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), Alexandra Meliou (2024 SIGMOD PC co-chair), S Sudarshan (2024 SIGMOD PC co-chair), Floris Geerts (Chair of PODS), Genoveva Vargas Solar (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:
Hui Li (SIGMOD Record), Georgia Koutrika (Blogging), Wim Martens (PODS)

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

SIGMOD Record Associate Editors:
Lyublena Antova, Manos Athanassoulis, Angela Bonifati, Renata Borovica-Gajic, Vanessa Braganholo, Aaron J. Elmore, George Fletcher, Wook-Shin Han, H V Jagadish, Alfons Kemper, Benny Kimelfeld, Samuel Madden, Kyriakos Mouratidis, Tamer Özsu, Kenneth Ross, Yufei Tao, Pınar Tözün, Immanuel Trummer, and Yannis Velegrakis

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

SIGMOD Awards Committee:
H.V. Jagadish (Chair), Sourav S Bhowmick, Angela Bonifati, David Maier, Sayan Ranu, Wang-Chiew Tan

Jim Gray Doctoral Dissertation Award Committee:
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:

Dan Suciu (2022)  Joseph M. Hellerstein (2023)  Samuel Madden (2024)

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 Reid, and Utkarsh Srivastava (2018); Xiaofeng Bao, Charlie Bell, Murali Bramhadesam, James Corey; Neal Fachen, 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, Greg Burd, Michael Cahill, Ron Cohen, Alex Gorrod, George Feinberg, Mark Hayes, Charles Lamb, Linda Lee, Susan LoVerso, Tom McCorkle, John Merrells, Mike Olson, Carol Sandstrom, Steve Sarette, David Schacter, David Segal, 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 Konszt, Gerard Lemson, Nolan Li, Dmitry Medvedev, Jeff Munn, Deoyani Nandrekar-Heinis, Maria Nieto-Santisteban, Wil O’Mullane, Victor Paul, Don Slutz, Alex Szalay, Gyula Szokolya, Mani Taghizadeh-Popp, Jordan Raddick, Bonnie Souter, Ani Thakar; Jan Vandenbergh, 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, Xiaol Li, Cheng Liu, Yanbo Liang, Xiangrui Meng, Sean Owen, Josh Rosen, Kousuke Saruta, Scott Shenker, Ion Stoica, Takuji Ueshin, Shivaram Venkataraman, Gengiang Wang, Yuming Wang, Patrick Wendell, Reynold Xin, Takeshi Yamamuro, Kent Yao, Matei Zaharia, Rufeng Zheng, and Shixiong Zhu (2022); Aljoscha Krettek, Andrey Zagrebin, Anton Kalashnikov, Arvid Heise, Asterios Katofidimos, Jianguo (Becket) Qin, Benchoo Li, Bowen Li, Caizhi Weng, ChengXiang Li, Chesney Schepler, Chihan Park, Congxian Qiu, Daniel Warneke, Danny Cranmer, David Anderson, David Moravec, Daid Wyszawokich, 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, Hao Hui, Mai, Heqin 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, Largu Andra, Martinvis Visser, Martin Balassi, Matthias J. Sax, Matthias Pohl, Matyas Orhidei, Maximilian Michels, Nico Kruber, Niels Basjes, Paris Carbone, Piotr Nowojski, Qingsheng Ren, Robert Metzger, Roman Khachatranyan, Rong Rong, Rui Fan, Rui Li, Sebastian Schelter, Seif Haridi, Sergey Nuyanzin, Seth Wiesman, Shaohuan Wang, Shengkai Fang, Shuyi Chen, Sihua Zhou, Stefan Richter, Stephan Ewen, Theodore 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, Zhijian Yang, Zhipeng Zhang, Zhu Zhu, Zili Chen (2023); Zhaojing Luo, Beng Chin Ooi, Wei
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:

<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1992</td>
<td>Maria Zemankova</td>
</tr>
<tr>
<td>1996</td>
<td>Jeffrey Ullman</td>
</tr>
<tr>
<td>1999</td>
<td>Raghu Ramakrishnan</td>
</tr>
<tr>
<td>2001</td>
<td>Daniel Rosenkrantz</td>
</tr>
<tr>
<td>2004</td>
<td>Surajit Chaudhuri</td>
</tr>
<tr>
<td>2007</td>
<td>Hans-Jörg Schek</td>
</tr>
<tr>
<td>2010</td>
<td>David Lomet</td>
</tr>
<tr>
<td>2013</td>
<td>H.V. Jagadish</td>
</tr>
<tr>
<td>2016</td>
<td>Samuel Madden</td>
</tr>
<tr>
<td>2019</td>
<td>Ahmed Elmagainrd</td>
</tr>
<tr>
<td>2020</td>
<td>Stratos Idreos</td>
</tr>
<tr>
<td>2020</td>
<td>Dennis Shasha</td>
</tr>
<tr>
<td>2023</td>
<td>K. Selcuk Candan</td>
</tr>
<tr>
<td>1995</td>
<td>Gio Wiederhold</td>
</tr>
<tr>
<td>1997</td>
<td>Avi Silberschatz</td>
</tr>
<tr>
<td>2000</td>
<td>Michael Carey</td>
</tr>
<tr>
<td>2002</td>
<td>Richard Snodgrass</td>
</tr>
<tr>
<td>2005</td>
<td>Hongjun Lu</td>
</tr>
<tr>
<td>2008</td>
<td>Klaus R. Dittrich</td>
</tr>
<tr>
<td>2011</td>
<td>Gerhard Weikum</td>
</tr>
<tr>
<td>2014</td>
<td>Kyu-Young Whang</td>
</tr>
<tr>
<td>2017</td>
<td>Yannis E. Ioannidis</td>
</tr>
<tr>
<td>2020</td>
<td>Philippe Bonnet</td>
</tr>
<tr>
<td>2020</td>
<td>Stefan Manegold</td>
</tr>
<tr>
<td>2021</td>
<td>Divesh Srivastava</td>
</tr>
<tr>
<td>2024</td>
<td>Sihem Amer-Yahia</td>
</tr>
<tr>
<td>1995</td>
<td>Yahiko Kambayashi</td>
</tr>
<tr>
<td>1998</td>
<td>Won Kim</td>
</tr>
<tr>
<td>2000</td>
<td>Laura Haas</td>
</tr>
<tr>
<td>2003</td>
<td>Michael Ley</td>
</tr>
<tr>
<td>2006</td>
<td>Tamer Özsu</td>
</tr>
<tr>
<td>2009</td>
<td>Beng Chin Ooi</td>
</tr>
<tr>
<td>2012</td>
<td>Curtis Dyreson</td>
</tr>
<tr>
<td>2018</td>
<td>Z. Meral Özoçoylu</td>
</tr>
<tr>
<td>2020</td>
<td>Juliana Freire</td>
</tr>
<tr>
<td>2020</td>
<td>Ioana Manolescu</td>
</tr>
<tr>
<td>2022</td>
<td>Christian S. Jensen</td>
</tr>
</tbody>
</table>

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

- **2006** Winner: Gerome Miklau. Honorable Mentions: Marcelo Arenas and Yanlei Diao
- **2007** Winner: Boon Thau Loo. Honorable Mentions: Xifeng Yan and Martin Theobald
- **2008** Winner: Ariel Fuxman. Honorable Mentions: Cong Yu and Nilesh Dalvi
- **2009** Winner: Daniel Abadi. Honorable Mentions: Bee-Chung Chen and Ashwin Machanavajjhala
- **2010** Winner: Christopher Ré. Honorable Mentions: Soumyadeb Mitra and Fabian Suchanek
- **2011** Winner: Stratos Idreos. Honorable Mentions: Todd Green and Karl Schnaitterz
- **2012** Winner: Ryan Johnson. Honorable Mention: Bogdan Alexe
- **2013** Winner: Sudipto Das. Honorable Mention: Herodotos Herodotou and Wenchao Zhou
- **2014** Winners: Aditya Parameswaran and Andy Pavlo.
- **2015** Winner: Alexander Thomson. Honorable Mentions: Marina Drosou and Karthik Ramachandra
- **2016** Winner: Paris Koutris. Honorable Mentions: Pinar Tozun and Alvin Cheung
- **2017** Winner: Peter Bailis. Honorable Mention: Immanuel Trummer
- **2018** Winner: Viktor Leis. Honorable Mention: Luis Galárraga and Yongjoo Park
- **2019** Winner: Joy Arulraj. Honorable Mention: Bas Ketsman
- **2020** Winner: Jose Faleiro. Honorable Mention: Silu Huang
- **2022** Winner: Chenggang Wu. Honorable Mentions: Pingcheng Ruan and Kexin Rong
- **2023** Winner: Supun Nakandala. Honorable Mentions: Benjamin Hilprecht and Zongheng Yang
- **2024** Winner: Daniel Kang. Honorable Mentions: Wei Dong, Jiabin Ding, and Yisu Remy Wang

A complete list of all SIGMOD Awards is available at: [https://sigmod.org/sigmod-awards/](https://sigmod.org/sigmod-awards/)

[Last updated: June 1, 2024]
Editor’s Notes

Welcome to the June 2024 issue of the ACM SIGMOD Record!

This issue starts with the Database Principles column presenting an article by Amarilli and Capelli on tractable circuits in database theory. The article discusses the state of the art on designing efficient algorithms for database tasks with the use of tractable circuit representation of Boolean circuits or relations. The authors argue that circuits enable a unifying view of seemingly disparate results, providing a modular and generic way to solve complex tasks. They also show how the circuit perspective can make it possible to design modular algorithms that leverage existing solver implementations. The article discusses existing results and applications, and concludes with an outline of potential directions for further investigation.

The Surveys column features a contribution by Stonebraker and Pavlo that analyzes the last 20 years of data-model and query-language activities in databases. The authors provide commentary on MapReduce systems, key-value stores, document databases, column stores, and text-search engines, as well as on array, vector, and graph databases. The article offers analysis of the state of the art of these and other systems, and concludes with recommendations and predictions on the future of databases.

The Research Articles column presents an article by Kakaraparthy and Patel. The authors consider the problem of fine-grained hardware profiling, that is of profiling that focuses on the execution of desired sections of the code. The article compares and validates three tools for performing fine-grained hardware profiling on Linux platforms, and provides analysis and recommendations that can help systems practitioners choose the right tools for performing fine-grained profiling at different time scales.

The Reminiscences on Influential Papers column features contributions by Matthias Boehm, Matteo Interlandi, Theodoros Rekatsinas, and Madelon Hulsebos.

The Advice to Mid-Career Researchers column presents a contribution by Angela Bonifati, who shares her experiences with and insights on what can be the long and winding road to mid-career academia. The article provides advice on avoiding research stagnation, striking the right balance between teaching and research, handling time management, going beyond one’s comfort zone, nurturing research collaborations with other teams, mentoring and being mentored, continuous education, working on impactful things and, last but not least, the importance of enjoying what you are doing.

The DBrainstorming column, whose goal is to discuss new and potentially controversial ideas that might be of interest and benefit to the research community, features an article by Hui Li. In the article, the author considers challenges and opportunities in technology-enabled database education, with a motivation to encourage development of learner-friendly technological support in the task of learning various core components of database systems. The article describes the author’s efforts and experiences in this space, and outlines research opportunities in technology-enabled database education.

The Reports column features three contributions. The first article, by Bikakis and colleagues, presents the outcomes of six years of organizing the International Workshop on Big Data Visual Exploration and Analytics (BigVis). The workshop has been held annually since 2018 in conjunction with the International Conference on Extending Database Technology (EDBT), providing opportunities for
researchers in data management and mining, information visualization, and human-computer interaction to revisit traditional problems and solutions in their areas for BigVis purposes. The article summarizes six years of organizing the BigVis workshops, discussing the challenges and presenting the main findings and outcomes.

The second contribution to the Reports column, by Kondylakis and colleagues, reports on the outcomes of the Fifth International Workshop on Health Data Management in the Era of AI (HeDAI), which was co-organized with the EDBT 2023 conference in Ioannina, Greece. The aim of the workshop was to bring together a diverse group of researchers and practitioners interested in developing next-generation solutions for challenging problems in healthcare systems. The article reports on the topics discussed during the workshop, and presents key observations and emerging research directions in this space.

The third contribution to the Reports column, by Amer-Yahia and colleagues, summarizes the 2023 activities of the Diversity, Equity and Inclusion (DEI) initiative, which started as the Diversity/Inclusion initiative in 2020. The article provides a report on the 2023 DEI@DB conferences, discusses the issues of undeclared COIs and PC diversity, and outlines plans going forward.

The issue closes with an Open Forum column, which presents an article by Bhowmick and Hose. The authors share their experiences with, and the results of, installing a novel data-driven PC-chair-in-the-loop PC formation framework for EDBT 2023, with the goal of mitigating some of the challenges brought by traditional PC-formation methods.

On behalf of the SIGMOD Record Editorial board, I hope that you enjoy reading the June 2024 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  
June 2024

Past SIGMOD Record Editors:

Tractable Circuits in Database Theory

Antoine Amarilli
LTCI, Télécom Paris
Institut Polytechnique de Paris
antoine.amarilli@telecom-paris.fr

Florent Capelli
Université d’Artois, CNRS, UMR 8188
Centre de Recherche en Informatique de Lens
capelli@cril.fr

ABSTRACT
This work reviews how database theory uses tractable circuit classes from knowledge compilation. We present relevant query evaluation tasks, and notions of tractable circuits. We then show how these tractable circuits can be used to address database tasks. We first focus on Boolean provenance and its applications for aggregation tasks, in particular probabilistic query evaluation. We study these for Monadic Second Order (MSO) queries on trees, and for safe Conjunctive Queries (CQs) and Union of Conjunctive Queries (UCQs). We also study circuit representations of query answers, and their applications to enumeration tasks: both in the Boolean setting (for MSO) and the multivalued setting (for CQs and UCQs).

1 Introduction
The field of knowledge compilation [62] studies how to efficiently reason about propositional knowledge bases, and how to represent logic formulas as data structures that ensure the tractability of tasks such as Boolean satisfiability. These data structures are often based on decision diagrams such as OBDDs [43] or on restricted versions of Boolean circuits; they often naturally correspond to the trace of an algorithm [77]. Such data structures can then be used to reason on the knowledge base, using different tools depending on the task (satisfiability, model counting, etc.).

These tasks are naturally connected to database problems: e.g., satisfiability may be seen as testing whether a query has an answer, and model counting can be seen as finding its number of answers. The main difference is that knowledge compilation focuses on propositional logic, which can be seen as the minimal setting where the techniques can be applied. It is hence only natural to use these techniques and tools from knowledge compilation and adapt them to the setting of databases. In this paper, we review this line of work which designs efficient algorithms for database tasks using tractable circuit representations of Boolean functions or relations.

We identify two main ways in which circuit classes are used in database theory. The first is via Boolean functions that naturally arise in databases, such as Boolean provenance, or answer functions that represent the output of queries. Knowledge compilation proposes efficient formalisms in which we can represent such Boolean functions and ensure that some tasks are tractable over them. Thus, if we can efficiently compute the Boolean provenance or answer functions and represent them in a tractable circuit class, then we can show some theoretical tractability results, and/or use practical tools. This approach has been used across different domains, e.g., query evaluation on probabilistic databases [80], enumeration for monadic second-order queries [8], and Shapley value computation [63]; we will review these works and more. Conversely, if we can show that the output to some problems cannot be tractably represented as a circuit in a given class, then we show that these problems cannot efficiently be solved by algorithms whose trace falls in that class: this approach is followed by [28] among others.

The second family of applications of circuit classes in databases is under the guise of factorized databases [114], in which relations are succinctly represented as circuits with restricted operators from relational algebra. This notion was introduced to understand various tasks on the answer set of restricted kinds of Conjunctive Queries (CQs) [26, 115, 135]. As we will explain, factorized databases can be seen as natural generalizations of the circuits used in knowledge compilation, going from the Boolean domain to a multivalued domain [110]. We will then explain how such circuits can be used to recover known results.

In both these settings, we see the strengths of circuits: they give a unifying view on scattered results, and they provide a modular and generic way to solve complex tasks. Namely, they make it possible to design query algorithms whose only task is to produce a circuit that falls in a given class. The circuit can then be fed to existing algorithms and software implementations which can solve various problems on the circuit independently from how it was built.

The paper is organized as follows. Section 2 first reviews how Boolean functions naturally occur in database
 tasks. We focus on aggregation tasks that we relate to Boolean provenance, and enumeration tasks that we relate to answer functions. We then study in Section 3 how to efficiently represent such functions: we review known circuit classes from knowledge compilation and the associated tractability results and software implementations. We then explain how the tractable circuit classes of Section 3 allow us to address the tasks of Section 2: we start in Section 4 with Monadic Second-Order (MSO) queries on trees, then study aggregation tasks for CQs and Union of CQs (UCQs) in Section 5. We then move in Section 6 to the multivalued perspective: we see how tractable circuit classes can be seen as factorized relations to succinctly represent the answer set of CQs. We close with an overview of other directions and questions for future research in Section 7.

2 Database Tasks and Boolean Functions

We first show in this section how various database tasks can be expressed in terms of Boolean functions, to be later represented by circuits. We focus on two main kinds of tasks. First, we present tasks expressed in terms of the Boolean provenance of a query on the input data. We focus on aggregation tasks on the provenance, which intuitively involve some form of counting. Second, we will study how tasks can be expressed in terms of Boolean functions that capture the answer of queries, e.g., in MSO. We focus there on enumeration tasks, which ask for the computation of witnesses. We last sketch other tasks related to circuits that we do not investigate in detail.

2.1 Boolean Provenance

We first recall some standard terminology. Provenance is defined over (relational) instances, which are sets of facts over a signature. Formally, a signature consists of a set of relation names with an associated arity, a fact for a relation name R of arity n is an expression of the form R(a1, ..., an) with a1, ..., an being values, and an instance is a set of facts. A Boolean query Q is then a function that maps instances ∆ to a Boolean value indicating whether the query is satisfied by ∆. The Boolean provenance of Q over an instance ∆ then simply describes the truth status of Q on all subinstances of ∆, i.e., all subsets of its facts. Formally:

**Definition 2.1.** Let ∆ be an instance and Q be a Boolean query. The provenance of Q on ∆ is the Boolean function from 2^∆ to {0, 1} that maps each subinstance ∆’ ∈ 2^∆ of ∆ to 1 if Q is true on ∆’ and to 0 otherwise.

As a Boolean function, provenance can be represented in multiple ways, e.g., as Boolean formulas like in the example below, or as Boolean circuits (see Section 3).

**Example 2.2.** Let ∆ = {R(a), R(a’), S(b)} be an instance and Q be the Boolean CQ □xy R(x), S(y) asking for the presence of an R-fact and of an S-fact. The provenance of Q on ∆ is the function denoted by the Boolean formula (R(a) ∨ R(a’)) ∧ S(b).

Semiring provenance and more general semirings. Boolean provenance as defined here is the special case of semiring provenance [73] for the semiring Bool[X] of Boolean functions. One advantage of Boolean provenance is that it is purely semantic, i.e., it considers the query as a black-box. By contrast, provenance for more general semirings like N[X] often depends on how the query is executed. We note that circuit notions have also been defined for such general provenance semirings [12, 64], in connection with arithmetic circuits; but we only discuss Boolean provenance from now on.

Now that we have defined Boolean provenance for queries, we explain how database tasks can be rephrased in terms of Boolean provenance. We will focus here on aggregation tasks, where we intuitively want to perform a kind of counting over subsets of the input instance.

Uniform reliability. The simplest aggregation task is uniform reliability (UR) for a Boolean query Q: given an instance ∆, count the subinstances of ∆ that satisfy Q. This can be solved via model counting. Formally:

**Definition 2.3.** Let φ be a Boolean function over variables X. The model counting problem (#SAT) for φ asks how many valuations of X satisfy φ. Formally, for Y ∈ 2^X, we write ν_Y for the Boolean valuation over X that maps x ∈ X to 1 if x ∈ Y and to 0 otherwise. #SAT is the problem of computing #φ := |{Y ∈ 2^X | ν_Y satisfies φ}|.

Hence, if φ if the provenance of Q on an instance ∆, then the answer to UR for Q is the answer to #SAT for φ.

Probabilistic query evaluation. A generalization of UR is probabilistic query evaluation (PQE) for Q. In this setting, we are given a so-called tuple-independent database (TID): it consists of an instance ∆ with a function π : ∆ → [0, 1] mapping each fact of ∆ to a rational probability value. We assume independence across facts, and consider the product probability distribution on subinstances of ∆, where the probability of ∆’ ⊆ ∆ is π(∆’) := Π_{F ∈ ∆’} π(F) × Π_{F ∈ ∆ \ ∆’} (1 − π(F)). We want to know the total probability of the subinstances of ∆ that satisfy Q, namely, Σ_{∆’ ⊆ ∆} π(∆’). This problem can be solved via weighted model counting:

**Definition 2.4.** Let (K, ⊕, +) be a semiring, and let φ be a Boolean function over X. We write Lit(X) the set of literals over X, i.e., X ∪ {¬x | x ∈ X}. Let w : Lit(X) → K be a weight function giving a weight in K to each literal over X. The weight of a Boolean valuation v : X → {0, 1} is then w(v) := Σ_{x∈X} w(x) ×
\( \bigotimes_{x \in X} w(x) = 0 \). Then, the weighted model counting problem (WMC) for \( \phi \) is to compute the total weight of satisfying valuations, i.e., \( \sum \nu \text{satisfies } \phi \cdot w(\nu) \).

Note that model counting for \( \phi \) reduces to WMC with a weight of 1 for each literal. We then have that PQE for \( Q \) on a TID \((\mathbb{D}, \pi)\) amounts to WMC for the provenance \( \phi \) of \( Q \) on \( \mathbb{D} \) in the semiring of rationals \((\mathbb{Q}, \times, +)\) with weights given by \( \pi \).

As the tasks UR and PQE are often intractable, we may prefer to study approximate model counting (ApproxMC) and its weighted variant (ApproxWMC). In these variants, instead of solving the problem exactly, we wish to compute an approximation of the model count or of the probability. We focus on multiplicative approximations: given an error \( \varepsilon > 0 \), we must compute an approximation \( \hat{x} \) of the true value \( x \) that ensures \( (1 - \varepsilon)x \leq \hat{x} \leq (1 + \varepsilon)x \). We often allow approximation algorithms to be randomized: then, the output of the algorithm must be a correct approximation with probability at least \( 2/3 \). We say that we have a fully polynomial-time approximation scheme (FPRAS) for Approx(W)MC when we have a randomized algorithm to compute a multiplicative approximation of the count with running time polynomial in the input instance and in the inverse of the desired error \( \varepsilon \).

**Shapley values.** Another aggregation task is the computation of Shapley values, which can be used to quantify the contribution of a fact to making the query true [35]. In this setting, we fix a query \( Q \) and we are given as input an instance \( \mathbb{D} \), which is partitioned between so-called exogenous facts \( \mathbb{D}_e \), which are always present, and endogenous facts \( \mathbb{D}_o \), which may be present or absent. The Shapley value is then an aggregate over all subinstances of \( \mathbb{D} \) that contain all the exogenous facts of \( \mathbb{D}_e \). We omit the formal definition of the Shapley value; see [35]. Note that the Shapley value reduces in particular to counting how many subsets of endogenous facts of a given cardinality satisfy \( Q \) together with the exogenous facts [100].

The computation of Shapley values can be posed on the Boolean provenance \( \phi \) of the query \( Q \), but imposing that all exogenous facts are kept. This amounts to partial evaluation of \( \phi \), namely, setting the variables of \( \mathbb{D}_e \) to 1.

### 2.2 Boolean Answer Functions

We now move on from Boolean provenance to a different kind of Boolean functions, this time defined to represent the results of queries. It will be easier to define these functions for queries with one free second-order variable, i.e., queries that return subset of domain elements as answers, in particular queries expressed in MSO. We will revisit this perspective in Section 6 to work on more conventional query results that consist of relations.

Formally, the semantics of a query \( Q(X) \) with a free second-order variable is that \( X \) stands for a set of elements taken from the active domain: given an instance \( \mathbb{D} \), letting \( D \) be the domain of \( \mathbb{D} \) (the set of elements occurring in facts of \( \mathbb{D} \), the answers of \( Q \) on \( \mathbb{D} \) is the set of subsets \( A \subseteq D \) such that \( Q(X := A) \) is true.

**Example 2.5.** On a signature with one binary relation \( R \) and one unary relation \( V \), consider the query \( Q(X) := \neg(yz \in R(z,y) \vee R(z,y)) \land V(y) \land V(z) \land X(y) \land X(z) \). Then, given an instance \( D \) with domain \( D \), we have that \( Q(X := A) \) holds precisely when \( A \subseteq D \) is an independent set of the graph represented by \( D \) with vertices coded by \( V \) and edges coded by \( R \).

In this setting, for a query \( Q \), given an instance \( \mathbb{D} \) with domain \( D \), we can naturally define a Boolean function \( \phi \) describing the answers of \( Q \) on \( \mathbb{D} \), called the answer function of \( \phi \) on \( \mathbb{D} \). Formally, the variables of the answer function are the domain elements in \( D \), and it is satisfied by a Boolean valuation \( v : D \rightarrow \{0,1\} \) precisely when the set \( A_v := \{a \in D \mid v(a) = 1\} \) is an answer to \( Q \).

For a query \( Q(X_1, \ldots, X_k) \) with multiple second-order variables, we can also define the answer function of \( Q \) on an instance \( \mathbb{D} \) with domain \( D \): it is a Boolean function with variable set \( D \times [k] \), satisfied by the valuations \( v : D \times [k] \rightarrow \{0,1\} \) such that the tuple of the sets \( A_i := \{a \in D \mid v((a, i)) = 1\} \) for \( i \in [k] \) is an answer to \( Q \).

We now study how problems over the set of answers of a query \( Q \) can be posed over their Boolean answer function. One example is answer counting: the number of answers of \( Q \) over an instance \( \mathbb{D} \) is the number of satisfying assignments of the answer function of \( Q \) on \( \mathbb{D} \). However, in this section, we focus on enumeration tasks: given \( \mathbb{D} \), we must compute a set of solutions of \( Q \) over \( \mathbb{D} \).

**Finding and enumerating satisfying valuations.** The simplest task on an answer function \( \phi \) is simply to decide if the query has an answer, i.e., if \( \phi \) has a satisfying valuation (called satisfiability or SAT); and to compute one if it exists. A variant is uniform sampling, i.e., a randomized algorithm that must produce one satisfying valuation uniformly at random. We note that the task can be made approximate by allowing a failure probability or doing near-uniform sampling.

However, it may also be important to compute all satisfying valuations. One challenge to formalize the task is that the number of answers to produce can be large, which makes it difficult to measure efficiency.

**Example 2.6.** Continuing Example 2.5, consider the task of enumerating all query answers, i.e., all independent sets of the input graph. A naive algorithm to compute the answers is to list each possible subset \( A \) and test if it is an independent set. The naive algorithm takes time \( \Omega(2^{|V|}) \), but it is hard to improve on this, because the worst-case complexity of the task is also \( \Omega(2^{|V|}) \). Indeed,
the running time cannot be less than the number of answers to produce, and given an instance \( D \) with \( n \) facts, there may be up to \( 2^n \) answers (e.g., for \( n \) isolated vertices). What we intuitively want is to beat \( \Omega(2^n) \) on the instances where the answer set is small; or to produce the first few solutions faster than \( \Omega(2^n) \).

This example illustrates why we study algorithms in terms of the output size. One way is enumeration algorithms, which have been studied in many contexts (see, e.g., [133]). In data management, enumeration algorithms distinguish two phases: a preprocessing phase to perform some precomputations, and an enumeration phase to produce all solutions with small delay between consecutive solutions. We omit the formal definition of enumeration algorithms; see, e.g., [120].

**Definition 2.7.** Given a Boolean function \( \phi \) on variables \( X \), the task of enumerating satisfying valuations (Enum) of \( \phi \) asks us to efficiently produce the list of all satisfying valuations, with small preprocessing time and small delay between any two successive valuations. Note that, in the enumeration, we may write each valuation \( \nu : X \to \{0, 1\} \) as the set \( \{x \in X \mid \nu(x) = 1\} \): this may be more concise when the Hamming weight of \( \nu \) is small.

Thus, for the database tasks presented above, we can devise enumeration algorithms for queries via answer functions: compute a representation of the answer function, then enumerate its satisfying valuations.

**Ranked enumeration and ranked access.** Beyond the mere listing of satisfying valuations, it is often relevant to list them in a specific order, or to find the top-\( k \) satisfying valuations in this order. We also want to do ranked access: quickly access the \( i \)-th solution in this order, which generalizes tasks such as quantile computation [49].

**Definition 2.8.** Given a Boolean function \( \phi \) on variables \( X \), given a total order \( \prec \) defined on the set \( 2^X \) of valuations of \( X \), the task of ranked enumeration of satisfying valuations of \( \phi \) under \( \prec \) is to enumerate the satisfying valuations of \( \phi \) in the order given by \( \prec \). The task of computing the top-\( k \) satisfying valuations, given \( k \in \mathbb{N} \), is to compute the \( k \) satisfying valuations that are first according to \( \prec \). The task of ranked access is to return, given \( i \in \mathbb{N} \), the \( i \)-th satisfying valuation in the order, or fail if the number of satisfying valuations is less than \( i \).

There are two main ways in which we can represent an order on valuations without materializing its comparability graph explicitly. One first method is lexicographic orders: we are given a total order \( \prec' \) on the set \( X \) of variables, and extend it lexicographically to valuations. Formally, given two valuations \( \nu, \nu' \in 2^X \), we have \( \nu < \nu' \) if, letting \( x \) be the smallest variable for \( \prec' \) such that \( \nu(x) \neq \nu'(x) \), we have \( \nu(x) = 0 \) and \( \nu'(x) = 1 \). The second method is weights: we fix a semigroup \( (K, \otimes) \), and define the weight \( w(\nu) \) of a valuation \( \nu \) for a weight function \( w \) like in Definition 2.4. Now, if we pick some total order \( \prec' \) on semigroup values, we can sort valuations according to their weight in \( K \): we may typically assume that \( \prec' \) is compatible in some sense with the semigroup law (e.g., subset-monotonicity [7, 128]), and often impose some tie-breaking rule to make the order total. In particular, when \( K = \mathbb{Q} \) and weights are probabilities (like for PQE), we can compute the most probable satisfying valuation or the most probable falsifying valuation.

### 2.3 Other Tasks via Boolean Functions

Boolean functions can be used for many data management tasks beyond the ones presented so far. We close the section by briefly alluding to other such tasks.

First, the notion of database repairs, specifically in the case of maximal subset repairs, can be studied via Boolean provenance by looking at the maximal satisfying valuations of the provenance. Thus, the problem of subset repair counting [44, 45, 93] can be seen as the aggregation task of counting the maximal satisfying valuations of the provenance of a certain Boolean query; and the task of enumerating subset repairs [93] amounts to the enumeration task of listing the maximal satisfying valuations. A connection could also be phrased for consistent query answering [33, 34].

Second, more generally, the setting of reverse data management [103] provides other examples of problems that can be phrased in terms of Boolean provenance. These problems can consider ways to delete facts in an input instance in order to satisfy a property, for instance minimum witness [76, 104] (delete as many facts as possible while making the query true) or resilience [71, 102] (delete as few facts as possible to make the query false). These problems can be posed on the Boolean provenance: find satisfying valuations of small Hamming weight, or falsifying valuations of large Hamming weight.

Third, in the setting of uncertain databases [1], we may consider instances where some facts are uncertain, i.e., they may be present or absent; and other facts are certain and are necessarily present. These correspond to endogenous and exogenous facts in the case of Shapley value computation. We can then study tasks such as certain query answers [79], i.e., is the query always true on all possible worlds of an uncertain instance, i.e., is the Boolean provenance tautological. A related task is query-guided uncertainty resolution [65] which interactively determines the truth value of the query, by probing individual facts while optimizing, e.g., the worst-case decision tree height. This amounts to stochastic Boolean function evaluation [5] of the Boolean provenance.

Fourth, many of the tasks defined so far can be extended
to incremental maintenance: compute the result (or its representation, e.g., as an enumeration index), and maintain this result efficiently under updates to the data. In general, circuit representations are amenable to incremental maintenance: we can reevaluate the circuit on a different valuation whenever the data changes. However, this idea assumes that the initial set of facts never grows, i.e., facts can only be removed or added back.

3 Knowledge Compilation

We have seen in Section 2 that many database tasks can be rephrased to problems on Boolean functions. However, these reductions do not directly give tractable algorithms, because many of these problems are hard. For example, SAT is NP-complete already for Conjunctive Normal Form formulas (CNFs) \( \phi \). Computing \( \# \phi \) is even harder: it is \#P-complete [129], and NP-hard to approximate to a \( 2^{\epsilon} \)-factor for every \( \epsilon > 0 \), even if \( \phi \) has no negation and clauses of size at most 2 [118]. However, tractability can hold for other representations of Boolean functions. For instance, for a Disjunctive Normal Form (DNF) formula \( \phi \), finding a satisfying valuation is easy. While computing \( \# \phi \) is also \#P-complete for DNFs, we can approximate it to a \( (1 + \epsilon) \)-factor with high probability via a Monte-Carlo-based FPRAS, namely, the Karp-Luby algorithm [86].

Thus, one approach to make the tasks of Section 2 tractable is to build representations of the Boolean functions that are tractable for the task at hand. This modular approach often makes it possible to show several tractability results at once, and allows us to use efficient practical implementations (see Section 3.3).

To achieve this, we turn to the field of knowledge compilation [62], which studies tractable representations for Boolean functions and their properties. We review definitions, results, and tools from this field in this section, which we will use in Sections 4 to 6 for various database tasks featuring aggregation and enumeration.

3.1 DNNF Circuits

Most representations of Boolean functions studied in the literature are restricted forms of Boolean circuits. The use of circuits allows for efficient sharing and factorization, while the restrictions ensure tractability. One of the most general such representation is Decomposable Negation Normal Form (DNNF) circuits [60]. We start by defining Boolean circuits before introducing DNNFs.

Circuits and NNF circuits. Formally, a Boolean circuit \( C \) on variables \( X \) is a Directed Acyclic Graph together with a distinguished node called the output. The nodes with indegree 0, called the inputs of the circuit, are each labeled either with a constant 1 or 0 or with a variable \( x \in X \). The internal nodes of \( C \), called gates, are labeled by \( \land, \lor \) or \( \neg \). Given a gate \( v \), every gate \( w \) that has an edge to \( v \) is called an input of \( v \). Every \( \neg \)-gate has exactly one input. Given a gate \( v \) of \( C \), we let \( \text{var}(v) \) be the set of variables \( x \in X \) that appear in the subcircuit rooted in \( C \), that is, we can reach \( v \) by a directed path from an input of \( C \) labeled by \( x \). The size \(|C|\) of a Boolean circuit is the number of edges in the DAG.

Every gate \( g \) of \( C \) computes a Boolean function \( f_g \) on \( \text{var}(g) \): given a valuation \( \nu \): \( \text{var}(g) \to \{0, 1\} \), the value of \( f_g \) is defined by setting the inputs of the circuit with \( \nu \) and evaluating internal gates inductively up to \( g \). The Boolean function on \( X \) computed by \( C \) is then the function \( f_o \) on \( \text{var}(o) \) for \( o \) the output of \( C \), extended to \( X \) by allowing the variables of \( X \setminus \text{var}(o) \) to take any value.

We focus on Boolean circuits in Negation Normal Form (NNF) where the input of every negation gate is an input of the circuit; alternatively, NNF circuits are circuits built on literals using \( \land \) and \( \lor \) on internal gates. See Figure 1 for an example of an NNF circuit. Any Boolean circuit can be put in NNF in linear time using De Morgan’s laws iteratively to push negations down.

DNNF circuits. We now move to restrictions on NNF circuits aimed at enforcing tractability of problems (e.g., SAT). Imposing NNF is not sufficient for this: as we explained, it is essentially without loss of generality. Intuitively, what makes SAT hard is that finding satisfying valuations \( \nu_1 \) for \( f_1 \) and \( \nu_2 \) for \( f_2 \) does not help to find one for \( f_1 \land f_2 \), because \( \nu_1 \) and \( \nu_2 \) may be inconsistent on their shared variables. To avoid this, we use decomposable \( \land \)-gates. A \( \land \)-gate \( g \) is decomposable if for every pair of distinct inputs \( g_1, g_2 \) of \( g \), we have \( \text{var}(g_1) \cap \text{var}(g_2) = \emptyset \). A Decomposable Negation Normal Form (DNNF) circuit is an NNF circuit \( C \) where every \( \land \)-gate is decomposable. Observe that decomposability makes SAT tractable: we can propagate satisfying valuations upwards in \( C \), building satisfying valuations for decomposable \( \land \)-gates by concatenating satisfying valuations of their inputs. The circuit depicted on Figure 1 is a DNNF.

Note that, in database terms, a decomposable \( \land \)-gate \( g \) is a Cartesian product: if we see the Boolean function

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 1: A DNNF and its satisfying valuations.
computed by each input \( g_i \) of \( g \) as a relation \( R_i \) over attributes \( \text{var}(g_i) \) and domain \( \{0, 1\} \), then the relation computed by \( g \) is \( R_1 \times R_2 \). We revisit this in Section 6.

DNNF circuits clearly generalize DNF formulas, while retaining some of their tractability. Indeed, we can solve SAT on a DNNF circuit \( C \) in time \( O(|C|) \), and enumerate the satisfying assignments with a delay of \( O(n|C|) \) where \( n = |X| \). DNNFs are also closed under partial evaluation, called conditioning in knowledge compilation [62]; given a DNNF circuit \( C \) on variables \( X \) computing \( f \) and a partial valuation \( v \) on variables \( Y \subseteq X \), we can build in time \( O(|C|) \) a DNNF computing \( f|_Y \). Indeed, simply replace each node labeled with a variable \( x \in Y \) by the constant \( v(x) \): this does not affect decomposability.

### 3.2 Restrictions on DNNF Circuits

We now introduce additional restrictions to make more tasks tractable. Figure 2 and Table 1 summarize the circuit classes studied and the complexity of tasks.

**Determinism.** Unlike enumeration, aggregation tasks such as \#SAT are not tractable on DNNFs, as they are already hard on DNFs. Intuitively, \#SAT is hard because \#1 and \#2 does not give \#(1 \lor 2): indeed, \#1 and \#2 may be sharing some of their satisfying valuations. Determinism [59] is a way to make such tasks tractable.

In a Boolean circuit, a \( \lor \)-gate \( g \) is said to be deterministic if for every pair of distinct inputs \( g_1, g_2 \) of \( g \), the function \( f_1 \land f_2 \) is not satisfiable. In other words, if \( v \) is a satisfying valuation of \( f_1 \) then there is exactly one input \( g_i \) of \( g \) such that \( v|_{\text{var}(g_i)} \) is a satisfying valuation of \( g_i \). A deterministic Decomposable Negation Normal Form (d-DNNF) circuit is a DNNF where every \( \lor \)-gate is deterministic. The DNNF in Figure 1 is not a d-DNNF.

Determinism makes it possible to count satisfiable valuations bottom-up in time \( O(|C|) \) (if arithmetic operations take constant time). Indeed, we can compute bottom-up in \( C \) the number of satisfying valuations of each gate: we have \#(1 \land f_2) = #1 \land #f_2 thanks to decomposability, and #1 \land #f_2 = #1 \land #f_2 thanks to determinism. Determinism also ensures the tractability of WMC for weights in any semiring \((K, \oplus, \otimes)\) [94], using only \( O(|C|) \) semiring operations, provided that the circuit is smooth, i.e., we have \( \text{var}(g') = \text{var}(g) \) for every input \( g' \) of a \( \lor \)-gate \( g \). Otherwise, an \(|\text{var}(C)|\) factor is usually needed to make the circuit smooth, which can sometimes be improved [124]. Further, when solving WMC with weights defined in a ring, it suffices to impose determinism and decomposability without NNF. This leads to d-Ds [107].

Determinism can also help for enumeration tasks: on a d-DNNF \( C \), we can efficiently enumerate satisfying valuations in increasing order of a semiring weighting of the literals [7, 36], or uniformly sample satisfying valuations in \( O(\text{depth}(C) \times |\text{var}(C)|) \) after a preprocessing in \( O(|C|) \) [123].

**Decision.** One limitation of determinism is that it is a semantic property that may be hard to check on arbitrary circuits. Indeed, given a DNNF \( C \) and a \( \land \)-gate \( g \) of \( C \), it is coNP-hard to check whether \( g \) is deterministic. This is not always a problem as circuits are sometimes deterministic by construction; but sometimes we prefer to enforce a sufficient condition for determinism, called decision gates, which is easy to check syntactically. A decision gate intuitively tests the value of one variable \( x \in X \) before proceeding further. More precisely, a \( \land \)-gate \( g \) is a decision gate on \( x \) if the inputs of \( g \) are exactly two \( \land \)-gates \( g_0, g_1 \), where \( g_0 \) has one input labeled by \( \neg x \) and \( g_1 \) has one input labeled by \( x \). In other words, \( g \) is of the form \((\neg x \land g_0) \lor (x \land g_1)\); note that the literals \( \neg x \) and \( x \) ensure that \( g \) is deterministic. For instance, the leftmost \( \land \)-gate of Figure 1 is a decision gate on \( x_3 \). A decision DNNF (dec-DNNF) is a DNNF where every \( \land \)-gate is a decision gate. All dec-DNNFs are d-DNNFs, and their depth can always be reduced to \( O(|X|) \) in linear time.

A circuit that only has decision gates is often called a binary decision diagram or a branching program: this includes in particular Ordered Binary Decision Diagrams (OBDDs) [43]. OBDDs can be seen as DNNFs having only decision gates and having an order \( < \) on \( X \) such that decision gates testing a variable \( x_i \) must come before gates testing variables \( x_j \) whenever \( x_i < x_j \). OBDDs without this order requirement but with the analogue of decomposability are called Free Binary Decision Diagrams (FBDDs). See [6] for a recent survey on the relationships between circuit classes and diagram classes.

**Structuredness.** Structuredness [116] is a generalization of orders in OBDD tailored for DNNFs. For a variable set \( X \), a variable tree or \( v \)-tree for \( X \) is a full binary tree whose leaves are in one-to-one correspondence with \( X \). Given a DNNF circuit \( C \) on variables \( X \) and a \( v \)-tree for \( X \), we say that a \( \land \)-gate \( g \) in \( C \) respects \( T \) if \( g \) has exactly two inputs \( g_1, g_2 \) and if there is a node \( t \) of \( T \) with inputs \( t_1, t_2 \) such that \( \text{var}(g_1) \subseteq \text{var}(t_1) \) and \( \text{var}(g_2) \subseteq \text{var}(t_2) \) where \( \text{var}(t_j) \) is the set of variables in the leaves of the subtree of \( T \) rooted at \( t_j \). A DNNF \( C \) respects \( T \) if every \( \land \)-gate of \( C \) does. We call \( C \) structured (denoted as SDNNF) if it respects some \( v \)-tree.

In essence, structuredness restricts how \( \land \)-gates are allowed to split the variables. It naturally appears in many algorithms building DNNFs from other Boolean function representations, e.g., building a DNNF from a bounded-treewidth circuit [14, 38]. Structuredness unlocks two new results: first, there is an FPRAS to approximately count the satisfying valuations of (non-deterministic) SDNNFs [23]. Second, enumerating the satisfying valuations of a d-SDNNF \( C \) can be done with preprocessing \( O(|C|) \) and output-linear delay \( O(|X|) \) [8].
Table 1: Tractable tasks for circuit classes. The input circuit $C$ is of size $s$, depth $d$, and $n$ variables; $k$ is the number of solutions to output (for Enum and Sampling).

<table>
<thead>
<tr>
<th>Task</th>
<th>Circuit class</th>
<th>Complexity</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMC</td>
<td>d-DNNF</td>
<td>$O(ns)$</td>
<td>[94]</td>
</tr>
<tr>
<td>WMC in ring</td>
<td>d-D</td>
<td>$O(ns)$</td>
<td>[107]</td>
</tr>
<tr>
<td>ApproxWMC</td>
<td>DNF</td>
<td>FPRAS</td>
<td>[86, 127]</td>
</tr>
<tr>
<td>ApproxWMC</td>
<td>SDNNF</td>
<td>FPRAS</td>
<td>[23]</td>
</tr>
<tr>
<td>Enum</td>
<td>d-SDNNF</td>
<td>$O(s + nk)$</td>
<td>[8]</td>
</tr>
<tr>
<td>Sampling</td>
<td>d-DNNF</td>
<td>$O(s + nk)$</td>
<td>[123]</td>
</tr>
</tbody>
</table>

3.3 Tools

An attractive feature of knowledge compilation is that many of its algorithms have implementations (at least experimental ones). We now survey these practical tools.

There are two main families of knowledge compilation tools. First, top-down tools are based on a generalization of the DPLL algorithm, known as exhaustive DPLL [24]. It is based on a recursive procedure originally devised for solving #SAT but which implicitly compiles into dec-DNNF formulas [77]. It compiles a CNF formula $\phi$ as follows: pick some variable $x$, recursively compile a circuit with gates $g_0$ computing $\phi[x := 0]$ and $g_1$ computing $\phi[x := 1]$, and add a decision variable $g$ on $x$ connected to $g_0$ and $g_1$. When $\phi = \phi_1 \land \phi_2$ with $\phi_1, \phi_2$ on disjoint variables, the algorithm compiles them independently and connects the two circuits with a decomposable $\land$-gate. Efficient algorithms rely on two main ingredients: (1) a caching mechanism that remembers previously-computed formulas to reuse them elsewhere in the circuit; and (2) a heuristic to choose variables to branch on, e.g., to break down the formula into smaller connected components. The knowledge compiler d4 [97, 98] implements this algorithm, trying, e.g., to find balanced cutsets in the formula and performing oracle calls to SAT solvers to cut unsatisfiable branches. The #SAT solver SharpSat-TD [95, 96] uses a heuristic that is guided by a tree decomposition computed via the FlowCutter algorithm [74, 126]. It has recently been modified into a knowledge compiler [91, 92].

Second, bottom-up knowledge compilers take a CNF formula $\phi$ and build circuits computing each clause of $\phi$, before combining them into a bigger circuit computing $\phi$. This works if the target circuit class efficiently supports conjunction; combine two circuits $C_1, C_2$ into a circuit computing $C_1 \land C_2$. Libraries such as CuDD [125] use this approach to compile CNFs to OBDDs. Further, the knowledge compiler SDD [58, 61] compiles into a subclass of structured d-DNNFs: these, unlike OBDDs [39, 117], can efficiently handle any bounded-treewidth CNF formula.

4 Circuits for MSO Queries over Trees

We have seen in Section 2 how database tasks can be expressed in terms of Boolean functions, and seen in Section 3 how such functions can be represented as tractable circuits. We now start surveying how circuit-based methods can be used to solve database tasks. We start in this section by queries in monadic second-order logic (MSO) over tree-shaped data. This covers in particular the evaluation of word automata over textual documents, including the so-called document spanners [68]; and the evaluation of MSO queries over bounded-treewidth data via Courcelle’s theorem [55]. We first give brief definitions of this setting, then study PQE for Boolean MSO queries and enumeration tasks for MSO queries with free variables. Throughout this section we adopt the data complexity perspective, i.e., the MSO query is always fixed, and the complexity is always a function of the input instance. We will study aggregation tasks (specifically, PQE) and enumeration tasks for MSO queries over trees. In both settings, the results will proceed by constructing tractable circuits to represent the Boolean provenance (for PQE) or the answer function (for enumeration).

Preliminaries. We consider queries over $\Sigma$-trees, or simply trees, which consist of nodes labeled with a fixed alphabet $\Sigma$. We assume trees to be rooted, ordered, binary, and full. The queries that we run over trees are expressed in MSO: this language extends first-order logic with quantification over sets, on a signature where we can test the label of tree nodes and the child and parent relationship between tree nodes. For example, we can express in MSO that there are two incomparable nodes with a certain label, or that nodes with a certain label are totally ordered by the descendant relation. An MSO query $Q$ may be Boolean, in which case it can equivalently be expressed as a tree automaton; or it may have free variables. We can always assume without loss of generality that each free variable $X$ is second-order, because $Q$ can assert if necessary that $X$ must be a singleton.
Aggregation tasks. We start with aggregation tasks for Boolean MSO queries $Q$, specifically probabilistic query evaluation (PQE). The PQE problem for $Q$ asks for the probability that $Q$ is satisfied on an input probabilistic tree, as we will define shortly. From there, by [12], using Courcelle’s theorem [55], these results generalize to PQE over tuple-independent databases of bounded treewidth; and bounded treewidth is in some sense the most general condition that ensures tractability [13, 18].

Formally, to define probabilistic trees, we distinguish a default label $e \in \Sigma$; a probabilistic tree then consists of a $\Sigma$-tree $T$ with a function $\pi$ giving a probability $\pi(n)$ to each tree node $n$ of $T$. The semantics is that $T = (T, \pi)$ represents a probability distribution on possible worlds which are $\Sigma$-trees with same skeleton as $T$: each node $n$ either keeps its label in $T$ with probability $\pi(n)$, or “reverts” to the default label $e$ with probability $1 - \pi(n)$, all these choices being independent. PQE for $Q$ on $T$ then asks for the total probability of the possible worlds of $T$ that satisfy $Q$.

Following earlier results on probabilistic XML [54], it is then known [12] that PQE for $Q$ can be computed in PTIME in the input $T$. Specifically, we can define as in Section 2 the provenance of $Q$ on $T$ as the Boolean function defined on the nodes of $T$ that maps each valuation to 0 or 1 depending on whether the corresponding possible world satisfies $Q$. One can then show, representing $Q$ as a tree automaton $A$, that the provenance can be computed in time $O(|A| \times |T|)$ as an SDNNF whose v-tree follows $T$. There is also a correspondence between properties of $A$ and of $C$, e.g., if $A$ is unambiguous then $C$ is a d-SDNNF [6]. The tractability of WMC for d-DNNF then implies that PQE for any fixed MSO query on probabilistic trees can be solved in PTIME.

Enumeration tasks. We now study MSO queries with free second-order variables $Q(X_1, \ldots, X_l)$, again running over trees. We first consider the task of enumerating query answers, again in data complexity. It is known that, in this setting, the answers to $Q$ on an input tree $T$ can be enumerated with preprocessing in $O(|T|)$, and with delay which is output-linear, i.e., which depends only on the size of each produced answer. This was shown first by Bagan [25], then by Kazana and Segoufin (only for free first-order variables) [88]. This result can be recaptured via knowledge compilation: a variant of provenance computation allows us to obtain d-SDNNF representations of the answer functions of MSO queries [8]. We can then enumerate the results of the query with linear preprocessing and output-linear delay with an algorithm to enumerate the satisfying valuations for this circuit class [8].

Similar results have also been shown for document spanners, which essentially amounts to evaluating MSO queries specified by automata over words, in particular via algorithms that also ensure tractability in the input automaton [10, 11]. In more expressive settings, the work of Muñoz and Riveros [108, 109] uses so-called enumerable compact sets, which resemble d-DNNF circuits, to achieve efficient enumeration on nested documents and SLP-compressed documents; and these are also used in [15] to enumerate the results of annotation grammars.

Enumeration algorithms for MSO over words and trees have been further extended to ranked enumeration: first by Bourhis et al. [37] on words, with weights defined by MSO cost functions; then on trees [7], with weights defined on partial assignments by so-called subset-monotone ranking functions. The latter work explicitly uses the circuit approach via smooth multivalued d-DNNF circuits.

We last mention the incremental maintenance of enumeration structures for MSO queries on trees. The point of such structures is to enumerate the answers of MSO queries while supporting updates to the underlying data; we want to handle updates (and restart the enumeration) without re-running the preprocessing phase from scratch. Note that this generalizes the task of incrementally maintaining Boolean MSO queries [27]. In this setting, for relabeling updates to the underlying tree, the best known bounds are obtained via d-DNNF representations of answer functions [9], improving on the work by Losemann and Martens [101]. Specifically, [9] shows that enumeration with linear preprocessing and output-linear delay can be extended to support relabeling updates in logarithmic time; tractability in the automaton is also possible [10].

5 Aggregative Tasks for CQs and UCQs

Having shown the uses of circuits for MSO queries on trees, we now move to Boolean CQs and UCQs over arbitrary data. We focus on aggregation tasks and study enumeration tasks in Section 6. We first focus on conjunctive queries (CQs), which are existentially quantified conjunctions of atoms; and on unions of conjunctive queries (UCQs). We show how circuits can be used for probabilistic query evaluation (PQE) on tuple-independent databases (TIDs), and its special case uniform reliability (UR). We first concentrate on exact (i.e., non-approximate) PQE, and we study it in data complexity, i.e., for fixed queries $Q$: first for CQs under a self-join-freeness assumption, then for UCQs. We then move to approximate PQE, and to combined complexity where both $Q$ and the TID are given as input. Last, we cover Shapley values as another aggregation task.

Exact PQE for self-join-free CQs. To study PQE, one first class of queries to consider are the so-called self-join-free CQs (SJFCQs). A CQ is self-join-free if each relation occurs only once, i.e., there are no two atoms with the same symbol. For such queries, a dichotomy on PQE
was shown by Dalvi and Suciu [56]: a SJFCQ is either hierarchical, in which case PQE is in polynomial-time data complexity; or it is non-hierarchical, in which case PQE is \#P-hard in data complexity (and so is UR [17]). The tractability of hierarchical SJFCQs can be explained via tractable circuit representations of the Boolean provenance, which is called the intensional approach to PQE. More specifically, for hierarchical SJFCQs, the provenance can be computed in polynomial-time as an OBDD, in fact even as a read-once formula [111].

Exact PQE for UCQs. Following this dichotomy on PQE for SJFCQs, Dalvi and Suciu have shown a far more general dichotomy on UCQs: the PQE problem enjoys PTIME data complexity for some UCQs (called safe), and for all others (the unsafe UCQs) there is a \#P-hardness result for PQE [57], indeed even for UR for most unsafe UCQs [89]. Ten years later, understanding this dichotomy in terms of tractable circuits is still an open research problem. Indeed, the algorithm of [57] follows the so-called extensional approach for PQE and directly computes the probability of the query. It does not follow the intensional approach of going via provenance. The intensional vs. extensional conjecture [106] thus asks whether we can solve PQE for any safe UCQ by computing a provenance representation in a tractable circuit class and invoking WMC on that class.

The intensional vs. extensional conjecture was investigated first by Jha and Suciu [80]: they characterize the strict subset of safe UCQs whose provenance can be expressed as read-once formulas (like for hierarchical SJFCQs), and also the larger strict subset, called inversion-free UCQs, for which we can build OBDDs in PTIME. It was then shown in [40] that safe UCQs that are not inversion-free do not admit polynomial-size provenance representations even as d-SDNNFs. Jha and Suciu [80] also give sufficient conditions on safe UCQs to admit polynomial-size FBDD provenance representations, but without a characterization. It was later shown in [28] that some safe UCQs admit no polynomial-size provenance representation as so-called DLDDs, implying the same for dec-SDNNFs and FBDDs.

However, the intensional vs. extensional conjecture is still open for more expressive circuit classes with tractable WMC. It remains open whether the class of d-SDNNFs can tractably represent the provenance of all safe UCQs (with [80] conjecturing that it does not). The question is also open for the more general class of d-Ds, which is in fact not yet separated from d-SDNNFs [106]. The ability to compute polynomial d-Ds to represent the provenance of all safe UCQs currently hinges on the unproven non-cancelling intersections conjecture [20].

Approximate PQE for UCQs and combined complexity. Faced with the general intractability of PQE for unsafe UCQs, it is natural to settle for approximate PQE. Additive approximations can be obtained simply via Monte Carlo sampling [127], and multiplicative approximations can always be obtained through the intensional approach. Namely, for any fixed UCQ, we can represent its provenance as a monotone DNF in PTIME data complexity, and we can then solve approximate PQE by solving approximate weighted model counting (Approx-WMC) on the DNF via the Karp-Luby algorithm [86] (see Section 3), giving an FPRAS for the task.

This tractability of approximate PQE leads to the question of finding efficient algorithms in combined complexity, i.e., when the query is also given as input. In this light, Van Bremen and Meel [130] have studied the combined complexity of SJFCQs of bounded hypertreewidth, which are tractable for non-probabilistic query evaluation [72], and extended this tractability result to approximate PQE. We will explain in Section 6 how their algorithm can be understood via SDNNF circuits and via the FPRAS of [22] mentioned in Section 3. Note that [130] generally does not extend to CQs with self-joins [21]. We also note that provenance-based approaches have also been used for tractable combined algorithms for exact PQE, e.g., via \( \beta \)-acyclic positive DNFs [19, 41].

Shapley values. We can use circuits for other aggregation tasks than PQE: one important example is computing the Shapley value of facts in relational instances. This task has been shown to be tractable whenever we can compute a representation of the query provenance as a d-D [63], and the same was shown for the related notion of Banzhaf values [4]. As another example of an aggregation task, the computation of the Shapley value (and variants) has also been recently extended to the setting of probabilistic instances [85], also using d-D circuits.

Implementation. We last mention ProvSQL [122], as it is a concrete instantiation of the circuit-based approach to provenance [64]. ProvSQL is a module of the PostgreSQL relational database management system, which adds the possibility to track the provenance of query results as a circuit throughout query evaluation. These general-purpose circuits can then be used for various applications. In particular, they can be used for PQE via existing knowledge compilation tools for WMC (see Section 3.3), or for Shapley value computation [85].

6 Circuits to Represent Query Answers
We now move to a different perspective on circuits, which we will use in particular for enumeration tasks with CQs. We do not use answer functions for CQs, because such queries cannot easily express that a variable is assigned to exactly one value. For this reason, unlike previous sections, we will go beyond Boolean functions and Boolean circuits, and adopt a multivalued perspective: we will
show how circuits can succinctly represent relations, in particular query answers.

**Representing Relations as Circuits.** Let \( f : \{0,1\}^X \rightarrow \{0,1\} \) be a Boolean function. We can easily see \( f^{-1}(1) \subseteq \{0,1\}^X \) as a relational table \( R \), whose domain is \( \{0,1\} \) and whose attributes (in the named perspective) is \( X \); see, e.g., the table right of Figure 1. What is more, we can see circuit representations of \( f \) as a factorized representation of \( R \). More specifically, a DNNF \( C \) on variables \( X \) can be seen as a factorized way of building up a relation from elementary relations of the form \( x = 0 \) or \( x = 1 \). Then, \( \wedge \)-gates are a natural join of their input relations (denoted by \( \triangleright \triangleright \)), while \( \vee \)-gates are unions.

Unfortunately, the semantics of \( \vee \)-gates does not precisely correspond to the union operator of relational algebra. Indeed, the union operator applies to tables with the same attributes, whereas two Boolean functions \( f \) and \( g \) on different variable sets \( X \neq Y \) can be disjoined as \( f \lor g \), giving a function on variables \( X \sqcup Y \). This issue does not arise with smooth circuits (see Section 3), but to interpret \( \lor \) for general circuits we must extend the union operator. Formally, given two relations \( R \subseteq D^X \) and \( S \subseteq D^Y \), the extended union \( R \cup S \) of \( R \) and \( S \) is the relation on attributes \( X \sqcup Y \) of \( R \cup S \) defined as \( (R \times D^Y(X)) \cup (S \times D^X(Y)) \). When \( X = Y \) then we clearly have \( R \cup S = R \cup S \).

We can now directly generalize NNFs and their variations to relational circuits on any finite domain: see Table 2 for a summary. Namely, a \( \{\cup,\triangleright\} \)-circuit \( C \) on attributes \( X \) and domain \( D \) is a circuit whose internal gates are labeled by either \( \cup \) or \( \triangleright \triangleright \) and whose input gates are labeled by relations of the form \( x/d \) where \( x \in X \) and \( d \in D \). The attributes \( \text{attr}(g) \subseteq X \) of a gate \( g \) of \( C \) are the attributes labeling the inputs of \( C \) that have a directed path to \( g \). Further, \( g \) computes a relation \( \text{rel}(g) \subseteq D^{\text{attr}(g)} \) inductively defined from the relations computed by its inputs and from the label of \( g \). The circuit \( C \) computes \( \text{rel}(C) \) which is \( \text{rel}(o) \) for the output \( o \) of \( C \), again extended by allowing arbitrary values in \( D \) for the attributes of \( X \setminus \text{attr}(o) \). One can easily check that if \( C \) is a \( \{\cup,\triangleright\} \)-circuit on domain \( \{0,1\} \), then we can get an NNF circuit computing \( \text{rel}(C) \) by renaming every input of the form \( x/1 \) by \( x \) and \( x/0 \) by \( \neg x \) and replacing every \( \triangleright \triangleright \)-gate by a \( \wedge \)-gate and every \( \cup \)-gate by a \( \vee \)-gate.

Now, if a \( \triangleright \triangleright \)-gate has inputs whose attributes are pairwise disjoint, it actually computes the Cartesian product of these inputs, and we can denote it by \( \times \) and call it decomposable. Hence, a \( \{\cup,\times\} \)-circuit is a \( \{\cup,\triangleright\triangleright\} \)-circuit wherever every \( \triangleright \triangleright \)-gate is in fact computing a Cartesian product. Such decomposable circuits correspond to DNNFs.

---

**Table 2: From Boolean circuits to relational circuits.**

<table>
<thead>
<tr>
<th>Boolean Circuits</th>
<th>Relational Circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNF</td>
<td>( {\cup,\triangleright} )</td>
</tr>
<tr>
<td>DNNF</td>
<td>( {\cup,\times} ) or ( {\cup,\triangleright\triangleright} ) if smooth</td>
</tr>
<tr>
<td>d-DNNF</td>
<td>( {\times,\times} ) or ( {\cup,\times} ) if smooth</td>
</tr>
<tr>
<td>dec-DNNF</td>
<td>( {\text{dec},\times} )</td>
</tr>
</tbody>
</table>

When restricted to the Boolean domain. Similarly, if a \( \cup \)-gate \( g \) has inputs \( g_1,\ldots,g_k \) such that \( \text{attr}(g_1) = \cdots = \text{attr}(g_k) \), it actually computes a union and we denote it by \( \cup \). This gives \( \{\cup,\times\} \)-circuits, which correspond to smooth DNNFs. We can also generalize structuredness: a \( \times \)-gate \( g \) with two inputs \( g_1,g_2 \) respects a vtree \( T \) on \( X \) if there is a node \( t \in T \) such that \( \text{attr}(g_1) \subseteq \text{attr}(t) \) and \( \text{attr}(g_2) \subseteq \text{attr}(t) \) where \( t_1,t_2 \) are the children of \( t \) in \( T \) and each \( \text{attr}(t_i) \) is the set of attributes labeling the leaves of the subtree of \( T \) rooted at \( t_i \).

We can also generalize determinism: denoting disjoint union by \( \cup \), we have \( \{\cup,\times\} \)-circuits which correspond to smooth d-DNNFs. Again, the disjointness of \( \cup \)-gates is a semantic property that may be intractable to verify. Like in Section 3, we can enforce it by a sufficient syntactic condition: a \( \cup \)-gate is a decision gate if it is of the form \( \{b \in D^X \mid x/d \} \times g_d \) for a subset \( D \subseteq D \), where \( \{x/d\} \) are input gates. A \( \{\text{dec},\times\} \)-circuit is a \( \{\cup,\times\} \)-circuit where every \( \cup \)-gate is a decision gate.

**Factorized Databases.** The notion of relational circuits that we present here is related to factorized databases as introduced by Olteanu and Závodny [113, 114]. Their \( d \)-representations correspond to \( \{\cup,\times\} \)-circuits, and their \( f \)-representations further require that the circuit is actually a tree, i.e., there is no sharing. Further, many factorized database algorithms follow a tree over the database attributes, so that they build structured \( \{\text{dec},\times\} \)-circuits. The link between factorized databases and knowledge compilation is also explored in [110].

**Building and Using Circuits.** We now turn to the use of relational circuits to solve enumeration tasks for CQs. Deciding whether a CQ has at least one answer is NP-hard in combined complexity [52] and the same reduction shows that counting query answers is #P-hard. Thus, a fruitful line of research studies how to design algorithms that use the query structure to enumerate or count CQ answers efficiently. For example, Yannakakis observed in [135] that one can test with linear data complexity whether an acyclic CQ has at least one answer: further, his algorithm is also tractable in the query size. This result has been generalized in many ways: to constant-delay enumeration for the so-called free-connex acyclic CQs by Bagan, Durand and Grandjean [26]; to counting for projection-free CQs in [115] and for arbitrary CQs.
in [67]; to direct access in [42, 51] for some lexicographical orders; or to semiring-based aggregation queries in [3, 81]. Relational circuits offer a unifying view of all these results and more, thanks to a claim initially shown in the setting of factorized databases:

**Theorem 6.1** ([114]). Given a free-connex acyclic conjunctive query \( Q \) and a database \( \mathbb{D} \), we can build a structured \( \{ \text{dec}, \times \} \)-circuit in time \( O(\text{poly} |Q| \cdot |\mathbb{D}|) \) computing \( Q(\mathbb{D}) \). The circuit is of size \( O(\text{poly} |Q| \cdot |\mathbb{D}|) \).

This result is actually also shown for non-acyclic CQs in [114]. In this case, the transformation will not be linear wrt \( |\mathbb{D}| \) anymore but of the form \( O(|\mathbb{D}|^k) \) for \( k \) a constant that depends only on the structure of \( Q \). When \( Q \) is in the class of free-connex acyclic CQs, we have \( k = 1 \). Otherwise, \( k \) intuitively measures how far \( Q \) is from being free-connex acyclic [69]; if \( Q \) is projection-free it is at most, e.g., the hypertreewidth of \( Q \) [72].

There are two main techniques to prove Theorem 6.1, that is, to build circuits of linear size for free-connex acyclic conjunctive queries. The first one can be seen as the trace of Yannakakis’s algorithm [135] and is intuitively the approach taken in [114]. This result decides whether \( Q(\mathbb{D}) \) is empty by alternatively joining and projecting the atoms of \( Q \) in an order that can be obtained from the fact that \( Q \) is free-connex acyclic (see [99, Chapter 6]). This ensures that all intermediate results have size \( O(|\mathbb{D}|) \). The trace of the joins and projections done during evaluation corresponds to a \( \{ \text{dec}, \times \} \)-circuit of size \( O(\text{poly} |Q| \cdot |\mathbb{D}|) \) computing \( Q(\mathbb{D}) \).

Another approach to build a circuit for \( Q(\mathbb{D}) \) is inspired by the DPLL algorithm from top-down knowledge compilers [24]. The circuit is built by following a static order on the attributes and recursively compiling \( Q[x_1 = d] \) for every value \( d \in \mathbb{D} \), which corresponds to building a decision gate on \( x_1 \). If the query can be written as \( Q_1 \land Q_2 \) at some recursive call with \( \text{attr}(Q_1) \cap \text{attr}(Q_2) = \emptyset \), then \( Q_1 \) and \( Q_2 \) are compiled independently and their subcircuits are joined by a \( \times \)-gate. By choosing an order that witnesses the free-connex acyclicity of \( Q \), and adding a caching mechanism to remember previously computed queries, one can show that this algorithm also builds a \( \{ \text{dec}, \times \} \)-circuit of size \( O(\text{poly} |Q| \cdot |\mathbb{D}|) \) computing \( Q(\mathbb{D}) \). If another order is chosen, the output of the algorithm is still a circuit but its size may not be linear in \( |\mathbb{D}| \). The detailed algorithm is given in [47] with order-dependent complexity bounds; it is also exemplified in [46].

The tractable tasks from Table 1 can straightforwardly be generalized to \( \{ \text{dec}, \times \} \)-circuits. In particular, if \( n \) is the number of attributes of \( C \), \( |\text{rel}(C)| \) can be computed in time \( O(n|\mathbb{C}|) \) and \( |\text{rel}(C)| \) can be enumerated with delay \( O(n) \); this fact was observed in [114] and used to motivate factorized databases. In particular, in data complexity, the value \( n \) is a constant, hence we recover the result from [26] that \( Q(\mathbb{D}) \) can be enumerated with constant-delay data complexity (more precisely \( O(\text{poly} |Q|) \) after a \( O(|\text{poly} |Q| |\mathbb{D}|) \) preprocessing when \( Q \) is free-connex acyclic, the preprocessing being here the construction of the circuit from Theorem 6.1. We note that, conditionally, for self-join-free conjunctive queries, it is known that only free-connex acyclic queries enjoy this tractability guarantee [26]. This conditional lower bound does not extend to CQs with self-joins [50], nor to UCQs [42].

More interestingly, the DPLL-based construction naturally produces \( \{ \text{dec}, \times \} \)-circuits of a very particular form: the circuit is built in a way where there exists an order \( \{x_1, \ldots, x_n\} \) on the attributes of \( Q \) such that, for every decision gate \( g \) on \( x_i \) with input \( g_1, \ldots, g_k \), we have \( \text{attr}(g) \subseteq \{x_{i+1}, \ldots, x_n\} \). This makes it possible, after a preprocessing time of \( O(|C|) \), to do direct access for the lexicographical order induced by \( x_1, \ldots, x_n \) in time \( O(|n \cdot \log |\mathbb{D}||) \) [47]. Combined with Theorem 6.1, it is an alternative proof of the results of [51]. The approach also generalizes to non-acyclic queries, matching results from [42], and to so-called signed conjunctive queries.

**From Relational Circuits to Provenance.** Representing answers of CQ with relational circuits is close to computing provenance, and has actually been one of the original motivations [112]. We make this intuition formal by explaining how relational circuits can be used to compute the Boolean provenance of queries.

Consider a signature \( \sigma \) and a projection-free CQ \( Q \) (that is, every variable of \( Q \) is free). We first define a signature \( \sigma' \): for each relation \( R \) in \( \sigma \) we add a relation \( R' \) in \( \sigma' \) with arity increased by 1. Then we denote by \( Q' \) a query on \( \sigma' \) obtained as follows: for each atom \( A = R(x) \) of \( Q \), add an atom \( R'(x, y_A) \) to \( Q' \), where \( y_A \) is a fresh attribute for the atom \( A \). Given an instance \( \mathbb{D} \) on \( \sigma \), we also let \( \mathbb{D}' \) be the instance on \( \sigma' \) where each symbol \( R' \) is interpreted as \( R \) in \( \mathbb{D} \) but each fact \( F \) is augmented with a fresh value \( a_F \). It is easy to check that \( Q(\mathbb{D}) \) and \( Q'(\mathbb{D}') \) are isomorphic up to the added identifiers. Moreover, if \( Q \) is acyclic, then \( Q' \) is also acyclic (they even have the same hypertreewidth).

Now, let \( C \) be a \( \{ \cup, \times \} \)-circuit computing \( Q'(\mathbb{D}') \). Its inputs are of two kinds: relations of the form \( x/a \) for \( x \) an attribute of \( Q \) and \( a \) a value of \( \mathbb{D} \), and relations of the form \( y_A/a_F \) where \( a_F \) is a fresh value added for the tuple \( F \). We modify \( C \) in two steps. First, let \( C' \) be the circuit obtained from \( C \) by replacing each input \( y_A/a_F \) of the second kind by a variable \( X_F \) on domain \( \{0, 1\} \). Note that \( C' \) is generally not decomposable, but decomposability is preserved if \( Q \) is self-join-free: for each fact \( F \) of \( \mathbb{D} \), the variable \( X_F \) of \( C' \) then corresponds to only one input label of \( C \), namely inputs of the form \( y_A/a_F \) where \( A \) is the one atom of \( Q \) for the relation used.
by $F$. In any case, a Boolean valuation $\nu$ of the variables $\{X_F | F \in D\}$ can be naturally identified as a subinstance $D_\nu \subseteq D$ containing the facts $F$ of $D$ such that $\nu(X_F) = 1$, and we can then easily see that $C'$ computes $\{\tau \times \nu | \nu \in 2^D, \tau \in Q(D_\nu)\}$. Second, let $C''$ be the circuit obtained from $C'$ by existentially projecting away every variable of the first kind, i.e., the variables that are not of the form $X_F$. If $C'$ is decomposable then $C''$ also is; however $C''$ is generally not deterministic. Hence, $C''$ is a $\{\cup, \times\}$-circuit computing $\{\nu | \nu \in 2^D \text{ s.t. } Q(D_\nu) \neq \emptyset\}$. That is, it computes the Boolean provenance of the Boolean query obtained by existentially quantifying $Q$. Further, $C''$ has size less than $|C|$. Last, if $Q$ is self-join-free then $C''$ is a DNNF, and $C''$ is structured if $C$ was.

Among other things, this result allows us to recover the result of Van Bremen and Meel [130] mentioned in Section 5. Recall that they give a combined FPRAS for approximate PQE with self-join-free CQs $Q$ of bounded hypertreewidth. This is straightforward in the data complexity sense (see Section 5), so the interesting point is the tractability in combined complexity. Applying Theorem 6.1 (extended for queries of hypertreewidth $k$) to the query $Q''$ defined as above, we get a structured $\{\text{dec, } \times\}$-circuit that computes $Q''(D')$ which has size $O(\text{poly} |Q| \cdot |D|^k)$. As $Q$ is self-join-free, our transformation above gives a SDNNF $C$ computing the Boolean provenance of $Q$ of size $O(\text{poly} |Q| \cdot |D|^k)$. We can then use the FPRAS from [23] on $C$ to solve approximate PQE for $Q$ on $\mathbb{D}$, which concludes.

7 Perspectives

We have seen how results from database theory can often be obtained via tractable circuit classes, or how some existing proofs can be rephrased in this vocabulary. We have focused on two main kinds of tasks: aggregation tasks, including PQE and Shapley value computation; and enumeration tasks, including ranked enumeration and direct access. We have focused on the settings of MSO queries and of CQs and UCQs. We believe that this survey illustrates the versatility of circuit techniques, which can sometimes offer a unified view on the tractability of multiple tasks in different areas. Circuits also serve as a convenient intermediate language between database algorithms (which deal with the query and data) and task-specific algorithms (e.g., satisfiability, counting, etc.). This view makes it possible to give modular algorithms and to leverage existing solver implementations.

We have focused on circuit classes from knowledge compilation, but we point out that circuits have been used in other interesting ways in database theory, for example, arithmetic circuits for semiring provenance [64], circuits for secure aggregation on data shared by different parties [131], or circuits for efficient parallel evaluation of queries [90, 132]. Similarly, we have chosen to focus on query evaluation applications in databases, but tractable circuits are also used in closely related topics such as CSPs [29, 48], homomorphism representations [32], or the computation of SHAP-scores [22].

We believe that tractable circuits still have a lot to bring to database theory, and vice-versa; we conclude by highlighting some directions for future work. We see two main research axes: studying how far we can bring circuit methods in established contexts, and introducing them in new contexts. For the first axis, the intensional-extensional conjecture for PQE of UCQs is a clearly identified setting where we do not know how far circuits can be taken to recapture existing results [20, 106]. A similar research direction would be the incremental maintenance of enumeration structures for MSO on trees: while circuit methods handle substitution updates in logarithmic time [9], better complexities are possible, at least in the case of Boolean queries on words [16]: it is unclear for now whether such results can be explained in terms of circuits. It is also unclear which incremental results can be recaptured by circuits in the setting of incremental PQE [30], incremental enumeration for CQs [31, 83, 84] or CQs with aggregates [78, 82], or more general results in algorithms on dynamic data [75]. Moreover, there remain tractability results for aggregate tasks in the database and CSP literature that have not been directly explained from a circuit perspective but which use similar counting techniques, for example results on counting the number of answers of UCQs [53, 70] or complex aggregation over semiring-annotated data [3, 81].

For the second axis, we believe that circuits could be applied to entirely new areas. One possibility is query enumeration for First-Order logic (FO), e.g., over bounded-degree structures [66, 87]. It is not known if such results, and their subsequent extensions [119, 121], can be captured in terms of circuits. A second setting in which circuits could be relevant is database repairs, in particular counting subset repairs [44, 45, 93], enumerating them [93], and more generally representing them in a factorized way. One last question is to connect circuits to the study of efficient algorithms for Datalog evaluation including provenance computation in various semirings [2]. Can tractable algorithms for these tasks be connected to algorithms producing tractable circuit representations? Can fine-grained complexity lower bounds be connected to circuit lower bounds?

Acknowledgements. This work was supported by project ANR KCODA, ANR-20-CE48-0004, and by project ANR CQFD, ANR-18-CE23-0003-02. We are grateful to Tim van Bremen and Benny Kimelfeld for their insightful feedback.
8 References


ABSTRACT

Two decades ago, one of us co-authored a paper commenting on the previous 40 years of data modelling research and development [188]. That paper demonstrated that the relational model (RM) and SQL are the prevailing choice for database management systems (DBMSs), despite efforts to replace either them. Instead, SQL absorbed the best ideas from these alternative approaches.

We revisit this issue and argue that this same evolution has continued since 2005. Once again there have been repeated efforts to replace either SQL or the RM. But the RM continues to be the dominant data model and SQL has been extended to capture the good ideas from others. As such, we expect more of the same in the future, namely the continued evolution of SQL and relational DBMSs (RDBMSs). We also discuss DBMS implementations and argue that the major advancements have been in the RM systems, primarily driven by changing hardware characteristics.

1 Introduction

In 2005, one of the authors participated in writing a chapter for the Red Book titled “What Goes Around Comes Around” [188]. That paper examined the major data modelling movements since the 1960s:

- Hierarchical (e.g., IMS): late 1960s and 1970s
- Network (e.g., CODASYL): 1970s
- Relational: 1970s and early 1980s
- Entity-Relationship: 1970s
- Extended Relational: 1980s
- Semantic: late 1970s and 1980s
- Object-Oriented: late 1980s and early 1990s
- Object-Relational: late 1980s and early 1990s
- Semi-structured (e.g., XML): late 1990s and 2000s

Our conclusion was that the relational model with an extendable type system (i.e., object-relational) has dominated all comers, and nothing else has succeeded in the marketplace. Although many of the non-relational DBMSs covered in 2005 still exist today, their vendors have relegated them to legacy maintenance mode and nobody is building new applications on them. This persistence is more of a testament to the “stickiness” of data rather than the lasting power of these systems. In other words, there still are many IBM IMS databases running today because it is expensive and risky to switch them to use a modern DBMS. But no start-up would willingly choose to build a new application on IMS.

A lot has happened in the world of databases since our 2005 survey. During this time, DBMSs have expanded from their roots in business data processing and are now used for almost every kind of data. This led to the “Big Data” era of the early 2010s and the current trend of integrating machine learning (ML) with DBMS technology.

In this paper, we analyze the last 20 years of data model and query language activity in databases. We structure our commentary into the following areas: (1) MapReduce Systems, (2) Key-value Stores, (3) Document Databases, (4) Column Family / Wide-Column, (5) Text Search Engines, (6) Array Databases, (7) Vector Databases, and (8) Graph Databases.

We contend that most systems that deviated from SQL or the RM have not dominated the DBMS landscape and often only serve niche markets. Many systems that started out rejecting the RM with much fanfare (think NoSQL) now expose a SQL-like interface for RM databases. Such systems are now on a path to convergence with RDBMSs. Meanwhile, SQL incorporated the best query language ideas to expand its support for modern applications and remain relevant.

Although there has not been much change in RM fundamentals, there were dramatic changes in RM system implementations. The second part of this paper discusses advancements in DBMS architectures that address modern applications and hardware: (1) Columnar Systems, (2) Cloud Databases, (3) Data Lakes / Lakehouses, (4) NewSQL Systems, (5) Hardware Accelerators, and (6) Blockchain Databases. Some of these are profound changes to DBMS implementations, while others are merely trends based on faulty premises.

We finish with a discussion of important considerations for the next generation of DBMSs and provide parting comments on our hope for the future of databases in both research and commercial settings.
Data Models & Query Languages

For our discussion here, we group the research and development thrusts in data models and query languages for database into eight categories.

2.1 MapReduce Systems

Google constructed their MapReduce (MR) framework in 2003 as a “point solution” for processing its periodic crawl of the internet [122]. At the time, Google had little expertise in DBMS technology, and they built MR to meet their crawl needs. In database terms, Map is a user-defined function (UDF) that performs computation and/or filtering while Reduce is a GROUP BY operation. To a first approximation, MR runs a single query:

\[
\text{SELECT map()} \text{ FROM crawl_table GROUP BY reduce()}
\]

Google’s MR approach did not prescribe a specific data model or query language. Rather, it was up to the Map and Reduce functions written in a procedural MR program to parse and decipher the contents of data files.

There was a lot of interest in MR-based systems at other companies in the late 2000s. Yahoo! developed an open-source version of MR in 2005, called Hadoop. It ran on top of a distributed file system HDFS that was a clone of the Google File System [134]. Several start-ups were formed to support Hadoop in the commercial marketplace. We will use MR to refer to the Google implementation and Hadoop to refer to the open-source version. They are functionally similar.

There was a controversy about the value of Hadoop compared to RDBMSs designed for OLAP workloads. This culminated in a 2009 study that showed that data warehouse DBMSs outperformed Hadoop [172]. This generated dueling articles from Google and the DBMS community [123, 190]. Google argued that with careful engineering, a MR system will beat DBMSs, and a user does not have to load data with a schema before running queries on it. Thus, MR is better for “one shot” tasks, such as text processing and ETL operations. The DBMS community argued that MR incurs performance problems due to its design that existing parallel DBMSs already solved. Furthermore, the use of higher-level languages (SQL) operating over partitioned tables has proven to be a good programming model [127].

A lot of the discussion in the two papers was on implementation issues (e.g., indexing, parsing, push vs. pull query processing, failure recovery). From reading both papers a reasonable conclusion would be that there is a place for both kinds of systems. However, two changes in the technology world rendered the debate moot.

The first event was that the Hadoop technology and services market cratered in the 2010s. Many enterprises spent a lot of money on Hadoop clusters, only to find there was little interest in this functionality. Developers found it difficult to shoehorn their application into the restricted MR/Hadoop paradigm. There were considerable efforts to provide a SQL and RM interface on top of Hadoop, most notable was Meta’s Hive [30, 197].

The next event occurred eight months after the CACM article when Google announced that they were moving their crawl processing from MR to BigTable [164]. The reason was that Google needed to interactively update its crawl database in real time but MR was a batch system. Google finally announced in 2014 that MR had no place in their technology stack and killed it off [194].

The first event left the three leading Hadoop vendors (Cloudera, Hortonworks, MapR) without a viable product to sell. Cloudera rebranded Hadoop to mean the whole stack (application, Hadoop, HDFS). In a further sleight-of-hand, Cloudera built a RDBMS, Impala [150], on top of HDFS but not using Hadoop. They realized that Hadoop had no place as an internal interface in a SQL DBMS, and they configured it out of their stack with software built directly on HDFS. In a similar vein, MapR built Drill [22] directly on HDFS, and Meta created Presto [185] to replace Hive.

Discussion: MR’s deficiencies were so significant that it could not be saved despite the adoption and enthusiasm from the developer community. Hadoop died about a decade ago, leaving a legacy of HDFS clusters in enterprises and a collection of companies dedicated to making money from them. At present, HDFS has lost its luster, as enterprises realize that there are better distributed storage alternatives [124]. Meanwhile, distributed RDBMSs are thriving, especially in the cloud.

Some aspects of MR system implementations related to scalability, elasticity, and fault tolerance are carried over into distributed RDBMSs. MR also brought about the revival of shared-disk architectures with disaggregated storage, subsequently giving rise to open-source file formats and data lakes (see Sec. 3.3). Hadoop’s limitations opened the door for other data processing platforms, namely Spark [201] and Flink [109]. Both systems started as better implementations of MR with procedural APIs but have since added support for SQL [105].

2.2 Key/Value Stores

The key/value (KV) data model is the simplest model possible. It represents the following binary relation:

\[(\text{key}, \text{value})\]

A KV DBMS represents a collection of data as an associative array that maps a key to a value. The value is typically an untyped array of bytes (i.e., a blob), and the DBMS is unaware of its contents. It is up to the application to maintain the schema and parse the value into its corresponding parts. Most KV DBMSs only provide get/set/delete operations on a single value.

In the 2000s, several new Internet companies built their own shared-nothing, distributed KV stores for nar-
rowly focused applications, like caching and storing session data. For caching, Memcached [131] is the most well-known example of this approach. Redis [67] markets itself as a Memcached replacement, offering a more robust query API with checkpointing support. For more persistent application data, Amazon created the Dynamo KV store in 2007 [125]. Such systems offer higher and more predictable performance, compared to a RDBMS, in exchange for more limited functionality.

The second KV DBMS category are embedded storage managers designed to run in the same address space as a higher-level application. One of the first stand-alone embedded KV DBMSs was BerkeleyDB from the early 1990s [170]. Recent notable entries include Google’s LevelDB [37], which Meta later forked as RocksDB [68].

**Discussion:** Key/value stores provide a quick “out-of-the-box” way for developers to store data, compared to the more laborious effort required to set up a table in a RDBMS. Of course, it is dangerous to use a KV store in a complex application that requires more than just a binary relation. If an application requires multiple fields in a record, then KV stores are probably a bad idea. Not only must the application parse record fields, but also there are no secondary indexes to retrieve other fields by value. Likewise, developers must implement joins or multi-get operations in their application.

To deal with these issues, several systems began as a KV store and then morphed into a more feature-rich record store. Such systems replace the opaque value with a semi-structured value, such as a JSON document. Examples of this transition are Amazon’s DynamoDB [129] and Aerospike [9]. It is not trivial to re-engineer a KV store to make it support a complex data model, whereas RDBMSs easily emulates KV stores without any changes. If an application needs an embedded DBMS, there are full-featured choices available today, including SQLite [71] and DuckDB [180]. Hence, a RDBMS may be a better choice, even for simple applications, because they offer a path forward if the application’s complexity increases.

One new architecture trend from the last 20 years is using embedded KV stores as the underlying storage manager for full-featured DBMSs. Prior to this, building a new DBMS requires engineers to build a custom storage manager that is natively integrated in the DBMS. MySQL was the first DBMS to expose an API that allowed developers to replace its default KV storage manager. This API enabled Meta to build RocksDB to replace InnoDB for its massive fleet of MySQL databases. Similarly, MongoDB discarded their ill-fated MMAP-based storage manager in favor of WiredTiger’s KV store in 2014 [120, 138]. Using an existing KV store allows developers to write a new DBMS in less time.

### 2.3 Document Databases

The document data model represents a database as a collection of record objects. Each document contains a hierarchy of field/value pairs, where each field is identified by a name and a field’s value can be either a scalar type, an array of values, or another document. The following example in JSON is a customer document that contain a nested list of purchase order records with their corresponding order items.

```
{ "name": "First Last",
  "orders": [ { "id": 123, "items": [...] },
              { "id": 456, "items": [...] }, ]
}
```

Document data models have been an active field of effort for several decades. This has given rise to data formats like SGML [117] and XML [118]. Despite the buzz with XML databases in the late 1990s, we correctly predicted in 2005 they would not supplant RDBMSs [188]. JSON has since overtaken XML to become the standard for data exchange for web-based applications. JavaScript’s popularity with developers and the accompanying ubiquity of JSON led several companies to create document-oriented systems that natively stored JSON in the 2000s.

The inability of OLTP RDBMSs to scale in the 2000s ushered in dozens of document DBMSs that marketed themselves using the catchphrase *NoSQL* [110]. There were two marketing messages for such systems that resonated with developers. First, SQL and joins are slow, and one should use a “faster” lower-level, record-at-a-time interface. Second, ACID transactions are unnecessary for modern applications, so the DBMS should only provide weaker notion of it (i.e., BASE [179]).

Because of these two thrusts, NoSQL came to stand for a DBMS that stored records or documents as JSON, supported a lower-level API, and weak or non-existent transactions. There are dozens of such systems, of which MongoDB [41] is the most popular.

**Discussion:** Document DBMSs are essentially the same as object-oriented DBMSs from the 1980s and XML DBMSs from the late 1990s. Proponents of document DBMSs make the same argument as their OO/XML predecessors: storing data as documents removes the impedance mismatch between how application OO code interacts with data and how relational databases store them. They also claim that denormalizing entries into nested structures is better for performance because it removes the need to dispatch multiple queries to retrieve data related to a given object (i.e., “N+1 problem” in ORMs). The problems with denormalization/prejoining is an old topic that dates back to the 1970s [116]: (1) if the join is not one-to-many, then there will be duplicated data, (2) prejoins are not necessarily faster than joins, and (3) there is no data independence.
Despite strong protestations that SQL was terrible, by the end of the 2010s, almost every NoSQL DBMS added a SQL interface. Notable examples include DynamoDB PartiQL [56], Cassandra CQL [15], Aerospike AQL [9], and Couchbase SQL++ [72]. The last holdout was MongoDB, but they added SQL for their Atlas service in 2021 [42]. Instead of supporting the SQL standard for DDL and DML operations, NoSQL vendors claim that they support their own proprietary query language derived or inspired from SQL. For most applications, these distinctions are without merit. Any language differences between SQL and NoSQL derivatives are mostly due to JSON extensions and maintenance operations.

Many of the remaining NoSQL DBMSs also added strongly consistent (ACID) transactions (see Sec. 3.4). As such, the NoSQL message has morphed from “Do not use SQL – it is too slow!” to “Not only SQL” (i.e., SQL is fine for some things).

Adding SQL and ACID to a NoSQL DBMS lowers their intellectual distance from RDBMSs. The main differences between them seems to be JSON support and the fact that NoSQL vendors allow “schema later” databases. But the SQL standard added a JSON data type and operations in 2016 [165, 178]. And as RDBMSs continue to improve their “first five minutes” experience for developers, we believe that the two kinds of systems will soon be effectively identical.

Higher level languages are almost universally preferred to record-at-a-time notations as they require less code and provide greater data independence. Although we acknowledge that the first SQL optimizers were slow and ineffective, they have improved immensely in the last 50 years. But the optimizer remains the hardest part of building a DBMS. We suspect that this engineering burden was a contributing factor to why NoSQL systems originally chose to not support SQL.

2.4 Column-Family Databases

There is another category of NoSQL systems that uses a data model called column-family (aka wide-column). Despite its name, column-family is not a columnar data model. Instead, it is a reduction of the document data model that only supports one level of nesting instead of arbitrary nesting: it is relation-like, but each record can have optional attributes, and cells can contain an array of values. The following example shows a mapping from user identifier keys to JSON documents that contain each user’s varying profile information:

```
User1000 = { "name": "Alice", "accounts": [ 123, 456 ], "email": "xxx@xxx.edu" }
User1001 = { "name": "Bob", "email": [ "yyy@yyy.org", "zzz@zzz.com" ] }
```

The first column-family model DBMS was Google’s BigTable in 2004 [111]. Instead of adopting SQL and emerging columnar storage, Google used this data model with procedural client APIs. Other systems adopted the column-family model in an attempt to copy Google’s bespoke implementation. Most notable are Cassandra [14] and HBase [28]. They also copied BigTable’s limitations, including the lack of joins and secondary indexes.

**Discussion**: All our comments in Sec. 2.3 about the document model are also applicable here. In the early 2010s, Google built RDBMSs on top of BigTable, including MegaStore [99] and the first version of Spanner. Since then, Google rewrote Spanner to remove the BigTable remnants [98], and it is now the primary database for many of its internal applications. Several NoSQL DBMSs deprecated their proprietary APIs in favor of SQL but still retain their non-relational architectures. Cassandra replaced their Thrift-API with a SQL-like language called CQL [15], and HBase now recommends the Phoenix SQL-frontend [57]. Google still offers BigTable as a cloud service, but the column-family model is a singular outlier with the same disadvantages as NoSQL DBMSs.

2.5 Text Search Engines

Text search engines have existed for a long time, beginning with the seminal SMART system in the 1960s [184]. SMART pioneered information retrieval and the vector space model, now nearly universal in modern search engines, by tokenizing documents into a “bag of words” and then building full-text indexes (aka inverted indexes) on those tokens to support queries on their contents. The system was also cognizant of noise words (e.g., “the”, “a”), synonyms (e.g., “The Big Apple” is a synonym for “New York City”), salient keywords, and distance (e.g., “drought” often appears close to “climate change”).

The leading text search systems today include Elasticsearch [23] and Solr [70], which both use Lucene [38] as their internal search library. These systems offer good support for storing and indexing text data but offer none-to-limited transaction capabilities. This limitation means that a DBMS has to recover from data corruption by rebuilding the document index from scratch, which results in significant downtime.

All the leading RDBMSs support full-text search indexes, including Oracle [52], Microsoft SQL Server [52], MySQL [43], and PostgreSQL [62]. Their search features have improved recently and are generally on par with the special-purpose systems above. They also have the advantage of built-in transaction support. But their integration of search operations in SQL is often clunky and differs between DBMSs.

**Discussion**: Text data is inherently unstructured, which means that there is no data model. Instead, a DBMS seeks to extract structure (i.e., meta-data, indexes) from text to avoid “needle in the haystack” sequential searches.
There are three ways to manage text data in application. First, one can run multiple systems, such as Elastic-search for text and a RDBMS for operational workloads. This approach allows one to run “best of breed” systems but requires additional ETL plumbing to push data from the operational DBMS to the text DBMS and to rewrite applications to route queries to the right DBMSs based on their needs. Alternatively, one can run a RDBMS with good text-search integration capabilities but with divergent APIs in SQL. This latter issue is often overcome by application frameworks that hide this complexity (e.g., Django Haystack [20]). The third option is a polystore system [187] that masks the system differences via middleware that exposes a unified interface.

Inverted index-centric search engines based on SMART and TileDB [76]. HDF5 [29] and NetCDF [46] are Arrays are also the core of most ML data sets. (Mean all variants of them [182]: vectors (one dimension – see Sec. 2.7), matrices (two dimensions), and tensors (three or more dimensions). For example, scientific surveys for geographic regions usually represent data as a multi-dimensional array that stores sensor measurements using location/time-based coordinates:

\[(\text{latitude, longitude, time, [vector-of-values]})\]

Several other data sets look like this, including genomic sequencing and computational fluid dynamics. Arrays are also the core of most ML data sets.

Although array-based programming languages have existed since the 1960s (APL [142]), the initial work on array DBMSs began in the 1980s. PICDMS is considered to be the first DBMS implementation using the array data model [114]. The two oldest array DBMSs still being developed today are Rasdaman [66, 103] and kdb+ [34]. Newer array DBMSs include SciDB [54, 191] and TileDB [76]. HDF5 [29] and NetCDF [46] are popular array file formats for scientific data.

There are several system challenges with storing and querying real-world array data sets. Foremost is that array data does not always align to a regular integer grid; for example, geospatial data is often split into irregular shapes. An application can map such grids to integer coordinates via metadata describing this mapping [166]. Hence, most applications maintain array and non-array data together in a single database.

Unlike row- or column-based DBMSs, querying array data in arbitrary dimensions presents unique challenges. The difficulty arises from storing multi-dimensional array data on a linear physical storage medium like a disk. To overcome these challenges, array DBMSs must employ indexing and data structures to support efficient traversal across array dimensions.

**Discussion:** Array DBMSs are a niche market that has only seen adoption in specific verticals (we discuss vector DBMSs next). For example, they have considerable traction in the genomics space. HDF5 is popular for satellite imagery and other gridded scientific data. But business applications rarely use dedicated array DBMSs, which is necessary for any product to survive. No major cloud provider offers a hosted array DBMS service, meaning they do not see a sizable market.

The challenge that array DBMS vendors have always faced is that the SQL includes support for ordered arrays as first-class data types (despite this being against the original RM proposal [115]). The first proposal to extend the unordered set-based RM with ordered rasters was in 1993 [155]. An early example of this was Illustra’s temporal (one-dimensional) data plugin [31]. SQL:1999 introduced limited support for single-dimension, fixed-length array data types. SQL:2003 expanded to support nested arrays without a predefined maximum cardinality. Later entrants include Oracle Georaster [4] and Teradata [73]. Data cubes are special-purpose arrays [135], but columnar RDBMSs have eclipsed them for OLAP workloads because of their better flexibility and lower engineering costs [113].

More recently, the SQL:2023 standard includes support for true multi-dimensional arrays (SQL/MDA) that is heavily inspired by Rasdaman’s RQL [166]. This update allows SQL to represent arrays with arbitrary dimensions using integer-based coordinates. In effect, this allows data cubes to exist in a SQL framework, but columnar DBMSs now dominate this market.

### 2.7 Vector Databases

Similar to how the column-family model is a reduction of the document model, the vector data model simplifies the array data model to one-dimensional rasters. Given that vector DBMSs are attracting the most attention right now from developers and investors (similar to the NoSQL fad), it is necessary to discuss them separately. The reason for this interest is because developers use them to store single-dimension *embeddings* generated from AI tools. These tools use learned transformations to convert a record’s data (e.g., text, image) into a vector representing its latent semantics. For example, one could convert each Wikipedia article into an embedding using Google BERT and store them in a vector database along with additional article meta-data:

\[(\text{title, date, author, [embedding-vector]})\]

The size of these embedding vectors range from 100s of dimensions for simple transformers to 1000s for high-end models; these sizes will obviously grow over time with the development of more sophisticated models.
The key difference between vector and array DBMSs is their query patterns. The former are designed for similarity searches that find records whose vectors have the shortest distance to a given input vector in a high-dimensional space. The input vector is another embedding generated with the same transformer used to populate the database. Unlike array DBMSs, applications do not use vector DBMSs to search for matches at an offset in a vector nor extract slices across multiple vectors. Instead, the dominant use case is this similarity search.

To avoid brute force scans for finding the most similar records, vector DBMSs build indexes to accelerate approximate nearest neighbor (ANN) searches. Applications issue queries with predicates on both the embedding index and non-embedding attributes (i.e., metadata). The DBMS then chooses whether to use the non-embedding predicate on records before (pre-filter) or after (post-filter) the vector search.

There are dozens of new DBMSs in this emerging category, with Pinecone [58], Milvus [40], and Weaviate [84] as the leading systems. Text search engines, including Elasticsearch [23], Solr [70], and Vespa [79], expanded their APIs to support vector search. Other DBMSs rebranded themselves as vector databases to jump on the bandwagon, such as Kdb+ [34].

One compelling feature of vector DBMSs is that they provide better integration with AI tools (e.g., ChatGPT [16], LangChain [36]) than RDBMSs. These systems natively support transforming a record’s data into an embedding upon insertion using these tools and then uses the same transformation to convert a query’s input arguments into an embedding to perform the ANN search; other DBMSs require the application to perform these transformations outside of the database.

**Discussion:** Unlike array DBMSs that require a customized storage manager and execution engine to support efficient operations on multi-dimensional data, vector DBMSs are essentially document-oriented DBMSs with specialized ANN indexes. Such indexes are a feature, not the foundation of a new system architecture.

After LLMs became “mainstream” with ChatGPT in late 2022, it took less than one year for several RDBMSs to add their own vector search extensions. In 2023, many of the major RDBMSs added vector indexes, including Oracle [7], SingleStore [137], Rockset [8], and Clickhouse [157]. Contrast this with JSON support in RDBMSs. NoSQL systems like MongoDB and CouchDB became popular in the late 2000s and it took several years for RDBMSs to add support for it.

There are two likely explanations for the quick proliferation of vector indexes. The first is that similarity search via embeddings is such a compelling use case that every DBMS vendor rushed out their version and announced it immediately. The second is that the engineering effort to introduce a new index data structure is small enough that it did not take that much work for the DBMS vendors to add vector search. Most of them did not write their vector index from scratch and instead integrated an open-source library (e.g., pgVector [145], DiskANN [19], FAISS [24]).

We anticipate that vector DBMSs will undergo the same evolution as document DBMSs by adding features to become more relational-like (e.g., SQL, transactions, extensibility). Meanwhile, relational incumbents will have added vector indexes to their already long list of features and moved on to the next emerging trend.

### 2.8 Graph Databases

There has been a lot of academic and industry interest in the last decade in graph databases [183]. Many applications use knowledge graphs to model semi-structured information. Social media applications inherently contain graph-oriented relationships (“likes,” “friend-of”). Relational design tools provide users with an entity-relationship (ER) model of their database. An ER diagram is a graph; thus, this paradigm has clear use cases.

The two most prevalent approaches to represent graphs are (1) the resource description framework (RDF) and (2) property graphs [126]. With property graphs, the DBMS maintains a directed multi-graph structure that supports key/value labels for nodes and edges. RDF databases (aka triplestores) only model a directed graph with labeled edges. Since property graphs are more common and are a superset of RDF, we will only discuss them. We consider two use cases for graph DBMSs and discuss the problems that will limit their adoption.

The first category of systems are for operational / OLTP workloads: an application, for example, adds a friend link in the database by updating a single record, presumably in a transactional manner. Neo4j [44] is the most popular graph DBMS for OLTP applications. It supports edges using pointers (as in CODASYL) but it does not cluster nodes with their “parent” or “offspring.” Such an architecture is advantageous for traversing long edge chains since it will do pointer chasing, whereas a RDBMS has to do this via joins. But their potential market success comes down to whether there are enough “long chain” scenarios that merit forgoing a RDBMS.

The second use case is analytics, which seeks to derive information from the graph. An example of this scenario is finding which user has the most friends under 30 years old. Notable entries like Tigergraph [74] and JanusGraph [32] focus on query languages and storage on a graph DBMS. Other systems, such as Giraph [26] and Turi [78] (formerly Graphlab [27]) provide a computing fabric to support parallel execution of graph-oriented programs, typically written by a user.

Unlike queries in relational analytics that are characterized by chains of joins, queries for graph analytics contain operations like shortest path, cut set, or clique...
determination. Algorithm choice and data representation will determine a DBMS’s performance. This argues for a computing fabric that allows developers to write their own algorithms using an abstraction that hides the underlying system topology. However, previous research shows that distributed algorithms rarely outperform single-node implementations because of communication costs [160]. A better strategy is to compress a graph into a space-efficient data structure that fits in memory on a single node and then run the query against this data structure. All but the largest graph databases are probably best handled this way.

Discussion: Regardless of whether a graph DBMS targets OLTP or OLAP workloads, the key challenge these systems have to overcome is that it is possible to simulate a graph as a collection of tables:

| Node (node.id, node.data) |
| Edge (node.id_1, node.id_2, edge.data) |

This means that RDBMSs are always an option to support graphs. But “vanilla” SQL is not expressive enough for graph queries and thus require multiple client-server roundtrips for traversal operations.

Some RDBMSs, including MSSQL [3] and Oracle [50], provide built-in SQL extensions that make storing and querying graph data easier. Other DBMSs use a translation layer on top of relations to support graph-oriented APIs. Amazon Neptune [45] is a graph-oriented veneer on top of Aurora MySQL. Apache AGE provides an OpenCypher interface on top of PostgreSQL [10].

More recently, SQL:2019 introduced property graph queries (SQL/PGQ) for defining and traversing graphs in a RDBMS [196]. The syntax builds on existing languages (e.g., Neo4j’s Cypher [49], Oracle’s PGQL [51], and TigerGraph’s GSQL [75]), and shares aspects of the emerging GQL standard [126]. Thus, SQL/PGQ further narrows the functionality difference between RDBMSs and native graph DBMSs.

The question is whether graph DBMS vendors can make their specialized systems fast enough to overcome the above disadvantages. There have been several performance studies showing that graph simulation on RDBMSs outperform graph DBMSs [130, 143]. More recent work showed how SQL/PGQ in DuckDB outperforms a leading graph DBMS by up to 10× [196]. This trend will continue with further improvements in worst-case optimal joins [132, 168] and factorized execution algorithms [100] for graph queries in RDBMSs.

2.9 Summary
A reasonable conclusion from the above section is that non-SQL, non-relational systems are either a niche market or are fast becoming SQL/RM systems. Specifically:

- **MapReduce Systems**: They died years ago and are, at best, a legacy technology at present.
- **Key-value Stores**: Many have either matured into RM systems or are only used for specific problems. These can generally be equaled or beaten by modern high-performance RDBMSs.
- **Document Databases**: Such NoSQL systems are on a collision course with RDBMSs. The differences between the two kinds of systems have diminished over time and should become nearly indistinguishable in the future.
- **Column-Family Systems**: These remain a niche market. Without Google, this paper would not be talking about this category.
- **Text Search Engines**: These systems are used for text fields in a polystore architecture. It would be valuable if RDBMSs had a better story for search so these would not have to be a separate product.
- **Array Databases**: Scientific applications will continue to ignore RDBMSs in favor of bespoke array systems. They may become more important because RDBMSs cannot efficiently store and analyze arrays despite new SQL/MDA enhancements.
- **Vector Databases**: They are single-purpose DBMSs with indexes to accelerate nearest-neighbor search. RM DBMSs should soon provide native support for these data structures and search methods using their extendable type system that will render such specialized databases unnecessary.
- **Graph Databases**: OLTP graph applications will be largely served by RDBMSs. In addition, analytic graph applications have unique requirements that are best done in main memory with specialized data structures. RDBMSs will provide graph-centric APIs on top of SQL or via extensions. We do not expect specialized graph DBMSs to be a large market.

Beyond the above, there are also proposals to rebrand previous data models as something novel. For example, graph-relational [158] is the same as the semantic data model [202]. Likewise, document-relational is the document model with foreign keys [199]. Others provide a non-SQL veneer over a RDBMS (e.g., PRQL [64], Mallory [39]). Although these languages deal with some of SQL’s shortcomings, they are not compelling enough to overcome its entrenched userbase and ecosystem.

3 System Architectures
There have been major new ideas in DBMS architectures put forward in the last two decades that reflecting changing application and hardware characteristics. These ideas range from terrific to questionable, and we discuss them in turn.

3.1 Columnar Systems
To understand the appeal of columnar DBMSs, we need to explain the origins of the data warehouse (OLAP) market. Beginning in the mid-1990s, enterprises started
collecting their customer facing (usually sales) data. Brick-and-mortar retailers (e.g., Walmart) were at forefront of constructing historical sales databases. These companies generally found that a sales data warehouse would pay for itself in better stock ordering and rotation decisions within six months. Such customer facing databases are now omnipresent in enterprises.

Data warehouse applications have common properties that are distinct from OLTP workloads:

1. They are historical in nature (i.e., they are loaded periodically and then are read-only).
2. Organizations retain everything as long as they can afford the storage — think terabytes to petabytes.
3. Queries typically only access a small subset of attributes from tables and are ad-hoc in nature.

Ralph Kimball was an early proponent of star schema data modelling for data warehouses [148, 149]. The idea was to construct a fact table that held item-level transactional data. The classic example is a fact table that contains a record for every item purchased in a retail enterprise. Then, one surrounds the fact table with dimension tables that contain common information factored out from the fact table to save space. Again, in a retail setting, these dimension tables would include information about customers, products, stores, and time.

Organizing the DBMS’s storage by columns instead of rows has several benefits [87]. First, compressing columnar data is more effective than row-based data because there is a single value type in a data block often many repeated bytes. Second, a Volcano-style engine executes operators once per row. In contrast, a column-oriented engine has an inner loop that processes a whole column using vectorized instructions [106, 147]. Lastly, row stores have a large header for each record (e.g., 20 bytes) to track nulls and versioning meta-data, whereas column stores have minimal storage overhead per record.

Discussion: Over the last two decades, all vendors active in the data warehouse market have converted their offerings from a row store to a column store. This transition brought about significant changes in the design of DBMSs. In addition, several new vendors have entered the market in the last two decades with column store offerings, for example Amazon’s Redshift [94] and Google’s BigQuery [162] along with offerings from independent companies (e.g., Snowflake [121]).

In summary, column stores are new DBMS implementations with specialized optimizers, executors, and storage formats. They have taken over the data warehouse marketplace because of their superior performance.

### 3.2 Cloud Databases

The rise of cloud platforms in the late 2000s has also greatly affected the implementation (and sales model) of DBMSs. Initial cloud DBMS offerings repackaged on-prem systems into managed VMs with direct-attached storage. But over the last 20 years, networking bandwidth has increased much faster than disk bandwidth, making network attached storage (NAS) attractive as an alternative to attached storage. This has caused a profound rethinking of DBMS architectures for the cloud.

All major cloud vendors offer NAS via object stores (e.g., Amazon S3) with some DBMS functionality (e.g., replication, filtering). Beyond better economics compared to direct-attached storage, object stores have several advantages that compensate for the cost of the added network link. First, because the compute nodes are disconnected from the storage nodes, a system can provide per-query elasticity; the DBMS can add new compute nodes dynamically without having to reshuffle data. It also allows the DBMS to use different hardware for its storage nodes than compute nodes. Second, the system can reassign compute nodes to other tasks if a DBMS is underutilized. On the other hand, in a shared-nothing DBMS, a node must always be online to handle incoming query requests. Lastly, pushing down computation into the storage nodes is possible (and generally advantageous). This execution strategy is known as “pushing the query to the data” versus “pulling the data to the query” and is well understood in DBMSs.

Generally, the first two ideas are called “serverless computing”; and was introduced for cloud-native DBMSs by Snowflake [121]. Other vendors have moved or are in the process of moving to a serverless environment for their cloud offerings. Effective utilization of this model requires a hosted multi-node environment in which multiple DBMS customers are grouped onto the same node(s) with a multi-tenant execution scheme.

Discussion: The advent of cloud databases is another example of “what goes around comes around”. Multi-node shared-disk DBMSs are an old idea that historically tended not to work out well. However, it is back in vogue with technology change (faster networking) and moving to the cloud. In addition, time-sharing services were popular in the 1970s when computers were big and expensive. Cloud platforms are big time-sharing services, so the concept is back after a few decades. Since enterprises are moving everything possible to the cloud, we expect this shared-disk to dominate DBMS architectures. Hence, we do not foresee shared-nothing architectures resurfacing in the future.

The cloud has profoundly impacted DBMSs, causing them to be completely re-architected. The movement of computing from on-prem to the cloud generates a once-in-a-lifetime opportunity for enterprises to refactor codebases and remove bad historical technology decisions. A cloud environment also provides several benefits to vendors that are not possible with on-prem deployments. Foremost is that vendors can track usage...
trends for all their customers: they can monitor unexpected behavior, performance degradations, and usage patterns. Moreover, they can push incremental updates and code patches without disrupting service.

From a business perspective, open-source DBMSs face the danger of becoming too popular and being monetized by the major cloud providers. The public spats between Amazon and ISVs like MongoDB [153] and Elasticsearch [101] are notable examples.

3.3 Data Lakes / Lakehouses

Another trend that the cloud platforms fomented is the movement away from monolithic, dedicated data warehouses for OLAP workloads and towards data lakes backed by object stores. With legacy data warehouses, organizations load data into the DBMS, which the system stashes in managed storage with proprietary formats. Vendors viewed their DBMSs as the "gatekeepers" for all things related to data in an organization. However, this has not been the model of many organizations, especially technology companies, for the last decade.

With a data lake architecture, applications upload files to a distributed object store, bypassing the traditional route through the DBMS [167]. Users then execute queries and processing pipelines on these accumulated files using a lakehouse (a portmanteau of data warehouse and data lake) execution engine [93]. These lakehouse systems provide a unified infrastructure supporting SQL and non-SQL workloads. The latter is crucial as the last decade has shown that data scientists and ML practitioners typically use Python-based notebooks that use Panda’s DataFrame API [159] to access data instead of SQL. Several projects leverage DBMS methods to optimize DataFrame processing, including Dask [181], Polars [61], Modin [177], and Bodo [198].

Instead of using DBMS-specific proprietary file formats or inefficient text-based files (e.g., CSV, JSON), applications write data to data lakes using open-source, disk-resident file formats [203]. The two most popular formats are Twitter/Cloudera’s Parquet [55] and Meta’s ORC [53, 140]. Both of them borrow techniques from earlier columnar storage research, such as PAX [90], compression [87], and nested-data (JSON) shredding [121, 161]. Apache Arrow [11] is a similar binary format for exchanging in-memory data between systems. Open-source libraries for reading/writing these formats allow disparate applications to create data files that other systems then parse and consume, thereby enhancing data sharing across services and business units.

Discussion: Data lakes are the successor to “Big Data” movement from the early 2010s, partly led by the popularity of MR systems (Sec. 2.1) and column stores (Sec. 3.1). At first glance, a data lake seems like a terrible idea for an organization: allowing any application to write arbitrary files into a centralized repository without any governance is a recipe for integrity, discovery, and versioning problems [167]. Lakehouses provide much-needed control over these environments to help mitigate many problems with meta-data, caching, and indexing services [93]. Additional middleware that tracks new data and supports transactional updates, such as Delta Lake [92], Iceberg [6], and Hudi [5], make lakehouses look more like a traditional data warehouse.

Data lakes introduce new challenges to query optimization. DBMSs have always struggled with acquiring precise statistics on data, leading to poor query plan choices [154]. However, a data lake system may completely lack statistics on newly ingested data files. Consequently, incorporating adaptive query processing strategies is imperative in the cloud to enable a DBMS to dynamically modify query plans during execution based on observed data characteristics [97, 105, 163].

All the major cloud vendors now offer some variation of a managed data lake service. Since data lake systems backed by object stores are much cheaper per gigabyte than proprietary data warehouses, the legacy OLAP vendors (e.g., Teradata, Vertica) have extended their DBMSs to support reading data from object stores in response to this pricing pressure. Several independent systems are also in this space, including Databricks [105], Dremio [21], PrestoDB [63], and Trino [77].

3.4 NewSQL Systems

In the late 2000s, there were multiple distributed NoSQL DBMSs available designed to scale horizontally to support online applications with large number of concurrent users [110]. However, many organizations could not use these NoSQL systems because their applications could not give up strong transactional requirements. But the existing RDBMSs (especially open-source ones) were not able to (natively) scale across multiple machines. In response, NewSQL systems arrived in the early 2010s seeking to provide the scalability of NoSQL systems for OLTP workloads while still supporting SQL [95, 171]. In other words, these new systems sought to achieve the same scalability of NoSQL DBMSs from the 2000s but still keep the RM and ACID transactions of the legacy DBMSs from the 1990s.

There were two main groups of NewSQL systems. The first was in-memory DBMSs, including H-Store [144, 189] (commercialized as VoltDB [83]), SingleStore [69], Microsoft Hekaton [128], and HyPer [146]. Other startup offerings included disk-oriented, distributed DBMSs like NuoDB [47] and Clustrix [17].

Discussion: There has yet to be a dramatic uptake in NewSQL DBMS adoption [96]. The reason for this lackluster interest is that existing DBMSs were good enough for the time, which means organizations are unwilling to take on the costs and risk of migrating existing applications to newer technologies. Companies are more risk-
averse with changing OLTP DBMSs than with OLAP. If an OLTP DBMS fails, companies cannot execute the transactions they need to generate revenue. In contrast, an OLAP DBMS failure could be limited to temporarily inconveniencing an analyst or data scientist.

There were other restrictions in NewSQL DBMSs, such as only supporting a subset of standard SQL or bad performance on multi-node transactions. Some NewSQL products, like Microsoft’s Hekaton, were only available as an extension to a legacy DBMS, requiring the faster engine to use the slower DBMS’s interfaces.

NewSQL vendors also incorrectly anticipated that in-memory DBMS adoption would be larger in the last decade. Flash vendors drove down costs while improving storage densities, bandwidth, and latencies. Higher DRAM costs and the collapse of persistent memory (e.g., Intel Optane) means that SSDs will remain dominant for OLTP DBMSs.

The aftermath of NewSQL is a new crop of distributed, transactional SQL RDBMSs. These include TiDB [141], CockroachDB [195], PlanetScale [60] (based on the Vitess sharding middleware [80]), and YugabyteDB [86]. The major NoSQL vendors also added transactions to their systems in the last decade despite previously strong claims that they were unnecessary. Notable DBMSs that made the shift include MongoDB, Cassandra, and DynamoDB. This is of course due to customer requests that transactions are in fact necessary. Google said this cogently when they discarded eventual consistency in favor of real transactions with Spanner in 2012 [119].

3.5 Hardware Accelerators

There has been a hunt for a cost-effective hardware accelerator for DBMSs for the last 50 years. The promise is obvious: specialized hardware designed for a DBMS should easily outperform a conventional CPU.

In the 1980s, vendors fabricated custom hardware to accelerate DBMSs and marketed them as database machines [107]. Britton-Lee released the first commercial accelerator product (IDM/500) in 1981 [192] that contained a conventional CPU with a hardware accelerator that offloaded portions of a query’s execution. This accelerator targeted a small subset of the execution path, and was not cost-effective. Teradata introduced its own database machine that provided network hardware for sorting in-flight tuples (Y-net [1]), but it was dropped for a software-only solution [85]. All other custom hardware DBMS acceleration during the 1980s failed.

Instead of building custom hardware for DBMSs, the last 20 years have been about using commodity hardware (FPGAs, GPUs) to accelerate queries. This is an enticing idea: a vendor can get the benefits of a DBMS accelerator without the cost of fabricating the hardware.

Netezza was one of the first FPGA-based DBMSs that started in the late 1990s as a fork of PostgreSQL. It used an FPGA to accelerate searches on disk-resident pages, but originally could not search in-memory pages. Netezza corrected this limitation in a later version [2]. Swarmp64 attempted to sell a FPGA accelerator for PostgreSQL but switched to a software-only architecture without the FPGA before they were acquired [91]. Vitesse’s Deepgreen DB [81] is the only remaining FPGA-enhanced DBMS available from an ISV.

There is more activity in the GPU-accelerated DBMS market. Notable GPU DBMSs include Kinetics [35], Sqream [35], Brytlyt [13], and HeavyDB [48]. If data does not fit in GPU memory, then query execution is bottlenecked on loading data into the device, thereby rendering the hardware’s parallelization benefits moot.

Discussion: There are several conclusions that we can draw from the above analysis. First, these systems are all focused on the OLAP market and only for RDBMSs; there are essentially no data model implications to the discussion in this section. Also, OLAP workloads will continue to move aggressively to the cloud, but special-purpose hardware is not likely to find acceptance unless it is built by the cloud vendor.

Creating custom hardware just for a DBMS is not cost-effective for most companies. Commodity hardware avoids this problem but there is still the challenge of integrating the hardware into a DBMS. The reason why there are more GPU DBMSs than FPGA systems is because there are existing support libraries available for GPUs (e.g., Nvidia CUDA [169]). But cloud CPU-based compute resources are incredibly cheap due to economies of scale. The success of any accelerator is likely to be limited to on-prem databases, but this market is not growing at the same rate as cloud databases.

Even if one could get an accelerator to market that showed orders of magnitude improvement over existing technologies, that only solves half the problem needed for adoption and success. A hardware-only company must find somebody to add support for its accelerator in a DBMS. If the accelerator is an optional add-on to the DBMS, then adoption will be low and thus a DBMS vendor will not want to spend engineering time on supporting it. If the accelerator is a critical component of the DBMS, then no vendor would outsource the development of such an important part to an outside vendor.

The only place that custom hardware accelerators will succeed is for the large cloud vendors. They can justify the $50–100m R&D cost of custom hardware at their massive scale. They also control the entire stack (hardware and software) and can integrate their hardware at critical locations. Amazon did this already with their Redshift AQUA accelerators [102]. Google BigQuery has custom components for in-memory shuffles [89].

In spite of the long odds, we predict that there will be many attempts in this space over the next two decades.
3.6 Blockchain Databases

As of this writing, a waning database technology fad is blockchains. These are decentralized log-structured databases (i.e., ledger) that maintain incremental checksums using some variation of Merkle trees. These incremental checksums are how a blockchain ensures that the database’s log records are immutable: applications use these checksums to verify that previous database updates have not been altered.

The ideal use case for blockchain databases is peer-to-peer applications where one cannot trust anybody. There is no centralized authority that controls the ordering of updates to the database. Thus, blockchain implementations use a BFT commit protocol to determine which transaction to apply to the database next.

At the present time, cryptocurrencies (Bitcoin) are the only use case for blockchains. In addition, there have been attempts to build a usable DBMS on top of blockchains, notably Fluree [25], BigChainDB [12], and ResilientDB [136]. These vendors (incorrectly) promote the blockchain as providing better security and auditability that are not possible in previous DBMSs.

Discussion: We are required to place trust in several entities in today’s society. When one sells a house, they trust the title company to manage the transaction. The only applications without real-world trust are dark web interactions (e.g., money laundering). Legitimate businesses are unwilling to pay the performance price (about five orders of magnitude) to use a blockchain DBMS. If organizations trust each other, they can run a shared distributed DBMS more efficiently without wasting time with blockchains. To the best of our knowledge, all the major cryptocurrency exchanges run their businesses off traditional RDBMSs and not blockchain systems.

Blockchain proponents make additional meaningless claims of achieving data resiliency through replication in a peer-to-peer environment. No sensible company would rely on random participants on the Internet as the backup solution for mission-critical databases.

There is possibly a (small) market for private blockchain DBMSs. Amazon’s Quantum Ledger Database (QLDB) released in 2018 [65] provides the same immutable and verifiable update guarantees as a blockchain, but it is not decentralized (i.e., no BFT commit protocol). Amazon built QLDB after finding no compelling use case for a fully decentralized blockchain DBMS [108].

3.7 Summary

The key takeaways from the major technological thrusts in database systems are as follows:

- **Columnar Systems**: The change to columnar storage revolutionized OLAP DBMS architectures.
- **Cloud Databases**: The cloud has upended the conventional wisdom on how to build scalable DBMSs. Except for embedded DBMSs, any product not starting with a cloud offering will likely fail.
- **Data Lakes / Lakehouses**: Cloud-based object storage using open-source formats will be the OLAP DBMS archetype for the next ten years.
- **NewSQL Systems**: They leverage new ideas but have yet to have the same impact as columnar and cloud DBMSs. It has led to new distributed DBMSs that support stronger ACID semantics as a counter to NoSQL’s weaker BASE guarantees.
- **Hardware Accelerators**: We do not see a use case for specialized hardware outside of the major cloud vendors, though start-ups will continue to try.
- **Blockchain Databases**: An inefficient technology looking for an application. History has shown this is the wrong way to approach systems development.

4 Parting Comments

Our analysis of the last two decades in databases has several takeaways. Unfortunately, some of these are repeats of the warnings from the 2005 paper.

**Never underestimate the value of good marketing for bad products.** The database market is highly competitive and lucrative. This competition drives vendors to claim that their new technologies will solve all sorts of problems and change developers’ lives for the better. Every developer has struggled with databases before, so they are especially amenable to such marketing. Inferior DBMS products have succeeded via strong marketing despite the existence of better options available at the time: Oracle did this in the 1980s, MySQL did this in the 2000s, and MongoDB did this in the 2010s. These systems got enough traction early on to buy them time to fix the engineering debt they accumulated earlier.

**Beware of DBMSs from large non-DBMS vendors.** One interesting aspect in the last ten years of databases is the trend of tech companies building DBMSs in-house that they then spin out as open-source projects. All these systems started life as purpose-built applications for a tech company. The company then releases the DBMS as an open-source project (often pushed to the Apache Foundation for stewardship) in hopes to achieve “free” development from external users.

Some times they come from large companies that can afford to allocate resources to developing new systems. Notable examples include Meta (Hive [197], Presto [63], Cassandra [14], RocksDB [68]) and LinkedIn (Kafka [33], Pinot [59], Voldemort [82]). Other systems are from start-ups building a data-intensive product where they felt the need to also build a DBMS. The most successful examples are 10gen (MongoDB) and PowerSet (HBase), but there also many failed endeavors.

This trend to avoid “not invented here” software is partly because many companies’ promotion path favors
engineers who make new internal systems, even if existing tools are sufficient. But this perversion led many teams without DBMS engineering experience to undertake building a new system. One should be wary of such systems when a company first open-sources them, as they are almost always immature technologies.

**Do not ignore the out-of-box experience.** One of the salient selling points of many non-relational DBMSs is a better “out-of-box” experience than RDBMSs. Most SQL systems require one first to create a database and then define their tables before they can load data. This is why data scientists use Python notebooks to analyze data files quickly. Every DBMS should, therefore, make it easy to perform in situ processing of local and cloud-storage files. DuckDB’s rising popularity is partly due to its ability to do this well.

Vendors should also consider additional challenges that customers will inevitably face with databases, including physical design, knob tuning, schema design, and query tuning. There is a crucial need for what one of us calls “self-driving” DBMSs [173].

**Developers need to query their database directly.** Most OLTP applications created in the last 20 years primarily interact with databases via an abstraction layer, such as an endpoint API (e.g., REST, GraphQL) or an object-relational mapper (ORM) library. Such layers translate an application’s high-level requests into database queries. ORMs also automatically handle maintenance tasks, such as schema migrations. One could argue that since OLTP developers never write raw SQL in their applications, it does not matter what data model their DBMS uses as these layers hide it.

ORMs are a vital tool for rapid prototyping. But they often sacrifice the ability to push logic into the DBMS in exchange for interoperability with multiple DBMSs. Developers fall back to writing explicit database queries to override the poor auto-generated queries. This is why using a RDBMS that supports SQL is the better choice.

**The impact of AI/ML on DBMSs will be significant.** How DBMSs should interact with modern AI/ML tools has recently become a crucial question, especially with the advent of LLMs (e.g., ChatGPT). Although this field is moving rapidly, we offer a few initial comments.

There is a resurgence in using natural languages (NLs) to query databases due to advancements in LLMs at converting NL to query code (e.g., SQL) [133]. Some have even suggested that such AI-powered query interfaces will render SQL obsolete. NL interfaces are an old research topic that dates back to the 1970s [139], but which historically has poor outcomes and thus little widespread use [88]. We acknowledge LLMs have impressive results for this task but caution those who think NL will replace SQL. Nobody will write OLTP applications using an NL, as most generate queries using ORMs. For OLAP databases, NL could prove helpful in constructing the initial queries for exploratory analysis. However, these queries should be exposed to a dashboard-like refinement tool since English and other NLs are rife with ambiguities and impreciseness.

There is a reluctance to depend on current LLM technology for decision-making inside the enterprise, especially with financial data. The biggest issue is that the output of an LLM is not explainable to a human. Second, LLM systems require more training data than “traditional” ML systems (e.g., random forests, Bayesian models). Companies generally cannot outsource the creation of training data for these models to unskilled people. For these reasons, the uptake of LLMs for enterprise data will be cautiously slow.

Lastly, there is a considerable amount of recent research on using AI/ML to optimize the DBMSs [174]. Examples include ML-oriented query optimizers [152, 156], configuration tuners [200, 204], and access methods [151, 193]. Although such ML-assisted optimizations are powerful tools to improve the performance of DBMSs, it does not obviate the need for high-quality systems engineering.

### 5 Conclusion

We predict that what goes around with databases will continue to come around in upcoming decades. Another wave of developers will claim that SQL and the RM are insufficient for emerging application domains. People will then propose new query languages and data models to overcome these problems. There is tremendous value in exploring new ideas and concepts for DBMSs (it is where we get new features for SQL). The database research community and marketplace are more robust because of it. However, we do not expect these new data models to supplant the RM.

Another concern is the wasted effort of new projects reimplementing the same components that are not novel but necessary to have a production-ready DBMS (e.g., config handlers, parsers, buffer pools). To accelerate the next generation of DBMSs, the community should foster the development of open-source reusable components and services [112, 176]. There are some efforts towards this goal, including for file formats (see Sec. 3.3), query optimization (e.g., Calcite [104], Orca [186]), and execution engines (e.g., DataFusion [18], Velox [175]). We contend that the database community should strive for a POSIX-like standard of DBMS internals to accelerate interoperability.

We caution developers to learn from history. In other words, stand on the shoulders of those who came before and not on their toes. One of us will likely still be alive and out on bail in two decades, and thus fully expects to write a follow-up to this paper in 2044.
References


[37] LeveDB. https://github.com/google/leveldb, 2024.


[73] Teradata – creating an array data type. https://docs.teradata.com/r/50v2/dw1/hf13/mdb6w000h0/1un3k-j-t3cm066lf4yuxlw, 2024.


ABSTRACT
We consider the problem of fine-grained hardware profiling, i.e., profiling the hardware while the desired section of the program is executing. Although this requirement is frequently encountered in practice, its importance has not been emphasized in literature so far. In this work, we compare and validate three tools for performing fine-grained profiling on Linux platforms – perf, PAPI, and a homegrown tool PMU-metrics. Perf has been used in the past for fine-grained profiling in an erroneous manner, producing inaccurate metrics as a result. On the other hand, PAPI and PMU-metrics produce accurate metrics for profiling at the ms-scale, while PMU-metrics enables profiling even at the µs-scale. Thus, we hope that our analysis will help systems practitioners choose the right tool for performing fine-grained profiling at different time scales.

1. INTRODUCTION
Hardware profiling is an important part of developing efficient data processing methods. This process frequently involves using fine-grained profiling to drill down different parts of the design to obtain valuable performance metrics such as cache misses, CPU cycles, TLB misses, etc. Fine-grained profiling also enables development of online adaptive algorithms that utilize performance metrics to make decisions at runtime [12, 16]. However, it can be challenging to obtain accurate hardware metrics at sub-ms time scales.

To highlight the challenge of profiling fast data processing methods, consider the example of a hash table. An optimized hash table typically takes a few hundred cycles for each probe operation, i.e., < 0.1µs. However, existing tools do not support accurate and non-intrusive profiling at such small time scales, thus forcing us to increase the duration of profiling using batching as shown in Listing 1.

We note that batching may not be possible if the target component is deeply integrated in a system with many hierarchical modules, and it may be important to examine the performance of just the target as it interacts with the rest of the system. For developing online adaptive algorithms, it may be necessary to collect metrics at a certain granularity for the technique to be effective. Fine-grained profiling enables isolating the performance of different components of a system, thus making it an essential part of developing and tuning systems.

Given the importance of fine-grained profiling, some questions that arise are: Which tools are available for this task? At what time scales do they work? What trade-offs do they offer? These questions have not been addressed in prior work utilizing fine-grained profiling [13, 14]. To address these questions, we make the following contributions:

- Applications of fine-grained profiling: To highlight the importance of fine-grained profiling, we study two applications. First, we profile the performance of a hash table as shown in Listing 1 [12]. Second, we study BLARE, an adaptive log processing framework for regular expression (regex) matching [16] that utilizes fine-grained profiling.
struct Perf {
    profile(int id, function<void()> body) {
        /* Launch perf stat to track parent */
        pid_t pid;
        stringstream parent_id;
        parent_id << getpid();
        pid = fork();
        if (pid == 0) {
            exit(exec1("/usr/bin/perf", "perf", "stat", "-e", "L2_RQSTS.REFERENCES", "-p", parent_id.str().c_str(), nullptr));
        }
        Perf::profile(i/BATCH_SIZE, [&]() {
            /* Collect metrics for each batch */
            hm.processReqs(&reqs[i:i+BATCH_SIZE]);
        });
    }
};

void main() {
    ...
    for (int i=0; i<N; i+=BATCH_SIZE) {
        Perf::profile(i/BATCH_SIZE, [&]() {
            /* Collect metrics for each batch */
            hm.processReqs(&reqs[i:i+BATCH_SIZE]);
        });
    }
}

Listing 2: Fine-grained profiling with perf. A child perf process is launched to track the parent while the section of code to profile (body) is executed.

- **Comparing multiple profiling tools:** We validate the metrics from three representative tools for profiling on Linux platforms – perf, PAPI, and PMU-metrics. The former is a popular command-line profiling tool, while the latter two libraries perform profiling from within the program.

- **Debunking erroneous usage of perf:** perf can potentially be used for fine-grained profiling as shown in Listing 2. However, we show that perf yields incorrect metrics and interferes with execution when used in this manner.

- **Fine-grained profiling at µs-scale:** PMU-metrics enables profiling at the µs-scale while imposing low overhead (+8% execution time) and providing good accuracy (2% error), which facilitates execution of BLARE with minimal overhead (10%).

- **Overall recommendations:** We recommend using either PAPI or PMU-metrics for profiling at the ms-scale, while PMU-metrics should be preferred for profiling at the µs-scale.

Thus, we evaluate the validity, accuracy and overhead of perf, PAPI, and PMU-metrics at time scales ranging from sub-s to sub-ms by studying applications to hash tables (Listing 1, [12]) and BLARE [16].

2. BACKGROUND

In this section, we briefly discuss performance monitoring in CPUs (§2.1–§2.2) to highlight the differences between perf, PAPI, and PMU-metrics, followed by describing how fine-grained profiling can be performed using each of the three tools (§2.3).

2.1 Performance Monitoring in CPUs

Present-day CPUs come with an integrated performance monitoring unit (PMU) that can track hardware events such as cache misses. Typically, PMUs have a fixed number of units that can independently count processor events. The set of supported hardware events and the implementation of the PMU varies across different CPU vendors. Even for the same vendor, the supported hardware events can vary between different models of CPUs.

2.2 Interfacing the PMU

In a nutshell, the interface to the PMU is a collection of registers that can be programmed for tracking hardware events. The low-level interface to the PMU varies across different CPU vendors. Even for the same vendor, the supported hardware events and the implementation of the PMU varies across different CPU vendors. Thus, accessing hardware metrics using the perf library is equivalent to reading from a file. Some prominent vendors supported by perf include Intel, AMD, ARM and IBM.

A cheaper interface compared to perf_event_open is the rdpmc [6] x86 assembly instruction that can directly read metrics from the performance counter registers without initiating a system call. The rdpmc instruction requires knowledge of which registers contain the output of the programmed PMU units.

2.3 Fine-Grained Profiling on Linux

We compare three different tools for fine-grained profiling – perf [11], PAPI [15], and PMU-metrics [10]. Both perf and PAPI are built on the perf_event library, whereas PMU-metrics uses a bare metal approach of directly accessing the PMU hardware counters. Below we describe how each of these tools can be used for fine-grained measurement of L1 cache misses on an Intel Xeon Silver 4114 CPU belonging to the Skylake microarchitecture family.

2.3.1 perf

Linux perf is a popular command-line profiling tool. The perf stat command can be used for fine-grained profiling as shown in Listing 2 – a child
void main () {
    ... 
    int EventSet1 = PAPI_NULL;
    long long metrics[1];
    PAPI_library_init ( PAPI_VER_CURRENT);
    PAPI_create_eventset (& EventSet1);
    /* Add PAPI_L1_TCM preset event */
    PAPI_add_named_event ( EventSet1 ,
        "PAPI_L1_TCM");
    for ( int i =0; i<N; i+= BATCH_SIZE ) {
        /* Start PAPI */
        PAPI_start ( EventSet1);
        /* Collect metrics for each batch */
        hm. processReqs (& reqs [i:i+BATCH_SIZE]);
        /* Stop PAPI */
        PAPI_stop (EventSet1, metrics); 
    }
}

Listing 3: Fine-grained profiling with PAPI. PAPI provides API functions to program, start, and stop performance event counters.

perf process is launched to track the parent when the code to profile is executed, thus (presumably) profiling only the desired section of the program. In Listing 2, L1 cache misses are measured using the event L2_RQSTS.

2.3.2 PAPI
PAPI is designed for fine-grained “first person” profiling, i.e., profiling from within the program. PAPI provides API functions to program the PMU (PAPI_add_named_event) and start/stop counting events (PAPI_start/stop) as shown in Listing 3. The library provides a collection of preset events with the goal of implementing cross-platform compatibility. Internally, the library uses one or more hardware-specific events (referred to as native events) to estimate the output of the preset event. In Listing 3, we use the preset event PAPI_L1_TCM to count L1 cache misses. Note that PAPI can also be used to directly count hardware-specific events.

2.3.3 PMU-metrics
PMU-metrics provides a fine-grained interface similar to PAPI, as shown in Listing 4. However, there are two key differences between PAPI and PMU-metrics. First, PAPI uses the perf_event_open system call to program the PMU at runtime, whereas the PMU-metrics library directly programs the PMU registers using wrmsr [9]. Second, PAPI relies on the file interface provided by the perf_event library to access the hardware metrics, whereas PMU-metrics uses the cheaper rdpmc assembly instruction to directly read the performance counter registers. Currently, PMU-metrics supports Intel CPUs released since 2007, and can be extended for other vendors.

Listing 4: Fine-grained profiling with PMU-metrics, with API functions to read performance counters using rdpmc x86 assembly instruction. The setup.sh script programs the PMU with performance events before running the application.

3. WHICH TOOLS ARE CORRECT?
In this section, we validate the metrics obtained from perf, PAPI, and PMU-metrics by profiling a hash table data structure (Listing 1).

3.1 Workload
We consider the example introduced in Listing 1 of measuring hardware metrics for each batch of fetch requests issued to a hash table. We process 100M fetch requests in batches of 10M (approximately 330ms processing time per batch without profiling), i.e., 10 batches of fetch requests are processed overall. The goal is to measure five hardware metrics for each batch – core cycles, instructions retired, L3, L2 and L1 cache misses. The workload executed is the same in all experiments.

3.2 Experimental Setup
All the experiments in this paper have been run on a dedicated server machine with an Intel Xeon Silver 4114 10-core CPU. The running process has been pinned to a specific core (two cores in case of profiling with perf) using taskset to mitigate the overhead of context switching. The frequency scaling governor has been set to “performance”, which drives the cores at maximum possible frequency, i.e., 3GHz on our hardware. All the experiments have been repeated five times, and we compare the median metrics in all cases.

In addition to profiling using the libraries discussed in §2, we use the clock_gettime [2] function to measure total time elapsed for processing all...
Figure 1: Comparing the aggregate metrics obtained from perf, PAPI, and PMU-metrics. The workload involved measuring hardware metrics individually for ten batches of 10M fetch requests each (approximately 330ms processing time per batch without profiling). The metrics obtained from perf are significantly different from the remaining two libraries, while PAPI and PMU-metrics report similar metrics in all cases (within 4% for core cycles, L3, L2, and L1 cache misses; within 10% for instructions retired). We validate the core cycles obtained from each of the libraries against estimated core cycles obtained from clock_gettime, and the three libraries have 23.9%, 99.3%, and 99.7% accuracy respectively as indicated in (a).

Figure 2: Increase in total execution time of the workload when different libraries are used for profiling. Relative to the baseline where profiling is not performed, perf increases execution time by 40% while PAPI and PMU-metrics impose negligible overhead of 3% and 2% respectively.

the 100M fetch requests each time. This function is known to provide good precision and accuracy for measuring time on Linux platforms [8].

3.3 perf vs PAPI vs PMU-metrics

Listings 2-4 show how to perform fine-grained profiling using perf, PAPI, and PMU-metrics respectively. Below we discuss the results obtained.

3.3.1 Overhead of profiling

Fig. 2 shows the increase in total running time of the whole workload (100M requests) when profiling using different tools. Compared to the baseline where profiling is not performed, PAPI and PMU-metrics impose a negligible overhead of 3% and 2% respectively. In contrast, profiling with perf causes a significant increase of 40% in total execution time of the workload, which suggests interference in the hash table execution from the child perf process.

3.3.2 Comparing the metrics obtained

We compare the metrics obtained from perf, PAPI, and PMU-metrics in Fig. 1. We find that the metrics obtained from perf are 3-4× lower compared to PAPI and PMU-metrics, both of which report similar metrics in all cases\(^1\). Since the same workload is being executed, we can conclude that not all of these tools give the correct metrics. This raises the question which of these tools are reliable.

3.3.3 Which metrics are correct?

To establish the correctness of the metrics, we validate the core cycles obtained from a tool against the core cycles estimated using the clock_gettime function. Given time \(t\) has elapsed while executing processRequests on a core with frequency \(f\),

\[
\text{estimated core cycles} = t \times f
\]

Fig. 1a plots the measured and estimated core cycles for each of the tools\(^2\). Both PAPI and PMU-metrics attain a high accuracy of 99.3% and 99.7% respectively, indicating that their measurements are correct. Since the same profiling methodology is applied for other hardware events, we conclude that the metrics obtained from these tools are reliable.

On the other hand, the cycles obtained from perf are 76% lower than estimated, which suggests that the metrics obtained using perf exhibit a large deviation from the true value. This observation is further corroborated by the fact that the aggregate execution time reported by perf is 1.2s, which is far lower than the value obtained using clock_gettime (4.3s). Thus, we conclude that perf produces incorrect metrics and interferes with execution (§3.3.1) when used as shown in Listing 2. We discuss the reasons behind these observations in Fig. 3.

\(^1\)The difference in metrics is within 4% for core cycles and cache misses. A deviation of 10% in instructions executed is encountered because PMU-metrics and PAPI use different hardware events (INST_RETD.ANY vs INST_RETD.ANY_P respectively) on Skylake CPUs.

\(^2\)Note that the time elapsed, and thus the estimated core cycles are different for each tool.
Figure 3: Investigating fine-grained profiling with perf. The metrics obtained are lower than estimated because profiling starts at later point compared to the execution of `processRequests` by the parent process. The execution time of `processRequests` increases by 30% (330ms vs 429ms) as forking a child process triggers the copy-on-write mechanism that causes page faults and TLB flushes in the parent. Additional overhead is imposed by `fork`, `kill`, and `waitpid`, which increases the total execution time for a batch by 37% relative to baseline (330ms vs 452ms).

4. PROFILING AT THE μS SCALE

In this section, we evaluate the accuracy and overhead of PAPI and PMU-metrics as we reduce the granularity of profiling to the μs-scale. We consider two applications – profiling a hash table (§4.1), and adaptive log processing with BLARE (§4.2).

4.1 Reducing the granularity of profiling

4.1.1 Experimental setup

We consider the example of profiling a hash table as shown in Listing 1. We decrease the batch size of requests from 10M to 100, which decreases the granularity of profiling from 0.33s to 3.3μs. Total 100M fetch requests are processed, and we compare the median metrics of five runs in each case. For both the tools, the metrics measured at a coarse granularity of 10M requests are used as the baseline, i.e., the performance of a tool is compared to itself.

4.1.2 Results

Fig. 4a and 4b respectively show the accuracy and overhead of PAPI and PMU-metrics as we reduce the batch size. At ms-scale granularity (batch size 100k), both the tools remain consistent with their respective baselines, while at the μs-scale (batch size 100) we observe a 12% and 2% deviation in instructions reported by PAPI and PMU-metrics respectively. A similar trend is observed for other hardware metrics as well. We note that PAPI imposes a high overhead which increases the total running time by 268% at the μs-scale, while PMU-metrics imposes a low overhead of just 8%.

Figure 4: (a) Total instructions reported and (b) total execution time as we reduce the granularity of profiling from 10M requests (≈ 0.33s) to 100 requests (≈ 3.3μs). Total 100M fetch requests are processed in all cases. At μs-scale (batch size 100), PAPI and PMU-metrics show a deviation of +12% and +2% compared to their respective baselines, while imposing 268% and 8% overhead respectively.

Overall, we conclude that either libraries can be used at the ms-scale, while we recommend using PMU-metrics at the μs-scale as it imposes low overhead (8%) and maintains good accuracy (2% error).

4.2 Adaptive log processing with BLARE

4.2.1 Experimental setup

Regex matching is a frequently encountered task while processing unstructured log data. BLARE [16] uses an adaptive algorithm to choose an efficient strategy for regex evaluation at runtime. In a nutshell, BLARE uses a multi-armed bandit approach which involves measuring the “reward” for different attempts of a given strategy, which informs future attempts. We measure the reward as the number of core cycles elapsed for a given attempt. To this end, we employ three different methods for counting cycles – `perf_event` [5], PAPI (Listing 3), and PMU-metrics (Listing 4). We refer the reader to [3] for details of using `perf_event` for counting cycles. Experiments are run on a dataset [7] of size 100MB as detailed in [1]. Each experiment is repeated ten times, and we report the median metrics in all cases.

4.2.2 Results

Fig. 5 shows the running time of Blare with different cycle counting methods and the overhead compared to the best regex evaluation strategy. Un-
surprisingly, PMU-metrics imposes the least overhead (10%) given that it utilizes an the rdpmc instruction to directly access performance counters. Cycle counting using perf_event (20% overhead) involves issuing system calls (ioctl) since performance counting is exposed via a file interface [5]. PAPI also utilizes the perf_event interface. However, it is a dynamically linked library, and imposes higher overhead (25%) compared to perf_event.

5. PAPI OR PMU-METRICS?

PAPI utilizes the PMU through the perf_event kernel library. Thus, the OS manages the hardware and automatically reprogrammes the PMU in the presence of context switches. However, the file descriptor interface of the perf_event library for accessing performance counters becomes expensive at the µs-scale (+268% execution time, Fig. 4b) and also affects the accuracy (12% deviation in instructions retired, Fig. 4a). On the other hand, PMU-metrics uses rdpmc to access performance counter registers, that enables the collection of metrics at µs-scale with good accuracy (2% deviation in instructions retired, Fig. 4a) and low overhead (+8% execution time, Fig. 4b). However, PMU-metrics directly programs the hardware, thus requiring privileged and exclusive access on part of the user.

6. CONCLUSIONS

Fine-grained profiling is an important requirement encountered by systems practitioners. Our analysis reveals that not all tools are suitable for this requirement. In particular, we discourage performing fork-based profiling using command-line tools such as perf, as it can result in incorrect metrics and impact the performance of the parent process due to the overhead of the copy-on-write mechanism. Tools designed for “first-person” fine-grained profiling such as PAPI and PMU-metrics should be preferred. Both of these libraries provide accurate metrics at the ms-scale, while we recommend using the lighter PMU-metrics library at the µs-scale.

7. ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under grant OAC-1835446 and a David DeWitt Fellowship.

8. REFERENCES

Reminiscences on Influential Papers

This issue’s contributors all work at the intersection of machine learning and databases. Coincidentally, three of them picked papers that received the VLDB 10-Year Best Paper award, and one of them picked a paper that, in my opinion, should receive the CIDR Test-of-Time award at some point. Their write-ups highlight very well why these four papers still continue to leave a mark on us personally and the real-world data systems. 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

Matthias Boehm
Technische Universität Berlin, Germany
matthias.boehm@tu-berlin.de

Stratos Idreos, Martin L. Kersten, and Stefan Manegold.

Database Cracking.

Selecting a single paper for the Reminiscences on Influential Papers (RIP) column is generally a difficult task. For me, however, it was surprisingly easy because the database cracking paper really stands out to me, for both, personal and technical reasons.

One of the first international conferences I attended as a young PhD student—working on cost-based optimization of message-oriented middleware—was ICDE 2008 in Cancún, Mexico. Apart from the magnificent environment, Martin Kersten gave an awe-inspiring keynote on “The Database Architecture Jigsaw Puzzle”, which described the pitfalls and opportunities of high-risk projects using database cracking as one of the examples. At its core, database cracking is all about workload-driven data reorganization in column stores. As queries come in, they partition the columns into qualifying and non-qualifying data, which in turn drives the reorganization of data, and thus, improves future data access. I came out of this keynote truly inspired, by the simplicity and effectiveness of the ideas, but also by the focus on system-oriented research of actually building MonetDB. Apparently, I was not alone – there are famous stories about Goetz Graefe wandering the beaches of Cancún after the keynote, thinking about adaptive indexing and merging.¹

Back at TU Dresden, my PhD supervisor Wolfgang Lehner had established a weekly paper session, cycled through all members of the group. So on my next opportunity, I deliberately chose to share the database cracking paper. In preparation of this presentation, I went through all the background start-

¹Interestingly, these stories are true. On confirming with Goetz Graefe, he shared the following: “Yes, that’s precisely how it happened. Something about database cracking was bothering me, but I couldn’t put my finger on it precisely. So I spent 1 1/2 days walking back and forth on the Cancún beach thinking about it. Much later I came to think of database cracking as “quicksort on a funny schedule”. Martin Kersten later agreed with that summary description. In contrast (or “in duality”), adaptive merging is “merge sort on a funny schedule”. One important difference is that quicksort and database cracking work best within memory, whereas merge sort and adaptive merging also work for external sorting and external indexes. The other related anecdote is that I visited CWI and presented adaptive merging as an alternative to database cracking. At first, that didn’t go over very well but within 1/2 hour the CWI team was engaging with me on deep understanding and comparisons and that eventually led to a productive collaboration. I am still impressed with their professionalism and their open minds!"
ing from the influential VLDB Journal 2000 paper “Optimizing database architecture for the new bottleneck: memory access”, over the core database cracking papers from CIDR 2005 and 2007, to advanced techniques for handling updates (SIGMOD 2007) and partial side-ways cracking for multi-attribute queries and tuple reconstruction (SIGMOD 2009). Besides the core cracking ideas, I was very impressed by the breadth and depth of the entire eco-system around the core systems MonetDB/SQL and MonetDB/X100, but also MonetDB/XQuery (semi-structured data), MonetDB/RAM (information retrieval and sciences), MonetDB/Armada (evolving databases), MonetDB/SkyServer (astronomy), and MonetDB/DataCell (streaming). After this paper session, multiple team members wrote papers on applying database cracking ideas to other areas (e.g., spatial indexing, index optimization). Not long after, I attended SIGMOD 2009—participating in the first SIGMOD programming contest—in Providence, Rhode Island. At this conference, Milena, Martin, Niels, and Romulo received a best-paper runner-up for another influential paper on “An architecture for recycling intermediates in a column-store”, and without knowing it, I was sitting next to Milena during the business meeting where they got the recognition. During my later time at IBM Research – Almaden, I also worked with Romulo who joined our larger data management group as a postdoc.

Apart from these anecdotes, the database cracking paper and larger MonetDB eco-system had a tremendous impact on me. First, I fully adopted a system-oriented research philosophy of building actual open-source systems (currently Apache SystemDS\(^2\) and DAPHNE\(^3\)) and integrating all our research into these umbrella systems, which provides grounding and a deeper understanding of existing trade-offs. This approach led to fewer but, as I believe, much better papers. Second, several of our research sub-projects were inspired by ideas from MonetDB. Examples are compressed linear algebra (applying column compression to numeric matrices), optimizing operator fusion plans (tuned vectorized execution for operator DAGs), and lineage-based reuse (recycling intermediates in ML systems). Third, in my data management courses, we still cover database cracking—typically in lectures on physical design—as a beautifully simple yet effective idea with connections to other techniques such as indexing, partitioning and partition pruning, as well as materialized views and result caching. Fourth, and finally, I largely agree with Martin on how to assemble teams for system-oriented research. After Martin’s too early death in 2022, CIDR 2023 held a great memorial sharing a variety of stories. Having played recreational soccer for 25+ years, I unknowingly used the same metaphor as Martin of assembling a research team like a soccer team with people of diverse backgrounds and skills (goalkeeper, defenders, forwards). So RIP (rest in peace) Martin Kersten, and thank you for inspiring generations of database researchers.

---

Matteo Interlandi
Microsoft Gray Systems Lab, USA
matteo.interlandi@microsoft.com

Thomas Neumann.
Efficiently Compiling Efficient Query Plans for Modern Hardware.


When I was asked to write about a paper that had a significant impact on me, one immediately springs to mind. While other papers had a more substantial influence on my PhD studies, this particular one not only shaped my view of database Query Processing (QP), but also consistently resurfaced over the years even after I graduated.

This paper, which won the Test of Time award in 2021, needs no introduction. It pioneered compiler-based QP, a paradigm shift from the traditional interpreted processing model for queries. Traditionally, databases have employed an interpreter model where single tuples or vectors of tuples are pulled from downstream data sources or operators using an iterator. This paper introduced several contributions that I continuously re-encountered in my life as a researcher.

Firstly, it proposed the division of query plans for execution into pipelines, identified by pipeline-breaking operations such as hash table creation. This method allows for more efficient processing and better utilization of modern hardware, while simultaneously blurring the boundaries between operators. Secondly, it flipped the conventional pull-based iterator model into a push-based execution. This approach enhances data flow and boosts the efficiency of query processing. Lastly, it advocated the use of compilers, like LLVM, to generate efficient code instead of relying on interpretation. This has paved the way for new possibilities in optimiz-

---

\(^2\)https://systemds.apache.org/

\(^3\)https://github.com/daphne-eu/daphne
ing database performance and has established a new benchmark in the field.

The influence of this paper extends beyond its initial publication, inspiring numerous follow-up works and stimulating discussions within the database community. For instance, several subsequent works have focused on enhancing the approach by finding ways to reduce the compilation overheads. There has been a continuous debate on whether a compiler approach surpasses a vectorized one. Both sides have significant proponents, and this discussion has led to a deeper understanding of the strengths and weaknesses of each approach. Some more recent works have attempted to amalgamate the best of both worlds by integrating vectorized execution with compilation. For example, the latest paper from Wagner et al., presented at ICDE 2024 [1], introduces an interesting incremental compilation framework. Lastly, the definition of “modern hardware” has evolved since 2011. Lately, we are seeing a significant trend towards hardware specialization. The lessons from this paper remain not only relevant but also applicable and valuable in the context of these specialized hardware environments. For instance, I have spent most of my time in the last few years on understanding and implementing QP on GPUs. And as far as I know, compilation for QP on GPUs is still not mainstream, although possible [2].

In conclusion, the influence of this paper on both my personal development and the broader database community extends far beyond its original contributions. It has ignited new research directions and ongoing discussions that continue to shape the field of database query processing. The paper’s enduring relevance and its significant role in advancing database technologies are a testament to its impact.


learning models. Moreover, the rigorous analysis and theoretical arguments of this work appealed to the engineer in me who always looks for correctness on the designs of proposed solutions. From that moment on I was convinced that I would always try to obtain such a deep understanding of a technical problem as the one demonstrated by the authors of this work. In retrospect, it was the perfect paper for me and I highly recommend that anyone interested in large-scale probabilistic reasoning reads this paper.

Madelon Hulsebos  
University of California, Berkeley, CA, USA  
madelon@berkeley.edu


In this paper, Cafarella et al. construct a large dataset of tables collected from web pages and describe a search system to navigate such tables at scale. The authors also introduce the Attribute Correlation Statistics Database (ACSDb), which they analyze and showcase in applications such as schema completion and attribute synonym identification. It has received VLDB’s 10-year Test-of-Time award, illustrating its high and broad impact, as also described in the paper “Ten Years of WebTables” in 2018.

While data management research is typically concentrated on systems and algorithms for storing and processing data, less attention is given to the content of the data itself. Large collections of data objects help us better understand what data we are actually dealing with, and anticipate it. WebTables is no different and helps us better understand the data stored in (web) tables. I learned about the WebTables project while helping out with a study comparing the effectiveness of data visualizations for synthetic data versus real-world data. Turns out, synthetic tables are nothing like real-world tables. From that point, I have become fascinated with the nature of real-world tables—the variation in shapes, sizes, structure, semantics, messiness. Especially during my research on tables stored in CSVs: it is wild to see what people store in CSVs, and how this stretches our assumptions.

Beyond data insights, this research was instrumental in surfacing tables from the web through Google search and unlocking advancements in data management research (e.g. data integration), as highlighted in the “Ten Years of WebTables” paper. But to me personally, this paper has been important because large-scale datasets are essential in training machine learning (ML) models, and so has WebTables\textsuperscript{5} been important in the development of ML models for structured data. These tables actually powered my first neural models for semantic table understanding, the starting point of my PhD.

The WebTables paper illustrates the value of patterns across tables by leveraging correlations among attributes in the millions of tables in the ACSDb database for, for example, automated schema completion. A similar idea underpins the GloVe word vectors, where word co-occurrence matrices are extracted from a large collection of texts, and used to obtain semantic word vectors through matrix factorization. A few years later, the transformer architecture fueled the training of token-level embeddings, and well, the rest is history. Recent research in data management, including my own, now intends to extend these capabilities to structured data.

Finally, this paper is also one of the early examples I know of a “dataset paper” that shows the value of constructing, documenting, and analyzing a dataset at scale. 15 years later, we even have tracks at VLDB and NeurIPS dedicated to the notion of dataset papers, and rightfully so. While the data management community, myself included, introduced new collections of tables over the past couple of years, I like to emphasize that we still have just scratched the surface of what is needed in terms of structured data to truly get to what is possible with AI. While WebTables unlocked many applications on web data, there is a pressing need for large-scale datasets representative of offline databases and interactions with this kind of data, which are harder to obtain at scale. So, I hope that my reflection on the importance of data can also serve as an influence on future contributions.

To close, I want to sincerely thank Pınar for inviting me to write this piece: it has been a long time since I have written a personal reflection on research. It reminds me how refreshing and liberating it is, complementary to writing research papers, while it is also fruitful for developing new questions and ideas.

\textsuperscript{5}The WebTables corpus, unfortunately, was never published, but similar datasets were later published through Web Data Commons and Dresden Web Table Corpus.
ADVICE TO MID-CAREER RESEARCHERS

We are starting a new series to provide advice to mid-career researchers. There are a number of programs that SIGMOD organizes for researchers at the beginning of their careers (PhD Symposium and the like) and senior people do not (or should not) need much help. There are considerable challenges for those who are about to transition from an early researcher to a more senior role. In academia, these are people who are about to get tenured that comes with starting to think of moving from shorter-term research objectives to longer-term ones. In industrial research, this corresponds to the transition from participating in projects to initiating and leading them. As a community we don’t seem to talk about these challenges much. That is the gap this series attempts to fill. We will get the views of senior researchers from diverse backgrounds and diverse geographies. We will continue as long as we find original advice and the views are not repetitions.

M. Tamer Özsu
University of Waterloo

The Long and Winding Road to Mid-Career Academia

Angela Bonifati
angela.bonifati@liris.cnrs.fr
Lyon 1 University, CNRS Liris and IUF (France)

I am extremely excited to contribute to this series and I would like to thank Tamer for giving me this great opportunity. When I started my PhD in Italy at the end of the nineties, it was still the time when you felt intimidated by your PhD advisor and, in general, by senior colleagues when you meet them at conferences. I did not imagine becoming one of them in a few years. Indeed, I remember that the first scientific event that I ever attended was the EDBT Summer School in La Baule-les-Pins (France)\(^1\) in 1999. I was a first year PhD student at that time, and I was meticulously attending all the lectures, taking notes and working on them during my free time. I learned over the years that an equally important part of conferences are the social interactions, the networking activities and the hobnobbing. When a senior colleague pitches an idea to you, you can agree or not agree. It is time to fight back with proper arguments! All along during our careers, we strive to reach independent critical thinking.

As long as you grow in your career from a fresh PhD student to a mid-career researcher and a senior researcher, you are on the right path for scientific independence. You have the power to decide which topics interest you, which collaborations you want to pursue, which students you want to hire, how to best supervise and mentor students and so on. The downside is that with time you become more in demand. The beauty of this independence process is that you focus on what you like, and you are more productive. It was my case since I started my research journey working on semi-structured data (all along my PhD, postdoc and a few more years after that) until I became a full professor in France in 2011. In the meanwhile, as soon as I approached the mid-career stage, I started working on a couple of new topics taking a certainly risky path. I began to be interested in data integration, which was a hot topic at the time (and still is). Data integration was also a topic broadly studied in the database community, but I decided to focus on novel aspects, such as user interventions and quality of the transformations. I decided to contact two senior colleagues to write a book on schema matching and mapping in the Data-Centric Series on Data management in 2011. The

\(^1\) For interested people, I amazingly found some pictures of the summer school on the Internet.
book is a great collection of chapters with contributions from several colleagues in our community.

As soon as NoSQL databases started to become popular, I began to investigate problems around graph data management. My first PhD students on this topic were very productive, and I could quickly build several new scientific collaborations with other teams as well as industrial collaborations. As graphs are the cornerstone for data integration, I could also continue on my former topic and combine the two topics. As part of this process, I co-authored another book on querying graphs in the Synthesis Lectures on Data Management in 2018. Apart from classical topics, such as graph query evaluation and learning, schema discovery, graph repairs, I also actively participated in the discussions on standardization of languages for graph databases. The latest work with colleagues from both academia and industry in SIGMOD 2023 addresses the design of schema and key constraints for property graphs, as the basis of newly born standard query languages, such as GQL (2024) and SQL/PGQ (2023).

My first piece of advice to mid-career researchers is to avoid stagnation on a single topic and select at least a couple of new topics to work on. Working on a trendy topic is not necessarily the best choice. It is important to choose a topic on which you can leave an imprint and to develop what I call a “natural instinct” for what is interesting next. But this choice is also guided by your previous expertise and background, and you might need a lot of advice for choosing the right path.

**Striking the right balance between teaching and research.** You are now a mid-career researcher but you might not have reached a balance between research and teaching (there are also other daunting administrative tasks in academic life, but I am oversimplifying here). I work in the French academic system, where we have a fairly heavy teaching load and this can be detrimental for people who want to do top-notch research. My advice here is to get organized and find possible ways to moderate your teaching service. This is extremely difficult because you have to show that you are a top researcher before being eligible for endowed research chairs and similar scientific accolades. It is the chicken and egg problem! From a practical viewpoint, it means that you have to be an equally good teacher and researcher for a few years and then get senior enough to have a lower teaching load. It was my case as soon as I got hired as a full professor – the university waived half of my teaching (2011-2013). Later on, I was on an endowed research Chair at INRIA (2018-2021). Lately, I have been nominated a Senior Member at the Institut Universitaire de France in Paris (2023-2028), a high distinction that recognizes top researchers across all disciplines in France. Now, you might appreciate why the road to mid-career academic success is long, winding and full of uncertainty!

**Coping with time management problems.** One important skill that we learn along the road is time management. When I see junior professors in my entourage, I often observe that they struggle with time management. This is due to the fact that as mid-career researchers and later as senior researchers, we have to juggle many tasks including research, teaching and service. The number of tasks exponentially grows (and at the moment I am writing this article I just received a new task from the VP of my university – I know that I should not have checked my email inbox!). How to cope with time management in an effective manner? My simple piece of advice here is to be extremely organized and to know how to prioritize activities that are the best to carry out at a given point in time. Working in batches is also a viable option with some round-robin scheduling. My working day is always like a puzzle in which eventually all pieces fit together.

**Going beyond your comfort zone.** You are a mid-career researcher and now you can settle down. You have your own research team, your own PhD students and postdocs as well as your well-earned research grants. If you have developed a research topic, and you start to be regarded as one of the top researchers in your area, this does not mean that you need to stop looking at other areas, whether they are subareas of data management or other areas in CS and even beyond CS. Keep training yourself by attending
online courses and webinars and always seek new ideas on your future research topics! This is the time when the fun starts.

**Nurturing research collaborations with other teams.** Along these lines, do not be afraid to start new collaborations during your sabbatical leaves or just during short research visits or while networking at conferences. You are a researcher after all, and it is important to keep connections with people in the community who work on similar topics as yours or on rather different topics. Collaborating with others fosters interdisciplinarity and cross-fertilization. I can mention one of my experiences when I was co-organizing a Dagstuhl workshop on Big Graph Processing Systems in 2019. I convinced my co-organizers to involve people from the HPC community and this is one of the most successful Dagstuhl workshops I have ever organized. We jointly published a paper in CACM 2021 on the “Future is Big Graphs: A Community View on Graph Processing Systems”.

**Mentoring, mentoring, mentoring and being mentored.** It is important to mentor young researchers. This is a relevant part of our job and it allows us to advise young people in our community. We are lucky that mentoring activities are nowadays organized in our main conferences (e.g. in SIGMOD and VLDB as part of the DEI initiatives) but you can also do it locally in your university and in your research team. Young researchers need a lot of help when they make crucial choices in their careers (industry vs. academia, a postdoc position vs. a job). As more experienced researchers, it is important to help them navigate the system. I always keep in touch with my former PhD students and postdocs and continue to mentor them all along their careers. Mentoring activities take time, but are also extremely rewarding!

As a mid-career researcher, you might also need actionable advice from your senior peers. Striving for excellence in research is not easy, and at any time during your career you might need help from a senior colleague. If you do not have a senior person in your team, you can contact people in other teams or abroad that you trust. I was lucky to be in touch with senior people in our community. Their advice has been indispensable and key to my success.

**The importance of continuous education.** Education is an important part of our job in academia as much as research. They are the two sides of the same coin. Within education (as well as within research), we have to address new unexpected challenges, such as the role of large language models. This leads us and especially mid-career researchers to reignite their educational and pedagogical skills and to work under the umbrella of this new technology with its pros and cons.

**Working on impactful things.** As researchers, we always wish that our papers are highly cited and our work is well respected in the community. There is not a unique definition of what is a breakthrough in research, as this might lead to industrial impact, impact in other research areas and/or in multidisciplinary applications. My advice here is to always go deeper in a topic and strive for excellence. Data management is a great field as it unifies theory and practice and has several applications. I remember my first interdisciplinary grants in France with colleagues working in personalized medicine and bioinformatics. They had a lot of “data” problems to solve, and I was fascinated by them.

If you were not impactful as a junior researcher for some reason, this does not mean that you cannot be impactful in your mid-career path or even later. But it is important to not put too much pressure on yourself and to enjoy what you are doing.

**Summary.** I started with reminiscences about the first scientific event I attended as a PhD student and about the fact that at that time senior researchers seemed out of reach (at least to me). Now, times are different and with social networks and other tools people keep in touch across continents on a regular basis. There are plenty of opportunities to meet senior people and ask their advice.

Concluding, be passionate about research and teaching, tackle a new challenge (e.g. write a book on your favorite
topic), strive for excellence, get organized, reach the right balance between teaching and research as you see fit, mentor young researchers in your team or elsewhere, participate in DEI programs, keep a lifelong connection with your PhD students and postdocs and help them in their future careers.

It is never too late to start new research collaborations and to be impactful, and now that you are a mid-career researcher you have acquired a lot of more experience!
Technology-Enabled Database Education: Challenges and Opportunities

Hui Li
Xidian University
hli@xidian.edu.cn

Motivation: User-friendly systems and tools are paramount for facilitating learning of database systems. In many universities around the world, database systems courses are supplemented with the use of industrial-strength relational database management systems (RDBMS). Although there exist a few efforts on mini RDBMS for learners to practice implementing a database kernel, e.g., Minibase [6], unfortunately, there is a lack of learner-friendly technological support to facilitate the learning of various core components of a database systems course, such as database storage and indexing, relational query processing and optimization, and transaction management.

Key challenges: Specifically, our longitudinal study and observation of learners learning the topic of relational query processing reveal that they face challenges in understanding the query execution plan (QEP) of an SQL query, understanding the alternative plan choices made by the underlying query optimizer, estimating cost of a plan, etc [7, 16]. We advocate that technologies can be leveraged to improve the understanding of the above issues. For instance, since natural language (NL)-based narratives aided with visual examples (as in textbooks and lectures) have been the traditional mode of learning for decades, an intuitive NL-based description of a QEP can greatly augment learning of the execution strategies of SQL queries by an RDBMS, where recent achievements in natural language processing may suggest potential solutions.

Our efforts and experiences: To address the aforementioned challenges of learning relational query processing, we are building technological frameworks to effectively supplement the learning of various subtopics. Our recent efforts in these aspects have demonstrated positive effects in practical educational scenarios.

NEURON [3, 13] is the first system that takes advantage of a rule-based interpretation engine to generate the NL description for QEP in PostgreSQL. Specifically, NEURON first parses and transforms a QEP into an operator tree, and then traverses the tree to generate a NL description of the node based on rule-based NL templates and information encoded in the QEP. LANTERN [1, 9, 18] further makes the solution generalizable and psychology-aware, i.e., generate diversified descriptions for the same QEP to avoid repetition of output messages that can lead to annoyance and boredom [8]. MOCHA [2, 17] aids learner-friendly interaction and visualization of the impact of alternative physical operator choices on a selected QEP for a given SQL query. ARENA [4, 19] further presents a novel problem called the informative plan selection problem (TIPS) which aims to discover a set of top-k informative alternative query plans from the underlying plan space. DBInsight [15] allows learners to visualize and interact with the optimization pipeline, e.g., to turn on/off specific optimization rules, and manually adjust the estimated cardinality, etc., so that it is easy for learners to observe and understand how each individual optimization rule (resp., cardinality) can affect the choice of plans (resp., a physical operator).

All these tools are publicly-available for pedagogical use [5]. Practical experiences and academic outcomes of learners taking database systems course in both Xidian University and Nanyang Technological University demonstrate that these platforms seem to facilitate understanding of various subtopics of relational query processing as well as improve academic outcomes on average. We also observe that these tools have been accessed by 74 universities and database companies around the world. More than 1300 users have used these tools.

Opportunities: We believe that these tools pave the way for designing technology-enabled solutions to improve learning and understanding of various complex topics of relational query processing. There are plenty of novel and non-trivial research challenges still waiting to be overcome, e.g., how the cost of a physical operator/plan is estimated. Moreover, it is easy to see that technology-enabled learning can be extended to other database education topics [11], such as enabling technologies to facilitate the learning of SQL [10, 12, 14], developing user-friendly tools to understand different mechanisms for concurrency control, exploring opportunities to unveil the recovery procedure with the help of logs, and leveraging LLM in database education.
REFERENCES

[1] LANTERN software.
https://dbedu.xidian.edu.cn/lantern/.

https://dbedu.xidian.edu.cn/mocha/.

https://dbedu.xidian.edu.cn/neuron/.

https://dbedu.xidian.edu.cn/arena/.

https://dbedu.xidian.edu.cn/.


SIGMOD Rec., 50(3), 2021.


Visualizing, Exploring and Analyzing Big Data: 
A 6-Year Story

Report on the Big Data Visual Exploration and Analytics Workshops 2018-2023

Nikos Bikakis\(^1\), George Papastefanatos\(^2\), Panos K. Chrysanthis\(^3\), Olga Papemmanuil\(^4\), David Auber\(^5\), Steffen Frey\(^6\), Issei Fujishiro\(^7\), Hanna Hauptmann\(^8\), Shixia Liu\(^9\), Kwan-Liu Ma\(^10\), Tobias Schreck\(^11\), Michael Sedlmair\(^12\), Mohamed A. Sharaf\(^13\)

1. INTRODUCTION

Information Visualization has been one of the cornerstones of Data Science, turning the abundance of Big Data being produced through modern systems into actionable knowledge. Indeed, the Big Data era has realized the availability of voluminous datasets that are dynamic, multidimensional, noisy and heterogeneous in nature. Transforming a data-curious user into someone who can access and analyze that data is even more burdensome now for a great number of users with little or no support and expertise on the data processing part.

In this context, several traditional problems from Data Management & Mining, Information Visualization, and Human-computer Interaction communities, such as efficient data storage, querying, and indexing to enable visual analytics, as well as new ways and AI techniques for visual presentation of massive data, need to be revisited. For instance, techniques that provide mechanisms for information abstraction, prefetching, sampling, progressive data visualization, and summarization to address problems related to visual information overplotting. Furthermore, it is essential to develop new methods that enhance user comprehension by providing customization options tailored to various user-defined exploration scenarios and preferences.

This article presents the main findings and outcomes of the International Conference on Extending Database Technology (EDBT). The series of BigVis workshops has offered a forum for scientists and engineers from different research areas to discuss, exchange and disseminate their work, as well as to highlight challenges that bridge together these communities.

Over the past 6 years (2018–2023), more than 250 participants, 43 published papers, 147 papers’ authors, 103 PC members, 3 Journal Special Issues, and 6 Keynotes have contributed to the success of the BigVis workshops. This report summarizes six years of organizing the BigVis workshop, presenting the main findings and outcomes.

2. HIGHLIGHTS

This section provides an overview of the key highlights, including keynotes, special issues, reports, and blog posts.

Keynotes

- 2023: Daniel Keim: "The Role of Interactive Visualization in Human-Centered AI"
- 2022: Danyel Fisher: "Co-Designing the Data Structure and the User Experience"
- 2022: Steffen Frey: "Visual Mapping, Comparison and Exploration of Large Multifield Data"
- 2021: Georgia Koutrika: "The Rise of Intelligent Data Assistants: Democratizing Data Access"
- 2021: Michael Sedlmair: "Machine Learning meets Visualization"
- 2018: Bill Howe: "Viziometrics: Mining Visualizations in the Scientific Literature"

---

\(^1\)Hellenic Mediterranean University, Greece  \(^2\)ATHENA Research Center, Greece  \(^3\)University of Pittsburgh, USA  
\(^4\)Brandeis University, USA  \(^5\)University Bordeaux, France  \(^6\)University of Groningen, Netherlands  \(^7\)Keio University, Japan  
\(^8\)Utrecht University, Netherlands  \(^9\)University of Groningen, Netherlands  \(^10\)University of California-Davis, USA  
\(^11\)Graz University of Technology, Austria  \(^12\)University of Stuttgart, Germany  \(^13\)United Arab Emirates University, United Arab Emirates
Table 1: Published Papers Categorization based on CFP Topics & Years

<table>
<thead>
<tr>
<th>Topic</th>
<th>2018</th>
<th>2019</th>
<th>2020</th>
<th>2021</th>
<th>2022</th>
<th>2023</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph Data</td>
<td>[40][43]</td>
<td></td>
<td>[24][27]</td>
<td></td>
<td></td>
<td>[17]</td>
</tr>
<tr>
<td>Knowledge Graph &amp; Linked Data</td>
<td>[42]</td>
<td>[35][36]</td>
<td>[21][26]</td>
<td></td>
<td></td>
<td>[5]</td>
</tr>
<tr>
<td>Spatiotemporal</td>
<td>[41]</td>
<td>[34]</td>
<td>[20][23]</td>
<td>[13]</td>
<td>[9]</td>
<td>[1]</td>
</tr>
<tr>
<td>Data Mining, Knowledge &amp; ML</td>
<td></td>
<td>[32][36]</td>
<td>[22][28]</td>
<td></td>
<td>[8][10]</td>
<td>[3]</td>
</tr>
<tr>
<td>Personalization, Recommendations &amp; Assistance</td>
<td></td>
<td>[30]</td>
<td></td>
<td></td>
<td>[25]</td>
<td></td>
</tr>
<tr>
<td>Data representation &amp; Novel interface</td>
<td></td>
<td></td>
<td>[29]</td>
<td></td>
<td></td>
<td>[4]</td>
</tr>
<tr>
<td>Data modeling, Storage, Indexing &amp; Query processing</td>
<td></td>
<td></td>
<td></td>
<td>[16]</td>
<td>[11]</td>
<td></td>
</tr>
<tr>
<td>Domain-specific approaches</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Journal Special Issues


Community Report & Blog Posts

- **2020**: Big Data Visualization and Analytics: Future Research Challenges and Emerging Applications [19]

- **2020**: Two ACM SIGMOD Blog posts

3. PUBLISHED WORKS

The BigVis workshops have hosted several types of submissions, including novel research works, completed or in-progress work, vision, and system or demonstration papers. These works covered a wide spectrum of research topics related to the types of visualized data and the methods used to prepare data for visualization, or the techniques used to visualize, analyze, and facilitate user interaction. Table 1 summarizes the presented contributions, organizing them based on their main topic and year of presentation.

Five papers referred to the visualization of graph data, mainly focusing on the effective visualization of complex networks and big data graphs, such as summarized or 3D visualizations of graph networks. In addition, six focused on the representation of the semantics of graph data, mainly working on ways to visualize knowledge graphs and Linked Data. Another line of presented works (7 papers) referred to spatiotemporal data, i.e., methods for visualizing spatial objects, traces, trajectories, timeseries and novel applications dealing with such data. Next, the use of data mining and machine learning techniques has been explored by seven works that aim at facilitating the

---

1 Part1: https://wp.sigmod.org/?p=3037  
Part2: https://wp.sigmod.org/?p=3123

SIGMOD Record, June 2024 (Vol. 53, No. 2)
interactive exploration, and visual analysis of different tasks, such as anomaly detection, opinion analysis, or popularity bias. Other works aimed at offering a user-centric approach to visual analytics, offering ways for the personalization of visual results, visual recommendations, and user assistance in accomplishing a visual task. Yet other works proposed new methods for data representation and user interaction, combining visual and chat-based interfaces, whereas a few works dealt with topics related to the efficiency of producing visual results, focusing on issues related to data modeling, caching, indexing and query processing. Finally, eight works presented domain-specific visual applications and demonstrations from different sectors, such as agriculture, energy, media industry, cultural heritage, and sports.

4. CHALLENGES & EMERGING APPLICATIONS

Another highlight in the context of the BigVis 2020, concerned the joint report prepared by fourteen distinguished scientists, from different communities, who were invited to provide their insights regarding promising research challenges and applications related to Big Data visualization and analytics [19].

The report identified that modern systems face significant user-centric challenges. They must comprehend users’ needs to help them solve problems and offer guidance (“Show the Data not Seen by Humans”). In this context, fundamental challenges arise, such as: (a) recommending data views that the users might want to analyze; (b) identifying the data segments that will be useful for specific tasks; (c) creating data stories and explanations; (d) designing novel interfaces that assist users to understand data types and properties; (e) integrating human factors related to vision and perception into the data analysis pipeline, enabling users to supervise or provide feedback to the systems.

Another key challenge pertains to the scalability and efficiency of these systems. This is to enable visualization systems to efficiently handle billion objects datasets, while ensuring that the response time is limited to just a few milliseconds. Addressing this challenge involves developing tools capable of performing interactive operations and complex analytics over massive sets of data. In that respect, there is the need to develop novel approaches (e.g., progressive data processing & indexing) that can handle large streaming, sampled, uncertain, high-dimensional, and noisy data.

Next, within the context of visual analysis, addressing data management problems reveals “new” challenges for data-intensive visual applications, such as visualization-centric algebras, design of visualization operators, optimization techniques, and effective storage and indexing scheme for visual analysis.

Finally, building interactive tools and enabling visual analysis for Machine Learning applications presents a significant challenge. For example, developing visual methods for interpreting and techniques for interacting with ML models; implementing visualization systems that facilitate model troubleshooting, debugging, and comparison.

5. REFERENCES

[14] T. Cross, R. Bovo, T. Heinis: Browsing Internet Content in Multiple Dimensions. BigVis 2021
[23] P. Quirós, B. Rosario C. Álvarez: The new Spanish political scenario: Twitter graph and opinion analysis with an interactive visualisation. BigVis 2020
[26] F. Desimoni, L. Po: Providing Effective Visualizations over Big Linked Data. BigVis 2020
[34] A. Gogolou, T. Tsandilas, T. Palpanas, A. Bezerianos: Progressive Similarity Search on Time Series Data. BigVis 2019
[35] I. Manolescu, M. Mazuran: Speeding up RDF Aggregation Discovery through Sampling. BigVis 2019
[40] D. Bleco, Y. Kotidis: Finding the Needle in a Haystack: Entropy Guided Exploration of Very Large Graph Cubes. BigVis 2018
[43] T. Repke, R. Krestel: Topic-aware Network Visualisation to Explore Large Email Corpora. BigVis 2018
Report on the Fifth International Workshop on Health Data Management in the Era of AI (HeDAI 2023)

Haridimos Kondylakis
Univ. of Crete & FORTH-ICS
Heraklion, Greece
kondylak@ics.forth.gr

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

Praveen Rao
Univ. of Missouri-Columbia
Columbia, USA
praveen.rao@missouri.edu

ABSTRACT

Artificial intelligence (AI) technologies have the potential to drastically improve healthcare delivery globally. However, healthcare data are diverse, complex, and very large in size. Better information management techniques are needed to deal with the volume and heterogeneity of healthcare data. Better AI techniques are needed to fully harness healthcare datasets to improve healthcare delivery and quality of care for patients. Our workshop aimed to bring together diverse group of researchers and practitioners interested in developing next-generation solutions for challenging problems in healthcare systems. A wide range of topics ranging from federated approaches for data sharing to machine learning for predictive analytics to medical question answering were discussed during the workshop.

1. INTRODUCTION

As AI technologies are increasingly employed for the management of health data, new challenges occur daily that dictate new solutions. For example, bringing data into centralized repositories has become more difficult, mainly due to regulatory, competing interests, and trust issues. As such federated approaches are emerging allowing models to be sent to the data to be trained instead of accumulating all data central and then training the models on top. This is not only a result of technological advancements in privacy-preserving machine learning but also of the increased requirements to preserve the confidentiality and privacy of individuals. Further various types of data are being used for predictive analytics such as electronic health records, electroencephalography portable devices, and ultrawideband radar sensors which dictate the effective management of the collected data and their usage for predictive downstream tasks. However, besides predictive tasks, other tasks such as personalized recommendations and intelligent question answering are equally important.

The goal of HeDAI 2023, co-organized with the EDBT conference in Ioannina, Greece, was to bring together researchers with interests cross-cutting the fields of Semantic Web, AI, data science, data management, and health informatics to discuss the challenges in healthcare data management and to propose novel and practical solutions for the next generation of data-driven healthcare systems. Developing optimal frameworks for integrating, curating, and sharing large volumes of clinical data has the potential for a tremendous impact on healthcare, enabling better outcomes at a lower and affordable cost. The ultimate goal is to enable innovations in Semantic Web, knowledge management, and data management for healthcare systems to move the needle to achieve the vision of precision medicine.

2. KEY TOPICS

In this section, we present the various topics that the workshop focused on through the various presentations and invited talks.

2.1 Real-World Federated Approaches for Healthcare Data Sharing and Analysis

The first keynote with title “Overcoming open issues and unmet needs in healthcare through the sharing, harmonization and federated analysis of unstructured medical data” was presented by Dimitrios I. Fotiadis, Prof. of Biomedical Engineering in the Department of Materials Science and Engineering, University of Ioannina, Greece that focused on overcoming open issues and unmet needs in healthcare through the sharing, harmonization, and federated analysis of unstructured medical data. The underlying heterogeneity and the reduced quality of the existing medical data across different clinical centers obscures the interlinking and co-analysis of such data. In addition, the legal and ethical barriers obscure the sharing of sensitive data and
highlight the need for the development of federated learning strategies to enable the federated analysis of medical data across different countries with inherent data protection policies. In this direction, a framework was presented allowing to overcome open issues and unmet needs through the design and development of (i) automated methods for data curation to address data inconsistencies and improve data quality in terms of relevance, conformity, and completeness, and (ii) hybrid data harmonization pipelines (i.e., pipelines for unifying disparate data fields, formats, dimensions, and columns into a composite dataset). Those are based on a combination of lexical and semantic matching methods with word embeddings which are utilized on top of medical index repositories and external knowledge bases to transform the heterogeneous data into a common standardized format, (iii) data augmentation through the design of robust virtual population generators to enhance the statistical power of databases with insufficient population size and improve the performance of the existing AI models, and (iv) federated AI algorithms to enable the training and evaluation of trustworthy and explainable AI workflows across high-quality and harmonized data stored in federated databases within a cloud environment. Multiple case studies were presented, and conducted across different clinical domains, including primary Sjögren’s Syndrome (pSS) and hypertrophic cardiomyopathy (HCM), among others, to demonstrate the efficacy of the proposed framework to address clinical unmet needs.

2.2 Supporting Medical Trials with a Data Lake Federation

The second keynote with title “Supporting Medical Trials with a Data Lake Federation: a Research Perspective” was presented by Letizia Tanca, a full professor in data management at Politecnico di Milano, Italy. The talk focused on how to adopt a data lake federation in order to obtain significant results and benefits for medical organizations. Letizia Tanca started with the fact that the collection of data needed for clinical trials is very critical since a complete picture of the patient’s status can only be obtained from real-world data, collected in different clinical institutions during their research and clinical history. A suitable solution to store and process the huge amount of necessary information, often coming from very heterogeneous devices and data sources, is needed to create the above-mentioned tremendous value. Data lake technology appears to be a promising solution for achieving the ability to manage and analyze data in healthcare. The goal is to be able to manage the complexity of the volume and variety of big data by providing data analysts with a self-service environment to which advanced techniques can be applied. Towards this direction, the adoption of a data lake federation, through which the involved medical organizations obtain significant results and new benefits is introduced. The extremely heterogeneous data collected in the data lakes of the federation must be accurately described, in order to document its quality, facilitate its discovery and integration, and define ethical, security, and privacy policies. Based on the experience in the Health Big Data project [5], the proposed architecture to collect and use data in the federation identifies the main IT research challenges we are facing nowadays.

2.3 Continuous Machine Learning for COVID-19

The COVID-19 pandemic caused a flurry of research in employing machine learning techniques for COVID-19 detection, mitigation, and treatment [20, 8, 16]. Avci et al. [4] in their paper entitled “Is My Model Up-to-date? Detecting COVID-19 Variants by Machine Learning” proposed a continuous machine learning approach to detect COVID-19 variants. Their work is motivated by the need to update machine learning models due to concept drift [22]. They proposed an online deep-learning approach that used two neural networks. The first network called the primary neural network is used for output prediction. The second network called the secondary neural network is used for updating the model in the background. A performance estimator continuously checks for performance degradation. When degradation is detected, the primary neural network is replaced by the secondary neural network. The detection of concept drift is done by evaluating the area under the curve (AUC) metrics. The approach was evaluated using a real dataset containing patients that were tested positive or negative for COVID-19 using an RT-PCR test. It achieved better accuracy than an offline technique by 6% and an online binary classification technique by 5%.

2.4 Predictive Analytics Using Machine/Deep Learning

As AI models are successfully applied in many medical areas, exploring how to integrate AI models with medical domain taxonomies seems natural. In the paper entitled “Diagnosis Prediction over Patient Data using Hierarchical Medical Taxonomies” [9], the authors investigate how hierarchical medical taxonomies can be used to improve AI-based diagnostic tasks. In this direction, patient
graphs are extracted from electronic health records, and then the node embeddings of that patient graph are pre-initialized using information from medical taxonomies such as ICD-9 [2], ATC [1], and LOINC [3]. This step improves the performance of graph convolution network models over the enriched patient graph on diagnostic downstream tasks. Experiments performed using the MIMIC-IV dataset indeed confirm the usefulness of the proposed approach.

In another approach, Massa et al. [14] with their paper “Monitoring Human Attention with a Portable EEG Sensor and Supervised Machine Learning” employ cheap and portable electroencephalography (EEG) sensors for monitoring the human attention level. The acquired signals are processed using machine learning to estimate the attention level of people, enabling the continuous and unobtrusive monitoring of people, especially useful in the fields of rehabilitation and psychology. The authors propose a feature extraction technique based on sliding windows, and supervised machine learning distinguishing between attentive and distracted states, whereas their results show that it is highly beneficial in terms of accuracy to train the model first on the specific subject under investigation.

Another similar study is focusing on the estimation of respiration and heartbeat rates using impulse response ultra-wideband (IR-UWB) radar sensors with title “Respiration and heartbeat rates estimation using IR-UWB non-contact radar sensor recordings: A pre-clinical study” [15]. The authors first propose an architecture composed of simulated chest and heart monitors and they perform extensive recordings of the simulator’s displacements, trying to identify the optimal mathematical estimation of the respiratory and heartbeat rates, compared to the initial frequencies given to the simulated procedure. The experimental results show that the proposed architecture is able to estimate sufficient quality rates in an accurate manner and even if these pre-clinical tests are made under ideal conditions, the UWB-radar can further be used for clinical assessment, with promising perspectives.

2.5 Privacy-Preserving Machine Learning

Federated learning has gained a lot of attention in healthcare [17, 23, 12] as it enables collaborative learning of machine learning models without sharing raw data between a client and a server. This protects the privacy of patient data, a critical requirement in healthcare data sharing. Split learning [7] improves upon federated learning wherein the model architecture and weights are not shared between a client and a server. However, split learning is prone to privacy leakage due to the reconstruction of the activation maps in deep learning models. Khan et al. [11] in their paper entitled “Split Ways: Privacy-Preserving Training of Encrypted Data Using Split Learning proposed to apply homomorphic encryption on the activation maps during split learning to protect user privacy. They focused on a U-shaped split learning model where the server only trains the fully connected layers of a convolutional neural network. The other layers are trained on a client. Using encrypted activation maps, they achieved comparable accuracy to training on plain text activation maps on an abnormal heart rhythm dataset. However, the training cost and communication overhead increased significantly. Further research is necessary to lower the overhead of employing homomorphic encryption in split learning.

2.6 Recommendation Systems

Recently, several approaches to recommendation systems exploit health-related information to provide suggestions, e.g., [19, 18]. Following a similar path, the paper entitled “SHARE: A Framework for Personalized and Healthy Recipe Recommendations” [24] proposes a personalized recommendation system that provides suggestions about recipes to users based on their health history and the preferences of similar users. Overall, the SHARE framework combines user tastes and nutritional information about the recipes in order to provide recommendations for recipes that meet the user’s preferences and specific health needs. An alternative that offers personalized filtering for the users of the system is also presented. An experiment with real uses was performed and shows the system’s ability to provide highly relevant personalized recommendations, using a large real-world data set of recipes.

2.7 Medical Question Answering Systems

Medical question answering systems [13, 6, 10] are growing in popularity due to the success of natural language processing (NLP) techniques and knowledge graphs. Most of the existing literature focuses on the English language. Tsampos et al. [21] with the paper entitled “A Medical Question Answering System with NLP and graph database” proposed a medical question answering system for the Greek language using NLP techniques and a graph database. Medical data in Greek are processed using sentence segmentation, tokenization, parts-of-speech tagging, and dependency parsing. The parse trees of sentences are represented as a graph and stored in an open-source graph database. In the
graph, the tokens are represented as nodes, each
dependency is represented as a relation, and mor-
phological features are represented as node prop-
erties. Questions are translated into graph queries
and executed by the graph database. However, due
to a lack of detailed evaluation, it was hard to judge
the effectiveness of the proposed system.

3. CONCLUSIONS

A number of key observations and research direc-
tions emerged in the discussions during the work-
shop. Particular attention was given to large lan-
guage models (LLMs) which are AI tools typically
used to process and generate text. LLMs can im-
prove the user-friendliness of a system since they
can offer answers to questions in natural language,
or they can summarize or even translate text on a
level that is understandable from human capabil-
ties. This way, users, or patients in our case, can ac-
tively interact with systems using LLMs in a trans-
parent way, facilitating access to healthcare. Still, a
systematic and comprehensive overview of the po-
tentials and limitations of LLMs in the health do-
main is missing. Overall, HeDAI 2023 was success-
ful in attracting international researchers to share
their findings in health data management in the era
of AI.

4. REFERENCES

https://www.whocc.no/atc_ddd_index/.

[2] International classification of diseases,ninth
revision, clinical modification (icd-9-cm).

[3] The international standard for identifying
health measurements, observations, and

Up-to-date? Detecting CoViD-19 Variants by
Machine Learning. In G. Fletcher and
V. Kantere, editors, Proceedings of the
Workshops of the EDBT/ICDT 2023 Joint
Conference, Ioannina, Greece, March, 28,
2023, volume 3379 of CEUR Workshop

R. S. Tavares. Big data for healthcare: A

A medical question-answering system
combining nlp techniques and semantic web
technologies. Information Processing

of deep neural network over multiple agents.
Journal of Network and Computer

Goldgof. Finding COVID-19 from chest
X-rays using deep learning on a small dataset.

prediction over patient data using hierarchical
medical taxonomies. In G. Fletcher and
V. Kantere, editors, Proceedings of the
Workshops of the EDBT/ICDT 2023 Joint
Conference, Ioannina, Greece, March, 28,
2023, volume 3379 of CEUR Workshop

medical question answering system based on
knowledge graph. IEEE Access,

ways: Privacy-preserving training of
encrypted data using split learning. CoRR,

evaluation of federated learning techniques for
secure and privacy-preserving machine
learning on medical datasets. In 2022 IEEE
Applied Imagery Pattern Recognition

V. Shanker, J. Ely, and H. Yu. Beyond
information retrieval—medical question
answering. In AMIA annual symposium
proceedings, volume 2006, page 469. American
Medical Informatics Association, 2006.

Monitoring human attention with a portable
EEG sensor and supervised machine learning.
In G. Fletcher and V. Kantere, editors,
Proceedings of the Workshops of the
EDBT/ICDT 2023 Joint Conference,
Ioannina, Greece, March, 28, 2023, volume
3379 of CEUR Workshop Proceedings.

G. Rigas, and M. Tsiknakis. Respiration and
heartbeat rates estimation using IR-UWB
non-contact radar sensor recordings: A
pre-clinical study. In G. Fletcher and
V. Kantere, editors, Proceedings of the
Workshops of the EDBT/ICDT 2023 Joint
Conference, Ioannina, Greece, March, 28,
2023, volume 3379 of CEUR Workshop


Diversity, Equity and Inclusion Activities in Database Conferences: A 2023 Report

https://dbdni.github.io


1. THE DEI@DB INITIATIVE


Our responsibility as a community is to ensure that attendees of DB conferences feel included, irrespective of their scientific perspective and personal background. In its leadership role, the DEI team advises DEI chairs at DB conferences, serving as a memory of DEI events at conferences, building an agreed-upon vision, and committing to working together for achieving DEI. That is pursued via actions led by our core members (Figure 1) and liaisons of individual executive bodies (Figure 2).

What did we achieve this year? The response from the DB community in various events in 2023 has been extremely positive. DBCARES¹ is now officially part of the DEI initiative. We consolidated our Code of Ethics and created guidelines for DB conference organizers. Some of our venues continued to use CLOSET [5] towards fairer reviewer assignments. Several of our actions have become mainstream and will no longer be managed by the DEI initiative.

To ensure early financial planning of DEI activities at conferences, SUPPORT has become the responsibility of conference organizers. INCLUDE, INFORM, ORGANIZE, and REACH OUT are now under the direct responsibility of DEI chairs at each conference. COORDINATE, liaisons, and the DEI chairs), SCOUT (collating DEI efforts from other communities), ETHICS (establishing and promoting ethics guidelines for paper writing and paper reviewing) and MEDIA (preserving and disseminating the digital media produced by DEI@DB events [7]) will continue to be managed by the DEI initiative. We are introducing a new action DIVERSIFY.

The initiative will continue in January 2024 with a new set of members.

2023 DEI statistics. An important goal of DEI@DB is to understand our community so that we can identify issues that need to be addressed as well as assess the effectiveness of our activities. To this end, we ran surveys at EDBT/ICDT, ADBIS and SIGMOD with 61, 109, 984 participants, respectively. Unfortunately, SIGMOD used a custom survey, making it harder to present aggregate results across the three conferences. Hence, we present aggregate results for EDBT/ICDT and ADBIS and discuss SIGMOD separately. About 1/3 of the respondents were females, and 3.1% were LGBTQ+, aligning with the findings in the previous years. The majority of the participants are either from academia (40.7%) or work in both academia and industry (34.9%), followed by students (18.2%), and industry-only participants (3%). Comparing the statistics in 2022 (mostly hybrid conferences) and 2023 (mostly in-person conferences), we observe similar low student and industry participation, which is concerning. This calls for actions to increase participation of these groups, in particular the number of students with or without a paper.

¹https://sigmod.org/sigmod-policies/

dbcares-policy
According to these surveys, people are most interested in research (finding topics, setting goals, determining success) followed by work-life balance and mentorship as the top-3 topics they want to discuss at DEI sessions. Regarding SIGMOD, 13.8% of the respondents mentioned that their research addresses DEI. They also suggested improvements such as maintaining double-blind reviews, diversifying program committees, funding students and parents with infants, reducing conference registration fees, providing more mentorship opportunities for students and pursuing visa reforms to facilitate conference attendance.

2. DEI@DB CONFERENCES 2023

Overall conclusion: DEI@DB started as a joint ACM SIGMOD/pVLDB initiative. In 2020, we invited other conferences to join the effort and now, we have participation from 11 different conferences. We collaborate on developing activities, interventions, survey design, as well as in assessing the effectiveness of different activities. This report provides an overview of the activities carried out at the conferences. They provide evidence of both the interest and concrete action towards ensuring diversity and inclusiveness in our community. We plan to continue monitoring and sharing these activities.

DEI@CIDR. U. Sirin (Harvard U.), S. Idreos (Harvard U.), and Y. Tian (Gray Systems Lab at Microsoft) chaired DEI@CIDR. The co-chairs offered DEI guidance for paper writing, presentations and peer reviewing. The chairs organized a successful mentoring event, wherein they paired junior and senior attendees for 30-minute sessions. Each pair had the flexibility to determine the timing and format of their meetings. The co-chairs successfully recruited 17 students and 13 mentors, some of whom mentored more than one session.

DEI@EDBT/ICDT. P. K. Chrysantis (U. of Pittsburgh), L. Peterfreund (CNRS/The Hebrew U. of Jerusalem) and G. Vargas-Solar (CNRS) chaired DEI. The program included a panel on “DEI Perspectives Around the World”. Panelists involved in analogous educational endeavours came from Brazil, Mexico, USA, and Italy. The discussion compared course material. Furthermore, Marie Plamondon from Open North, UK gave a keynote on “An Intersectional Approach to Data Governance.”

DEI events were organized in a hybrid format. Thanks to the support of the EDBT endowment and the organizing committee, 17 online and 7 on-site registrations were free. The DEI keynote and a 150-euro PhD workshop award were also sponsored.

DEI@ICDE. A. Meliou (UMass) chaired DEI@ICDE. The program focused on three efforts: (1) Affinity stickers were included in registration packages; one could attach stickers that denoted particular interests to their badge to serve as conversation starters. (2) A mentoring lunch round-table offered junior researchers the opportunity to interact with seniors. (3) A panel discussion on “Ethics in paper submissions and reviewing” with L. Chen (moderator), A. Bonifati, Y. Ioannidis, J. R. Haritsa and S. Roy. The panel addressed the following topics: Reviewing quality: How to incentivise reviewers to write more conscientious reviews? Should we penalize bad reviewers (e.g., bans from PC lists)? What organizational efforts can positively impact review quality? Are implicit biases affecting our reviewing processes, and how do we overcome them? COIs: Policies vary across venues. How do we settle as a community towards a consistent COI policy? What are the right limits and restrictions? Ethics in COI declaration: How do we address under-declared COIs (e.g., penalties, desk-rejection, or alternative mechanisms)? Submission quality: To increase publication counts, we see different submissions using the same approach to address different problems or an approach divided across multiple submissions. How should we address such challenges to the novelty bar? Preempting future ethics challenges: Will AI-assisted technologies like ChatGPT become factors in paper writing and reviewing? What should we monitor?

The panel triggered an ongoing Task Force on Review Processes, chaired by A. Meliou and S. S. Bhownick, in which the chairs of database conferences are involved (EDBT, ICDT, VLDB, ICDE, ACM SIGMOD/PODS).

DEI@MDM. R. Borovica-Gajic (U. of Melbourne) and V. Kalogeraki (Athens U. of Economics and Business) chaired DEI intending to understand the meaning of DEI. The first action was to utilize the rich set of DEI material generated in past MDM, EDBT/ICDT and SIGMOD to prepare guidelines and examples of inclusive and discriminatory behavior and language, encouraging participants in papers and presentations. A on “Navigating the Social and Ethical Responsibilities of Computing” was delivered by G. Dobbie (U. of Auckland) who shed light on the multifaceted dimensions of the social and ethical responsibilities of computing and emphasized the urgent need for conscientious action. Her talk delved into the ethical considerations that emerge with the rapid progression of computing technologies, underlining the significance of fostering transparency, accountability, and inclusivity in the design and deployment of computing systems.
DEI@SIGMOD. The DEI chairs Y. Amsterdamer (Bar-Ilan U.) and C. Curino (Microsoft) organized: (1) A special lunch funded by Amazon and Microsoft, where a small group of ~40 participants were invited, putting a special emphasis on students, newcomers along with long-time conference participants and organizers. The participants were randomly seated at small tables in a restaurant where they could have a quiet conversation.

The second event was a keynote by S. Sadiq (U. of Queensland) who described challenges arising from diverse student populations and from changes in learning delivery modes. She also shared strategies for incorporating DEI in Information Technology education.

The third event was a birds-of-a-feather and introduced the main activities undertaken by the DEI initiative. Participants were invited to propose and discuss selected topics in small groups. The ideas and insights were summarized in a concluding forum. The event was very well received by 30+ people. The discussion was broad and lively, and heated at times, demonstrating a good level of passion for DEI topics. This type of event is key to making our own DEI efforts more inclusive and more representative of a broader audience and to increasing our community commitment to DEI.

SIGMOD added to the review form a yes/no question on whether a submission under review is compliant with the DEI guidelines on the SIGMOD Website, and an open question for reviewers to enter their comments.

DEI@VLDB. This year’s DEI efforts at VLDB included a mentoring session and a keynote. The mentoring session gave mentees (students and researchers of any stage in their career) a chance to gain first-hand guidance and inspiration on grand challenges and future research opportunities from experienced researchers. The event was organized in thematic tables based on broad areas of research within the VLDB community: Data Mining, Machine Learning and AI (mentor: C. Mohan); Data Security and Privacy, Blockchain Data Management (mentor: E. Bertino); Novel Infrastructures and Techniques for Big Data and Large Language Models (mentor: A. Halevy); Information Integration, Retrieval and Visualization (mentor: R. Miller); Data Integrity, Quality and Provenance (mentor: W. Tan); Diversity, Equity and Inclusion Issues (mentor: G. Koutrika).

The DEI keynote “On the Cusp: Computing Thrills and Perils of Professional Awakening” by N. Milic-Frayling (Intact Digital Ltd. and QCRI) discussed the perils of generative AI. She pointed out that this should be a turning point in our professional awakening.

DEI@DASFAA. Gender and geospatial diversity was extensively discussed among the steering committee members and the organizing committee members of DASFAA 2023. Statistics were gathered on female PC members and female organizing committee members. Suggestions and a list of female researchers as potential organizing committee members were provided. S. Amer-Yahia and A. Bonifati delivered keynotes.

DEI@MEDES. MEDES joined DEI this year. S. Sellami (Aix Marseille U.) and P. Fatourou (U. of Crete and FORTH) organized a DEI plenary session. E. Ntoutsi (Bundeswehr U.) delivered a talk on How to make AI more fair and unbiased. She discussed the discriminative impact of AI-driven decision-making and presented research on discrimination for multiple protected attributes, discrimination under class imbalance and multitask discrimination. P. Fatourou closed the session with a talk in which she invited participants to integrate related DEI aspects in their research and careers.

To facilitate discussions among PhD students and peers about navigating their careers, a panel on “Plurality of Excellent Career Profiles, Wellbeing, and Meritocracy” was organized with panelists from academia and industry. The discussion centered on “Balancing productivity and meaningful research: How can one achieve it all without overextending oneself?”. 
3. UNDECLARED COIS

Automated check for COIs is now standard practice in SIGMOD, EDBT and VLDB who requested in 2023 to leverage CLOSET [5] for managing COIs. VLDB 2023 experimented with a pre-generated list of coauthorship-based COIs that is periodically made available to all authors to aid COI declaration in CMT. Undeclared COIs still introduced further delays to the review process.

ICDE was an early adopters of CLOSET in 2020. While CLOSET was used to check for COIs for each submission cycle in 2022 and post-facto for all cycles in 2021, it was only used post-facto for only the third cycle in 2023. CLOSET detected a significant number of undeclared COIs for all cycles it was deployed in.

4. DIVERSITY OF PC

Since a diverse group of individuals tends to surface different perspectives, the PC of a venue should have good diversity w.r.t. various dimensions such as topic, location, institution, ethnicity, and gender. While our venues typically ensure that subject areas of interest and institutional diversity are adequately represented in a PC, diversity w.r.t. other dimensions may not receive the same degree of attention during PC formation. We report our observations w.r.t. location and ethnicity diversity. By location, we mean the country where a PC member is located at the time of serving for a venue. Since ethnicity may not be publicly available, we use country of tertiary/secondary education (COE) as a proxy for ethnicity. Specifically, it represents the country where a PC member undertook his or her high school or undergraduate education. Such information is often available in a PC member’s homepage or LinkedIn.

Figure 4 reports some stats. The largest COE groups for SIGMOD 2023, VLDB 2023, and ICDE 2023 represent around 29%, 43%, and 55% of the PC, respectively. We can observe two key trends here. First, in contrast to SIGMOD, the COE diversity is getting more skewed for VLDB and ICDE. Second, the location diversity trend is similar and relatively stable across all venues.

5. GOING FORWARD

Job descriptions. We are working with the ACM to ensure the job descriptions of DEI members and chairs are aligned with the ACM policies.

COIs. Although major venues use CLOSET, several others still depend on conference management tools. We are working on a consistent COI policy across all venues.

Code of Ethics and additional committees. From now on, DEI chairs will ensure the CoE is clearly displayed on conference websites and on site. We look forward to introducing new committees [1, 2]— such as the Accessibility Committee, Family Committee, and Sustainability Committee. We will also promote conference websites to use Accessibility Menus as in [3].

MEDIA action. We have established a DEI-DB-MEDIA YouTube channel2 where we organized recordings of DEI events from our conferences into talks, briefs, panel discussions and workshops by conference and by year.

ETHICS action. We are working on establishing and promoting ethics guidelines for publications, similar to other efforts [6]. This involves creating a living document specifying major ethics aspects that authors and reviewers should consider. To enhance inclusion, we plan to compile a set of guidelines for session chairs, presenters, and participants for handling panels and Q&A.

DIVERSIFY action. DEI extends beyond merely tallying the number of male and female participants. The DIVERSIFY action will ensure regular data collection from: Responses to demographic questions required by authors, reviewers, and attendees before (via registration forms) and after conferences. This allows to understand community diversity. Observations from the staff or volunteers, who can document cases of diversity during the conference (e.g. assessing accessibility and noting incidents). DEI members can provide guidelines. Feedback mechanisms: online feedback can be provided to report diversity-related issues. Social Media monitoring can be used for discussions and feedback related to diversity. PhD workshops should incorporate activities that raise awareness about DEI in both the formulation of research projects and dissemination of findings. The assessment of presentations for paper awards can help estimate how PhD students apply DEI principles and determine which areas require discussion.

Finally, additional actions such as sustainability will be discussed in the future.

2https://tinyurl.com/2v4ed98n
6. REFERENCES


Data-driven PC-chair-in-the-loop Formation of Program Committees: An EDBT 2023 Experience

Sourav S Bhowmick
Nanyang Technological University
Singapore
assourav@ntu.edu.sg

Katja Hose
TU Wien
Austria
katja.hose@tuwien.ac.at

ABSTRACT
The formation of a quality program committee (PC) for a conference venue is critical for a high-quality scientific program. Traditionally, PC chairs take a “manual” approach to form a PC. In practice, however, such an approach, might not create a diverse PC w.r.t. certain dimensions. Furthermore, it has been reported that the traditional manual approach may lead to dense co-authorship networks among PC members. All these aspects can easily make it challenging in practice to ensure fair and quality assignments of reviewers to submissions. In this article, we share our experiences and results of installing a novel data-driven PC-chair-in-the-loop PC formation framework for EDBT 2023 to mitigate some of the challenges brought by traditional PC formation methods.

1. INTRODUCTION
Program committee (PC) members of a conference venue play a pivotal role in ensuring quality and fairness of the review process. A high-quality review process does not only facilitate a high-quality scientific program but also enhances trust of a venue among community members by providing constructive feedback on scientific work that benefits authors in their scientific endeavours. PC chairs often strive to form PCs to serve this intended purpose. With the increasing number of submissions in recent years, however, sizes of PCs in major conferences can easily be in the hundreds, making quality PC formation a challenging task.

The traditional approach of PC formation for a venue is “manual” in nature. PC chairs typically invite candidate reviewers independent of other candidates primarily based on the recommendations from meta-reviewers and themselves, and lists of PC members in recent venues (e.g., PC members in the last two editions of SIGMOD). Although they may invest efforts to ensure high coverage of all topics of interest (based on publication profiles of candidates) as well as diversify PC members along various dimensions (e.g., gender, location, experience), these are largely ad-hoc and manual in nature. Consequently, even if the initial list of candidate reviewers is diverse with adequate coverage, the final set of PC members may not necessarily be so due to declination of invitations from many candidates during PC formation. Furthermore, a recent report [3] revealed that such a traditional approach of PC formation often leads to dense co-authorship networks between reviewers that may make fair assignments of reviewers to submissions challenging.

In this article, we report our experiences as PC chairs in installing a novel data-driven PC chair-in-the-loop PC formation framework for EDBT 2023 [1] to address some of the challenges of the traditional approach. Specifically, we undertook a data-driven, iterative approach to carefully select a set of PC members that is diverse w.r.t. multiple dimensions, adequately covers the topics of a venue, but forms a sparse co-authorship network with a low average clustering coefficient, a smaller giant component (i.e., largest connected component), and low average and maximum degree. We observe that the formed PC not only exhibits features that are closer to the PC selection criteria of EDBT 2023 but also several quantitative measures related to the review process quality show promising results. Nevertheless, we do not claim that the our framework has a causal relationship with the review process quality. This article aims to nudge future PC chairs to adopt and explore the nexus between the proposed framework and quality review process.

The rest of this article is organized as follows. In Section 2, we introduce the criteria we have adopted for selecting PC members for EDBT 2023. We describe the novel framework for PC formation based on these criteria in Section 3. Section 4 reports the impact of the formed PC on the review process. The last section concludes the article. Note that the majority of the content reported here was presented during the EDBT 2023 Opening Session.
2. CRITERIA FOR PC MEMBERS

There are many ways PC chairs may form a PC. We adopted the following criteria to select high-quality and diverse PC members for EDBT 2023:

1. The PC members should adequately cover all topics of interest of EDBT 2023. The coverage of topics does not have to be uniformly distributed. Popular topics typically attract a higher number of submissions and hence a larger pool of experts needs to be recruited on these topics compared to less popular ones to balance reviewer workload and assign reviewers with the respective expertise.

2. All PC members should have prior experience in publishing their research in SIGMOD, VLDB, or EDBT. We use the past publication record of an individual in these venues as a proxy for ensuring high scientific quality of the PC.

3. The PC should have a good balance of “senior” and “junior” researchers. By “senior”, we refer to researchers who are tenured faculty members or have been active in research for more than 8 years since their doctoral degree. “Junior” researchers are typically tenure-track faculty, post docs, or researchers with 8 or less years of research experience. This balance is essential as it enables us to inject sufficient experience in the review process while at the same time provide junior researchers with an opportunity to serve as reviewers.

4. Since a diverse group of individuals tends to surface different perspectives, the PC should have good geographic, institutional, and country of origin diversity. That is, no particular group should be overly represented in the PC.

5. It is desirable for the PC to form a sparse co-authorship network instead of a dense one. Not only do dense co-authorship networks among PC members make unbiased assignment of reviewers to submissions challenging, but they might also increase the likelihood of unethical reviewing practices such as collusion [3].

6. PC members should not be serving as PC chairs of other major venues during the review period of EDBT 2023. This is to mitigate the adverse impact of competing service workload on the review process.

Observe that Criteria 1, 3, and 6 are usually considered during PC formation in major venues. However, Criteria 2 may not be adopted strictly. For example, the VLDB 2023 (resp. SIGMOD 2023) PC contained at least 9 (resp. 11) members who have not published in these two venues at the start of the review process. Although some degree of geographic and institutional diversity is considered by existing venues, country of origin (Criteria 4) is often ignored, leading to over-representation of certain groups (detailed in Section 3.3). Lastly, Criterion 5 is typically not considered by major data management venues.

3. FRAMEWORK FOR PC FORMATION

In this section, we describe our data-driven PC-chair-in-the-loop framework for PC formation.

3.1 Approach

First, we gathered recommendations from the 13 Senior Program Committee (SPC) members of EDBT 2023. This list contained 87 recommendations. Instead of inviting all of them for PC directly, we pruned the list based on Criteria 2 and 6 first. This resulted in a final list of 75 candidate PC members. Note that the check for Criterion 2 is undertaken automatically by leveraging CLOS-ET [2, 4]. Since the number of candidates is lower than the desired size of the PC (i.e., 85-90), we iteratively added additional candidates satisfying the two criteria as discussed below. These candidates are retrieved from the reviewer database of CLOS-ET, which contains details of PC members in major venues in the last five years. We also exploit CLOS-
ET's capability of searching for candidates in DBLP based on topics and publication profile.

Each candidate is associated with the following attributes: name, email, institution, country of residence, country of tertiary/secondary education (COE), seniority, expertise area, DBLP, iteration, and decision. The attribute COE represents the country where a candidate undertook his or her high school or undergraduate education. Such information is often available in a candidate’s homepage or LinkedIn page. We use it as a proxy for country of origin since the latter may not be publicly available. Seniority takes one of the following two values: senior or junior (Criterion 3). The DBLP attribute records the DBLP URL of a candidate. The iteration attribute records the iteration id when a candidate is invited (detailed below) and decision captures whether a candidate accepted or declined the PC invitation or did not respond.

Next, we formed the PC iteratively until the desired number of PC members was attained by adopting the following steps (Figure 1). We set the desired PC size to be between 85-90 given the expected submission numbers to EDBT. In each iteration, we undertook the following steps:

1. Compute research area, level, diversity distributions, and network features (Criteria 1, 3–5) of the list of candidates and accepted PC members by leveraging CLOSET. We added or removed candidates to ensure that the aforementioned criteria are satisfied.
2. Send PC invitation to the refined candidate set from Step 1 and give them 10-14 days to respond. We record their decisions and update the decision attribute of the candidate list.
3. Add new candidates satisfying Criteria 2 and Figure 3: Distribution of expertise of reviewers and submissions. The Y-axis shows the percentage of reviewers in the PC (resp. submissions in Cycle 3) for a specific topic.

Figure 3: Distribution of expertise of reviewers and submissions. The Y-axis shows the percentage of reviewers in the PC (resp. submissions in Cycle 3) for a specific topic.

We took 8 iterations spanning over a period of 3.5 months to recruit 88 PC members. Figure 2 depicts the number of candidates invited per iteration as well as acceptance rates of the candidates. In total, we invited 194 candidates with an overall acceptance rate of 45.4%.

3.2 Looking into the Data

In every iteration of the PC formation process, we generated various distributions and statistics (related to Criteria 1, 3–5) of the currently accepted list of PC members, invited candidates who have not yet responded, and candidates whom we intend to send invitations to. We updated the candidate list for invitations, if necessary, based on these distributions and statistics so that the aforementioned criteria are satisfactory in each iteration. This enabled us to form the PC iteratively in a data-driven manner. In this subsection, we first present data to reveal this iterative process. In the next subsection, we shall compare some of these quantitative characteristics of our PC with the PCs of recent major data management venues that are formed using the traditional approach.

Our foremost criterion was to ensure that the PC adequately covers the topics of interest of EDBT 2023 (Criterion 1). Figure 3 plots the distributions of expertise at iteration 8 (final) and the primary
subject areas of the submissions in Cycle 3 (as declared by the authors). Observe that the final PC covers the areas of submissions adequately. In particular, since graphs, DB & AI, and data analytics are popular topics, we recruited more PC members in these areas. Furthermore, at each iteration of PC invitations we maintained a healthy balance of junior and senior members (Criteria 3). Around 60% of the final PC members were seniors, which we believe is a good distribution.

Another important criterion during this process is the reviewer network of the PC members in each iteration. The reviewer network $G_i = (V_i, E_i)$ at iteration $0 < i \leq 8$ is an undirected, labeled, weighted graph where $V_i$ is a set of (candidate) reviewers and $E_i$ is a set of co-authorship edges between them. Given a pair of $u, v \in V_i$, $(u, v) \in E_i$ if $u$ and $v$ have co-authored one or more articles. A node $u \in V_i$ is labeled with a unique identifier and $(u, v) \in E_i$ is labeled with a weight $w$ representing the number of co-authored articles by $u$ and $v$. We generate a reviewer network by exploiting the DBLP dataset of candidate reviewers. $V_i$ comprises accepted PC members till iteration $i - 1$ (denoted by $V_a$), invited candidates at iterations $0 < j \leq i - 1$, denoted by $V_u$, who have not responded yet (i.e., not marked as decline or accept), and candidates whom we intend to send invitations at iteration $i$ (denoted by $V_c$). That is, $V_i = V_a \cup V_u \cup V_c$. Note that when $i = 1$, $V_a = V_u = \emptyset$. When $i = 8$, we generate $G_8$ after the desired size was attained by updating the decision attribute of unresponsive candidates to ‘decline’. That is, $V_c = V_u = \emptyset$.

In line with Criterion 5, at each iteration, we ensured that the reviewer network was sparse and there were no significant hubs (PC members who have collaborated with many other members). Figure 4 depicts the final reviewer network. Note that the goal here is not visual clarity but to visually appreciate the sparseness of the network. Figure 5 reports the numbers of nodes, edges, and connected components, the size of the giant component (GC), density, and average clustering coefficient (CC) of the reviewer network at each iteration. The key observation here is that at each iteration of PC invitations we maintain a sparse network (low density and CC) and the size of the giant component is less than half of the number of nodes in most iterations. As we shall see later, these values are significantly superior to other major data management venues. Note that several network properties may not monotonically increase with $i$ if there are declinations from candidates invited in the prior iterations.

Figure 6 plots the degree distributions of the network at two iterations. Observe that the maximum degree of a node is 7 in the final PC and many PC members do not have any co-authorship relationships with other members (i.e., many isolates).

Figure 7 plots the distributions of COE and location of candidate reviewers at two iteration points. At each iteration we ensured that the PC is diverse
and no single group dominates significantly. In particular, the maximum size of any group does not exceed 30% of the PC for these measures.

### 3.3 Comparison

In the following, we compare the reviewer network and PC properties of EDBT 2023 with the 2023 editions of SIGMOD, VLDB, and ICDE whose PC formation follows the traditional approach. Reviewer networks of all venues are generated using CLOSET using the approach described in [4]. Figure 8 plots the results for average clustering coefficient (CC), the fraction of nodes in the giant component (GC), density, and average degree of the networks. Clearly, the EDBT 2023 reviewer network is sparser than these venues.

Figure 9 reports the degree distributions. Observe that the maximum degree of nodes is lowest in EDBT 2023 (Figure 6, right). Furthermore, the number of isolates (i.e., nodes with 0 degree) forms the largest group size in EDBT 2023. This is not the case for any of the other venues.

Finally, we make some observations related to the country of tertiary/secondary education (COE) and location distributions. The largest group sizes for COE (resp. location) distribution for ICDE 2023, VLDB 2023, and SIGMOD 2023 involve 53% (resp. 28%), 42% (resp. 33%), and 27% (resp. 42%), respectively, of the PC members. Hence, the COE distribution of EDBT 2023 is comparable to SIGMOD 2023 and significantly less skewed compared to ICDE 2023 and VLDB 2023. Similarly, the location distribution is comparable to ICDE 2023 and less skewed compared to the other two venues. It is evident that the manual approach of PC formation may result in overrepresentation of certain groups.

In summary, the data-driven PC formation framework deployed in EDBT 2023 facilitated the formation of a PC whose features are closer to the PC selection criteria in Section 2.

### 4. REVIEW PROCESS ANALYSIS

In this section, we quantitatively analyse the review process of EDBT 2023 undertaken by the PC formed by the proposed framework. There were three submission cycles for EDBT 2023. As PC chairs we were in charge of the second and third cycles. The first cycle was managed by the PC chairs of EDBT 2022. We received 29 and 88 submissions for the second and third cycles, respectively. We deployed a bidding-based automated reviewer assignment process for submissions hosted on Microsoft’s CMT. We utilized CLOSET [4] to manage submissions and reviewer assignments that violate EDBT 2023 COI policy. Each reviewer was assigned a total of 5-6 submissions. Each senior PC (i.e., meta-reviewer) managed 8-11 submissions. Each submission was assigned 5 reviewers and at most 4 reviewers in the second and third cycles, respectively.

#### Diversity of Reviewer Assignments.

Since the reviewer assignment technique in CMT is opaque to end users, we could not tinker with it. Instead, we diversified the input (i.e., reviewers) through our PC formation framework so that the assignments could be diverse. Figure 10 plots the results (diversity score vs number of submissions). The higher the diversity score the greater is the diversity of the reviewer assignment. For instance, COE diversity score of 1 (resp. 0) indicates all reviewers for a submission have distinct (resp. same) COE. Observe that the diversity of assignments is good across all
Figure 11: Quality of the review process: timeliness (left), reviewer confidence (right).

the three dimensions with very few submissions, if any, have low diversity (< 0.5).

Reviewer connectivity. Next, we report the impact of maintaining a sparse reviewer network on the reviewer assignment. We compute the number of edges between the set of reviewers in each submission. Then, we compute the distribution of the number of submissions with reviewers having co-authorship relationships. We observe that 44.8% and 62% submissions have no edges between reviewers (i.e., isolated nodes) in the second and third cycles, respectively. Figure 10 (bottom-right) reports the distributions for submissions. Observe that for the majority of these cases only one or two edges exist among reviewers. That is, the sparseness of the reviewer network facilitates assignment of reviewers to submissions with very low connectivity without tinkering CMT’s assignment algorithm.

Quality of the review process. Lastly, we report on the quality of the review process. We quantify it by measuring the following three dimensions: review timeliness, author complaints, and reviewer confidence. We measure the timeliness by computing the number of reviews received per submission before the start of the author feedback phase. Figure 11 (left) reports the numbers. We received all five reviews for each submission on time in the second cycle. For the third cycle, except for one submission, all had at least three reviews prior to the deadline (maximum is four reviews) In particular, the missing reviews involved only one or two reviewers. That is, 92% of the PC completed their reviews before the author feedback phase. This is remarkable given that PC chairs have frequently lamented on late or missing reviews in major venues. For example, in the first three submission cycles of SIGMOD 2024, only about 20% of submissions had all three reviews by the review deadline [5].

Next, we report the number of complaints from authors regarding review quality and decision making process after the notification of results. For both

We acknowledge review process quality is multi-faceted with many dimensions (e.g., discussion quality, engagement) at play. Here we only consider a subset of them that is easily measurable. Lastly, we measure reviewer confidence, i.e., how confident reviewers were of their reviews w.r.t. the contents of submissions. To this end, in the review form we have an item on reviewer confidence. Specifically, a reviewer must select one of the following w.r.t. a submission he or she reviewed:

1. “I am certain that my evaluation is correct and I have worked on this problem/main topic.”
2. “I am willing to defend my evaluation and have a solid overview of the state of the art on the main topic of the paper.”
3. “I am willing to defend my evaluation but am lacking a detailed knowledge of the state of the art on the main topic of the paper.”
4. “I learnt the topic as I reviewed the paper.”

Note that Item 4 indicates lack of confidence of a reviewer on his or her review. Figure 11 (right) plots the results. Observe that very few submissions have a review where the reviewer has selected item 4. Overall, the PC of EDBT 2023 is confident of their reviews, which is a cornerstone for any quality review process.

5. CONCLUSIONS

In this paper, we report our experiences with the novel data-driven PC-chair-in-the-loop PC formation framework that we adopted for EDBT 2023. We depart from the traditional approach of PC formation by iteratively guiding the selection of candidate reviewers using various diversity, publication profile, and collaboration network-related data. Our experiences as well as data related to the review process convince us that such a data-driven approach may contribute to a high-quality review process. We emphasize that we neither claim that the proposed framework has a causal relationship with the quality of the review process nor the review quality is superior to other venues. In particular, the latter issue demands access to the private review data of different venues which is unavailable to us.

We hope that this article will inspire future PC chairs to adopt a data-driven framework for PC formation and review management and explore its impact on the review quality. We also encourage PC chairs of existing venues to undertake a comparative analysis of the aforementioned measures related to their review processes. This will provide deeper insights on the impact of the two PC formation approaches on the review process.

SIGMOD Record, June 2024 (Vol. 53, No. 2) 73
6. REFERENCES
   Peer-Reviewing Processes and Incentives: Data Management Community Survey Results. SIGMOD Rec., 52(4), Dec 2023.