SIGMOD Officers, Committees, and Awardees

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

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

Secretary/Treasurer
Fatma Ozcan
Google
Sunnyvale
California
USA
fozcan <at> google.com

SIGMOD Executive Committee:
Juliana Freire (Chair), Ihab Francis Ilyas (Vice-Chair), Fatma Ozcan (Treasurer), K. Selçuk Candan, Rada Chirkova, Chris Jermaine, Wang-Chiew Tan, AnHai Doan, Leonid Libkin, and Curtis Dyreson

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

SIGMOD Information Director:
Curtis Dyreson, Utah State University

Associate Information Directors:
Huiping Cao, Georgia Koutrika, Wim Martens, and Sourav S Bhowmick

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

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

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

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

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

SIGMOD Awards Committee:
M. Tamer Özsu (Chair), Stefano Ceri, Yanlei Diao, Volker Markl, Renee Miller, and Sunita Sarawagi

Jim Gray Doctoral Dissertation Award Committee:
Pınar Tözün (co-Chair), Viktor Leis (co-Chair), Peter Bailis, Alexandra Meliou, Bailu Ding, Vanessa Braganholo, Immanuel Trummer, and Joy Arulraj

SIGMOD Edgar F. Codd Innovations Award

For innovative and highly significant contributions of enduring value to the development, understanding, or use of database systems and databases. Recipients of the award are the following:

Michael Stonebraker (1992)
David DeWitt (1995)
Serge Abiteboul (1998)
Rudolf Bayer (2001)
Ronald Fagin (2004)
Jennifer Widom (2007)
Umeshwar Dayal (2010)
Stefano Ceri (2013)
Gerhard Weikum (2016)
Anastasia Ailamaki (2019)
Jim Gray (1993)
C. Mohan (1996)
Hector Garcia-Molina (1999)
Patricia Selinger (2002)
Michael Carey (2005)
Moshe Y. Vardi (2008)
Surajit Chaudhuri (2011)
Martin Kersten (2014)
Goetz Graefe (2017)
Beng Chin Ooi (2020)
Philip Bernstein (1994)
David Maier (1997)
Rakesh Agrawal (2000)
Don Chamberlin (2003)
Jeffrey D. Ullman (2006)
Masaru Kitsuregawa (2009)
Bruce Lindsay (2012)
Laura Haas (2015)
Raghu Ramakrishnan (2018)

SIGMOD Systems Award

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

Michael Stonebraker and Lawrence Rowe (2015); Martin Kersten (2016); Richard Hipp (2017); Jeff Hammerbacher, Ashish Thusoo, Joydeep Sen Sarma; Christopher Olston, Benjamin Reed, and Utkarsh Srivastava (2018); Xiaofeng Bao, Charlie Bell, Murali Brahmadesam, James Corey, Neal Fachan, Raju Gulabani, Anurag Gupta, Kamal Gupta, James Hamilton, Andy Jassy, Tengiz Kharatishvili, Sailesh Krishnamurthy, Yan Leshinsky, Lon Lundgren, Pradeep Madhavarapu, Sandor Maurice, Grant McAlister, Sam McKelvie, Raman Mittal, Debanjan Saha, Swami Sivasubramanian, Stefano Stefani, and Alex Verbitski (2019); Don Anderson, Keith Bostic, Alan Bram, Grg Burd, Michael Cahill, Ron Cohen, Alex Gorrod, George Feinberg, Mark Hayes, Charles Lamb, Linda Lee, Susan LoVerso, John Merrells, Mike Olson, Carol Sandstrom, Steve Sarette, David Schacter, David Segleau, Mario Seltzer, and Mike Ubell (2020)

SIGMOD Contributions Award

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

Maria Zemankova (1992)
Jeffrey Ullman (1996)
Raghu Ramakrishnan (1999)
Daniel Rosenkrantz (2001)
Surajit Chaudhuri (2004)
Hans-Jörg Schek (2007)
David Lomet (2010)
H.V. Jagadish (2013)
Samuel Madden (2016)
Ahmed Elmagarmid (2019)
Stratos Idreos (2020)
Dennis Shasha (2020)
Gio Wiederhold (1995)
Avi Silberschatz (1997)
Michael Carey (2000)
Richard Snodgrass (2002)
Hongjun Lu (2005)
Klaus R. Dittrich (2008)
Gerhard Weikum (2011)
Kyu-Young Whang (2014)
Yannis E. Ioannidis (2017)
Philipe Bonnet (2020)
Stefan Manegold (2020)
Yahiko Kambayashi (1995)
Won Kim (1998)
Laura Haas (2000)
Michael Ley (2003)
Tamer Özsu (2006)
Beng Chin Ooi (2009)
Marianne Winslett (2012)
Curtis Dyreson (2015)
Z. Meral Özsoyoğlu (2018)
Juliana Freire (2020)
Ioana Manolescu (2020)

SIGMOD Jim Gray Doctoral Dissertation Award

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

- **2006** *Winner*: Gerome Miklau. *Honorable Mentions*: Marcelo Arenas and Yanlei Diao
- **2007** *Winner*: Boon Thau Loo. *Honorable Mentions*: Xifeng Yan and Martin Theobald
- **2008** *Winner*: Ariel Fuxman. *Honorable Mentions*: Cong Yu and Nilesh Dalvi
- **2009** *Winner*: Daniel Abadi. *Honorable Mentions*: Bee-Chung Chen and Ashwin Machanavajjhala
- **2010** *Winner:* Christopher Ré. *Honorable Mentions*: Soumyadeb Mitra and Fabian Suchanek
- **2011** *Winner*: Stratos Idreos. *Honorable Mentions*: Todd Green and Karl Schnaitterz

- **2012** *Winner*: Ryan Johnson. *Honorable Mention*: Bogdan Alexe
- **2013** *Winner*: Sudipto Das, *Honorable Mention*: Herodotos Herodotou and Wenchao Zhou
- **2014** *Winners*: Aditya Parameswaran and Andy Pavlo.
- **2015** *Winner*: Alexander Thomson. *Honorable Mentions*: Marina Drosou and Karthik Ramachandra
- **2016** *Winner*: Paris Koutris. *Honorable Mentions*: Pinar Tozun and Alvin Cheung
- **2017** *Winner*: Peter Bailis. *Honorable Mention*: Immanuel Trummer
- **2018** *Winner*: Viktor Leis. *Honorable Mention*: Luis Galárraga and Yongjoo Park
- **2019** *Winner*: Joy Arulraj. *Honorable Mention*: Bas Ketsman
- **2020** *Winner:* Jose Faleiro. *Honorable Mention:* Silu Huang

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

[Last updated: September 30, 2020]

Editor's Notes

Welcome to the September 2020 issue of the ACM SIGMOD Record!

This issue starts with the Database Principles column featuring an article by Hu and Yi. The authors survey recent algorithms for join evaluation under the Massively Parallel Computation (MPC) model exemplified by the MapReduce and Spark frameworks. The article provides a formal perspective on processing join-aggregate queries over annotated relations; this formalization covers many query types, including standard aggregate queries and conjunctive queries. The authors study the data complexity of algorithms for multi-round processing of joins under the MPC model, conveying crisp and practically applicable characterizations of the algorithms by cases. The article also provides lower-bound results, whose importance is in ruling out the possibility of certain types of join algorithms for certain query classes. The formal results discussed in the article are summarized in two detailed tables, and the exposition contains illustrations featuring specific join examples. The article also presents open problems and provides a discussion of related and further work.

The Surveys column features an article by Hameed and Naumann that studies the state of the art in data-preparation tools, including the accompanying research and development opportunities. Here, data preparation is understood as a collection of processes for preparing raw data for ingestion into downstream applications, such as data-analytics tools and data-management systems. The article provides a thorough classification of data-preparation features and commercial tools, a wealth of examples, and extensive documentation and evaluation results of the features on three data sets obtained from public data repositories. The recommendations include an articulation of the features and abilities that are still lacking in the state-of-the-art tools, as well as a discussion of challenging research problems in the emerging area of data preparation.

The Distinguished Profiles column features Goetz Graefe, recipient of the SIGMOD Innovations Award, the SIGMOD Test of Time Award, the ICDE Distinguished Paper Award, and the ACM Software System Award, all for his work on query processing. Goetz's Ph.D. is from the University of Wisconsin-Madison. He was an HP Fellow, and currently works for Google. In this interview, Goetz talks about the research topics to which he has made contributions over the years, and about how traditional results can be extended to apply in new contexts. He shares his thoughts about open problems in query optimization, outlining ideas for sources of potential solutions for lack of robustness in query performance. Goetz talks about the success of the Cascades framework, as well as about how he would build a query optimizer today from scratch. He provides insights on a range of other topics, including thoughts on ways in which academia, industry, and industry research labs do research, as well as on his teaching at Dagstuhl. Goetz shares ideas on promising long-term hard systems problems, discusses what he would change about himself as a computer-science researcher, and provides (surprising) advice for fledgling or midcareer database researchers.

The Industry Perspectives column features an article by Jindal about lessons that he has learned from the CloudViews project at the Gray Systems Lab at Microsoft. While using his experience from the CloudViews project as a detailed running example, the author provides excellent advice that can be readily used by any new researchers to set themselves up for success in a product group. The article walks the reader through all the steps of the process, from initial selection of a promising product and of the right problem to work on, all the way to publishing and to then moving on. The author discusses common challenges and pitfalls, and shares excellent advice, including tips on building collaborations, on success metrics, and on publishing. Much of the advice in this engaging

and enjoyable article would also be of interest to beginner researches in general, both in graduate school and in the workplace, including in academia.

This issue features two reports. First, it is our pleasure to include a detailed report on running the SIGMOD/PODS conference in June 2020 in the online-only mode due to the COVID-19 situation. Preparing and running the conference was challenging, as it had originally been planned to take place as a "live" event in Portland, Oregon, with many months of work already invested into realizing that plan. The organizers of the conference detail in the article the sequence of the events and decisions in the process of rearranging the conference into the fully remote-access mode. (At some point, there was even discussion of canceling the conference altogether.) The article presents experiences of the conference organizers, information gathered in the process, quotes from participants, and recommendations on running analogous events. The authors also outline and discuss the most-requested features that ended up not being supported by SIGMOD/PODS 2020, and provide statistics from the post-conference survey and from logs of Zoom sessions during the conference. Lessons from the conference would be of interest to those who organize online-only conferences or conferences with an online component. The second article in the Reports column, by Stonebraker, Mattson, Kraska, and Gadepally, reports on the outcomes of the fourth annual workshop on Polystores (POLY'19), which took place in Los Angeles, CA USA in August 2019 in conjunction with the VLDB conference. The article focuses on the theme of the workshop that explored the implications of data-privacy regulations such as General Data Privacy Regulations (GDPR) in the context of heterogeneous data and data-management systems. The authors outline some major approaches and directions presented during the data-privacy portion of the POLY'19 workshop, specifically in the scope of the GDPR right to be forgotten and of support for restrictions on data usage known as "purposes." The authors also outline the unaddressed issues and emerging research directions. The workshop materials are available online; in addition, the authors point the readers to the POLY'20 workshop, which also featured data privacy in its program.

On behalf of the SIGMOD Record Editorial board, I hope that you enjoy reading the September 2020 issue of the SIGMOD Record!

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

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

Rada Chirkova

September 2020

Past SIGMOD Record Editors:

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

Massively Parallel Join Algorithms

Xiao Hu
Duke University
xh102@cs.duke.edu

Ke Yi
HKUST
yike@cse.ust.hk

ABSTRACT

Due to the rapid development of massively parallel data processing systems such as MapReduce and Spark, there have been revived interests in designing algorithms in a massively parallel computational model. Computing multi-way joins, as one of the central algorithmic problems in databases, has received much attention recently. This paper surveys some of the recent algorithms, as well as lower bounds. We focus on multi-round algorithms, while referring readers to [27] for single-round algorithms.

1. INTRODUCTION

1.1 Join and Join-aggregate Queries

A (natural) *join* is represented as a hypergraph $\mathcal{Q} = (\mathcal{V}, \mathcal{E})$, where the vertices $\mathcal{V} = \{x_1, \ldots, x_n\}$ model the *attributes* and the hyperedges $\mathcal{E} = \{e_1, \ldots, e_m\} \subseteq 2^{\mathcal{V}}$ model the *relations*. Let $\mathrm{dom}(x)$ be the *domain* of attribute $x \in \mathcal{V}$. An *instance* of $\mathcal{Q}$ is a set of relations $\mathcal{R} = \{R(e) : e \in \mathcal{E}\}$, where $R(e)$ is a set of *tuples* on attributes e. We use $\mathrm{IN} = \sum_{e\in\mathcal{E}} |R(e)|$ to denote the size of $\mathcal{R}$. The *join result* of $\mathcal{Q}$ on $\mathcal{R}$ is

$$\mathcal{Q}(\mathcal{R}) = \{t \mid \pi_e t \in R(e), \forall e \in \mathcal{E}\}.$$

Let $\mathrm{OUT} = |\mathcal{Q}(\mathcal{R})|$ be the output size. We study the *data complexity* of algorithms, namely, the query size (i.e., $n + m$) is treated as a constant.

We consider join-aggregate queries over *annotated relations* [17, 23, 25] with one semiring. Let $(\mathbb{R}, \oplus, \otimes)$ be a commutative semiring. We assume that every tuple t is associated with an *annotation* $w(t) \in \mathbb{R}$. The annotation of a join result $t \in \mathcal{Q}(\mathcal{R})$ is

$$w(t) := \otimes_{t_e\in R(e), \pi_e t = t_e, e\in\mathcal{E}} w(t_e).$$

Let $\mathbf{y} \subseteq \mathcal{V}$ be a set of *output attributes* (a.k.a. *free variables*) and $\bar{\mathbf{y}} = \mathcal{V} - \mathbf{y}$ the non-output attributes (a.k.a. *bound variables*). A *join-aggregate query* $\mathcal{Q}_{\mathbf{y}}(\mathcal{R})$ asks us to compute $\oplus_{\bar{\mathbf{y}}}\mathcal{Q}(\mathcal{R}) =$

$$\left\{(t_{\mathbf{y}}, w(t_{\mathbf{y}})) : t_{\mathbf{y}} \in \pi_{\mathbf{y}}\mathcal{Q}(\mathcal{R}), w(t_{\mathbf{y}}) = \oplus_{t\in\mathcal{Q}(\mathcal{R}):\pi_{\mathbf{y}} t = t_{\mathbf{y}}} w(t)\right\}.$$

In plain language, a join-aggregate query first computes the join $\mathcal{Q}(\mathcal{R})$ and the annotation of each join result, which is the $\otimes$-aggregate of the tuples comprising the join result. Then it partitions $\mathcal{Q}(\mathcal{R})$ into groups by their projection on $\mathbf{y}$. Finally, for each group, it computes the $\oplus$-aggregate of the annotations of the join results.

Many queries can be formulated as special join-aggregate queries. For example, if we take $\mathbb{R}$ to be the domain of integers, $\oplus$ to be addition, $\otimes$ to be multiplication, and set $w(t) = 1$ for all t, then it becomes the `COUNT(*) GROUP BY` $\mathbf{y}$ query; in particular, if $\mathbf{y} = \emptyset$, the query computes $|\mathcal{Q}(\mathcal{R})|$. The join-project query $\pi_{\mathbf{y}}\mathcal{Q}(\mathcal{R})$, also known as a *conjunctive query*, is a special join-aggregate query by discarding the annotations.

1.2 Model of Computation

We adopt the *Massively Parallel Computation* (MPC) model [28, 8, 9, 27]. In the MPC model, there are p servers connected by a complete communication network. Data are initially distributed across p servers with each server holding IN/p tuples. Computation proceeds in *rounds* or *super steps*. In each round, each server first receives messages from other servers (if there are any), performs some local computation, and then sends messages to other servers. The complexity of the algorithm is measured by the number of rounds, and the load, denoted as L, which is the maximum message size received by any server in any round. We will only consider constant-round algorithms. In this case, whether a server is allowed to keep messages it has received from previous rounds is irrelevant: if not, it can just keep sending all these messages to itself over the rounds, increasing the load by a constant factor.

The MPC model can be considered as a simplified version of the *bulk synchronous parallel* (BSP) model [32], but it has enjoyed more popularity in recent years. This is mostly because the BSP model takes too many measures into consideration, such as communication costs, local computation time, memory consumption, etc. The MPC model unifies all these costs with one parameter L, which makes the model much simpler. Meanwhile, although L is defined as the maximum incoming message size of a server, it is also closely related to the local computation time and memory consumption, which are both increasing functions of L. Thus, L serves as a good surrogate of these other cost measures. This is also why the MPC model does not limit the outgoing message size of a server, which is less relevant to other costs.

We will adopt the mild assumption $\mathrm{IN} \geq p^{1+\epsilon}$ where $\epsilon > 0$ is any small constant. This assumption clearly holds on any reasonable values of IN and p in practice;

theoretically, this is the minimum requirement for the model to be able to compute some almost trivial functions, like the "or" of N bits, in $O(1)$ rounds [15]. When $\text{IN} \geq p^{1+\epsilon}$, many basic operations (see Section 2) can be performed in $O(1)$ rounds with $O(\text{IN}/p)$ load, which is often called "linear load", as it is the load needed to shuffle all input data once.

When proving lower bounds, we confine ourselves to *tuple-based* join algorithms, i.e., the tuples are atomic elements that must be processed and communicated in their entirety. The only way to create a tuple is by making a copy, from either the original tuple or one of its copies. We say that an MPC algorithm computes the join $\mathcal{Q}$ on instance $\mathcal{R}$ if the following is achieved: For any join result $t \in \mathcal{Q}(\mathcal{R})$, the tuples (or their copies) t_e such that $t_e \in R(e), \pi_e \in R(e)$ for all $e \in \mathcal{E}$ must be present on the same server at some point. Then the server will emit the join result. Recall that we allow a server to keep all messages it has received, so this requirement is equivalent to requiring that the tuples t_e all arrive at some server. For join-aggregate queries, we assume that the only way for a server to create new semiring elements is by multiplying and adding existing semiring elements currently residing on the same server.

1.3 Classification of Join Queries

Various classes of join queries have been studied in the literature. The relationships of join queries to be mentioned are illustrated in Figure 2.

Acyclic joins [10]. A join $\mathcal{Q} = (\mathcal{V}, \mathcal{E})$ is *acyclic* (a.k.a. α-acyclic) if there exists a tree $\mathcal{T}$ whose nodes are in one-to-one correspondence with the hyperedges in $\mathcal{E}$, such that for any vertex $v \in \mathcal{V}$, all nodes containing v are connected in $\mathcal{T}$. Such a tree $\mathcal{T}$ is called the *join tree* of $\mathcal{Q}$. Note that the join tree may not be unique for a given $\mathcal{Q}$.

Hierarchical joins [14]. A join $\mathcal{Q} = (\mathcal{V}, \mathcal{E})$ is *hierarchical* if for every pair of vertices x, y, there is $\mathcal{E}_x \subseteq \mathcal{E}_y$, or $\mathcal{E}_y \subseteq \mathcal{E}_x$, or $\mathcal{E}_x \cap \mathcal{E}_y = \emptyset$, where $\mathcal{E}_x = \{e \in \mathcal{E} : x \in e\}$ is the set of hyperedges containing attribute x. This is equivalent to the condition that all attributes can be organized into a forest, such that x is a descendant of y iff $\mathcal{E}_x \subseteq \mathcal{E}_y$.

r-hierarchical joins [20]. We consider a slightly larger class of hierarchical joins. A *reduce* procedure on a hypergraph $(\mathcal{V}, \mathcal{E})$ is to remove an edge $e \in \mathcal{E}$ if there exists another edge $e' \in \mathcal{E}$ such that $e \subseteq e'$. We can repeatedly apply the reduce procedure until no more edge can be reduced, and the resulting hypergraph is said to be *reduced*. A join is *r-hierarchical* if its reduced join hypergraph is hierarchical. A hierarchical join must be r-hierarchical, but not vice versa. For example, the join $R_1(A) \bowtie R_2(A, B) \bowtie R_3(B)$ is r-hierarchical but not hierarchical. On the other hand, an r-hierarchical join must be acyclic.

Note that the reduction procedure can be done in linear load using semijoins (see Section 2).

Tall-flat joins [28]. A join $\mathcal{Q} = (\mathcal{V}, \mathcal{E})$ is *tall-flat* if one can order the attributes as $x_1, x_2, \cdots, x_h, y_1, y_2, \cdots, y_l$ such that (1) $\mathcal{E}_{x_1} \supseteq \mathcal{E}_{x_2} \supseteq \cdots \supseteq \mathcal{E}_{x_h}$; (2) $\mathcal{E}_{x_h} \supseteq \mathcal{E}_{y_j}$ for $j = 1, 2, \cdots, l$; and (3) $|\mathcal{E}_{y_j}| = 1$ for $j = 1, 2, \cdots, l$. Obviously, a tall-flat join is hierarchical. For a tall-flat join, this attribute forest takes the form of a special tree, which consists of a single "stem" plus a number of leaves at the bottom. Consider two examples: $\mathcal{Q}_1 = R_1(x_1) \bowtie R_2(x_1, x_2) \bowtie R_3(x_1, x_2, x_3, x_4) \bowtie R_4(x_1, x_2, x_3, x_5) \bowtie R_5(x_1, x_2, x_3, x_6)$ is a tall-flat join, and $\mathcal{Q}_2 = R_1(x_1, x_2) \bowtie R_2(x_1, x_3, x_4) \bowtie R_3(x_1, x_3, x_5)$ is a hierarchical join (but not tall-flat). Their attribute forests (actually, trees for these cases) are shown in Figure 1.

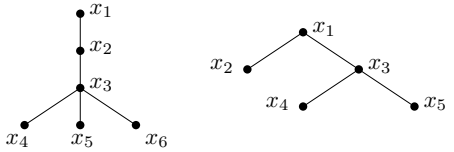

Figure 1: Tall-flat and hierarchical join.

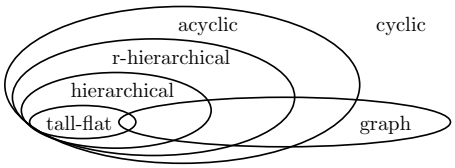

Figure 2: Relationships of joins.

Graph joins [24, 31]. A join is a *graph join* if every relation contains at most two attributes. If each relation contains exactly two attributes, the hypergraph becomes an ordinary graph. If a graph join is also acyclic, it is a *tree join*, i.e., the hypergraph is a tree.

Free-connex join-aggregate queries [6]. With respect to join-aggregate queries, *free-connex* queries are an important subclass. To define a free-connex query, we introduce the notion of a *width-1 GHD*, which can be considered as a generalized join tree. A *width-1 GHD* of a hypergraph $\mathcal{Q} = (\mathcal{V}, \mathcal{E})$ is a tree $\mathcal{T}$, where each node $u \in \mathcal{T}$ is a subset of $\mathcal{V}$, such that (1) for each attribute $x \in \mathcal{V}$, the nodes containing x are connected in $\mathcal{T}$; (2) for each hyperedge $e \in \mathcal{E}$, there exists a node $u \in \mathcal{T}$ such that $e \subseteq u$; and (3) for each node $u \in \mathcal{T}$, there exists a hyperedge $e \in \mathcal{E}$ such that $u \subseteq e$.

Given a set of output attributes $\mathbf{y}$, $\mathcal{T}$ is free-connex if there is a subset of connected nodes of $\mathcal{T}$, denoted as $\mathcal{T}'$ (such a $\mathcal{T}'$ is said to be a connex subset), such that $\mathbf{y} = \bigcup_{u \in \mathcal{T}'} u$. A join-aggregate query $\mathcal{Q}_{\mathbf{y}}$ is free-connex if it has a free-connex width-1 GHD. For example, the join-project query $\pi_A R_1(A, B) \bowtie R_2(B, C)$ is free-connex while $\sum_B R_1(A, B) \bowtie R_2(B, C)$ is not.

1.4 Complexity Measures

In worst-case analysis, the entire space of instances is divided into classes by the input size IN, and the running time is measured on the worst instance in each class. Let $\mathfrak{R}(\text{IN})$ be the class of instances with input size IN. The load of an MPC Algorithm $\mathbb{A}$ is a function

of IN, defined as

$$L_{\mathcal{A}}(\text{IN}) = \max_{\mathcal{R} \in \mathfrak{R}(\text{IN})} L_{\mathcal{A}}(\mathcal{R}),$$

where Algorithm $\mathcal{A}$ is *worst-case optimal* if

$$L_{\mathcal{A}}(\text{IN}) = O(L_{\mathcal{A}'}(\text{IN})),$$

for every algorithm $\mathcal{A}'$.

A more refined approach is *parameterized analysis*, which further subdivides the instance space into smaller classes by introducing more parameters that can better characterize the difficulty of each class. For the join problem, the output size OUT is a commonly used parameter, and each class of instances share the same input and output size. Let $\mathfrak{R}(\text{IN}, \text{OUT})$ be the class of instances with input size IN and output size OUT. Then the load of an MPC algorithm $\mathcal{A}$ is thus a function of both IN and OUT, defined as

$$L_{\mathcal{A}}(\text{IN}, \text{OUT}) = \max_{\mathcal{R} \in \mathfrak{R}(\text{IN}, \text{OUT})} L_{\mathcal{A}}(\mathcal{R}),$$

Algorithm $\mathcal{A}$ is *output-optimal* if

$$L_{\mathcal{A}}(\text{IN}, \text{OUT}) = O(L_{\mathcal{A}'}(\text{IN}, \text{OUT})),$$

for every algorithm $\mathcal{A}'$.

Further subdividing the instance space leads to more refined analyses. In the extreme case when each class contains just one instance, we obtain instance-optimal algorithms. Algorithm $\mathcal{A}$ is *instance-optimal* if

$$L_{\mathcal{A}}(\mathcal{R}) = O(L_{\mathcal{A}'}(\mathcal{R})),$$

for every instance $\mathcal{R}$ and every algorithm $\mathcal{A}'$.

By definition, an instance-optimal algorithm must be output-optimal, and an output-optimal algorithm must be worst-case optimal, but the reverse direction may not be true.

In the RAM model, the Yannakakis algorithm [34] can compute any acyclic join in time $O(\text{IN} + \text{OUT})$, which is both output-optimal and instance-optimal, because on any instance $\mathcal{R}$, any algorithm has to spend at least $\Omega(\text{IN})$ time to read inputs and $\Omega(\text{OUT})$ time to enumerate the outputs. Thus, the two notions of optimality coincide (but both are stronger than worst-case optimality). Fundamentally, this is because the difficulty of any instance $\mathcal{R}$ is precisely characterized by its input size and output size, and all instances in $\mathfrak{R}(\text{IN}, \text{OUT})$ have exactly the same complexity $O(\text{IN} + \text{OUT})$.

1.5 Overview of Results

Earlier efforts have been devoted to one-round MPC algorithms; please see [27] for an excellent survey. However, for many queries, there can be a polynomial difference between one-round and multi-round algorithms. For example, the optimal load for the triangle join $\mathcal{Q}_\triangle = R_1(B,C), R_2(A,C), R_3(A,B)$ is $O(\frac{\text{IN}}{p^{1/2}})$, while a two-round algorithm can achieve $O(\frac{\text{IN}}{p^{2/3}})$ [26].

In this paper, we focus on multi-round (but still a constant) algorithms in the MPC model. We give a brief overview of results below, while describing some selective algorithms in more detail in later sections. Tables 1 and 2 provide a summary of the results.

Instance-optimal join algorithms. We start from computing the Cartesian product of m sets of sizes $N_1, \ldots, N_m$. Since the output size is $\prod_{i=1}^m N_i$ and each server can emit at most L^m results if the load is L, an immediate lower bound on L is $(\frac{\prod_{i=1}^m N_i}{p})^{\frac{1}{m}}$. In addition, any algorithm computing the full Cartesian product must also implicitly compute the Cartesian product of any subset of the m sets. Applying the same argument for each subset S, we obtain a lower bound of

$$L_{\text{Cartesian}}(p, \mathcal{R}) := \max_{S \subseteq \{1,\ldots,m\}} \left(\frac{\prod_{i \in S} N_i}{p} \right)^{\frac{1}{|S|}}. \quad (1)$$

It has been shown that the HyperCube algorithm [2] incurs a load of $L_{\text{Cartesian}}(p, \mathcal{R}) \cdot \log^{O(1)} p$ on any instance $\mathcal{R}$ [8]. Thus, it can be considered as an instance-optimal algorithm for computing Cartesian products, with an optimality ratio of $\log^{O(1)} p$.

We can extend the Cartesian product lower bound to a join query $\mathcal{Q} = (\mathcal{V}, \mathcal{E})$. For any subset of relations $S \subseteq \mathcal{E}$, define

$$\mathcal{Q}(\mathcal{R}, S) := (\bowtie_{e \in S} R(e)) \ltimes \mathcal{Q}(\mathcal{R}),$$

i.e., the join results of relations in S that are part of a full join result. Clearly, any algorithm computing $\mathcal{Q}(\mathcal{R})$ must implicitly also compute $\mathcal{Q}(\mathcal{R}, S)$ for every S. Because each join result in $\mathcal{Q}(\mathcal{R}, S)$ consists of $|S|$ tuples, one from each relation in S, a server can emit at most $O(L^{|S|})$ join results of $\mathcal{Q}(\mathcal{R}, S)$, so we must have $p \cdot L^{|S|} = \Omega(|\mathcal{Q}(\mathcal{R}, S)|)$. Thus, we obtain the following per-instance lower bound on the load:

$$L_{\text{instance}}(p, \mathcal{R}) := \max_{S \subseteq \mathcal{E}} \left(\frac{|\mathcal{Q}(\mathcal{R}, S)|}{p} \right)^{\frac{1}{|S|}}. \quad (2)$$

It has been shown that r-hierarchical queries are precisely the class of queries that admit instance-optimal algorithms [8, 20]. More precisely, there is an algorithm with load $O(\frac{\text{IN}}{p} + L_{\text{instance}}(p, \mathcal{R}))$ for computing any r-hierarchical query, while for every acyclic join that is not r-hierarchical, there is an instance $\mathcal{R}$ with $L_{\text{instance}}(p, \mathcal{R}) = O(\frac{\text{IN}}{p})$ but any multi-round algorithm must incur a load of $\tilde{\Omega}(\frac{\text{IN}}{p^{1/2}})$ on $\mathcal{R}$. Section 3 gives more details on these results.

Output-sensitive algorithms. By plugging the two-way join algorithm [8, 22] into the classical Yannakakis algorithm [34], one can compute any acyclic join with load $O(\frac{\text{IN}}{p} + \frac{\text{OUT}}{p})$ [1], but this is not output-optimal. As mentioned, an instance-optimal algorithm must also be output-optimal, so we have automatically obtained output-optimal algorithms for r-hierarchical joins. In fact, it has been shown that $L_{\text{instance}}(p, \mathcal{R}) = O(\frac{\text{IN}}{p} +$

[2]For a join query $\mathcal{Q} = (\mathcal{V}, \mathcal{E})$, the edge quasi-packing number is defined as follows. Let $x \subseteq \mathcal{V}$ be any subset of vertices of $\mathcal{V}$. Define the residual hypergraph derived by removing attributes in x as $\mathcal{Q}_x = (\mathcal{V}_x, \mathcal{E}_x)$, where $\mathcal{V}_x = \mathcal{V} - x$ and $\mathcal{E}_x = \{e - x : e \in E\}$. The edge quasi-packing number of $\mathcal{Q}$ is the maximum optimal fractional edge packing number over all $\mathcal{Q}_x$'s, i.e., $\psi^* = \max_{x \subseteq \mathcal{V}} \tau^*(\mathcal{Q}_x)$.

<table>
<tr><th rowspan="2">Joins</th><th colspan="2">Instance-optimal</th><th colspan="2">Output-optimal</th><th colspan="2">Worst-case Optimal</th></tr>
<tr><th>one-round</th><th>multi-round</th><th>one-round</th><th>multi-round</th><th>one-round</th><th>multi-round</th></tr>
<tr><td>tall-flat</td><td rowspan="2">$\widetilde{O}(L^*)$ [8]</td><td rowspan="2">$\Theta(L^*)$ [20]</td><td colspan="2" rowspan="2">$O\left(\frac{\text{IN}}{p^{1/\max\{1,k^*-1\}}} + (\frac{\text{OUT}}{p})^{\frac{1}{k^*}}\right)$ [20]</td><td rowspan="4">$\widetilde{O}\left(\frac{\text{IN}}{p^{1/\psi^*}}\right)$ [26]</td><td rowspan="3">$\widetilde{O}\left(\frac{\text{IN}}{p^{1/\rho^*}}\right)$ [18]</td></tr>
<tr><td>r-hierarchical w/o dangling tuples</td></tr>
<tr><td>r-hierarchical w/ dangling tuples</td><td rowspan="3">$\omega(L^*)$ [20]</td><td rowspan="3">$\omega\left(\frac{\text{IN}}{p} + \frac{\text{OUT}}{p}\right)$ [28]</td></tr>
<tr><td>acyclic</td><td>$\Theta\left(\frac{\text{IN}}{p} + \frac{\sqrt{\text{IN}\cdot\text{OUT}}}{p}\right)$ [20]
LB for $\text{OUT} \le p\cdot\text{IN}$</td></tr>
<tr><td>cyclic</td><td>$\tilde{\Omega}\left(\min\{\frac{\text{IN}+\text{OUT}}{p}, \frac{\text{IN}}{p^{2/3}}\}\right)$
for triangle join [20]</td><td>$\widetilde{O}\left(\frac{\text{IN}}{p^{1/\rho^*}}\right)$
for LW join [26],
graph join [24, 31]</td></tr>
<tr><td colspan="5"></td><td>$\Omega\left(\frac{\text{IN}}{p^{1/\tau^*}}\right)$
for $\boxminus$-join [18]</td></tr>
</table>

Table 1: Join algorithms in the MPC model. IN is the input size, OUT is the output size and p is the number of servers in the MPC model. For instance-optimal algorithms, $L^* = \frac{\text{IN}}{p} + L_{\text{instance}}(p, \mathcal{R})$. For output-optimal algorithm on r-hierarchical joins, $k^* = \lceil \log_{\text{IN}} \text{OUT} \rceil$. ψ^* is the optimal fractional edge quasi-packing number[2]; ρ^* is the optimal fractional edge cover number; τ^* is the optimal fractional edge packing number.

$\sqrt{\frac{\text{OUT}}{p}}$) for all r-hierarchical joins [20], improving the Yannakakis algorithm by a quadratic factor.

Unfortunately, such a quadratic improvement is not possible beyond r-hierarchical joins, even for the line-3 join $R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(C,D)$ [22], which is the simplest non-r-hierarchical join. Nevertheless, one can achieve a load of $O(\frac{\text{IN}}{p} + \frac{\sqrt{\text{IN}\cdot\text{OUT}}}{p})$ [20] for all acyclic joins (see Section 4.1 for more details), which improves upon Yannakakis algorithm as long as $\text{OUT} > \text{IN}$. This bound has also been shown to be optimal for $\text{OUT} = O(p \cdot \text{IN})$. Note that some restriction on OUT is inherent, because the $O(\frac{\text{IN}}{p} + \frac{\sqrt{\text{IN}\cdot\text{OUT}}}{p})$ bound cannot be optimal for all values of OUT. When OUT is large enough, a worst-case optimal algorithm will take over, as will be seen next.

What kind of output-sensitive bounds are achievable for cyclic joins remains an open problem, even for the triangle join. While there is an output-sensitive algorithm for the triangle join in the RAM model [11], we currently don't have any non-trivial upper bounds in the MPC model. On the lower bound side, a lower bound of $\tilde{\Omega}(\min\{\frac{\text{IN}+\text{OUT}}{p}, \frac{\text{IN}}{p^{2/3}}\})$ has been shown for cyclic joins in the MPC model [20]; please see Section 7.1 for more details. This shows a separation from acyclic joins, i.e., cyclic joins are harder than acyclic ones by at least a factor of $\tilde{\Omega}(\sqrt{\frac{\text{OUT}}{\text{IN}}})$.

Worst-case Optimal Join Algorithms. For worst-case optimal algorithms, we consider the simpler case where all relations $R(e)$ have equal size. In the RAM model, there is a unified algorithm computing all joins in $O(\text{IN}^{\rho^*})$ time [30, 33, 8], where ρ^* is the optimal fractional edge cover of the query hypergraph, i.e., it is the optimal solution of the following linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_{e\in\mathcal{E}} x_e \\ \text{subject to} \quad & \sum_{e:v\in e, e\in\mathcal{E}} x_e \ge 1, \forall v \in \mathcal{V} \\ & x_e \ge 0, \forall e \in \mathcal{E}. \end{aligned}$$

The conjecture had been an MPC algorithm with load $O(\frac{\text{IN}}{p^{1/\rho^*}})$. Such a load can be easily shown to be optimal: Each server can only produce $O(L^{\rho^*})$ join results in a constant number of rounds when the load is limited to L (implied by the AGM bound [4]), so all the p servers can produce at most $O(p \cdot L^{\rho^*})$ join results. On the other hand, the output size can be as large as IN^{ρ^*} on certain instances, so the load L has to be $\Omega(\frac{\text{IN}}{p^{1/\rho^*}})$ in the worst case. Indeed, this load has been achieved on certain classes of joins, such as graph joins [24, 31], Loomis-Whitney (LW) joins [26], and acyclic joins [18]; we describe some of these algorithms in Section 4.2 and Section 6.

Until very recently, an $\Omega(\frac{\text{IN}}{p^{1/\tau^*}})$ lower bound has been proved for the $\boxminus$-join $\mathcal{Q}_{\boxminus} = R_1(A,B,C) \bowtie R_2(D,E,F) \bowtie R_3(A,D) \bowtie R_4(B,E) \bowtie R_5(C,F)$ [18], where τ^* is the optimal fractional edge packing of the query, i.e., the optimal solution of the following linear program:

$$\begin{aligned} \text{maximize} \quad & \sum_{e\in\mathcal{E}} x_e \\ \text{subject to} \quad & \sum_{e:v\in e, e\in\mathcal{E}} x_e \le 1, \forall v \in \mathcal{V} \\ & x_e \ge 0, \forall e \in \mathcal{E}. \end{aligned}$$

On $\mathcal{Q}_{\boxminus}$, we have $\rho^* = 2$ and $\tau^* = 3$, so this result rules

out the possibility of achieving $O\left(\frac{\text{IN}}{p^{1/\rho^*}}\right)$ for all joins. Please see Section 7.2 for more details.

Join-Aggregate Algorithms. For a join-aggregate query, computing the full join and then performing the aggregation can be far from optimal, since the full join size can be much larger than the final output size OUT. In the RAM model, the Yannakakis algorithm can be modified to push down the aggregation as early as possible. Specifically, after removing the dangling tuples, we perform a bottom-up traversal of the join tree. For each $R(e)$ and $R(e')$ such that e is a leaf and e' is the parent of e, and we replace $R(e')$ with $\pi_{\mathbf{y}\cup\text{anc}(e')}R(e) \bowtie R(e')$, where $\text{anc}(e')$ is the set of attributes in e' that appear in the ancestors of e'. Then $R(e)$ is removed and the step is repeated until only one relation remains. It has been noted that this algorithm can be easily modified to handle join-aggregate queries, by replacing the projection $\pi_{\mathbf{y}\cup\text{anc}(e')}$ by an aggregation [23].

The running time of this algorithm is proportional to the largest intermediate join size $|R(e) \bowtie R(e')|$. It is known that if the query is *free-connex*, then the maximum intermediate join size is $O(\text{OUT})$ [23, 7]. For non-free-connex queries, Yannakakis gave an upper bound of $O(\text{IN} \cdot \text{OUT})$ in his original paper [34]. For matrix multiplication $\sum_B R_1(A,B) \bowtie R_2(B,C)$, which is the simplest non-free-connex query, this has been tightened to $O(\text{IN}\sqrt{\text{OUT}})$ [3], which is also shown to be optimal in the semiring model, as there are instances with $\Omega(\text{IN}\sqrt{\text{OUT}})$ elementary products. This bound is also extended to star queries[3], for which the bound becomes $O(\text{IN} \cdot \text{OUT}^{1-1/n})$.

Plugging the optimal two-way join algorithm to the Yannakakis algorithm, together with the MPC primitives for semi-join and aggregation (Section 2), we are able to compute join-project or join-aggregate queries in the MPC model. This is referred to as *distributed Yannakakis algorithm* in [23, 1]. The load of this algorithm is $O(\frac{\text{IN}}{p} + \frac{J}{p})$, where J is the maximum intermediate join size. Combined with the previously known bounds on J [34, 23, 7, 3], this implies that it can compute free-connex queries with load $O(\frac{\text{IN}}{p} + \frac{\text{OUT}}{p})$, matrix multiplication with load $O(\frac{\text{IN}}{p} + \frac{\text{IN}\sqrt{\text{OUT}}}{p})$, star queries with load $O(\frac{\text{IN}}{p} + \frac{\text{IN}\cdot\text{OUT}^{1-1/n}}{p})$, and general acyclic join-aggregate queries with load $O(\frac{\text{IN}}{p} + \frac{\text{IN}\cdot\text{OUT}}{p})$.

For any free-connex query, we can reduce it to a full acyclic join and then invoke the output-sensitive algorithm. This improves the load to $O(\frac{\text{IN}}{p} + \frac{\sqrt{\text{IN}\cdot\text{OUT}}}{p})$ (see Section 4.3). However, for non-free-connex queries, such a reduction is not possible. In fact, matrix multiplication, which is the simplest non-free-connex query, already requires a different treatment. Recently, it has been shown that matrix multiplication can be solved in $O(1)$ rounds with load $\widetilde{O}(\min\{\frac{\text{IN}}{\sqrt{p}}, \frac{\text{IN}}{p} + \frac{\text{IN}^{2/3}\cdot\text{OUT}^{1/3}}{p^{2/3}}\})$ which is also optimal [21]; Section 5 describes this algorithm in more detail. Improved bounds have also been obtained for other non-free-connex queries, as summarized in Table 2.

[3] A star query is defined as $\sum_B R_1(A_1,B) \bowtie R_2(A_2,B) \bowtie \cdots \bowtie R_m(A_m,B)$.

2. MPC PRIMITIVES

Assume $\text{IN} > p^{1+\epsilon}$ where $\epsilon > 0$ is any small constant. We first introduce the following primitives in the MPC model, all of which can be computed with linear load $O(\frac{\text{IN}}{p})$ in $O(1)$ rounds.

Sorting [16]: Given IN elements, redistribute them so that each server has $O(\frac{\text{IN}}{p})$ elements in the end, while any element on server i is smaller than or equal to any element on server j, for any $i < j$.

Reduce-by-key [22]: Given IN (key, value) pairs, compute the "sum" of values for each key, where the "sum" is defined by any associative operator. An aggregate $\oplus_{\mathbf{y}}\mathcal{R}$ can be computed as a reduce-by-key operation.

This primitive will also be frequently used to compute data statistics, for example the degree information. The degree of value $a \in \text{dom}(x)$ in relation $R(e)$ is defined as the number of tuples in $R(e)$ having this value in attribute x, i.e., $|\sigma_{x=a}R(e)|$. Each tuple $t \in R(e)$ is considered to have "key" $\pi_x t$ and "value" 1.

Multi-search [22]: Given N_1 elements $x_1, x_2, \cdots, x_{N_1}$ as set X and N_2 elements $y_1, y_2, \cdots, y_{N_2}$ as set Y, where all elements are drawn from an ordered domain. Set $\text{IN} = N_1 + N_2$. For each x_i, find its predecessor in Y, i.e., the largest element in Y but smaller than x_i.

Semi-Join: Given two relations R_1 and R_2 with a common attribute x, the semijoin $R_1 \ltimes R_2$ returns all the tuples in R_1 whose value on x matches that of at least one tuple in R_2. This can be reduced to a multi-search problem: For each $t \in R_1$, if its predecessor on the x attribute in R_2 is the same as that of t, then it is in the semi-join.

Note that we can remove all *dangling tuples*, i.e., those do not appear in the join results, of an acyclic-join [34] by a constant number of semi-joins, so it can be done in $O(1)$ rounds with linear load. Moreover, semi-joins are also used to *reduce* a join and join-aggregate query. If there exists a pair of relations $R(e), R(e')$ such that $e \subset e'$, then we can replace $R(e')$ with $R(e) \bowtie R(e')$ and then discard $R(e)$. Note that by the earlier definition, the annotation of a join result is the $\otimes$-aggregate of the annotations of tuples comprising the join result, so the annotations in $R(e)$ are aggregated into those in $R(e')$ correctly. As mentioned, a join query can be reduced by applying a set of semi-joins until no relation is a subset of another relation.

Parallel-packing [22]: Given IN numbers $x_1, x_2, \cdots, x_{\text{IN}}$ where $0 < x_i \le 1$ for $i = 1, 2, \cdots, \text{IN}$, group them into m sets $Y_1, Y_2, \cdots, Y_m$ such that $\sum_{i\in Y_j} x_i \le 1$ for all j, and $\sum_{i\in Y_j} x_i \ge \frac{1}{2}$ for all but one j. Initially, the IN numbers are distributed arbitrarily across all servers, and the algorithm should produce all pairs (i, j) if $i \in Y_j$ when done. Note that $m \le 1 + 2\sum_i x_i$.

Join Aggregate Query	The Yannakakis algorithm	New results [21]
Matrix Multiplication	$O\left(\frac{\text{IN}}{p} + \frac{\text{IN}\cdot\sqrt{\text{OUT}}}{p}\right)$ [23, 3]	$\widetilde{O}\left(\frac{N_1+N_2}{p} + \min\left\{\sqrt{\frac{N_1 N_2}{p}}, \frac{N_1^{1/3}\cdot N_2^{1/3}\cdot \text{OUT}^{1/3}}{p^{2/3}}\right\}\right)$ (1) optimal for $N_1, N_2 \geq 2$ and $\max\{N_1, N_2\} \leq \text{OUT} \leq N_1 N_2$; (2) $\widetilde{O}\left(\frac{\text{IN}}{p} + \min\left\{\frac{\text{IN}}{\sqrt{p}}, \frac{\text{IN}^{2/3}\cdot\text{OUT}^{1/3}}{p^{2/3}}\right\}\right)$ if $N_1 = N_2$;
Star	$O\left(\frac{\text{IN}}{p} + \frac{\text{IN}\cdot\text{OUT}^{1-\frac{1}{n}}}{p}\right)$ [23, 3]	$\widetilde{O}\left((\frac{\text{IN}\cdot\text{OUT}}{p})^{2/3} + \frac{\text{IN}\cdot\text{OUT}^{1/2}}{p} + \frac{\text{IN}+\text{OUT}}{p}\right)$
Line	$O\left(\frac{\text{IN}}{p} + \frac{\text{IN}\cdot\text{OUT}}{p}\right)$ [23]	
Tree		$\widetilde{O}\left(\frac{\text{IN}\cdot\text{OUT}^{2/3}}{p} + \frac{\text{IN}+\text{OUT}}{p}\right)$

Table 2: Summary of Join-Aggregate Queries in the MPC model. In the sparse matrix multiplication, N_1, N_2 are the number of non-zero entries in two input matrices respectively. Generally, any instance for the join-aggregate query has input size IN and output size OUT. p is the number of servers.

Output size computation [20, 23, 21]: For any acyclic join $\mathcal{Q}$, the value of OUT can be computed exactly as a special case of our join-aggregate algorithm, which will be described in Section 4.3. This result also applies for free-connex join-aggregate queries after applying a primitive transformation.

However, for cyclic full join and non-free-connex join-aggregate queries, how to compute OUT effectively is still open. Fortunately, a constant-factor approximation of OUT for line join-aggregate queries (including matrix multiplication as a special case) can be computed in $O(1)$ rounds with linear load. A similar idea has been used by [13] in the RAM model.

3. R-HIERARCHICAL JOINS

It is known that r-hierarchical joins are precisely the class of joins that admit instance-optimal algorithms. In this section, we first present an instance-optimal algorithm, and then the lower bound. There are two such algorithms. The BinHC algorithm [8] is randomized and has some extra polylogarithmic factors, while the one in [20] is deterministic without any logarithmic factors.

Note that the BinHC algorithm is a generalization of the HyperCube algorithm to general joins. The load of the BinHC algorithm is parameterized by the degrees of all subsets of attribute values. Beame et al. [8] show that BinHC is optimal (up to polylog factors) within the class of instances sharing the same degrees, among all one-round MPC algorithms. A stronger analysis of the BinHC algorithm shows that it is actually instance-optimal (up to polylog factors) for (1) all tall-flat joins, and (2) all r-hierarchical joins provided that the instance does not contain dangling tuples. Furthermore, since the per-instance lower bound (2) also holds for multi-round algorithms, these instance-optimality results extend to multi-round algorithms as well. For r-hierarchical joins with dangling tuples, one-round algorithms cannot achieve $O(\frac{\text{IN}}{p} + L_{\text{instance}}(p, \mathcal{R}))$ load. But, removing dangling tuples (an MPC primitive) first and then running BinHC algorithm, leads to a multi-round algorithm of $O((\frac{\text{IN}}{p} + L_{\text{instance}}(p, \mathcal{R})) \cdot \log^{O(1)} p)$ load, where the $O(1)$ exponent depends on the query size, and is at least m, the number of relations.

Below we describe the latter deterministic algorithm with load $O(\frac{\text{IN}}{p} + L_{\text{instance}}(p, \mathcal{R}))$, i.e., improving the instance-optimality ratio from $\log^{O(1)} p$ to $O(1)$.

3.1 An Instance-Optimal Algorithm

We give a sketch of the recursive algorithm in [20]. It chooses a fixed threshold L, whose value will be determined later.

Inductive hypothesis: For an r-hierarchical join $\mathcal{Q}$ with an instance $\mathcal{R}$, the join results $\mathcal{Q}(\mathcal{R})$ can be computed using $\Psi(\mathcal{Q}, \mathcal{R}, L)$ servers in $O(1)$ rounds with $O(L)$ load, where

$$\Psi(\mathcal{Q}, \mathcal{R}, L) = \max_{S \subseteq \mathcal{E}} \left\lceil \frac{\bowtie_{e \in S} R(e)}{L^{|S|}} \right\rceil.$$

Base case: When $\mathcal{Q}$ has just one relation, say $\mathcal{E} = \{e\}$, in which case the algorithm simply emits all tuples in the relation. This step can be done using $O(\frac{|R(e)|}{L})$ servers with $O(L)$ load. Note that $\frac{|R(e)|}{L} \leq \Psi(\mathcal{Q}, \mathcal{R}, L)$.

General case: In general, we first reduce the query and remove all the dangling tuples, which can be done by MPC primitives. Then we are left with a hierarchical join $\mathcal{Q}$ on an instance $\mathcal{R}$ with no dangling tuples. Note that $\mathcal{Q}$ has an attribute forest, denoted as $\mathcal{T}$, where each relation corresponds to a root-to-leaf path of $\mathcal{T}$. The algorithm will proceed by the following two cases.

Case (1): $\mathcal{Q}$ is connected. Suppose the root attribute of $\mathcal{T}$ is x. Since x is included in all relations, we can decompose the original join into disjoint subsets by the value on x. Each $a \in \text{dom}(x)$ induces a sub-instance $\mathcal{R}_a = \{\sigma_{x=a} R(e) : e \in \mathcal{E}\}$. A sub-instance is *heavy* if it contains more than L tuples, and *light* otherwise.

All light sub-instances are packed into groups (an MPC primitive) and send a group as whole to one server for computation. Then, it remains to compute $\mathcal{Q}_x$ on each heavy $\mathcal{R}_a$, where $\mathcal{Q}_x$ is the residual query by removing x from all relations in $\mathcal{Q}$. The challenge is to allocate p servers in total to these residual queries appropriately so as to compute all $\mathcal{Q}_x(\mathcal{R}_a)$'s in parallel

while ensuring a uniform load of $O(L)$. To do so, we allocate for instance $\mathcal{R}_a$

$$p_a = \max_{S \subseteq \mathcal{E}} \left\lceil \frac{\mathcal{Q}_x(\mathcal{R}_a, S)}{L^{|S|}} \right\rceil$$

servers and compute $\mathcal{Q}_x(\mathcal{R}_a)$'s recursively in parallel. By hypothesis, for each heavy sub-instance $\mathcal{R}_a$, $\mathcal{Q}_x(\mathcal{R}_a)$ can be computed using p_a servers with $O(L)$ load.

Case (2): $\mathcal{Q}$ is disconnected. Let $\mathcal{Q}_1, \mathcal{Q}_2, \cdots, \mathcal{Q}_k$ be the connected components of $\mathcal{Q}$. In this case, the join becomes a Cartesian product $\mathcal{Q}_1(\mathcal{R}_1) \times \cdots \times \mathcal{Q}_k(\mathcal{R}_k)$, where each $\mathcal{Q}_i(\mathcal{R}_i)$ is a join under the Case (1).

The idea is to arrange servers into a $p_1 \times p_2 \times \cdots \times p_k$ hypercube, where each server is identified with coordinates $(c_1, c_2, \cdots, c_k)$, for $c_i \in [p_i]$. For every combination $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_k$, the p_i servers with coordinates $(c_1, \cdots, c_{i-1}, *, c_{i+1}, \cdots, c_k)$ form a group to compute $\mathcal{Q}_i(\mathcal{R}_i)$ (using the algorithm under Case (1)). Yes, each $\mathcal{Q}_i(\mathcal{R}_i)$ is computed $p_1 \cdots p_{i-1} p_{i+1} \cdots p_k$ times, which seems to be a lot of redundancy. However, as we shall see, there will be no redundancy in terms of the final join results, and it is exactly due to this redundancy that we avoid the shuffling of the intermediate result and achieve an optimal load. Consider a particular server $(c_1, \ldots, c_k)$. It participates in k groups, one for each $\mathcal{Q}_i(\mathcal{R}_i), i = 1, \ldots, k$. For each $\mathcal{Q}_i(\mathcal{R}_i)$, it emits a subset of its join results, denoted $\mathcal{Q}_i(\mathcal{R}_i, c_1 \ldots, c_k)$. Then the server emits the Cartesian product $\mathcal{Q}_1(\mathcal{R}_1, c_1 \ldots, c_k) \times \cdots \times \mathcal{Q}_k(\mathcal{R}_k, c_1 \ldots, c_k)$. Note that for each group of servers computing $\mathcal{Q}_i(\mathcal{R}_i)$, the p_i servers in the group emit $\mathcal{Q}_i(R_i)$ with no redundancy, so there is no redundancy in emitting the Cartesian product.

It remains to show how to allocate the p servers for each sub-query so that $p_1 \cdots p_k = O(p)$ If $|\mathcal{R}_i| < L$ is light, we set $p_i = 1$; otherwise set

$$p_i = \max_{S \subseteq \mathcal{E}_i} \left\lceil \frac{|\mathcal{Q}_i(\mathcal{R}_i, S)|}{L^{|S|}} \right\rceil.$$

By hypothesis, $\mathcal{Q}_i(\mathcal{R}_i)$ can be computed using p_i servers with $O(L)$ load. Although each server participates in k sub-queries, it still has a load of $O(L)$.

Combining two cases completes the inductive proof.

Choosing L. At last, we show how to choose an appropriate value of L. For an input join $\mathcal{Q}$ and an instance $\mathcal{R}$, we first compute the value of $L_{\text{instance}}(p, \mathcal{R})$ by a constant number of MPC primitives. Setting $L = L_{\text{instance}}(p, \mathcal{R}) + \frac{\text{IN}}{p}$ will yield $\Psi(\mathcal{Q}, \mathcal{R}, L) = O(p)$, thus leading to an instance-optimal algorithm.

3.2 Lower Bound

The instance-optimal load $O(\frac{\text{IN}}{p} + L_{\text{instance}}(p, \mathcal{R}))$ is not achievable beyond r-hierarchical joins. More precisely, the following lower bound is proved in [20]:

THEOREM 1. *For any $\text{IN} \geq p^{3/2}$, there exists an instance $\mathcal{R}$ with input size $\Theta(\text{IN})$ for any acyclic but non-r-hierarchical join, such that any tuple-based algorithm that computes the join in $O(1)$ rounds must have a load of $\Omega\left(\frac{\text{IN}}{p^{1/2} \log \text{IN}}\right)$, while $L_{instance}(p, \mathcal{R}) = O(\frac{\text{IN}}{p})$.*

4. ACYCLIC JOIN

Theorem 1 has ruled out the possibility of achieving an instance-optimal algorithm for non-r-hierarchical joins. In this section, we review two algorithms for general acyclic joins, one is worst-case optimal algorithm [18] and the other is output-sensitive [20] (but optimal on specific range of OUT). We illustrate the high-level idea of these two algorithms through the line-3 join $R_1(A, B) \bowtie R_2(B, C) \bowtie R_3(C, D)$, which is the simplest acyclic but non-r-hierarchical join. Finally, we turn to free-connex join-aggregate query. After applying a linear transformation procedure, any free-connex join-aggregate query can be reduced to a full join query, thus benefits from any results achieved for acyclic joins.

4.1 Output-sensitive algorithm

Note that in the RAM model, the Yannakakis algorithm first removes all the dangling tuples and then performs pairwise joins in some arbitrary order. It is shown that the join order does not affect the asymptotic running time: After dangling tuples have been removed, any intermediate join result is part of a full join result, so the running time of the last join, which is $\Theta(\text{OUT})$, dominates that of any intermediate join. Interestingly, the join order does matter in the MPC model. Moreover, it is shown that a global best order may not exist even for the line-3 join. The basic idea of the output-sensitive algorithm is to decompose the join into multiple pieces, and find a provably good join order of the Yannakakis algorithm for each piece.

Line-3 join. The value of OUT should be computed in advance (an MPC primitive). Set $\tau = \sqrt{\frac{\text{OUT}}{\text{IN}}}$. We first compute degrees for values of attribute B in relation R_1. A value $b \in \text{dom}(B)$ is *heavy* if it has degree greater than τ in R_1, and *light* otherwise. Let B^H, B^L be the set of heavy and light values in B respectively. In this way, we decompose the join into the following two parts as $\mathcal{Q}_1, \mathcal{Q}_2$ and compute them with aggregated Yannakakis algorithm using different join orderings:

$$\mathcal{Q}_1 = R_1(A, B^H) \bowtie \big(R_2(B^H, C) \bowtie R_3(C, D)\big)$$
$$\mathcal{Q}_2 = \big(R_1(A, B^L) \bowtie R_2(B^L, C)\big) \bowtie R_3(C, D)$$

The observation is that the intermediate join $R_2(B^H, C) \bowtie R_3(C, D)$ has its size bounded by $\frac{\text{OUT}}{\tau}$, since each intermediate join result has a heavy B value, so it joins with at least τ tuples in R_1. Meanwhile, the intermediate join $R_1(A, B^L) \bowtie R_2(B^L, C)$ has its size bounded by $\text{IN} \cdot \tau$, since each tuple from R_2 can join with at most τ tuples from R_1. The load of computing two sub-queries is $O(\frac{\text{IN}}{p} + \frac{\text{IN} \cdot \tau}{p} + \frac{\text{OUT}}{p \cdot \tau} + \sqrt{\frac{\text{OUT}}{p}}) = O(\frac{\text{IN}}{p} + \frac{\sqrt{\text{IN} \cdot \text{OUT}}}{p})$. Note that the value of τ is set to achieve the minimum.

This algorithm can be extended to arbitrary acyclic joins with the same load complexity, but the decomposition is much more complicated based on the join

tree of acyclic join. The challenge is still to bound the size of any intermediate join result as $O(\text{IN}\sqrt{\text{OUT}})$. We refer readers to [20] for algorithmic details. Meanwhile, a lower bound has been presented for any non-r-hierarchical acyclic join: For any $\text{IN} \leq \text{OUT} \leq c \cdot p \cdot \text{IN}$ for some constant c, there exists an instance $\mathcal{R}$ with input size $\Theta(\text{IN})$ and output size $\Theta(\text{OUT})$, such that any tuple-based algorithm computing it in $O(1)$ rounds must have a load of $\Omega(\min\{\frac{\sqrt{\text{IN}\cdot\text{OUT}}}{p\cdot\log \text{IN}}, \frac{\text{IN}}{\sqrt{p}}\})$. This establishes the output-optimality of the output-sensitive algorithm for $\text{OUT} = O(p \cdot \text{IN})$.

Remark. We have obtained a complete understanding of line-3 join in terms of output-optimality: (1) when $\text{OUT} \leq \text{IN}$, the Yannakakis algorithm has linear load $O(\frac{\text{IN}}{p})$; (2) when $\text{IN} < \text{OUT} \leq c\cdot p\cdot\text{IN}$, the lower bound becomes $\tilde{\Omega}(\frac{\sqrt{\text{IN}\cdot\text{OUT}}}{p})$, which is matched by the output-sensitive algorithm; (3) when $\text{OUT} \geq c\cdot p\cdot\text{IN}$, the lower bound is $\Omega(\frac{\text{IN}}{\sqrt{p}})$, which is matched by the worst-case optimal algorithm [19, 26, 20]. In particular, this means that when OUT is large enough, the load complexity of the join is no longer output-sensitive. This also stands in contrast with the RAM model, where the complexity of acyclic joins always grows linearly with OUT. On more complicated joins, the worst-case optimal algorithms have a higher load, and the output-optimality for OUT values in the middle is still unclear.

4.2 Worst-case Optimal Algorithm

In the output-sensitive algorithm, the original join is divided into $O(2^{|\mathcal{E}|})$ pieces, which is still a constant as long as the query has constant size. However, to target the worst-case optimal algorithm, a more fine-grained decomposition of the original join is needed, and intermediate join results should be dealt with more carefully.

The framework of this worst-case optimal algorithm is also based on the join tree of acyclic join: Each time it peels one leaf relation off and reduces the original join into a smaller one until it becomes empty. Eventually, each relation is divided into disjoint partitions of size $\Theta(L)$, where $L = O(\frac{\text{IN}}{p^{1/\rho^*}})$ is the target of the worst-case optimal algorithm for computing acyclic joins, and each piece of subjoin query involves exactly one partition from each relation. In this way, each subjoin can be computed locally and join results are emitted directly without generating any intermediate join result.

Line-3 join [19, 26, 20]. We next present an algorithm for line-3 join with load $O(L)$, where $L = \frac{\text{IN}}{\sqrt{p}}$.

Similarly, we first compute degrees for values of attribute B in relation R_1. A value $b \in \text{dom}(B)$ is *heavy* if it has degree greater than L in R_1 and *light* otherwise. Let B^H, B^L be the set of heavy and light values in B respectively. We further divide B^L into $k = O(\frac{N_1}{\tau})$ disjoint groups $B_1, B_2, \cdots, B_k$ such that values in each group have total degree $\Theta(\tau)$ in relation R_1. The original join is decomposed into following subjoins:

$$\mathcal{Q}_1 = \bigcup_{b\in B^H} (R_1(A, b) \times R_2(b, C) \bowtie R_3(C, D))$$

$$\mathcal{Q}_2 = \bigcup_{i} (R_1(A, B_i) \bowtie R_2(B_i, C) \bowtie R_3(C, D))$$

For a subjoin query induced by heavy value b in $\mathcal{Q}_1$, we compute the Cartesian product between tuples in $R_1(A, b)$ and results of $R_2(b, C) \bowtie R_3(C, D)$. To compute these subjoins in parallel, we allocate servers proportional to the degree of b in relation R_1. For a subjoin query induced by group B_i in $\mathcal{Q}_2$, we allocate $\sqrt{p}$ servers and compute $R_1(A, B_i) \bowtie R_2(B_i, C) \bowtie R_3(C, D)$ by broadcasting tuples in $R_1(A, B_i)$ and invoking the optimal binary-join algorithm for $R_2(B_i, C) \bowtie R_3(C, D)$.

4.3 Free-Connex Join-Aggregate Queries

We now present a primitive through which any free-connex join-aggregate query can be transformed into a full join, running in $O(1)$ rounds with linear load.

In the preprocessing step, we remove the dangling tuples and reduce the query. We find a free-connex width-1 GHD $\mathcal{T}$ of $\mathcal{Q}$ [6, 5]. Note that the nodes of $\mathcal{T}$ also define a hypergraph, and can be regarded as another join-aggregate query, but with the property that it has a free-connex subset $\mathcal{T}'$ such that $\mathbf{y} = \bigcup_{u\in\mathcal{T}'} u$. We construct an instance $\mathcal{R}_\mathcal{T} = \{R(u) : u \in \mathcal{T}\}$ such that $\mathcal{Q}_\mathbf{y}(\mathcal{R}) = \mathcal{T}(\mathcal{R}_\mathcal{T})$, where $\mathcal{T}(\mathcal{R}_\mathcal{T})$ denotes the result of running the query defined by $\pi_\mathbf{y}\mathcal{T}$ on $\mathcal{R}_\mathcal{T}$. Observe that on a reduced $\mathcal{Q}$, the condition $e \subseteq u$ in property (2) of a width-1 GHD can be replaced by $e = u$, since if $e \subset u$ and $u \subseteq e'$ for some other $e' \in \mathcal{E}$ due to property (3), we would find $e \subset e'$. This implies that $\mathcal{T}$ has only two types of nodes: (1) all hyperedges in $\mathcal{E}$, and (2) nodes that are a proper subset of some $e \in \mathcal{E}$. Then we construct $\mathcal{R}_\mathcal{T}$ as follows. For each $u \in \mathcal{T}$ of type (1), we set $R(u) := R(e)$ where $e = u$; for each $u \in \mathcal{T}$ of type (2), we set $R(u) := R(e)$ for any $e \in \mathcal{E}, u \subset e$, but the annotations of all tuples in $R(u)$ are set to 1 (the $\otimes$-identity). Then, we only focus on computing $\mathcal{T}(\mathcal{R}_\mathcal{T})$.

LEMMA 1. *Given any free-connex width-1 GHD $\mathcal{T}$ and an instance $\mathcal{R}_\mathcal{T}$, an instance $\mathcal{R}_{\mathcal{T}'}$ can be returned in $O(1)$ rounds with linear load such that $\mathcal{T}(\mathcal{R}_\mathcal{T}) = \mathcal{T}'(\mathcal{R}_{\mathcal{T}'})$, where $\mathcal{T}'$ is the free-connex subset of $\mathcal{T}$.*

By plugging to the optimal two-way join algorithm in [8, 22], the aggregated Yannakakis algorithm [23] can aggregate over all the non-output attributes, returning a modified query $\mathcal{T}'(\mathcal{R}_{\mathcal{T}'})$ that only has the output attributes. Because $\mathcal{T}'$ is an acyclic join, thus any results in Section 4.2 and Section 4.1 can be applied to $\mathcal{T}'$.

Moreover, the join size of a (non-aggregate) join is a special join-aggregate query with $\mathbf{y} = \emptyset$, without any circular dependency here, which must be free-connex. Thus, for any acyclic join $\mathcal{Q}$ and any instance $\mathcal{R}$, $|\mathcal{Q}(\mathcal{R})|$ can be computed in $O(1)$ rounds with linear load.

5. SPARSE MATRIX MULTIPLICATION

In this section, we review the output-optimal algorithm [21] for sparse matrix multiplication problem, i.e., $\sum_B R_1(A, B) \bowtie R_2(B, C)$, which is the simplest non-free-connex query. Let N_1, N_2 be the sizes of R_1, R_2 respectively.

First, if $N_1 = 1$ (resp. $N_2 = 1$), the problem can be trivially solved by simply broadcasting the only tuple in R_1 (resp. R_2) with $O(1)$ load. In general, for any $N_1, N_2 \geq 2$, it can be solved in $O(1)$ rounds with

$$\widetilde{O}\left(\frac{N_1+N_2}{p} + \min\{\sqrt{\frac{N_1N_2}{p}}, \frac{N_1^{1/3}\cdot N_2^{1/3}\cdot \mathrm{OUT}^{1/3}}{p^{2/3}}\}\right)$$

load, with probability at least $1 - 1/N^{O(1)}$.

One can verify that this presents an asymptotic improvement over the Yannakakis algorithm for $\mathrm{OUT} = \omega(1)$. In fact, our algorithm performs the same amount of computation as the Yannakakis algorithm and computes all $O(\mathrm{IN}\sqrt{\mathrm{OUT}})$ elementary products, which is unavoidable in the semiring model. The key to the reduction in load is *locality*, namely, we arrange these elementary products to be computed on the servers in such a way that most of them can be aggregated locally. The standard Yannakakis algorithm has no locality at all, and all the elementary products are shuffled around.

Observation 1. *If $N_1 > pN_2$ or $N_2 > pN_1$, matrix multiplication can be computed with linear load.*

We start with Observation 1, in which two cases can be tackled just by sorting (an MPC primitive). Below, we assume $1/p < N_1/N_2 < p$.

5.1 Worst-case optimal algorithm

We first describe an algorithm with load $O(\sqrt{\frac{N_1N_2}{p}})$. This is actually worst-case optimal because when there is a single value in the domain of attribute B, there are N_1N_2 elementary products. A server with load L can compute $O(L^2)$ of them in a constant number of rounds, so we have $pL^2 = \Omega(N_1N_2)$, i.e., $L = \Omega(\sqrt{\frac{N_1N_2}{p}})$.

Set $L = \sqrt{\frac{N_1N_2}{p}}$. We first compute all degrees for values in attributes A and C. A value $a \in \mathrm{dom}(A)$ (resp. $c \in \mathrm{dom}(C)$) is *heavy* if it has degree greater than L in R_1 (resp. R_2), and *light* otherwise. The set of heavy and light values in A (resp. C) is denoted as A^H and A^L (resp. C^H and C^L). Then, the original query can be decomposed into four subqueries:

$$\sum_B R_1(A^?, B) \bowtie R_2(B, C^!),$$

where $?, !$ can be either H or L. Note that the results produced by these subqueries are disjoint and the final aggregated result is just their union. We handle each subquery separately.

Case (1): At least one of A, C is heavy. W.l.o.g., assume A is heavy. We use the aggregated Yannakakis to compute $\sum_B R_1(A^H, B) \bowtie R_2(B, C)$ with load $O(\frac{J}{p})$, where $J = |R_1(A^H, B) \bowtie R_2(B, C)| = O\left(\frac{N_1N_2}{\tau}\right)$ since each tuple in R_2 can join with at most $\frac{N_1}{\tau}$ values in A^H.

Case (2): Both A, C are light. We divide A^L into $k = O(\frac{N_1}{L})$ disjoint groups $A_1, A_2, \cdots, A_k$ such that each group has total degree $O(L)$ in $R_1(A^L, B)$ (an MPC primitive) as well as $l = O(\frac{N_2}{L})$ disjoint groups $C_1, C_2, \cdots, C_l$ for C^L such that each group has total degree $O(L)$ in $R_2(B, C^L)$. Then we arrange all servers into a $\lceil\frac{N_1}{L}\rceil \times \lceil\frac{N_2}{L}\rceil$ grid, where each one is associated with (i, j) for $i \in [\lceil\frac{N_1}{L}\rceil], j \in [\lceil\frac{N_2}{L}\rceil]$. The server (i, j) receives all tuples in $R_1(A_i, B), R_2(B, C_j)$ and then compute the subquery $\sum_B R_1(A_i, B) \bowtie R_2(B, C_j)$ locally.

5.2 Output-sensitive algorithm

We first compute a constant-factor approximation of OUT, which should be known by the algorithm in advance. Another important observation on OUT is that any matrix multiplication can be computed with a load $O(\mathrm{OUT} + \frac{\mathrm{IN}}{p})$, through sorting and reduce-by-key primitives. Below, we consider the case that $\mathrm{OUT} > \frac{N_1+N_2}{p}$.

Observation 2. *Matrix multiplication can be computed with load $\widetilde{O}(\mathrm{OUT} + \frac{N_1+N_2}{p})$.*

We next show an algorithm with load $O(L)$, where

$$L = \frac{N_1^{1/3}\cdot N_2^{1/3}\cdot \mathrm{OUT}^{1/3}}{p^{2/3}} + \frac{N_1+N_2}{p}.$$

Slightly different from the previous algorithm, value $a \in \mathrm{dom}(A)$ is *heavy* if it participates in more than $\tau = \sqrt{\frac{N_2\cdot\mathrm{OUT}\cdot L}{N_1}}$ final aggregate results, and *light* otherwise. Note that there are at most $\frac{\mathrm{OUT}}{\tau}$ values in A^H since aggregate results by different a's are disjoint.

Case (1): A is heavy. We use the aggregated Yannakakis to compute $\sum_B R_1(A^H, B) \bowtie R_2(B, C)$ with load $O(\frac{J}{p})$, where $J = |R_1(A^H, B) \bowtie R_2(B, C)| = O\left(\frac{N_2\cdot\mathrm{OUT}}{\tau}\right)$, since each tuple in R_2 can join with at most $\frac{\mathrm{OUT}}{\tau}$ values in A^H.

Case (2): A is light. We divide A^L into $k_1 = O(\frac{\mathrm{OUT}}{\tau})$ disjoint groups $A_1, A_2, \cdots, A_{k_1}$ such that values in each group appear in $O(\tau)$ final results. On group A_i, value $c \in \mathrm{dom}(C)$ is *heavy* if it appears in more than L results of the subquery $\sum_B \sigma_{A\in A_i} R_1(A, B) \bowtie R_2(B, C)$, and *light* otherwise. The set of heavy and light values in $\mathrm{dom}(C)$, with respect to A_i, is denoted as C_i^H and C_i^L. Observe that $|C_i^H| \leq \frac{\tau}{L}$.

Case (2.1): C is heavy. For each group A_i, we use the aggregated Yannakakis algorithm to compute $\sum_B R_1(A_i, B) \bowtie R_2(B, C_i^H)$ with $J_i = |R_1(A_i, B) \bowtie R_2(B, C_i^H)| = O(|R_1(A_i, B)|\cdot \frac{\tau}{L})$, since each tuple in $R_1(A_i, B)$ can join with at most $\frac{\tau}{L}$ values in C_i^H. To compute all groups in parallel, we allocate servers proportional to the input sizes of each group and achieve a uniform load of $O(L)$ for all groups.

Case (2.2): C is light. For each group A_i, we divide C_i^L into $k_2 = O(\sqrt{\frac{\mathrm{OUT}}{L}}\cdot\sqrt{\frac{N_2}{N_1}})$ disjoint groups C_1^i, C_2^i, $\cdots$, $C_{k_2}^i$ such that values in each group appear together in $O(L)$ results of the subquery $\sum_B R_1(A_i, B) \bowtie R_2(B, C)$. Note that each pair of (A_i, C_j^i) further defines a subquery as $\sum_B R_1(A_i, B) \bowtie R_2(B, C_j^i)$, which

has output size smaller than L. Thus, by allocating servers proportional to the input sizes, we can reduce it to the case in Observation 2, achieving a uniform load of $O(L)$ for all subqueries.

5.3 Lower Bound

We mention the following two lower bounds, which together show that the upper bound achieved is optimal when $N_1, N_2 \geq 2$ and $\max\{N_1, N_2\} \leq \text{OUT} \leq N_1 N_2$.

THEOREM 2. *For any $N_1, N_2 \geq 2$, there exists an instance $\mathcal{R}$ for matrix multiplication with input sizes N_1, N_2 such that any algorithm computing it must incur a load of $\Omega\left(\frac{N_1+N_2}{p}\right)$ in the semiring MPC model.*

THEOREM 3. *For any $1/p \leq N_1/N_2 \leq p$ and $1 \leq \text{OUT} \leq N_1 N_2$, there exists an instance $\mathcal{R}$ for sparse matrix multiplication with input sizes N_1, N_2 and output size* OUT*, such that any algorithm computing it in $O(1)$ rounds in the semiring MPC model must incur a load of $\Omega\left(\min\left\{\sqrt{\frac{N_1 N_2}{p}}, \frac{N_1^{1/3} \cdot N_2^{1/3} \cdot \text{OUT}^{1/3}}{p^{2/3}}\right\}\right)$.*

5.4 General Tree Queries

In [21], an output-sensitive algorithm based on matrix multiplication is also proposed for general tree queries. However, an inherent difficulty for tree query is that it is not known how to compute a constant-factor approximation of OUT without actually computing all the query results. One standard technique is to repeatedly double a guess of OUT and try to run the algorithm, until the guess is correct (i.e., within a constant factor of the true value). This would work in a sequential model, since the running times of the successive guesses will form a geometric series, increasing the total running time by only a constant factor. In the parallel model like the MPC, although the total load can still be bounded, but the repeated guesses would lead to $O(\log N)$ rounds of computation. The idea to get around this is to make the algorithm *oblivious* to the value of OUT, i.e., the value of OUT is not needed by the algorithm but only used in the analysis. All details, including how to reduce the orignal query into a matrix multiplication problem, and how to bound the sizes of intermediate query results with OUT, are referred to [21].

6. GRAPH JOIN

Graph join, as a special class of cyclic joins, enjoy very nice properties as follows: (1) $\tau^* \leq \rho^*$; (2) $\tau^* + \rho^* = |\mathcal{V}|$; (3) τ^* and ρ^* admit half-integral solutions. In plain language, a graph join always has its optimal fractional edge packing number smaller than edge covering number; moreover, every edge takes a value in $\{0, \frac{1}{2}, 1\}$ in the optimal solution for edge cover and packing. Those three properties are taken full use by algorithm design.

In [24], Ketsman and Suciu gave an algorithm in the MPC model for computing the graph join with load $O(\frac{\text{IN}}{p^{1/\rho^*}})$. Later, this complicated algorithm was simplified by Tao [31], and the number of rounds required was decreased from 7 to 3. These two algorithms are based on the HYPERCUBE algorithm [2, 8], which arranges servers into a hypercube where each dimension corresponds to one attribute. We first mention one important property of this algorithm on graph joins, resilient to the data skew. Let $\mathbf{p}$ be a function mapping each attribute x to a positive integer $\mathbf{p}_x$. Instance $\mathcal{R}$ is *skew-free* with respect to $\mathbf{p}$ if for each relation $R(e)$ with $e = \{x, y\}$, each value of $\text{dom}(x)$ has degree at most $\frac{|R(e)|}{\mathbf{p}_x}$ and each value of $\text{dom}(y)$ has degree at most $\frac{|R(e)|}{\mathbf{p}_y}$. For a graph join $\mathcal{Q}$ and a skew-free instance $\mathcal{R}$ under $\mathbf{p}$, the join result $\mathcal{Q}(\mathcal{R})$ can be computed using $\prod_{x \in \mathcal{V}} \mathbf{p}_x$ servers in a single round with load complexity $O(\text{IN}/\min_{e \in \mathcal{E}} \prod_{v \in e} \mathbf{p}_v)$.

For a graph join $\mathcal{Q}$ and an instance $\mathcal{R}$, if each value of any attribute has degree smaller than $\frac{\text{IN}}{p^{1/2\rho^*}}$, then this is a skew-free instance w.r.t. $\mathbf{p}_x = p^{1/2\rho^*}$ for every attribute $x \in \mathcal{V}$. Implied by the result above, such an instance can be computed using $p^{|\mathcal{V}|/2\rho^*} \leq p$ servers in a single round with load $O(\frac{\text{IN}}{p^{1/\rho^*}})$. The challenge comes when skew exists. The high-level idea is to decompose the original join into a set of skew-free subjoins and allocate servers appropriately for computing all subjoins in parallel while achieving a uniform load of $O(\frac{\text{IN}}{p^{1/\rho^*}})$.

Set $\tau = \frac{\text{IN}}{\lambda}$. For each attribute x, it divides values in $\text{dom}(x)$ into *heavy* and *light*. More specifically, value $a \in \text{dom}(x)$ is *heavy* if there exists a relation e for $x \in e$ such that a has degree more than τ in $R(e)$, and *light* otherwise. Then, the join results can be distinguished into $O(2^{|\mathcal{V}|})$ cases, such that each case corresponds to one subset of attributes $S \subseteq \mathcal{V}$ and each join results fall into this case has heavy values in S and light values in $\mathcal{V} - S$. Fixing one subset of attributes S, there are $O(\lambda^S)$ different combinations of heavy values over S, and each one is noted as *configuration*. In this way, the original join is divided into a set of subjoins, each one corresponds to a residual query by fixing a configuration. Consider a subjoin defined by a configuration over attributes S. Observe that all values in $\text{dom}(x)$ for $x \in \mathcal{V} - S$ are light,thus this is a skew-free instance. The HYPERCUBE algorithm is then applied for each subjoin independently. More algorithmic details, including how to allocate servers for each subjoin, and semi-join reduction, can be found in [31].

7. LOWER BOUNDS FOR CYCLIC JOINS

In this section, we review two lower bounds for cyclic joins, one is an output-sensitive lower bound for the triangle join $\mathcal{Q}_\triangle = R_1(B, C) \bowtie R_2(A, C) \bowtie R_3(A, B)$ [20], and the other is an worst-case optimal lower bound for the box-minus join $\mathcal{Q}_\boxminus = R_1(A, B, C) \bowtie R_2(D, E, F) \bowtie R_3(A, D) \bowtie R_4(B, E) \bowtie R_5(C, F)$ [18], both of which verify that cyclic joins are inherently more difficult than acyclic joins. The high-level idea of these lower bound proofs is still resorted to the counting argument. It first show how to construct an probabilistic instance such that it will have a bounded $J(L)$, the maximum number of join results a server can produce, if it loads at most

L tuples from each relation. Then setting $p \cdot J(L) = \Omega(|\mathcal{Q}(\mathcal{R})|)$ yields a lower bound on L. Any attempts in lowering this bound further would break the counting argument.

7.1 A lower bound for Triangle Join

For $\mathcal{Q}_\triangle$, a worst-case lower bound of $\Omega(\frac{\text{IN}}{p^{2/3}})$ is known, by the counting argument. However, if OUT is also used as a parameter, this argument only leads to a lower bound of $\Omega((\frac{\text{OUT}}{p})^{\frac{3}{2}})$. This lower bound has been improved to $\Omega(\min\{\frac{\text{IN}}{p} + \frac{\text{OUT}}{p \log N}, \frac{\text{IN}}{p^{2/3}}\})$. The proof given in [20], is quite technical, but the intuition is simple: When $\text{OUT} = \Theta(\text{IN}^{\frac{3}{2}})$, the triangles are "dense" enough, so a server can achieve the maximum efficiency and emit $\Theta(L^{\frac{3}{2}})$ triangles. However, for small OUT, we can construct an instance in which the triangles are "sparse" so that a server cannot be as efficient. In fact, an instance constructed randomly (in a certain way) would have this property with high probability. This lower bound has the following consequences:

When $\text{OUT} \geq \text{IN} \cdot p^{1/3}$ for some constant c, the lower bound becomes $\tilde{\Omega}(\frac{\text{IN}}{p^{2/3}})$, which means that the worst-case optimal algorithm of [26] is actually also output-optimal in this parameter range. Finding $\tilde{\Omega}(\text{IN} \cdot p^{1/3})$ triangles is as difficult as finding $\Theta(\text{IN}^{3/2})$ triangles.

When $\text{IN} \leq \text{OUT} \leq \text{IN} \cdot p^{1/3}$, the lower bound becomes $\tilde{\Omega}(\frac{\text{OUT}}{p})$ while we do not have a matching upper bound yet. Nevertheless, this already exhibits a separation from acyclic joins, which can be done with load $O(\frac{\sqrt{\text{IN} \cdot \text{OUT}}}{p})$, with a gap being at least $\tilde{\Omega}(\sqrt{\frac{\text{OUT}}{\text{IN}}})$.

7.2 A lower bound for Box-minus Join

When studying the worst-case optimal algorithms for cyclic joins, it is surprisingly observed that $O(\frac{\text{IN}}{p^{1/\rho^*}})$ is not necessarily a correct target for multi-round worst-case optimal join algorithms [18]. An open question posed in [29, 24] was answered: For $\mathcal{Q}_\boxminus$, whether there exists a better upper bound than $\widetilde{O}(\frac{\text{IN}}{p^{1/3}})$, or a better lower bound than $\Omega(\frac{\text{IN}}{p^{1/2}})$? Note that $\mathcal{Q}_\boxminus$ has optimal fractional edge covering number $\rho^* = 2$ and optimal fractional edge packing number $\tau^* = 3$. [18] proves a higher lower bound $\Omega(\frac{\text{IN}}{p^{1/3}})$ for $\mathcal{Q}_\boxminus$.

The intuition for proving this lower bound is that a probabilistic instance could be shown when there are $\Theta(\text{IN}^2)$ join results, a server cannot be as efficient since the input instance is "sparse" enough. For the remaining cyclic joins (except LW join and graph join), there are no other lower bounds. Meanwhile, the existing algorithm [26] can compute it in a single round with load $\widetilde{O}(\frac{\text{IN}}{p^{1/3}})$[4], which is already worst-case optimal implied by this new lower bound.

8. CONCLUSION

In this article, we have surveyed recent results on computing join and join-aggregate queries in the MPC model. While most of them are theoretical in nature, experimental results have been presented in [12, 22], showing that some of the algorithmic ideas can lead to practical performance gains with certain engineering efforts. We conclude by summarizing several key results and posing some open questions:

- The instance-optimality can be achieved if and only if the join query is r-hierarchical.
- Beyond r-hierarchical joins, the output-optimality has only been achieved on acyclic joins if the output size is small, by an output-sensitive algorithm. Note that this algorithm is not always optimal, at least when the output size is large. However, there is no result on the output-optimal upper bound for cyclic joins in the MPC model, even for the triangle join. On the other hand, a lower bound for the triangle join indicates an inherent gap between the acyclic joins and cyclic joins in terms of the dependence on OUT.
- Worst-case optimal algorithms with load $O(\frac{\text{IN}}{p^{1/\rho^*}})$ have been discovered for acyclic joins and some specific class of cyclic joins (graph join and LW join). On the other hand, a recent edge-packing lower bound for the box-minus join $\mathcal{Q}_\boxminus$ shows that $O(\frac{\text{IN}}{p^{1/\rho^*}})$ is not achievable for all queries. It is thus an intriguing question to determine the worst-case complexity for the remaining cyclic joins. Would it be $\Omega(\text{IN}/p^{1/\max\{\rho^*,\tau^*\}})$?
- For join-aggregegate query, if it is free-connex, it can be transformed into a full join query through primitive operations and then enjoy all the results for full-join queries. For non-free-connex queries, the output-optimal algorithm has been achieved only for the matrix multiplication query. For other non-free connex queries, some output-sensitive algorithms have been developed, but without any non-trivial lower bound.

9. REFERENCES

[1] F. Afrati, M. Joglekar, C. Ré, S. Salihoglu, and J. D. Ullman. GYM: A multiround join algorithm in MapReduce. In *Proc. International Conference on Database Theory*, 2017.

[2] F. N. Afrati and J. D. Ullman. Optimizing multiway joins in a map-reduce environment. *IEEE Transactions on Knowledge and Data Engineering*, 23(9):1282–1298, 2011.

[3] R. R. Amossen and R. Pagh. Faster join-projects and sparse matrix multiplications. In *Proceedings of the 12th International Conference on Database Theory*, pages 121–126. ACM, 2009.

[4] A. Atserias, M. Grohe, and D. Marx. Size bounds and query plans for relational joins. *SIAM Journal on Computing*, 42(4):1737–1767, 2013.

[4] $\mathcal{Q}_\boxminus$ has optimal fractional edge quasi-packing number as 3.

[5] G. Bagan. *Algorithmes et complexité des problèmes d'énumération pour l'évaluation de requêtes logiques*. PhD thesis, Université de Caen, 2009.

[6] G. Bagan, A. Durand, and E. Grandjean. On acyclic conjunctive queries and constant delay enumeration. In *International Workshop on Computer Science Logic*, pages 208–222. Springer, 2007.

[7] N. Bakibayev, T. Kocisky, D. Olteanu, and J. Zavodny. Aggregation and ordering in factorised databases. In *Proc. International Conference on Very Large Data Bases*, 2013.

[8] P. Beame, P. Koutris, and D. Suciu. Skew in parallel query processing. In *Proc. ACM Symposium on Principles of Database Systems*, 2014.

[9] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. *Journal of the ACM (JACM)*, 64(6):40, 2017.

[10] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM (JACM)*, 30(3):479–513, 1983.

[11] A. Björklund, R. Pagh, V. V. Williams, and U. Zwick. Listing triangles. In *International Colloquium on Automata, Languages, and Programming*, pages 223–234. Springer, 2014.

[12] S. Chu, M. Balazinska, and D. Suciu. From theory to practice: Efficient join query evaluation in a parallel database system. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2015.

[13] E. Cohen. Estimating the size of the transitive closure in linear time. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 190–200. IEEE, 1994.

[14] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007.

[15] M. T. Goodrich. Communication-efficient parallel sorting. *SIAM Journal on Computing*, 29(2):416–432, 1999.

[16] M. T. Goodrich, N. Sitchinava, and Q. Zhang. Sorting, searching and simulation in the mapreduce framework. In *Proc. International Symposium on Algorithms and Computation*, 2011.

[17] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *Proc. ACM Symposium on Principles of Database Systems*, 2007.

[18] X. Hu. Cover or pack: New upper and lower bounds for massively parallel joins, `https://users.cs.duke.edu/~xh102/cover.pdf`. Technical report, Duke University, 2020.

[19] X. Hu and K. Yi. Towards a worst-case I/O-optimal algorithm for acyclic joins. In *Proc. ACM Symposium on Principles of Database Systems*, 2016.

[20] X. Hu and K. Yi. Instance and output optimal parallel algorithms for acyclic joins. In *Proc. ACM Symposium on Principles of Database Systems*, 2019.

[21] X. Hu and K. Yi. Parallel algorithms for sparse matrix multiplication and join-aggregate queries. In *Proc. ACM Symposium on Principles of Database Systems*, 2020.

[22] X. Hu, K. Yi, and Y. Tao. Output-optimal massively parallel algorithms for similarity joins. *ACM Transactions on Database Systems*, 44(2):6, 2019.

[23] M. R. Joglekar, R. Puttagunta, and C. Ré. AJAR: Aggregations and joins over annotated relations. In *Proc. ACM Symposium on Principles of Database Systems*, 2016.

[24] B. Ketsman and D. Suciu. A worst-case optimal multi-round algorithm for parallel computation of conjunctive queries. In *Proc. ACM Symposium on Principles of Database Systems*, 2017.

[25] M. A. Khamis, H. Q. Ngo, and A. Rudra. FAQ: Questions asked frequently. In *Proc. ACM Symposium on Principles of Database Systems*, 2016.

[26] P. Koutris, P. Beame, and D. Suciu. Worst-case optimal algorithms for parallel query processing. In *Proc. International Conference on Database Theory*, 2016.

[27] P. Koutris, S. Salihoglu, and D. Suciu. *Algorithmic Aspects of Parallel Data Processing*. Now Publishers, 2018.

[28] P. Koutris and D. Suciu. Parallel evaluation of conjunctive queries. In *Proc. ACM Symposium on Principles of Database Systems*, 2011.

[29] P. Koutris and D. Suciu. A guide to formal analysis of join processing in massively parallel systems. *ACM SIGMOD Record*, 45(4):18–27, 2017.

[30] H. Q. Ngo, E. Porat, C. Ré, and A. Rudra. Worst-case optimal join algorithms. In *Proc. ACM Symposium on Principles of Database Systems*, pages 37–48, 2012.

[31] Y. Tao. A simple parallel algorithm for natural joins on binary relations. In *23rd International Conference on Database Theory*, 2020.

[32] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[33] T. Veldhuizen. Leapfrog triejoin: A simple, worst-case optimal join algorithm. In *Proc. International Conference on Database Theory*, 2014.

[34] M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. International Conference on Very Large Data Bases*, pages 82–94, 1981.

Data Preparation: A Survey of Commercial Tools

Mazhar Hameed
Hasso Plattner Institute
University of Potsdam, Germany
mazhar.hameed@hpi.de

Felix Naumann
Hasso Plattner Institute
University of Potsdam, Germany
felix.naumann@hpi.de

ABSTRACT

Raw data are often messy: they follow different encodings, records are not well structured, values do not adhere to patterns, etc. Such data are in general not fit to be ingested by downstream applications, such as data analytics tools, or even by data management systems. The act of obtaining information from raw data relies on some *data preparation* process. Data preparation is integral to advanced data analysis and data management, not only for data science but for any data-driven applications. Existing data preparation tools are operational and useful, but there is still room for improvement and optimization. With increasing data volume and its messy nature, the demand for prepared data increases day by day.

To cater to this demand, companies and researchers are developing techniques and tools for data preparation. To better understand the available data preparation systems, we have conducted a survey to investigate (1) prominent data preparation tools, (2) distinctive tool features, (3) the need for preliminary data processing even for these tools and, (4) features and abilities that are still lacking. We conclude with an argument in support of automatic and intelligent data preparation beyond traditional and simplistic techniques.

Keywords

data quality, data cleaning, data wrangling

1. THE NEED FOR DATA PREPARATION

Raw data appears in many situations: logs, sensor output, government data, medical research data, climate data, geospatial data, etc. It accumulates in many places, such as file systems, data lakes or online repositories. In typical scenarios, raw data from various sources is accrued without any standardized formats or structure and with no specific target use-case; thus, it can appear messy, contain invalid characters, use different encodings, lack necessary columns, contain unwanted rows, have missing values, not follow valid patterns, etc.

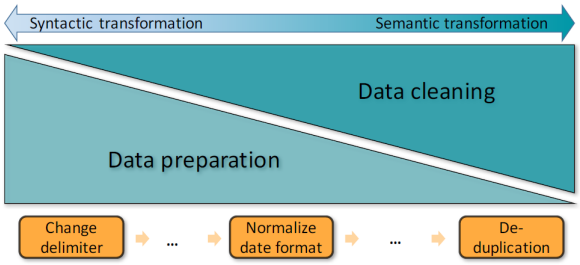

Figure 1: Data preparation vs. data cleaning

We define data preparation as the set of preprocessing operations performed in early stages of a data processing pipeline, i.e., data transformations at the structural and syntactical levels. We provide many examples of such transformations throughout the article. In contrast, data cleaning concerns subsequent data transformations and corrections at the semantic level (Figure 1).

One example scenario in need of data preparation technology are data lakes to store heterogeneous raw data [20]. They can turn into vast repositories or collections of unstructured, semi-structured, unformulated, messy, and unclean data. The large volumes of data in data lakes are compelling and can generate valuable information, provided they are thoroughly pre-processed, cleaned, and prepared [29]. With the ever-increasing amount of raw data, the need for data preparation has become more apparent.

Preparing data yields many advantages, such as prompt error detection, improved analytics, improved data quality, enhanced scalability, accelerated data usage and more easy data collaboration [9, 12].

1.1 The data-to-application process

To understand data processing in the data to application life-cycle, it is important to identify the phases required to create data that is valid for the consuming application. Data creation occurs typi-

cally in raw format, possibly to be stored in data lakes. Before these raw data are sent to applications it is crucial to enhance its structure and, if needed, its content: To make data readable and machine understandable, a number of steps are typically performed, such as (1) data exploration [5, 14, 26, 27], (2) data collection [18, 31], (3) data profiling [8, 21, 22], (4) data preparation [3, 12, 25], (5) data integration [7, 17, 28], and (6) data cleaning [2, 4, 24] in various orders and iterations.

These aforementioned steps are applied to originally 'raw data', before they are sent to the main application for further processing. In our research focus, and based on evidence from noted surveys, a critical and important step is data preparation. Trifacta's data preparation study shows that 72% of respondents indicated that data preparation by data users is critical, while 88% indicated at least its importance, and only 4% indicated that it is not important for the user [1]. Data scientists spend approximately 80% of the time on preparing the data and about 20% on actual model implementation and deployment [12,23,29]. Clearly, these numbers cannot be reduced to 0%, due to the semantic difficulties of understanding and interpreting data. However, the time spent on data preparation can be decreased to a significant amount using sophisticated data preparation techniques, and, in turn, data scientists attain more time for model implementation and deployment.

Keeping in mind the importance and impact of data preparation, developers and researchers have contributed various techniques that facilitate the data preparation process [9–11, 15, 29, 32].

To address the aforementioned challenges and the general importance of data preparation, many tools have been designed by not only the industry, but also by research and academia to address varying use cases. In light of that, we have surveyed commercial data preparation tools to analyze available features and methods. Our survey is not comparative, nor do we explicitly evaluate the individual tools. Rather, we want to show the current state of the art and identify research and development opportunities for the data preparation community at large.

1.2 A preparation example

Let us explain, with the help of an example, the usefulness and importance of data preparation. For instance a data scientist has been handed a csv-formatted file as shown in Figure 2a, from a government data portal[1] to examine and answer how much time each employee is spending on training besides their 9-5 job. Although the csv format defines rows of records, once the file is opened it is clear that there is no coherent relational structure: data is laid out in a somewhat human-readable format, making automated analysis impossible. More-

Q.1. Please think about any time away from your day-to-day job that you spend in training. Is your training ...?				
Base : All apprentices				
		Wave		
	Wtd Total (a)	Wave 1 (b)	Wave 2 (c)	Wave 3 (d)
Unweighted Total	4979	1667	1667	1645
Weighted Total	4979	1548	1715	1716
Effective Base	3283	1112	1136	1049
Based at a college only	567	206	165	196
	11%cf	13%ac	10%	11%
	j			
Based at a training provider only	232	68	78	86
	5%i	4%	5%	5%
Within your workplace only	1732	486	640	606
	35%be	31%	37%ab	35%
	ghikt			
	w			
Based within your workplace and at a college or training provider	2440	786	828	826
	49%fj	51%	48%	48%
	osv			
Don't know	8	2	4	2
	*	*	*	*
Fieldwork dates : 17 February 2009 - 31 July 2009				
Respondent Type : Learners				
Source : Ipsos MORI (J34262)				
*=Less than 0.5 %				
Tested (5% risk level) - a/b/c/d - a/e/f - a/g/h/i/j - a/k/l/m/n/o/p - a/r/s/t/u/v/w/x				
* small base	** very small base (under 30) ineligible for sig testing			

(a) Unprepared data

(Category)	Unweighted Total	Weighted Total	Effective Base	Based at a college only	(Percentage)
Wtd Total (a)	4979	4979	3283	567	11.00%
Wave 1 (b)	1667	1548	1112	206	13.00%
Wave 2 (c)	1667	1715	1136	165	10.00%
Wave 3 (d)	1645	1716	1049	196	11.00%
Male (e)	2901	2689	2045	394	15.00%
Female (f)	2078	2290	1306	173	8.00%
16-18 (g)	2175	1195	1703	181	15.00%
16-19 (h)	2936	2175	1833	321	15.00%
19-24 (i)	2149	2738	1651	345	13.00%
25+ (j)	655	1046	503	41	4.00%
Entry level/ Level 1 (k)	2192	1974	1404	240	12.00%
Level 2 (l)	2109	2224	1448	248	11.00%
Level 3 (m)	272	349	212	37	11.00%
Level 4 or above (n)	15	29	10	0	0.00%
No qualification (o)	368	379	215	38	10.00%
No level / don't know (p)	23	23	19	4	16.00%
Level 1 and entry (r)	14	15	11	0	2.00%
Level 2 (s)	2839	2592	1745	286	11.00%
Level 3 (t)	2121	2366	1525	279	12.00%
Level 4 or 5 or higher (u)	3	2	3	1	34.00%
Level 2 or below (v)	2853	2607	1756	286	11.00%
Level 3 or higher (w)	2124	2369	1528	280	12.00%
No level / don't know (x)	2	3	2	1	23.00%
Unwtd Total	4979	4979	4979	631	13.00%

(b) Prepared data

Figure 2: Example of data preparation

[1] http://webarchive.nationalarchives.gov.uk/+/http://www.bis.gov.uk/assets/biscore/further-education-skills/docs/n/11-708-data-nlss-2009.csv (February, 2019)

over in this case, almost 1,000 tables are stacked one below each other (not shown), interleaved by metadata information in the form of preambles and comments that more often than not repeat themselves without meaningful addition. On top of that, inside the actual data tables, alphanumeric characters appear in what seem to be other-wise numeric records, and there are apparently inconsistent representations for zeros/null values (e.g., '*','-' or empty cell). The data scientist might perform the sequence of steps listed below to each file making their data more comprehensive, prepared, structured and machine-readable as depicted in Figure 2b., before feeding them to the analysis tool. In this way, the data scientist avoids the cumbersome and time-consuming manual execution of these tasks and could use this sequence again for future use cases and tasks.

1. Split the file to isolate one data table at a time. For each obtained file:
2. Remove preamble and comment rows.
3. Unify null-value representations.
4. Remove rows with no meaningful information, e.g., empty rows or rows with only null-values.
5. Clean numeric data rows by removing special characters.
6. Fill missing values, e.g., by value imputation or using functional dependencies.
7. Transpose table.
8. Add missing header.

It is evident from the aforementioned example that with the help of various data preparation steps and tools we were able to target messy data and convert it into clean and machine-readable data, highlighting the significance of data preparation in the market for both industry and academia. The application of simple data preparation tasks on raw data files improves their usability, readability, interpretability, etc. Software vendors have identified the importance and need of data preparation and offer dedicated tools. To provide a snapshot of the current state of development, we have conducted a detailed survey of seven commercial data preparation tools. Our paper makes the following contributions:

1. **Organisation**: We propose six broad categories of data preparation and identify 40 common data preparation steps, which we classify into those categories (Section 2).
2. **Documentation**: We validate the availability of these features and broader categories for seven selected tools and document them in a feature matrix (Section 3).
3. **Evaluation**: We evaluate the selected features of surveyed tools to identify whether the tool offers the stated functionalities or not (Section 4).
4. **Recommendation**: We identify shortcomings of commercial data preparation tools in general and encourage researchers to explore further in the field of data preparation (Section 5).

2. DATA PREPARATION TASKS

Data preparation is not a single step process. Rather, it usually comprises many individual preparation steps, implemented by what we call *preparators*, and which we have organized anew into six broader categories, defined here.

Data discovery is the process of analyzing and collecting data from different sources, for instance to match data patterns, find missing data, and locate outliers.

Data validation comprises rules and constraints to inspect the data, for instance for correctness, completeness, and other data quality constraints.

Data structuring encompasses tasks for the creation, representation and structuring of information. Examples include updating schema, detecting & changing encoding and, transform data by example [13].

Data enrichment adds value or supplementary information to existing data from separate sources [30]. Typically, it involves augmenting existing data with new or derived data values using data lookups, primary key generation, and inserting metadata.

Data filtering generates a subset of the data under consideration, facilitating manual inspection and removing irregular data rows or values. Examples include extracting text parts, and keeping or deleting filtered rows.

Data cleaning refers to removal, addition, or replacement of less accurate or inaccurate data values with more suitable, accurate or representative values. Typical examples are deduplication, fill missing values, and removing whitespace.

Despite our definition, which distinguishes data preparation and cleaning, we include data cleaning steps here as well, as most data preparation tools also venture into this area.

Our set of 40 individual preparators is shown and categorized in Table 2, which is introduced in the next section.

3. PREPARATION TOOLS AND TASKS

Data preparation tools are vital to any data preparation process. They usually provide implementations of various preparators and a frontend to sequentially apply preparations or to specify data preparation pipelines. The flexibility, robustness and intelligence of these tools contribute significantly towards the data analysis and data management tasks. In this section, we discuss in detail a selection of tools for our research study that are supported by supplementary documentation for experimentation and guidance. Section 3.1 discusses the selected data preparation tools (see Table 1 for an overview) and Section 3.2 highlights our approach to populate the preparator matrix (Table 2), organized by data preparator categories with selected preparation tasks.

3.1 Available data preparation tools

In general, data preparation is an expensive and time-consuming activity, especially without automated and mature data preparation tools. Traditionally, data scientists write specific preparation scripts to accomplish the project-specific goals. Recently, the market has answered to some of the general needs of data preparation by providing commercial preparation tools that can lower the burden of data scientists.

To better understand commercial tools and their capabilities, we initiated our study with a discovery phase. We collected notable commercial data preparation tools gathered from business reports and analyses, company portals, and online demonstration videos. Our preliminary investigation resulted in 42 initial commercial tools (shown in Table 3 in the appendix), which we then examined for the extent of their *data preparation* capabilities.

Not all collected tools were dedicated to data preparation. Rather, many tools were primarily targeting data visualization, data analysis, and business intelligence applications, with only some added data preparation features. To focus on the topic of our survey, we established, necessarily soft, criteria for tool selection.

- Domain specificity: tools that specifically address the data preparation task.
- Comprehensiveness: the extent and sophistication to which tools adequately covered preparation features listed in Section 2.
- Guides and documentation: the availability of proper documentation for the tools, i.e., useful, up-to-date documentation with listings of features and how-to guides
- Trial availability: the availability of a trial version, giving us the opportunity to test the tools and validate their features
- GUI: the availability of a comprehensive and intuitive graphical user interface to select and apply preparations.
- Customer assistance: compliant support teams that assisted users with generic and specific tool queries, when needed.

Finally, we selected seven tools for detailed investigation (shown in Table 1). We now discuss (in alphabetical order) the seven qualifying tools for our data preparation survey. In the appendix we have collected additional functional and non-functional features that are not specific data preparation tasks.

Altair Monarch Data Preparation, called Datawatch until the company's merger with Altair, provides common data preparators for structured data but also transforms tables from within PDF and text files to tabular data. The extracted files from Altair's table extractor feature can be used independently as a table or they can be merged with other tables or files using a variety of join and union operations.

Paxata Self-Service Data Preparation offers many features to organize and prepare structured data and also deals efficiently with semi-structured data. In addition to common data preparation features, Paxata offers so-called data filtergrams, which allow various visual interactions to perform filter operations on data, such as, text filtergrams, numeric filtergrams, Boolean filtergrams, and source filtergrams. The user experience is emphasized in this tool, which is designed to support also non-experts.

SAP Agile Data Preparation runs on top of SAP's HANA database system. It offers many common data preparators with some specific system features, such as Schedule Snapshot, which allows the user to take periodic snapshots and retrieve data from a remote source on demand. It offers interactive suggestions to help users navigate and prepare data efficiently. Multi-user access allows to prepare data in collaboration.

SAS Data Preparation is part of SAS Viya System Management, which runs its operations with

Table 1: Selected data preparation tools

Tool name	URL
Altair Monarch Data Preparation	https://www.datawatch.com/in-action/monarch-draft/
Paxata Self Service Data Preparation	https://www.paxata.com/self-service-data-prep/
SAP Agile Data Preparation	https://www.sap.com/germany/products/data-preparation.html
SAS Data Preparation	https://www.sas.com/en_us/software/data-preparation.html
Tableau Prep	https://www.tableau.com/products/prep
Talend Data Preparation	https://www.talend.com/products/data-preparation/
Trifacta Wrangler	https://www.trifacta.com/products/wrangler-editions/

distributed in-memory processing. In addition to common features, SAS offers code-based transformations for users to write and share custom code to transform data, supporting re-usability of preparation pipelines.

Tableau Prep implements a workflow approach to organize and prepare messy data. With its interactive interface and workspace plans, users have the freedom to perform multiple operations simultaneously. Tableau prep comprises two parts, namely Tableau Prep Builder, which is designed to develop so-called flows, manage data and apply operations on data, and Tableau Prep Conductor to share, schedule, and monitor the flows.

Talend Data Preparation offers many specific data preparation functionalities tailored to the task at hand. For instance, for data cleaning, different functions exist for cleaning numeric data values, strings and date inputs. One of its main features is "selective sampling" of data for insights and operations that can be later deployed on the entire dataset. Talend actively contributes to solving system-level challenges, e.g., one of its intelligent system features is pipeline automation, to save and reuse data preparation tasks or steps.

Trifacta Wrangler prepares data using multiple data preparation functions and intelligently predicts patterns to provide suggestions that help users to transform data. Apart from common preparation tasks, it offers additional interesting features, such as primary key generation, transform data by example, and permitted character checks. Wrangler uses regular expressions for most of its pattern-based features. The significance of Wrangler preparators is their degree of sophistication. For example, the locate outlier not only identifies the outliers, but also plots a histogram of the entire column. The tool was spun out of the Wrangler project [16].

3.2 Preparator matrix

Table 2 provides a feature matrix showing which preparator is supported by which tool in each of the six categories. We evaluated each of these preparators on three datasets downloaded from public data repositories: (i) Kaggle – 120 years of Olympic history (athletes and results)[2], (ii) IMDb – data about movies[3], and (iii) UK government web archive, as mentioned in Section 1.2.

The population of this preparator matrix was not a trivial task. Initially, we analyzed the tool's documentation to gather all available preparators. We then downloaded trial versions of all tools and (generously) evaluated for each of the seven tools and each of the 40 preparators whether they offer this functionality. Section 4 describes in more detail how we populated the feature matrix. All tools and their corresponding documented preparators were gathered before September 2, 2019.

The basic functionality of most preparators is self-explanatory by their name – their precise implementation and parameterization might differ from tool to tool and it would be beyond the scope of this article to describe each. Instead, we have selected three exemplary preparators to illustrate their function and the intricacies involved in even simple data preparation tasks. We use the same three preparators in Section 4 to highlight some capabilities of individual tools.

Keep or Delete Filtered Rows: Filtering operations customize data views and provide output based on specified predicates, for instance to filter data that can be deleted, extracted or altered for further analysis. In its basic form, filtering allows simple predicates, akin to SQL conditions. A more intelligent approach would be to use a richer language, such as regular expressions, for filtering.

Value Standardization: A typical preparation operation is to change the values of a column to follow some standard. That standard could be a frequent pattern derived from the data itself or taken from an external authority. A more sophisticated preparator could help in automatically detecting relevant data clusters for standardization. Popular techniques include fuzzy matching for clustering to

[2] https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results
[3] ftp://ftp.fu-berlin.de/pub/misc/movies/database/frozendata/

provide a better representation of data.

Split Column: Messy data can include values that consist of multiple atomic parts. Split column can separate (split) data into multiple columns based on defined criteria (e.g., split after ',' or at last whitespace in string, etc). A more sophisticated preparator could identify split column cases by using existing patterns in data, and be able to handle splits into more than two columns.

4. EVALUATION OF EXISTING TOOLS

To better explain how we evaluated the preparators, we provide an example for each of the three preparators discussed in the previous section. In general, even the simplest versions of the respective preparators earned the tool a checkmark in our matrix (Table 2). More sophisticated versions could incorporate preparators that intelligently detect relevant problems and actively provide suggestions for their configuration, e.g., suitable regular expressions or standard formats.

Keep or Delete Filtered Rows: Data filtering techniques improve data quality using predefined criteria, such as removing records that contain empty values or that do not conform to some user-defined pattern. The majority of data preparation tools offers various types of filters. For instance, Talend Data Preparation offers filters based on patterns using pre-defined syntactic data types:

EXAMPLE 1. *Using pattern filtering, a user might want to keep only official email addresses. Using Talend's syntax, corresponding patterns might be:*

`[word]@ibm.[word],[char].[word]@ibm.[word]`

Thus, private addresses such as `bob1992@gmail.com` *or* `alice25@yahoo.com` *would be filtered, while* `a.peter@ibm.com` *would be retained.*

Value Standardization: A typical step in case of heterogeneously formatted values is standardization using patterns, e.g., phone number patterns, datetime patterns, patterns by example, etc. For instance, Trifacta Wrangler provides suggestions for applicable patterns and transforms data to the suggested or a selected standard.

Also, in case of different representations of the same real-world value within a column, value standardization groups those values and transforms them to a single, common representation.

EXAMPLE 2. *Trifacta might group records with city values* `NY`, `NYC`, `New York` *and* `New York City` *and standardize all occurrences to* `New York City`. *Alternatively, users can review the cluster and manually choose the correct standard value.*

Split Column: Multi-valued columns reduce flexibility in handling data (and also their readability). Split column splits such columns based on some criterion. For instance, SAS Data Preparation implements this technique in several ways, e.g., split based *on*, *before*, or *after* a delimiter, on a fixed length, and "quick split", which intelligently identifies a split criterion.

EXAMPLE 3. *Using comma as a delimiter, the user wants to split the location column (and implicitly trim accrued whitespace). In addition, the user specified headers for the output columns. As can be seen in the example, due to a missing value in the original data, the value "USA" is misplaced; a later validation step might identify this error.*

Input:

Location	...
Melbourne, Victoria, Australia	
San Francisco, USA	
Potsdam, Brandenburg, Germany	

Output:

City	***State***	***Country***	...
Melbourne	*Victoria*	*Australia*	
San Francisco	*USA*		
Potsdam	*Brandenburg*	*Germany*	

5. CHALLENGES AND FUTURE WORK

Some of the most prominent challenges that we came across during our research and survey are the following:

Dataset pre-processing: Interestingly, despite being data preparation tools, all tools that we have surveyed and explored require a pre-prepared or cleaned dataset as their input. For example, if the file had comment-lines, additional header or footer information, or poorly placed quotation marks, it was misinterpreted and loaded improperly. In fact, most tools make the following broad assumptions:

- Single table file (no multi-table files)
- Specific file encoding
- No preambles, comments, footnotes, etc.
- No intermediate headers
- Specific line-ending symbol
- Homogeneous delimiters
- Homogeneous escape symbols
- Same number of fields per row

Table 2: Data preparation tool feature matrix

Categories	Available features	Data preparation tools						
		Altair	Paxata	SAP	SAS	Tableau	Talend	Trifacta
Data discovery	Locate missing values (nulls)	✓	✓	✓	✓	✓	✓	✓
	Locate outliers		✓		✓			✓
	Search by pattern	✓	✓	✓	✓	✓	✓	✓
	Sort data	✓	✓	✓	✓	✓	✓	✓
Data validation	Compare values (selection and join)	✓	✓	✓		✓	✓	✓
	Check data range	✓	✓	✓		✓	✓	✓
	Check permitted characters							✓
	Check column uniqueness	✓	✓	✓		✓	✓	✓
	Find type-mismatched data		✓	✓		✓	✓	✓
	Find data-mismatched datatypes		✓				✓	✓
Data structuring	Change column data type	✓	✓	✓	✓	✓	✓	✓
	Delete column	✓	✓	✓	✓	✓	✓	✓
	Detect & change encoding						✓	✓
	Pivot / unpivot	✓	✓	✓		✓		✓
	Rename column	✓	✓	✓	✓	✓	✓	✓
	Split column	✓	✓	✓	✓	✓	✓	✓
	Transform by example [13]						✓	✓
Data enrichment	Assign semantic data type				✓	✓	✓	
	Calculate column using expressions	✓	✓	✓	✓	✓	✓	✓
	Discover & merge external data	✓	✓	✓			✓	✓
	Duplicate column	✓	✓	✓		✓	✓	✓
	Generate primary key column			✓				✓
	Join & union	✓	✓	✓	✓	✓	✓	✓
	Merge columns	✓		✓		✓	✓	✓
	Normalize numeric values	✓	✓	✓	✓	✓	✓	✓
Data filtering	Delete/keep filtered rows	✓	✓	✓	✓	✓	✓	✓
	Delete empty and invalid rows	✓	✓	✓	✓	✓	✓	✓
	Extract value parts	✓			✓		✓	✓
	Filter with regular expressions							✓
Data cleaning	Change date & time format	✓	✓	✓	✓	✓	✓	✓
	Change letter case	✓	✓	✓	✓	✓	✓	✓
	Change number format	✓	✓	✓	✓	✓	✓	✓
	Deduplicate data	✓	✓	✓	✓		✓	✓
	Delete by pattern	✓	✓		✓	✓	✓	✓
	Edit & replace cell data	✓	✓	✓	✓	✓	✓	✓
	Fill empty cells	✓	✓				✓	✓
	Remove extra whitespace	✓	✓	✓	✓	✓	✓	✓
	Remove diacritics			✓				
	Standardize strings by pattern		✓	✓	✓	✓	✓	✓
	Standardize values in clusters		✓	✓	✓	✓	✓	✓

- Relational data (no nested or graph-structured data, such as XML, JSON or RDF)

Some of the aforementioned assumptions pose interesting research problems in themselves, which have been addressed in isolation by other researchers, such as detecting tables in complex spreadsheets [6] or converting HTML tables to relations [19].

User expertise needed: Another challenge we experienced was the need of the combination of domain knowledge and IT-knowledge for tool usability. Most tools require the user to be an expert in the dataset domain and have prior knowledge and understanding of the datasets and of the data preparation goal.

Moreover, beyond simple predicates, most tools allow the use of regular expressions to match, split, or delete data. A typical domain-expert cannot be expected to formulate often intricate regular expressions.

Lack of intelligent solutions: All surveyed tools offer useful data preparation functions. However, most tools and most preparators lack intelligent solutions for more automated data preparation tasks. For example Deduplicate data removes duplicate records from a source. The surveyed tools deduplicate data only on exact match conditions, a more sophisticated version would involve deduplication based on similarity measures. Another problem for many tools is column heterogeneity, i.e., if columns contain data in multiple formats. Currently, users need to manually filter those different groups and prepare them separately. An automatic homogenization would be helpful but also poses a challenging

research problem.

Unstructured data: The scope of our survey is that of preparing structured data. However, many datasets include some textual component, such as product descriptions, plot synopses, etc. Such textual data can also benefit from basic preparation steps, such as stopword-removal, lemmatization, or sentence breaking, to then e.g. perform named entity extraction or sentiment analysis.

One outlook is to include such capabilities in the existing tools for structured data preparation. Another is to develop a dedicated framework and toolset for the case of unstructured data preparation (or text preparation), similar to the tools survey in this article.

Preparation pipelining: Data preparation is not a one-step process. Rather, it involves many subsequent steps, organized in a preparation pipeline to gradually transform a dataset towards the desired output.

Creating and managing pipelines yields many system-level challenges and opportunities. For instance, preparation suggestion, pipeline adaption, and pipeline optimization, that need to be addressed accordingly. Such systematic data preparation can benefit from a comprehensive and well-defined yet extensible set of operators. By incorporating the ability to create and manage preparation pipelines, data preparation tools can be massively improved and generalized for more intelligent and self-service techniques. After a pipeline has been established, optimization and customization policies can be designed according to needs of the problem at hand or business use cases under considerations.

To summarize, existing tools already cover basic data preparation needs by implementing simple and obvious preparators. In few cases we observed more sophisticated abilities, such as automatic suggestion of patterns or even of preparators for the data at hand. All of these tools are excellent platforms for further development in several dimensions, as outlined above. In our opinion, the need for self-service data preparation and tool capabilities goes beyond current technology and we encourage research in this emerging field.

6. CONCLUSION

In this paper, we have discussed and surveyed major commercial tools for data preparation. We have gathered and organized their capabilities in the form of "preparators", organized in six categories.

As more and more data are produced there is more and more opportunity to create value by integrating and analyzing them. Thus, the need for data preparation and data cleaning grows: Data have many types of syntactic and semantic issues that can be bridged by careful automated or manual preparation and cleaning steps. Current technology is still far from enabling a fully automatic transformation of data from their raw form to a shape and quality that can be readily consumed by downstream applications. Commercial (and academic) tools provide good user-support and tooling for a wide range of preparation needs. Nevertheless, data preparation remains a largely manual task to be performed by data experts or by domain experts with data engineering skills.

Acknowledgments

This research was funded by the HPI research school on Data Science and Engineering.

7. REFERENCES

[1] Trifacta end user data preparation. `https://www.trifacta.com/wp-content/uploads/2018/02/End-User-Data-Preparation-Market-Study-2018.pdf`. Accessed: 2019-09-19.

[2] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. Detecting data errors: Where are we and what needs to be done? *PVLDB*, 9(12):993–1004, 2016.

[3] Gregorio Convertino and Andy Echenique. Self-service data preparation and analysis by business users: New needs, skills, and tools. In *Proceedings of the CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1075–1083. ACM, 2017.

[4] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F Ilyas, Mourad Ouzzani, and Nan Tang. Nadeef: a commodity data cleaning system. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 541–552. ACM, 2013.

[5] Yanlei Diao, Kyriaki Dimitriadou, Zhan Li, Wenzhao Liu, Olga Papaemmanouil, Kemi Peng, and Liping Peng. Aide: an automatic user navigation system for interactive data exploration. *PVLDB*, 8(12):1964–1967, 2015.

[6] Haoyu Dong, Shijie Liu, Shi Han, Zhouyu Fu, and Dongmei Zhang. TableSense: Spreadsheet table detection with convolutional neural networks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, volume 33, pages 69–76, 2019.

[7] Xin Luna Dong and Divesh Srivastava. Big data integration. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 1245–1248. IEEE, 2013.

[8] Jens Ehrlich, Mandy Roick, Lukas Schulze, Jakob Zwiener, Thorsten Papenbrock, and Felix Naumann. Holistic data profiling: Simultaneous discovery of various metadata. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 305–316, 2016.

[9] Florian Endel and Harald Piringer. Data wrangling: Making data useful again. *IFAC-PapersOnLine*, 48(1):111–112, 2015.

[10] Tim Furche, Georg Gottlob, Leonid Libkin, Giorgio Orsi, and Norman W Paton. Data wrangling for big data: Challenges and opportunities. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 473–478, 2016.

[11] Anders Haug, Frederik Zachariassen, and Dennis Van Liempd. The costs of poor data quality. *Journal of Industrial Engineering and Management (JIEM)*, 4(2):168–193, 2011.

[12] Joseph M Hellerstein, Jeffrey Heer, and Sean Kandel. Self-service data preparation: Research to practice. *IEEE Data Engineering Bulletin*, 41(2):23–34, 2018.

[13] Zhongjun Jin, Michael R Anderson, Michael Cafarella, and HV Jagadish. Foofah: Transforming data by example. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 683–698. ACM, 2017.

[14] Manas Joglekar, Hector Garcia-Molina, and Aditya G Parameswaran. Interactive data exploration with smart drill-down (extended version). *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, (1):1–1, 2017.

[15] Sean Kandel, Jeffrey Heer, Catherine Plaisant, Jessie Kennedy, Frank Van Ham, Nathalie Henry Riche, Chris Weaver, Bongshin Lee, Dominique Brodbeck, and Paolo Buono. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization*, 10(4):271–288, 2011.

[16] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the International Conference on Human Factors in Computing Systems (CHI)*, pages 3363–3372, 2011.

[17] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, pages 233–246. ACM, 2002.

[18] Yanying Li, Haipei Sun, Boxiang Dong, and Hui Wendy Wang. Cost-efficient data acquisition on online data marketplaces for correlation analysis. *PVLDB*, 12(4):362–375, 2018.

[19] George Nagy, Sharad Seth, and David W. Embley. End-to-end conversion of HTML tables for populating a relational database. In *Proceedings of the IAPR International Workshop on Document Analysis Systems*, pages 222–226, 2014.

[20] Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. Data lake management: challenges and opportunities. *PVLDB*, 12(12):1986–1989, 2019.

[21] Felix Naumann. Data profiling revisited. *SIGMOD Record*, 42(4):40–49, 2014.

[22] Thorsten Papenbrock, Tanja Bergmann, Moritz Finke, Jakob Zwiener, and Felix Naumann. Data profiling with Metanome. *PVLDB*, 8(12):1860–1863, 2015.

[23] Gil Press. Cleaning data: Most time-consuming, least enjoyable data science task. *Forbes*, March 2016.

[24] Vijayshankar Raman and Joseph M Hellerstein. Potter's wheel: An interactive data cleaning system. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2001.

[25] Tye Rattenbury, Joseph M Hellerstein, Jeffrey Heer, Sean Kandel, and Connor Carreras. *Principles of data wrangling: Practical techniques for data preparation*. O'Reilly Media, Inc., 2017.

[26] Thibault Sellam and Martin Kersten. Ziggy: Characterizing query results for data explorers. *PVLDB*, 9(13):1473–1476, 2016.

[27] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *PVLDB*, 10(4):457–468, 2016.

[28] Michael Stonebraker and Ihab F Ilyas. Data integration: The current status and the way forward. *IEEE Data Engineering Bulletin*, 41(2):3–9, 2018.

[29] Ignacio G Terrizzano, Peter M Schwarz, Mary Roth, and John E Colino. Data wrangling: The challenging journey from the wild to the

lake. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2015.

[30] Pei Wang, Yongjun He, Ryan Shea, Jiannan Wang, and Eugene Wu. Deeper: A data enrichment system powered by deep web. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 1801–1804. ACM, 2018.

[31] Susan C Weller and A Kimball Romney. *Systematic data collection*, volume 10. Sage publications, 1988.

[32] Shichao Zhang, Chengqi Zhang, and Qiang Yang. Data preparation for data mining. *Applied artificial intelligence*, 17(5-6):375–381, 2003.

Appendix

Our initial survey found 42 software tools that asserted some for of data preparation functionality. These tools are listed in Table 3. Section 3.1 describes our selection process to reach the seven tools (in bold) that we analyzed more closely.

In our survey of commercial tools, we came across many functional and non-functional system features that did not cater to our data preparation focus. Nonetheless, these features are important and interesting when explored and utilized. Thus, we have gathered them in Table 4.

Table 3: Discovered tools with asserted data preparation capabilities

Tool name	URL
Altair Monarch Data Preparation	https://www.datawatch.com/in-action/monarch-draft/
Alteryx Data Preparation	https://www.alteryx.com/solutions/analytics-need/data-preparation
BigGorilla Data Preparation	https://www.biggorilla.org/
Cambridge Semantics Anzo	https://www.cambridgesemantics.com/
Datameer	https://www.datameer.com/
EasyMorph Data Preparation and Automation	https://easymorph.com/
Erwin	https://erwin.com/
FICO	https://www.fico.com/
Google Cloud Data Prep by Trifacta	https://cloud.google.com/dataprep/
Hitachi-Pentaho Business Analytics	https://www.hitachivantara.com/en-us/products/data-management-analytics.html
IBM Data Refinery	https://www.ibm.com/cloud/data-refinery
INFOGIX	https://www.infogix.com/data3sixty/analyze/
Informatica Enterprise Data Preparation	https://www.informatica.com/products/data-catalog/enterprise-data-prep.html
Looker	https://looker.com/
Lore IO	https://www.getlore.io/
Microsoft Power BI	https://powerbi.microsoft.com/en-us/
MicroStrategy	https://www.microstrategy.com/us/product/analytics/data-visualization
Modak-nabu	https://modakanalytics.com/nabu.html
OpenRefine	http://openrefine.org/
Oracle Analytics Cloud	https://www.oracle.com/business-analytics/analytics-cloud.html
Paxata Self Service Data Preparation	https://www.paxata.com/self-service-data-prep/
Qlik Data Catalyst	https://www.qlik.com/us/products/qlik-data-catalyst
Quest Toad Data Point	https://www.quest.com/products/toad-data-point/
Rapid Insight	https://www.rapidinsight.com/solutions/data-preparation/
RapidMiner Turbo Prep	https://rapidminer.com/products/turbo-prep/
SAP Agile Data Preparation	https://www.sap.com/germany/products/data-preparation.html
SAS Data Preparation	https://www.sas.com/en_us/software/data-preparation.html
Smarten Advanced Data Discovery	https://www.smarten.com/self-serve-data-preparation.html
Solix Common Data Platform	https://www.solix.com/products/solix-common-data-platform/
Sparkflows	https://www.sparkflows.io/data-science
Tableau Prep	https://www.tableau.com/products/prep
Talend Data Preparation	https://www.talend.com/products/data-preparation/
Tamr	https://www.tamr.com/
Teradata Vantage	https://www.teradata.com/Products/Software/Vantage
TIBCO Spotfire Analytics	https://www.tibco.com/products/tibco-spotfire
TMMData	https://www.tmmdata.com/
Trifacta Wrangler	https://www.trifacta.com/products/wrangler-editions/
Unifi Data Platform	https://unifisoftware.com/platform/
Waterline Data	https://www.waterlinedata.com/
Workday-Prism Analytics	https://www.workday.com/en-us/applications/analytics/prism-analytics.html
Yellowfin Data Prep	https://www.yellowfinbi.com/suite/data-prep
Zoho Analytics	https://www.zoho.com/analytics/

Table 4: Further features of data preparation tools

Altair	**Paxata**	**SAP**	**SAS**	**Tableau**	**Talend**	**Trifcata**
Advanced Filtering	Add Comments	Action History	Create Custom Code	Adjust Sample Size	Advanced Filtering	Add Comments
Audit User Actions	Advanced Filtering	Advanced Filtering	Data Lineage	Advanced Filtering	Aggregation Using Charts	Advanced Filtering
Comparison Functions	Aggregation	Aggregation	Data Sampling	Aggregation	Audit User Actions	Aggregation
Copy and Paste Columns	Check Spelling	Comparison Functions	Job Monitoring	Change Color Scheme	Calendar Formats	Comparison Functions
Create Summaries	Cluster Data Prep Steps	Copy and Paste Columns	Job Scheduling	Check Spelling	Country Name into Codes	Copy and Paste Columns
Data Histogram	Comparison Functions	Data Quality Statistics	Maintain Log	Data Size Details	Data Masks	Data Histogram
Data Lineage	Copy and Paste Columns	Date & Time Formats	Math functions	External Data Use	Date & Time Formats	Data Profiling
Data Sampling	Data Histogram	Deduplication Statistics	Multi User Access	Group and Replace	External Data Use	Date & Time Formats
Data Size Details	Data Lineage	External Data Use	Preparation Versions	Group Tasks	Extract Quarter from Date	Diagnose Failed Jobs
Edit Field Values	Data Profiling	Hide Column	Refresh Data from Source	Intelligent Bar	Find and Group	External Data Use
External Data Use	Data Sampling	Intelligent Bar	Reorder Preparation Steps	Maintain Log	Intelligent Bar	Fix Dependency Issues
Hide Column	Data Size Details	Maintain Log	Search and Replace	Math Functions	Maintain Log	Group and Replace
Job Scheduling	Date & Time Formats	Math Functions	Share Dataset	Mini Maps	Math Functions	Initial Parsing Steps
Maintain Log	External Data Use	Multi User Access	String Functions	Multi Language Support	Multi Language Support	Intelligent Bar
Math Functions	Find and Group	Multiple Graphs for Visuals	Transpose Data	Preparation Versions	Preparation Versions	Logical Functions
Move Column	Group and Replace	Preparation Versions	View Table Properties	Publish Flows	Reorder Preparation Steps	Maintain Log
Preparation Versions	Intelligent Bar	Refresh Data from Source	Visual Feedback	Refresh Data from Source	Search and Replace	Manage Flows with Folders
Refresh Data from Source	Intelligent Ingest	Reorder Preparation Steps	Work with Plans	Reorder Preparation Steps	Share Dataset	Manage String Lengths
Row and Column Counts	Maintain Log	Search and Replace		Schedule Flows	String Functions	Math Functions
Search and Replace	Math Functions	Share Dataset		Search and Replace	Suggestions	Multi Language Support
String Functions	Multi Language Support	String Functions		Share Dataset	Swap Column Content	Preparation Versions
Transpose Data	Preparation Versions	Suggestions		String Functions	Use Metric Symbols	Reorder Preparation Steps
Visual Feedback	Publish Dataset			Suggestions	Visual Feedback	Row and Column Counts
	Reorder Preparation Steps			Visual Feedback		Search and Replace
	Search and Replace					Sequence Datasets
	Send Notifications					Share Dataset
	Share Dataset					String Functions
	String Functions					Suggestions
	Suggestions					Target-driven preparation
	Version History					Track Data Changes
	Visual Feedback					Visual Feedback
						Workflow Automation

Goetz Graefe Speaks Out on (Not Only) Query Optimization

Marianne Winslett and Vanessa Braganholo

Goetz Graefe
https://dblp.org/pers/hd/g/Graefe:Goetz

Welcome to ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are at the 2017 SIGMOD and PODS conference in Chicago. I have here with me Goetz Graefe, who is the recipient of the SIGMOD Innovations Award, the SIGMOD Test of Time Award, the ICDE Distinguished Paper Award, and the ACM Software System Award, all for his work on query processing. His Ph.D. is from the University of Wisconsin − Madison. Goetz was an HP Fellow and he currently works for Google. So, Goetz, welcome!

Thank you.

You've worked at a lot of different places but always on query processing. Everybody wants to know what it's like to work on the same topic for 30 years.

I've worked at multiple places. I've taught at Oregon Graduate Institute and then at the University of Colorado at Boulder, and then at the Portland State University. Since then, I worked for 12 years at Microsoft and then for ten years at HP. So, I wouldn't say I worked at a lot of places in the last 20 years.

I also didn't always work on query processing. I worked on query processing as a graduate student, and then as a professor, and then for a while at Microsoft. But then I switched to working on indexing within the SQL Server product. And at HP, I worked on query execution again. And then, I also worked on concurrency control and on write-ahead logging and recovery. So, I've always worked within the database engine, but not exactly on one topic only.

Okay. So, why does everyone think you've been working on the same thing for 30 years? Is it because you're Mr. Query Optimization?

Actually, I don't know whether that is even true because, in Germany, I'm actually often known as Mr. B-Tree. Obviously, Rudy Bayer invented B-trees, but I have published probably more than half a dozen papers and surveys on B-Trees. So, in Germany, people think I'm Mr. B-Tree.

Okay, we're correcting that misperception then of 30 years of the same thing.

I like to believe I'm neither one. I'm neither only query processing nor only B-Trees. And I think there are a couple of pieces of research hopefully coming out soon on concurrency control, and there actually are several pieces out already on logging and recovery[1]. So, hopefully, that perception will vanish over time.

It sounds like you're a full-stack guy for the database engine itself.

Perhaps something like that.

People would like to know, are there still open problems in query optimization?

There most certainly are. Plenty of them. But they all boil down to fairly few topics. So, the way I think about query processing is No. 1; people expand the functionality of query processing. People put more and more analytic functions into query processing. People want to put all kinds of clever machine learning into the query processor. So, expanding the functionality is one thing.

The other thing (No. 2) is most people when they think query processing, they think about traditional relational query processing, and then they think about performance. And performance, I really divide into three things: (i) efficiency, meaning clever algorithms that process data fast; (ii) scalability, clever ways of using many, many computers (or at least many, many cores). And the third one (iii) is really robustness of performance, and with robustness, I mean more than predictability. If I have a car that never starts when it's wet but always starts when it's dry, it's a predictable car, but it's not a useful car. So, what I want is more than predictability. I want robustness, something I can rely on. I want the good performance every day. I'm okay if I don't get best performance. Maybe I don't get best performance ever.

And my standard analogy for that is if you own a house, every year you pay good money yet you hope you will never get anything back for it. It's called fire insurance, right? You pay it, and you pay it willingly as long as it's a small fraction of the value of the house. And as long as you can count on if you lose the house due to fire, it'll get replaced. So, then similarly with robust performance, you're probably willing to forego some small amount of efficiency, if, in return, you get robust performance, predictable performance, reliable performance, which among other things, permits you to load your service much higher.

If you get random load spikes and you never know when, you end up running a service at 20 percent utilization. But if your performance is very steady and utilization is very steady, it's perfectly reasonable to run at 60 percent utilization. And suddenly, the ten percent overhead – or maybe even the factor two overhead – comes back and more, if you can load your service substantially higher.

[1] The interested reader can refer to these publications:

Goetz Graefe: On transactional concurrency control. Synthesis Lectures on Data Management, Morgan & Claypool Publishers 2019.

Goetz Graefe, Wey Guy, Caetano Sauer: Instant recovery with write-ahead logging: page repair, system restart, media restore, and system failover, Second Edition. Synthesis Lectures on Data Management, Morgan & Claypool Publishers 2016.

That's a great lead-in for something else I wanted to ask you about. There's this recent work from Rick Snodgrass's group that suggests that there's an inherent hard limit on how well top-down rule-based optimizers can do. And the kind of behavior that they're seeing in their experiments with commercial engines is exactly what you describe as the key thing to avoid if you wanna have robust performance.

> ***[…] choice is confusion.***

It's like the system somehow ends up with too many options to consider, and it does worse and worse on an average random query. Not the ones it was tuned for, but just something that's slightly different. Do you think there is a wall there and that rule-based query optimizers have hit that wall or will hit that wall?

Well, I don't think it has anything to do with what kind of query optimizer you have. Whether it's top-down, bottom-up, rule-based, transformation-based, dynamic programming – they all have the same problem. I think the key issue here is also not how many logical operators you have like join and select. I think the bigger issue is how many physical operators you have. So, not how many algebra operations you have in your specification algebra, but in your execution algebra. And so, if you have 17 join algorithms in your system, chances are you'll hardly ever pick the optimal one. In fact, you should be happy if you always pick a good one. And it's unlikely to be the case.

So, the fewer algorithms you have – ideally, if you only have one – you can never choose a wrong one. So, yes, there is a practical limit, and I think the practical limit comes from two things. No. 1, choice is confusion. If you have too many choices, you get confused. No. 2, cardinality estimation will always be inaccurate. No matter how sophisticated your model is to describe the distribution of data values, there will always be perhaps an adversarial case – as a test case – where the model that you have chosen to implement does not capture the distribution you truly have.

And so, I don't believe that the solution for the lack of robustness in query performance will come from the planning part of query processing. I actually believe it will come from the execution part from query processing.

So, what do you mean?

Well, I think plans will often be right and good. And there will always be cases where, in particular, the compile-time planner will choose a bad plan. And what we really need is execution engines that are much more forgiving. So, the word that I choose here is graceful degradation. It's very important. The problem in products is that the customer complains about a bad plan having been chosen for a query. Now, a bad plan chosen sounds of course like a defect, a bug, a complaint that should go to the query optimization team. So, the query optimization team will do what they can to have a different plan chosen or a better plan chosen.

Maybe they make the cardinality estimation or the cost calculation more sophisticated. But I think in some sense, it's a futile battle. I think that in many cases, the solution will come from the query execution engine being more forgiving about what plan actually got handed to the query execution. So, can the query execution engine somehow avoid performance deterioration that is not graceful? And can the query execution engine execute the plan in a way that doesn't show the mistake as badly as a naïve query execution engine would?

So, what exactly should be done differently at runtime?

So, the algorithms executing at runtime have to be implemented in such a way that they transition from an execution mode optimized for small data to an execution mode optimized for large data in a graceful way and in an incremental way, as opposed to having a big switch. For a simple example but something that's nonetheless used heavily by systems and customers, imagine you want to sort data. If the sort input fits in memory, you'll probably use an in-memory sort, like quicksort, and the data gets loaded into the sort workspace and then gets scanned out of the sort workspace. If you have one record more than fits in memory, how much data gets written to temporary storage?

It seems it should only be one record or one page. But in the naïve implementation that might have been done under time pressure – "Let's get the release out" – there might actually be a sort that spills the entire memory content. Now, if you have a gigabyte sort space, and you spill at 100 megabytes per second, and you load it back in at 100 megabytes per second, that's 20 seconds right there. So, for one extra record, we have 20 seconds extra runtime. And customers are guaranteed to come back and say, "Bad plan chosen."

Can you give another example? That was a great example.

Well, let's take hash join. There are different versions of hash join, different versions of hybrid hash join. In particular, in terms of when you need to know how big the inputs are. And if you implement hybrid hash join from the get-go, from the start, anticipating unknown input sizes (inputs that are smaller or larger than the optimizer might have said), then the hybrid hash join should spill incrementally. So, it should start running as an in-memory join and then spill a little bit, and if necessary, spill a little bit more. So, that would be a graceful behavior.

Can hardcore database engine internals research still be done in academia?

Absolutely. Yes. And lots of people do. In fact, in every SIGMOD, every VLDB, you see a number of papers where somebody has done maybe only a twist on something previously or something fundamental. And yes, there's a lot of interesting work coming out of academia. Not everybody who can get a program to run necessarily and implicitly and immediately has interesting work. But I think there is absolutely interesting work to be done in academia but also in industrial research.

Are there any hardcore database engine internal problems where the research really needs to be done in industry rather than in academia?

I don't think so. Much of it can be done either place. When you say industry, you also have to distinguish between product groups and research groups. I think they really have different roles. But academia has yet another role. The way I see it, it's clear what a product group does. A product group produces product and either provides it as a software product on a DVD or as download, or also as a cloud service. Academics do research. They create IP and of course pass it on to the next generation. Industrial research labs have a very different role from both of them. Industrial research labs, in my opinion, should enable informed decisions.

This is something that the product groups don't do. The product groups have to execute. The product groups adopt technologies that they can adopt with a predictable effort (say in software development and testing) and with a predictable result (say in performance, efficiency, scalability, or robustness), whereas industrial research labs should take promising intellectual property and promising techniques and technologies and develop them to the point that leaders in product groups can make informed decisions about whether to adopt a technology, how to adopt a technology, or whether to skip a technology.

That's very interesting. But isn't it the product groups who have the most insight into what the pain points are for the customers, in other words, what IP needs to be developed?

Yes. But understanding the pain points, that can easily be transferred from a product group into an industrial research lab. I totally agree that the product groups should somewhat guide the industrial research labs. On the other hand, only "somewhat" because I think there is this famous quote attributed to Henry Ford: "If I had done what my customers wanted me to do, I would have produced faster horses." And I'm sure there are variants to that one.

But I think what industrial research groups also ought to do is prevent the product groups from getting scooped. So, explore outside technology that may or may not disrupt the products in some form. So, I think there too, the research labs should help the product leaders to make informed decisions, what to prototype, what to adopt, what to skip.

[…] if you have 17 join algorithms in your system, chances are you'll hardly ever pick the optimal one. In fact, you should be happy if you always pick a good one. And it's unlikely to be the case.

I like what you're saying. But isn't it true that if the industrial research lab came back and said, "You guys should really take a serious look at this disruptive technology," wouldn't the product groups not be very happy to hear that since it would disrupt their entire income stream?

Well, let's take an example. A traditional database product has a product group and a research lab. The research lab says "in-memory databases are going to be there", what the product leader might want to know is when. Also, what do we know about what the competition is already doing in terms of what public material is out there in websites or conference papers or something. And also, the next question that the product leader will ask is: what actually works? Just saying, "in-memory databases are coming, in-memory databases are coming," is not sufficient. It doesn't enable informed development decisions. It doesn't enable informed investment decisions.

At some point, the product leader has to say, "I'm not going to have five people work on a faster backup. But I'm gonna take three of them and have them work on in-memory transactions," or something like that. And that's a decision. And making that decision an informed decision, that's where the industrial research lab can create tremendous value for the industry, but also for the progress of the customers and of the applications and of whatever benefits they will provide.

Great. Young researchers would like to know what long-term hard systems problems you see. Not the hot topics but the long-term issues.

Well, that's a difficult one. And given that you are asking about long-term questions, I have a high probability of being wrong. So, I think scalability and robustness in scalability is going to be a big issue. I think we are going to reinvent a number of systems issues repeatedly. So, for example, today, we achieve robustness in scalable systems by mirroring like crazy. Every data page is written in multiple places. And if one of those places breaks down, we'll rely on the multiple copies. Now that is very expensive.

And if data keeps exploding in size in an exponential growth curve, and hardware is not growing as fast in storage capacity and processing capacity, then we might actually find that we can't have as many copies anymore. We have to do something with fewer copies. And I think that probably is going to be with us for a while.

Another problem that will be with us for a while is the problem we already talked about briefly: robust performance. I think designing efficient algorithms, making things parallel and scalable, those are trickier at times, but more manageable. Robustness is much harder, partially because it's much harder to measure. And if we don't have a clear agreed-upon metric, it's very hard to prove that my technique is better than your technique or the technique published last year.

You traditionally work on very intricate details of relational database engine internals. And this isn't an obvious match with your current employer, Google. Although of course, Google also cares about issues like fast recovery from failures. Can your results also be applied in some kind of way to Google's kind of massively parallel infrastructure?

I believe very much so. So, let's look at the query processing work that I've done and that I'm still doing. Google actually has multiple SQL engines. So, Google has multiple query optimizers and multiple query execution engines. All of them of course, designed and implemented from the get-go to be very scalable. So, efficiency, scalability, and robust performance are issues on all of these engines.

If you look at the indexing things I've worked on in the past, Google, like everybody else, will store more and more data in memory, meaning with very low latency. And in the past, a go-to on disk, a random access on disk, was considered very expensive. On a traditional disk drive, you can scan a megabyte in the time to read one byte in a random place. So, therefore, there are a number of systems that are optimized for fast scanning. And the principle optimizations for fast scannings are column stores and compression. So, in my mind, I think of column stores as optimized for disk-based data centers, disk-based data collections (traditionally, historical data collections have been disk-based). But when it comes to transaction processing or shorter history – not years of history but shorter history – and in particular looking into the future, I think more and more data will be in memory, where random accesses are much cheaper.

So, I think indexing and index-based query processing, and that means index maintenance techniques, index concurrency control, index recovery, index compression, all those things will definitely be used at Google, but also elsewhere. And I think whatever companies that are out there that Google competes on a business level, it also competes on a technology level, what internal technology is used. Google is clearly interested in in-memory processing, in-memory indexing, SQL query processing, and so on.

Think about concurrency control with many-core processors. Concurrency is an issue because there are multiple threads, multiple transactions running in any sphere of control, in any operating system instance. Concurrency control is a big issue, and I think the work I have been doing recently on precision in concurrency control – lock sizes and lock durations – can very much have an impact on Google.

Thinking about recovery and availability, obviously, Google very much depends on continuous processing of its logs. Google collects a lot of logs from online activity. And those logs need to be processed. One day's worth of log needs to be processed in less than a day, otherwise, we fall behind. Keeping that log processing pipeline and all its components up and running is very important too, You can think of all of those things as invented and designed and perhaps publicly described in the context of traditional relational databases. But many or all of those things are very much transferable into other environments.

Today's commercial query optimizers are all based on Cascades, the top-down rule-based approach to query

optimization that you put together 25 years ago. Does it surprise you that even new optimizers like Orca still use the Cascades framework?

Very much. Yes, it does. Orca actually not only uses the approach, but I think Orca is somewhat of a reimplementation of the Cascades paper[2]. And I just heard today at the SIGMOD conference here in Chicago that somebody had as a student semester project a reimplementation of Cascades. And that is now an open-source piece of software, apparently. Yes, it surprises me very much that people still follow this approach. I think this approach is very good with respect to extensibility. So, if you want to bring in a new operation into your specification algebra or into your execution algebra, then yes. Cascades is very nice because it's very extensible.

On the other hand, Cascades doesn't do anything for anybody with respect to cardinality estimation, which is really the Achilles' heel of compile-time query planning.

I don't believe that the solution for the lack of robustness in query performance will come from the planning part of query processing. I actually believe it will come from the execution part from query processing.

The other thing is I think if you look at the core of most relational queries, it is still joins. And I think the group around Pat Selinger at IBM Almaden, and their paper from 1979 is still a foundation[3]. I think Thomas Neumann has done excellent work with his advisor Guido Moerkotte and then since on extending that to more complex join predicates, for example.

If I were to build a query optimizer today from scratch, I would use dynamic programming for join optimization. And I would use a Cascades-style transformation approach for extensibility. But as I said earlier in this conversation, I would also build a query execution engine to complement my optimizer, in a way that it is very forgiving of poor plan choices.

What do you think of key-value stores?

I think key-value stores have their place in the scheme of things. Key-value stores come in a wide variety of scalability, capability, and so on. At some places, they are the right tool. And that's what it's all about, choosing the right tool. At some other places, they are not. Personally, I am very convinced that application programmers want serializable transactions, meaning application programmers have the freedom, the liberty, the simplicity of thinking whatever transaction they run is the only thing going. I think that's a powerful paradigm. Some people strongly agree with me. Some people strongly disagree with me. And that's okay. I happen to have one belief. There you have it.

Some key-value stores are better about it than others. And I think some people trade performance for concurrency, for cleanliness of transactions. As I said, I usually would forego performance and scalability if I can get cleanliness of the application model. But then I think we, as data engine experts, should try to make the clean application programming model highly efficient in the engine.

So, I mentioned earlier concurrency control, the granularity of concurrency control, the duration of locks, how many false conflicts do we detect and treat them as if they are conflicts. In my concurrency control work that is basically the theme: avoiding false conflicts. And I think there is probably a factor 100 in that.

Wow. Okay. You teach a one-week course on – we won't pigeonhole it. We'll just say database engines – every year at Dagstuhl. In this day and age of education over the internet, why don't you just record your class and leave it on the web for posterity?

Well, there are many reasons for that. I think the students, typically fresh masters graduates, get much more out of it if it's interactive. Even when I was teaching undergraduates, I always was trying to learn names, basically have conversations rather than lectures. So, I think it's much better for the students if it's interactive. I think it's also much better for the students if they in some sense experience, what for many, is the first international event.

And *Dagstuhl* is a very nice and protective environment. For many of the participants, that's a very positive experience. Personally, I enjoy it very much. Yes, I miss teaching. I used to like it very much.

[2] Goetz Graefe: The Cascades Framework for Query Optimization. IEEE Data Eng. Bull. 18(3): 19-29 (1995).

[3] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, Thomas G. Price: Access Path Selection in a Relational Database Management System. SIGMOD Conference 1979: 23-34.

And so, this is my outlet. And I love *Dagstuhl*. In fact, I'm on one of their boards, so I have to go there for their board meetings at least once a year.

Do you have any other words of advice for fledgling or midcareer database researchers?

Well, what advice? Never give up. That's really the advice I have because there have been a number of times where things have not gone well in my career. I had to leave a university because clearly my tenure was going down the drain. In retrospect, they probably would be happy to have had me. I think other things have not gone my way. You just keep plugging away, and you show them. And that would be my advice.

Work on real problems. Solve problems that you know exist. And then have confidence that you can solve them and keep working on them.

Work on problems nobody cares about because in particular, if you don't have a large group, that's the best way to make progress without fierce competition. For example, at Hewlett-Packard, I felt at times I was the only database expert in Hewlett-Packard Labs. And so, I worked on stuff like concurrency control and join algorithms. And I knew there wouldn't be competition. If I get it published this year, get it published next year, nobody cares. Nobody will scoop me because nobody was working on concurrency control and join algorithms. So, just keep plugging away, and you'll get there.

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

I would really, really love to have a team to implement a new system that actually is innovative by simplicity. Simple is absolutely important because if it's not simple, I don't understand it. And every system I know has gotten so unbelievably complex. And people revel in the complexity, it feels to me. Building something really simple, that would be fun. But it would require a small team to build something that is still robust say, against data loss, but also robust in terms of query performance.

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

Perhaps I would have stayed an extra year in graduate school and had learned about artificial intelligence and basically had done more with it. I mean, I went through graduate school in four years, which was fast. And I think maybe if I had stayed an extra year, that could have been fun.

Thank you very much for talking with us today.

It's been my pleasure.

Applied Research Lessons from CloudViews Project

Alekh Jindal

Gray Systems Lab
Microsoft

ABSTRACT

Industry research has a rich legacy in computer science [9]. However, as opposed to the blue-sky approach to research, increasingly there is a trend to align industry research more closely with the products. This is manifested in several new trends in industry research: (i) emphasis on product impact, e.g., improving existing products or seeing new ones coming around the bend, (ii) popularity of blended job functions, such as scientist, research scientist, and data scientist, and (iii) setting up research teams that are integrated within the product organization to forge closer collaborations. The latter is a case in point in Azure Data group at Microsoft, where the Gray Systems Lab (GSL) [2] is an applied research team within the product group. Such integrated research labs offer beautiful opportunities for combining research with product impact. Yet, due to their product focus from the get-go, applied research labs could also be challenging to get started. Fortunately, it turns out that there are a set of things that new researchers could do in order to set themselves up for success in a product group.

In this paper, we describe the key lessons learned from the CloudViews project [1] at GSL. CloudViews project started with identifying and reusing common subexpressions in big data workloads at Microsoft, however, it was also successful in spinning up a number of follow-up projects, establishing the GSL ties with the SCOPE[1] team, and seeding the bigger vision of workload optimization, resulting in the Peregrine [7] and Flock [4] projects. Although the lessons we discuss below are derived from the CloudViews project at GSL, we believe the learnings are applicable to other industry research settings as well. Note that there could be several successful ways of going about applied research, however, in this paper, we only discuss the things that we found useful in our experience from the CloudViews project.

[1]SCOPE [12] is the query processing engine in Cosmos, the big data infrastructure for running massive scale analytics, consisting of hundreds of thousands of machines and hundreds of thousands of analytical jobs, across the whole of Microsoft, including teams such as Bing, Office, Windows, XBox, etc.

1. GETTING STARTED

Researchers at Microsoft, and other places, often have the freedom to pick their projects. However, this could also be overwhelming given wide range of products and possible opportunities at a large company like Microsoft. Therefore, before getting started, it could be useful to familiarize oneself with some of the products and pick one that is promising and big enough, in terms of usage, revenue, etc. The choice may be narrowed down by talking to colleagues, managers, or potential business groups, and it is important to be open to learn new areas.

After picking a product, get ready for a long term investment with the product team. This means getting hands dirty by diving into the production codebases — often a leap for researchers who might be used to building standalone prototypes. Getting hands-on on production codebases is useful in keeping the research project grounded in practicality. Also, it is easy to get distracted with too many conversations across several products and there could be temptation to spin projects with each of them. However, it takes a lot of work to onboard and work with a product team, so an individual contributor might be better off by focusing on few things in the early stages. This also means to start sunsetting prior projects and collaborations from the graduate school.

To illustrate, the CloudViews project started with the intent to build connections with the SCOPE product team, in particular the SCOPE query optimizer team. There was a feeling that query optimization in cloud workloads is going to be an important area for long term investments from a research team, and the guidance from the leadership was to understand the SCOPE engine better, get hands dirty by going into the guts of the query optimizer, and explore opportunities for impact. This was a challenge since our prior experience was on building flexible storage systems and query optimizer was quite a shift. In summary, getting started in applied industry research at Microsoft involves taking a leap of faith on a product and preparing oneself for the long haul. Thinking of it as a strategic opportunity can certainly make things easier.

2. FINDING PROBLEMS

Research teams at Microsoft are typically separate from the product teams and so they often have the flexibility in picking problems. However, problem selection is an art and some people have a natural inclination for identifying big and interesting problems. Therefore, it is hard to come up with a recipe for finding the right problems. However, there are certain things one can do to increase their chances. A common pitfall is to jump to a solution, e.g., from one's previous work or from the state of the art, without fully understanding the problem. This often leads to solving something which is either not really a significant problem in the first place or solved in a manner that could not be applied in practice. While it is natural to try hard and polish the problem to fit to the prior solution, this could be very frustrating to the product teams. Instead, focus on understanding the real problems. Start with talking to the product teams, program managers, and customers if possible (e.g., if there are internal customers of the product). One may often end up with their laundry list of TODOs and bugs. One may be also shocked to see that many of the standard concepts and techniques from research papers have simply not made it to production. Don't get disheartened. Instead, consider this an opportunity to understand what works and what doesn't, why is it hard to get things right, and what are the core issues.

Once an engagement is established with a specific product team and there is some understanding of their open problems, the next step is to get access to their data and workloads to validate those problems. There is a tremendous value in being data-driven when looking for a research problem. It is also okay to shape a problem from the product team's TODO list, using ideas and intuition that are backed by data. The key is to convince the product team that the problem is worth solving. While navigating the problem space, loop back with the product teams, the PMs, and the customers, to share and corroborate the insights from their workloads. Ultimately, the project should answer positively to one of the following two questions: (i) will it improve an existing product or feature such that the customers could measure it, e.g., more efficient or more usable, or (ii) will it add new capabilities that allows the customers to do something more, e.g., enabling newer scenarios. The selected problem should also be feasible, i.e., estimate the work required early on. The goal should be to find a problem that is simple to get started, and yet has a big potential for impact.

Before arriving at the core problems to solve in CloudViews, we started talking to the SCOPE team and early conversations indeed resulted in a list of open bugs and issues that the SCOPE team needed help with. They also had a plethora of research work from the previous generation of team members that they were looking to transition into the product. These included optimizing recurring queries [3] and optimizing queries progressively [5]. Our observation was that while the primary focus was on optimizing each individual SCOPE query, there was a demand for reducing the overall operational costs, i.e., we could consider optimizing across queries as well. At the same time, there were existing pieces of infrastructure, e.g., plan identifiers called signatures [3] that were captured in the telemetry and could be used for multi-query optimization (MQO). These two observations led us to analyze the SCOPE workloads for MQO opportunities. Interestingly, we found common subexpressions to be a major problem in SCOPE workloads, where portions of the SCOPE queries were duplicated, thereby incurring redundant computations [8]. This early leg work went a long way in getting support from the SCOPE team.

3. GETTING FUNDED

A research team at Microsoft usually has limited resources, while taking a research idea all the way to a production ready solution requires a lot more effort[2]. Therefore, once there is an interesting problem that emerged from conversations with a product team and is backed by data, the next thing to think about is how to get the project funded. The first thing to do before expecting any funding is to align the problem with the product team's roadmap and priorities. A problem that is super interesting but does not fit into the product team's agenda is unlikely to get any funding. But wait what kind of funding are we talking about? The funding may consist engineers to help design, build, debug, test, and maintain your code, program managers to identify the market, make a business case, define priorities, and manage deliverables of your project, and customers to try out your solution, provide feedback, and finally to validate your success. In spite of this clear set of things that would be needed, the seed funding is always the researcher herself. This is simply because people are unlikely to to invest into a researcher's idea if she herself is not invested. So be prepared to be a one-person army in the beginning, and this is what we are going to focus on in the rest of this section.

Get things rolling by getting access to code, attending weekly meetings and scrums, and establishing a point of contact from the product team. Given the early stage of the project, be explicit about the fact that the point of contact is for lightweight consulting, e.g., one hour per week. Product teams are often driven by planning cadence and so it is unrealistic to ask too much from them

[2]The exception of course is if the project is completely research-oriented in nature without any plans for immediate product integration.

until things are put into the planning (otherwise the engineer's work may not end up being accounted). Digging into the code is important to understand the approaches to implement the key ideas. Large product teams often try a number of things, and the current idea could be one of them, may be in some other form. Learn about those attempts, their current status, and key learnings. It is important to know why things failed the last time they were tried in order to not make the same mistake. More importantly, see what pieces of code could be salvaged from those prior attempts. There is no point reinventing the wheel, so be open to building on top of prior art.

Once there is some understanding of the code and the prior art, the next step is to come up with an initial design. Often there is a big value in making the implementation iterative, e.g., thinking what could be the bare minimal version (v0) to demonstrate the ideas; what would be v1, v2, and so on. Identifying these versions helps stage the project into smaller achievable steps while also providing the bigger vision to the audience. Work closely with the engineering contact point to define the v0 and building a proof of concept (POC) around it. While this would be far from the production version, it is still important for understanding the system, getting a good sense of what pieces need to be built, and for earning early credits/credibility for being a doer.

Once there is a POC in place, pick a catchy project name. The good thing about names is that they can identify things succinctly, ideally in one word, rather than describing it in a sentence. The bad thing, however, is that it takes time for a name to stick in people's working memory. Although, this is exactly why it is important to pick a name early on and keep using it in everyday conversations, hoping for it to stick eventually.

In CloudViews, we jumped right into the SCOPE codebase to get a hang of things. While we got one person from the SCOPE team doing lightweight consulting, most of the driving and prototype building was on us. Given that CloudViews was the first MQO feature in SCOPE, we had to think through how to make it automatic in the SCOPE job service. We defined the v0 implementation and managed to get one pilot customer willing to try our feature if we were to build it. Once we got the POC working, we organized a demo, however, it crashed quickly since they had no cycles for offline data processing while our initial implementation would materialize common subexpressions in additional offline SCOPE jobs. This led us to think of materializing common subexpressions as part of query processing. We held on to this pilot customer for a long time and they were super useful in helping us understand the production scenarios. We also got a PM onboard to help productize the solution going forward. It turned out that PMs could be one of the best friends for the research teams and they should be actively leveraged for getting the customer perspective, driving product planning and priorities, interfacing product releases, and leading customer adoption. Finally, the goal for selecting the name of CloudViews project was to keep it simple, relatable, and consistent (last one being very important).

4. BUILDING COLLABORATIONS

Developers at Microsoft are part of the product teams and even after a problem has been identified that the product team agrees on, and a POC has been built that the PM team (and hopefully one early customer) is willing to buy, the researcher still needs to builds active collaborations with the developers in the product team. This is because typical funding for research projects at Microsoft does not result in a dedicated set of people, at least not for a fresh new researcher. Rather, the researcher needs to identify the key challenges and work items that need to be done to fully flesh out the POC, and rally people for their support. Of course, this would require back and forth discussions with the product teams. Remember the goal is to create a production ready feature/system and it needs to be conveyed as new work (and it should be new work), since nobody is interested in wrapping up a POC where all credits have been already claimed. While discussing, also consider who is the best person for each of the tasks and why. People could have different roles such as general interest, direction setting, brainstorming, consulting, planning, developing, customer interfacing, deployment, testing, etc. Try getting as many people onboard as possible for the long term, although everyone should have a clear well-defined role. Thereafter, talk to the managers and get approvals for the peoples' time. Work with the PMs to have the above plan added to the future planning cadence, so that people's time is officially accounted for.

One thing to note is that there is a value in creating a virtual team (also referred to as V-team) for the project. This could be as simple as creating an email distribution list or a SharePoint page. The goal is to give collaborators a sense of belonging and a means to communicate across a set of people working towards a common goal. Its good to be as inclusive as possible when adding people to the V-team. The entire process of discussing project implementation details with the product team and getting specific support from them by creating a V-team could go a long way in building sustained collaborations, even beyond the current project.

There was a long code review cycle for the initial CloudViews POC. We got people from different SCOPE component teams involved. Initially, we pushed for dedicated resources, but then we divided our ask into fine-grained tasks and convinced the component teams to add them to their planning with people hours assigned. The

CloudViews V-team had people from different component teams, with 8 members at its peak. We created mailing lists, held regular scrums, and even organized team events for the V-team. This was also a good time for us to present the problem statement, the POC, and the collaborations to the leadership team.

5. WALKING THE TALK

By now a lot of ground work has already been done, however, now comes the most important phase of the project — the execution phase. All the planning and preparation could go to waste if things are not executed properly. So now is the time for hands down execution using all the people and resources that have been gathered. Remember the project is now visible and people have committed their time to it, so one needs to be very diligent at how they go about things. Also, remember that the V-team was created for a reason, so delegate tasks to people in the V-team. Assign people, including the researchers, ownership of smaller pieces and make them accountable for it. Organize daily scrums in the beginning so that everyone is clear about their tasks and is able to make/report progress. The goal should be to orchestrate the execution and making sure people are unblocked as quickly as possible. At the same time, it is important to be hands on and not lose touch from details. Since the founding researchers have the best understanding of the project, they need to remain on top of things for a while. As the project progresses, there might be possible conflicts amongst V-team members: people not agreeing on the design or the APIs, not coordinating the action items, or loosing interest in the project. Overcoming these conflicts and bringing people on the same page is a crucial people skill to develop. All along, it is important to listen to diverse opinions, even though one may or may not agree.

Rarely a product is developed in a single shot, therefore be constantly reminded on applying an iterative model, i.e., building versions v1, v2, v3, and so on. Test the feature end-to-end as early as possible, preferably in production environments, to iron out the bugs or correct design decisions. All along, keep people informed about the progress. Once the productized version (v1), works, present it to the customers and get their feedback.

CloudViews had a long march towards getting the basic things working, including things like unit testing, integration testing, flighting, debugging, etc. Our first checkpoint was to make CloudViews as optional feature in an upcoming SCOPE release, i.e., a private preview. Next, we focussed on making it real for a pilot customer, discovering and fixing several bugs along the way. Once we gained more confidence, we announced it as public preview it in the next release. This was followed by further stabilization and getting it working for more customers, eventually leading to general availability (GA). Finally, there was a march towards getting customers on-boarded and addressing their adoption pain. All along we learned valuable lessons.

6. QUANTIFYING IMPACT

Now that the first version of the feature is built, it is time to market your work. The easiest way to sell something is by providing quantitative metrics, e.g., by validating the feature on a subset of customer data. Most companies have a way to do A/B testing before releasing new products. Get a hold of those testing tools and estimate the wins on a subset of real customer data. It may very well happen that the wins are not exactly the same as the projected wins from the initial workload analysis. Don't be defensive. Instead, analyze the gap between the expected and the actuals, and reconcile with it. Also, try to think ahead on what the observations translate into: are there more roadblocks that one should anticipate? is this leading to other interesting problems for the future? are there some fundamental properties about the problem space that could be distilled out? These may help plan some of the next steps going forward.

Once there are observed wins on one subset of the customer data, use that to motivate and drive adoption by all other customers. Note that customer adoption of any new product could often be a long journey and PMs are often the best buddies in this journey; work with them closely. In fact, after the first few instances, the PMs should be completely owning (as well as credited henceforth) for driving the customer adoption. In this phase, there will also be the inevitable cycle of bugs-fixes, releases, and usage. So, this is the time to harden the feature for production readiness. And given that multiple people will be helping with these different stages, be very open to share ownership, credits, and accolades. In particular, work actively towards making everyone involved visible and rewarded.

We took several steps to capture the impact of CloudViews. First of all, we added the tooling to make the feature visible to the end users. Then, we added telemetry to log the use of the feature for offline analysis. Finally, we ran reporting scripts to analyze and compare the query logs before and after the feature was enabled, and cross channeled them to drive further adoption.

7. PUBLISHING

There is a tremendous value in publishing applied research work. First of all, it is important to develop a sense of clarity around the key ideas. The process of writing forces one to convey things in a crisp, concise, and understandable manner, which alone is very useful for communicating with peers, collaborators, or anyone interested in the work. Ideas well communicated are

long lasting and have greater impact. Well written ideas are also well thought through and serve as a good basis when preparing for a presentation. In fact, presentation quality improves dramatically if it is based on a published work. Second, having a work reviewed and accepted for publication in a journal/conference gains credibility. Therefore, it is also important to target top journals/conferences in order to get higher credibility. Published works are easier to discover and reference to, making the researchers more credible over time. Third, it is desirable to have visibility, both for individuals and the team. Personal visibility can help people stay motivated, grow their career, and build future collaborations. Team visibility is needed for attracting top talent and for continued funding to the team, both essential for the team's survival. Fourth, it is useful to engage with the broader research community by sharing the results, the systems built, or just the directions pursued. These can be helpful to motivate industry relevant problems that other people can catch up or contribute. Wherever possible, consider sharing production data, or code, or customer insights to enable people benchmark and validate their solutions, or even derive newer problems.

Writing a paper from an industry research lab could turn out to be relatively easy. In fact, just writing down the previous six sections can help to motivate the problem, show data on why it is significant, discuss the ideas, describe the implementation, and finally show empirical evidence on how good it works. These steps backed by real problems, real data, and real workload can certainly go a long way in making the paper impactful. Consider both research and industry tracks — research tracks for problems where you have gone deeper and innovated on specific solutions, and industry track where you have considered the end-to-end scenarios, abstractions, or platforms, and their practicality in industry setting. Finally, be forthright in what is actually deployed in production and what is not. Apart from intellectual honesty, it is also important to not upset people from the product side with exaggerated claims, which the reviewers and editors have practically no way to verify before accepting the paper for publication.

Regarding authorship, consider that product teams may have built the system while not necessarily contributing to the writeup, and may therefore deserve authorship. As the paper draft develops, get feedback from product teams to be more precise about: (i) the prior state of things before the feature came around, (ii) how the feature was implemented, (iii) what is deployed and what is not. Involving product people generally improves the quality of the paper, makes it more detailed, and avoids feelings of discredit or misrepresentation.

In CloudViews, we found a new take on the older problem of materialized views, namely applying it to a cloud-based job service. As we began writing the CloudViews paper, we soon realized that there was too much material for a single paper, and so we split it into two papers, a system paper [8] and algorithm paper [6], and crediting the V-team in these papers was super helpful in boosting morale and keeping the motivation high.

8. SUCCESS METRICS

It is easy to see research output in binaries — whether it worked or whether it failed. Some people go even further to judge research output purely in terms of dollar amount. However, such a perspective misses the valuable lessons learned along the way, in each of the seven steps described before. For instance, a fresh understanding of product with the intent of working on it may reveal several gaps, analysis of product related data can discover exciting opportunities to make things better, getting ideas funded can surface the cost and feasibility of the effort (or even related efforts), teaming up with set of people can generate feedback on different aspects of the problem, going ahead and actually implementing the ideas could provide a good engineering experience, objectively measuring the impact could validate whether the ideas would actual work or not, and trying to get things published can tell how much and why the external world cares about these ideas. All of these intermediate lessons are equally important and should be incorporated when defining the success metrics. In fact, success metrics should also equally embrace the negative results, since not everything is going to be a success, particularly in a research team. And so people should be incentivized to still take measured amount of risks and report back the negative results, if any. Once the success metric is agreed upon, it is important to use that to evaluate the project. A good time to do that is while trying to get the work published, since the project must be at an advanced stage by then. In fact, use the success metrics to prove the point in the publication.

The CloudViews project was fairly successful on all of the above steps, however, given the duration of the project over multiple years, it also helped us realize the value of each step along the way. This leads us to believe that the key question when evaluating the success of a research project is to check whether one or more of the above steps were completed or not, and those should be presented as success to the management for the individual rewards and career progression.

9. MOVING ON

It is important to wind down a project once the expected success metrics have been achieved. In fact, winding down a project is just as important as spinning it up, especially in a research lab where often the goal is to explore and expand on newer frontiers. But there are a

set of things that need to be done before winding down. First of all, product people need to be trained to be able to run, operate, and debug the feature independently. The researcher needs to remove herself from the critical path and transfer ownership to the product teams. This is an important step and the technology transfer is only truly complete when the show goes on even when the researcher is not around.

Moving on involves not just winding down the current project but also finding the next one. Consider expanding the scope of the current problem, or looking at related problems, or defining the next level of abstraction that becomes relevant beyond the current one. Think about the expertise built so far that could be leveraged or the unfair advantage that has been gained from the previous project. While shaping the next problem, based on the experiences from the previous ones, articulate it as the vision for the next few years. This is timely to reflect back on how things worked in the past, think ahead on how you would like them in the future, and motivate or influence the future directions of the team. There will also be an expectation to present the larger vision as one grows more senior in the team. Not to mention it will help validate the course of action and get feedback for any corrective measures, if needed.

Two followup directions emerged from the CloudViews project, namely, to apply computation reuse to other query engines, and to consider other workload optimizations for the SCOPE (and other) query engines. As a result, we started multiple efforts in both directions, e.g., applying CloudViews to the Spark engine [10] and learning cardinality models from SCOPE workloads [11]. We further abstracted our ideas into a common workload optimization platform, called Peregrine [7], to make it easier for developing workload optimization features for different cloud query engines.

10. PORTFOLIO MANAGEMENT

One of the last lessons that we found valuable for being effective at Microsoft is to gradually develop a portfolio of projects, in particular as one transitions from the first project to the next ones. Managing a portfolio has several key benefits. First of all, research projects generally have a longer end to end arc, from ideation all the way to production deployment and adoption, and different stakeholders are involved differently in each of the stages, e.g., there could be more role for the researchers in the early ideation stage, more role for software developers in the system building stage, and more role for program managers in production deployment and onboarding. When one project does not require active involvement in its current stage, having a portfolio helps to remain occupied in other ones.

A portfolio also allows to pipeline projects such that they reach subsequent stages in succession. This can help in sustaining productivity at workplace, which is important both for professional success and for personal happiness. Researching on a single idea for too long can also make things boring and dampen the creativity. Juggling more than one project or idea can keep things interesting and can even cross pollinate ideas.

A portfolio also helps explore alternate ideas that are hard to pursue in a one project regime. This not only keeps the curiosity burning but also provides room to pursue riskier bets, which may or may not see the light of the day, or hot new trends, which need timely attention to not lose the early bird advantage. In a way, considering projects with different magnitude of ambition and feasibility, e.g., crazy new idea that may be completely impractical versus a straight forward extension of the earlier work, also hedges the bets and improves the chances of success, both in terms of making incremental moves and shooting for groundbreaking shifts.

CloudViews lead us to a vibrant portfolio consisting of multi-query optimizations, ML for systems, and platforms for workload optimization. Such a portfolio sets us up for an interesting research landscape, where we hope to create both personal and team identities.

Acknowledgements. We would like to thank the CloudViews and SCOPE teams for their continuous feedback, support, and openness to take on riskier bets. We are grateful to Hiren Patel for championing the CloudViews project from the PM side. And finally, special thanks to the GSL management and Raghu Ramakrishnan for supporting the project all along.

11. REFERENCES

[1] CloudViews Project. https://www.microsoft.com/en-us/research/project/cloudviews/.
[2] Gray Systems Lab. https://azuredata.microsoft.com/labs/gsl.
[3] S. Agarwal, S. Kandula, N. Bruno, M.-C. Wu, I. Stoica, and J. Zhou. Re-optimizing Data-parallel Computing. In *NSDI*, 2012.
[4] A. Agrawal, R. Chatterjee, C. Curino, A. Floratou, N. Gowdal, M. Interlandi, A. Jindal, K. Karanasos, S. Krishnan, B. Kroth, J. Leeka, K. Park, H. Patel, O. Poppe, F. Psallidas, R. Ramakrishnan, A. Roy, K. Saur, R. Sen, M. Weimer, T. Wright, and Y. Zhu. Cloudy with high chance of DBMS: A 10-year prediction for Enterprise-Grade ML. In *CIDR (to appear)*, 2020.
[5] N. Bruno, S. Jain, and J. Zhou. Continuous cloud-scale query optimization and processing. In *VLDB*, 2013.
[6] A. Jindal, K. Karanasos, S. Rao, and H. Patel. Thou Shall Not Recompute: Selecting Subexpressions to Materialize at Datacenter Scale. In *VLDB*, 2018.
[7] A. Jindal, H. Patel, A. Roy, S. Qiao, J. Yin, R. Sen, and S. Krishnan. Peregrine: Workload Optimization for Cloud Query Engines. In *SOCC (to appear)*, 2019.
[8] A. Jindal, S. Qiao, H. Patel, Z. Yin, J. Di, M. Bag, M. Friedman, Y. Lin, K. Karanasos, and S. Rao. Computation Reuse in Analytics Job Service at Microsoft. In *SIGMOD*, 2018.
[9] R. Levin. A Perspective on Computing Research Management. *Operating Systems Review*, 41(2):3–9, 2007.
[10] A. Roy, A. Jindal, H. Patel, A. Gosalia, S. Krishnan, and C. Curino. Sparkcruise: Handsfree computation reuse in spark. In *VLDB*, 2019.
[11] C. Wu, A. Jindal, S. Amizadeh, H. Patel, S. Qiao, W. Li, and S. Rao. Towards a Learning Optimizer for Shared Clouds. In *VLDB*, 2019.
[12] J. Zhou, N. Bruno, M.-C. Wu, P.-Å. Larson, R. Chaiken, and D. Shakib. SCOPE: parallel databases meet MapReduce. *VLDB J.*, 21(5):611–636, 2012.

Advice from SIGMOD/PODS 2020

David Maier, Rachel Pottinger, AnHai Doan, Eduard Dragut, Bill Howe, Joanne Lateulere, John Lateulere, Mostafa Milani, Tilmann Rabl, Dan Suciu, Yufei Tao, Wang-Chiew Tan, Kristin Tufte

1 Audience

This document collects the experiences and advice from the organizers of the SIGMOD/PODS 2020, which shifted on short notice to an online-only conference. It is mainly intended for others who are organizing online conferences, but some of it may be of use in the future to people organizing "live" conferences with an online component.

2 Timeline

SIGMOD/PODS 2020 was originally planned to take place in Portland, Oregon on 14-19 June 2020. While we contemplated early in January 2020 that the coronavirus outbreak might interfere with attendee travel, the realization that we would need to support some kinds of remote access came to the fore around January 21, with the first detection of a case in the US (in Washington state, adjacent to Oregon). By the first week of February we were hearing about in-person conferences with low turnout because of the ban on direct travel from China, and the organizers started discussing capacity for streaming and recording most sessions. We also recognized that we might have to provide for remote or prerecorded presentations. In early March, the US had 400 detected cases of COVID-19, and was experiencing problems with testing. There was a call then of some of the conference organizers with the SIGMOD Executive Committee (EC). We discussed the possibilities of canceling, postponing or going completely virtual. No final decision was taken, however canceling was unattractive—given that most of the paper reviewing was nearly completed—and postponing could mean dealing with the same issues farther down the road. Thus hybrid and completely virtual were the most likely choices. On 10 March we announced that the conference was going forward at the scheduled time, but that there would be provision for authors who couldn't attend.

By the second week of March, the situation was shifting rapidly. Companies and universities were banning non-essential travel, with no clear end time to the bans. On 11 March, the governor of Oregon banned gatherings of over 250 people for the next four weeks (but with no guarantee the ban would be lifted then). SIGCSE 2020, which had just started in Portland, canceled the remainder of their in-person conference. On 12 March we started exploring in earnest alternatives for remote participation by both presenters and audience members. It was a period of high uncertainty. We hoped to learn something from other conferences, such as EDBT/ICDE 2020 scheduled for the end of March, and ICDE 2020, which was considering postponing from their April dates (but ended up retaining their original dates in virtual mode). On 19 March the EC in consultation with conference organizers decided on an all-virtual conference. While that decision simplified some aspects of our planning (e.g., no food and beverage menus), we now had to deal with our contracts with the hotel and banquet venue. At this point we were less than three months out from the conference start, and needed to quickly determine what parts of the program to retain (which ended up being almost everything except end-of-day poster sessions). At the request of ACM, we held off on announcing the cancellation of the in-person part of the conference, while they negotiated with the hotel. On 24 March we announced that the conference was taking place, but the extent of

the in-person component (if any) was still to be determined. On 2 April the hotel agreed to let us cancel the contract without a penalty payment. In the days that followed, we notified the organisers that the conference would be all virtual, followed by the sponsors. On 10 April, we announced the change generally.

Overall response was positive. Of the 175 respondents to a post attendance survey, 67% thought that the conference was slightly better or much better than expected:

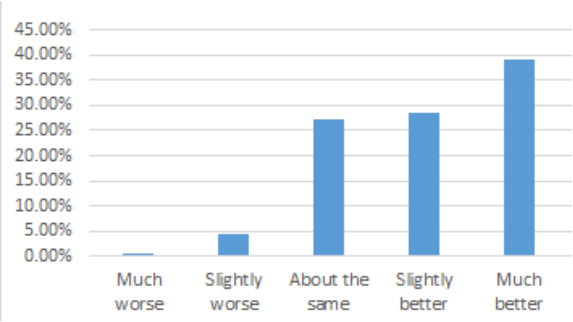

Figure 1: How well the conference met attendees expectations.

The remainder of this document covers some of the main decisions we made leading up to the conference, then touches on some of the details, plus the most-requested features that we didn't support. The final section includes additional selected statistics from the post-conference survey and the logs of Zoom sessions.

3 Preserve the Core, Retain the Schedule

Our first cut at a detailed plan for a fully virtual conference dates from 14 April. Our general goals were:

- Reuse as much of the planning as possible.
- Preserve the core of the conference.
- Regulate expectations.

Reuse our planning: The original conference schedule had consumed a lot of time and effort, both because of logistical constraints at the hotel, plus a program that included significantly more papers than recent years and a desire to have most tutorials during the conference proper, rather than in tandem with the workshops. While many of the logistical constraints went away with abandoning the in-person component in the conference, any significant rescheduling would mean more rounds of negotiations with SIGMOD and PODS PC chairs, plus the demos and tutorial chairs, plus the Student Research Competition. There was discussion of compressing the conference into a shorter period of time by reducing talk times and cutting the lengths of breaks and lunches. However, even though we no longer needed the time for coffee breaks and meals, we felt that people would need breaks to get away from their screens and stretch. In the end, the breaks in the schedule were useful for social-networking events and sponsor talks. There was also brief consideration of an asynchronous format, which we saw a few other conferences using. However, we wanted to retain the possibility of some live elements, plus 24-hour staffing for technical support and monitoring would have been difficult. In the end, our schedule was similar to the one we had for the in-person conference, with a few adjustments, such as parallel demo sessions and shifting a couple events (PODS business meeting, New Researchers Symposium) to early morning to make them more accessible to participants in Europe.

Preserve the core: We sought to "preserve" the core in two senses: 1) Retain the main elements of the conference, and 2) Have a record of the conference that people could access in the future. In terms of retaining elements, we certainly wanted to keep presentations for all PODS, SIGMOD technical and SIGMOD industrial papers, and we focused on those sessions initially. We ended up retaining almost all other elements, including demos, keynotes, panels, tutorials, business meetings, awards session, Student Research Competition, New Researcher Symposium, and workshops (though one workshop decided to cancel). The main thing that went away was the poster sessions each afternoon for all presenters for a given day. We did not identify a good way to support the large parallelism needed, and the time would have been after midnight in Europe (though manageable in much of Asia). In terms of retaining content, the papers would be available in the ACM Digital Library in any case. We also wanted to preserve as many presentations as possible, plus the associated Q&A. For the latter, since we weren't sure at first how it would be handled (via chat, within the streaming

channel, live), we were uncertain about capturing discussions.

Regulate expectations: Given the very short lead time, and the relative inexperience of all involved organizing a fully virtual conference, we did not want to over-promise on what we could deliver. Thus, as our initial baseline, we targeted pre-recorded talks, with questions in a chat channel, likely Slack. As more pieces became clear, such as the ACM subscription of Zoom and the capabilities of our A/V company, and as we saw what was working for other conferences such as EDBT/ICDT 2020 and ICDE 2020, we raised our sights to include live Q&A and some live sessions. We also wanted to give value to our sponsors, but were uncertain at first what we could offer and what would be appreciated. Based on much back-and-forth between our sponsor chairs and sponsor representatives, we added the option of sponsors getting half-hour talk slots and the opportunity to host other events (that they would set up and we would link to). We also provided "booths" in our virtual interaction space (Gather) for all sponsors.

4 Professional Help versus All Volunteers

We ended up being lucky in that the companies we had engaged to help with the in-person conference were able to stay with us and adapt to the changing needs for our virtual conference. We had engaged Integrated Management Solutions (IMS) to help with onsite logistics, such as food and beverage planning, A/V requirements, room scheduling and set-ups, tracking registration and monitoring and troubleshooting during the conference itself. They agreed to stay on in their support role as the conference shifted to online, helping collect and organize information for the detailed schedule, tracking video uploads, interfacing with our technical team, helping sort registration problems, monitoring Zoom and Slack for problems during the conference, and myriad other tasks. IMS in turn helped us connect with Gateway Production Services (now Equipment Asset Management) as a lower-cost alternative to the in-house A/V service at the hotel, to handle projection, audio, streaming and recording at the in-person conference. We were fortunate to have technical support that was not tied to the hotel. While Gateway did not have much prior experience with Zoom, they mastered the nuances quickly, and took over storing pre-recorded videos, organizing them for playback during the sessions, providing technical hosts for all conference-supported sessions, setting up Slack channels, making training materials for Zoom, and engaging a web designer to set up our schedule pages with all the Zoom and Slack links. While the conference was not without glitches, it on the whole ran smoothly. We do not believe it would have done so without the help of IMS and Gateway.

5 Live versus Recorded Presentations

A key question is whether presentations should be delivered live or prerecorded. Some people advocated for live presentations as being more spontaneous and interactive. However, there are risks with that approach: a presenter or session chair might have trouble connecting to your meeting platform, there might be background noise in the audio, there could be network interruptions, someone might get the time-zone difference wrong. (I (DM) am writing this just after our first SIGMOD plenary session, where our speaker had problems connecting to Zoom and also dropped out for a couple minutes in the middle of his talk.) For SIGMOD/PODS, we used prerecorded presentations with live Q&A for the most part. There were some keynotes and awards talks that were done live, as were some tutorials. Other tutorials interspersed recorded segments with discussion periods.

We also provided links from the online schedule page from each talk to the corresponding paper in the ACM Digital Library (except for a couple workshops whose proceedings weren't finalized at conference time). Access to papers provided an added way for participants to get additional information about a talk. We are intending to provide links to recordings of our sessions through the online schedule page as well. However, as of this writing (27 July 2020), the videos are just starting to be posted—there have been some delays involved in editing out segments for papers where the presenter did not give permission to post recordings on their rights form.

Some observations and suggestions:

- Having pre-recorded talks helped keep things on schedule. Session chairs didn't have to be timekeepers for presenters.
- Authors could monitor questions in the Zoom Q&A and Slack and answer them as the talk was proceeding.
- Make clear to people as early as possible that they will need to record their presentations, what the length is, the required format and when the deadline will be. You can follow up later with upload instructions.
- There were a handful of videos that exhibited a problem with audio lagging video by about 4 seconds (which might be due to limitations of some free editing tools). If we had been able to collect videos earlier and post them, then authors (or someone else) would have been able to check for problems.
- If someone wants to present live, insist on a test with them in advance. It would also be good if the host had the slides, which would permit the talk to go on with an audio-only connection.
- Think about a system for collecting videos. Mapping videos to the correct sessions is a logistical challenge. We relied on a naming scheme for the video files. It would have been easier for us if we had had time to set up an upload site where the submitter could supply some metadata with the video, including selecting a session from a pull-down menu.

6 Webinar versus Meeting Mode

Shortly after we decided to go to a fully virtual format, ACM subscribed to a Zoom meeting plan, and let conferences use it without cost. We decided to use Zoom for our conference sessions, based on familiarity of most potential participants with it and the budget savings for us. (We did need to pay to upgrade some sessions beyond the 300-person-per-session limit of ACM's plan.) We reserved the use of up to nine of ACM's slots for the conference.

There was then the question of whether to run sessions in meeting or webinar mode. In meeting mode there are hosts and participants, where participants can share audio and video at will. Webinars have hosts, panelists and participants. Panelists can share audio, video and desktop, while participants can only view and listen. However, the host can promote a participant to a panelist at any time. Also, webinar mode supports a Q&A pane to which everyone can post questions, and hosts and panelists can add answers. While meeting mode makes participants more visible, we decided to conduct nearly all sessions in webinar mode, to give the session chair and presenters a bit more control, and to avoid issues with intentional or unintentional disruptions, which others using meeting mode had reported. Using the same options and settings across sessions helps people get used to the "style" of interaction as the conference progressed.

For each session, we had (at least) two hosts. One was the technical host, provided by Gateway, and the other was the session chair. We needed to collect the names of session chairs in advance of the sessions, as they needed to be added as hosts at the start. In most cases, we did not have names of speakers ahead of sessions. Rather, they would join the session and identify themselves to the hosts, who could then promote them to panelists. Hosts could also promote someone with a question to a panelist, to ask the question live, though some hosts chose to read the question to the presenter.

Some observations and suggestions:

- We tried to start each Zoom webinar 15 minutes before the actual program started, so people could check their connections.
- Some people report being disquieted by being constantly visible, or constantly seeing themselves. Note that in a typical conference setting, audience members' faces are visible only to the speaker.
- Some workshops and sponsor events provided their own Zoom links. There was sometimes an issue with getting problem reports to the right place—participants were not necessarily aware that a link was not for a conference-supported session, and posted to the general tech-support Slack channel, rather than the one for the event. Some of those "external" Zoom sessions required registration, which caused problems for people connecting to the session from within their browser. Some companies ban employees from installing the Zoom application on work machines.
- We did do some testing in advance that our Zoom links were accessible from other countries.

- Gateway provided a training video and slide deck for session chair, plus set up practice sessions. There was also a video for participants, plus the slide roll before each session with basic instructions for participants. The program chairs also prepared guides for session chairs and speakers.
- PODS was configured with a single Zoom webinar per day. That meant that the Zoom logs did not break out attendance information by session, though the program chair did note this information.
- It wasn't feasible for a single technical host to handle back-to-back Zoom sessions (where one would start immediately after the other ended). It takes time to launch a Zoom session, plus we wanted to have a 15-minute buffer period before each session. That limitation was one reason that some sponsor talks and social-networking events ended up with "external" Zoom links. In retrospect, we should have arranged for one or two "tracks" in addition to those for regular conference sessions, to handle these additional conference elements.

7 Slack and Bulletins

Our choice of Slack as a discussion platform was mainly based on our familiarity with it (and assumed familiarity of most participants), plus the availability of a free tier with 10K messages visible. (Older messages are preserved, but are not visible without payment.) Our original baseline for the conference was streamed talks and all Q&A on Slack. When we adopted Zoom for our meeting platform, with its own Q&A support, we decided to retain Slack, as a vehicle for post-session discussions. That capability was especially important given that some sessions were at times not conducive to live viewing in some time zones.

We had observed some previous conferences where most Slack channels were lightly used. However, our experience was that nearly all channels had significant traffic, and we crossed the 10K-message threshold by the end of the conference (hence the messages from the beginning of the conference were no longer accessible). There were 1330 participants who signed up for the conference workspace. We provided a channel per session, plus one per sponsor and a few others (see bullet below). Session chairs often seeded their channel with a description of talks, and some transferred the unanswered questions from the Zoom Q&A to the channel after the session. Presenters almost always followed up to answer these questions. Some presenters posted links to slides, datasets and software in the channel for their session. Some channels saw further discussion around the theme for the session, on topics beyond those in the specific papers.

In addition to Slack, we sent an email bulletin every evening to all registrants. Those bulletins contained information about accessing the conference schedule, Slack and Gather. They contained highlights for the following day's program, contained answers to some frequent questions, and linked to other sources of information. We also listed all sponsor talks and social networking events for the coming day, as those were new elements to the conference, and participants might not be explicitly looking for them.

Additional notes:

- The Slack workspace wasn't protected, but we did not see issues with inappropriate content being posted. Session chairs or others would sometimes post the Zoom links for their sessions, which meant they were no longer password protected, as they were in the online schedule.
- In addition to session and sponsor channels, there were channels for general announcements, a bulletin board, tech support, and conference help.
- It appears that a workspace administrator needs to create a channel if people who newly join the workspace are to automatically see it. Others who create channels can subscribe all current users, but not (automatically) those who join after the channel is created.
- A channel per session was about the right granularity. Fewer channels would have made it hard to find posts relevant to a given paper. However, a channel per paper would have overwhelmed people with channels (there were already some complaints about how many there were), plus it would not provide a place for discussions related to the session topic generally.
- There may be alternatives to Slack worth considering, perhaps with a more generous free

tier or an alternative pricing structure (such as per message versus per user/month).

8 Free Participation for Most

Once the decision was made to go fully virtual, the budget picture changed greatly. We wouldn't, for example, have expenditures for food and beverage, nor on-site A/V rental. However, there were still many uncertainties at that point. On the income side, we did not know if sponsors would leave or lower their sponsorships levels. On the expense side, we were trying to estimate our sunk costs. However, we needed to have a new budget relatively soon to reopen registration (which we had shut down prior to the announcement of canceling the in-person component). Looking at other conferences that had shifted to virtual mode, it was fairly common to require one author per paper to register at the full (albeit reduced) rate, and for other participants to have free or nominal-cost registration. We followed that model, with a $300 registration fee for regular authors and $100 for workshop-only authors. All other participants could register for free. Our budget was conservative, with a $100K+ cushion between anticipated income and expenses. In the end, the surplus was less, due mainly to a few additional expenses and lower than forecast registration income because of duplication of authors between papers and student waivers. (We set up a waiver program for papers where all authors were students.) Sponsors got a number of free registrations based on their sponsorship level. Given that most participants were free, these registrations only had value if someone from the company was registering.

We decided on a target of 3000 registrants, based on what we thought were limits on the ACM Zoom license: 10 hosts at 300 people per session. We held back 100 slots for those who should have registered but didn't, such as session chairs, organizers and panelists. We ended up using about 50 of those slots, so total registration was around 2950. We maxed out before the conference started, so there were likely additional people who wanted to register, but couldn't. We could probably have accommodated more, as obviously not everyone who registers is going to attend every session, plus it is possible to purchase "upgrades" to increase attendance at a given session (which we did). It appears we only hit the attendance limit (of 1000) on one session, the first SIGMOD session, which included welcoming remarks and a keynote talk. There was also a possibility that some non-registrants were able to attend Zoom sessions, as we only password-protected the online schedule as a whole, and not individual sessions. Links to some Zoom sessions may have "escaped" by people posting them in non-protected places.

While it was useful to have a $0 participant fee this year to gauge the level of interest, we suggest a $20-$40 fee in the future. It will cut down on the number of people who register and do not attend, plus there are certain costs that accrue on a per-head basis (such as the fee to the registration company). Such a charge could be accompanied by a generous waiver program, so as not to exclude those who truly want to attend but have limited means.

9 Supporting Social Networking

The biggest drawback of a fully virtual conference is the absence of the "hallway track": the ability to easily have impromptu conversations with small groups. To partially remedy this gap, we added two elements to the conference: Social Networking events and the Gather virtual interaction space.

The **Social Networking** events were organized by the social events chair and the SIGMOD PC chairs. The goal of these events is to enliven the social aspects of conferences and in particular, provide more-junior members of the community an opportunity to hear from and interact with more senior people. These events took three forms:

1. Zoomtables: The typical SIGMOD technical session had five 12-minute presentations—plus 2 minutes each for questions—in an hour-and-a-half sessions. As this is the first time zoomtables are implemented, only some of the of the sessions chairs were encouraged to turn the remaining 20 minutes into a roundtable discussion with experts in the session topic whom they invited. Each had a "spillover" Zoom session where the conversation could continue beyond the end of the allocated session. Every zoomtable was very well-attended, and we would recommend this feature to continue for the next conferences.

2. Zoomside chats: There were a number of sessions with senior researchers with topics ranging from "Ask me anything about life in the academia" to "Experience Sharing: How to conduct research" with senior researchers in the field. These were separately scheduled Zoom sessions, either before the first program session of the day, or during breaks. Again, every zoomside chat was well-attended, and we would recommend this feature to continue for the next conference.
3. Women in DB. This event was advertised as "a roundtable discussion on research, mentorship, career paths, failures, work-life balance" with seven mid-career and senior women in the field. It was targeted at women beginning their careers in database research (but not restricted to them) and had women researchers at various stages in their careers leading the discussions. This session was very well-attended and should continue in the future.

In addition, the program and executive committees for SIGMOD organized an online retirement party for C. Mohan, who retired from IBM at the end of June. In contrast to the other social networking events, attendees for this event tended to be more senior. In particular, it attracted a number of retired members of the community, most of whom would not have come to a live conference just to attend such an event.

In retrospect, an additional Zoom track for social networking and similar events would have been worthwhile, at least for the Tuesday–Thursday run of the SIGMOD Conference proper.

After we decided to shift to a fully virtual conference, we learned of **Gather** (gather.town), which is a new platform that supports informal virtual interaction. Briefly, users are represented by small avatars in a 2-D meeting space. When two avatars approach each other, the video and audio for the two users fades in, and they can converse. Groups of 2 to 6 or so can form dynamically, much as in break spaces at conferences. (However, as at conferences, you might not be able to hear someone on the far side of a large group.) We also worked with the developers of Gather to add support for sponsor booths, which included branding, private conversation areas, private rooms and pop-up content, depending on the sponsorship level.

While only a fraction of participants visited Gather, many of those who did were enthusiastic about it. People were able to both connect with existing acquaintances and meet new people. We announced a couple of "parties" in Gather at times where no other events were scheduled, and that served to bring people into the space. Two ideas we had to encourage usage that we ran out of time to implement: 1) one or more "preview parties" before the conference started for organizers, sponsors, student helpers and others to familiarize them with the space, 2) creating a short video to orient people to the space. We did provide a written guide, plus gave advance "tours" to some people.

We did get suggestions about new features and improvements for Gather, such as making it easier to find a particular person in the space. However, the platform is advancing rapidly, and many of those items are already being addressed, so we won't list them here.

10 Retaining Sponsors

Even before we announced the conference would be fully online, we were hearing concerns from sponsors both that their staff might have trouble attending because of company travel restrictions and to what degree in-person attendance might be reduced. We were obviously concerned about how much sponsorship support we would retain as our plans evolved. Many sponsors participate for the networking and recruiting opportunities, while others, such as book publishers, are there mainly for marketing purposes. We only had a few sponsors in the latter category this year. Some sponsors just want to support the community, or particular aspects, such as diversity and the Student Research Competition.

The SIGMOD Sponsorships Chairs handled the bulk of communication with sponsors, trying to keep them abreast of conference developments, and soliciting suggestions of what they might find valuable in this new format. The accommodations we made this year for sponsors included:

- Bumping sponsorships levels up. For example, a sponsor who paid for Gold level was listed at the Platinum level.

- We added sponsor talks, or other events of their choosing. These were ½-hour long. Some talks used conference Zoom sessions, some sponsors provided their own. These talks were well attended, with more viewers on average than the technical sessions. (The talks were not scheduled in parallel with technical sessions nor each other.)
- There was a Slack channel for each sponsor.
- Sponsor logos were included in the slide roll before each session, and extended thanks were included in the welcoming remarks from the General Chairs.
- Each sponsor got a "booth" in Gather, whose size and placement depended on the sponsorship level. Looking back, it would have been useful to have functionality for participants to get to booths more easily, such as a special link that could take you to a Gather "spawn point" near a particular sponsor's booth.

11 Frequent Requests

There were several requests and suggestions for additional capabilities that we lacked. Most of these we considered in some form, but were not able to implement given short lead time and the need to focus on essential elements.

- Posting of the talk videos before the conference.
- Making slides for the talks and tutorials available. (Some presenters posted their slides or a link in the corresponding Slack channel after their sessions.)
- Links from the Overview Schedule to appropriate parts of details pages. (We had hoped to do so, but the schedule pages were still in flux as the conference was starting.)
- Local time adjustment: Having the times of session appear in a viewer's local time zone on the schedule.

12 Suggestions for Future Conference Organizers

- Have a Video Chair. We envisage that most SIGMOD/PODS conferences in the future will provide for prerecorded video presentations (at least as a back-up) and capture of most sessions to video. This position will need to interface closely with both the program side and local arrangements side of organization—monitoring the collection of videos, checking their quality, organizing them into the appropriate sessions and order, possibly advance posting of them, and planning and providing for posting of videos that are captured from sessions. Also, this person can follow up with authors who do not opt to give permission for recording on their rights form. (We found that at least half the authors who had selected "no" on the rights form had done so in error.)
- Collect additional information from authors, such as who the presenter will be and what time zone he or she is in.
- Even with a fully virtual conference, last-minute registration can be a problem: it's hard to set up credentials on different platforms instantly when someone registers. We ended up having to set up a temporary password for the online conference site for "day of" registrants.
- Also, make clear to all organizers, sessions chairs, keynoters, panelists and so forth that they need to register (even if registration is free). You want the registration site to have a complete record of registrants in order to reliably reach everyone by email.
- Consider an Award Coordinator position. The number of awards and recognitions announced at the conference has grown steadily over the years. It would help to have a single person who is collecting information about winners, arranging the session where awards are presented and talks given, and organizing plaques and payments where appropriate.
- Having the capacity for remote attendance by both presenters and audience members definitely broadened participation. It will be for others to decide whether to retain these options when the conference returns to live format. It will be a challenge to keep remote participants from being left out of the informal parts of the conference, and to dissuade local participants from spending even more time with their screens.
- Having free registration for most participants meant there wasn't a direct way to incentivize student volunteers. We did recruit some students to help with monitoring Slack channels and the

Gather space—they were for the most part students working with organizing and program-committee members.

- **Note to 2021 organizers:** We did obtain an NSF grant to support student travel, but did not make any awards from it. We have been informally told by the program director that we can use the funds in 2021, but the grant will need a no-cost extension.

13 Favorite Quotes:

We make no claim that these remarks are statistically representative, but they made us feel good.

On opening SIGMOD keynote: "What an amazing session. I wouldn't have been able to attend the conference in-person, so this having this virtual session is turning out to be a blessing! Thanks to the SIGMOD committee for offering this virtually and free for all!!"

From a sponsor: "This is the second virtual conference we do, and this is by far the best organized."

From an attendee: "thanks for the great organization overall, I know it is an incredible effort!"

Mohan: "Gather was also a lot of fun and a very novel experience."

Trip report: "This virtual conference was FANTASTIC."

Another trip report: "However, with Zoom, the magic happens. I can open up all sessions I'm interested in and mute the speaker via drop audio setting in Zoom. If I find the topic I want to hear more, I can instantly switch to the desired Zoom window, reset the audio setting, and listen to the talk."

One more trip report: "Overall, the conference is life-changing, and I felt grateful for the opportunity to participate." & "But thanks to Gather, I found that it became easier for my personality to come through when I was oblivious to who I was talking to, or when I was so carried away by my curiosity and burning questions."

From our final bulletin to participants: "The last workshop has finished, and SIGMOD/PODS 2020 is now history. We suspect it will be a landmark in most of your minds, separating SIGMOD/PODS Conferences into those pre-2020 and those post-2020. Even before all the adjustments brought on by the COVID-19 crisis, we planned to stream more of the sessions. Our registration of ~3000 shows that there is high demand for online access to the conference. If our community is serious about fostering diversity and inclusion, then remote participation should become a permanent option. We are proud of the adaptability and flexibility of the organizers, many of whom found themselves doing jobs much different than those they anticipated when they agreed to help. We are pleased at the level of engagement of participants, with substantial interaction in the Zoom Q&As, Slack channels, Social Networking events, and Gather space. We feel we largely succeeded in delivering a conference that preserved the core elements of an in-person conference: a high-quality technical program, provocative keynotes, timely tutorials and lively panels. There were some elements that we couldn't readily emulate in the on-line format (conference banquet, sponsor swag), but maybe someone will figure those out for the future. We received many good suggestions for additions and modifications leading up to the conference that we couldn't pursue for lack of lead time and cycles. Undo-redo recovery is resource intensive—ask Mohan!"

14 Selected Survey Responses and Zoom Analysis

We conducted a survey of attendees by sending them email on the last day of the conference. We received 175 responses (out of ~2950 people who registered). In this section we list some selected responses. Many of the questions are the same as those used at EDBT for those wishing to draw larger conclusions about online conferences.

For the Zoom log analysis, we note that we have only partial information, since some of the logs were in different formats, rendering it impossible to perform good aggregation. The number of distinct participants

was at least 1,912 from 55 countries, which is surely an underestimate, since not all logs were available in readily consumable form, and in particular, were not available for the SIGMOD keynotes. Additionally, the information that we had only recorded the total number of attendees for a session, which was generally greater than the maximum attendance observed at any particular point. Finally, workshop attendance was calculated only per day, not per session, since workshops were structured as a single Zoom meeting.

14.1 Sessions

Overall, session attendance was reported as high among respondents and according to the Zoom logs. While we only have partial information from those logs, the average number of people who attended a SIGMOD Research talk was 135. The highest attendance was the first keynote with 762.

Overall, the average number of attendees per session type was:

Demos	41
Industry	152
PODS	105
SIGMOD Research	135
Sponsors	173
SIGMOD Tutorials	97
SRC	39
Workshops	281

The average for PODS days was 214 attendees, though we note that this was highly variable and measured across days. (The PODS keynote had 334, and the Test-of-Time + Gems session reached 225.) The workshop numbers are only for those that used the conference provided hosting; workshops that used their own hosting are not included.

We saw 64% of respondents to the survey report going to fewer sessions than they would normally go to:

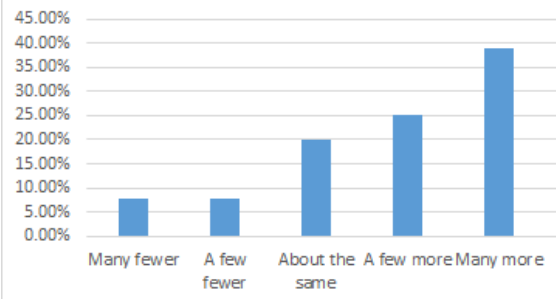

Figure 2: Compared to how many sessions you attended, how many sessions would you have attended if the conference had been physically located?

Respondents reported being overall neutral or happy both with the talks compared to in person talks:

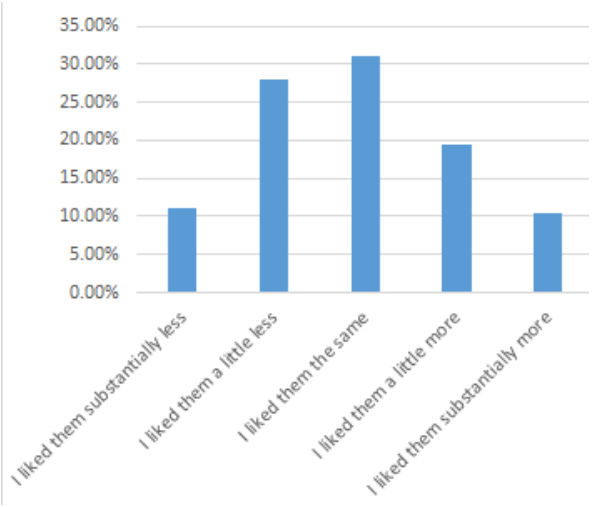

Figure 3: How did the online video presentations compare to conventional conference talks?

And with the questions and answers:

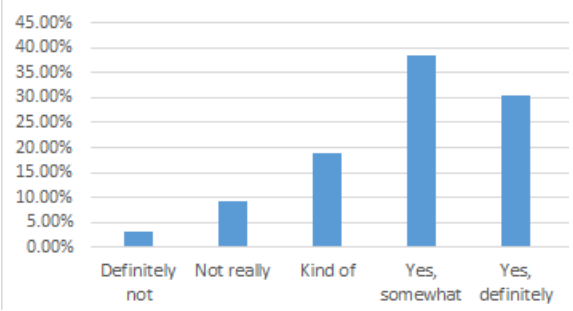

Figure 4: Did the interactiveness of the Q&A during sessions meet your needs and expectations?

14.2 Slack

Slack usage was higher than anticipated based on experiences in other conferences. As of Friday afternoon, 1328 people had Slack accounts for the conference. We exceeded the 10,000 message limit for message archives on a free account. (A paid account for the month would have cost >$10K.) This behavior may result from several factors, including the session and PC chairs being very proactive in seeding information in their sessions, and the frequent posts by Mohan. Overall, Slack was viewed quite positively.

People were generally happy with how helpful it was for questions and answers when the talks were not in session:

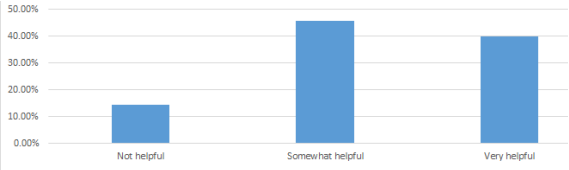

Figure 5: How helpful did you find the Slack channels for asking questions when the session was not being held?

14.3 Sponsor Talks

Due to the online format raising a concern as to how much visibility the sponsors would get, we included talks for the sponsors. These were both very popular by the numbers (there was an average of 173 people per sponsor talk—higher than the number of attendees at the research sessions) and with responses from attendees.

We saw that 44% of respondents attended at least one sponsor talk:

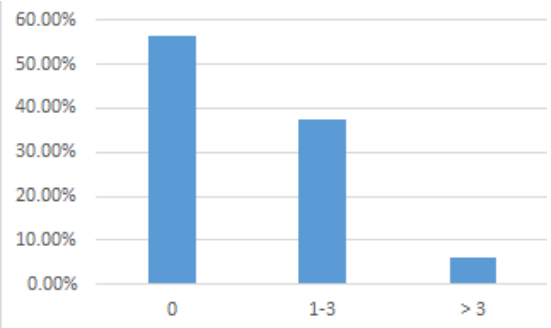

Figure 6:How many sponsor talks did you attend?

Of those who attended a sponsor talk 91% reported them to be somewhat or very useful:

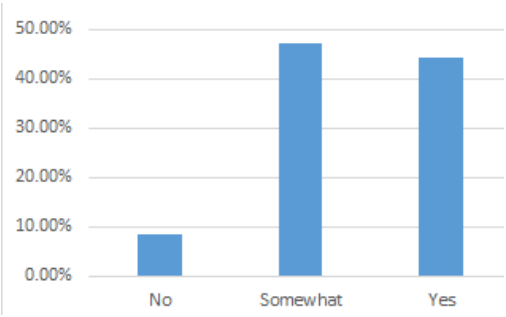

Figure 7: Did you find the sponsor talks to be useful?

There is appetite for continuing sponsor talks in the future, whether the conference is physical or virtual, with 75% of those who responded (117 individuals) saying that we should continue having the talks even at physical conferences:

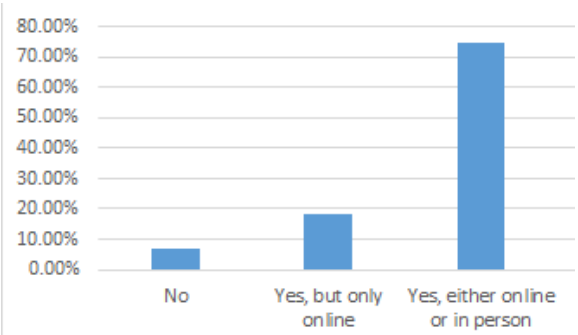

Figure 8: Should we consider having sponsor talks at future SIGMODs?

14.4 Social and networking options

The social and networking events were well received, even though obviously nothing can replace in-person options.

46% of respondents reported that they attended at least one social or networking event:

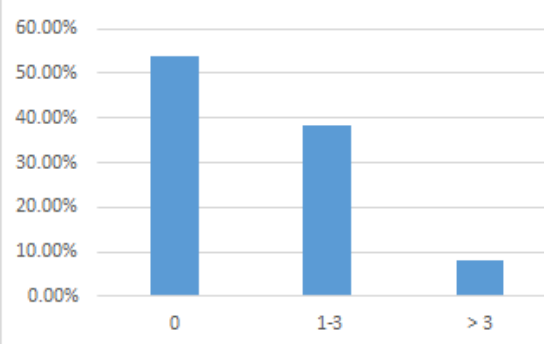

Figure 9: How many social and networking events (Zoomtables, Zoomside Chats, Retirement Party, Gather Parties) did you attend?

Given the limitations of an online platform, the fact that 36% of attendees thought that there were enough social and networking options should be seen as a positive:

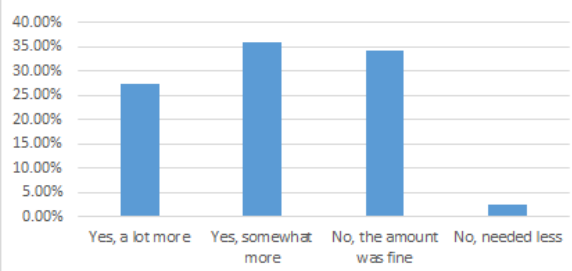

Figure 10: Did the conference need more social interaction?

Gather was a mixed success. Only 40% of respondents (who, given that they took the time to respond to the survey, seem more likely than the average attendee to be interested in such things) used Gather:

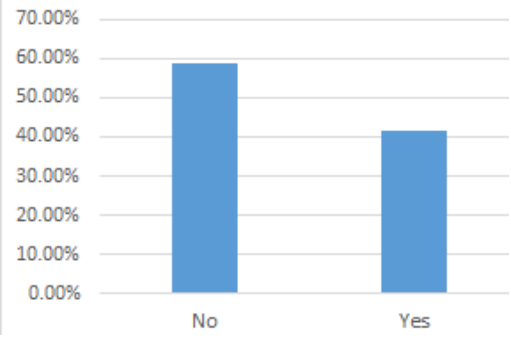

Figure 11: Did you use Gather?

However, of those who used Gather, 70% liked it either a lot or a great deal:

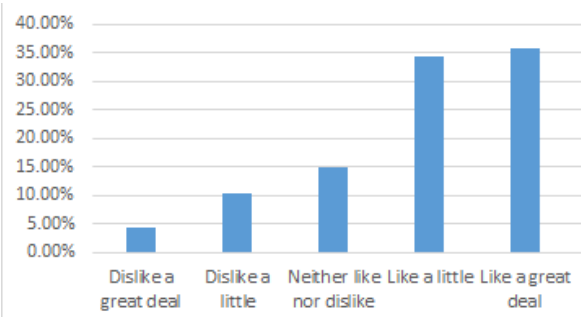

Figure 12: How much did you like using Gather?

Given that the platform was just in its infancy, this response is highly encouraging, and we recommend those who are putting on future virtual conferences to consider this or similar platforms.

Poly'19 Workshop Summary: GDPR

Michael Stonebraker
CSAIL
MIT
stonebraker@csail.mit.edu

Timothy Mattson
Parallel Computing Lab
Intel Corporation
timothy.g.mattson@intel.com

Tim Kraska
CSAIL
MIT
kraska@mit.edu

Vijay Gadepally
Lincoln Laboratory
MIT
vijayg@ll.mit.edu

ABSTRACT

Data privacy within the context of heterogenous data and data management systems continues to be an important issue. At the Poly'19 workshop, held in conjunction with VLDB 2019 in Los Angeles, CA, one of the major themes explored was the implication of data privacy regulations such as GDPR to systems composed of multiple heterogenous databases. This summary outlines some of the major approaches and directions presented by various presenters during the privacy portion of the Poly'19 workshop.

1. INTRODUCTION

In conjunction with VLDB'19, we organized the fourth annual workshop on Polystores (POLY'19 [1]). Half of the workshop focused on traditional topics concerning polystore data systems [1, 2]. The other half of the workshop focused on privacy; in particular, regulations such as the General Data Privacy Regulations (GDPR) [9] and their implications for data systems composed of multiple heterogeneous databases. Tim Kraska joined the Poly'19 organizing committee specifically to help us create a strong program on privacy. In this workshop summary, our focus is only on the GDPR and privacy related portion of the workshop.

2. PRIVACY REGULATIONS

It might be tempting to deemphasize GDPR, treating it as largely a European issue. We note, however, that a new law in California based on GDPR (CCPA [7]) went into effect in January 2020. It is expected that other US states will soon follow California's lead. Hence, the European Union's developments in regulating privacy, as defined by GDPR, will have a global impact.

Workshop keynotes were given by Daniel Weitzner (Founding Director, MIT Internet Policy Research Initiative) and Blaise Aguera y Arcas (Google Incorporated). Two articles [4, 5], from the privacy portion of the workshop, addressed legal and semantic topics of implementing GDPR. In [4], the authors highlight a number of privacy related topics propose a research agenda to bridge the gap between traditional security mechanisms and the needs amplified by polystore and federated databases. The authors of [4] also discuss specific policies such as GDPR, within the context of these research questions. Specifically, the authors propose the following research priorities: (1) matching legal requirements with technology capablities, (2) development of accountable privacy preserving systems, and (3) bringing privacy preserving techniques to private islands. The other policy article, [5], focused on GDPR compliance of large cloud providers and analyzed cases of potential non-compliance. They use these examples to propose best-practices for organizations interested in developing their own GDPR privacy policies. The authors provide a thorough analysis of ten cloud services such as Bloomberg and Uber and highlight particular patterns that may indicate GDPR non-compliance. For readers developing their own privacy policy, the authors also provide practical policy and technology recommendations that can be used.

Four additional papers from the session [10, 8, 6, 3] from the workshop dealt with implementing GDPR privacy guarantees and the majority of the discussion in this short summary will focus on these four technical articles. We will further constrain the scope of this discussion by focusing on just two of

[1] https://sites.google.com/view/poly19/

the guarantees defined by GDPR:

- **The right to be forgotten.** Upon a request, personal data on an individual must be deleted unless other laws take precedent. This requirement applies primarily to social media and other web sites that sell personal profiling information for monetary gain. Users of such services must be able to "opt out". There are numerous anecdotes of enterprises spending millions of dollars to find such information in legacy systems where copying of data is rampant. Supporting GDPR (or the similar California regulation) requires such expenditures, and future systems will have to be cognizant of such requirements.
- **Support for restrictions on data usage, so-called "purposes".** Database Management Systems (DBMSs) have long supported access control services. In SQL DBMSs, such services are standardized and are based on users and roles and apply to tables and views. Essentially all SQL DBMSs support such access control mechanisms using metadata that is checked at run time. However, GDPR expands this notion to the concept of purposes. For example, a user might be allowed to access certain data while helping a customer debug a problem, but not to run a marketing campaign. Presently, SQL access control cannot distinguish between these two cases.

We will discuss these two topics in terms of a simple running example. Consider the following data stored about an individual:

- `Email address (key)`
- `Preferred IP address`
- `Zipcode`
- `Department`
- `Expected salary`
- `Expected age`
- `Expected sex`
- `Expected political persuasion`

Additionally, consider the following three purposes:

1. Retrieve a cell value for some unknown purpose
2. Aggregate a cell value with at least 20 other cell values for statistical purposes
3. Use cell value to help target advertising messages

GDPR allows a consumer to specify, for each cell, which purposes they will allow. That could ultimately be three purposes times seven different attributes, each of which could have a separate "yes" or "no" answer. Much "coarser" granularity is possible, and sites may restrict the number of purposes and group cells into collections to reduce the amount of information required. In time, we expect fine-grained granularity will be the required service, potentially organized in a hierarchy. A typical site might have 1 million customers, each of which may constrain allowed purposes and make requests to be forgotten.

3. IMPLEMENTATION APPROACHES

The four implementation papers in the Poly'19 workshops can be grouped into three technical approaches:

3.1 Approach #1: [8, 10, 6]

Store each user's data in a separate physical "segment", which we term a "shard". Hence, raw data is partitioned into shards, one per user. In our running example, there will be 1 million shards. Then, a pipeline of operations is proposed to generate materialized views (MVs) of interest. These MVs aggregate information from the 1 million source shards. A user query simply goes to the correct MV and runs his/her particular query. In this approach, the pipeline will contain normalization operations (into common units for example) as well as SQL operations. However, to be mindful of purposes, each MV must have data allowed for only one purpose. Consider an MV which aggregates salary by department. There will have to be three versions of this MV, one for each purpose, and a user will have to query the one that corresponds to their purpose. In such a system, significant redundancy is inevitable.

In read-almost-always decision support applications, this approach will probably work. However, on every update some-to-most of the MVs will have to be recreated. If one is lucky, this can be done incrementally. For example, if the department of "Joe" is changed from "candy" to "shoe" and a particular MV is storing the average salary by department, then the two departments can be readily adjusted without examining other data. On the other hand, if the MV contains a join, then incremental update becomes much more difficult. In fact, some vendors make no effort to update MVs, choosing to let them "decay" over time, recreating them periodically. Whenever, a user shard is deleted,

the same propagation issues arise. For applications where there are substantial updates, approach 1 is not likely to work well.

Lastly, there is a substantial data discovery problem to decide which MVs must be interrogated. If a consumer wants information on both age and salary, they will likely have to interrogate two MVs, one for age and one for salary. This will be required whenever a consumer allows different access by cell for the various purposes.

Wang et. al. [10] have a similar pipeline approach. However, the "shard" for each user is associated with a collection of constraints on access. Think of these as a mix of SQL access control and GDPR purposes managed through a rule-oriented constraint language. The paper demonstrates that a rule language is rich enough to define GDPR specifications. That is not surprising since GDPR deals mostly with constrained access where rule systems are an obvious mechanism. If an analysis program wishes to access multiple shards to create an output shard (think of this as constructing a materialized view), then the system proposed in [10] can automatically and quickly figure out which constraints apply to the output shard. Hence, [10] has a mechanism to propagate access constraints through a pipeline of materialized views. However, for GDPR compliance, this amounts to simply taking the intersection of all the allowed purposes for the various cells. Lastly, this approach has all of the issues noted above concerning the solution in [8].

On the other hand, Pasquier et al [6] deal with GDPR at the operating system level and do not assume all data is stored in a DBMS. Given a pipeline of operations, this paper allows them to be a "mix and match" of operations in various subsystems (web servers, Hadoop, etc.). They assume provenance is stored about each operation. In effect, this is "who did what to whom". At the OS level all semantics is lost, and it is not clear how to support purposes. In addition, it is not clear what purposes even means in [6]. Lastly, it is not clear how the approach in [6] will continue to work as systems scale to large sizes.

In summary, Approach #1 is delete optimized. It is straightforward to support the right to be forgotten; one merely deletes the indicated user shard and recreates or updates all downstream MVs. It seems plausible to include a purpose system into this pipeline as proposed in [2]

3.2 Approach #2: [3]

Here, the focus is on a "clean" entity-relationship (E-R) DBMS schema for data that stores each fact exactly once. Materialized views in this approach must be disallowed, since they introduce replication, which make it difficult to find and delete personal data. Therefore, this approach advocates merely adding a table of user identifiers which are connected to the "clean" schema with the relationship "is relevant to". As such, deleting a user's information is easy, as one need only follow the E-R diagram from a user's entry point deleting information along the way. In fact [3] shows an implementation where this deletion can be performed in a lazy fashion. In effect, this is a different way to optimize for deletions.

Physically clustering E-R data to optimize retrievals is an interesting future exercise. As such, this schema will have no redundancy and consume minimal space. In contrast to Approach 1, this scheme will optimize space consumed (redundancy) to achieve better deletion performance. There is no recomputation time on deletes, but worse (perhaps significantly worse) query performance.

In addition, [3] proposed storing the purposes that apply to each cell as a bit vector for each cell. This will optimize purpose-oriented processing but, of course, require (perhaps significant) additional space.

Note that both Approaches #1 and #2 make onerous restrictions on the schemas/applications supported. It is widely known that the "clean" schemas required by Approach #2 may have poor run-time performance, since most enterprises diverge from clean schemas for performance reasons, and materialized views are introduced for the same reason. Arguably, Approach #2 also requires substantial changes to existing applications. Approach #1, on the other hand, will fail badly when there are significant updates. Therefore, we now turn to a third approach, that can deal effectively with legacy environments.

3.3 Approach #3:

Our final approach assumes the schema is arbitrary. We see this approach in [3] which includes a proposal for situations where the schema is unconstrained. This case, of course, applies to legacy information systems where the schema has evolved without regard for GDPR. Unlike in Approach #1 where there is a stylized way for information to flow among MVs, Approach #3 must cope with arbitrary data movement. There seems no other way to support this, other than maintaining detailed lineage to keep track of such information redundancy. The overhead of such a scheme is expected to be very onerous unless GDPR is supported at a coarse-grained level, which might have limitations of its

own.

In summary, the three approaches differ in:

- Generality of schemas supported
- Space consumed
- Cost of supporting the right to be forgotten
- Cost of supporting purposes

Hopefully, our discussions in this summary will generate additional proposals and a more detailed analysis of the options for addressing GDPR. Additional work is certainly warranted. Also, one is always reminded that the devil is in the details. There are many unaddressed issues:

- What happens when data is not associated with a single user? I.e. what about a marriage between two persons?
- As is often pointed out in these papers, a log fundamentally violates the right to be forgotten. I.e. one can "forget" information on a human in the database, but that information is still in the database log. Selectively purging the log will make certain kinds of recovery impossible. In other words, one is well advised to trust the DBAs, who have access to the log. Multiple "corner cases" exist of this sort.
- Record-level provenance, whether supported by the OS [6] or the DBMS [3] should be merged into a DBMS log to avoid duplication of effort and data. How to structure such a log is an interesting question.
- Compilation. To achieve high performance, purposes and provenance will have to be aggressively compiled. This will, of course, interfere with flexibility and changeability.
- How to support external applications or data movement across companies. For example, what happens if a user creates a Python notebook and copies some data into it to build an ML model. If a user requests to be forgotten, the whole Python notebook has to be deleted as an update to the notebook is not always possible.
- How can an entire company be efficiently transitioned to be GDPR compliant?

4. CONCLUSIONS

Hopefully, this summary and the privacy papers from Poly'19 will spur others to consider implementation issues raised by GDPR, especially ideas that make it relatively easy to support the right to be forgotten. The Poly'20 workshop[2] is scheduled to be held in conjunction with VLDB 2020 and we look forward to additional research and proposals in this area.

5. ACKNOWLEDGEMENTS

The authors wish to thank the program committee of the Poly'19 workshop.

6. REFERENCES

[1] Gadepally, V., Mattson, T., Stonebraker, M., Wang, F., Luo, G., Laing, Y., and Dubovitskaya, A. *Heterogeneous Data Management, Polystores, and Analytics for Healthcare: VLDB 2019 Workshops, Poly and DMAH, Los Angeles, CA, USA, August 30, 2019, Revised Selected Papers*, vol. 11721. Springer Nature, 2019.

[2] Gadepally, V., Mattson, T., Stonebraker, M., Wang, F., Luo, G., and Teodoro, G. *Heterogeneous Data Management, Polystores, and Analytics for Healthcare: VLDB 2018 Workshops, Poly and DMAH, Rio De Janeiro, Brazil, August 31, 2018, Revised Selected Papers*, vol. 11470. Springer Nature, 2019.

[3] Kraska, T., Stonebraker, M., Brodie, M., Servan-Schreiber, S., and Weitzner, D. Schengendb: A data protection database proposal. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 24–38.

[4] Kroll, J. A., Kohli, N., and Laskowski, P. Privacy and policy in polystores: A data management research agenda. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 68–81.

[5] Mohan, J., Wasserman, M., and Chidambaram, V. Analyzing gdpr compliance through the lens of privacy policy. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 82–95.

[6] Pasquier, T., Eyers, D., and Seltzer, M. From here to provtopia. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 54–67.

[7] Rothstein, M. A., and Tovino, S. A. California takes the lead on data privacy law. *Hastings Center Report 49*, 5 (2019), 4–5.

[8] Schwarzkopf, M., Kohler, E., Kaashoek, M. F., and Morris, R. Position: Gdpr compliance by construction. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 39–53.

[9] Tankard, C. What the gdpr means for businesses. *Network Security 2016*, 6 (2016), 5–8.

[10] Wang, L., Near, J. P., Somani, N., Gao, P., Low, A., Dao, D., and Song, D. Data capsule: A new paradigm for automatic compliance with data privacy regulations. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 3–23.

[2] https://sites.google.com/view/poly20/home