SIGMOD Officers, Committees, and Awardees

Chair	Vice-Chair	Secretary/Treasurer
Donald Kossmann Systems Group ETH Zürich Cab F 73 8092 Zuerich SWITZERLAND +41 44 632 29 40 <donaldk AT inf.ethz.ch>	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>	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:

Denilson Barbosa, Pablo Barceló, Vanessa Braganholo, Marco Brambilla, Chee Yong Chan, Rada Chirkova, Anish Das Sarma, Glenn Paulley, 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 : June 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 : June 30th, 2014]

Editor's Notes

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

The issue opens with a Database Principles article by Ameloot on recent theoretical work on declarative networking. This article addresses three important aspects of declarative networking: (a) *coordination* where nodes of a cloud are trying to obtain a global consensus, and in particular, the CALM conjecture by Hellerstein, (b) *correctness* of distributed computation, including decidability results, and (c) *languages* in declarative networking and *declarative semantics* for such languages, including the CRON conjecture by Hellersterin. The article concludes by outlining several directions for future work. This article presents a timely review of the recent theoretical work on declarative networking and offers a great introduction to this topic for the interested reader.

The Research and Vision Articles column features three articles. First, Morton, Balazinska, Grossman, Kosara, and Mackinlay, present an analysis of usage patterns of Many Eyes and Tableau Public, two popular Web-based, collaborative visual analytics systems. The analysis explores a number of primary dimensions of online visual analytics including the types of users, how users collaborate and interact, and how they analyze single datasets versus multiple data sources. The results of this study offer valuable information and insights towards building online visual analytics systems in the future. The second article, by Torres, Galante, and Pimenta, addresses the issue that today's Object-Relational Mappings are often platform dependent and deeply embedded in the code, which is difficult to read, understand, or evolve. This article presents ENORM, a notation that extends class models representing all the essential mappings and does so in a platform-independent manner. The third article, by Lin, Chang, and Chao, addresses keyword search queries over XML documents which may contain arbitrary combinations of AND, OR, and NOT operators. This article presents the concept of *valid Smallest Least Common Ancestors* (*SLCA*s) as query results, which eliminates erroneous results returned by previous algorithms, as well as an efficient algorithm to process such queries.

The Systems and Prototypes Column features the Medusa system developed by Zhong and He, which is a parallel graph processing system on graphics processors (GPUs). The core design of Medusa is to enable developers to leverage the massive parallelism and other hardware features of GPUs by writing sequential C/C++ code for a small set of APIs. The runtime system of Medusa then automatically executes the user-defined APIs in parallel on the GPU, with a number of optimizations based on the architecture features of GPUs and the characteristics of graph applications. A case study in social network analysis shows how Medusa improves both the coding productivity and the performance of graph operations.

In the Research Centers column, Haas, Cefkin, Kieliszewski, Plouffe, Roth, et al. describe the design and activities of the IBM Research Accelerated Discovery Lab. The lab is built on an analytics cloud environment with a unique software system that supports the process of discovery, facilitating collaboration and fostering insight. While many other groups have or are creating institutes that focus in one way or another in data-driven discovery, the IBM Research Accelerated Discovery Lab is unique in its emphasis on supporting the overall discovery process and the focus on understanding, from a social science perspective, how discovery happens and how it may be accelerated. These aspects are illustrated through a diversity of analytic and systems research projects that span disciplines and institutions, as well as through a study of the practice of discovery, with the goal to use the findings to better enable and accelerate discovery.

This issue features three event reports. Koutrika, Lakshmanan, Riedewald, and Stefanidis report on the First International Workshop on Exploratory Search in Databases and the Web (ExploreDB 2014), held in conjunction with EDBT/ICDT 2014. The report highlights the keynote by Prof. Keim on "Exploring Big

Data using Visual Analytics", outlines a collection of six papers on various topics ranging from data exploration with structured database queries to search and ranking on the Web, and summarizes the panel discussion at the workshop. The report concludes by pointing out a number of research directions in areas including databases, Web search, multimedia exploration. In the second article, Ma, Meng, and Wang report on the Sixth International Workshop on Cloud Data Management (CloudDB 2014), co-located with ICDE 2014. The report covers the two keynotes speeches, "Building Big Data Processing Systems under a New Computing Model" by Prof. Xiaodong Zhang and "Multi-faceted Classification of Big Data Uses and Proposed Architecture Integrating High Performance Computing and the Apache Stack" by Prof. Fox, as well as 10 research papers covering a wide range of topics from Quality of Service, to Query Processing, System Architecture, and Benchmarks. The report closes by pointing out some open problems including big data management in the cloud and cloud data security and privacy. The third article, by Manghi, Bolikowski, Houssos, and Schirrwagen, reports on the First Workshop on Linking and Contextualizing Publications and Datasets, held in conjunction with the 3rd International Conference on Theory and Practice of Digital Libraries (TPDL). This workshop addresses the need to interlink and contextualize datasets and scientific publications in sectors of scholarly communication and digital libraries. The report covers two keynote speeches and 10 research papers on topics including dataset contextualization, interlinking datasets and publications, and representing and visualizing datasets. The report concludes by outlining main considerations and future issues with respect to publications and datasets.

On behalf of the SIGMOD Record Editorial board, I hope that you will all enjoy reading the June 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

June 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)

Declarative Networking: Recent Theoretical Work on Coordination, Correctness, and Declarative Semantics*

Tom J. Ameloot†
Hasselt University &
Transnational University of Limburg
Diepenbeek, Belgium
tom.ameloot@uhasselt.be

ABSTRACT

We discuss recent theoretical results on declarative networking, in particular regarding the topics of coordination, correctness, and declarative semantics.

1. INTRODUCTION

Cloud computing refers to the principle that computations are distributed over a network of computing nodes to increase parallelism [54]. Cloud computing is challenging to implement because messages can be delayed, computing nodes can crash, network links can be broken, etc. It seems desirable to abstract away from some of these technical aspects, and let them be automatically handled by a suitable framework.

Paradigms for cloud computing have emerged that hide some technical aspects and provide intuitive concepts instead. Well-known examples are MapReduce [24], Pregel [44], and GraphLab [41]. These paradigms suggest concrete and yet simple ways to think about cloud computing. The programmer typically provides the functionality in the form of a few modules, and the runtime engine takes care of the actual distributed execution of these modules. In general, the modules are specified with imperative programming languages.

Now, declarative networking is another proposal to simplify programming of cloud computing, using high-level declarative languages instead of imperative languages. The programmer expresses what should happen instead of how to effectively achieve this. The runtime engine will generate a distributed physical query plan to perform the desired cloud computation. For example, regarding messages, a declarative program will only generate the event to send a specific message to a certain recipient, and the runtime engine chooses some efficient delivery strategy. Languages for declarative networking elegantly combine messaging features with local computation. Originally, the term declarative networking referred specifically to Datalog-inspired languages, and also more specifically to networking protocols [39]. In the meantime, the languages remain mostly Datalog-inspired, but several works now also consider their use for general distributed database queries. In this context, cloud data is typically viewed as a distributed database.

In this paper, we discuss recent theoretical results on declarative networking, thereby complementing the surveys of Hellerstein [30] and Loo et al. [40] that discuss various applications and practical aspects of declarative networking.

We give an overview of the paper. First, Section 2 briefly introduces basic database and Datalog terminology. The following two sections discuss theoretical results on distributed execution. Section 3 discusses coordination, and studies in particular the CALM conjecture by Hellerstein. Section 4 discusses correctness of distributed computations, including decidability results. Next, Section 5 highlights features of languages in declarative networking and reviews declarative semantics for such languages; in this context, we also examine a second conjecture by Hellerstein, namely, the CRON conjecture. Section 6 provides directions for further work.

2. PRELIMINARIES

The purpose of this section is to introduce some concepts that are frequently used in this paper [3].

*Database Principles Column. Column editor: Pablo Barceló. Department of Computer Science, University of Chile, Santiago, Chile. E-mail: pbarcelo@dcc.uchile.cl

†PhD Fellow of the Fund for Scientific Research, Flanders (FWO).

A *database schema* $\mathcal{D}$ is a set of pairs (R, k), where R is a relation name and $k \in \mathbb{N}$ is its associated arity. A *fact* over $\mathcal{D}$ is of the form $R(\bar{a})$ where R is a relation name from the schema and $\bar{a}$ is a tuple of values matching the arity of the relation. An *atom* over $\mathcal{D}$ is of the form $R(\bar{u})$ where R is again a relation from the schema, and $\bar{u}$ is now a possibly mixed tuple of values and variables, matching the arity of the relation.

A *conjunctive query with negation* over $\mathcal{D}$ is of the following form:

$$T(\bar{u}) \leftarrow R_1(\overline{v_1}), \ldots, R_p(\overline{v_p}), \neg S_1(\overline{w_1}), \ldots, \neg S_q(\overline{w_q}).$$

where $T(\bar{u})$ and all $R_i(\overline{v_i})$ and $S_j(\overline{w_j})$ are atoms over $\mathcal{D}$. A conjunctive query with negation may also be called a (Datalog) *rule*. Atom $T(\bar{u})$ is called the *head* and the other atoms constitute the *body*. The order of body atoms is usually irrelevant. The R_i-atoms are called *positive*: they test the presence of facts. The S_j-atoms are called *negative*: they test the absence of facts; the symbol '$\neg$' stands for negation. For simplicity, we make the common assumption that all variables of a rule occur in its positive body atoms.

To evaluate a rule on a set of input facts, we seek a substitution of the rule variables by values so that facts resulting from positive body atoms occur in the input and facts resulting from negative body atoms do not occur in the input. Applying this substitution to the head atom results in a fact, the so-called *derived* fact.

A Datalog program over a database schema is a set of rules over this schema. A Datalog program is called *positive* when its rules contain only positive body atoms.[1] A Datalog program is called *recursive* when some head relations of rules also occur in rule bodies.

3. COORDINATION

Coordination means that nodes of a cloud are trying to obtain a global consensus. For example, by exchanging messages about the presence or absence of data in the cloud, coordination protocols could ensure that all nodes have the desired data before applying negation in their local computation. Because computation at all nodes is halted during coordination, which reduces parallelism, we want to avoid coordination as much as possible.

Recent research on coordination consists of two main approaches. Both approaches provide indications about how efficient the distributed runtime engines for declarative networking can be made. The first approach, as embodied by the CALM conjecture, investigates which distributed computations can completely avoid coordination and are thus "embarrassingly parallel" [30]. For distributed computations that can not completely avoid coordination, the other approach quantifies the required amount of coordination. Even for computations that can avoid coordination, a quantitative approach can shed light on the costs involved. These approaches are complementary, and are discussed in Sections 3.1 and 3.2 respectively.

3.1 CALM Conjecture

During his PODS 2010 keynote, Hellerstein presented a number of intriguing conjectures to the database community [30]. The first conjecture is called the CALM conjecture (Consistency And Logical Monotonicity), that we repeat for convenience:

CONJECTURE 1 (CALM [30]). *A program has an eventually consistent, coordination-free execution strategy if and only if it is expressible in (monotonic) Datalog.*

We explain the meaning of this conjecture. Eventual consistency is a correctness notion: it indicates that the program can tolerate arbitrary message delays, as occurring in asynchronous communication settings (cf. Section 4). Coordination-freeness means that coordination is completely avoided. Monotonic Datalog refers to positive Datalog, that is indeed restricted to so-called *monotone* computations where previous output facts remain valid whenever the input is extended with new facts (and new output facts may also be produced). So, the CALM conjecture suggests that a distributed program can avoid coordination (and stay correct) if and only if that program is expressible in positive Datalog.

One direction of the CALM conjecture was already known: positive Datalog programs can be implemented without coordination [39]. This actually holds more generally for all monotone programs, even those that are not expressible in positive Datalog: the main intuition here, is that the nodes can send the input data to each other and steadily accumulate these messages; whenever a new message arrives, a node can always again evaluate the monotone program. Because the program is monotone, no wrong outputs are produced by previous evaluations. This strategy results in eventual consistency.

But the other direction of the CALM conjecture appears new: it suggests an upper bound on the ex-

[1] The term "Datalog" originally denoted programs with only positive bodies [3]. But to simplify terminology, this paper also uses the term for programs with negative body atoms.

pressivity of distributed programs that can proceed without coordination. This direction has prompted several investigations, that we discuss below.

Coordination-freeness and monotonicity. Because the CALM conjecture was only stated informally, it had to be formalized first. Ameloot et al. [12] have proposed a formal definition of coordination-freeness: a program is called coordination-free when for each set of input facts, there is some right way to distribute these facts over the network so that the nodes can already compute the entire output without communicating. Intuitively, the program still has to be correct for all possible input distributions, but there is a right distribution enabling an embarrassingly parallel execution. Although the original CALM conjecture mentioning Datalog was disproved in the formal framework of Ameloot et al., nonetheless the main intuition of the conjecture turns out to hold [12]: a distributed program is coordination-free if and only if it is monotone.

Coordination-freeness and non-monotonicity. Using the same definition of coordination-freeness as Ameloot et al. [12], Zinn et al. [55] have subsequently obtained additional insights on the CALM conjecture. Surprisingly, it turns out that some *non-monotone* programs are coordination-free when each node is given knowledge about the *distribution policy* of the global input data. This way, a node can sometimes locally conclude that certain input facts are globally absent, allowing some non-monotone programs to proceed without coordination. In the previous model [12], where this policy is not exposed, a node would always have to *coordinate* with all other nodes to conclude such global absences. Moreover, it turns out that in some variations of the model considered by Zinn et al., all programs can be made coordination-free; these variations, however, are quite expensive in terms of how much additional data each node should have.

Weaker forms of monotonicity. Recently, the results by Ameloot et al. [12] and Zinn et al. [55] have been combined in a more unified theory on the CALM conjecture [11]. In particular, the non-monotone programs that can avoid coordination, as identified by Zinn et al. [55], have been characterized semantically with weaker forms of monotonicity: two classes have been identified, called *domain-distinct-monotone* and *domain-disjoint-monotone* programs, that we explain below.

Recall that for an "ordinary" monotone program, previous output facts remain valid whenever the input is extended with arbitrary new facts (and new output facts may also be produced). Now, a program is called domain-distinct-monotone if previous output facts remain valid when we extend the input with facts that each contains at least one new value not yet occurring in the old input. For example, the difference $R \setminus S$ of two unary relations R and S is domain-distinct-monotone: indeed, new input facts added to R will not shrink the output, and new facts added to S will not subtract from R if they contain at least one value not yet occurring in the old input. Note that $R \setminus S$ is not monotone.

A program is called domain-disjoint-monotone if previous output facts remain valid when we extend the input with facts that share no values with the old input. For example, the complement of the transitive closure on a binary edge relation R is domain-disjoint-monotone: new edges that share no values with the previous input can not establish a path between old vertices. This program is not domain-distinct-monotone.

Note that ordinary monotonicity implies domain-distinct-monotonicity and that the latter implies domain-disjoint-monotonicity.

In accordance with the results of Zinn et al. [55], domain-distinct-monotone programs can be implemented without coordination if the nodes of the cloud are made aware of how input facts are distributed. The same awareness is needed for domain-disjoint-monotone programs to be implemented without coordination, but also with the additional assumption that nodes are now "responsible" for input *values*: each node is initialized with all input facts containing any value the node is responsible for.

Efficient coordination-free strategies. The above works theoretically relate distributed coordination to program monotonicity. They are complemented by works that investigate efficient implementation strategies for coordination-free programs. For example, the works of Loo et al. [38, 39] and Nigam et al. [47] provide concrete algorithms for the case of distributed positive Datalog programs. Whenever some input facts change, these algorithms efficiently update the state at the nodes of a cloud. To avoid recomputing the entire state at every node, only incremental changes are propagated. This reduces communication and needless recomputation. These algorithms are coordination-free; handle recursive Datalog; and, tolerate messages delayed by the network, i.e., they are eventually consistent.

3.2 Quantification

The second approach to understanding coordination is to quantify the amount of coordination, and any related costs.

First, Alvaro et al. [7, 8] propose program analysis techniques to detect code fragments where coordination is perhaps overused. This way, some uses of coordination could be replaced with strategies like eventual consistency, reducing the overall amount of coordination.

Koutris and Suciu [33] define the massively parallel model of computation (MP). An execution in this model is a sequence of global MP steps. In each step, the nodes first communicate and then they perform local computation. Each step represents a global round of coordination. Koutris and Suciu also define when an algorithm is load-balanced in this model: this intuitively means that each server locally processes an equal share of the total problem size. In this setting, Koutris and Suciu prove that the *tall-flat conjunctive queries* are precisely those conjunctive queries that can be computed in one MP step in a load-balanced way.

Beame et al. [18] extend the work of Koutris and Suciu by considering a parameter to control the amount of data that each node may receive during a step. Higher values of the parameter allow more replication of data. Less replication is viewed as more efficient. Beame et al. quantify the replication required when a computation may only use one global communication step. Beame et al. also quantify the number of global steps required when the allowed replication is fixed.

Interestingly, the positive conjunctive queries considered for load-balanced algorithms [33] and replication [18] can be implemented with a coordination-free strategy in the models used for the CALM conjecture [12, 55]. However, these coordination-free strategies would not be efficient because they gradually replicate the input over the network. Thus, the notion of coordination-freeness is only part of a larger picture, where costs can be formalized and measured in multiple ways.

4. CORRECTNESS

Cloud computing often works over an asynchronous communication model, where messages can be arbitrarily delayed. Larger networks are typical settings with asynchronous communication, because routers can forward messages differently depending on network congestion, subjecting messages to unpredictable latencies.

It is important to design distributed programs that tolerate message delays. We may call a distributed program correct if, for each input, it succeeds in producing the desired output no matter how much messages are delayed. We discuss two main strategies for ensuring correctness, namely, construction and verification, given in Sections 4.1 and 4.2 respectively.

4.1 Constructive Approach

A first main strategy to obtain correct distributed programs, is to use certain principles for program construction.

On one side of this spectrum, we have eventual consistency [52, 30, 16]. This means that the output is eventually produced if messages are eventually delivered, in some arbitrary fashion. There is no coordination here. It is well-known that monotone computations can be executed in an eventually consistent way: indeed, whenever a node receives a new message, it can simply recompute the local result, which is guaranteed to be part of the overall output by monotonicity (cf. Section 3.1). Another approach to eventual consistency consists of so-called commutative replicated data types, where messages represent commutative operations, that are thus resilient to unpredictable reorderings [50, 22, 16].

Coordination protocols are at the other side of the spectrum, e.g., used when the computation is not monotone or if messages do not commute. Note, however, that some classes of non-monotone computations can be implemented without coordination [55].

4.2 Deciding Correctness

A second main strategy, is to decide correctness for distributed programs.

Ameloot and Van den Bussche [13] have investigated decidability of correctness for distributed programs in which each computing node of a cloud is represented by a (relational) transducer [4, 25, 26, 27, 51]; such a distributed program is referred to as a transducer network. Here, a transducer is a collection of queries over a database schema, where each of the relations is used for either input, output, memory, messages, or auxiliary system relations; the queries update the output and memory relations, and generate messages.

Now, Ameloot and Van den Bussche [13] define correctness as a confluence notion: a distributed program is called confluent if for any two finite execution traces on the same input, the second trace can always be extended to obtain the (partial) output of the first trace. Intuitively, the prior execution of the program will not prevent outputs from being produced. The opposite of confluence is called

diffluence. Deciding diffluence for so-called *simple* transducer networks, where transducers are implemented with restricted conjunctive queries, turns out to be NEXPTIME-complete. The restrictions of simple transducer networks are: *(i)* the network is recursion-free, where rules cannot be mutually recursive through positive body atoms; *(ii)* deleting from output and memory relations is forbidden; *(iii)* negation on message relations is forbidden; *(iv)* rules inserting into output and memory relations must be "message-bounded";[2] finally, *(v)* message-sending rules only use input and message relations.

Ameloot and Van den Bussche [13] have shown that simple transducer networks compute exactly all distributed queries expressible by unions of conjunctive queries with negation, or equivalently, the existential fragment of first-order logic. Compared to standard database queries, the location of facts matters for distributed queries. We may conclude that simple transducer networks are indeed a weaker computational model, but that is not totally useless.

Ameloot [10] has investigated decidability of a second formalization of correctness, referred to as consistency, also appearing in prior works [2, 12]: a distributed program is called consistent if any two infinite fair execution traces on the same input yield the same output. The fairness conditions demand that all sent messages are eventually delivered and that all nodes are made active infinitely often. Deciding inconsistency for simple transducer networks is again NEXPTIME-complete. The expressivity of simple transducer networks is the same under both confluence and consistency.

5. DECLARATIVE LANGUAGES

This section is devoted to languages in declarative networking, and their semantics. Section 5.1 highlights some important features of languages in declarative networking. Section 5.2 discusses operational semantics. Section 5.3 discusses declarative semantics as an alternative to operational semantics; we also use this context to discuss a second conjecture by Hellerstein, namely, the CRON conjecture [30].

5.1 Datalog Variants

As we have mentioned in the Introduction, declarative networking originally developed around Datalog [39, 30]. Today, Datalog is still an attractive foundation for declarative networking, because it allows expressing advanced algorithms with relatively few lines of code [30]. We are also seeing a more general interest in Datalog [23, 31, 17].

A notable language proposed in declarative networking is Dedalus [9, 30], a minimalistic extension of Datalog to the distributed setting: it only provides basic features for reasoning about distributed facts, and it provides a simple way to designate some facts as messages between nodes. Initial expressivity and complexity properties of Dedalus are provided by Ameloot and Van den Bussche [14].

Dedalus [9] and its predecessor languages [39] have influenced other recent language designs in declarative networking such as WebdamLog [2, 1], Bloom [7, 8], and several other works [29, 32, 37].

Location specifiers. A frequently occurring feature in declarative networking, is the use of *location specifiers* to tag facts with the node that stores that fact [39]. Accordingly, rules have additional variables for location specifiers in head and body atoms; each atom contains precisely one such variable. Often, the same location specifier is used in all body atoms, meaning that the rule can be evaluated locally on a single node. Now, if the head location specifier variable is different from the body location specifier variable, derived facts are sent as messages to the node indicated by the head variable. Otherwise, derived facts are stored locally. For each case, the runtime engine handles the details of message sending or local storage, respectively.

Delegation. The language WebdamLog [2, 1] has introduced the novel feature to *delegate* at runtime a piece of functionality, as represented by a set of rules, to the node with the best opportunity to evaluate these rules; this typically means that the node has the required data. Also, an important design principle of WebdamLog is that previously unseen nodes can start to participate in a computation that is already running, each contributing new local rules. To make different WebdamLog rules still globally interoperate, a programmer could write variables in place of relation names.

Time. In its "unsugared" presentation, Dedalus explicitly exposes *time variables* in its rules. The intention of exposing time, is to more clearly reason about dynamic changes to the memory of the computing nodes; all from within the declarative program itself, instead of deferring this aspect to the runtime engine. Concrete values for time variables may be called *timestamps*, and are often just natural numbers. The exposure of time connects Dedalus to temporal deductive databases and tem-

[2]This corresponds to *input-boundedness*, as first identified by Spielmann [51] and further investigated by Deutsch et al. [26, 27].

poral logic programming (cf. Section 5.3.2).

5.2 Operational Semantics

To describe how programs in declarative networking are distributedly executed, often an *operational semantics* is used. This represents how the runtime engine underneath the declarative language works. The results on coordination (Section 3) and correctness (Section 4) are about such operational semantics.

It is well understood how such an operational semantics might be defined [27, 46, 29, 2, 12]. Typically, a transition system is used, describing how the cloud moves from one global state to another global state as the result of local computation at nodes and message sending between nodes. This transition system is infinite because nodes can run indefinitely and keep sending messages so that an unbounded number of messages can be floating around in the network. In addition, the transition system is highly nondeterministic, because each transition chooses which node becomes active and which messages are delivered. This allows representing asynchronous communication, where messages can be delayed and eventually be delivered out of order.

Ameloot et al. [12] have defined an operational model for declarative networking where each computing node is implemented with a local relational transducer (cf. Section 4.2). Fragments of Datalog may be used to implement such transducers, for example, unions of conjunctive queries with negation or first order logic. Ameloot et al. [12] also provide expressivity results in this operational model. For example, non-monotone distributed computations require each node to have access to its own identifier and the identifiers of all the other nodes. Also, the transducer model turns out to be quite natural: it only introduces a kind of iteration to the local query language of the transducers.

Because an operational semantics might become difficult for a programmer to imagine, it is useful to look at suitable abstractions. This may be called a *declarative semantics*, which is discussed next.

5.3 Declarative Semantics

By hiding technical details of operational executions, a declarative semantics can help separate the meaning of a program from the actual distributed execution strategies. This way, old distributed execution strategies can be replaced with new strategies, as long as the new strategies satisfy the same declarative semantics.

Based on their Datalog origin, languages in declarative networking have already explored some well-known semantics of Datalog as candidates for their own declarative semantics: Section 5.3.1 discusses simple fixpoint semantics; Section 5.3.2 discusses syntactically and temporally stratified semantics; and, Section 5.3.3 discusses the stable model semantics. Although the stable model semantics might be less intuitive for the programmer, it provides avenues for new theoretical and practical research. In particular, Section 5.3.4 discusses how stable models allow reasoning about message causality.

5.3.1 Simple Fixpoint Semantics

Although strictly speaking it is still an operational semantics, a fixpoint semantics can be an intuitive semantics for declarative networking. Essentially, this semantics transforms an initial set of input facts by successively applying updates generated by triggered rules. Updates could be insertions, or deletions when rule heads contain negation. The computation ends when no more facts can be added or removed, i.e., when a fixpoint has been reached. Sometimes no fixpoint is reached.

In declarative networking, the programmer can imagine that the fixpoint semantics is applied to a centralized, holistic Datalog-like program having access to all data and rules in the cloud. Here, communication is viewed as happening instantaneously, that is, asynchronous communication is abstracted away. It is important, of course, to prove that output under this fixpoint semantics really corresponds to the output produced by the distributed execution.

Positive programs. For positive Datalog-like languages, i.e., programs not using negation (and not doing deletions), several works establish a connection between a centralized fixpoint semantics and a distributed execution [38, 39, 2, 47]. Here, the fixpoint always exists.[3] The intuition for monotone programs applies (cf. Section 3.1): using mild syntactic assumptions [2], positive programs will steadily accumulate any received messages, thereby creating the opportunity for delayed facts to still participate in the monotone computation, and relational joins in particular.

Semi-monotone programs. Zinn et al. [55] have investigated a Datalog variant called *semi-monotone*. Besides allowing rules to trigger fact insertions and deletions, this language only allows two kinds of computed relations: *(i)* relations only tested positively in rules and only inserted into, and *(ii)* rela-

[3]The fixpoint semantics for positive Datalog programs corresponds to their minimal model semantics [3].

tions only tested negatively in rules and only deleted from. For this language, Zinn et al. prove that a deterministic fixpoint semantics corresponds to distributed executions that are eventually consistent. Again, the deterministic semantics appears more intuitive compared to the nondeterministic distributed execution.

5.3.2 Stratified Semantics

In declarative networking, two variants of stratified programs have been studied.

Syntactically stratified programs. A Datalog program is *syntactically stratified* when its rules can be divided into sets, called strata, that are ordered in such a way that rule bodies apply negation only to relations computed in previous strata [3]. So, there is no recursion through negation. The program is evaluated by successively evaluating the strata: we start with the first stratum, then the next stratum, etc. Each stratum itself is evaluated under the fixpoint semantics, and it may read the facts generated by the previous stratum.

Syntactic stratification can also be defined for languages in declarative networking, e.g., for fragments of Dedalus [9] or WebdamLog [2]. The connection between a centralized semantics (cf. Section 5.3.1) and distributed executions can also be established for syntactically stratified programs, but this is more challenging compared to positive programs [30]. Indeed, a relation T could be partially computed by multiple nodes in the distributed setting. So, whenever a node x wants to apply negation to relation T (as computed in a previous stratum), node x needs to communicate with the other nodes before it can determine the presence or absence of some T-facts. One way to achieve this, is to let the global computation proceed in rounds: each round corresponds to one stratum of the centralized program, and each round is followed by a coordination phase to make sure that nodes have their required facts from the previous stratum; then the next round begins and nodes can safely apply local negation. Here, although the centralized stratified semantics is still intuitive for the programmer, the distributed execution may need to employ some expensive coordination mechanisms.[4]

Temporally stratified programs. A Datalog program is called *locally stratified* when for each input, all possible ground rules based on the input values can be grouped into strata such that ground atoms appearing negatively in rule bodies can be rule-heads only in lower strata [15].[5] Intuitively, no ground atom can negatively depend on itself. So, this condition is very much like syntactic stratification, except we now use ground rules.

A particular kind of local stratification is *temporal stratification*, that is well-studied in temporal deductive databases and temporal logic programming [53, 34, 42]. Seminal work in this field is by Chomicki and Imieliński [21, 20]. In this setting, all facts are tagged with an additional timestamp, to indicate the discrete moment on which the fact exists. Accordingly, rules will mention an additional timestamp variable for each head and body atom. Intuitively, a program is said to be temporally stratified when each head timestamp variable always represents a larger timestamp value than all timestamp variables in the accompanying rule body. This ensures that facts are only derived in the future. Negation is thus only applied to relations computed in the past, preventing cyclic dependencies involving negation through time. Besides using a model-based semantics [15], we can imagine that the program evolves from timestamp to timestamp, where facts available at the current timestamp may contribute to deriving facts at future timestamps.

In the context of declarative networking, Dedalus without (asynchronous) message rules is temporally stratified [9, 30]. In the remaining rules, all body atoms use the same timestamp variable, and the head timestamp variable is either *(i)* the same as the body timestamp variable, or *(ii)* it is restricted to be the successor of the body timestamp variable. Dedalus assumes that rules of the first kind, called deductive rules, are syntactically stratified. This ensures temporal stratification for the language without message rules.

In a similar vein, Interlandi et al. [32] give a Dedalus-inspired language for *synchronous* systems. Here, nodes of the network proceed in rounds and messages are not arbitrarily delayed. During each round, nodes share the same global clock. Interlandi et al. show that an operational semantics for their language coincides with a declarative model-based semantics of a single holistic Datalog program; this declarative semantics is enabled by the temporal stratification.

More generally, it seems that when ignoring message rules, many languages used in declarative networking can be (strictly) embedded in some of the prior languages [53, 34, 42], because the remaining rules represent local computation, which typically

[4] Hellerstein [30] makes initial suggestions to reduce the coordination complexity of stratified programs.

[5] Ground rules are obtained from original program rules by replacing their variables with concrete values.

only deals with the current time and the next time.

As an intermediate conclusion, the previous works mentioning temporal stratification for declarative networking have not yet investigated asynchronous communication, where arrival timestamps of messages can be arbitrarily into the future. Asynchronous communication can, however, be represented in detail by the stable model semantics, as discussed next.

5.3.3 Stable Model Semantics

We now discuss uses of the stable model semantics [28] for languages in declarative networking. Although it is perhaps less intuitive for the programmer, this semantics provides an interesting framework for theoretical and practical research.

Dedalus stable models. Stable models have been proposed as a way of thinking about the semantics of Dedalus programs [45]. The main idea is that, for a given distributed input, each stable model represents another way in which nondeterminism is caused by asynchronous communication.

A formal proof was provided to show the correspondence between the stable model semantics and an operational semantics [5, 6].[6] In this proof, a Dedalus program is first translated to a pure Datalog program (with negation), to which the stable model semantics is applied. This Datalog program represents the computation of the entire cloud, thus providing a kind of centralized semantics as in Section 5.3.1. The pure Datalog program gives each relation two dedicated components: one component for the location of facts and the other component for the local timestamp of facts at their location (cf. Section 5.1). Now, asynchronous communication is modeled with the choice construct by Saccà and Zaniolo [49], allowing to nondeterministically select an arrival timestamp at the addressee for each message. The pure Datalog program is also extended with auxiliary rules to enforce natural properties occurring in an operational semantics. The first natural property is *causality*: messages are only delivered in the future, and not in the past. We elaborate on this property in Section 5.3.4. The second natural property is that only a finite number of messages arrive at each timestamp of a node.

Practical answer set programming. Stable model semantics also enables verification and testing. For example, Lobo et al. [37] provide a semantics for a Dedalus-like language based on answer set programming (ASP), i.e., stable models. This is again done by translating an original program into a pure Datalog program that holistically describes the distributed computation, resembling the translation of Dedalus programs to Datalog mentioned above [5, 6]. To enforce execution properties in their semantics, like causality, Lobo et al. also specify auxiliary rules in the syntactical translation. By varying certain rules, the communication semantics can be specified, for example, whether messages are delivered synchronously or asynchronously.

By giving this holistic Datalog program to available ASP solvers, the original distributed program can be simulated and thus analyzed [37]. Under asynchronous communication, nondeterminism can occur: multiple answer sets exist, each representing a different execution of the distributed program. One might, for example, verify whether the distributed algorithm is correct in the sense of Section 4 by enumerating or sampling the answer sets. However, enumerating all possible answer sets under asynchronous communication poses scalability issues [37].

The work of Lobo et al. is extended by Ma et al. [43], who also formalize an operational semantics of distributed systems. Global properties of the system can again be analyzed by translating it into a logic program, to which an ASP solver can be applied.

In this context we can also mention an area of artificial intelligence closely related to declarative networking: programming multi-agent systems in declarative languages. The knowledge of an agent can be expressed by a logic program and agents update their knowledge by modifying their rules [36, 48, 35]. The semantics of such dynamic agents is often given by a stable model semantics, implemented in practice with ASP solvers.

5.3.4 The CRON Conjecture

The term *causality* means that an effect can only happen after its cause. In the distributed setting, this implies that a message can only arrive after it was sent. To illustrate, if node x at local timestamp 2 sends a message A that arrives at local timestamp 1 of node y, then any message B that node y sends at local timestamp 1 or later may not arrive on x at local timestamp 2 or less; put differently, local timestamp 2 of x lies in the past with respect to local timestamp 1 of y.

In practice, causality seems to be satisfied in general, except in situations like crash recovery: there,

[6] To the best of our knowledge, this is the only work to rigorously establish a connection between stable models and an operational semantics of the form discussed in Section 5.2.

an arriving or logged message could appear to come from the future when it is put side-by-side with an old state snapshot. To illustrate, suppose in our above example that x sets a local flag when it has sent message A to y and that y sends B as a reply when receiving A. If x crashes, x could now be reverted to a state without the local flag; when receiving message B in this state, node x will not expect B, as if B is coming from the future in the viewpoint of x. We may call this *non-causality*.

In this context, we can now study a second conjecture by Hellerstein, namely the CRON conjecture (Causality Required Only for Non-monotonicity):

CONJECTURE 2 (CRON [30]).
Program semantics require causal message ordering if and only if the messages participate in non-monotonic derivations.

The CRON conjecture relates causality of messages to the nature of the computations in which those messages participate, for example monotone versus non-monotone. In particular, the conjecture suggests that the order of messages (causality) is only important for non-monotone computations. Perhaps more generally, the CRON conjecture asks us to think about the need for time, like when temporal delay on messages is needed for expressivity.

One way to investigate this conjecture is as follows. As explained in Section 5.3.3, the stable model semantics for languages in declarative networking allows representing asynchronous communication in detail. In particular, the choice construct [49] allows us to nondeterministically select an arrival timestamp at the addressee for each message. But auxiliary rules were be added to enforce causality [5, 37, 6]. Now, to formalize the CRON conjecture in the Dedalus setting, Ameloot and Van den Bussche [14] study the effect of *omitting* such auxiliary rules, to see how the behavior of the program changes as a result. Concretely, omitting the auxiliary rules allows certain stable models where messages are sent "into the past", representing non-causality. Then, a Dedalus program that is already correct in an operational semantics (cf. Section 4) is said to *tolerate non-causality* when these non-causal stable models yield the same output as the operational executions.[7] In this setting, the CRON conjecture can be seen as suggesting a link between tolerance to non-causality and monotonicity, much like the CALM conjecture relates a semantic property (coordination-freeness) to monotonicity.

[7]Inside a stable model, the output at a node x is the set of all facts $\boldsymbol{f}$ for which there is a local timestamp of x so that $\boldsymbol{f}$ is present at x in all the following local timestamps.

However, Ameloot and Van den Bussche [14] show that the CRON conjecture does not hold when formalized purely semantically, where a Dedalus program is seen as expressing a database query. More concretely, both directions of the following formal conjecture can be disproved: *A Dedalus program computes a monotone query if and only if it tolerates non-causality.* However, on a more syntactical level, it can be shown that all positive Dedalus programs tolerate non-causality.[8] This result establishes a class of programs that no not require causality to be maintained on messages, for example during crash recovery.

Outside crash recovery, the extreme non-causality of sending messages "into the past" is unlikely to occur. Yet, the result that positive Dedalus programs tolerate non-causality seems distantly related to the result that positive programs have eventually consistent distributed execution strategies (cf. Section 4.1): indeed, eventually consistent programs can deal with unpredictable message reorderings, as occurring under asynchronous communication.

6. FURTHER WORK

We provide directions for further work.

Coordination.

For the CALM conjecture, we might need additional formal definitions of coordination-freeness. A problem with the previous definition [12] is perhaps that a distributed program is already coordination-free when there is one right distribution of the input on which nodes can locally compute the output, while on more general distributions the program may gradually replicate the input at all nodes.

From this perspective, theoretical work is also needed on quantifying the costs required for computing certain queries. Example costs are the number of coordination steps [33], and the amount of replication [18].

Also interesting, is to investigate how increased local knowledge on nodes allows non-monotone computations to avoid coordination [55, 11].

Correctness.

The mentioned decidability results [13, 10] provide an indication that automatically deciding correctness of a distributed system might not be a useful strategy in practice. Indeed, we have to severely restrict expressivity to obtain decidability, and yet

[8]For completeness, we mention that this result depends on the assumption that only a finite number of messages arrive at each local timestamp of a node; a property also mentioned in Section 5.3.3.

the time complexity remains prohibitively high.

It might be more promising to achieve correctness through practical mechanisms and protocols [50, 16, 22], design guidelines [19], and insights about monotonicity such as the CALM conjecture [30, 7, 55, 12, 11].

Declarative semantics.

The motivation behind declarative networking is to simplify programming of cloud computing. Besides offering a convenient syntax, declarative networking should offer intuitive ways to think about the programmed functionality, also in asynchronous communication models. This could be done with a suitable declarative semantics, for which some initial explorations have been mentioned in Section 5.3.

We believe that declarative semantics for declarative networking deserves more attention. The difficulty is that a declarative semantics should at the same time hide technical operational details, whilst at the same time preserving an understanding of the distributed setting. In particular, further work seems needed to provide an intuitive semantics for distributed negation. This investigation could also encompass features like aggregation. But the intuitiveness of the semantics should not force execution engines to resort to heavy coordination. Perhaps this will only work for some restricted classes of programs.

Also, the stable models considered by previous work [5, 37, 6] are typically infinite, because they represent an infinite time domain in which the distributed computation unfolds. This is partly caused by asynchronous communication, where messages have arbitrary delays. In further work, it would be interesting to find finite representations of these stable models, again perhaps only for restricted classes of programs.

Regarding the CRON conjecture, further work is needed to understand the spectrum of causality: besides positive programs, perhaps richer classes of programs can tolerate some relaxations of causality as well. Also, it might be intriguing to link the CRON conjecture more concretely to crash recovery mechanisms, or other application scenarios giving rise to non-causality.

7. REFERENCES

[1] S. Abiteboul, E. Antoine, G. Miklau, J. Stoyanovich, and J. Testard. Rule-based application development using webdamlog. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 965–968. ACM, 2013.

[2] S. Abiteboul, M. Bienvenu, A. Galland, and E. Antoine. A rule-based language for Web data management. In *Proceedings 30th ACM Symposium on Principles of Database Systems*, pages 293–304. ACM Press, 2011.

[3] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[4] S. Abiteboul, V. Vianu, B. Fordham, and Y. Yesha. Relational transducers for electronic commerce. *Journal of Computer and System Sciences*, 61(2):236–269, 2000.

[5] P. Alvaro, T.J. Ameloot, J.M. Hellerstein, W.R. Marczak, and J. Van den Bussche. A declarative semantics for Dedalus. Technical Report UCB/EECS-2011-120, EECS Department, University of California, Berkeley, Nov 2011.

[6] P. Alvaro, T.J. Ameloot, J.M. Hellerstein, W.R. Marczak, and J. Van den Bussche. A declarative semantics for Dedalus. Hasselt University, Technical report, `http://hdl.handle.net/1942/14572`, 2013.

[7] P. Alvaro, N. Conway, J. Hellerstein, and W.R. Marczak. Consistency analysis in Bloom: A CALM and collected approach. In *Proceedings 5th Biennial Conference on Innovative Data Systems Research*, pages 249–260. www.cidrdb.org, 2011.

[8] P. Alvaro, N. Conway, J.M. Hellerstein, and D. Maier. Blazes: Coordination analysis for distributed programs. Technical Report UCB/EECS-2013-133, EECS Department, University of California, Berkeley, Jul 2013.

[9] P. Alvaro, W.R. Marczak, N. Conway, J.M. Hellerstein, D. Maier, and R. Sears. Dedalus: Datalog in time and space. In de Moor et al. [23], pages 262–281.

[10] T.J. Ameloot. Deciding correctness with fairness for simple transducer networks. In *Proceedings of the 17th International Conference on Database Theory*, 2014 (to appear).

[11] T.J. Ameloot, B. Ketsman, F. Neven, and D. Zinn. Weaker forms of monotonicity for declarative networking: a more fine-grained answer to the CALM-conjecture. In *Proceedings 33rd ACM Symposium on Principles of Database Systems*, 2014 (to appear).

[12] T.J. Ameloot, F. Neven, and J. Van den Bussche. Relational transducers for declarative networking. *Journal of the ACM*, 60(2):15:1–15:38, 2013.

[13] T.J. Ameloot and J. Van den Bussche. Deciding eventual consistency for a simple class of relational transducer networks. In *Proceedings of the 15th International Conference on Database Theory*, pages 86–98. ACM Press, 2012.

[14] T.J. Ameloot and J. Van den Bussche. Positive Dedalus programs tolerate non-causality. *Journal of Computer and System Sciences*, to appear.

[15] K.R. Apt and R.N. Bol. Logic programming and negation: A survey. *The Journal of Logic Programming*, 19-20, Supplement 1(0):9–71, 1994.

[16] P. Bailis and A. Ghodsi. Eventual consistency today: Limitations, extensions, and beyond. *ACM Queue*, 11(3), 2013.

[17] P. Barceló and R. Pichler, editors. *Datalog in Academia and Industry*, volume 7494 of *Lecture Notes in Computer Science*. Springer, 2012.

[18] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM Symposium on Principles of Database Systems*, pages 273–284. ACM Press, 2013.

[19] M. Cavage. There's just no getting around it: You're building a distributed system. *ACM Queue*, 11(4), 2013.

[20] J. Chomicki. Depth-bounded bottom-up evaluation of logic programs. *The Journal of Logic Programming*, 25(1):1–31, 1995.

[21] J. Chomicki and T. Imieliński. Finite representation of infinite query answers. *ACM Transactions on Database Systems*, 18(2):181–223, 1993.

[22] N. Conway, W.R. Marczak, P. Alvaro, J.M. Hellerstein, and D. Maier. Logic and lattices for distributed programming. In *Proceedings of the Third ACM Symposium on Cloud Computing*, pages 1:1–1:14. ACM Press, 2012.

[23] O. de Moor, G. Gottlob, T. Furche, and A. Sellers, editors. *Datalog Reloaded: First International Workshop, Datalog 2010*, volume 6702 of *Lecture Notes in Computer Science*, 2011.

[24] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[25] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Proceedings 12th International Conference on Database Theory*, 2009.

[26] A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven Web applications. *Journal of Computer and System Sciences*, 73(3):442–474, 2007.

[27] A. Deutsch, L. Sui, V. Vianu, and D. Zhou. Verification of communicating data-driven Web services. In *Proceedings 25th ACM Symposium on Principles of Database Systems*, pages 90–99. ACM Press, 2006.

[28] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[29] S. Grumbach and F. Wang. Netlog, a rule-based language for distributed programming. In M. Carro and R. Peña, editors, *Proceedings 12th International Symposium on Practical Aspects of Declarative Languages*, volume 5937 of *Lecture Notes in Computer Science*, pages 88–103, 2010.

[30] J.M. Hellerstein. The declarative imperative: experiences and conjectures in distributed logic. *SIGMOD Record*, 39(1):5–19, 2010.

[31] S.S. Huang, T.J. Green, and B.T. Loo. Datalog and emerging applications: an interactive tutorial. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 1213–1216. ACM, 2011.

[32] M. Interlandi, L. Tanca, and S. Bergamaschi. Datalog in time and space, synchronously. In L. Bravo and M. Lenzerini, editors, *AMW*, volume 1087 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[33] P. Koutris and D. Suciu. Parallel evaluation of conjunctive queries. In *Proceedings of the 30th ACM symposium on Principles of Database Systems*, pages 223–234. ACM Press, 2011.

[34] G. Lausen, B. Ludäscher, and W. May. On active deductive databases: The statelog approach. In B. Freitag, H. Decker, M. Kifer, and A. Voronkov, editors, *Transactions and Change in Logic Databases*, volume 1472 of *Lecture Notes in Computer Science*, pages 69–106. Springer Berlin Heidelberg, 1998.

[35] J. Leite and L. Soares. Adding evolving abilities to a multi-agent system. In *Proceedings of the 7th International Conference on Computational Logic in Multi-agent Systems*, CLIMA VII'06, pages 246–265. Springer-Verlag, 2007.

[36] J.A. Leite, J.J. Alferes, and L.M. Pereira.

Minerva – a dynamic logic programming agent architecture. In *Revised Papers from the 8th International Workshop on Intelligent Agents VIII*, ATAL, pages 141–157. Springer-Verlag, 2002.

[37] J. Lobo, J. Ma, A. Russo, and F. Le. Declarative distributed computing. In E. Erdem, J. Lee, Y. Lierler, and D. Pearce, editors, *Correct Reasoning*, volume 7265 of *Lecture Notes in Computer Science*, pages 454–470. Springer, 2012.

[38] B.T. Loo, T. Condie, et al. Declarative networking: language, execution and optimization. In S. Chaudhuri, V. Hristidis, and N. Polyzotis, editors, *SIGMOD Conference*, pages 97–108. ACM, 2006.

[39] B.T. Loo, T. Condie, et al. Declarative networking. *Communications of the ACM*, 52(11):87–95, 2009.

[40] B.T. Loo, H. Gill, C. Liu, Y. Mao, W.R. Marczak, M. Sherr, A. Wang, and W. Zhou. Recent advances in declarative networking. In C. Russo and N. Zhou, editors, *Practical Aspects of Declarative Languages*, volume 7149 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012.

[41] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J.M. Hellerstein. Graphlab: A new framework for parallel machine learning. In P. Grünwald and P. Spirtes, editors, *UAI*, pages 340–349. AUAI Press, 2010.

[42] L. Lu and J.G. Cleary. An operational semantics of starlog. In G. Nadathur, editor, *Principles and Practice of Declarative Programming*, volume 1702 of *Lecture Notes in Computer Science*, pages 294–310. Springer Berlin Heidelberg, 1999.

[43] J. Ma, F. Le, D. Wood, A. Russo, and J. Lobo. A declarative approach to distributed computing: Specification, execution and analysis. *Theory and Practice of Logic Programming*, 13:815–830, 2013.

[44] G. Malewicz, M.H. Austern, A.J.C Bik, J.C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 135–146. ACM, 2010.

[45] W.R. Marczak, P. Alvaro, N. Conway, J.M. Hellerstein, and D. Maier. Confluence analysis for distributed programs: A model-theoretic approach. In Barceló and Pichler [17], pages 135–147.

[46] J.A. Navarro and A. Rybalchenko. Operational semantics for declarative networking. In A. Gill and T. Swift, editors, *Proceedings 11th International Symposium on Practical Aspects of Declarative Languages*, volume 5419 of *Lecture Notes in Computer Science*, pages 76–90, 2009.

[47] V. Nigam, L. Jia, B.T. Loo, and A. Scedrov. Maintaining distributed logic programs incrementally. *Computer Languages, Systems & Structures*, 38(2):158–180, 2012.

[48] V. Nigam and J. Leite. A dynamic logic programming based system for agents with declarative goals. In *Proceedings of the 4th International Conference on Declarative Agent Languages and Technologies*, DALT, pages 174–190. Springer-Verlag, 2006.

[49] D. Saccà and C. Zaniolo. Stable models and non-determinism in logic programs with negation. In *Proceedings of the Ninth ACM Symposium on Principles of Database Systems*, pages 205–217. ACM Press, 1990.

[50] M. Shapiro, N.M. Preguiça, C. Baquero, and M. Zawirski. Convergent and commutative replicated data types. *Bulletin of the EATCS*, 104:67–88, 2011.

[51] M. Spielmann. Verification of relational transducers for electronic commerce. *Journal of Computer and System Sciences*, 66(1):40–65, 2003.

[52] W. Vogels. Eventually consistent. *Communications of the ACM*, 52(1):40–44, 2009.

[53] C. Zaniolo, N. Arni, and K. Ong. Negation and aggregates in recursive rules: the ldl++ approach. In S. Ceri, K. Tanaka, and S. Tsur, editors, *Deductive and Object-Oriented Databases*, volume 760 of *Lecture Notes in Computer Science*, pages 204–221. Springer Berlin Heidelberg, 1993.

[54] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1:7–18, 2010.

[55] D. Zinn, T.J. Green, and B. Ludäscher. Win-move is coordination-free (sometimes). In *Proceedings of the 15th International Conference on Database Theory*, pages 99–113. ACM Press, 2012.

Public Data and Visualizations: How are Many Eyes and Tableau Public Used for Collaborative Analytics?

Kristi Morton[1], Magdalena Balazinska[1], Dan Grossman[1],
Robert Kosara[2], and Jock Mackinlay[2]

[1]University of Washington
{kmorton,magda,djg}@cs.washington.edu

[2]Tableau Software
{rkosara,jmackinlay}@tableausoftware.com

ABSTRACT

Recently, online visual analytics systems have emerged as popular tools for data analysis and sharing. The database community has an important role to play in shaping the design and implementation of these new types of systems. Little, however, is known about how these systems are used today. In this paper, we address this shortcoming by presenting an analysis of usage patterns of Many Eyes and Tableau Public, two popular Web-based, collaborative visual analytics systems.

1. INTRODUCTION

As data has become more publicly available on the Web, for example, through local and national government initiatives such as the Open Data movement [6], a broader audience has emerged as data consumers and knowledge-seekers (referred to as *data enthusiasts*) [11]. These people are not statisticians or programmers, yet want to use data to answer a question or solve a problem. A typical example is a news reporter who wants to use data and visualizations to illustrate a story and make it available online (*e.g.,* on her blog).

Over the past few years, increasingly many online data visualization systems have appeared to meet the demands of such users for data analysis and sharing [2, 1, 4, 18, 5, 9, 10]. The core functionality of these systems is threefold: (1) They enable users to *visually* explore their data: users have access to a graphical user interface through which they can easily create various charts and graphs. Importantly, through these interfaces, users are basically executing queries to filter, group by, and aggregate datasets. (2) These systems also facilitate the integration and study of *multiple* datasets. (3) Finally, they support *collaboration* between users through sharing visualizations and data *online* for both viewing and editing by others [12, 20]. These services are thus a new type of easy-to-use data management and analytics systems.

While different systems have different architectures, several are based on the integration of a visualization front-end with a database management system back-end [19, 9]. For example, Tableau supports analysis across a variety of structured, heterogeneous data sources (*e.g.,* delimited text files, cubes, data marts, and databases), and issues live queries to these sources to obtain the necessary data to render each visualization. With live query support, interactive analysis is possible: visualizations can be altered on-the-fly and multiple data sources can be joined together.

Unlike Tableau Public, Many Eyes visualizations are created and published through a Web browser. Either structured (*i.e.,* tables) or unstructured (*i.e.,* bag of words) data is ingested through a browser using cut-and-paste operations from a text file up to 5 MB in size. Once a visualization is chosen for a given data set it cannot be arbitrarily altered nor combined with other data. Moreover, while both systems share many of the same visualization types (*i.e.,* bar, line, text, pie, area, scatter, and maps), Many Eyes includes a number of unique text analysis techniques that are not available in other systems. Such visualizations include word clouds, phrase nets, and word trees.

Despite their growing popularity, little is known about how these systems are being used. Even basic statistics such as the number of users are often not published (*e.g.,* Fusion Tables [9]), let alone any details of user activity. The most prominent system, Many Eyes, started in early 2007, and initial studies [8] indicated a significant uptake, as well as collaboration between users; but there have been no follow-up studies on usage, nor have there been comparable studies of other web-based or web-centric visual data analysis systems. Shortly before Many Eyes, in December 2006, Swivel.com was launched. Swivel was much simpler and less academically ambitious than Many Eyes, but run as a start-up rather than an experiment. It shut down in summer 2010, casting doubt on whether there was a market for web-based visual data analysis systems. At the same time, there is clearly broad interest in data integration, analysis, and visualization. *The New York Times*, *The Washington Post*, *The Guardian*, and other news media are not only increasingly using visual data analysis as part

System	Start Date	# Visualizations	# Workbooks	# Datasets	Users
Many Eyes	January 1, 2007	149,395 (3.2/user)	n/a	358,880 (7.8/user)	46,048
Tableau Public	February 10, 2010	269,609 (11/user)	73,404 (3/user)	107,596 (4.4/user)	24,563

Table 1: Summary of the collected data from Many Eyes and Tableau Public, from each system's inception until December 31, 2012.

of news stories, but also experimenting with more sophisticated types of visualizations.

As our society continues to become "data-enabled", it is important that we continue to improve data management and analysis tools. If we are to build better online data visualization and sharing systems, the first step is to understand how they are being used today. The key contribution of this paper is to shed light on this exact question: *How are online visual data analysis and sharing systems being used?*

We take a first step toward answering this question through a longitudinal measurement study of two popular online data visualization and analysis systems: Tableau Public [4] and Many Eyes [18, 3]. Both systems allow users to create visualizations online, and both are free to use. Tableau Public requires the download of a Windows-only client, while Many Eyes is used entirely in the browser. Both systems provide a variety of different visualization techniques, which not only generate static images, but which the viewer can interact with in the browser. The data used in visualizations can be downloaded in both systems.

We tackle the question of how both of these systems are being used from the perspective of the database community. Through our study, we thus focus on the following core set of questions: (1) How popular are these systems? How many users do they attract and how active are these users? (2) How heavily do users leverage the collaborative features of these tools? (3) What do users actually do with the data? How do they analyze it? How much data (in terms of relation cardinality and degree) do users choose to visualize at any given time? And finally (4) Do users integrate multiple data sources in their visualizations? And how do they perform these integrations? To the best of our knowledge, this is the first formal study of these types of systems.

2. METHOD

Our study is based on traces of Many Eyes and Tableau Public as summarized in Table 1. The traces span six and three years respectively and include detailed information about the data and visualizations that are published to each system. The Tableau Public trace also includes detailed traffic and impression data for each visualization. For Tableau Public, each workbook specifies the data sources analyzed (including all schema metadata), the types of visualizations produced, and all of the specific VizQL[1] definitions [17] that produce each visualization. For Many Eyes, metadata was collected from the visualization types used and inferred from data sets uploaded.

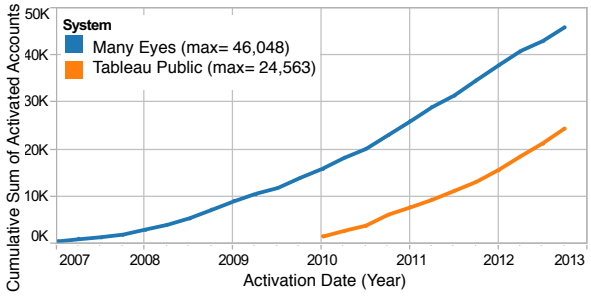


Figure 1: Cumulative growth of Many Eyes and Tableau Public activated user accounts.

3. RESULTS

We present the key results of our analysis, organized around our four core questions related to the user-base, collaborations, single-dataset analytics, and analytics of integrated datasets.

3.1 User-base

The first question that we ask is whether web-based visual analytics systems are at all popular. To answer this question, we measure the size of the user-base for each system. Figure 1 shows how the systems are growing over time in terms of the number of opened accounts. As the figure shows, since its inception in January 2007, Many Eyes, has grown to over 46,000 authors who have published over 358,000 data sources and more than 149,000 visualizations. For Tableau Public, its user-base includes 24,500 authors who have contributed over 73,000 workbooks, 107,500 datasets, and 269,000 visualizations (Table 1). We define authors to be users who have published at least one data set or visualization. **These systems thus have moderate numbers of users today, but their popularity is continuing to grow significantly each year.**

The natural next question is how active are these authors over time. Interestingly, as Table 2 shows, **half the users (or almost half) are *one-time* users who publish only one dataset or visualization. The remaining users are mostly *light* users who publish two to four**

[1]Visual Query Language (VizQL) a formal declarative language for describing visualizations

	Number of Data Sources Published				
	1	≤2	≤3	≤4	≤5
Many Eyes	44%	65%	76%	83%	86%
Tabeau Public	45%	63%	73%	79%	83%
	Number of Visualizations Published				
	1	≤2	≤3	≤4	≤5
Many Eyes	52%	72%	82%	87%	90%
Tabeau Public	53%	71%	80%	85%	88%

Table 2: Cumulative fraction of users who publish up to a given number of data sources or visualizations (e.g., 80% of Tableau users publish 3 visualizations or less).

visualizations. Only 10% to 17% are *prolific* users who publish five or more data sets or visualizations. Despite the significant age difference between the two systems, it is still interesting to note the most prolific author who is not directly affiliated with either system: on Many Eyes such a user contributed 1,617 data sets and visualizations; and on Tableau Public 2,927 data sets and visualizations were published by one user.

Since the majority of users are one-time or light users, does it mean that most of the activity in the systems is due to novices? Figure 2 confirms this hypothesis. In the figure, we group users into cohorts based on the quarter in which they publish their first visualization (workbook on Tableau Public). For each quarter, the figure shows the fraction of active accounts that come from each *returning* cohort. For example, in the third quarter, 8% of active accounts in Many Eyes belong to the second cohort and 11% of active accounts belong to the first cohort. The remaining 81% active accounts (not shown) belong to users who joined the system that quarter. On average only 17% of active accounts in Many Eyes belong to returning users from any cohort. The average is 31% for Tableau Public. **Hence, web-based data analysis systems today need to provide good support for novices.**

One hypothesis for high user churn is bad system performance.According to a study published on Web users' tolerable waiting time [15], two seconds is considered an acceptable waiting time for loading Web pages. We measure, however, that 84% of all visualizations on Tableau Public take less than two seconds to load (includes both query and rendering time) and 98% are under ten seconds (the accepted limit for keeping a user's attention focused on a given task [14]). Although attitudes and expectations change over time, the basic capability of human attention has not changed over the decades [7, 14]. Thus, our results indicate that the majority of load times should not negatively impact Tableau Public's users. Performance alone thus cannot explain the high degree of user churn. It could, however, simply be that the systems do not offer the visualization capabilities users want (*e.g.,* limited to no support for unstructured data). Users explore the systems but walk away when they find them unsuitable for their needs.

If we frame these retention results in the context of other free, web-based services such as Twitter, we see that low retention after initial use is common. According to a 2009 Nielsen report [13] only 40% of Twitter users returned to use the site after the first month.

Bottom Line: Web-based visual analytics systems continue to attract thousands of users, but most of them use these systems lightly right after registering and then stop. Only a small fraction of users grows into power users. The implication for the database community is that Web-based visual analytics systems must be geared toward supporting novice users and there is significant room for improvement in retaining these users.

3.2 User Interaction and Collaboration

Since both systems are designed for sharing visualizations and collaboratively analyzing data, we explore the frequency of viewership, collaboration, and sharing in this section.

3.2.1 Viewership

Based on a distinct count of user cookies, we found that there are approximately 52 million unique visitors to Tableau Public. **Visitors are thus several orders of magnitude more numerous than the authors** (only ≈ 24,500 authors). Additionally, we found that the top 50% of all Tableau Public traffic is attributed to 244 distinct workbooks (or 0.3% of all workbooks). For Many Eyes, however, we did not have access to the equivalent traffic and viewership information.

3.2.2 Collaboration

On Tableau Public and Many Eyes, users can download, edit, and republish any visualization and supporting data set. To get a sense of the degree of such collaborative activities among authors, we explore how often authors take existing content and evolve it for their own analytical needs (*e.g.,* by changing the visualization content to explore some other dimension or measure) and then republish it with their insights. In our approach, we trace the provenance of Tableau Public workbooks that were created by one author and edited and republished by a different author (called a *derivation*). Figure 3(top) shows that a workbook is about five times more likely to be derived if it is not the author's first publication. Nevertheless, the probability of derivation remains small at only 6%. Very few workbooks are derived more than once. Only 28 workbooks were derived > 4 times.

In Figure 3(bottom) we observe two types of collaborative behaviors. Some workbooks are derived multiple times by alternating between the same two authors as in a *Direct Collaboration* while others are derived by a dif-

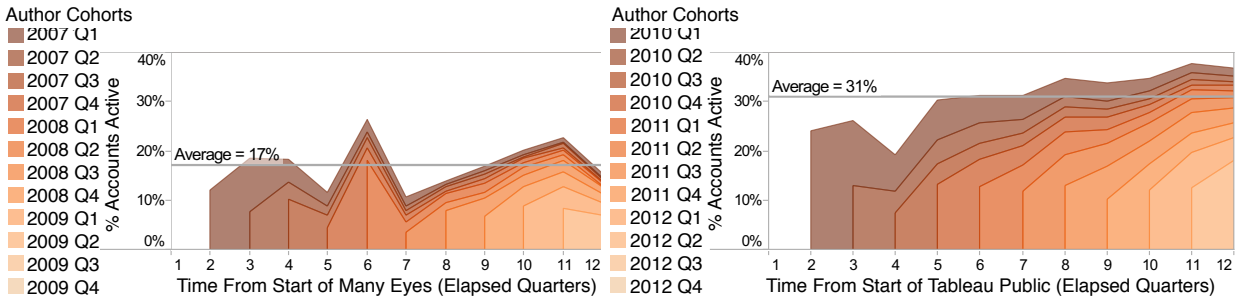


Figure 2: Many Eyes (left) and Tableau Public (right) author cohorts for the first 12 quarters (3 years). Authors are grouped into cohorts based on the quarter in which they published their first visualization or workbook. The fraction of authors that returned to the site to publish a dataset or visualization is shown as the *percent of accounts still active* for each quarter.

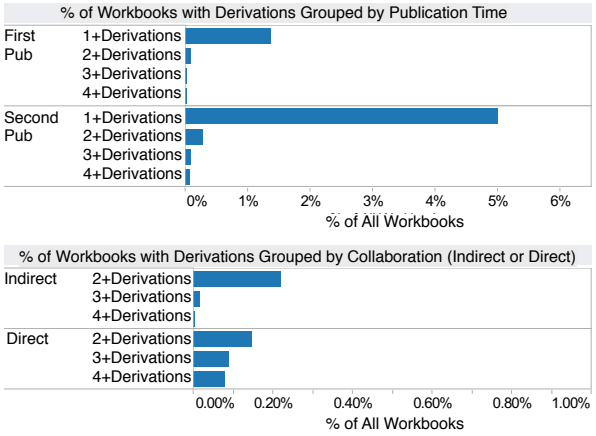


Figure 3: Derived Tableau Public workbooks partitioned by publication time (top) and nature of collaboration (bottom).

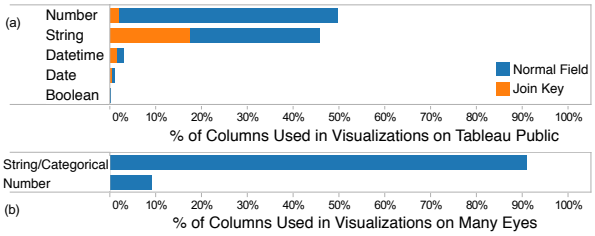


Figure 4: Data types in visualizations on Tableau Public (a) and Many Eyes (b). Tableau Public is split between numbers and strings, and word-oriented Many Eyes is heavily skewed toward strings.

ferent author each time as in an *Indirect Collaboration*.

No equivalent derivation information is available for Many Eyes. However, in order to get a sense of the degree of influence one author's contributions have on other authors, we measure how often authors reuse data uploaded and shared by others for their visual analysis in Many Eyes. We find that only 6% of datasets are used by multiple authors, which is consistent with Tableau's workbook derivation statistics, and that 20% of datasets are used in multiple visualizations. We cannot compute this statistic for Tableau Public because published workbooks make a copy of the data being visualized.

3.2.3 *New Content Published: Data Types*

We next consider what are the predominant data types that are being visualized in both systems. First, in Figure 4(a), we see that `Number` (51%) and `String` (44%) are the most common data types in visualizations on Tableau Public. It is interesting that their use is fairly balanced, while intuition would indicate that numbers might be more common due to the quantitative nature of business analytics. The `Number` data type includes both integers and reals. Finally, we see fewer specialized types such as `Datetime` and `Date`, which indicates that visualizations of time-based data are less prevalent.

Many Eyes, however, has a skewed distribution of `String/Categorical` types. In Figure 4(b), we see that 91% of columns on Many Eyes are of this type. This finding is consistent with Many Eyes's greater emphasis on text-based visualizations (*i.e.,* word clouds, phrase nets, and word trees) that are not available elsewhere.

Bottom Line: Online visual analytics systems are **read-heavy today: Orders of magnitude more people are viewers compared to authors**. Additionally, as is typically the case for database access patterns, viewership is skewed toward a **small fraction of *hot* visualizations**. Interestingly, while publishing visualizations is common, **collaborations among users remain infrequent**. Incentivizing and supporting collaborations thus remain critical challenges for these systems.

3.3 Single-Dataset Analytics

Today's online visual analytics systems are designed for small data. Most of these systems put a bound on the size of datasets that can be processed. On Many Eyes, data sizes are limited to 5MB, while on Tableau Public, each user gets a 50MB account and a visualization can operate on at most 100,000 rows. Interestingly, we find

System	Number of Rows in Visualizations				
	≤100	≤1K	≤10K	≤50K	≤100K
Many Eyes	63%	90%	98%	99%	100%
Tabeau Public	28%	53%	84%	95%	100%

Table 3: Cardinality of visualized relations.

System	Number of Columns in Data Source				
	≤2	≤10	≤20	≤100	≤300
Many Eyes	49%	84%	93%	99%	100%
Tabeau Public	2%	28%	52%	90%	99%

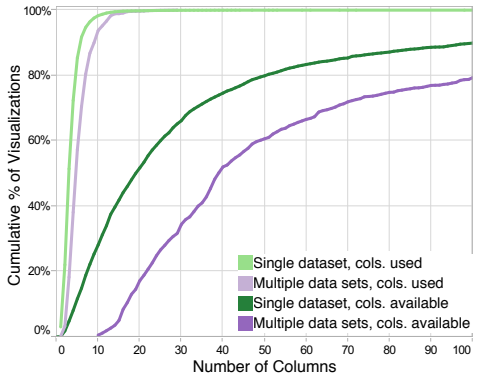


Figure 5: Degree of input relations (top). CDF of the number of columns in visualizations with one or multiple (joined) data sets in Tableau Public (bottom).

that 90% of user accounts in Tableau Public use less than half of their 50MB quotas. Hence most users do not push the pre-set limit. The focus on small-size datasets affects the size of the visualized relations as shown in Table 3: **the majority of visualized relations stays well below the pre-set cap of 100K rows.**

Tableau Public also offers a paid (*a.k.a.*, *Premium*) tier, which allows accounts to go beyond the 100,000 rows limit. These accounts (along with some accounts on Many Eyes) visualize more than an order of magnitude more data, which seems to **imply the need for the online visualization of bigger data too**.

Interestingly, datasets in both systems contain on average a large number of attributes, especially in Tableau Public, where the median dataset has 20 attributes and the top 10% have more than 100 attributes. Only a small fraction of these attributes, however, is visualized simultaneously as shown in Figure 5: 52% of visualizations with a single data source use at most 3 columns and 90% use at most 6. A similar trend appears for visualizations over integrated data sources as we discuss further in Section 3.4. The figure shows results for Tableau Public. No equivalent information was available for Many Eyes.

Table 4 shows the breakdown of the most common visualization types used for a given number of columns. The values denoted with a '⋆' in Table 4 show that a second visualization type was within 5% from the top choice for that given number of columns. For single data sources, we see that the text table is the most common

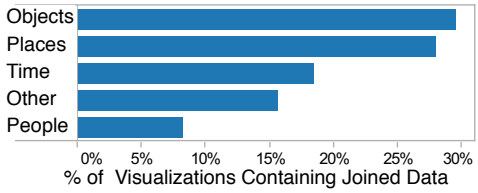


Figure 6: Common semantic entities of join keys in visualizations with multiple (joined) data sources.

type when there is only one data column present in the visualization. As the number of columns increases, we see a shift in visualization techniques used: bar views become the dominant technique for 2–4 columns and maps are the most popular for 5–8 columns.This behavior is not too surprising since map views always include two virtual columns that represent the latitude and longitude coordinates.

Bottom Line: Most visualizations have modest data sizes, well below the limits set in existing systems although some users with special privileges visualize datasets with more than one million rows. Visualizations also focus on a small number of attributes at any one time, even though many more attributes are available. Finally, as the number of columns used increases, so does the complexity of the visualization type (*e.g.,* maps require more columns than other types like bar views.)

3.4 Integrating Multiple Data Sets

In this section, we study the trends in data and visualization on Tableau Public in the context of data integration from multiple data sources. We omit Many Eyes from this section because the platform currently does not support data integration.

3.4.1 Semantic Entities for Data Integration

On Tableau Public, there are 5,532 visualizations that were created by joining multiple data sets. Of these visualizations, we ask *how do authors combine data sets for their analysis?* To answer this question we manually categorized all of the join keys for the 5,532 visualizations (2%) that have integrated data to get a sense of the most popular semantic entities. This process entailed inspecting the column name, data type, and data values of each join key. In the case where the column name was in a foreign language, we used Google Translate on the name and (in some cases) values of that column. If we were still unsure, we opened the workbook to inspect the visualization that was associated with that join key. Figure 6 summarizes the semantic entities of the join keys in five different categories: people, places, time, objects, and other. The people category contains any information pertaining to people, including names and demographics. The places category is restricted to

Number of Data Sets	Number of Columns in Visualization 1	2	3	4	5	6	7	8	9	10	11	12	13	14
One	Text (68%)	Bar (53%)	Bar (47%)	Bar (32%)	Map* (27%) Bar	Map (32%)	Map* (27%) Bar	Map* (26%) Bar	Bar* (25%) Map	Line (32%)	Map* (24%) Bar	Circle (34%)	Bar (30%)	Bar (20%)
Multiple	Text (75%)	Bar (48%)	Bar (50%)	Bar (41%)	Map* (22%) Bar	Map (35%)	Map (40%)	Text (46%)	Map (49%)	Map (56%)	Scatter* (38%) Map	Scatter* (36%) Bar	Circle (64%)	Map (42%)

Table 4: Most common visualization types for different numbers of columns in the visualization.

geolocations and other identifying characteristics such as zip codes, regions, states, countries, continents, etc. As expected, the time category refers to dates and date times and objects refer to any physical entity that is not a person, place, or time. Objects consist mainly of opaque identifiers like alphanumeric product codes as well as more well-known, descriptive entities such as "university", "department", or "team". As the figure shows, users integrate data most often by adding attributes to the same object identifier (30%), or by bringing together items located in the same places (28%) or occurring at the same time (18%).

3.4.2 Data Columns per Visualization

Figure 5 shows the number of available and visualized columns for single and integrated datasets. We see that visualizations on top of integrated datasets are significantly richer as they display a larger number of columns than visualizations over a single data source. For example, 43% of visualizations over integrated data use 5 or more columns, compared to only 15% of visualizations over a single dataset. Furthermore, we see a familiar trend as with single data sources: there is a sizable gulf between the number of columns used and the number of columns available in the integrated data sources.

Bottom Line: Data integration often occurs by combining multiple attributes about the same uniquely identified entities from different data sources. This type of data integration is more common than simply placing multiple entities at the same location or at the same point in time, although the latter two dominate when considered together. This finding is especially interesting for data integration tools. For example, a recent tool provides recommendations of potentially useful data to integrate with a given database [16]. This tool does not consider joining on place or time. It only considers extending semantic entities with additional attributes, which covers less than half of all integration scenarios. **Additionally, integrated visualizations tend to be more complex (*i.e.,* use more columns and have more columns available) than single-source ones. Interestingly, the distribution of the most common visualization types for a given number of columns is similar for visualizations over one or more datasets.**

4. CONCLUSIONS

We studied four primary dimensions of two popular online visual analytics systems: (1) what types of users are leveraging these systems and how heavily, (2) how are users collaborating and interacting with the published content, (3) how do users analyze a single-dataset and (4) how they integrate data sources. We find that web-based visual analytics systems have much room for growth: they attract large numbers of users but most users do not push the limit of what these tools can do.

Acknowledgments. This work is partially supported by NSF CDI grant IIA-1028195.

5. REFERENCES

[1] Gapminder. `http://www.gapminder.org/`, 2012.
[2] iCharts. `http://www.icharts.net/`, 2012.
[3] Many Eyes. `http://many-eyes.com/`, 2012.
[4] Tableau Public. `http://www.tableaupublic.com/`, 2012.
[5] ViewShare: Interfaces to our Heritage. `http://viewshare.org/`, 2012.
[6] T. Berners-Lee. The year open data went worldwide. TED Talk, `http://on.ted.com/eU5`, 2009.
[7] S. Card et al. The information visualizer, an information workspace. In *CHI*, 1991.
[8] C. Danis. et al. Your place or mine?: visualization as a community component. In *CHI*, 2008.
[9] H. Gonzalez et al. Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud. In *SOCC*, 2010.
[10] H. Gonzalez et al. Google fusion tables: Web-centered data management and collaboration. In *SIGMOD*, 2010.
[11] P. Hanrahan. Analytic database technologies for a new kind of user: the data enthusiast. In *SIGMOD*, 2012.
[12] J. Heer et al. Design considerations for collaborative visual analytics. In *IEEE VAST*, 2007.
[13] D. Martin. Twitter Quitters Post Roadblock to Long-Term Growth. `http://blog.nielsen.com/nielsenwire/online_mobile/twitter-quitters-post-roadblock-to-long-term-growth/`, 2009.
[14] R. Miller. Response Time in Man-Computer Conversational Transactions. In *Proc. of the AFIPS Fall Join Computer Conference*, volume 33, 1968.
[15] F. Nah. A Study on Tolerable Waiting Time: How Long Are Web Users Willing to Wait? In *Behaviour and Information Technology*, volume 23, 2004.
[16] A. D. Sarma et al. Finding Related Tables. In *SIGMOD*, 2012.
[17] C. Stolte et al. Polaris: a system for query, analysis, and visualization of multidimensional databases. *IEEE TVCG*, 2002.
[18] F. B. Viegas et al. Many eyes: A site for visualization at internet scale. *IEEE TVCG*, 13(6), 2007.
[19] K. Wesley et al. An Analytic Data Engine for Visualization in Tableau. In *SIGMOD*, 2011.
[20] W. Willett et al. Commentspace: Structured support for collaborative visual analytics. In *CHI*, 2011.

ENORM: An Essential Notation for Object-Relational Mapping

Alexandre Torres
Instituto de Informática, UFRGS
Porto Alegre, Brazil
atorres@inf.ufrgs.br

Renata Galante
Instituto de Informática, UFRGS
Porto Alegre, Brazil
galante@inf.ufrgs.br

Marcelo Pimenta
Instituto de Informática, UFRGS
Porto Alegre, Brazil
mpimenta@inf.ufrgs.br

ABSTRACT

Despite the growing adoption of object-relational mapping frameworks, UML and its most widespread extensions do not represent these mappings in a platform independent way. Maintaining mappings scattered in the code is difficult and error prone, specially if the schema is large and serves several systems. This paper proposes ENORM, a notation that extends class models representing all the essential mappings. ENORM is platform independent, providing a meta-model based on design patterns employed by three frameworks of Java, Ruby, and Python languages. An empirical evaluation indicates that ENORM performs well in comparison to separated models.

1. INTRODUCTION

Relational databases (RDB) are the backbone of information systems, and nobody knows when (or if) this will change [3]. However, the Impedance Mismatch Problem (IMP) continues to haunt object oriented designs that tend to underestimate the Object-Relational Mapping (ORM) difficulties.

In the past decade we saw a growing adoption of ORM frameworks by information system developers of distinct platforms such as Java, C#, Python, and Ruby on Rails. These frameworks have most of their resources based upon established patterns [6, 11, 14], and its use spread a more standardized approach for the IMP. Nevertheless, mappings scattered in the code, annotations and/or XML files are difficult to read, understand, and reason about changes.

The Model Driven Architecture (MDA) proposes that models take on the main role on the system development process [4, 17]. For an effective MDA approach, the information represented by models should be coherent, integrated, and computable, so that automatic transformations could turn models into executable system [16]. The UML notation lacks a specific notation for persistence, or to map classes to database. The absence of mapping information poses a challenge for developing transformations.

This paper presents ENORM, a general purpose notation that represents the essential structural concepts of ORM by extending the UML class model with a profile, and offering a concise set of new visual elements specific for ORM designs. These essential concepts are based upon persistence patterns adopted by distinct ORM frameworks in the market. The goal of ENORM is to facilitate the design by the clear application of ORM patterns, document mappings with a platform independent notation, and be a repository for MDA transformations and code generation.

The focus of ENORM is designing with structural patterns within a domain modeling logic, with objects of the domain incorporating both behavior and data [11]. ENORM does not encompasses the design of queries or the use of dynamic diagrams.

A controlled experiment was performed to evaluate modeling using ENORM. The results indicates that using only models, ENORM has a lower mean of missed goals than separated models.

This paper is organized as follows: Section 2 presents related works; Section 3 presents the notation; Section 4 presents the meta-model; Section 5 presents examples using ORM tools; Section 6 presents limitations and special cases; Section 7 summarizes empirical evaluation; and section 8 has the conclusions.

2. RELATED WORK

The agile database modeling [1] is a well known proposal for database modeling using UML extensions. It is mainly based upon the class diagrams for representing data models with a set of stereotypes. The Object Management Group (OMG) has also an underway proposal for data modeling representation [18]. None of the two notations have the focus on ORM, ORM frameworks or patterns.

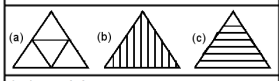
The Entity-Framework proposes the EDM model based on the EER notation [5]. EDM is focused on multipurpose conceptual modeling for distinct persistence mechanisms using the .NET platform. ENORM takes a distinct approach by encompassing general ORM design patterns, in a cross-platform way.

On a previous work, we proposed a notation based upon the Java Persistence API (*JPA*) standard named MD-JPA [20]. Although *JPA* is a standard, it is focused at the Java platform, including many concepts particular only to Java. It was not clear at that time what concepts are particular for *JPA*, and what was missing from other frameworks and platforms.

3. ESSENTIAL NOTATION (ENORM)

The notation here proposed is a lightweight UML profile, represented by a set of graphical extensions for class models, encompassing the essential structural concepts of ORM. ENORM was designed to be easily understood by developers and rich enough for MDA tools, allowing the specification of the relevant persistence details or hiding what can be inferred.

Table 1. New visual elements and their meaning

Graphical Element	Description
class || *table1, table2, ...*	Persistent class. Optional tables
property || *column def*	Definition of the column mapping
«PK»	Part of the primary key
«Embed»	Dependent or embedded
	Inheritance types: (a) Flat, (b) Vertical or (c) Horizontal. Discriminators can be specified.
join table = *table*	Association table name
join columns = *col1, col2, ...*	Columns that implement association
«Override» *property path* || *column def*	Override an inherited or embedded property or association mapping
property or assoc. end θ	Transient feature should not be mapped
«Map»	Set key property of qualified associations

ENORM elements (Table 1) are derived from ORM patterns following the domain model pattern [11]. Besides, ENORM reflects common practices of various ORM frameworks, such as *activerecord* for *Ruby (AR)*, *JPA*, and *SQLAlchemy* (SA) for Python [2, 12, 19].

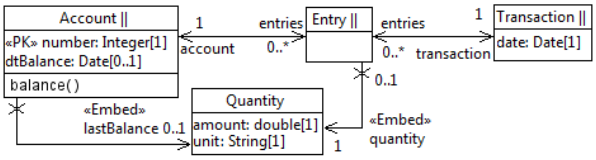


Figure 1: Simple Transaction example

A *Persistent* class (marked with "||") represents a class implemented as an *Active Record*, *Data Mapper,* or mapped in such a way by a framework. The class is persisted by a table with the same name; or one or more specified tables. Each property of a persistent class maps to a column, that can be detailed in the model. Associations between persistent classes are implemented with Foreign Keys (FKs) detailed by join columns and tables. Inheritance can be *flat* for single table pattern; *vertical*, for joined table pattern; or *horizontal* for the concrete table pattern. Non persistent classes can be persisted by associations marked as *embed* within *persistent* classes. A persistent class can have transient properties by using the transient symbol.

3.1. A simple example

Figure 1 shows a simple design for the *Accounting* patterns [10]. *Account*, *Entry*, and *Transaction* are persistent classes, each persisted by tables with the same name. *Account* has a meaningful Primary Key (PK) named *number*. *Entry* and *Transaction* will also have PKs, but they are not specified (inferred).

Quantity is not persistent and does not correspond to a table. However, each *Entry* instance refers to a *Quantity* with the *Embed* stereotype. Since the upper multiplicity is one, quantity association is persisted along the *Entry* table, by columns *amount* and *unit*. *Quantity* is similarly embedded by *Account*.

Finally, the associations between persistent classes are mapped as FKs connecting the PKs of each table. *Entry* will have a column referencing account *number* and a column referencing the PK of Transaction.

3.2. A not so simple example

Database information systems usually refer to centralized databases serving multiple systems, that must adapt to the existing schema. Often that means a break between nomenclature used by the system and the database, and a more complicated mapping.

Figure 2 introduces the *SummaryAccount* class, that aggregates accounts implementing multiple summary accounts [10]. Each account can be part of one or more summary accounts, and the *entries* of the summary are the union of all underlying *DetailAccount* instances.

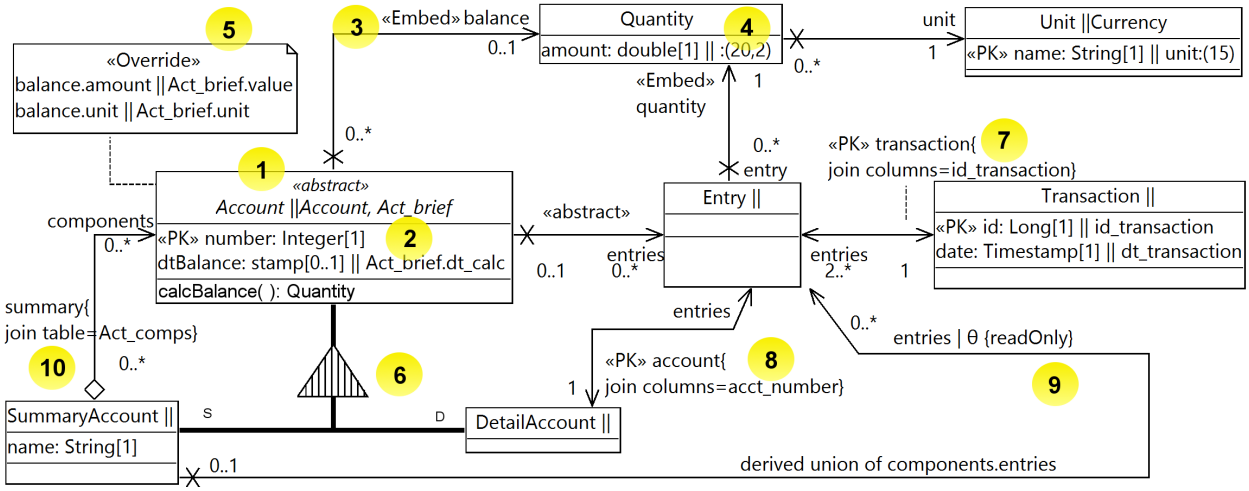


Figure 2: Summary account example

The *Unit* class now replaces the free text unit property. Several changes were introduced in the mapping, and Figure 3 presents the database derived from the model:

1. *Account* is mapped to two tables joined by the PK. The table *Act_brief* has an FK to *Account*.

2. Property *dtBalance* is mapped to column *dt_calc* on table *Act_brief*.

3. *Quantity* now refers to a *Unit* persisted by the *Currency* table. When *Account* references a *Quantity*, it stores a reference (FK) to the *Currency* table.

4. Property *amount* with default SQL precision/scale of *(20,2)*.

5. *Account* overrides the *quantity*: *amount* is persisted by the column *value* of table *Act_brief*; the association end *unit* is stored by the column *unit* in table *Act_brief,* that references the table *Currency*. By default, all columns would be stored along the primary table *Account*.

6. The account inheritance tree is persisted with the joined table pattern. Each class has its own tables, and each PK of the specializations refers to the *Account* PK. The discriminator column can assume 'S' or 'D'.

7-8. *Entry* refers to *Transaction* with a column named *id_transaction,* and refers to *DetailAccount* with a column named *acct_number,* setting the PK of *Entry*.

9. *Account* defines the association *entries* as *abstract*. *DetailAccount* implements *entries* by an FK, but for *SummaryAccount* this association is derived from its components. The transient symbol tells that this association should not be stored by an FK column.

10. The *components* association is many-to-many, and therefore is mapped by an association table. The *join table* specifies that this table is *Acct_Comps*. By default it will have FK columns referring to *Account* and *SummaryAccount*.

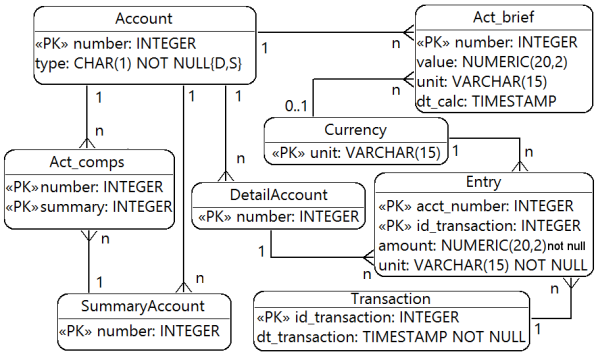


Figure 3: Database model of account example

3.3. Maps

UML allows the specification of *Qualified Associations* that represents partitions in the association between two classes. When the qualified property has an upper value of one, the association represents what is commonly referred as *Map* or *Dictionary* by object-oriented languages [22].

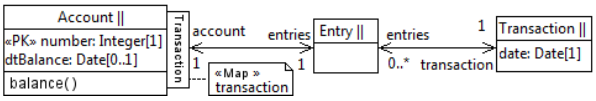


Figure 4: Map with key reference

Figure 4 presents an example where the association end of *Account* is a map with a <*Transaction, Entry*> form, where the qualified variable of type *Transaction* is the key. The *Map* stereotype allows the specification that the key is in fact the transaction property of entry, what is common on ORM. The goal is that when the user adds a pair <*tx*, *ey*> to the map, it will associate *ey* both with the account and the *tx* transaction.

The property key can also be user defined, derived from a complex operation. In such cases, it can be a read-only map. Qualified associations without a property key are also allowed. In the transaction example, the map would be persisted in a separate many-to-many table, instead of using the association between entries and transactions. Qualifier properties can also assume non-persistent and scalar types.

4. ENORM METAMODEL

Backing up the visual notation there is a profile providing compatibility between ENORM and UML implementations, such as the Eclipse UML2 package. Figure 5 summarizes the stereotypes, the extended UML elements, meta-classes, and its properties and relationships detailed by this section.

The *Persistent* Stereotype is applied to a class marking the class with the double bars (‖) of Table 1. The *source* property allows the direct definition of one *Table*, a reference to an already defined table by *TableRef*, or a *JoinedSource* comprising two or more tables connected by *JoinColumn* objects. The use of *Table* or *TableRef* determines the class that "owns" the table definition, preventing duplicate specification of tables. If *source* is unspecified, the class is persisted by a table with the same name of the class.

Properties owned by a persistent class are, by default, persisted, and scalar values are stored as columns. The *ColumnMapping* Stereotype allows the definition of these columns, informing column name, if it accept nulls, length, precision, scale, unique constraint, database type and so on. The column can be owned by a *Table*, but the table may be inferred if the *Persistent* class does not define a table, or if the class is not persistent. Again a *ColumnDefinition* can be a *Column* owned by the property or a *ColumnRef* that references a *Column*. A property without mapping will have a column with an inferred definition.

The *Embedded* stereotype is applied to association ends or simple properties whose types are not persistent classes. This means that this class is persisted

as a dependent table (if to-many) or embedded in the table (if to-one). Properties of non persistent classes can have the *ColumnMapping* stereotype applied in order to specify how is its preferred way of being persisted, such as *length*, *precision* and so on. These definitions will not have a table.

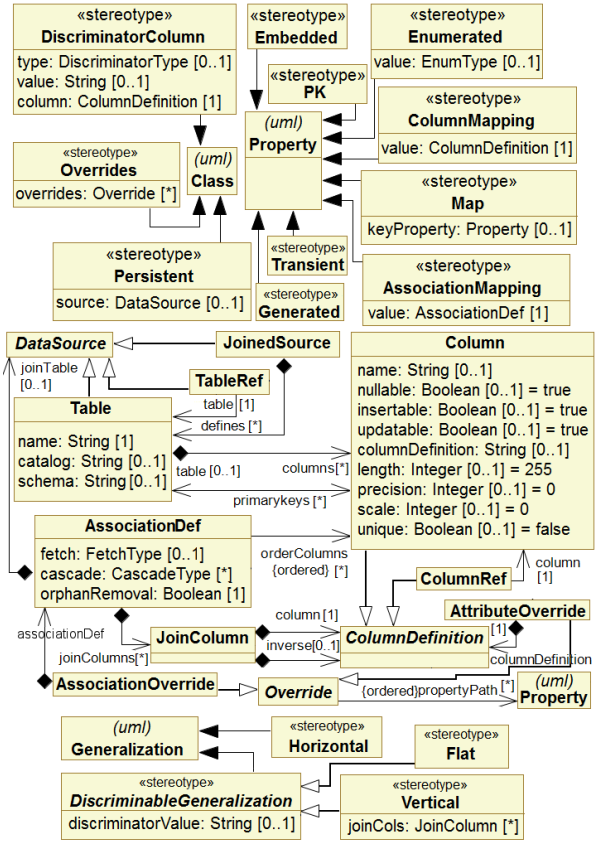


Figure 5: Profile of ENORM

The *AssociationMapping* stereotype allows the definition of mapping details for one association by the application in one of the association ends. The *AssociationDef* class allows the definition of fetch strategies, cascade delete, orphan removal policy, columns used by an order by clause, join columns, and a join table. The *JoinColumn* class defines the FK column in the detail side, and optionally the corresponding PK in the master side (for multiple PK, or ad-hoc joins). The *joinTable* is usually defined on many-to-many situations to specify the table(s) that implement the relationship.

The *PK* stereotype marks a property as part of the PK of some persistent class. It can be applied on association ends, meaning that the FK(s) columns are also part of the PK. *PK* can be combinated with *ColumnMapping*, *AssociationMapping* and so on. *Generated* marks a column with generated values.

Horizontal, *Flat*, and *Vertical* stereotypes can be applied to a generalization to specify which pattern will be used to emulate inheritance on the database. With *Flat*, all columns necessary to represent the inheritance tree are stored in the same table. Usually, the instance type is determined by a discriminator column, that can be defined by applying the *DiscriminatorColumn* stereotype at the general class, and filling the property *discriminatorValue* for each generalization with the *Flat* application.

The *Vertical* stereotype stores each class along its properties in a distinct table, that is by default joined by a common PK. It is possible to specify what columns perform the join by the *joinCols* property. It is also possible to define a discriminator. Finally the *Horizontal* stereotype stores each concrete class independently, and the origin table determines the type.

A class may specify an inheritance pattern even when it inherits from a non persistent class. In this situation (and only this), the properties and associations of the general class will be persisted along the persistent specializations. The *Overrides* stereotype allows a class to override such properties (*Attribute Override*) and associations (*Association Override*), defining the *columns*, *join columns*, *join tables*, among other details.

A class may also override properties and associations of embedded/dependent classes. The tricky part here is that one class can embed a class that embed another class. The property path of embedded overrides is stored by the ordered association *propertyPath*. In the example of Figure 2 the path "*balance.amount*" refers to the sequence {*Account.balance, Quantity.amount*}. This allows the override to differentiate when the class has more than one relationship to the same class.

The *Enumerated* stereotype allows the definition of how enumerations are mapped (string or ordinal values). The *Transient* stereotype marks a property or association end to be ignored on persistence mapping.

5. ENORM AND ORM FRAMEWORKS

The way *JPA*, *SA*, and *AR* implements each ORM pattern is distinct. *AR* separates database from class definitions on migration files, where each table is specified with its columns and references. *JPA*, in the other hand, infers much of the database structure from annotations placed before each class (or XML), but does not have a central place where the database is defined. In the middle ground, *SA* allows the definition of tables, classes, and its mappings separately (classical) or together (declarative), but the table definitions are clearly separated at *runtime*.

JPA advanced a lot in the field of embedded and dependent mapping, providing several resources to automate complex collections of elements and embedded classes. *SA* has a simple mechanism called *composites* that deals with embedded objects, but not

with dependent objects. *AR* also has a similar mechanism named *composed_of*. *JPA* allows the partial mapping of plain classes (such as *Quantity*) and a later override by the container classes. *SA* and *AR* does not have this resource.

SA and *JPA* supports all three inheritance patterns. AR only supports the *Flat* strategy, and other strategies can at best be emulated with a simple relationship.

SA allows the definition of queries based on polymorphism for inheritance or class mapping. *JPA* relies on one-to-one relationship between tables for mappings and multi-table inheritance. *AR* does not allows a class mapped to multiple tables. The implementation of *Account,* with *AR, JPA*, and *SA*, is available at the web site of our modeling tool [9].

6. LIMITATIONS AND SPECIAL CASES

This section enumerates some known limitations and special use cases of ENORM.

- **Flexible data sources.** Currently, the profile only supports the mapping of one class to many tables if each table has a one-to-one relationship to the first table. This is an easy way to specify the data source without caring about checking how a complex mapping would be persisted. A more flexible rule for data sources would be equivalent to a side effect free *updatable view* [7].
- **Qualified associations.** Qualified associations can have more than one qualifier properties. This kind of construct would need keys with *tuples* of objects, what can be quite complicated to implement using ORM tools. Qualified properties with upper cardinality over one is a special case, representing a map of collection elements, where each key can have more than one associated value.
- **Multiple Inheritance, multiple types.** The profile does not include resources to deal with the persistent specialization of more than one persistent class, and the resulting mapping would be unknown. However, a class can specialize any number of other classes as long as it only inherits persistent information from one tree branch. Single relation with multiple type attributes [8] was not included in ENORM.
- **Association class and "n-ary".** The profile does not have any specific mapping for the *Association Class* element of UML, it is as any other class. ENORM does not yet support persistent associations with more than two classes. These associations must be separated on binary associations.
- **Generics and Template parameters.** Mechanisms such as generics can be specified using template parameters on UML [20], and they are useful for strong typed languages such as Java and C#. We did not identify any additional extension necessary to the use of template parameters.

7. EMPIRICAL EVALUATION

The goal of the empirical evaluation is to check if ENORM had a greater rate of success in the activity of changing models, regardless of any impact related to implementation. Changing models was our choice because it is more common than creating new models, and captures both comprehension and application of the notation.

Controlled experiments comparing the use of ENORM and separated UML/Relational models were performed to test our hypothesis. In this paper we summarizes the results of an experiment performed in 2012 with 69 students[1].

The tasks were designed as modeling activities, showing models based upon Analysis Patterns and asking the participants to apply a set of modifications, creating an output model. These models and instructions were extracted from the Analysis Pattern literature [10], in order to reduce the artificiality of the tasks, and augmented with ORM details. Each task was as objective as possible, avoiding misinterpretations. One of the tasks was similar to the evolution of the *accounting* models (Figures 1 and 2), the other tasks related to *accountability* and *planning domains*.

The subjects were senior undergraduate students and graduate students, selected among those already approved on the basic database, object oriented development, and software engineering courses. Each participant received a training in the format of a tutorial with videos, and a small scale task just like the experiment itself.

The experiment had a *within-subjects* design, in which each treatment was applied to each subject, and the starting order was randomized (*counterbalancing*) so that the same number of subjects started with each treatment [13, 15]. The treatment (*method*) is the main independent variable assuming ***A*** (not using ENORM) or ***B*** (using only ENORM).

The dependent variable is the number or missed goals (*misses*) based on expected model. The time to execute each task was fixed due to external constraints, and was not evaluated in this experiment.

Other factors were controlled as follows: both hardware and software used on the experiment was the same to all participants; a specially developed modeling tool was employed to guarantee a similar environment, and detect the number of missed goals.

The Analysis of Variance (ANOVA) was employed to compare the treatments, making it possible to verify the residual effect in the *sequence* of activities. In other words, it checks if there are significant difference in the sequences *AB* or *BA*, an indication of learning effect.

[1] Full technical report available at [21].

Table 2 presents the least square means of *misses,* and *p-value* results analyzing *method* and *sequence (seq.)* on each of the four tasks (*T*). The other variables are the number of participants (*P*), and time in minutes available to perform each task (*Tim*).

Table 2. ANOVA results, per task.

T	P	Tim	P-value		Misses	
			Method	Seq.	A	B
1	69	10	<0.001	0.56	16.6	11.6
2	69	20	<0.001	0.42	11.2	6.5
3	35	20	0.006	0.29	13.1	10.7
4	35	23	<0.001	0.25	7.5	2.4

Assuming results as statistically relevant at $\alpha = 0.05$, there is a significant difference ($p<0.05$) between *methods **A*** and ***B***, with *method **B*** presenting a lower mean of misses at all tasks. The sample evidence does not confirm the presence of residual effect, given the absence of statistical significance for the effect of *sequence* ($p>0.05$ at all tasks).

8. CONCLUSION

Despite the growing popularization of ORM patterns by the adoption of persistence frameworks, the mappings between objects and database are dispersed in the code. Distinct frameworks employ distinct ways of presenting these mappings, despite following the same patterns.

This paper proposes ENORM, a new notation implemented as an extension of UML class models that allows the design of database based systems, providing the essential patterns of ORM in a platform independent way. ENORM unifies classes and mappings focused on the structural aspects of persistence, with the necessary detail for MDA tools. Our controlled experiment indicated that ENORM had a lower mean of missed goals when improving models, in comparison to separated models.

9. REFERENCES

[1] A UML Profile for Data Modeling: 2003. *http://www.agiledata.org/essays/umlDataModelingProfile.html*. Accessed: 2013-10-01.

[2] Active Record - Object-relation mapping put on rails: 2012. *http://ar.rubyonrails.org/*. Accessed: 2013-10-01.

[3] Atzeni, P. et al. 2013. The relational model is dead, SQL is dead, and I don't feel so good myself. *SIGMOD Rec.* 42, 1 (Jul. 2013), 64–68.

[4] Beydeda, S. et al. 2005. *Model-Driven Software Development*. Springer.

[5] Blakeley, J.A. et al. 2006. The ADO.NET entity framework: making the conceptual level real. *SIGMOD Rec.* 35, 4 (Dec. 2006), 32–39.

[6] Brown, K. and Whitenack, B.G. 1996. Crossing Chasms: a pattern language for object-RDBMS integration: the static patterns. *Pattern languages of program design 2*. Addison-Wesley Longman Publishing Co., Inc. 227–238.

[7] Dayal, U. and Bernstein, P.A. 1982. On the correct translation of update operations on relational views. *ACM Trans. Database Syst.* 7, 3 (Sep. 1982), 381–416.

[8] Elmasri, R. and Navathe, S.B. 2003. *Fundamentals of Database Systems*. Addison Wesley.

[9] Essential ORM Modeler: 2013. *http://sourceforge.net/projects/eorm/*. Accessed: 2013-12-05.

[10] Fowler, M. 1996. *Analysis Patterns: Reusable Object Models*. Addison-Wesley Professional.

[11] Fowler, M. 2002. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc.

[12] JSR-000317 Java Persistence 2.0 - Final Release: 2009. *http://jcp.org/aboutJava/communityprocess/final/jsr317/index.html*. Accessed: 2013-10-01.

[13] Juristo, N. and Moreno, A.M. 2001. *Basics of Software Engineering Experimentation*. Springer.

[14] Keller, W. 1997. Mapping Objects to Tables - A Pattern Language. *Proceedings of the 1997 European Pattern Languages of Programming Conference* (Irrsee, Germany, 1997).

[15] Ko, A.J. et al. 2013. A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering*. (Sep. 2013), 1–32.

[16] Mellor, S.J. et al. 2004. *MDA Distilled: Principles of Model-Driven Architecture*. Addison-Wesley.

[17] OMG 2001. OMG's Model Driven Architecture.

[18] OMG 2005. Request For Proposal Information Management Metamodel (IMM).

[19] SQLAlchemy - The Database Toolkit for Python: 2012. *http://www.sqlalchemy.org/*. Accessed: 2013-10-01.

[20] Torres, A. et al. 2011. A synergistic model-driven approach for persistence modeling with UML. *Journal of Systems and Software*. 84, 6 (Jun. 2011), 942–957.

[21] Torres, A. et al. 2013. Technical Report - Comparing ENORM and separated modeling using Relational and UML class models: a within-subjects experimental study. *http://www.inf.ufrgs.br/~atorres/sigmod2013/*.

[22] UML 2.4.1 Superstructure: 2011. *http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/*. Accessed: 2013-10-01.

Locating Valid SLCAs for XML Keyword Search with NOT Semantics

Rung-Ren Lin[1], Ya-Hui Chang[2]*, and Kun-Mao Chao[1]
[1]Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan.
{r91054, kmchao}@csie.ntu.edu.tw
[2]Department of Computer Science and Engineering
National Taiwan Ocean University, Keelung, Taiwan.
yahui@ntou.edu.tw

ABSTRACT

Keyword search provides an easy way for users to pose queries against XML documents, and it is important to support queries with arbitrary combinations of AND, OR, and NOT operators. The previous RELMN algorithm processed such kind of queries by extending the original SLCA definition in a straightforward way, but it did not work correctly in some cases. In this paper, we propose the concept of *valid SLCAs* as query results. Basically, nodes in an XML document are classified according to their usages, which is further used to define the *scope* affected by a negative keyword. Only *valid* nodes, which are not affected by any negative keyword, are qualified to identify *valid SLCAs*. The experimental results show that the proposed algorithm achieves higher precision and recall, and is more efficient than the previous work.

1. INTRODUCTION

Keyword search provides an easy way for users to obtain desired information from XML documents by just giving some keywords they are interested in, but irrelevant data may be returned due to lacking exact query semantics. Many researchers advocate the idea that each returning result should be based on an LCA (Lowest Common Ancestor), which contains every keyword under its subtree at least once [1][3][4][5][6][7]. Among all the LCA-based techniques, the smallest-LCA (SLCA) [6] concept is particularly popular since it can locate nodes semantically closer to the query keywords. Specifically, a node n is said to be an SLCA node if: (i) n is an LCA node, and (ii) none of n's children are LCA nodes. Consider the sample XML tree depicted in Figure 1, which represents the information about course offerings at a particular school, and each node is identified by a Dewey encoding.

*To whom all correspondence should be sent.

Table 1: Sample keyword queries.

Q_1	Subject $\wedge$ Friday
Q_2	Subject $\wedge$ Friday $\wedge$!R101
Q_3	2010 $\wedge$ Subject $\wedge$!R101
Q_4	Red Wood $\wedge$ Subject $\wedge$ Friday $\wedge$!R103
Q_5	Subject $\wedge$ Friday $\wedge$!R102 $\wedge$!2010
Q_6	Subject $\wedge$ Friday $\wedge$ (R101 $\vee$ R103)

Query Q_1 in Table 1 is used to retrieve those subjects whose classes are held on Friday. Since only nodes 1.2.2 and 1.3.3 contain $Subject$ and $Friday$ under their subtrees and none of their children do, these two nodes are the SLCAs for Q_1.

The limitation of the works above is that they only concern the keywords with the AND operator. It is therefore an important research issue to process queries with arbitrary combinations of AND, OR, and NOT operators, and the challenge mainly lies in how to identify results satisfying the NOT semantics. A straightforward idea is that the returning SLCA should contain every "positive" keyword but no "negative" keyword (keywords with NOT operators). Consider the sample query Q_2 in Table 1, where the exclamation mark "!" is used to represent the NOT operator. The semantics of query Q_2 is similar to that of query Q_1, except that Q_2 should not output the courses held in room 101. Since the course rooted at node 1.2.2 contains the unwanted negative keyword "R101", only the subtree rooted at 1.3.3 should be returned. Although intuitive, such approach is sometimes too restrictive. Consider query Q_3, asking for subjects whose classes are offered in 2010 but not held in room 101. We cannot find an SLCA satisfying the above definition, but actually node 1.2.3.1, whose class is held in room 103, is an answer reasonable for a human user, which we call *human answers* in short. To avoid the occurrences of such false negatives, when

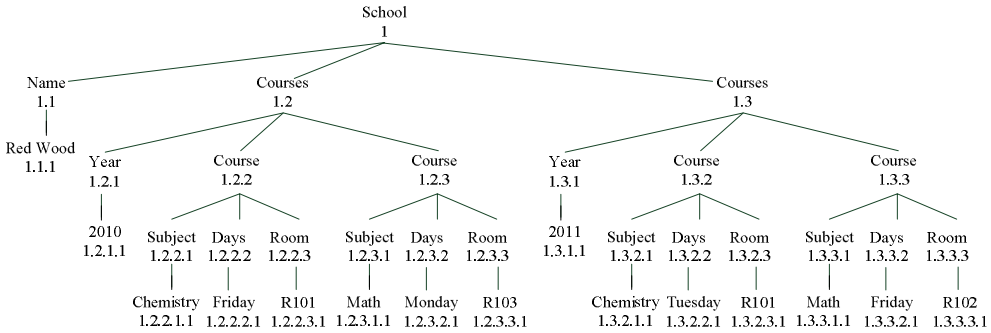


Figure 1: A sample XML tree.

no satisfied nodes are found, our previously proposed RELMN algorithm [2] will identify SLCAs only based on positive keywords, and then *prune out* nodes according to negative keywords.

Although the RELMN algorithm basically performs well, the process of result identification is complex, and it still might cause false negatives or false positives in some cases, as will be discussed in detail in Section 2. Therefore, we propose a new approach to improve the effectiveness and efficiency of processing XML keyword search with NOT semantics. The main idea is to classify all nodes into several groups based on their usage. This classification is used to define the *scope* that a negative keyword can affect. We then propose to return *valid SLCA* as the query result, where a valid SLCA contains every positive keyword under its subtree, and each of which is not *affected* by any negative keyword. Consider query Q_3 again. Node 1.2 is the only SLCA based on positive keywords. Since its descendant nodes 1.2.1.1 and 1.2.3.1 match the positive keywords, and are not within the scope affected by the negative keyword $R101$, node 1.2 is qualified as a *valid SLCA*, which contains human answers and should be returned. In summary, the contributions of this paper are as follows:

- Based on a simple observation of XML documents, we propose a set of definitions which identify nodes which should not be output due to negative keywords.
- We design an efficient algorithm called ValidSLCA to process queries with arbitrary combinations of AND, OR, and NOT operators.
- We perform an empirical evaluation by measuring the precision, recall, and processing time of two approaches. The experimental results show that the new ValidSLCA algorithm is more efficient and gives better query results than the previous RELMN algorithm [2].

The rest of this paper is organized as follows. Section 2 briefly describes the RELMN algorithm. The basic observation and definitions underlying the new approach are presented in Section 3, and the corresponding algorithm is discussed in Section 4. At last, experimental studies and conclusions are given in Section 5 and Section 6, respectively.

2. THE RELMN APPROACH

We briefly describe the RELMN algorithm in this section, and direct the readers to [2] for detailed discussion.

The RELMN algorithm basically consists of two steps. First, it searches the *satisfied* SLCAs of the input query. Specifically, a node n *satisfies* the query Q if n contains every "positive" keyword under its subtree but does not contain any "negative" keyword. Further, n is a *satisfied SLCA* if n satisfies Q but none of n's children do. As discussed in Section 1, node 1.3.3 is a *satisfied* SLCA for query Q_2. When no satisfied SLCAs are found, the RELMN algorithm will locate SLCAs purely based on positive keywords. Second, for every subtree rooted at the identified SLCAs, it then prunes the unwanted part based on the following rule: node n_1 is able to prune its sibling node n_2 if n_1 contains more positive keywords or contains less negative keywords under its subtree than n_2 does. Consider query Q_3 again. RELMN will first identify node 1.2 based on the positive keywords "2010" and "Subject", and then exclude the subtree rooted at node 1.2.2, since it has the negative keyword $R101$ and is pruned by node 1.2.3.

The time complexity of the above two steps are respectively $O(d|M| \cdot w)$ and $O(\Sigma_{i=1}^{z} b_i)$, where d is the depth of the XML tree, $|M|$ is the sum of the keyword frequencies, w is the number of clauses of query Q in conjunctive normal form, z is the number of nodes in the match tree for Q[1], and b_i is

[1]A match (in RELMN) is a node containing a positive/negative keyword. The match tree for a given query is a subtree of the XML tree, which consists of the nodes along the path from each match up to the root.

the number of siblings of the i^{th} node.

We now discuss the cases where the RELMN algorithm might cause false negatives or false positives. Consider query Q_4 in Table 1, asking for the course information at the school "Red Wood", and only the subjects whose classes are held on Friday but not in room 103 should be returned. Note that nodes 1.2.2 and 1.3.3 represent human answers. Nevertheless, RELMN will only return 1.3.3. The reason is that RELMN will first identify node 1 as the only SLCA, and then prune node 1.2 by node 1.3 because it contains the negative keyword $R103$. Hence, node 1.2.2 is excluded too. Take query Q_5 as another example, asking for subjects whose classes are held on Friday, but not in room R102, and not in year 2010. Observe that the human answer is empty. However, RELMN will return both nodes 1.2.2 and 1.3.3, since they are both identified as SLCAs based on positive keywords and cannot prune each other.

3. BASIC OBSERVATIONS AND DEFINITIONS

In this section, we explain the observation on XML documents which is underlying the new approach, and deliver the basic definitions. The main idea is to identify the *scope* which is affected by a negative keyword according to node types. Since the entity-relationship (ER) model has been widely used for data modeling, we borrow some terms from that model to classify the nodes in an XML tree based on their usage as follows:

1. A node is a *text* node if it is a value.
2. A node is an *attribute* node if it has only one child, which is a text node.
3. A node denotes an *entity* if its tag name is identical to its siblings, such as a *-node in the XML DTD (Document Type Definition).
4. A node is a *dummy* node if it is none of the three nodes mentioned above.

For example, nodes 1.1.1 (*Red Wood*), 1.2.3.2.1 (*Monday*), and 1.3.1.1 (2011) are text nodes. Nodes 1.2.1 (*Year*) and 1.2.2.3 (*Room*) are attribute nodes. Nodes 1.2 (*Courses*) and 1.2.3 (*Course*) are entity nodes. In addition, the *closest entity* of node n refers to the lowest entity node that are the ancestor of n. For instance, node 1.3 (*Courses*) is the closest entity of node 1.3.1.1 (2011), and node 1.2.3 (*Course*) is the closest entity of node 1.2.3.2.1 (*Monday*).

We observe that a text node is generally used as the attribute value of its closest entity. For example, node 1.1.1 (*Red Wood*) is used to describe the name of the school (node 1), node 1.3.1.1 (2011) indicates the courses (node 1.3) belonging to year 2010, and node 1.2.3.2.1 (*Monday*) represents that the course (node 1.2.3) is held on Monday. Therefore, if a negative keyword matches a text node n, it is reasonable to assume that the subtree rooted at n's closest entity is unwanted. For instance, if the negative keyword is $!R101$, we assume that the subtrees rooted at nodes 1.2.2 and 1.3.2 should not be returned. In other words, we propose that the query result is composed of the subtrees which represent positive keywords and are not "affected" by the negative keywords. For ease of discussion, we assume that the query is in DNF in the rest of this section. Hence, the clauses in a query are connected with the OR operators, and keywords in the same clause are connected with the AND operators. The formal definitions are then given as follows:

Definition 1. If node n is a text node and corresponds to a given negative keyword, the closest entity of n is termed a *negator*.

Definition 2. A node is a *match* of keyword k if its tag name or content contains k. Moreover, a match is *valid* if it has no ancestors that are negators formed within the same clause.

Consider the positive keyword *Friday* in query Q_2. Among the two matches 1.2.2.2.1 and 1.3.3.2.1, the former one will become *invalid* due to the negator (node 1.2.2) caused by the negative keyword $!R101$.

Definition 3. For a given clause c, a node n is said to be an SLCA if n contains every positive keyword of c at least once under its subtree and none of n's children do. Furthermore, n is a candidate valid SLCA (VSLCA) if n contains at least one valid match for every positive keyword of c.

Continuing the running example, both nodes 1.2.2 and 1.3.3 are SLCAs in our framework, but only node 1.3.3 is a candidate VSLCA since node 1.2.2 has no valid matches for both *Friday* and *Subject* under its subtree.

Definition 4. Given a keyword query in DNF with only one clause, all the candidate VSLCAs n are *valid SLCAs*. If there is more than one clause, n is a *valid* SLCA only if n has no descendants that are also candidate VSLCAs formed by other clauses.

The proposed new approach will return all identified valid SLCAs as the query result. Consider query Q_6, which searches the subjects whose classes are held on Friday and in room 101 or 103. The

DNF of Q_6 has two clauses: $(Subject \wedge Friday \wedge R101)$ and $(Subject \wedge Friday \wedge R103)$, and the candidate VSLCAs of these two clauses are node 1.2.2 and node 1.2, respectively. The query result will be node 1.2.2, since it is the descendant of node 1.2.

Note that in Definition 1, we only consider text nodes for negative keywords. The reason is that a negative non-text keyword is mainly used to describe unnecessary information, and does not affect the validity of an SLCA. For instance, suppose the negative non-text matches have no ancestor-descendant relationships with other positive keywords, such as in the query (Subject ∧ R101 ∧ !Days). We can see that the SLCA node, 1.2.2 (Course), is a human answer and should not be affected by $!Days$. Even if the negative non-text matches have ancestor-descendant relationships with other positive keywords, such as in the query (Subject ∧ R101 ∧ !Room), node 1.2.2 (Course) should still be returned.

4. THE VALIDSLCA ALGORITHM

The proposed ValidSLCA algorithm is listed in Figure 2. Briefly, for the input query Q, we will convert it into DNF Q' (line 1). To get the candidate VSLCAs of each clause, we first identify the SLCAs based only on positive keywords by the approach given in [6] (line 5). We then collect the negators in line 6. The negators are used to determine the valid matches by procedure $ValidateMatches$. Namely, some of the matches will be removed if they are the descendants of the negators. Next, we identify candidate VSLCAs by ensuring that there exists at least one valid match for each positive keyword (lines 9-12). At last, those candidate VSLCAs that are ancestors of others are removed (line 13).

Observe that the procedures ValidateMatches and CombineSLCA are variants of merge sort algorithms and are therefore linear to the input. Besides, two B-tree indices are maintained for efficient processing. In the first index, the data associated with each keyword k is a sorted list of Dewey numbers of all the nodes which contain keyword k. If a node is a text-node, we additionally record its closest entity. Hence, we can efficiently retrieve the match array of the keyword (lines 3-4) and negators (line 6). The second index is used to output meaningful answers for users (line 14), which retrieves the tag name by a given Dewey number. Both indices are created and accessed based on the Oracle Berkeley DB[2].

We next give the time complexity of ValidSLCA. The details are omitted due to space constraints. Let d be the maximum depth of the XML tree and

[2]http://www.oracle.com/database/

Input: A keyword query Q of arbitrary combination with AND, OR, and NOT operators.
Output: All of the valid SLCAs of Q.

ValidSLCA(Q)
1 convert Q into DNF Q'
2 **for** each clause c_i of Q' **do**
3 **for** each positive keyword k_j of c_i **do**
4 $M_j \leftarrow$ all the matches of k_j
5 $S_i \leftarrow$ compute SLCAs by { M_1, M_2, M_3…}
6 $E \leftarrow$ collect the negators by the negative keywords of c_i
7 **for** each M_j **do**
8 $M'_j \leftarrow$ *ValidateMatches*(M_j, E)
9 **for** each $n \in S_i$ **do**
10 **for** each M'_j **do**
11 if n has no descendants belongs to M'_j **then**
12 remove n from S_i
13 $S \leftarrow$ *CombineSLCA*(S_1, S_2, S_3…)
14 Output S /* all the valid SLCAs of query Q */

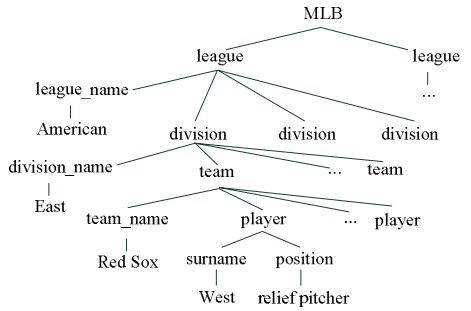
ValidateMatches(M, E)
1 $i \leftarrow 1, j \leftarrow 1$
2 **while** $i \leq |M|$ **and** $j \leq |E|$ **do**
3 **if** $M[i]$ precedes $E[j]$ **then**
4 $i \leftarrow i + 1$ /* $M[i]$ is valid */
5 **else if** $E[j]$ is the ancestor of $M[i]$ **then**
6 remove $M[i]$ from M /* $M[i]$ is invalid */
7 $i \leftarrow i + 1$
8 **else**
9 $j \leftarrow j + 1$
10 return M

CombineSLCA(S_1, S_2, S_3…)
1 $S \leftarrow$ merge (S_1, S_2, S_3…) by their Dewey Numbers
2 $i \leftarrow 1, j \leftarrow 2$
3 **while** $j \leq |S|$ **do**
4 **if** $S[i]$ is the ancestor of $S[j]$ **then**
5 remove $S[j]$ from S
6 **else**
7 $i \leftarrow j$
8 $j \leftarrow j + 1$
9 return S

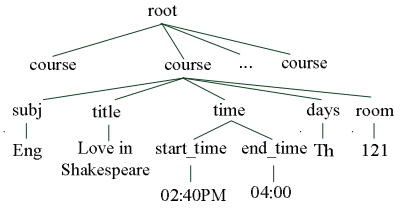
Figure 2: The *ValidSLCA* algorithm.

Q' be the DNF of the original input query Q. Suppose Q' has k clauses, and the i^{th} clause has p_i positive keywords and n_i negative keywords. That is, the matches of positive keywords of c_i are $M_1, M_2, ..., M_{p_i}$, and the matches of negative keywords of c_i are $N_1, N_2, ..., N_{n_i}$. Let $|M_i^{sum}| = \Sigma_{j=1}^{p_i} M_j$ and $|N_i^{sum}| = \Sigma_{j=1}^{n_i} N_j$ be the sum of the size of positive and negative matches. The overall time complexity of algorithm ValidSLCA is $O(\Sigma_{i=1}^{k} \Sigma_{j=1}^{p_i} (|M_j| + |N_i^{sum}|) \cdot d)$.

This section is concluded by showing how Algorithm ValidSLCA can correctly process queries Q_4 and Q_5. Recall that nodes 1.2.2 and 1.3.3 are two



(a) Baseball



(b) Reed College

Figure 3: Portions of the XML trees.

Table 2: The information for the data sets.

Data Set	Doc. Size	Nodes	Max/Avg Depth
Baseball	1.01 MB	51,599	7/6.5
Reed	277 KB	18,855	5/3.7
Mondial	1.7 MB	124,736	7/4.6
DBLP	127 MB	7,146,530	7/3.5

courses that should be returned by Q_4, but RELMN only identifies one, as discussed in Section 2. In ValidSLCA, only node 1.2.3.3.1 (R103) matches the negative keyword and the corresponding negator is node 1.2.3. This negator will not affect the positive matches under the subtrees rooted at 1.2.2 and 1.3.3, so we will identify node 1 as the valid SLCA and return both courses. Consider another query Q_5, where the answer should be empty. The two negative keywords (R102 and 2010) yield two negators (nodes 1.2 and 1.3.3) in total. Besides, the positive keyword $Friday$ has two matches 1.2.2.2.1 and 1.3.3.2.1. Since both of them are descendants of negators, our approach will correctly return empty.

5. EXPERIMENTAL EVALUATION

In this section, we compare the performance of RELMN and ValidSLCA algorithms. Both of them were implemented in C++, and the experiments were performed on a 1.67GHz dual-core CPU with 1.5GB RAM. Note that in order to be fair with the RELMN approach, after identifying valid SLCAs, we further adopt the pruning rules described in [5] to remove subtrees containing less positive keywords than their siblings. The time complexity of ValidSLCA therefore becomes $O(\Sigma_{i=1}^{k}\Sigma_{j=1}^{p_i}(|M_j|+|N_i^{sum}|)\cdot d) + O(\Sigma_{i=1}^{k}|M_i^{sum}|\cdot d\cdot 2^{p_i})$.

Table 3: Test queries.

No.	query
QB_1	(Indians ∨ Tigers) ∧ Starting Pitcher ∧ SURNAME
QB_2	SURNAME ∧ !Starting Pitcher ∧ !Relief Pitcher
QB_3	American ∧ TEAM_NAME ∧ !Central
QB_4	!American ∧ TEAM_NAME
QB_5	Yankees ∧ SURNAME ∧ !Starting Pitcher ∧ !Relief Pitcher
QB_6	East ∧ !Yankees ∧ Second Base ∧ HOME_RUNS
QB_7	American ∧ (East ∨ West) ∧ TEAM_NAME
QB_8	Red Sox ∧ HITS ∧ !Outfield
QR_1	subj ∧ 04:10PM ∧ !T ∧ !W
QR_2	subj ∧ !T
QR_3	subj ∧ room ∧ 301 ∧ !M
QR_4	instructor ∧ CHEM ∧ M ∧ 01:10PM
QR_5	instructor ∧ CHEM ∧ !M ∧ 01:10PM
QR_6	room ∧ Love in Shakespeare ∧ !02:40PM
QR_7	room ∧ Love in Shakespeare ∧ !F
QR_8	title ∧ (M ∨ W) ∧ !01:10PM

We used four data sets, DBLP.xml[2], Mondial.xml[2], Reed.xml[3], and Baseball.xml[4], to perform the experiments. Some statistics about these data sets, including their sizes, are listed in Table 2 for comparison. Besides, the metrics to compare the two algorithms are precision, recall, and processing time. The first two are calculated based on *human answers*, which are obtained from three human experts with differences resolved by voting.

We first use the Baseball and Reed data sets to analyze the precision and recall of RELMN and ValidSLCA. Figure 3 shows the portions of these two data sets to assist in analyzing the outcomes of experiments. The test queries of these two data sets are listed in Table 3, and the precision and recall on the Baseball data set is displayed in Figure 4. We can see that ValidSLCA gets better precision than RELMN in QB_2. In RELMN algorithm, all the nodes with "surname" tag names satisfy QB_2 and will be returned. Since some of the players are starting pitchers or relief pitchers, it gets imperfect precision and 100% recall. However, those players who are starting pitchers or relief pitchers are considered as invalid matches in ValidSLCA. Hence, ValidSLCA gets 100% precision and recall in QB_2. RELMN also gets imperfect precision in QB_4 due to the same reason. On the other hand, the precision and recall on the Reed data set is displayed in Figure 5. The semantics of QR_6 is to search the room of course "Love in Shakespeare" which does not start at 02:40PM. However, there only exists one course titled "Love in Shakespeare", and it coincidentally starts at 02:40PM. That is, the answer should be empty. Refer to Figure 3 (b). The closest entity of !02:40PM is the node with tag name *course*. Hence, ValidSLCA returns empty while RELMN still returns course "Love in Shakespeare". The similar situation happens in QR_7, which makes RELMN get poor precision and recall, too.

The processing time is measured with four data sets mentioned above. Since the query result may be different between RELMN and ValidSLCA, we only consider those queries which yield the same

[3]http://www.cs.washington.edu/research/xmldatasets/
[4]http://www.cafeconleche.org/books/biblegold/examples/

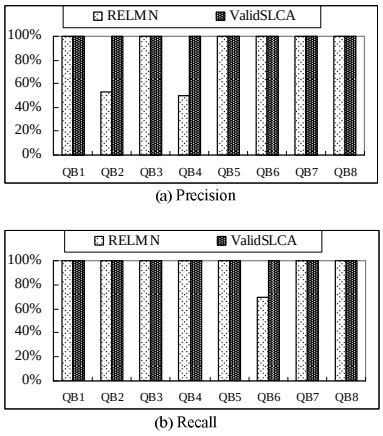


Figure 4: The Baseball data set.

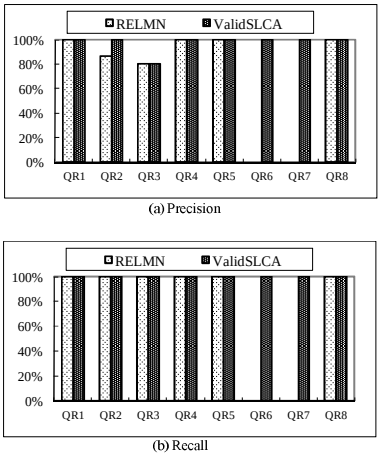


Figure 5: The Reed data set.

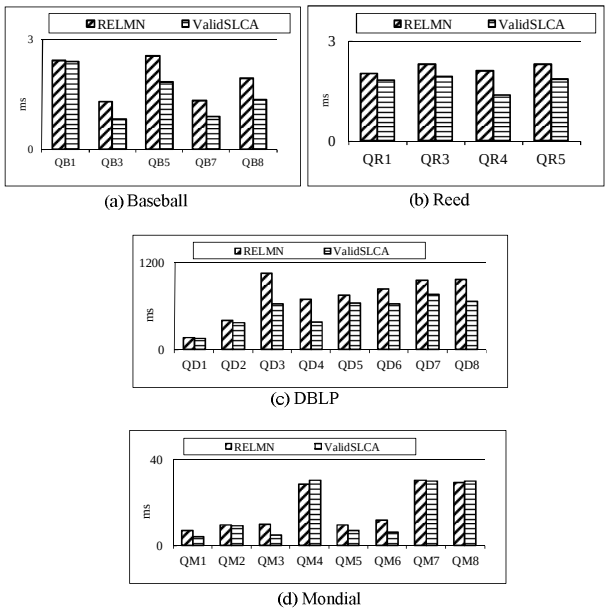


Figure 6: The processing time.

query results by the two algorithms. Due to space limitation, we do not explicitly list the test queries. As shown in Figure 6, we can see that our new approach is more efficient than the previous work in most cases. One reason is that the pruning rules defined in RELMN are very complex to process. Besides, as shown in the time complexity of the two algorithms, RELMN is affected by the branch factor of the XML trees (b_i), which may be very large. On the other hand, ValidSLCA is affected by the number of the query keywords (p_i), which is comparably small. Hence, the RELMN algorithm often takes more time than ValidSLCA does.

6. CONCLUSIONS

In this paper, we propose an algorithm to process keyword search with NOT semantics. The idea is to classify matches as valid or invalid based on negative keywords, and only valid nodes are qualified to identify valid SLCAs as query outputs. We empirically evaluate the precision, recall, and processing time by comparing the previous work and our new approach. The experiments show that our new approach gives more reasonable query results and is even more efficient. As part of our future work, we are interested in designing a novel ranking scheme to order the query results so that users can focus on most desirable ones.

7. REFERENCES

[1] G. Li, J. Feng, J. Wang, and L. Zhou. Effective Keyword Search for Valuable LCAs over XML Documents. In *CIKM*, 2007.

[2] R.-R. Lin, Y.-H. Chang, and K.-M. Chao. Identifying Relevant Matches with NOT Semantics over XML Documents. In *DASFAA*, 2011.

[3] R.-R Lin, Y.-H. Chang, and K.-M. Chao. Improving the Performance of Identifying Contributors for XML Keyword Search. In *SIGMOD Record*, 40(1), pp. 5-10, 2011.

[4] Y.-H. Chang. Optimizing XML Twig Queries with Full-Text Predicates. In *SIGMOD Record*, 41(1), pp. 5-10, 2012.

[5] Z. Liu and Y. Chen. Reasoning and Identifying Relevant Matches for XML Keyword Search. In *PVLDB*, 2008.

[6] Y. Xu and Y. Papakonstantinou. Efficient Keyword Search for Smallest LCAs in XML Databases. In *SIGMOD*, 2005.

[7] J. Zhou, Z. Bao, W. Wang, T. Ling, Z. Chen, X. Lin, and J. Guo. Fast SLCA and ELCA computation for XML Keyword Queries Based on Set Intersection. In *ICDE*, 2012.

Medusa: A Parallel Graph Processing System on Graphics Processors

Jianlong Zhong
Nanyang Technological University
jzhong2@ntu.edu.sg

Bingsheng He
Nanyang Technological University
bshe@ntu.edu.sg

ABSTRACT

Medusa is a parallel graph processing system on graphics processors (GPUs). The core design of Medusa is to enable developers to leverage the massive parallelism and other hardware features of GPUs by writing sequential C/C++ code for a small set of APIs. This simplifies the implementation of parallel graph processing on the GPU. The runtime system of Medusa automatically executes the user-defined APIs in parallel on the GPU, with a series of optimizations based on the architecture features of GPUs and characteristics of graph applications. In this paper, we present an overview of the Medusa system and a case study of adopting Medusa to a research project on social network simulations. With Medusa, users without GPU programming experience can quickly implement their graph operations on the GPU, which accelerates the discovery and findings of domain-specific applications.

1. INTRODUCTION

GPGPU (General-Purpose Computation on Graphics Processing Units) is gaining increasing popularity in performance acceleration for many applications, including graph processing [8, 10]. A Modern GPU can have over an order of magnitude higher memory bandwidth and computation power than a multi-core CPU. With intensive and application-specific optimizations, GPU-based graph algorithms have shown significant performance improvement over CPU-based implementations. For example, the GPU accelerated breadth first search (BFS) algorithm is up tp 14 times faster than multi-core implementation [10]. However, despite the recent improvements on GPGPU programmability, writing a correct and efficient GPU program is challenging in general and even more difficult for the highly irregular graph applications.

To address the above-mentioned issues and simplify programming graph processing algorithms on the GPU, we propose the Medusa parallel graph processing programming framework. Like existing programming frameworks such as Hadoop [17] and Mars [9], Medusa provides a small set of APIs for developers to implement their applications by writing sequential (C/C++) code. Different from Hadoop and Mars, which adopts the MapReduce programming model [3], Medusa adopts our novel "Edge-Message-Vertex" (EMV) graph programming model for fine-grained processing on edges, messages and vertices. Medusa embraces an efficient message passing based runtime. It automatically executes user-defined APIs in parallel within one GPU and across multiple GPUs, and hides the complexity of GPU programming from developers. Thus, developers can write the same APIs, which automatically run on one or multiple GPUs. To maximally leverage the power of GPUs, Medusa embraces a series of optimizations based on the architecture features of GPUs and characteristics of graph applications.

As a case study, we adopt Medusa to implement GPU accelerated simulation of information propagation over social networks. Simulation of information propagation is computationally intensive and fortunately highly parallelizable, which makes it viable for GPU acceleration. By the case study, we demonstrate that Medusa both improves the coding productivity and brings significant performance speedups.

This paper gives an overview of Medusa based on work reported in [20, 18, 11, 12, 13]. We first present the design of the main components of Medusa's system architecture and evaluation results. We then present the case study developed based on Medusa. Finally, we conclude this paper and present the future work.

2. RELATED WORK

Parallel graph processing. It has been observed that many common graph algorithms can be formulated using a form of the bulk synchronous parallel (BSP) model [14] (we call it *GBSP*). Under

the GBSP model, local computations on each vertex are performed in parallel iteratively. At the end of each iteration, vertices exchange data by message passing, followed by a global synchronization. More recently, the asynchronous model has been applied to improve the convergence speed of some graph applications [14]. Due to the synchronous nature of the GPU programming model, we only support the synchronous GBSP model on GPU at the moment.

GPGPU. We use NVIDIA CUDA's terminology. The GPU consists of multiple of streaming multiprocessors (SM), inside which there is an array of scalar cores. A CUDA program, which is called a *k*ernel, usually consists of thousands of threads. The massive number of threads of a kernel runs on all the SMs of a GPU in parallel. Each 32 of the threads are grouped into a warp and execute synchronously. Divergence inside a warp introduces severe performance penalty since different paths are executed serially. The GPU requires *coalesced access* to its memory to ensure high utilization of its memory bandwidth. To achieve coalesced access, threads in the same warp must access the same memory segment each time. Another important feature for performance optimization on the GPU is the *shared memory*, which is a small piece of scratch pad memory on the SM. Shared memory has much lower latency compared with the GPU memory. Utilizing shared memory can greatly improve the data access performance of CUDA programs.

3. SYSTEM OVERVIEW

In this section, we give an overview on how users implement their graph processing algorithms based on Medusa, and outline the key modules of Medusa.

3.1 Programming with Medusa

We propose the EMV model as the programming model of Medusa. EMV is similar to the GBSP model and specifically tailored for parallel graph processing on the GPU. To facilitate efficient and fine-grained graph processing on the GPU, EMV decomposes each iterative graph operation in GBSP into three sub-iterations, i.e., processing on edges, messages and vertices. Medusa hides the GPU programming details from users by offering two kinds of APIs, EMV APIs and system-provided APIs, as shown in Tables 1 and 2, respectively. Through those APIs, Medusa enables programmability and efficiency for parallel graph processing on the GPU.

Each EMV API is either for executing user-defined computation on vertices (*VERTEX*), edges (*ELIST*, *EDGE*) or messages (*MESSAGE*, *MLIST*). The vertex and edge APIs can also send messages

```
Device code APIs:
/* EDGE API */
struct SendDistance{
__device__ void operator() (Edge e){
  int head_id = e.get_head_id();
  Vertex head(head_id);
  if(head.get_updated())
  {
    unsigned int msg = v.get_distance() +
                       e.get_length();
    e.sendMsg(msg);
  }
}
/* VERTEX API */
struct UpdateDistance{
__device__ void operator() (Vertex v){
  unsigned int min_msg = v.combined_msg();
  if(min_msg < v.get_distance())
  {
    v.set_distance(min_msg);
    v.set_updated(true);
    Medusa_Continue();
  }
  else
    v.set_updated(false);
}
```

```
Iteration definition:
void SSSP() {
  /* Initiate message buffer to UINT_MAX */
  InitMessageBuffer(UINT_MAX);
  /* Invoke the EDGE API */
  EMV<EDGE>::Run(SendDistance);
  /* Invoke the message combiner */
  Combiner();
  /* Invoke the VERTEX API */
  EMV<VERTEX>::Run(UpdateDistance);
}
Configurations and API execution:
int main(int argc, char **argv) {
  /* Load the input graph. */
  Graph my_graph;
  conf.combinerOpType = MEDUSA_MIN;
  conf.combinerDataType = MEDUSA_UINT;
  conf.gpuCount = 1;
  conf.continueIteration = false;
  /*Setup device data structure.*/
  Init_Device_DS(my_graph);
  Medusa::Run(SSSP);
  /* Retrieve results to my_graph. */
  Dump_Result(my_graph);
  ......
  return 0;
}
```

Figure 1: User-defined functions in SSSP implemented with Medusa.

to neighboring vertices. The idea of providing these APIs is mainly for efficiency. It decouples the single vertex API of previous GBSP-based systems into separate APIs which target individual vertices, edges or messages. Each GPU thread executes one instance of the user-defined EMV API. The fine-grained data parallelism exposed by the EMV model can better exploit the massive parallelism of the GPU. In addition, a *Combiner* API is provided to aggregate results of *EDGE* and *MESSAGE* using an associative operator. This enhances the execution performance since segmented-scan has very efficient and load balanced implementation on the GPU [15].

A small set of system provided APIs is designed to hide the GPU-specific programming details. Particularly, Medusa provides $EMV < type >:: Run()$ to invoke the device code APIs, which automatically sets up the thread block configurations and calls the corresponding user-defined functions. Medusa allows developers to define an *iteration* which executes a sequence of $EMV < type >:: Run()$ calls in one host function (invoked by $Medusa :: Run()$). The iteration is performed iteratively until predefined conditions are satisfied. Medusa offers a set of configuration parameters and utility functions for iteration control.

To demonstrate the usage of Medusa, we show an example of the SSSP (Single Source Shortest Path) implementation with Medusa, as shown in Figure 1. The function *SSSP()* consists of three user-defined EMV API function calls, which is the three main steps of the algorithm. First, we use an *EDGE* type API (*SendDistance*) to send tentative new distance values to neighbors of updated vertices. Second, we use a message *Combiner* to calculate the minimums of received distances of each vertex. Third, we

Table 1: EMV APIs

API Type	*Parameters*	*Variant*	*Description*
ELIST	Vertex v, Edge-list el	Collective	Apply to edge-list el of each vertex v
EDGE	Edge e	Individual	Apply to each edge e
MLIST	Vertex v, Message-list ml	Collective	Apply to message-list ml of each vertex v
MESSAGE	Message m	Individual	Apply to each message m
VERTEX	Vertex v	Individual	Apply to each vertex v
Combiner	Associative operation o	Collective	Apply an associative operation to all edge-lists or message-lists

Table 2: System provided APIs and parameters in Medusa

API/Parameter	*Description*
AddEdge (void* e), *AddVertex*(void* v)	Add an edge or a vertex into the graph
InitMessageBuffer(void* m)	Initiate the message buffer
maxIteration	The maximum iterations that Medusa executes ($2^{31}-1$ by default)
halt	A flag indicating whether Medusa stops the iteration
Medusa :: *Run*(Func f)	Execute f iteratively according to the iteration control
EMV<*type*>:: *Run*(Func f')	Execute EMV API f' with *type* on the GPU

invoke a *VERTEX* type API (*UpdateDistance*) to update the distances of vertices which receive new distances lower than their current distances. In the main function, we load the input graph and configure the execution parameters such as the *Combiner* data type and operation type, the number of GPUs to use and the default iteration termination behavior. *Medusa::Run(SSSP)* invokes the *SSSP* function.

3.2 System Internals

We proposed various optimization techniques for Medusa internals to ensure the high performance. For example, we optimize the graph data layout on the GPU memory, as well as layout of user-defined data structures for the efficiency of GPU memory access. Specifically, Medusa mainly consists of the following key modules.

Graph Storage. The storage module of Medusa allows developers to load the graph data through adding vertices and edges with the system provided APIs *AddEdge* and *AddVertex*. During loading of the graph data, data are stored in our novel graph layout optimized for GPU access [20]. Compared with the classic adjacency list layout, the optimized layout facilitates coalesced access during execution of graph algorithms to ensure high memory bandwidth utilization. After the graph is loaded into main memory, it is automatically transfered to the GPU memory.

Medusa code generation tool chain. Users build Medusa-based programs with standard C/C++. The Medusa code generation tool chain translates the user code into CUDA code. Firstly, Medusa translates user-defined EMV APIs into kernel codes, and generates corresponding kernel invocation codes. Secondly, Medusa translates user-defined data structures, such as data structures for vertex and edge, into GPU data structures and corresponding data transfer codes. Thirdly, Medusa inserts segmented-scan code for the *Combiner* APIs.

Medusa runtime. This module provides runtime support for user applications and manages GPU resource and kernel executions. In particular, this module has three main functionalities. First, it supports the message passing interface. We develop a novel graph aware message passing mechanism for efficiency [20]. Second, Medusa runtime enables transparent execution of user applications on the multi-GPU environment. We further propose techniques such as overlapping kernel execution with data transfer and multi-hop replication to increase the scalability of multi-GPU execution. Third, Medusa runtime supports concurrent execution of multiple Medusa tasks from different users. In the multi-user environment, Medusa coordinates the GPU memory allocation and kernel execution commands to prevent resource allocation deadlocks and exploit opportunities for performance improvement. Meudsa runtime also exploits the complementary resource requirements among the kernels to improve the throughput of concurrent kernel executions. More details about our concurrent kernel scheduling mechanism can be found in our paper [19].

4. RESULTS

We evaluate Medusa with both synthetic graphs generated by GTGraph [7] and publicly available real world graphs [16, 1]. The details of the graph data, including the numbers of vertices and edges, and standard deviations of the vertex degree, are shown in Table 3. Our workload includes a set of common graph computation and visualization primitives. The graph computation primitives include PageRank, BFS, maximal bipartite matching (MBM), ans SSSP. Our experimental platform is a work station with four NVIDIA Tesla C2050 GPUs and two six-core Intel Xeon E5645 CPUs at 2.4 GHz. We developed a set of visualization primitives

Table 3: Details of graphs used in the experiments

Graph	Vertices (10^6)	Edges (10^6)	σ
RMAT	1.0	16.0	32.9
Random (Rand)	1.0	16.0	4.0
BIP	4.0	16.0	5.1
WikiTalk (Wiki)	2.4	5.0	99.9
RoadNet-CA (Road)	2.0	5.5	1.0
kkt_power (KKT)	2.1	13.0	7.5
coPapersCiteseer (Cite)	0.4	32.1	101.3
hugebubbles-00020 (Huge)	21.2	63.6	0.03

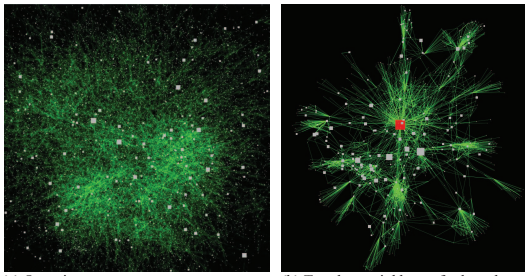
Table 4: Coding complexity of Medusa implementation and hand-tuned implementations.

	Baseline	Warp-centric	Medusa (N/Q)
GPU code lines (BFS)	56	76	9/7
GPU code lines (SSSP)	59	N.A.	13/11
GPU memory management	Yes	Yes	No
Kernel configuration	Yes	Yes	No
Parallel programming	Thread	Thread+Warp	No

such as graph layout and drilling up/down. We implement the force direct layout algorithm [4]. The drilling up/down operation is implemented using BFS. The visualization experiments are conducted on a workstation with one NVIDIA Quadro 5000 GPU and one Intel Xeon W3565 processor with 4 GB memory.

Comparison with Hand-Tuned Implementations. We compare the traversed edges per second (TEPS) of Medusa-based BFS with the basic implementation from Harish et al [8] and the warp-centric method from Hong et al [10]. Compared to the basic implementation, Medusa performs much better on all graphs except KKT. The average speedup of Medusa over the basic implementation is 3.4. Medusa has comparable performance with the warp-centric method, while the latter has much more complicated implementation. We also compare Medusa-based SSSP with Harish et al's implementation. Medusa achieves comparable performance with Harish et al's implementation except on Road and Huge. This is because Road and Huge have large diameters, which lead to large numbers of iterations of Medusa. Table 4 shows the coding complexity of the three implementations. Medusa significantly reduces the developing effort by hiding the GPU programming details and reducing the lines of code.

Comparison with CPU-based Implementations. We compare Medusa with MTGL [2] based multi-core implementations running on 12 cores. Similar to Medusa, MTGL offers a set of data structures and APIs for building graph algorithms. MTGL is optimized to leverage shared memory multithreaded machines. For all the computation primitives, Medusa is significantly faster than MTGL on most graphs and delivers a performance speedup of 1.0-19.6 with an average of 5.5.



(a) Overview (b) Two-hop neighbors of selected author

Figure 2: Visualization results on DBLP co-authorship graph.

We also evaluate the performance of our graph visualization primitives. The input is the co-author graph extracted from DBLP (http://dblp.uni-trier.de/xml/). Figure 2(a) shows the results of our layout primitive on the DBLP graph. On the Quadro 5000 GPU, Medusa takes only 120 seconds to compute the layout, while the 4-thread CPU implementation on the Intel Quad-core CPU takes over 1000 seconds. Figure 2(b) shows the two-hop neighbors of a selected author Jiawei Han obtained by a drill-down operation. Medusa greatly improves the responsiveness of graph visualization tasks.

Scalability. The memory sizes of our input graphs ranges from as small as less than 100 MB (Wiki) to as large as 1 GB (Huge). Our experiments show that Medusa fully utilizes the GPU for all the graph sizes. We also evaluate the scalability of Medusa on the multi-GPU environment with BFS and PageRank. The speedup of BFS and PageRank on four GPUs is 1.8 and 2.6, respectively. BFS is lightweight on computation compared with PageRank. Hence, the communication overhead is larger for BFS, which leads to fewer speedups.

5. USER EXPERIENCE AND LESSONS

In this section, we present a case study on applying Medusa to accelerate information propagation simulations over social networks. Information propagation simulations provide a flexible and valuable method to study the behaviors over complex social networks.

Large-scale network-based simulations involving information propagation often require a large amount of computing resources. It is therefore necessary to develop performance-oriented simulation techniques and to map those optimized simulation methods onto high performance computing (HPC) platforms such as GPUs. For complex networks, the network structures are highly irregular due to

the complicated relationships among the verticess. Such irregularity of the data structure may exhibit very poor memory access locality. The irregularity of the network data structure poses great challenges on efficient GPU acceleration.

5.1 Models of Information Propagation

The simulation of information propagation is to investigate the interactive behaviors between *Active* vertices and *Inactive* vertices within a given network. Currently, the Independent Cascade Model (ICM) [5] and the Linear Threshold Model (LTM) [6] are widely used in studying the behaviors of information propagation over networks. In the ICM, we define an initial set of active vertexs A_0 at step 0. The information propagation unfolds in discrete time steps: at step t, the newly active vertex v_i has a single chance to activate its inactive neighbor u with an independent probability $p_{(v_i,u)} \in [0,1]$. If v_i succeeds in activating u, u will transit its status from inactive to active at step $t+1$ and remain active afterwards [5]. Such a process continues until no more new activations are made in a step. In the LTM, each vertex on the network is assigned with a random threshold $T_u \in [0,1]$. At step t, each inactive vertex is influenced by its active neighbors (a set A_t, where A_t is $\emptyset$ if no active neighbor exists). The influence weight between the active vertex v_i and the inactive vertex u can be expressed as a probability $b_{(v_i,u)}$. Thus, vertex u's influence weight from its active neighbors can be calculated and represented by $\sum_{i=1}^{l} b(v_i,u)$, where l denotes the number of active neighbors and v_i is the ith active neighbor of u [6]. If the transition probability $\sum_{i=1}^{l} b(v_i,u)$ is larger than the predefined threshold value, u's status will transit from inactive to active at step $t+1$ and remain afterwords.

5.2 Implementations and Evaluations

Based on the ICM and LTM models (with adaptions to our application scenarios [11]), we first introduce two types of simulation algorithms named as C-Loop and T-Loop. 1) C-Loop: Starting from the active vertices in the network, each active vertex goes through its neighbors at each simulation step and tests whether it can propagate the information to the inactive neighbors with a specific probability. If the inactive vertices receive the information, they will change status to be active at the next step. 2) T-Loop: In contrast to the C-Loop, the T-Loop starts from the inactive vertices and traces the neighbors' status at each step. The inactive vertex can be activated at the next step by any active neighbor if the transmission probability is satisfied.

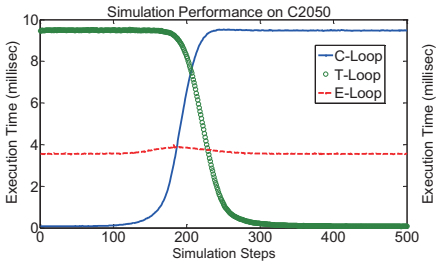


Figure 3: Execution time per simulation step.

Both C-Loop and T-Loop can be easily implemented using the *ELIST* API provided by Medusa. Since the processing of neighbors in C-Loop and T-Loop imposes no edge order constraint, we also propose to use *EDGE* API to implement similar functionalities as C-Loop and T-Loop. This is inspired by the analysis from Medusa that *EDGE* API alleviates the load imbalance problem of *ELIST* API and exhibits better data access performance. We name the *EDGE* API based algorithm as E-Loop. Different from the vertex-oriented approach (C-Loop and T-Loop), the E-Loop approach starts from each edge element and checks the status of the connected pair of vertices. If the two connected vertices have different status such as Active-Inactive, the information can be propagated from active to inactive with the given probability.

Figure 3 shows the execution time per simulation step on C2050. The dataset is a random graph generated by GTGraph [7] with one million vertices and 8 million edges. Due to different memory access patterns, the above three algorithms can exhibit different costs in each step of the simulation. For example, the cost of a C-Loop step is initially small due to the small number of active vertices. As the number of active vertices increases, the cost of a C-Loop step increases. The cost of T-Loop iterations is opposite to that of C-Loop. In contrast, the cost of a E-Loop stays stable in different iterations. Compared to the CPU serial simulation performance, the C2050 GPU based simulation shows 12.5x, 13.1x, 15.6x speedup with CLoop, T-Loop, and E-Loop, respectively [11]. The different characteristics of the algorithms allow us to adaptively choose the best algorithm based on the per-step simulation information. More details on this adaptation can be found in our paper [11]. We also experiment with synthetic and real world graphs with varying sizes and obtained consistent speedups [11, 13].

Simulation on Multiple GPUs. Using multiple GPUs for the simulation is a fast way to increase the memory and computation capacities of the system. We use the default graph partitioning method of Medusa to partition the graph. Medusa automatically builds the replicas for each partition and handles the update of the replicas. Thus, with Medusa, the network-based information propagation is enabled on multiple GPUs with minimized effort. The simulation performance is improved by 2.7 times on four GPUs compared with on one GPU. More details can be found in our paper [12].

5.3 Experience on Using Medusa

Medusa requires no knowledge of GPU programming and greatly simplifies our work on utilizing GPUs for information propagation simulation. First, the programming model of Medusa fits well with the real information propagation process. Information propagation over social networks usually happens among neighboring vertices. Similarly, Medusa allows users to formulate their algorithms with the granularity of individual vertices or edges. Second, the individual and collective APIs of Medusa allow us to develop different approaches (i.e., vertex-oriented and edge-oriented) for the simulation, leading to more opportunities for improving the overall performance of the simulation. Third, despite the fact that Medusa provides an abstract data model and hides the GPU related implementation details from users, experienced users can still easily access the underlying data structures and conduct further customization. Forth, a Medusa program can transparently run on multiple GPUs. This feature of Medusa allows the user to enjoy the benefits of multiple GPUs (more memory space and computation power) with little extra implementation effort.

6. CONCLUSIONS AND FUTURE WORK

The design and implementation of Medusa show that parallel graph computation can efficiently and elegantly be supported on the GPU with a small set of user-defined APIs. The fine-grained API design and graph-centric optimizations significantly improve the performance of graph computation on the GPU. As for future work, we are considering offering dynamic graph processing support in Medusa, and extending Medusa to large scale systems such as clusters and clouds.

The source code of Medusa is available at `http://code.google.com/p/medusa-gpu/`.

7. ACKNOWLEDGEMENT

The authors would like to thank anonymous reviewers for their valuable comments. This work is supported by a MoE AcRF Tier 2 grant (MOE2012-T2-2-067) in Singapore.

8. REFERENCES

[1] 10th DIMACS implementation challenge. http://www.cc.gatech.edu/dimacs10/index.shtml.
[2] J. W. Berry, B. Hendrickson, S. Kahan, and P. Konecny. Software and algorithms for graph queries on multithreaded architectures. In *IPDPS*, pages 1–14, 2007.
[3] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
[4] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21:1129–1164, 1991.
[5] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters*, 12(3):211–223, 2001.
[6] M. Granovetter. Threshold models of collective behavior. *American journal of sociology*, pages 1420–1443, 1978.
[7] GTGraph Generator. *http://www.cc.gatech.edu/ kamesh/GTgraph/*.
[8] P. Harish and P. Narayanan. Accelerating large graph algorithms on the GPU using CUDA. In *HiPC*, pages 197–208. 2007.
[9] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang. Mars: a MapReduce framework on graphics processors. In *PACT*, pages 260–269, 2008.
[10] S. Hong, S. K. Kim, T. Oguntebi, and K. Olukotun. Accelerating CUDA graph algorithms at maximum warp. In *PPoPP*, pages 267–276, 2011.
[11] J. Jin, S. J. Turner, B.-S. Lee, J. Zhong, and B. He. HPC simulations of information propagation over social networks. *Procedia Computer Science*, 9:292–301, 2012.
[12] J. Jin, S. J. Turner, B.-S. Lee, J. Zhong, and B. He. Simulation of information propagation over complex networks: Performance studies on multi-GPU. In *DS-RT*, pages 179–188, 2013.
[13] J. Jin, S. J. Turner, B.-S. Lee, J. Zhong, and B. He. Simulation studies of viral advertisement diffusion on multi-GPU. In *Winter Simulation Conference (WSC)*, pages 1592–1603, 2013.
[14] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. GraphLab: A new parallel framework for machine learning. In *UAI*, 2010.
[15] S. Sengupta, M. Harris, and M. Garland. Efficient parallel scan algorithms for GPUs. NVIDIA. Technical report, 2008.
[16] Stanford Large Network Dataset Collections. *http://snap.stanford.edu/data/index.html*.
[17] T. White. *Hadoop: The Definitive Guide: The Definitive Guide*. O'Reilly Media, 2009.
[18] J. Zhong and B. He. Parallel graph processing on graphics processors made easy. *PVLDB*, 6(12):1270–1273, 2013.
[19] J. Zhong and B. He. Kernelet: High-throughput gpu kernel executions with dynamic slicing and scheduling. *Parallel and Distributed Systems, IEEE Transactions on*, 25(6):1522–1532, 2014.
[20] J. Zhong and B. He. Medusa: Simplified graph processing on gpus. *IEEE Transactions on Parallel and Distributed Systems*, 25(6):1543–1552, 2014.

The IBM Research Accelerated Discovery Lab

Laura Haas, Melissa Cefkin, Cheryl Kieliszewski, Wil Plouffe, Mary Roth
lmhaas, mcefkin, cher, plouffe, torkroth@us.ibm.com

1. INTRODUCTION

Data analytics is becoming central to modern society. In the business world, financial institutions rely on data analysis to detect and prevent fraud; retailers combine transaction data with social media and emails to build a better understanding of their clients; industrial giants use sensor data to improve their carbon footprint. Meanwhile, bioinformatics, astronomy, and particle physics are just a few of the sciences that are being transformed by the availability of large data sets and new techniques for analyzing data. Cities, governments and social agencies are leveraging data analytics for important causes such as improving public health and planning for (and reacting to) natural disasters.

But the path from raw data to insight, or better yet, predictive or prescriptive capabilities, is still long, error-prone, and expensive. First, data must be acquired – not only the pertinent domain data, but often reference and/or contextual data, such as data on weather, economics, demographics, or maps. An appropriate systems infrastructure is needed to store and process the data. Once that infrastructure is acquired, the data must be cleansed, integrated and transformed before the real analysis even begins. Each of these steps requires different tools, and often expertise not only in those tools but also in the various data sets, in data management, and mathematics. Analysis itself involves more tools, deep knowledge of the domain of inquiry, and, if the volume or velocity of data to be analyzed is high, substantial systems and algorithmic skills may be required as well to achieve acceptable performance, with the necessary accuracy. Few people have this broad range of knowledge; thus, many experts need to collaborate across disciplines to achieve the desired insights.

The IBM Research Accelerated Discovery Lab is a unique, collaborative environment specifically designed to facilitate analytic research projects that require multiple participants who may be from several disciplines, and even several institutions. Discovery, in our context, means the gaining of new insight or understanding, often with the intent of attaining predictive or prescriptive capability, where the analysis of data plays a central role. The Lab's objective is to accelerate this type of discovery by (1) enabling research in and improvements to the tools and systems that facilitate discovery, and (2) enabling the business person or domain expert who uses the environment to focus on their investigation instead of the systems and data challenges. To accomplish the first two objectives, we also need (3) to understand how discovery occurs, and how it can be accelerated.

2. LAB OVERVIEW

To achieve the Accelerated Discovery Lab's objectives, we focus on the *discovery platform* the Lab provides to support the discovery process, the *partner projects* that leverage the platform, and the studies that explore the *practice of discovery*.

The discovery platform includes a secure cloud environment that supports large-scale data-intensive computations and a software system that encourages discovery. The cloud environment includes several hundred compute nodes, over 12 petabytes (PBs) of online storage, and a high-speed network as the hardware infrastructure; it leverages IBM's Platform Cluster Management and supports a wide variety of information management tools and analytics platforms. The software system includes data curation tools, support for collections of data called data lakes, and a library of analytics tools and models. It also provides LabBook, a social user experience in which our partner projects pursue their research. Both data lakes and the analytics library allow contribution of new elements (data sets or analytics, respectively), which may be created as a result of projects run in the Lab.

We are supporting a diverse set of partner projects of two types. *Analytics* projects tackle challenges from multiple domains. They range in scale from month-long investigations by a few researchers of a narrowly-defined question requiring one or two data sets to answer, to multi-year studies by multiple teams that require tens of data sets and petabytes of data. *Systems* projects also range from short-term performance studies of new algorithms or architectures, to longer-term creation of new analytics tools or information integration capabilities. While some projects may be done "in residence" in our collaboration space, described below, most are done by teams who may not be local and may, in fact, be geographically distributed. Hence the discovery platform needs to support collaboration across locations and time zones. Partner projects vary over time; potential partners are chosen based on the alignment of their interests with the Accelerated Discovery Lab's mission, their ability to exploit the Lab's platform, and their tolerance for running in an experimental environment.

Finally, we are exploring the human and social dimensions of large-scale data-intensive research and discovery practices, studying how discovery is conducted to iden-

tify essential technological, informational, and environmental characteristics that can encourage and perhaps even accelerate discovery. At some level, the discovery process can be seen as a set of analytics experiments run over some set of data [1]. But what are the right experiments? What tools should be used? What data? Often an individual researcher relies on familiar or available tools or the advice of colleagues. However, examples such as the discovery of the link between fish oil and Renaud's syndrome [17] show that discovery may also happen when previously isolated projects collide in new and unanticipated ways, or when individuals with different (technical) backgrounds collaborate or exchange ideas. Our studies include observations of our partner projects, as well as experiments with the software and physical environments to understand how to provide the best conditions for discovery, including, perhaps, serendipitous interaction that sparks insight.

One of the affordances of the Accelerated Discovery Lab is a 7500+ square foot workspace that provides a flexible work environment for individuals and groups. The space is outfitted to facilitate creativity and collaboration through access to simple, yet effective tools such as whiteboards and displays that can be moved and configured for the needs of those using the space. Also, with our researchers and partners scattered worldwide, we need to be sure that all can participate in planned and *ad hoc* engagement. The workspace thus includes standard collaboration technologies such as video and web conferencing, along with less standard tele-presence robots that allow remote wandering through the room. The space not only affords our researchers and clients a place to work and explore, it also provides a rich environment for collecting data to support our discovery practices research, using methods such as interview, observation, and log data analysis.

Each of our three research thrusts, the platform, partner projects and discovery practice studies, is driven by researchers from different disciplines. The platform research is driven by computer scientists; our core team includes database, human-computer interaction, and systems researchers. The partner analytics projects are typically staffed by domain researchers or analysts; some are "data scientists" with strong data or algorithmic skills. The systems projects, by contrast, are led by computer science researchers, some of whom may have analytic skills. Finally, the discovery practice studies are led by teams of social scientists from such disciplines as anthropology, sociology and social computing. Thus the Lab itself is a multi-disciplinary research environment, mirroring the discovery projects it supports.

Many other groups have or are creating institutes that focus in one way or another on data-driven discovery. Physical science labs[1] have built substantial cyber-infrastructure to support sharing data and tools for analytics. As interest in data analytics has expanded, many universities have formed data science institutes[2], typically multi-disciplinary endeavors that attempt to bring computer scientists, statisticians and domain researchers together to solve domain-specific problems. In the commercial realm, data marketplaces[3] are starting to add computational analytics capabilities to the collections of data sets they provide. Meanwhile, computer science efforts such as CLDS[4] and Berkeley's AMPLab[5] bring together several branches of expertise to improve the systems for doing analysis, and to prove them on real domain-specific challenges. The Accelerated Discovery Lab has many elements in common with each of these efforts; however, to the best of our knowledge we are unique in our emphasis on supporting the overall discovery process (section 4) and our focus on understanding, from a social science perspective, how discovery happens and how it may be accelerated (section 6).

This paper is organized as follows. In the next two sections we provide more detail on the cloud environment and the software systems of the discovery platform, respectively. Section 5 gives a few examples of current partner projects, both analytics and systems, while Section 6 addresses our studies of discovery practices. We discuss where we are today, the research that is currently underway, and where we hope to go in the future.

3. DISCOVERY CLOUD ENVIRONMENT

The discovery cloud is the backbone of the discovery platform, where data lives, and algorithms are tested. It consists today of almost 500 multi-core servers, a high-speed network from our partner, Juniper Networks[6], and capacious storage. The flexible hardware infrastructure provides a rich experimental platform tuned to run large-scale data-intensive analytics, and supports the key analytics tools our partners need. A key consideration in our design is ensuring that data and systems are protected. The architecture allows secure access by authorized researchers (IBM and external) to both private and open data. Measures taken to ensure privacy include restricting access to the systems, secured logins (LDAP), role assignments, security scans, and controlling internet access.

[1] E.g., SLAC: https://www6.slac.stanford.edu/ or CERN: http://home.web.cern.ch/about/computing

[2] For example: http://datascience.nyu.edu/ or http://vcreearch.berkeley.edu/datascience/overview-berkeley-institute-for-data-science or the joint Argonne and Univ. of Chicago Computation Institute: http://www.ci.anl.gov/data-computation

[3] E.g., Microsoft Azure, http://datamarket.azure.com/ or Amazon, https://aws.amazon.com/datasets

[4] The Center for Large-Scale Data System Research, http://clds.sdsc.edu

[5] https://amplab.cs.berkeley.edu/

[6] QFabric, from www.juniper.net

Projects can be physically isolated from each other, or run in a shared pool, depending on their sensitivity.

The physical machines are of two types. The compute server is characterized by a 1:1 ratio between hardware cores (processors) and drives (spindles), using physical drives so that seeks may be overlapped with other I/O. This configuration is best for physical or virtual (VM) systems that need to optimize the parallel I/O or to use local storage. Such systems include Hadoop and its various implementations, e.g., IBM BigInsights and Cloudera, and IBM SPSS Statistics Server. Compute servers have 12 Intel x86 cores, 12 2TB drives and either 96 or 192 GBs of main memory.

Used as a physical server, the compute server is dedicated to a single purpose – usually a single partner project. When hosting VMs, it may be shared by multiple projects. High performance computing applications run on physical servers for best network utilization. Finally, projects that use sensitive client data would have dedicated servers (even if running VMs) so that the data can be permanently destroyed by wiping the disks at the project's end.

The second type of server, the hypervisor server, is used to host multiple VMs, which may be for multiple projects. These would be VMs that are either compute or memory bound and do not need locally attached drives because of either low I/O bandwidth requirements or a small drive footprint. Such servers are used by web servers, database, and user interface servers. Hypervisor servers are bigger systems, with 32 to 40 Intel x86 cores, 128 to 512 GBs of memory, and 6 1TB drives that can be configured as needed by the applications.

All servers are connected by a Juniper Networks QFabric Ethernet backbone (four 40Gb links to each top-of-rack switch) with two 10Gb links to each compute server. A separate 1Gb Ethernet network is used to support management and monitoring services. Shared data services are provided by GPFS-SAN for data sets not requiring large bandwidth, while GPFS-FPO (a cluster file system utilizing local drives) provides substantially higher, distributed bandwidth for larger datasets requiring parallel access.

Today, we can support a large number of projects with flexible, scalable runtime environments for discovery. Each project can experiment with configurations and software as needed, providing ultimate flexibility. Most of our projects run on Red Hat Linux for stability, but a few use Ubuntu or Fedora to gain access to particular features or because the analytics packages require it. We use IBM's Platform Cluster Management Advanced Edition for basic Hadoop clusters, and leverage OpenStack for other application images. Over time we are standardizing images for our analytics projects, enabling us to relieve them of the demands of systems set-up and management, while allowing the systems projects to exploit the underlying hardware as needed.

4. SOFTWARE TO FOSTER DISCOVERY

The second piece of the discovery platform is software that fosters discovery. We focus on enabling two key elements of discovery: *insight* (the aha!) and *collaboration*. While no one can force insight, our software gives researchers new ways to look at a problem. The software presents contextual data and analytics to enrich core domain data and algorithms; it provides exposure to other researchers' ideas and work, aiming to spark new hypotheses. The analytics projects we support represent collaborations by individuals and teams, often spanning multiple domains of expertise. Our software lowers the barriers to cross-fertilization and supports collaboration across individuals and projects, creating the right conditions for insight and "strategic" serendipity. This section elaborates on these themes.

4.1 Contextual Data and Analytics

Contextual data and analytics can enrich core domain data and algorithms, providing new insights. For example, DNA samples from surfaces in a city such as turnstiles, public railings, and elevator buttons can be analyzed to identify what microbes are present at each location, but it is contextual data and analytics such as demographic data and traffic pattern computations that bring insight into patterns of microbes across neighborhoods, income level and populations. Contextual data and analytics can be used and reused across projects and in a variety of domains, and access to both are central to the mission of the Accelerated Discovery Lab. For example, data provided by government agencies such as the Census Bureau and the Bureaus of Labor Statistics and of Economic Analysis can provide location-specific population and income data across many domains. Other important contextual information includes worldwide patent data, medical journals, SEC filings and geo-spatial analytics packages such as those offered by Esri[7], one of our business partners.

Finding and preparing the right contextual data for a project are crucial to deriving insight, but are difficult tasks, particularly for non-technical users. Most data providers supply a simple hierarchical catalog of data sets organized by topic or category. Browsing the catalog of a large provider such as data.gov, with over a hundred thousand data sets, can be daunting, as users rarely know exactly what they are looking for. Once found, preparing data is a tedious process involving manual downloading and at least lightweight data modeling and transformation, skills that most non-technical users lack. For example, **a single** zip file from the Bureau of Economic

[7] www.esri.com

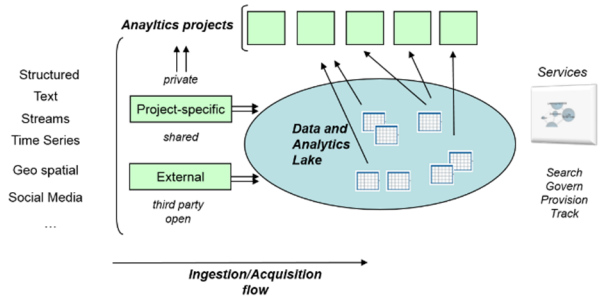


Figure 1. A data lake enables governed reuse of data sets.

Analysis containing National Income and Product (NIPA) data was found to contain nine spreadsheets, which can be transformed ultimately into **116** structured tables.

The Accelerated Discovery Lab provides *data lakes*[8] that can ingest data and analytics from a variety of sources, both open sources (e.g., data.gov) and third party providers, making both contextual and project-specific data available to our researchers (Figure 1). Data lakes store and catalog data, making it easy to track, govern, and repurpose, and ensure compliance with individual licensing terms and conditions. Specific projects may contribute data or analytics to a common lake (and combine them with contextual sources), but data or algorithms need not be shared if there are privacy or security concerns. Multiple data lakes are supported; this allows, e.g., aggregating data on particular themes. A project may also transform data and contribute the result back to a lake where it is cataloged and made available to others.

Data lakes include tools to facilitate and automate the data acquisition process, including tools to pull from standard publishing APIs such as Socrata[9] and ckan[10], and tools that analyze files such as the NIPA zip file. These tools derive and store structured tables and record provenance information about them, including the source and any additional metadata such as semantic tags, publishing organization, etc., that were captured as part of the analysis. Such metadata provides valuable governance and provenance information. Without governance, the use and re-use of data can lead to data management and legal challenges.

As shown on the right of the figure, a data lake provides a set of services to search for and provision data and analytics for use with multiple runtimes. These include direct access services, Extract, Transform, Load (ETL) platforms such as IBM InfoSphere Information Server[11], and multiple Hadoop distributions. Data lakes store the licensing terms and conditions associated with the data and analytics, automatically record use, and ensure compliance. Data lake services are provided via APIs and surfaced through a user experience that encourages discovery.

4.2 A Socially-Inspired User Experience

While access to a rich set of data, analytic tools and runtimes are critical for a data analytics project, so are the experts who do the work. These may include domain experts, e.g., the microbiologists or meteorologists, as well as "general purpose" analysts, e.g., statisticians and experts in data mining. Our software, therefore, needs to support interactions across such multi-disciplinary teams. Each team member may use their own tools consistent with their area of expertise, but also need to exchange ideas with others in the group, or with other experts they consult.

We believe that the 'in-between' knowledge generated as these experts work together or separately can be critical to the discovery process, as it supports both insight and collaboration. LabBook, our user experience [11], places both data lake services and analytics work in a social context. Our system supports social actions such as following and tagging not only other users, but also data, analytics and associated metadata, such as the publisher of the data. In effect, we think of data, analytics and metadata along with users as first-class social entities. The user experience facilitates a meaningful conversation among these entities to guide discovery, suggesting additional or alternative pathways for new insights, and explicating provenance and process.

Accelerated Discovery Lab users have a home page with their profile and access to a set of applications and services appropriate for their role (Figure 2). Users interact with services in the context of a *notebook*, which captures and persistently stores the users' activities – e.g. data sets accessed, analytics run, and comments made. Notebooks can be private, shared, or made public, allowing one or more users to exchange ideas, knowledge and expertise to facilitate collaboration between team members and among a community of individuals with similar interests, with both the flow and dialog of the exchange captured in the notebook. Thus, notebooks themselves constitute collaborative metadata that capture relationships between individuals, data and applications. This allows the system to provide governance and track provenance of data and analytics assets used within the Accelerated Discovery Lab naturally and seamlessly.

Further, this collaborative metadata supports new models of exploration, which may lead to new insights. We have seen that a search for the right data or analytic often relies on a combination of clues from a user's social network, technical knowledge and semantic understanding of the

[8] For simplicity, we will refer to a *data lake*, although in practice the data lake contains both data and analytics.
[9] http://www.socrata.com/
[10] The open source data portal platform, http://ckan.org/
[11] http://www-01.ibm.com/software/data/integration/info_server/

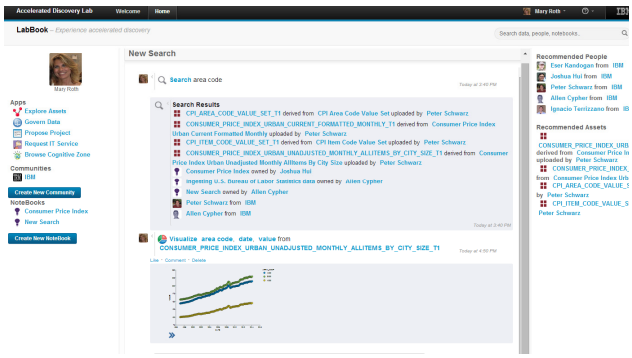


Figure 2. The user experience fosters discovery by combining and leveraging social, technical and semantic metadata.

problem. For example, a user might be looking for data from a particular government agency they had heard was used by a colleague of a friend on a different project. This is exactly the type of association captured in notebooks. To facilitate this more wandering notion of search, notebooks and the implicit and explicit relationships within them are captured as a large federated graph that connects social metadata with semantic tags and schema metadata such as data types. The graph enables powerful new contextual search and recommendation services. As shown in Figure 2, a search by the topic "area code" leads not only to data and analytics related to that topic, but to notebooks in which data related to the topic were referenced, as well as people who have worked with relevant data on other projects. Users may find previous explorations that might be related to their current project, giving them immediate insight into that discovery process by means of the captured dialog and tags, and allowing them to quickly find, use, or extend existing data or analytics from the previous exploration in the new project. The context of the new project is likewise saved and indexed, making it available for search and to serve as a reference for future projects.

5. PARTNER RESEARCH PROJECTS

We started hosting our first partner research projects in late 2012, and have over a dozen projects underway. Here we describe a few of both the analytics and the systems projects.

5.1 Analytics Projects

We look for analytics projects for the Accelerated Discovery Lab that have cross-disciplinary interactions and the need for our infrastructure and tools. We try to balance diverse projects (to study discovery patterns across domains and technologies) with projects that have some commonality in either domain or tools (to explore how serendipitous interactions and cross-project collaboration may aid in the discovery process). Our goal is that the output of these projects will enrich the Lab's portfolios of data assets and analytic building blocks to further speed discovery for future projects, ensuring that the more the Lab is used, the better it gets. To illustrate how these considerations work, we look at examples of two subclasses of analytics projects.

5.1.1 Projects that leverage text analytics

Text analytics is one of IBM Research's strengths [3, 4], so many of our research partners are finding novel ways to exploit this technology in a range of fields, from medicine to finance to marketing. Here we sketch three representative projects, and discuss the assets they use and those they produce.

Knowledge Integration Toolkit (KnIT). The scientific literature is vast, and increasing exponentially [13]. In the field of cancer biology, over 70,000 papers have been written on a single critical tumor protein, p53 [7]. Since no researcher can read all of the relevant literature, most work from only a fraction of the available knowledge. A joint research team from Baylor College of Medicine and IBM Research is using text analytics, knowledge representation, and machine learning to build a model of what is known about p53, and then to suggest opportunities for experiments that could increase our knowledge. Such a tool could spark new discovery by oncology researchers. KnIT extends previous work [18] on a platform for chemical literature search, creating new information extraction and entity resolution rules for the biology domain, then mining the literature and analyzing the results. Since the rate of certain types of discovery in this field is known, we can measure the acceleration of discovery we achieve. For example, the rate of discovery of kinases that phosphorylate p53[12] has averaged one a year for the last decade; the KnIT team has already predicted several previously unknown p53 kinases, with two showing promise in experimental (wet lab) validation.

Waterfund. This project uses the same underlying text analytics in a different domain, finance, along with entity resolution and integration technology [2] and creates new text extraction rules, entity integration rules and data sets. IBM researchers worked with Waterfund[13] analysts to produce a financial index, the Water Cost Index, to track the cost of water in different geographies around the world. The goal is to encourage investments in water treatment facilities by giving a clearer view to lenders, insurers, and governments of the value, costs and associated risks of these projects. The information needed to understand these elements, however, is scattered across many different documents, including reports from public

[12] Kinases are enzymes; phosphorylation adds phosphates to a protein (p53), changing the cell's behavior.

[13] http://Worldswaterfund.com

utilities, newspaper articles, and so on. The team defined a Normalized Production Cost Statement to compare the costs of different agencies and populated these statements through scalable text analysis and integration of company filings. The index has been published regularly since Oct. 2013 and we expect the resulting index stream data to be of interest to other finance projects.

System U. Social media data can teach companies a lot about their clients [8] and their brand image [14]. The System U project is using analytics that derive an individual's personality portrait from his/her social media stream to help companies gain a deeper understanding of their customers, employees, and partners. Such people insights can then be used to help a company to optimize its business outcome, including delivering more individualized products and services to their customers by better matching between their brand and their patrons. With as few as 200 Twitter tweets, System U can derive a personality portrait of an individual or a "company" (as expressed through its social media posts) based on the words used and their frequency of use [6]. The researchers are working with multiple companies to understand if this type of analysis can improve business results. Leveraging the same entity resolution tool as described above, System U helps companies combine existing enterprise customer data (e.g., transaction records) with the personality portraits derived from social media to create enriched, actionable customer portraits.

This sampling of text analytics projects shows both their diversity and their underlying commonality. While these projects involve different researchers, in different domains, they have each benefited from and contributed to the expertise, data and assets available through the Accelerated Discovery Lab. As we enhance our discovery software, we expect to stimulate further interactions, and measure their impact on projects.

5.1.2 Prescriptive analytics projects

Another important subclass of projects leverages a combination of sensor and contextual data, and makes heavy use of machine learning and statistical packages. Here we highlight two such projects, reflecting on their commonalities and differences.

Equipment Condition Monitoring. A key challenge for the mining industry is equipment maintenance. Servicing equipment too soon costs millions in lost revenue; running it too long may result in damage that costs even more to repair. Today, most companies use time in service to decide when to pull a machine in for maintenance, since, in practice, it can be difficult to find good predictors of failure. Using DB2, SPSS, and R, the first phase of this project [9] analyzed data from monitoring 39 components of 50 mining haul trucks over six years, to build a predictive model of part failures, and to create a tool that provides an easy way for field foremen to see when a given machine needs service. The next phase of this project is looking at more data for more types of equipment, and at contextual data on terrain and weather.

Bioinformatics. Several projects have leveraged the Lab to develop or test new parallel algorithms for genome-wide association studies (GWAS). As an example, the BlueSNP R package [10] implements GWAS statistical tests in the R language. These calculations are then executed on Hadoop, using the MapReduce formalism. BlueSNP14 makes computationally intensive analyses feasible for large genotype-phenotype datasets. The team is currently focusing on metagenomics, with a goal of creating a new system that will allow routinely testing many thousands of samples against thousands of reference genomes. This work may someday allow sequencing whole ecosystems, with potential to improve food safety and public health.

Although in unrelated domains, these projects benefit from common tools, such as R and Hadoop. Further, both projects are interested in adding weather and geo-spatial data as context to their analyses. They even have similar challenges in dealing with high-dimensional low sample size data in both fields. Hence there is potential for more synergies and interactions going forward.

5.2 Systems Projects

As with the analytics projects, our partner systems projects cover a broad range of topics. We rely on the work of some of these projects, for example, the scalable storage architecture (GPFS-FPO) that provides a robust alternative to HDFS, or the declarative machine learning platform that our analytics projects are starting to exploit. Other projects will likewise become part of our infrastructure as they mature. We give two examples here.

5.2.1 Benchmarking

A number of our systems partners use the environment for testing or benchmarking. For example, our IBM development colleagues used the Lab to benchmark and certify the performance of IBM BigInsights against that of Apache Hadoop. They used the Statistical Workload Injector for MapReduce (SWIM) developed by the University of California at Berkeley, and had their results certified by the Securities Technology Analysis Center (STAC). SWIM provides a large set of diverse MapReduce jobs based on production Hadoop traces obtained from Facebook, along with information to enable characterization of each job. The STAC report [16] showed that BigInsights completed the jobs four times faster than Apache Hadoop running on the same 18-node environment. BigInsights was about eleven times faster using the

[14] Implementation: http://github.com/ibm-bioinformatics/bluesnp

"sleep" test of scheduling speed. These types of experiments help us tune the configurations we offer the Lab's analytics users, by providing insight into what works best for particular workloads.

5.2.2 Novel Analytics Platforms

The Accelerated Discovery Lab provides a diverse set of applications that both inspire and test new analytic platform ideas. Some platforms, such as SystemT [3], become indispensable to a set of projects. These projects provide proof points for the technology and speed its adoption into products, and the products are then brought into the Lab to accelerate future analytics projects. In this way, the environment is continually improved by the projects within it.

One systems project being brought into the environment for use by our analytics projects is SystemML [5]. Expressing and running analytics for complex data at scale is challenging for mathematicians and systems researchers alike. SystemML raises the level of abstraction and lessens the burden of programming these algorithms by providing a declarative, high-level language using an R-like syntax extended with machine-learning-specific constructs. This language is compiled to a MapReduce runtime and automatically optimized to the specific data set and cluster configuration the analytics need to run against. We expect providing this system to the analytics projects in the Lab will drive further innovations and improvements to the technology.

6. DISCOVERY IN PRACTICE

As described in the previous sections, we have created an environment and a discovery platform that facilitate the exchange of ideas, technology sharing, and collaboration. Within the Accelerated Discovery Lab, we see an opportunity to better understand individual and team customs, actions, and processes (a.k.a., practices) in the context of a large-scale data-intensive discovery paradigm. Our discovery practices research focuses on addressing questions such as 'can discovery be identified as it is being enacted or only in hindsight?', 'how is discovery organized over time?', and 'how do different ways of organizing work affect discovery?' We examine discovery across the many domains of our partner projects, looking at how the practice varies, and across differently-constituted teams to see the role collaborations play. In short, we are investigating the human and social dimensions of what it means to accelerate the ability to 'discover'.

Business and scientific efforts exist in a complex ecosystem composed of many relationships. Thus, we view the Accelerated Discovery Lab as a system-of-systems in which work and discovery is enabled and performed through an arrangement of technical, social, and spatial elements that form into identifiable patterns [12] that can be studied for the purpose of understanding, augmenting, enhancing, or hastening discovery. Each system brings with it a set of resources, whether the resource is an algorithmic tool or data set (technical), an expert or specialist (social), or a particular place (spatial). For example, our metagenomics project entails genomic data, spectral data analysis, bioinformaticians and a locale whence the microbiome is sampled. Captured and examined, these elements could lead to a better understanding of how discovery organically occurs.

Through both field and lab studies, we are investigating how these systems are configured by participants and teams over time and analyzing the key characteristics of the discovery process. Central to our research is the identification of system relationships, patterns of work and interaction, and typologies of discovery that will lead to a fuller understanding of how to represent, measure, and better enable discovery.

7. SUMMARY

In this paper we sketched the design and activities of the IBM Research Accelerated Discovery Lab. The lab is built on an analytics cloud environment with a unique software system that supports the process of discovery, facilitating collaboration and fostering insight. It facilitates a diversity of analytic and systems research projects that span disciplines and institutions. We are studying the practice of discovery, and using our findings to better enable and accelerate it.

8. ADDITIONAL AUTHORS

E. Kandogan, P. Schwarz, J. Hui, A. Cypher, I. Terrizzano, M. Bencala, L. Anderson, J. Vaughan, P. Selinger, R. Moore, O. Anya, P. Maglio, D. Pease, J. Janiak, J. Colino, G. Weber, M-T Schmidt.

9. ACKNOWLEDGMENTS

We thank our many research partners for allowing us to brag about their work. In particular, R. Krishnamurthy, S. Spangler, B. Reinwald and M. Zhou reviewed and improved our descriptions of their work. Any mistakes, however, are the authors' own.

10. REFERENCES

[1] Akil, H. 2003. Scientific strategy in neuroscience: discovery science versus hypothesis-driven research. Message from the President, *Neuroscience Quarterly*. Society for Neuroscience. https://am2012.sfn.org/index.aspx?pagename=neuroscienceQuarterly_03summer_message&print=on

[2] Burdick, D., et al. Extracting, Linking and Integrating Data from Public Sources: A Financial Case Study. *IEEE Data Eng. Bull.* 34, 3 (2011), 60-67.

[3] Chiticariu, L., et al. SystemT: An Algebraic Approach to Declarative Information Extraction. *ACL 2010*, 128-137.

[4] Ferrucci, D., et al. Building Watson: An Overview of the DeepQA Project. *AI Magazine* 31, 3 (2010), 59-79.

[5] Ghoting, A. et al. SystemML: Declarative machine learning on MapReduce. *ICDE 2011* (April 2011), 231-242.

[6] Gou, L., Zhou, M. X., Yang, H. KnowMe and ShareMe: understanding automatically discovered personality

traits from social media and user sharing preferences. *CHI 2014* (April 2014), 955-964.
[7] Hager, K.M. and Gu, W. Understanding the non-canonical pathways involved in p53-mediated tumor suppression. *Carcinogenesis* 35, 4 (Apr 2014), 740-6.
[8] Hernández, M.A, et al. Constructing consumer profiles from social media data. *BigData Conference* (2013), 710-716.
[9] Hochstein, A., et al. Survival Analysis for HDLSS Data with Time Dependent Variables: Lessons from Predictive Maintenance at a Mining Service Provider, *IEEE SOLI* (July 2013).
[10] Huang H, Tata S, Prill R.J. BlueSNP: R package for highly scalable genome-wide association studies using Hadoop clusters. *Bioinformatics.* 29, 1 (2013 Jan 1) 135-6. doi: 10.1093/bioinformatics/bts647. Epub 2012.
[11] Kandogan, E., et al. Data for All: A Systems Approach to Accelerate the Path from Data to Insight. *Big Data Congress, 2013 IEEE International Congress*, 427-428.
[12] Kieliszewski, C.A., Anderson, L.C., and Stucky, S.U. A case study: Designing the service experience for big data discovery. In *Proc. 5th Int'l Conference on Applied Human Factors and Ergonomics* (2014).
[13] Larsen, P.O. and Von Ins, M. The rate of growth in scientific publication and the decline in coverage provided by Science Citation Index. *Scientometrics* 84, 3 (Sep 2010), 575-603. http://dx.doi.org/10.1007/s11192-010-0202-z.
[14] Spangler, S, et al.: COBRA – mining the web for COrporate Brand and Reputation Analysis. *Web Intelligence and Agent Systems* 7, 3 (2009), 243--254.
[15] Spangler, S., Wilkins, A. et al. Automated Hypothesis Generation Based on Mining Scientific Literature. To appear in *KDD 2014* (Aug 2014).
[16] STAC report: Comparison of IBM InfoSphere BigInsights Enterprise Edition with Adaptive MapReduce and Apache Hadoop, using Berkeley SWIM. (October 2013) http://stacresearch.com/node/15370.
[17] Swanson, D. Fish oil, Raynaud's syndrome, and undiscovered public knowledge. *Perspectives in Biology and Medicine* 30, 1 (1986), 7-18.
[18] Yan, S., Spangler, S., Chen, Y.: Chemical Name Extraction Based on Automatic Training Data Generation and Rich Feature Set. *IEEE/ACM Trans. Comput. Biology Bioinform*. 10, 5 (2013), 1218-1233.

Report on the First International Workshop on Exploratory Search in Databases and the Web (ExploreDB 2014)

Georgia Koutrika
HP Labs, Palo Alto
koutrika@hp.com

Laks V.S. Lakshmanan
Department of Computer Science, University of British Columbia
laks@cs.ubc.ca

Mirek Riedewald
College of Computer and Information Science, Northeastern University, Boston
mirek@ccs.neu.edu

Kostas Stefanidis
ICS-FORTH, Heraklion
kstef@ics.forth.gr

1. INTRODUCTION

Databases are well-organized collections of data. Structured query languages, such as SQL, XQuery, and SPARQL, enable users to formulate precise queries over the data stored in a database. To be successful, users need to be familiar with the query language and the underlying data organization. They also need to understand, to some extent, the data stored, and have a fairly clear idea of what they are looking for.

In contrast, the World Wide Web represents the largest and arguably the most complex repository of content, where the above assumptions do not hold. Structured and unstructured data, files and records, multimedia and text, scientific and user-generated data co-exist peacefully on the Web. Free-text queries provide an easy way for users to express their information seeking needs without having to worry about the underlying data organization. Search engines typically act as mediators for user-data interactions on the Web. Given a query, results are most commonly presented as a ranked list. Users subsequently peruse the list to satisfy their information needs through browsing the links and by issuing further queries. This information seeking paradigm has prevailed on the Web for many years as witnessed by the success of search engines, such as Google and Bing.

Despite the popularity and success of querying combined with browsing of data and results, it is worth exploring if this paradigm is still suitable and sufficient in the age of Big Data.

Consider for example `www.data.gov`: this site alone provides access to more than 100,000 different datasets, making it difficult for users to determine which of them could be relevant for their analysis. At the same time, increasingly more applications no longer rely on queries specified by experts. Instead, queries are generated by a diverse and not necessarily programming-aware audience. For instance, consider the Sloan Digital Sky Survey, accessed by domain scientists for a variety of analysis purposes, or the Amazon catalog, where users search over a huge number of products. In these settings, knowing what to look for or how to find it is not easy. With seemingly infinite options, long and repetitive query-browse sessions frustrate users and shake their confidence in the results: "Did I find what I was really looking for?" "Are these good results?" "Did I explore all my options?"

Information on the Web gets rapidly diversified both in terms of its complexity as well as the media through which it is encoded, spanning from large amounts of unstructured and semi-structured data to semantically rich knowledge. Many useful facts about entities (e.g., people, locations, organizations, and products) and their relationships can be found in a multitude of semi-structured and structured data sources, such as Wikipedia, Linked Data cloud[1], Freebase[2], and many others. Increasing demands for sophisticated discovery capabilities are now being imposed by numerous applications in various domains, such as social media, healthcare, e-commerce and web analytics, telecommunications, business intelligence, and cyber-security. Yet, many of these data are hidden behind barriers of language constraints, data heterogeneity, ambiguity, and the lack of proper interfaces.

[1] `http://linkeddata.org`
[2] `http://freebase.com`

Furthermore, the complexity and heterogeneity of the information implies that the associated semantics is often user-dependent and emergent. Individual aspects, such as age, gender, profession, or experience, are often not taken into account, for example, the difference in searching between children and adults. In addition, most systems still assume that the user has a static information need, which remains unchanged during the seeking process. Hence, they are strongly optimized for lookup searches, expecting that the user is only interested in facts and not in complex problem solving.

Consequently, there is a need to develop novel paradigms for exploratory user-data interactions that emphasize user context and interactivity with the goal of facilitating exploration, interpretation, retrieval, and assimilation of information. Ranked retrieval techniques for relational, XML, RDF and graph databases, text, multimedia, scientific and statistical databases, social networks and many others, comprise a first step towards this direction. From a different perspective, recommendation applications tend to anticipate user needs by automatically suggesting the information which is most appropriate to the users and their current context. Recently, new aspects of exploratory search, such as preferences, diversity, novelty, and surprise, are gaining increasing importance. Also, a new line of research in the area of exploratory search is fueled by the growth of online social interactions within social networks and Web communities.

The purpose of the ExploreDB workshop is to bring together researchers and practitioners that approach data exploration from different angles, ranging from data management and information retrieval to data visualization and human computer interaction, in order to study the emerging needs and objectives for data exploration, as well as the challenges and problems that need to be tackled, and to nourish interdisciplinary synergies. We summarize the outcomes of the first workshop instance held in conjunction with EDBT/ICDT 2014 in Athens, Greece.

2. WORKSHOP OUTLINE

The workshop program included a keynote talk, six research papers, and a panel, which examined data exploration from the standpoints of data visualization, information retrieval, Web search, data mining, and database queries.

2.1 Invited Talk

The keynote talk was given by Prof. Daniel A. Keim from the University of Konstanz, Germany, and was entitled "*Exploring Big Data using Visual Analytics.*" With ever-growing volumes of data, Prof. Keim highlighted that effective large-data exploration has to include the human in the process. Specifically, it is important to combine the flexibility, creativity, and general knowledge of the human with the enormous storage capacity and the computational power of today's computers.

Visual analytics naturally integrate the human in the data analysis process and enable her to apply her perceptual abilities to large data sets. Presenting data in an interactive, graphical form often fosters new insights and encourages the formation and validation of new hypotheses for better problem solving and gaining deeper domain knowledge. Visual analytics techniques have proven to be of high value both for the first steps of the data exploration process, namely understanding the data and generating hypotheses about the data, as well as for the actual knowledge discovery by guiding the search using visual feedback.

However, in putting visual analysis to work on big data, it is not obvious where the boundary lies between what can be done by automated analysis and what should be done by interactive visual methods. In dealing with massive data, the use of automated methods is mandatory—and for some problems it may be sufficient. On the other hand, there is a wide range of problems where the use of interactive visual methods is necessary. The keynote speaker discussed when it is useful to combine visualization and analytics techniques, as well as the options on how to combine techniques from both areas. He provided examples from a wide range of application areas that illustrated the benefits of visual analytics techniques.

2.2 Paper Presentations

Sean Chester, Michael Lind Mortensen, and Ira Assent in their paper entitled "*On the Suitability of Skyline Queries for Data Exploration*" studied the data exploration problem based on a sequence of incrementally changing queries to the data. They focused on the skyline operator as a tool of exploratory querying both analytically and empirically. They showed how the results evolve as users modify their queries, and suggested different ways to guide users in formulating reasonable queries.

From a different perspective, George Valkanas, Apostolos N. Papadopoulos, and Dimitrios Gunopulos in their paper "*Skyline Ranking a la IR*" studied a quality-based ranking technique of the results of a skyline query. They described a novel IR-style ranking mechanism for generating such results,

based on the renowned tf-idf weighting scheme, and efficient algorithms to compute the quality of a result and induce a total ordering of the skyline set.

Panagiotis Papadakos and Yannis Tzitzikas in their paper "*Preference-enriched Faceted Exploration*" presented Hippalus, a system for exploratory search enriched with preferences. Hippalus supports the popular interaction model of Faceted and Dynamic Taxonomies, enriched with user actions. These allow users to define preferences, while offering automatic conflict resolution. Preferences can be expressed over attributes (facets), whose values can be hierarchically valued and/or multi-valued.

Marina Drosou and Evaggelia Pitoura in their paper "*The DisC Diversity Model*" considered that diversification can be used as a means for exploration, and they described the notion of DisC diversity. A DisC diverse subset of a query result contains objects, such that each object in the result is represented by a similar object in the diverse subset and the objects in the diverse subset are dissimilar to each other. Locating a minimum DisC diverse subset is an NP-hard problem, hence, they provided heuristics for its approximation.

Haridimos Kondylakis and Dimitris Plexousakis in their paper titled "*Exploring RDF/S Evolution using Provenance Queries*" discussed how to reduce the human effort spent on understanding ontology evolution. They presented a module, named ProvenanceTracker, which receives as input the list of changes between two or more RDF/S ontology versions and can answer fine-grained provenance queries about ontology resources. The module can identify when and how a resource was created, as well as the sequence of changes that led to the creation of that specific resource.

Finally, Steven Simske, Igor Boyko, and Georgia Koutrika in their paper "*Multi-Engine Search and Language Translation*" summarized approaches that focus on improving the quality and accuracy of the search and language translation tasks in the process of interaction between a user and a database. Specifically, multi-engine and related meta-algorithmic approaches are shown to be promising means of improving the performance of both search and translation. They also described a vision of how these tasks can be combined to create a more robust overall text mining project.

2.3 Panel

The panel's theme was "*Exploratory search in databases and the Web—New name for an old hat?*" The moderator, Georgia Koutrika, challenged five panelists on the novelty and future of data exploration and its connection to databases and the Web: Amelie Marian (Rutgers University, USA), Melanie Herschel (Université Paris Sud 11, France), Daniel Keim (University of Konstanz, Germany), Yannis Tzitzikas (University of Crete, Greece), and K. Selçuk Candan (Arizona State University, USA).

Amelie Marian viewed exploratory search from the personalization perspective, and she pointed out the importance of exploring the past when aiming at personalizing the user experience. Such process may include exploration based on personal data, e.g., data from e-mails, Skype, calendars, browser history and file systems, as well as exploration based on social data, e.g., data from Facebook, Twitter, and Foursquare.

Melanie Herschel took the OLAP angle, arguing that learning and investigation are important components of exploratory search. Aspects of both, such as knowledge acquisition, comparison (through the global view of data), aggregation (data warehouses), analysis (OLAP, data mining) and synthesis (provided by reports), have already been solved. However, traditional OLAP querying is constrained by hierarchical dimensions and cube operations, and the limited capabilities for changing and evolving information needs. Intuitively, it is about harvesting what has been planted. On the other hand, exploratory search is about the unknown, creating opportunities for future work, including the *search-refine-expand* paradigm, the freedom to adapt to changing user questions and the fact that learning is an iterative process.

Daniel Keim talked about visual analytics in exploratory search. He described the general goals of data visualization as presentation (visualization of data), confirmatory analysis (visualization of data that allows confirmation or rejection of input hypotheses), and exploratory analysis (visualization of data that provides hypotheses about data). He claimed that exploratory analysis is an open topic with many applications, such as visually exploring complex, semantically ambiguous, dynamic, and uncertain data.

Yannis Tzitzikas spoke from the Web search perspective. He pointed out that Web searching has mainly focused on ranking, but ranking alone is not adequate for exploration. The integration of interaction models for exploring structured and unstructured data, faceted browsing of search results and gradual restrictions for different types of queries and different domains are important topics that have not been (adequately) covered yet. He also highlighted the issue of the applicability of this interac-

tion mode to distributed and heterogeneous sources of varying structural complexity, and the need for better support of the decision making process with user-provided and user-controllable preferences.

Finally, K. Selçuk Candan spoke about multimedia data exploration. Challenges in multimedia exploration arise from special characteristics of such data, including imprecision, sparsity, volume, velocity, variety, high-dimensionality, and multimodality. Selçuk pointed out that there are several directions for future work not (fully) covered so far, including media summarization and dimensionality reduction, multi-modal and richly structured/linked data exploration, dynamic/evolving multimedia exploration, and bridging the semantic gap in media exploration.

3. WORKSHOP CONCLUSIONS

One important message was made clear by the workshop presentations and the participants: given the proliferation of data and applications, there is a need to view data exploration at various levels and from different perspectives. A number of key observations and research directions emerged that we summarize below.

- In databases, much work has been done on generating precise answers for precise queries. In contrast, exploratory search being about the unknown opens up the door to novel information seeking paradigms that focus on the *user-data interaction*, and the need to support an *iterative learning process that adapts to evolving user objectives*.

- In the Web, ranking alone is not adequate for exploration. For developing novel paradigms for exploratory interactions between users and data, *user context and interactivity* need to be emphasized. The integration of interaction models for exploring structured and unstructured data, faceted browsing of search results and gradual restrictions for different types of queries and different domains are important topics not adequately covered yet.

- Exploratory analysis is an open topic with many applications, such as exploring complex, semantically ambiguous, dynamic, and uncertain data. *Different types of data* bring different research challenges at the table.

- Challenges in multimedia exploration arise from special characteristics of such data, including imprecision, sparsity, and multimodality. *Media summarization and dimensionality reduction, multi-modal and richly structured/linked data exploration, dynamic/evolving multimedia exploration* are some critical challenges in multimedia data exploration.

- As yet another example, personal data can be leveraged for *exploring the past and personalizing the user experience*. Personal data exploration can take into account psychological and behavioral patterns to build novel exploration paradigms. For example, in an *associative learning paradigm*, where exploration becomes a learning experience that allows the individual to learn and remember the relationship between unrelated items such as the name of someone in an article and the name of a company in a personal email.

Report on the Sixth International Workshop on Cloud Data Management (CloudDB 2014)

Shuai Ma
SKLSDE Lab
Beihang University
Beijing, China
mashuai@buaa.edu.cn

Xiaofeng Meng
School of Information
Renmin University of China
Beijing, China
xfmeng@ruc.edu.cn

Fusheng Wang
Dept of Biomedical Informatics
Emory University
Atlanta, USA
fusheng.wang@emory.edu

1. INTRODUCTION

The workshop series on Cloud Data Management (CloudDB) was held successfully from 2009 to 2013, co-located with the ACM Conference on Information and Knowledge Management (CIKM) [1, 3–6]. The sixth International Workshop on Cloud Data Management was held in Chicago, IL, USA on March 31, 2014, co-located with the 30th IEEE International Conference on Data Engineering (ICDE) [2]. CloudDB is dedicated to address the challenges in managing big data in the cloud computing environment, and identifying information of value to business, science, government, and society, and it continues serving as a premier forum for researchers and practitioners to present research progress and share ideas in the cloud data management area.

Data management is one of the most important research areas in cloud computing. The huge volumes of data in cloud computing environments pose big infrastructure challenges, including data storage at Petabyte scale, massive parallel query execution, facilities for analytical processing, and online query processing. Meanwhile, the emergence of large data centers and computer clusters has created a new business model, cloud-based computing, for the provision of large-scale computer facilities, where businesses and individuals can rent storage and computing capacities, rather than make significant capital investments to construct. Cloud-based data storage and management is a rapidly expanding business. Whilst these emerging services have substantially reduced the cost of data storage and delivery, there is significant complexity involved in ensuring that they can sustain consistent and reliable operations under peak loads. A cloud-based environment has technical requirements to manage data center virtualization, lower cost and boost reliability by consolidating systems in the cloud. In addition, cloud systems ideally should be geographically dispersed, both to reduce their vulnerability to natural disasters and other catastrophes and to bring data and computation closer to a possibly global user base. This trend brings rise to new and complex technical challenges in the areas of distributed data interoperability and mobility.

This year, the program committee accepted ten papers from nineteen submissions, by authors from Australia, Brazil, China, Germany, Japan, USA and Switzerland. The program covered a variety of topics, including quality of services, query processing, system architecture, and benchmarking. In addition, the program included two invited keynote talks from leading cloud computing researchers.

Many people contributed to the success of this year's CloudDB. First, we would like to thank all authors for submitting their contributions and all attendees for being generous and warm-hearted in all interactions. We would also like to express our deepest gratitude to the program committee members who worked hard in reviewing papers and providing suggestions for improvements. We also give special thanks to our keynote speakers, Geoffrey Fox and Xiaodong Zhang. Finally, we would express our great appreciation to Beihang University, Renmin University of China and Emory University for their supports.

2. KEYNOTE TALKS

The two keynote talks covered topics on big data processing systems and on big data uses and architecture integrating high performance computing and the Apache stack, respectively.

- The first keynote talk was delivered by Xiaodong Zhang from the Ohio State University on "Building Big Data Processing Systems under New Computing Model". Firstly, the speaker introduced the implications of big data processing from a system perspective. (a) Conventional database systems are not designed for big data. (b) Big data users require cost-

effective solutions for their analytics because conventional solutions are not scalable and affordable. (c) System designers and practitioners highly demand various new software tools for big data processing and analytics. (d) Computing paradigm for data processing has been shifted from a scale-up model for high performance to the one for high throughput as the main role of computers becomes data centers. Then, the speaker described how the system community addressed the above mentioned issues with a case study on major technical advancements in Apache Hive, widely adopted by many organizations for big data analytics. In short, the speaker presented a community-based effort and showed how academic research laid a foundation for Hive to improve its daily operations in production systems.

- The second keynote talk was delivered by Geoffrey Charles Fox from Indiana University on "Multi-faceted Classification of Big Data Uses and Proposed Architecture Integrating High Performance Computing and the Apache Stack". The speaker firstly gave a nice introduction of the NIST collection of 51 use cases and their scope over industry, government and research areas. Then, the speaker proposed that in many cases it was wise to combine the well known commodity best practice (often Apache) Big Data Stack (with 120 software subsystems) with high performance computing technologies. Finally, the speaker identified key layers where HPC Apache integration was particularly important: File systems, Cluster resource management, File and object data management, Inter process and thread communication, Analytics libraries, Workflow and Monitoring.

3. RESEARCH PAPERS

The ten accepted papers were divided into four sessions on quality of services, query processing, system architecture and benchmarking, chaired by Dr. Ablimit Aji from HP Labs and Prof. Raymond Chi-Wing Wong from the Hong Kong University of Science and Technology.

3.1 Quality of Services

The proliferation of cloud computing has attracted the deployment of many applications based on the cloud platforms. This obviously raises the issue that how the cloud providers could support high quality services and eventually lead to the complete satisfaction of all sorts of requirements from the cloud users, e.g., in a pay-as-you-go fashion.

(1) Paper "Towards Improvements on the Quality of Service for Multi-Tenant RDBMS in the Cloud" by Leonardo O. Moreira, Victor A. E. Farias, Flávio R. C. Sousa, Gustavo A. C. Santos, José Gilvan Rodrigues Maia, and Javam C. Machado. This paper focuses on the multi-tenant approaches to improving the use of resources, by reducing the operation cost of services, and it proposes an approach to improving quality of service for multi-tenant RDBMS, by employing the migration techniques of tenants, system monitoring, allocation strategy, forecast approach, and benefits of cloud infrastructure to improve performance and reduce provider cost.

(2) Paper "PolarDBMS: Towards a Cost-Effective and Policy-Based Data Management in the Cloud" by Ilir Fetai, Filip M. Brinkmann and Heiko Schuldt. This paper reports the work in progress PolarDBMS towards a flexible and dynamically adaptable system for managing data in the Cloud. PolarDBMS derives policies from application and service objectives, from which it automatically deploys the most efficient and cost-optimized set of modules and protocols, and monitors their compliance. Further, the modules and their customization are changed at running time if necessary.

(3) Paper "SLA-driven Workload Management for Cloud Databases" by Dimokritos Stamatakis and Olga Papaemmanouil. This paper focuses on the challenges related to Service-Level-Agreements (SLAs) specification and management, and argues that SLA management for cloud databases should itself be offered as a cloud-based automated service. For this, it talks about the design of a framework that (a) enables the specification of custom application level performance SLAs and (b) offers workload management mechanisms that can automatically customize their functionality towards meeting these application-specific SLAs.

3.2 Query Processing

The next three papers investigate query processing in cloud based systems for different types of data, from relational data to graphs to data streams.

(4) Paper "Parallel Join Executions in RAMCloud" by Christian Tinnefeld, Donald Kossmann, Joos-Hendrik Boese and Hasso Plattner. This paper studies the utilization of the processing power of large-scale storage systems for supporting

query execution, and it evaluates the parallel execution of join operations in Stanford's RAMCloud, a DRAM-based storage system connected via RDMA-enabled network adapters. To do this, a system model is proposed to derive the execution costs for the Grace Join, the Distributed Block Nested Loop Join, and the Cyclo Join algorithm and their corresponding implementations in RAMCloud, together with a set of heuristics for parameterizing the execution of multiple join operations in parallel for maximizing the throughput.

(5) Paper "Data Stream Partitioning Re-Optimization Based on Runtime Dependency Mining" by Emeric Viel and Haruyasu Ueda. This paper focuses on the optimization of communication cost in distributed data stream processing systems. As programs made of multiple queries can be parallelized by partitioning input streams according to partitioning keys, different partitioning keys for different queries often require intermediary re-partitions, which, as a result, causes extra communication cost and reduces the throughput. It is known that re-partitionings could be avoided by detecting dependencies among the partitioning keys applicable to each query. This paper extends existing (compile-time) partitioning optimization methods by adding a runtime re-optimization module, based on the usage of temporal approximate dependencies among partitioning keys, a type of dependency approximately valid over a sliding window.

(6) Paper "Neighbor-base Similarity Matching for Graphs" by Hang Zhang, Hongzhi Wang, Jianzhong Li and Hong Gao. This paper investigates approximate graph pattern matching which has various cloud data management related applications.

3.3 System Architecture

The following two papers aim at novel cloud systems that could provide better services to deal with the requirements of various applications.

(7) Paper "Curracurrong Cloud: Stream Processing in the Cloud" by Vasvi Kakkad, Akon Dey, Alan Fekete and Bernhard Scholz. This paper focuses on the stream processing systems in a cloud environment, and describes a novel system *Curracurrong Cloud* that allows the computation and data origins to share a cloud-hosted cluster, offers a lightweight algebraic-style description of the processing pipeline, and supports automated placement of computation among computing resources.

(8) Paper "ORESTES: a Scalable Database-as-a-Service Architecture for Low Latency" by Felix Gessert, Florian Bcklers and Norbert Ritter. This paper first describes three major problems that hinder the applicability of database systems in cloud environments: (a) high network latencies for remote/mobile clients, (b) lack of elastic horizontal scalability mechanisms, and (c) missing abstraction of storage and data models. It then introduces an architecture, a REST/HTTP protocol and a set of algorithms towards solving these problems with a Database-as-a-Service middleware called ORESTES, which exposes cloud-hosted NoSQL database systems through a scalable tier of REST servers. These together provide database-independent and object-oriented schema design, a client-independent REST-API for database operations, globally distributed caching, cache consistency mechanisms and optimized database ACID transactions.

3.4 Benchmarking

Database system benchmarks like TPC-C and TPC-E are very useful on evaluating the performance of DBMS systems. The last two papers study the benchmarking of Web-scale transactional databases and cloud-based tagging services, respectively.

(9) Paper "YCSB+T: Benchmarking Web-scale Transactional Databases" by Akon Dey, Alan Fekete, Raghunath Nambiar, Uwe Röhm. Cloud service benchmark frameworks like YCSB are designed for performance evaluation of distributed NoSQL key-value stores, which initially did not support transactions. Recent implementations of Web-scale distributed NoSQL systems, such as Spanner and Percolator, offer transaction features to cater to new Web-scale applications. To fix this gap in standard benchmarks, this paper first identifies the issues to be addressed when evaluating transaction support in NoSQL databases. It then introduces YCSB+T, an extension of YCSB, that wraps database operations within transactions, incorporates a validation stage to detect and quantify database anomalies resulting from any workload, and gathers metrics that measure the transactional overhead.

(10) Paper "Benchmarking Cloud-based Tagging Services" by Tanu Malik, Kyle Chard and Ian Foster. Tagging services have emerged as a

useful and popular way to organize data resources. However, an efficient implementation of tagging services is a challenge since highly dynamic schemas and sparse, heterogeneous attributes must be supported within a shared, openly writable database. This case-study paper describes a benchmark for tagging services, and proposes benchmarking modules that can be used to evaluate the suitability of a database for workloads generated from tagging services. The modules have been incorporated as part of OLTP-Bench, a cloud-based benchmarking infrastructure, to understand performance characteristics of tagging systems on several relational DBMSs and cloud-based database-as-a-service (DBaaS) offerings.

We would like to encourage the interested readers to look into our workshop proceedings for the details of the ten accepted research papers.

4. CONCLUSIONS

CloudDB was held successfully six times associated with CIKM from 2009 to 2013 and with ICDE in 2014. During these six years, cloud computing has undergone significant development and attracted major interests from both industry and academia. Topics of CloudDB cover all sorts of aspects on cloud data management, such as cloud computing infrastructure for big data storage and computing, cloud privacy and security, query processing and indexing access control in cloud computing systems, service-level agreements, business models and pricing policies, novel data-intensive computing applications, massive parallel query execution, data intensive scalable computing, and large-scale analytical methodology and algorithm.

Further, all the participants agreed that many open challenges remain open for cloud data management, such as big data management in the cloud and cloud data security and privacy.

5. ACKNOWLEDGMENTS

The workshop was partially supported by 973 program (No. 2014CB340300), NSFC (No. 61322207), 863 program (No. 2013AA01A213), by Specialized Research Fund for the Doctoral Program of Higher Education (No. 201300041-30001), the Research Fund of Renmin University of China (No. 11XNL010) and by grant R01LM009239 from the National Library of Medicine.

6. REFERENCES

[1] D. W.-L. Cheung, I.-Y. Song, W. W. Chu, X. Hu, and J. J. Lin, editors. *The 18th ACM Conference on Information and Knowledge Management, Hong Kong, China, November 2-6*. ACM, 2009.

[2] I. Cruz, E. Ferrari, and Y. Tao, editors. *The 30th IEEE International Conference on Data Engineering, Chicago, IL, USA, March 31-April 4*. IEEE Computer Society, 2014.

[3] Q. He, A. Iyengar, W. Nejdl, J. Pei, and R. Rastogi, editors. *The 22nd ACM International Conference on Information and Knowledge Management, San Francisco, CA, USA, October 27 - November 1*. ACM, 2013.

[4] J. Huang, N. Koudas, G. J. F. Jones, X. Wu, K. Collins-Thompson, and A. An, editors. *The 19th ACM Conference on Information and Knowledge Management, Toronto, Ontario, Canada, October 26-30*. ACM, 2010.

[5] C. Macdonald, I. Ounis, and I. Ruthven, editors. *The 20th ACM Conference on Information and Knowledge Management, Glasgow, United Kingdom, October 24-28*. ACM, 2011.

[6] X. wen Chen, G. Lebanon, H. Wang, and M. J. Zaki, editors. *The 21st ACM International Conference on Information and Knowledge Management, Maui, HI, USA, October 29 - November 02*. ACM, 2012.

Report on the First Workshop on Linking and Contextualizing Publications and Datasets

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

Lukasz Bolikowski
ICM, University of Warsaw, Poland
l.bolikowski@icm.edu.pl

Nikos Houssos
National Hellenic Research Foundation, Greece
nhoussos@ekt.gr

Jochen Schirrwagen
Bielefeld University Library, Germany
jochen.schirrwagen@uni-bielefeld.de

1. INTRODUCTION

Contemporary scholarly communication is undergoing a paradigm shift, which in some ways echoes the one from the start of the Digital Era, when publications moved to a digital form. There are multiple reasons for this change, and three prominent ones are: (*i*) emergence of data-intensive science (Jim Gray's Fourth Paradigm), (*ii*) evolving reading patterns in modern science, and (*iii*) increasing heterogeneity of research communication practices (and technologies).

Motivated by e-Science methodologies and data-intensive science, contemporary scientists are increasingly embracing new data-centric ways of conceptualizing, organizing and carrying out their research activities. Such paradigm shift strongly affects the way scholarly communication is conducted, promoting datasets as first class citizen of the scientific dissemination. Scientific communities are eagerly investigating and devising solutions for scientists to publish their raw and secondary datasets – e.g. sensor data, tables, charts, questionnaires – to enable: (*i*) discovery and re-use of datasets and (*ii*) rewarding the scientists who produced the datasets after often meticulous and time-consuming efforts. Data publishing is still not a reality in many communities, while for others it has already solidified into procedures and policies.

Due to the ability to have immediate Web access to all published material, be them publications or datasets, scientists are today faced with a daily wave of new potentially relevant research results. Several solutions have been devised to go beyond the simple digital article and facilitate the identification of relevant and quality material. Approaches aim at enriching publications with semantic tags, quality evaluations, feedbacks, pointers to authority files (for example persistent identifiers of authors, affiliation, and funding) or links to other research material. Such trends find their motivations not only from the need of scientists to share a richer perspective of research outcome, but also from traditional and novel needs of research organisations and funding agencies to: (*i*) measure research impact in order to assess and reward their initiatives, e.g. research outcome must be linked to affiliations, authorships, and grants, and (*ii*) guarantee the results of public research is made available as interlinked and contextualized Open Access material, e.g. research datasets are interlinked to related publications and made available via online data repositories and publication repositories. The most prominent example of such requirements is provided by the European Commission with the Open Access mandates for publications and data in Horizon2020.

Finally, researchers rely on different technologies and systems to deposit and preserve their research outcome and their contextual information. Datasets and publications are kept into data centres and institutional and thematic repositories together with descriptive metadata. Contextual information is scattered into other systems, for example CRIS systems for funding schemes and affiliation, national and international initiatives and registries, such as ORCID and VIAF for authors and notable people in general. The construction of *Modern Scholarly Communication Systems* capable of collecting and assembling such information in a meaningful way has opened up several research challenges in the fields of Digital Library, e-Science, and e-Research.

Solving the above challenges would foster multi-disciplinarity, generate novel research opportunities, and endorse quality research. To this aim, sectors of scholarly communication and digital libraries are investigating solutions for "interlinking" and "contextualizing" datasets and scientific publications. Such solutions span from publishing methodologies, processes, policies, to technical aspects involving

data modelling, systems, architectures, and applications. The goal of the first workshop on Linking and Contextualizing Publications and Datasets (LCPD)[1] was to provide researchers and practitioners in the fields of Digital Library, e-Science, and e-Research with a forum where they can constructively explore foundational, organizational and systemic challenges in contexts having publishing, interlinking, preservation, discovery, access, and reuse of publications and datasets.

2. WORKSHOP PRESENTATIONS

All submitted contributions were peer reviewed by three of the seventeen members of the Program Committee and ten were accepted. The workshop structure comprised an invited speakers session followed by the presentation of the ten contributions. Each session is introduced as a separate subsection.

2.1 Keynotes

The workshop had two invited talks respectively covering the aspects of dataset creation and publishing and how Linked Data can be exploited as a mean to leverage scientific publishing.

Sarah Callaghan in her talk entitled "Datasets: from creation to publication" made a clear statement of how research data is becoming central to many scientific disciplines, for which repeatability, verification and transparency of experiments is the key to reward and enable good research. As a consequence, research data should be published in ways like data journals are supporting, e.g. peer review of data stored in data repositories and cited in scientific literature. Access to the data is important to understand and validate conclusions made in research papers. This requires a change in the current scholarly communication practices [4], but also in the culture of scientists, who must ensure that their research stories are transparent, their outcomes viable to sharing, in order to make their research used and trusted by others.

Sören Auer in his talk entitled "How can Linked Data facilitate scientific publishing and knowledge exchange?" suggested the possibility of integrating the Linked Data approach to traditional scientific literature by using tools that would allow researchers to embed structure and semantics to their articles in order to represent them as conceptual RDF graphs, to be then processed by applications more sophisticated than traditional viewers. Examples are document ontologies (e.g. identifying sections, paragraphs, figures, sentence), rhetorical ontologies (e.g. claim, explanation, argument), or semantic annotations (e.g. concepts, external links). Semantic annotation [5] may help finding related work, gaining reputation on social networks, improve visualization, engage researchers with games, and be implemented by researchers in a distributed fashion. In the long term, such practices would increase the number of citations and provide evidence to achieve research funding.

[1] *LCPD2013's web site*, `http://lcpd2013.research-infrastructures.eu`

2.2 Papers Presentation

The presentations from the ten contributions were organized in different sessions, which covered the areas of dataset contextualisation, interlinking datasets and publications, representing and visualizing datasets, and issues regarding packaging datasets and metadata for datasets.

Dataset Contextualization. With respect to dataset contextualisation, Łukasz Bolikowski presented the paper "Tagging Scientific Publications using Wikipedia and Natural Language Processing Tools". The authors propose and evaluate the effectiveness of two methods for contextualizing scientific publications by tagging them with labels reflecting their content. Labels correspond to Wikipedia pages or to noun phrases obtained with NLP tools and can be used as new forms of document representations, for example to enable machine learning tasks, such as document similarity, clustering, topic modelling.

Next, Jon Blower presented the paper "Understanding Climate Data through Commentary Metadata: the CHARMe project". CHARMe is an EC funded project, which aims to link climate datasets with publications, user feedback and other items of commentary metadata. The resulting information system will help users learn from previous community experience and select datasets that best suit their needs, as well as providing direct traceability between conclusions and the data that supported them. Although the project focuses on climate science, the technologies and concepts are very general and could be applied to other fields.

Interlinking Publications and Datasets. Exploring the areas of interlinking research outcomes, Mark Depauw and Tom Gheldof presented the paper "Trismegistos. An interdisciplinary Platform for Ancient World Texts and Related Information". Trismegistos is a metadata platform for the study of texts (e.g. inscriptions) from the Ancient World (roughly 800 BC – AD 800) whose descriptions are kept in several databases worldwide. Its aim is to correlate, i.e. interlink, these digital descriptions to surmount

barriers of language, discipline and geography.

Next, Paolo Manghi presented the work "Preliminary Analysis of Data Sources Interlinking - Data Searchery: a case study". Data Searchery is a lightweight configurable tool supporting researchers at real-time relating publications and/or research data across online data sources by identifying relationships between their metadata descriptions.

Finally, Nuno Lopez reported on the paper "Linked Logainm: Enhancing Library Metadata using Linked Data of Irish Place Names". Linked Logainm is a newly created Linked Data version of Logainm.ie, an online database holding the authoritative hierarchical list of Irish and English language place names in Ireland. The authors demonstrate the benefit of Linked Data to the library community using Linked Logainm to enhance the Longfield Map collection, a set of digitised 18th-19th century maps held by the National Library of Ireland.

Dataset representation and visualization. With regard to dataset representation and visualisation aspects, Marcin Skulimowski presented the paper titled "From Linked Data to Concept Networks". The author proposes the usage of concept networks for scientific publications making use of RDF links between relevant entities from publications. In particular, a web tool supporting creation of concept networks for quantum mechanics was presented.

Subsequently, András Micsik reported on the work "LODmilla: shared visualization of Linked Open Data". LODmilla is a browser embedding the basic commodity features for generic LOD browsing including views, graph manipulation, searching, etc. Users can navigate and explore multiple LOD datasets and they can also save LOD views and share them with other users.

The session concluded with Harry Dimitropoulos presenting the paper "Content visualization of scientific corpora using an extensible relational database implementation". The authors describe a method for the supervised classification and visualization of collections of scientific publications. By integrating a text classification module, which leads to class probability estimation, along with a dimensionality reduction technique, which represents each class in the 2-D space, any collection of unlabelled documents can be intelligibly visualized.

Metadata and packaging of datasets and publications. In this session Nikos Houssos, on behalf of Anna Clements, presented the work "CERIF for Datasets (C4D)". The aim of CERIF for Datasets (C4D) is to develop a framework for incorporating metadata into CERIF (the Common European Research Information Format) such that research organisations and researchers can better discover and make use of existing and future research datasets, wherever they may be held.

The session ended with Catherine Jones presenting the paper "Investigations as research objects within facilities science". The authors explore the notion of data publication in the context of large-scale scientific facilities and propose to publish an investigation, a more complete record of the experiment, including its parameters and context details. In particular they relate this investigation to the emerging concept of a research object [2].

3. WORKSHOP DISCUSSION

The concluding brainstorming session brought up the following main relevant considerations and future issues with respect to publications and datasets.

Publications: beyond traditional papers. Research is focusing on novel forms of publication (e.g. enhanced publications [1]), which are interpreting interlinking and contextualisation of publications and datasets in different, often discipline-specific, scenarios. "Modern" publishing should try to preserve the narrative spirit of literature while integrating procedural aspects of the underlying research (e.g. experiments, workflows), other material, and post-publication information (e.g. semantic tags), etc. Areas of interest identified in the discussion regard annotations, experiments, and linked open data.

Annotation of publications for human and machine interpretation: this area addresses "horizontal" alternatives to the traditional "vertical" reading of paper. Tools, models, and practices are conceived in order to describe the structure of papers (e.g. ordering and interlinking of concepts, rhetorical and reading structure), to enrich papers with semantics (e.g. LOD), and to interlink papers with other research outcome via semantic relationships. Such techniques improve publication discovery capabilities, enable navigation by concepts and improve the ability to interpret research outcomes.

Enabling experiment repeatability and reproducibility: this area focuses on the realization of new digital publications, including a narrative part describing the research and the components necessary to execute the experiment underlying such research. Several solutions exist (e.g. research objects [2]), more or less specific to a different execution environment, but a lot of work has to be done, especially on embedding the life-cycle of such publications within the practices of research infrastructures.

Linked Open Data: this area investigates the potential benefits of adopting LOD techniques in a cross-disciplinary scenario, where common dataset and publication standards cannot be adopted. While it is evident how LOD may fit with the needs of modern scholarly communication when applied within the semantic boundaries of one discipline, it is less clear on how existing experiences could help at leveraging multi-disciplinarity across several domains.

While traditionally an article was the "unit" of publication, contemporary scholarly communication is more fine-grained. Proposed solutions must tackle the heterogeneity of scientific disciplines where the notion of experiment as well as that of datasets changes considerably, e.g. different concept networks, ontologies, metadata descriptions, digital encodings of research datasets. This inherent complexity introduces discipline-oriented practices, standards, and technologies, which can sometime hardly be reused to serve different disciplines. On the other hand, it also leverages effective quality measurement and reuse of scientific results, and introduces a level of granularity which facilitates the identification of cross-links between disciplines.

Datasets: "Publishing without datasets is publishing without evidence, is not research but advertising" (Graham Steel). A crucial challenge in the years to come is addressing the cultural barriers behind data publishing and data citation. These practices are often disregarded by researchers once their results have been published. Research communities should take a rigorous approach and identify the optimal procedures to ensure publications and datasets are published, interlinked and properly described to maximise their discovery and re-usability [3]. In order to achieve real benefits, such procedures should be standardised and imposed to the scientists. Some of the aspects to be studied are: (*i*) guidance about data citation as good research practice, possibly involving publishers and data centres, (*ii*) reward and give visibility to re-use of datasets (e.g. measuring dataset downloads, surveys about the re-use of data), (*iii*) changing the scientific reward paradigm to include dataset production.

On the other hand, data publishing and citation would lose all their appeal without enabling proper data evaluation mechanisms and processes capable of supporting dataset quality control. An area worth investigations is that of annotation of datasets for human and machinery interpretation, adopting techniques which are similar to those used for publications. Datasets could be enriched during their life-cycle (from the collection of raw data to the generation of intermediate secondary datasets) by collecting provenance and lineage information or other application-specific information. Automating dataset peer-review is another challenge. In most scenarios (e.g. data papers) data to be published is quality-checked by humans who verify if datasets are well described and complete, e.g. checking usage of standard formats. This approach cannot scale for large datasets (i.e. big data) or for datasets with non-legible formats (time-series). In such scenarios, dataset quality in publishing workflows should be demanded to proper machinery, e.g. research infrastructure tools used to run scientific experiments.

4. ACKNOWLEDGMENTS

Our sincere gratitude goes to all program committee members, invited speakers, authors, and participants, whose enthusiasm and vision constituted the soul of this workshop and of the future research in the field. The event was funded by the OpenAIREplus project (grant agreement: 283595, Call: FP7-INFRA-2011-2) and the EuroCRIS Consortium.

5. REFERENCES

[1] Bardi, A., and Manghi, P. Enhanced publications: Data models and information systems. *LIBER Quarterly 23*, 4 (2014).

[2] Bechhofer, S., Roure, D. D., Gamble, M., Goble, C., and Buchan, I. Research Objects: Towards Exchange and Reuse of Digital Knowledge. In *The Future of the Web for Collaborative Science (FWCS 2010)* (February 2010). Co-located with WWW'10 Event Dates: April 2010.

[3] Callaghan, S., Murphy, F., Tedds, J., Allan, R., Kunze, J., Lawrence, R., Mayernik, M. S., and Whyte, A. Processes and procedures for data publication: A case study in the geosciences. *IJDC 8*, 1 (2013), 193–203.

[4] CODATA-ICSTI Task Group on Data Citation Standards and Practices. Out of Cite out of Mind: The Current State of Practice, Policy and Technology for the Citation of Data. *Data Science Journal 12* (2013).

[5] Ngomo, A.-C. N., and Auer, S. LIMES: a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three* (2011), AAAI Press, pp. 2312–2317.