SIGMOD Officers, Committees, and Awardees

Chair
Divyakant Agrawal
Department of Computer Science
UC Santa Barbara
Santa Barbara, California
USA
+1 805 893 4385
agrawal <at> cs.ucsb.edu

Vice-Chair
Fatma Ozcan
Systems Research Group
Google
Sunnyvale, California
USA
+1 669 264 9238
Fozcan <at> google.com

Secretary/Treasurer
Rachel Pottinger
Department of Computer Science
University of British Columbia
Vancouver
Canada
+1 604 822 0436
Rap <at> cs.ubc.ca

SIGMOD Executive Committee:

Divyakant Agrawal (Chair), Fatma Ozcan (Vice-chair), Rachel Pottinger (Treasurer), Juliana Freire (Previous SIGMOD Chair), Chris Jermaine (SIGMOD Conference Coordinator and ACM TODS Editor in Chief), Rada Chirkova (SIGMOD Record Editor), Angela Bonifati (2022 SIGMOD PC co-chair), Amr El Abbadi (2022 SIGMOD PC co-chair), Floris Geerts (Chair of PODS), Sihem Amer-Yahia (SIGMOD Diversity and Inclusion Coordinator), Sourav S Bhowmick (SIGMOD Ethics)

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 Directors:

Sourav S Bhowmick, Nanyang Technological University
Byron Choi, Hong Kong Baptist University

Associate Information Directors:

Huiping Cao (SIGMOD Record), Georgia Koutrika (Blogging), Wim Martens (PODS)

SIGMOD Record Editor-in-Chief:

Rada Chirkova, NC State University

SIGMOD Record Associate Editors:

Lyublena Antova, Marcelo Arenas, Manos Athanassoulis, Renata Borovica-Gajic, Vanessa Braganholo, Susan Davidson, Aaron J. Elmore, Wook-Shin Han, Wim Martens, Kyriakos Mouratidis, Dan Olteanu, Tamer Özsu, Kenneth Ross, Pınar Tözün, Immanuel Trummer, Yannis Velegrakis, Marianne Winslett, and Jun Yang

SIGMOD Conference Coordinator:

Chris Jermaine, Rice University

PODS Executive Committee:

Floris Geerts (chair), Pablo Barcelo, Leonid Libkin, Hung Q. Ngo, Reinhard Pichler, Dan Suciu

Sister Society Liaisons:

Raghu Ramakhrishnan (SIGKDD), Yannis Ioannidis (EDBT Endowment), Christian Jensen (IEEE TKDE)

SIGMOD Awards Committee:

H.V. Jagadish (Chair), Stefano Ceri, Yanlei Diao, Samuel Madden, Volker Markl, Sayan Ranu, Barna Saha, Wang-Chiew Tan

Jim Gray Doctoral Dissertation Award Committee:

Wolfgang Lehner (co-chair), Gustavo Alonso (co-chair), Azza Abouzied, Daniel Deutch, Evaggelia Pitoura, Xiaofang Zhou, Huanchen Zhang, and Chenggang Wu

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)
Dan Suciu (2022)
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)
Joseph M. Hellerstein (2023)
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)
Alon Halevy (2021)

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); Michael Blanton, Adam Bolton, Bill Boroski, Joel Brownstein, Robert Brunner, Tamas Budavari, Sam Carliles, Jim Gray, Steve Kent, Peter Kunszt, Gerard Lemson, Nolan Li, Dmitry Medvedev, Jeff Munn, Deoyani Nandrekar-Heinis, Maria Nieto-Santisteban, Wil O'Mullane, Victor Paul, Don Slutz, Alex Szalay, Gyula Szokoly, Manu Taghizadeh-Popp, Jordan Raddick, Bonnie Souter, Ani Thakar, Jan Vandenberg, Benjamin Alan Weaver, Anne-Marie Weijmans, Sue Werner, Brian Yanny, Donald York, and the SDSS collaboration (2021); Michael Armbrust, Tathagata Das, Ankur Dave, Wenchen Fan, Michael J. Franklin, Huaxin Gao, Maxim Gekk, Ali Ghodsi, Joseph Gonzalez, Liang-Chi Hsieh, Dongjoon Hyun, Hyukjin Kwon, Xiao Li, Cheng Lian, Yanbo Liang, Xiangrui Meng, Sean Owen, Josh Rosen, Kousuke Saruta, Scott Shenker, Ion Stoica, Takuya Ueshin, Shivaram Venkataraman, Gengliang Wang, Yuming Wang, Patrick Wendell, Reynold Xin, Takeshi Yamamuro, Kent Yao, Matei Zaharia, Ruifeng Zheng, and Shixiong Zhu (2022); Aljoscha Krettek, Andrey Zagrebin, Anton Kalashnikov, Arvid Heise, Asterios Katsifodimos, Jiangji (Becket) Qin, Benchao Li, Bowen Li, Caizhi Weng, ChengXiang Li, Chesnay Schepler, Chiwan Park, Congxian Qiu, Daniel Warneke, Danny Cranmer, David Anderson, David Morávek, Dawid Wysakowicz, Dian Fu, Dong Lin, Eron Wright, Etienne Chauchot, Fabian Hueske, Fabian Paul, Feng Wang, Gabor Somogyi, Gary Yao, Godfrey He, Greg Hogan, Guowei Ma, Gyula Fora, Haohui Mai, Henry Saputra, Hequn Cheng, Igal Shilman, Ingo Bürk, Jamie Grier, Jark Wu, Jincheng Sun, Jing Ge, Jing Zhang, Jingsong Lee, Junhan Yang, Konstantin Knauf, Kostas Kloudas, Kostas Tzoumas, Kete (Kurt) Young, Leonard Xu, Lijie Wang, Lincoln Lee, Lungu Andra, Martijn Visser, Marton Balassi, Matthias J. Sax, Matthias Pohl, Matyas Orhidi, Maximilian Michels, Nico Kruber, Niels Basjes, Paris Carbone, Piotr Nowojski, Qingsheng Ren, Robert Metzger, Roman Khachatryan, Rong Rong, Rui Fan, Rui Li, Sebastian Schelter, Seif Haridi, Sergey Nuyanzin, Seth Wiesman, Shaoxuan Wang, Shengkai Fang, Shuyi Chen, Sihua Zhou, Stefan Richter, Stephan Ewen, Theodore Vasiloudis, Thomas Weise, Till Rohrmann, Timo Walther, Tzu-Li (Gordon) Tai, Ufuk Celebi, Vasiliki Kalavri, Volker Markl, Wei Zhong, Weijie Guo, Xiaogang Shi, Xiaowei Jiang,

Xingbo Huang, Xingcan Cui, Xintong Song, Yang Wang, Yangze Guo, Yingjie Cao, Yu Li, Yuan Mei, Yun Gao, Yun Tang, Yuxia Luo, Zhijiang Wang, Zhipeng Zhang, Zhu Zhu, Zili Chen (2023)

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)
K. Selcuk Candan (2023)

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)
Divesh Srivastava (2021)

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)
Christian S. Jensen (2022)

SIGMOD Jim Gray Doctoral Dissertation Award

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

- **2006** *Winner*: Gerome Miklau. *Honorable Mentions*: Marcelo Arenas and Yanlei Diao
- **2007** *Winner*: Boon Thau Loo. *Honorable Mentions*: Xifeng Yan and Martin Theobald
- **2008** *Winner*: Ariel Fuxman. *Honorable Mentions*: Cong Yu and Nilesh Dalvi
- **2009** *Winner*: Daniel Abadi. *Honorable Mentions*: Bee-Chung Chen and Ashwin Machanavajjhala
- **2010** *Winner:* Christopher Ré. *Honorable Mentions*: Soumyadeb Mitra and Fabian Suchanek
- **2011** *Winner*: Stratos Idreos. *Honorable Mentions*: Todd Green and Karl Schnaitterz
- **2012** *Winner*: Ryan Johnson. *Honorable Mention*: Bogdan Alexe
- **2013** *Winner*: Sudipto Das, *Honorable Mention*: Herodotos Herodotou and Wenchao Zhou
- **2014** *Winners*: Aditya Parameswaran and Andy Pavlo.
- **2015** *Winner*: Alexander Thomson. *Honorable Mentions*: Marina Drosou and Karthik Ramachandra
- **2016** *Winner*: Paris Koutris. *Honorable Mentions*: Pinar Tozun and Alvin Cheung
- **2017** *Winne*r: Peter Bailis. *Honorable Mention*: Immanuel Trummer
- **2018** *Winne*r: Viktor Leis. *Honorable Mention*: Luis Galárraga and Yongjoo Park
- **2019** *Winne*r: Joy Arulraj. *Honorable Mention*: Bas Ketsman
- **2020** *Winner:* Jose Faleiro. *Honorable Mention:* Silu Huang
- **2021** *Winner:* Huanchen Zhang, *Honorable Mentions:* Erfan Zamanian, Maximilian Schleich, and Natacha Crooks
- **2022** *Winner:* Chenggang Wu, *Honorable Mentions:* Pingcheng Ruan and Kexin Rong
- **2023** *Winner:* Supun Nakandala, *Honorable Mentions:* Benjamin Hilprecht and Zongheng Yang

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

[Last updated: June 1, 2023]

Editor's Notes

Welcome to the September 2023 issue of the ACM SIGMOD Record!

This issue starts with the Database Principles column presenting an article by Dong and Yi on query evaluation under differential privacy. In the scope of the pressing challenges in processing sensitive information at scale and in privacy-preserving ways, the authors consider the holy grail of developing a general-purpose SQL-based query engine that is capable of supporting a broad range of queries while maintaining differential privacy. Toward that goal, the article presents recent research advances in query evaluation under differential privacy, and outlines interesting directions for further investigation.

The Surveys column features a contribution by Seufitelli and colleagues. The article presents a systematic literature review on the areas of overlap between digital forensics and database systems. In an effort to promote better categorization and synthesis, the authors introduce a new taxonomy, which brings to light a pattern of publications that would enable researchers and organizations to quickly find solutions grouped by forensic purpose. The authors draw conclusions from their findings and list potential opportunities for future research.

The Systems and Prototypes column presents an article by Beedkar and colleagues that introduces Apache Wayang (Incubating), an open-source framework for unifying data analytics in a systematic way by integrating multiple heterogeneous data-processing platforms. The authors present the architecture of Apache Wayang, describe its components, and give an outlook on future directions of work.

The Reminiscences on Influential Papers column features contributions by Renata Borovica-Gajic and Bailu Ding.

The DBrainstorming column, whose goal is to discuss new and potentially controversial ideas that might be of interest and potentially of benefit to the research community, presents an article by Salihoglu. The article ponders research questions around architecting modern graph database-management systems (GDBMS), under the premise that they are relational at the core but can process "beyond relational" workloads. The author shares the experiences of the research team working on the Kùzu GDBMS, and formulates areas of inspiration for student researchers, including studies of systems with general deductive capabilities.

The Industry Perspectives column features an article by Poppe and colleagues that discusses challenges encountered with reactive resource allocations in modern cloud services, and introduces the proactive resource-allocation policy for Microsoft Azure Cognitive Search. The authors discuss the insights gained from the available production-workload patterns, and outline directions of further work.

The Open Forum column presents an article by Amer-Yahia and colleagues that explores the challenges and opportunities related to large language models (LLMs) in database and data-science research and education. The authors discuss a number of intriguing topics, including the pros and cons of LLMs, uses of LLMs in database systems and in education, and the need for reasoning in LLMs.

The issue closes with an announcement from the ACM on the new authorship policy, which covers a range of key topics, including the use of generative AI tools.

On behalf of the SIGMOD Record Editorial board, I hope that you enjoy reading the September 2023 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 2023

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)

Query Evaluation under Differential Privacy

Wei Dong
Hong Kong University of Science and Technology
wdongac@cse.ust.hk

Ke Yi
Hong Kong University of Science and Technology
yike@cse.ust.hk

ABSTRACT

Differential privacy has garnered significant attention in recent years due to its potential in offering robust privacy protection for individual data during analysis. With the increasing volume of sensitive information being collected by organizations and analyzed through SQL queries, the development of a general-purpose query engine that is capable of supporting a broad range of queries while maintaining differential privacy has become the holy grail in privacy-preserving query release. Towards this goal, this article surveys recent advances in query evaluation under differential privacy.

1. INTRODUCTION

Suppose a data analyst is interested in the total number of items sold this year where the customer and supplier are from the same nation. S/he would issue the following SQL query (assuming the TPC-H schema):

```
SELECT count(*)
FROM Customer, Orders, Supplier, Lineitem
WHERE Orders.Orderdate > 2023-01-01
  AND Lineitem.SK = Supplier.SK
  AND Supplier.NK = Customer.NK
  AND Customer.CK = Orders.CK
  AND Orders.OK = Lineitem.OK;
```

Such queries are very common in today's data analytical tasks and are a central problem in databases, which have been extensively studied in the literature. Sophisticated query processing algorithms and systems have been and are continually being developed and optimized throughout the years.

In recent years, a new direction for query processing concerns with the problem of how to release query results while respecting the privacy of the individuals who have contributed their data to the database. For example, the query above clearly relies on the data from the customers and suppliers, and it has been shown that, if the results of a certain number of such queries are available, then the data of the customers/suppliers can be reconstructed with enough accuracy [15]. Meanwhile, privacy-protection laws, such as the General data protection regulation (GDPR) [43], have been enacted across the world, making it a legal responsibility that companies and governments must handle personal data carefully.

Among the many privacy definitions, *differential privacy* (DP) [24] has become the de facto standard for privacy-preserving query release. It requires that the presence or absence of any individual's data should not change the distribution of the query result significantly, so that the adversary cannot infer (with a certain level of confidence) whether any individual has contributed to the database or not. More formally, let $\mathcal{I}$ be the space of all database instances, Q a query, and $M_Q : \mathcal{I} \to \mathcal{Y}$ a query-answering algorithm, often called a *mechanism* in the DP literature. The mechanism M_Q is said to satisfy (ε, δ)-DP if

$$\mathsf{Pr}[M_Q(\mathbf{I}) \in Y] \leq e^{\varepsilon} \cdot \mathsf{Pr}[M_Q(\mathbf{I}') \in Y] + \delta \qquad (1)$$

for any subset of outputs $Y \subseteq \mathcal{Y}$ and any pair of *neighboring instances* $\mathbf{I} \sim \mathbf{I}'$ (to be elaborated shortly). Here, ε, δ are the privacy parameters, also called the privacy budget. Typically, ε is a constant ranging from 0.1 to 10, with smaller values corresponding to stronger privacy guarantees. On the other hand, δ should be much smaller than $1/N$ to ensure the privacy of individual tuples, where $N = |\mathbf{I}|$ is the instance size; in particular, the case where $\delta = 0$ is referred to as *pure DP*, which is more desirable. Note that a DP mechanism must be randomized by definition, and some noise has to be injected to the true query result $Q(\mathbf{I})$. Thus, the central problem in DP is to find the optimal trade-off between privacy (i.e., ε, δ) and utility (i.e., how much noise is injected).

1.1 DP Policies in Relational Databases

The unspecified neighboring relationship $\mathbf{I} \sim \mathbf{I}'$ in the definition above depends on the data model and privacy requirement. For a single table (relation), the standard definition is that $\mathbf{I} \sim \mathbf{I}'$ if one contains one more tuple than the other. However, in a database with multiple relations possibly with foreign-key (FK) constraints, the situation is more subtle. Two neighboring relationships have been proposed and extensively studied, resulting in two different DP policies: tuple-DP [33, 41, 37, 42, 28, 19, 20] and user-DP [45, 34, 16, 18]. The nomenclature of these two policies reflects their respective aims: tuple-DP safeguards the tuples, while user-DP preserves the privacy of users, who may possess multiple tuples.

Tuple-DP is a straightforward generalization of the single-table case: neighboring instances differ by the

addition or deletion of a single tuple in any relation, while FK constraints are ignored. In contrast, user-DP employs a more intricate definition of neighboring instances by taking the FK constraints into consideration. First, one or more relations are designated as the *primary private relations*, whose tuples are the "users" whose information we aim to protect, such as `Customer` and/or `Supplier`. Then, any relation that has an FK reference, direct or indirect, to a primary private relation is called a *secondary private relation*. A tuple in a secondary relation that has an FK reference (directly or indirectly) to a user, such as a lineitem in an order placed by a customer, is considered as data belonging to the user. Relations having no FK references to the primary private relations are public. Then $\mathbf{I}$ and $\mathbf{I}'$ are neighbors if one can be obtained from the other by deleting one user from the primary private relation and all his/her data from the secondary private relations.

Note that when all relations are taken as primary private relations and there are no FK constraints, user-DP degenerates into tuple-DP. Thus, user-DP is more general, hence more difficult to achieve and often makes the utility worse, but it offers stronger and more flexible privacy policies. Which policy to adopt depends on what information is considered private and needs protection. For example, in the TPC-H schema, tuple-DP only protects the privacy of tuples in the relations, such as whether a customer has placed a particular order, whether a particular item is in a given order, and whether a supplier provides a certain item. In contrast, user-DP protects all information about each customer/supplier. Note that when applied to the database schema $\{\texttt{Edge}(\texttt{src}, \texttt{dst}), \texttt{Node}(\texttt{ID})\}$, where `src` and `dst` have FK references to `ID`, user-DP degenerates into *node-DP* (by designating `Node` as the primary private relation) [31, 10, 14] while tuple-DP becomes *edge-DP* [39, 10, 47, 30], both of which have been extensively studied in private graph analysis.

1.2 Classification of Queries

A wide range of queries have been studied under both tuple-DP and user-DP. Below we classify them according to the operators used: Selection, Projection, Join, and Aggregation (Count, Sum, and Max/Min). We assume that the aggregation attribute takes values from the non-negative integer domain[1] in this article, as most works do in this area. Note that for queries that return a subset of the tuples from the input, such as plain conjunctive queries, DP is hard to achieve, so they have not been considered in the literature.

SA Queries.

An SA query imposes a selection condition on a single relation, followed by an aggregation. The following is an example:

```
SELECT count(*) FROM Lineitem
WHERE Lineitem.Shipdate > 2023-01-01;
```

[1]The integer domain can be handled by separately processing the query for the non-negative and negative domains.

While SA queries have been extensively studied under tuple-DP, they are not considered under user-DP. To see why, consider the query above in a TPC-H database where `Customer` is designated as the primary private relation. Such a query is said to be *incomplete* under user-DP, as it does not include the primary private relation, which contains information of the users whose privacy we aim to protect. Thus, under user-DP, the query must first be made complete by iteratively adding relations whose PKs are referenced, together with the necessary PK-FK join condition, until the primary private relations are included (if possible). For example, the query above should be augmented to the following query, which becomes an SJA query:

```
SELECT count(*)
FROM Customer, Orders, Lineitem
WHERE Lineitem.Shipdate > 2023-01-01
  AND Customer.CK = Orders.CK
  AND Orders.OK = Lineitem.OK;
```

SJA Queries.

An SJA query uses selections, joins, followed by an aggregation. The query above is one such example. This example has no self-joins. On the other hand, many useful queries, such as pattern counting queries in a graph, extensively use self-joins. Another scenario where self-joins arise implicitly is user-DP with multiple primary private relations, as this case is first reduced to a single primary private relation. For example, when both `Customer` and `Supplier` are primary private relations, a virtual relation $\texttt{User}(\underline{\texttt{ID}})$ is built that contains all the PKs of these two relations while `Customer.CK` and `Supplier.SK` have FK references to `User.ID`. Then `User` is designated as the only primary private relation, while `Customer` and `Supplier` become secondary private relations. Now consider a query that involves both `Customer` and `Supplier`. After making the query complete as described above, the query contains a self-join on `User`, even if it is self-join-free originally. As will be seen later, the presence of self-joins makes the problem significantly more difficult, thus is often treated separately.

SPJA Queries.

Finally, the query may involve a (distinct) projection on certain attributes before the aggregation; the following is an example.

```
SELECT count(DISTINCT Customer.CK)
FROM Customer, Orders, Lineitem
WHERE Lineitem.Shipdate > 2023-01-01
  AND Customer.CK = Orders.CK
  AND Orders.OK = Lineitem.OK;
```

SPJA queries are the most general and the most difficult queries, and only Count aggregation has been studied in the literature. Thus, in this article, we only discuss Count aggregation when SPJA queries are concerned.

Query type		State-of-the-art Result	
		Tuple-DP	User-DP
Count /Sum	SA	[24]: $O(1)$-worst-case optimal	
	Self-join -free SJA	[19, 20]: $\big(O(1), O(1)\big)$-neighborhood optimal	[21]: $\big(1, O(1)\big)$-down neighborhood optimal
	Self-join SJA		[16]: $\big(1, \tilde{O}(1)\big)$-down neighborhood optimal
			[25]: $\big(1, \tilde{O}(1)\big)$-down neighborhood optimal
	SPJA	[19, 20]: No optimal guarantee	[16] and [25]: No optimal guarantee
		[25]*:$\big(\tilde{O}(1), \tilde{O}(1)\big)$-downward neighborhood optimal	
Max	SA	[21]: $\big(\tilde{O}(1), 2\big)$-down neighborhood optimal	
	SJA	[25]: $\big(\tilde{O}(1), 2\big)$-down neighborhood optimal	

Table 1: Summary of state-of-the-art results for answering SQL queries under tuple-DP and user-DP. All results achieve ε-DP and have polynomial running time except *.

1.3 Optimality Measures for Utility

As the output of a DP mechanism must be randomized, we often use a constant-probability error bound to measure its utility:

$$\mathrm{Err}(M_Q, \mathbf{I}) = \inf\Big\{\xi : \Pr\left[\|M_Q(\mathbf{I}) - Q(\mathbf{I})\| \le \xi\right] \ge 2/3\Big\}.$$

Most classical DP mechanisms are based on the notion of *sensitivity* of the query Q. First, the *local sensitivity* of Q at instance $\mathbf{I}$ is how much $Q(\mathbf{I})$ can change when $\mathbf{I}$ changes to one of its neighbors, i.e.,

$$\mathrm{LS}_Q(\mathbf{I}) = \sup_{\mathbf{I}', \mathbf{I} \sim \mathbf{I}'} \|Q(\mathbf{I}) - Q(\mathbf{I}')\|.$$

The *global sensitivity* of Q is

$$\mathrm{GS}_Q = \sup_{\mathbf{I}} \mathrm{LS}_Q(\mathbf{I}).$$

For a 1-dimensional query $Q : \mathcal{I} \to \mathbb{R}$, adding a Laplace noise proportionate to $\mathrm{GS}_Q/\varepsilon$ preserves ε-DP. This mechanism is referred to as the *Laplace mechanism*, which yields an error of $O(\mathrm{GS}_Q)$[2]. A simple case that can be handled by the Laplace mechanism is any SA-`Count` query under tuple-DP, for which we have $\mathrm{GS}_Q = 1$.

Worst-case optimality.

The classical optimality notion is *worst-case optimality*. Let $\mathcal{M}_Q$ be the class of all (ε, δ)-DP mechanisms for query Q. The worst-case lower bound is

$$\mathcal{L}_{\mathrm{wst}} = \inf_{M'_Q \in \mathcal{M}_Q} \sup_{\mathbf{I}' \in \mathcal{I}} \mathrm{Err}(M'_Q, \mathbf{I}').$$

It can be shown that for any constant ε, $\mathcal{L}_{\mathrm{wst}} \ge \mathrm{GS}_Q/2$ for any Q. So the Laplace mechanism is already worst-case optimal, not just for SA-`Count` queries, but for all Q. However, the Laplace mechanism is hardly a satisfactory, or even valid, solution to any query other than SA-`Count`, as GS_Q is often large or unbounded for many Q. Consider the SJA-`Count` query given at the beginning of the article. We can construct an $\mathbf{I}$ in which there is just one supplier and one customer, while all lineitems are shipped from this supplier to this customer. Then we delete the customer to obtain $\mathbf{I}'$ (for user-DP, we also need to delete all the lineitems). Such a pair of neighboring instances imply that $\mathrm{GS}_Q = \infty$, so the Laplace mechanism cannot be applied. One may artificially impose a limit on GS_Q, but this is not a theoretically elegant solution; in practice, this is not satisfying, either, since this limit must be set *a priori*, so is often conservatively large.

Instance optimality.

The failure of the Laplace mechanism means that some instance-specific optimality should be employed. The strongest such notion is *instance optimality*. More precisely, let

$$\mathcal{L}_{\mathrm{ins}}(\mathbf{I}) := \inf_{M'_Q \in \mathcal{M}_Q} \mathrm{Err}(M'_Q, \mathbf{I})$$

be the smallest error any $M'_Q \in \mathcal{M}_Q$ can achieve on $\mathbf{I}$. Then an DP mechanism M_Q is *c-instance optimal* if $\mathrm{Err}(M_Q, \mathbf{I}) \le c \cdot \mathcal{L}_{\mathrm{ins}}(\mathbf{I})$ for every $\mathbf{I}$, where c is called the *optimality ratio*. Unfortunately, for every $\mathbf{I}$, one can design a trivial $M'_Q(\cdot) \equiv Q(\mathbf{I})$ that has 0 error on $\mathbf{I}$ (but fails miserably on other instances), so $\mathcal{L}_{\mathrm{ins}}(\cdot) \equiv 0$, which rules out instance-optimal DP mechanisms.

Neighborhood optimality.

As instance optimality is unattainable, [8, 20] consider a relaxed version of instance optimality where we compare M_Q against any M'_Q that is required to work well not just on $\mathbf{I}$, but also on its k-neighbors, i.e., instances within distance[3] k from $\mathbf{I}$. More precisely, we

[2]The O notation omits the dependency on ε and log log factors, and the $\tilde{O}$ notation further omits polylogarithmic factors.

[3]Distance between $\mathbf{I}$ and $\mathbf{I}'$ is the number of individuals'

define the target error on $\mathbf{I}$ as

$$\mathcal{L}_{\text{nbr}}(\mathbf{I}, k) := \inf_{M'_Q \in \mathcal{M}_Q} \sup_{\mathbf{I}': d(\mathbf{I}, \mathbf{I}') \le k} \text{Err}(M'_Q, \mathbf{I}').$$

Then, M_Q is (k, c)-*neighborhood optimal* if $\text{Err}(M_Q, \mathbf{I}) \le c \cdot \mathcal{L}_{\text{nbf}}(\mathbf{I}, k)$ for every $\mathbf{I}$. Note that neighborhood optimality interpolates between instance optimality ($k = 0$) and worst-case optimality ($k = \infty$), with smaller values of k corresponding to stronger optimality.

Neighborhood optimality has been adopted for analyzing DP mechanisms for certain machine learning problems [8] and SJA-COUNT queries under tuple-DP [20]. However, it degenerates into worst-case optimality (for any $k \ge 1$), hence meaningless, when the query has a MAX or SUM aggregation. Consider an SA-MAX query that returns the highest salary of any customer. For every $\mathbf{I}$, we can construct a neighboring instance $\mathbf{I}'$ by adding a customer with an arbitrarily high salary, implying $\text{LS}_Q(\mathbf{I}) = \infty$. Vadhan [46] shows that $\mathcal{L}_{\text{nbr}}(\mathbf{I}, 1) \ge \text{LS}_Q(\mathbf{I})/2$, so $\mathcal{L}_{\text{nbr}}(\mathbf{I}, 1)$ is also unbounded. A similar construction works for a SA-SUM query or an SJA-COUNT query under user-DP, by just adding a user contributing to arbitrarily many tuples.

The reason why neighborhood optimality fails to work in these cases is that we require M'_Q to work well on any neighbor $\mathbf{I}'$ of $\mathbf{I}$. For these queries, there always exists a bad neighbor that contains a heavy contributor. This is too high a requirement for M'_Q, hence too low an optimality notion for M_Q.

Down-neighborhood optimality.

To address the issue, Dong et al. [16] revised $\mathcal{L}_{\text{nbr}}(\cdot, \cdot)$ to

$$\mathcal{L}_{\text{d-nbr}}(\mathbf{I}, k) := \min_{M'_Q \in \mathcal{M}_Q} \max_{\mathbf{I}': d(\mathbf{I}, \mathbf{I}') \le k, \mathbf{I}' \subseteq \mathbf{I}} \text{Err}(M'_Q, \mathbf{I}'),$$

namely, we require M'_Q to work well only on $\mathbf{I}'$ and its k-*down-neighbors*, which can be obtained by removing at most k users' data from $\mathbf{I}$. Then (k, c)-*down neighborhood optimality* is defined analogously. The $k = 1$ case is of particular interest, as it can be shown that $\text{DS}_Q(\mathbf{I}) \ge \mathcal{L}_{\text{nbr}}(\mathbf{I}, 1) \ge \text{DS}_Q(\mathbf{I})/2$ where $\text{DS}_Q(\mathbf{I})$ is the *downward local sensitivity* of Q at $\mathbf{I}$:

$$\text{DS}_Q(\mathbf{I}) = \max_{\mathbf{I}', \mathbf{I} \sim \mathbf{I}', \mathbf{I}' \subseteq \mathbf{I}} \big\| Q(\mathbf{I}) - Q(\mathbf{I}') \big\|.$$

Thus, we can use $\text{DS}_Q(\mathbf{I})$ as a proxy to prove down neighborhood optimality, which is easier and has simple interpretations for many queries. For example, for a query Q with COUNT or SUM aggregation, $\text{DS}_Q(\mathbf{I})$ is maximum user contribution in $Q(\mathbf{I})$. For an SA-MAX query, $\text{DS}_Q(\mathbf{I})$ is the gap between the maximum value and the second maximum value. Unlike $\text{LS}_Q(\mathbf{I})$, $\text{DS}_Q(\mathbf{I})$ is always bounded and usually small on most instances, so down-neighborhood optimality is more meaningful.

information they differ. Under user-DP, that refers to the number of different users while under tuple-DP, that represents the number of different tuples.

1.4 Overview of Results

Table 1 provides an overview of the state-of-the-art solutions for answering various queries under DP. Under tuple-DP, as mentioned, the Laplace mechanism [24] already achieves an error of $O(1)$ for SA-COUNT queries, which is $O(1)$-worst-case optimal. For SA-MAX queries, [21] achieves $\big(\tilde{O}(1)), 2\big)$-down neighborhood optimality. For SJA queries, [19, 20] achieve $\big(O(1), O(1)\big)$-neighbor -hood optimal error. For SJA queries with the MAX aggregation, while there is no dedicated work for this problem, the mechanism [25] designed for user-DP can be employed. This is possible because any user-DP mechanism can also handle tuple-DP, as mentioned in Section 1.1. For SPJA queries, [20] currently delivers the best performance but still lacks an optimal error guarantee.

Moving towards user-DP, for SJA queries, when the queries are self-join-free, [21] achieves $\big(1, O(1)\big)$-down neighborhood optimal error. For self-join queries, both [16] and [25] achieve $\big(1, \tilde{O}(1)\big)$-down neighborhood optimal error and the log factors hidden by $\tilde{O}$ in these solutions are not comparable. For SJA queries with the MAX aggregation, [25] achieves the $\big(\tilde{O}(1), 2\big)$-downward neighborhood optimal error. For SPJA queries, both [16] and [25] are applicable, yet neither guarantees optimal utility. Meanwhile, [25] gives another solution achieving $\big(\tilde{O}(1), \tilde{O}(1)\big)$-downward neighborhood optimal error while taking super-polynomial time. Throughout the article, we use *data complexity* [2] when talking about running times, i.e., the running time is measured as a function of the database size N, while the query size (i.e., number of relations and attributes in the query) is considered a constant. All the aforementioned solutions have polynomial running times, except for the last one. Furthermore, all of these solutions achieve pure-DP.

2. DP PROPERTIES

The following properties of DP will be useful:

LEMMA 1 (POST PROCESSING [24]). *If $M_{Q_1} : \mathcal{I} \to \mathcal{Y}$ satisfies (ε, δ)-DP and $M_{Q_2} : \mathcal{Y} \to \mathcal{Z}$ is any randomized mechanism, then $M_{Q_2}(M_{Q_1}(\mathbf{I}))$ satisfies (ε, δ)-DP.*

LEMMA 2 (COMPOSITION THEOREM [24]). *If M_Q is an adaptive composition[4] of differentially private mechanisms $M_{Q_1}, \ldots, M_{Q_k}$, where each M_{Q_k} satisfies (ε, δ)-DP, then M satisfies (ε', δ')-DP, where*

1. *$\varepsilon' = k\varepsilon$ and $\delta' = k\delta$; [Basic Composition]*
2. *$\varepsilon' = \varepsilon\sqrt{2k \log \frac{1}{\delta''}} + k\varepsilon(e^{\varepsilon} - 1)$ and $\delta' = k\delta + \delta''$ for any $\delta'' > 0$. [Advanced Composition]*

LEMMA 3 (GROUP PRIVACY [24]). *If M_Q is an $(\varepsilon_0, \delta_0)$-DP mechanism, then for any two instances $\mathbf{I}, \mathbf{I}'$ with $d(\mathbf{I}, \mathbf{I}') = \lambda$, M_Q satisfies $(\lambda\varepsilon_0, \lambda e^{\lambda\varepsilon_0}\delta_0)$-DP.*

[4] Adaptive composition refers to a sequence of mechanisms, where the choice of each mechanism can depend on the outcomes of the previous mechanisms.

LEMMA 4 (PARALLEL COMPOSITION [36]). *If M_{Q_1}, M_{Q_2} satisfy ε_1-DP and ε_2-DP, and $\mathcal{X}_1, \mathcal{X}_2 \subseteq \mathcal{X}$ are two disjoint input domains, then $\big(M_{Q_1}(\mathbf{I} \cap \mathcal{X}_1), M_{Q_2}(\mathbf{I} \cap \mathcal{X}_2)\big)$ satisfies $\max(\varepsilon_1, \varepsilon_2)$-DP.*

3. QUERY EVALUATION UNDER TUPLE-DP

3.1 SA Queries

As mentioned, SA-COUNT queries can be readily handled by the Laplace mechanism. SA-SUM queries can be handled as self-join-free SJA-COUNT queries under user-DP, which will be discussed in Section 4.1. This section thus only discusses SA-MAX queries. In this problem, $\mathbf{I}$ can be regarded as a multiset of integers $\{x_1, x_2, \ldots, x_N\}$, we reorder $\mathbf{I}$ such that $x_1 \leq \cdots \leq x_N$, and our goal is to design a DP mechanism M_Q such that

$$x_{N-\rho} \leq Q(\mathbf{I}) \leq x_N.$$

Such a guarantee is said to have a *rank error* of ρ. Note that this is equivalent to $(\rho, 2)$-down neighborhood optimality.

For this problem, Asi and Duchi introduced the *inverse sensitivity mechanism*, which instantiates the *exponential mechanism* [24], a fundamental ε-DP framework. The inverse sensitivity mechanism operates based on an assumption of data boundedness within the range $[0, U]$ and achieves ε-DP while maintaining a rank error of $O(\log(U))$.

Later, Huang et al. [26] proposed an alternative mechanism by transforming the maximum problem into a counting problem. This algorithm also requires a similar assumption of data boundedness as the inverse sensitivity mechanism. The high-level idea is to find the largest r such that $[r, U]$ contains more than c elements, where c is a predetermined parameter. By employing a binary search, the desired r can be located using $\log(U)$ counting queries. The composition theory is then utilized to allocate the privacy budget for each counting query. Their mechanism is under the *concentrated differential privacy* (CDP) [11], which is a DP notation between ε-DP and (ε, δ)-DP. By carefully selecting the parameter c, they achieve a rank error of $O\big(\sqrt{\log(U)\log\log(U)}\big)$. Moreover, there exists a corresponding ε-DP version of this mechanism by replacing the Gaussian mechanism with the Laplace mechanism. However, this version yields a higher rank error of $O\big(\log(U)\log\log(U)\big)$ than the inverse sensitivity mechanism.

Dong and Yi [21] also reduced the problem to a counting problem. However, they took a different route: Instead of seeking an interval $[r, U]$, they identify the smallest value of r such that the interval $[0, r]$ encompasses the majority of elements. More precisely, they iteratively set $r = 1, 2, \ldots$ and inquire whether $[0, r]$ contains more than $N - c$ data, where c is another predefined parameter. When applying the composition theory, this method has a large error for each counting query, further leading to a large rank error. Instead, they use the *sparse vector technique* [23]. The algorithm first uses constant noise to obscure the threshold $N - c$. During each iteration, the constant noise is added to mask the counter result for $[0, r]$, and this noisy counter is then compared with the noisy threshold. Finally, the algorithm returns the first r for which its noisy counter surpasses the noisy threshold. The entire process can be shown to preserve ε-DP. In comparison to binary search, this approach doesn't require the division of privacy budget but generates more noisy outcomes. Additionally, the boundedness assumption becomes unnecessary. Through careful selection of $c = \tilde{O}(1)$, the algorithm achieves an instance-specific rank error of $O(\log(x_{\text{Max}}))$, where x_{Max} is the maximum value in $\mathbf{I}$.

3.2 SJA Queries

Let us begin by discussing JA queries. Joins make the problem more challenging, as a single tuple can now influence numerous join results, and the global sensitivity becomes ∞. A relatively easy approach is to add constraints so as to reduce global sensitivity. McSherry [36] solves the problem by restricting to one-to-one joins. Proserpio et al. [42] propose wPINQ to extend the work of McSherry to support general equijoins: by assigning weights to tuples and scaling down the weights, their algorithm ensures each tuple can at most affect one on final counting result. However, this only works well when one tuple affects a fixed number of results. Palamidessi and Stronati [41] add constraints on the attribute range. Arapinis et al. [7] and Narayan et al. [37] consider functional dependencies and cardinality constraints.

Another way to deal with the issue of a high global sensitivity is tempting to use the sensitivity of the query on the particular given instance like local sensitivity $\text{LS}_Q(\mathbf{I})$. However, as pointed out by Nissim et al. [39], using the local sensitivity to calibrate noise is not DP. This is because the local sensitivity can be very different on two neighboring databases, so the noise level may reveal information about an individual tuple. Essentially, the problem is that local sensitivity, when considered as a query, has high global sensitivity.

To get around the problem, the idea is to use a smooth (i.e., having low global sensitivity) upper bound of the local sensitivity, named *smooth sensitivity* (SS_Q) [39]. Similar to local sensitivity, smooth sensitivity is also instance-dependent and usually can be much smaller than global sensitivity. But different from local sensitivity, it eliminates abrupt changes between neighboring instances, hence the name "smooth sensitivity". More precisely, for any $\mathbf{I} \sim \mathbf{I}'$, we have $\text{SS}_Q^\beta(\mathbf{I}) \leq \text{SS}_Q^\beta(\mathbf{I}) \cdot e^\beta$. By selecting β to be $\Theta(\varepsilon)$ and incorporating noise sampled from a general Cauchy distribution[5], scaled by $\text{SS}_Q^\beta(\mathbf{I})$, we get an ε-DP mechanism. Furthermore, it can be demonstrated that smooth sensitivity achieves $(O(1), O(1))$-neighborhood optimal error for answering multi-way join counting queries under tuple-DP [20]. However, computing the smooth sensitivity by defini-

[5]The general Cauchy distribution has pdf $h(z) \propto \frac{1}{1+|z|^\gamma}$.

tion in general takes exponential time. Dong and Yi devised a method to reduce its computational cost for multi-way join counting queries to $N^{O(\log N)}$, which is still super-polynomial.

Due to the absence of an efficient algorithm for calculating the smooth sensitivity for multi-way join counting queries under tuple-DP, Johnson et al. [28] introduced *elastic sensitivity*. Elastic sensitivity is an approximation of smooth sensitivity while preserving its "smoothness property". In contrast to smooth sensitivity, elastic sensitivity can be computed in linear time. However, it lacks any utility guarantee. Theoretically, the gap between elastic sensitivity and smooth sensitivity can be as large as $O\left(N^{n-1}\right)$.

To tackle this challenge, Dong and Yi proposed *residual sensitivity* [19, 20], which is another valid approximation of smooth sensitivity. For utility, residual sensitivity is a constant-factor upper bound on smooth sensitivity, which can be used to add noise, resulting in an $(O(1), O(1))$-neighborhood optimal error. In terms of efficiency, residual sensitivity can be computed through a constant number of AJAR/FAQ queries [27, 4] with $\tilde{O}(1)$ additional computations. Each AJAR/FAQ query can be processed within $O(N^w)$ time [40, 9], where w is its AJAR/FAQ width, a constant depending on the query only.

Then, let us consider the selection operations. The traditional approach to dealing with selection operation [34, 28, 19] is to evaluate the query with selection but compute the sensitivity without considering the selection conditions. This yields a valid DP mechanism but loses optimality. To see this, just consider an extreme case where a selection condition always returns False. Then the query becomes a trivial query and the optimal (under any notion of optimality) mechanism is $M(\cdot) \equiv 0$, i.e., $\mathrm{Err}(M, \mathbf{I}) = 0$ for all $\mathbf{I}$, but the sensitivity of the query without the selection condition must be nonzero. Meanwhile, Dong and Yi demonstrated how to extend residual sensitivity to incorporate selection operations while maintaining its neighborhood optimality [20]. Additionally, when all selection conditions are inequalities and comparisons, the algorithms can still be run in polynomial time.

3.3 SPJA Queries

The conventional approach for answering SPJA queries under tuple-DP simply disregards the projection. Dong and Yi extended residual sensitivity to more effectively handle projection so as to reduce the noise [20]. This extended algorithm can also be executed using a constant number of AJAR/FAQ queries. However, it does not have neighborhood optimality.

4. QUERY EVALUATION UNDER USER-DP

Let us now move towards the user-DP. As mentioned, user-DP exclusively focuses on join queries, and self-joins can bring unique challenges. In this section, we first review the works for self-join-free queries. Subsequently, we delve into the techniques employed to handle self-joins. Next, we talk about how to answer SJA-Max queries and then SPJA queries.

4.1 Self-join-free SJA Queries

As mentioned, the challenge that arises with user-DP compared to tuple-DP is that each individual can own arbitrarily many tuples. Consider the self-join-free SJA query introduced in Section 1.2, where we count items while protecting the privacy of customers. In this context, a customer could theoretically possess an unbounded number of items and adding such a customer to the database can cause an unbounded change in the query result. A simple fix is to assume a finite GS_Q, which can be justified in practice because we may never have a customer with, say, more than a million items. However, assuming such a GS_Q limits the allowable database instances, one tends to be conservative and sets a large GS_Q. This allows the Laplace mechanism to work, but adding noise of this scale clearly eliminates any utility of the released query answer. Furthermore, it is clear that this issue persists across all instances, leading to $\mathrm{LS}_Q(\mathbf{I}) \equiv \mathrm{GS}_Q = \infty$. This means the sensitivity-based techniques used for answering SJA queries under tuple-DP will lose utility since those sensitivity measures are all upper bounds of local sensitivity.

The issue above was first identified by Kotsogiannis et al. [34], who also formalized the user-DP. Their solution is the *truncation mechanism*, which simply deletes all customers with more than τ items before applying the Laplace mechanism, for some threshold τ. After truncation, the query has sensitivity τ, so adding noise of scale τ is sufficient. Such an idea has also been used in a later work [45]. A well-known issue for the truncation mechanism is the bias-variance trade-off: In one extreme $\tau = \mathrm{GS}_Q$, it degenerates into the naive Laplace mechanism with a large noise (i.e., large variance); in the other extreme $\tau = 0$, the truncation introduces a bias as large as the query answer. Both [34] and [45] use a heuristic approach to find such a τ, without offering any optimal guarantee.

As mentioned, self-join-free SJA queries under user-DP are equivalent to the sum estimation problem, and the issue of how to choose a near-optimal τ has been extensively studied in the statistics and machine learning community [1, 5, 6, 26, 21]. In this context, $\mathbf{I}$ is treated as an ordered multiset of integers $x_1, x_2, \ldots, x_N$, where each x_i corresponds to an individual user's contribution to $Q(\mathbf{I})$, and $Q(\mathbf{I}) = \sum_i x_i$. The truncation mechanism is to delete those $x_i > \tau$.[6] Furthermore, it is clear that x_N is the maximum user contribution, i.e., $\mathrm{DS}_Q(\mathbf{I})$. An observation is that by setting $\tau = x_N$, we can eliminate bias and introduce noise at a scale of $O(\mathrm{DS}_Q(\mathbf{I}))$, thereby achieving $(1, O(1))$-down neighborhood optimal error. Then, the problem is reduced to estimate x_N. The discussion of works on estimating x_N under DP can be found in Section 3.1, where the state-of-the-art algorithm yields a rank error of $O(\log(x_N))$. Using such

[6]Some works use clipping instead of truncation, i.e., clipping x_i to τ if $x_i > \tau$. Since will not affect the result asymptotically, we will use the truncation mechanism in our discussion.

a noisy estimated x_N as the truncation threshold leads to an error of $O\Big(\mathrm{DS}_Q(\mathbf{I}) \cdot \log\big(\mathrm{DS}_Q(\mathbf{I})\big)\Big)$.

Can we further enhance this result? Recall that the $O(\log(x_N))$ rank error has already been proven to be optimal, so it seems that this outcome can't be improved if we employ x_N as the truncation threshold. However, for the maximum problem, our focus is solely on achieving rank error, with the aim of avoiding relative error. Astute readers would recognize that when x_N serves as the truncation threshold, we essentially need only a constant approximation of x_N. In other words, we can tolerate some relative error in locating such x_N.

Dong and Yi [21] leveraged this finding to devise a mechanism for identifying a τ such that $x_{N-k} \le \tau \le 2x_N$. By permitting relaxation on the upper boundary, they manage to reduce the rank error from $O(\log(x_N))$ to $O(\log\log(x_N))$. Implementing such a τ results in an error of $O\Big(\mathrm{DS}_Q(\mathbf{I}) \cdot \log\log\big(\mathrm{DS}_Q(\mathbf{I})\big)\Big)$ in the sum estimation, which is $\Big(1, O\big(\log\log(\mathrm{DS}_Q(\mathbf{I}))\big)\Big)$-down neighborhood optimal. Furthermore, [21] points out the optimality ratio $O\big(\log\log(\mathrm{DS}_Q(\mathbf{I}))\big)$ cannot be improved.

4.2 SJA Queries with Self-joins

When addressing SJA queries under user-DP, the presence of self-joins brings another challenge. Specifically, all the aforementioned techniques for selecting a truncation threshold τ heavily depend on the assumption of individual independence, i.e., the addition or removal of one individual doesn't impact the data of another individual. However, this assumption no longer holds when the query involves self-joins. In fact, the truncation mechanism itself falls as illustrated in the following example.

Using the query provided at the beginning of the article as an example, let us assume that both `Customer` and `Supplier` are primary private relations. This is like an edge counting query in a bipartite graph, where the nodes on the left side represent suppliers, the nodes on the right side represent customers, and the edges represent items. With a given truncation threshold τ, we intend to eliminate all nodes with degrees higher than τ. To illustrate a failure of the truncation mechanism, let us construct an instance $\mathbf{I}$ as follows: each customer only purchases a single item, while each supplier provides τ items. In other words, each left-side node has a degree of τ, while each right-side node has a degree of 1. In total, there are N items. Now, consider a neighboring instance $\mathbf{I}'$, which we create by inserting a right-side node that is connected to every existing left-side node. In $\mathbf{I}'$, every left-side node has a degree of $\tau + 1$. When we do truncation by τ, the truncated result for $\mathbf{I}$ is N, whereas for $\mathbf{I}'$, it is 0 due to the truncation of all left-side nodes. Adding noise at a scale of τ cannot obscure their difference, consequently violating the DP.

The truncation mechanism fails because, after truncation, the query's sensitivity is no longer bounded by τ. More fundamentally, this is due to the correlation among the individuals introduced by self-joins. In the example above, we see that the addition of one node may cause the degrees of many others to increase. For the problem of graph pattern counting under node-DP, which can be formulated as a self-join SJA query under user-DP as previously mentioned, Kasiviswanathan et al. [31] proposed a linear program (LP)-based truncation mechanism to fix the issue. Dong et al. [16] then extended this solution to support general SJA queries.

Now, the remaining task is how to determine the appropriate τ when dealing with self-joins. On one hand, [31] does not study the selection of τ for graph pattern counting queries, rendering their mechanism devoid of any utility guarantee. On the other hand, adopting a similar approach to selecting τ as self-join-free queries does not work. This is also because a single individual can influence the contributions of numerous others. In the above example, all suppliers contribute τ in $\mathbf{I}$, while in $\mathbf{I}'$ each supplier contributes $\tau + 1$. Running a DP algorithm to estimate the maximum contribution is likely to yield τ and $\tau + 1$ for $\mathbf{I}$ and $\mathbf{I}'$ respectively, making them distinguishable.

To address this challenge, Dong et al. proposed *Race-to-the-Top* (R2T) [16] for adaptively choosing τ in combination with any valid DP truncation mechanism that satisfies specific properties: (1) $Q(\mathbf{I}, \tau) \le Q(\mathbf{I})$; (2) the sensitivity of $Q(\mathbf{I}, \tau)$ is bounded by τ; (3) $Q(\mathbf{I}, \tau) = Q(\mathbf{I})$ for $\tau \ge \mathrm{DS}_Q(\mathbf{I})$. Intuitively, such a $Q(\mathbf{I}, \tau)$ gives a stable (property (1)) underestimate (property (2)) of $Q(\mathbf{I})$, while reaches $Q(\mathbf{I})$ for τ sufficiently large (property (3)). Notably, $Q(\mathbf{I}, \tau)$ itself is not DP. Instead of directly determining τ, R2T directly provides a privatized query answer. The central idea is to try out $Q(\mathbf{I}, \tau)$ with $\tau = 2, 4, 8, \ldots, \mathrm{GS}_Q$, and somehow pick the "winner" of the race (the maximum) to estimate $Q(\mathbf{I})$. To ensure DP in this process, noise of $\mathrm{Lap}(\tau/\varepsilon')$ is added to each $Q(\mathbf{I}, \tau)$, requiring the privacy budget to be divided as $\varepsilon' = \varepsilon/\log(\mathrm{GS}_Q)$ since multiple $Q(\mathbf{I}, \tau)$ are attempted. Yet, this noise-masked $Q(\mathbf{I}, \tau)$ can turn out to be extremely uncertain and potentially much greater than $Q(\mathbf{I})$, particularly with a larger value of τ. To get out of this problem, we shift $Q(\mathbf{I}, \tau)$ down by an amount that roughly equals the scale of the noise, i.e., $\tilde{O}(\tau)$. This step ensures that the noise-masked $Q(\mathbf{I}, \tau)$ is generally underestimated compared to the true result $Q(\mathbf{I})$ thus we can pick the "winner" of the race. With this idea alongside the LP-based truncation mechanism, R2T achieves an error of $O\big(\mathrm{DS}_Q(\mathbf{I}) \cdot \log(\mathrm{GS}_Q(\mathbf{I})) \cdot \log\log(\mathrm{GS}_Q(\mathbf{I}))\big)$, which is indeed $\Big(1, O\big(\log(\mathrm{GS}_Q(\mathbf{I})) \cdot \log\log(\mathrm{GS}_Q(\mathbf{I}))\big)\Big)$-down neighborhood optimal. Additionally, for efficiency, the computation bottleneck of R2T is the $\log(\mathrm{GS}_Q)$ LPs, each contains $|J(\mathbf{I})|$ variables and $|J(\mathbf{I})| + N$ constraints, with $J(\mathbf{I})$ denotes join results, which is equivalent to $|Q(\mathbf{I})|$ for counting queries. Dong et al. also provide techniques to speed up this process and build a system employing PostgreSQL and the CPLEX LP solver.

Following R2T, Fang et al. [25] introduced another LP-based mechanism (to be discussed in detail in the upcoming section). This mechanism is also applicable

to answering SJA queries. Rather than an assumption of a fixed GS_Q, their algorithm necessitates a predefined output range $[1, R]$ and achieves an error of $O(\mathrm{DS}_Q(\mathbf{I}) \cdot \log(R))$. Notably, GS_Q must inherently be smaller than R, but the error of R2T possesses an additional loglog factor leading to these two upper bounds do not dominate each other. Additionally, their algorithm requires solving $\Theta(\log(R))$ LPs, each of which has a more complex structure than those used in R2T. The experiments show that both mechanisms yield comparable error levels, with R2T having significantly lower running times.

4.3 SJA Queries with MAX Aggregation

Now, let us discuss SJA queries with MAX aggregation. The objective of these queries is to identify the highest value among the aggregated attributes over the join results. This variation of the query introduces new challenges to achieving optimal utility and current techniques cannot effectively address them. These difficulties even exist in self-join-free queries. In this problem, one intuitive approach is to apply the truncation mechanism to establish an upper bound on the number of join results associated with each individual. Consequently, with the group privacy property of DP, we can transform this problem into a SA query with MAX aggregation under tuple-DP. During the truncation phase, the optimal choice for τ is to use $\kappa(\mathbf{I})$, which is the maximum number of join results corresponding to a single user in $\mathbf{I}$. By allocating the privacy budget accordingly, the ultimate result has a rank error of $\tilde{O}(\kappa(\mathbf{I}))$.

However, despite that $\kappa(\mathbf{I})$ is the maximum number of join results corresponding to one user, a rank error of $\tilde{O}(\kappa(\mathbf{I}))$ doesn't necessarily imply $(\tilde{O}(1), c)$-down neighborhood optimality for arbitrary c. To see this, consider the query $R_1(\underline{A}) \bowtie R_2(A, B)$ that outputs the maximum value on attribute B, with relation R_1 designated as the primary private relation. The instance $\mathbf{I}$ contains two distinct user types. For the first $N/2$ users, each corresponds to a tuple (a_i) in R_1, and one tuple (a_i, N) in R_2. Meanwhile, for the remaining $N/2$ users, each corresponds to one tuple (a_i) in R_1 as well but $N/2$ tuples in R_2: $(a_i, 0), (a_i, 1), \ldots, (a_i, N-1)$. It is trivial to see that $Q(\mathbf{I}) = N$, and after removing any arbitrary k users where $k < N/2$, the query result will still remain unchanged at N. This implies that $\mathcal{L}_{\text{d-nbr}}(\mathbf{I}, k) = 0$ for $k < \frac{N}{2}$, as we can construct a mechanism M' where $M'(\cdot) \equiv N$. Nonetheless, it is clear that $\kappa(\mathbf{I}) = N$. Since there are only N join results with a value of N, a rank error of $\tilde{O}(\kappa(\mathbf{I}))$ means returning a value lower than N, which fails to achieve (d, c)-down neighborhood optimal error for any $d < \frac{N}{2}$ and any $c > 0$. Another approach is to transform the maximum problem into a counting problem, similar to what was discussed in Section 3.1. However, the $\tilde{O}(\mathrm{DS}_Q(\mathbf{I}))$ additive error of the counting problem would also lead to a $\tilde{O}(\kappa(\mathbf{I}))$ rank error. Acute readers would realize that the problem arises due to two main factors. First, the data distribution is skewed, implying that not all users in $\mathbf{I}$ correspond to $\kappa(\mathbf{I})$ join results. Second, we cannot guarantee that those high-value join results originate from the same individuals.

To address this issue, Fang et al. [25] have devised a general DP mechanism applicable to any monotonic query under the user-DP model called *ShiftedInverse*. Their algorithm requires a predefined output range $[1, R]$ and achieves a $\Big(O\big(\log(R)\big), O\big(\log(R)\big)\Big)$-down neighborhood optimal error. The high-level idea is, for each value $r \in [1, R]$, they determine $\mathrm{len}(\mathbf{I}, r)$, which is how many individuals should be excluded to achieve a query result less than r. Subsequently, they sample each r as an output with a probability proportional to $\mathrm{len}(\mathbf{I}, r)$. This process can be shown to satisfy the ε-DP under the user-DP while the challenge of guaranteeing down-neighborhood optimal error is to ensure that the output is underestimated. This can be achieved by "shifting" the target downward. More precisely, they use $\widehat{\mathrm{len}}(\mathbf{I}, r) = |\mathrm{len}(\mathbf{I}, r) - \Theta(\log(R))|$ in place of $\mathrm{len}(\mathbf{I}, r)$ during sampling. As a result, they will sample an r such that $\underline{\mathrm{len}(\mathbf{I}, r) = O(\log(R))$ with high probability, implying a $(O(\log(R)), O(\log(R)))$-down neighborhood optimal error$}$. Note that this underlined statement holds for arbitrary monotonic queries, while for specific queries, more favorable outcomes might be attainable. For instance, for maximum queries, this approach can lead to a $(O(\log(R)), 2)$-down neighborhood optimal error.

For self-join-free SJA queries with MAX aggregation, all $\mathrm{len}(\mathbf{I}, r)$ values for $r \in [1, R]$ can be computed in linear time. However, for other functions like self-join SJA queries with MAX and SUM aggregation, computing $\mathrm{len}(\mathbf{I}, r)$ requires a time complexity of $O(N^{\mathrm{len}(\mathbf{I}, r)})$. Even though, Fang et al. emphasize it is unnecessary to compute all $\mathrm{len}(\mathbf{I}, r)$ values. Nonetheless, implementing ShiftedInverse still requires a running time of $N^{O(\log(R))}$ in general, which is super-polynomial unless R is very small. To address this challenge, they propose an approximation of $\mathrm{len}(\mathbf{I}, \mathrm{r})$ specifically for certain functions. This modified mechanism, equipped with the approximate $\mathrm{len}(\mathbf{I}, \mathrm{r})$, is termed *approximate ShiftedInverse*. While approximate ShiftedInverse maintains ε-DP, the utility guarantee might not be upheld. When addressing self-join SJA queries with MAX aggregation, each approximate $\mathrm{len}(\mathbf{I}, r)$ is formulated as an LP, and the approximate ShiftedInverse mechanism achieves a $(O(\log(R)), 2)$-down neighborhood optimal error. By the way, an approximation for SJA queries with SUM aggregation is also proposed to achieve the result mentioned in the last section.

4.4 SPJA Queries

For SPJA queries, first, since ShiftedInverse can handle arbitrary monotonic queries under user-DP, we can use it to answer SPJA queries and achieve $O\big(\log(R), \log(R)\big)$-down neighborhood error. However, in this case, ShiftedInverse cannot be computed in polynomial time. Additionally, Fang et al [25] also proposed an approximate ShiftedInverse mechanism for this problem, which can be computed using a logarithmic number of LPs.

However, this mechanism does not hold an optimal guarantee in utility. In parallel, Dong et al [16] proposed an LP-based truncation mechanism specifically designed for SPJA queries. This mechanism can also be integrated with R2T. However, their algorithm also lacks an optimal utility guarantee as well. Moreover, they presented a negative result indicating that achieving an error dependent on $\mathrm{DS}_Q(\mathbf{I})$ for answering SPJA queries under user-DP is unattainable. This essentially means that achieving a $(1, c)$-down neighborhood optimal error for any value of c is not achievable.

5. ANSWERING QUERIES UNDER DP IN MORE COMPLEX SETTING

All the aforementioned discussions consider the straightforward scenario of answering a single SQL query. Subsequently, more complex settings have been studied in the literature.

5.1 Multi-query Answering

In practice, queries often arrive in batches, thus it is natural to consider the multi-query problem in relational databases, which includes group-by queries as an important special case (i.e., each group corresponds to one query). For instance, if a data analyst is interested in the total number of items shipped for each date of the first month this year, s/he would issue the following query,

```
SELECT Lineitem.Shipdate, count(*)
FROM Customer, Orders, Lineitem
WHERE Lineitem.Shipdate > 2023-01-01
  AND Lineitem.Shipdate < 2023-04-30
  AND Customer.CK = Orders.CK
  AND Orders.OK = Lineitem.OK
GROUP BY Lineitem.Shipdate;
```

This query is equivalent to answering $d = 100$ SJA queries, with each query corresponding to a specific date. Let $\mathbf{Q} = (Q_1, \ldots, Q_d)$ represent the set of d queries that we aim to answer privately. We use the standard metric of root-mean-square error (RMSE), i.e., the ℓ_2 error to measure the utility.

The general approach to this multi-query problem is to use the privacy composition theory, i.e., we divide the privacy budget to the d queries and answer each query with the single-query mechanism. Using advanced composition, the utility suffers an $\tilde{O}(\sqrt{d})$-factor degradation which is the best we can if to answer SA queries with Count aggregation under tuple-DP. For more complex queries like SJA queries under user-DP, this method leads to an error of $\tilde{O}\big(\sqrt{d} \cdot \mathrm{DS}_{Q_k}(\mathbf{I})\big)$ for Q_k hence an RMSE of

$$\tilde{O}\left(\sqrt{d} \cdot \sqrt{\sum_{k=1}^{d} \mathrm{DS}_{Q_k}(\mathbf{I})^2}\right) \leq \tilde{O}\Big(d \cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})\Big).$$

However, this error is not optimal. An observation is that answering d self-join-free SJA queries under user-DP is equivalent to the sum estimation problem in d dimensions, where each user's contribution to these d queries can be seen as a vector, and the task is to compute their summation. This equivalence has an immediate consequence: the lower bound established for the sum estimation problem also leads to a lower bound for the multi-query problem, which is $\tilde{\Omega}\left(\sqrt{d} \cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})\right)$ [26, 29]. For self-join-free queries, [26] extends their algorithm designed for single self-join-free SJA query under user-DP to the multiple-query scenario. In their approach, they first estimate the maximum user contribution and employ that as the threshold for truncating heavy contributors. Subsequently, they use the Gaussian mechanism to add noise. Ultimately, they achieve an error of $\tilde{O}(\sqrt{d} \cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}))$. However, their mechanism requires a predefined $\mathrm{GS}_{\mathbf{Q}}$. On the other hand, Dong et al. [18] extended the 1-dimensional mechanism from [21] to d dimensions. This extension enables them to also achieve the optimal error of $\tilde{O}(\sqrt{d} \cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}))$ without the need for a predefined $\mathrm{GS}_{\mathbf{Q}}$.

For self-joins, akin to the single query case, the truncation mechanism encounters problems. Additionally, the LP-based mechanisms [16, 25] fundamentally do not work for multiple queries, as LP optimization is limited to one-dimensional queries. Dong et al. [18] also highlight that the straightforward extension of these LP-based approaches also does not work. Thereafter, they proposed an alternative approach to the multi-query problem. The initial version of their algorithm has an exponential running time, but they subsequently reduce it to polynomial time using quadratically constrained quadratic programming (QCQP), which can be computed in polynomial time. This novel approach enables them to achieve an error of $\tilde{O}(\sqrt{d} \cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}))$, matching the lower bound up to polylogarithmic factors.

Furthermore, Cai et al. [12] addressed the multi-query problem using an alternative approach. They generated a synthetic relational database under DP and used that to answer the subsequent queries. This methodology provides the advantage of accommodating a wide array of query types while maintaining error independence from the query dimensions. However, their proposed algorithm lacks any utility guarantee and only performs well when the domain of attributes is small.

5.2 Continual Observation

Data is seldom static. When private data evolves over time, there is a need to continually release sanitized query results about the data while preserving the privacy of the users who contribute to the data. This is precisely the problem of *continual observation under differential privacy*, introduced in the pioneering work of Dwork et al. [22]. Here, time is divided into discrete steps, and data is modeled as a (possibly infinite) stream of tuples arriving over time, one per time step and we release the query result after each time step. In the dynamic setting, Dwork et al. [22] proposed two natural DP definitions: In *event-DP*, two streams are neighbors if one can be obtained from the other by removing one item. In *user-DP*, each tuple is associ-

Query type		Current Result	
		Event-DP	User-DP
Count/Sum	SA	✓ [22], [13]	
	Self-join-free SJA	Open question	✓ [17]
	Self-join SJA		
	SPJA		
Max	SA	✓ [22], [13]	
	SJA	Open question	× [17]

Table 2: Whether the same asymptotic error as that in the static setting can be achieved with a continual observation setting.

ated with a user, and two streams are neighbors if one can be obtained from the other by removing all or any subset of items associated with one user. Clearly, the event-DP/user-DP in the dynamic setting aligns with the tuple-DP/user-DP in the static setting. Furthermore, under event-DP/user-DP, the problem at each given moment can be viewed as a static problem under tuple-DP/user-DP. Our target is to achieve the same error as the static setting each time. Up to now, this objective has been successfully realized for certain specific SQL queries.

The major result in [22] on event-DP is a black-box reduction to the static problem with only a $\mathrm{poly}\log(T)$-factor increase in the error for any *union-preserving* query Q, where T is an upper bound on the stream length. Chan et al. [13] extend this result to infinite streams, with the $\mathrm{poly}\log(T)$ factor replaced by $\mathrm{poly}\log(t)$, where t is the current length of the stream. A union-preserving query Q is one such that $Q(\mathbf{I} \cup \mathbf{I}') = Q(\mathbf{I}) + Q(\mathbf{I}')$ for any $\mathbf{I}, \mathbf{I}'$. Most natural functions (e.g., count, sum, max) are union-preserving. The high-level idea is to build a binary decomposition over all the T time steps $\log T$ levels of intervals and the DP mechanism for the static problem is invoked on each interval to return a noisy query result. Then the query result at any time can be obtained by at most $\log T$ such noisy results, one from each level. To set the privacy budgets of these intervals, it suffices to allocate $\varepsilon/\log T$ to each interval by basic composition (across levels) and parallel composition (within a level).

By applying state-of-the-art algorithms for static count, sum, and max queries within this framework, for any time t, we achieve an error of $O(\log^{1.5} t)$ error for count queries, an error of $O(x^t_{\max} \cdot \log\log(x^t_{\max}) \cdot \log^2 t)$ for sum queries, and a rank error of $O(\log(x^t_{\max}) \cdot \log^2 t)$ for max queries, where $x^t_{\max}$ is the maximum value of the instance at time t. These errors correspond to $\tilde{O}(1)$-worst-case optimal error, $\big(1, \tilde{O}(1)\big)$-down neighborhood optimal error, and $\big(\tilde{O}(1), 2\big)$-down neighborhood error for SA queries with Count, Sum, and Max aggregation, respectively. It is worth mentioning that, in the dynamic setting, Sum queries under event-DP differ from those under user-DP, thus needing separate consideration for event-DP.

When moving towards user-DP, one natural idea is to truncate the user contributions. More precisely, given some truncation threshold τ, we only retain the first τ items from each user. Then, we can use group privacy to divide the privacy budget and call the mechanism for event-DP. However, this approach encounters two key issues: Estimating a good τ requires a strong prior knowledge, which is impossible for infinite time domain cases; This leads to a non-time-specific error, i.e., errors across all time steps share the same dependency on τ. To address this issue, Dong at el. [17] used a dynamic τ to constrain user contributions. More precisely, they monitor the count of users with contributions surpassing τ and subsequently double τ when a sufficient number of users meet this criterion. Following each doubling iteration, the entire data stream is truncated using the updated τ and the DP mechanism over the truncated stream is also re-initialized. As a result, for sum queries, they match the error as the static setting up to polylogarithmic factors. For max queries, unfortunately, they show a negative result that no ε-DP mechanisms can achieve $(O(\sqrt{T}), c(T))$-down neighborhood optimality at each time, where T is the length of the stream and $c(T)$ is an arbitrary function of T. It is clear that under user-DP, self-join-free SJA queries with Sum and Max can be treated as a sum and max query since each join result only belongs to one user. We have corresponding positive and negative results for these two queries. For self-join queries, so far no known algorithm can handle them.

6. CONCLUSION

In this article, we have surveyed some recent results on query evaluation under differential privacy. There are two predominant DP policies in the relational model, namely tuple-DP and user-DP, and two instance-specific optimality notions, neighborhood optimality and down-neighborhood optimality. The choice of an appropriate optimality notion depends on the nature of the query under consideration and the DP policy adopted.

We conclude this article by mentioning two interesting directions for further investigation.

More Complex Scenarios.

As mentioned, the problems in the single-query setting have been reasonably well solved. However, there are many open questions in more complex scenarios, especially for multiple queries and under continuous observation. For multi-query answering, one open question is how to effectively handle queries involving max-

imum aggregation and projection operations. For continual observation, current studies are limited to queries involving single relations or their equivalent constructs, and how to handle join operations still remains an uncharted domain waiting for more exploration.

Integrating DP with Artificial Intelligence in a Relational Model.

Training machine learning models within the framework of a relational database has attracted lots of attention from the database community [44, 35, 3, 38, 32]. In this context, the training procedure operates over outcomes from join queries, with substantial efforts devoted to enhancing operational efficiency and curtailing storage overhead by avoiding explicit materialization of join results. Concurrently, how to integrate DP with the training of machine learning models over a flatted table has also been extensively studied. However, it remains an open question how to integrate DP into the training of machine learning models within a relational framework.

7. REFERENCES

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.

[3] M. Abo Khamis, H. Q. Ngo, X. Nguyen, D. Olteanu, and M. Schleich. In-database learning with sparse tensors. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 325–340, 2018.

[4] M. Abo Khamis, H. Q. Ngo, and A. Rudra. Faq: questions asked frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 13–28, 2016.

[5] K. Amin, A. Kulesza, A. Munoz, and S. Vassilvtiskii. Bounding user contributions: A bias-variance trade-off in differential privacy. In *International Conference on Machine Learning*, pages 263–271. PMLR, 2019.

[6] G. Andrew, O. Thakkar, H. B. McMahan, and S. Ramaswamy. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

[7] M. Arapinis, D. Figueira, and M. Gaboardi. Sensitivity of counting queries. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2016.

[8] H. Asi and J. C. Duchi. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. *Advances in neural information processing systems*, 33, 2020.

[9] N. Bakibayev, T. Kociskỳ, D. Olteanu, and J. Závodnỳ. Aggregation and ordering in factorised databases. *Proceedings of the VLDB Endowment*, 6(14), 2013.

[10] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96, 2013.

[11] M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[12] K. Cai, X. Xiao, and G. Cormode. Privlava: synthesizing relational data with foreign keys under differential privacy. *Proceedings of the ACM on Management of Data*, 1(2):1–25, 2023.

[13] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Transactions on Information and System Security*, 2011.

[14] S. Chen and S. Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 653–664, 2013.

[15] T. Dick, C. Dwork, M. Kearns, T. Liu, A. Roth, G. Vietri, and Z. S. Wu. Confidence-ranked reconstruction of census microdata from published statistics. *Proceedings of the National Academy of Sciences*, 120(8):e2218605120, 2023.

[16] W. Dong, J. Fang, K. Yi, Y. Tao, and A. Machanavajjhala. R2T: Instance-optimal truncation for differentially privatequery evaluation with foreign keys. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2022.

[17] W. Dong, Q. Luo, and K. Yi. Continual observation under user-level differential privacy. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2190–2207. IEEE Computer Society, 2023.

[18] W. Dong, D. Sun, and K. Yi. Better than composition: How to answer multiple relational queries under differential privacy. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2023.

[19] W. Dong and K. Yi. Residual sensitivity for deferentially private multi-way joins. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2021.

[20] W. Dong and K. Yi. A nearly instance-optimal differentially private mechanism for conjunctive queries. In *Proc. ACM Symposium on Principles of Database Systems*, 2022.

[21] W. Dong and K. Yi. Universal private estimators. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 195–206, 2023.

[22] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724, 2010.

[23] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390, 2009.

[24] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[25] J. Fang, W. Dong, and K. Yi. Shifted inverse: A general mechanism for monotonic functions under user differential privacy. 2022.

[26] Z. Huang, Y. Liang, and K. Yi. Instance-optimal mean estimation under differential privacy. *Advances in Neural Information Processing Systems*, 2021.

[27] M. R. Joglekar, R. Puttagunta, and C. Ré. Ajar: Aggregations and joins over annotated relations. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 91–106, 2016.

[28] N. Johnson, J. P. Near, and D. Song. Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539, 2018.

[29] G. Kamath, J. Li, V. Singhal, and J. Ullman. Privately learning high-dimensional distributions. In *Proceedings of the 32nd Annual Conference on Learning Theory*, COLT '19, pages 1853–1902, 2019.

[30] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, 2011.

[31] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*, pages 457–476. Springer, 2013.

[32] M. A. Khamis, H. Q. Ngo, X. Nguyen, D. Olteanu, and M. Schleich. Learning models over relational data using sparse tensors and functional dependencies. *ACM Transactions on Database Systems (TODS)*, 45(2):1–66, 2020.

[33] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204, 2011.

[34] I. Kotsogiannis, Y. Tao, X. He, M. Fanaeepour, A. Machanavajjhala, M. Hay, and G. Miklau. Privatesql: a differentially private sql query engine. *Proceedings of the VLDB Endowment*, 12(11):1371–1384, 2019.

[35] A. Kumar, M. Boehm, and J. Yang. Data management in machine learning: Challenges, techniques, and systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1717–1722, 2017.

[36] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30, 2009.

[37] A. Narayan and A. Haeberlen. Djoin: Differentially private join queries over distributed databases. In *USENIX Symposium on Operating Systems Design and Implementation*, pages 149–162, 2012.

[38] M. Nikolic, H. Zhang, A. Kara, and D. Olteanu. F-ivm: learning over fast-evolving relational data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2773–2776, 2020.

[39] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84, 2007.

[40] D. Olteanu and J. Závodnỳ. Size bounds for factorised representations of query results. *ACM Transactions on Database Systems (TODS)*, 40(1):1–44, 2015.

[41] C. Palamidessi and M. Stronati. Differential privacy for relational algebra: Improving the sensitivity bounds via constraint systems. In *QAPL*, 2012.

[42] D. Proserpio, S. Goldberg, and F. McSherry. Calibrating data to sensitivity in private data analysis. *Proceedings of the VLDB Endowment*, 7(8), 2014.

[43] P. Regulation. General data protection regulation. *Intouch*, 25:1–5, 2018.

[44] M. Schleich, D. Olteanu, and R. Ciucanu. Learning linear regression models over factorized joins. In *Proceedings of the 2016 International Conference on Management of Data*, pages 3–18, 2016.

[45] Y. Tao, X. He, A. Machanavajjhala, and S. Roy. Computing local sensitivities of counting queries with joins. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 479–494, 2020.

[46] S. Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.

[47] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 731–745, 2015.

Where do Databases and Digital Forensics meet? A Comprehensive Survey and Taxonomy

Danilo B. Seufitelli[1], Michele A. Brandão[1,2], Ayane C. A. Fernandes[2],
Kayque M. Siqueira[2], Mirella M. Moro[1]
[1]Universidade Federal de Minas Gerais
[2]Instituto Federal de Minas Gerais
daniloboechat@dcc.ufmg.br,michele.brandao@ifmg.edu.br
{ayanecristinamb,kayque.siq}@gmail.com,mirella@dcc.ufmg.br

ABSTRACT

We present a systematic literature review and propose a taxonomy for research at the intersection of Digital Forensics and Databases. The merge between these two areas has become more prolific due to the growing volume of data and mobile apps on the Web, and the consequent rise in cyber attacks. Our review has identified 91 relevant papers. The taxonomy categorizes such papers into: Cyber-Attacks (subclasses SQLi, Attack Detection, Data Recovery) and Criminal Intelligence (subclasses Forensic Investigation, Research Products, Crime Resolution). Overall, we contribute to better understanding the intersection between digital forensics and databases, and open opportunities for future research and development with potential for significant social, economic, and technical-scientific contributions.

1. INTRODUCTION

Digital Forensics (DF) helps reconstruct cybercrimes and develop prevention mechanisms. Indeed, it searches, analyzes, identifies, and categorizes data that may become crime evidence [28, 30, 59]. For instance, some authors have used DF in preventing and detecting *SQL Injection* attacks [61], analyzing digital evidence manipulation [49], and others. Also, its number of publications has risen, reaching thousands of works indexed by Google Scholar when using search *strings* (see §3), which also challenges a person to start studying the field. Hence, we apply a Systematic Literature Review (SLR) protocol to extract Computing-related insights from DF, focusing on Database (DB) aware publications. Overall, the SLR covers works mainly from Digital Forensics that also use, explore, or advance DB topics.

The field of DF is widely recognized for its applications beyond the realm of Computing, including legal medicine plus civil, criminal, and corporation investigations. An extensive search and filtering process is essential to identify relevant publications specific to the computational context. However, an SLR can only solve half the problem by identifying such publications. The other half is organizing such articles comprehensively, promoting effective categorization and synthesis, and supporting future research on DF and DB.

For the second challenge (comprehensive organization), we can use a taxonomy to identify and classify approaches and concepts based on their purposes. For instance, Dave et al. [22] introduce a taxonomy for implicit requirements identification, a crucial part of requirements engineering. Through taxonomies, researchers can efficiently categorize their findings and develop a better understanding of the research landscape, ultimately advancing the field and supporting future research endeavors.

Then, our goal starts from the high availability of data on DF and tackles its *intersection* with DB. An initial study connects them on data infrastructure and availability [53]. The goal now is to cover works on how to identify and prevent *cyber attacks* as well as exploit data for *criminal intelligence*. This article aims to achieve two primary objectives: (*i*) identify how exploring data can aid in DF and identify the challenges faced by data engineers and scientists in doing so, and (*ii*) develop a better categorization of works in this area. As we note "data is the new oil" is not enough,[1] promoting a better understanding between databases and digital forensics is crucial as data plays a vital role in digital forensics. Hence, the contributions of this article include a survey and a corresponding taxonomy of publications at their intersection, as well as discussions on the various phases of forensics tasks related to data and the associated opportunities. Overall, researchers may better grasp how exploring data can aid to DF.

The rest of this article is structured as follows. We discuss the related work in §2. Then, we outline the SLR methodology in §3 and review concepts on the taxonomy classes in §4. Next, we focus the SLR on data from/to criminal intelligence and cyber attacks in §5. Finally, we map the taxonomy and its papers to digital forensic phases in §6, and discuss challenges for the survey questions in §7.

[1]Data as The New Oil: `https://bit.ly/f-dataoil`

Table 1: Search strings over digital libraries.

Search Strings
1."database forensic" OR "database forensics" OR "forensic database" OR "forensic databases"
2."criminal database" OR "criminal databases" OR "database auditing"
3.(database OR databases) AND ("forensic access" OR "forensic analysis" OR "forensic purpose" OR "forensic purposes")
4.(forensic OR forensics) AND ("database analysis" OR "database access")
5.(forensic OR forensics) AND ("database analysis" OR "database access")

2. RELATED WORK

There is no related work on the intersection of DB and DF besides ours [53], which goes over DBMS (Database Management System) and data building, solely. Others cover different aspects of digital forensics. First, Khan et al. [34] review its evolution between 1980 and 2020. Then, Sikos [55] focuses on the evolution of forensic packet analysis. Finally, Al-Dhaqm and collaborators cover Database Forensic Investigation (DBFI) from two aspects: (*i*) categorization of processes (planning, preparation and pre-response; acquisition and preservation; and analysis and reconstruction) in [3]; and (*ii*) common limitations (regarding investigation processes; concepts and terminologies; and lack of unified models) and their solutions in [4].

Our contributions are novel for focusing on cyber attacks and criminal intelligence linking databases and forensics (whereas [53] goes over DBMS and data building). They shall support youngsters and experts in forensics in quickly finding works as classified in a taxonomy (four classes and three subcategories) according to the forensics phase.

3. METHODOLOGY

In this SLR, we adapt the methodology of seven steps from Kitchenham and Charters' protocol [35] (which is the same applied in [53]).

Step 1: Define research questions. We first define questions to guide over the state of knowledge in Forensics and DB. Our questions and goals are:
– *When and where the studies were published?* Define interests and trends over time.
– *What kinds of research are there?* Classify into qualitative, quantitative or mixed.
– *What is the focus of data-driven digital forensics?* Define the sub-areas, themes and trends.
– *Which are the advances and potential challenges in the area?* Identify methods, models and tools.
– *What issues are still open?* Find new challenges.

Step 2: Define search strings. Such questions require looking for works that could answer them. First, we consider the most extensive Computing digital library – DBLP (Digital Bibliography and Library Project), and search for "data forens" within title, abstract and keywords. Its results serve as input for filtering the most relevant keywords and define the final search strings, see Table 1.

Step 3: Define inclusion criteria and general exclusion criteria. Keeping focus on our questions, we define the following criteria: *Inclusion criteria* verify if the paper is related to both DB and DF; and *General exclusion criteria* check if it has no abstract, is only an abstract, is an old version of another study already considered, is not a primary study, and is not possible to access its full content.

Step 4: Search for publications. Looking for works in one digital library may compromise coverage. Hence, we search for the pre-defined strings over: IEEE Xplore, Scopus, Science Direct, and Web of Science. All returned papers were collected – except from Scopus, as we consider only publications on Computing and Engineering. The numbers of papers found in each library are: IEEE 289; Scopus 3,836; Science Direct 2,042; and Web of Science 756; making a total of 6,923 records.

Step 5: Define specific exclusion criteria. We create exclusion criteria based on the publication titles. We discard those outside the domains of Computing and Engineering and those about other fields – e.g., biology, genetic forensics, biomedicine. By applying such criteria to titles and keywords, we identified 493 relevant publications for further analysis.

Step 6: Select publications and identify common themes. After reviewing abstracts of the 493 publications, we eliminate those outside the inclusion criteria; resulting in 151 articles, to which the exclusion criteria was reapplied. In the end, we have identified 101 articles for the next phase.

Step 7: Classify publications. We elaborate a taxonomy by using the identified themes in the previous step, with four major classes: DBMS, data building, cyber attacks, and criminal intelligence. Three volunteers manually labeled 101 publications (from step 6) by considering these four classes. We also evaluate the agreement of the classification through the Fleiss' Kappa coefficient [20], which achieved a value of 0.30 with 95% of confidence. The volunteers then discussed the content and reached a final verdict to the labels, which resulted in a better classification and the exclusion of 10 publications. Thus, we present the SLR for 91 publications; from which 52 fit the two categories used in this article: Cyber Attacks and Criminal Intelligence.

4. TAXONOMY OVERVIEW

Forensic science is the area interested in and concerned with finding the relationship among people, places, and things involved in criminal activities [46]. It is an essential science to assist in the investigation and judgment of civil and criminal cases. It also considers expertise from different areas; e.g., forensic chemistry, forensic biology, forensic physics, computer forensics (also called cyber forensics and digital forensics), and so on [46]. Here, we focus on *digital forensics*, a science whose goal is to identify,

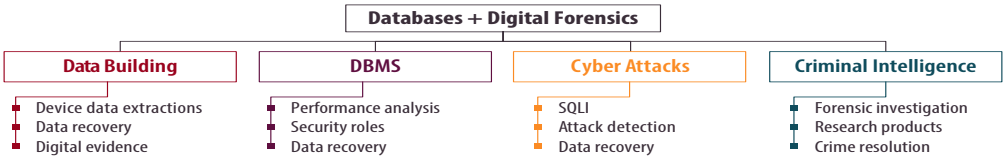


Figure 1: Proposed classifications and their guidelines. In this article, we focus on Cyber Attacks and Criminal Intelligence, as Data Building and DBMS are covered in [53].

preserve, retrieve, analyze and present digital data during a digital-related investigation [29]. Although one may use the terms computer forensics, digital forensics, and cyber forensics interchangeably, they are different: computer forensics is mainly about the investigation of crimes related to computers, whereas cyber and digital forensics are mainly about digital data from various digital devices [34].

Digital forensics uses data as evidence, which is often, but not only, associated with cybercrimes. DF helps to: support or refute assumptions, reconstruct criminal events, and predict unauthorized actions [29]. Studying DF also aids socially, by serving the criminal justice and defense systems [16]. More, investigating the solution to each crime is different. Thus, investigating human misbehavior on computers and digital devices creates new logic and technological challenges, especially when the offender tries to hide the evidence and related activities [16].

We now describe the main attributes of the publications for each class identified in the SLR. We present how we identify the topics of each class (§4.1) and then overview main definitions (§4.2).

4.1 Classification Summary

We propose a taxonomy with four classes: Data Building, DBMS, Cyber Attacks, and Criminal Intelligence, as shown in Figure 1. Note that we have empirically defined these classes based on the extensive literature analysis and by considering the link between topics on databases and digital forensics. Further, Table 2 provides an overview of each class by presenting LDA[2] topics. LDA considers the title and abstract of all papers in each class and returns 15 words to describe the classes. Terms in bold are more representative of each class defined as follows.
– **Data Building**: includes publications that enrich the forensic frameworks, with subcategories Device Data Extraction, Data Recovery, and Digital Evidence. Here, the publications involve the use of data to solve a problem in the digital forensics area.
– **DBMS**: has publications about forensic tools and data architecture, presenting subcategories for Performance Analysis, Security Rules, and Data Recovery. It is an important tool in the DB area; thus, this class includes publications on digital forensics whose main focus is related to DBMS usage.

[2]Latent Dirichlet Allocation – a generative probabilistic model of a collection of texts [13].

Table 2: Topics found by LDA for each class.

Class	LDA Topic
Data Building	forensic **cloud computing** analysis similarity deep incident **traces** learning response **reconstruction** monitoring **storage auditing** artifacts
DBMS	**database** internet analysis things systems **nosql** management iot control machine **recovery** clustering multi application networking
Cyber Attacks	data databases investigation analysis **detection intrusion** big sql dataset log file collection **injection** mining model
Criminal Intelligence	**forensics digital** security network **evidence crime** android applications computer **web cyber intelligence** based encryption center

– **Cyber Attacks**: has publications about digital attacks, with subcategories SQL Injection, Attack Detection, and Data Recovery (further discussed in §5.1). In other words, this class covers publications that address attacks on databases or sensitive data.
– **Criminal Intelligence**: for publications that improve systems and security, with subcategories Forensic Investigation, Research Products, and Crime Resolution (further described in §5.2). The publications in this class include those that apply data to investigate and solve digital crimes.

4.2 Classes and their Categories

This section overviews each class and their categories. The first two classes (DBMS and Data Building) focus on aspects of DB architecture and availability, being well described in [53]. This article highly improves the literature review by describing works related to cyber-attacks and criminal intelligence, whose intersections with DB have not appeared in any survey (to the best of our knowledge).

4.2.1 Data Base Management System (DBMS)

A DBMS allows creating, modifying and deleting a database, plus inserting, deleting and updating its data. Different DBMSs are studied in forensics. For example, Beirami et al [11] prepare DBMSs to optimize the results of forensic queries. We classify forensic-related publications as DBMS when they focus on: *Performance Analysis*, papers aim to improve the performance analysis of DBMS, compare different storage solutions, or propose frameworks for better operating forensic databases; *Security Rules*, papers present a formula that defines the conditions for granting access control, or access controls that specify the access permissions; and *Data*

Recovery, papers promote the search for digital or digitized artifacts stored in a DBMS, introduce optimized storage format for digital evidence, or reconstruct damaged or deleted data.

4.2.2 Data Building

Data building gets valuable content from DB through software, analysis tools, web forms, and so on. Data collection, analysis and layout define the work methodology and help researchers build the desired results. Then, anyone may also benefit from data building by using data science approaches [29]. Works here also study ethical and privacy issues related to using data posted online and complying with General Data Protection Regulation (GDPR) or its variations. We classify a publication as data building when it focuses on properly collecting data, describing, examining, and developing a dataset. Such publications are then categorized as: *Device Data Extraction*, works that analyze data stored in devices to explore their vulnerabilities or security issues; *Data Recovery*, papers focus on recovering deleted files that compose evidence; and *Digital Evidence*, works focus on models, experiments, and data directly related to digital proof.

4.2.3 Cyber Attacks

There are several examples of criminal hacking actions that endanger sensitive data. Cybercriminals try to find weaknesses in systems through different techniques, which become increasingly sophisticated as security systems improve their operation.

In this context, DF becomes an essential tool allied to companies and citizens, as it seeks to identify cybercriminals and their attacks in a well-structured way. Indeed, the main goal is to access sensitive data for identity theft, fraud, extortion, scams, and other criminal and malicious practices.

Categories. Different papers address DF focused on cyberattacks, including domain hijacking [15], network intrusion [38], and dictionary attack [36]. Here, we classify a paper as a cyber attack when the authors address attacks related to databases in three different aspects: *SQL Injection*, papers focus on how the SQLi affects a DB or proposes tools for forensic analysis; *Attack Detection*, papers focus on detecting other kinds of cyber-attacks in forensic DB; and *Data Recovery*, papers lead to recovering data or systems after a cyber attack. Among those, SQLi is the most common one and explained next.

SQL Injection. SQLi is an attack approach based on manipulating SQL code, and is one common concern of DB professionals. A common example is injecting SQL code through the application login screen: "`select * from users where username = 'admin' -- and password = '1234';`". The issue is absence of verification and validation of data entered by a user, as the system concatenates the parameters to the query string. Thereby, when the attacker sends the comment symbol (−−), the DBMS ignores the remain of SQL code, disabling the password field.

4.2.4 Criminal Intelligence

Publications are classified as criminal intelligence when their goal is three-fold: *forensics investigation* to analyze DB contents, investigate DB incidents and build a timeline of illegal activities; *research products* to acquire a timely, valuable and accurate product from the logical processing of forensic case data (research products); and *crime resolution* to solve digital crimes by using forensic and computational techniques. Publications that address any of such branches are covered here.

5. LITERATURE REVIEW

This section discusses the publications filtered by the SLR. We now discuss the publications found in the methodology regarding Cyber Attacks (§5.1), and Criminal Intelligence (§5.2).

5.1 Cyber Attacks

Hackers hold criminal actions that put data about individuals or companies at risk. Here, papers mainly address *SQL Injection*, attack detection, and data recovery. We classify **nine** papers as Cyber Attacks, as they investigate digital crimes that expose, alter, disable, destroy, steal, gain access to or make unauthorized use of a system or device. Table 3 summarizes those publications: where **four** deal with SQLi, **three** perform attack detection methods, and **two** try to recover data from cyber attacks.

SQL Injection. Publications that address attackers who use SQLi to: obtain unauthorized access to a DB or read-protected data, corrupt DB, or grant access to unauthorized users. Pomeroy and Tan [44] highlight the challenges in recovering data after a SQLi attack. They develop a method for detecting SQLi attacks and recovering the excluded data. Likewise, Alam et al [7] explore SQLi vulnerabilities within the web applications of Bangladesh with *.bd* domain. They evaluate and analyze such vulnerabilities through a black-box penetration test. The results point SQLi exposure in over 600 web apps of the 900 considered in the study. The findings suggest experienced administrators do not acceptably maintain the web servers, as user input authentication and regular updates could prevent issues.

Kao et al [33] cover SQLi attacks with descriptive and investigative methods. They also propose a framework of SQLi Investigation Architecture (SIA) and prove its feasibility against SQLi attacks. Such solution can find hackers within the defined criteria of the SQLi attack, and detect issues or protect data against further attacks. Through machine learning, Xie et al [61] show a method for SQLi detection based on Elastic-Pooling Convolutional Neural Network (EP-CNN) and compare it with traditional

Table 3: Publications classified as Cyber Attacks, chronologically sorted.

	Reference, Keywords	Content
RECV	[38] Network Invasion	– Collect network data to support network forensic analysis, and store it in a MySQL DBMS
SQLI	[44] SQL Injection, Attack Reconstruction	– Propose a network recording solution to detect and capture SQLi. To validate it, they simulate an SQL injection attack on a local MySQL server as a data source
SQLI	[7] SQL Injection	– Evaluate about 900 public domain web apps searched on *Google* using a set of keywords
DETC	[15] Hijacking Detection	– Develop LUDIC (LookUp DIstributed Cache) for detecting domain hijacking attacks
DETC	[36] Wordpress	– Use *John the Ripper*, *Cain* and *Abel* dictionaries to decipher distinct types of passwords based on the password strengthening technique
SQLI	[33] SQL Injection	– Propose a framework (SQLi Investigation Architecture) to detect and combat SQLi attacks
SQLI	[61] SQL Injection, CNN	– Use 4.48 million real weblogs, of which 1/4 are SQLi logs and the others, common logs
RECV	[45] Industrial control	– Propose a forensic framework to analyze and retrieve data of control logic injection attacks
DETC	[57] Intrusion Detection	– List the different Intrusion Detection System (IDS) datasets used to evaluate IDS models

RECV: Data recovery. SQLI: SQL Injection. DETC: Attack detection.

ones. The method automatically extracts the hidden features (unrecognized by humans) of SQLi and identifies the attack traffic, bypassing the regular SQLi. Results show it to be effective, with high recognition accuracy compared to traditional ones.

Attack Detection. These publications focus on detecting many types of cyber attacks. For example, Borgwart et al [15] tackle the DNS Hijacking attack — a name service provided by the Domain Name System (DNS) that is essential for locating resources on the Internet, distributing security mechanisms in an authenticated manner, and facilitating future applications. The authors introduce LUDIC (LookUp DIstributed Cache) to detect and prevent outcomes of such attacks. LUDIC does not changes the existing infrastructure and can be easily integrated into an Intrusion Detection System (IDS) or a firewall while providing immediate benefits.

Kyaw et al [36] carry a dictionary attack experiment against WordPress handled by a fictional person. The attack broke the seven-character password by using well-known online dictionaries. The authors affirm applying password strengthening techniques can mitigate the attack. They also provide insights into implementing a forensic-ready WordPress system and investigating attacks on web applications, such as lockout systems, multi-factor authentication, strong passwords, and fake names.

Finally, Thakkar and Lohiya [57] overview Machine Learning and Data Mining techniques used for IDS, and discuss recent datasets that contain and organize network attack features and new attack categories. The types of network attacks changed over the years; hence, updating the datasets used for evaluating IDS is crucial. Then, the authors discuss recent advances in the attack detection datasets that are available for various research communities.

Data Recovery. These publications focus on recovering data or systems after a cyber attack or during forensics analyses. For example, Ming and Zhong [38] develop a network intrusion model as a forensic tool. The model performs intrusion detection while recovers all network data from the target system to simplify network forensic analysis. From another perspective, some cyberattacks inject malicious control logic into programmable logic controllers (PLCs) to sabotage physical processes (e.g., traffic light signals, nuclear plants). Hence, Qasim et al. [45] propose Reditus, a novel control-logic forensics framework for injection attacks that recovers control logic from suspicious industrial control systems (ICS) network traffic. Reditus assumes there is a built-in decompiler that can transform the control logic into its source code.

By analyzing the papers, we note three relevant aspects: *(i)* attacks collect and recover essential data from applications that use DB; *(ii)* having a DB with attack evidence is mandatory to perform forensic analyses; and *(iii)* when the target is a DB system, criminals prefer SQLi attacks.

5.2 Criminal Intelligence

Criminal intelligence include information compiled, analyzed, or disclosed to anticipate, prevent, or monitor illegal activity. A key point for criminal intelligence is DB, which store vast features on individuals, organizations, and transactions. Hence, gathering data is essential in any law enforcement agency. When acquired, the information from such data may anticipate or prevent crime by building a timeline of criminal activities.

Hence, papers classified as Criminal Intelligence show how DB can gather data and act as an investigation tool. By analyzing data within a DB, law enforcement agencies can better understand illegal activities and build a case to bring offenders to justice. This section then describes **40** papers classified as *Criminal Intelligence*, i.e., publications that analyze DB contents to investigate incidents and build a timeline of criminal activities (**13** in Forensic Investigation), process forensic case data to construct a general a product (**21** in Research Products) and solve forensic problems (**six** in Crime Resolution).

Forensic Investigation. Table 4 lists articles on forensic investigation processes that mainly use DB. Chang et al [17] describe how to use the Windows Registry for forensic analysis and investigation. Windows Registry is a central hierarchical DB with information for configuring system, applications and hardware devices. Hence, it is a signifi-

Table 4: Publications on Forensic Investigation, chronologically sorted.

Reference, Keywords	Content
[17] Windows Registry	– Use Windows registry database, which contains vital data used by windows and users
[39] Forensics on email evidence	– Use Enron Corp. E-mails, 8.70 GB, for 148 mailboxes with 517,431 messages and 3,299 folders
[60] Database Audit System	– Record and audit DB communication packets, which are mirrored by the switch on network traffic
[48] File Forensic Investigation	– Collect and store browser logs in a database considered as evidence of cybercrime
[2] IoT Investigation	– Consider different pieces of evidence that represent suspicious transactions
[9] CHAID, SQL	– SQL code for an automatic fraud-detection software application
[14] Digital traces, investigations	– Consider more than 200 real cases of cybernetics crimes registered by Geneva policy
[19] Database files, messaging	– User location and personal data of three message app in China and South Korea
[10] Privacy impact	– Assess privacy impact by considering DFaaS platform as a case study
[47] Decision making	– Discuss how scientific interpretation principles strengthen investigative process
[54] Mobile, Cloud Traceability	– Use WeChat app, which has data of photos and messages
[64] IoT Botnet Forensics	– Use the Mirai botnet server, available on GitHub
[40] Blockchain investigations	– Analyze blockchain to ensure data integrity on databases

cant forensic resource, and one may use it to prove the authenticity of judgment within the examination process and the forensic analysis phase.

Paglierani et al [39] share a systematic process for email forensics that integrates workflow into the normal forensic analysis and fits the distinct features of email evidence. They focus on detecting non-obvious artifacts related to email accounts, retrieving the data from the service provider, and describing email in a well-structured format.

Wu et al [60] design and implement a DB auditing system for distinct DB types. They use bypass mode to deflect any DB performance delays. Also, it provides a flexible audit system that decides which DB should be recorded or not.

Salunkhe et al [48] study how Decision Trees allow systems to quickly, easily, and affordably analyze log data on many formats for file forensic analysis. They propose an analysis strategy that aids investigators to detect criminal activities by collecting log files. When a crime occurs, the system investigates and stores the shreds of evidence in DB.

Al-Dhaqm et al [2] propose the Common Database Forensic Investigation Process (CDBFIP) to investigate cybercrime activities and cyber breaches over the Internet of Things. They propose a four-phase process: identification, artifact collection, artifact analysis, and documentation and submission process. Unifying these processes into an abstract diagram increases the knowledge available to users, newcomers, and professionals, and reduces the complexity and ambiguity of the investigation.

Another issue is to discern inexact or non-obvious similarities between cybercrimes. Hence, Bollé and Casey [14] propose a solution to finding links and repetitions between cases through the quasi-similarity calculation of distinct digital traits and the Levenshtein distance. Automatically detecting such similarities gives investigators a better understanding of the criminal context and the actual phenomenon, and can reveal many related crimes.

Bach et al [9] apply data mining techniques (CHAID decision tree) to discover patterns in fraud related to internal controls in a project-based organization. They increase the efficiency of internal fraud detection, which results in a SQL code used to develop an automatic fraud-detection application.

Choi et al [19] analyze the local and the formats of personal data files in three instant messaging apps (KakaoTalk, NateOn, and QQ) that use encryption. Thus, the authors apply reverse engineering to examine the encryption and decryption procedures of the internal databases of such applications.

Zhang et al [64] study a famous family of IoT bot malware – Mirai. They design a Mirai botnet network and run forensics analyses on its server. They identify and discuss forensic items left on the attacker's terminal command, control server, DB server, scan receiver, loader, and network packets.

Bas Seyyar and Geradts [10] explore and measure privacy risks specific to law enforcement activities that require processing large amounts of data. They assess privacy impact (PIA) on a big data forensics platform. They also answer the question of how a PIA should be performed for large-scale DF operations and describe privacy risks and threats. Finally, they articulate concrete privacy measures to demonstrate compliance with the Policy Directive.

Sharma et al [54] present a mobile cloud forensics process that incorporates inter and intra-application analysis and time synchronization allied to traditional forensics. Time synchronization enables the investigator to perform forensic analysis of the mobile cloud application concisely; then, inter and intra-application analysis process ensures the extraction of forensic evidence and enriches the performance of event traceability in the cloud, using the metadata of possible mobile evidence.

Collecting and analyzing digital and multimedia evidence require many decisions from forensic professionals. Then, Ryser et al [47] present a well-established logical framework for making structured decisions at all stages of an inquiry in DF, aiming to mitigate the risks of errors and establish adequate trust in digital and multimedia evidence.

Palanisamy and Nataraj [40] review the application of blockchain in an enterprise information management system (IMS) and explore how to incor-

Table 5: Publications that propose Research Products, chronologically sorted.

Reference, Keywords	Content
[26] Anti-Money Laundering	– Use a database with 100 fictitious records to emulate banking transactions
[42] Information Accountability	– Present a tool to detect tampering in high-performance DB.
[51] Enterprise Rights	– Microsoft RMS and Adobe LiveCycle
[43] Android Timestamps	– Introduce the Authenticity Framework for Android Timestamps (AFAT), tested over SQLite DB
[31] Education	– Interviews with forensic experts as judges, lawyers and prosecutors
[37] Forensic analysis on Android	– Present a tool to forensic analysis Android apps (Fordroid) tested over a 100 android app DB
[56] Digital forensics language	– Present a new digital forensics language called Nugget
[32] AI Speaker Ecosystems	– Use a DB with audio files from 4 models: Clova da NAVER, Kakao I da KAKAO, NUGU da SKT and GiGA Genie from KT
[5] Integrated Incident Database Forensic	– Use data from IEEE Xplore, Scopus, ACM, SpringerLink and Elsevier to develop an Integrated Incident Response Model
[30] Learning Efficacy of Digital Forensics Concepts	– Present an openly available virtual reality (VR) digital forensic education game via the Immersive VR Education ENGAGE platform
[50] Metadata-based storage reconstruction and carving	– Present the tool LAYR (available on GitHub) and use data from the Digital Forensics Tool Testing (DFTT) project to evaluate it
[49] Main Memory Images	– Perform experiments to build a dataset with adulterated images from main memory
[59] Digital forensics as a service	– Describe learned lessons of using DF as a Service ([58]) platform in a forensic and legal context
[62] True source of cyber crime	– Propose a framework (Root-Tracker) for identifying real source of cybercrime, and evaluate it over a network infrastructure built with different devices
[24] Cyber attacks	– Propose a framework for reviewing/ investigating cyber-attacks (D4I) and evaluate it over phishing
[6] Database Forensic	– Validate the Database Forensic Investigation Metamodel
[23] IA forensics	– Show how databases serve as valuable input for IA models supporting forensic analysis
[52] VISU criminal suspect	– VISU is connected to a criminal and suspect DB provided by law enforcement authority
[25] MORPH database	– Empirically evaluate on the academic MORPH database using a facial recognition system
[63] real-time image	– Provides YOLO v5, a new method for database auditing
[8] Unified Forensic Model	– Propose the Unified Forensic Model (UFM) for the database forensics analysis

porate blockchain technology in CampusStack, an integrated IMS, to audit the DB and ensure data integrity. They conclude decentralization must be considered and adopt the data auditing algorithm with the blockchain technology in critical DB.

Research Products. Here, papers propose new frameworks, tools, models resulting from research, as listed in Table 5. By analyzing **21** papers in this group, we found: three tools – DRAGOON, Fordroid, LAYR [37, 42, 50]; three frameworks – AFAT, Root-Tracker, D4I) [24, 43, 62]; and one digital forensics language – Nugget [56]. The search strings returned works since 2006, but the oldest one classified as Criminal Intelligence that proposes a tool is from 2012, indicating a recent research field in forensics. Also, the three tools have distinct goals: DRAGOON detects tampering in DB; Fordroid analyzes Android mobile apps; and LAYR combines methods to optimize storage reconstruction techniques. The frameworks have also distinct goals: AFAT investigates the authenticity of timestamps on Android smartphones; Root-Tracker aims to identify the source of network security attacks; and D4I improves the digital forensics process, especially the examination of cyber-attacks. Then, Nugget is a domain-specific language to specify the data flow of a forensic inquiry, execute forensic computation, and return a log of the inquiry.

There are also two works on services. Van Beek et al [59] provide Digital Forensics as a Service (DFaaS) implementations to agencies and share lessons learned within a forensic and legal context. Then, Henseler and van Loenhout [31] depict standards and requirements of the Dutch Register of Judicial Experts (NRGD), with requirements for a person to qualify as an NRGD Registered Specialist.

The remaining works tackle other current issues. First, in Internet of Things (IoT) and Digital Forensic, an AI speaker is a cloud-based IoT system built by merging an AI speaker and IoT devices. Such AI speakers are continuously operating and may provide vital evidence for digital forensics. However, privacy issues may arise. Hence, Jo et al [32] propose five digital forensic analysis models for four distinct AI speakers. They introduce a forensic tool for collecting user command history for NAVER Clova (Korean IA Speaker) as a research product. Likewise, Al-Dhaqm et al [5] present an Integrated Incident Response Model (IIRM) to recognize, respond, mitigate and recover from a potential database incident. IIRM is a hybrid model with four main goals: establish a plan to prevent DB disasters, investigate and seek possible evidence, recover DB operations, and share DB disaster knowledge.

Hassenfeldt et al [30] present an openly available virtual reality (VR) DF education game via the Immersive VR Education ENGAGE platform. They investigate methods to hack and extract data from the Nintendo 3DS storage system, the NAND.

Flores et al [26] use data mining and data warehousing to assist digital forensic investigations related to money laundering in compliance with the Know-Your-Customer 'KYC' policies defined inside an organization. They prove BI tools could support the analysis of money laundering evidence by using simple DB transactional logs to present the investigation results more comprehensively than using extensive written Suspicious Activity Reports (SARs).

Table 6: publications that propose Crime Resolution, chronologically sorted.

Reference, Keywords	Content
[18] Mining Criminal Databases	– Build and use a data warehouse with records of 378,000 cars stolen in 11 years in Taiwan
[21] Crimes from Web	– Consider over 33k crime reports in the "Onde Fui Roubado" platform within [2012,2016]
[27] Cryptocurrency investigations	– Provide an empirical analysis of CoinJoin transactions
[41] Vulnerabilities, Forensic Issues	– Discuss various types of cloud attacks and inform how to mitigate them
[1] Crime-scene image	– Apply machine learning to a subset of 60,520 images from the illicit drug database
[12] Suspect Identification	– Train a model for suspect id over 100k images of CelebaFaces

Schrittwieser et al [51] explore forensic techniques for Enterprise Rights Management (ERM) systems and develop application-specific guidelines for forensic investigations, targeting Microsoft Active Directory Rights Management Services (RMS) and Adobe LiveCycle Rights Management. Such ERM systems heavily use databases to store keys and relevant forensic investigation data. Also, they show the critical role of database forensics for inquiries in ERM systems. They conclude that MySQL InnoDB storage engine's data and log files store enough data to reveal older versions of LiveCycle policies and even allow recovering deleted cryptographic keys.

Schneider et al [49] study how to manipulate main memory copies obtained during a digital investigation based on controlled experiments. Note, tampered digital evidence may compromise its interpretation. Handling prominent memory dumps is problematic, as they detect most spoofs. Overall, tampering with main memory dumps seems more difficult than tampering with hard disk images, but the likelihood of misleading an analyst is also higher.

Al-Dhaqm et al [6] aim to validate the Database Forensic Investigation Metamodel (DBFIM) by using the qualitative method of face validity. DBFIM proposes resolving interoperability, heterogeneity, complexity, and ambiguity in a database forensics investigation, where various models were identified, collected, and reviewed to develop DBFIM.

Delgado et al [23] argue that AI connects scientists and forensic investigators. Indeed, they highlight the growth of data analytics in all fields of life, starting from enterprises to public health solutions during the COVID-19 pandemic. Such examples are relevant instances of data analytics for society and are suitable for forensic intelligence.

Sethuraman et al [52] present VISU, a prototype of a three-dimensional printed robot for crowd surveillance. It captures the surround with a built-in camera and sends it to a cloud DB. Then, VISU connects to the criminal and suspect DB provided by an authority to identify any suspicious activity. The authors suggest VISU works for crowd surveillance purposes in both crowded and non-crowded locations with minimal production cost.

On facial recognition analysis, Drozdowski et al [25] analyze the watch list imbalance effect, which can cause (unintentional) discrimination based on individual demographic properties; i.e., the impacts of an unbalanced DB regarding a specific demographic group (e.g., containing many more men than women). They conduct an empirical evaluation of the MORPH DB by using a facial recognition system. The authors show unbalanced samples influence the fairness of biometric identification systems. Further, it indicates demographically equitable biometric verification systems do not necessarily guarantee demographically fair identification.

Zhang et al [63] propose a real-time DB operation recognition method based on the YOLO algorithm. It performs real-time image recognition on the DB operation and maintenance interface. The instruction image converts into a character sequence, and the DB audit is complete according to the rules.

Alhussan et al [8] propose the Unified Forensic Model (UFM): initialization, acquisition, investigation, restoration and recovery, and evaluation (each with processes and activities). UFM aims to collect, preserve, identify, analyze, reconstruct, and document DB incidents in the forensic field.

Crime Resolution. Here, the publications discuss new computational solutions that may help in crime resolution, as listed in Table 6. Chen [18] applies data mining techniques (such as classification, clustering, association rule, and prediction) to manipulate data on stolen automobiles in Taiwan. He explores information hidden in such data and provides knowledge to transportation, insurance, and police agencies for decision support. The results allow understanding of automobile theft, finding stolen automobiles, questioning theft claims, etc.

Silveira and Brandão [21] collect data from websites with crime occurrences and apply clustering analysis to discover crime patterns. They highlight that more than 41% of the crimes were not reported, most of which are thefts and robberies occurring at night and dawn. Moreover, minor offenses present different patterns of serious crimes, and crime patterns differ in rich and poor neighborhoods.

Analyzing cryptocurrency payment flows has become a critical forensic method in law enforcement and is used to investigate a broad spectrum of criminal activities. Then, Fröwis et al [27] identify internationally accepted standards and rules for supporting suspicions and providing evidence in court, and project them onto current cryptocurrency forensics.

Pandi et al [41] discuss the STRIDE threat model and the misuse of cloud services for many malicious

Table 7: Proposed classification considering the phases of digital forensics and their guidelines.

Phase / Timeline	Goal	Methodology
Prevention Before crime	Prevent digital crimes	Vulnerability analysis and planning
Detection During a crime	Identify crimes – where how when	Find malicious activities among normal ones
Recovery After a crime	Restore data & processes	Uncover deleted, lost data; restore damaged evidence

purposes, as most criminals flee for lack of evidence. The article addresses some forensic issues in a cloud computing environment, such as collecting and analyzing evidence in the digital world.

Abraham et al [1] present two proof of concept for crime-scene image classification models. They evaluate two ML classifiers using images from an illicit drug database as ground truth for automatically cataloging and classifying crime scene images.

Bhajaj et al [12] propose a system to generate a face image by using a technology called Conditional Generative Adversarial Networks (C-GANs). The system considers gender, hair color, face shape, etc., as input data. It then shows a real-time face preview after inputting the data. Such a face-matching feature is evaluated over criminal databases to check whether the suspect already exists.

Here, publications reveal digital forensics has been proposing and using increasingly robust computational tools, models, or frameworks to identify and track crimes committed through digital devices. Also, exciting research presents methods or processes to guarantee that collected digital evidence serves as proof in a court of law. Hence, some works seek to solve or prevent crimes from occurring based on prior knowledge through data analysis.

Overall, criminal intelligence plays a critical role in DF. Indeed, our work may inspire or open paths for future research in integrating AI with DF. Combining such areas can provide potential solutions for forensic purposes. AI can help automate simple and repetitive tasks, freeing up time for investigators to focus on more complex issues. This integration may also inspire new advanced tools and techniques that can aid in preventing and detecting cybercrime. Integrating databases, digital forensics, and AI holds tremendous promise for the future of criminal intelligence and law enforcement, and further research in this area may yield significant benefits.

6. PUBLICATIONS BY PHASE

Simply put, within digital forensics, experts can act over three critical goals: prevention, detection, and recovery; as listed in Table 7. After explaining the taxonomy, we now draw a parallel between selected works according to each phase in Table 8. Next, we discuss the main aspects of such perspectives and how DB may relate to them.

Prevention. Preventing cybercrime requires digital forensic experts proactively identify vulnerabilities intrinsic to an organization's processes and systems. Such vulnerabilities may be understood as possible flaws in systems or processes that allow cybercriminals to exploit them to access data or even control the system. Thus, actions aimed at identifying and reducing the risks of vulnerabilities may include: implementing new information security technologies; acquiring or developing computational tools to perform periodic expert examinations; implementing methods of ensuring that systems are traceable (i.e., using logs); and constant monitoring and tracking any suspicious activity.

By drawing a parallel between this phase and our goal (identifying works that act direct or indirectly with DB), we classify as Prevention those papers whose goals (primary or not) include working to assist in: identifying vulnerabilities in DB systems and minimizing their risks; and ensuring data is traceable after a potential intrusion (Table 8).

Detection. Once a cyber attack is detected, DF experts must identify all evidence of such an incident and which systems, data and processes were affected. It is also crucial to keep all digital evidence found intact and unaltered to avoid inconclusive results. Further, tracking and analyzing all related data are needed to identify the attack causes.

We classify as *Detection* articles that propose models or methods to identify attacks in database environments. In addition, we also include works that aim to avoid violating the integrity of digital evidence found for ensuring its use as evidence in court. One of the most common methods of detecting and tracing attacks is the use of logs.

Recovery. After proper identifying and tracking all data and systems that have suffered a digital attack, recovering such losses is paramount. The damage levels in data and computational systems depend on the episode degree, i.e., detection time (to end such an attack asap), number of systems and volume of data involved, and so on. We classify as *Recovery* articles that propose efficient methods of recovering data or data systems to return to their total capacity. Recovery can include restoring corrupted, altered, or deleted data.

7. DISCUSSION AND CHALLENGES

Overall, Digital Forensics primarily collects, preserves and analyzes digital evidence for crime and legal related scenarios; whereas Databases are responsible for managing and storing data in an organized and efficient system. The intersection between these areas is vast, with many studies on device data extraction, data recovery, and digital evidence as instances of data building. Hence, this section answers each SLR question. We note such discussions may guide research in the intersection

Table 8: Papers on prevention (P), detection (D) and recovery (R), sorted by taxonomy class and year.

Ref.	Class	Forensic Phase			Ref.	Class	Forensic Phase			Ref.	Class	Forensic Phase			Ref.	Class	Forensic Phase		
[38]	CA		D		[51]	CI			R	[19]	CI	P			[41]	CI	P		
[44]	CA		D		[39]	CI			R	[5]	CI	P	D	R	[1]	CI		D	
[7]	CA		D		[60]	CI	P			[64]	CI		D		[6]	CI	P	D	
[15]	CA		D		[43]	CI		D		[10]	CI	P			[23]	CI	P	D	
[36]	CA	P			[48]	CI		D		[30]	CI		D		[52]	CI		D	
[33]	CA		D		[2]	CI		D		[49]	CI	P			[25]	CI	P		
[61]	CA		D		[21]	CI		D		[54]	CI		D		[63]	CI		D	
[45]	CA		D		[31]	CI	P			[59]	CI		D		[40]	CI		D	R
[57]	CA		D		[37]	CI			R	[47]	CI	P	D		[8]	CI	P		
[17]	CI	P			[56]	CI			R	[27]	CI			R	[12]	CI		D	
[18]	CI		D		[14]	CI	P	D		[62]	CI		D						
[26]	CI		D		[9]	CI		D		[24]	CI		D		**CA:** Cyber Attacks				
[42]	CI		D		[32]	CI		D		[50]	CI			R	**CI:** Criminal Intelligence				

between databases and digital forensic, reducing the gap between both knowledge areas.

When and where the studies were published? The search strings retrieved relevant works since the year 2006. However, only from 2017 onwards there was a growth in the number of publications that address digital forensics and databases. In general, *String 3* retrieved more papers, which may indicate trending keywords. Also, Scopus and Science Direct are the digital libraries that returned more publications.

What kinds of research are there? Works use different statistical strategies to prove or validate their hypotheses; i.e., mostly quantitative. However, we also find exceptions that use a qualitative approach and present reviews or analyses of forensic methods or issues in a non-numerical way, e.g., [31].

What is the focus of data-driven digital forensics? We identified four classes of works, each with three categories: *Data Building* – Device Data Extraction, Data Recovery, and Digital Evidence; *DBMS* – Performance Analysis, Security Roles, and Data Recovery; *Cyber Attacks* – SQLi, Attack Detection and Data Recovery; and *Criminal Intelligence* – Forensic Investigation, Research Products, and Crime Resolution. SLR results for the first two are in [53]. Still, almost half of the articles focuses on Criminal Intelligence, i.e., digital forensics follows the current trend of intelligent solutions.

We have also mapped all articles according to the phases of DF covered: Prevention, Detection, and Recovery. Grouping publications on stages allows companies and researchers to quickly find current solutions for one or a combination of phase(s).

What are the advances and potential challenges in the area? It is unquestionable that forensic investigation is data-driven. Indeed, DF and DB have an almost interdependent relationship as both areas have data as a central element. On the one hand, research on databases (mostly) aims to discover how to optimize data storage and retrieval (querying) as well as improve internal tools and mechanisms, e.g., logging and transaction control. On the other hand, research on data-oriented digital forensics acts on at least two main points: organizing/storing data for further investigation or using the data stored for investigations. We also note advances in using machine learning techniques (mostly mining) over forensics data. Chen [18] applies data mining over stolen automobile data and uses classification, clustering, association, and prediction algorithms.

What issues are still open? In 2010, Garfinkel [28] pointed out some technological challenges for digital forensics, including: the increase in device storage capacity, which makes processing data promptly difficult; the diversity of hardware that complicates data access standardization; and the proliferation of different data, operating systems and file systems that increase the complexity and cost of developing tools to exploit data. Technology has since changed a lot, but so have the issues. For example, one challenging opportunity is to keep in sync forensics and database updates on data formats, systems programming languages, etc. Such changes may favor security as technologies evolve and possible failures are already known (especially for cybercrimes). Still, new technologies also mean new forms of cyberattacks. Especially for databases, another challenge is maintaining compatibility between systems.

Further, new technologies demand preparing specialists and building approaches to forensic investigation. For example, many works focus on extracting data from devices with specific systems, such as android, IOS, cameras and games. One research opportunity is to define a way to standardize: extracting data from different devices; and the terminologies and concepts most used in the intersection between the areas. Another opportunity is to define a knowledge base that enables to store and share knowledge involving these two areas.

Amidst the challenges that new technologies represent, cyberculture promotes a series of social changes with significant impacts on human relationships. The main reason is that it integrates individuals, companies, devices, networks, artificial intelligence, and the internet of things without the barriers of time or geographic limitations. The constant

exchange of information means the virtual environment needs to be regulated in the ethical field (using and treating personal and private data) within private and public settings.

Further, privacy has changed through the years. Still, individual guarantees and freedoms (e.g, the right to privacy, intimacy, and the inviolability of the home) are protected by the legal system of states and countries. Hence, a key open issue is dealing with data during investigations with proper ethical conduct, which is still fuzzy worldwide.

In summary, the intersection of databases with digital forensics is essential for investigating cybercrimes and legal disputes, as well as preserving data security. Professionals in such areas must know the techniques and tools available to ensure that data can be collected, preserved and analyzed correctly during investigations while adopting ethical behavior. After all, having ethics is crucial when conducting forensic database analyses to ensure proper, responsible work (and research).

8. CONCLUSION

This article presented a systematic literature review on the intersection between digital forensics and DB areas. One goal was to promote a better categorization and synthesis and, hence, to simplify the search and access to related work. We proposed a new taxonomy (focusing on cyber-attacks and criminal intelligence), and uncovered a specific pattern of publications – i.e., the three phases: prevention, detection, and recovery. Besides classifying all papers within the taxonomy, we have also mapped them to the patterns. Such a mapping enables researchers and organizations to quickly find leading solutions grouped by forensic purpose.

Based on the SLR, we may conclude: the number of publications on digital forensics has increased, given the evolution of computing and the relevance of solving digital crimes; forensic expertise has expanded in various computational and digital contexts; and there are distinct digital forensics functions related to data usage and development. Advances in DB are crucial to forensic investigations as well, not only in the search for better processing speed (through indexes and specialized access methods) but also in the accuracy of results (through new query techniques and correlation, for example).

Future work shall expand the coverage of publications over other areas related to Databases, such as Data Mining and Machine Learning, which may aid forensics analysis. Also, we plan to study further privacy and ethical issues uncovered here.

Acknowledgments. Funded by Research Scholarship Programs - IFMG, CNPq, and CAPES, Brazil.

9. REFERENCES

[1] J. Abraham et al. Automatically classifying crime scene images using machine learning methodologies. *Forensic Sci Int'l: Dig Investigation*, 39, 2021.

[2] A. Al-Dhaqm et al. Cdbfip: Common database forensic investigation processes for internet of things. *IEEE Access*, 5:24401–24416, 2017.

[3] A. Al-Dhaqm et al. Categorization and organization of database forensic investigation processes. *IEEE Access*, 8:112846–112858, 2020.

[4] A. Al-Dhaqm et al. Database forensic investigation process models: A review. *IEEE Access*, 8:48477–48490, 2020.

[5] A. Al-Dhaqm et al. Towards the development of an integrated incident response model for database forensic investigation field. *IEEE Access*, 8:145018–145032, 2020.

[6] A. Al-Dhaqm et al. Face validation of database forensic investigation metamodel. *Infrastructures*, 6(2):1 – 19, 2021.

[7] D. Alam et al. A case study of sql injection vulnerabilities assessment of .bd domain web applications. In *CyberSec*, pages 73–77, 2015.

[8] A. A. Alhussan et al. A unified forensic model applicable to the database forensics field. *Electronics (Switzerland)*, 11(9), 2022.

[9] M. P. Bach et al. Internal fraud in a project-based organization: Chaid decision tree analysis. *Procedia Computer Science*, 138:680–687, 2018.

[10] M. Bas Seyyar and Z. Geradts. Privacy impact assessment in large-scale digital forensic investigations. *FSI: Dig. Investigation*, 33:200906, 2020.

[11] A. Beirami et al. Trusted relational databases with blockchain: design and optimization. *Procedia Computer Science*, 155:137–144, 2019.

[12] P. Bhajaj et al. Figsi—facial image generation for suspect identification. *LNNS*, 351:877 – 891, 2022.

[13] D. M. Blei et al. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[14] T. Bollé and E. Casey. Using computed similarity of distinctive digital traces to evaluate non-obvious links and repetitions in cyber-investigations. *Dig. Investigation*, 24:S2–S9, 2018.

[15] A. Borgwart et al. Detection and forensics of domains hijacking. In *GLOBECOM*, 2015.

[16] E. Casey. *Digital evidence and computer crime: Forensic science, computers, and the internet.* Academic press, 2011.

[17] K. Chang et al. Initial case analysis using windows registry in computer forensics. In *FGCN*, 2007.

[18] P. S. Chen. Discovering investigation clues through mining criminal databases. In H. Chen and C. Yang, editors, *Intelligence and Security Informatics: Techniques and Applications*, pages 173–198. Springer Berlin Heidelberg, 2008.

[19] J. Choi et al. Digital forensic analysis of encrypted database files in instant messaging applications on windows operating systems: Case study with kakaotalk, nateon and qq messenger. *Dig. Investigation*, 28:S50–S59, 2019.

[20] J. Cohen. A coefficient of agreement for nominal scales. *EPM*, 20(1):37–46, 1960.

[21] M. da Silveira and W. Brandão. Characterizing crimes from web. In *BraSNAM*, 2017.

[22] D. Dave et al. Management of implicit requirements data in large SRS documents: Taxonomy and techniques. *SIGMOD Rec.*, 51(2):18–29, 2022.

[23] Y. Delgado et al. Forensic intelligence: Data analytics as the bridge between forensic science

and investigation. *FSI: Synergy*, 3, 2021.
[24] A. Dimitriadis et al. D4i - digital forensics framework for reviewing and investigating cyber attacks. *Array*, 5:100015, 2020.
[25] P. Drozdowski et al. The watchlist imbalance effect in biometric face identification: Comparing theoretical estimates and empiric measurements. In *ICCVW*, pages 3750–3758, 2021.
[26] D. A. Flores et al. Combining digital forensic practices and database analysis as an anti-money laundering strategy for financial institutions. In *EIDWT*, 2012.
[27] M. Fröwis et al. Safeguarding the evidential value of forensic cryptocurrency investigations. *FSI: Dig. Investigation*, 33:200902, 2020.
[28] S. L. Garfinkel. Digital forensics research: The next 10 years. *Dig. Investigation*, 7:S64–S73, 2010.
[29] A. Guarino. Digital forensics as a big data challenge. In *ISSE*, pages 197–203, 2013.
[30] C. Hassenfeldt et al. Exploring the learning efficacy of digital forensics concepts and bagging & tagging of digital devices in immersive virtual reality. *FSI: Dig. Investigation*, 33:301011, 2020.
[31] H. Henseler and S. van Loenhout. Educating judges, prosecutors and lawyers in the use of digital forensic experts. *Dig. Investigation*, 24:S76–S82, 2018.
[32] W. Jo et al. Digital forensic practices and methodologies for AI speaker ecosystems. *Dig. Investigation*, 29:S80–S93, 2019.
[33] D. Kao et al. A framework for sql injection investigations: Detection, investigation, and forensics. In *SMC*, pages 2838–2843, 2018.
[34] M. Z. Khan et al. Cyber forensics evolution and its goals. In *Critical Concepts, Standards, and Techniques in Cyber Forensics*, pages 16–30. IGI Global, 2020.
[35] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical report, Un of Durham, 2007.
[36] A. K. Kyaw et al. Dictionary attack on wordpress: Security and forensic analysis. In *InfoSec*, pages 158–164, 2015.
[37] X. Lin et al. Automated forensic analysis of mobile applications on android devices. *Dig. Investigation*, 26:S59–S66, 2018.
[38] H. Ming and S. LiZhong. A new system design of network invasion forensics. In *ICCEE*, volume 2, pages 596–599, 2009.
[39] J. Paglierani et al. Towards comprehensive and collaborative forensics on email evidence. In *CollaborateCom*, pages 11–20, 2013.
[40] A. M. Palanisamy and R. V. Nataraj. A novel methodology to ensure data integrity in enterprise information systems using blockchain technology. In *ICEEICT*, pages 1–5, 2022.
[41] G. S. Pandi (Jain) et al. Exploration of vulnerabilities, threats and forensic issues and its impact on the distributed environment of cloud and its mitigation. *Procedia Computer Science*, 167:163–173, 2020.
[42] K. E. Pavlou and R. T. Snodgrass. Dragoon: An information accountability system for high-performance databases. In *ICDE*, 2012.
[43] H. Pieterse et al. Playing hide-and-seek: Detecting the manipulation of android timestamps. In *ISSA*, 2015.
[44] A. Pomeroy and Q. Tan, Effective sql injection attack reconstruction using network recording. In *IEEE CIT*, pages 552–556, 2011.
[45] S. A. Qasim et al. Control logic forensics framework using built-in decompiler of engineering software in industrial control systems. *FSI: Dig. Investigation*, 33:301013, 2020.
[46] Q. Rossy et al. Integrating forensic information in a crime intelligence database. *FSI*, 230(1-3):137–146, 2013.
[47] E. Ryser et al. Structured decision making in investigations involving digital and multimedia evidence. *FSI: Dig. Investigation*, 34:301015, 2020.
[48] P. Salunkhe et al. Data analysis of file forensic investigation. In *SCOPES*, pages 372–375, 2016.
[49] J. Schneider et al. Tampering with digital evidence is hard: The case of main memory images. *FSI: Dig. Investigation*, 32:300924, 2020.
[50] J. Schneider et al. Unifying metadata-based storage reconstruction and carving with layr. *FSI: Dig. Investigation*, 33:301006, 2020.
[51] S. Schrittwieser et al. Digital forensics for enterprise rights management systems. In *iiWAS*, 2012.
[52] S. C. Sethuraman et al. Visu: A 3-d printed functional robot for crowd surveillance. *IEEE Consumer Electronics Mag.*, 10(1):17–23, 2021.
[53] D. B. Seufitelli, M. A. Brandão, and M. M. Moro. Exploring the intersection between databases and digital forensics. *Journal of Information and Data Management*, 13(3), Sep. 2022.
[54] P. Sharma et al. Enhanced forensic process for improving mobile cloud traceability in cloud-based mobile applications. *Procedia Computer Science*, 167:907–917, 2020.
[55] L. F. Sikos. Packet analysis for network forensics: A comprehensive survey. *FSI: Dig. Investigation*, 32:200892, 2020.
[56] C. Stelly and V. Roussev. Nugget: A digital forensics language. *Dig. Investigation*, 24:S38–S47, 2018.
[57] A. Thakkar and R. Lohiya. A review of the advancement in intrusion detection datasets. *Procedia Computer Science*, 167:636–645, 2020.
[58] R. Van Baar et al. Digital forensics as a service: A game changer. *Digital Investigation*, 11:S54–S62, 2014.
[59] H. van Beek et al. Digital forensics as a service: Stepping up the game. *FSI: Dig. Investigation*, 35:301021, 2020.
[60] K. Wu et al. The design and implementation of database audit system framework. In *ICSESS*, 2014.
[61] X. Xie et al. Sql injection detection for web applications based on elastic-pooling cnn. *IEEE Access*, 7:151475–151481, 2019.
[62] P. R. Yogesh and D. S. R. Backtracking tool root-tracker to identify true source of cyber crime. *Procedia Computer Science*, 171:1120–1128, 2020.
[63] L. Zhang et al. Research and implementation of database operation recognition based on yolo v5 algorithm. In *CISAI*, pages 367–372, 2021.
[64] X. Zhang et al. Iot botnet forensics: A comprehensive digital forensic case study on mirai botnet servers. *FSI: Dig. Investigation*, 32:300926, 2020.

Apache Wayang: A Unified Data Analytics Framework

Kaustubh Beedkar[△,‡], Bertty Contreras-Rojas[◇], Haralampos Gavriilidis[◇], Zoi Kaoudi[⋆,‡], Volker Markl[◇], Rodrigo Pardo-Meza[◇], Jorge-Arnulfo Quiané-Ruiz[⋆,‡*]

Indian Institute of Technology Delhi[△] Technische Universität Berlin[◇] IT University of Copenhagen[⋆] Databloom Inc.[‡]

ABSTRACT

The large variety of specialized data processing platforms and the increased complexity of data analytics has led to the need for *unifying data analytics* within a single framework. Such a framework should free users from the burden of (i) choosing the right platform(s) and (ii) gluing code between the different parts of their pipelines. Apache Wayang (Incubating) is the only open-source framework that provides a systematic solution to unified data analytics by integrating multiple heterogeneous data processing platforms. It achieves that by decoupling applications from the underlying platforms and providing an optimizer so that users do not have to specify the platforms on which their pipeline should run. Wayang provides a unified view and processing model, effectively integrating the hodgepodge of heterogeneous platforms into a single framework with increased usability without sacrificing performance and total cost of ownership. In this paper, we present the architecture of Wayang, describe its main components, and give an outlook on future directions.

1. INTRODUCTION

The research and industry communities have developed a variety of data processing platforms (platforms, for short) to enable users to efficiently extract value from their data. Each platform excels in different aspects of the design space. For instance, PostgreSQL performs better than Vertica for OLTP workloads, but Vertica performs better for OLAP workloads. Apache Spark, on the other side, can outperform both database systems for batch data processing on big datasets.

Consequently, users face a *zoo of specialized platforms* to perform data analytics. They typically run their data analytics at a higher cost than necessary, as selecting the right platform is daunting. Furthermore, modern applications often require to perform data analytics that goes *beyond the limits of a single* platform, making the selection of platforms even more difficult.

To ease the platform selection task, we require *unifying data analytics* within a single framework, i.e., applications should run over any set of platforms seamlessly and efficiently. The need for unified data analytics can stem from simple tasks, such as k-means clustering, to very complex tasks, such as a data science pipeline that includes data cleaning, preparation, feature extraction, and model training. Unified data analytics is quickly becoming essential as new applications emerge.

We distinguish between two general cases of unified data analytics: (i) an entire task is executed on a single platform and, based on the circumstance, this platform can vary, and (ii) a task is split into sub-tasks which are executed on multiple platforms. In particular, we identify four situations when unified data analytics is required [10]: **Platform Independence** refers to the situation where one needs to run an entire task on any arbitrary platform. This requires re-implementing applications when new platforms emerge or when the workload changes. **Opportunistic Cross-Platform** refers to the situation where performing a single task using multiple platforms brings significant performance reasons. **Mandatory Cross-Platform** refers to the fact that modern applications need to go beyond the functionalities offered by a single platform. **Polystore** refers to the situation where applications need to access and process data stored in different data stores (data lakes). In all the above cases, developers typically must write ad-hoc programs to wire multiple platforms together. However, integrating platforms is tedious, repetitive, and error-prone. Figure 1 illustrates these four cases with systems that can handle each case.

Therefore, supporting unified data analytics is crucial in many cases. Apache Wayang (Incubating)[1], Wayang for short, is the first open-source

*To the memory of Jorge: the originator of Wayang who passed away unexpectedly in May 2023.

[1]https://wayang.apache.org/

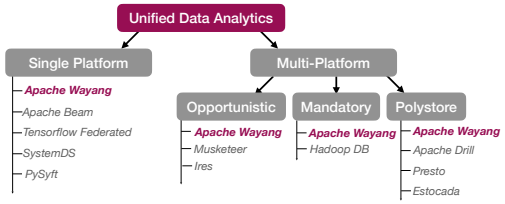


Figure 1: Taxonomy for unified data analytics.

framework that provides applications with unified data analytics capabilities. The main goal of Wayang is to decouple applications from platforms so that they can run analytics on one or more platforms seamlessly and efficiently. Developers can provide their data analytics tasks programmatically (using Java, Scala, or Python) or declaratively (via the SQL and ML libraries). Wayang, in turn, takes an input task and *optimizes* it to produce efficient execution plans, which might run over multiple platforms. We refer to an efficient execution plan as the plan that allows Wayang to execute a given task with a low cost. By default, it considers the cost to be the execution time, but users can provide their own cost function, such as monetary cost.

Wayang presents itself as a full-fledged and efficient cross-platform data processing system for unified data analytics. As of today, it supports a variety of platforms: Spark, Flink, PostgreSQL, GraphX, Giraph, and its in-memory Java-based executor[2]. Wayang originated from the Rheem project [3, 13], is currently incubating in the Apache Software Foundation, and is used by several companies. In particular, Databloom, an AI startup, has been created around Wayang [2].

Our contributions in this paper are as follows. We introduce Apache Wayang (Incubating), which comes with a novel system architecture allowing the integration of different platforms (Section 2). Then, we present the core components of Wayang, including a cross-platform query optimizer that alleviates users from any platform decisions (Section 3). Furthermore, we introduce Wayang's approach to running data analytics on any platform (Section 4). We additionally present the new Polyglot module, which will allow developers to add support for UDFs in any desired programming language by implementing only two core abstractions. Finally, we briefly discuss Wayang's adoption (Section 5), related work (Section 6), and our current efforts towards Wayang 2.0 (Section 7).

2. OVERVIEW

Wayang's main goal is to unify data analytics by

[2]GraphChi is outdated and is going to be removed in our next release.

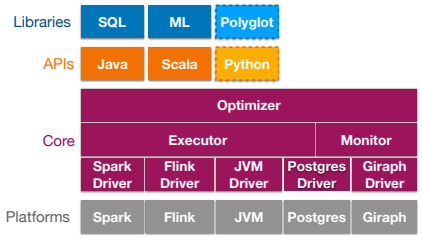


Figure 2: Wayang's software stack.

decoupling applications from the underlying platforms and to provide cross-platform data processing. Figure 2 shows the software stack of Wayang.

In the bottom layers, there are the different data storage mediums and the supported data processing platforms. On top of these, Wayang's core consists of the following main components: the optimizer, the executor, the monitor, and platform-specific drivers. Wayang currently supports two main APIs: the Java one and the Scala one. A Python API is currently under development. Besides using any of the supported languages, users can directly input SQL queries via the SQL library, which transforms them into a Wayang plan. Wayang also comes with an ML library for running ML tasks. Users can directly utilize the provided algorithms or can implement their own algorithm using a simple ML abstraction [11]. To enable support for more programming languages in an efficient way, Wayang will soon come with a Polyglot library (see Section 4.3).

Wayang relies on *data quanta*, the smallest processing units of the input datasets. A data quantum can express a large spectrum of data formats, such as database tuples, edges in a graph, or data points required by machine learning. Wayang's main building block is a *Wayang plan*: a directed dataflow graph whose vertices are platform-agnostic operators and whose edges represent data flowing among the operators. An example Wayang plan for the Wordcount task is depicted in Figure 3(a). A user or application can specify a Wayang plan by using any of the three supported languages (Java, Scala, and Python). Importantly, one does not have to specify the platform on which each plan's operator will be executed. Given a Wayang plan, the optimizer is responsible for determining the platform on which each operator has to be executed, thereby composing a platform-specific execution plan. The executor is then responsible for assigning the operators of the execution plan to the corresponding drivers and coordinating the execution. The monitor checks whether the estimations used during the optimization are correct, and if not, requests a new execution plan from the optimizer.

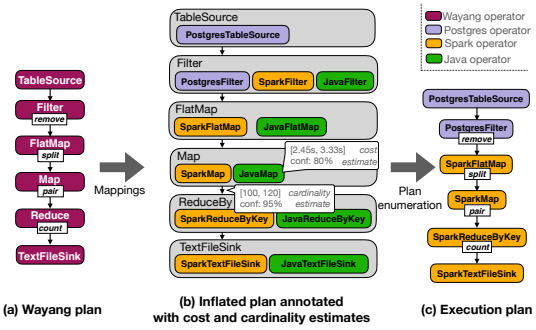


Figure 3: Wayang's optimization phase by example.

3. THE CORE OF WAYANG

Wayang's core comprises a query optimizer and an executor. The former determines which underlying platform each (sub)query has to be executed on to achieve the best performance. The latter is responsible for scheduling and submitting the (sub)queries for execution. The main building block of Wayang is a *Wayang plan*, a graph where each node is a Wayang (platform-agnostic) operator of the following types: unary, binary, loop, source, or sink. In addition, a Wayang operator can be either atomic (e.g., `map`) or composite (e.g., `pagerank`), i.e., composed of many atomic operators.

Query Optimization. The optimizer receives as input a Wayang plan and outputs an execution (platform-specific) plan with the goal of minimizing the total execution cost. To achieve this, the optimizer first "inflates" the plan (Figure 3): For each node that corresponds to a Wayang operator, it adds all the corresponding execution operators. This is done via *flexible graph-based mappings* that map one or more Wayang operator(s) to one or more platform-specific execution operator(s). A composite Wayang operator, such as `pagerank`, is mapped either to a platform-specific composite operator (e.g., MLlib's pagerank) or to a graph of platform-specific operators that perform the required functionality.

Once the inflated plan is created, the optimizer attaches not only the operator's costs but also the costs for moving intermediate data from one platform to another (omitted in the figure for simplicity reasons). Currently, Wayang uses linear cost formulas to estimate these costs. The system administrator needs to fine-tune the coefficients of these formulas. Although Wayang comes with profiling tools to facilitate this tuning effort, our near plans include the ability to easily port machine learning models for estimating the costs, as discussed in [12].

At the last step of query optimization, an enumeration algorithm considers available options to output the optimal execution plan w.r.t. a defined cost. The metric for the optimization cost can be anything, from runtime to monetary cost or energy consumption. As the search space is exponential (a plan with n Wayang operators, each having k execution operators, leads to k^n possible execution plans), pruning is crucial. Wayang's enumeration algorithm is based on an algebra consisting of two main operations: *Join* for concatenating subplans and *Prune* for pruning subplans that lead to inferior execution plans. Importantly, the pruning strategy is lossless. It is based on the notion of boundary operators which are the start and end operators of a subplan and is guaranteed to not prune a subplan that is part of the optimal execution plan.

Note that users can control the optimizer by specifying in their code where an operator has to be executed via the `withTargetPlatform(plat)` call on the desired operator. Then, the optimizer takes into consideration the decisions of the user and outputs an execution plan by navigating a reduced search space during the plan enumeration.

Data movement. We now detail how Wayang computes the costs incurred when moving data from one platform to another during the query optimization process. As there may be multiple ways to move data from platform A to platform B, Wayang represents the space of different communication ways as a channel conversion graph. This graph contains the different data types (channels) as vertices (e.g., RDD or Relation). Two channels are connected with a direct edge denoting that the source channel can be converted to the destination channel via one or more conversion operators. Conversion operators can be the standard source and sink operators of the underlying platforms. During query optimization, Wayang finds the optimal communication path from one channel to another by formulating the problem as a minimum conversion tree problem (proved to be NP-hard [14]). The interested reader is referred to [13] for more details.

Execution. Given an execution plan output by the query optimizer, the executor of Wayang is responsible for scheduling its execution. First, it divides the plan into stages so that each stage forms a subplan where all its execution operators are of the same platform. Stages are connected by data flow dependencies. The executor dispatches a stage to the relevant platform driver, which in turn submits the sequence of operators as a job to the underlying platform. If there are stages with no dependencies, the executor dispatches them in parallel. In any other case, it dispatches a stage once its input dependencies are satisfied. After each stage, the plat-

form gives back the control to the executor so that it either initiates the next stage or it materializes the output in the case of the final stage. The executor may also create more than one stage for a sequence of execution operators of the same platform in cases where the executor needs the control (e.g., when the executor needs to evaluate the loop condition in an iterative operator).

Re-optimization. It is well-known that poor cardinality and cost estimates can negatively impact the effectiveness of an optimizer. This is even worse in our setting where the semantics of UDFs and data distributions are usually unknown because of the little control over the underlying platforms. For this reason, Wayang's optimizer allows for re-optimizing a plan whenever observed cardinalities greatly mismatch the estimated ones. Similar to [15], it achieves this by adding optimization checkpoints between stages in the execution plan whenever the cardinality estimates have low confidence or the data is at rest (e.g., file). When the executor encounters a checkpoint between stages, it pauses the plan execution and gives the control to the optimizer to consider a re-optimization of the plan beyond the checkpoint. The optimizer uses the observed cardinalities and recomputes the most efficient plan since the last optimization checkpoint. It then gives the new execution plan to the executor, which resumes execution considering the new plan.

4. ANY ANALYTICS ANYWHERE

We now describe the libraries that Wayang currently supports (SQL and ML) and one that is under development (Polyglot). These libraries are built atop the native Java API of Wayang. Note that Wayang also provides a Scala API and a Python API is currently under development. In the following, we first describe its SQL and ML libraries. We, then, detail the Polyglot library, which is the Wayang's approach to support UDFs coded in different programming languages.

4.1 SQL analytics anywhere

A feature of Wayang is its unified SQL interface for cross-platform data processing. The SQL library allows users to embed SQL queries in their applications via the `SqlContext` object, which holds the configurations about different data sources. The following snippet shows how users can specify SQL queries using its Java API.

```
SqlContext sqlContext = new SqlContext(conf);
Collection<Record> result = sqlContext.
        executeSql("SELECT ... FROM ...");
```

The `sqlContext` provides methods that return

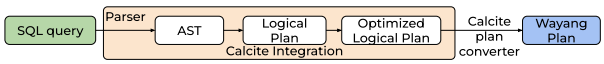


Figure 4: SQL query preparation in Wayang.

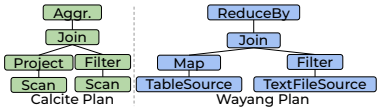


Figure 5: Example Calcite plan converted to a Wayang plan.

the result of the `SELECT` statement as a collection of Records, which can be converted to a data quanta or used in subsequent SQL queries. This allows Wayang to seamlessly integrate SQL queries into applications and holistically optimize them.

Wayang's SQL library utilizes Apache Calcite [4] to support the SQL standard. Yet, to execute an SQL query, we first have to translate it into a Wayang plan, as shown in Figure 4.

Wayang comes with a Calcite-based SQL parser and optimizer. The SQL query is first translated into a Calcite logical plan from its AST, which is then optimized and subsequently converted into a Wayang plan. Our Calcite integration facilitates database optimizations, such as operator pushdown, reordering, and elimination.

Converting a Calcite plan into a Wayang plan is done on a per-operator basis. Figure 5 shows an example translation. Common translations include: `tableScan` operators to Wayang source operators; `project`, `filter`, `join`, and `aggregation` operators to Wayang's `Map`, `Filter`, `Join`, and `ReduceBy` operators, respectively. During the plan conversion step, a SQL operator is translated into Java functions, which are then wrapped by a single UDF.

Wayang currently offers support for developing applications with the SQL interface in Java. Our future work will include bindings for Scala as well as for Python. Moreover, the SQL language support in Wayang only includes support for `SELECT` statements as the core focus of the Apache Wayang project is to enable cross-platform data analytics (rather than data management). In future, we also plan to include a JDBC client in Wayang. This will enable Wayang's compatibility with other external BI tools, such as Tableau.

4.2 Machine learning anywhere

Wayang also comes with a machine learning (ML) library, which allows users to create ML pipelines using Wayang's operators and/or write their ML algorithms using a predefined small set of primitives [11]. This set of primitives is sufficient for implementing a wide variety of iterative ML algorithms, such as any gradient descent algorithm,

k-means clustering, or expectation-maximization algorithms. After analyzing such ML algorithms, we found that they all can be split into three different phases: preparation, processing, and convergence. Wayang's ML library, thus, provides the following seven operators:

(1) `Transform` receives a data point to transform (e.g., normalize it) and outputs a new data point.

(2) `Stage` initializes all the required global parameters (e.g., centroids for the k-means algorithm).

(3) `Compute` performs user-defined computations on the input data point and returns a new data point. For example, it can compute the nearest centroid for each input data point.

(4) `Update` updates the global parameters based on a user-defined formula. For example, it can update the new centroids based on the output computed by the `Compute` operator.

(5) `Sample` takes as input the size of the desired sample and the data points to sample from and returns a reduced set of sampled data points.

(6) `Converge` specifies a function that outputs a convergence dataset required for determining whether the iterations should continue or stop.

(7) `Loop` specifies the stopping condition on the convergence dataset.

All the above operators serve as UDFs. While we provide reference implementations for common algorithms, users can easily customize or override them. The first two operators are used in the preparation phase, while `Compute`, `Update`, and `Sample` are used iteratively in the processing phase. The interested reader can find more details in [11].

Once these operators are defined, the ML library transforms them into a Wayang plan. The plan is then passed to the optimizer to determine the right platform. Thus, data scientists can use Wayang to develop new algorithms and test them with small datasets, which will be run locally. The same code can seamlessly be used on deployment for larger datasets potentially running in a big data platform. However, the data scientist does not have to worry about the underlying deployment of a plan.

4.3 UDFs coded anywhere

Besides its Java, Scala, and Python APIs, which are dedicated interfaces for programming languages, Wayang provides a library to support UDFs coded in different programming languages, such as Go. Continuing with its search for platform interoperability, Wayang goes one step forward when it comes to supporting UDFs. It offers the Polyglot library to support the execution of UDFs in any programming language and in an efficient way. The execution of UDFs in other programming languages is crucial for exposing Wayang to other programming languages besides Java, Scala, and Python.

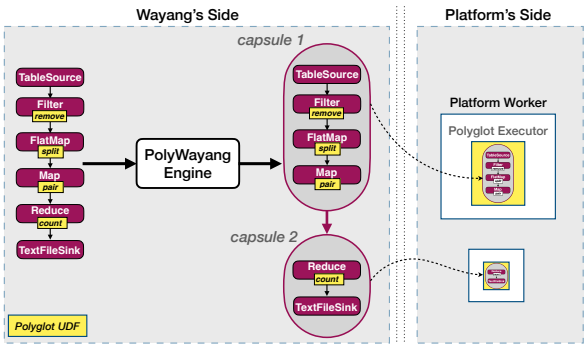


Figure 6: Execution flow with Polyglot.

The execution flow with Polyglot is shown in Figure 6. To support UDFs in a particular programming language, Wayang developers must implement a `FunctionWrapper` and a `WorkerManager`. While the former allows Wayang users[3] to provide their UDFs in any supported programming language (e.g., Go), the latter allows Wayang to launch the required runtime (e.g., the Go runtime). As a result, Wayang can invoke the programming language runtime (e.g., the Go runtime via a `GoWorkerManager` implementation) with the UDF (e.g., the Go UDF via a `GoFunctionWrapper`). For performance reasons, Wayang does so by *encapsulating* operators that belong to the same stage (i.e., pipelined operators) into a `MapPartition` operator. This results in a single call to the programming language runtime for the entire data quanta of the `MapPartition` instead of having a runtime call per input data quantum.

5. ADOPTION

Wayang is increasingly gaining traction in both industry and academia. In academia, Wayang has fostered several database research projects in query optimization, data integration, and polyglot data management. In industrial settings where multi-platform infrastructures are routine, Wayang has provided a cost-effective alternative to run complex analytics without having to develop platform-specific solutions. Wayang is being used, among others, in machine learning, data cleaning, and data analytics applications. For instance, an airline company is assessing Wayang to carry out large-scale data analysis for optimizing air cargo revenues [16].

[3]We distinguish between developers and users in the way they interact with Wayang: while the developers write code to extend or fine-tune Wayang, the users only use it via its libraries and APIs.

NADEEF [5], a commodity data cleaning system, uses Wayang to boost its performance through platform independence. Wayang has also led to the creation of Databloom [2], an AI startup that aims at providing an easy-to-use, cost-efficient, and data-compliant solution to companies having distributed and heterogeneous data infrastructures.

6. RELATED WORK

Open source. To our knowledge, Wayang is the only open source system that not only decouples applications from the underlying platforms but also provides a way for automatically determining on which platform(s) a given task should be executed. Perhaps, most related to Wayang is Apache Beam [1], which focuses on providing a unified model for batch and streaming data processing and being portable to any data processing platforms. The latter means that a user's pipeline is entirely executed on one data platform (runner), which users need to specify. In contrast, Wayang users are free to either not specify at all where their pipelines are executed or create hybrid pipelines integrating multiple data processing platforms.

Academic. There have been early efforts to unify data analytics in a systematic way [9, 7]. However, [7] requires expertise from users for deciding when to use a data processing platform and the design of [9] is not flexible enough to allow for continuous extensions with new platforms, i.e., developers have to modify the source code. IReS [6] on the other hand, provides a flexible and automatic way to choose data processing platforms. However, in contrast to Wayang, it focuses more on coarse-grained operators (e.g., `k-means` instead of `filter`, `map`) and provides 1-to-1 mappings from abstract to execution operators, which may lead to suboptimal execution plans. For instance, for a simple stochastic gradient descent algorithm that could be viewed as an operator by itself, Wayang can provide significant performance benefits by splitting it into more fine-grained operators [3].

7. TOWARDS WAYANG 2.0

Apache Wayang (Incubating) facilitates automatic cross-platform data processing. Its extensible framework integrates various data processing platforms, decoupling applications from specific platforms. Wayang provides a unified framework for analytics and aims to support fully decentralized applications through a *delegation phase* and *direct communication channels*.

Task Delegation. The goal is to decentralize execution by offloading processing and communication tasks to underlying platforms, avoiding a centralized Executor. This involves introducing delegation tasks (akin to [8]) that combine data manipulation and movement instructions. Task delegation will reduce the need for resource-intensive execution coordination and platform communication.

Direct Communication Channels. To enable task delegation, we will prioritize direct communication between platforms and develop new abstractions within the Wayang operator model. These abstractions will enhance platform interoperability and eliminate the need for generic conversion channels where communication operators require intermediate mediums (e.g., CSV files or Java collections). Wayang can then also optimize data movement on a platform level by implementing techniques such as data layouts and compression.

8. REFERENCES

[1] The Unified Apache Beam Model. https://beam.apache.org. Retrieved May, 2023.
[2] Databloom ai: https://www.databloom.ai/, 2022.
[3] D. Agrawal et al. Rheem: enabling cross-platform data processing – may the big data be with you! In *PVLDB*, 2018.
[4] E. Begoli, J. Camacho-Rodríguez, J. Hyde, M. J. Mior, and D. Lemire. Apache calcite: A foundational framework for optimized query processing over heterogeneous data sources. In *SIGMOD*, page 221–230, 2018.
[5] M. Dallachiesa, A. Ebaid, A. Eldawy, A. K. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. NADEEF: a commodity data cleaning system. In *SIGMOD*, pages 541–552. ACM, 2013.
[6] K. Doka, I. Mytilinis, N. Papailiou, V. Giannakouris, D. Tsoumakos, and N. Koziris. Multi-engine analytics with ires. In *BIRTE*, pages 133–154, 2016.
[7] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. Zdonik. The BigDAWG polystore system. *SIGMOD Record*, 44(2):11–16, 2015.
[8] H. Gavriilidis, K. Beedkar, J.-A. Quiané-Ruiz, and V. Markl. In-situ cross-database query processing. In *ICDE*, 2023.
[9] I. Gog, M. Schwarzkopf, N. Crooks, M. P. Grosvenor, A. Clement, and S. Hand. Musketeer: all for one, one for all in data processing systems. In *EuroSys*, 2015.
[10] Z. Kaoudi and J. Quiané-Ruiz. Unified data analytics: State-of-the-art and open problems. *Proc. VLDB Endow.*, 15(12):3778–3781, 2022.
[11] Z. Kaoudi, J.-A. Quiané-Ruiz, S. Thirumuruganathan, S. Chawla, and D. Agrawal. A cost-based optimizer for gradient descent optimization. In *SIGMOD*, 2017.
[12] Z. Kaoudi, J.-A. Quiané-Ruiz, B. Contreras-Rojas, R. Pardo-Meza, A. Troudi, and S. Chawla. ML-based Cross-Platform Query Optimization. In *ICDE*, pages 1489–1500, 2020.
[13] S. Kruse, Z. Kaoudi, B. Contreras-Rojas, S. Chawla, F. Naumann, and J. Quiané-Ruiz. RHEEMix in the data jungle: a cost-based optimizer for cross-platform systems. *VLDB J.*, 29(6):1287–1310, 2020.
[14] S. Kruse, Z. Kaoudi, J.-A. Quiané-Ruiz, S. Chawla, F. Naumann, and B. Contreras-Rojas. Optimizing Cross-platform Data Movement. In *ICDE*, 2019.
[15] V. Markl, V. Raman, D. E. Simmen, G. M. Lohman, and H. Pirahesh. Robust query processing through progressive optimization. In *SIGMOD*, pages 659–670, 2004.
[16] S. G. Rizzo, Y. Chen, L. Pang, J. Lucas, Z. Kaoudi, J. Quiané-Ruiz, and S. Chawla. Prescriptive learning for air-cargo revenue management. In *ICDM*, pages 462–471, 2020.

Reminiscences on Influential Papers

This issue's contributors chose papers that address challenges at the heart of database systems: physical design tuning for index selection and transaction isolation levels. Both contributions emphasize the elegant, modular, and long-lasting design choices of the respective work. Enjoy reading!

While I will keep inviting members of the data management community, and neighboring communities, to contribute to this column, I also welcome unsolicited contributions. Please contact me if you are interested.

Pınar Tözün, *editor*
IT University of Copenhagen, Denmark
pito@itu.dk

Renata Borovica-Gajic
University of Melbourne, Australia
renata.borovica@unimelb.edu.au

Surajit Chaudhuri and Vivek Narasayya.
An Efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server.
In Proceedings of the International Conference on Very Large Data Bases (VLDB), pages 146–155, 1997.

Pondering on which paper impacted you most is hard - so when Pınar invited me to contribute to this column, I had a hard time picking it. Yet, I immediately knew which thread of papers had the greatest impact on my work. Thus, as is only natural, I chose the first paper from this thread that started an avalanche of amazing work in physical design tuning of databases through the AutoAdmin project.

The AutoAdmin project started in mid-1996 at Microsoft, with the goal of developing technology that makes database systems more self-tuning and self-managing. The task was a tall order at the time, when the database administrators were deeply involved in performance tuning of databases, and when very little automation had seen the light in production systems. Yet, the problem of choosing the right physical design was critical, since together with the execution engine and the optimizer, the physical database design determines the efficiency of executed queries, and hence ultimately it affects the overall user experience when using a database. The VLDB 1997 paper was the first paper of this line of work, and it looked at recommending indexes (a.k.a. index selection) as one (important) aspect of physical database design.

The index selection problem had been studied since the early 70's and the importance of the problem was well recognized at the time. The goal of automated index selection is to automatically pick a set of indexes, referred to as a configuration, estimated to (maximally) boost the performance of the given workload, often under memory budget constraints. While the index selection problem was proven to be NP-complete (through the work of Shapiro in 1983), the VLDB 1997 paper presented a practical, efficient, and elegant solution to this rather challenging problem. And this is what struck me the most about this paper. The authors had taken an extremely challenging problem, broke it down into small pieces, and proposed a set of elegant techniques to address all the challenges efficiently and effectively, while keeping the entire architecture fully modularized. This end-to-end systems approach not only resulted in an extremely clean design, but also opened doors for future improvements, since any component could easily be replaced by another, more efficient algorithm in the future. Thanks to this modularized design, over a dozen of papers had appeared in the following decade, many leveraging the proposed architecture and adding new dimensions to the problem (e.g., considering materialized views, partitioning, statis-

tics, etc).

So, let's dive into the architecture. The index selection tool proposed in this paper consists of candidate index selection module, configuration enumeration module, "what-if" index creation module, cost evaluation module and multi-column index generation module.

The index selection module proposes a set of candidate indexes for the given workload. Since the number of possible index candidates is too large (exponential in the number of columns and tables), the authors proposed a heuristic of determining the best configuration for each query independently, and only consider indexes belonging to one or more of such best configurations as a candidate index set. This neat trick allowed them to use their own tool to generate candidate indexes, since each individual query could be considered as a workload consisting of that single query - which is yet another example of the elegance of the approach.

Next, out of N candidate indexes chosen as the overall candidate set, the goal of configuration enumeration modules is to pick the best K. Exhaustively enumerating all possible combinations is again too large, and the authors employed a greedy incremental approach where in each round the algorithm selects an optimal configuration of the size M, where M <= K, greedily adding the next most beneficial index to the existing configuration until M == K or until no further cost reduction is possible.

Probably the most influential component is the "what-if" API in the query optimizer. The "what-if" component stood the test of time and remained a critical component of many subsequent tools as well as served as an independent component for manual performance tuning of databases employed by database administrators. The "what if" API simulates the presence of different physical design structures without materializing them. When the index selection tool needs to evaluate the cost of a workload, it simulates the presence of the configuration by loading the catalog tables of a database system with metadata and statistical information about defined structures. It then optimizes the queries from the workload in a no-execute mode, in which the optimizer returns a plan and a cost estimate for each query without executing them. Using the query optimizer's cost estimates as the basis for the physical design tool has several advantages. First, it can guarantee that any proposed index, if materialized, will actually be used by the optimizer. Second, it is much more efficient than paying the cost to materialize candidate indexes. Finally, as the query optimizer's cost model evolves over time the tool will just benefit from those improvements, which is yet another forward-looking aspect of this paper.

Still, calling the optimizer to cost all possible configurations across the entire workload may be too expensive. To reduce the number of optimizer calls, the authors introduce a concept of atomic configurations and show that it is sufficient only to evaluate all atomic configurations across the workload, as the cost of all other configurations could be derived from atomic configurations without requiring any additional optimizer calls. On top of reducing the number of atomic configurations to cost, the authors propose a way of reducing the cost of evaluating atomic configurations by costing only a subset of relevant indexes from the configuration whose columns are part of the query set. By caching the results of the optimizer calls for atomic configurations, optimizer invocations for other configurations for the same query could be eliminated (often even by orders of magnitude).

Finally, the authors proposed a search algorithm to incrementally examine the space of multi-column indexes. The approach the authors employed is to iteratively expand the space of multi-column indexes by choosing only the winners of one iteration and augmenting the search space of the next iteration by expanding such winners with an additional column. This heuristic allows for a structured and tractable exploration of what would be an enormous space of alternative choices.

The impact of this paper was manyfold. It was the first approach that looked at creating an automated tool for index selection. This work formed the basis of the Index Tuning Wizard (ITW) that shipped in Microsoft SQL Server 7.0, and many commercial vendors have followed suit, using somewhat similar techniques. The paper rightfully received the 10-Year Best Paper Award at VLDB 2007 due to its novelty, clean architecture but also the broad impact it had on the research community and database vendors at large. As the work progressed, so did the product, resulting in a fully-fledged Database Tuning Advisor (DTA) that shipped as part of Microsoft SQL Server 2005. DTA went beyond index selection and supported selection of materialized views, and horizontal partitioning. Today's SQL Server DTA also supports selection of partial indexes and columnstore indexes.

On a personal note, I first discovered this paper more than a decade ago when embarking on a PhD journey, and immediately appreciated its elegant architecture, and pragmatic approach to solving challenging problems. This pragmatic approach stayed with me throughout my professional life, for which I

will be forever grateful to the authors. Finally, while we as a research community naturally evolve and sometimes outgrow research problems, the problem of automated physical design tuning is more important than ever, considering the strong presence of cloud platforms where the workload complexity and the absence of on-premise database administrators make such tools a necessity. When we will be able to completely solve this problem is hard to say, since as authors conclude in their 10-year paper summary (published in VLDB 2007) "it will probably be impossible to make database systems self-tuning by a single architectural or algorithmic breakthrough" and that "demand for self-manageability could lead to development of newer structured store that is built grounds-up with self-manageability as a critical requirement". Almost two decades later, we are seeing first strides towards making database systems self-driving from the ground up. And I am sure that the next two decades will be as exciting, with fully adaptive and self-driving systems becoming common.

Bailu Ding
Microsoft Research Redmond, USA
bailu.ding@microsoft.com

Atul Adya, Barbara Liskov, and Patrick E. O'Neil. ***Generalized Isolation Level Definitions.*** In Proceedings of the 16th International Conference on Data Engineering (ICDE), pages 67-78, 2000.

When I first started working on transaction processing, one of the first papers I read was "Generalized Isolation Level Definitions" by Adya, Liskov, and O'Neil. This paper argues that isolation levels should be a logical property of transactions, rather than being defined based on how transactions are implemented in a locking-based concurrency control scheme, as was the case in earlier ANSI-SQL 92 standards. The authors propose a new way to define transaction isolation levels based on the dependencies of transactions, which decouples the abstraction of isolation levels from their implementation. This definition can be applied to different concurrency control schemes, such as optimistic concurrency control or multi-version concurrency control. I was impressed by this work for its elegance and practicality. The technique of analyzing isolation levels based on transaction dependency graphs has also become a crucial tool used in my later work on relaxed concurrency control for transaction processing, such as in watermarking [1] and transaction reordering [2].

[1] Bailu Ding, Lucja Kot, Alan Demers, and Johannes Gehrke."Centiman: Elastic, High Performance Pptimistic Concurrency Control by Watermarking." In Proceedings of the Sixth ACM Symposium on Cloud Computing, pages 262-275, 2015.

[2] Bailu Ding, Lucja Kot, and Johannes Gehrke. "Improving Optimistic Concurrency Control through Transaction Batching and Operation Reordering." In Proceedings of the VLDB Endowment 12.2, pages 169-182, 2018.

Kùzu: A Database Management System For "Beyond Relational" Workloads

Semih Salihoğlu
University of Waterloo
semih.salihoglu@uwaterloo.ca

I would like to share my opinions on the following question: *how should a modern graph DBMS (GDBMS) be architected?* This is the motivating research question we are addressing in the Kùzu project at University of Waterloo [4, 5].[1] I will argue that a modern GDBMS should optimize for a set of what I will call, for lack of a better term, "beyond relational" workloads. As a background, let me start with a brief overview of GDBMSs.

Overview: Modern GDBMSs [7, 8, 11, 15] adopt the property graph data model, where applications model their data as a set of node and relationship records and query these records using SQL-like high-level languages that have specialized graph syntax, such as the arrow syntax to describe the joins between node records. As other DBMSs with high-level query languages, GDBMSs are relational at their cores as these high-level constructs compile to relational operators, such as joins, filters, and projections. Yet, GDBMSs support workloads that require a set of features that are not traditionally optimized in RDBMSs. Here are some examples of such "beyond relational" capabilities:

- Complex many-to-many (m-n) joins: Datasets that are modeled as graphs often contain many-to-many relationships across nodes, e.g., friendships in social networks. Many applications search for patterns in these datasets, which translate to complex m-n joins.
- Recursive joins: Many queries of "graph workloads" can be recursive, e.g., to find indirect connections between accounts. While recursion is an afterthought in SQL, it is a first-class citizen feature in GDBMSs.
- Schema-flexible queries: Some applications require answering questions that require flexibility in the type of records to process, such as finding "any type of connections between accounts u and v". GDBMSs have elegant means to ask these queries, while in SQL such queries are asked by unioning many queries.
- Heterogeneous datasets: Some datasets, such as knowledge graphs like Wikidata [1], model very complex domains, which: (i) cannot be tabulated; and (ii) are best modeled as URI-heavy RDF triples.

[1] I have previously written about the vision of Kùzu in a longer blog post without space constraints here [16].

Techniques For Beyond Relational Workloads: Our work in Kùzu has so far focused on integrating state-of-the-art techniques to evaluate complex many-to-many joins efficiently [5, 6, 9]. For example, Kùzu has a factorized [3, 13] query processor, which compresses intermediate results of many-to-many joins and implements novel worst-case optimal join algorithms [12, 17]. We are also implementing common recursive joins, such as shortest path and variable-length joins, and plan to implement a URI data type. However, a lot remains undone, such as integrating automata-based techniques for regular path queries and advanced string compression to manage URIs to achieve our vision of a feature-rich, competent GDBMS for beyond relational workloads. We are actively developing Kùzu to achieve this goal.

An Ideal Worth Remembering: With the hope of inspiring some PhD students, let me bring up another beyond relational capability that may at first seem unrelated to GDBMSs: *the ideal of systems with general deductive capabilities*. Consider a database of 3 items and their colors, 1 red, 1 blue, and 1 with a NULL color, and a constraint that every item must be red or blue. Suppose we ask: "what is the maximum of the count of red items and blue items?" With relational capabilities of joins and group by-aggregates, one would compute the answer as 1, yet with a simple deduction, we can see that the answer is 2, as the unknown value is either red or blue. The ideal of information systems that can perform advanced logical deductions, for example proofs by contradictions or if-then implications, has existed since the birth of CS [2], with deep connections to AI. Because logical deductions are often recursive and recursion is a first-class citizen in GDBMSs, GDBMSs can integrate deductive capabilities. In fact, some RDF systems [10], which are graph-based DBMSs, perform limited OWL-based deductions [14]. With the current momentum around symbolic AI, it is a good time for our community to revisit these ideals, and maybe some research groups can attempt to develop systems with such capabilities and without forgetting our relentless focus on performance and scalability!

REFERENCES

[1] Wikidata. https://www.wikidata.org/wiki/Wikidata:Main_Page, 2023.

[2] Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, 2004.

[3] Factorization & Great Ideas from Database Theory. https://kuzudb.com/blog/factorization.html, 2023.

[4] Xiyang Feng, Guodong Jin, Ziyi Chen, Chang Liu, and Semih Salihoğlu. Kùzu Source Code. https://github.com/kuzudb/kuzu, November 2022.

[5] Xiyang Feng, Guodong Jin, Ziyi Chen, Chang Liu, and Semih Salihoğlu. Kùzu Graph Database Management System. In *The Conference on Innovative Data Systems Research*, 2023.

[6] Pranjal Gupta, Amine Mhedhbi, and Semih Salihoglu. Columnar Storage and List-based Processing for Graph Database Management Systems. *Proceedings of the VLDB Endowment*, 14(11):2491–2504, 2021.

[7] Chathura Kankanamge, Siddhartha Sahu, Amine Mhedbhi, Jeremy Chen, and Semih Salihoglu. Graphflow: An Active Graph Database. In *Proceedings of the ACM International Conference on Management of Data*, 2017.

[8] Memgraph. https://memgraph.com/, 2023.

[9] Amine Mhedhbi, Chathura Kankanamge, and Semih Salihoglu. Optimizing One-time and Continuous Subgraph Queries using Worst-Case Optimal Joins. *ACM Transactions on Database Systems*, 46(2):1–45, 2021.

[10] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. RDFox: A Highly-Scalable RDF Store. In Marcelo Arenas, Oscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab, editors, *The Semantic Web*, 2015.

[11] Neo4j. http://neo4j.com, 2023.

[12] H. Ngo, C. Ré, and A. Rudra. Skew Strikes Back: New Developments in the Theory of Join Algorithms. *SIGMOD Record*, 42(4):5–16, 2014.

[13] Dan Olteanu and Jakub Závodnỳ. Size Bounds For Factorised Representations of Query Results. *ACM Transactions on Database Systems*, 40(1):1–44, 2015.

[14] OWL Semantic Web Standard. https://www.w3.org/OWL/, 2022.

[15] Tigergraph. http://tigergraph.com, 2023.

[16] What Every Competent GDBMS Should Do. https://kuzudb.com/blog/what-every-gdbms-should-do-and-vision.html, 2023.

[17] Why (Graph) DBMSs Need New Join Algorithms: The Story of Worst-case Optimal Join Algorithms. https://kuzudb.com/blog/wcoj.html, 2023.

Proactive Resource Allocation Policy for Microsoft Azure Cognitive Search

Olga Poppe, Pablo Castro, Willis Lang, and Jyoti Leeka
olpoppe|pablocas|wilang|jyleeka@microsoft.com

ABSTRACT

Modern cloud services aim to find the middle ground between quality of service and operational cost efficiency by allocating resources if and only if these resources are needed by the customers. Unfortunately, most industrial demand-driven resource allocation approaches are reactive. Given that scaling mechanisms are not instantaneous, the reactive policy may introduce delays to latency-sensitive customer workloads and waste operational costs for cloud service providers. To solve this catch-22, we define the proactive resource allocation policy for Microsoft Azure Cognitive Search. In addition to the current resource demand, the proactive policy takes the typical resource usage patterns into account. We gained the following valuable insights from these patterns over several months of production workloads. One, 87% of the workload is stable due to continuous resource demand. Two, 90% of varying demand is predictable based on a few weeks of historical traces. Three, resources can be reclaimed 52% of the time due to extensive idle intervals of varying workload. Given the size and scope of our analysis, we believe that our approach applies to any latency-sensitive cloud service.

1. INTRODUCTION

We are currently witnessing the growing popularity of demand-driven resource allocation among all cloud service providers, including Microsoft Azure [3], Amazon Web Servoces [7], and Google Cloud Platform [4]. As a result, the customers do not have to provision a fixed amount of resources up front. Instead, they send workloads to the systems that dynamically and transparently manage resources to handle the changing demand over time. These systems deploy low-latency resource allocation mechanisms. While the workload is running, resources are allocated. When the workload stops, resources are reclaimed and possibly reused to serve current workloads. In this way, demand-driven resource allocation reduces the amount of maintained resources and thus saves the operational costs.

The natural progression for demand-driven resource allocation is proactive decision making. In addition to the current demand, proactive decisions take the predicted future demand into account to allocate resources ahead of demand. The proactive policy improves the quality of service by reducing delays due to the reaction time between demand signal and effective change in resource availability [23, 25, 27, 30]. Furthermore, the proactive policy allows to re-target resources during predicted long idle intervals to serve the current workloads.

We define the proactive policy for Microsoft Azure Cognitive Search, which is a PaaS solution that allows to build sophisticated search capabilities within customer applications on customer data [1, 2]. It is a rapidly growing cloud service which currently runs tens of billions of queries per month. It offers all the functionality needed to create rich search scenarios such as automatic content ingestion, fast full-text search, auto-complete, customizable scoring, and a natural language understanding stack that ensures high relevance of search results.

Challenges. Defining the proactive policy for any latency-sensitive cloud service is not trivial.

(1) *High latency sensitivity.* Search is highly sensitive to execution latency. Currently, low latency is achieved through a combination of index data structure design, a favorable index size-memory ratio, and by maintaining warm compute and cache to execute search queries as soon as they arrive. However, current solution results in low efficiency of compute utilization when search requests are not continuous. We aim to move the efficiency needle without sacrificing the low latency requirement.

(2) *Benefit versus overhead of proactive policy.* Proactive resource allocation is not always applicable. For example, if the demand is stable or idle time is too fragmented for effective reuse of resources, then the resources must be provisioned to guarantee high quality of service. A proactive policy will

do more harm than good in such cases due to its overhead of load prediction. We aim to identify the prerequisites of the proactive policy to exclude part of the workload from further consideration.

(3) *Wide range of applicable techniques.* There are many approaches to load prediction for proactive resource allocation. They range from simple statistics to complex machine learning models. Each of them has multiple tunable parameters. Each of them has advantages with respect to prediction accuracy, execution latency, and supportability long term in production worldwide. The search space is too large to be explored exhaustively. We aim to compare several techniques and propose an effective approach to proactive resource alloction.

State-of-the-Art. Several existing approaches implement reactive demand-driven resource allocation [10, 11, 12]. They might cause delays in resource availability after long idle intervals during which resources are reclaimed. Thus, the reactive policy is not suitable for latency-sensitive cloud services. To optimize quality of service, we apply proactive resource allocation policy. While academic approaches leverage complex and computationally expensive machine learning models to predict the load [8, 13, 14, 15, 20, 26, 29], industrial approaches deploy light-weight yet accurate forecast techniques to production [17, 19, 21, 22, 23, 25, 27, 30]. Unfortunately, the existing approaches fail to identify cases when the proactive policy is not applicable. Hence, they introduce significant computational overhead of load prediction to latency-sensitive cloud services. To avoid this overhead, our approach identifies the prerequisites of the proactive policy and focuses on cases when the proactive policy saves operational costs without sacrificing high quality of service.

Proactive Resource Allocation Policy. We analyzed several months of production telemetry for Microsoft Azure Cognitive Search and concluded that no single resource allocation policy is effective for all workloads. Instead, the policy must be tailored for each type of workload. Therefore, we first characterize the search requests along various dimensions, including stability, predictability, and fragmentation of idle time. Afterwards, we identify the prerequisites of proactive resource allocation. In particular, resources are proactively allocated for varying predictable workloads to guarantee high quality of service. In addition, resources are reclaimed and reused during extensive idle intervals to save operational costs. Furthermore, resources are provisioned for stable workloads to avoid the overhead of load prediction. Lastly, resources are allocated reactively for unpredictable workloads.

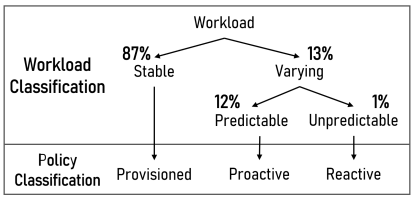


Figure 1: Workload and policy classification

Contributions. Our key contributions include:

(1) We classify the workload of search requests into stable and varying, predictable and unpredictable (Figure 1). 87% of the workload is stable. 12% is varying and predictable. Only 1% is varying and unpredictable. We study the fragmentation of idle time. 63% of the total idle time is composed of idle intervals that exceed two hours.

(2) We define the spectrum of resource allocation policies, including provisioned, reactive, proactive, and optimal. We review the range of demand prediction techniques from simple probabilistic algorithms to advanced time series forecast models.

(3) We identify five prerequisites of the proactive policy and apply it to predictable varying workload with extensive idle intervals. In this way, we exclude 87% of the workload and thus reduce the execution latency of the proactive policy by 2X compared to the proactive policy applied to the entire workload.

(4) We define the KPI metrics that measure the effectiveness of the resource allocation policies in addressing the business needs for various workloads. In particular, we measure demand predictability, quality of service, correctness of resource allocation, operational cost efficiency, and execution latency.

(5) We experimentally compare the resource allocation policies. In contrast to the reactive policy, the proactive policy correctly predicts 90% of varying search requests to guarantee high quality of service. In contrast to the provisioned policy, the proactive policy reclaims resources 52% of the time for varying workload to save operational costs.

Outline. We define the optimization objective in Section 2. We identify the prerequisites of the proactive policy in Section 3. We define the resource allocation policies in Section 4 and the accuracy metrics in Section 5. We present the experiments in Section 6. We review the related work in Section 7 and conclude the paper in Section 8.

2. OPTIMIZATION OBJECTIVE

Index. To speed up search requests, the search engine builds indexes and allocates resources per index. Let $\mathbb{I}$ be the set of indexes.

Time is represented by a linearly ordered set of

time points $(\mathbb{T}, \leq)$ where $\mathbb{T} \subseteq \mathbb{R}^+$ are the non-negative real numbers.

Resource Demand $D : \mathbb{I} \times \mathbb{T} \rightarrow \{0, 1\}$ is a function that maps an index $i \in \mathbb{I}$ and a time point $t \in \mathbb{T}$ to a binary value indicating whether the resources of i are needed at t. $\forall i \in \mathbb{I}\ \forall t \in \mathbb{T}$ if the resources of i are needed at t then $D(i,t) = 1$, else $D(i,t) = 0$.

Predicted resource demand, denoted as $P(i,t)$, is defined analogously to the actual demand $D(i,t)$.

Resource Allocation $A : \mathbb{I} \times \mathbb{T} \rightarrow \{0, 1\}$ is a function that maps i and t to a binary value indicating whether the resources are allocated for i at t.

Quality of Service (QoS) is measured as the ratio of the time when the resources are needed and allocated $T_{na}(i) = \{t \in \mathbb{T} \mid D(i,t) > 0 \text{ and } A(i,t) > 0\}$ to the time when the resources are needed $T_n(i) = \{t \in \mathbb{T} \mid D(i,t) > 0\}$ per index.

$$QoS = \sum_{i \in \mathbb{I}} T_{na}(i) \Big/ \sum_{i \in \mathbb{I}} T_n(i) \tag{1}$$

If the resources are always available when they are needed, then QoS equals to 1. If the resources become available after a delay, then QoS is lower.

Operational Cost Efficiency is measured as the ratio of the time when the resources are needed and allocated to the time when the resources are allocated $T_a(i) = \{t \in \mathbb{T} \mid A(i,t) > 0\}$ per index.

$$Eff = \sum_{i \in \mathbb{I}} T_{na}(i) \Big/ \sum_{i \in \mathbb{I}} T_a(i) \tag{2}$$

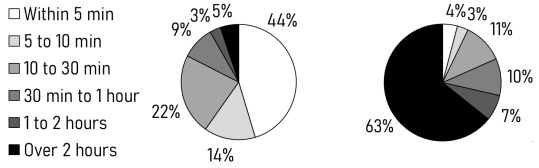
If the resources are only allocated when they are needed, then efficiency equals to 1. If the resources stay idle, then efficiency is lower.

Optimization Objective. An optimal policy allocates resources if and only if they are needed. Figure 3(d) illustrates this policy. Then, QoS and efficiency are equal to 1. An optimal policy requires a perfect demand prediction, which is hard to achieve in practice due to varying workloads. Nevertheless, the ultimate goal of a resource allocation policy is to maximize both QoS and efficiency, while keeping the execution latency low to guarantee real-time response of a latency-sensitive cloud service.

3. PREREQUISITES OF PROACTIVE RESOURCE ALLOCATION

3.1 Fine-Grained Production Telemetry

To guarantee accurate load prediction, the production telemetry must be fine-grained, cover several months, and contain all features that can be useful for prediction. We analyze two months of production telemetry in one Azure region where tens of thousands of indexes are currently deployed. Each event carries a timestamp in milliseconds, an index identifier, and a subscriber identifier.



(a) Number of idle intervals (b) Duration of idle time

Figure 2: Fragmentation of idle time

3.2 Varying Resource Demand

The resource demand of an index $i \in \mathbb{I}$ is stable if $\forall t \in \mathbb{T}\ D(i,t) = 1$. Otherwise, the demand of i is varying. Resources are provisioned to stable indexes to guarantee high QoS and efficiency, while avoiding the latency of load prediction. 32% of indexes receive requests every few minutes. 87% of requests belong to them (Figure 1).

Resources can be shared among indexes with varying demand. We focus on optimizing resource allocation for these indexes below. 68% of indexes have varying demand. 13% of requests belong to them.

3.3 Extensive Idle Intervals

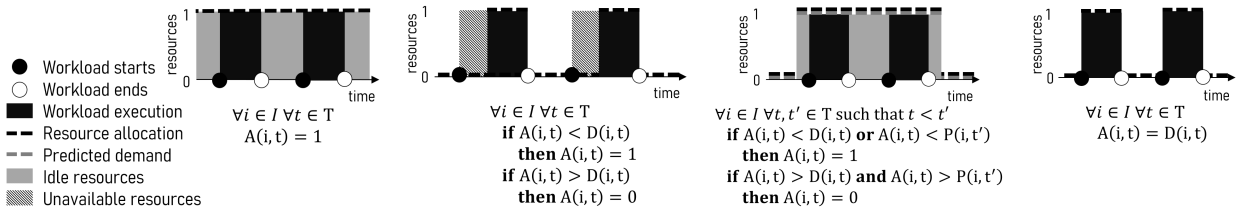
While the number of requests per index and hour can reach several hundreds, indexes receive requests only 18% of the time and stay idle 82% of the time on average. While 44% of idle intervals are within 5 minutes (Figure 2(a)), their total duration contributes only 4% to the total idle time (Figure 2(b)). Resources are not reclaimed during such short idle intervals to relieve the backend from the overhead of frequent scaling operations [23]. Even though only 5% of idle intervals exceed two hours, the total duration of these intervals contributes 63% to the total idle time. Resources can be effectively reused during such extensive idle intervals.

3.4 Accurate Demand Prediction

The proactive resource allocation policy relies on highly accurate workload prediction. Predicted long duration of idle intervals enables resource reclamation. Predicted start of customer workload enables resource allocation ahead of demand.

3.5 Low-Latency Prediction and Scaling

Low-latency workload prediction and scaling mechanisms are indispensable for the effectiveness of proactive resource allocation. Computationally expensive predictions and slow mechanisms reduce the time intervals during which resources can be reused. Worst yet, they introduce delays and jeopardize high



(a) Provisioned policy (b) Reactive policy (c) Proactive policy (d) Optimal policy

Figure 3: Resource allocation policies

quality of service for unpredictable workloads for which resources have to be allocated reactively.

4. RESOURCE ALLOCATION POLICIES

4.1 Base-Line Policies

Provisioned Policy always allocates resources, independently from demand. Figure 3(a) illustrates this policy. Analysis of historical load traces reveals that resources stay idle (i.e., $D(i,t) < A(i,t)$), unless the customer manually de-allocates resources during idle intervals [9, 10, 17, 19, 22, 30]. Manual resource allocation is labor-intensive, time-consuming, error-prone, neither scalable, nor durable.

Reactive Policy allocates resources based on the current demand [10, 11, 12]. Figure 3(b) illustrates this policy and defines its algorithm.

Unfortunately, resource scaling mechanisms are not instantaneous. Therefore, resources may be unavailable (i.e., $D(i,t) > A(i,t)$) when the workload starts in Figure 3(b). These delays make the reactive policy less suitable for latency-sensitive applications than the provisioned policy.

To reduce operational costs, the reactive policy reclaims resources when the workload stops. However, if idle intervals are short, then resource availability time is too fragmented for effective reuse. Worst yet, frequent scaling operations may introduce a significant backend overhead.

4.2 Proactive Policy

The proactive policy analyzes the historical resource usage patterns, predicts future demand, and allocates resources based on both current and predicted demand [23, 25, 27, 30]. Figure 3(c) illustrates this policy and defines its algorithm.

On the up side, the proactive policy reduces or even avoids delays in resource availability when the workload starts compared to the reactive policy (compare Figures 3(b) and 3(c)). Furthermore, the proactive policy relieves the backend from frequent scaling operations due to short idle intervals.

On the down side, idle time might increase compared to the reactive policy since resources are allocated ahead of demand and thus not immediately used by the customers. Also, in contrast to the base-line policies, the proactive policy introduces the computational overhead of demand prediction.

4.3 Demand Prediction Techniques

Any demand prediction algorithm can be plugged in the proactive policy. We now briefly summarize the range of commonly used techniques in industry.

Persistent forecast algorithm looks up the demand on previous day (or weekday), assumes that it stays the same on the following day (or weekday), and makes proactive decisions per index [22, 23].

Probabilistic algorithm analyzes historical traces and computes probability of requests per index and time window. Resources are proactively allocated during a window if the probability exceeds a threshold during the window. Resources are reclaimed once the workload stops and the probability falls below the threshold. Proactive resource allocation decisions can also leverage any other statistics, for example, count of requests [19, 23, 25, 27, 30].

Predictive algorithm trains a machine learning model on historical traces, predicts the demand, and makes proactive decisions based on both current and predicted resource demand per index. Time series forecast models, linear regression, exponential smoothing, classification models, and Neural Networks are commonly used [21, 22, 23, 27].

5. ACCURACY METRICS

While the standard metrics (such as hinge loss) allow to measure the overall accuracy of resource demand prediction, they provide little insight into the effectiveness of a policy in addressing the business needs. Table 1 summarizes the accuracy metrics for in-depth evaluation of the policies [23, 30].

Demand Predictability. To measure QoS, we differentiate between predictable and unpredictable demand. Resource demand for an index i at a time point t is predictable if $D(i,t) = 1$ and $P(i,t) = 1$. It is unpredictable if $D(i,t) = 1$ and $P(i,t) = 0$.

Correctness of Resource Allocation. To measure

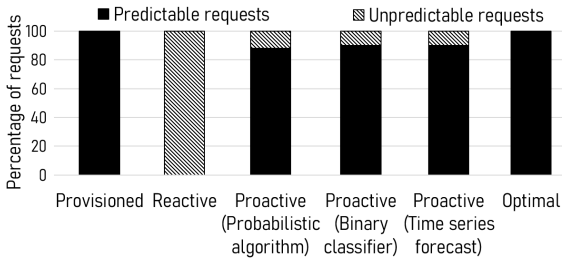


Figure 4: Demand predictability

Resource allocation	Conditions
Correctly allocated due to predictable demand	$P(i,t) = A(i,t) = D(i,t) = 1$
Correctly de-allocated (reclaimed)	$P(i,t) = A(i,t) = D(i,t) = 0$
Wrongly allocated (idle)	$P(i,t) = A(i,t) = 1$ but $D(i,t) = 0$
Wrongly de-allocated due to unpredictable demand	$P(i,t) = A(i,t) = 0$ but $D(i,t) = 1$

Table 1: Accuracy metrics

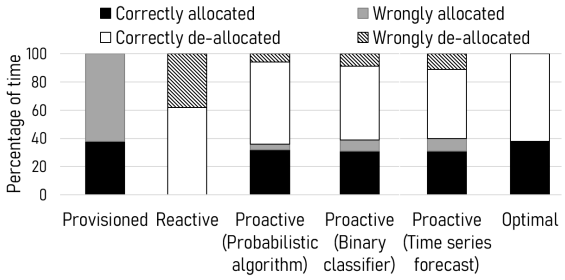


Figure 5: Correctness of resource allocation

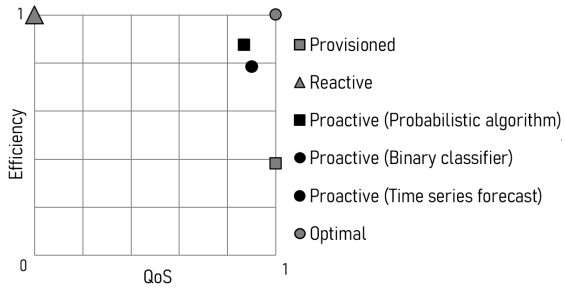


Figure 6: Quality of service versus efficiency

efficiency, we differentiate between the time intervals when the resources are correctly or wrongly allocated or de-allocated. Resource allocation for an index i at a time point t is correct if $D(i,t) = 1$. Analogously, de-allocation of resources of an index i at a time point t is correct if $D(i,t) = 0$.

Impact of Resource Allocation Decisions. If demand is predictable, then the resources are correctly allocated and high QoS is guaranteed. If demand is unpredictable, then the resources are wrongly de-allocated. In this case, resources must be allocated reactively. Thus, low-latency scaling mechanisms are indispensable even for the proactive policy.

While the resources are correctly de-allocated, they can be reclaimed to improve efficiency without sacrificing high QoS. While the resources are wrongly allocated, they stay idle and efficiency suffers. This waste of operational costs can be mitigated by the high accuracy of load prediction.

6. EXPERIMENTAL EVALUATION

6.1 Experimental Setup

Input Data contains two month of search requests for several thousands of indexes in one Azure region.

Hardware. We run all experiments on a Windows 11 machine with 10 Intel 2.80GHz CPUs, 128GB RAM, and 3TB SSD.

Methodology. We implemented the workload classification, the accuracy evaluation, and the probabilistic algorithm in Python 3.7. We use the existing libraries for ML.NET binary classifier [5] and the time series forecast model NimbusML [6]. We train these models based on three weeks of history and predict search requests one day ahead per index.

Metrics. We measure accuracy per Section 5 and execution latency. We run each latency experiment ten times and report the average result.

6.2 Quality of Service versus Efficiency

We consider indexes with varying workload in Figures 4–6. We exclude indexes with stable workload since the provisioned policy is the most effective for them (Figure 1). We omit the request processing time since it is the same for all policies (shown as black area in Figure 3).

Provisioned Policy is one extreme of the spectrum. Its QoS is optimal since resources are always allocated. However, resources stay idle 62% of the time in Figure 5 and efficiency is 0.38 in Figure 6.

Reactive Policy is the opposite extreme of the spectrum. On the up side, resources are de-allocated once the workload stops and efficiency is optimal. On the down side, resources are always wrongly de-allocated when the workload starts and QoS suffers.

Optimal Policy is the third extreme of the spectrum. It makes no mistakes in resource allocation. Thus, both QoS and efficiency are equal to 1.

Proactive Policy can leverage any demand prediction technique. We compare the probabilistic algorithm, ML.NET binary classifier, and the time series forecast model NimbusML.

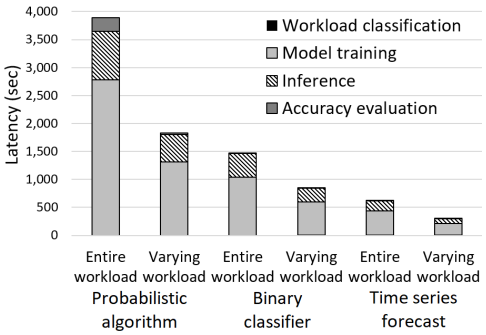


Figure 7: Execution latency

The accuracy of the binary classifier and the time series forecast model is quite similar. They correctly predict 90% of requests (Figure 4) and correctly allocate resources 31% of the time (Figure 5). Due to 10% of unpredictable requests, 10% of the time resources are wrongly de-allocated. 52% of the time resources are correctly de-allocated. Only 8% of the time resources are wrongly allocated.

As Figure 6 illustrates, these ML models achieve high QoS score of 0.9 at the cost of slightly lower efficiency score of 0.78 than the probabilistic algorithm. However, the probabilistic algorithm is also quite accurate. It correctly predicts 88% of requests and achieves QoS and efficiency scores of 0.87. Based on the results in Figures 4–6, we conclude that the proactive policy finds a reasonable balance between QoS and operational cost efficiency.

6.3 Execution Latency

In Figure 7, we compare the execution latency of the probabilistic algorithm, ML.NET binary classifier, and the time series forecast model NimbusML for the entire workload and the varying workload. Each bar is broken down into the following four sections: (1) Workload classification into stable and varying, (2) Model training based on three weeks of historical traces, (3) Inference one day ahead per index, and (4) Accuracy evaluation per Figures 4–6. The latency of workload classification is within one second and thus not visible at the scale of Figure 7.

Per our workload analysis in Section 3.2, 32% of indexes have stable resource demand and receive 87% of all search requests (Figure 1). Excluding such large portion of the workload reduces the latency by 2X for all models in Figure 7.

The latency of time series prediction using NimbusML is the lowest compared to other models in Figure 7. Its training is 3 minutes for all indexes with varying workload, while accuracy evaluation is 30 milliseconds for all indexes with varying workload. The average inference latency per index with varying workload is 110 milliseconds. Based on the results in Figures 6 and 7, we conclude that NimbusML is an accurate and light-weight model.

7. RELATED WORK

Demand-driven allocation of resources in the cloud has recently become a popular research direction [10, 11, 12, 13, 14, 16, 20, 26, 28, 29]. Some of these approaches are reactive [10, 11, 12]. A reactive policy reclaims resources once the customer workload stops, to save operational costs. Thus, quality of service may be jeopardized due to delays in resource availability when customer workload starts. In contrast, our approach makes proactive decisions based on recently observed resource usage patterns.

There are approaches to load analysis [9, 16, 18, 24] and load prediction using machine learning models [8, 13, 14, 15, 20, 26, 29]. Some of these models are computationally expensive and unintuitive for non-experts [19, 22, 27]. Thus, industrial approaches tend to deploy simple and light-weight forecast techniques to production [17, 19, 21, 22, 23, 25, 27, 30]. However, even light-weight techniques introduce the overhead of detailed continuous workload analysis which can be avoided in many cases. This is the approach we followed in this paper.

8. CONCLUSIONS AND FUTURE WORK

We define and evaluate the proactive resource allocation policy for Microsoft Azure Cognitive Search. We classify the workload of search requests and tailor the policy for each type of the workload. Such tailored approach significantly improves the quality of service, operational cost efficiency, and execution latency compared to other policies.

In this paper, we focus on the binary problem of proactive allocation and de-allocation of resources. Our solution caters to the application at hand and is a stepping stone towards a more general problem of proactive auto-scale of resources to any percentage of capacity. Solving this general problem is a subject for future research.

This paper presents the pre-deployment evaluation of the proactive policy. The post-deployment evaluation is subject for future work. It will measure how much operational costs (i.e., physical machines) are indeed saved by the proactive policy. It will depend on the effectiveness of placement policies which aim to place indexes that receive requests during mutually exclusive time intervals on the same physical machine to facilitate resource sharing among them. In-depth analysis of effective index placement is a subject for a follow-up publication.

REFERENCES

[1] Azure Cognitive Search. https://azure.microsoft.com/en-us/services/search/, 2023.

[2] Azure Cognitive Search Documentation. https://docs.microsoft.com/en-us/azure/search/search-what-is-azure-search, 2023.

[3] Azure SQL Database Serverless. https://docs.microsoft.com/en-us/azure/azure-sql/database/serverless-tier-overview, 2023.

[4] Google Serverless Computing. https://cloud.google.com/serverless, 2023.

[5] ML.NET Binary Trainer. https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.fasttree.fastforestbinarytrainer, 2023.

[6] NimbusML. https://docs.microsoft.com/en-us/python/api/nimbusml/nimbusml.timeseries.ssaforecaster, 2023.

[7] Serverless on AWS. https://aws.amazon.com/serverless/, 2023.

[8] R. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya. Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS. *IEEE Transactions on Cloud Computing*, 3:449–458, 08 2014.

[9] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *SOSP*, page 153–167, 2017.

[10] S. Das, F. Li, V. R. Narasayya, and A. C. König. Automated Demand-driven Resource Scaling in Relational Database-as-a-Service. In *SIGMOD*, pages 1923–1924, 2016.

[11] C. Delimitrou and C. Kozyrakis. Quasar: Resource-Efficient and QoS-Aware Cluster Management. *SIGPLAN Not.*, 49(4):127–144, 2014.

[12] A. Floratou, A. Agrawal, B. Graham, S. Rao, and K. Ramasamy. Dhalion: Self-Regulating Stream Processing in Heron. In *Proc. VLDB Endow.*, pages 1825–1836, 2017.

[13] Z. Gong, X. Gu, and J. Wilkes. PRESS: PRedictive Elastic ReSource Scaling for cloud systems. In *TNSM*, pages 9–16, 2010.

[14] S. Islam, J. Keung, K. Lee, and A. Liu. Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud. *Future Generation Comp. Syst.*, 28:155–162, 01 2012.

[15] A. Khan, X. Yan, S. Tao, and N. Anerousis. Workload Characterization and Prediction in the Cloud: A Multiple Time Series Approach. In *IEEE Network Operations and Management Symposium*, pages 1287–1294, 2012.

[16] C. Kilcioglu, J. M. Rao, A. Kannan, and R. P. McAfee. Usage Patterns and the Economics of the Public Cloud. In *WWW*, page 83–91, 2017.

[17] W. Lang, K. Ramachandra, D. J. DeWitt, S. Xu, Q. Guo, A. Kalhan, and P. Carlin. Not for the Timid: On the Impact of Aggressive over-Booking in the Cloud. *Proc. VLDB Endow.*, 9(13):1245–1256, 2016.

[18] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das. Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters. *SIGMETRICS Perform. Eval. Rev.*, 37(4):34–41, Mar. 2010.

[19] J. Moeller, Z. Ye, K. Lin, and W. Lang. Toto - Benchmarking the Efficiency of a Cloud Service. In *SIGMOD*, pages 2543–2556, 2021.

[20] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant. Automated Control of Multiple Virtualized Resources. In *EuroSys*, page 13–26, 2009.

[21] J. Picado, W. Lang, and E. C. Thayer. Survivability of Cloud Databases - Factors and Prediction. In *SIGMOD*, page 811–823, 2018.

[22] O. Poppe, T. Amuneke, D. Banda, A. De, A. Green, M. Knoertzer, E. Nosakhare, K. Rajendran, D. Shankargouda, M. Wang, A. Au, C. Curino, Q. Guo, A. Jindal, A. Kalhan, M. Oslake, S. Parchani, V. Ramani, R. Sellappan, S. Sen, S. Shrotri, S. Srinivasan, P. Xia, S. Xu, A. Yang, and Y. Zhu. Seagull: An Infrastructure for Load Prediction and Optimized Resource Allocation. *Proc. VLDB Endow.*, 14(2):154–162, 2020.

[23] O. Poppe, Q. Guo, W. Lang, P. Arora, M. Oslake, S. Xu, and A. Kalhan. Moneyball: Proactive Auto-Scaling in Microsoft Azure SQL Database Serverless. *Proc. VLDB Endow.*, 15(6):1279–1287, 2022.

[24] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch. Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis. In *SOCC*, pages 1–13, 2012.

[25] F. Romero, G. I. Chaudhry, I. Goiri, P. Gopa, P. Batum, N. J. Yadwadkar, R. Fonseca,

C. Kozyrakis, and R. Bianchini. Faa$T: A Transparent Auto-Scaling Cache for Serverless Applications. In *SoCC*, pages 122–137. ACM, 2021.

[26] N. Roy, A. Dubey, and A. Gokhale. Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting. In *CLOUD*, pages 500–507, 2011.

[27] M. Shahrad, R. Fonseca, I. Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, and R. Bianchini. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. In A. Gavrilovska and E. Zadok, editors, *USENIX*, pages 205–218. USENIX Association, 2020.

[28] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. CloudScale: Elastic Resource Scaling for Multi-tenant Cloud Systems. In *SOCC*, pages 1–14, 2011.

[29] R. Taft, N. El-Sayed, M. Serafini, Y. Lu, A. Aboulnaga, M. Stonebraker, R. Mayerhofer, and F. Andrade. P-Store: An Elastic Database System with Predictive Provisioning. In *SIGMOD*, page 205–219, 2018.

[30] L. Viswanathan, B. Chandra, W. Lang, K. Ramachandra, J. M. Patel, A. Kalhan, D. J. DeWitt, and A. Halverson. Predictive Provisioning: Efficiently Anticipating Usage in Azure SQL Database. In *ICDE*, pages 1111–1116, 2017.

From Large Language Models to Databases and Back: A Discussion on Research and Education

Sihem Amer-Yahia (CNRS, Univ. Grenoble Alpes, France), Angela Bonifati (Univ. Lyon 1, CNRS, IUF, France), Lei Chen (HKUST(GZ) & HKUST, China), Guoliang Li (Tsinghua University, China), Kyuseok Shim (Seoul National University, Korea), Jianliang Xu (Hong Kong Baptist University), Xiaochun Yang (Northeastern University, China)

1. LLMS AND DBS

In recent years, large language models (LLMs) have garnered increasing attention from both academia and industry due to their potential to facilitate natural language processing (NLP) and generate high-quality text. Despite their benefits, however, the use of LLMs is raising concerns about the reliability of knowledge extraction. The combination of DB research and data science has advanced the state of the art in solving real-world problems, such as merchandise recommendation and hazard prevention [30]. In this discussion, we explore the challenges and opportunities related to LLMs in DB and data science research and education.

LLMs for DB research. LLMs have proven to be highly useful for language writing, as they can identify and correct grammar errors. Additionally, LLMs can serve as a valuable resource for knowledge acquisition and analysis. However, the accuracy of extracted knowledge is not guaranteed, and bias can be introduced, leading to potential inaccuracies in the data analysis process.

LLMs can be highly effective in data preparation and labeling tasks, such as text mining, text parsing, keyword extraction, and sentiment analysis. It also has great potential to improve feature extraction, selection, and parameter tuning.

DB research for LLMs. DB research can support LLM development from data cleaning and preprocessing to training and optimization. For instance, domain-specific knowledge can be incorporated into training data to create more accurate and reliable models. Furthermore, DB research can be used to optimize prompt engineering to improve the effectiveness of LLM.

LLMs for education. LLMs can be used to reform DB education and address the challenges and concerns related to their use. LLMs can provide students with a wealth of knowledge and practical skills, such as techniques for handling dirty data. However, care must be taken to ensure that the information and programming styles provided by LLMs are accurate and free from bias or misinformation. As such, it is crucial to consider how to detect plagiarism when people use LLMs to generate scientific articles, papers, or assignments. We conclude that, although there are challenges associated with the use of LLMs in DB research and education, these can be addressed through careful research and thoughtful integration of LLMs into the data science curriculum.

ChatGPT, today's famous LLM. A Generative Pre-Trained Transformer (GPT) is a language model relying on deep learning that takes a text-based input and generates natural language. Chat Generative Pre-trained Transformer (ChatGPT) is a chatbot released by OpenAI in November 2022 and is built on top of OpenAI's GPT-3.5 large language model (LLM). It is trained on a large body of text from a variety of sources in 2020 and can write any form of text such as essays, poems, paragraphs and computer programs. It is able to understand and generate natural language with a high level of accuracy and fluency. A new version based on GPT-4 was released on March 2023 and is available for paid subscribers on a limited basis.

While ChatGPT can help people with a lot of well-suited tasks such translation of foreign languages, summarization of a text and generation of human-like conversational responses, it has several drawbacks. Since large language models perform the task of predicting the next word, it produces the output text without any concern of originality, plagiarism and privacy. Furthermore, since the training text data is derived from publicly available data before 2021, it cannot provide accurate information in a timely manner and may not know the most up-to-date information. Moreover, ChatGPT is not yet able to solve sophisticated math or high-level tasks. Since its outputs may contain false or outdated information, we should carefully evaluate the outputs of ChatGPT and use them cautiously.

2. PROS AND CONS OF LLMS

LLMs for Data Science Research. Calculators and word processors are useful tools for people that allow not to worry about complex arithmetic calculations and incorrect spellings/grammars as well as citation labels of their writing, respectively. The positive implications of using both tools are that people can focus and concentrate more on the content of their work without worrying about inaccurate calculations or spelling/grammatical errors.

Similarly, data scientists can utilize ChatGPT for their data science research to focus more on high-level creative thinking including getting the big picture, original idea generation, and analytical thinking. For example, they can utilize it to summarize the texts about related works, learn about a particular research topic, brainstorm about research directions for their new project and improve their writing skills of technical papers. ChatGPT can also assist non-native data scientists in understanding, interpreting and writing English texts. On the other hand, they can use ChatGPT to produce a high-quality code with explanations and improve their coding skills. It can even help a data scientist rewrite his old code in a programming language to an equivalent code in another programming language. Note that an important skill required for data scientists is the ability to produce a good quality of code. Thus, instead of spending time in learning how to code or producing code for data analysis, data scientists can concentrating more on their research by utilizing ChatGPT.

While more training data is likely to produce a more accurate model [6], there are many applications such as named entity recognition [33], relation extraction [23] and image classification [36] where producing a large-scale training data by manual labeling is expensive and time-consuming. To quickly obtain a large-scale training data with low cost, one approach is to use weak supervision that automatically annotates unlabeled data by heuristic rules or machine learning models. For example, one of the most popular techniques for weak supervision is distant supervision that utilizes external knowledge bases to produce weak labels [27]. We can also utilize ChatGPT as an alternative method for weak supervision. For instance, a recent investigation augments training data for few-shot classification by rephrasing each sentence in the training data into multiple similar sentences [18].

Data Science Research for LLMs. Since an LLM model is only as knowledgeable as the training texts that have been provided for learning, its knowledge is limited according to the training data and it may become unfair by absorbing the biases from the training data. Furthermore, it lacks the capability of ethical thinking too. Thus, developing learning techniques to overcome such handicaps of LLM models will be very helpful to LLMs. Publicly available text data on the Web has a lot of sensitive information and training LLMs with public data can thus disclose sensitive and private information of people. On the other hand, as users input more data with conversations into ChatGPT, it may potentially leak the sensitive information to other users of ChatGPT. Thus, developing the privacy preserving schemes, such as the differential privacy, with high utility for training LLM modles will help LLMs to protect the privacy of individuals.

LLMs for Computer Science and Data Science Education. ChatGPT can be a useful tool for both disciplines and we need to reform the curricular to include ChatGPT. In [22], opportunities of utilizing ChatGPT are addressed and several tasks are suggested for computer science education. For instance, teachers can ask students to generate a code for a given problem, and then explain, analyze and improve the code. In addition, we can also ask students to write their own code for the same problem and find the similarities as well as differences between two codes.

Students can utilize ChatGPT to summarize, understand, and learn texts about existing works for a particular research topic, enhance their coding as well as debugging skills to generate a high quality code, brainstorm about research topics and improve their writing skills of technical papers. Thus, students can utilize ChatGPT to focus more on high-level creative thinking including getting the big picture, original idea generation, analytical thinking and the detailed steps of their methods to solve a given problem. To do so, since the outputs of ChatGPT may contain false or outdated information, students should learn how to use ChatGPT effectively and cautiously.

While there are many advantages of including ChatGPT in the curriculum, students who consistently depend on ChatGPT may lose or cannot improve their skills of summarizing the texts, searching for relevant materials about a particular topic and writing technical papers by themselves. Furthermore, while ChatGPT is proficient in generating fluent text, it may produce the contents with lack of clarity as well as originality. Moreover, the outputs of ChatGPT may contain false or outdated information, and may even present a plagiarized writing from another source without citing properly. Thus, we need to provide precise guidelines of using

ChatGPT to students so that they can learn how to use ChatGPT effectively and utilize the ChatGPT outputs with caution.

3. WHAT CAN LLM DO FOR DBS?

LLMs typically report probabilistic results but cannot be used to report fully deterministic results [20, 29]. Thus, LLM can be leveraged to handle inexact data/query processing problems that can tolerate approximate results, e.g., approximate query processing and data integration. However, it is hard to use LLMs to support exact data/query processing, e.g., query rewriting. So we discuss how to use LLMs to support data/query processing.

LLMs for DB Research. For some of DB problems, e.g., data discovery, cleansing, and integration, users are satisfied with approximate results. Intuitively, we can utilize LLMs to improve generalizability. However, several challenges arise. The first is automatic prompt engineering that automatically generates appropriate prompts to guide LLMs to find correct answers. LLMs have limitations on the number of token constraints and long latency, and the automatic prompt engineering tool should optimize these two factors. The second is how to integrate domain knowledge into LLMs. Current LLMs use open Web corpus to pretrain a large language model and thus can well support data cleaning and integration on Web data but cannot effectively support vertical domains that are absent on open Web. Hence, the challenge is to fine-tune LLMs to support domain knowledge or use prompt engineering to teach LLMs to do this. The third is how to combine LLMs and existing data science tools, as it is expensive to call LLMs and it is beneficial to utilize some existing tools to reduce the cost. For example, there are many good DB tools, e.g., blocking tools and entity-matching tools, and we can design tool learning that enable LLMs to call effective tools to reduce the cost.

DB Research for LLMs. The theory and model architecture of LLMs are mature, and the researchers and scientists that are working on LLMs focus on providing high-quality data to train LLMs, e.g., discovering data, cleaning data and integrating data. There exists a plethora of tools for the above DB tasks that can be used to prepare the data on which LLMs are trained. A challenge is how to make a win-win loop between DB systems and LLMs, which uses DB techniques to provide high-quality of LLMs and uses LLMs to optimize the DB tools.

Querying data with natural language. An interesting application is Text-to-SQL, which converts natural language queries into SQL statements. This has been a long-studied research problem. The state-of-the-art is currently a work presented at AAAI 2023 [26], which achieved an accuracy of 79.9% using a seq2seq pre-trained language model. With continuous prompts, ChatGPT enables users to interactively refine the generated SQL queries and could further improve their accuracy. This unique feature presents a potential opportunity to integrate LLMs with existing Text-to-SQL techniques to generate more precise SQL statements.

Additionally, ChatGPT allows users to query a dataset with natural language. This could eliminate the need for SQL queries and increase the efficiency of data retrieval and analysis for certain applications, which poses the question of whether SQL remains necessary or if it is possible to translate text into a query evaluation plan for DB result evaluation. The integration of LLMs with DB techniques has the potential to open up new opportunities for research and advancement in data science.

LLMs for Logical Query Optimization. LLMs could be used to support logical query optimization. The key challenge is that the translated query plan should be exactly equivalent to the original plan. Therefore, there are several possible solutions. The first uses LLMs to obtain an optimized query plan and then verifies the equivalence using existing techniques (and then keeps the equivalent query and drops the in-equivalent one). The second uses LLMs to optimize using query rewriting rules, including discovering new query-rewrite rules and judiciously using the rules (including whether to use a rule and to determine the order of different rules).

LLMs for Physical Query Optimization. Different from logical query optimization, physical query optimization should utilize the physical DB statistics and it is vital to provide these information to LLMs. However, the current LLMs cannot effectively support numerical values. Two challenges arise in this context. The first is to fine-tune the LLMs that enables LLMs to support numerical statistics. The second is to embed DB statistics into the prompt to facilitate that LLMs can use such information to get an optimized physical plan.

4. LLMS NEED REASONING

LLMs and data integration To understand the differences between LLMs and DBs, let us compare them with the process of integrating heterogeneous data sources. Data integration is a long-standing research problem in data management [7, 32]. Schema mapping, data deduplication, schema and data fusion are all tasks that involve considering up-to-date data sources, as well as personal and pro-

prietary data. By leveraging the inherent semantics of mappings (correspondences between queries or views on different data sources), these tasks do not need re-training on large corpuses of data and can easily capture new incoming data. As recently argued in a vision paper, these data of different nature are not considered so far in the LLM [21]. Moreover, integrated data can easily cater for privacy constraints and become trustworthy, for instance by blending mappings with policy views [9] or by having humans as first-class citizens in the data integration process [5, 8, 16].

Provenance and lineage information, in particular why, how and where provenance characterizing query results [15], could serve the need of filling an existing gap of large language models, missing the key capability of locating the sources of information.

But what is missing in LLMs to take advantage of DBs and integrated data sources? LLMs need to retain provenance and schema as well as other kinds of metadata, which is not merely raw data and should not be unified with raw data; They need to process and compute provenance throughout the learning process and be able to annotate the results with provenance information (and, thus, citation sources); They need to capture privacy constraints and privacy policies in the data acquisition and data fusion process. These are only some examples.

LLMs and Graphs. Graphs are a great source of knowledge, which is typically curated by humans and, as such, can be seen as high-quality and trustworthy integrated information. Examples of such highly curated graphs are Wikidata and DBPedia, that are typically used by search engines [34], whereas LLM are trained on large text corpora, such as Wikipedia, books, news and open datasets. Knowledge graphs such as DBPedia and Wikidata can be navigated and queried by leveraging query endpoints. Queries collected at the endpoints allow to understand what the users search within the knowledge graphs and thus indirectly to characterize the underlying structure of the data.

To illustrate the difference between semantics in graph DBs and language models, we choose to confront a query from the DBPedia graph query logs with question answering in ChatGPT.

Figure 1 shows the shape of a 7-node clique. All edges between Henry The VIII and his six wifes are labeled "dbpedia-owl:spouse" and the edges between each pair of wifes are labeled with a property path "!dbpedia-owl:same-As" [1], (to retrieve his distinct wifes). Figure 2 shows question answering on ChatGPT on the possible relationships (in terms of RDF properties) between Henry The VIII and his wives. It shows the RDF syntax of the "spouse" relationship and it substantiates the result with the information about the source (DBPedia) from which this information has been retrieved. As such, ChatGPT can clearly identifies the spouses of Henry The VIII and enumerate them, but he cannot do further reasoning, bringing to the visual clique-shaped query as in Figure 1. Thus, it cannot perform advanced querying and analysis on databases such as DBPedia or Wikidata at the time being and, as a text-based AI model, is unable to draw query shapes and search query logs [11, 12].

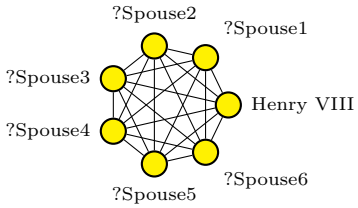


Figure 1: The Henry VIII query, a 7-clique with one constant and six variables. All edges between Henry VIII and the variables are labeled "dbpedia-owl:spouse" and all edges between variables are labeled with the property path "!dbpedia-owl:sameAs" [12].

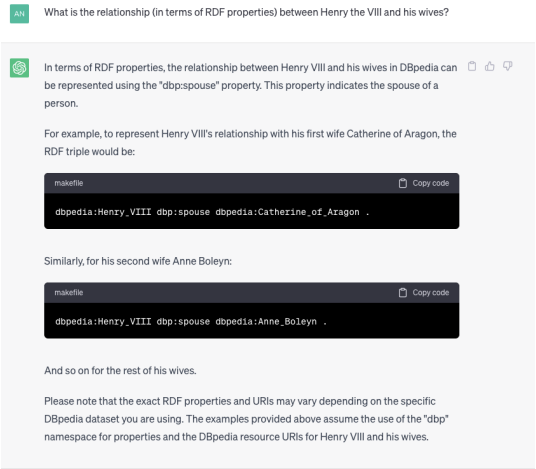


Figure 2: Snapshot of question answering on Henry The VIII's wives on ChatGPT.

LLMs and Reasoning. Contrarily to LLMs, graphs enable symbolic reasoning, when logic-based existential rules and path queries are used to augment existing knowledge graphs with additional inferred data [14]. Moreover, DB queries including graph queries return certain answers [10], whereas

[1] A property path is a path with a regular expression allowing to navigate graph data.

output of LLM is highly uncertain depending on the frequency of values in the training data.

Combining results from LLMs with graphs could improve the results of LLMs by unifying machine reasoning with symbolic and logic-based reasoning [31]. The knowledge graph lifecycle with maintenance operations, update propagation and error fixing is nonexistent for LLMs. On the other hand, the outputs of LLM could be used to enrich knowledge graphs for question answering tasks by leveraging graph attention networks [35].

Concluding, we cannot disagree with Gary Markus' blog 'Hoping for the Best as AI Evolves' [28]: "Large language models lack mechanisms for verifying truth; we need to find new ways to integrate them with the tools of classical AI, such as DBs, Webs of knowledge, and reasoning.".

5. LLMS AS A RESEARCH ASSISTANT

Helping with Scientific Writing. LLMs like ChatGPT can help with scientific writing in several ways, such as proofreading, rewriting, summarization, and even suggesting titles for research papers. It can also help improve the language to better communicate research ideas and results, so as to facilitate research collaborations. However, the key challenge lies in creating effective prompts that generate high-quality responses. Examples include revising a paragraph from the introduction of a Computer Science academic paper so that citations are kept and the text has a better structure; Suggesting five creative titles for a paper base don an abstract; Writing a 1-page sensational press. With well-crafted prompts, ChatGPT can deliver results that rival those of paid editing services.

Assisting with Data Analytics. LLMs can assist with data creation. It can generate data based on input parameters, which can be useful in situations where large amounts of synthetic data are needed for research. Additionally, LLMs can help generate code for data analytics tasks [19]. One such task is exploratory data analysis (EDA). For example, suppose we have a loan dataset that we want to perform EDA on. We can simply prompt ChatGPT with a request to write Python code to load and perform EDA on the loan dataset, and it will provide us with a code that we can use to analyze and visualize the data. ChatGPT can also assist with other data analytics tasks such as data cleaning and preprocessing, feature engineering, hyperparameter tuning, and model selection and evaluation. It can help save time and effort by automating some of these tedious tasks and focus on more complex aspects of data analytics.

Limitation of Hallucination in LLMs. Despite their many benefits, LLMs do have some limitations that need to be considered. One of these limitations is their potential to generate incorrect content that appears plausible, known as "hallucination". This is particularly relevant in research paper writing, where ChatGPT may suggest nonexisting references or provide inaccurate information. For example, when asked to recommend a paper on the topic of data science authored by Lei Chen, ChatGPT suggested a paper titled "Crowdsourced Data Management: A Survey" authored by Lei Chen, Reynold Cheng, Silviu Maniu, and Wang-Chien Lee and published in IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 9, pp. 1959-1977, September 2013. However, although this paper title does exist, it was not authored by Lei, rather by Guoliang et al. and published in 2016 (see [3]). In fact, Lei, Sihem, and Anand did author a survey paper on a related topic but their title is different (see [4]). To address this issue, one possible solution is to integrate ChatGPT with a knowledge graph and ground truth facts, which can help verify the accuracy of the generated information by cross-referencing it with existing data.

Incorporating External Data into LLMs. While LLMs have access to a vast amount of data, their knowledge is still limited by the data they have been trained on. There are some recent efforts to incorporate external data into LLMs through the use of prompts. However, LLMs may have a limitation on the length of input they can process, which can impact their ability to understand complex or lengthy inputs. For example, the GPT-4 base version allows up to 8,192 tokens, which may not be sufficient for processing longer texts. One approach to overcome this limitation, known as chunking, is to split the longer text into smaller segments and process each segment separately [25]. It can be combined with traditional data science techniques such as embedding and indexing to improve the model's performance on longer inputs. Yet, chunking can also introduce challenges such as maintaining coherence and consistency between segments, which may require further research.

Ethical and legal issues. Ethical concerns such as bias, plagiarism, and data privacy and security are also significant issues when using LLMs. LLMs may be biased towards the data they were trained on, which can lead to unfair or inaccurate results. Additionally, there is a risk of privacy breaches if the input contains sensitive information. Moreover, legal and copyright issues must also be considered when using LLMs. If the model generates copy-

righted material without proper licensing, it could lead to legal repercussions. Thus, it is essential to responsibly and ethically use LLMs by thoroughly vetting and verifying generated content before using or publishing it. The new ACM policy requires disclosure of the use of generative AI tools for content generation in published work, with specific details regarding their usage provided in the acknowledgments section or elsewhere in the work [1].

6. LLMS FOR EDUCATION

One area that is receiving both scientific and media coverage these days is the impact of LLMs on education. Several concerns have been raised about students using ChatGPT to complete tests, ChatGPT passing bar exams and professors using it to devise quizzes. As scientists, we focus on discussing LLMs in teaching and LLMs for doing research on education, that we refer to as LLM4ED.

Teaching LLMs. We need to teach LLMs just like other models. This will contribute to demystifying them and to raising awareness about their lack of transparency as well as their benefits and pitfalls. We also need to encourage students to treat LLMs like any other recommender. Their prediction accuracy should be tested keeping in mind that the best paper award at RecSys in 2019 showed that KNN outperformed 6 Deep Learning recommendation methods on MovieLens data [17]. These includes Collaborative Variational Autoencoder and and Neural Collaborative Filtering methods. Students need to learn to build on top of LLMs and treat them just like other models. In recent work, we built a meta-recommender that learns the best algorithm to apply given a user and a dataset or a user, and a question [13]. This approach could integrate LLMs as a recommendation option.

LLM4ED. There are many challenges and opportunities of leveraging LLMs for education [2,24]. An LLM could be modeled as a learner or to support learners, teachers or administrators.

LLMs as Learners. This would require to model learners' data and behavior. This data is highly diverse. *Student and curriculum records* capture individual learners' records such as their demographics which are usually provided by learners at registration time as well as information on learning material such as artefacts, and assessment and outcome requirements. *Learning records* capture data on learners' achievements such as grades and assessment outcomes. *Learning logs* record learners' engagement with artefacts, feedback to learners and collaboration. This would encourage students to see LLMs as a peer from which to learn and to criticize.

LLMs for Learners, Teachers and Administrators. This opens new opportunities for LLM-in-the-loop research in education. For instance, in the context of collaborative learning, team formation algorithms could be revisited to consider LLMs as team members. In the context of developing an intelligent teaching assistant, LLMs are already in use for grading students which raises the question of accuracy and fairness. They are also used help teachers create content. In both grading and content creation, ensuring teachers' agency will allow them to guide the process and override automatic decisions. LLMs can also be used for admission support and student dropout prediction analytics. A particular point of attention in all these applications is the study of bias in ranking (student ranking) and in classification (student dropout prediction). Adding provenance to LLMs to better identify their sources of flaws would also address some of these concerns.

7. CONCLUSION

DBs and LLMS are on either side of the spectrum of data science research. DBs are collections of data, while LLMS are viewed as summaries and profiles of experiences based on data. DBs can help data scientists to query and statistically analyze the stored data accurately and efficiently. LLMs, on the other hand, are learned from textual data and can help data scientists to solve semantic application problems related to natural language. However, LLMs cannot guarantee an accurate or complete answer.

We discussed the advantages and limitations that LLMs brought to data science and DB research and education. Regarding the discussion of LLMs, the optimistic view is LLMs can facilitate most of data science and data management tasks, including data discovery, data cleaning, data integration, and data visualization, and LLMs can bring great benefit to education. While the pessimistic view is LLMs are hard for data modeling, data analytics, and data interpretation, meanwhile, they might weaken learning skills and training LLMs could bring bias, plagiarism, privacy, legal, and copyright issues. People should be careful about the results obtained from an AI model and take them with a grain of salt.

We also showed our insights that data science and DB research could also help LLMs, including how to use provenance and lineage information to fill the existing gap of LLMs, how to take advantage of DBs and integrated data sources, how to unify machine reasoning with symbolic and logic-based reasoning by combining results from LLM with graphs, and how to combine model-centric and data-centric approaches altogether.

8. REFERENCES

[1] ACM. ACM Policy on Authorship , 2023. [Online] https://www.acm.org/publications/policies/frequently-asked-questions. Accessed on May 20, 2023.

[2] Sihem Amer-Yahia. Towards ai-powered data-driven education. *Proc. VLDB Endow.*, 15(12):3798–3806, 2022.

[3] Guoliang Li and Jiannan Wang and Yudian Zheng and Michael J. Franklin. Crowdsourced data management: A survey. *IEEE Trans. Knowl. Data Eng.*, 28(9):2296–2319, 2016.

[4] Anand Inasu Chittilappilly and Lei Chen and Sihem Amer-Yahia. A survey of general-purpose crowdsourcing techniques. *IEEE Trans. Knowl. Data Eng.*, 28(9):2246–2266, 2016.

[5] Abdallah Arioua and Angela Bonifati. User-guided repairing of inconsistent knowledge bases. In Michael H. Böhlen, Reinhard Pichler, Norman May, Erhard Rahm, Shan-Hung Wu, and Katja Hose, editors, *Proceedings of the 21st International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018*, pages 133–144. OpenProceedings.org, 2018.

[6] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, Toulouse, France, July 2001. Association for Computational Linguistics.

[7] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm, editors. *Schema Matching and Mapping*. Data-Centric Systems and Applications. Springer, 2011.

[8] Angela Bonifati, Ugo Comignani, Emmanuel Coquery, and Romuald Thion. Interactive mapping specification with exemplar tuples. In Semih Salihoglu, Wenchao Zhou, Rada Chirkova, Jun Yang, and Dan Suciu, editors, *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 667–682. ACM, 2017.

[9] Angela Bonifati, Ugo Comignani, and Efthymia Tsamoura. Exchanging data under policy views. In Yannis Velegrakis, Demetris Zeinalipour-Yazti, Panos K. Chrysanthis, and Francesco Guerra, editors, *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, pages 1–12. OpenProceedings.org, 2021.

[10] Angela Bonifati, George H. L. Fletcher, Hannes Voigt, and Nikolay Yakovets. *Querying Graphs*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018.

[11] Angela Bonifati, Wim Martens, and Thomas Timm. Navigating the maze of wikidata query logs. In *The World Wide Web Conference*, WWW '19, page 127–138, New York, NY, USA, 2019. Association for Computing Machinery.

[12] Angela Bonifati, Wim Martens, and Thomas Timm. An analytical study of large SPARQL query logs. *VLDB J.*, 29(2-3):655–679, 2020.

[13] Nassim Bouarour, Idir Benouaret, and Sihem Amer-Yahia. How useful is meta-recommendation? an empirical investigation. In *2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, December 15-18, 2021*, pages 600–606, 2021.

[14] David Carral, Irina Dragoste, Markus Krötzsch, and Christian Lewe. Chasing sets: How to use existential rules for expressive reasoning. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1624–1631. ijcai.org, 2019.

[15] James Cheney, Laura Chiticariu, and Wang Chiew Tan. Provenance in databases: Why, how, and where. *Found. Trends Databases*, 1(4):379–474, 2009.

[16] Laura Chiticariu and Wang Chiew Tan. Debugging schema mappings with routes. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 79–90. ACM, 2006.

[17] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. *CoRR*, abs/1907.06902, 2019.

[18] Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming

Liu, and Xiang Li. Auggpt: Leveraging chatgpt for text data augmentation, 2023.
[19] DataCamp. A guide to using ChatGPT for data science projects, 2023. [Online] https://www.datacamp.com/tutorial/chatgpt-data-science-projects. Accessed on May 4, 2023.
[20] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(1):452–459, 2015.
[21] Alon Y. Halevy and Jane Dwivedi-Yu. Learnings from data integration for augmented language models. *CoRR*, abs/2304.04576, 2023.
[22] Orit Hazzan. Chatgpt in computer science education, 2023. BLOG@CACM on January 23, 2023.
[23] Woohwan Jung and Kyuseok Shim. Dual supervision framework for relation extraction with distant supervision and human annotation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6411–6423, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.
[24] Hassam Khosravi, Shazia Sadiq, and Sihem Amer-Yahia. Data management of ai-powered education technologies: Challenges and opportunities. In *Learning Letters, 0.*, page (to appear), 2023.
[25] LangChain. LangChain Chat, 2023. [Online] https://blog.langchain.dev/langchain-chat/. Accessed on May 4, 2023.
[26] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. RESDSQL: Decoupling schema linking and skeleton parsing for text-to-SQL. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI)*, 2023.
[27] Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. BOND: bert-assisted open-domain named entity recognition with distant supervision. In *KDD 2020*, pages 1054–1064. ACM, 2020.
[28] Gary Marcus. Hoping for the best as ai evolves. *Commun. ACM*, 66(4):6–7, mar 2023.
[29] Kevin Patrick Murphy. Probabilistic machine learning: An introduction. *MIT Press, March*, 2022.
[30] M. Tamer Özsu. Data science - A systematic treatment. *Commun. ACM*, 66(7):106–116, 2023.
[31] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid G. Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. The future is big graphs: a community view on graph processing systems. *Commun. ACM*, 64(9):62–71, 2021.
[32] Michael Stonebraker and Ihab F. Ilyas. Data integration: The current status and the way forward. *IEEE Data Eng. Bull.*, 41(2):3–9, 2018.
[33] Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. Meta self-training for few-shot neural sequence labeling. In *KDD 2021*, pages 1737–1747, 2021.
[34] Gerhard Weikum. Knowledge graphs 2021: A data odyssey. *Proc. VLDB Endow.*, 14(12):3233–3238, 2021.
[35] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. QA-GNN: reasoning with language models and knowledge graphs for question answering. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 535–546. Association for Computational Linguistics, 2021.
[36] Bowen Zhang, Yidong Wang, Wenxin Hou, HAO WU, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. In *Advances in Neural Information Processing Systems*, volume 34, pages 18408–18419. Curran Associates, Inc., 2021.

Announcement from the ACM on the New Authorship Policy

Chair
Divyakant Agrawal
Department of Computer Science
UC Santa Barbara
Santa Barbara, California
USA
+1 805 893 4385
agrawal@cs.ucsb.edu

Vice-Chair
Fatma Ozcan
Systems Research Group
Google
Sunnyvale, California
USA
+1 669 264 9238
Fozcan@google.com

Secretary/Treasurer
Rachel Pottinger
Department of Computer Science
University of British Columbia
Vancouver
Canada
+1 604 822 0436
Rap@cs.ubc.ca

ACM has recently released its new policy on authorship, which covers a range of key topics, including the use of generative AI tools. It is available at https://www.acm.org/publications/policies/new-acm-policy-on-authorship . Please familiarize yourself with the new policy and the associated list of Frequently Asked Questions at https://www.acm.org/publications/policies/frequently-asked-questions .

Please reach out to Sara Kate of ACM (heukerott@hq.acm.org) for further questions.