SIGMOD Officers, Committees, and Awardees

Chair	Vice-Chair	Secretary/Treasurer
Donald Kossmann Systems Group ETH Zürich Cab F 73 8092 Zuerich SWITZERLAND +41 44 632 29 40 <donaldk AT inf.ethz.ch>	Anastasia Ailamaki School of Computer and Communication Sciences, EPFL EPFL/IC/IIF/DIAS Station 14, CH-1015 Lausanne SWITZERLAND +41 21 693 75 64 <natassa AT epfl.ch>	Magdalena Balazinska Computer Science & Engineering University of Washington Box 352350 Seattle, WA USA +1 206-616-1069 <magda AT cs.washington.edu>

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

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

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

Associate Information Directors:
Manfred Jeusfeld, Georgia Koutrika, Wim Martens, Mirella Moro

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

SIGMOD Record Associate Editors:
Pablo Barceló, Vanessa Braganholo, Marco Brambilla, Chee Yong Chan, Rada Chirkova, Anastasios Kementsietsidis, Olga Papaemmanouil, Aditya Parameswaran, Anish Das Sarma, Alkis Simitsis, Nesime Tatbul, Marianne Winslett, and Jun Yang.

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

PODS Executive Committee:
Tova Milo (Chair), Diego Calvanse, Wenfei Fan, Martin Grohe, Rick Hull, Maurizio Lenzerini

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

Awards Committee:
Elisa Bertino (Chair), Surajit Chaudhuri, Maurizio Lenzerini, Kartin Kersten, Umesh Dayal

Jim Gray Doctoral Dissertation Award Committee:
Tova Milo (Co-Chair), Juliana Freire (Co-Chair), Ashraf Aboulnaga, Minos Garofalakis, Chris Jermaine, Renee Miller, Aditya Parameswaran, Andy Pavlo, Kian-Lee Tan.

SIGMOD Officers, Committees, and Awardees (continued)

SIGMOD Edgar F. Codd Innovations Award

For innovative and highly significant contributions of enduring value to the development, understanding, or use of database systems and databases. Formerly known as the "SIGMOD Innovations Award", it now honors Dr. E. F. (Ted) Codd (1923 - 2003) who invented the relational data model and was responsible for the significant development of the database field as a scientific discipline. Recipients of the award are the following:

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

SIGMOD Contributions Award

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

Maria Zemankova (1992)	Gio Wiederhold (1995)	Yahiko Kambayashi (1995)
Jeffrey Ullman (1996)	Avi Silberschatz (1997)	Won Kim (1998)
Raghu Ramakrishnan (1999)	Michael Carey (2000)	Laura Haas (2000)
Daniel Rosenkrantz (2001)	Richard Snodgrass (2002)	Michael Ley (2003)
Surajit Chaudhuri (2004)	Hongjun Lu (2005)	Tamer Özsu (2006)
Hans-Jörg Schek (2007)	Klaus R. Dittrich (2008)	Beng Chin Ooi (2009)
David Lomet (2010)	Gerhard Weikum (2011)	Marianne Winslett (2012)
H.V. Jagadish (2013)	Kyu-Young Whang (2014)	Curtis Dyreson (2015)

SIGMOD Jim Gray Doctoral Dissertation Award

SIGMOD has established the annual SIGMOD Jim Gray Doctoral Dissertation Award to *recognize excellent research by doctoral candidates in the database field.* Recipients of the award are the following:

- **2006** *Winner*: Gerome Miklau, University of Washington. *Honorable Mentions*: Marcelo Arenas and Yanlei Diao.
- **2007** *Winner*: Boon Thau Loo, University of California at Berkeley. *Honorable Mentions*: Xifeng Yan and Martin Theobald.
- **2008** *Winner*: Ariel Fuxman, University of Toronto. *Honorable Mentions*: Cong Yu and Nilesh Dalvi.
- **2009** *Winner*: Daniel Abadi, MIT. *Honorable Mentions*: Bee-Chung Chen and Ashwin Machanavajjhala.
- **2010** *Winner:* Christopher Ré, University of Washington. *Honorable Mentions*: Soumyadeb Mitra and Fabian Suchanek.
- **2011** *Winner*: Stratos Idreos, Centrum Wiskunde & Informatica. *Honorable Mentions*: Todd Green and Karl Schnaitterz.
- **2012** *Winner*: Ryan Johnson, Carnegie Mellon University. *Honorable Mention*: Bogdan Alexe.
- **2013** *Winner*: Sudipto Das, University of California, Santa Barbara. *Honorable Mention*: Herodotos Herodotou and Wenchao Zhou.
- **2014** *Winners*: Aditya Parameswaran, Stanford University, and Andy Pavlo, Brown University.
- **2015** *Winners*: Alexander Thomson, Yale University. *Honorable Mentions*: Marina Drosou, University of Ioannina and Karthik Ramachandra, IIT Bombay

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

[Last updated : June 30, 2015]

Editor's Notes

Welcome to the September 2015 issue of the ACM SIGMOD Record!

This issue opens with a letter from the SIGMOD Executive Committee, summarizing some of the major changes implemented by the SIGMOD Executive Committee and other SIGMOD news of the last two years. The major changes range from the revisions of the SIGMOD conference, to the new special issues of the SIGMOD Record, to the development of a new "sigmod.org" website, and to the new rewards in the SIGMOD community.

This issue continues with a Database Principles article by Wenfei Fan, which provides an overview of recent advances in the study of data quality, from theory to practice. The fundamental research questions covered in this article include data consistency, data deduplication, information completeness, data currency, and data accuracy. Following the theoretical questions, the article surveys a range of practical techniques for profiling (discovery of data quality rules), cleaning (error detection and data repairing), and matching (data deduplication). The article closes by pointing out challenges introduced by big data to data quality management.

The Research and Vision Articles Column features a vision article, by Gawlick et al., on "Mastering Situation Awareness". This article is motivated by the observation that in applications such as cloud operation and customer care, situation awareness requires a system to support the management of data, knowledge, processes, and other services such as social networking in an integrated way. To this end, the authors propose a model, called KIDS (Knowledge Intensive Data-processing System), which enables the development and maintenance of situation-aware applications in a declarative and therefore economical manner. The article also outlines a list of opportunities and challenges for the database community, including a theoretical foundation based on category theory, time dimension support, performance and scalability.

The Surveys Column features two articles. The first survey, by Yin et al., examines "Robust Query Optimization Methods With Respect to Estimation Errors." Query optimization has been a key component of a database management system (DBMS) since the design of the System-R optimizer. Effective query optimization, however, depends on accurate estimation of data and query related statistics, which remains as a technical challenge. This survey particularly focuses on a recent notion of "robust query optimization", which aims to find a plan (or several plans) whose query execution time differs from that of an optimal plan by only a small fraction. Toward this goal, the article surveys a wide range of techniques, including cardinality injection, dynamic re-optimization, deferred plan selection, tuple routing, etc., under a set of criteria for comparison. As such, the article provides a good introduction to the topic of robust query optimization and characterizes a large set of recent techniques developed for this purpose.

The second survey by Kim and Lee focuses on "Community Detection in Multi-Layer Graphs". The goal of community detection (graph clustering) is to partition vertices in a complex graph into densely-connected components, called communities. In recent applications, however, an entity is often associated with multiple aspects of relationships—capturing such multiple aspects of interactions requires multi-layer graph clustering, where each graph represents an aspect of the interactions. This article presents a large number of multi-layer graph datasets used in recent studies, surveys and compares recent relevant techniques, and suggests future directions for research.

The Distinguished Profiles column features Rick Cattell. Rick spent over 20 years at Sun Microsystems, where he was involved in many technology innovations that we take for granted today, such as ODBC, JDBC, and J2EE. Rick was one of Sun's first Distinguished Engineers, and his dissertation on compiler technology won the ACM Dissertation Award. In this interview, Rick shared his experiences and thoughts regarding software patents, standards, and performance, scalability, and availability of database systems.

This issue includes three event reports. The first article, by Pedersen, reports on the International Workshop on Energy Data Management (EnDM 2014), co-located with EDBT/ICDE 2014. The workshop, targeting at novel schemes for large-scale energy data processing, featured 5 research papers on modeling, semantics, and analytics of energy data. The second article, by Chen et al., reports on the International Workshop on Big Data Management on Emerging Hardware (HardDB 2015), co-located with ICDE 2015. The workshop included a keynote speech and four research papers that exploit new hardware trends, such as multi-core CPU and GPU clusters, for big data analytics. The third article, by Castellanos et al., reports on the International Workshop on Business Intelligence for the Real-time Enterprise (BIRTE 2014), co-located with VLDB 2014. The workshop attracted a large audience with a full program including a keynote speech by C. Mohan from IBM Research, two industrial talks from Microsoft and HP Labs, respectively, and four research papers on various aspects of real-time business analytics.

Finally, this issue closes with the call for participation for ICDE 2016, to be held in Helsinki, Finland in May 2016.

On behalf of the SIGMOD Record Editorial board, I hope that you all enjoy reading the September 2015 issue of the SIGMOD Record!

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

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

Yanlei Diao

September 2015

Past SIGMOD Record Editors:

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)
Mario Nascimento (2005–2007)
Jennifer Widom (1995–1996)
Jon D. Clark (1984–1985)
Randall Rustin (1974-1975)

Letter from the SIGMOD Executive Committee

Dear SIGMOD members:

We are now half way through our four-year term as elected SIGMOD officers and SIGMOD Executive Committee. In these first two years, SIGMOD has continued to thrive. We continue to be a very healthy community with many new people, new scientific challenges, and new ideas. We are living in a golden age of data management research in many ways, and it seems that things will get even better. Some of the success is just good fortune, but most of it is the result of the efforts of the people in this community, for which we would like to thank all of you.

As SIGMOD Executive Committee, our primary goal is to stay out of your way and silently make sure that all the organizational issues of our community are taken care of. So, we want to change as little as possible. Nevertheless, there is always room for improvement, and we continue to be eager to hear about your ideas and problems. Some ideas are simple and can be implemented in a straightforward way. An example is to relax the 12-page limit of SIGMOD papers to accommodate more references and a more detailed discussion of related work. This idea was brought to our attention at SIGMOD 2014 in Snowbird, became implemented at SIGMOD 2015, and has now become a standard. Some ideas, however, are more disruptive and require a deeper involvement of the community. This letter describes some of these changes and other SIGMOD news of the last two years.

Conference: Starting in 2016, the SIGMOD Conference will have only two parallel tracks, plus one track for PODS. In comparison, SIGMOD 2015 had up to five parallel sessions, plus PODS. The goal of this change is to have more lively discussions in the sessions, increase attendance at the talks, and create a greater sense of a shared experience at the conference. We hope that this change will be particularly beneficial to students and young researchers because they will be getting a much larger stage than before.

Before implementing this change, we had discussions with many of you. The feedback has been overwhelmingly positive and that is why we are moving forward with the plan. Unfortunately, there is also a cost associated with this change. In order to make room, the regular research talks will be shorter. For detailed discussions, each paper will also present a poster and we will put much more emphasis on the poster sessions. Furthermore, there will be a change in the tutorial program: All SIGMOD tutorials will be presented on Friday, rather than Tuesday to Thursday as in the past. (PODS tutorials will continue to be part of the regular PODS program.) There are also a number of smaller changes aimed at tightening the program and improving the conference experience, but we speculate that you will not even notice them.

We hope that we will be able to implement these changes as seamlessly as possible and that you will enjoy the more intense experience during the sessions. Nevertheless, it is likely that this transition will not be perfectly smooth. Please, do let us know if you notice anything during SIGMOD 2016 and have ideas for improvements so that we can learn and improve for SIGMOD 2017 and the following years.

SIGMOD Record: As you may have already noticed, there have been a number of exciting changes with regard to the SIGMOD Record. The September 2015 issue of the SIGMOD Record was a special issue on "Vision Papers". The response to the call for papers was overwhelming and we were thrilled to produce a very strong first edition of this special issue. In the long run, we plan to have such special issues every two years, alternating with the CIDR conference for publishing visionary ideas in data management.

Another exciting change is the introduction of a new award, the "SIGMOD Research Highlight Award". Every year, the authors of the best papers of our major conferences (and journals in the long run) will be invited to prepare a version of the paper for a broader audience. These papers will be published in another special issue of the SIGMOD Record. We expect to publish about a dozen such papers every year and hope that people from other communities will find this a useful resource. Furthermore, we will forward the best of the best for consideration as a Research Highlight in Communications of the ACM.

A welcome side effect of these special issues is that it will become more competitive to publish a regular paper in the SIGMOD Record. The SIGMOD Record editorial team is currently revising the reviewing process and will announce changes as soon as they have been implemented. The other columns of SIGMOD Record (e.g., distinguished profiles, database principles, surveys, etc.), however, will not change: They have been great and we will all continue to enjoy them in pretty much the same way as before.

TODS: A number of ideas are being discussed and developed at TODS. One recent change is that the authors of an "original" TODS paper, which is not an extension of a previously published conference paper, are now invited to present their paper as a poster at the SIGMOD Conference. For details, see the June 2015 issue of the SIGMOD Record. We will keep you posted by means of the TODS column in the SIGMOD Record as other ideas take form.

Website: The SIGMOD website is currently undergoing a major revamp. The goal is to put more content online and to have fresher and more dynamic content. In the future, we should all be checking "sigmod.org" first thing in the morning. Please, stay tuned.

Awards: This year has been a particularly exciting year for SIGMOD in terms of awards. First, we have been extremely successful in winning awards at the ACM level. Jennifer Widom won the ACM Athena Lecture Award; this is the first time that somebody from our community won this extremely prestigious award. We continue to be super-successful with our nominations of ACM Fellows. Last but not least, Mike Stonebraker won this year's Turing Award.

Within our community, the biggest news is that we established the "SIGMOD Systems Award" and announced the first winner at SIGMOD 2015 in Melbourne: The winners were Mike Stonebraker and Larry Rowe for Postgres. Another important change is that we raised the prize for many awards. The Codd Innovation Award and the SIGMOD Systems Award now each come with a prize of 10,000 USD. The prize for the SIGMOD Systems Award is sponsored by Microsoft. The prize for the Codd Innovation Award is partly supported by a personal gift from Mike Stonebraker. We are very grateful for these sponsorships and gifts.

We would like to congratulate all award winners. Their achievements are fantastic and they reflect back on the whole community so that we all benefit from them. What might be less obvious is that these awards are the result of hard work of many people in the community. First, serving in an award committee is very hard work. There are too many volunteers to list them all, but as representatives we would like to thank Elisa Bertino, David DeWitt, Tova Milo, and Juliana Freire for chairing the three SIGMOD Award committees in 2015.

Second, and may be even more important, there are no awards without nominations. We need to share this task and everybody in the community should think at least twice a year about whom to nominate for which kind of award (within SIGMOD and beyond SIGMOD). Even though we have been extraordinarily successful recently, maintaining this success requires that more people become involved in nominations. We would like to particularly encourage you to nominate women for awards: While we are lucky to have so many women doing great work in our community, these women are not getting their fair share in terms of awards. Even if your nomination is not successful, this exercise of thinking about other people in the community and nominating them for awards is extremely important for the health of the community. It helps build up respect and better put the contributions of our community into perspective.

Executive Committee: There have been a number of changes in the SIGMOD Executive Committee. After six years as TODS Editor in Chief, Meral Ozsoyogul retired from this role and Christian Jensen took over, including Meral's seat in the SIGMOD Executive Committee. After four years as SIGMOD Record Editor in Chief, Ioana Manolescu retired and Yanlei Diao took over. Furthermore, Tova Milo is now the PODS representative in the SIGMOD Executive Committee, replacing Rick Hull in this role. We would like to thank all past members for their hard work and dedication and welcome the new members.

We have also established a new SIGMOD Advisory Board that is significantly larger and more diverse than the old SIGMOD Advisory Board. We are discussing strategic initiatives such as the SIGMOD Conference format with that Advisory Board.

Last but not least, we would like to thank the many volunteers who have made our lives so enjoyable and who are really the heart of this community by running the conferences, journals & publications, and other services (e.g., DBjobs, DBLP, etc.). Without this volunteer work, there would be no "SIGMOD and VLDB" community. We all do a bit of this volunteer work every day, e.g., by reviewing papers, writing letters, making nominations for awards, providing feedback to discussions, etc. Please, continue to do all this work and stay involved.

Donald Kossmann, Anastasia Ailamaki, Magda Balazinska, K. Selçuk Candan, Yanlei Diao, Curtis Dyreson, Yannis Ioannidis, Christian Jensen, Tova Milo, Fran Spinola.

Data Quality: From Theory to Practice

Wenfei Fan
School of Informatics, University of Edinburgh, and RCBD, Beihang University
wenfei@inf.ed.ac.uk

ABSTRACT

Data quantity and data quality, like two sides of a coin, are equally important to data management. This paper provides an overview of recent advances in the study of data quality, from theory to practice. We also address challenges introduced by big data to data quality management.

1. INTRODUCTION

When we talk about big data, we typically emphasize the quantity (volume) of the data. We often focus on techniques that allow us to efficiently store, manage and query the data. For example, there has been a host of work on developing scalable algorithms that, given a query Q and a dataset D, compute query answers $Q(D)$ when D is big.

But can we trust $Q(D)$ as correct query answers?

EXAMPLE 1. *In table D_0 of Fig. 1, each tuple specifies the name (*FN, LN*), phone (country code* CC*, area code* AC*,* landline*,* mobile*), address (*street*,* city *and* zip*), and marital* status *of an employee. Consider the following queries.*

(1) Query Q_1 is to find distinct employees in Edinburgh whose first name is Mary. A textbook answer to Q_1 in D_0 is that $Q_1(D_0)$ consists of tuples t_2 and t_3.

*However, there are at least three reasons that discredit our trust in $Q_1(D_0)$. (a) In tuple t_1, attribute t_1[AC] is 131, which is the area code of Edinburgh, not of London. Hence t_1 is "inconsistent", and t_1[city] may actually be Edinburgh. (b) Tuples t_2 and t_3 may refer to the same person, i.e., they may not be "*distinct*". (c) Relation D_0 may be incomplete: there are possibly employees in Edinburgh whose records are not included in D_0. In light of these, we do not know whether $Q_1(D_0)$ gives us all correct answers.*

(2) Suppose that t_1,t_2 and t_3 refer to the same Mary, and that they were once correct records (except the address of t_1). Query Q_2 is to find her current last name. It is not clear whether the answer is Smith or Luth. Indeed, some attributes of t_1, t_2 and t_3 have become obsolete and thus inaccurate. □

The example shows that if the quality of the data is bad, we cannot find correct query answers no matter how scalable and efficient our query evaluation algorithms are.

Unfortunately, real-life data is often dirty: inconsistent, inaccurate, incomplete, obsolete and duplicated. Indeed, "more than 25% of critical data in the world's top companies is flawed" [53], and "pieces of information perceived as being needed for clinical decisions were missing from 13.6% to 81% of the time" [76]. It is also estimated that "2% of records in a customer file become obsolete in one month" [31] and hence, in a customer database, 50% of its records may be obsolete and inaccurate within two years.

Dirty data is costly. Statistics shows that "bad data or poor data quality costs US businesses $600 billion annually" [31], "poor data can cost businesses 20%-35% of their operating revenue" [92], and that "poor data across businesses and the government costs the US economy $3.1 trillion a year" [92]. Worse still, when it comes to big data, the scale of the data quality problem is historically unprecedented.

These suggest that quantity and quality are equally important to big data, *i.e., big data = data quantity + data quality*.

This paper aims to provide an overview of recent advances in the study of data quality, from fundamental research (Section 2) to practical techniques (Section 3). It also identifies challenges introduced by big data to data quality management (Section 4). Due to the space constraint, this is by no means a comprehensive survey. We opt for breadth rather than depth in the presentation. Nonetheless, we hope that the paper will incite interest in the study of data quality management for big data. We refer the interested reader to recent surveys on the subject [7, 11, 37, 52, 62, 78].

	FN	LN	CC	AC	landline	mobile	street	city	zip	status
t_1:	Mary	Smith	44	131	3855662	7966899	5 Crichton	London	W1B 1JL	single
t_2:	Mary	Luth	44	131	*null*	*null*	10 King's Road	Edinburgh	EH4 8LE	married
t_3:	Mary	Luth	44	131	6513877	7966899	8 Mayfield	Edinburgh	EH4 8LE	married
t_4:	Bob	Webber	01	908	6512845	3393756	PO Box 212	Murray Hill	NJ 07974	single
t_5:	Robert	Webber	01	908	6512845	*null*	9 Elm St.	Murray Hill	NJ 07974	single

Figure 1: An employee dataset D_0

2. FOUNDATIONS OF DATA QUALITY

Central to data quality are data consistency, data deduplication, information completeness, data currency and data accuracy. The study of data quality has been mostly focusing on data consistency and deduplication in relational data. Nonetheless, each and every of the five central issues introduces fundamental problems. In this section we survey fundamental research on these issues. We highlight dependency-based approaches since they may yield a uniform logical framework to handle these issues.

2.1 Data Consistency

Data consistency refers to the validity and integrity of data representing real-world entities. It aims to detect errors (inconsistencies and conflicts) in the data, typically identified as violations of *data dependencies* (integrity constraints). It is also to help us *repair* the data by fixing the errors.

There are at least two questions associated with data consistency. What data dependencies should we use to detect errors? What repair model do we adopt to fix the errors?

Data dependencies. Several classes of data dependencies have been studied as data quality rules, including

- functional dependencies (FDs) and inclusion dependencies (INDs) [14, 23] found in textbooks (*e.g.,* [1]);
- conditional functional dependencies (CFDs) [38] and conditional inclusion dependencies (CINDs) [15], which extend FDs and INDs, respectively, with a pattern tableau of semantically related constants;
- denial constraints (DCs) [8, 23], which are universally quantified first-order logic (FO) sentences of the form $\forall \bar{x} \neg(\phi(\bar{x}) \wedge \beta(\bar{x}))$, where $\phi(\bar{x})$ is a non-empty conjunction of relation atoms over $\bar{x}$, and $\beta(\bar{x})$ is a conjunction of built-in predicates $=, \neq, <, >, \leq, \geq$;
- equality-generating dependencies [2] (EGDs [9]), a special case of DCs when $\beta(\bar{x})$ is of the form $x_i = x_j$; our familiar FDs are a special case of EGDs;
- tuple-generating dependencies [2] (TGDs [9]), FO sentences of the form $\forall \bar{x}(\phi(\bar{x}) \rightarrow \exists \bar{y}(\psi(\bar{x}, \bar{y}))$, where $\phi(\bar{x})$ and $\psi(\bar{x}, \bar{y})$ are conjunctions of relation atoms over $\bar{x}$ and $\bar{x} \cup \bar{y}$, respectively, such that each variable of $\bar{x}$ occurs in at least one relation atom of $\phi(\bar{x})$;
- full TGDs [2], special case of TGDs without existential quantifiers, *i.e.,* of the form $\forall \bar{x}(\phi(\bar{x}) \rightarrow \psi(\bar{x}))$; and
- LAV TGDs [2], a special case of TGDs in which $\phi(\bar{x})$ is a single relation atom; LAV TGDs subsume INDs.

EXAMPLE 2. *We may use the following* CFDs *as data quality rules on the employee relation of Figure 1:*

$$\varphi_1 = ((\mathsf{CC}, \mathsf{zip} \rightarrow \mathsf{street}),\ T_{P1}),$$
$$\varphi_2 = ((\mathsf{CC}, \mathsf{AC} \rightarrow \mathsf{city}),\ T_{P2}),$$

where CC, zip → street *and* CC, AC → city *are* FDs *embedded in the* CFDs, *and T_{P1} and T_{P2} are pattern tableaux:*

T_{P1}:

CC	zip	street
44	_	_

T_{P2}:

CC	AC	city
_	_	_
44	*131*	*Edinburgh*
01	*908*	*Murray Hill*

CFD φ_1 *states that in the UK (when* CC *= 44),* zip *code uniquely determines* street. *In other words,* CC, zip → street *is an* FD *that is enforced only on tuples that match the pattern* CC *= 44, e.g., on t_1–t_3 in D_0, but not on t_4–t_5. Taking φ as a data quality rule, we find that t_2 and t_3 violate φ_1 and hence, are inconsistent: they have the same* zip *but differ in* street. *Such errors cannot be caught by conventional* FDs.

CFD φ_2 *says that country code* CC *and area code* AC *uniquely determine* city. *Moreover, in the UK (i.e.,* CC *= 44), when* AC *is 131,* city *must be Edinburgh; and in the US (*CC *= 01), if* AC *is 908, then* city *is Murray Hill. It catches t_1 as a violation, i.e., a single tuple may violate a* CFD. *Note that φ_2 subsumes conventional* FD CC, AC → city, *as indicated by the first tuple in T_{P2}, in which '_' is a "wildcard" that matches any value (see [38] for details).* □

To decide what class of dependencies we should use as data quality rules, we want to strike a balance between its "expressive power", *i.e.,* whether it is capable

Dependencies	Implication
FDs	$O(n)$ (cf. [1])
INDs	PSPACE-complete (cf. [1])
FDs + INDs	undecidable (cf. [1])
CFDs	coNP-complete [38]
CINDs	EXPTIME-complete [15]
CFDs + CINDs	undecidable [15]
DCs	coNP-complete [8]
TGDs	undecidable (cf. [1])

Table 1: Complexity of implication analysis

of catching errors commonly found in practice, and the complexity for reasoning about its dependencies and for repairing data.

There are two classical problems for reasoning about dependencies: the satisfiability and implication problems.

Satisfiability. For a class $\mathcal{C}$ of dependencies and $\varphi \in \mathcal{C}$, we use $D \models \varphi$ to denote that a database D satisfies φ, depending on how $\mathcal{C}$ is defined. For a set $\Sigma \subseteq \mathcal{C}$, we use $D \models \Sigma$ to denote that D satisfies all dependencies in Σ. The *satisfiability problem* for $\mathcal{C}$ is to decide, given a finite set $\Sigma \subseteq \mathcal{C}$ defined on a relational schema $\mathcal{R}$, whether there exists a nonempty finite instance D of $\mathcal{R}$ such that $D \models \Sigma$. That is, whether the data quality rules in Σ are consistent themselves.

We can specify arbitrary FDs without worrying about their satisfiability. Indeed, every set of EGDs (or TGDs) can be satisfied by a single-tuple relation [8]. However, a set of DCs or CFDs may *not* be satisfiable by a nonempty database. While the satisfiability problem for DCs has not been settled, it is known that it is NP-complete for CFDs [38], owing to the constant patterns in CFDs. That is, the expressive power of CFDs and DCs come at a price of a higher complexity.

Implication. Consider a finite set $\Sigma \subseteq \mathcal{C}$ of dependencies and another $\varphi \in \mathcal{C}$, both defined on instances of a relational schema $\mathcal{R}$. We say that Σ *implies* φ, denoted by $\Sigma \models \varphi$, if for all instances D of $\mathcal{R}$, $D \models \varphi$ as long as $D \models \Sigma$. The *implication problem* for $\mathcal{C}$ is to decide, given $\Sigma \subseteq \mathcal{C}$ and $\varphi \in \mathcal{C}$ over a relational schema $\mathcal{R}$, whether $\Sigma \models \varphi$. The implication analysis helps us remove redundant data quality rules and hence, speed up error detection and data repairing processes.

Table 1 summarizes known complexity of the implication analysis of data dependencies used as data quality rules.

Data repairing. There are two approaches to obtaining consistent information from an inconsistent database, both proposed by [6]: *data repairing* is to find another database that is consistent and minimally differs from the original database; and *consistent query answering* is to find an answer to a given query in every repair of the original database. Both approaches are based on the notion of repairs. We focus on data repairing in this paper, and refer the interested reader to a comprehensive survey [11] and recent work [12, 67, 86, 87] on consistent query answering.

Repair models. Assume a function $\mathsf{cost}(D, D_r)$ that measures the difference between instances D and D_r of a relational schema $\mathcal{R}$, such that the smaller it is, the closer D_r is to D. Given a set Σ of dependencies and an instance D of $\mathcal{R}$, a *repair* of D *relative to* Σ and $\mathsf{cost}(,)$ is an instance D_r of $\mathcal{R}$ such that $D_r \models \Sigma$ and $\mathsf{cost}(D, D_r)$ is minimum among all instances of $\mathcal{R}$ that satisfy Σ. Several repair models have been studied, based on how $\mathsf{cost}(D, D_r)$ is defined:

- *S-repair* [23]: $\mathsf{cost}(D, D_r) = |D \setminus D_r|$, where $D_r \subseteq D$; assuming that the information in D is inconsistent but complete, this model allows tuple deletions only;
- C-repair [6]: $\mathsf{cost}(D, D_r) = |D \oplus D_r|$, where $D \oplus D_r$ is defined as $(D \setminus D_r) \cup (D_r \setminus D)$; assuming that D is neither consistent nor complete, this model allows both tuple deletions and tuple insertions;
- *CC-repair* [2]: a C-repair such that $|D \oplus D_r|$ is strictly smaller than $|D \oplus D'_r|$ for all D'_r that satisfies Σ; and
- *U-repair* [91, 14]: $\mathsf{cost}(D, D_r)$ is a numerical aggregation function defined in terms of distances and accuracy of attribute values in D and D_r; this model supports attribute value modifications.

For example, the repair model of [14] assumes (a) a *weight* $w(t, A)$ associated with each attribute A of each tuple t in D, and (b) a *distance* function $\mathsf{dis}(v, v')$ for values v and v' in the same domain. Intuitively, $w(t, A)$ indicates the confidence in the *accuracy* of $t[A]$, and $\mathsf{dis}(v, v')$ measures how close v' is to v. The cost of changing the value of an attribute $t[A]$ from v to v' is defined as: $\mathsf{cost}(v, v') = w(t, A) \cdot \mathsf{dis}(v, v')$. That is, the more accurate the original $t[A]$ value v is and the more distant the new value v' is from v, the higher the cost of the change is. The cost of changing a tuple t to t' is the sum of $\mathsf{cost}(t[A], t'[A])$ for A ranging over all attributes in t in which the value of $t[A]$ is modified. The cost of changing D to D_r, denoted by $\mathsf{cost}(D, D_r)$, is the sum of the costs of modifying tuples in D. In practice, repairing is typically carried out via U-repair (see Section 3).

The repair checking problem. Consider a class $\mathcal{C}$ of dependencies and a repair model T with which function $\mathsf{cost}_T(,)$ is associated. The *repair checking problem* for

Dependencies	Repair model	Repair checking
full TGDs	S-repair	PTIME [85]
one FD + one IND	S-repair	coNP-complete [23]
DCs	S-repair	LOGSPACE (cf. [2])
WA LAV TGDs + EGDs	S-repair	LOGSPACE [2]
full TGDs + EGDs	S-repair	PTIME-complete [2]
WA TGDs + EGDs	S-repair	coNP-complete [2]
DCs	C-repair	coNP-complete [73]
full TGDs + EGDs	C-repair	coNP-complete [2]
WA TGDs + EGDs	C-repair	coNP-complete [2]
DCs	CC-repair	coNP-complete [2]
full TGDs + EGDs	CC-repair	coNP-complete [2]
WA TGDs + EGDs	CC-repair	coNP-complete [2]
fixed FDs	U-repair	coNP-complete [14]
fixed CINDs	U-repair	coNP-complete [14]

Table 2: Complexity of repair checking

$(\mathcal{C}, T)$ is to decide, given a finite set $\Sigma \subseteq \mathcal{C}$ of dependencies defined over a relational schema $\mathcal{R}$, and two instances D and D_r of $\mathcal{R}$, whether D_r is a repair of D relative to Σ and $\mathsf{cost}_T(,)$?

The repair checking problem has been studied for various dependencies and repair models; some of the complexity bounds are presented in Table 2. Here a set of TGDs is said to be weakly acyclic (WA) if its dependency graph does not have a cycle going through a special edge that indicates an existentially quantified variable in Σ (see [2] for details).

Table 2 tells us that data repairing is rather expensive, especially for U-repair when attribute values are allowed to be updated: following [14], one can show that its data complexity is already intractable when only FDs or INDs are used.

2.2 Data Deduplication

Data deduplication is the problem of identifying tuples from one or more (possibly unreliable) relations that refer to the same real-world entity. It is also known as record matching, record linkage, entity resolution, instance identification, duplicate identification, merge-purge, database hardening, name matching, coreference resolution, identity uncertainty, and object identification. It is a longstanding issue that has been studied for decades [49], and is perhaps the most extensively studied data quality problem.

The need for data deduplication is evident in, *e.g.*, data quality management, data integration and fraud detection. It is particularly important to big data, which is often characterized by a large number of (heterogeneous) data sources. To make practical use of the data, it is often necessary to accurately identify tuples from different sources that refer to the same entity, so that we can fuse the data and enhance the information about the entity. This is nontrivial: data from various sources may be dirty, and moreover, even when the data sources are seemingly reliable, inconsistencies and conflicts often emerge when we integrate the data [14].

A variety of approaches have been proposed for data deduplication: probabilistic (*e.g.*, [49, 65, 95]), learning-based [27, 82], distance-based [60], and rule-based [3, 44, 61] (see [33, 62, 78] for surveys). In this paper we focus on rule-based collective and collaborative deduplication.

Data deduplication. To simplify the discussion, consider a single relation schema R. This does not lose generality since for any relational schema $\mathcal{R} = (R_1, \ldots, R_n)$, one can construct a single relation schema R and a linear bijective function $f()$ from instances of $\mathcal{R}$ to instances of R, without loss of information. Consider a set E of *entity types*, each specified by $e[X]$, where X is a set of attributes of R.

Given an instance D of R and a set E of entity types, *data deduplication* is to determine, for all tuples t, t' in D, and for each entity type $e[X]$, whether $t[X]$ and $t'[X]$ should be identified, *i.e.*, they refer to the same entity of type e. Following [13], we call $t[X]$ and $t'[X]$ *references* to e entities.

EXAMPLE 3. *On the employee relation of Figure 1, we may consider two entity types:* address *specified by* (CC, street, city, zip), *and* person *as the list of all attributes of employee. Given employee tuples t and t', deduplication is to decide whether t*[address] *and t'*[address] *refer to the same address, and whether t and t' are the same person.* □

As observed in [13], references to different entities may co-occur, and entities for co-occurring references should be determined jointly. For instance, papers and authors co-occur; identifying two authors helps identify their papers, and vice versa. This is referred to as *collective entity resolution* (deduplication) [13]. A graph-based method is proposed in [13] to propagate similarity among references, for collective deduplication. A datalog-like language is introduced in [5], with recursive rules for collective deduplication.

Matching rules. Rules were first studied in [3] for deduplication. Extending [3], a class of *matching dependencies* is defined in [44] in terms of similarity predicates and a matching operator $\rightleftharpoons$, based on a dynamic semantics [34].

EXAMPLE 4. *Matching dependencies on the employee relation of Figure 1 include the following:*

$$\psi_1 = \forall t, t' (t[\mathsf{CC}, \mathsf{AC}, \mathsf{landline}] = t'[\mathsf{CC}, \mathsf{AC}, \mathsf{landline}] \rightarrow t[\mathsf{address}] \rightleftharpoons t'[\mathsf{address}]),$$

$\psi_2 = \forall t, t'(t[\mathsf{LN}, \mathsf{address}] = t'[\mathsf{LN}, \mathsf{address}] \wedge t[\mathsf{FN}] \approx t'[\mathsf{FN}] \rightarrow t[\mathsf{person}] \rightleftharpoons t'[\mathsf{person}])$,
$\psi_3 = \forall t, t'(t[\mathsf{CC}, \mathsf{AC}, \mathsf{mobile}] = t'[\mathsf{CC}, \mathsf{AC}, \mathsf{mobile}] \rightarrow t[\mathsf{person}] \rightleftharpoons t'[\mathsf{person}])$,

Intuitively, (a) ψ_1 states that if t and t' have the same landline phone, then t[address] and t'[address] should refer to the same address and be equalized via updates; (b) ψ_2 says that if t and t' have the same address and last name, and if they have similar *first names, then they refer to the same person; and (c) ψ_3 states that if t and t' have the same mobile phone, then they should be identified as the same person. Here $\approx$ denotes a predicate for similarity of* FN*, such that, e.g., Bob $\approx$ Robert, since Bob is a nickname of Robert.*

These rules identify t_4 and t_5 in Figure 1 as follows. (a) By ψ_1, t_4[address] and t_5[address] should be identified although their values are radically different*; and (b) by (a) and ψ_2, t_4 and t_5 refer to the same person. Note that matching dependencies can be "recursively" applied: the outcome of (a) is used to deduce (b), for collective deduplication.* □

There exists a sound and complete axiom system for deducing matching dependencies from a set of known matching dependencies, based on their dynamic semantics [34]. The deduction process is in quadratic time. Moreover, "negative rules" such as "a male and a female cannot be the same person" can be expressed as matching dependencies without the need for introducing negation [45].

An operational semantics is developed for matching dependencies in [12] by means of a chase process with matching functions. It is shown that matching dependencies can also be used in data cleaning, together with related complexity bounds for consistent query answering [12]. Other types of rules have also been studied in, *e.g.*, [4, 16, 89].

Collaborative deduplication. Data repairing and deduplication are often taken as separate processes. To improve the accuracy of both processes, the two should be unified [45].

EXAMPLE 5. *We show how data repairing and deduplication interact to identify t_1–t_3 of Figure 1 as follows.*

(a) By CFD *φ_1 of Example 2, we have that t_2 and t_3 have the same address. By matching dependency ψ_2 of Example 4, we deduce that t_2 and t_3 refer to the same person. Moreover, we can enrich t_2 by t_2[landline, mobile] := t_3[landline, mobile].*

(b) By ψ_3 of Example 4, we deduce that t_1 and t_3 refer to the same person. Therefore, t_1–t_3 refer to the same Mary. □

The example shows that repairing helps deduplication and vice versa. This is also observed in [5]. Algorithms for unifying repairing and deduplication are given in [45]. In addition to data consistency, it has also been verified that data deduplication should also be combined with the analyses of data currency (timeliness) and data accuracy [42, 70].

Putting these together, we advocate *collaborative deduplication* that incorporates the analyses of data consistency (repairing), currency, accuracy and co-occurrences of attributes into the deduplication process, not limited to co-occurring references considered in collective deduplication [13].

2.3 Information Completeness

Information completeness concerns whether our database has complete information to answer our queries. Given a database D and a query Q, we want to know whether Q can be correctly answered by using only the data in D.

A database is typically assumed either closed or open.

- Under the Closed World Assumption (CWA), our database includes all the tuples representing real-world entities, but some *attribute values* may be *missing*.
- Under the Open World Assumption (OWA), our database may only be a proper subset of the set of tuples that represent real-world entities. That is, both tuples and values may be missing.

The CWA is often too strong in the real world [76]. Under the OWA, however, few queries can find correct answers.

To deal with missing values, representation systems are typically used (*e.g.*, c-tables, v-tables [59, 64]), based on certain query answers, which are recently revised in [71]. There has also been work on coping with missing tuples, by assuming that there exists a virtual database D_c with "complete information", and that part of D is known as a view of D_c [69, 77, 80]. Given such a database D, we want to determine whether a query posed on D_c can be answered by an equivalent query on D, via query answering using views.

Relative information completeness. We can possibly do better by making use of master data. An enterprise nowadays typically maintains *master data* (*a.k.a. reference data*), a single repository of high-quality data that provides various applications with a synchronized, consistent view of *the core business entities* of the enterprise [74].

Given a database D and master data D_m, we specify a set V of *containment constraints* [36]. Such a constraint

RCDP($\mathcal{L}_Q, \mathcal{L}_C$)	combined complexity [36]	data complexity [17]
(FO, CQ)	undecidable	undecidable
(CQ, CQ)	Π_2^p-complete	PTIME
(UCQ, UCQ)	Π_2^p-complete	PTIME

Table 3: Relative information completeness

is of the form $q(D) \subseteq p(D_m)$, where q is a query on D, and p is a simple projection on D_m. Intuitively, D_m is closed-world, and the part of D that is constrained by V is bounded by D_m, while the rest is open-world. We refer to a database D that satisfies V as a *partially closed* database *w.r.t.* (D_m, V). A database D_e is a *partially closed extension* of D if $D \subseteq D_e$ and D_e is partially closed *w.r.t.* (D_m, V) itself.

A partially closed database D is said to be *complete for a query Q relative to* (D_m, V) if for all partially closed extensions D_e of D *w.r.t.* (D_m, V), $Q(D_e) = Q(D)$. That is, there is no need for adding new tuples to D, since they either violate the containment constraints, or do not change the answer to Q in D. In other words, D already contains complete information necessary for answering Q [36].

EXAMPLE 6. *Recall that relation D_0 of Figure 1 may not have complete information to answer query Q_1 of Example 1. Now suppose that we have a master relation D_m of schema* (FN, LN, city)*, which maintains complete employee records in the UK, and a containment constraint ϕ: $\pi_{\mathsf{FN,LN,city}}\sigma_{CC=44}(D_0) \subseteq D_m$, i.e., the set of UK employees in D_0 is contained in D_m. Then if $Q_1(D_0)$ returns all employees in Edinburgh found in D_m, we can safely conclude that D_0 is complete for Q_1 relative to* $(D_m, \{\phi\})$. □

Several problems have been studied for relative information completeness [17, 36]. One of the problems, denoted by RCDP($\mathcal{L}_Q, \mathcal{L}_C$), is to determine, given a query Q, master data D_m, a set V of containment constraints, and a partially closed database D *w.r.t.* (D_m, V), whether D is complete for Q relatively to (D_m, V), where $\mathcal{L}_Q$ and $\mathcal{L}_C$ are query languages in which Q and q (in containment constraints) are expressed, respectively. Some complexity bounds of RCDP($\mathcal{L}_Q, \mathcal{L}_C$) are shown in Table 3, where CQ, UCQ and FO denote conjunctive queries (SPJ), unions of conjunctive queries (SPJU) and FO queries (the full relational algebra), respectively. The complexity bounds demonstrate the difficulty of reasoning about information completeness. Relative information completeness has also been studied in the setting where both values and tuples may be missing, by extending representation systems for missing values [35].

Containment constraints are also able to express dependencies used in the analysis of data consistency, such as CFDs and CINDs [36]. Hence we can study data consistency and information completeness in a uniform framework.

2.4 Data Currency

Data currency (*timeliness*) aims to identify the current values of entities represented by tuples in a (possibly stale) database, and to answer queries with the current values.

There has been work on how to define current tuples by means of timestamps in temporal databases (see, *e.g.*, [24, 83] for surveys). In practice, however, timestamps are often unavailable or imprecise [96]. The question is how to determine data currency in the absence of reliable timestamps.

Modeling data currency. We present a model proposed in [43]. Consider a database D that possibly contains stale data. For each tuple $t \in D$, $t[\mathsf{eid}]$ denotes the id of the entity that t represents, obtained by data deduplication (see Section 2.2).

(1) The model assumes *a currency order* $\prec_A$ for each attribute A of each relation schema R, such that for tuples t_1 and t_2 of schema R in D, if $t_1[\mathsf{eid}] = t_2[\mathsf{eid}]$, *i.e.*, when t_1 and t_2 represent the same entity, then $t_1 \prec_A t_2$ indicates that t_2 is more up-to-date than t_1 in the A attribute value. This is to model partially available currency information in D.

(2) The model uses *currency constraints* to specify currency relationships derived from the semantics of the data, expressed as denial constraints equipped with constants.

EXAMPLE 7. *Extending relation D_0 of Figure 1 with attribute* eid*, currency constraints on D_0 include:*

$$\forall s, t\big((s[\mathsf{eid}] = t[\mathsf{eid}] \wedge s[\mathsf{status}] = \text{“}\textit{married}\text{”} \wedge t[\mathsf{status}] = \text{“}\textit{single}\text{”}) \rightarrow t \prec_{\mathsf{status}} s\big),$$
$$\forall s, t\big((s[\mathsf{eid}] = t[\mathsf{eid}] \wedge t \prec_{\mathsf{status}} s \rightarrow t \prec_{\mathsf{LN}} s\big).$$

These constraints are derived from the semantics of the data: (a) marital changes from “single” to “married”, but not the other way around; and (b) LN *and* status *are correlated: if t has more current status than s, it also has more current* LN.

Based on these, query Q_2 of Example 1 can be answered with the most current LN *value of Mary, namely, Luth.* □

Based on currency orders and constraints, we can define (3) *consistent completions* D^c of D, which extend $\prec_A$ in D to a total order on all tuples pertaining to the same

CCQA($\mathcal{L}_Q$)	combined complexity [43]	data complexity [43]
FO	PSPACE-complete	coNP-complete
CQ, UCQ	Π_2^p-complete	coNP-complete

Table 4: Certain current answers

entity, such that D^c satisfies the currency constraints; and

(4) from D^c, we can extract the *current tuple* for each entity eid, composed of the entity's most current A value for each attribute A based on $\prec_A$. This yields the *current instance* of D^c consisting of only the current tuples of the entities in D, from which currency orders can be removed.

(5) We compute *certain current answers* to a query Q in D, *i.e.,* answers to Q in *all* consistent completions D^c of D.

Several problems associated with data currency are studied in [43]. One of the problems, denoted by CCQA($\mathcal{L}_Q$), is to decide, given a database D with partial currency orders $\prec_A$ and currency constraints, a query $Q \in \mathcal{L}_Q$ and a tuple t, whether t is a certain current answer to Q in D. Some of the complexity results for CCQA($\mathcal{L}_Q$) are shown in Table 4.

2.5 Data Accuracy

Data accuracy refers to the closeness of values in a database to the true values of the entities that the data in the database represents, when the true values are not known.

While it has long been recognized that data accuracy is critical to data quality [7], the topic has not been well studied. Prior work typically studies the reliability of data sources, *e.g.,* dependencies [30] and lineage information [90] of data sources to detect copy relationships and identify reliable sources, vote counting and probabilistic analysis based on the trustworthiness of data sources [51, 97].

Complementary to the reliability analysis of sources, relative accuracy is studied in [18]. Given tuples t_1 and t_2 that pertain to the same entity, it is to infer whether $t_1[A]$ is more accurate than $t_2[A]$ for attributes A of the tuples. The inference is conducted by a chase process, by combining the analyses of data consistency, currency and correlated attributes.

3. DATA CLEANING TECHNIQUES

As Gartner [54] put it, the data quality tool market is "among the fastest-growing in the enterprise software sector". It reached $1.13 billion in software revenue in 2013, about 13.2% growth, and will reach $2 billion by 2017, 16% growth. While data quality tools have mostly been dealing with customer, citizen and patient data, they are rapidly expanding into financial and quantitative data domains.

What does the industry need from data quality tools? Such tools are expected to automate key elements, including: (1) data profiling to discover data quality rules, in particular "dependency analysis (cross-table and cross-dataset analysis)"; (2) cleaning, "the modification of data values to meet domain restrictions, integrity constraints or other business rules"; and (3) matching, "the identifying, linking and merging of related entries within or across sets of data", and in particular, "matching rules or algorithms" [54].

In this section we briefly survey techniques for profiling (discovery of data quality rules), cleaning (error detection and data repairing) and matching (data deduplication).

3.1 Discovering Data Quality Rules

To clean data with data quality rules, the first question we have to answer is how we can get the rules. It is unrealistic to rely on domain experts to design data quality rules via an expensive and long manual process, or count on business rules that have been accumulated. This highlights the need for automatically *discovering* and *validating* data quality rules.

Rule discovery. For a class $\mathcal{C}$ of dependencies that are used as data quality rules, the *discovery problem* for $\mathcal{C}$ is stated as follows. Given a database instance D, it is to find a *minimal cover*, a non-redundant set of dependencies that is logically equivalent to the set of all dependencies in $\mathcal{C}$ that hold on D.

A number of discovery algorithms are developed for, *e.g.,*

- FDs, *e.g.,* [63, 93], and INDs (see [72] for a survey);
- CFDs, *e.g.,* [21, 39, 56, 58], and CINDs [56];
- denial constraints DCs [25]; and for
- matching dependencies [84].

Discovery algorithms are often based on the levelwise approach proposed by [63], *e.g.,* [21, 39], depth-first search of [93], *e.g.,* [25, 39], and association rule mining [39, 56].

Rule validation. Data quality rules are discovered from possibly dirty data, and are likely "dirty" themselves. Hence given a set Σ of discovered rules, we need to identify what rules in Σ make sense, by checking their satisfiability. In addition, we want to remove redundant rules from Σ, by making use of implication analysis (see Section 2.1).

It is nontrivial to identify sensible rules from Σ. Recall that the satisfiability problem is NP-complete for CFDs, and is nontrivial for DCs. Nevertheless, approximation algorithms can be developed. For CFDs, such algorithms have been studied [38], which extract a set Σ' of satisfiable dependencies from Σ, and guarantee that Σ' is "close" to a maximum satisfiable subset of Σ, within a constant bound.

3.2 Error Detection

After data quality rules are discovered and validated, the next question concerns how to effectively catch errors in a database by using these rules. Given a database D and a set Σ of dependencies as data quality rules, *error detection* (*a.k.a. error localization*) is to find all tuples in D that violate at least one dependency in Σ. Error detection is a routine operation of data quality tools. To clean data we have to detect errors first. Many users simply want errors in their data to be detected, without asking for repairing the data.

Error detection methods depend on (a) what dependencies are used as data quality rules, and (b) whether the data is stored in a local database or distributed across different sites.

Centralized databases. When D resides in a centralized database and when Σ is a set of CFDs, *two SQL queries* Q^c and Q^v can be *automatically* generated such that $Q^c(D)$ and $Q^v(D)$ return all and only those tuples in D that violate Σ [38]. Better still, Q^c and Q^v are independent of the number and size of CFDs in Σ. That is, we can detect errors by leveraging existing facility of commercial relational DBMS.

EXAMPLE 8. *To detect violations of* $\varphi_2 = ((\mathsf{CC}, \mathsf{AC} \to \mathsf{city}),\ T_{P2})$ *of Example 2, we use the following* Q^c *and* Q^v*:*

Q^C
```
SELECT * FROM R t, T_P2 t_p
WHERE t[CC, AC] ≍ t_p[CC, AC] AND t[city] ≭ t_p[city]
```

Q^V
```
SELECT DISTNCT CC, AC FROM R t, T_P2 t_p
WHERE  t[CC, AC] ≍ t_p[CC, AC] AND t_p[city] = '_'
GROUP BY CC, AC HAVING COUNT(DISTNCT city) > 1
```

where $t[\mathsf{CC}, \mathsf{AC}] \asymp t_p[\mathsf{CC}, \mathsf{AC}]$ *denotes (*$t[\mathsf{CC}] = t_p[\mathsf{CC}]$ *OR* $t_p[\mathsf{CC}] =$ '_'*) AND (*$t[\mathsf{AC}] = t_p[\mathsf{AC}]$ *OR* $t_p[\mathsf{AC}] =$ '_'*); and* R *denotes the schema of employee datasets. Intuitively,* Q^C *catches single-tuple violations of* φ_2*, i.e., those that violate a pattern in* T_{P2}*, and* Q^V *identifies violations of the* FD *embedded in* φ_2*. Note that* Q^C *and* Q^V *simply treat pattern tableau* T_{P2} *as an "input" relation, regardless of its size. In other words,* Q^C *and* Q^V *are determined only by the* FD *embedded in* φ_2*, no matter how large the tableau* T_{P2} *is.*

When Σ *consists of multiple* CFDs*, we can "merge" these* CFDs *into an equivalent one, by making use of a new wildcard [38]. Thus two SQL queries as above suffice for* Σ*.* □

The SQL-based method also works for CINDs [20].

Distributed data. In practice a database is often fragmented and distributed across different sites. In this setting, error detection necessarily requires data shipment from one site to another. For both vertically or horizontally partitioned data, it is NP-complete to decide whether error detection can be carried out by shipping a bounded amount of data, and the SQL-based method no longer works [40]. Nevertheless, distributed algorithms are in place to detect CFD violations in distributed data, with performance guarantees [40, 47].

3.3 Data Repairing

After errors are detected, we want to fix the errors. Given a database D and a set Σ of dependencies as data quality rules, *data repairing* (*a.k.a. data imputation*) is to find a repair D_r of D with minimum $\mathsf{cost}(D, D_r)$. We focus on the U-repair model based on attribute-value modifications (see Section 2.1), as it is widely used in the real world [54].

Heuristic fixes. Data repairing is cost-prohibitive: its data complexity is coNP-complete for fixed FDs or INDs [14]. In light of this, repairing algorithms are mostly heuristic, by enforcing dependencies in Σ one by one. This is nontrivial.

EXAMPLE 9. *Consider two relation schemas* $R_1(A, B)$ *and* $R_2(B, C)$*, an* FD *on* R_1*:* $A \to B$*, and an* IND $R_2[B] \subseteq R_1[B]$*. Consider instances* $D_1 = \{(1,2), (1,3)\}$ *of* R_1 *and* $D_2 = \{(2,1), (3,4)\}$*, where* D_1 *does not satisfy the* FD*. To repair* (D_1, D_2)*, a heuristic may enforce the* FD *first, to "equalize"* 2 *and* 3*; it then needs to enforce the* IND*, by ensuring that* D_1 *includes* $\{2, 3\}$ *as its* B*-attribute values. This yields a repairing process that does not terminate.* □

Taking both FDs and INDs as data quality rules, a heuristic method is proposed in [14] based on equivalence classes, which group together attribute values of D that must take the same value. The idea is to separate the decision of which values should be equal from the decision of what values should be assigned to the equivalence classes. Based on the $\mathsf{cost}(,)$ function given in Section 2.1, it guarantees to find a repair. The method has been extended to repair data based on CFDs [28], EGDs and TGDs [55] with a partial order on equivalence classes to specify preferred updates, and DCs [26] by generalizing equivalence classes to conflict hypergraphs. An approximation algorithm for repairing data based on FDs was developed in [66].

A semi-automated method is introduced in [94] for data repairing based on CFDs. In contrast to [14], it interacts with users to solicit credible updates and improve the accuracy. Another repairing method is studied in [45], which picks reliable fixes based on an analysis of the relative certainty of the data, measured by entropy. There have also been attempts to unify data repairing and deduplication [45] based on CFDs, matching dependencies and master data.

Certain Fixes. A major problem with heuristic repairing methods is that they do not guarantee to find correct fixes; worse still, they may introduce new errors when attempting to fix existing errors. As an example, to fix tuple t_1 of Figure 1 that violates CFD φ_2 of Example 2, a heuristic method may very likely change $t_1[\mathsf{city}]$ from London to Edinburgh. While the change makes t_1 a "repair", the chances are that for the entity represented by t_1, AC is 020 and city is London. That is, the heuristic update does not correct the error in $t_1[\mathsf{AC}]$, and worse yet, it changes $t_1[\mathsf{city}]$ to a wrong value. Hence, while the heuristic methods may suffice for statistical analysis, *e.g.*, census data, they are often too risky to be used in repairing critical data such as medical records.

This highlights the need for studying certain fixes for critical data, *i.e.*, fixes that are guaranteed to be correct [46]. To identify certain fixes, we make use of (a) master data (Section 2.3), (b) editing rules instead of data dependencies, and (c) a chase process for inferring "certain regions" based on user confirmation, master data and editing rules, where certain regions are attribute values that are validated.

Editing rules are dynamic constraints that tell us which attributes should be changed and to what values they should be changed. In contrast, dependencies have a static semantics; they are capable of detecting the presence of errors in the data, but they do not tell us how to fix the errors.

EXAMPLE 10. *Assume master data D_m with schema $R_m(\mathsf{postal}, \mathsf{C}, \mathsf{A})$ for postal code, city and area code in the UK. An editing rule for D_0 of Fig. 1 is as follows:*

$$\sigma\text{:}\ (\mathsf{postal}, \mathsf{zip}) \rightarrow ((\mathsf{C}, \mathsf{city}), (\mathsf{A}, \mathsf{AC})),$$

specified with pairs of attributes from D_m and D_0. It states that for an input tuple t, if $t[\mathsf{zip}]$ is validated and there exists a master tuple $s \in D_m$ such that $t[\mathsf{zip}] = s[\mathsf{postal}]$, then update $t[\mathsf{city}, \mathsf{AC}] := s[\mathsf{C}, \mathsf{A}]$ is guaranteed a certain fix, and $t[\mathsf{AC}, \mathsf{city}]$ becomes a certain region (validated). Suppose that there is s = (W1B 1JL, London, 020) in D_m, and that $t_1[\mathsf{zip}]$ of Figure 1 is validated. Then $t_1[\mathsf{AC}]$ should be changed to 020; here $t_1[\mathsf{city}]$ remains unchanged. □

A framework is developed in [46] for inferring certain fixes for input tuples. Although it may not be able to fix all the errors in the data based on available information, it guarantees that each update fixes at least one error, and that no new errors are introduced in the entire repairing process. The process may consult users to validate a minimum number of attributes in the input tuples. Static analyses of editing rules and certain regions can also be found in [46].

Editing rules are generalized in [29] by allowing generic functions to encompass editing rules [46], CFDs and matching dependencies. However, it remains to be justified whether such generic rules can be validated themselves and whether the fixes generated are sensible at all.

Beyond data repairing. Data repairing typically assumes that data quality rules have been validated. Indeed, in practice we use data quality rules to clean data only after the rules are confirmed correct themselves. A more general setting is studied in [22], when both data and data quality rules are possibly dirty and need to be repaired.

There has also been work on (a) causality of errors [75] and its connection with data repairs [81], and (b) propagation of errors and dependencies in data transformations [19, 48].

3.4 Data Deduplication

A number of systems have been developed for data deduplication, *e.g.*, BigMatch [95], Tailor [32], Swoosh [10] AJAX [50], CrowdER [88] and Corleone [57], as stand-alone tools, embedded packages in ETL systems, or crowd-sourced systems. Criteria for developing such systems include (a) accuracy, to reduce *false matches* (false positives) and *false non-matches* (false negatives); and (b) scalability with big data. To improve accuracy, we advocate collaborative deduplication (Section 2.2), including but not limited to collective deduplication [13]. For scalability, parallel matching methods need to be developed and combined with traditional blocking and windowing techniques (see [33]). We refer the interested reader to [62, 78] for detailed surveys.

4. CHALLENGES INTRODUCED BY BIG DATA

The study of data quality has raised as many questions as it has answered. In particular, a full treatment is required for each of data accuracy, currency and information completeness, as well as their interaction with data consistency and deduplication. Moreover, big data introduces a number of challenges, and the study of big

data quality is in its infancy.

Volume. Cleaning big data is cost-prohibitive: discovering data quality rules, error detection, data repairing and data deduplication are all expensive; *e.g.,* the data complexity of data repairing is coNP-complete for FDs and INDs [14]. To see what it means in the context of big data, observe that a linear scan of a dataset D of PB size (10^{15} bytes) takes days using a solid state drive with a read speed of 6GB/s, and it takes years if D is of EB size (10^{18} bytes) [41].

To cope with the volume of big data, we advocate the following approaches, taking data repairing as an example.

Parallel scalable algorithms. We approach big data repairing by developing parallel algorithms. This is often necessary since in the real world, big data is often distributed.

It is *not* always the case that the more processors are used, the faster we get. To characterize the effectiveness of parallelization, we formalize parallel scalability following [68].

Consider a dataset D and a set Σ of data quality rules. We denote by $t(|D|, |\Sigma|)$ the worst-case running time of a *sequential algorithm* for repairing D with Σ; and by $T(|D|, |\Sigma|, n)$ the time taken by a parallel algorithm for the task *by using n processors*, taking n as a parameter. Here we assume $n \ll |D|$, *i.e.,* the number of processors does not exceed the size of the data, as commonly found in practice.

We say that the algorithm is *parallel scalable* if

$$T(|D|, |\Sigma|, n) = O(t(|D|, |\Sigma|)/n) + (n|\Sigma|)^{O(1)}.$$

That is, the parallel algorithm achieves a polynomial reduction in sequential running time, plus a "bookkeeping" cost $O((n|\Sigma|)^l)$ for a constant l that is *independent of* $|D|$.

Obviously, if the algorithm is parallel scalable, then for a given D, it *guarantees* that the more processors are used, the less time it takes to repair D. It allows us to repair big data by adding processors when needed. If an algorithm is not parallel scalable, it may not be able to efficiently repair D when D grows big *no matter how many* processors are used.

Entity instances. We propose to deal with entity instances instead of processing the big dataset D directly. An *entity instance* I_e is a set of tuples in D that pertain to the same entity e. It is *substantially smaller* than D, and typically retains a manageable size when D grows big. This suggests the following approach to repairing big data: (1) cluster D into entity instances I_e, by using a parallel data deduplication algorithm; (2) for each entity e, deduce "the true values" of e from I_e, by processing all entities in parallel; and (3) resolve inconsistencies across different entities, again in parallel.

We find that this approach allows us to effectively and efficiently deduce accurate values for each entity, by reasoning about data consistency, data deduplication with master data, data accuracy and data currency together [18, 42].

Bounded incremental repairing. We advocate *incremental data repairing*. Given a big dataset D, a set Σ of data quality rules, a repair D_r of D with Σ, and updates ΔD to D, it is to find *changes* ΔD_r to the repair D_r such that $D_r \oplus \Delta D_r$ is a repair of $D \oplus \Delta D$ with Σ, where $D \oplus \Delta D$ denotes the updated dataset of D with ΔD; similarly for $D_r \oplus \Delta D_r$.

Intuitively, small changes ΔD to D often incur a small number of new violations to the rules in Σ; hence, changes ΔD_r to the repair D_r are also small, and it is more efficient to find ΔD_r than to compute a new repair starting from scratch. In practice, data is frequently updated, but the changes ΔD are typically small. We can minimize unnecessary recomputation of D_r by incremental data repairing.

The benefit is more evident if there exists a bounded incremental repairing algorithm. As argued in [79], incremental algorithms should be analyzed in terms of |CHANGED| = $|\Delta D|$ + $|\Delta D_r|$, indicating the updating costs that are *inherent to* the incremental problem itself. An incremental algorithm is said to be *bounded* if its cost can be expressed as a function of |CHANGED| and $|\Sigma|$, *i.e.,* it depends only on |CHANGED| and $|\Sigma|$, *independent of* the size of big D.

This suggests the following approach to repairing and maintaining a big dataset D. (1) We compute repair D_r of D *once*, in parallel by using a number of processors. (2) In response to updates ΔD to D, we *incrementally* compute ΔD_r, by reducing the problem of repairing big D to an incremental problem on "small data" of size |CHANGED|. The incremental step may not need a lot of resources.

Besides the scalability of repairing algorithms with big data, we need to ensure the accuracy of repairs. To this end, we promote the following approach.

Knowledge bases as master data. Master data is extremely helpful in identifying certain fixes [46], data repairing [45] and in deducing the true values of entities [18, 42]. A number of high-quality knowledge bases are already developed these days, and can be employed as master data. We believe that repairing algorithms should be developed by taking the knowledge bases as master data, to improve the accuracy.

Velocity. Big datasets are "dynamic": they change fre-

quently. This further highlights the need for developing bounded incremental algorithms for data cleaning. When CFDs are used as data quality rules, incremental algorithms are in place for error detection in centralized databases [38] and distributed data [47], and for data repairing [28]. Nonetheless, parallel incremental algorithms need to be developed for error detection, data repairing and deduplication.

Variety. Big data is also characterized by its heterogeneity. Unfortunately, very little is known about how to model and improve the quality of data beyond relations. In particular, graphs are a major source of big data, *e.g.*, social graphs, knowledge bases, Web sites, and transportation networks. However, integrity constraints are not yet well studied for graphs to determine the consistency of the data. Even keys, a primary form of data dependencies, are not yet defined for graphs. Given a graph G, we need keys that help us uniquely identify entities represented by vertices in G.

Keys for graphs are, however, a departure from their counterparts for relations, since such keys have to be specified in terms of both attribute values of vertices and the topological structures of neighborhoods, perhaps in terms of graph pattern matching by means of subgraph isomorphism.

Acknowledgments. The author thanks Floris Geerts for his thorough reading of the first draft and for helpful comments. The author is supported in part by NSFC 61133002, 973 Program 2012CB316200, ERC 652976, Shenzhen Peacock Program 1105100030834361, Guangdong Innovative Research Team Program 2011D005, EPSRC EP/J015377/1 and EP/M025268/1, NSF III 1302212, and a Google Faculty Research Award.

5. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
[2] F. N. Afrati and P. G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *ICDT*, pages 31–41, 2009.
[3] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, pages 586–597, 2002.
[4] A. Arasu, S. Chaudhuri, and R. Kaushik. Transformation-based framework for record matching. In *ICDE*, pages 40–49, 2008.
[5] A. Arasu, C. Ré, and D. Suciu. Large-scale deduplication with constraints using dedupalog. In *ICDE*, pages 952–963, 2009.
[6] M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79, 1999.
[7] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006.
[8] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-generating dependencies. *JCSS*, 59(1):94–115, 1999.
[9] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *JACM*, 31(4):718–741, 1984.
[10] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *VLDB J.*, 18(1):255–276, 2009.
[11] L. Bertossi. *Database Repairing and Consistent Query Answering*. Morgan & Claypool Publishers, 2011.
[12] L. E. Bertossi, S. Kolahi, and L. V. S. Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. *TCS*, 52(3):441–482, 2013.
[13] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *TKDD*, 1(1), 2007.
[14] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD*, pages 143–154, 2005.
[15] L. Bravo, W. Fan, and S. Ma. Extending inclusion dependencies with conditions. In *VLDB*, 2007.
[16] D. Burdick, R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. A declarative framework for linking entities. In *ICDT*, 2015.
[17] Y. Cao, T. Deng, W. Fan, and F. Geerts. On the data complexity of relative information completeness. *Inf. Syst.*, 45:18–34, 2014.
[18] Y. Cao, W. Fan, and W. Yu. Determining the relative accuracy of attributes. In *SIGMOD*, pages 565–576, 2013.
[19] A. Chalamalla, I. F. Ilyas, M. Ouzzani, and P. Papotti. Descriptive and prescriptive data cleaning. In *SIGMOD*, pages 445–456, 2014.
[20] W. Chen, W. Fan, and S. Ma. Analyses and validation of conditional dependencies with built-in predicates. In *DEXA*, 2009.
[21] F. Chiang and R. J. Miller. Discovering data quality rules. *PVLDB*, 1(1):1166–1177, 2008.
[22] F. Chiang and R. J. Miller. A unified model for data and constraint repair. In *ICDE*, pages 446–457, 2011.
[23] J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Information and Computation*, 197(1-2):90–121, 2005.
[24] J. Chomicki and D. Toman. Time in database systems. In M. Fisher, D. Gabbay, and L. Víla, editors, *Handbook of Temporal Reasoning in Artificial Intelligence*, pages 429–467. Elsevier, 2005.
[25] X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. *PVLDB*, 6(13):1498–1509, 2013.
[26] X. Chu, I. F. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *ICDE*, pages 458–469, 2013.
[27] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *KDD*, 2002.
[28] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: Consistency and accuracy. In *VLDB*, pages 315–326, 2007.
[29] M. Dallachiesa, A. Ebaid, A. Eldawy, A. K. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. NADEEF: a commodity data cleaning system. In *SIGMOD*, 2013.
[30] X. Dong, L. Berti-Equille, and D. Srivastava. Truth discovery and copying detection in a dynamic world. In *PVLDB*, 2009.
[31] W. W. Eckerson. Data quality and the bottom line: Achieving business success through a commitment to high quality data. Technical report, The Data Warehousing Institute, 2002.
[32] M. G. Elfeky, A. K. Elmagarmid, and V. S. Verykios. TAILOR: A record linkage tool box. In *ICDE*, 2002.
[33] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *TKDE*, 19(1):1–16, 2007.
[34] W. Fan, H. Gao, X. Jia, J. Li, and S. Ma. Dynamic constraints for record matching. *VLDB J.*, 20(4):495–520, 2011.
[35] W. Fan and F. Geerts. Capturing missing tuples and missing values. In *PODS*, pages 169–178, 2010.
[36] W. Fan and F. Geerts. Relative information completeness. *ACM Trans. on Database Systems*, 35(4), 2010.
[37] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Morgan & Claypool Publishers, 2012.
[38] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies.

TODS, 33(1), 2008.
[39] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *TKDE*, 23(5):683–698, 2011.
[40] W. Fan, F. Geerts, S. Ma, and H. Müller. Detecting inconsistencies in distributed data. In *ICDE*, pages 64–75, 2010.
[41] W. Fan, F. Geerts, and F. Neven. Making queries tractable on big data with preprocessing. *PVLDB*, 6(8):577–588, 2013.
[42] W. Fan, F. Geerts, N. Tang, and W. Yu. Conflict resolution with data currency and consistency. *J. Data and Information Quality*, 5(1-2):6, 2014.
[43] W. Fan, F. Geerts, and J. Wijsen. Determining the currency of data. *TODS*, 37(4), 2012.
[44] W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. In *VLDB*, 2009.
[45] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Interaction between record matching and data repairing. In *SIGMOD*, 2011.
[46] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Towards certain fixes with editing rules and master data. *VLDB J.*, 21(2):213–238, 2012.
[47] W. Fan, J. Li, N. Tang, and W. Yu. Incremental detection of inconsistencies in distributed data. *TKDE*, 2014.
[48] W. Fan, S. Ma, Y. Hu, J. Liu, and Y. Wu. Propagating functional dependencies with conditions. *PVLDB*, 1(1):391–407, 2008.
[49] I. Fellegi and A. B. Sunter. A theory for record linkage. *J. American Statistical Association*, 64(328):1183–1210, 1969.
[50] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. AJAX: An extensible data cleaning tool. In *SIGMOD*, page 590, 2000.
[51] A. Galland, S. Abiteboul, A. Marian, and P. Senellart. Corroborating information from disagreeing views. In *WSDM*, 2010.
[52] V. Ganti and A. D. Sarma. *Data Cleaning: A Practical Perspective*. Morgan & Claypool Publishers, 2013.
[53] Gartner. 'Dirty data' is a business problem, not an IT problem, 2007. *http://www.gartner.com/newsroom/id/501733*.
[54] Gartner. Magic quardrant for data quality tools, 2014.
[55] F. Geerts, G. Mecca, P. Papotti, and D. Santoro. The LLUNATIC data-cleaning framework. *PVLDB*, 2013.
[56] B. Goethals, W. L. Page, and H. Mannila. Mining association rules of simple conjunctive queries. In *SDM*, 2008.
[57] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. W. Shavlik, and X. Zhu. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*, 2014.
[58] L. Golab, H. Karloff, F. Korn, D. Srivastava, and B. Yu. On generating near-optimal tableaux for conditional functional dependencies. *PVLDB*, 1(1):376–390, 2008.
[59] G. Grahne. *The Problem of Incomplete Information in Relational Databases*. Springer, 1991.
[60] S. Guha, N. Koudas, A. Marathe, and D. Srivastava. Merging the results of approximate match operations. In *VLDB*, 2004.
[61] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *SIGMOD*, pages 127–138, 1995.
[62] T. N. Herzog, F. J. Scheuren, and W. E. Winkler. *Data Quality and Record Linkage Techniques*. Springer, 2009.
[63] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. TANE: An efficient algorithm for discovering functional and approximate dependencies. *COMP. J.*, 42(2):100–111, 1999.
[64] T. Imieliński and W. Lipski, Jr. Incomplete information in relational databases. *JACM*, 31(4), 1984.
[65] M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa Florida. *J. American Statistical Association*, 89:414–420, 1989.
[66] S. Kolahi and L. V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, 2009.
[67] P. G. Kolaitis and E. Pema. A dichotomy in the complexity of consistent query answering for queries with two atoms. *Inf. Process. Lett.*, 112(3):77–85, 2012.
[68] C. P. Kruskal, L. Rudolph, and M. Snir. A complexity theory of efficient parallel algorithms. *TCS*, 71(1):95–132, 1990.
[69] A. Y. Levy. Obtaining complete answers from incomplete databases. In *VLDB*, pages 402–412, 1996.
[70] P. Li, X. L. Dong, A. Maurino, and D. Srivastava. Linking temporal records. *PVLDB*, 4(11):956–967, 2011.
[71] L. Libkin. Certain answers as objects and knowledge. In *KR*, 2014.
[72] J. Liu, J. Li, C. Liu, and Y. Chen. Discover dependencies from data - a review. *TKDE*, 24(2):251–264, 2012.
[73] 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.
[74] D. Loshin. *Master Data Management*. Knowledge Integrity, Inc., 2009.
[75] A. Meliou, W. Gatterbauer, S. Nath, and D. Suciu. Tracing data errors with view-conditioned causality. In *SIGMOD*, pages 505–516, 2011.
[76] D. W. Miller Jr., J. D. Yeast, and R. L. Evans. Missing prenatal records at a birth center: A communication problem quantified. In *AMIA Annu Symp Proc.*, pages 535–539, 2005.
[77] A. Motro. Integrity = validity + completeness. *ACM Trans. on Database Systems*, 14(4):480–502, 1989.
[78] F. Naumann and M. Herschel. *An Introduction to Duplicate Detection*. Morgan & Claypool Publishers, 2010.
[79] G. Ramalingam and T. Reps. On the computational complexity of dynamic graph problems. *TCS*, 158(1-2):213–224, 1996.
[80] S. Razniewski and W. Nutt. Completeness of queries over incomplete databases. *PVLDB*, pages 749–760, 2011.
[81] B. Salimi and L. E. Bertossi. From causes for database queries to repairs and model-based diagnosis and back. In *ICDT*, 2015.
[82] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *KDD*, pages 269–278, 2002.
[83] R. T. Snodgrass. *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann, 1999.
[84] S. Song and L. Chen. Efficient discovery of similarity constraints for matching dependencies. *TKDE*, 87:146–166, 2013.
[85] S. Staworko. *Declarative inconsistency handling in relational and semi-structured databases*. PhD thesis, the State University of New York at Buffalo, 2007.
[86] 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.
[87] B. ten Cate, G. Fontaine, and P. G. Kolaitis. On the data complexity of consistent query answering. In *ICDT*, pages 22–33, 2012.
[88] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. CrowdER: Crowdsourcing entity resolution. *PVLDB*, 2012.
[89] S. Whang, O. Benjelloun, and H. Garcia-Molina. Generic entity resolution with negative rules. *VLDB J.*, 18(6):1261–1277, 2009.
[90] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, 2005.
[91] J. Wijsen. Database repairing using updates. *TODS*, 30(3), 2005.
[92] Wikibon. A comprehensive list of big data statistics, 2012. *http://wikibon.org/blog/big-data-statistics/*.
[93] C. M. Wyss, C. Giannella, and E. L. Robertson. Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances. In *DaWak*, 2001.
[94] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. *PVLDB*, pages 279–289, 2011.
[95] W. Yancey. BigMatch: A program for extracting probable matches from a large file. Technical Report Computing 2007/01, U.S. Census Bureau, 2007.
[96] H. Zhang, Y. Diao, and N. Immerman. Recognizing patterns in streams with imprecise timestamps. *PVLDB*, pages 244–255, 2010.
[97] B. Zhao, B. I. P. Rubinstein, J. Gemmell, and J. Han. A bayesian approach to discovering truth from conflicting sources for data integration. *PVLDB*, 2012.

Mastering Situation Awareness:
The Next Big Challenge?

Dieter Gawlick Eric S. Chan Adel Ghoneimy Zhen Hua Liu
Oracle Corporation
500 Oracle Parkway Redwood Shores, California, USA
{dieter.gawlick, eric.s.chan, adel.ghoneimy, zhen.liu}@oracle.com

ABSTRACT
John Boyd recognized in the 1960's the importance of **situation awareness** for military operations and introduced the notion of the OODA loop (Observe, Orient, Decide, and Act). Today we realize that many applications have to deal with situation awareness: Customer Relationship Management, Human Capital Management, Supply Chain Management, patient care, power grid management, and cloud services management, as well as any IoT (Internet of Things) related application; the list seems to be endless. Situation awareness requires applications to support the management of data, knowledge, processes, and other services such as social networking in an integrated way. These applications additionally require high personalization as well as rapid and continuous evolution. They must provide a wide variety of operational and functional requirements, including real time processing.

Handcrafting these applications is an almost impossible task requiring exhaustive resources for development and maintenance. Due to the resources and time involved in their development, these applications typically fall way short of the desired functionality, operational characteristics, and ease and speed of evolution. We – the authors – have developed a model enabling the development and maintenance of situation-aware applications in a declarative and therefore economical manner; we call this model **KIDS** – Knowledge Intensive Data-processing System.

1. INTRODUCTION
We published a formal definition of the KIDS model in [4] which deals with the situation awareness as documented in [16][21]. An effective support of situation awarness is at the core of many applications. This has been attempted by developing appropriate data structures, procedural code, and processes. To simplify the application development, the IT community has for years abstracted out these three important aspects; with varying degree of success and level of maturity. Data management was the first technology to be abstracted out; we could not write modern and mission critical applications without the existing database technology. Process management in the form of workflow systems and knowledge management in the form of rules, analytics, etc., have significantly simplified the management of the other two aspects of the applications.

However, data, knowledge, and process management are treated as three orthogonal and independent aspects of applications and are therefore separately managed by three independent technology platforms. If we try to use these technologies to solve more complex problems, such as situation awareness, we find that data, knowledge and process management are increasingly intertwined. Unfortunately, there is no high-level model managing these three aspects in a consistent way. To deal with this challenge we propose a model that helps applications to manage data, knowledge, and processes in a synergistic, consistent, and well structured way. This model is based on the observation that humans typically solve problems using a loop where they capture facts, condense the facts by applying knowledge, reason about the findings, and act. More specifically, we capture *quantitative* facts, classify the facts to derive compact *qualitative* perceptions, assess the perceptions to arrive at one or more hypotheses, and use these hypotheses to formulate directives, unless we decide that nothing should or could be done. The resulting directives will be acted upon to create new facts – see Figure 1.

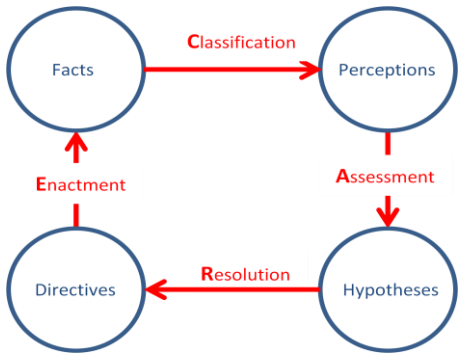


Figure 1: The KIDS CARE-loop

When we look at the CARE-loop from the perspective of processing we see the **CARE** (**C**lassification, **A**ssessment, **R**esolution, and **E**nactment) loop consisting of four distinct categories of knowledge

acting on and producing specific categories of data. If we look at the same loop from the data perspective we see **F**acts, **P**erceptions, **H**ypotheses, and **D**irectives representing four distinctive types of data. The CARE-loop represents a normalized workflow; Facts, Perceptions, Hypotheses, and Directives of a CARE-loop instance represents a **situation**.

Data describes things that are typically stored in modern files or databases without any distinction. Formal knowledge is stored in articles, books, application code, workflow, case management systems, or decision support systems without any distinction about which knowledge is best used at what time and how everything interacts. To make up for the deficiencies, the CARE-loop provides a much needed structure of interaction between data, knowledge, and processes management in applications with the goal of providing a framework for applications dealing with situation awareness.

2. RELATED WORK

Boyd developed a model to deal with situation awareness by proposing the OODA loop (Observe, Orient, Decide, and Act) [2] [15]. The OODA loop was originally applied to military operations. Boyd conjectured that any operational system faced with rapidly changing and incomplete knowledge of the reality can thrive by rapid iteration using the OODA loop by continuously interacting with the environment to assess and adapt to the changes. Our proposed KIDS CARE-loop is fundamentally an enhanced computerized version of the OODA loop.

Paper [13] by Horvitz and Mitchell presented the need for transforming large data (facts) into insights and decisions support for evidence-based decision making process. Paper [21], motivated by Barwise, Perry, and Devlin, defined the situation theory ontology that materialized a situation as an object like any other physical or conceptual objects. In KIDS model [4], a situation is a quadruple of fact, perception, hypothesis, and directive objects. Paper [6] articulates a Data-Action-Model loop that has some conceptual similarity to the CARE-loop of KIDS.

3. THE KIDS MODEL

KIDS provides a model to support situation awareness by structuring applications according to the CARE-loop (see figure 1) using three core technologies: Data, knowledge, and process managament. The management of data is a relatively mature technology, which is currently going through a rapid evolution. KIDS contributes to this evolution by defining and managing four distinct categories of data:

Facts are what we measure in the world around us. The number, rate, and quantity of facts make it in many environments impossible for the human cognitive system to deal with these measurements directly. Therefore, we have to compact these facts in an ever increasing way for human consumptions.

Perceptions are compact, temporal, and qualitative representations of facts. They are optimized for use by the human cognitive system; they represent the most important characteristics of evolving situations, with high focus on exceptions. Perceptions depend on the perspective of the user.

Hypotheses are descriptions of possible root causes explaining the facts and the perceptions leading to the directives.

Directives describe what needs to be done to react to a specific set of facts, perceptions, and hypotheses. Directives specify action plans often in the form of workflows or processes. Obviously, the directive will most likely influence the evolving facts and also determine which facts should be captured in the future.

Managing these data requires a wide variety of data types/structures, extensibility, declarative access across data types/structures, time travel, flexibility in evolving data structures (support for well structured data as well as data first/structure later or never), OLTP, analytic, and so forth. There is also a need for extended functionality such as (fine grain) security, and provenance. Last but not the least important operational characteristics are disaster recovery, high availability, reliability, scalability, performance, and rapid development tools. Obviously, the management of data requires such a broad functional and operational support that is only available in mature and widely used databases. The collection and management of facts does not require the classical transaction model and limited loss of data may be acceptable; provenance requirements will drive these specifications. Mature databases need to optimize the management of facts and provide support with significantly reduced resource consumption. This can be achieved with well understood technologies; unlimited scalability is very achievable as well.

Knowledge is divided into four categories as well: Classification, Assessment, Resolution, and Enactment. These categories are based on the mode of reasoning that is required to process each category of data. A substantial subset of each category of knowledge can be automated by suitable computation models.

The **Classification** knowledge - transforming facts into perceptions - is primarily represented by **deductive** reasoning. Some Classification knowledge that produces prediction or norm may involve inductive reasoning as well. The computation model for

classification includes Support Vector Machines [1], naïve-Bayes classifier [22], Multivariate State Estimation Technique [5], Clustering, Association Rules, Decision Trees, Cognitive computing, etc.

The **Assessment** knowledge - transforming perceptions into hypotheses - is typically implemented by **abductive** reasoning that derives the *Hypotheses* from *Perceptions*. The computation model for assessment includes Bayesian Belief Network [22] and Least-Squares Optimization or Regression of solutions for inverse problem [9].

The **Resolution** knowledge - transforming hypotheses into directives - involves **making decisions** under the uncertainty of outcomes by considering the relative merit of the different outcomes and the associated payoffs/costs. The computation model for resolution includes Bayesian Belief Network extended with decision nodes and payoff/cost nodes, known as Influence Diagrams [14], Dempster-Shafer theory [7][25], Decision Trees, and Prognosis of Remaining Useful Life [5].

The **Enactment** knowledge - transforming directives into actions (and new facts) - involves **control structures** encoded in scripts, plans, schedules, BPEL workflows, and business processes in BPMN. The actions often include capturing of new facts.

Knowledge will be applied in the proper sequence as specified by the CARE-loop. In some cases not all steps of the CARE-loop need to be executed. Knowledge - including each version of it - has to be stored in databases to take advantage of the state management with declarative access and manipulation. Knowledge has to be applicable ad-hoc and in real time. An important ad-hoc use case is the ability to revisit data (especially facts and perception) with new knowledge to find out what has been missed and has been overrated.

Additionally, each category of formal knowledge is complemented by tacit knowledge. Applications may also require social networking services where we can profile the tacit knowledge [11] and social preferences of the actors in the system. This allows identifying the most qualified individuals or teams for a task by adjusting the tacit knowledge profiles based on recent activities to ensure that profiles are as up-to-date as possible.

An important requirement of applications is the ability for continuous improvements. Methods to enable continuous improvements are:

- Improve the rules, queries, models, and procedures leveraging insights from users and domain experts.
- Re-run the existing models using additional data or new algorithms.

Knowledge can be exchanged between experts of a field; this exchange should be as formal as possible. Papers should be considered as the equivalent of Venn Diagrams helping to understand intuitively models or whatever formalism is used. Obviously, any new knowledge has to be carefully reviewed before it is generally used. KIDS allows the use of evolving and existing knowledge concurrently and is able to show both results. The application of knowledge is driven by the CARE-loop, which is a standardized process management structure.

4. KIDS USE CASES

In the following sections we discuss three use cases, which are in various states of development.

4.1 Cloud Operation

A basic postulate of cloud computing is the economy of scale by consolidation and pooling of the physical resources and providing virtually unlimited resources by dynamic resource management. The control system needs to manage the dynamic entity model to provide an accurate awareness of the system which changes due to frequent new software releases, patches for bug fixes, hardware upgrades, capacity scale out, and dynamic resource allocation. To conform to service level agreements (SLA), the cloud operations need continuous monitoring of vital signs and predictive diagnosis capability to detect and circumvent the SLA violations. A typical cloud operation has to monitor millions of hardware and software components of the data centers. The reactive fault detection and manual diagnosis techniques of traditional IT operations are labor intensive, require extensive domain expertise, often too little or too late in responsiveness, often resulting in disproportionate responses involving restart of large parts of the system instead of isolating and fixing the faulty components, and obviously cannot scale out for the cloud.

Cloud operation can only thrive by rapid iteration of the CARE-Loops to get inside the dynamics of seasonal cycles, load trends, load spikes, system response characteristics, transient glitches, gradual degradations, aging, and performance drifts of millions of components in the environment. The framework must automatically transform facts from different entities at different points of time into a representation of comprehendible and actionable perceptions for effective human or automated decision making. The framework must also perpetually evolve the entity model by discovering new entities, new relationships, and new knowledge that transforms vital measures about entities into appropriate representations [21]. This involves managing terabytes of data combined with read-time analytics in Big Data systems [4] and large scale management of millions of CARE-Loop

instances using Oracle bi-temporal database, expression filters, registered queries, and orchestration engine to integrate many inference engines.

The KIDS model is a viable framework to enable the information fusion and situation awareness at the level of the complexity and heterogeneity needed to support the effective sequence of facts, perceptions, hypotheses, and directives in CARE-Loops. For example, a classification model predicts a load surge in the next two months based on an annual seasonal trend model learned from abduction. An assessment model hypothesizes that the CPU and memory usage will exceed the capacity during the seasonal peak. It issues a directive to scale out the number of virtual machines for the next two months. The system response to such a change would be observable in the new facts.

4.2 Software & Hardware Product Support

SW and HW support applications facilitate collaborative and iterative problem solving activities involving product support and customer's IT personnel. The objective of such activities is to minimize system unavailability by minimizing the time required to either, find and apply existing knowledge (tacit, explicit, or automated) to resolve known issues, or to discover remedy for new issues. And therefore, justifies the necessity for maximizing the automation of diagnosing and fixing known issues whenever economically possible, in order to free support and customer personnel to focus on newly emerging issues that require a great deal of collective human experience and intelligence. To achieve economical automation, it is essential to ensure that data and knowledge about product issues, encountered throughout the product lifecycle (in bug database, support tickets), are captured with precise articulation and provenance. This includes consistent terminology with accurate definitions, accurate and validated causality relationships, system configuration, and personnel's contributions. For example, computing a perception about invalid objects in the data dictionary with potential states (hi, lo, none) implies that we need to define the source where the fact (a query, a view, a diagnostic tool). A hypothesis such as database migration failure could be caused by invalid objects. And the resolution for such hypothesis could be an action plan that calls for recompilation the dictionary and run a query to evaluate the state of the dictionary after such recompilation, which will generate a new set of facts that will help determine whether our resolution worked or not. Such articulation and provenance enables accurate statistics for the likely recurrence of an issue and the degree of complexity in recognizing and resolving the issue automatically or semi-automatically. Next, it is essential to standardize the process for diagnosing issues by leveraging data collected for provenance. For example, diagnosing ORA-4031 Error (an oracle database memory allocation error), are most effectively done using facts from alert log and ORA-4031 trace. Such standardization aims at establishing standardized data collection, standardized parsing and interpretation of such data, standardized diagnoses, and standardized remediation methods as well as standardization of the entire process of issue resolution.

4.3 Patient Care

In patient care facts are representing various measurements and images such as vitals, blood tests, and ultra sound. Classification is used to express perceptions of concerns, such as a high and increasing risk for an impending heart attack. The assessment identifies the root cause such as fibrillations, which will be used as the working hypothesis. The resolution determines the directives: what medicine to take or procedure to perform or, if in doubt, what other facts need to be captured how often and when. Finally, the directives are acted upon, and new facts are captured to follow the progress of the patient; e.g., the perception will be re-computed to provide an updated high level view and the hypothesis will be re-evaluated. The hypothesis may entail prediction of expected future values. In this case the doctor can ask KIDS to inform him/her if the evolution of the patient is not as expected; the context determines the exact meaning of this sentence. The medical background of different doctors can lead to different – even conflicting – perceptions; this is not a problem for KIDS [10] [12].

Any new knowledge can be adapted immediately and not only be used to treat existing and future patients but also be used to review the history of former patients. Significant deviations in the perception and hypothesis based on new knowledge will be brought to the attention of the doctors [23].

5. ACTIVITIES DRIVING THE EVOLUTION OF KIDS

Within Oracle the KIDS model is actively used for the management of cloud services and customer care. In both cases the use of the KIDS model significantly helped the evolution of these projects. Other groups are in the process of evaluating KIDS.

A research project with IIT (Illinois Institute of Technology) is focused on providing unambiguous provenance in bi-temporal environments. Provenance allows the use of qualitative languages for perception without losing the precision that is inherent in facts [19][20]. Another research project with the University of Bonn (Germany) is focused on a detailed architecture for perceptions, with flight supervision as

the primary use case. This use case requires significant advances in temporal spatial technology [18]. An additional research project with the University of Buffalo is focused on estimating the uncertainty of the results after applying knowledge to data. This uncertainty can among others be the result of imprecise, insufficient, or decaying data as well as imprecise knowledge [17].

While the above projects are funded by Oracle, there are other informal collaborations with universities without funding from Oracle; e.g. with Ken Baclawski from the Northeastern University [21].

6. OPPORTUNITIES & CHALLENGES

A very important value of databases is that they provide a declarative data management abstraction to manage persistent states. Furthermore, the success of databases is due to the underlying set-based abstract algebra. However, databases currently provide declarative abstraction for data only; they do not provide declarative abstraction to manage knowledge, processes, nor the interactions among data, knowledge and processes in the KIDS CARE-loops. Instead, these tasks have been procedurally coded in every application that needs KIDS semantics. We think it is time for databases to extend its declarative state management service to provide KIDS abstraction. There are a number of opportunities and challenges.

Theoretical Foundation: The set-based relational model is the theoretical corner stone for databases. To support KIDS semantics, however, we need set algebra that can embrace and reason about transformations among set elements and discover morphisms among transformations. Investigation of the KIDS abstraction using category theory [8] can lead to a theoretical foundation of KIDS and new principles of data, knowledge, and process management systems.

Declarative Language Support: Based on a theoretical foundation, we can extend the declarative languages such as DDL, DML and query component of SQL so that they can manage the facts, knowledge, processes, and CARE-loops. For example, a DDL extension can allow the applications to manage the life cycles of facts, knowledge, processes in a repository. A DML extension can support loading instances of KIDS elements into a repository, establishing the links among the KIDS elements in the CARE-loops, and navigating the CARE-loops stored in the repository, all with full integrity checks. It is essential to support automated CARE-loop execution. Finally, a comprehensive query language extension is needed to query not only facts, but also knowledge, processes, CARE-loops, and their interactions.

Paper [24] explored the database extensibility mechanism to accomplish flexible schema data management. We will continue to explore the database extensibility framework, which provides an engineering foundation, to naturally evolve databases to support KIDS abstraction. Certainly, designing a proper indexing and storage structure to provide efficient access of knowledge and process will require a lot of innovative techniques.

Time Dimension Support: KIDS model has both implicit and explicit dependency on tracking the valid time and transaction time dimensions for data and knowledge; KIDS semantics is based on the valid times of the data and knowledge when transforming them through the CARE-loops. The bi-temporal and provenance support of modern databases provides a valuable direction to understand data management in time. However, they need to be further enhanced to support the notion of multi-version of data with branching so as to support "what if" KIDS analysis. The goal is to eventually make the KIDS system behave like a time machine with a notion of the parallel universe of KIDS leveraging the idea of data version and branching.

Performance & Scalability: A DBMS supporting KIDS is expected to support the rapid growth of the number of KIDS instances, in particular, the growth of fact. While some of the facts can be discarded after a short time period, others may need to be kept and made easily accessible for an extended period of time. This requires a highly optimized data service providing data flexibility, durability, exactly once semantic, security, compression, compaction, time travel, provenance, along with extreme performance and scalability.

Practical KIDS Migration Strategy with Social Networking: Few applications have the luxury to start from scratch; therefore, a KIDS migration strategy is important. One idea is to keep the mature legacy applications running to support the existing functionality while creating a 'shadow' application based on the KIDS model. The shadow application can crawl the information of existing systems to automatically categorize the data, knowledge, process, and CARE-loop interactions so that application users can gradually gain the benefits of the KIDS model through the 'shadow' applications. Such a shadow KIDS application can be tuned based on the preferences of the application users by integrating with social networking services [3]. Like the Google Internet index, the shadow KIDS application provides the KIDS index for the applications data to let the community to search for matching problems and contribute solutions. Such a KIDS index can represent a situation aware service.

7. CONCLUSIONS

KIDS provides a framework to organize and support the applications to deal with situation awareness. KIDS integrates three technologies: data, knowledge, and process management. It does so by providing a high level model that normalizes the use of these technologies using the CARE-loop. By doing so, KIDS allows applications designers to develop applications with a new methodology leading to applications that are well structured and can evolve in real time. The evolution can be achieved through collaboration among many developers working on different applications with some overlap.

KIDS offers an incentive for the research communities of the three core technologies to work more closely to better leverage each other's work.

8. ACKNOWLEDGMENTS

The authors extend their thanks to colleagues who helped with many constructive suggestions and critics. Especially, we thank Andrew Mendelsohn, Rafiul Ahad, and Vikas Arora for encouraging and supporting this work. We like to thank Ken Baclawski, Andreas Behrend, Boris Glavic, Ying Hu, and Oliver Kennedy for their many contributions to the refinement of the KIDS model.

9. REFERENCES

[1] Boser, B.E., Guyon, I.M., Vapnik, V.N., 1992, *A training algorithm for optimal margin classifiers*. Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, ACM Press.

[2] Boyd, J.R., 1976. *Destruction and Creation*. *U.S.* Army Command and General Staff College.

[3] Chan, E.S., Behrend, A., Gawlick, D., Ghoneimy, A., Liu, Z.H., 2012, *Towards a Synergistic Model for Managing Data, Knowledge, Processes, and Social Interaction, SDPS-2012*, Society for Design and Process Science.

[4] Chan E.S., Gawlick D., Ghoneimy A., and Liu Z.H., "*Situation Aware Computing for Big Data*," SemBIoT 2014

[5] Cheng, S., Pecht, M., 2007, *Multivariate State Estimation Technique for Remaining Useful Life Prediction of Electronic Products*, Association for the Advancement of Artificial Intelligence.

[6] Crankshaw, D., Bailis, P., Gonzalez, J., Li, H., Zhang, Z., Franklin, M., Ghodsi, A., Jordan, Mi: *The Missing Piece in Complex Analysis: Low Latency, Scalable Model Management and Serving with VELOX*, CIDR 2015.

[7] Dempster, A.P., 1968, A generalization of Bayesian Inference. *Journal of the Royal Statistical Society*.

[8] D.I. Spivak: "*Category Theory for the Scientists*," ISBN-13: 978-0262028134

[9] Fletcher, R., 1970, *Generalized Inverses for Nonlinear Equations and Optimization*. *Numerical Methods for Non-Linear Algebraic Equations*. Gordon and Breach, London.

[10] Gawlick, D., Ghoneimy, A., Liu, Z.H., 2011, *How to Build a Modern Patient Care Application*. HEALTHINF.

[11] Gilmour D.L, et al. 2003, *Automatic Management of Terms in a User Profile in a Knowledge Management System*. United States Patent 6,640,229.

[12] Guerra, D., Gawlick, U., Bizarro, P., Gawlick, D., 2011, *An Integrated Data Management Approach to Manage Health Care Data*. BTW 2011.

[13] Horvitz, E., Mitchell, T., 2010. *From Data to knowledge to Action: A Global Enabler for the 21st Century*. *Data Analytic Series*, Computing Community Consortium.

[14] Howard, R.A., Matheson, J.E., 1984, *Influence Diagrams. Readings on the Principles and Applications of Decision Analysis*, v.2. Strategic Decisions Group, Menlo Park, CA.

[15] http://en.wikipedia.org/wiki/OODA_loop

[16] http://en.wikipedia.org/wiki/Situation_awareness

[17] http://mjolnir.cse.buffalo.edu/

[18] https://www3.uni-bonn.de/idb/research/states

[19] http://www.cs.iit.edu/~dbgroup/research/gprom.php

[20] http://www.cs.iit.edu/~dbgroup/research/oracletprov.php

[21] Kokar, M.M., Matheus, C.J., Baclawski, K., (2009), *Ontology-based situation awareness*, Journal Information Fusion, Vol 10, Issue 1.

[22] Koller, D., Friedman, N., 2009, *Probabilistic Graphical Models: Principles and Techniques*, The MIT Press, Cambridge, Massachusetts.

[23] Liu, Z.H., Behrend, A., Chan, E., Gawlick, D., Ghoneimy A., *KIDS - A Model for Developing Evolutionary Database Applications*. DATA 2012: 129-134.

[24] Liu Z.H., Gawlick, D., *Management of Flexible Schema Data in RDBMSs*, CIDR 2015

[25] Shafer, G., 1976, *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ.

Robust Query Optimization Methods With Respect to Estimation Errors: A Survey

Shaoyi YIN, Abdelkader HAMEURLAIN, Franck MORVAN
IRIT Laboratory, Paul Sabatier University, France
firstname.lastname@irit.fr

ABSTRACT

The quality of a query execution plan chosen by a Cost-Based Optimizer (CBO) depends greatly on the estimation accuracy of input parameter values. Many research results have been produced on improving the estimation accuracy, but they do not work for every situation. Therefore, "robust query optimization" was introduced, in an effort to minimize the sub-optimality risk by accepting the fact that estimates could be inaccurate. In this survey, we aim to provide an overview of robust query optimization methods by classifying them into different categories, explaining the essential ideas, listing their advantages and limitations, and comparing them with multiple criteria.

1. INTRODUCTION

The query optimizer is an indispensable component in a relational DBMS engine. Since the publication of the System-R paper [61], cost-based optimizers have been widely adopted. For a given query, the optimizer enumerates a large number of possible execution plans, estimates the cost of each plan using a cost model, and picks the one with the lowest estimated cost. The parameters of the cost model fall into two categories: the database profile and the available amount of system resources. The database profile contains mainly: (1) basic information which represents the properties of the data, e.g., relation sizes and number of distinct attribute values, which is thereafter called catalog statistics, and (2) derived information specific to a given query, which is mainly the cardinality (i.e., number of tuples returned by a relational operator). The accuracy of parameter values has a significant impact on the quality of the chosen execution plan.

It has been shown that, even if estimation errors on the basic information are small, their transitive effect on estimates of the derived information can be devastating (e.g., the error propagates exponentially with respect to the number of joins) [42]. Consequently, the optimizer may choose the wrong plan. However, due to non-uniform distribution of attribute values and correlations between attributes, the cardinality estimation problem remains very challenging. Many efforts have been made to improve the estimation accuracy by constructing and maintaining more precise catalog statistics, using histograms [43], sampling [47, 48, 56], maximum entropy [51] and probabilistic graphical model [66, 67], etc. Nevertheless, they do not work well for every situation, in particular for complex predicates and skewed data. In addition, the amount of system resources available may change dynamically during query execution.

Having accepted the fact that parameter value estimates could be inaccurate or even missing, it is still desirable to minimize the plan sub-optimality risk, so the notion of "robustness" was introduced in the query optimizer. Informally, robustness is related to the ability of error resistance. However, there is yet no consensus on a formal definition of robustness for query optimizer. Recently, Graefe et al. have organized two seminars [31, 33] and one research panel [32] on the "robust query processing" topic. Before that, the authors tried to visualize and benchmark the robustness of query execution [30, 69]. [69] distinguished three types of robustness: query optimizer robustness ("the ability of the optimizer to choose a good plan as expected conditions change"), query execution robustness ("the ability of the query execution engine to process a given plan efficiently under different run-time conditions") and workload management robustness ("characterizes how database system performance is vulnerable to unexpected query performance"). Each type deserves an in-depth study. In this survey, the focus is on the query optimizer robustness. To make the concept more concrete, we propose the following definition: a query optimizer is robust with respect to estimation errors, if it is able to find a plan (or several plans) such that the query response time T is not greater than $(1 + \varepsilon) * T(P_{opt})$ despite estimation errors, where $T(P_{opt})$ is the query response time by executing the optimal plan P_{opt} implied by exact input parameter values and ε is a user-defined tolerance threshold. Note that obtaining efficiently all the exact parameter values to find P_{opt} is challenging [19], but outside the scope of our survey.

The above statement is the main objective for a robust query optimizer. Although it has not yet been achieved completely, some "best-effort" research results are worth being analyzed. Some of them have been analyzed in previous surveys under the terms like "dynamic query optimization" [22, 52] or "adaptive query processing" [6, 24, 34, 35]. Indeed, dynamic or

adaptive query optimization is one way to improve robustness. However, as will be seen, there are further interesting approaches proposed for this purpose. The aim of this survey is to give an overview of the representative robust query optimization methods, including many recent proposals [14-17, 25, 26, 28, 39, 40, 53-55, 58, 65] not yet covered in any survey. Note that we are interested here in relational DBMS engines running in different environments (single-node, distributed or parallel), but not in query execution engines based on the Map-Reduce model. The major contributions are: (1) proposing a two-level classification framework for robust query optimization methods, (2) pointing out the inherent advantages and limitations of each method, as well as the relationship between them, and (3) comparing the methods using multiple well-chosen criteria.

The remainder of the paper is organized as follows. In Section 2, we describe the proposed classification framework and choose multiple criteria for later comparisons. Section 3 and Section 4 are organized in accordance with the classification framework. We analyze some representative methods and compare them using the chosen criteria. In Section 5, we present a global comparison of the main approaches and their principle strategies. Finally, we conclude the paper in Section 6.

2. CLASSIFICATION AND CRITERIA

2.1 Classification Framework

Depending on the query optimizer output (for a query), we distinguish the following two approaches: (1) single-plan based approach, where the output of the optimizer is a single plan satisfying the optimization criteria, and (2) multi-plan based approach, where the output of the optimizer is a set of plans. The main difference between them is that the latter often requires a more sophisticated execution model.

The methods which adopt the single-plan based approach rely mainly on the following three strategies: **(1) Cardinality Injection** (CI). Instead of deriving cardinalities from basic database statistics, the optimizer tries to obtain directly the cardinality values for some operators. The main objective is to overcome the correlation problem (w.r.t. multi-predicate selections) and limit the error propagation effect (w.r.t. joins). One way is to collect information from execution feedback of previous queries; another way is to execute some important sub-queries during optimization. **(2) Plan Modification** (PM). The system collects statistics and detects estimation errors during query execution, then reacts to them by modifying the plan dynamically. Sometimes, the optimizer may be recalled repeatedly. **(3) "Robust Plan" Selection** (RPS). Instead of finding an "optimal" plan, the optimizer chooses a "robust" plan, i.e., a plan which is less sensitive to estimation errors. Note that, these strategies are not mutually exclusive. They may be used together by the same optimizer. In addition, some of them are even compatible with strategies adopted by a multi-plan based method. We present them as methods using the single-plan based approach, because they constitute the core contributions, while for the multi-plan based approach, they only serve as a tool.

The methods which adopt the multi-plan based approach rely mainly on the following strategies: **(1) Deferred Plan Choosing** (DPC). The optimizer proposes several potentially optimal plans, and the final choice is taken during execution time. One way is to run these plans in a competition mode. Another way is to start with one plan and smoothly switch to others if necessary. **(2) Tuple Routing through Eddies** (TRE). Avnur et al. proposed a special operator called "eddy" [5] which receives all base relation tuples and intermediate result tuples, then routes them through the relational operators in different orders. Since different tuples may follow different routing orders, and each routing order corresponds to a specific execution plan, we consider this mechanism belongs to the multi-plan based approach. **(3) Optimizer Controlled Data Partitioning** (OCDP). The essential idea is that the optimizer partitions the data explicitly according to their inherent characteristics such as skewed distribution or correlations, such that different optimal plans may be executed for different data partitions. The main difference between these strategies lies in how to decide which plan is used for which data. With DPC, only one plan will be finally chosen and used for the complete dataset, even though the optimizer generates multiple plans as the output; with TRE, the eddy operator chooses a routing order (i.e., a plan) for each tuple or a group of tuples, and the decision is based on local statistics collected by the eddy; with OCDP, the mapping between sub-datasets and multiple plans is decided by the optimizer, based on global statistics.

2.2 Comparison Criteria

For each approach, different methods will be compared using the following criteria. The first three criteria define the application scope; the fourth is related to query performance; and the last concerns the software engineering aspect. Here, we list the options for each criterion and their abbreviations.

C1: Estimation Error Sources. The existing methods deal with one or several of the following estimation error sources: non-uniform data distribution (DD), data correlation (DC), statistics obsolescence due to data modification (DM), missing statistics (MS, e.g., for complex predicates or remote data sources), unavailability of resources (UR), data arrival delay (AD), data arrival rate changing (AR), and so on.

C2: Target Query Types. Some methods aim at optimizing the currently running query (C); some serve for future executions (F) of the same (Sa) or similar (Si) queries; others only deal with predefined parametrized queries (P).

C3: Target Optimization Decisions. Due to estimation errors, the optimizer may make wrong decisions in the following aspects: base relation access methods (AM), join methods (JM), join order (JO), operator execution order (OEO), execution site (ES) of an operator, CPU allocation (CA), memory allocation (MA), parallelism degree (PD), partitioning key (PK), etc. Different methods may cover different aspects.

C4: Performance Degradation Risk. Sometimes, for a given query, the "robustly" chosen plan becomes even worse than the "naively" chosen plan (i.e., the plan chosen by a classical optimizer). This may be caused for various reasons, e.g., too much preparation work should be done before finding an ideal plan, costly follow-up work such as removing duplicates may be required, wrong decisions are made on whether to discard or reuse intermediate results, or only part of the uncertainty is removed but the remaining part defeats the new solution. To compare the methods, we classify the performance degradation risk as high (H) if there is no worst-case guarantee, low (L) if the degradation is constant or linear to the original performance, and medium (M) if the maximum degradation is bound, but non-linear to the original performance. Sometimes, the risk level is low, but only under certain conditions (LC). It is also possible that there is no degradation risk (N) or that the user is allowed to choose a certain risk level (UC).

C5: Engineering Cost. After many years of maintenance, commercial DBMS engines become extremely complex. Modifications should be made very carefully to avoid system regression. We assess the engineering cost this way: low (L), if only adding a stand-alone module which could be turned on/off easily; medium (M), if just modifying a few modules of the optimizer or the executor; or high (H), if rewriting most of the optimizer or the executor code.

3. SINGLE-PLAN BASED APPROACH

3.1 Cardinality Injection

"Cardinality Injection" is a term introduced by Chaudhuri in [18], meaning that the cardinality information is obtained from other sources rather than derived from basic database statistics. Two different ways mainly exist to obtain the cardinality: learning from previous query executions and running some sub-queries during optimization process.

3.1.1 Exploiting the execution feedback

LEO (DB2's LEarning Optimizer) [63] is the most representative work of using query execution feedback for cardinality estimation. LEO captures the number of rows produced by each operator at run-time, by adding a counter in the operator implementation. Then, the <predicate, cardinality> pairs are stored in a feedback cache, which can be consulted by the query optimizer in conjunction with catalog statistics when optimizing a future query.

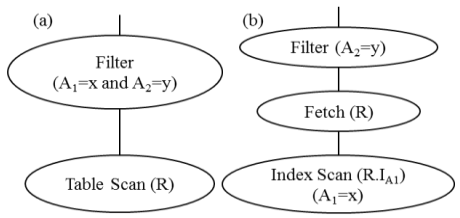


Figure 1. Example to show the limitation of LEO

However, using this mechanism, we can only obtain the cardinalities of a subset of predicates used by the optimizer to estimate the costs. For example, given the query "select * from R where A_1=x and A_2=y", if the executed plan is Plan (a) in Figure 1, we can obtain the cardinality Card(A_1=x and A_2=y) from the first execution, so in the future, the cost estimation for Plan (a) will be more accurate. Nevertheless, to estimate the cost of Plan (b), the optimizer still needs to derive the cardinality Card(A_1=x) from catalog statistics. To solve this problem, the pay-as-you-go (PAYG) framework [17] uses proactive plan modification and monitoring mechanisms, in order to obtain cardinality values for some sub-expressions (e.g., Card(A_1=x)) which are not included in the running plan. For example, when running Plan (a), in addition to counting the number of rows satisfying the predicate "A_1=x and A_2=y", the operator keeps another counter, which is increased every time when "A_1=x" is true for an input tuple. Thus, in the future, the optimizer can also estimate the cost of Plan (b) more precisely. Obviously, this will increase the overhead of query execution, however, the DBA or the user is allowed to specify a limit on the additional overhead for a query.

Some other methods [1, 13, 20, 62] use feedback information to refine catalog statistics. They are less general than the above CI methods, because each of them limits to a specific statistical data structure. They have gone further with regard to previous methods mentioned in Section 1 which maintain precise catalog statistics, and most of them have been covered by earlier surveys [43, 49], so are not discussed in detail in this paper.

3.1.2 Querying during optimization process

This idea was first adopted for multi-databases (MIND system [27]). In case that there are not enough statistics for generating a complete execution plan, the query optimizer first decomposes the query into sub-queries, sends them to different remote sites, and then

decides the order of inter-site operators (e.g., joins) based on the sub-query results. A recent work of Neumann et al called "Incremental Execution Framework" (IEF) [53] adopted a similar principle to optimize queries in uni-processor environment. The main steps are: (1) construct the optimal execution plan using the cost model, (2) identify sensitive plan fragments, i.e., the fragments whose cardinality estimation errors might lead to wrong plan decisions for higher level operators, (3) execute those plan fragments, materialize the results, and thus retrieve the cardinality (i.e., the number of tuples), and (4) find a new optimal plan using the obtained cardinality. IEF to some extent removes the uncertainty of the cardinality estimation. However, it still has some limitations. For example, it identifies the sensitive query fragments based on "estimation error rates", but this information is often unknown or inaccurate.

Different from the above work, Xplus [40] focused on offline tuning of repeatedly-running queries. When a query plan is claimed to be sub-optimal, the optimizer picks some candidate (sub)plans heuristically, calls the executor to run these (sub)plans iteratively, collects the actual cardinalities from these runs, and stops the iterations when finding a plan which is δ% better than the original one, where δ is a user-defined threshold. This proposal is only suitable for read-only applications running in a stable environment.

3.1.3 Comparison

In Table 1, the above methods are compared using the five criteria defined in Section 2.2. C1: They deal with all kinds of cardinality estimation errors, except that Xplus is not resistant to data modifications. C2: The feedback-based methods serve for future similar queries; others optimize the currently running query, except that Xplus tunes a query for future runs. C3: LEO, PAYG and Xplus try to improve all kinds of decisions; IEF focuses on JM and JO. C4: As LEO and PAYG only correct part of the estimation errors, the uncertainty of other values may lead to a worse plan. However, for repeatedly-running queries, after several runs, a stable plan can be obtained. Normally, PAYG could converge earlier, because it collects information more efficiently. IEF introduces some degradation risk due to materialization, but this cost is rather limited. Xplus does not have degradation risk, because it uses only exact cardinalities. C5: LEO comprises four components: one to save the optimizer's plan, a monitoring component, a feedback analysis component and a feedback exploitation component. The analysis component is a stand-alone process, and the others are minor modifications to the DB2 server. PAYG needs to modify the optimizer generated plan and identify important expressions to run, so it is more complicated than LEO. IEF optimizer needs to identify critical plan fragments to execute, interact with the executor, and materialize intermediate results, but the plan executor stays the same, so we consider the engineering cost as medium. Finally, Xplus can work as a stand-alone module.

Table 1. Comparison of cardinality injection methods

Criterion / Method	C1: Estim. error sources	C2: Target query	C3:Target opt. decisions	C4:Deg. risk	C5:Eng. cost
LEO	DD,DC,DM,MS	F, Si	All	LC	L
PAYG	DD,DC,DM,MS	F, Si	All	LC	M
IEF	DD,DC,DM,MS	C	JM,JO	L	M
Xplus	DD,DC,MS	F, Sa	All	N	L

3.2 Plan Modification

With this different strategy, the optimizer uses catalog statistics to generate a plan. However, the execution plan is monitored at run-time. Once a sub-optimality is detected, the plan is modified: either by rescheduling, or by re-optimization. Rescheduling is to update only the execution order of the operators or to update the order of the base relations. Re-optimization is to generate a new plan for the remainder of the query using run-time collected statistics. Based on the new cost estimation, re-optimization might throw away the intermediate results and start a new plan from scratch.

3.2.1 Rescheduling

In distributed environments, the relations participating in a query plan are often stored in remote sites, and the arrival of data may be delayed. In this situation, to avoid idling, Amsaleg et al proposed a query plan scrambling algorithm [2]. The algorithm contains two phases: (1) materializing sub-trees. During this phase, each iteration of the algorithm identifies a plan fragment that is not dependent on any delayed data (the fragment is called a "runnable sub-tree"), then the fragment is executed and the result is materialized. (2) Creating new joins between relations that were not directly joined in the original query tree. When no more runnable sub-trees can be found by Phase 1, the scrambling algorithm moves into Phase 2, so that the plan execution could continue.

Query plan scrambling can improve the response time in many cases, but it only deals with the initial delay (i.e., the arrival of the first tuple is delayed). If the delay happens during the execution of the fragment, it is blocked and has to wait. To solve this problem, Bouganim et al proposed a dynamic query scheduling strategy (DQS) [12] that interleaves the scheduling phase and the execution phase. Each time, the scheduler only schedules the query fragments that can be executed immediately (i.e., all their inputs are available and there is enough memory). These fragments are executed concurrently. During execution, the data arrival rate is monitored

continuously. Once a problem is detected or the execution is finished, rescheduling is triggered. If the rescheduling cannot solve the problem, a re-optimization (see Section 3.2.2) may be triggered.

3.2.2 Re-optimization

The dynamic Re-Optimization (ReOpt) algorithm proposed by Kabra et al [46] detects the estimation errors during query execution and re-optimizes the rest of the query if necessary. At specific intermediate points in the query plan, statistics collector operators are inserted to collect various statistics. During query execution, the collected statistics are compared with the estimated ones. If there is a large difference, some heuristics are triggered to evaluate whether a re-optimization is beneficial. If so, the optimizer is recalled to modify the execution plan for the rest of the query. Instead of suspending a query in mid-execution, the currently executing operator is run to completion and re-directs the output to a temporary file on disk. Then, SQL corresponding to the rest of the query is generated by using this temporary file. The new SQL is re-submitted to the optimizer as a regular query. In ReOpt, if the difference between the collected parameter value and the estimated one exceeds a threshold, the re-optimization procedure will be considered. A later work [50] argued that this threshold is chosen arbitrarily so could be blind. [50] introduced POP algorithm which uses the "validity range" concept of the chosen plan for each input parameter. If the actual value of the parameter violates the validity range, a re-optimization is triggered; otherwise, the current plan continues execution. The violation of validity ranges is detected by a CHECK operator. Another difference between POP and ReOpt is that, when a re-optimization is triggered, ReOpt always modifies the plan for the remainder of the query in order to reuse the intermediate results, while POP allows the optimizer to discard the intermediate results and choose a completely new plan, if the cost model estimates that is to be beneficial. Han et al [37] extended the POP algorithm to a parallel environment with a shared-nothing architecture.

Continuous query optimization (CQO) [15] extends ReOpt for query optimization in massively parallel environments. On the same principle, query execution is continuously monitored, run-time statistics are collected and the parallelism degree or the partition key choice is dynamically modified.

Re-optimization is also applied for recursive queries in [28], where the estimation errors may be propagated to later iterations. The authors proposed two mechanisms. The first mechanism is called "lookahead planning": the optimizer generates plans for k iterations, the executor executes them, provides collected statistics, and then the optimizer generates plans for the next k iterations using these statistics, and so on. The second mechanism is called "dynamic feedback": it detects the divergence of the cardinality estimates and decides to re-optimize the remainder of the running plan if needed. The proposal is named "lookahead with feedback" (LAWF).

Bonneau et al [11] and Hameurlain et al [36] focused on the re-optimization problem for the shared-nothing architecture and the multi-user environment. The main objective is to improve the physical resource (CPU and memory) allocation by exploiting the collected statistics at run-time. When an estimation error is detected, they re-optimize not only the mapping between the remaining tasks and the CPUs, but also the allocation of memory. Incremental memory allocation (we call it IMA in short) heuristics were proposed to avoid unexpected extra I/Os caused by lack of memory. During re-optimization, the parallelism degree may also be modified to satisfy the memory requirements.

3.2.3 Comparison

In Table 2, we compare the above methods using the five criteria. C1: The rescheduling methods were mainly designed to deal with data arrival delay and rate changing problems, while the re-optimization methods were originally used to solve the other estimation error problems. However, they are not contradictory and can co-exist to handle both kinds of problems, as Tukwila [44] does. In CQO, statistics are missing during optimization. In IMA, the unavailability of resources is also taken into account. C2: They all optimize the currently running query. C3: They focus on different decision aspects, but none of them deal with all aspects. C4: Scrambling may degrade the performance dramatically if a bad join order is chosen during Phase 2. DQS reduces this risk by estimating the increased cost before changing the join order. ReOpt reacts to every detected estimation error. When there is only one wrongly estimated parameter, it works very well. However, when there are many uncertain parameters, each re-optimization may generate just another wrong plan. POP, LAWF and IMA may face the same problem. We consider the degradation risk level as high. CQO is different, because for the moment, it only modifies parallelism degree and partition key. The decision is based on run-time collected statistics, so the degradation risk is rather low. C5: Scrambling only modifies slightly the scheduler in order to detect data arrival delays and run the two phases iteratively. DQS not only rewrites the scheduler, but also modifies slightly the optimizer to generate annotated query plans and enhances the executor to be able to interact with the scheduler. Re-optimization methods only add statistics collectors and re-optimization triggers, except POP. It suspends the query during re-optimization, and

it provides different strategies for placing the CHECK operators, in order to support pipelined execution.

Table 2. Comparison of plan modification methods

Criterion / Method	C1: Estim. error sources	C2:Target query	C3:Target Opt. decisions	C4:Deg. risk	C5:Eng. cost
Scrambling	AD	C	OEO,JO	H	L
DQS	AD,AR	C	OEO,JO	L	M
ReOpt	DD,DC,DM	C	JM,JO,MA	H	L
POP	DD,DC,DM	C	AM,JM,JO	H	M
CQO	MS	C	PD,PK	L	L
LAWF	DD,DC,DM,MS	C	AM,JM,JO	H	L
IMA	DD,DC,DM,UR	C	CA,MA,PD	H	L

3.3 "Robust Plan" Selection

In section 3.2, were presented the methods which react to estimation errors by modifying the plan; in this section, are examined the methods which take into account the uncertainty during optimization process. Instead of an "optimal" plan, they choose a "robust" plan.

3.3.1 Robust cardinality estimation

The traditional sampling-based cardinality estimation methods compute a single-point value: if the population size is N, the sample size is s, and the observed cardinality for the sample is C', then the estimated cardinality C for the whole dataset should be C'*N/s. The estimation error could be small; however, the optimizer may still make a big mistake when choosing the plan. For example, in Figure 2 (a), we have two candidate plans P1 and P2 for query Q, where x-axis represents the value space of the uncertain cardinality and y-axis represents the cost of the plan. Suppose that the real cardinality is between C_{low} and C_{high}. If the estimated cardinality is C_{low}, the optimizer will choose P2. Otherwise, if the estimated cardinality is C_{high}, the optimizer will choose P1. By comparing these two situations, we find that the first one is more risky, because the worst case cost is high.

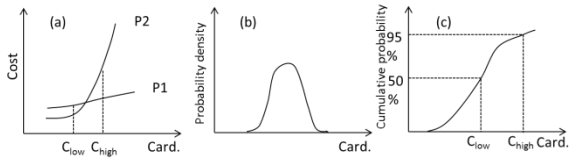


Figure 2. Robust cardinality estimation

Instead of estimating the cardinality by a single value C, the Robust Cardinality Estimation (RCE) method in [8] derives a probability density function of C from the sampling result, as shown in Figure 2(b). Then it transforms the probability density function into a cumulative probability function *cdf(c)*, as shown in Figure 2(c). The user is allowed to choose a confidence threshold *T* which represents the level of risk (a big *T* corresponds to a small risk). The estimated cardinality is computed by: $C = cdf^{-1}(T)$. For example, if *T*=95%, the model returns C_{high} as the estimated cardinality, so the more stable plan P1 will be chosen. If *T*=50%, the more risky plan P2 will be chosen. The authors claim that this solution is robust, because users are aware of the risk and take the responsibility for it.

3.3.2 Proactive re-optimization

Babu et al [7] proposed another way to take into account the estimation uncertainty during optimization process. In the Rio prototype, the authors estimate cardinalities using intervals, instead of single point values. If the optimizer is very certain of the estimate, then the interval should be narrow; otherwise, the interval should be wider. Using intervals allows the optimizer to generate robust plans that minimize the need for re-optimization. A robust plan is a plan whose cost is very close to optimal at all points within the interval. For example, in Figure 2(a), we assume that the estimated cardinality is C_{low} and the actual cardinality is C_{high}. With re-optimization methods such as POP, the optimizer will first choose P2 which is optimal at point C_{low}, and then choose P1 during re-optimization. For more complicated queries (e.g., with multiple joins and selection predicates), the situation could be even worse: re-optimization may happen repeatedly when multiple errors are detected.

We come back to the example in Figure 2(a). With Rio, the cardinality could be estimated as interval [C_{low}, C_{high}], so P1 will be chosen directly at the beginning, because it is robust within this interval. Thus, the re-optimization is avoided. The authors claim that their method is "proactive", because instead of reacting to the disaster caused by a wrong plan, they tried to prevent the optimizer from choosing that plan. Unfortunately, very often, a robust plan does not exist for the estimated interval. In this case, the authors propose to choose a set of plans which are "switchable". We will talk about this in Section 4.1. Actually, sometimes, we cannot even find a switchable plan. If this is the case, the authors propose to do like POP: choose an optimal plan using a single-point estimate and re-optimize the query if necessary. Note that, even if a robust or switchable plan is found, re-optimization may still be triggered, because the detected cardinalities may be outside of the estimated intervals.

Ergenç et al [26] extends the proactive re-optimization idea to deal with the query optimization problem in large-scale distributed environments. In such environments, the amount of data transferred between sites has a big impact on the overall performance. If the optimizer decides to place a relational operator at a wrong site due to cardinality estimation errors, huge amount of data may be transferred. To minimize the risk of wrong placement, the authors estimate the

cardinality as an interval instead of a single point value. If at any point in the interval, placing an operator on site S provides near-optimality (i.e., the performance degradation compared to the optimal placement is less than a threshold), then the site S is called a robust site. A Robust Placement (RP) for a query is to place recursively each operator in the plan tree on a robust site.

Least Expected Cost (LEC) optimization [21] also used intervals to estimate cardinalities. LEC treats statistics estimates as random variables to compute the expected cost of each plan and picks the one with lowest expected cost. It is an interesting approach for query optimization in general, but in our opinion cannot be considered as a robust optimization method, because its objective is to minimize the average running time of a compile-once-run-many query, but not to improve the worst case performance of a specific query execution.

3.3.3 Robust plan diagram reduction

A "plan diagram" is a color-coded pictorial enumeration of the plans chosen by the optimizer for a parameterized query template over the relational selectivity space [60]. The diagram is generated offline [57] by repeatedly invoking the query optimizer, each time with a different selectivity value. Then, for an instance of the query template, the optimizer first calls the selectivity estimator, and then picks the corresponding plan from the diagram. In the original paper, the authors gave examples with two-dimensional diagrams, each dimension representing the possible selectivity of one parametrized predicate in the query template. In this paper, for ease of comprehension, we illustrate the principle with a one-dimensional example. The sample query is "select * from R, S where $R.A_2 = S.A_2$ and $R.A_1 = \$x$", where $x is a variable. Figure 3(a) shows the diagram in the lower part, and the corresponding cost function curves above for more information. For example, if the estimated selectivity is between b and c, plan P3 will be chosen.

Using the plan diagram can avoid running the complete optimization algorithm for each instance of the parametrized query, thus the query optimization is more efficient. However, a plan diagram may contain a large number of plans in the selectivity space, making the diagram maintenance difficult. Therefore, Harish et al [38] proposed to reduce the dense diagrams to simpler ones, without degrading too much the quality of each individual plan. The principle is as follows: a plan Pa can be replaced by another plan Pb if and only if at each query point covered by Pa, the increased cost *(C(Pb) – C(Pa))* is less than a tolerance threshold defined by the user (such as 10%). For example, P1 and P3 can be replaced by P2, as shown in Figure 3(b).

When the cardinality estimation is precise, the reduced plan diagram does not degrade the performance too much compared to the original diagram. However, when there are estimation errors, use of the reduced diagram could be at high risk. For example, with the reduced diagram in Figure 3(b), if the estimated selectivity is between b and c, P2 will be chosen. However, if the actual selectivity is much higher than c, the cost of P2 becomes very high compared to P3.

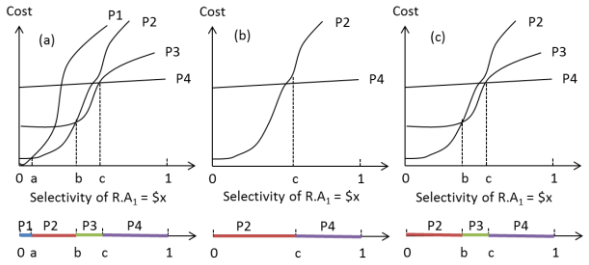


Figure 3. Example of robust plan diagram reduction

To reduce this risk, the same authors [39] proposed to make the plan replacement policy stricter: during the plan reduction, a plan Pa can be replaced by another plan Pb only if at each query point in the **whole selectivity space**, the increased cost *(C(Pb) – C(Pa))* is less than a tolerance threshold defined by the user. Considering this condition, P3 cannot be replaced anymore, so we get the reduction result in Figure 3(c). This reduction is called "Robust Diagram Reduction" (RDR), because the risk of significant performance degradation is limited in case of estimation errors.

3.3.4 Comparison

The above methods are compared in Table 3. C1: They all assume that the running environment is stable. RCE derives the probability density function by using fresh random samples which are pre-computed manually or updated periodically whenever a sufficient number of database modifications have occurred. Thus, it does not deal with DM and MS. Rio and RP still work when catalog statistics are missing or outdated. However, their effectiveness may be affected. In fact, they compute the estimation intervals based on catalog statistics. If statistics are missing or outdated, it may happen that the intervals become too large for the optimizer to find a robust plan, or that the intervals are erroneous and re-optimization should be triggered. RDR has no constraints on estimation error sources. C2: They all deal with the currently running query, except RDR, which works for predefined parametrized queries. C3: RCE, Rio, RDR could avoid wrong decisions on base relation access methods, join methods and join ordering; RP extends Rio to improve the execution site selection. C4: RCE allows the user to choose the risk level. For Rio, if the actual cardinalities fall into the estimated intervals, and if a

robust plan exists, then the performance degradation risk is limited to a predefined threshold. However, these two conditions are difficult to satisfy. RP has the same risk level as Rio. The degradation risk of RDR is always limited, thanks to the strict replacement policy. C5: RCE only modifies the cardinality estimation module of the optimizer. Rio requires more modifications to the DBMS engine. RP works for large-scale distributed environments and is based on a mobile execution model [4]. Similar to Rio, it also requires significant modifications to the optimizer in order to support the interval-based estimation. RDR develops a stand-alone tool to prepare a set of robust plans for a predefined query template.

Table 3. Comparison of robust plan selection methods

Criterion / Method	C1: Estim. error sources	C2: Target query	C3: Target opt. decisions	C4:Deg. risk	C5:Eng. cost
RCE	DD,DC	C	AM,JM,JO	UC	L
Rio	DD,DC,DM,MS	C	AM,JM,JO	LC	M
RP	DD,DC,DM,MS	C	AM,JM,JO,ES	LC	M
RDR	DD,DC,DM,MS	P	AM,JM,JO	L	L

4. MULTI-PLAN BASED APPROACH

4.1 Deferred Plan Choosing

Parametric query optimization process [10] determines for each point in the parameter space, an optimal plan. It defers choosing the plan until the start of execution. However, its objective is to avoid compiling the query for each run, but not to achieve robustness. In this section, we will study other methods, which make the choice in the middle of execution.

4.1.1 Access method competition

Whether to use indexes and which ones to use for a single-relation access depends strongly on the selectivity of the predicate. To avoid wrong optimization decisions due to the selectivity estimation uncertainty, Antoshenkov [3] proposed access method competition (AMC), i.e., to run simultaneously different base relation access processes for a small amount of time. The author argues that there is a high probability that one of them finishes during this time, and others can be canceled. Otherwise, if none of them finishes quickly, the execution engine should guess and continue only one that has the least estimated cost.

4.1.2 Plan switching

In Rio [7], if the optimizer fails to find a robust plan within the estimated interval, it tries to find a "switchable plan" (SP), which is a set S of plans such that: (1) at any point in the intervals, there is a plan p in S whose cost is close to optimal; (2) according to the detected statistics, the system can switch from one plan to another in S without losing any significant fraction of work done so far. Figure 4 gives an example of a switchable plan. Assuming that the result size of R⋈S is estimated to be small, then the first plan is executed; during execution, if the tuples produced by R ⋈ S cannot fit in memory, the second plan will be switched on (i.e., changing the join algorithm from NLJ to HJ); later on, if the result size of R⋈S is detected to be much bigger than the relation T, the third plan will be switched on. The switching process is smooth thanks to a "switch" operator integrated in the plan tree.

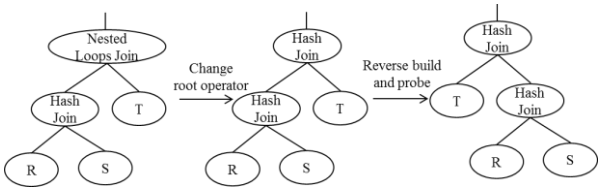


Figure 4. Example of a switchable plan

The "switch" operator can be seen as a variant of the "choose-plan" operator (CP) proposed earlier by Graefe et al. in [29]. "Choose-plan" operators are run-time primitives that permit optimization decisions to be prepared at compile-time and evaluated at run-time. It was initially designed to deal with the situation where parameters in the query template are unknown; however, other estimation errors like non-uniform distribution could also be addressed by adding a "choose-plan" operator at an appropriate position in the decision tree.

A different way of switching plans based on "Plan Bouquets" (PB) has been proposed recently [25]. First, through repeated invocations of the optimizer, a "parametric optimal set of plans" (POSP) that covers the entire selectivity space of the predicates is identified. Second, a "POSP infimum curve" (PIC) which is the trajectory of the minimum cost from the POSP plans is constructed. Then, the PIC is discretized by some predefined isocost (IC) steps, IC1, IC2, … , which are progressive cost thresholds, for example, each IC value doubles the preceding one. The intersection of each IC with the PIC corresponds to a selectivity value and the best POSP plan for this selectivity. The set of plans associated with these ICs is called a "Plan Bouquet". At run-time, the plan associated with the cheapest IC step is executed first. If the partial execution overheads exceed the IC value, it means that the actual selectivity is beyond the range where the current plan is optimal, so a switch to the plan associated with the next IC value is triggered. Otherwise, if the current plan completes execution before reaching the IC value, it means that the actual selectivity is inside the range covered by this plan. Since the cost of executing a sequence of plans before discovering the actual selectivity and then switching to the optimal plan is bounded by the IC values, Plan Bouquet method guarantees worst-case performance.

4.1.3 Comparison

In Table 4, we compare the above methods. C1: They deal with all kinds of estimation errors in a stable environment. C2: AMC and SP work for the currently running query, while CP and PB were initially designed for predefined parametrized queries. C3: AMC focuses on base relation access method selection; SP deals with join methods and join order; CP and PB cover all those three decisions. C4: AMC has low degradation risk, at the condition that one of the concurrent processes finishes quickly; SP also has low degradation risk, at the condition that the actual cardinalities are inside the estimated intervals. CP and PB have low degradation risk. C5: All methods require major modifications to the optimizer and the executor.

Table 4. Comp. of deferred plan choosing methods

Criterion / Method	C1: Estim. error sources	C2: Target	C3:Target opt. decisions	C4:Deg. risk	C5:Eng. Cost
AMC	DD,DC,DM,MS	C	AM	LC	M
SP	DD,DC,DM,MS	C	JM,JO	LC	M
CP	DD,DC,DM,MS	P	AM,JM,JO	L	M
PB	DD,DC,DM,MS	P	AM,JM,JO	L	M

4.2 Tuple Routing through Eddies

With the methods described in Section 4.1, although the optimizer proposes multiple execution plans, only one of them "survives". Thus, all tuples flow through the same plan tree (route). Differently, Avnur et al [5] allow different tuples to flow through different routes using a special operator "eddy". The eddy mechanism was extended for different environments [59, 64, 70].

4.2.1 The eddy

A tree-like plan fixes the execution order of the operators in advance, i.e., tuples always flow from leaves to the root. This order can be changed at run-time by using rescheduling or re-optimization, but it is too costly to change it frequently. Avnur et al. [5] proposed a more flexible mechanism which can continuously reorder the operators. They use a star-like query plan, where the relational operators surround a coordinating operator called an eddy. Tuples from base relations and intermediate results are sent to the eddy which routes each tuple to an operator according to the routing policy. The eddy sends a tuple to the output only if it has been handled by all the operators.

We illustrate the effectiveness of eddies using the query "select * from R, S, T where R.B=S.B and S.C=T.C". Suppose that tuples arrive from the three relations with different delay and rate. In the plan, if there are two join operators Op1 ($R \bowtie S$) and Op2 ($S \bowtie T$), tuples from R can be only routed to Op1, and tuples from T can be only routed to Op2, while tuples from S can be routed either to Op1 or to Op2. The eddy makes the routing decision for tuples from S dynamically according to a predefined policy, which tries to minimize the execution cost. A specific join algorithm called Symmetric Hash Join (SHJ) [68, 41] is recommended: two hash tables are maintained, one for each input relation; the arriving tuple is built immediately into the corresponding hash table and probed against the existing tuples in the other hash table, so the intermediate result can be produced immediately and returned to the eddy. This is actually the "secret" of eddies to enable the reorder-ability: (1) the operator state is continuously maintained, regardless of the execution order; and (2) the "faster" relation is never blocked by the "slower" relation.

We can find that, an eddy is equivalent to a set of tree-like query plans, each one handling a subset of tuples. The tuple routing policy is used to make the mappings. Advanced routing policies [9] were proposed later on. Note that, since the eddy implementations rely on the symmetric hash join, they are more adequate to streaming scenarios where existing relations fit in memory.

4.2.2 Extensions of the eddy

The adaptability of the eddy is limited to operator re-ordering, whereas access methods and join algorithms are pre-chosen and fixed during the execution. A more flexible version [59] also allows continuously changing the choice of access methods and join algorithms, etc. To do this, Raman et al [59] made the following main modifications to the eddy architecture: (1) each join operator is replaced by two State Modules (SteMs) which encapsulate data structures (such as hash tables or indexes) used in join algorithms; (2) one or several Access Modules (AMs) are added to each base relation, each AM encapsulating one access method to the data source; and (3) new routing policies are used by the eddy module. At the beginning, different AMs are run concurrently (thus redundantly). In fact, they are in competition: when the eddy finds that one is much more efficient than the others, it will stop the slower ones. Tian and DeWitt [64] extended the eddy and SteMs architecture to a distributed version. Instead of using a centralized eddy module which could become a bottleneck, the authors proposed to integrate the routing function into each operator. Zhou et al [70] designed another distributed query processing architecture called SwAP, building on eddies and SteMs. The authors proposed to use one eddy module for each execution site, which routes tuples between the local operators and remote eddies.

4.2.3 Comparison

We compare the tuple routing methods in Table5. C1: They deal with all kinds of estimation errors. C2: They optimize the currently running query. C3: Eddy only

optimizes the join order and operator execution order; the extensions of eddy also optimize base relation access methods and join methods. C4: Although there is no theoretical guarantee, Deshpande [23] has shown experimentally that the performance degradation of eddies is low. Other methods have the same risk level, under the condition that one of the competing access methods wins quickly. C5: In these methods, most of the optimization decisions are made by the eddy module, so the classical optimizer is reduced to a pre-optimizer, and the execution engine becomes more complicated. Thus, the engineering cost is high.

Table 5. Comparison of tuple routing methods

Criterion / Method	C1: Estim. error sources	C2: Target query types	C3: Target opt. decisions	C4:Deg. risk	C5:Eng. Cost
Eddy	All	C	JO,OEO	L	H
SteM	All	C	AM,JM,JO,OEO	LC	H
Tian	All	C	AM,JM,JO,OEO	LC	H
SwAP	All	C	AM,JM,JO,OEO	LC	H

4.3 Optimizer Controlled Data Partitioning

In TRE based methods, the mapping between tuples and multiple plans is decided by the eddy module, according to local indicators, such as input rate and output rate of an operator. In OCDP based methods, the mapping is decided by the optimizer, according to global statistics. TRE tends to avoid worst-case performance, while OCDP also aims at exploring best-case opportunities. We will present some representative methods in this section.

4.3.1 Run-time partitioning

Ives et al [45] proposed an adaptive data partitioning (ADP) method. During query execution, the data are dynamically partitioned into sub-datasets, each following a specific plan. Three partitioning strategies are illustrated: (1) sequential partitioning. The query execution is divided into multiple phases. All tuples arriving during Phase Ph_N follow a plan Pl_N. Pl_N is chosen using the statistics collected during the previous N-1 phases. To guarantee the correctness of the result, a "stitch-up" phase is added at the end. For example, two relations S and T are joined through two phases. During Ph_0, S_0 and T_0 are joined; during Ph_1, S_1 and T_1 are joined. According to the following equation: $S \bowtie T = (S_0 \bowtie T_0) \cup (S_1 \bowtie T_1) \cup (S_0 \bowtie T_1) \cup (S_1 \bowtie T_0)$, the "stitch-up" phase has to compute $(S_0 \bowtie T_1)$ and $(S_1 \bowtie T_0)$. (2) Dynamic splitting. Multiple plans are run concurrently, and the arriving tuple is sent to one plan by a "split" operator according to some criteria. For example, to join two relations which are quasi-sorted, tuples respecting the expected order will be sent to the plan with merge-join, and others will be sent to the plan with hash join. (3) Partitioning used for plan competition. Multiple plans are run concurrently, each processing a small subset of data. If one is much faster than the others, it will process all the remaining data. Note that, for the last two strategies, a "stitch-up" phase is also needed.

4.3.2 Compile-time partitioning

Different from ADP, Selectivity-Based Partitioning (SBP) [58] and Query Mesh (QM) model [54, 55] decide the data partitions and corresponding plans at query compile time. The author of SBP noticed that there often exist join correlations among relation fragments, for example, given two relations R and S, where $S = S_1 \cup S_2$, it may happen that the join $R \bowtie S_1$ is selective while $R \bowtie S_2$ is much less selective. Based on this observation, for a chain query with equality predicates, the author proposed to horizontally partition one relation in the chain, and rewrite the original query as the union of a set of sub-queries. For different sub-queries, the optimizer can choose several join orders, such that the overall performance is better than using a single plan without partitioning. The search space of this optimization problem is large: the optimizer has to decide which relation to partition, choose the number of partitions and compute the optimal join order corresponding to each partition. The author proposed a heuristic algorithm for computing an effective solution without exploring the complete search space. With the QM model, for a given query, a decision tree-based classifier is learned from a training dataset. Each decision node is a predicate (such as A>x) which distributes the arriving tuples into different classes. For each tuple class, a best plan is chosen. The choice of execution plans and the classifier are mutually dependent, so they should be considered as a whole, meaning that the execution cost of each plan for each possible data subset should be estimated and compared. The search space is too big to use an enumerative search strategy, so the authors chose randomized search strategies. Similar to QM, correlation-aware multi-route stream query optimizer (CMR) [16] also partitions the data and computes an optimal plan for each partition. The difference is that it explores explicitly data correlations, which not only makes the partitioning more effective but also reduces the optimization complexity. Horizontal Partitioning with Eddies HPE [65] is another work using different plans for different data partitions. The originality is: first, the authors introduced the notion of conditional join plans (CJP), a new representation of search space which captures both the partitioning and the join orders for each partition combination; second, they use the eddy mechanism as the execution model, in order to share intermediate results between different plans.

4.3.3 Comparison

We compare the above methods in Table 6. C1: They are all resistant to (or even take advantage of) non-uniform data distribution and data correlations. HPE uses eddies, so it is also resistant to data arrival delay and rate changing, etc. C2: They all optimize the current query. C3: SBP focuses only on join order. Others focus also on access methods and join methods. Again, with eddies, HPE can also optimize the operator execution order. C4: The degradation risk is low, because characteristics of each sub-dataset are well-known by the optimizer. C5: The engineering cost is high, because both the optimizer and the plan executor need to be rewritten.

Table 6. Comparison of data partitioning methods

Criterion / Method	C1: Estim. error sources	C2:Target query types	C3: Target opt. decisions	C4:Deg. risk	C5:Eng. cost
SBP	DD,DC	C	JO	L	H
ADP	DD,DC	C	AM,JM,JO	L	H
QM	DD,DC	C	AM,JM,JO	L	H
CMR	DD,DC	C	AM,JM,JO	L	H
HPE	All	C	AM,JM,JO,OEO	L	H

5. GLOBAL COMPARISON

In this section, we make a global comparison of the two approaches and their adopted strategies. With single-plan based approach, methods are easier to implement, but none of them can handle all types of estimation error sources; different methods could be combined to enlarge the application scope, but when there are too many uncertain factors, the degradation risk becomes high. With multi-plan based approach, the degradation risk is limited, but the engineering cost is higher. Eddy-based methods can handle all kinds of estimation error sources, however, most of them require that the hash tables fit in memory. In addition, how eddies can be used in a highly parallel environment has not been well studied.

In Table 7, we list briefly the advantages and limitations of the strategies used by each approach.

6. CONCLUSION

Robust query optimization methods take into account the uncertainty of estimated parameter values, in order to avoid or recover from bad decisions caused by estimation errors. In this paper, the representative methods were classified into two main approaches: single-plan based approach and multi-plan based approach. For each approach, we highlight the principle strategies. We analyzed and compared the methods using five well-selected criteria: estimation error sources, target query types, target optimization decisions, performance degradation risk and engineering cost. Finally, a global comparison of the approaches and the strategies is given.

Table 7. Global comparison

Approach	Strategy	Advantage	Limitation
Single-Plan Based	CI	Good for repeatedly-running queries	For current query, only JM, JO are optimized
	PM	Could be extended to improve all kinds of opt. decisions	May have high degradation risk
	RPS	Degradation risk is low if a robust plan exists	Difficult to handle too many uncertain factors
Multi-Plan Based	DPC	Easier to implement than TR and DP	AMC may consume too many resources
	TRE	Deal with all kinds of estimation error sources	Memory consuming; Parallelization problem not addressed
	OCDP	Take advantage of inherent data characteristics	Optimization time may be long

The main conclusions to be drawn are: (1) different strategies of the single-plan based approach can be combined to enlarge the application scope, as the AutoAdmin project [14] does, (2) single-plan based approach is easier to be integrated into the main commercial DBMSs, but it only works well when there are few uncertain parameters, and (3) hence when there are too many uncertain parameters, the multi-plan based approach is a safer choice.

7. REFERENCES

[1] Aboulnaga, A. and Chaudhuri, S. 1999. Self-tuning Histograms: Building Histograms without Looking at Data. In *SIGMOD*. New York, USA, 181-192.

[2] Amsaleg, L., al.1996. Scrambling Query Plans to Cope with Unexpected Delays. In *PDIS*. Miami, USA, 208-219.

[3] Antonshenkov, G. 1993. Dynamic Query Optimization in Rdb/VMS. In *ICDE*. Vienna, Austria, 538-547.

[4] Arcangeli, J.P., et al. 2004. Mobile Agent Based Self-Adaptive Join for Wide-Area Distributed Query Processing. *Journal of Database Management*, 15(4): 25-44.

[5] Avnur, R. and Hellerstein, J.M. 2000. Eddies: Continuously Adaptive Query Processing. *SIGMOD*. Dallas, USA, 261-272.

[6] Babu, S. and Bizarro, P. 2005. Adaptive Query Processing in the Looking Glass. In *CIDR*. Asilomar, USA, 238-249.

[7] Babu, S., et al. 2005. Proactive Re-Optimization. In *SIGMOD*. Baltimore, USA, 107-118.

[8] Babcock, B. and Chaudhuri, S. 2005. Towards a Robust Query Optimizer: A Principled and Practical Approach. In *SIGMOD*. Baltimore, USA, 119-130.

[9] Bizarro, P., et al. 2005. Content-Based Routing: Different Plans for Different Data. In *VLDB*. Trondheim, Norway, 757-768.

[10] Bizarro, P., et al. 2009. Progressive Parametric Query Optimization. *KDE*, 21(4): 582 – 594.

[11] Bonneau, S. and Hameurlain, A. 1999. Hybrid Simultaneous Scheduling and Mapping in SQL Multi-Query Parallelization. In *DEXA*. Florence, Italy, 88-98.

[12] Bouganim, L., et al. 2000. Dynamic Query Scheduling in Data Integration Systems. In *ICDE*. San Diego, USA, 425-434.

[13] Bruno, N., et al. 2001. STHoles: a multidimensional workload-aware histogram. In *SIGMOD*. Santa Barbara, USA, 211-222.

[14] Bruno, N., et al. 2011. AutoAdmin Project at Microsoft Research: Lessons Learned. *IEEE Data Eng. Bull*, 34(4): 12-19.

[15] Bruno, N., et al. 2013. Continuous Cloud-Scale Query Optimization and Processing. *PVLDB*, 6(11): 961-972.

[16] Cao, L. and Rundensteiner, E.A. 2013. High Performance Stream Query Processing With Correlation-Aware Partitioning. PVLDB, 7(4): 265-276.

[17] Chaudhuri, S., et al. 2008. A Pay-As-You-Go Framework for Query Execution Feedback. *PVLDB*, 1(1): 1141-1152.

[18] Chaudhuri, S. 2009. Query Optimizers: Time to Rethink the Contract? In *SIGMOD*. Providence, USA, 961-968.

[19] Chaudhuri, S., et al. 2009. Exact Cardinality Query Optimization for Optimizer Testing. *PVLDB*, 2(1): 994-1005.

[20] Chen, C.M. and Roussopoulos, N. 1994. Adaptive Selectivity Estimation Using Query Feedback. In *SIGMOD*. Minneapolis, USA, 161-172.

[21] Chu, F., Halpern, J., Gehrke, J. 2002. Least expected cost query optimization: what can we expect? In *PODS*. Madison, USA, 293-302.

[22] Cole, R.L. and Graefe, G. 1994. Optimization of Dynamic Query Evaluation Plans. In *SIGMOD*. Minneapolis, 150-160.

[23] Deshpande, A. 2004. An Initial Study of Overheads of Eddies. *SIGMOD Record*, 33(1): 44-49.

[24] Deshpande, A., et al. 2007. Adaptive Query Processing. *Foundations and Trends in Databases*, 1(1): 1-140.

[25] Dutt, A. and Haritsa, J. 2014. Plan bouquets: query processing without selectivity estimation. In *SIGMOD*. Snowbird, USA, 1039–1050.

[26] Ergenç, B., et al. 2007. Robust Placement of Mobile Relational Operators for Large Scale Distributed Query Optimization. In *PDCAT*. Adelaide, Australia, 227-235.

[27] Evrendilek, C., et al. 1997. Multidatabase Query Optimization. *Distributed and Parallel Databases*, 5(1):77-114.

[28] Ghazal, A., et al. 2012. Adaptive Optimizations of Recursive Queries in Teradata. In *SIGMOD*. Scottsdale, USA, 851-860.

[29] Graefe, G. and Ward, K. 1989. Dynamic query evaluation plans. In *SIGMOD*. Portland, USA, 358-366.

[30] Graefe, G., et al. 2009. Visualizing the Robustness of Query Execution. In *CIDR*. Asilomar, USA.

[31] Graefe, G., et al. 2010. Robust Query Processing. *Dagstuhl Workshop Summary 10381*, Wadern, Germany.

[32] Graefe, G. 2011. Robust Query Processing (Research Panel). In *ICDE*. Hannover, Germany, 1361.

[33] Graefe, G., et al. 2012. Robust Query Processing. *Dagstuhl Workshop Summary 12321*, Wadern, Germany.

[34] Gounaris, A., et al. 2002. Adaptive Query Processing: A Survey. In *BNCOD*. Sheffield, UK, 11-25.

[35] Gounaris, A., et al. 2013. Adaptive Query Processing in Distributed Settings. *Advanced Query Processing*, Vol. 1: 211-236.

[36] Hameurlain, A. and Morvan, F. 2002. CPU and Incremental Memory Allocation in Dynamic Parallelization of SQL Queries. *Journal of Parallel Computing*, 28(4): 525-556.

[37] Han, W., et al. 2007. Progressive Optimization in a Shared-Nothing Parallel Database. In *SIGMOD*. Beijing, 809-820.

[38] Harish, D., et al.. 2007. On the Production of Anorexic Plan Diagrams. In VLDB. Vienna, Austria, 1081-1092.

[39] Harish, D., et al. 2008. Identifying Robust Plans through Plan Diagram Reduction. *PVLDB*, 1(1): 1124-1140.

[40] Herodotou, H. and Babu, S. Xplus. 2010. A SQL-Tuning-Aware Query Optimizer. *PVLDB*, 3(1): 1149-1160.

[41] Hong, W. and Stonebraker, M.1993. Optimization of Parallel Query Execution Plans in XPRS. *Distributed and Parallel Databases*, 1(1): 9–32.

[42] Ioannidis, Y. and Christodoulakis, S. 1991. On the Propagation of Errors in the Size of Join Results. In *SIGMOD*. Denver, USA, 168-177.

[43] Ioannidis, Y. 2003. The History of Histograms (abridged). In *VLDB*. Berlin, Germany, 19-30.

[44] Ives, Z. G., et al. 1999. An Adaptive Query Execution System for Data Integration. In *SIGMOD*. Philadelphia, USA, 299-310.

[45] Ives, Z.G., et al. 2004. Adapting to Source Properties in Processing Data Integration Queries. In *SIGMOD*. Paris, France, 395-406.

[46] Kabra, N. and DeWitt, D. 1998. Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans. In *SIGMOD*. Seattle, USA, 106-117.

[47] Larson, P., et al. 2007. Cardinality Estimation Using Sample Views with Quality Assurance. In *SIGMOD*. Beijing, 175-186.

[48] Lipton, R.J., et al. 1990. Practical Selectivity Estimation through Adaptive Sampling. In *SIGMOD*. Atlantic City, 1-11.

[49] Mannino, M., et al. 1988. Statistical profile estimation in database systems. *ACM Computing Surveys*, 20(3): 191-221.

[50] Markl, V., et al. 2004. Robust Query Processing through Progressive Optimization. *SIGMOD*. Paris, France, 659-670.

[51] Markl, V., et al. 2007. Consistent Selectivity Estimation via Maximum Entropy. *VLDB Journal*, 16(1): 55-76.

[52] Morvan, F. and Hameurlain, A. 2009. Dynamic Query Optimization: Towards Decentralized Methods. *International Journal of Intelligent Information and Database Systems*, 4(3): 461-482.

[53] Neumann, T. and Calindo-Legaria, C. 2013.Taking the Edge off Cardinality Estimation Errors using Incremental Execution. In *BTW*. Magdeburg, Germany, 73-92.

[54] Nehme, R.V., et al. 2009. Query Mesh: Multi-Route Query Processing Technology (Demo). *PVLDB*, 2(2): 1530-1533.

[55] Nehme, R.V., et al. 2013. Multi-Route Query Processing and Optimization. Journal of Computer and System Sciences, 79(3): 312-329.

[56] Olken, F. and Rotem, D. 1986. Simple Random Sampling from Relational Databases. In *VLDB*. Kyoto, Japan, 160-169.

[57] Picasso Database Query Optimizer Visualizer. http://dsl.serc.iisc.ernet.in/projects/PICASSO/

[58] Polyzotis, N. 2005. Selectivity-based partitioning: a divide-and-union paradigm for effective query optimization. In *CIKM*. Bremen, Germany, 720-727.

[59] Raman, V., et al. 2003. Using State Modules for Adaptive Query Processing. In *ICDE*. Bangalore, India, 353-364.

[60] Reddy, N. and Harista, J. 2005. Analyzing Plan Diagrams of Database Query Optimizers. In *VLDB*. Trondheim, Norway, 1228-1239.

[61] Selinger, P.G., et al. 1979. Access Path Selection in a Relational DBMS. In *SIGMOD*. Boston, USA, 23-34.

[62] Srivastava, Uet al. 2006. ISOMER: Consistent Histogram Construction Using Query Feedback. In *ICDE*. Atlanta, 39.

[63] Stillger, M., et al.2001. LEO-DB2's Learning Optimizer. *VLDB*. Roma, Italy, 19-28.

[64] Tian, F. and DeWitt, D.J. 2003. Tuple Routing Strategies for Distributed Eddies. In VLDB. Berlin, Germany, 333-344.

[65] Tzoumas, K., et al. 2010. Sharing-Aware Horizontal Partitioning for Exploiting Correlations during Query Processing. *PVLDB*, 3(1): 542-553.

[66] Tzoumas, K., et al. 2011. Lightweight Graphical Models for Selectivity Estimation without Independence Assumptions. *PVLDB*, 4(11): 852-863.

[67] Tzoumas, K., et al. 2013. Efficient Adapting Graphical Models for Selectivity Estimation. *VLDB Journal*, 22(1): 3-27.

[68] Wilschut, A. N. and Apers, P. M. G. 1991. Dataflow Query Execution in a Parallel Main-Memory Environment. In PDIS, Miami Beach, USA, 68–77.

[69] Wiener, J.L., et al. 2009. Benchmarking Query Execution Robustness. In TPC Technology Conference on Performance Evaluation & Benchmarking. Lyon, France, 153-166.

[70] Zhou, Y., et al. 2005. An Adaptable Distributed Query Processing Architecture. *Knowledge and Data Engineering*. 53(3): 283-309.

Community Detection in Multi-Layer Graphs: A Survey

Jungeun Kim, Jae-Gil Lee*
Department of Knowledge Service Engineering, KAIST
Daejeon, Republic of Korea
{je_kim, jaegil}@kaist.ac.kr

ABSTRACT

Community detection, also known as graph clustering, has been extensively studied in the literature. The goal of community detection is to partition vertices in a complex graph into densely-connected components so-called *communities*. In recent applications, however, an entity is associated with multiple aspects of relationships, which brings new challenges in community detection. The multiple aspects of interactions can be modeled as a *multi-layer graph* comprised of multiple interdependent graphs, where each graph represents an aspect of the interactions. Great efforts have therefore been made to tackle the problem of community detection in multi-layer graphs. In this survey, we provide readers with a comprehensive understanding of community detection in multi-layer graphs and compare the state-of-the-art algorithms with respect to their underlying properties.

1. INTRODUCTION

Graph mining in complex networks has attracted significant attention during the past several years. One of the important tasks in graph mining is community detection, in which the objective is to partition a graph into several densely-connected components. Such components correspond to sets of similar vertices, and can thus be regarded as a *community* [17]. Since this problem arises in a broad range of applications, a large number of approaches have been proposed in the literature [10, 13, 15].

In contrast to the traditional problem, recent applications, such as mobile and social network analyses, give rise to intriguing new challenges [6]. In this context, assumably, data encapsulates multiple aspects of human interactions, *e.g.*, those among coworkers and those among friends. The multiple aspects of relationships can be represented by a multi-layer graph comprised of multiple interdependent graphs, where each graph represents an aspect of the relationships. Therefore, great efforts have been made to solve the challenge of community detection in multi-layer graphs.

The goal of this survey is to provide a timely remark on the status of improving community detection in multi-layer graphs. We offer a brief overview of primary algorithms and classify them with respect to their underlying strategies.

The rest of this paper is organized as follows. Section 2 discusses the background information regarding multi-layer graphs. Section 3 presents multi-layer graph datasets used in recent studies. Section 4 introduces community detection approaches in two-layer graphs. Section 5 introduces community detection approaches in multi-layer graphs. Section 6 presents comparisons of community detection approaches in multi-layer graphs. Section 7 suggests promising future research directions in multi-layer graphs. Finally, Section 8 concludes this survey.

2. BACKGROUND

In this section, we discuss some background information about multi-layer graphs. We present the formal definitions of multi-layer graphs [14] in Section 2.1. Then, we briefly summarize the community detection approaches for single graphs and the challenges for developing those for multi-layer graphs in Section 2.2. To enhance the readability of this survey, frequently used symbols are summarized in Table 1.

Table 1: The summary of symbols.

Symbol	Description
G	a graph
V	a set of vertices
S	a set of attributes
L	a set of layers
n	the number of vertices
m	the number of edges
k	the number of clusters
t	the number of attributes
l	the number of layers

*Jae-Gil Lee is the corresponding author.

2.1 Multiple Network Models

2.1.1 Multi-Layer Graphs

The definition of a multi-layer graph depends on that of a single-layer graph.

Definition 1. [14] A *single-layer graph* is a weighted graph (V, w) where V is a set of vertices and w is a set of edge weights: $(V \times V) \rightarrow [0,1]$.

Figure 1 shows an example of a single undirected graph (without specifying the edge weights). Assume that it is a subgraph of Facebook's network. Each vertex represents the user, and each edge denotes the relationship between users. The weight of the edge is the strength of the relationship.

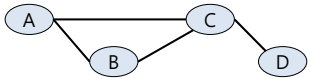


Figure 1: A single-layer graph.

When we start characterizing multi-layer graphs, understanding which vertices in one graph correspond to vertices in the other is important because the multi-layer graph is comprised of multiple *interdependent* graphs. A *node mapping* can formalize this task.

Definition 2. [14] A *node mapping* from a graph layer $L_1 = (V_1, w_1)$ to another graph layer $L_2 = (V_2, w_2)$ is a function $f : V_1 \times V_2 \rightarrow [0, 1]$. For each $u \in V_1$, the set $C(u) = \{v \in V_2 | f(u, v) > 0\}$ is the set of V_2 vertices corresponding to u.

Figure 2 illustrates an example of a multi-layer graph. Assume that layer 1 is the Facebook network and layer 2 is the Twitter network. If the users in the Facebook network also have an account on Twitter, then the Twitter network can be used to represent these users and their relationships. Note that every user can be identified by one account on each layer. This graph is generally called a *pillar multi-layer graph* since every user can be seen as a pillar traversing every layer denoting the level of physical reality [14]. A pillar multi-layer graph is formally defined by node mapping, $|C(u)| \in \{0, 1\}$.

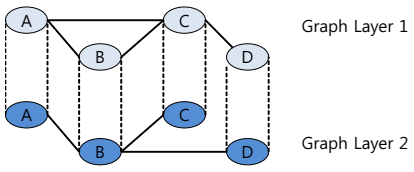


Figure 2: A pillar multi-layer graph.

A generic multi-layer graph is formally defined based both on a set of single layers and a matrix of node mappings.

Definition 3. [14] A *multi-layer graph* is a tuple $MLN = (L_1, ..., L_l, IM)$ where $L_i = (V_i, w_i), i \in 1, ..., l$ are graph layers and IM (Identity Mapping) is an $l \times l$ matrix of node mappings, with $IM_{i,j} : V_i \times V_j \rightarrow [0, 1]$.

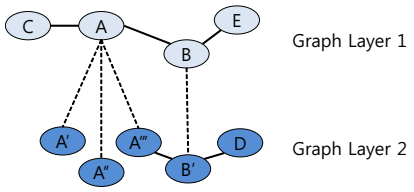


Figure 3: A general multi-layer graph.

Figure 3 shows an example of a multi-layer graph that is more complex than a pillar multi-layer graph. For example, in Figure 3, on layer 1, A is an account of a user in FriendFeed, and A′, A″, and A‴ on layer 2 are the social media accounts that the user has registered. We call this network a *general multi-layer graph*. Note that a vertex in one graph layer corresponds to multiple vertices in another. This case is typically shown in *social media aggregators*, such as FriendFeed, which support various social network services with a single access point as long as a user has registered for those services. Thus, vertices do not necessarily denote users, but more generally, accounts.

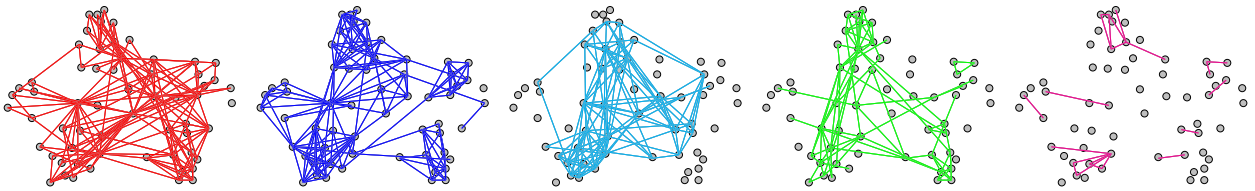
2.1.2 Heterogeneous Information Networks

The definition of a heterogeneous information network depends on that of an information network.

Definition 4. [23, 24] An *information network* is defined as a directed graph $G = (V, E)$ with an object type mapping function $\phi : V \rightarrow \mathcal{A}$ and a link type mapping function $\psi : E \rightarrow \mathcal{R}$, where each object $v \in V$ belongs to one particular object type $\phi(v) \in \mathcal{A}$, and each link $e \in E$ belongs to a particular relation $\psi(e) \in \mathcal{R}$.

If the number of object types $|\mathcal{A}| > 1$ or the number of link types $|\mathcal{R}| > 1$, the network is called a *heterogeneous information network* [23, 24]. A bibliographic information network is a typical example, containing objects from four types of entities: papers, venues, authors, and terms. Each paper has distinct types of links to a set of authors, a venue, a set of words, a set of citing papers, and a set of cited papers, respectively.

A heterogeneous information network can be translated to a *general* multi-layer graph in Definition 3, and vice versa. More specifically, an object type corresponds to a layer L_i, the links within an object type correspond to w_i, and the links between



(a) Work. (b) Lunch. (c) Facebook. (d) Friend. (e) Coauthor.

Figure 4: An example of a multi-layer graph called the AUCS dataset.

different object types correspond to $IM_{i,j}$. That is, these two definitions are syntactically equivalent.

Despite this equivalence, the two definitions are actually being used for slightly different meanings. General multi-layer graphs emphasize *multiple types of relationships* between similar types of entities. For example, in Figure 3, all the entities are social media accounts. On the other hand, heterogeneous information networks emphasize *heterogeneous types of entities* connected by different relationships. Overall, we rely on the typical meaning of multi-layer graphs in this paper.

2.2 Current Status and Challenges

2.2.1 Community Detection in Single Graphs

Many community detection approaches have been proposed for single-layer graphs. Fortunato [7] and Schaeffer [19] conducted really extensive survey on this topic. Representative algorithms include graph partitioning algorithms, modularity-based algorithms, spectral algorithms, and structure definition algorithms [7, 19]. The objective of *graph partitioning algorithms* is to divide the vertices such that cut size is minimal. *Cut size* is determined by the number of edges lying between partitions. The goal of *modularity-based algorithms* is to partition the vertices such that modularity is maximal. *Modularity* is defined by the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random. *Spectral algorithms* partition the graph into communities using the eigenvectors of graph matrices. A graph Laplacian matrix is typically used for the graph matrix. *Structure definition algorithms* discover communities such that a very strict structural property is satisfied. In other words, they find communities satisfying the meta definitions of a community such as k-clique, r-quasi clique, and s-plex.

2.2.2 Challenges for Multi-Layer Graphs

In contrast to the community detection problem in single graphs, new challenges arise for community detection in multi-layer graphs. Intuitively, each single layer has a piece of meaningful information from its own perspective; however, one can expect improved community detection results through the proper and efficient merging of information in each layer. Thus, an important open question is how to exploit and fuse the multiple aspects of information to generate improved understanding of vertices and their relationships. In addition, since we are confronted with managing multiple layers (often called *networks of networks*), scalability remains a significant challenge because of the larger resulting search spaces [2].

3. MULTI-LAYER GRAPH DATASETS

In this section, we introduce various multi-layer graph datasets. Figure 4 illustrates an example of a multi-layer graph called the AUCS dataset. In this graph, the multiple layers represent relationships between 61 employees of a University department in five different aspects: (i) coworking, (ii) having lunch together, (iii) Facebook friendship, (iv) offline friendship (having fun together), and (v) coauthorship. Popular datasets used in academic papers are as follows. Note that some layers are constructed *by using attribute information*. In these cases, an edge between two vertices is formed if attribute similarity is higher than a given threshold. This list is also available at `http://dm.kaist.ac.kr/datasets/multi-layer-network/`.

- **MIT Reality Mining [6]**
  This is a mobile phone dataset including 87 users on the MIT campus. Each layer represents the relationships defined by physical locations, bluetooth scans, and phone calls, respectively.
- **Enron Email [17]**
  This is an email message dataset between employees of the Enron corporation. It contains 200,399 messages belonging to 158 members. One layer contains the relationships defined by the existence of email communications, and the other contains those defined by the similarity of text messages.
- **Mobile Phone [6]**
  This is a mobile phone dataset collected by Nokia Research Center(NRC) Lausnne [8]. It contains

about 200 mobile users in Lausanne, Switzerland. Each layer represents the relationships defined by physical locations, bluetooth scans, and phone calls, respectively.

- **Cora [6]**
  This is a bibliographic dataset including 292 research papers. Layers represent three different research fields such as natural language processing, data mining, and robotics, respectively.
- **IMDB [1]**
  This is a movie database managed by IMDB, which contains 300 vertices and 18,368 edges. Vertices represent actors and edges are formed if two actors worked together. In this dataset, there exists four layers: (i) the first year of collaboration, (ii) the last year of collaboration, (iii) the average incomes, and (iv) the average number of sold tickets. In other words, four layers have the same edges but different edge labels.
- **Airline Transportation Multiplex [3]**
  This is a network composed of the airline operating in Europe. It contains 450 vertices and 3,588 edges. This data includes total thirty-seven layers and each one corresponds to a different airline.
- **SIAM Journal [25]**
  This is a bibliographic dataset containing 5,022 vertices which are papers. Five layers are formed from five different similarity matrices. First three are defined by the text similarity based on the abstract, title, and keyword, respectively. The other two are obtained by the number of common authors between papers and the citation relation.
- **Political Blogs [26] [28]**
  This is a webblog network on US politics, which contains 1,490 vertices and 19,090 edges. Vertices represent webblogs, and edges hyperlinks between webblogs. Each blog in the dataset has an attribute denoting its political position as either liberal or conservative. Thus, one layer depicts explicit hyperlinks, and the other depicts political preferences.
- **CiteSeer [12] [18] [21]**
  This is a citation network of computer science publications containing 3,312 vertices and 4,536 edges. One layer contains the relationships defined by citation, and the other contains those defined by content similarity.
- **US Stock Market [27]**
  This is a US stock market graph database containing 11 graph layers. On average it contains 3,636 vertices and 206,747 edges. Each layer is a graph made by setting the different correlation coefficient value based on stock price.
- **Arxiv Publication Database [28]**
  This is a bibliographic dataset including 13,396 vertices and 673,800 edges. Each layer corresponds to citation relationships with different research topics. Thus, the number of layers is equivalent to that of topics.
- **Flickr [17] [18]**
  This is a social network with tagged photos including 16,710 vertices and 716,063 edges. Each vertex represents a user, and the edge exists if the user is in another's contact list or if they favor the same images. In other words, one layer represents the relationships defined by the explicit contact list, and the other represents those defined by the common interest retrieved from photo sharing between two users.
- **DBLP [1] [20] [26] [28]**
  This is a bibliographic dataset including up to 108,030 vertices and 276,658 edges. A vertex stands for an author, and an edge is formed if two authors write a research paper together or share the same research interest. Using this dataset, we can make a two-layer graph as well as a general multi-layer graph whose layers are more than 2. In the two-layer graph, one layer contains the relationships defined by coauthorship while the other contains those defined by the sameness of the research interest. In the general multi-layer graph, each layer corresponds to the coauthorships in the different venues (conferences or journals).
- **LastFm [20]**
  This is a social music network which consists of 272,412 vertices and 350,239 edges. One layer represents the relationships based on friendships between users, and the other represents those based on the sameness of the musical tastes.
- **Higgs Twitter [5]**
  This is the multiplex of social interactions in Twitter including 456,631 vertices and 16,070,185 edges. Each layer represents friendship, replying, mentioning, and retweeting, respectively.
- **Wikipedia [18]**
  This dataset is from the static dump of English Wikipedia pages. It consists of 3,580,013 vertices and 162,085,383 edges. One layer contains the relationships defined by explicit page links, and the other contains those defined by text similarity between pages.

Table 2: The summary of multi-layer datasets.

No.	Name	# Vertices	# Edges	# Layers	Type	Publicly Available
1	AUCS	61	620	5	pillar	Y [1]
2	MIT Reality Mining [6]	87	-	3	pillar	Y [2]
3	Enron Email [17]	158	200,399	2	pillar	Y [3]
4	Mobile Phone [6]	200	-	3	pillar	N
5	Cora [6]	292	-	3	pillar	Y [4]
6	IMDB [1]	300	18,368	4	pillar	△ [5]
7	Airline Transportation Multiplex [3]	450	3,588	37	pillar	Y [6]
8	SIAM Journal [25]	5,022	-	5	pillar	N
9	Political Blogs [28] [26]	1,490	19,090	2	pillar	Y [7]
10	CiteSeer [12] [18] [21]	3,636	4,536	2	pillar	Y [8]
11	US stock market [27]	3,312	206,747	11	pillar	N
12	Arxiv publication [28]	13,396	673,800	7	pillar	N
13	Flickr [17] [18]	16,710	716,063	2	pillar	Y [9]
14	DBLP [1] [20] [26] [28]	108,030	276,658	various	pillar	Y [10]
15	LastFm [20]	272,412	350,239	2	pillar	△ [11]
16	Higgs Twitter [5]	456,631	16,070,185	4	pillar	Y [12]
17	Wikipidia [18]	3,580,013	162,085,383	2	pillar	N
18	FriendFeed [4]	9,717,499	15,000,000	various	general	Y [13]

- **FriendFeed [4]**
  This is one of social media aggregators. It contains about 400,000 users and 1 million posts with 15 million subscription relationships. In general, vertices stand for users, and edges various relationships between users. On the other hands, vertices can also be posts, and edges relationships between users and posts. Thus, layers can be various, for example, the types of services as well as the different relationships between users and posts.

Table 2 shows a brief summary of the multi-layer network datasets. If certain information of datasets does not exist in the reference papers, we fill in the blank with "−". For the "public available" column, if we can directly get the dataset through the web, we assign "Y". If we need extra efforts (*e.g.*, crawling) to get datasets, we assign "△".

4. COMMUNITY DETECTION IN TWO-LAYER GRAPHS

In this section, we introduce community detection algorithms in two-layer graphs. All algorithms described in this section can only support two-layer graphs and mostly consider structural and attribute information. One layer represents the original topology of a graph as structural information, and the other layer is derived by calculating the similarity between the vertices based on their attribute information. Such graphs with additional attribute information do not seem to conform to the definition multi-layer graphs. However, they have been regarded as a typical case of two-layer graphs since attribute information can be easily transformed to a layer—*e.g.*, by creating an edge between vertices if the attribute similarity between them is above a certain threshold. Thus, we categorize such graphs into two-layer graphs.

4.1 Cluster Expansion

Li *et al.* [12] proposed a hierarchical community detection algorithm based on both relations and textual attributes using the cluster expansion philosophy. This algorithm focuses on quickly finding initial cores as seeds of communities and expanding the cores into the communities in order to enhance scalability. In this paper, the CiteSeer dataset (No.10 in Table 2) was used. In the No.10 dataset, one layer represents the citation relationship between papers, and the other represents the degree of content similarity of the titles and abstracts of papers.

The algorithm consists of four major steps: core probing, core merging, affiliation, and classification. Figure 5 shows an overview of the algorithm (without specifying attribute information).

First, structural information is used solely to find cores, denoted as K_i, using the frequent itemset mining method derived from the Apriori algorithm.

[1] http://sigsna.net/impact/datasets/
[2] http://realitycommons.media.mit.edu/index.html
[3] http://bailando.sims.berkeley.edu/enron_email.html
[4] http://www.cs.umass.edu/~mccallum/data.html
[5] http://imdb.com
[6] http://complex.unizar.es/~atnmultiplex/
[7] http://networkdata.ics.uci.edu/data.php?id=102
[8] http://www.cs.umd.edu/projects/linqs/projects/lbc/index.html
[9] http://staff.science.uva.nl/~xirong/index.php?n=DataSet.Flickr3m
[10] http://informatik.uni-trier.de/~ley/db/
[11] http://www.last.fm
[12] http://www.plexmath.eu/?page_id=320/
[13] http://sigsna.net/impact/datasets/

After the set of all outgoing relations is listed for each document, the process of finding cores can be transformed into that of computing frequent itemsets. Each core will be used as a community seed. This step will enhance the scalability of the subsequent steps since the analysis scope is limited to each core. Then, cores are merged based on textual analysis using text similarity (*i.e.*, attribute information). In the core merging step of Figure 5, K_3 and K_4 are merged since they are linked and also topically relevant (not shown in the figure). In the affiliation step, initial communities are constructed through relation propagation. For each vertex v_i in a cluster C, the algorithm finds all vertices that are adjacent to v_i and adds them to C. Now every merged K_i is expanded to C_i in Figure 5. Since finding communities based solely on relation propagation may generate false hits, communities are refined based on classification using attribute analysis. In this step, LDA is used to reduce dimensionality, and all vertices are transformed into the feature vectors to represent their topical positions. Then, vertices are classified based on the SVM, and negatively labeled vertices are removed. For example, v_D is dropped from C_1.

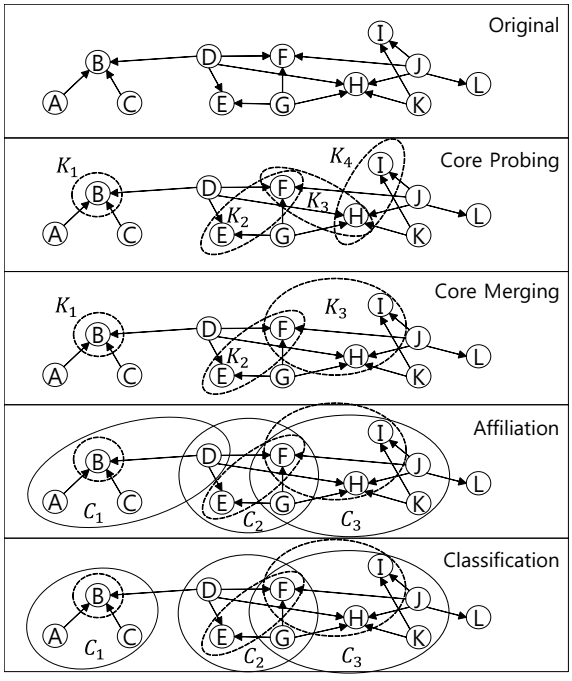


Figure 5: The overview of the community discovery algorithm [12].

4.2 Matrix Factorization

Qi *et al.* [17] proposed a community detection algorithm based both on link structure and edge content using the Edge-Induced Matrix Factorization (EIMF). In this paper, the Enron email and Flickr datasets (No.3 and No.13 in Table 2) were used. In the No.13 dataset, one layer depicts the relationships defined by the contact list, and the other depicts those defined by the favorite photo shared by the users.

The main contribution of this algorithm is using edge content for the community detection process. Edge content can be a useful source of information when nodes interact with multiple communities, since it can assist in distinguishing between the different interactions of nodes. Figure 6 shows an example of an edge-based social network in the No.13 dataset. Intuitively, edges can be divided into two different groups, such as a family (AB, BC, CD, AD) and people with similar musical interests (AE, AF). Moreover, it is clear that the user A belongs to both communities based on the edge content, whereas the same finding is unclear in terms of a vertex-centric perspective.

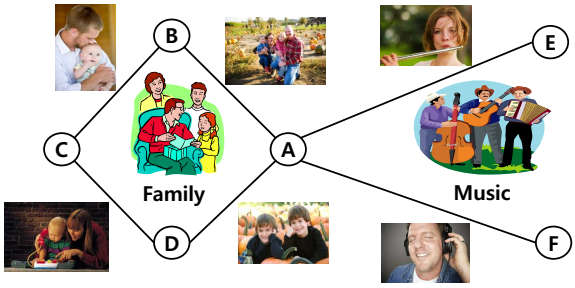


Figure 6: An example of an edge-based social media network [17].

This algorithm consists mainly of two parts: the EIMF based purely on the link structure, and incorporation of the edge content into the EIMF.

In the first part, an incidence matrix is formed using the link structure. Then, the latent edge matrix E is constructed from the incidence matrix using matrix factorization, which is obtained by minimizing Eq.(1).

$$O^l(E) = \| E^T \cdot E \cdot \triangle - \Gamma \|_F^2 \qquad (1)$$

Here, E is a $k \times m$ matrix with each column corresponding to a k-dimensional feature vector for an edge, Γ is an $m \times n$ incident matrix, and Δ is a normalization of the incident matrix such that every column-wise sum becomes 1. Then, by the definition of matrix factorization, Eq.(1) indicates the error of the approximation by E when compared with the link structure Γ. Since each column of E represents the membership of the edge to k communities, this procedure of matrix factorization can be regarded as a community detection technique using the link structure.

In the second part, two approaches are proposed in order to consider the edge content by way of reflecting the similarity among the edge content in matrix factorization. The former approach is to optimize Eq.(2).

$$O(E) = O^l(E) + \lambda \cdot O^c(E) \tag{2}$$

Here, $O^c(E)$ denotes the error of the approximation by E when compared with the similarity of edge content instead of the link structure, and λ is a weighting factor to consider the degree of importance of the link structure and edge content. The other approach is developed to avoid the necessity of tuning the parameter λ, and the reader can refer to [17] for the details.

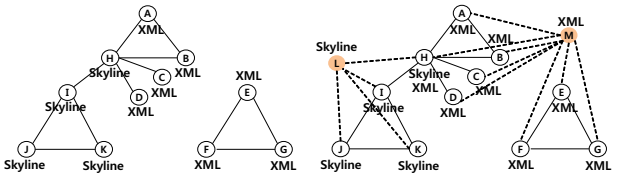
4.3 Unified Distance

Zhou *et al.* [28] proposed a community detection algorithm, called SA-Cluster, based on both structural and attribute similarities using a unified distance measure. In this paper, political blogs and the DBLP datasets (No.9 and No.14 in Table 2) were used. In the No.14 dataset, one layer represents the relationships created by coauthorship between researchers, and the other layer represents those defined by the similarity of research interests.

The main contribution of SA-Cluster is twofold: (1) a unified distance measure to fuse structural and attribute similarities; (2) a weight self-adjustment method to modulate the degree of importance of structural and attribute similarities.

First, the unified distance measure is formulated based on the *attribute-augmented graph* using the Random Walk with Restart (RWR). Figure 7a shows the original coauthor network, and Figure 7b shows the attribute-augmented graph with research topics. In the attribute-augmented graph, *attribute vertices* are added to represent attribute values, and the original vertices are connected to the corresponding attribute vertices. For example, the research topics, "Skyline" and "XML", are added as attribute vertices (two shaded vertices L and M in Figure 7b). Then, the researchers are connected via attribute vertices if they are interested in the same research topic. Intuitively, the larger the number of common attribute values between two vertices, the higher the degree of similarity between the two vertices, since more random walk paths can exist.

Second, the graph clustering algorithm that follows k-medoids clustering is performed based on the unified distance measure. More importantly, weight self-adjustment is conducted in each iteration of the algorithm. The weight of an attribute a_i in the



(a) A coauthor graph. (b) An augmented graph.

Figure 7: A coauthor network example with a topic attribute [28].

$(t+1)^{th}$ iteration is computed as Eq.(3).

$$w_i^{t+1} = \frac{1}{2}(w_i^t + \Delta w_i^t) \tag{3}$$

The weight increment Δw_i is measured by a majority voting mechanism. The voting mechanism counts the number of vertices within clusters that share the same attribute values for estimating clustering tendency of the attribute, and then adjusts the attribute weight. That is, if a large number of vertices within clusters have the same value of an attribute a_i, it denotes that a_i has high clustering tendency and increases the weight w_i of a_i accordingly.

4.4 Model-Based Method

Xu *et al.* [26] proposed a model-based community detection approach based on both structural and attribute aspects of a graph. In this paper, the datasets and graph layers were the same as those used in the authors' previous work [28].

The key point of this approach is the use of a *probabilistic model* that fuses both structural and attribute information instead of an artificial distance measure. The algorithm consists of two major parts: the construction of the probabilistic model and a variational approach to solve the model.

In the first part, a Bayesian probabilistic model is proposed for community detection over a clustered attributed graph. The clustered attributed graph used in this model is represented by $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$, where $\mathbf{X} = [X_{ij}]$ is an $n \times n$ adjacency matrix, $\mathbf{Y} = [Y_t^i]$ is an $n \times t$ attribute matrix, and $\mathbf{Z} = [Z_i]$ is a $n \times 1$ cluster vector that contains the label of a given vertex's cluster. This model defines a joint probability distribution $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$ for all possible communities and attributed graphs. α, θ, ϕ are parameters for generating the probabilistic model, where α denotes the vertex distribution of each cluster, θ implies the attribute distribution of each cluster, and ϕ denotes the edge occurrence probabilities between clusters.

Based on the model, the problem of community detection is transformed into a probabilistic inference problem, finding the maximum-a-posteriori

(MAP) configuration of communities $\mathbf{Z}$ with conditions $\mathbf{X}$ and $\mathbf{Y}$, as formulated by Eq.(4).

$$\mathbf{Z}^* = \underset{\mathbf{Z}}{\operatorname{argmax}}\ p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) \tag{4}$$

However, it is computationally infeasible to find the global maximum for a large set of $\mathbf{Z}$.

In the second part, a variational algorithm is introduced to solve the probabilistic inference problem. The major principle is to approximate the distribution $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$ using a variational distribution $q(\alpha, \theta, \phi, \mathbf{Z})$. Additionally, if we restrict the variational distribution to a family of distributions that factorize as Eq.(5), finding the global maximum translates as finding the local maximum in Eq.(6). Please refer to [26] for the details of the mathematical derivations.

$$q(\alpha, \theta, \phi, \mathbf{Z}) = q(\alpha)q(\theta)q(\phi)\prod_i q(Z_i) \tag{5}$$

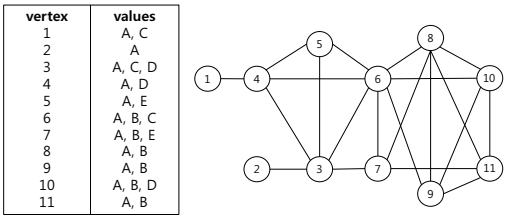
$$\begin{aligned}\mathbf{Z}^* &= \underset{\mathbf{Z}}{\operatorname{argmax}}\ p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) \\ &= [\underset{Z_1}{\operatorname{argmax}}\, q(Z_1), \underset{Z_2}{\operatorname{argmax}}\, q(Z_2), ..., \underset{Z_N}{\operatorname{argmax}}\, q(Z_N)]\end{aligned} \tag{6}$$

4.5 Pattern Mining

Silva *et al.* [21] proposed a community detection algorithm based on structural correlation pattern mining, called SCPM. In this paper, the CiteSeer, DBLP, and LastFm datasets (No.10, No.14, and No.15 in Table 2) were used. In the No.15 dataset, one layer contains friendships between users, and the other contains their shared musical preferences, *e.g.*, favorite singers.

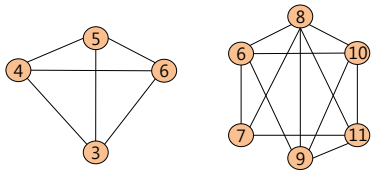
The main contribution of SCPM is to uncover the interaction between vertex attributes and dense subgraphs using both frequent itemset mining and quasi-clique mining. Here, a dense subgraph is defined by a γ-quasi-clique. The *structural correlation pattern* is formed if the proportion of the vertices in the dense subgraph that contain a given set of attribute values is above a threshold. In more detail, the algorithm first finds a frequent itemset S (*i.e.*, a set of attribute values appearing together in many vertices) from the entire graph G and obtains the subgraph G' induced by S. Then, it identifies a γ-quasi-clique Q from G'. Finally, the *structural correlation* of S is calculated by checking whether each vertex in G' belongs a quasi-clique Q. A structural correlation pattern should preserve a high value of structural correlation.

Figure 8 shows a toy example of SCPM. Figure 8a contains a set of attribute values for each vertex as well as an entire graph; Figure 8b depicts two examples of dense graphs. For a frequent itemset {A,B}, a subgraph {6,7,8,9,10,11} is induced from the entire graph, since these vertices include {A,B}. Then, a dense graph (the second one in Figure 8b) is obtained from this subgraph. Last, ({A,B}, {6,7,8,9,10,11}) is a structural correlation pattern with structural correlation 1, implying that the value set {A,B} appears on every vertex of the subgraph {6,7,8,9,10,11}.



vertex	values
1	A, C
2	A
3	A, C, D
4	A, D
5	A, E
6	A, B, C
7	A, B, E
8	A, B
9	A, B
10	A, B, D
11	A, B

(a) The graph with vertex attributes.



(b) The dense subgraphs.

Figure 8: An example of structural correlation pattern mining [21].

However, simply combining frequent itemset mining and quasi-clique mining will suffer from high computational overhead since the two problems are known to be #P-hard. Thus, two pruning techniques are proposed: (1) vertex pruning and (2) candidate set pruning. The former eliminates vertices that do not belong to quasi-cliques in the graph derived by a given attribute-value set or any quasi-clique in each iteration. The latter excludes candidate sets after the $(i+1)^{th}$ step if they do not satisfy the condition in the i^{th} step.

4.6 Graph Merging

Ruan *et al.* [18] proposed a community detection approach, called CODICIL, to combine structural and attribute information using the graph merging process. In this paper, the Wikipedia, Flickr, and CiteSeer datasets (No.17, No.13, and No.10 in Table 2) were used. In the No.17 dataset, one layer represents explicit hyperlinks, and the other represents content similarities.

The main contribution of this algorithm is to strengthen the community signal by eliminating noise in the link structure using content information. Figure 9 shows the work flow of the proposed approach. This approach consists of four steps: creating content edges, combining edges, sampling edges with bias, and clustering. First, for each vertex v_i, its k most content-similar neighbors are computed by calculating cosine similarity. Then, con-

tent edges are formed between the vertex v_i and its top-k neighbors. Second, the newly-created content edge set and the original topological edge set are simply unified. Third, for each vertex v_i, the edges to retain are selected from its local neighborhood based on either cosine similarity or Jaccard similarity. Last, clustering is performed on the merged graph. Since the process of merging graphs is performed independently of community detection algorithms, any conventional community detection algorithms can be applied.

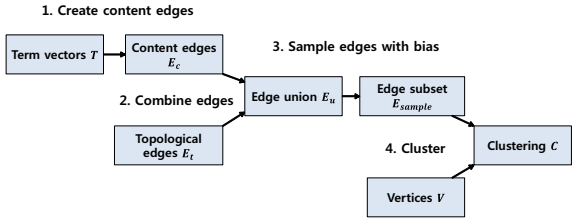


Figure 9: The work flow of CODICIL [18].

5. COMMUNITY DETECTION IN MULTI-LAYER GRAPHS

In this section, we introduce community detection algorithms that can support multi-layer graphs containing *more than or equal to two* layers.

5.1 Matrix Factorization

Tang *et al.* [25] and Dong *et al.* [6] proposed graph clustering algorithms for multi-layer graphs based on matrix factorization. In these papers, the MIT Reality Mining, mobile phone, Cora, and SIAM Journal datasets (No.2, No.4, No.5, and No.8 in Table 2) were used. In the No.2 and No.4 datasets, the layers represent the relationships defined by physical locations, bluetooth scans, and phone calls, respectively. In the No.5 dataset, the layers contain three different research domains: natural language processing, data mining, and robotics. In the No.8 dataset, the layers depict five different similarity matrices retrieved from the abstract, title, keywords, author, and citation fields.

The main idea of these two algorithms is to fuse different information by extracting common factors from multiple layers, which may then be used by general clustering methods. The major difference is that Tang *et al.* [25] approximates *adjacency matrices* while Dong *et al.* [6] approximates *graph Laplacian matrices*. To achieve this goal, they approximate each layer through a low-rank matrix factorization $O \approx P\Lambda P^t$, where O is an object matrix they try to approximate, which is either an adjacency matrix or a Laplacian matrix, P is an $n \times n$ eigenvector matrix, and Λ is an $n \times n$ eigenvalue matrix. When multiple layers are being considered, O is naturally extended to $O^{(i)}$, for $i = 1, \ldots, l$. Also, a common factor matrix should be reflected by the multiple factorizations. Hence, the objective function is defined as minimizing Eq.(7), where P is an $n \times n$ matrix representing the common factor of all layers, $\Lambda^{(i)}$ is an $n \times n$ matrix capturing the characteristics of i^{th} layer, $||\cdot||$ is the Frobenius norm, and α is a regularization parameter.

$$G = \frac{1}{2}\sum_{i=1}^{l} ||O^{(i)} - P\Lambda^{(i)}P^T||_F^2 + \frac{\alpha}{2}(\sum_{i=1}^{l} ||\Lambda^{(i)}||_F^2 + ||P||_F^2) \quad (7)$$

However, the solution of this objective function is not jointly convex in P and $\Lambda^{(i)}$. Thus, they proposed an alternative method that transforms the problem of finding the global minimum into that of finding the local minimum. In brief, they first fix P and optimize $\Lambda^{(i)}$, and then fix $\Lambda^{(i)}$ and optimize P. This procedure is repeated until the solution converges.

5.2 Pattern Mining

Zeng *et al.* [27] proposed a subgraph mining algorithm for finding quasi-cliques that appear on multiple layers with a frequency above a given threshold. In this paper, the US stock market database (No.11 in Table 2) was used. In the No.11 dataset, each layer represents a graph formed by different correlation coefficient values in terms of stock prices.

The main contribution of this algorithm is to find *cross-graph quasi-cliques* in a multi-layer graph that are frequent, coherent, and closed. Generally, the cross-graph quasi-clique has been defined as a set of vertices belonging to a quasi-clique that appears on all layers and must be the maximal set [16]. However, this algorithm does not limit the minimum support to be 100%, meaning that it attempts to find quasi-cliques on above a certain percentage of the layers in a multi-layer graph. The final output does not contain a quasi-clique Q if any superset of Q forms a quasi-clique with the same support, because the output must be closed.

To satisfy this goal, the algorithm first converts the subgraphs into their canonical forms. Since the algorithm does not take the exact topology of a quasi-clique into account as long as it satisfies given properties, the subgraph can be represented by the minimum string with the assumption that all vertices have the total order. Then, the algorithm enumerates feasible candidates for γ-quasi-cliques by using the DFS strategy with pruning techniques. Finally, the algorithm selects *closed* γ-quasi-cliques based on the closure-checking scheme. A naive approach of the closure-checking scheme

scans all γ-quasi-cliques, and then checks whether those quasi-cliques can be subsumed by other quasi-cliques. Since this naive approach is very costly, the algorithm adopts an efficient variational approach using the enumeration tree satisfying the condition that a descendant must subsume an ancestor. The key principle of the variational approach is to conduct the closure checking for each quasi-clique Q after all of its descendants have been processed.

Boden *et al.* [1] proposed a graph clustering algorithm in multi-layer graphs with edge labels, called MiMAG. In this paper, the IMDM, Arxiv, and DBLP datasets (No.6, No.12, and No.14 in Table 2) were used. In the No.6 dataset, each layer depicts different information about movies in which two actors star together. In the No.12 or No.14 datasets, each layer represents the citation or coauthorship relationships in different topics or conferences.

The main contribution of MiMAG is to find clusters, called MLCS (Multi-Layer Coherent Subgraph), satisfying both aspects of structural density and edge label similarity. In order to achieve the structural density of MLCS, a γ-quasi-clique model is used. For the edge label similarity of MLCS, a cell-based cluster model is used. Putting them together, the algorithm finds the densely-connected subgraphs whose edge labels vary at most by a certain threshold w. Such a subgraph is called an MLCS when it satisfies the two conditions on at least two layers.

However, listing all MLCSs produces numerous similar clusters, possibly containing redundant information, since MiMAG allows MLCSs to overlap with each other. For example, in Figure 10, the clusters C_2 and C_3 are redundant since they share a large number of the same vertices, *i.e.*, $\{f, g, h\}$, on layer 1.

In order to avoid redundancy, a redundancy relation is introduced [1]. It defines a cluster C to be redundant with respect to a cluster C' if the edges of C and those of C' overlap at a high rate and the quality of C' is higher than that of C. The quality

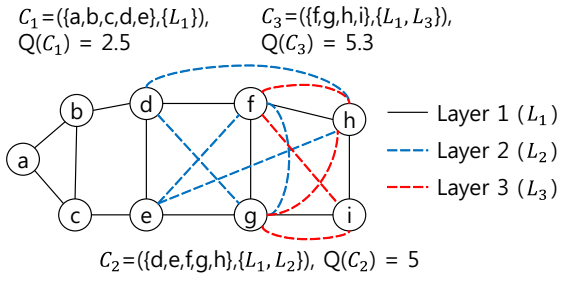


Figure 10: An example of overlapping clusters [1].

of a cluster $C = (V, L)$ is defined as Eq.(8), where V is a set of vertices, L denotes a set of layers, and $\gamma_L(V)$ represents the average density of the cluster on L.

$$Q(C) = \begin{cases} |V| \cdot |L| \cdot \gamma_L(V), & \text{if } |V| \geq 8 \wedge |L| \geq 2 \\ -1, & \text{otherwise} \end{cases} \tag{8}$$

Thus, MiMAG prefers the clusters that contain more vertices, contain more edges (*i.e.*, denser), and appear on more layers. In Figure 10, it is formally defined that C_2 is redundant with respect to C_3.

6. COMPARISON

In this section, we compare ten community detection algorithms introduced in Sections 4 and 5 with respect to the following seven properties. When selecting properties, we refer to the popular properties of subspace clustering [9] since community detection in multi-layer graphs resembles subspace clustering in considering that both methods deal with multiple dimensions of datasets. Among the properties in [9], only those closely related to community detection in multi-layer graphs are selected. Then, if a property is satisfied by none of the algorithms in this paper, we exclude it. Overall, P.1∼P.7 except P.4 correspond to a subset of the properties in [9]. P.4 is inspired by a widely-known categorization of attribute selection: the filter model and the wrapper model [22].

- **Property 1: Multiple layer ($l \geq 2$) applicability**
  All algorithms we introduced are designed for the community detection problem in multi-layer graphs. However, some algorithms support only a two-layer graph, while the others support a multi-layer graph containing more than two layers.
- **Property 2: Consideration of each layer's importance**
  Since each aspect of relationships may have different importance in the real world, considering the importance of each layer differently is more applicable than assigning uniform importance. Thus, it is crucial to automatically find the importance of each layer based on the layer's characteristics. We call the importance of each layer its *layer coefficient*.
- **Property 3: Flexible layer participation**
  The layer coefficient can vary across communities. Thus, capturing the optimal layer coefficient *specific to each community* is an important ability since it can distinguish the layer participation in each community. In this case, an algorithm can freely construct a community involved with a sub-

set of layers rather than the entire or common set of available layers.

- **Property 4: Algorithm insensitivity**
  Some approaches are tightly coupled with a specific graph clustering algorithm. This tight coupling may limit the freedom of users to choose a graph clustering algorithm. It is well-known that certain graph clustering algorithms tend to perform particularly well or poorly on certain kinds of graphs [11]. Thus, an ability of applying any clustering algorithms can improve the quality of community detection.
- **Property 5: No layer locality assumption**
  Some approaches find initial communities from a specific layer and then discover final communities by expanding and refining the initial communities on other layers. Those algorithms are regarded to have *locality assumption*. In other words, it is assumed that all hidden communities can be derived from a local region of the layer.
- **Property 6: Independence from the order of layers**
  The results of community detection could be sensitive to the order of processing layers. This limitation typically happens when an algorithm processes layers *sequentially* with a dedicated policy for each layer. In this case, an improper ordering will result in lower-quality results.
- **Property 7: Overlapping layers**
  The communities can be defined in an overlapping way across layers. That is, a vertex can belong to a community C_1 on a certain set of layers but to a community C_2 on another set of layers.

Table 3 shows whether each algorithm supports the seven properties. Our perspective is that more Y's indicate that the algorithm has more powerful and advanced features. Nevertheless, we cannot definitely say that the number of Y's determines the superiority of an algorithm over another. Some algorithm does not need all the properties if it is designed for specific environments. In addition, the performance in terms of efficiency or accuracy is not addressed in Table 3, since an apple-to-apple comparison is not possible owing to the differences in problem settings. Overall, despite of these limitations, we believe that this comparison will give useful insights into various approaches.

Table 3: The comparisons of community detection algorithms for multi-layer graphs.

Algorithm	P1	P2	P3	P4	P5	P6	P7
Li *et al.* [12]	N	N	N	N	N	N	Y
Qi *et al.* [17]	N	Y	N	Y	N	N	Y
Zhou *et al.* [28]	N	Y	N	N	Y	N	N
Xu *et al.* [26]	N	Y	N	N	Y	N	N
Silva *et al.* [21]	N	N	Y	N	Y	N	Y
Ruan *et al.* [18]	N	N	N	Y	Y	Y	N
Tang *et al.* [25]	Y	Y	N	Y	Y	Y	N
Dong *et al.* [6]	Y	Y	N	Y	Y	Y	N
Zeng *et al.* [27]	Y	N	Y	N	Y	Y	Y
Boden *et al.* [1]	Y	N	Y	N	Y	Y	Y

7. FUTURE RESEARCH DIRECTIONS

In this section, we present a few challenging but interesting future research directions.

- **General multi-layer graph applicability**
  Most algorithms covered are only applicable to *pillar* multi-layer graphs. It is definitely true that they are simple but effective to model various real-world situations. However, since a one-to-one correspondence between vertices of different layers is not always guaranteed in the real world, it is more natural to consider an extension of the algorithms into *general* multi-layer graphs.
- **Uncertainty in multi-layer graphs**
  Most studies assume that multi-layer graphs are already cleaned completely. However, in the real world, both vertices and edges could be noisy and ambiguous [23]. For example, in bibliographic datasets, different authors may have the same name. Even worse, information extracted from the real world may not be reliable. Thus, constructing multi-layer graphs with entity resolution and/or trustworthy analysis certainly enhances the quality of the community detection process.
- **Scalability issues**
  In the era of Big Data, the amount of available information grows rapidly. Thus, scalability of both computational time and memory requirement has become a critical issue. Although many researchers are trying to enhance scalability, most studies are being conducted with relatively small datasets because of unsatisfactory scalability. One of feasible solutions is to implement parallel and distributed versions of a community detection algorithm. Another is to use sampling for feature-vector matrices of multi-layer graphs.
- **Temporal analysis**
  Graphs evolve over time, and the communities in graphs also change as time goes by. Thus, understanding and exploiting temporal characteristics are helpful for discovering deep insights about the communities. Although many researchers have studied this problem for single-layer graphs, there is almost no work done for multi-layer graphs. The complexity of modeling the evolution in multi-layer graphs is extremely high since it involves multiple layers and the connections between the multiple layers.

8. CONCLUSIONS

In this paper, we presented a comprehensive understanding of multi-layer graphs and the state-of-the-art community detection algorithms for multi-layer graphs. In recent applications, each entity often engages in multiple relations. Hence, the qualified communities in multi-layer graphs can be discovered by the way of exploiting and fusing all these different aspects of information. We classified community detection algorithms in multi-layer graphs into the six types based on their underlying strategies: cluster expansion, matrix factorization, unified distance, model-based, pattern mining, and graph merging. These algorithms were compared with each other using seven properties. Also, various multi-layer graph datasets used in related studies were summarized for ease of reference. Finally, we tried to provide insights and directions for further research in this domain.

9. ACKNOWLEDGMENTS

This research, "Geospatial Big Data Management, Analysis and Service Platform Technology Development," was supported by the MOLIT (The Ministry of Land, Infrastructure and Transport), Korea, under the national spatial information research program supervised by the KAIA (Korea Agency for Infrastructure Technology Advancement) (15NSIP-B081011-02).

10. REFERENCES

[1] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl. Mining coherent subgraphs in multi-layer graphs with edge labels. In *Proc. 2012 ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 1258–1266, Beijing, China, Aug. 2012.

[2] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028, Apr. 2010.

[3] A. Cardillo, J. Gómez-Gardeñes, M. Zanin, M. Romance, D. Papo, F. del Pozo, and S. Boccaletti. Emergence of network features from multiplexity. *Scientific Reports*, 3(1344), Feb. 2013.

[4] F. Celli, F. M. L. D. Lascio, M. Magnani, B. Pacelli, and L. Rossi. Social network data and practices: The case of friendfeed. In *Proc. 3rd Int'l Conf. on Social Computing, Behavioral Modeling, and Prediction*, pages 346–353, Bethesda, Maryland, Mar. 2010.

[5] M. D. Domenico, A. Lima, P. Mougel, and M. Musolesi. The anatomy of a scientific rumor. *Scientific Reports*, 3(2980), Oct. 2013.

[6] X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov. Clustering with multi-layer graphs: A spectral perspective. *IEEE Trans. on Signal Processing*, 60(11):5820–5831, Dec. 2011.

[7] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, Feb. 2010.

[8] N. Kiukkonen, J. Blom, O. Dousse, D. Gatica-Perez, and J. Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. In *Proc. The 7th Int'l Conf. on Pervasive Services*, Berlin, Germany, July 2010.

[9] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. on Knowledge Discovery from Data*, 3(1):1–58, Mar. 2009.

[10] S. Lee, M. Ko, K. Han, and J.-G. Lee. On finding fine-granularity user communities by profile decomposition. In *Proc. 2012 ASONAM Int'l Conf. on Advances in Social Networks Analysis and Mining*, pages 631–639, Istanbul, Turkey, Aug. 2012.

[11] J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In *Proc. 19th Int'l World Wide Web Conf.*, pages 631–640, Raleigh, North Carolina, Apr. 2010.

[12] H. Li, Z. Nie, W.-C. Lee, L. Giles, and J.-R. Wen. Scalable community discovery on textual data with relations. In *Proc. 17th Int'l Conf. on Information and Knowledge Management*, pages 1203–1212, Napa Valley, California, Oct. 2008.

[13] S. Lim, S. Ryu, S. Kwon, K. Jung, and J.-G. Lee. LinkSCAN*: Overlapping community detection using the link-space transformation. In *Proc. 30th Int'l Conf. on Data Engineering*, pages 292–303, Chicago, Illinois, Apr. 2014.

[14] M. Magnani and L. Rossi. The ML-model for multi-layer social networks. In *Proc. 2011 ASONAM Int'l Conf. on Advances in Social Networks Analysis and Mining*, pages 5–12, Kaohsiung City, Taiwan, July 2011.

[15] S. Moon, J.-G. Lee, and M. Kang. Scalable community detection from networks by computing edge betweenness on mapreduce. In *Proc. 2014 Int'l Conf. on Big Data and Smart Computing*, pages 145–148, Bangkok, Thailand, Jan. 2014.

[16] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *Proc. 2005 ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 228–238, Chicago, Illinois, Aug. 2005.

[17] G.-J. Qi, C. C. Aggarwal, and T. Huang. Community detection with edge content in social media networks. In *Proc. 28th Int'l Conf. on Data Engineering*, pages 534–545, Brisbane, Australia, Apr. 2012.

[18] Y. Ruan, D. Fuhry, and S. Parthasarathy. Efficient community detection in large networks using content and links. In *Proc. 22nd Int'l World Wide Web Conf.*, pages 1089–1098, Rio de Janeiro, Brazil, May 2013.

[19] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, Aug. 2007.

[20] A. Silva, W. M. Jr., and M. J. Zaki. Structural correlation pattern mining for large graphs. In *Proc. 8th Workshop on Mining and Learning with Graphs*, pages 119–126, Washington D.C., Aug. 2010.

[21] A. Silva, W. M. Jr., and M. J. Zaki. Mining attribute-structure correlated patterns in large attributed graphs. *Proc. of the VLDB Endowment*, 5(5):466–477, Sept. 2012.

[22] K. Sim, V. Gopalkrishnan, A. Zimek, and G. Cong. A survey on enhanced subspace clustering. *Data Mining and Knowledge Discovery*, 26(2):332–397, Feb. 2013.

[23] Y. Sun and J. Han. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD Explorations Newsletter*, 14(2):20–28, Dec. 2013.

[24] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proc. of the VLDB Endowment*, 4(11):992–1003, Aug. 2011.

[25] W. Tang, Z. Lu, and I. S. Dhillon. Clustering with multiple graphs. In *Proc. 9th Int'l Conf. on Data Mining*, pages 1016–1021, Mianmi, Florida, Dec. 2009.

[26] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *Proc. 2012 ACM SIGMOD Int'l Conf. on Management of Data*, pages 505–516, Indianapolis, Indiana, June 2012.

[27] Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *Proc. 2006 ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 797–802, Philadelphia, Pennsylvania, Aug. 2006.

[28] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *Proc. of the VLDB Endowment*, 2(1):718–729, Aug. 2009.

Rick Cattell Speaks Out on Patenting, Reinventing and Standardizing Things

Marianne Winslett and Vanessa Braganholo

Rick Cattell
http://www.cattell.net/

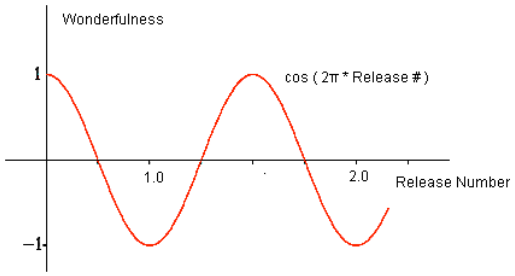
Welcome to ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are in Tiburon, California, at the home of Rick Cattell, who is an independent consultant. Rick spent over 20 years at Sun Microsystems, where he was involved with many things that we take for granted today, such as ODBC, JDBC, and J2EE. Rick was one of Sun's first Distinguished Engineers, and his dissertation on compiler technology won the ACM Dissertation Award. So, Rick, welcome!

Thank you Marianne, it's good to be here.

What is a patent troll?

So I've been working a lot with companies that are dealing with patent trolls. A patent troll is a company who basically does no useful work, but they have patents and go after big companies with those patents. They either bought the patents or acquired them in some fashion and they threaten lawsuits. In almost all the cases I've been involved in, Fortune 500 companies will generally settle out of court because they do not want to spend a couple million dollars proving that the patent is not valid. In all of the cases I've been involved in, the patents are definitely not valid (in my opinion). The US patent office, since they've started granting patents on software in about 1990, seems to not understand software very well, so people get patents for the silliest things!

Wonderfulness is equal to the cosine of 2π times the release number [of a product].



Like what?

Like putting a cursor in a field on the screen. Also, multiple people get patents on the same thing! Right now, there are about 15 different patents on object-relational mapping from an object-oriented programming language to a database system. They use different and confusing words, so apparently the patent examiner thought they were different things. And as far as I can tell, none of these people actually invented object-relational mapping, which was invented as early as 1990 or 1989 in academia and in products, independently. It's tedious helping with this patent work, but it's also satisfying in a sense that I feel like these patent trolls are standing in the way of progress in computer science and in the way of people building real useful products -- they're just trying to make a buck. So I've been doing a lot of that work lately.

Okay so it sounds like a vampire sucking our creative blood, but who is the original filer of the patents? Are these big companies themselves? Or random evil individuals? Or startups that fail? Or what?

They are typically startups that fail. Startups almost always file patents on the things that they are doing. When they fail, their patents go on the auction block, or get acquired by another company in some fashion. Then they turn into live ammunition that's out there in a dangerous spot.

So do you think those individual startups thought they had come up with a new idea when filing these object-relational patents? Or just hadn't read the literature? Or what?

That's a good question, I often wonder about that. I've certainly talked to some people who think that they invented object-relational mapping and then I had to show them the previous work. It's a natural thing when you have a new idea. There are new ideas whose time has come and everybody sort of thinks of them at the same time. So here there was object-oriented programming, and there were relational database systems, and people wanted to connect them. Well, let's do object-relational mapping! So I would expect that a number of people think that they actually invented it, even though they didn't invent it first.

You actually told me earlier that "the system works in favor of invalid patents".

Yes.

That's awful!

Yes, here is the problem. I think this is probably not such a bad problem in other areas where the patent office has a hundred years worth of prior art and they understand it and it's well established what's new and what's not, but in software, things are very confused. So a patent troll with a patent threatens a lawsuit with a Fortune 1000 company or whatever. The company looks at the patent and on the surface it seems like they violate this patent. Proving that the patent is invalid is very difficult. Typically, a patent holder will bring a lawsuit in the Eastern District of Texas that is generally viewed as favorable to patent holders. The

problem, being faced by a big company, is they have to explain to a jury of laymen why some piece of prior art, which is a fairly complicated piece of software, invalidates this patent with a fairly complicated set of claims. That is difficult to do. They often view it as a roll of the dice -- they don't know what is going to happen when the jury makes a decision. It's a risk for them. So rather than spending 2 million dollars and go to court and fighting, they all settle. They all give $50,000 or $100,000 to the patent troll or whatever. Then the patent troll can go on to another big company with their patents.

Surely, there must be a better way to do things.

Yes, I've put a lot of thought into that. I often thought that the jury of peers should actually be people knowledgeable in computer science or whatever the patent is about, but that's not the system we have had for a couple hundred years. It would be easy to explain invalidity to someone with a bachelor's degree in computer science.

Well how do they do it in other science and engineering? So, what's the secret there?

It's the same problem, but for some reason software is more complicated. Even with a new drug, it's pretty clear whether it is or isn't the new drug. In automobiles… where people can generally understand how the parts fit together, you can explain it to them. In software, it's complicated to explain it to a layman.

So do you think we should get rid of software patents?

That's a good question. I think they serve a good purpose between companies that really have invented something and want to protect that intellectual property. I think the threshold for the obviousness of patents and the innovation in the patent has to be higher than it is right now in software, and then they would work better.

How would that happen?

That's a good question. I often ask attorneys about changing the system. There was an attempt in 2006 to improve the patent system, but it's still broken in my mind. It's a system we've had for hundreds of years… 200 years? So it's hard to change, especially because when congress considers a change, there is certainly a lot of invested interest in existing patents so there will be a lot of lobbyists in there. I don't have a good solution or I would have written a paper about it by now.

Back when people first started talking about object databases there were a lot of arguments about "my data model is better than yours". That's a really hard kind of argument to make. It is kind of religious in nature. Was that the right argument for them to be making?

I don't think so. I think the right argument is to say "people like object-oriented programming languages and they need to make their data persistent". These systems solved that problem. And then later with object-relational mapping, they actually solved that in a different way, storing the data in a relational database system. That is the strong argument for object oriented-databases and object relational mapping, respectively.

So you helped change that argument.

Yes.

What did you do?

Well, when object-oriented databases first came out, I was concerned that there was no standard like SQL for relational database systems so they would fail just because people didn't want to count on something that had a different API in every other company. So we made some progress on that. And then later at Sun, I was involved in standards in various ways. We can talk about that later if you want.

You can mention them. We might come back to them.

So I learned something from the ODMG object database standard experience that I applied to JDBC when I was at Sun trying to get a dozen different companies to agree on a standard.

Yeah?

Standards typically turn into a design by committee that is very confusing and perhaps inefficient and clumsy to use. The system that I used with JDBC was that there was a specification lead (that was me) and I took input from everybody. We often took votes on things, but there was one person responsible for the integrity of the document and the standard, so I believe it came out better as a result. The same thing got applied in other areas of the Java community. I wrote up what I learned from JDBC and passed it among the Sun management about how to make a successful, simple yet powerful standard. Those ideas eventually evolved into the Java community process, which I believe has worked pretty well over and over again in different standard arenas.

Do you think something like that could have worked for SQL?

It did work for SQL, because in that case, IBM was the specification lead for something that became the standard. And in fact if you look at almost every successful standard like Unix, it was originally done by one company or one small group of people and then was adopted as a *de-facto* standard. The standards that were designed by a committee have often failed. You can find some exceptions, but generally, the best standards were initially done by one person or one company.

What about the Web and the W3C, where they were working on standardizing things that don't exist yet?

That's hard, but it can be done. You can do standards by committee and try to do innovation, but it's difficult to do innovation in a group.

Back to the "my data model is better than your data model" argument… In a sense, that was the argument for Java and they actually won that argument.

Yeah, Java is a nice programming language and it has a pretty good model, but I claim that it didn't succeed just because it was better. I think it succeeded because it was in the right place at the right time. In fact I was very frustrated about the time Java came out because at Sun we were stuck with C and C++. After being at Xerox PARC working with languages that did automatic garbage collection and were safe, I found myself making stupid errors: forgetting to free memory, clobbering memory… errors that are very difficult to fix. Java wasn't the first language to solve that problem. There were other languages that were equally good, yet they were not popular. They didn't take off in the way that C and C++ did. So, here with Java, finally there was a language that was good and it was popular and so I could expect there to be lots of libraries and activity around it. So I got excited about it around 1995. I left my work on relational and object database systems at Sun, and focused on Java. Of course I then went back and did JDBC, so I was still in the database area. Those were exciting times -- there were only about a dozen people in the Java group when I joined and it grew very fast.

What was Java originally intended to be used for?

That's a good question. James Gosling and his group had been looking at devices like set-top boxes where they wanted to be able to send programmed material to a box in a remote location. Yet you wanted it to be secure and not crash the box if there was a faulty program. So Java took off with the Internet even though that wasn't the original intent. There were some sharp people at Sun that figured out that this was a way to convey programs in a browser and jumped on it.

Okay, so they started using it on the client side?

Yes. And my contribution to Java was thinking, "wow, this would also be a great language for servers", because there, you are also trying to build websites with some application server logic that you need to be safe and that you need to be able to easily move around. So I started this group doing what I called Enterprise Java at the time, which is now known as J2EE (or Java Enterprise Edition) to build a set of APIs, a platform for building server side Java programs which required additional functionality that wasn't necessary on the client.

People look at the wrong things when they are trying to tune [database systems]. They don't realize that the number of machine instructions that are used in a database call is now a critical factor they are trying to minimize.

How have things changed in the past decade, for database applications?

Two things. One is that the hardware has changed. RAM has gotten much cheaper, so that you can store a lot of your database in memory, and flash also gives you a way to make very fast reads and writes to databases, which was not possible with disk. The other thing is a change in the market in that there are a lot more people out there trying to deal with really big databases with lots of users because every web company is at least dreaming of having millions of users and they want to be able to scale up to that. So there's a lot of interest in scalable database systems today, which is where I've been focusing in my consulting practice.

What's the right way to respond to those two trends?

I think there's a lot of excitement around scalable database systems. There's also a lot of confusion around them. There are perhaps a dozen new database

systems just in the last 18 months to 24 months[1] that are so called NoSQL database systems or are scalable SQL database systems. So there is plenty of work for consultants like myself who are familiar with the various products and their strengths and weaknesses. Customers just need to make a careful choice of what are the requirements and which database systems really satisfy those. That's the age-old story in software, making sure it solves your problem.

Did you say a dozen new products?

Yes, I would say that.

> ***The system works in favor of invalid patents.***

That's a lot! So it is really responding to this need.

Yes, and most of these systems are open source. So there are communities of people who work around them trying to tune them. In my experience, a number of the systems are still immature. The bugs haven't been shaken out and they have some performance characteristics that are lacking when you go beyond the simple case they were originally designed for. So I think we're going to see some fallout in this industry and I've been doing a lot of benchmarking of systems with my clients to look at which ones actually scale, instead of running into all kinds of communication bottlenecks when there are more than 4 servers.

So I would love to ask you which systems perform best on your benchmarks but one of the lessons that database researchers have learned is that it's not good to talk about that. So I won't ask you about that but I will ask you about what are the types of scaling that the systems need to do and what are the good ways of achieving it.

So, in some ways that hasn't changed in the last 10 or 20 years. You can achieve horizontal scaling across multiple servers by doing replication. This gives you scalability for reads and also gives you a way to recover from crashes. And by doing partitioning of the database across multiple servers, you can split the load of writes and reads over multiple servers.

That's not new, right? I remember learning about that in grad school. So what's new? I guess nothing is new under the sun. That's what the patent story is. It's nothing really new, but there must be something new here?

Well, with the two things I mentioned… more people are interested in this. They care and there are some open source systems built around these ideas. The other thing is that RAM is cheap and flash memory is cheap and that there are new mechanisms to do faster communication in between machines that will reduce the overhead. Also, you get more cores per computer now than ever before, which allows you to do better vertical scaling on each machine.

So what is vertical scaling?

Vertical scaling is using all of those cores that you have on a modern computer effectively. In the bad old days, you weren't too far off just having a single process per machine that processed all the database calls. Now you can't afford to do that if you want to utilize all the power of the computer that you have in front of you.

So are we still in a world where we are trying to minimize the number of disk accesses?

No, that has changed for most database systems. Oracle, Informix, Ingres, all major database systems were originally designed with the goal of minimizing the number of disk accesses and now that story has changed. There are papers about that like the one "It's Time for a Complete Rewrite"[2]. The things you're trying to optimize in a relational database have changed, so it's time for a complete rewrite. Some of the NoSQL systems are doing their style of rewrite and other people with products like Clustrix and VoltDB are going in a different direction with relational databases, taking advantage of splitting the data across multiple machines, using RAM effectively, and so on.

Some of the age-old rules still apply and people forget that. People forget that if you're going to scale to a dozen or a hundred machines, you can't have any manual intervention. You can't have the database go down and have an operator come in and fix it. The system has to be self-repairing. When a server fails, it has to be replaced online automatically. If you have to change your schema, you can't bring a hundred or a thousand machines down while you upgrade your

[1] Editor's note: recall that this interview was conducted in 2011.

[2] STONEBRAKER, M., MADDEN, S., ABADI, D., HARIZOPOULOS, S., HACHEM, N., HELLAND, P. The end of an architectural era: it's time for a complete rewrite. In: VLDB, 2007. pp 1150-1160.

schema. It has to be always online. That is something that I often find: clients in a startup that are not familiar with. They say, "okay, we'll just use more servers". They forget you also need high availability and continuous online operations.

Are there any new research issues there that people haven't already looked at?

Well there are in the performance arena, I believe, because a lot of the performance is counter intuitive. Where is the time actually going in a complex system like that?

> ***It would be easy to explain invalidity [of a software patent] to someone with a bachelor's degree in computer science.***

Counter intuitive? Like what?

So people look at the wrong things when they are trying to tune. They don't realize that the number of machine instructions that are used in a database call is now a critical factor they are trying to minimize. They don't realize that the inter-machine cost can be fairly expensive. It can take a thousand machine instructions just to move one byte from computer A to computer B. So your database design has to be built for minimizing the inter-machine calls with a new sense of urgency that wasn't the case in the past.

I see. Sounds like it's time for a rewrite for the textbooks also.

Yes, I would say so.

You spent much of your career at Sun, which didn't have any database product. That's counterintuitive too, but you exerted a lot of influence from there, including on the standards like we were talked about and de-facto standards. Could you have been as influential if you were in a company that did have a database product?

Yes and no. I did have some advantage, being in a neutral position. For example, by working with various database companies at Sun, which didn't have its own database product, I was able to get a lot of cooperation on tuning for their database systems with our operating system. Also with the ODMG with the object databases companies, I was in a neutral position. Even with Java, I was in a relatively neutral position, so that helped.

In retrospect, working for Sun did limit my career in many ways because we didn't have a database system product that I could work on. In fact, when I came to Sun, I was hired by Eric Schmitt to start a database group and the first thing I decided is that we actually didn't need a database group to build a database system. At that time, IBM, HP, DEC and even Apollo had their own database system and my conclusion was that it was better for us to just work with all of the vendors and they worked harder to have the best performance on our platform and to sell on our platform instead of selling against say, IBM, who had both a database system and a server they were trying to sell.

I flipped my position on that around 2000, 15 years after I went to Sun, saying, "now maybe it makes sense for Sun to have its own in-memory database cache in front of these relational database systems and that in-memory cache can actually evolve into being an in-memory distributed database system". Unfortunately around 2001 and 2002, Sun's profits were falling and there was not a lot of money to start a whole new project. I actually wasted some time at Sun thinking each year that I was just 6 months away from getting funding for building a distributed in-memory database system but it never happened and I gave up around 2007, when then I went out on my own.

Speaking of being out on your own, you're self-employed and doing research. I've never heard of a self-employed person who is doing research. So how does that work? Are you your own funding agency?

I am, I guess. I think every software consultant actually has to do some amount of research doing a benchmark here, doing a study of the details of an implementation in order to be familiar with what they are talking about. So I might be doing a little bit more of that than others, but I think that is actually necessary in order to give advice on database systems.

Speaking of advice, do you have any words of advice for fledging or mid-career database researchers or practitioners?

Yes, that's an area near and dear to my heart because I started writing this book that's been on the back burner for a long time. It's entitled, "Things I Wish I Learned in Engineering School". When I got my PhD, I went

out into research and then went out into industry building products. I learned over and over again that people waste a lot of time, spending many months or years building something that you could have known was not going to succeed, if you had had some experience with things that companies do wrong.

So how can you tell if something is not going to succeed?

Well my book is a list of rules to follow, advice to follow to avoid the errors that people commonly fall into. For example, rule #1 in the book is that most organizations, most startups, most projects, try to do too many things. They're almost always trying to do too many things. The landscape is littered with startups that are trying to do too many things. In fact I've consulted with for at least one of them where they didn't take my advice on it. So Steve Jobs for example, whom you think is the epitome of a great leader and a successful leader, started NeXT computer, which failed. He tried to do his own hardware from the bottom up. His own operating system, a new programming language, a new window system, a new programming environment, new tools, new ideas… and he was trying to do it all at once with finite resources, competing against existing players like Apple and Microsoft on Intel, which had established application bases. So he was working uphill in more than one way. He violated multiple rules in my book. He was trying to do too many things. And he was trying to displace an established market player without enough "better" to displace them.

Yeah there aren't enough places for being the enough better to displace the established leader.

So my book has four chapters. One is about successful organizations: why organizations fail and succeed. A second is about technology, errors that people make in technology. Like error #33 where you come up with a new idea that has a 30% chance of success and base it on another new idea that has a 30% chance of success. Now you have like a 10% chance of that your system succeeds. The third chapter is about successful products. There's a rule in there that says "wonderfulness is equal to the cosine of 2π times the release number".

What??

What this means here is that when you initially come up with a new idea, it sounds wonderful and you throw in all this other new stuff that sounds wonderful and you finally get to a proposed release. At Release #1 you're at the top of the curve in terms of the wonderfulness of what you're going to do. And then the product gets released and you discover all of the problems that people come in with. There are issues and there are bugs and you realize that there are missing features that were necessary. So you go down the wave again, and then the second release begins where you say, "Oh, we can fix all of that". We're going to put in this and that and it's going to be wonderful again. We go back up the curve until Release #2 comes out. So this is something you should be aware of. If you go to a startup, everyone is really excited about this first release. When you listen to Steve Jobs talk about his product, for example, he has this reality distortion field and everyone in the room will be convinced that this is going to be the best thing since sliced bread. So you just have to be aware of these trends when you put out a product.

The final chapter is about career advice. For example, you need to spend time increasing your effectiveness. For example, one of the best things I ever did was take a typing class in 8th grade. I didn't realize at the time how useful that was going to be to me. It's useful as a programmer to spend a lot of time learning about tools and what you need to do because if you've got it all in your head you're not constantly stalled, not realizing there's a solution to the problem you have or a better tool to solve the problem that you have. There are 80 rules in the book, twenty in each of those chapters, about things that you ought to know.

Now where can I read it? Or did you just get that?

On my website, Cattell.net, there is a presentation that I have given at the University of Illinois and a couple other universities that summarizes some of the ideas. If they are interested in giving me comments on the manuscript, then they can send me an email and I'd be happy to share it with them.

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

That's a hard question. I like the work I did back in Xerox PARC, on new kinds of user interfaces to database systems where you can see things laid out spatially or browse around the databases by clicking on things. I did a little bit of work on that at Sun initially but I was in the wrong place. I was in a hardware company trying to do an innovative software product. That would've been fun to have spent more time on.

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

In database systems, I'd like to have a little bit more time to experiment with the properties of these distributed systems. I'd like to setup a lab of a hundred computers and experiment with the characteristics and see what happens when you change the way database systems work, but that's a bit too big of a project.

No, it's a fine answer because the question was if you magically had enough time (and money, I guess), right? If you could change one thing about yourself as a computer science researcher, what would it be?

I think I made a mistake in my career in not choosing to focus on either doing research or doing products and trying to do everything at the same time for my entire career. It would be good to do one and then do the other and then the one. For example, I often considered starting a company myself and doing a new kind of database system. I didn't want to make the sacrifice of the time out of my personal life to do that. It also would put my research work at a standstill. In retrospect, it probably would have been good to go out and make some mistakes starting a new company (maybe more than once). Then go back into research or whatever.

Okay, thank you very much for talking with me today.

Thank you! I've enjoyed your interviews on SIGMOD. This is a great service you're doing for our community.

Thank you!

Report on the Third International Workshop on Energy Data Management (EnDM 2014)

Torben Bach Pedersen
Center for Data-intensive Systems (Daisy)
Aalborg University
9220 Aalborg, Denmark
tbp@cs.aau.dk

1. INTRODUCTION

The energy sector is in transition–being forced to re-think the current practice and apply data-management based IT solutions to provide a scalable and sustainable supply and distribution of energy. Novel challenges range from renewable energy production over energy distribution and monitoring to controlling and moving energy consumption. Huge amounts of "Big Energy Data," i.e., data from smart meters, new renewable energy sources (RES–such as wind, solar, hydro, thermal, etc), novel distributions mechanisms (Smart Grid), and novel types of consumers and devices, e.g., electric cars, are being collected and must be managed and analyzed to yield their potential. Energy is at the top of the worldwide political agenda. For example, The European Union has stated the "20-20-20 goals" (20% renewable energy, 20% better energy efficiency, and 20% CO2 reduction by 2020). Even more ambitious goals are set for 2030 and 2050. This situation is reflected in research funding schemes such as Horizon 2020 as well as national programs. Increasingly, such programs include joint calls involving both energy and IT partners. Data management is at the heart of this development.

Thus, data management within the energy domain becomes increasingly important. The International Workshop on Energy Data Management (EnDM) focuses on conceptual and system architecture issues related to the management of very large-scale data sets specifically in the context of the energy domain. The overall goal of the EmDM workshop is a) to bridge the gap between domain experts and data management scientists and b) to create awareness of this emerging and very challenging application area. For the workshop's research program, the organizers especially try to attract contributions that push the envelope towards novel schemes for large-scale data processing with special focus on energy data management.

The Third International Workshop on Energy Data Management (EnDM'14)[1] was held in conjunction with EDBT/ICDT 2014 in Athens, Greece, on March 28, 2014. This half-day event brought together researchers and engineers from academia and industry to discuss and exchange ideas related to energy data management and related topics. The workshop featured five research papers, and finished off with a panel/roundtable discussion. The accepted papers spanned a number of exciting topics within energy data management. Three papers concerned modeling of energy data: a pipeline production data model, a semantic web ontology for renewable energy sources, and an ontology for prosumer-oriented smart grids. Two papers were on energy analytics: one on extracting energy consumption profiles from smart meter data and one on a benchmark for renewable energy forecasting systems. The workshop proceedings have been published in a joint volume of all EDBT/ICDT 2014 workshops [1].

2. RESEARCH PAPERS

The first paper "Pipeline Production Data Model" by Jitao Yang, Yu Fan, Yinliang Liu, Hui Deng, and Yang Lin proposed a data model for pipeline production that could be used to support planning, scheduling, distribution, metering, energy consumption as well other busines processes within pipeline production. The model was specified in a function-free first-order predicate language. The model could be queries using a corresponding formula language. Two implementations of the model in a pipeline production system were discussed, one with Datalog queries on top of relational storage, and another based on RDF. Only the first had been realized so far: for this system, the functional, system, and software architectures were discussed, along with the hardware used for deployment. The system is running in production, and will be extended to in-

[1] http://endm2014.endm.org

tegrate data from a number of company systems.

The second paper "Renewable Energy Data Sources in the Semantic Web with OpenWatt" by Davide Lamanna and Antonio Maccioni proposed the OpenWatt ontology which is meant to solve a number of problems in the renewable energy sector including data that is only partially available, noisy and inconsistent data, sparse data in heterogeneous sources, unstructured data, and data represented through non-standard and proprietary formats, making the process of using the data ineffective and error-prone. OpenWatt is based on the Linked Open Data paradigm. It proposes a methodology with the steps of data gathering, data cleansing, data modeling, data recognition, ontology definition, data generation, metadata generation, external linking, data and ontology validation, and finally data publication. The OpenWatt ontology captures the types (consumption, production, potential) and categories (wind, biomass,..) of renewable energy sources, their geo-location, organized in a hiearchy, and a description of the associated measures (what quantity was measured, how, and the data source), and allows many kinds of data to be described and linked. The paper discussed the potential impact of OpenWatt and experiences from its implementation.

The third paper "A Generic Ontology for Prosumer-Oriented Smart Grid" by Syed Gillani, Frederique Laforest, and Gauthier Picard, presented a generic and layered ontology for complex prosumer-oriented smart grids. Prosumers are a new type of entity occuring in smart grid that can both produce and consume energy, which is a paradigm shift compared to traditional central energy production, and requires a detailed model of the context. The ontology enabled the integration and management in real-time of many distributed and heterogeneus sources, along with allowing the right granularity level for information. The paper discussed a number of use cases and modeling different types of physical infrastructures, electrical appliances, electrical generation, storage, weather, events, service contracts, and components. Rule-based inductive inference was supported on top of the ontology, with patterns for appliance consumption, alternative energy production, and producer performance.

The fourth paper "Computing Electricity Consumption Profiles from Household Smart Meter Data" by Omid Ardakanian, Negar Koochakzadeh, Rayman Preet Singh, Lukasz Golab, and S Keshav presented a profiling framework for residential consumers that take the variations in electricity consumption at different times of day and at different outside temperatures into account. Concretely, the effect of the outside temperature was isolated and a time-series autoregressive model used for the residual. The profiles can be used for personalized savings recommendations, outlier detection, as well as generaring realistic synthetic data. An experiment on a real-world data set of one thousand homes showed that the approach had better root-mean-squared prediction error than competing approaches.

The final paper "ECAST: A Benchmark Framework for Renewable Energy Forecasting Systems" by Robert Ulbricht, Ulrike Fischer, Lars Kegel, Wolfgang Lehner, and Hilko Donker discussed the need for evaluating and comparing the many new energy supply forecasting techniques developed in recent years. Although more and more data sets are available, it is still difficult and time-consuming to compare techniques with each other. The paper discusses the requirements for benchmarks of energy supply forecasting techniques, followed by presenting the ECAST benchmark framework that simplies the process by automating many tasks. The framework is demonstrated on a real-world scenario comparing different forecasting tools against a naive method. Finally, the paper points to a number of future developments of ECAST.

3. ROUNDTABLE/PANEL

The workshop finished off with a panel/roundtable discussion on "Energy Data Management: Where Are We Headed?" The workshop organizer first asked some questions: on what was already done within energy data management, and what is still missing, what the scientific challenges are, what the technical challenges are, and what challenges that necessitate an interdisciplinary approach, and provided his personal opinions on this.

A broad range of open benchmark datasets that can be used to develop robust and effective methods for various energy data management tasks, e.g., detailed device-level datasets for a larger number of households, is missing. Scientific challenges include a) the development of robust and effective methods and techniques for prediction of energy production and consumption down to the device level; b) the development of methods capable of extracting and predicting flexibilities in energy usage; c) the development of scalable etchniques for aggregating, scheduling, and disaggregating micro-level flexibilities, e.g., in individual device consumptions, to large-scale macro-level units suitable for balancing energy supply and demand at the higher levels; On the technical level, there is still a lack of community-wide agreed-upon common definitions of data and information concepts, e.g., standardized ontologies

specifying common concepts. Also, standardizing communication protocols, e.g., for available flexibilities, is very important. Interdisciplinary challenges are hard to meet, and include the interplay between computer scientists developing scalable techniques for energy data management, human-computer interaction designers exploring how and at which level of detail to interact with a smart grid system, and economists developing new business and energy taxation schemes.

The roundtable discussion added further perspectives. For data sets, production data is available but consumption data not yet, due to privacy problems. Data collection and generation can be bottom-up, using a home operating system controlling different appliances, but this is problematic for large appliances like central air conditioning. It can also be top-down, but this creates the problem of proper dis-aggregation. It is not feasible to put a sensor in every device, but it is possible to get a long way without dedicated HW. For privacy, consumers generally do not know how critical the data is, e.g., which movie is watched can be determined by analyzing the TV's energy consumption. A solution could be to build a virtual home, an energy data container in the cloud, where people can store their own data, and share it with whom they choose, compare with their neighbors, and give feedback. Getting utilities to release data is difficult, especially to get continued support after the initial data release. The French energy company EDF provides analysis to customers at the household level, but still not at the device level. Scientific challenges include model selection and predicting load of transformers. Technical challenges include that companies make their own proprietary ontologies, there is no standard or reuse. For interdisciplinary challenges, non-cash incentives and social networks are important. An example is the Ontario PeakSaver program, where the set point of the heating system can be changed, and the benefit is distribubed among the participants.

4. DISCUSSION AND OUTLOOK

Summing up, if we first look at the topics of the presented papers, we note that they address two main topics, modeling and analytics. In some sense, the three modeling papers all try to tackle the above-mentioned problem of a lack of standardization of common concepts, by proposing generic and extensible ontologies. This is an encouraging trend, which will however, have to be accompanied by a consolidation and integration phase leading up to a formal standardization. The papers on analytics also try to propose generic solutions to common problems, one to the problem of customer segmentation based on the consumption profiles, and the other by providing a common automated framework for benchmarking energy forecasting techniques against each other. Benchmarks are typically a sign that an area is reaching a certain level of maturity, which hopefully is the case for energy data management. Compared to the previous workshops, the topics were more focused. Several papers describe inter-disciplinary collaborations.

Next, when looking at the topics which occurred in the Call for Papers, but not within the accepted papers, we see that about half of the topics are covered in one way or another. Missing are more systems-oriented topics such as data processing architectures and schemes, query processing and optimization, and robustness aspects. We believe this is because energy data management is still new, and thus most systems are still in the development phase. While most papers are based on small case studies, only one paper described a system running in industrial production. We again attribute this to the fact that smart grids are still in development.

Summing up, we conclude that there is a lot of interesting work going on in the area of energy data management, with many remaining challenges to be met. This supports the need for venues that focus on this issue. The EnDM workshop series will continue at EDBT/ICDT 2015 in Bruxelles where the 4thInternational Workshop on Energy Data Management will be held on March 27, 2015[2].

5. ACKNOWLEDGEMENTS

The EnDM 2014 Chairs would like to thank all the authors of submitted papers for their support for the workshop and the high paper quality and the distinguished PC members for their careful and dedicated work during the reviewing and discussion phases. Finally, we would like to thank the Organizing Committee of EBDT/ICDT 2014, especially General Chair Vassilis Christophides, Workshop Chair Selcuk Candan, and Proceedings Chair Vincent Leroy. for their support.

6. REFERENCES

[1] K. S. Candan, S. Amer-Yahia, N. Schweikardt, V. Christophides, and V. Leroy, editors. *Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014), Athens, Greece, March 28, 2014*, volume 1133 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[2]http://endm2015.endm.org

Report on the International Workshop on Big Data Management on Emerging Hardware (HardBD 2015)

Shimin Chen
Institute of Computing Technology
Chinese Academy of Sciences
chensm@ict.ac.cn

Qiong Luo
Hong Kong University of
Science and Technology
luo@cse.ust.hk

Xiaofeng Meng
School of Information
Renmin University of China
mengxf@ruc.edu.cn

1. INTRODUCTION

HardBD 2015 (International Workshop on Big Data Management on Emerging Hardware) was held in Seoul, Korea on April 13, 2015, in conjunction with ICDE 2015 (the 31st IEEE International Conference on Data Engineering) [1]. The aim of this half-day workshop is to bring together researchers, practitioners, system administrators, and others interested in management of big data over new hardware platforms to share their perspectives, and to discuss and identify future directions and challenges in this area.

Data properties and hardware characteristics are two key aspects for efficient data management. A clear trend in the first aspect, data properties, is the increasing demand to manage and process *Big Data* in both enterprise and consumer applications, characterized by the fast evolution of *Big Data Systems*. Examples of big data systems include NoSQL storage systems, MapReduce/Hadoop, data analytics platforms, search and indexing platforms, messaging infrastructures, event log processing systems, as well as novel extensions to relational database systems. These systems address needs for processing structured, semi-structured, and unstructured data across a wide spectrum of domains such as web, social networks, enterprise, mobile computing, sensor networks, multimedia/streaming, cyber-physical and high performance systems, and for a great many application areas such as e-commerce, finance, health care, transportation, telecommunication, and scientific computing. At the same time, the second aspect, hardware characteristics, is undergoing rapid changes, imposing new challenges for the efficient utilization of hardware resources. Recent trends include massive multi-core processing systems, high performance co-processors, very large main memory, emerging non-volatile memory technology, fast networking components, big computing clusters, and large data centers that consume massive amount of energy. Utilizing new hardware technologies for efficient Big Data management is of urgent importance.

The program committee accepted four regular papers that cover a variety of interesting topics, by authors from China, Germany, Korea, and USA. The program also features a keynote speech by Sangyeun Cho, VP at Samsung for advanced solutions research and development, on recent advances in Flash solutions.

In the following, we will overview the keynote talk in Section 2 and the research papers in Section 3, and summarize the workshop in Section 4.

2. KEYNOTE TALK

The first session is keynote talk. Sangyeun Cho, our keynote speaker, became a tenured associate professor at the University of Pittsburgh in 2010. He recently joined Samsung's Memory Division as VP for advanced solutions research and development. His research interests are in the area of computer architecture and systems with particular focus on performance, power and reliability of memory and storage hierarchy design for next-generation data centers.

The speaker first described recent trends of Flash technology. Flash encounters significant challenges in scaling down its feature sizes. As density increases, cell interference goes up and the chance of data corruption increases. To address this problem (if temporarily), the memory industry has developed 3D vertical NAND flash (a.k.a. V-NAND). V-NAND stacks layers of flash cells on top of one another in order to increase flash capacity instead of shrinking feature sizes. SSDs with 32-layer V-NAND are already available on the market.

Then, the speaker discussed two topics that his team was working on. The first topic concerned multi-streaming SSDs, which allow upper-level software to specify multiple (currently up to 8) I/O streams. For each stream, a multi-streaming SSD will monitor the stream's I/O pattern and optimize Flash management tasks accordingly. Therefore, upper level software can use the streams to segregate data that have different properties. Experimental study with popular NoSQL engines have shown significant gains in terms of steady-state SSD performance and device lifetime. Multi-streaming has been accepted by the SSD industry standard. Products with the multi-streaming feature will soon appear on the

market. The second part of the talk focused on Intelligent SSDs, which support In-Storage Computing (ISC) and run user-written codes within the SSDs. ISC exploits the internal bandwidth of flash chips and the processing cycles within the SSD. Operations (e.g., filtering) that process a large amount of data but produce only a small result set can significantly benefit from ISC. The speaker presented the promising results of several case studies using database and data analytics workloads.

3. RESEARCH PAPERS

The second session featured four paper presentations:

(1) "Scalable and Efficient Spatial Data Management on Multi-Core CPU and GPU Clusters: A Preliminary Implementation based on Impala" by Simin You, Jianting Zhang, and Le Gruenwald. This paper integrates a spatial indexing and query processing engine with Cloudera Impala. The engine was previously developed by the authors. It supports both multi-core processors and GPUs. The integration effort modifies the Impala frontend to support spatial query syntax and implements a spatial data management module in impala that invokes the spatial query engine as a shared library. Compared to a traditional single-core technique, the resulting solution has achieved orders of magnitude of speedups on a high-end GPU-equipped workstation. In addition, the resulting solution has shown high efficiency and good scalability on a 10-node Amazon EC2 cluster equipped with multi-core CPUs and GPUs.

(2) "Optimizing CPU Cache Performance for Pregel-Like Graph Computation" by Songjie Niu and Shimin Chen. In-memory graph computation systems have been used to support many important applications, such as PageRank on the web graph and social network analysis. This paper studies the CPU cache performance of graph computation. To facilitate the study, the authors implemented a graph computation system, called GraphLite, in C/C++ based on the description of Pregel. The paper analyzes the CPU cache behavior of the internal data structures and operations of graph computation, then exploits CPU cache prefetching techniques to improve the cache performance. Preliminary experiments on a real machine show that the proposed technique can achieve about 2x speedups for PageRank computation.

(3) "Query Processing on Low-Energy Many-Core Processors" by Annett Ungethüm, Dirk Habich, Tomas Karnagel, Wolfgang Lehner, Nils Asmussen, Marcus Völp, Benedikt Nöthen and Gerhard Fettweis. This paper studies the techniques to implement query processing on Tomahawk, a low-energy heterogeneous multiprocessor system-on-a-chip. Tomahawk consists of an application core, a larger number of processing elements (PEs), and a core manager controller, all connected through a network-on-chip. The paper investigates two APIs of Tomahawk, a CUDA-like programming interface and a micro-kernel interface. Analysis and experimental evaluation show that the former has a simpler programming interface, while the latter allows programmers more controls to place different operators on different PEs, thereby avoiding unnecessary data transfers incurred by the former API.

(4) "Flash-aware Index Scan in PostgreSQL" by Da-som Hwang, Woon-hak Kang, and Sang-won Lee. This paper combines sorted index scan with parallel I/Os on SSDs to improve index scan performance. Sorted index scan is a technique to sort the record IDs from the range scan of the secondary index before retrieving the records. Parallel I/Os can take advantage of the internal parallelism of SSDs. The paper modifies the index scan implementation in PostgreSQL. Preliminary experimental results show that the optimization achieves dramatic improvements for index scans, and the optimized index scan can be faster than a full table scan even with 100% selectivity.

4. CONCLUSION

Over 30 people attended the half-day meeting. Both the keynote talk and the research paper presentations stimulated interesting questions and discussions. This shows significant interests of the ICDE community in the theme of HardBD: exploiting new hardware technologies for efficient Big Data management. From the discussions, it is clear that there are many open research problems, and both academia and industry have been actively working in this area.

5. ACKNOWLEDGMENT

Many people contributed to the successful organization of HardBD 2015. We would like to thank all the authors for their contributions, the PC members for excellent reviewing work, the keynote speaker for delivering an insightful speech, and the ICDE 2015 Workshops co-chairs for their guidance and support.

This workshop is partially supported by the CAS Hundred Talents program, State Key Laboratory of Computer Architecture (ICT, CAS), NSFC Innovation Research Group No. 61221062, and the National 863 High-tech Program (No. 2013AA013204).

6. REFERENCES

[1] J. Gehrke, W. Lehner, K. Shim, S. K. Cha, and G. M. Lohman, editors. *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*. IEEE, 2015.

Report on the 8th International Workshop on Business Intelligence for the Real-Time Enterprise (BIRTE'14)

Malu Castellanos
HP Labs, USA
malu.castellanos@hp.com

Umeshwar Dayal
Hitachi Labs, USA
umeshwar.dayal@hal.hitachi.com

Nesime Tatbul
Intel Labs and MIT, USA
tatbul@csail.mit.edu

Damianos Chatziantoniou *
AUEB, Greece
damianos@aueb.gr

Qiming Chen *
HP Labs, USA
qiming.chen@hp.com

1. OVERVIEW

The 8th International Workshop on Business Intelligence for the Real-Time Enterprise (BIRTE) was held on September 1, 2014 in conjunction with the VLDB 2014 Conference in Hangzhou, China. Like in previous years, the workshop was well attended by an engaged audience from both academia and industry, breaking an attendance record in the history of the BIRTE workshop series with more than 80 participants during the keynote session. In addition to the keynote speech, the workshop included two invited industrial talks, and four presentations of peer-reviewed papers covering a wide range of real-time BI topics with an overarching emphasis on big data analytics. The workshop developed as follows: After the official opening of the workshop, two papers were presented - one on BI analytics on graph-structured data and another on contextual analysis in temporal databases. The invited industrial talks session that followed included a talk from Microsoft Research about their data processing solutions for complex big data analytics and a talk from HP Labs about a new fault tolerance technique that they developed for distributed stream processing. The afternoon sessions opened with the highlight of the workshop: Dr. C. Mohan's keynote speech providing a critical survey of the current big data landscape. The workshop closed with a session that consisted of the two remaining papers - one on exploratory OLAP over linked data and another on data stream partitioning.

2. KEYNOTE SPEECH

This year's keynote speaker was Dr. C. Mohan from the IBM Almaden Research Center. Dr. Mohan has been a well-known pioneer in database systems and has made numerous contributions to relational database research and technology in various different roles at IBM for more than 30 years. In his talk "Big Data: Hype and Reality", Mohan delighted the audience with a concrete and detailed picture of the current landscape of big data systems. According to Mohan, as users and developers gain a deeper understanding of the needs of real use cases (including real-time BI applications), the initial hype around big data systems (including NoSQL, NewSQL, and others) has been fading away. It is now becoming clearer that most of the so-called distinctive features of big data systems have in fact been well-known principles of relational database systems for decades. Mohan's comprehensive and critical survey of this popular field attracted much attention from a big audience and was very well received.

3. INDUSTRIAL TALKS

Inviting industrial speakers to present their perspective on real-world BI problems, solutions, and applications has been a tradition of BIRTE since its inception in 2006. This year's workshop featured two industrial talks. First, in his talk entitled "Building Analytics Engines for the Big Data Age", Dr. Badrish Chandramouli of Microsoft Research presented the challenges of a temporal streaming engine called Trill. Trill has been architected as a library to support embedded execution within cloud applications and distributed fabrics. Second, Dr. Qiming Chen of HP Labs gave a talk about "Optimistic Failure Recovery in Distributed Stream Processing". More specifically, he presented the backtrack-based and the window-oriented recovery mechanisms implemented as part of the Fontainebleau distributed stream analytics system built on top of

*This author served as a session chair at the workshop.

the Storm platform. Both of these talks, covering industry-scale stream processing engines, intrigued an audience of around 50 participants.

4. PAPER PRESENTATIONS

The BIRTE'14 program committee selected a total of 4 paper submissions presented at the workshop in two research sessions, one being the opening and the other being the closing session of the workshop. The first of these sessions consisted of a position paper by co-authors from TU Dresden in Germany and the SAP Labs in USA, and a research paper by co-authors from University at Buffalo, SUNY and the Oracle Corporation in USA. First, Michael Rudolf presented a flexible approach for multi-dimensional graph data analysis in their paper entitled "SynopSys: Foundations for Multidimensional Graph Analytics". The key feature that distinguishes SynopSys from existing technologies, which require upfront modeling of analytical scenarios and are difficult to adapt to changes, is the ability to express ad-hoc analytical queries over graph data. The second paper, entitled "Detecting the Temporal Context of Queries" and presented by Ying Yang, focuses on the concept of contextual dependency - a term used by the authors to explain and attribute mistaken assumptions made by end users of BI applications. A formal definition for contextual dependence is given, followed by several strategies to efficiently detect and quantify the effects of contextual dependence on query outputs.

The second paper session consisted of an application paper jointly written by co-authors from Aalborg University in Denmark and Universite Libre de Bruxelles in Belgium, and a research paper from University of Southern Denmark. First, Dilshod Ibragimov explained during his talk, entitled "Towards Exploratory OLAP over Linked Open Data - A Case Study", how to integrate real-time data from web sources described in RDF into the analysis process in BI environments. To achieve this, a system that uses a multi-dimensional schema of the OLAP cube expressed in RDF vocabularies is proposed. Finally, the last presentation of the workshop was on "Efficient Pattern Detection over a Distributed Framework" by Ahmed Khan Leghari. In this talk, Leghari described an event stream partitioning scheme that partitions streams over time windows without considering any key attributes.

Though not as popular as the invited speaker sessions, there were 30+ attendees during both of these paper sessions.

5. CONCLUSIONS

Overall, we are proud to report that the 8th edition of the VLDB-colocated BIRTE workshop series has been a great success. We have once again witnessed that real-time business intelligence continues being a topic of great relevance for both database researchers and practitioners. Talks included a diverse set of real-world BI use cases, and indicated strong collaborations between academic and industrial community in this field. This year, we have seen that big data analytics has been a common theme for all BIRTE presentations, with a striking emphasis on the analysis of streaming and graph-structured data. BIRTE'14 presentation slides and a preliminary version of the corresponding papers can be found on our workshop webpage at http://db.csail.mit.edu/birte2014/. The final version of the papers (including papers from our invited speakers) have also been published as part of a joint BIRTE 2013-2014 post-proceedings in Springer LNBIP Volume 206.

6. ACKNOWLEDGEMENTS

We would like to thank all the authors of submitted papers for their interest in the workshop and their high-quality submissions, as well as the distinguished PC members for their dedicated work in reviewing those submissions. This workshop would not have been as successful as it was without the keynote given by Dr. C. Mohan, as well as the invited talks given by Dr. Badrish Chandramouli and Dr. Qiming Chen, to whom we are deeply grateful. We would also like to thank the VLDB 2014 organizers for supporting BIRTE, especially the general chairs (Chun Chen and Sharad Mehrotra), the workshop chairs (Anastasia Ailamaki and Kaushik Chakrabarti), and the local organization chair (Lidan Shou). Finally, we would like to acknowledge our proceedings chair Jennie Duggan, who did an excellent job in putting together the workshop post-proceedings.

32nd IEEE International Conference on

Data Engineering

May 16-20, 2016 · Helsinki, Finland

http://icde2016.fi/

Call for Papers

Conference Organization

General Chairs
Boris Novikov (Saint Petersburg University, Russia)
Eljas Soisalon-Soininen (Aalto University, Finland)

Program Committee Chairs
Mei Hsu (HP Labs, USA)
Alfons Kemper (TU Munich, Germany)
Timos Sellis (RMIT University, Australia)

Program Committee Chair for Industrial and Applications Papers
C. Mohan (IBM Almaden, USA)

Program Committee Chair for Demos
Stefan Manegold (CWI, the Netherlands)

TKDE Posters Chair
Xuemin Lin (University of New South Wales, Australia)

Panel Chair
Divesh Srivastava (AT&T Labs-Research, USA)

Workshop and Tutorial Chairs
Giovanna Guerrini (University of Genova, Italy)
Georgia Koutrika (HP Labs, USA)

Organizing Committee Chair
Sami El-Mahgary (Aalto University, Finland)

Publicity Chair
Antoni Wolski (AWO Consulting, Finland)

Web Chair
Anna Yarygina (Saint Petersburg University, Russia)

The annual ICDE conference addresses research issues in designing, building, managing, and evaluating advanced data systems and applications. It is a leading forum for researchers, practitioners, developers, and users to explore cutting-edge ideas and to exchange techniques, tools, and experiences. We invite submissions for research papers, industrial and applications papers, demonstrations, tutorials, and workshops.
The conference proceedings are published by IEEE Computer Society.

Topics of Interest

- Cloud Computing and Database-as-a-Service
- Big Data and Data-Warehousing System Architectures
- Data Integration, Metadata Management, and Interoperability
- Modern Hardware and In-Memory Database Architecture and Systems
- Privacy, Security, and Trust
- Query Processing, Indexing, and Optimization
- Social Networks, Social Web, Graph, and Personal Information Management
- Crowdsourcing, Distributed, and P2P Data Management
- Streams and Sensor Networks
- High Performance Transaction Management
- Temporal, Spatial, Mobile, and Multimedia Data
- Strings, Texts, and Keyword Search
- Uncertain and Probabilistic Data
- Visual Data Analytics, Data Mining, and Knowledge Discovery

Important Dates

Submission due:	Oct. 19, 2015
Author's feedback (research papers only):	Dec. 1-4, 2015
Notification:	Dec. 20, 2015
Camera-ready copy due:	Jan. 25, 2016
Tutorial proposals due:	Nov. 15, 2015
Tutorial notification:	Dec. 20, 2015
Workshop proposals due:	Sep. 10, 2015
Workshop notification:	Oct. 10, 2015

Aalto University
School of Science

IEEE computer society