SIGMOD Officers, Committees, and Awardees

Chair
Juliana Freire
Computer Science & Engineering
New York University
Brooklyn, New York
USA
+1 646 997 4128
juliana.freire <at> nyu.edu

Vice-Chair
Ihab Francis Ilyas
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario
CANADA
+1 519 888 4567 ext. 33145
ilyas <at> uwaterloo.ca

Secretary/Treasurer
Fatma Ozcan
IBM Research
Almaden Research Center
San Jose, California
USA
+1 408 927 2737
fozcan <at> us.ibm.com

SIGMOD Executive Committee:
Juliana Freire (Chair), Ihab Francis Ilyas (Vice-Chair), Fatma Ozcan (Treasurer), K. Selçuk Candan, Rada Chirkova, Curtis Dyreson, Christian S. Jensen, Donald Kossmann, and Dan Suciu.

Advisory Board:
Yannis Ioannidis (Chair), Phil Bernstein, Surajit Chaudhuri, Rakesh Agrawal, Joe Hellerstein, Mike Franklin, Laura Haas, Renee Miller, John Wilkes, Chris Olsten, AnHai Doan, Tamer Özsu, Gerhard Weikum, Stefano Ceri, Beng Chin Ooi, Timos Sellis, Sunita Sarawagi, Stratos Idreos, and Tim Kraska.

SIGMOD Information Director:
Curtis Dyreson, Utah State University

Associate Information Directors:
Huiping Cao, Georgia Koutrika, Wim Martens, Asterios Katsifodimos

SIGMOD Record Editor-in-Chief:
Rada Chirkova, NC State University

SIGMOD Record Associate Editors:
Azza Abouzied, Lyublena Antova, Vanessa Braganholo, Aaron J. Elmore, Wim Martens, Kyriakos Mouratidis, Dan Olteanu, Divesh Srivastava, Pınar Tözün, Immanuel Trummer, Yannis Velegrakis, Marianne Winslett, and Jun Yang.

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

PODS Executive Committee:
Dan Suciu (Chair), Tova Milo, Diego Calvanese, Wang-Chiew Tan, Rick Hull, and Floris Geerts.

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

SIGMOD Awards Committee:
Martin Kersten (Chair), Surajit Chadhuri, David DeWitt, Sunita Sarawagi, and Michael Carey.

Jim Gray Doctoral Dissertation Award Committee:
Ioana Manolescu (co-Chair), Lucian Popa (co-Chair), Peter Bailis, Michael Cafarella, Feifei Li, Qiong Luo, Felix Naumann, and Pinar Tozun.

SIGMOD Systems Award Committee:
Michael Cafarella (Chair), Michael Carey, David DeWitt, Yanlei Diao, Paul Larson, and Gustavo Alonso.

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. Recipients of the award are the following:

Michael Stonebraker (1992)
David DeWitt (1995)
Serge Abiteboul (1998)
Rudolf Bayer (2001)
Ronald Fagin (2004)
Jennifer Widom (2007)
Umeshwar Dayal (2010)
Stefano Ceri (2013)
Gerhard Weikum (2016)
Anastasia Ailamaki (2019)
Jim Gray (1993)
C. Mohan (1996)
Hector Garcia-Molina (1999)
Patricia Selinger (2002)
Michael Carey (2005)
Moshe Y. Vardi (2008)
Surajit Chaudhuri (2011)
Martin Kersten (2014)
Goetz Graefe (2017)
Philip Bernstein (1994)
David Maier (1997)
Rakesh Agrawal (2000)
Don Chamberlin (2003)
Jeffrey D. Ullman (2006)
Masaru Kitsuregawa (2009)
Bruce Lindsay (2012)
Laura Haas (2015)
Raghu Ramakrishnan (2018)

SIGMOD Systems Award

For technical contributions that have had significant impact on the theory or practice of large-scale data management systems.

Michael Stonebraker and Lawrence Rowe (2015); Martin Kersten (2016); Richard Hipp (2017); Jeff Hammerbacher, Ashish Thusoo, Joydeep Sen Sarma; Christopher Olston, Benjamin Reed, Utkarsh Srivastava (2018); Xiaofeng Bao, Charlie Bell, Murali Brahmadesam, James Corey, Neal Fachan, Raju Gulabani, Anurag Gupta, Kamal Gupta, James Hamilton, Andy Jassy, Tengiz Kharatishvili, Sailesh Krishnamurthy, Yan Leshinsky, Lon Lundgren, Pradeep Madhavarapu, Sandor Maurice, Grant McAlister, Sam McKelvie, Raman Mittal, Debanjan Saha, Swami Sivasubramanian, Stefano Stefani, Alex Verbitski (2019)

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)
Jeffrey Ullman (1996)
Raghu Ramakrishnan (1999)
Daniel Rosenkrantz (2001)
Surajit Chaudhuri (2004)
Hans-Jörg Schek (2007)
David Lomet (2010)
H.V. Jagadish (2013)
Samuel Madden (2016)
Ahmed Elmagarmid (2019)
Gio Wiederhold (1995)
Avi Silberschatz (1997)
Michael Carey (2000)
Richard Snodgrass (2002)
Hongjun Lu (2005)
Klaus R. Dittrich (2008)
Gerhard Weikum (2011)
Kyu-Young Whang (2014)
Yannis E. Ioannidis (2017)
Yahiko Kambayashi (1995)
Won Kim (1998)
Laura Haas (2000)
Michael Ley (2003)
Tamer Özsu (2006)
Beng Chin Ooi (2009)
Marianne Winslett (2012)
Curtis Dyreson (2015)
Z. Meral Özsoyoğlu (2018)

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. *Honorable Mentions*: Marcelo Arenas and Yanlei Diao.
- **2007** *Winner*: Boon Thau Loo. *Honorable Mentions*: Xifeng Yan and Martin Theobald.
- **2008** *Winner*: Ariel Fuxman. *Honorable Mentions*: Cong Yu and Nilesh Dalvi.
- **2009** *Winner*: Daniel Abadi. *Honorable Mentions*: Bee-Chung Chen and Ashwin Machanavajjhala.
- **2010** *Winner:* Christopher Ré. *Honorable Mentions*: Soumyadeb Mitra and Fabian Suchanek.
- **2011** *Winner*: Stratos Idreos. *Honorable Mentions*: Todd Green and Karl Schnaitterz.
- **2012** *Winner*: Ryan Johnson. *Honorable Mention*: Bogdan Alexe.
- **2013** *Winner*: Sudipto Das, *Honorable Mention*: Herodotos Herodotou and Wenchao Zhou.
- **2014** *Winners*: Aditya Parameswaran and Andy Pavlo.
- **2015** *Winner*: Alexander Thomson. *Honorable Mentions*: Marina Drosou and Karthik Ramachandra
- **2016** *Winner*: Paris Koutris. *Honorable Mentions*: Pinar Tozun and Alvin Cheung

- **2017** *Winne*r: Peter Bailis. *Honorable Mention*: Immanuel Trummer
- **2018** *Winne*r: Viktor Leis. *Honorable Mention*: Luis Galárraga and Yongjoo Park
- **2019** *Winne*r: Joy Arulraj. *Honorable Mention*: Bas Ketsman

A complete list of all SIGMOD Awards is available at: **https://sigmod.org/sigmod-awards/**

[Last updated: September 30, 2019]

Editor's Notes

Welcome to the September 2019 issue of the ACM SIGMOD Record!

This issue starts with the Database Principles column featuring an article by Wijsen that discusses the problem of query answering on inconsistent databases. A database is called inconsistent if it violates some integrity constraints; the key issue is what information can and cannot be inferred from inconsistent data in the process of query answering. Intuitively, an inconsistent database can be viewed as a collection of consistent databases, each of which results from some minimal corrections, or repairs, to the original data. Then, an answer to a query on such a database is called consistent if it can be obtained by processing the query on each of the repaired databases. The article summarizes core concepts and theoretical developments that have arisen in foundations of consistent query answering in the past twenty years. The report on the rich body of theoretical results covers studies for common classes of integrity constraints and emphasizes results in computational and descriptive complexity. In general, consistent query answering is intractable. However, as the intractability usually arises in worst-case results, the article calls for research in refinement of the complexity results for practically important cases. The article also discusses open problems, and provides references on related areas.

The Research Articles column features two articles. The first article, by McCullough, Mokfi, and Almaeenjad, studies the problem of accuracy of SQL software for statistical purposes. The study focuses on testing elementary statistical calculations with a collection of benchmark tests known as Wilkinson's tests. These tests have a long track record of being applied to uncover flaws in statistical packages; this article describes their application to six well-known SQL packages. The article discusses the flaws that were identified in the analysis of statistical functions in the SQL packages, provides pointers to relevant algorithms, and gives the best choice of data types for statistical purposes in each package. The authors call for developers to fix the discovered errors, and propose that accuracy of algorithms be incorporated into the SQL standards.

The second article in the Research Articles column, by Cavero, Vela, and Cáceres, discusses the problem of simulating SQL assertions via materialized views. While assertions have been in the SQL standard since 1992 as a powerful means of specifying cross-row and cross-table constraints, they are usually not supported in commercial database-management systems, and are typically implemented by triggers or application programs. The article shows how assertions can be simulated using materialized views that count the number of violations of the assertion's conditions. With the help of a series of tests, the authors demonstrate that materialized views are easier to program and less error prone than triggers or application programs, as well as more efficient than triggers in some situations. The article provides recommendations on application scenarios that can benefit from the proposed assertion-codification approach, and specifies relevant DMBS requirements.

The Distinguished Profiles column features Michael Franklin, inaugural holder of the Liew Family Chair of Computer Science and senior advisor to the provost on computation and data science at the University of Chicago. Before that, for many years, Mike was a professor at Berkeley, where he also served as a chair of the Computer Science division. Mike was a co-founder and director of the Algorithms, Machines, and People Lab at Berkeley, better known as the AMPLab, a leading academic big data analytics research center that received a National Science Foundation CISE "Expeditions in Computing" award. At the AMPLab, Mike was one of the creators of the Spark (now Apache Spark) data analytics and machine learning platform. Mike is a Fellow of the Association for Computing Machinery and a two-time recipient of the ACM SIGMOD "Test of Time" award. Mike's Ph.D.

is from the University of Wisconsin Madison. In this interview, Mike talks about his experience with building computer science and data science at the University of Chicago in a way that integrates these disciplines into the fabric of the university, with opportunities for people with widely varying interests to work together. He shares his views on the prospects for real-time streaming analytics, and discusses how people fit into overall systems architectures. Mike also outlines aspects of computational and data literacy that an educated person in the 21st century needs to know, gives advice on research, and shares information about his own advising style and research.

The Reports column features two articles. The first article, by Palpanas and Beckmann, reports on the First and Second Interdisciplinary Time Series Analysis Workshops (ITISA), which took place in Paris in June and December 2016. Time-series data arise naturally in many applications, and their analysis can push the computational power and other resources to their limits. Over 80 participants in the two ITISA workshops participated in discussions of the challenges and requirements on the new technologies and algorithms. The workshops included 14 keynote talks, hands-on sessions, and panel discussions. The article summarizes the ideas presented and discussed in the workshops, highlighting the relevant state-of-the-art techniques and advancements in time-series management and analysis. The ITISA programs and slides are available from the authors of the article.

The second article in the Reports column, by Kondylakis, Stefanidis, and Rao, reports on the outcomes of the First International Workshop on Semantic Web Technologies for Health Data Management (SWH). The workshop took place in 2018 in Monterey, CA USA, in conjunction with the International Semantic Web Conference. The SWH workshop aimed to bring together an interdisciplinary audience, to discuss challenges in healthcare data management and to propose novel and practical solutions for next-generation data-driven healthcare systems. The article summarizes the outcomes of the workshop, and outlines key observations and emerging research directions.

On behalf of the SIGMOD Record Editorial board, I hope that you enjoy reading the September 2019 issue of the SIGMOD Record!

Your submissions to the SIGMOD Record are welcome via the submission site:
https://mc.manuscriptcentral.com/sigmodrecord

Prior to submission, please read the Editorial Policy on the SIGMOD Record's website:
https://sigmodrecord.org/sigmod-record-editorial-policy/

Rada Chirkova

September 2019

Past SIGMOD Record Editors:

Yanlei Diao (2014-2019)
Mario Nascimento (2005–2007)
Jennifer Widom (1995–1996)
Jon D. Clark (1984–1985)
Randall Rustin (1974-1975)
Ioana Manolescu (2009-2013)
Ling Liu (2000–2004)
Arie Segev (1989–1995)
Thomas J. Cook (1981–1983)
Daniel O'Connell (1971–1973)
Alexandros Labrinidis (2007–2009)
Michael Franklin (1996–2000)
Margaret H. Dunham (1986–1988)
Douglas S. Kerr (1976-1978)
Harrison R. Morse (1969)

Foundations of Query Answering on Inconsistent Databases

Jef Wijsen
University of Mons
Mons, Belgium
jef.wijsen@umons.ac.be

ABSTRACT

Notwithstanding the traditional view that database instances must respect all integrity constraints imposed on them, it is relevant to develop theories about how to handle database instances that violate some integrity constraints, and more particularly, how to cope with query answering in the presence of inconsistency. Such a theory developed over the past twenty years is currently known as *consistent query answering* (CQA). The aim of this article is to summarize and discuss some core concepts and theoretical developments in CQA.

1. INTRODUCTION

Consistent query answering (CQA) started with an article at PODS 1999 by Arenas, Bertossi and Chomicki [2]. Twenty years later, the significance of their contribution was acknowledged through a *Gems of PODS* session, at which occasion Leopoldo Bertossi flashed back to the origins of CQA [6]. Among these origins is the question about "what is consistent in an inconsistent database" [6, p. 49]. The next example illustrates this question, as well as the answer provided by the CQA approach.

Following the well-known running example of C.J. Date's textbook [15], we represent suppliers by a table with name S. Every supplier has a supplier number, unique to that supplier, a supplier name, a rating or status value, and a location. The requirement that supplier numbers be unique is violated by the following table; the available information about the status of S2 happens to be inconsistent.

S	S#	SNAME	STATUS	CITY
	S1	Smith	20	London
	S2	Jones	10	Paris
	S2	Jones	15	Paris

We ask the question of what information can and cannot be inferred from this inconsistent table. A useful answer to that question should rely on some *paraconsistent* inference relation, i.e., one that abandons the principle that "everything follows from an inconsistency" (*ex contradictione quodlibet*). In general terms, the answer provided in [2] goes as follows. Every inconsistent database instance represents a set of possible consistent database instances, which are called *repairs*. Repairs should be obtained by fixing inconsistencies in some minimal way. *Consistently true* information, then, is defined as information that holds true in every repair.

We will formalize shortly the idea of minimal fixing. For the example table, minimal fixing could mean deleting either of the two tuples that agree on S#. This yields two repairs which differ in the status of supplier S2. Other repairs could be obtained by replacing either occurrence of S2 with some fresh supplier number. This would yield many repairs, differing only in the choice of the new supplier number. For the purpose of this example, assume that there are no other repairs than the ones just described. Then, the assertion that "S2 is a supplier having a lower status than S1" is consistently true, because it is true in every repair. It is also consistently true that "the number of suppliers is two or three." On the other hand, the claim that "supplier S2 has status 10" is not consistently true.

The aim of this article is to present and discuss theoretical foundations of CQA, with an emphasis on results in computational and descriptive complexity. We will not report on the deployment of CQA in operational systems. We have tried to be complimenatry to the previously cited *Gems of PODS* article of Bertossi [6], which gives a broad overview, intentionally based on representative examples rather than formal definitions. The treatment in the current paper is more formal and focuses on some core concepts and results.

Organization.

Section 2 recalls some standard notions in database theory. Section 3 formalizes the framework of consistent query answering. The idea of minimal fixing is introduced there in an original manner es-

pecially developed for the purpose of this article. Section 4 discusses different ways for indicating the complexity of consistent query answering, referring to both computational and descriptive complexity. In Section 5, we focus on a fine-grained complexity classification that has been achieved for consistent query answering with respect to primary keys, but which is largely open for other classes of integrity constraints. The case of primary keys is also interesting because of its connections to two other fields: constraint satisfaction problems (CSPs) and probabilistic database systems. Section 6 discusses the complexity of repair checking. Finally, Section 7 concludes the paper.

2. PRELIMINARIES

For every positive integer n, we assume denumerably many *relation names* of *arity* n. A *database schema* is a finite set of relation names. In the sequel, we will often assume that some database schema has been fixed. A *database instance* is a finite set of *facts* $R(c_1, \ldots, c_n)$ where R is an n-ary relation name of the database schema, and each c_i is a constant. An m-ary *query* q, with $m \geq 0$, maps every database instance $\mathbf{db}$ to an m-ary relation, denoted $q(\mathbf{db})$. A 0-ary query is also called a Boolean query: a 0-ary relation represents *false* if it is empty, otherwise it represents *true*. In the complexity study of CQA, much attention has been paid to Boolean conjunctive queries, i.e., queries defined by first-order logic sentences, possibly with constants, of the form

$$\exists \vec{x} \left(R_1(\vec{x}_1) \wedge R_2(\vec{x}_2) \wedge \cdots \wedge R_n(\vec{x}_n) \right).$$

Such a query is said to be *self-join-free* if $i \neq j$ implies $R_i \neq R_j$. Each $R_i(\vec{x}_i)$ is called a *relational atom*.

We assume that our database schema is equipped with a set Σ of *integrity constraints*. All integrity constraints in this paper can be expressed as sentences in first-order logic. In what follows, by a *set of integrity constraints*, we will always mean a finite set of integrity constraints that can be satisfied by some database instance. A database instance is *consistent* if it satisfies all integrity constraints in Σ; otherwise it is *inconsistent*. Common integrity constraints, called *dependencies*, are recalled next. Readers familiar with common classes of integrity constraints can skip the remainder of this section.

Inclusion dependencies (IND).

If R and S are relation names, of arities m and n respectively, then $R[i_1, \ldots, i_k] \subseteq S[j_1, \ldots, j_k]$ is an *inclusion dependency*, where $i_1, \ldots, i_k$ is a sequence of distinct integers in $\{1, \ldots, m\}$, and $j_1, \ldots, j_k$ is a sequence of distinct integers in $\{1, \ldots, n\}$. A database instance $\mathbf{db}$ is said to *satisfy* this inclusion dependency if for every $R(a_1, \ldots, a_m) \in \mathbf{db}$, there exists $S(b_1, \ldots, b_n) \in \mathbf{db}$ such that for every $\ell \in \{1, \ldots, k\}$, $a_{i_\ell} = b_{j_\ell}$.

Functional dependencies (FD).

If R is a relation name of arity n, then a *functional dependency* is an expression $R : X \to Y$ where $X, Y \subseteq \{1, \ldots, n\}$. A database instance $\mathbf{db}$ *satisfies* $R : X \to Y$ if for all $R(a_1, \ldots, a_n)$, $R(b_1, \ldots, b_n) \in \mathbf{db}$, if $a_i = b_i$ for all $i \in X$, then $a_j = b_j$ for all $j \in Y$. If $X \cup Y = \{1, \ldots, n\}$, then $R : X \to Y$ is called a *key dependency*.

Tuple-generating dependencies (tgd).

A $\vee$-*tgd* is a constant-free first-order logic sentence of the form

$$\forall \vec{x} \left(\varphi(\vec{x}) \to \bigvee_{i=1}^{n} \exists \vec{y}_i \psi_i(\vec{x}, \vec{y}_i) \right), \tag{1}$$

where φ is a nonempty conjunction of relational atoms, each ψ_i is a conjunction of relational atoms, and every variable in $\vec{x}$ appears in φ (but not necessarily in $\bigvee_{i=1}^{n} \exists \vec{y}_i \psi_i(\vec{x}, \vec{y}_i)$). $\vee$-tgds can be further restricted as follows:

- a *tgd* is a $\vee$-tgd where $n = 1$;
- a $\vee$-tgd or tgd without existentially-quantified variables is called *full*; and
- a *LAV tgd* is a tgd of the form

  $$\forall \vec{x} \left(R(\vec{x}) \to \exists \vec{y} \psi(\vec{x}, \vec{y}) \right),$$

  where $R(\vec{x})$ is a relational atom.

For a set of tgds, the notion of being *weakly acyclic* is a structural property that guarantees termination of the *chase*. The definition of weakly acyclic can be found in [17].

Universal constraints (UC).

A *universal constraint* is a constant-free first-order logic sentence of the form

$$\forall \vec{x} \left(\varphi(\vec{x}) \wedge \beta(\vec{x}) \to \bigvee_{i=1}^{n} \psi_i(\vec{x}) \right), \tag{2}$$

where φ and each ψ_i are conjunctions of relational atoms, β is a Boolean combination of equalities, and every variable in $\vec{x}$ appears in φ.

Special cases of UCs are obtained by letting $n = 0$, and by considering that the empty disjunction is **false**:

- a *denial constraint* is commonly written in the form $\forall\vec{x}\neg\left(\varphi(\vec{x})\wedge\beta(\vec{x})\right)$, a sentence logically equivalent to $\forall\vec{x}\left(\varphi(\vec{x})\wedge\beta(\vec{x})\rightarrow\mathbf{false}\right)$.

- an *equality-generating dependency (egd)* takes the form $\forall\vec{x}\left(\varphi(\vec{x})\rightarrow x_i = x_j\right)$, which is equivalent to $\forall\vec{x}\left(\varphi(\vec{x})\wedge\neg\left(x_i = x_j\right)\rightarrow\mathbf{false}\right)$.

The form (2) was chosen because of its resemblance to (1). The disjunction $\bigvee_{i=1}^{n}\psi_i(\vec{x})$ in (2) is equivalent to a formula in CNF, say $\bigwedge_{i=1}^{m}\chi_i(\vec{x})$, where each χ_i is a disjunction of relational atoms. The set $\{\forall\vec{x}\left(\varphi(\vec{x})\wedge\beta(\vec{x})\rightarrow\chi_i(\vec{x})\right)\}_{i=1}^{m}$ is then equivalent to (2). Since we always consider sets of integrity constraints, universal constraints can be (and often are) defined in different forms [2, 31]:

$$\forall\vec{x}\neg\left(\begin{array}{l}\neg R_1(\vec{x}_1)\wedge\cdots\wedge\neg R_m(\vec{x}_m)\wedge\\ R_{m+1}(\vec{x}_{m+1})\wedge\cdots\wedge R_n(\vec{x}_n)\wedge\beta(\vec{x})\end{array}\right),$$

$$\forall\vec{x}\left(\begin{array}{l}R_1(\vec{x}_1)\vee\cdots\vee R_m(\vec{x}_m)\vee\\ \neg R_{m+1}(\vec{x}_{m+1})\vee\cdots\vee\neg R_n(\vec{x}_n)\vee\neg\beta(\vec{x})\end{array}\right),$$

$$\forall\vec{x}\left(\begin{array}{l}R_{m+1}(\vec{x}_{m+1})\wedge\cdots\wedge R_n(\vec{x}_n)\wedge\beta(\vec{x})\rightarrow\\ R_1(\vec{x}_1)\vee\cdots\vee R_m(\vec{x}_m)\end{array}\right),$$

where each R_i is a relation name, and each variable in $\vec{x}$ appears in some $\vec{x}_i$ with $m{+}1\leq i\leq n$. The latter requirement is called *safety*. Denial constraints, then, are universal constraints where $m = 0$.

Figure 1 relates different classes of integrity constraints. An upward line from IC_1 to IC_2 means that every set Σ_1 of integrity constraints in the class IC_1 is equivalent to some set Σ_2 in IC_2. Note, for example, that the functional dependency $R:\{1\}\rightarrow\{2,3\}$ expresses a pair of egds.

3. CONSISTENT ANSWERS

In Section 1, we have already given a general but informal introduction to CQA. We will now enter into more technical details.

3.1 Querying Inconsistent Data

Let q be a query and Σ a set of integrity constraints. If $\mathbf{db}$ is a consistent database instance, the *query answer* $q(\mathbf{db})$ can reasonably be called "consistent" as well. The CQA paradigm was developed in the first place to define "consistent query answers" for the case where $\mathbf{db}$ is inconsistent.

When $\mathbf{db}$ is an inconsistent database instance, it looks like a good idea to change it, in some minimal way, such that the new database instance is consistent. Such a consistent database instance obtained by some minimal change is called a *repair*. Let us skip for a moment the details of minimal change, and denote by $\mathsf{repairs}(\mathbf{db},\Sigma)$ the set of all repairs of $\mathbf{db}$ with respect to Σ. Then a tuple t belongs to the consistent answer to q on $\mathbf{db}$ if for each repair $\mathbf{r}$, we have that t belongs to $q(\mathbf{r})$:

$$\mathsf{cqa}(q,\mathbf{db},\Sigma)\triangleq\bigcap\{q(\mathbf{r})\mid\mathbf{r}\in\mathsf{repairs}(\mathbf{db},\Sigma)\}.$$

Of course, this definition makes sense only when the notion of being a repair is well-defined, which is the topic of the next subsection.

3.2 Fixing Inconsistency

The literature on CQA contains many proposals for formalizing the idea of minimal change. For the purpose of this article, we next develop a generalization that captures the essence of most proposals. Our generalization assumes that, for a given database instance $\mathbf{db}$, there is a binary relation $\leq_{\mathbf{db}}$ (which depends on $\mathbf{db}$) on the set of all consistent database instances. The intended informal meaning is that for all consistent database instances $\mathbf{r}_1$ and $\mathbf{r}_2$, we have $\mathbf{r}_1\leq_{\mathbf{db}}\mathbf{r}_2$ if transforming $\mathbf{db}$ into $\mathbf{r}_1$ requires not more effort than transforming $\mathbf{db}$ into $\mathbf{r}_2$. We define the strict version of $\leq_{\mathbf{db}}$, denoted $<_{\mathbf{db}}$, as follows: $\mathbf{r}_1<_{\mathbf{db}}\mathbf{r}_2\triangleq\mathbf{r}_1\leq_{\mathbf{db}}\mathbf{r}_2$ and not $\mathbf{r}_2\leq_{\mathbf{db}}\mathbf{r}_1$. The principle that repairs must be obtained by some minimal change can now be made formal: a *repair* of $\mathbf{db}$ is a consistent database instance $\mathbf{r}$ such that there exists no consistent database instance $\mathbf{r}'$ satisfying $\mathbf{r}'<_{\mathbf{db}}\mathbf{r}$. Some repairs are of a special sort: a repair of $\mathbf{db}$ is called a *subset-repair* if it is included in $\mathbf{db}$, and is called a *superset-repair* if it includes $\mathbf{db}$.

To guarantee the existence of repairs, some additional properties should be imposed on $\leq_{\mathbf{db}}$ (or on $<_{\mathbf{db}}$). A very common and sufficient requirement for the existence of repairs is the acyclicity of $<_{\mathbf{db}}$.

In the CQA literature, the binary relation $\leq_{\mathbf{db}}$ (or $<_{\mathbf{db}}$) is never explicitly given, but instead implicitly specified. Different specifications of $\leq_{\mathbf{db}}$ now lead to different repair notions; the two most common are the following:

- Let $\oplus$ denote the symmetric difference between sets. When we define $\mathbf{r}_1\leq_{\mathbf{db}}\mathbf{r}_2\triangleq(\mathbf{r}_1\oplus\mathbf{db})\subseteq(\mathbf{r}_2\oplus\mathbf{db})$, we obtain *symmetric-difference repairs*, also called *$\oplus$-repairs*. With this definition, $\leq_{\mathbf{db}}$ is a partial order. In terms of minimal change, symmetric-difference repairs minimize (with respect to $\subseteq$) the set of inserted and deleted facts.

- When we define $\mathbf{r}_1\leq_{\mathbf{db}}\mathbf{r}_2\triangleq|\mathbf{r}_1\oplus\mathbf{db}|\leq|\mathbf{r}_2\oplus\mathbf{db}|$, we obtain *cardinality repairs*, also called *C-repairs*. With this definition, $\leq_{\mathbf{db}}$ is reflexive and transitive.

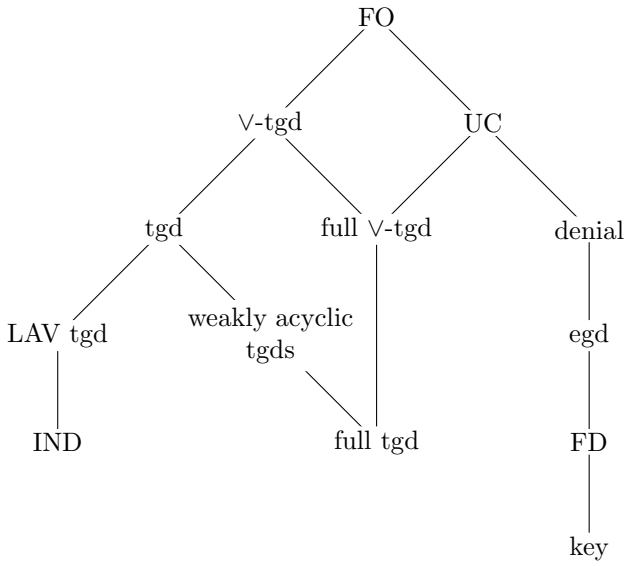

Figure 1: Common classes of integrity constraints. Adapted from [4].

In the literature on CQA, the notion of subset-repair is frequently introduced as an independent repair notion—one in which deletions are the only allowed repairing actions. We, instead, have defined it as a property: a $\oplus$-repair, cardinality repair, or any other repair of a database instance $\mathbf{db}$ is called a subset-repair if it is included in $\mathbf{db}$. Clearly, every maximal (with respect to $\subseteq$) consistent subset of a database instance $\mathbf{db}$ is a $\oplus$-repair of $\mathbf{db}$. A significant observation (see [1, Proposition 3]) is that for sets of denial constraints, every $\oplus$-repair is a subset-repair. This observation is no longer true for universal constraints, as illustrated next.

Example 1. *The database instance $\{R(a)\}$ has two $\oplus$-repairs with respect to $\Sigma = \{R(a) \rightarrow S(b)\}$: $\{\}$ and $\{R(a), S(a)\}$, which are a subset-repair and a superset-repair, respectively.*

In the definition of $\oplus$-repairs, the symmetric difference $\mathbf{r}_1 \oplus \mathbf{db}$ treats deletions (i.e., facts in $\mathbf{db} \setminus \mathbf{r}_1$) and insertions (i.e., facts in $\mathbf{r}_1 \setminus \mathbf{db}$) on equal footing. Alternatively, it is conceivable, although less common, to treat deletions and insertions asymmetrically. In some situations, for example, inconsistency may be primarily attributed to missing facts rather than to erroneously stored facts. In such situations, one may want to keep deletions to a minimum, which can be achieved by defining $\mathbf{r}_1 \leq_{\mathbf{db}} \mathbf{r}_2 \triangleq$ either $(\mathbf{r}_1 \cap \mathbf{db}) \supsetneq (\mathbf{r}_2 \cap \mathbf{db})$ or both $(\mathbf{r}_1 \cap \mathbf{db}) = (\mathbf{r}_2 \cap \mathbf{db})$ and $(\mathbf{r}_1 \oplus \mathbf{db}) \subseteq (\mathbf{r}_2 \oplus \mathbf{db})$. With this definition, the repair $\{R(a), S(a)\}$ would become the only repair in Example 1, because it preserves the database fact $R(a)$. Many other repair notions have been proposed and studied over the past twenty years.

Prioritized repairs.

The framework introduced in [32] and further investigated in [16, 20, 24] deduces $<_{\mathbf{db}}$ from a preference relation on the set of database facts. For the sake of simplicity, let Σ be a set of functional dependencies. An *inconsistent prioritizing database instance* is a pair $(\mathbf{db}, \succ)$ where $\mathbf{db}$ is an inconsistent database instance, and $\succ$ is an acyclic binary relation on $\mathbf{db}$ such that if $f \succ g$, then $\{f, g\}$ falsifies Σ. The intended informal meaning of $f \succ g$ is that we prefer to keep f rather than g. Note that in the case of functional dependencies, conflicting facts always come in pairs. A relation $<_{\mathbf{db}}$ can be defined in terms of $\succ$, as follows: for all distinct consistent subsets $\mathbf{r}_1$ and $\mathbf{r}_2$ of $\mathbf{db}$, define $\mathbf{r}_1 <_{\mathbf{db}} \mathbf{r}_2$ if for every $g \in \mathbf{r}_2 \setminus \mathbf{r}_1$, there exists $f \in \mathbf{r}_1 \setminus \mathbf{r}_2$ such that $f \succ g$. Informally, $\mathbf{r}_1 <_{\mathbf{db}} \mathbf{r}_2$ states that $\mathbf{r}_1$ can be obtained from $\mathbf{r}_2$ by exchanging facts with more preferred facts. It can be verified that $<_{\mathbf{db}}$ is acyclic (by using that $\succ$ is acyclic). With this definition of $<_{\mathbf{db}}$, we obtain *g-repairs*: a consistent subset $\mathbf{r}$ of $\mathbf{db}$ is called a *globally optimal repair* (or *g-repair*) if there exists no consistent subset $\mathbf{r}'$ of $\mathbf{db}$ such that $\mathbf{r}' <_{\mathbf{db}} \mathbf{r}$.

EXAMPLE 2. *Take the following database instance:*

$$R \begin{array}{|cc|l} \hline 1 & 2 & \\ \hline a & b & (f_1) \\ c & b & (f_2) \\ c & d & (f_3) \\ \hline \end{array}.$$

Let $\Sigma = \{R : \{1\} \to \{2\}, R : \{2\} \to \{1\}\}$*, which expresses that neither column should contain duplicate values. The* $\oplus$*-repairs are:*

$$\begin{array}{|cc|} \hline 1 & 2 \\ \hline a & b \\ c & d \\ \hline \end{array} \quad \textit{and} \quad \begin{array}{|cc|} \hline 1 & 2 \\ \hline c & b \\ \hline \end{array}.$$

The left-hand relation is the only C-repair. If we assume $f_2 \succ f_1$ *and* $f_2 \succ f_3$*, then the right-hand relation is the only g-repair.*

4. COMPLEXITY OF CQA

We will restrict our attention to Boolean queries. Under this restriction, consistent query answering becomes a decision problem: given q, Σ, and **db**, does q evaluate to *true* on every repair of **db**? The input to this problem consists of three parts, and in the study of its complexity zero, one, or two parts can be fixed. In this paper, we take a *data complexity* perspective: we will fix both the query and the set of integrity constraints, and measure complexity with respect to the database instance. Alternative complexity analyses, which consider also queries and/or integrity constraints as part of the input, can be found in [4]. Thus, for every Boolean query q and set Σ of integrity constraints, we have the following problem:

CERTAINTY(q, Σ)

INSTANCE: A database instance **db**.

QUESTION: Does q evaluate to *true* on every repair of **db** with respect to Σ?

Of course, this definition is only meaningful when a repair notion has been established beforehand. We will write $\oplus$-CERTAINTY(q, Σ) if $\oplus$-repairs are intended. Furthermore, it is always understood that the database schema contains all relation names used in q or Σ.

We so far have defined consistent query answering for every individual pair q, Σ. It is also of interest to measure the complexity of consistent query answering for classes of queries and classes of integrity constraints. To this extent, let Q be a class of queries, and IC a class of integrity constraints. Let C be a complexity class.

- Consistent query answering for Q and IC is said to be *in* C if CERTAINTY(q, Σ) is in C for all $q \in$ Q and $\Sigma \subseteq$ IC.
- Consistent query answering for Q and IC is said to be C-*complete* if it is in C and, moreover, CERTAINTY(q, Σ) is C-complete for some $q \in$ Q and $\Sigma \subseteq$ IC.

For example, for symmetric-difference repairing, a correct claim is: *"Consistent query answering is* coNP*-complete for conjunctive queries and key dependencies."* Indeed, if q is a conjunctive query and Σ a set of key dependencies, then membership of $\oplus$-CERTAINTY(q, Σ) in coNP is straightforward: a succinct disqualification for a "no"-instance is any repair that falsifies q. coNP-completeness holds since it is known that $\oplus$-CERTAINTY(q_0, Σ_0) is coNP-hard for $q_0 = \exists x \exists y \exists z\,(R(x, y, z) \wedge S(z, x))$ and $\Sigma_0 = \{R : \{1, 2\} \to \{3\},\ S : \{1\} \to \{2\}\}$. For a reason that will become apparent in Section 5.2, note that q_0 is self-join-free and that Σ_0 has only one key dependency per relation.

Arming et al. in [4] have carried out a thorough study on the complexity of consistent query answering for conjunctive queries and $\oplus$-repairs; data complexity results for different classes of integrity constraints are shown in Table 1. Triangles indicate whether the lower ($\triangle$) or the upper ($\triangledown$) complexity bound is of interest. For example, for coNP-completeness, $\triangledown$ and $\triangle$ denote, respectively, membership in coNP and coNP-hardness. If a line contains no reference to the literature, then its complexity result follows from other lines in the table.

Table 1 conveys an unpleasant message: consistent query answering is intractable already for very common queries and integrity constraints. However, some nuance is needed: coNP-completeness of consistent query answering for Q and IC tells us that the set $\{\text{CERTAINTY}(q, \Sigma) \mid q \in \text{Q}, \Sigma \subseteq \text{IC}\}$ contains at least one intractable problem, but does not give us any hint about the boundary between tractable and intractable problems. It may still be the case that this set contains many tractable problems of practical interest. We will bring a more nuanced story in Section 5.

Descriptive Complexity of CQA.

Table 1 uses common *computational complexity* classes (P, coNP, Π_2^{P}). In the realm of (consistent) query answering, it may be more relevant to utilize *descriptive complexity*, which describes the complexity of the problem CERTAINTY(q, Σ) in some logic formalism, as explained next.

Notice first that every problem CERTAINTY(q, Σ) is actually a Boolean query (as introduced in the first paragraph of Section 2), mapping each database instance to either *true* or *false*. Let $\mathcal{L}$ be some logic language. We say that CERTAINTY(q, Σ) is

IC	$\oplus$-CERTAINTY(Σ, q)	Reference
FO	undecidable	
$\vee$-tgd	undecidable	
tgd	undecidable	[33, Theorem 7.2]
egds + weakly acyclic tgds	Π_2^{P}-complete $\triangledown$	[33, Theorem 6.3]
weakly acyclic tgds	Π_2^{P}-complete $\triangle$	[33, Theorem 6.3]
LAV tgd	in P	[33, Theorem 4.7]
IND	in P	
UC	Π_2^{P}-complete $\triangledown$ $\triangle$	[31, Lemma 4,Theorem 6]
full $\vee$-tgd	Π_2^{P}-complete $\triangle$	Modification of the $\triangle$ proof for UC [4]
full tgd	coNP-complete $\triangledown$ $\triangle$	[31][33, Theorem 5.5]
denial	coNP-complete $\triangledown$	[31]
egd	coNP-complete	
FD	coNP-complete	
key	coNP-complete $\triangle$	[12, Theorem 3.3]

Table 1: Data complexity results for consistent query answering, for conjunctive queries and $\oplus$-repairs.

expressible in $\mathcal{L}$ if there exists a formula Q in $\mathcal{L}$ such that the following are equivalent for every database instance **db**:

1. q is true on every repair of **db** with respect to Σ;
2. Q is true on **db**.

Such a formula Q, if it exists, is called a *consistent $\mathcal{L}$-rewriting of q* (with respect to Σ). The practice of constructing Q is often referred to as "query rewriting."

From a database perspective, the most attractive candidate for $\mathcal{L}$ is probably first-order predicate calculus, denoted FO. Indeed, if we can construct a consistent FO-rewriting of q with respect to Σ, then the problem CERTAINTY(q, Σ) can be solved by a single SQL query on existing RDBMS engines. Another good candidate for query rewriting is Datalog with stratified negation, whose data complexity is in P (and is complete for P). For the higher complexities in Table 1, more expressive logics are needed, such as variants of disjunctive Datalog [3, 19]. In general, by studying the descriptive complexity of problems CERTAINTY(q, Σ), we get a handle on the database languages that can be used to solve them.

Integrity Constraints from Different Classes.

In database applications, it is normal to have integrity constraints belonging to different classes, for example, a combination of inclusion and functional dependencies. Table 2 is somewhat unsatisfactory insofar as it does not consider unions of common classes of integrity constraints, except for unions of a set of egds and a weakly acyclic set of tgds. Note incidentally that in Fig. 1, the smallest class that includes both inclusion and functional dependencies is the class of all first-order logic constraints.

5. PRIMARY KEYS

The notion of primary key is fundamental and ubiquitous in relational database systems. Its study in the context of CQA has therefore attracted considerable attention and has revealed some interesting connections with other fields. This section shows that a nuanced complexity landscape hides behind the last line of Table 1, and discusses connections to CSP and probabilistic database systems.

5.1 Some Terminology

Recall that a key dependency is a functional dependency $R : X \to Y$ such that $X \cup Y$ contains every positive integer up to the arity of R. A database instance satisfies this key dependency if and only if it does not contain two distinct facts that agree on all positions in X. By a *set of primary keys*, we mean a set of key dependencies containing exactly one key dependency for each relation name in the database schema under consideration. If such a set of primary keys contains $R : X \to Y$, then X is said to be the *primary key* of R. For the sake of simplicity, it is commonly assumed that the primary key X of R is not empty and formed by the $|X|$ leftmost positions (i.e., $X = \{1, 2, \ldots, |X|\}$ with $|X| \geq 1$).

A natural way for specifying repairs with respect to primary keys goes as follows. We say that two database facts are *key-equal* if they share the same relation name and agree on the primary key of their shared relation name. Given a database instance

db, the binary relation *"is key-equal to"* is obviously an equivalence relation on **db**; its equivalence classes are called the *blocks* of **db**. Obviously, **db** is consistent if and only if none of its blocks contains two or more database facts. Every *repair* of **db** is obtained by picking exactly one database fact from each of its blocks. The repairs defined in this way are both $\oplus$-repairs and C-repairs, and moreover are subset-repairs. In this section, we do not consider other conceivable ways of fixing primary key violations, such as replacing a duplicate primary key value with a fresh value.

The notion of block also occurs in probabilistic databases for modeling uncertainty: only one database fact of a block can be true, but we do not know which one holds true.

5.2 A Complexity Trichotomy

In Section 4, we explained what it means that consistent query answering is coNP-complete for conjunctive queries and primary keys. This is a rough result because it does not say anything about the boundary between tractable and intractable problems. The following trichotomy theorem from [21, 23] offers a more fine-grained complexity classification, albeit only for queries that are self-join-free.

THEOREM 1. *For every set Σ of primary keys and self-join-free Boolean conjunctive query q, the problem* CERTAINTY(q, Σ) *is in* FO, L*-complete, or* coNP*-complete. Moreover, it is decidable which of the three cases applies.*

The proof of Theorem 1 also yields a significant result in descriptive complexity, because it shows membership in L by expressing CERTAINTY(q, Σ) in symmetric stratified Datalog with some aggregation operator. Another promising observation in [23] is that tractability in L or FO obtains, roughly speaking, when all joins are foreign-to-primary key joins, which is undoubtedly the most common kind of join. In conclusion, for primary keys and self-join-free conjunctive queries, the most common cases of consistent query answering happen to be tractable, and all tractable cases can be solved by query rewriting in stratified Datalog with some aggregation operator.

For illustration, compare the following queries:

$$q_0 = \exists x \exists y \exists z \left(S(\underline{z}, x) \wedge R(\underline{x}, y, z) \right),$$
$$q_1 = \exists x \exists y \exists z \left(S(\underline{z}, x) \wedge T(\underline{x}, z) \right),$$

where primary keys are underlined. It is known that consistent query answering is coNP-complete for q_0 (see Section 4), and in L (but not in FO) for q_1. This difference in complexity between q_0 and q_1 occurs because $S(\underline{z}, x)$ uses *all* primary key variables of the other atom in q_1, but does not so in q_0 (in particular, y does not occur in $S(\underline{z}, x)$). The join in q_0 is therefore not a foreign-to-primary key join.

An obvious open question is how to extend Theorem 1 to conjunctive queries with self-joins; we will have more to say about this in Section 5.3. A different partial extension of Theorem 1 appears in [22]: membership of CERTAINTY(q, Σ) in FO remains decidable if q is a self-join-free Boolean conjunctive query with clique-guarded negated atoms. Negation is called clique-guarded if whenever some variables x and y occur together in a negated atom, they also occur together in some non-negated atom.

For a reason that will become apparent in the next subsection, we state a P-coNP-complete dichotomy that immediately follows from Theorem 1:

COROLLARY 1. *For every set Σ of primary keys and self-join-free Boolean conjunctive query q, the problem* CERTAINTY(q, Σ) *is either in* P *or* coNP*-complete.*

5.3 Connections with CSP

A famous open conjecture is the following.

CONJECTURE 1. *For every set Σ of primary keys, for every query q that is a disjunction of Boolean conjunctive queries,* CERTAINTY(q, Σ) *is either in* P *or* coNP*-complete.*

Conjecture 1 generalizes Corollary 1 to unions of conjunctive queries, possibly with self-joins. The proof of Theorem 1 uses a predominantly logic approach, and it may be tempting to attack Conjecture 1 by the same logic apparatus. However, for reasons explained in the next paragraph, such a line of attack is unlikely to lead to success.

In her article with the questioning title *"Why Is It Hard to Obtain a Dichotomy for Consistent Query Answering?"* [18], Fontaine establishes connections between computational complexities in CQA and constraint satisfaction problems (CSPs). These connections were further investigated in [26]. One of Fontaine's findings is that a proof of Conjecture 1 would imply Bulatov's dichotomy theorem for conservative CSPs. The three published proofs [5, 7, 8] of the latter theorem use an algebraic approach developed over many years. Therefore, it is hardly conceivable that Conjecture 1 can be settled by the logic approach developed for Theorem 1. A different way to attack Conjecture 1 would be to show that it is implied by the recently proved CSP dichotomy theorem [9, 35].

5.4 Counts and Probabilities

For a set Σ of primary keys, the number of repairs of a given database instance is finite and can easily be calculated. Rather than asking whether all repairs satisfy a Boolean query q, a computationally more difficult problem is to determine how many repairs satisfy q. It has been shown [28] that for every set Σ of primary keys and self-join-free Boolean conjunctive query q, the following problem is either in FP or ♯P-complete: given a database instance **db**, determine the number of repairs of **db** that satisfy q. It is an open conjecture that this dichotomy result remains true over the class of all Boolean conjunctive queries. When all primary keys are singletons, this conjecture has been shown to be true [29, 34]. In these results, ♯P-hardness is with respect to polynomial-time Turing reductions; Calautti et al. [10] have recently studied the consequences of using many-one logspace reductions instead, which are a weaker form of reduction.

We close this section by pointing out an intimate relationship between the counting variant of CERTAINTY(q, Σ) and query answering in probabilistic databases. Assume a probability distribution over the set of all repairs of some database instance **db**. The probability of a Boolean query q, denoted $Pr(q)$, is then defined to be the sum of the probabilities of the repairs that satisfy the query. A common assumption is that facts of distinct blocks are independent, i.e., if $A_1, A_2, \ldots, A_k$ are facts belonging to k distinct blocks, then $Pr\left(\bigwedge_{i=1}^{k} A_i\right) = \prod_{i=1}^{k} Pr(A_i)$. Notice here that (conjunctions of) atomic facts are Boolean queries, for which we have defined Pr. If this independence assumption holds true, then the probability distribution over the set of all repairs is fully determined if we know the marginal probability $Pr(A)$ for each fact A in **db**. These probabilities can then be listed as illustrated in the following table; it is also common to underline primary keys, and to separate blocks by dashed lines.

S	S#	SNAME	STATUS	CITY	*Pr*
	S1	Smith	20	London	1
	S2	Jones	10	Paris	0.7
	S2	Jones	15	Paris	0.3

Then, for a fixed Boolean query q, there is a natural shift from counting to calculating probabilities: given a database instance and the marginal probabilities of its facts, determine $Pr(q)$.

The probabilistic data model just described is almost the same as the *block-independent disjoint* (BID) probabilistic data model [13]. The only difference is that in BID probabilistic database instances, the marginal probabilities of the facts of a same block need not sum up to one. This difference emerges because *possible worlds* in the BID probabilistic data model are merely restricted not to contain two distinct facts from a same block, which leaves the possibility of selecting no fact from a block. Repairs, on the other hand, must contain exactly one fact from each block. A complexity dichotomy similar to the one previously cited holds: for every self-join-free Boolean conjunctive query q, the data complexity of evaluating q on BID probabilistic database instances is either in FP or ♯P-hard [14].

6. REPAIR CHECKING

Assume a repair notion has been fixed, for example, symmetric-difference repairing. Repair checking, then, is the following decision problem: given a set Σ of integrity constraints and two database instances **db** and **r**, determine whether **r** is a repair of **db**. Since this paper's focus is on data complexity, we define this problem for any fixed set Σ:

REPAIR-CHECKING(Σ)

INSTANCE: Database instances **db** and **r**.

QUESTION: Is **r** a repair of **db** with respect to Σ?

We will write $\oplus$-REPAIR-CHECKING(Σ) when we assume $\oplus$-repairs.

Repair checking is relevant to (the complexity of) consistent query answering: a method for solving the complement of CERTAINTY(q, Σ), with **db** as input, consists in non-deterministically guessing a database instance **r**, and checking whether **r** falsifies q and whether the pair **db**, **r** is a "yes"-instance of REPAIR-CHECKING(Σ). This method is effective when the size of **r** can be polynomially bounded in the size of **db**, as is the case for $\oplus$-repairs with respect to denial constraints and full tgds.

The complexity of repair checking for a class IC of integrity constraints can be expressed in the following terms, where C denotes a complexity class:

- Repair checking for IC is said to be *in* C if REPAIR-CHECKING(Σ) is in C for all $\Sigma \subseteq$ IC.
- Repair checking for IC is said to be C-*complete* if it is in C and, moreover, for some $\Sigma \subseteq$ IC, REPAIR-CHECKING(Σ) is C-complete.

Arming et al. in [4] have carried out a thorough study on the complexity of $\oplus$-repair checking; data complexity results are shown in Table 2. Results on C-repair checking appear in [1, 25]. A similar caveat

IC	$\oplus$-REPAIR-CHECKING(Σ)	Reference
FO	coNP-complete $\triangledown$	[1, Proposition 4]
$\vee$-tgd	coNP-complete	
tgd	coNP-complete	
weakly acyclic tgds	coNP-complete $\triangle$	[1, Theorem 7]
LAV tgd	in P	[33, Theorem 4.9]
weakly acyclic LAV tgds	in L	[1, Theorem 3]
IND	in P	
UC	coNP-complete $\triangledown$ $\triangle$	[31, Lemma 4, Corollary 3]
full $\vee$-tgd	coNP-complete $\triangle$	Modification of the $\triangle$ proof for UC [4]
full tgd	P-complete $\triangledown$ $\triangle$	[30, Theorem 3.7][1, Theorem 5]
denial	in L	[1, Proposition 5]
egd	in L	
FD	in L	
key	in L	

Table 2: Data complexity results for $\oplus$-repair checking.

as in Section 4 is in order here: coNP-completeness of repair checking for IC tells us nothing about the boundary between tractable and intractable problems in the set $\{\text{REPAIR-CHECKING}(\Sigma) \mid \Sigma \subseteq \text{IC}\}$.

For g-repairs (see Section 3 for the definition of g-repairs), it follows from Proposition 5 and Theorem 2 in [32] that g-repair checking is coNP-complete for functional dependencies. Note that in the case of g-repairs, the input to REPAIR-CHECKING(Σ) also contains the binary preference relation $\succ$ on **db**. A more fine-grained complexity classification for g-repair checking appears in [16], where it is shown that for every set Σ of functional dependencies, the problem REPAIR-CHECKING(Σ) is either in P or coNP-complete, and it is decidable which of the two cases applies. More complexity results on variants of g-repair checking appear in [20].

Repair Counting.

Livshits and Kimelfeld in [24] study the problem of counting database repairs. They show, among others, that for every set of functional dependencies, the data complexity of the following problem is either in FP or $\sharp$P-complete: given a database instance **db**, determine the number of $\oplus$-repairs of **db**. Remind here that for functional dependencies, all $\oplus$-repairs are subset-repairs.

Integrity Constraints from Different Classes.

Few complexity studies in CQA consider combining integrity constraints from different classes, which is nevertheless most common in practice. The following two results show that such combinations can entail an increase in the data complexity of repair checking. First, from Theorem 4.6 in [12], it follows that $\oplus$-repair checking for INDs and FDs taken together is coNP-complete. This is to be contrasted with the tractable complexities for IND and FD in Table 2. Second, by [1, Theorem 8], there are a weakly acyclic set Σ_1 of LAV tgds and a set Σ_2 of egds such that $\oplus$-REPAIR-CHECKING($\Sigma_1 \cup \Sigma_2$) is coNP-complete, whereas Table 2 shows that for $i \in \{1, 2\}$, $\oplus$-REPAIR-CHECKING(Σ_i) is tractable.

7. CONCLUDING THOUGHTS

Consistent query answering has been studied for all common classes of integrity constraints. Tables 1 and 2 show some of the rich body of theoretical results obtained over the past twenty years. These results, however, are still open to refinement. For example, it has been known since the early years that consistent query answering is coNP-complete for conjunctive queries and key dependencies, as reported by the last line of Table 1. But still today it remains an open conjecture that for every Boolean conjunctive query q and set Σ of key dependencies, the problem CERTAINTY(q, Σ) can be classified as either coNP-complete or in P. It took until recently to show this conjecture under the serious restrictions of self-join-free queries and primary keys. For more expressive queries and integrity constraints, such detailed complexity classifications are missing.

We conclude with a reflection on the notion of repairing. In Section 3.2, we have modeled database repairing by an acyclic binary relation $\leq_{\mathbf{db}}$ on the set of consistent database instances, where the intuition behind $\mathbf{r}_1 \leq_{\mathbf{db}} \mathbf{r}_2$ is that $\mathbf{r}_1$ is at least as close to **db** as $\mathbf{r}_2$. The repairs, then, are the consistent database instances that are most close to **db**. The relation $\leq_{\mathbf{db}}$ is never explicitly given, but rather implicitly specified by using some embodiment of the principle of minimal change, nearly always in

a manner that is agnostic of the meaning of the data. We believe that it would be worthwhile to explore capabilities for further restricting "the legal" repairs—in the same way as integrity constraints restrict "the legal" databases. Such a capability is already provided, for example, by the preference relation on database facts in prioritized repairs (see Section 3). But this preference relation, rather than being given explicitly, could be specified implicitly in some declarative fashion, capturing more of the meaning of the data. Proposals for such formalism can be found in [11, 27].

8. REFERENCES

[1] F. N. Afrati and P. G. Kolaitis. Repair checking in inconsistent databases: Algorithms and complexity. In *ICDT*, pages 31–41, 2009.

[2] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79, 1999.

[3] M. Arenas, L. E. Bertossi, and J. Chomicki. Answer sets for consistent query answering in inconsistent databases. *TPLP*, 3(4-5):393–424, 2003.

[4] S. Arming, R. Pichler, and E. Sallinger. Complexity of repair checking and consistent query answering. In *ICDT*, pages 21:1–21:18, 2016.

[5] L. Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *LICS*, pages 301–310, 2011.

[6] L. E. Bertossi. Database repairs and consistent query answering: Origins and further developments. In *PODS*, pages 48–58, 2019.

[7] A. A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Log.*, 12(4):24:1–24:66, 2011.

[8] A. A. Bulatov. Conservative constraint satisfaction re-revisited. *J. Comput. Syst. Sci.*, 82(2):347–356, 2016.

[9] A. A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *FOCS*, pages 319–330, 2017.

[10] M. Calautti, M. Console, and A. Pieris. Counting database repairs under primary keys revisited. In *PODS*, pages 104–118, 2019.

[11] L. Caroprese, S. Greco, and E. Zumpano. Active integrity constraints for database consistency maintenance. *IEEE Trans. Knowl. Data Eng.*, 21(7):1042–1058, 2009.

[12] J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.

[13] N. N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: Diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009.

[14] N. N. Dalvi, C. Ré, and D. Suciu. Queries and materialized views on probabilistic databases. *J. Comput. Syst. Sci.*, 77(3):473–490, 2011.

[15] C. J. Date. *An introduction to database systems (7. ed.)*. Addison-Wesley-Longman, 2000.

[16] R. Fagin, B. Kimelfeld, and P. G. Kolaitis. Dichotomies in the complexity of preferred repairs. In *PODS*, pages 3–15, 2015.

[17] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[18] G. Fontaine. Why is it hard to obtain a dichotomy for consistent query answering? *ACM Trans. Comput. Log.*, 16(1):7:1–7:24, 2015.

[19] G. Greco, S. Greco, and E. Zumpano. A logical framework for querying and repairing inconsistent databases. *IEEE Trans. Knowl. Data Eng.*, 15(6):1389–1408, 2003.

[20] B. Kimelfeld, E. Livshits, and L. Peterfreund. Detecting ambiguity in prioritized database repairing. In *ICDT*, pages 17:1–17:20, 2017.

[21] P. Koutris and J. Wijsen. Consistent query answering for self-join-free conjunctive queries under primary key constraints. *ACM Trans. Database Syst.*, 42(2):9:1–9:45, 2017.

[22] P. Koutris and J. Wijsen. Consistent query answering for primary keys and conjunctive queries with negated atoms. In *PODS*, pages 209–224, 2018.

[23] P. Koutris and J. Wijsen. Consistent query answering for primary keys in logspace. In *ICDT*, pages 23:1–23:19, 2019.

[24] E. Livshits and B. Kimelfeld. Counting and enumerating (preferred) database repairs. In *PODS*, pages 289–301, 2017.

[25] A. Lopatenko and L. E. Bertossi. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *ICDT*, pages 179–193, 2007.

[26] C. Lutz and F. Wolter. On the relationship between consistent query answering and constraint satisfaction problems. In *ICDT*, pages 363–379, 2015.

[27] M. V. Martinez, F. Parisi, A. Pugliese, G. I. Simari, and V. S. Subrahmanian. Policy-based

inconsistency management in relational databases. *Int. J. Approx. Reasoning*, 55(2):501–528, 2014.

[28] D. Maslowski and J. Wijsen. A dichotomy in the complexity of counting database repairs. *J. Comput. Syst. Sci.*, 79(6):958–983, 2013.

[29] D. Maslowski and J. Wijsen. Counting database repairs that satisfy conjunctive queries with self-joins. In *ICDT*, pages 155–164, 2014.

[30] S. Staworko. *Declarative Inconsistency Handling in Relational and Semi-Structured Databases*. PhD thesis, State University of New York at Buffalo, 2007.

[31] S. Staworko and J. Chomicki. Consistent query answers in the presence of universal constraints. *Inf. Syst.*, 35(1):1–22, 2010.

[32] S. Staworko, J. Chomicki, and J. Marcinkowski. Prioritized repairing and consistent query answering in relational databases. *Ann. Math. Artif. Intell.*, 64(2-3):209–246, 2012.

[33] B. ten Cate, G. Fontaine, and P. G. Kolaitis. On the data complexity of consistent query answering. *Theory Comput. Syst.*, 57(4):843–891, 2015.

[34] J. Wijsen. Corrigendum to "Counting database repairs that satisfy conjunctive queries with self-joins". *CoRR*, abs/1903.12469, 2019.

[35] D. Zhuk. A proof of CSP dichotomy conjecture. In *FOCS*, pages 331–342, 2017.

Wilkinson's Tests and SQL Packages

B. D. McCullough
Decision Sciences & MIS
Drexel University
Philadelphia, PA USA
bdmccullough@drexel.edu

Taha Mokfi
Independent Researcher
Connecticut, USA
mokfi.taha@gmail.com

Mahsa Almaeenjad
Independent Researcher
Connecticut, USA
almaee.mahsa@gmail.com

ABSTRACT

Wilkinson's Tests are used to benchmark the accuracy of some statistical functions in six SQL packages: Apache Hive, Microsoft Access, Microsoft SQL Server, MySQL, Oracle 11g SQL, and SAP Hana. Using the best choice of data type, we find that different packages use different rounding schemes, two packages use unreliable algorithms to compute the sample variance, one package returns the population standard deviation when the sample standard deviation is called, and one package has an unstable algorithm for computing the correlation coefficient. Using the wrong data type all but guarantees inaccurate results.

1. INTRODUCTION

A database is an organized collection of data. A database is managed by a database management system (DBMS). The popularity of DBMSs increased with the advent of relational database management systems (RDBMS). Query operations on tables were introduced through the Structured Query Language (SQL). Even though the SQL syntax was standardized, many vendors developed SQLs on their own. Hence, some of the standard functionality might not be the same and the underlying code might also make a difference. This is especially important since many authors advocate the use of SQL for data mining activities: Wei et al [12], Trueblood and Lovett [20], Linoff [10], Alexander [1], Celko [4], Fotache and Strimbel [7], Ordonez and Pitchaimalai [16], and Pearson and Mackey [17]. None of these authors warns the reader that SQL software might be inaccurate for statistical purposes.

In particular, different SQL packages might give different answers to the same problem. Indeed, the situation is even more problematic: the same package can give different answers to the same problem! Recently Niranjan and Nandi [15] found errors in various SQL packages in the implementation of the calculation of sample variance; not all packages could accurately compute the sample variance. We extend their work with an eye toward elementary statistical calculations using a collection of benchmark tests known as "Wilkinson's Tests" based on Wilkinson's [21] *Statistics Quiz* that presents six suites of tests: reading an ASCII file; real numbers; missing data; regression; analysis of variance; and operating on a database. Wilkinson's Tests have a long track record of being applied to statistical packages by, among others, Wilkinson [22], Sawitzki [18], Bankhofer and Hilbert [3], McCullough [13], Choi and Kiefer[6], Lomax [11], Keeling and Pavur [8], and McCullough and Yalta [14].

While *Statistics Quiz* offers six suites of tests, for present purposes only the "Real Numbers," "Missing Data," and "Regression" suites are relevant. All the applicable tests are applied to the most recent versions of RDBMS packages. We have tested:

- Hive version 1.2.1
- MS Access 2016 v16.0 (64 bit)
- Microsoft SQL Server Management Studio v12.0 (henceforth, MS SQL)
- MySQL workbench v6.3.6 buit 511 CE (64 bit)
- Oracle Database Express edition 11g
- SAP Hana version 2.0 [Express Edition]

All programs were run on a laptop wtih Intel(R) Core(TM) i5-5200U CPU 2.2 RAM: 8 GB System with a 64 bit Windows 10 Operating System.

To date, we were unable to find any research assessing the accuracy of statistical functionality in SQL packages. This is important, because SQL is frequently advocated for both statistical analysis of data and data mining:

2. THE DATA

Table 1 below displays the data set "Nasty". The values for BIG are about the size of the population of Egypt, while the values of HUGE are the same order of magnitude as the American federal budget deficit.

LABEL$	X	ZERO	MISS	BIG
ONE	1	0	.	99999991
TWO	2	0	.	99999992
THREE	3	0	.	99999993
FOUR	4	0	.	99999994
FIVE	5	0	.	99999995
SIX	6	0	.	99999996
SEVEN	7	0	.	99999997
EIGHT	8	0	.	99999998
NINE	9	0	.	99999999

LITTLE	HUGE	TINY	ROUND
0.99999991	1.0E12	1.0E-12	0.5
0.99999992	2.0E12	2.0E-12	1.5
0.99999993	3.0E12	3.0E-12	2.5
0.99999994	4.0E12	4.0E-12	3.5
0.99999995	5.0E12	5.0E-12	4.5
0.99999996	6.0E12	6.0E-12	5.5
0.99999997	7.0E12	7.0E-12	6.5
0.99999998	8.0E12	8.0E-12	7.5
0.99999999	9.0E12	9.0E-12	8.5

Table 1: Data Set NASTY.DAT

Immediately we encountered difficulties reading the data. For SQL and MySQL we tried to import the data from a .csv file but both packages read all the columns as character when they should be numeric, so we manually changed the type of all columns to be double precision. In contrast, Oracle could import data from csv with numeric types. In general, after reading in the data for each package, we had to set the type for each column. This turned out to be a critical step. One might think that NUMERIC, BINARY_FLOAT and BINARY_DOUBLE would give similar answers, but such is not the case. For example, when computing the standard deviation of the variables, to four decimals all answers should be 2.7386 raised to some power of ten, with the exceptions of ZERO which should be 0 and MISS which should be missing. Yet, Table 2 shows some results from Oracle for various data types.

Observe that the first column, NUMERIC, gives correct answers. In contrast, BINARY_FLOAT and BINARY_DOUBLE correctly compute the sample standard deviation for X, HUGE, TINY and ROUND, but fail for other variables. Oracle was not alone in exhibiting this type of behavior. Specifically why Oracle and other packages erratically compute the standard deviation for the BINARY_FLOAT and BINARY_DOUBLE types is beyond the scope of this paper, but the point is that specifying the best data type is critical. Therefore we ran all the tests for all the data types for all the packages and only used the best choice.

variable	NUMERIC	BINARY_ FLOAT	BINARY_ DOUBLE
X	2.7386	2.7386	2.7386
ZERO	0	0	0
MISS	–	–	–
BIG	2.7386	0	2.4494
LITTLE	2.7386E-08	0	1.4901 E-08
HUGE	2.7386E12	2.7386E+12	2.7386E12
TINY	2.7386E-12	2.7386E-12	2.7386E-12
ROUND	2.7386	2.7386	2.7386

Table 2: Oracle Std. Dev. Results for Various Data Types

The failures of MS SQL are presented in Table 3. None of the types provides correct answers for all the variables. See that the standard deviation of LITTLE can take on three different values, none of which is correct. REAL gives the same answers as SINGLE, and so is omitted from the table.

variable	NUMERIC	DECIMAL	SINGLE	DOUBLE
X	2.7386	2.7386	2.7386	2.7386
ZERO	0	0	0	0
MISS	NULL	NULL	NULL	NULL
BIG	2.4495	2.4495	4.2426	2.4495
LITTLE	0	2.9802E-8	3.6500E-8	2.9802E-8
HUGE	2.7386E12	2.7386E12	2.7386E12	2.7386E12
TINY	0	2.7386E-12	2.7386E-12	2.7386E-12
ROUND	2.7386	2.7386	2.7386	2.7386

Table 3: MS SQL Std. Dev. Results for Various Data Types

MS SQL server supports different numeric data types. The type DECIMAL has fixed precision and scale and takes two arguments: precision (Maximum total number of decimal digits which is between 1 to 38) and scale (The number of decimal digits that will be stored to the right of the decimal point which is between 0 and precision). FLOAT and DOUBLE types are approximate and have only one argument n (precision and storage size which can be between 1 and 53). A precision from 0 to 23 results in a 4-byte single-precision FLOAT column. A precision from 24 to 53 results in an 8-byte double-precision DOUBLE column. All the other types such as: INT, BIGINT, SMALLINT, and TINYINT are exact-number data types that use integer data depending on the data points. For MS SQL, DECIMAL is specified as DECIMAL(38,12), SINGLE as FLOAT(24), and DOUBLE as FLOAT(53). In MySQL, NUMERIC is implemented as DECIMAL, so the following remarks about DECIMAL

apply equally to NUMERIC.

For the six packages, the options and our best choices for data type are given in Table 4 (we exclude integer types), where an asterisk denotes that the data type could not correctly compute the standard deviation. The prevalence of asterisks shows that whether a package can correctly compute the standard deviation depends on the data type specified.

package	variable types	most accurate
Hive	FLOAT*, DOUBLE, DECIMAL	DOUBLE
Access	SINGLE*, DOUBLE*	DOUBLE
MS SQL	NUMERIC*, DECIMAL*, SINGLE*, DOUBLE*, REAL*	FLOAT(53)
MySQL	NUMERIC*, DECIMAL, FLOAT*, DOUBLE	DOUBLE
Oracle	NUMERIC, BINARY_FLOAT*, BINARY_DOUBLE*	NUMERIC
Hana	FLOAT, REAL*, DOUBLE, DECIMAL	DOUBLE or FLOAT

Table 4: Options and best data type for each package

3. THE TESTS

3.1 Real Numbers

3.1.1 Test II-A: Print ROUND to one digit.

The program should follow IEEE-754, which specifies "round to even", so the correct answer is $0, 2, 2, 4, 4, 6, 6, 8, 8$. For all packages, we used the command: Round(X,1) and results are presented in Table 5.

package	result	package	result
Apache Hive	fail	MySQL	pass
MS Access	fail	Oracle 11g	fail
MS SQL	fail	SAP Hana	fail

Table 5: Results of Test II-A: print ROUND

Only MySQL passes this test. All other packages return $1, 2, 3, 4, 5, 6, 7, 8, 9$ instead of the correct answer, which indicates that they are not using IEEE-754 rounding. We see again that different SQL packages will return different answers to the same problem.

For an example of why correct rounding is important, consider this example from an SQL discussion board [2]:

> I am currently working on POC for migrating Oracle to SQL Server 2008 R2. Stuck with an issue related to rounding off calculations.
>
> There is this "rate" value we are calculating using POWER functions (it has nested POWER functions actually). I am getting mismatches because the values are not correctly round off during calculation. There are around 258 rows where the values are 0.01 less than the required value (ex. I get the rate value as 6.31 where actually it should be 6.32).
>
> In order to correct this issue, I changed the data types of the fields in the calculation from FLOAT to NUMERIC to correct out the precision. This is able to resolve the issue with the above 258 rows, but now I have another 61 rows where the value is 0.01 more than the required value (I get the rate value as 7.42 where actually it should be 7.41).
>
> All the fields in the Oracle calculation is using NUMBER datatype (without any precision), so cant really figure out what equivalent data type should be used in SQL Server. Currently I am using FLOAT datatype.
>
> Please advise if you have any pointers.

With different SQL packages using different rounding schemes, migrations are needlessly complicated and can produce unintended errors.

3.1.2 Test II-B: Plot HUGE against TINY and plot BIG against LITTLE in a scatter plot.

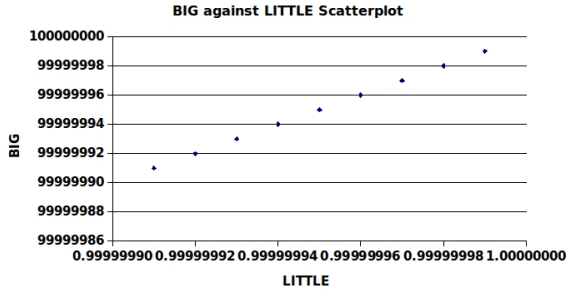

Figure 1: Test II-B Results for MS Access.

For each case the answer is a 45-degree straight line. MS ACCESS is the only package offering plotting capability. As Figure 1 shows, MS ACCESS passes this test; the other two graphs are correct and are omitted to conserve space.

	MS ACCESS
HUGE v. TINY	pass
BIG v. LITTLE	pass
X v. ZERO	pass

Table 6: Results of Test II-B

3.1.3 *Test II-C: Calculate the mean and standard deviation of each variable.*

The mean should be equal to the fifth value of each variable. Each package passes this test.

Standard deviations should be "undefined" or missing for MISS, zero for ZERO, and 2.738612788 (times 10 to some power) for all other variables (in the table the powers of ten are omitted). The relevant command for each package is: STDDEV_SAMP. The standard deviation results by different packages are listed in Table 7.

	Hive	ACCESS	MS SQL	MySQL	Oracle	Hana
X	2.582	p	p	p	p	p
ZERO	p	p	p	p	p	p
MISS	p	p	p	p	p	p
BIG	p	2.309	2.450	p	p	p
LITTLE	p	2.810	2.980	p	p	p
HUGE	p	p	p	p	p	p
TINY	p	p	p	p	p	p
ROUND	p	p	p	p	p	p

Table 7: Results of Test II-C – calculate the standard deviation (correct answers indicated by 'p' for 'pass')

Accurately computing the sample standard deviation is a solved problem, and there is no excuse for any software developer to use a bad algorithm. For a discussion of the various algorithms, see Ling [9] or Chan, Golub and Leveque [5]. Of course, if one uses the command STDDEV, one gets different answers depending on the package. STDDEV is supposed to be an alias for STDDEV_SAMP, but Hive returns 2.5820: in Hive, STDDEV is an alias for STDDEV_POP, so the documentation for STDDEV is wrong. Observe that MS SQL and MS Access both are using bad algorithms for computing the sample standard deviation, but apparently different bad algorithms! The same software company appears to be using two different algorithms, and neither of them is correct.

Consider performing a one-sample test of means on 29 observations. Let $H_0 : \mu = 11$ against $H_1 : \mu \neq 11$. Suppose the correct standard deviation is 2.7386; then the calculated t-statistic is 1.966 and the null hypothesis is just barely rejected. If the standard deviation were incorrectly calculated as 2.980 then the calculated t-statistic would be 1.81 and the null hypothesis would not be rejected. If the standard deviation were incorrectly calculated as 2.309 and many would incorrectly conclude that the null hypothesis had been even more strongly rejected.

	X	ZERO	BIG	LITTLE	HUGE	TINY	ROUND
X	1	NA	1	1	1	1	1
ZERO	NA	NA	NA	NA	NA	NA	NA
BIG	1	NA	1	1	1	1	1
LITTLE	1	NA	1	1	1	1	1
HUGE	1	NA	1	1	1	1	1
TINY	1	NA	1	1	1	1	1
ROUND	1	NA	1	1	1	1	1

Table 8: Correct answer for II-D, Correlation Matrix

3.1.4 *Test II-D: Compute the correlation between all the variables.*

The correlation coefficient is calculated as

$$\rho_{wz} = \frac{cov(w,z)}{\sigma_w \sigma_z} \tag{1}$$

where $cov(w,z)$ is the covariance between w and z and and σ_w is the standard deviation of w. Since the standard deviation of ZERO is zero, Equation 1 has zero in the denominator and so its correlation with any other variable is undefined. Only Oracle and Hive offer a correlation function. The correct answer is given in the top of Table 8.

For the data type FLOAT, in addition to sometimes giving the correct answer of 1.0, running HIVE with FLOAT also produces answers 0.614, 0.836 and 0.866 instead of 1.0 for some pairs of variables. For the preferred data type DOUBLE, correlating ROUND with X or ROUND produces NaN instead of the correct answer of 1.0.

Oracle, using the preferred data type NUMERIC, correctly calculates the correlation matrix. However, when using either BINARY_FLOAT or BINARY_DOUBLE it gives wildly incorrect answers.

The user-guides for these packages should warn that accuracy depends on choosing the correct data type, as the packages will not warn the user if a bad data type has been chosen.

3.1.5 *Test II-E: Plot X against X.*

In this test we try to plot the same variable with itself. The answer should be a graph with a 45 degree line.

Only MS Access lets us make a plot but it lets us select a variable only once and we cannot select it again. Therefore, we are unable to make this graph.

3.1.6 Test II-F: Regress BIG on X.

If the constant intercept is 99999990, then we can expect the slope to be 1. Only Oracle 11g offers regression, and it gives the correct answer. Oracle passes.

3.2 Missing Data

Missing values have been encountered commonly in all the areas. It could be due to bad data entry or simply no reading at all at that respective time. SQL queries do handle NULL or missing values quite efficiently and it might simply exclude the readings or include them in the results of queries. We however define handling missing values and simply excluding them in a different way.

3.2.1 Test III-A: MISSING in a conditional statement.

Implement the below pseudo-code in the package:

```
IF MISS = 3
THEN TEST = 1
ELSE TEST = 2
```

Ideally, the correct result should be equal to 2, since the MISS values do not have any value and definitely do not equal to 3. We use a select statement with the below syntax:

package	command
Hive	if(Miss = 3,1,2)
SQL/MySQL	Iif(Miss = 3,1,2)
Oracle	CASE WHEN Miss = 3 THEN 1 ELSE 2 END
MS Access	IIf(Miss=3,1,2)
SAP Hana	CASE WHEN Miss = 3 THEN 1 ELSE 2 END

All packages pass this test. In cases, where an IF..ELSE.. statement could not be used, we could use a similar functional statement. For example, we have used NVL2 function to test the IF.ELSE loop since it offers a similar functionality. All packages pass.

3.2.2 Test III-B: Is MISSING summable?

We apply the following test in this section

```
IF MISS = <missing> THEN MISS = MISS + 1
```

Ideally, the correct answer is missing, since we cannot add to any NULL value. The specific command used for each package is given in Table 9.

MS Access does not return a missing value indicator, but simply an empty table. This is an error.

package	command
SQL/MySQL	isnull(x, x+1)
Oracle	NVL(x, x+1)
MS Access	IIf(IsNull([miss]),[MISS]+1)
Hive	if(isnull(Miss), Miss +1, Miss)
Hana	ifnull(MISS,MISS+1)

Table 9: Commands for IF MISS... test

Hive	MS Access	MS SQL	MySQL	Oracle	Hana
pass	fail	pass	pass	pass	pass

Table 10: Results of Test III-B

3.3 Regression

Another very common computation is the computation of regression models. These models are useful in many ways and form the basis of data analysis. We can calculate the regression models in different ways by using the formula. Only Oracle offers functions to calculate the regression values and coefficients. It does not give us all the information about the regression model, but it gives us specifically what we might query. For example, the slope and intercept are displayed separately.

3.3.1 Test IV-A

In this test, we regress X on a constant term and powers of X. However, none of the packages offers multiple regression, so we cannot apply this test.

3.3.2 Test IV-B: Regress X with a constant and X.

The intercept should be zero and the slope should be one. Oracle passes this test.

3.3.3 Test IV-C: Regress X on a constant, BIG and LITTLE.

This is a multiple regression, and no package can perform this test.

3.3.4 Test IV-D: Regress ZERO vs a constant and X.

Because ZERO has no variance, the program should either fail to compute coefficients and report that ZERO has no variance, or it should return zero for both the intercept and the slope. Oracle returns zero for the intercept and the slope. Oracle passes this test.

4. CONCLUSIONS

Wilkinson's test are designed to uncover flaws in statistical function of software packages. In our analysis of six SQL packages, we have identified several flaws:

1. An incorrect choice of data type all but guarantees inaccurate statistical results; a correct choice does not guarantee correct results.

2. Different packages use different rounding schemes. Only MySQL adheres to IEEE-754 rounding; the others do not.

3. MS Access and MS SQL use unreliable algorithms to compute the sample variance.

4. Hive returns the population standard deviation instead of the sample standard deviation using STDDEV.

5. The Oracle correlation function is unstable. It gives a correct answer for the preferred data type and incorrect answers for other data types.

6. MS Access fails to correctly handle a missing value case.

Developers should fix these errors qiuckly and, until then, users should avoid them. Further, "accuracy of algorithms" should be incorporated into the SQL standards [19].

5. REFERENCES

[1] Michael Alexander. *Microsoft Access 2007 Data Analysis*. Wiley, 2007.

[2] anonymous. `https://social.msdn.microsoft.com/forums/sqlserver/en-US/9daa1b60-d11c-421b-8b87-e38a299e372c/rounding-off-issue-during-oracle-to-sql-server-migration`, 2013. Accessed: 2019-07-15.

[3] U. Bankhofer and A. Hilbert. Statistical software packages for windows: A market survey. *Statistical Papers*, 38:393–407, 1997.

[4] Joe Celko. *SQL for Smarties: Advanced SQL Programming, 5e*. Morgan Kaufman, 2015.

[5] T. F. Chan, Golub G. H. and R. J. Leveque. Algorithms for computing the sample variance: Analysis and recommendations. *American Statistician*, 37:242–247, 1983.

[6] Hwan-sik Choi and Nicholas Kiefer. Software evaluation: Easyreg international. *International Journal of Forecasting*, 21(3):609–616, 2005.

[7] Marin Fotache and Catalin Strimbel. Sql and data analysis. some implications for data analysis and higher education. *Procedia Economics and Finance*, 20:243–251, 2015.

[8] Kellie Keeling and Robert Pavur. Statistical accuracy of spreadsheet software. *The American Statistician*, 65(4):265–273, 2011.

[9] R. F. Ling. Comparison of several algorithms for computing sample means and variances. *Journal of the American Statistical Association*, 69:859866, 1974.

[10] Gordon S. Linoff. *Data analysis using SQL and Excel*. Wiley, 2015.

[11] Richard G. Lomax. Statistical accuracy of ipad applications: An initial examination. *The American Statistician*, 67(2):105–108, 2009.

[12] Wei Lu, Jiajia HouYing, YanMeihui, Zhang Xiaoyong, and Thomas Moscibroda. Msql: efficient similarity search in metric spaces using sql. *The VLDB Journal*, 26(3):829–854, 2017.

[13] B. D. McCullough. Wilkinson's tests and econometric software. *Journal of Economic and Social Measurement*, 29(1-3):261–270, 2004.

[14] B. D. McCullough and A. Talha Yalta. Spreadsheets in the cloud – not ready yet. *Journal of Statistical Software*, 52(7):1–14, 2013.

[15] Kamat Niranjan and Arnab Nandi. A closer look at variance implementations in modern database systems. *SIGMOD Record*, 45(4):28–33, 2016.

[16] C. Ordonez and S. K. Pitchaimalai. Bayesian classifiers programmed in sql. *IEEE Transactions on Knowledge and Data Engineering*, 22(1):139–144, 2010.

[17] William R. Pearson and Aaron J. Mackey. Using sql databases for sequence similarity searching and analysis. *Current Protocols in Bioinformatics*, 59(1):1–22, 2017.

[18] G. Sawitzki. Report on the numerical reliability of data analysis systems. *Computational Statistics and Data Analysis*, 18(2):289–301, 1994.

[19] Charles Severance. Elizabeth Fong: Creating the SQL Database Standards. *Computer*, 47(8):7–8, 2014.

[20] Robert P. Trueblood and Jr. John N. Lovett. *Data mining and statistical analysis using SQL*. Apress, 2001.

[21] Leland Wilkinson. *Statistics Quiz*. Systat Inc., 1985. `http://web.stanford.edu/~clint/bench/wilk.txt`.

[22] Leland Wilkinson. Practical guidelines for testing statistical software. In P. Dirschedl and R. Ostermann, editors, *Computational Statistics*. Physica-Verlag, Heidelberg, 1994.

Evaluation of an Implementation of Cross-Row Constraints Using Materialized Views

José María Cavero Barca
Rey Juan Carlos University
C/ Tulipán s/n
28933 Móstoles (Spain)
josemaria.cavero@urjc.es

Belén Vela Sánchez
Rey Juan Carlos University
C/ Tulipán s/n
28933 Móstoles (Spain)
belen.vela@urjc.es

Paloma Cáceres García de Marina
Rey Juan Carlos University
C/ Tulipán s/n
28933 Móstoles (Spain)
paloma.caceres@urjc.es

ABSTRACT

SQL assertions are a powerful means used to specify cross-row constraints, and have been available in the SQL standard since 1992. Unfortunately, assertions are not supported in commercial database management systems. Although triggers and application programs can be efficiently used to constrain database content, they are more complex to write and more error-prone. The objective of this paper is to analyze whether the use of materialized views could be a viable solution as regards the automatic implementation of SQL assertions. Materialized views are views that physically store the result of a query and are periodically updated. The method consists of defining a materialized view which contains the number of tuples that violate the condition expressed in the assertion. The materialized view will contain a CHECK constraint that guarantees that the number of tuples that violate the assertion is equal to zero. The proposed method is an easy and automatic means of implementing the integrity constraints described using assertions. We have carried out a series of tests, and although triggers perform better than materialized views in most situations, there are some in which materialized views would be an efficient option. They are easily automatable and less error prone than triggers.

1. INTRODUCTION

In relational databases, an integrity constraint is basically a Boolean expression that must be evaluated as TRUE [1]. A database integrity constraint, therefore, constrains the values that can appear in a given database.

The standard language for relational databases (Structured Query Language, SQL) provides several means to deal with integrity constraints [2]: table constraints, column constraints, domain constraints and assertions. The first three include UNIQUE, PRIMARY KEYS, FOREIGN KEYS and CHECKS, and are supported in most commercial Relational Database Management Systems (RDBMSs). The last (assertions) are the most general form of integrity constraint, and they are a simple and easy method by which to enforce cross-row constraints (that is, constraints across related rows, possibly in different tables). In short, their structure includes a condition that must be fulfilled. The RDBMS is responsible for ensuring that the condition is satisfied in every state of the database.

Unfortunately, although assertions have been part of the SQL standard since 1992 [3,4], commercial RDBMSs do not support assertions, and many cross-row integrity constraints are, therefore, usually implemented by means of triggers (which are used "to detect some conditions that happen in a database and then react to the database" [5]), or are included in the applications used to access the database. Recently, a few works related with constraints that affect collectively several tables have appeared. For example, in [6] the authors propose adding more functionality to core SQL by means of package queries to support constraints that the set of result rows to a query must satisfy. However, if a RDBMS supported assertions, it would be easy to control integrity constraints, thus eliminating the need to use triggers or application programs to control the integrity of the database or any other mechanism. While assertions only specify the condition that must be fulfilled to satisfy the constraint, triggers and application programs must be programmed by taking into account every possible situation that could violate the constraint.

In this paper we show an automatic implementation of assertions using materialized views. Materialized views are database objects that contain the result of a query at a specific time, are updated from time to time, and are used to increase the speed of queries in very large databases. As will be shown in the following sections, the implementation of the method requires a RDBMS that supports materialized views and that can include CHECK constraints, along with an automatic procedure that can be used to refresh the view. We have chosen the Oracle database [7] since we have prior expertise in using it, and because it supports all the conditions that must be satisfied in order to carry out the implementation and even some additional features. This approach has some basic advantages that

could make it useful in certain environments, although in many situations it performs worse than when using triggers.

The organization of the remainder of this paper is as follows. In Section 2, we summarize SQL assertions and the example that will be used in the rest of the paper. Section 3 focuses on materialized views and how they can be used to implement assertions. Section 4 shows the results of some tests in which triggers are compared with materialized views. Finally, Section 5 discusses the results and presents some conclusions.

2. SQL ASSERTIONS

In standard SQL, users can specify general constraints via the CREATE ASSERTION statement. An SQL assertion is "a CHECK constraint at the database level that is allowed to contain queries" [8]. One basic technique employed to write assertions is that of specifying a query that selects the tuples that violate the condition. By including this query in a NOT EXISTS clause, the assertion will specify that the query result must be empty. Any constraint that can be expressed in terms of a query that selects the tuples that violate the desired constraint could, therefore, be expressed using an assertion in the following way:

```
CREATE ASSERTION Assertion_Name
CHECK (NOT EXISTS (Query that selects
the tuples that violate the
constraint));
```

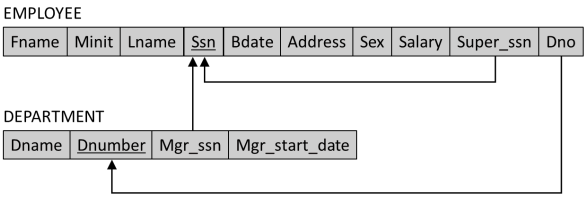

Fig. 1. Relational database schema.

In the remainder of the paper, we use a simplified version of an example regarding the storage of information concerning employees and departments from [9]. Every Employee belongs to one Department, and every Department has an Employee who works as a Manager. The schema is shown in Figure 1.

The following assertion constrains "that the salary of an employee must not be greater than the salary of the manager of the department that the employee works for":

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK (NOT EXISTS
       (SELECT *
       FROM EMPLOYEE E,
            EMPLOYEE M,
            DEPARTMENT D
       WHERE E.Salary > M.Salary
            AND E.Dno = D.Dnumber
            AND D.Mgr_ssn = M.Ssn));
```

This simple assertion guarantees that none of the employees violate the condition in any state of the database. The RDBMS should reject insertions, deletions and updates that make an employee's salary greater than that of the manager of his/her department.

None of the most frequently used commercial RDBMS support assertions, although some attempts have been made to build support for assertions, particularly in Oracle. For example, back in 2016, this possibility was considered in Oracle's Database Ideas [8]. Implementation alternatives were proposed, and complexity and performance problems that could appear were also discussed. It would, however, appear that the idea was eventually discarded owing to its complexity.

As mentioned previously, no commercial product supports assertions. A previous assertion is, therefore commonly implemented in the application programs, or by using a set of triggers that should be executed in the case of events that could provoke a violation of the constraint. A very simple assertion like that previously described could be implemented by, for example, using four triggers that will be fired when an employee is inserted or updated, when the manager of a department is updated, etc. Sometimes, a lot of situations have to be taken into account when programming triggers to support constraints, that is, sometimes, as stated in [10] "transforming integrity constraints into triggers for verifying database consistency produces a serious and complex problem". Moreover, "manually checking integrity constraint enforcement at the application level is usually difficult, as the code base to be examined could be large" [11].

Therefore, although triggers and application programs can be efficiently used to constrain database content, they are more complex to write and more error-prone. In the following section, we show how to implement constraints in a RDBMS by simulating assertions and using materialized views.

3. IMPLEMENTING ASSERTIONS USING MATERIALIZED VIEWS

An assertion checks that the condition that follows the keyword CHECK holds true for every database state. The NOT EXISTS (query) clause returns TRUE if the query returns no tuples. This is equivalent to saying that the result of applying the COUNT function to the query must be zero. That is, the previous condition is

equivalent to saying that the following query returns zero:

```
SELECT COUNT (*) AS Invalid_Tuples
FROM
       (SELECT *
       FROM EMPLOYEE E,
              EMPLOYEE M,
              DEPARTMENT D
       WHERE E.Salary > M.Salary
              AND E.Dno = D.Dnumber
              AND D.Mgr_ssn = M.Ssn));
```

In a RDBMS, a view is a virtual table that represents the result of a query. If a view contains the result of the previous SELECT, a simply CHECK column constraint (for example, CONSTRAINT CK_ASC CHECK (Invalid_Tuples = 0)) should therefore be sufficient to guarantee this constraint. Unfortunately, traditional views are not allowed to include CHECK constraints. Simple integrity constraints stated in a view could imply very complex constraints in the base tables (base tables are the tables from which the view obtains the data). Nevertheless, CHECK constraints can be included in a kind of views called materialized views, as will be shown in the following paragraphs.

Some RDBMSs also implement "materialized views". Materialized views were first implemented in the Oracle Database. A materialized view (also called a snapshot) is a database object that contains the result of a query at a specific time, and is periodically updated on the basis of certain criteria. A materialized view eventually enables efficient access to the cost of some data that may potentially be out-of-date. In Oracle, the content of a materialized view may be updated manually or automatically: for example, every day by using the clause START WITH SYSDATE NEXT SYSDATE + 1, or when a commit occurs in the base tables by using the ON COMMIT clause. Materialized views are very useful in data warehousing environments, in which frequent queries regarding historical data can be expensive.

In Oracle, materialized views can include CHECK constraints, as common tables (or can even be stored in a previously created table). It would, therefore, be possible to define CHECK constraints in order to constrain the values stored in the materialized view. This would, in turn, constrain the values of the base tables used in the query defined in the materialized view.

Oracle therefore satisfies all the conditions that must be fulfilled in order to automatically implement assertions using materialized views. The basic procedure is as follows: a materialized view has to be created for each assertion. The materialized view will contain only one column, Invalid_Tuples, which will contain the number of tuples that violate the constraint specified in the assertion. Moreover, a CHECK column constraint constrains that the value of the column must be equal to zero.

The refresh method used will be ON COMMIT. This guarantees that for every commit the materialized view will be updated and, therefore, if the CHECK column constraint is violated, the operation that provoked the commit will be rejected. We therefore guarantee that when a/some invalid tuple/s appear/s, the materialized view is immediately updated. This obviously implies that if a particular operation in a transaction violates the constraint, the whole transaction will be rolled back instead of committed

We, therefore, first create the materialized view with a REFRESH ON COMMIT option:

```
CREATE MATERIALIZED VIEW
  ASSERTION_SALARY_CONSTRAINT
  REFRESH ON COMMIT
  AS SELECT COUNT(*) AS Invalid_Tuples
      FROM
      (SELECT *
      FROM EMPLOYEE E,
            EMPLOYEE M,
            DEPARTMENT D
      WHERE E.Salary > M.Salary
            AND E.Dno = D.Dnumber
            AND D.Mgr_ssn = M.Ssn);
```

We then modify the materialized view using an ALTER TABLE sentence, adding a CHECK constraint that guarantees that the number of invalid tuples is always zero:

```
ALTER TABLE
ASSERTION_SALARY_CONSTRAINT
       ADD CONSTRAINT CK_ASC
          CHECK (Invalid_Tuples = 0);
```

If we make some updates in the database that cause an employee to have a salary that is greater than that of the manager of the department that the employee works for, we will obtain the following error message when a commit occurs:

```
Error SQL: ORA-12008: error in
materialized view refresh path

ORA-02290: check constraint
(SYSTEM.CK_ASC) violated
```

The main disadvantage of the REFRESH ON COMMIT option is that the time required to complete the commit will be longer because of the extra

processing involved. Although this should not be an issue in a data warehouse environment, because it is unlikely that concurrent processes will be attempting to update the same table [12], this could be a problem in a transactional environment. Oracle has an option (the FAST REFRESH option) that can be used to improve the performance of the refreshing. The FAST REFRESH option performs an incremental refresh and requires the creation of a series of materialized view logs that store the changes made to the base tables since the last commit. The FAST REFRESH option also has some restrictions, such as the type of SELECT, aggregations, remote tables, etc [13].

4. RESULTS

We have carried out various experiments with the example shown in Sections 2 and 3, in an Oracle database, version 12c. Each execution of the example consisted of creating the tables and inserting the test data from scratch, in order to avoid malfunctions owing to data kept in the cache. We have carried out various tests applied to the previous example in order to compare the use of a set of triggers with that of materialized views (one normal and one using the FAST REFRESH option). We have specifically developed four triggers, because there are four situations in which the constraint can be violated:

- The first three are fired when a new employee is inserted, when the salary of an existing employee is modified, or when an existing employee is assigned to a different department. These triggers share virtually the same code, and need only compare the salary of the new or existing employee with the salary of the manager of the department.

- The fourth trigger is fired when the manager of an existing department is modified. In this case, the trigger needs to check that the salary of every employee in the department is not greater than the salary of the new manager. Note that this trigger does not need to be fired when a new department is inserted, because in this case no employees are still assigned to it.

Table 1. Inserting 100 employees (seconds)

	ONE FINAL COMMIT			ONE COMMIT AFTER EACH INSERT		
	MATERIALIZED VIEW	MATERIALIZED VIEW (FAST REFRESH)	TRIGGERS	MATERIALIZED VIEW	MATERIALIZED VIEW (FAST REFRESH)	TRIGGERS
10,000 EMPLOYEES	0.0765	0.1190	0.0264	0.9057	2.2282	0.0253
50,000 EMPLOYEES	0.0886	0.1182	0.0247	2.2106	2.4138	0.0271
100,000 EMPLOYEES	0.1098	0.1229	0.0253	3.7018	2.8495	0.0278
200,000 EMPLOYEES	0.1314	0.1266	0.0261	6.7539	3.4744	0.0293
300,000 EMPLOYEES	0.1650	0.1357	0.0254	9.8549	4.1711	0.0277

Table 2. Updating 100 managers (seconds)

	ONE FINAL COMMIT			ONE COMMIT AFTER EACH UPDATE		
	MATERIALIZED VIEW	MATERIALIZED VIEW (FAST REFRESH)	TRIGGERS	MATERIALIZED VIEW	MATERIALIZED VIEW (FAST REFRESH)	TRIGGERS
10,000 EMPLOYEES	0.0564	0.1102	0.0701	0.9656	2.9299	0.0934
50,000 EMPLOYEES	0.0666	0.1118	0.3174	2.2747	3.2582	0.3738
100,000 EMPLOYEES	0.0836	0.1178	0.6468	3.8715	3.8114	0.7125
200,000 EMPLOYEES	0.1182	0.1256	1.3163	7.1413	4.4121	1.4020
300,000 EMPLOYEES	0.1445	0.1370	1.9195	10.4417	5.3521	2.0144

Table 1 shows the average results obtained after repeating the insertion of 100 new employees into the EMPLOYEE table 10 times with 10,000, 50,000, 100,000, 200,000 and 300,000 tuples (in seconds). The first three columns show the results when only one commit is executed after the 100 insertions. The last three columns show the results when one commit is forced after each insertion.

The performance of the triggers is similar in both cases (with or without commits), and the size of the table has no significant effect, because the trigger only compares the salary of the employee with the salary of the manager of the department. Moreover, the triggers clearly perform better than materialized views, especially in the case of forcing a commit after each insert. In the case of

materialized views, the FAST REFRESH option only has a clear effect when the tables affected have a lot of tuples and a lot of commits have to be done. It seems reasonable that in almost empty tables the fact of having the additional task of managing a set of materialized view logs does not have any effect, or even a negative effect.

Table 2 shows the average results obtained after repeating the modification of 100 managers 10 times, again when the EMPLOYEES table has 10,000, 50,000, 100,000, 200,000 and 300,000 tuples.

In this case, the performance of the materialized views is similar to the previous one (Table 1), but the performance of the triggers has worsened. In this case, the performance of the triggers is affected by the size of the tables, because this trigger has to compare the salary of the new manager with the salary of all the employees in the department.

The main difference between both results (Table 1 and Table 2) is the performance of the triggers. The triggers that fire in the case of inserting a new employee (Table 1), or modifying an existing employee, only have to compare his/her salary with the salary of the manager of the department to which he/she belongs. Nevertheless, the trigger that fires in the case of the modification of the manager of one department (Table 2) is more complex, because it has to check that every employee in that department has a salary which is not greater than the salary of the new manager.

The following figures provide a graphic summary of the aforementioned tests. They show the performance of the triggers and materialized views after inserting 100 employees plus modifying the manager of a department 100 times (that is, the result of adding the results of Table 1 and Table 2). Figure 2 shows the results when a single commit is executed after each 100 operations and Figure 3 shows the result of forcing one commit after each operation. A linear trend line has also been added for each situation.

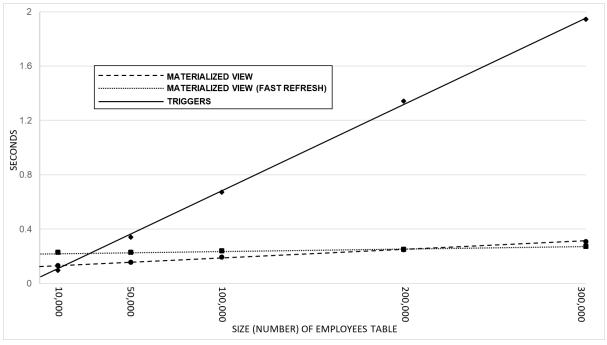

Figure 2. Inserting 100 employees and updating 100 managers, one final commit.

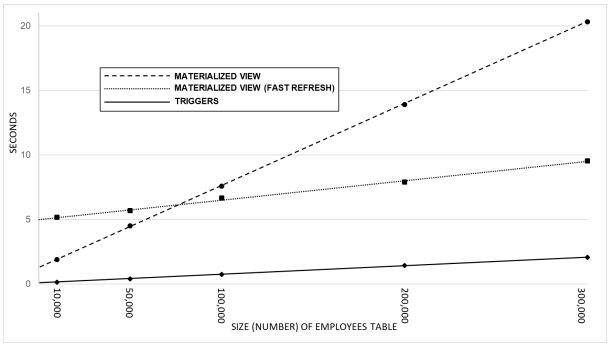

Figure 3. Inserting 100 employees and updating 100 managers, one commit after each insert/update.

As can be seen, the trend line fits very well with the results obtained. In Figure 2, when commits are executed only after the operations, triggers are the worst option when the Employees table has more than 20,000-25,000 tuples. With regard to the FAST REFRESH option, it starts to provide its benefits when the Employees table has about 200,000 tuples.

In Figure 3, when one commit is forced after each operation, triggers are always the best option. Although materialized views perform reasonably well in small tables, their performance worsens in the case of large tables, especially when the FAST REFRESH option is not used. The FAST REFRESH option starts to provide its benefits when the Employees table has about 75,000 tuples.

5. CONCLUSION

In an RDBMS, almost any constraint can be specified as an SQL query. Virtually any constraint could, therefore, be expressed using an assertion. Unfortunately, no commercial RDBMS product supports assertions. This signifies that cross-row constraints are usually implemented using a set of complex triggers, or are included in the applications. In some cases, this method is more efficient, although it is error-prone owing to the quantity of possible situations that must be taken into account. In some situations, it might therefore be better to automate this codification task and delegate it to the RDBMS.

In this paper, we have proposed an automatic and easy means of implementing assertions in RDBMSs which support materialized views and allow an ON COMMIT refresh of these views.

To the best of our knowledge, the only approach for the automatic implementation of assertions is that of Oriol et al. [14], who propose an incremental approach consisting of implementing assertions that create several triggers in order to capture the update requested by the user and placing it in auxiliary tables in SQL Server. Their method consists of generating a set of triggers and views that check the assertions, all in a semi-automatic manner (manual intervention is necessary because the user has to

manually invoke a procedure called safeCommit at the end of each transaction). The results provided by the authors in order to check the assertions range from 0.01 to 1.29 seconds, which are similar to those obtained in our experiments in the case of using triggers, signifying that they are reasonable. Unfortunately, the aforementioned authors' method does not take into account the possibility of including aggregation functions in assertions, which is a fundamental disadvantage, since it works only for very simple assertions.

Our main objective was to analyze whether implementing assertions by means of materialized views is a viable method. We have shown that it is possible, and that it can be easily automated. Although our proposal is specific to Oracle, it would be applicable to any system that supports CHECK constraints and REFRESH ON COMMIT on materialized views. However, to the best of our knowledge, no system other than Oracle currently supports all of these aspects.

We have also carried out some preliminary tests in order to evaluate the efficiency of our approach. These tests showed that the method provides good results (even better than triggers) when making a unique commit after a set of inserts or updates. Nevertheless, it performs worse than when using a set of triggers when a commit is forced after each insertion or update. The following conclusions may be obtained from the aforementioned experiments:

- Materialized views perform similarly, independently of the cause that violates the constraint. It depends mainly on the size of the table.

- The performance of the triggers depends mainly on the particular constraint, and less on the size of the table.

- Using the FAST REFRESH option in materialized views provides benefits only in large tables.

- In general, materialized views perform better than triggers in the case of issuing a single commit for all the modifications made, as the condition is evaluated only once (when the commit is performed). In the case of the triggers, which are part of the same transaction of the firing statement, the condition is evaluated for each operation. The performance of both triggers and materialized views obviously also depends on the complexity of the condition to be evaluated and the operation that can fire the assertions.

Our method, therefore, fits better in environments with a high number of complex constraints but a low number of transactions. It is less appropriate for environments in which a high number of simple transactions are present. We are currently working on a detailed categorization of when it is worth using this method, along with its automatic implementation in a tool.

6. ACKNOWLEDGMENTS

This work has been partially supported by the Access@City research project (TIN2016-78103-C2-1-R), funded by the Spanish Ministry of Science, Innovation and Universities.

7. REFERENCES

[1] C.J. Date, 2015. SQL and Relational Theory: How to Write Accurate SQL Code. Third ed., O'Reilly.

[2] A. Behrend, R. Manthey and B. Pieper, 2001. An Amateur's Introduction to Integrity Constraints and Integrity Checking in SQL, Datenbanksysteme in Büro, Technik und Wissenschaft. A. Heuer, F. Leymann and D. Priebe, eds, Informatik aktuell. Springer, Berlin, Heidelberg, 405-423.

[3] ANSI Standard, 1992. The SQL 92 Standard. http://savage.net.au/SQL/sql-92.bnf.htm

[4] J. Melton and A. R. Simon, 2002. SQL 1999: Understanding Relational Language Components, Morgan Kaufmann.

[5] Y.I. Chang and F.L. Chen, 1997. RBE: A Rule-by-example Active Database System, Software: Practice and Experience, 27(4):365-394.

[6] M. Brucato, A. Abouzied and A. Meliou, 2017. A Scalable Execution Engine for Package Queries. SIGMOD Record, 46(1): 24-31.

[7] Oracle. http://www.oracle.com

[8] T. Koppelaars, 2016. SQL Assertions / Declarative multi-row constraints". https://community. oracle.com/ideas/ 13028.

[9] R. Elmasri and S. Navathe,2010. Fundamentals of Database Systems, Sixth ed., Addison-Wesley.

[10] H.T. Al-Jumaily, D. Cuadra and P. Martínez, 2008 "OCL2Trigger: Deriving active mechanisms for relational databases using Model-Driven Architecture", Journal of Systems and Software, 81(12):2299-2314.

[11] H. Zhang, H.B.K. Tan, L. Zhang, X. Lin, X. Wang, C. Zhang and H. Mei, 2011. Checking enforcement of integrity constraints in database applications based on code patterns", Journal of Systems and Software, 84(12):2253-2264.

[12] Oracle Database SQL Language Reference, 11g Release 2 (11.2).http://docs.oracle.com/cd/E11882_01/server.112/e41084.pdf

[13] P. Lane and P. Potineni, 2014. Oracle Database Data Warehousing Guide, 12c Release 1 (12.1). Oracle.

[14] X. Oriol, E. Teniente and G. Rull, 2016. TINTIN: a Tool for Incremental INTegrity checking of Assertions in SQL Server", 19th International Conference on Extending Database Technology (EDBT): 632-635.

Michael Franklin Speaks Out on Data Science

Marianne Winslett and Vanessa Braganholo

Mike Franklin

https://cs.uchicago.edu/people/profile/michael-franklin/

Welcome to ACM SIGMOD Record series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we're at the 2017 SIGMOD and PODS conference in Chicago. I have here with me Mike Franklin, who is the chair of the Computer Science department at the University of Chicago. Before that, for many years, Mike was a professor at Berkeley where he also served as a chair of the Computer Science division. Mike was a co-founder and director of the Algorithms, Machines, and People Lab, better known as the AMPLab. He is an ACM fellow, a two-time winner of the SIGMOD Ten Year Test of Time Award, and a founder of the successful startup, Truviso. Mike's Ph.D. is from the University of Wisconsin Madison. So, Mike, welcome!

Everyone wants to know why you moved from Silicon Valley, the epicenter of all things computer, to the Midwest?

I had a great 17 years at UC Berkeley and being in and around Silicon Valley. But I moved to Chicago to take advantage of an amazing opportunity here to help build computer science and data science in a new way that's integrated into the fabric of the university. The University of Chicago has tremendous programs across a huge range of fields, ranging from biological sciences to public policy, economics, of course, social sciences, and humanities and they've decided as a university that they want computer science and data science to play an increasingly central role across all those different fields.

And so, the opportunity I have at Chicago is to build a modern Computer Science department that in its very nature is built to work with people across all these different disciplines. And that combined with the opportunities of the growing tech field in Chicago and the Midwest more generally, it just seemed that after a great run at Berkeley it was time to do something new and so that's what I signed up for.

What do you miss most about Silicon Valley?

Silicon Valley is really a unique place. The energy, the sense of adventure, the sense of just trying to make something big happen that permeates the whole place is something that is hard to replicate somewhere else. So, I miss that, yeah.

You miss it, but it is a bubble.

Yeah, so the downside of all that energy is it's really all-encompassing. And when you're there, you're out talking to people, you're sitting in a café, you're at a restaurant, the topic of conversation is stock options and the next round of funding and the minimal viable product and all this. At some point, it does get to be a bit much.

One of the great things about Chicago and the Midwest in general is it's a much more diversified place. There's no one industry that dominates the Chicago economy and because of that you get people with widely varying interests all talking together, working together. It's, in some ways a refreshing change.

You weren't very bullish on the short-term prospects for real-time streaming analytics when you gave a keynote on that topic at a VLDB 2015 workshop. Have your views shifted in the two years since then?

So, I think some people might have misunderstood what I was saying in that talk, and I've given versions of that talk in different places. So, just for some history, we were working on streaming in the early 2000s when that was a hot topic in the database community, and the company you mentioned, Truviso, was a streaming analytics company. My view has always been that stream processing is absolutely going to be an integral part of any data analytics platform because it's just the most efficient way to answer queries that you already know you want to ask.

If you already have a bunch of queries, which often you do, that you know you're going to want the answer to, it's much more efficient to answer them incrementally on the fly as the data's arriving as opposed to storing the data off and then going to find it later and then starting the query from scratch. So, my view has always been that streaming would be a component of analytic systems. Now, what I did push back on is this idea that everybody is going to want instantaneous answers to queries, no matter what they're doing, and that's just not the way it turned out the first time we did it, and it's not going to turn out this way either, and there are a few reasons for that.

One reason is: business processes and other types of processes just have a natural cadence and that for a lot of reasons you can only make decisions every so often. And getting people the freshest data instantaneously when they're not able to make a decision or act on a decision is, in the best case, a waste of time, and in the worst case, a big distraction.

The other problem is, if you're trying to answer queries instantly when the data comes in, you don't have time to deal with problems in that data. So, for example, if there's out of order data, which is very typical in streaming environments, if you're trying to answer things instantaneously, you're going to miss a lot of data. If there's an error in the data and you're not able to analyze it properly, you're going to cause problems that way. So, really, my view on it is that if you ask people, do they want faster query answers, they're going to say yes. If you ask people, "for the problem you're trying to solve, how often do you need a correct answer," you'd get probably a very different and in many cases a much slower rate. So my view on it is that stream processing is really important because it's a fundamental way of dealing with large volumes of data, but you've got to take into account what people are actually trying to do and then target the solution to the latency needs of that application.

That's true of life in general, wouldn't you say? You need to have situational awareness. Too much situational awareness is bad; you end up with helicopter parents and things like that. And too little can lead to a disaster because you don't know what's going on. So,

it's a lesson that we need to think about in applications for data.

Yeah, I think that's right.

> ***[…] getting people the freshest data instantaneously when they're not able to make a decision or act on a decision is, in the best case, a waste of time, and in the worst case, a big distraction.***

The AMPLab is about algorithms, machines, and people. Unlike the other parts of the AMPLab, the people component hasn't produced results that have made their way into industry. Why is that?

Right, that's a great question. So, the premise of the AMPLab when we started it was exactly that. If you're going to try to make sense of big data, you have three types of resources you can use. You can use algorithms in terms of machine learning and statistical processing. The machines part of the agenda was about cloud computing and cluster computing and basically throwing more scalable hardware at the problems. And the people part was initially about crowdsourcing and about bringing people to bear on the parts of the problems that weren't adequately addressed by the algorithms and the machines. The dream was that we would build an integrated system that combined all of these.

Now, what happened was certain parts of the AMPLab agenda just took off like rocket ships. The one that, of course, is most famous is Apache Spark and all the things around it. Apache Spark and its ecosystem has taken a leadership role, not just in academia, but more so in industry in terms of what people are doing with big data. And so when people look at the AMPLab, they see the Spark part of the agenda, and they say, wow, that was a huge success and really beyond what you'd expect from an academic project. But when you look at some of the other parts of what we were doing, they didn't have that same industrial impact as your question says – at least they haven't had it yet.

But one thing I like to point out (it's a little defensive about the people part of the agenda) is if you look at what happened for the people who worked on that part of the agenda, we had best paper awards, we had a series of papers including, I believe, the most referenced paper in SIGMOD from the previous five years[1]. The students and postdocs who worked on that project are now at some of the top universities in the country. So, by any metric, the people part of the agenda was a huge research success, but when you stand it up next to Apache Spark, it doesn't have that same industrial impact, which was your question.

So, now the question is: Why is that? There are two things. One is the nature of the way people think about people in an overall systems architecture. If you look at large web companies that are ingesting and trying to make sense of lots of data, and you ask them to draw out their systems architecture, you'll see racks of machines running certain processes and communicating in certain ways. All of those companies have armies of people that are doing exactly what we set out to do in the AMPLab, which is to have the people do those things that you just couldn't get the right fidelity out of the algorithms and the machines to handle properly. But nobody draws their architecture saying, "oh, and the hard stuff that we can't afford to get wrong, we're going to show to this group of 500 people that we're paying". And so part of it is just there isn't the set of systems abstractions yet for how people fit into the architecture. Everyone thinks about it as those people are somehow separate from the architecture.

Well, yeah, now that Facebook has to deal with the fake news in a very people-intensive manner, do you think that that will cause a change in people's thinking of what an architecture consists of?

I'm not sure because they've been doing that all along. Companies have been bringing in people for solving those hard problems.

True, but 2,000 of them, I think that's the number.

Right, yeah.

[1] At the time of the interview (2017), this was referring to: Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, Reynold Xin: CrowdDB: Answering Queries with Crowdsourcing, SIGMOD Conference 2011: 61-72. As of the time of publication (2019), a different AMPLab paper now holds this position: Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, Matei Zaharia: Spark SQL: Relational Data Processing in Spark. SIGMOD Conference 2015: 1383-1394.

Do you think that will show up on future diagrams?

I don't know what it would take to make it just standard practice to have the people part show up on diagrams. I think if there are enough stories like that where it starts coming out more in the open, maybe it will.

The other reason I think that industrial impact of the research has been a little slower than in other areas is because we don't yet have the relational algebra of crowdsourcing. We don't have a standard set of abstractions, of operators, of benchmarks that you need to make it so that people in industry can get their heads around a particular set of concepts and then start understanding what their needs are and what the alternative solutions are and how they compare. So, part of it is just this awareness about the fact that people are already integral parts of these systems. And the other part is that we in the research community have to do a better job of determining what are the fundamental abstractions of crowdsourcing to make it happen.

What's next for you in your research life?

Well, so I've taken on a pretty big administrative role these days, and I'm focused on basically doubling the size of our department. We're building a new building[2], we're building a lot of new programs, and so I think that's going to keep me pretty occupied for a little while. But as I talk to people around campus about what their needs are for data science, it's a lot of those fundamental problems that the SIGMOD community has been beating their heads against the wall on for a long time that still need to be addressed: data integration, data cleaning, and data quality.

There's an improved awareness now of how bias and other problems in the underlying data can impact the results that are coming out of analytics, and so really what I want to focus my research on, for the next wave of my research, is those types of data quality issues.

You are currently building up a big new data science center in Chicago. How do you decide what to include in the scope?

The challenge of data science is exactly that. That if you look across the disciplines in a modern university, pretty much all of them are doing more and more with data and feel that they need to be involved in data science. And so I've been taking a pragmatic approach, looking to see who's willing to step up and contribute and basically using that as my guide for what to keep in because intellectually, you really can't make an argument that these people are doing data, but that's not data science whereas these people are. But I think finding out who really wants to collaborate, who's willing to put some hard work into thinking about curricular issues and putting some time into it, that's what's going to drive who's involved, at least in the beginning.

Could there be a problem later on if initial successes make a bunch more departments want to get on board, but you're built up or scoped out or whatever?

I think whatever we design is going to have to be designed with the assumption that eventually everybody is going to want to be involved in some way.

> ***[…] there isn't the set of systems abstractions yet for how people fit into […] [a system's] architecture. Everyone thinks about it as those people are somehow separate from the architecture.***

The University of Chicago has a great books curriculum. What do you think are the equivalent of great books in computer science?

The University of Chicago curriculum, as far as I understand it, has moved away from the great books a little bit and is more of what they call now as a core. The core is determined less by specific titles and more by concepts and techniques and philosophies that you need to understand to be an educated person in the 21st century. And if you look at it from that point of view, I think it's pretty obvious that to be an educated person in the 21st century you need to understand something about computation, right? – this idea of computational thinking and how algorithms work and what they are and what they can do and what they can't do. And also you need to understand data, and you need to understand how to make arguments given data. You need to understand when somebody's presenting you an argument that supposedly comes from data, how to determine what might be right or wrong about that, and so that's more sort of a data literacy. If you think about

[2] The U Chicago CS Department moved into its new facility in August 2018.

computational thinking and data literacy, I fully believe that they will end up in the core set of things that the university decides all undergraduates need to know, and I think that's going to happen everywhere.

Do you think there'll ever be classics?

I think that there certainly will be classics about understanding the relationship between computation and thought, right, and around the limits and the opportunities of artificial intelligence and things like that. If I had to give everybody a book to read, it would probably be the Gray and Reuter Transaction Processing book.

I thought of that as a possibility, but you give it to the average or even a computer science major, and they're going to be "oh my God, I had no idea, I don't want to know."

Right, so that's the problem. The amount of things you need to learn and do until you can understand what's in that book is too high.

The level of complexity is kind of astonishing.

Yup.

Alright, we'd like to hear your advice for having a successful startup in the data space beyond all the advice we can already find on the internet.

Okay, well, I'll say a couple of things, and you can tell me if it's already out there. It probably is.

Okay.

Related to that, one piece of advice I got when I started my company was that you're going to find people who have a similar or the same idea that you have and that's actually not a bad thing. If you're the only person who's come up with an idea in a hot area, there's probably something fundamentally wrong with it. And so, if I'm giving you advice, it's probably already out on the internet somewhere or it's wrong advice.

But the thing that I learned about data-driven startups that surprised me was – you know, our first customer in our company was a hedge fund that was doing currency trading and we built an amazing application for them that let them see things in real-time that they couldn't see before. And in that business, you really do need to see things in real-time. We had that running, and we then went over to have a meeting with a computer security company, and they said, "oh, well, do you have a demo of your product" and we said "absolutely". Then we showed them the demo of this currency trading application. And they said, "well, what's that?", and we explained it. They said, "well, what does that have to do with computer security?" We said, "well, you understand, right, there's streams of data coming in, there are these comparisons and metrics and thresholds and other things are being computed in real-time and they're being shown. So, you can imagine that these are now network events coming in and the things that you're querying are security events…" But they just totally couldn't get it – and these were smart people, these were not dumb people. And then this repeated in a bunch of other industries as well.

What if you'd framed it as situational awareness because security community does have that concept?

It could be that we were missing the terminology, but really, I think what I learned from that is as data people, we're able to think about data and queries in the abstract – we see patterns, we see similarities. A customer that's trying to solve their problem sees only their problem, and they see it in the set of data that they deal with, and the set of questions that they ask. It's extremely rare, if not impossible, to find somebody at a company that's trying to solve a problem that has that same idea of abstraction that a database person would have.

So, my advice for people doing data-driven startups is to really put yourself in your potential customer's shoes and gear whatever you're presenting to solving their problems, not a bottom-up way about the technology itself.

Interesting, so in the particular example you gave, does that mean you need to cobble together a fake network monitoring demo for them to get it?

Yeah, I think cobbling a demo or…

That's a lot of work!

It's a lot of work, and I guess a corollary to that is you should very quickly pick a small number of verticals, maybe just one, and focus your energies on that for exactly that reason.

Times have definitely changed, so do you think if you come back to a big network-oriented company today with your saying "here's your real-time dashboard situational awareness," would that now be an easy sell or would it be the exact same thing all over again?

I think there's increased awareness. I mean, the story I told is probably a ten-year-old story. So, you're right, in the past ten years it's possible that there's a greater

appreciation now for data science in general, but I still think at the end of the day things that are very natural for you as a database technologist are not natural for a domain expert, and you just have to be aware of that.

> ***[…] we don't yet have the relational algebra of crowdsourcing.***

You spent years working at MCC before you went for your Ph.D. Did you plan to get a Ph.D all along?

I actually bounced between industry and academia quite a bit during my career. I worked after my bachelor's degree and then the MCC job was after I got a master's degree. I really had no intention of getting a Ph.D at the time when I took that job, but I worked with a lot of Ph.Ds and during the course of that project, we built a system called Bubba[3], which was one of the first massively parallel database systems. And working with the people who were Ph.Ds kind of impressed me. I liked the way they thought, I liked the way they thought about problems, and the way they attacked big problems, and it kind of gave me an appreciation for that and realized that that was something I wanted to learn how to do.

Now, that's interesting. So, I guess master's degrees were the highest degree offered by where you got your master's degree. Do you think if you'd been at a place that also had Ph.D students you would have had that same reaction at that time?

That's a good question. I always encourage people at universities now (especially undergrads), to do some research, to get involved in a research project to see if they like it, because you're right, had I been at a place that was actively doing research projects, I might have had that same experience.

Interesting. Did your time away from the university affect what you ultimately chose to do for your Ph.D?

Oh, absolutely. So, at MCC, in addition to working with some amazing people at MCC, I also had the opportunity to work with Mike Carey who eventually became my Ph.D advisor and Dave DeWitt, both of whom were consultants on that project and they were the reason I went to the University of Wisconsin.

Most people go straight through school for a variety of reasons. For people who think they want to get an additional degree, under what situations do you think they should spend time away from school first?

Well, I often encourage people – if you're in a place that has a one-year master's program, there are a number of schools that you do your bachelor's, if you stay another year, you get the master's… I think that's a really great opportunity for people and I also encourage people to do that. To go for your Ph.D, that's a whole other thing. One thing that students who are considering a Ph.D don't often understand is that everything up to the Ph.D, often including the master's, is basically course driven, so you take your courses, you do the exams, you do the projects. At the end of the semester, you're done, and you move on.

As you know, a Ph.D isn't like that at all. It's kind of an unbounded enterprise that you're getting involved in and it's a very different set of criteria for success. And you often don't get that regular feedback, that regular sense of accomplishment – certainly in the first few years when you're just trying to find your way around the research world. So I always encourage people, even who think they want to get a Ph.D, to maybe take a little time and spend some time in industry and see what's out there, to see if that's going to be a better path for them. My feeling, and it's just based on my own personal experience, is as long as you're not out too long, if you really want to go back, you will, and there will be opportunities to do that.

What approach to advising do you take with your own Ph.D students?

My approach with students has, I think, tended a little more towards the hands-off approach. I try to give students a direction and pose problems and then turn them loose on those problems and not dictate what I think the solution should be.

When I first became a professor, I ran into Jeff Naughton, who was one of my professors at Wisconsin, and he had asked me how it was, was I enjoying faculty life and whatever. I said, well, I really am, except that I really wish I could figure out what it takes for a given student to be successful because things that work for one student don't work for another. There are some people who need a little bit of pressure, there are some people

[3] Haran Boral, William Alexander, Larry Clay, George P. Copeland, Scott Danforth, Michael J. Franklin, Brian E. Hart, Marc G. Smith, Patrick Valduriez: Prototyping Bubba, A Highly Parallel Database System. IEEE Trans. Knowl. Data Eng. 2(1): 4-24 (1990).

who crack under pressure… I haven't figured out an approach to make students successful in general. And Jeff, who had been teaching for a number of years at that point, said to me, well, yeah, when you figure it out, let me know.

So, I think you do have to understand that people are motivated in different ways, but the best way to work with me is to be open-minded about the problems that you're going to work on and then be creative about the solutions you come up with.

Do you have any other words of advice for fledgling or mid-career database researchers?

We're really lucky in the database world. Those of us who have been in the business for a long time remember when it was one of the less glamorous parts of the field, where it was hard to get people to take the classes, it was hard to get people to want to do research compared to some of the sexier parts of computer science. But because of the big data revolution and because of data science and all the companies that are clearly being driven by data, that's not true anymore. I guess my advice is really to just enjoy being in a field that's having such an impact on the world and that has so many open problems.

Among all your past research, do you have a favorite piece of work?

Well, the AMPLab is hard to beat as a research project for a number of reasons. One, the impact that we had was really just, as I said, well beyond what you would ever hope for from an academic project and that's been really wonderful. But the other great thing about the AMPLab was that it was a collaboration of a large number of faculty, a large number of students, and these were people not just from databases – actually not even just from systems, but we had machine learning people, we had HCI people, we had security people, and then we had applications people around the campus who we were working with. So overall, it's hard to beat that experience as a research project.

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

I haven't written a book yet and I think I need to write a book, so if I magically had time, that's what I would probably do.

I fully believe that [computational thinking and data literacy] will end up in the core set of things that the university decides all undergraduates need to know.

What would the topic be?

Well, clearly, we need a new database textbook for undergraduates and somebody has to do that. The other one would be about building systems for large-scale machine learning.

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

If I could go back to my education, there's definitely some math classes I would have paid more attention to, because I'm finding now that a lot of the techniques that are taught in those courses are more important to me as a database researcher than I thought they would have been.

Is that, in some sense, a failure of the professor or was the class so generic that the teachers themselves could not have envisioned all the different ways that their students might put that work to use?

Well, I think, particularly for databases, advanced analytics and machine learning have now become so important that a lot of techniques around linear algebra and stochastic processes and optimization are just much more important to the field than they were even ten years ago.

Okay, great, thank you very much for talking with us today.

Thank you.

Report on the First and Second Interdisciplinary Time Series Analysis Workshop (ITISA)

Themis Palpanas
Paris Descartes University
themis@mi.parisdescartes.fr

Volker Beckmann
CNRS, Paris Diderot University
beckmann@in2p3.fr

ABSTRACT

The analysis of time-series data associated with modern-day industrial operations and scientific experiments is now pushing both computational power and resources to their limits. In order to analyze the existing and (more importantly) future very large time series collections, new technologies and the development of more efficient and smarter algorithms are required. The two editions of the Interdisciplinary Time Series Analysis Workshop brought together data analysts from the fields of computer science, astrophysics, neuroscience, engineering, electricity networks, and music. The focus of these workshops was on the requirements of different applications in the various domains, and also on the advances in both academia and industry, in the areas of time-series management and analysis. In this paper, we summarize the experiences presented in and the results obtained from the two workshops, highlighting the relevant state-of-the-art-techniques and open research problems.

1. INTRODUCTION

Time series[1] have gathered the attention of the data management community for more than two decades [1, 15, 30]. They are one of the most common data types, present in virtually every scientific and social domain: they appear as audio sequences [13], shape and image data [29], financial [26], environmental monitoring [25] and scientific data [11], and they have many diverse applications, such as in health care, astronomy, biology, economics, etc.

Recent advances in sensing, networking, data processing and storage technologies have significantly eased the process of generating and collecting data series. It is not unusual for applications to involve numbers of sequences in the order of hundreds of millions to billions [21]. These data have to be analyzed to identify patterns, gain insights, detect anomalies, and extract new knowledge.

A key observation is that analysts need to process a sequence (or subsequence) of values as a single object, rather than the individual points independently, which is what makes the management and analysis of data sequences a hard problem. Note that even though a sequence can be regarded as a point in n-dimensional space, traditional multi-dimensional approaches fail in this case, mainly due to the combination of the following two reasons: (a) the dimensionality is typically very high, i.e., in the order of several hundreds to several thousands, and (b) dimensions are strictly ordered (imposed by the sequence) and neighboring values are correlated.

Current time series analysis solutions require custom code, which implies huge investments in time and effort, and duplication of effort across different teams. Existing systems (e.g., based on DBMSs, Column Stores, or Array Databases) do not provide a viable solution, since they have not been designed for managing and processing sequence data [21]. Therefore, they do not offer a suitable declarative query language, storage model, auxiliary data structures, and optimization mechanism that can support a variety of sequence query workloads in an efficient manner [6,31]. (In Section 4, we discuss more reasons why existing solutions are inadequate.)

The Interdisciplinary Time Series Analysis Workshop provided a forum for researchers and practitioners that approach time series from different angles, ranging from data management and processing, to analysis, mining and machine learning. The core research issues considered in the workshop include: management and indexing, interactive visualization, machine learning, privacy preserving analytics, uncertainty and missing values, and applications of those in astrophysics, neuroscience, engineering, electricity networks, and music.

The program of the two editions of the workshop included 14 keynote talks, 2 hands-on sessions, and 2 panel discussions. We summarize here the ideas

[1] *Time series*, or *data series*, or *sequences* are values measured and ordered over a dimension (usually time, but could also be mass in mass spectroscopy, angle in radial chemical profiles, or position in genome sequences).

that were presented and discussed in the two workshops that took place in June and December 2016 (in Paris, France) with over 80 participants in total. The detailed program and the slides for the talks are available at the ITISA web pages.
1st edition: https://indico.in2p3.fr/event/13186/
2nd edition: https://indico.in2p3.fr/event/13934/

2. KEYNOTE TALKS

2.1 Computer Science

Prof. Anthony Bagnall (University of East Anglia) focused on Time series classification problems (TSC). He described the recent advances in time series classification, and presented a taxonomy of algorithms based on the nature of discriminatory features used to classify. Finally, he presented an experimental comparison of over 20 algorithms on 85 of the UCR-UEA datasets. The results showed that the collective of transformation-based ensembles (COTE) was significantly more accurate than all other approaches, because it could utilize features from each of the five promising representations identified in the algorithm taxonomy.

Prof. Abdullah Mueen (University of New Mexico) talked about algorithms and applications of three primitive temporal patterns, namely, motifs, shapelets, and discords. Motifs are repeating segments in seemingly random time series data; Shapelets are small segments of long time series characterizing their sources; and Discords are anomalous waveforms in long time series that do not repeat anywhere else. He discussed efficient algorithms to discover these patterns, and presented corresponding use cases. Applications included activity classification using accelerometer and brain activity data, correlated clusters in social media data, and anomaly detection in online review data.

Prof. Themis Palpanas (Paris Descartes University) presented techniques for time series indexing. He described recent efforts in designing techniques for indexing and mining massive collections of time series, and showed that the main bottleneck is the time taken to build the index. He presented the state of the art techniques that adaptively create time series indexes, allowing users to correctly answer queries before the indexing task is finished.

Prof. Anastasia Bezerianos (University Paris-Sud), Dr. Theophanis Tsandilas (Inria), and Ms. Anna Gogolou (Inria) talked about interactive visual exploration of large time series collections. They provided an overview of existing interaction and visualization techniques for time series exploration and analysis, and noted the absence of focus on their scalability to multi-terabyte time series collections. To this end, they described work directions for achieving both visual scalability (how can we visualize billions of data series) and response-time scalability (how can we get answers quickly in interactive response times), including approximate and progressive result mechanisms.

2.2 Astrophysics

Dr. Dimitrios Emmanoulopoulos (University of Southampton) talked about astrophysical time series, and in particular AGN light curves[2], whose variability allow astrophysicists to study the physical conditions around a black hole. In order to test any theoretical model though, it is crucial to attribute a precise statistical significance to any timing property. Dr. Emmanolopoulos presented a new statistical method that can produce random light curves that contain all the genuine statistical and variability properties of the observed ones, i.e., the same flux distribution (quantified by the probability density function) and same power spectral density.

Dr. Jerome Rodriguez (CEA) discussed methods for diagnosing and analyzing the fast time variability in X-ray binaries[3]. He introduced the Fourier analysis and generic techniques used in high energy astrophysics, and explained how these tools help understand the properties of fast variabilities in X-ray binaries time series.

Dr. Gabriele Ponti (Max Planck Institute) also focused on the variabilities of X-ray binaries time series. He showed that the characteristic time-scales of such variations depend linearly on the mass of the black hole, and that by studying the correlations between the emission at various energy bands (through cross spectra and lag frequency spectra), it is possible to determine delays between radiation produced by different components of the system. From this, it is possible to draw conclusions about the geometry of the regions around black holes.

Dr. Vivien Raymond (Cardiff University) focused on Gravitational Wave (GW)[4] detection using the Advanced Laser Interferometer Gravitational-wave Observatory (LIGO). He stressed that at the core of this new observational medium for GW-astronomy

[2] An *Active Galactic Nucleus (AGN)* is a compact region at the center of a galaxy that has an unusually high luminosity. A *light curve* is an astrophysical time series that measures the amount of light as a function of time.
[3] *X-ray binaries* (a.k.a. microquasars) are a class of binary stars that host the most compact objects (neutron stars and black holes), and are luminous in X-rays.
[4] *Gravitational waves* are ripples in space-time caused by violent and energetic processes in the universe (e.g., the merge of two black holes). Albert Einstein predicted the existence of gravitational waves in 1916 in his general theory of relativity, and they were first detected in 2016.

is the analysis of the time series of space-time deformations recorded by the GW detectors. He presented the time series analysis techniques currently used in signal detection, detector noise analyses, and source properties inference.

Dr. Eric Chassande-Mottin (CNRS) talked about the detection of GW in space. It is believed that the space-based LISA GW detector data stream will contain approximately 60 million simultaneous sources. He emphasized that in analyzing these time series data, there are two important goals: the first is to separate the various sources from each other, and the second is to estimate the astrophysical parameters of each source. In this task, matched filtering is the main tool of GW astronomy, a method that requires the use of accurate theoretical templates for each source type. development of sophisticated Bayesian algorithms.

2.3 Neuroscience

Dr. Katia Lehongre (ICM Institute for Brain and Spinal Cord) talked about times series analysis in neuroscience, and elaborated on the electrophysiology of epilepsy. Patients with epilepsy present abnormal brain activity, like epileptic spikes and seizures that can be recorded with electroencephalography (EEG). In order to localize the region of the brain that produces this abnormal activity, EEG from the patients is recorded continuously for 2 to 3 weeks. Usual clinical practice involves a neurologist reviewing visually the signal in order to determine the spatial localization and the temporal dynamics of the epileptic activity. As Dr. Lehongre pointed out, several studies tried to develop an automatic and reliable detection / characterization of the epileptic events in time and space, however, no fully non-supervised methods are commonly used by the neurologists, because they are not accurate enough. An efficient time series analysis could be of great interest to speed up the signal analysis, and in turn to increase the number of patients handled.

Prof. Uri Hasson (University of Trento) discussed time series in relation to the brain. He began with a brief overview of the sorts of time series that modern cognitive neuroscience can obtain from human participants and the principles of the instrumentation used to obtain those. The second part of the talk focused on approaches for analyzing these times series. These include quantification of correlations between different brain regions and network partitioning strategies. It was noted that more recent work is focusing on fast, non-oscillatory signatures in brain dynamics that also contain important information. These signatures are either driven by an input or internally generated. Several methods based on the analysis of peaks and pits in neural time series were discussed, as well as methods for decomposing spatiotemporal data into series of micro-states and motifs. The last part summarized these technologies from the perspective of temporal search engines, highlighting the importance of approximate searches on multivariate time series, and in the context of real-time analysis.

2.4 Engineering and Electricity Networks

Dr. Dohy Hong (Safran) shared his experience on multivariate time series analysis in aeronautics, and aircraft engines in particular. He pointed out that one of the main future challenges in aeronautics is the use of available data in order to enable the optimal management of maintenance processes (e.g., a per engine-based individual management strategy), and later its integration in the design process. The overall data processing along the engine cycle includes several technical challenges: weak signal detection in continuous multivariate usage time series (hidden layer/rules learning/extraction), management of heterogeneous data (maintenance repair and operation data, test bench or inspection data, configuration data, etc.), integrating and consolidating the existing expert knowledge (from design model to residual life time estimate practice), and others. All these challenges have to be addressed under the constraint of certifiability, which implies interpretability and robustness.

Dr. Georges Hebrail (EDF) made the case for privacy-preserving use of individual smart meter data for customer services. The advent of on-body/at-home sensors connected to personal devices leads to the generation of fine grain highly sensitive personal data at an unprecedented rate. However, despite the promises of large scale analytics there are obvious privacy concerns that prevent individuals to share their personal data. Dr. Hebrail presented Chiaroscuro, a solution for clustering personal time series, with strong privacy guarantees. The execution sequence produced by Chiaroscuro is massively distributed on personal devices, coping with arbitrary connections/disconnections. Chiaroscuro builds on a novel data structure, which allows the participating devices to collaborate privately by combining encryption with differential privacy.

2.5 Music

Prof. Philippe Esling (IRCAM) talked about musical time series. Music inherently conveys several open and interesting scientific questions, which all embed the notion of time. Specifically, musical orchestration is the subtle art of writing musical pieces

for orchestra, by combining the spectral properties specific to each instrument in order to achieve a particular sonic goal. Prof. Esling described novel learning and mining algorithms on multivariate time series that can cope with the various time scales that are inherent in musical perception, and can be used for orchestration. His current research is focused on automatic inference through deep representational learning to allow the automatic deciphering of these dimensions in order to provide optimal features for orchestration, by targeting correlations existing in the work of notorious composers.

3. HANDS-ON SESSIONS

The workshop also offered two hands-on sessions.

The first hands-on session was organized by Dr. Vivien Raymond, and walked participants through the process of analyzing real time series signals collected at LIGO, and visualize a GW that was buried in the signal. This session aimed at teaching participants all the preprocessing steps necessary for cleaning the time series, and for amplifying the true signal, i.e., the GW. This process involved filtering and noise removal and downsampling steps[5], which had to be performed either in the time- or frequency-domain. The main take-away message of this session was that the time series processing workflow that analysts apply (often times) requires many preprocessing and data transformation steps.

The second hands-on session was led by Ms. Anna Gogolou, and dealt with the challenges in interactive visual exploration of large time series. The participants were asked to reply to a questionnaire that was divided in three parts: background information, scenarios, and detailed examples of their questions and problems at hand. Participants came mainly from two different domains: neuroscience and astrophysics. Analyzing their answers, led to the conclusion that their goal on multi-dimensional time series data is to find: similar patterns, abnormal patterns, time length of events, specific times of variability in data (periodicity), and correlation. Moreover, some are interested in working with quick, but rough results (e.g., approximations or incomplete answers), while they wait for the complete and exact answer.

4. DISCUSSION SESSIONS

Finally, we report on the discussion sessions of the workshop, which greatly helped in putting all previous information in perspective, and in identifying the research directions that are useful and promising. Below, we summarize the main points of the discussion, and relevant open research problems.

[5]`https://www.gw-openscience.org/tutorials/`

(1) Preprocessing: In most cases, time series must be preprocessed before being analyzed: this involves selecting the series and intervals of interest, and applying techniques from signal processing (e.g., band filters, denoising), several of which operate in the frequency domain. Thus, time series management systems [12] and analysis workflows should allow analysts to easily extract subsets of interest, and embed their data in different spaces (e.g., time, frequency), suitable for the various analysis techniques.
(2) Analysis Operations: The analysis task itself encompasses several different operations, including similarity search, correlation, clustering, classification, anomaly (or discord) detection, motif (or frequent patterns) discovery, and causal modeling[6]. Similarity search is a key operation that is expensive per se, and if performed fast then it can help speedup several of the other analysis operations, as well. We note that in some cases, the analysis operations need to be performed in a way that takes into account spatial information[7], pointing to the need for the development of corresponding query languages and index structures.
(3) Versatility: Even though there exists a sizable number of studies on time series analysis techniques in the literature (and some of them have found their way into real systems), these techniques are usually not versatile enough for use in the real world. Real applications require scalable techniques that can serve ad-hoc queries and analysis workflows, have the ability to select and operate on sets of sequences selected using complex conditions[8], operate on both entire sequences and subsequences, support the analysis of variable-length subsequences[9], treat value uncertainty[10] [4] as a first class citizen and be able to carry confidence and significance values along the analysis workflow, as well as allow for privacy-preserving analytics.
(4) Interaction with Users: Despite the significant effort of the visualization community in this area [2], most of the available systems are far from scaling to the sizes of time series collections that are used in practice. Recent advances in time series indexing can be of help here, though, novel approaches are required in order to really address the current and future needs. Interactive visualizations are important for users, and when the datasets

[6]For example, Granger causality.
[7]Neuroscientists are interested in correlations among signals recorded by sensors that are spatially close.
[8]Based on metadata, time intervals, value thresholds.
[9]Consider for example that most of the current time series indexes only support fixed-length queries.
[10]In several cases, this uncertainty is inherent in the measurement instrument.

grow very large some of the most promising ways to achieve interactive response times is through the use of fast approximate and/or progressive answers [27], with the support of appropriate visualizations.

[**Summary**] Overall, we observed a slight disconnect between the needs of scientists and practitioners that process and analyze time series, and Computer Science (CS) researchers that work on time series. The problems that CS researchers have been studying are for the most part simplified, clean, and sanitized versions of the real problems and analysis workflows that practitioners have to address in the real world. Despite the remarkable progress in this area by the CS community during the recent years [6, 8, 12, 14, 18, 22], there are still many challenging open problems.

Efficiently supporting similarity search [3, 16, 17, 23,24,28] is still challenging for large data series collections [6,7]. Only very recently attention has been given to solutions that can support variable-length queries [19, 20], and there are still a lot to be done in terms of supporting uncertain series [5]. Scalable visualization solutions are direly needed, especially in support of progressive analytics [10, 27]. At the same time, even some basic problems, such as the interplay between visual perception and similarity measures [9], deserves to be studied in more detail. Evidently, in order to be used in practice, all the above components should be combined in general, easy-to-use by non-experts, time series management systems [6, 12], a task that is by itself a challenge.

5. WORKSHOP CONCLUSIONS

Even though time series are a very common data type, no available system can inherently accommodate and support the dataset sizes and complex analytics required by users. Our discussions showed that applications across different domains share common requirements: fulfilling them is a challenging goal, involving many interesting research problems.

[**Acknowledgements**] The workshops were supported by the CNRS Mastodons TimeClean project.

References

[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *FODO*, 1993.

[2] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data.* Springer, 2011.

[3] G. Chatzigeorgakidis, D. Skoutas, K. Patroumpas, T. Palpanas, S. Athanasiou, and S. Skiadopoulos. Local pair and bundle discovery over co-evolving time series. In *SSTD*, 2019.

[4] M. Dallachiesa, B. Nushi, K. Mirylenka, and T. Palpanas. Uncertain time-series similarity: Return to the basics. *PVLDB*, 5(11), 2012.

[5] M. Dallachiesa, T. Palpanas, and I. F. Ilyas. Top-k nearest neighbor search in uncertain data series. *PVLDB*, 8(1), 2014.

[6] K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim. The Lernaean Hydra of Data Series Similarity Search: An Experimental Evaluation of the State of the Art. *PVLDB*, 12(2), 2018.

[7] K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim. Return of the Lernaean Hydra: Experimental Evaluation of Data Series Approximate Similarity Search. *PVLDB*, 2019.

[8] C. Faloutsos, J. Gasthaus, T. Januschowski, and Y. Wang. Forecasting big time series: Old and new. *PVLDB*, 11(12), 2018.

[9] A. Gogolou, T. Tsandilas, T. Palpanas, and A. Bezerianos. Comparing similarity perception in time series visualizations. *TVCG*, 25(1), 2019.

[10] A. Gogolou, T. Tsandilas, T. Palpanas, and A. Bezerianos. Progressive similarity search on time series data. In *EDBT BigVis Workshop*, 2019.

[11] P. Huijse, P. A. Estévez, P. Protopapas, J. C. Principe, and P. Zegers. Computational intelligence challenges and applications on large-scale astronomical time series databases. *IEEE Comp. Int. Mag.*, 9(3):27–39, 2014.

[12] S. K. Jensen, T. B. Pedersen, and C. Thomsen. Time series management systems: A survey. *TKDE*, 29(11), 2017.

[13] K. Kashino, G. Smith, and H. Murase. Time-series active search for quick retrieval of audio and video. In *ICASSP*, 1999.

[14] E. J. Keogh. Indexing and mining time series data. In *Encyclopedia of GIS.*, pages 933–939. 2017.

[15] E. J. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD*, 2001.

[16] H. Kondylakis, N. Dayan, K. Zoumpatianos, and T. Palpanas. Coconut Palm: Static and Streaming Data Series Exploration Now in your Palm. In *SIGMOD*, 2019.

[17] H. Kondylakis, N. Dayan, K. Zoumpatianos, and T. Palpanas. Coconut: Sortable Summarizations for Scalable Indexes over Static and Streaming Data Series. *VLDBJ*, accepted for publication, 2019.

[18] J. Large, P. Southam, and A. J. Bagnall. Can automated smoothing significantly improve benchmark time series classification algorithms? *CoRR*, abs/1811.00894, 2018.

[19] M. Linardi and T. Palpanas. Scalable, variable-length similarity search in data series: The ULISSE approach. *PVLDB*, 11(13), 2018.

[20] M. Linardi, Y. Zhu, T. Palpanas, and E. J. Keogh. Matrix profile X: VALMOD - scalable discovery of variable-length motifs in data series. In *SIGMOD*, 2018.

[21] T. Palpanas. Data series management: The road to big sequence analytics. *SIGMOD Record*, 44(2), 2015.

[22] T. Palpanas. Big sequence management: A glimpse of the past, the present, and the future. In *SOFSEM*, 2016.

[23] B. Peng, P. Fatourou, and T. Palpanas. Paris: The next destination for fast data series indexing and query answering. In *IEEE BigData*, 2018.

[24] B. Peng, P. Fatourou, and T. Palpanas. MESSI: In-Memory Data Series Indexing. In *ICDE*, 2020.

[25] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco. Practical data prediction for real-world wireless sensor networks. *TKDE 27(8)*, 2015.

[26] D. Shasha. Tuning time series queries in finance: Case studies and recommendations. *DEBull*, 22(2), 1999.

[27] C. Turkay, N. Pezzotti, C. Binnig, H. Strobelt, B. Hammer, D. A. Keim, J. Fekete, T. Palpanas, Y. Wang, and F. Rusu. Progressive data science: Potential and challenges. *CoRR*, abs/1812.08032, 2018.

[28] D.-E. Yagoubi, R. Akbarinia, F. Masseglia, and T. Palpanas. Massively distributed time series indexing and querying. *TKDE (to appear)*, 2019.

[29] L. Ye and E. J. Keogh. Time series shapelets: a new primitive for data mining. In *KDD*, 2009.

[30] K. Zoumpatianos, S. Idreos, and T. Palpanas. ADS: the adaptive data series index. *VLDBJ*, 25(6), 2016.

[31] K. Zoumpatianos and T. Palpanas. Data series management: Fulfilling the need for big sequence analytics. In *ICDE*, 2018.

Report on the First International Workshop on Semantic Web Technologies for Health Data Management (SWH 2018)

Haridimos Kondylakis
ICS-FORTH
Heraklion, Greece
kondylak@ics.forth.gr

Kostas Stefanidis
Tampere University
Tampere, Finland
kostas.stefanidis@uta.fi

Praveen Rao
University of Missouri-Kansas
Kansas City, USA
raopr@umkc.edu

ABSTRACT

Better information management is the key to a more intelligent health and social system. To this direction, many challenges must be first overcome, enabling seamless, effective and efficient access to various health data sets and novel methods for exploiting the available information. The First International Workshop on Semantic Web Technologies for Health Data Management aimed at bringing together an interdisciplinary audience interested in the fields of semantic web, data management and health informatics to discuss the challenges in health-care data management and to propose novel and practical solutions for the next generation data-driven health-care systems. In this paper, we summarize the outcomes of the first instance of the workshop, and we present interesting conclusions and key messages.

1. INTRODUCTION

Key in achieving the vision of affordable, less intrusive and more personalized care, is the efficient and effective exploitation of health data. Ultimately this has the potential to increase the quality of life as well as to lower mortality. However, the lifelong patient data to be exploited for this purpose are complex, with hundreds of attributes per patient record, that will continually evolve as new types of calculations and analysis/assessment results are added to these records over time. In addition, data exist in many different formats, from textual documents and web tables to well-defined relational data and APIs. Furthermore, they pertain to ambiguous semantics and quality standards resulted from different collection processes across sites. The vast amount of data generated and collected comes in so many different streams and forms from physician notes, personal health records, images from patient scans, health conversations in social media, to continuous streaming information collected from wearables and other monitoring devices.

To this direction, semantic technologies can provide effective solutions for enabling interoperability and common language among health systems, and can lead to the disambiguation of the information through the adoption of various terminologies and ontologies available. On the other hand, cloud-based technologies and micro-services are the key for large-scale health systems deployment, data storage and analysis. The goal of this workshop is to bring together researchers cross-cutting the fields of semantic web, data management and health informatics to discuss the challenges in health-care data management and to propose novel and practical solutions for the next generation of data-driven health-care systems. Developing optimal frameworks for integrating, curating and sharing large volumes of Health Record data has the potential for a tremendous impact on health-care, enabling better outcomes at a lower cost.

Next, we summarize the outcomes of the first workshop instance held in conjunction with ISWC 2018 in Monterey, USA.

2. INVITED TALK

2.1 Benchmarking Big Linked Data: The case of the HOBBIT Project

Irini Fundulaki, in her keynote talk, discussed the results of the European H2020 HOBBIT (Holistic Benchmarking for Big Linked Data) Project. More specifically, she talked about the benchmarks developed in the context of the project, that target all the steps of the Big Data Value Chain. Special focus was given on the following two benchmarks: the link discovery benchmarks that aim at testing link discovery systems which address one of the most important problems of data integration, and the versioning benchmark that can be used to check the performance of systems that manage versions of linked datasets.

The HOBBIT project worked towards providing innovative benchmarks with the following characteristics:

- Realistic benchmarks: Benchmarks are commonly generated with synthetic data that reflect a single and specific domain. HOBBIT created mimicking algorithms to generate synthetic data from different domains.
- Universal benchmarking platform: HOBBIT developed a generic platform that is able to execute large scale benchmarks across the Linked Data lifecycle. The platform provides reference implementations, as well as dereferenceable results and automatic feedback to tools developers.
- Industry relevant Key Performance Indicators (KPIs): In addition to the classical KPIs developed over the last decades, HOBBIT collected relevant KPIs from industry to assess technologies based on the industrial needs.

Finally, Irini Fundulaki discussed what are the benchmarks that are of interest for health-care systems and applications, and provided some vision as to what are the lines of research that we should pursue in order to be able to have good quality benchmarks for such critical systems.

3. PAPER PRESENTATIONS

3.1 An Ontology-Driven Elderly People Home Mobilization Approach

Proposing exercise games to elderly is a challenging area of research. Age-related changes in balance, gait, strength, visual and hearing senses, memory and attention, and their deterioration over time make it difficult to assess individual status and adapt appropriately the corresponding recommendations. The authors in [5] propose an intelligent agent, that automatically and continuously adapts to the user profile, and provides corresponding incentives for mobilization at home. In order to construct the user profile, the agent incrementally builds a knowledge base capturing behavioral characteristics and movement sequences. The tracking of the users is realized by 3D sensors, which capture individual tracks throughout the day. Processing those tracks, information regarding *active time in room*, *active time of day*, *average gait velocity*, *average stand up time* and *average walking time* is calculated and stored in the knowledge base. The information in the knowledge base is modeled using an ontology. Instances of this ontology match the condition of the available rules for providing personalized recommendations. Those rules are in essence SPARQL queries, which propose personalized recommendations regarding games which include walking exercises or mind games. REST APIs implemented on top provide CRUD functionality on the available data and expose the movement sequences processed by other components of the agent. The evaluation performed confirm the effectiveness and efficiency of the approach.

3.2 Integrating clinical data from hospital databases

Research in various fields of medicine often requires the process and analysis of large amounts of possibly heterogeneous data that appear in different sources, like hospitals or scientific laboratories. By integrating such data, researchers extract new knowledge related to their field of study, that are not able to obtain when working with each data source independently. In general, the goal of data integration is to provide a uniform access over a set of data sources that have been created and stored autonomously [4, 3].

To fully exploit the integrated clinical data, it is important to be able to reveal the semantic relationships among them. For example, as stated in [6], to translate patients data describing dementia symptoms into effective Alzheimer's disease diagnosis, it is important that these data are related to additional patients information, such as genetic data, as well as to biological markers, such as proteins and electroencephalography. Specifically, the authors are interested in integrating clinical data related to the human brain. This work has been developed to meet the needs of the Medical Informatics Platform (MIP) of the Human Brain Project (HBP) that aims to develop technologies that enhance the scientific research related to human brain. Towards this effort, MIP provides a data integration mechanism to collect clinical data, such as Electronic Health Records (EHR) and imaging features stored in hospitals local databases.

3.3 Knowledge Engineering Framework to Quantify Dependencies between Epidemiological and Biomolecular Factors in Breast Cancer

The relationship between social determinants of health and chronic disease risks is crucial for the prevention of chronic diseases. Such associations are relatively easier to uncover for simple diseases, like obesity. But for complex diagnoses like cancer, a large number of factors contribute to the onset of the disease. Cancer Registries as the source of health data are used widely in epidemiological

research. Being collected by health professionals, they reduce research costs and embrace the whole population. However, the primary purpose of those sources is not being used for research, e.g., many times the structure of records is not appropriate in order to build an epidemiological model. Therefore, a data adjusting issue arises. To fit data from Cancer Registries to the epidemiological model, the authors create a knowledge engineering framework utilizing controlled vocabularies, using Bayesian Networks to quantify and predict factors that influence hormonal patterns of breast cancer, which can lead to better patient care.

3.4 The FairGRecs Dataset

FairGRecs is a synthetic dataset that can be used for evaluating and benchmarking methods [8] that produce recommendations related to health documents based on individual health records. Specifically, FairGRecs can create, via a fully parametrized API, synthetic patients profiles, containing the same characteristics that exist in a real medical database, including both information about health problems and also relevant documents. More specifically, [10] relies on the EMRBots dataset[1], which contains synthetic patients profiles, containing the same characteristics that exist in a real medical database, such as patients admission details, demographics, socioeconomic details, labs and medications, extending it with a document corpus and a rating dataset. By exploiting the FairGRecs dataset, interested users can create patients that have provided rankings for health documents. To link document contents with patients, the authors use the ICD10[2] ontology, namely the International Statistical Classification of Diseases and Related Health Problems, which is a standard medical classification list maintained by the World Health Organization. FairGRecs is fully parametrized, is offered via an API, and has been used already in [11].

3.5 Towards the Development of a National eHealth Interoperability Framework to Address Public Health Challenges in Greece

Large amounts of health data are daily generated and stored in regional health systems across Europe. Opening and reusing these data can be the key for improving healthcare efficiency and effectiveness. As such, the development of national interoperability frameworks (NIF) is essential and towards this direction the EU has announced guidelines for a European interoperability framework (EIF) [1], including 47 concrete recommendations for legal, organizational, semantic and technical interoperability. In Greece, healthcare is provided by the national health system with multiple of services already available such as ePrescription, eReferral for primary care, eConfirmation for insurance status verification, eReimbursement, eAppointments for doctors in the primary care etc. [7]. The Greek national health system has recognized the importance of implementing a NIF. The prerequisites for enabling data reuse include a well-defined process model, available and agreed terminology and reliable clinical content.

3.6 The Case for Designing Data-Intensive Cloud-Based Healthcare Applications

Cloud computing is a major source of revenue for companies like Amazon, Google, and Microsoft. Today, it is attracting a lot of interest among the healthcare community due to its benefits such a lower cost and ease of deployment. [2] argues for a microservices-based architecture for designing data-intensive cloud-based healthcare applications. A healthcare vendor may consider moving an existing software application by deploying them through virtual machines (VMs). However, VMs are heavyweight and are not suited for rapid deployment and recovery. Rethinking the design of the software application using microservices will provide the ability to quickly scale, be fault-tolerant, and provide high levels of security and availability. Microservices allow an application to be composed of loosely coupled services. Kubernetes[3] is a popular framework for orchestrating microservices. The authors propose a generic architecture for designing healthcare applications using microservices by decoupling storage, compute, and non-functional requirements such as availability, security, scalability, capacity planning, and others. Services for data analytics and machine learning can be incorporated using the software-as-a-service model.

3.7 A De-centralized Framework for Data Sharing, Ontology Matching and Distributed Analytics

The HarmonicSS platform [9] is a decentralized platform with the target to address all the aforementioned needs, tackling appropriately all ethical, legal and privacy issues for data sharing. The data sharing framework includes the data assessment and the data sharing management modules, ensuring that the framework respects all General Data Pro-

[1]http://www.emrbots.org
[2]http://www.icd10data.com/
[3]https://kubernetes.io

tection Regulation requirements for both the data providers and data processors. The clinical data are stored on a private cloud, specifically designed for each cohort, whereas a data curation module performs data cleaning. For data harmonization, an ontology has been defined and ontology matching is used for mapping the schema of each individual cohort to the ontology.

The proposed architecture is used to integrate and harmonize 7500 records out of 22 cohorts, including a variety of patients with primary Sjogren syndrome. A data processor who wishes to process cohort data has first to request access. Then the corresponding data providers can allow or deny data access. Data analytic services on top are then executed locally on the private cloud and the results are combined in a distributed learning fashion ensuring that the data never leave the clinical center.

4. WORKSHOP CONCLUSIONS

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

- Semantic technology can leverage behavioral characteristics into personalized recommendations. In this direction, user context and interactivity need to be emphasized.
- Personal health data can be leveraged for exploring the past and personalizing the user experience. Personal data exploration even in the health domain can take into account psychological and behavioral patterns to build novel exploration paradigms.
- System performance, in particular response time experienced by the user, remains a major challenge in the domain of health data management.
- Given the growing interest among the healthcare community to adopt cloud services for large-scale health data management, microservices come to the foreground, holding the promise for designing the next generation of data-intensive health-care applications.
- Researchers can employ different clinical terminologies (e.g., ICD10, SNOMED CT) for building knowledge/data management solutions for healthcare data based on the country of deployment.

This first instance of the Semantic Web Technologies for Health Data Management Workshop made clear that a lot of research work still needs to be done in the general area of semantic health data management. Given the growing interest in industry and academia, we are looking forward to the next instance of this workshop.

5. REFERENCES

[1] The new european interoperability framework. `https://ec.europa.eu/isa2/eif_en`. Accessed: 2018-10-30.

[2] S. Bhagavan, K. Alsultan, and P. Rao. The case for designing data-intensive cloud-based healthcare applications. In *SWH*, 2018.

[3] V. Christophides, V. Efthymiou, and K. Stefanidis. *Entity Resolution in the Web of Data*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, 2015.

[4] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[5] S. Karagiorgou, D. Ntalaperas, G. Vafeiadis, D. Alexandrou, K. Perakis, D. Baltas, C. Amza, A. Wanka, H. Freitag, M. Blok, M. Kampel, V. de Rond, T. Münzer, and R. Planinc. An ontology-driven elderly people home mobilization approach. In *SWH*, 2018.

[6] K. Karozos, I. Spartalis, A. Tsikiridis, D. Trivela, and V. Vassalos. Integrating clinical data from hospital databases. In *SWH*, 2018.

[7] D. G. Katehakis, A. Kouroubali, and I. Fundulaki. Towards the development of a national ehealth interoperability framework to address public health challenges in greece. In *SWH*, 2018.

[8] H. Kondylakis, L. Koumakis, E. Kazantzaki, M. Chatzimina, M. Psaraki, K. Marias, and M. Tsiknakis. Patient empowerment through personal medical recommendations. In *MEDINFO 2015*, page 1117, 2015.

[9] V. C. Pezoulas, K. D. Kourou, T. P. Exarchos, V. Andronikou, T. A. Varvarigou, A. G. Tzioufas, S. de Vita, and D. I. Fotiadis. A de-centralized framework for data sharing, ontology matching and distributed analytics. In *SWH*, 2018.

[10] M. Stratigi, H. Kondylakis, and K. Stefanidis. The fairgrecs dataset: A dataset for producing health-related recommendations. In *SWH*, 2018.

[11] M. Stratigi, H. Kondylakis, and K. Stefanidis. Fairgrecs: Fair group recommendations by exploiting personal health information. In *DEXA*, 2018.