SIGMOD Officers, Committees, and Awardees

**Chair**
Divyakant Agrawal  
Department of Computer Science  
UC Santa Barbara  
Santa Barbara, California  
USA  
+1 805 893 4385  
agraval <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), K. Selçuk Candan (SIGMOD Conference Coordinator), Rada Chirkova (SIGMOD Record Editor), Chris Jermaine (ACM TODS Editor in Chief), Divesh Srivastava (2021 SIGMOD PC co-chair), Stratos Idreos (2021 SIGMOD PC co-chair), Leonid Libkin (Chair of PODS), Sihem Amer-Yahia (SIGMOD Diversity and Inclusion Coordinator), Curtis Dyreson (Information Director)

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

SIGMOD Information Director:
Curtis Dyreson, Utah State University

Associate Information Directors:
Huiping Cao (SIGMOD Record), Georgia Koutrika (Blogging), Wim Martens (PODS), and Sourav S Bhowmick (SIGMOD Record)

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

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

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

PODS Executive Committee:
Leonid Libkin (chair), Christoph Koch, Reinhard Pichler, Dan Suciu, Yufei Tao, Jan Van den Bussche

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

SIGMOD Awards Committee:
Sunita Sarawagi (Chair), Volker Markl, Renée Miller, H. V. Jagadish, Yanlei Diao, Stefano Ceri
Jim Gray Doctoral Dissertation Award Committee:
Vanessa Braganholo (co-chair), Viktor Leis (co-chair), Bailu Ding, Immanuel Trummer, Joy Arulraj, Jose Faleiro, Gustavo Alonso, Wolfgang Lehner

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:

<table>
<thead>
<tr>
<th>Year</th>
<th>Recipient 1</th>
<th>Recipient 2</th>
<th>Recipient 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1992</td>
<td>Michael Stonebraker</td>
<td>Jim Gray</td>
<td>Philip Bernstein</td>
</tr>
<tr>
<td>1993</td>
<td>David DeWitt</td>
<td>C. Mohan</td>
<td>David Maier</td>
</tr>
<tr>
<td>1994</td>
<td>Serge Abiteboul</td>
<td>Hector Garcia-Molina</td>
<td>Rakesh Agrawal</td>
</tr>
<tr>
<td>1995</td>
<td>Rudolf Bayer</td>
<td>Patricia Selinger</td>
<td>Don Chamberlin</td>
</tr>
<tr>
<td>1996</td>
<td>Ronald Fagin</td>
<td>Michael Carey</td>
<td>Jeffrey D. Ullman</td>
</tr>
<tr>
<td>1997</td>
<td>Jennifer Widom</td>
<td>Moshe Y. Vardi</td>
<td>Masaru Kitsuregawa</td>
</tr>
<tr>
<td>1998</td>
<td>Umeshwar Dayal</td>
<td>Surajit Chaudhuri</td>
<td>Bruce Lindsay</td>
</tr>
<tr>
<td>1999</td>
<td>Stefano Ceri</td>
<td>Martin Kersten</td>
<td>Laura Haas</td>
</tr>
<tr>
<td>2000</td>
<td>Gerhard Weikum</td>
<td>Goetz Graefe</td>
<td>Raghu Ramakrishnan</td>
</tr>
<tr>
<td>2001</td>
<td>Anastasia Ailamaki</td>
<td>Beng Chin Ooi</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Recipient 1</th>
<th>Recipient 2</th>
<th>Recipient 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>Michael Stonebraker and Lawrence Rowe</td>
<td>Richard Hipp</td>
<td>Jeff Hammerbacher, Ashish Thusoo, Joydeep Sen Sarma, Christopher Olston, Benjamin Reed, and Utkarsh Srivastava</td>
</tr>
<tr>
<td>2016</td>
<td>Martin Kersten</td>
<td>Xiaofeng Bao, Charlie Bell, Murali Brahmade, James Corey, Neal Fagan, Raju Gulabani, Anurag Gupta, Kamal Gupta, James Hamilton, Andy Jassy, Tengiz Kharatishvili, Saelesh Krishnamurthy, Yan Leshinsky, Lon Lundgren, Pradeep Madhavarapu, Sandor Maurice, Grant McAlister, Sam McKelvie, Raman Mittal, Debanjan Saha, Swami Sivasubramanian, Stefano Stefani, and Alex Verbitski</td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td>Don Anderson, Keith Bostic, Alan Bram, Grg Burd, Michael Cahill, Ron Cohen, Alex Gorrod, George Feinberg, Mark Hayes, Charles Lamb, Linda Lee, Susan LoVerso, John Merrells, Mike Olson, Carol Sandstrom, Steve Sarette, David Schacter, David Sgleau, Mario Seltzer, and Mike Ubell</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>Recipient 1</th>
<th>Recipient 2</th>
<th>Recipient 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1992</td>
<td>Maria Zemankova</td>
<td>Gio Wiederhold</td>
<td>Yahiko Kambayashi</td>
</tr>
<tr>
<td>1995</td>
<td>Jeffrey Ullman</td>
<td>Avi Silberschatz</td>
<td>Won Kim</td>
</tr>
<tr>
<td>1999</td>
<td>Raghu Ramakrishnan</td>
<td>Michael Carey</td>
<td>Laura Haas</td>
</tr>
<tr>
<td>2002</td>
<td>Dan Rosenkrantz</td>
<td>Richard Snodgrass</td>
<td>Michael Ley</td>
</tr>
<tr>
<td>2004</td>
<td>Surajit Chaudhuri</td>
<td>Hongjun Lu</td>
<td>Tamer Özsu</td>
</tr>
<tr>
<td>2007</td>
<td>Hans-Jörg Schek</td>
<td>Klaus R. Dittrich</td>
<td>Beng Chin Ooi</td>
</tr>
<tr>
<td>2010</td>
<td>David Lomet</td>
<td>Gerhard Weikum</td>
<td>Marianne Winslett</td>
</tr>
<tr>
<td>2013</td>
<td>H.V. Jagadish</td>
<td>Kyu-Young Whang</td>
<td>Curtis Dyreson</td>
</tr>
<tr>
<td>2016</td>
<td>Samuel Madden</td>
<td>Yannis E. Ioannidis</td>
<td>Z. Meral Özsoyoğlu</td>
</tr>
<tr>
<td>2019</td>
<td>Ahmed Elmagarmid</td>
<td>Philippe Bonnet</td>
<td>Juliana Freire</td>
</tr>
<tr>
<td>2020</td>
<td>Stratos Idreos</td>
<td>Stefan Manegold</td>
<td>Ioana Manolescu</td>
</tr>
<tr>
<td>2020</td>
<td>Dennis Shasha</td>
<td></td>
<td></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
- **2021** Winner: Huanchen Zhang, **Honorable Mentions:** Erfan Zamanian, Maximilian Schleich, and Natacha Crooks

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

[Last updated: September 30, 2021]
Editor’s Notes

Welcome to the June 2022 issue of the ACM SIGMOD Record!

This issue starts with the Database Principles column featuring an article by Khamis, Ngo, Pichler, Suciu, and Wang. The authors present Datalog®, a recursive language that combines the elegance of Datalog with the power of semiring abstraction. This combination allows Datalog® to express iterative computations prevalent in modern data analytics, yet retain the declarative fixpoint semantics of Datalog. The authors also present novel optimization techniques, report experimental results, and outline open problems for future research.

The Surveys column presents an article by Dave, Celestino, Varde, and Anu. The authors focus on missed user requirements, also known as implicit requirements (IMRs), which are a major contributor to failures in software projects. The article presents a background of IMR data along with a detailed taxonomy of terms, definitions, and examples associated with this paradigm, provides novel insights for the data-science community, and outlines directions of future work.

The Research column features two articles. The first article, by Psallidas and colleagues, shares with the community the key findings resulting from amassing and analyzing approximately 40M data-science projects, both publicly available and internal to Microsoft. The authors provide actionable interpretations, share how the analysis has been used internally at Microsoft, and detail how the analysis is pragmatically useful to better inform investments by both practitioners and systems builders. The second article, by Ghosh, Gupta, Mehrotra, and Sharma, introduces EnrichDB – a system for supporting data enrichment within data-management systems. EnrichDB is based on a powerful enrichment data model and provides tools for making application programming very simple for developers.

It is a pleasure to reintroduce the column containing reminiscences on influential papers. This column used to be run in SIGMOD Record by Richard Snodgrass and Ken Ross, and is now being managed by Pınar Tözün. The first issue of the column presents contributions by M. Tamer Özsu, Alberto Lerner, and Tianzheng Wang.

The Advice to Mid-Career Researchers column features a contribution by David Maier entitled “Deciding what not to do.” The article details the author’s perspective on considerations that you could use to decide to accept or decline offers of professional service and activities.

The Future of Data(base) Education column presents two articles. The first article, by de Waal and Echeverri, discusses a recent effort called OpenDS4All, whose goal is to accelerate the creation of data-science curricula by providing a “starter set” of open-source training materials in data science, including PowerPoint slide decks and Jupyter Notebooks with Python code. The second article, by Fekete and Röhm, presents a process for making pedagogical decisions on constructing data-management classes in the presence of the wealth of content that can be included.

The DBrainstorming column, whose goal is to discuss new and potentially controversial ideas that might be of interest and potentially of benefit to the research community, features in this issue an article by Appuswamy making a case for passive, migration-free, standardized, long-term database archival. The author overviews the current challenges in archiving culturally significant historical data in databases, and presents potential solutions that could make low-cost, migration-free data archival feasible.
The Research Centers column presents an article by Cheng and colleagues that introduces the Social Technology and Research (STAR) Laboratory in the University of Hong Kong. The article describes the STAR team, details its focus and goals, and provides an overview of the team's efforts and achievements along the axes of research, technology, synergy, teaching, and impact.

The Reports column features two articles. The first article, by Amer-Yahia and colleagues, presents the diversity and inclusion initiative in database conferences, D&I@DB, and discusses the latest achievements in this space with a focus on specific conferences that took place in 2021. The authors also present D&I tools and outline the planned efforts going forward. The second article, by Athanassoulis and colleagues, summarizes the discussion of a panel held during VLDB 2021 titled "Artifacts, Availability & Reproducibility," detail the recent efforts in this space, and outline the goals and a roadmap for the future.

On behalf of the SIGMOD Record Editorial board, I hope that you enjoy reading the June 2022 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 2022

Past SIGMOD Record Editors:

Datalog in Wonderland∗

Mahmoud Abo Khamis
RelationalAI
Hung Q. Ngo
RelationalAI
Reinhard Pichler
TU Wien
Dan Suciu
University of Washington
Yisu Remy Wang
University of Washington

ABSTRACT
Modern data analytics applications, such as knowledge graph reasoning and machine learning, typically involve recursion through aggregation. Such computations pose great challenges to both system builders and theoreticians: first, to derive simple yet powerful abstractions for these computations; second, to define and study the semantics for the abstractions; third, to devise optimization techniques for these computations.

In recent work we presented a generalization of Datalog called Datalog◦, which addresses these challenges. Datalog◦ is a simple abstraction, which allows aggregates to be interleaved with recursion, and retains much of the simplicity and elegance of Datalog. We define its formal semantics based on an algebraic structure called Partially Ordered Pre-Semirings, and illustrate through several examples how Datalog◦ can be used for a variety of applications. Finally, we describe a new optimization rule for Datalog◦, called the FGH-rule, then illustrate the FGH-rule on several examples, including a simple magic-set rewriting, generalized semi-naïve evaluation, and a bill-of-material example, and briefly discuss the implementation of the FGH-rule and present some experimental validation of its effectiveness.

1. INTRODUCTION
The database community has developed an elegant abstraction for recursive computations, in the context of Datalog [1, 45]. Query evaluation and optimization techniques such as semi-naïve evaluation and magic-set transformation have led to efficient implementations [24, 3, 4]. Datalog and its optimization, however, are not sufficiently powerful to handle the computations commonly found in the modern data stack; for example, new applications often involve iterative computations with aggregations such as summation or minimization over the reals. Even under the Boolean domain, clean and practical fixpoint semantics require strong assumptions on the input rules such as stratification [1]. In addition to difficult semantic questions, semi-naïve evaluation and magic-set transformation also impose strong assumptions about the input such as monotonicity in terms of set-containment. These assumptions typically do not hold in the new world.

A powerful abstraction to address these challenges can be found in the study of aggregation with the aid of semirings [21]. This research line generalizes relational algebra beyond sets and multisets. The semiring semantics can capture a wide range of operations including tensor algebra.

In recent works [27, 26, 46, 47], we combine the elegance of Datalog and the power of semiring abstractions into a new language called Datalog◦. We studied its semantics, optimization, and convergence behavior. In particular, we derived a novel optimization primitive called the FGH-rule, which serves as a key component for generalizing both semi-naïve evaluation and magic-set transformations to Datalog◦. This article highlights our findings.

Semantics of Datalog◦. A Datalog program is a collection of (unions of) conjunctive queries, operating on relations. Analogously, a Datalog◦ program is a collection of (sum-) sum-product queries on $S$-relations for some (pre-) semiring $S$ [20]. In short, Datalog◦ is like Datalog, where $\land, \lor$ are replaced by $\otimes, \oplus$. Recall that an $S$-relation is a function from the set of tuples to a (pre-) semiring $S$; the domain of the tuples is called the key space, while $S$ is the value space. The value space of standard relations is the Booleans (see Sec. 3).

EXAMPLE 1.1. A matrix $A$ over the real numbers is an $R$-relation, where each tuple $A[i,j]$ has the value $a_{ij}$; $R$ denotes the sum-product semiring $(\mathbb{R}, +, \cdot, 0, 1)$. Both the objective and gradient of the ridge linear regression problem $\min_J J(x)$, with $J(x) = \frac{1}{2} ||Ax - b||^2 + (\lambda/2)||x||^2$, are expressible in Datalog◦, because they

$^*$Suciu and Wang were partially supported by NSF IIS 1907997 and NSF IIS 1954222. Pichler was supported by the Austrian Science Fund (FWF):P30930.

$^1K$-relations were introduced by Green et al. [21]; we call them $S$-relations in this paper where $S$ stands for “semiring”.

SIGMOD Record, June 2022 (Vol. 51, No. 2)
are sum-sum-product queries. The gradient \( \nabla J(x) = A^T Ax - A^T b + \lambda x \), for example, is the following sum-sum-product query:

\[
\nabla[j] = \sum_{i} a[k,i] \cdot a[k,j] \cdot x[i] + \sum_{j} -1 \cdot a[j,i] \cdot b[j] + \lambda \cdot x[j]
\]

The gradient has the same dimensionality as \( x \), and the group-by variable is \( i \). Gradient descent is an algorithm to find the solution of \( \nabla J(x) = 0 \), or equivalently to solve for a fixpoint solution to the Datalog\(^*\) program \( x = f(x) \) where \( f(x) = x - \alpha \nabla J(x) \) for some step-size \( \alpha \).\(^2\)

**Example 1.2.** The all-pairs shortest paths (APSP) problem is to compute the shortest path length \( P[x,y] \) between any pair \( x,y \) of vertices in the graph, given the length \( E[x,y] \) of edges in the graph. The value space of \( E[x,y] \) can be the reals \( \mathbb{R} \) or the non-negative reals \( \mathbb{R}_{\geq 0} \). The APSP problem in Datalog\(^*\) can be expressed very compactly as:

\[
P[x,y] := \min(E[x,y], \min(P[x,z] + E[z,y]))
\]  

(1)

where \((\min,+)\) are the “addition” and “multiplication” operators in the partial order \( S \). See Ex. 3.1.

By changing the semiring, Datalog\(^*\) is able to express similar problems in exactly the same way. For example, (1) becomes transitive closure over the Boolean semiring, the \( p + 1 \) shortest paths over the \( \top_{p} \) semiring, and so forth.

In Datalog, the least fixpoint semantics was defined w.r.t. set inclusion [1]. To generalize this semantics for Datalog\(^*\), we generalize set inclusion to a partial order over \( S \)-relations. We define a partially ordered, pre-semiring (POPS, Sec. 3) to be any pre-semiring \( S \) [20] with a partial order, where both \( \ominus \) and \( \otimes \) are monotone operations. Thus, in Datalog\(^*\) the value space is always some POPS. Given this partial order, the semantics of a Datalog\(^*\) program is defined naturally, as the least fixpoint of the immediate consequence operator.

**Optimizations.** While Datalog is designed for iteration, Datalog engines typically optimize only the loop body but not the actual loop. The few systems that do, are limited to a small number of hard-coded optimizations, like magic-set rewriting and semi-naïve evaluation. Datalog\(^*\) supports these classic optimizations, and more. We describe these optimizations in Sec. 4.1, but give here a brief preview, and start by illustrating how the semi-naïve algorithm extends to Datalog\(^*\). Consider the program computing the transitive closure of \( E \):

\[
P(x,y) := E(x,y) \lor \bigvee_{z} (P(x,z) \land E(z,y))
\]  

(2)

We use parentheses like \( E(x,y) \) for standard relations, whose value space is the set of Booleans, and use square brackets, like \( E[x,y] \), when the value space is something else. The rule (2) deviates only slightly from standard Datalog syntax, in that it uses explicit conjunction and disjunction, and binds the variable \( z \) explicitly. After initializing \( P_0(x,y) = \delta_0(x,y) = E(x,y) \), at the \( t \)’th iteration, the semi-naïve algorithm does the following:

\[
\delta_t(x,y) = \left( \bigvee_{z} (\delta_{t-1}(x,z) \land E(z,y)) \right) \ominus P_{t-1}(x,y)
\]  

(3)

\[
P_t(x,y) = P_{t-1}(x,y) \ominus \delta_t(x,y)
\]

By computing the \( \delta \) relation so, we avoid re-deriving many facts in each iteration. Another way to see this is that, when \( \delta \) is much smaller than \( P \), then the join between \( \delta \) and \( E \) in (3) is much cheaper than the join between \( P \) and \( E \) in (2). The set difference operation aims precisely to keep \( \delta \) small. Somewhat surprisingly, the same principle can be extended to Datalog\(^*\), as we illustrate next.

**Example 1.3 (APSP-SN).** The semi-naïve algorithm for the APSP problem (Example 1.2) is:

\[
\delta_t[x,y] = \left( \min_{z} (\delta_{t-1}[x,z] + E[z,y]) \right) \ominus P_{t-1}[x,y]
\]  

(4)

The difference operator \( \ominus \) is defined as follows:

\[
b \ominus a = \begin{cases} 
  b & \text{if } b < a \\
  \infty & \text{if } b \geq a
\end{cases}
\]

As in the standard semi-naïve algorithm, our goal is to keep \( \delta \) small, by storing only tuples with a finite value, \( \delta[x,y] \neq \infty \). We use the \( \ominus \) operator for that purpose. Consider the rule (4). If \( b = \delta_{t-1}[x,z] + E[z,y] \) is the newly discovered path length, and \( a = P_{t-1}[y,x] \) is the previously known path length, then \( b \ominus a \) is finite if \( b < a \), i.e. only when the new path length strictly decreases. Correctness of the semi-naïve algorithm follows from the identity \( \min(a,b \ominus a) = \min(a,b) \). We note that, recently, Budiu et al. [7] have developed a very general incremental view maintenance technique, which also leads to the semi-naïve algorithm, for the case when the value space is restricted to an abelian group.

We have introduced in [47] a simple, yet very general optimization rule, called the FGH-rule. The semi-naïve algorithm is one instance of the FGH-rule, but so are many other optimizations, as we illustrate in Sec. 4.
a brief preview, we illustrate the FGH-optimization with the following example:

**Example 1.4.** Soufflé [24] is a popular open source Datalog system that supports aggregates, but does not allow aggregates in recursive rules. This means that we cannot write \( \text{APSP} \) as in rule (1). The common workaround is to stratify the program: we first compute the lengths of all paths between each pair of vertices, then take the minimum:

\[
\begin{align*}
P_{\text{all}}(x,y,d) & : = E(x,y,d). \\
P_{\text{all}}(x,y,d_1 + d_2) & : = E(x,z,d_1), P_{\text{all}}(z,y,d_2). \\
P(x,y) & = \min_d \{ d \mid P_{\text{all}}(x,y,d) \}
\end{align*}
\]

Of course, this program diverges on graphs with cycles, and is quite inefficient on acyclic graphs. The FGH-rule rewrites this naïve program into (1).

**Organization.** Sec. 2 discusses related work; Sec. 3 introduces the syntax and semantics of Datalog\(^2\); Sec. 4 describes the FGH-framework for optimizing Datalog\(^2\) programs, including magic-set transformation and semi-naïve evaluation; finally, Sec. 5 concludes.

2. RELATED WORK

Researchers have extended Datalog in many ways to enhance its expressiveness. Some of these extensions also reveal opportunities for various optimizations. This section surveys some existing research on Datalog extensions (Sec. 2.1) and their optimization (Sec. 2.2).

2.1 Datalog Extensions

Pure Datalog is very spartan: neither negation nor aggregate is allowed. Therefore, the literature on Datalog extensions is vast, and we will not attempt to cover the whole space. Instead, we focus on extensions that aim to support aggregates in Datalog. These include, but are not limited to, the standard \( \min, \max, \sum, \text{and COUNT} \) aggregates in SQL.

The main challenge in having aggregates is that they are not monotone under set inclusion, yet monotonicity is crucial for the declarative semantics of Datalog, and optimizations like semi-naïve evaluation. Consider the APSP Example 1.2. We could attempt to write it in Datalog, by extending the language with a min-aggregation:

\[
\begin{align*}
P(x,y,d) & : = E(x,y,d) \\
P(x,y,\min(d)) & : = P(x,z,d_1), E(z,y,d_2). \quad d = d_1 + d_2
\end{align*}
\]

However, the second rule is not monotone w.r.t. set inclusion. This is a subtle, but important point. For example, fix \( E = \{ (b,c,20), (b',c,10) \} \); if \( P \) is \( \{ (a,b,1) \} \), then the output of the rule is \( \{ (a,c,21) \} \), but when \( P \) is the superset \( \{ (a,b,1), (a,b',1) \} \), then the output is \( \{ (a,c,11) \} \), which is not a superset of the previous output. \( \{ (a,c,21) \} \nsubseteq \{ (a,c,11) \} \).

Approaches to resolve the tension between aggregates and monotonicity mainly follow two strategies: break the program into strata, or generalize the order relation to ensure that aggregates become monotone.

**Stratified Aggregates.** The simplest way to add aggregates to Datalog while staying monotone is to disallow aggregates in recursion. Proposed by Mumick et al. [35], the idea is inspired by stratified negation, where every negated relation must be computed in a previous stratum. Ex. 1.4 is stratified: the first two rules form the first stratum and compute \( P_{\text{all}} \) to a fixpoint in regular Datalog\(^3\), and the last rule performs the \( \min \) aggregate in its own stratum. Stratifying aggregates has the benefit that the semantics, evaluation algorithms, and optimizations for classic Datalog can be applied unchanged to each stratum. However, stratification limits the programs one is allowed to write – Ex. 1.2 is not stratified, and would therefore be invalid. Since Ex. 1.2 is equivalent to but more efficient than Ex. 1.4, disallowing the former leads to suboptimal performance. The stratification requirement can also be a cognitive burden on the programmer. In fact, the most general notion of stratification, dubbed “magic stratification” [35], involves both a syntactic condition and a semantic condition defined in terms of derivation trees.

**Generalized Ordering.** In this article, we follow the approach that restores monotonicity by generalizing the ordering on which monotonicity is defined. The key idea is that, although a program like that shown in Eq. (5) is not monotone according to the \( \subseteq \) ordering on sets, we can pick another order under which \( P \) is monotone. Ross and Sagiv [39] define the ordering\(^4\) \( \sqsubseteq \) on \( P \) as:

\[
\forall (x,y,d) \in P, \exists (x,y,d') \in P' : (x,y,d) \sqsubseteq (x,y,d')
\]

where \( (x,y,d) \sqsubseteq (x',y',d') \) \( \overset{\text{def}}{=} x = x' \land y = y' \land d \geq d' \).

That is, \( P \) increases if we replace \( (x,y,d) \) with \( (x,y,d') \) where \( d' < d \), for example \( \{ (a,c,21) \} \nsubseteq \{ (a,c,11) \} \).

In general, to define a generalized ordering we need to view a relation as a map from a tuple to an element in some ordered set \( S \). For example, the relation \( P \) maps a pair of vertices \( (x,y) \) to a distance \( d \). We will call such generalized relations \( S \)-relations. Different approaches in existing work have modeled \( S \) using different algebraic structures: Ross and Sagiv [39] require it to be a complete lattice, Conway et al. [9] require only a semi-lattice, whereas Green et al. [21] require \( S \) to be an \( \omega \)-continuous semiring. These proposals bundle the order-

\(^3\)The second rule uses the built-in function +.

\(^4\)If \( (x,y) \) is not a key in \( P \), then \( \subseteq \) is only a preorder.
ing together with two operations $\otimes$ and/or $\oplus$. In this article, we follow this line of work and ensure monotonicity by generalizing the ordering. However, in contrast to prior work we decouple operations on $S$-relations from the ordering, and allow one to freely mix and match the two as long as monotonicity is respected.

**Other Approaches.** There are other approaches to support aggregates in Datalog that do not fall into the two categories above. We highlight a few of them here. Ganguly et al. [14] model min and max aggregates in Datalog with negation, thereby supporting aggregates via semantics defined for negation. Mazuran et al. [34] extend Datalog with counting quantification, which additionally captures $\text{SUM}$. Kemp and Stuckey [25] extend the well-founded semantics [16, 15] and stable model semantics [17] of Datalog to support recursive aggregates. They show that their semantics captures various previous attempts to incorporate aggregates into Datalog. They also discuss a semantics using closed semirings, and observe that under such a semantics some programs may not have a unique stable model. Semantics of aggregation in Answer Set Programming has been extensively studied [12, 18, 2]. Liu and Stoller [33] give a comprehensive survey of this space.

## 2.2 Optimizing Extended Datalog

New opportunities for optimization emerge once we extend Datalog to support aggregation. We have already seen one instance in Ex. 1.4 and Ex. 1.2. Intuitively, the optimization can be seen as applying the group-by pushdown rule to recursive programs; or, it can be seen as a variant of the magic-set transformation, where we push the aggregate instead of a predicate into recursion. The optimizations we study in this article are inspired by a long line of work done by Carlo Zaniolo and his collaborators [14, 51, 50, 49]. Indeed, our FGH-rule is a generalization of Zaniolo et al.’s notion of pre-mappability (PreM) [51], which evolved from earlier ideas on pushing extrema (min / max aggregate) into recursion [50]. A different and more recent solution to aggregate pushdown by Shkapsky et al. [42] is to use set-monotonic aggregation operators. Unlike PreM, pushing monotonic aggregates requires no preconditions, but may result in slightly less efficient programs.

In addition to new optimizations like aggregate pushdown, classic techniques including semi-naive evaluation and magic-set transformation also exhibit interesting twists in extended Datalog variants. For example, Conway et al. [9] generalize the set-based semi-naive evaluation into one over $S$-relations where $S$ is a semilattice, and Mumick et al. [35] adapt magic-set transformation to work in the presence of aggregate-in-recursion. In this article, we will show how the FGH-rule can capture both the magic-set transformation and the general semi-naive evaluation.

To evaluate a recursive Datalog program is to solve fixpoint equations over semirings, which has been studied in the automata theory [28], program analysis [10, 36], and graph algorithms [8, 32, 31] communities since the 1970s. (See [40, 23, 29, 20, 52] and references thereof). The problem took slightly different forms in these domains, but at its core, it is to find a solution to the equation $x = f(x)$, where $x \in S^n$ is a vector over the domain $S$ of a semiring, and $f : S^n \rightarrow S^n$ has multivariate polynomial component functions. A literature review on this fixpoint problem can be found in [26].

## 3. DATALOG

Datalog$^*$ extends Datalog in two ways. First, all relations are $S$-relations over some semiring $S$. Second, the semiring needs to be partially ordered; more precisely, it needs to be a POPS.

**POPS** A partially ordered pre-semiring, or POPS, is a tuple $S = (S, \oplus, \otimes, 0, 1, \sqsubseteq)$, where:

- $(S, \oplus, \otimes, 0, 1)$ is a pre-semiring, meaning that $\oplus$, $\otimes$ are commutative and associative, have identities $0$ and $1$ respectively, and $\otimes$ distributes over $\oplus$.
- $\sqsubseteq$ is a partial order with a minimal element, $\bot$.
- Both $\oplus$, $\otimes$ are monotone operations w.r.t. $\sqsubseteq$.

Pre-semirings have been studied intensively [20], and we need to straighten up some terminology before proceeding. A pre-semiring only requires $\oplus$ to be commutative: if $\otimes$ is also commutative, then it is called a commutative pre-semiring. All pre-semirings in this paper are commutative. If $x \otimes 0 = 0$ holds for all $x$, then $S$ is called a semiring. We call $\otimes$ strict if $x \otimes z = \bot$ for all $x$; throughout this paper we will assume that $\otimes$ is strict.

When the relation $r \sqsubseteq y$ is defined by $\exists z : x \oplus z = y$, then $\sqsubseteq$ is called the natural order. In that case, the minimal element is $\bot = 0$. Naturally ordered semirings appear often in the literature [20, 21, 11], but we do not require POPS to be naturally ordered (see Example 3.2).

**S-Relations** Fix a POPS $S$, and a domain $D$, which, for simplicity, we will assume to be finite. An $S$-relation is a function $R : D^k \rightarrow S$. We call $D^k$ the key space and $S$ the value space. When $S$ is the set of Booleans, which we denote $\mathbb{B}$, then a $\mathbb{B}$-relation is a standard relation, i.e. a set. Next, we need to define sum-product, and sum-sum-product expressions, which are generalizations of Conjunctive Queries (CQ), and Unions of Conjunctive Queries (UCQ):

$$T[x_1, \ldots, x_k] ::= \bigoplus_{x_{k+1}, \ldots, x_p \in D} \{A_1 \otimes \cdots \otimes A_m | C\}$$

$$F[x_1, \ldots, x_k] ::= T_1[x_1, \ldots, x_k] \oplus \cdots \oplus T_q[x_1, \ldots, x_k]$$
The sum-product expression (6) defines a new \( S \)-relation \( T \), with head variables \( x_1, \ldots, x_k \). It consists of a summation of products, where the bound variables \( x_{k+1}, \ldots, x_p \) range over the domain \( D \), and may be further restricted to satisfy a condition \( C \). Each factor \( A_n \) is either a relational atom, \( R_i[x_{i1}, \ldots, x_{i\eta_i}] \), where \( R_i \) is a relation name from some given vocabulary, or an equality predicate, \( [x_i = x_j] \). The sum-sum-expression (7) is a sum of sum-product expressions, with the restriction that all summands have the same head variables.

**Datalog** The input to a Datalog\(^\text{\textsuperscript{c}}\) program consists of \( m \) EDB predicates\(^\text{\textsuperscript{5}}\) \( E = (E_1, \ldots, E_m) \), and the output consists of \( n \) IDB predicates \( P = (P_1, \ldots, P_n) \). The Datalog\(^{\text{\textsuperscript{c}}} \) program has one rule for each IDB:

\[
P_i[\text{vars}_i] := F_i[\text{vars}_i]
\]

\[
\ldots
\]

\[
P_n[\text{vars}_n] := F_n[\text{vars}_n]
\]

where each \( F_i[\text{vars}_i] \) is a sum-sum-product expression using the relation symbols \( E_1, \ldots, E_m, P_1, \ldots, P_n \). We say that the program is linear if each product contains at most one IDB predicate.

**Semantics** The tuple of all \( n \) sum-sum-product expressions \( F = (F_1, \ldots, F_n) \) is called the Immediate Consequence Operator, or ICO. Fix an instance of the EDB relations \( E \). The ICO defines a function \( P \mapsto F(E, P) \) that maps the IDB instances \( P \) to new IDB instances. The semantics of a Datalog\(^{\text{\textsuperscript{c}}} \) program is the least fixpoint of the ICO, when it exists. Equivalently, the Datalog\(^{\text{\textsuperscript{c}}} \) program is the result returned by the following naïve evaluation algorithm:

\[
P_0 = \bot; \quad t = 0;
\]

repeat \( P_{t+1} = F(E, P_t) \);

\( t = t + 1 \);

until \( P_t = P_{t-1} \).

The reader may have recognized that Datalog\(^{\text{\textsuperscript{c}}} \) is quite similar to Datalog, with minor changes: the operations \( \lor, \land \) are replaced with \( \oplus, \odot \), and multiple rules for the same IDB predicate are combined into a single sum-product rule. Importantly, Datalog\(^{\text{\textsuperscript{c}}} \) retains the same simple fixpoint semantics, but it generalizes from sets to \( S \)-relations. We illustrate this with two examples.

**Example 3.1.** Consider the one-rule program:

\[
P[x, y] := E[x, y] \odot \bigoplus_z (P[x, z] \odot E[z, y])
\]

We will interpret it over several POPS and use it to compute quite different things.

\(^5\)EDB and IDB stand for extensional database and intentional database respectively [1].

**Booleans** Choose \( B = \{0, 1\}, \land, \lor, \neg, 0, 1 \) to be the value space, where \( 0 \leq 1 \). Then the program in Eq. (9) becomes the transitive closure program in Eq. (2).

**Tropical Semiring** \( Trop^+ = (\mathbb{R}_+, \min, +, +, 0, \geq) \) is a naturally ordered POPS, called the tropical semiring. When we choose it as value space, then the program in Eq. (9) becomes the APSP program in Eq. (1). We briefly illustrate its semantics on a graph with three nodes \( a, b, c \) and edges (we show only entries with value \( < \infty \)):

\[
E[a, b] = 1 \quad E[a, c] = 10 \quad E[b, c] = 1
\]

During the iterations \( t = 0, 1, 2, \ldots \) of the naïve algorithm, \( P \) “grows” as follows (notice that the order relation in \( Trop^+ \) is the reverse of the usual one, thus \( \infty \) is the smallest value):

\[
\begin{array}{cccc}
\text{t} & P[a, b] & P[a, c] & P[b, c] \\
0 & \infty & \infty & \infty \\
1 & 1 & 10 & 1 \\
2 & 1 & 2 & 1 \\
\end{array}
\]

\( p \)-**Tropical** We can use the same program over a different POPS to compute the \( p + 1 \) shortest paths, for some fixed number \( p \geq 0 \). We need some notations. If \( x = \{x_0 \leq x_1 \leq \cdots \leq x_n\} \) is a bag of numbers, then we denote by \( \min_p(x) \) \( \triangleq \{x_0, x_1, \ldots, x_{\min \{p, n\}}\} \). In other words, \( \min_p \) retains the smallest \( p + 1 \) elements of the bag \( x \). The \( p \)-tropical semiring is:

\[
Trop^+_p \triangleq (B_{p+1}(\mathbb{R}_+ \cup \{\infty\}), \oplus_p, \odot_p, 0_p, 1_p)
\]

where \( B_{p+1} \) represents bags of \( p + 1 \) elements, and:

\[
x \oplus_p y \triangleq \min_p(x \cup y) \quad x \odot_p y \triangleq \min_p(x \cap y)
\]

\[
0_p \triangleq \{\infty, \infty, \ldots, \infty\} \quad 1_p \triangleq \{0, 0, \ldots, \infty\}
\]

\( Trop^+_p \) is naturally ordered. Now the program (9) computes the length of the \( p + 1 \) shortest paths from \( x \) to \( y \).

\( \eta \)-**Tropical** Finally, we illustrate how the same program can be used to compute the length of all paths that differ from the shortest path by \( \leq \eta \), for some fixed \( \eta \geq 0 \). Given any finite set \( x \) of real numbers, define:

\[
\min_{\leq \eta}(x) = \{u \mid u \in x, u - \min(x) \leq \eta\}
\]

In other words, \( \min_{\leq \eta} \) retains from the set \( x \) the elements at distance \( \leq \eta \) from its minimum. Define the POPS:

\[
Trop^+_{\leq \eta} \triangleq (B_{\leq \eta}(\mathbb{R}_+ \cup \{\infty\}), \oplus_{\leq \eta}, \odot_{\leq \eta})
\]

where \( B_{\leq \eta} \) is the set of finite sets \( x \) where \( \max(x) - \min(x) \leq \eta \), and:

\[
x \oplus_{\leq \eta} y \triangleq \min_{\leq \eta}(x \cup y) \quad x \odot_{\leq \eta} y \triangleq \min_{\leq \eta}(x \cap y)
\]

\[
0_{\leq \eta} \triangleq \{\infty\} \quad 1_{\leq \eta} \triangleq \{0\}
\]

\( \forall x, y: x + y \triangleq \{u + v \mid u \in x, v \in y\} \).

For sets or bags \( x, y \):

\[
\frac{1}{\infty} = 0
\]
Trop^\leq_\eta is naturally ordered. The program (9) computes now the length of all paths that differ by \leq \eta from the shortest.

**Example 3.2.** Consider now the bill-of-material example: the relation SupPart(x,z) represents the fact that z is a subpart of x, and the goal is to compute, for each x, the total cost of all its direct and indirect subparts. This is written in Datalog^o as follows:

\[
Q[x] :- \text{Cost}[x] + \sum_z\{Q[z] \mid \text{SupPart}(x,z)\}
\]

We will interpret it over two different POPS:

1. **Natural Numbers** We start by interpreting the program over the POPS (\(\mathbb{N},+,*\), 0, 1, \(\leq\)). If the SupPart hierarchy is a tree, then Eq. (10) computes correctly the total cost, as intended. If SupPart is a DAG, then the program may over-count some costs; we will return to this issue in Ex. 4.5. For now, we consider termination, in the case when SupPart happens to have a cycle: in that case the program diverges.

2. **Lifted Reals (or Lifted Naturals)** Alternatively, consider the following POPS: \(\mathbb{R}_\bot = (\mathbb{R} \cup \{\bot\}, +, *, 0, 1, \leq)\), where \(x + \bot = x\) and \(\bot = \bot\) for all \(x\), and \(a \leq b\) if \(a = b\) or \(a = \bot\). This POPS is not naturally ordered. When we interpret the program in Eq. (10) over \(\mathbb{R}_\bot\), then it always converges, even on a graph with cycles, because all nodes on a cycle will converge with \(Q[x] = \bot\).

Note that, in the above examples, only \(\mathbb{R}_\bot\) (likewise \(\mathbb{N}_\bot\)) is a pre-semiring. All other POPS discussed here are actually semirings.

4. Optimizing Datalog

Traditional Datalog has two major advantages: first, it has a clean declarative semantics; second, it has some powerful optimization techniques such as the semi-naïve evaluation, magic-set rewriting, and the PreM optimization [51]. Datalog^o generalizes both: we have seen its semantics in Sec. 3, while here we show (following [26, 47]) that the previous optimizations are special cases of a general, yet very simple optimization rule, which we call the FGH-rule (pronounced “fig-rule”).

4.1 The FGH-Rule

Consider an iterative program that repeatedly applies a function \(F\) until some termination condition is satisfied, then applies a function \(G\) that returns the final answer \(Y\):

\[
X \leftarrow X_0 \\
\text{loop } X \leftarrow F(X) \text{ end loop} \quad (11)
\]

\[
Y \leftarrow G(X)
\]

We call this an FG-program. The FGH-rule provides a sufficient condition to compute the final answer \(Y\) by another program, called the GH-program:

\[
Y \leftarrow G(X_0) \\
\text{loop } Y \leftarrow H(Y) \text{ end loop} \quad (12)
\]

The FGH-Rule [47] states: if the following identity holds:

\[
G(F(X)) = H(G(X)) \quad (13)
\]

then the FG-program (11) and the GH-program (12) are equivalent. We supply here a “proof by picture” of the claim:

\[
\begin{align*}
X_0 \xrightarrow{F} X_1 \xrightarrow{F} X_2 & \ldots \xrightarrow{F} X_n \\
Y_0 \xrightarrow{H} Y_1 \xrightarrow{H} Y_2 & \ldots \xrightarrow{H} Y_n
\end{align*}
\]

Our goal is to use the FGH-rule to optimize Datalog^o programs, and we proceed as follows. Consider two Datalog^o programs \(\Pi_1\) and \(\Pi_2\) given below:

\[
\Pi_1 : \quad X :- F(X) \quad Y :- G(X) \\
\Pi_2 : \quad Y :- H(Y)
\]

Here \(X\) and \(Y\) are tuples of IDBs (for example \(X = (P_1,\ldots,P_n)\) with the notation in Sec. 3), and \(F,G,H\) represent sum-sum-product expressions over these IDBs. In both cases, only the IDBs \(Y\) are returned. Then, if the FGH-rule (13) holds, and, moreover, \(G(\bot) = \bot\), then \(\Pi_1\) is equivalent to \(\Pi_2\). We notice that, under these conditions, if \(\Pi_1\) terminates, then \(\Pi_2\) terminates as well.

We illustrate several applications of the FGH-rule in Sec. 4.2, then describe its implementation in an optimizer in Sec. 4.3.

4.2 Applications of the FGH-Rule

We start with some simple applications. Throughout this section we assume that the function \(H\) is given; we discuss in Sec. 4.3 how to synthesize \(H\).

**Example 4.1 (Connected Components).** We are given an undirected graph, with edge relation \(E(x,y)\), where each node \(x\) has a unique numerical label \(L[x]\). The task is to compute for each node \(x\), the minimum label \(CC[x]\) in its connected component. This program is a well-known target of query optimization in the literature [51]. A naïve approach is to first compute the reflexive and transitive closure of \(E\), then apply a min-aggregate:

\[
\begin{align*}
TC(x,y) & :- [x = y] \vee \exists z (E(x,z) \land TC(z,y)) \\
CC[x] & :- \min_y \{L[y] \mid TC(x,y)\}
\end{align*}
\]
Then we prove that $G(F(X)) = H(G(X))$ by exploiting the fact that $\mathcal{S}$ is a complete distributive lattice. In Example 4.3 (General Semi-Naïve). The algorithm for the naïve evaluation of (positive) Datalog rediscover each fact from step $t$ again at steps $t + 1, t + 2, \ldots$. The semi-naïve algorithm aims at avoiding this, by computing only the new facts. We generalize the semi-naïve evaluation from the Boolean semiring to any POPS $\mathcal{S}$, and prove it correct using the FGH-rule.

We require $\mathcal{S}$ to be a complete distributive lattice and $\oplus$ to be idempotent, and define the “minus” operation as:

$$ b \ominus a \triangleq \bigwedge \{ c \mid b \leq a \oplus c \} $$

then prove using the FGH-rule the following programs equivalent:

$$ \Pi_1 : \quad TC(x,y) := [x = y] \lor \exists z(E(x,z) \land TC(z,y)) $$

$$ Q(y) := TC(a,y) $$

(14)

where $a$ is some constant, into this program:

$$ \Pi_2 : \quad Q(y) := [y = a] \lor \exists z(Q(z) \land E(z,y)) $$

(15)

This is a powerful optimization, because it reduces the run time from $O(n^2)$ to $O(n)$. Several Datalog systems support some form of magic-set optimizations. We check that (14) is equivalent to (15) by verifying the FGH-rule. The functions $F, G, H$ are shown in Fig. 3. One can verify that $G(F(TC)) = H(G(TC))$, for any relation $TC$. Indeed, after converting both expressions to normal form (i.e. in sum-sum-product form), we obtain

$$ \Pi_0 : \quad \Pi_2 : 

\begin{align*}
X_0 & := \emptyset; \\
Y_0 & := \emptyset; \\
\Delta_0 & := F(\emptyset); \\
\Delta_0 & := F(Y_0) \\
\text{loop} & \ X_i := F(X_{i-1}); \\
\text{loop} & \ Y_i := Y_i + \Delta_i; \\
\Delta_i & := F(Y_i) \oplus Y_i;
\end{align*}$$

To prove their equivalence, we define:

$$ G(X) \triangleq (X, F(X) \land X) $$

$$ H(X, \Delta) \triangleq (X \lor \Delta, F(X \lor \Delta) \lor (X \lor \Delta)) $$

Then we prove that $G(F(X)) = H(G(X))$ by exploiting the fact that $\mathcal{S}$ is a complete distributive lattice. In

$SIGMOD$ Record, June 2022 (Vol. 51, No. 2)
practice, we compute the difference
\[ \Delta t = F(Y_t) \ominus Y_t = F(Y_{t-1} \oplus \Delta t_{-1}) \ominus F(Y_{t-1}) \]
using an efficient differential rule that computes \( \delta F(Y_{t-1}, \Delta t_{-1}) = F(Y_{t-1} \oplus \Delta t_{-1}) \ominus F(Y_{t-1}) \), where \( \delta F \) is an incremental update query for \( F \), i.e., it satisfies the identity \( F(Y) \oplus \delta F(Y, \Delta) = F(Y \oplus \Delta) \).

Thus, semi-naïve query evaluation generalizes from standard Datalog over the Booleans to Datalog over any complete distributive lattice with idempotent \( \oplus \), and, moreover, is a special case of the FGH-rule.

We remark that the FGH-rule is a generalization of an optimization rule introduced by Zaniolo et al. [51] and called Pre-mappability, or PreM. The PreM property asserts that the identity \( G(F(X)) = G(F(G(X))) \) holds: in this case one can define \( H \) as \( H(X) = G(F(X)) \), and the FGH-rule holds automatically. The PreM rule is more restricted than the FGH-rule, in two ways: the types of the IDBs of the F-program and the H-program must be the same, and the new query \( H \) is uniquely defined by \( F \) and \( G \), which limits the type of optimizations that are possible under PreM.

**Loop Invariants** We now describe a more powerful application of the FGH-rule, which uses loop invariants. The general principle is the following. Let \( \phi(X) \) be any predicate satisfying the following three conditions:

\[
\begin{align*}
\phi(X_0) \\
\phi(X) \Rightarrow \phi(F(X)) \\
\phi(X) \Rightarrow (G(F(X)) = H(G(X)))
\end{align*}
\]

then the FG-program (11) and the GH-program (12) are equivalent. This conditional FGH-rule is very powerful; we briefly illustrate it with an example.

**Example 4.4 (Beyond Magic).** Consider the following program:

\[
\begin{align*}
\Pi_1: & \quad TC(x, y) \Rightarrow [x = y] \lor \exists z (E(x, z) \land TC(z, y)) \\
Q(y) & \Rightarrow TC(a, y)
\end{align*}
\]  
(17)

which we want to optimize to:

\[
\Pi_2: \quad Q(y) \Rightarrow [y = a] \lor \exists z (Q(z) \land E(z, y))
\]  
(18)

Unlike the simple magic program in Ex. 4.2, here rule (17) is right-recursive. As shown in [6], the magic-set optimization using the standard sideways information passing optimization [1] yields a program that is more complicated than our program (18). Indeed, consider a graph that is simply a directed path \( a_0 \rightarrow a_1 \rightarrow \cdots \rightarrow a_n \) with \( a = a_0 \). Then, even with magic-set optimization, the right-recursive rule (17) needs to derive quadratically many facts of the form \( T(a_i, a_j) \) for \( i \leq j \), whereas the optimized program (18) can be evaluated in linear time. Note also that the FGH-rule cannot be applied directly to prove that the program (17) is equivalent to (18). To see this, denote by \( P_1 \triangleq G(F(TC)) \) and \( P_2 \triangleq H(G(TC)) \), and observe that \( P_1, P_2 \) are defined as:

\[
\begin{align*}
P_1(y) & \triangleq [y = a] \lor \exists z (E(a, z) \land TC(z, y)) \\
P_2(y) & \triangleq [y = a] \lor \exists z (TC(a, z) \land E(z, y))
\end{align*}
\]

In general, \( P_1 \neq P_2 \). The problem is that the FGH-rule requires that \( G(F(TC)) = H(G(TC)) \) for every input \( TC \), not just the transitive closure of \( E \). However, the FGH-rule does hold if we restrict \( TC \) to relations that satisfy the following loop-invariant \( \phi(TC) \):

\[
\exists z_1(E(x, z_1) \land TC(z_1, y)) \iff \exists z_2(TC(x, z_2) \land E(z_2, y))
\]  
(19)

If \( TC \) satisfies this predicate, then it follows immediately that \( P_1 = P_2 \), allowing us to optimize program (17) to (18). It remains to prove that \( \phi \) is indeed an invariant for the function \( F \). The base case (16) holds because both sides of (19) are empty when \( TC = \emptyset \). It remains to check \( \phi(TC) \Rightarrow \phi(F(TC)) \). Denote \( TC' \triangleq F(TC) \), then we need to check that, if (19) holds, then the predicate \( \Psi_1(x, y) \triangleq \exists z_1 (E(x, z_1) \land TC'(z_1, y)) \) is equivalent to the predicate \( \Psi_2(x, y) \triangleq \exists z_2 (TC'(x, z_2) \land E(z_2, y)) \). Using (19) we can prove the equivalence of the predicates \( \Psi_1 \) and \( \Psi_2 \).

We describe in [47] how to infer the loop invariant given a program and constraints on the input.

**Semantic optimization** Finally, we illustrate how the FGH-rule takes advantage of database constraints [38]. In general, a priori knowledge of database constraints can lead to more powerful optimizations. For instance, in [5], the counting and reverse counting methods are presented to further optimize the same-generation program if it is known that the underlying graph is acyclic. We present a principled way of exploiting such a priori knowledge. As we show here, recursive queries have the potential to use global constraints on the data during semantic optimization; for example, the query optimizer may exploit the fact that the graph is a tree, or the graph is connected. We will denote by \( \Gamma \) the set of constraints on the EDBs. Then, the FGH-rule (13) needs to be checked only for EDBs that satisfy \( \Gamma \), as we illustrate in this example:

**Example 4.5.** Consider again the bill-of-material problem in Ex. 3.2. SubPart \( (x, y) \) indicates that \( y \) is a subpart of \( x \), and \( \text{Cost}[x] \in \mathbb{N} \) represents the cost of the part \( x \). We want to compute, for each \( x \), the total cost \( \text{Cost}[x] \) of all its subparts, sub-subparts, etc. Recall from Ex. 3.2 that, if we insist on interpreting the program (10) over the natural numbers or reals (and not the lifted naturals \( \mathbb{N}_1 \), or lifted reals \( \mathbb{R}_1 \)), then a cyclic graph will cause the program to diverge. Even if the subpart hierarchy is a DAG, we have to be careful not to double count costs. Therefore, we first compute the transitive closure, and then sum up all costs:
Consider now the case when our subpart hierarchy is a tree. Then, we can compute the total cost much more efficiently by using the program in Eq. (10), repeated here for convenience:

\[\Pi_2: \quad Q[x] := \sum_y \{\text{Cost}[y] \mid S(x,y)\}\]

(21)

Optimizing the program (20) to (21) is an instance of semantic optimization, since this only holds if the database instance is a tree. We do this in three steps. First, we define the constraint \(\Gamma\) stating that the data is a tree. Second, using \(\Gamma\) we infer a loop-invariant \(\Phi\) of the program \(\Pi_1\). Finally, using \(\Gamma\) and \(\Phi\) we prove the FGH-rule, concluding that \(\Pi_1\) and \(\Pi_2\) are equivalent. The constraint \(\Gamma\) is the conjunction of the following statements:

\[
\begin{align*}
\forall x_1, x_2, y. \text{SubPart}(x_1, y) \land \text{SubPart}(x_2, y) \Rightarrow x_1 = x_2 \\
\forall x, y. \text{SubPart}(x, y) \Rightarrow T(x, y) \\
\forall x, y, z. T(x, z) \land \text{SubPart}(z, y) \Rightarrow T(x, y) \\
\forall x, y. T(x, y) \Rightarrow x \neq y
\end{align*}
\]

(22)

The first asserts that \(y\) is a key in \(\text{SubPart}(x, y)\). The last three are an Existential Second Order Logic statement: they assert that there exists some relation \(T(x, y)\) that contains \(\text{SubPart}\), is transitively closed, and irreflexive. Next, we infer the following loop-invariant of the program \(\Pi_1\):

\[\Phi : S(x, y) \Rightarrow [x = y] \lor T(x, y)\]

(24)

Finally, we check the FGH-rule, under the assumptions \(\Gamma, \Phi\). Denote by \(P_1 \overset{\text{def}}{=} G(F(S))\) and \(P_2 \overset{\text{def}}{=} H(G(S))\). To prove \(P_1 = P_2\), we simplify \(P_1\) using the assumptions \(\Gamma, \Phi\), as shown in Fig. 4. We explain each step. Line 2-3 are inclusion/exclusion. Line 4 uses the fact that the term on line 3 is \(0\), because the loop invariant implies:

\[
\begin{align*}
S(x, z) \land \text{SubPart}(z, y) \\
\Rightarrow \{S(x, z) \lor T(x, z) \land \text{SubPart}(z, y)\} & \quad \text{by (24)} \\
\equiv \{\text{SubPart}(x, y) \lor T(x, z) \land \text{SubPart}(z, y)\} & \quad \text{by (24)} \\
\Rightarrow T(x, y) \lor \text{SubPart}(z, y) \equiv T(x, y) & \quad \text{by (22)} \\
\Rightarrow x \neq y & \quad \text{by (23)}
\end{align*}
\]

The last line follows from the fact that \(y\) is a key in \(\text{SubPart}(z, y)\). A direct calculation of \(P_2 = H(G(S))\) results in the same expression as line 5 of Fig. 4, proving that \(P_1 = P_2\).

---

P_1 [x] = \sum_y \{\text{Cost}[y] \mid [x = y] \lor T(x, y)\}

\[
\begin{align*}
&= \text{Cost}[x] + \sum_y \{\text{Cost}[y] \mid \exists z. (S(x, z) \land \text{SubPart}(z, y))\} \\
&= \sum_y \{\text{Cost}[y] \mid [x = y] \lor \exists z. (S(x, z) \land \text{SubPart}(z, y))\} \\
&= \sum_y \{\text{Cost}[y] \mid (S(x, z) \land \text{SubPart}(z, y))\}
\end{align*}
\]

Figure 4: Transformation of \(P_1 \overset{\text{def}}{=} G(F(S))\) in Ex. 4.5.

### 4.3 Program Synthesis

In order to use the FGH-rule, the optimizer has to do the following: given the expressions \(F, G\) find the new expression \(H\) such that \(G(F(X)) = H(G(X))\). We will denote \(G(F(X))\) and \(H(G(X))\) by \(P_1, P_2\) respectively. There are two ways to find \(H\): using rewriting, or using program synthesis with an SMT solver.

**Rule-based Synthesis** In query rewriting using views we are given a query \(Q\) and a view \(V\), and want to find another query \(Q'\) that answers \(Q\) by using only the view \(V\) instead of the base tables \(X\); in other words, \(Q(X) = Q'(V(X))\) [22, 19]. The problem is usually solved by applying rewrite rules to \(Q\), until it only uses the available views. The problem of finding \(H\) is an instance of query rewriting using views, and one possibility is to approach it using rewrite rules; for this purpose we used the rule engine egg [48], a state-of-the-art equality saturation system [47].

**Counterexample-based Synthesis** Rule-based synthesis explores only correct rewritings \(P_2\), but its space is limited by the hand-written axioms. The alternative approach, pioneered in the programming language community [43], is to generate candidate programs \(P_2\) from a much larger space, then using an SMT solver to verify correctness. This technique, called Counterexample-Guided Inductive Synthesis, or CEGIS, can find rewritings \(P_2\) even in the presence of interpreted functions, because it exploits the semantics of the underlying domain.

**Rosette** We briefly review Rosette [44], the CEGIS system used in our optimizer. The input to Rosette consists of a specification and a grammar, and the goal is to synthesize a program defined by the grammar that satisfies the specification. The main loop is implemented with a pair of dueling SMT-solvers, the generator and the checker. In our setting, the inputs are the query \(P_1\), the database constraint \(\Gamma\) (including the loop invariant), and a small grammar \(\Sigma\), described below. The specification is \(\Gamma \models (P_1 = P_2)\), where \(P_2\) is defined by the grammar \(\Sigma\). The generator generates syntactically correct programs \(P_2\), and the verifier checks \(\Gamma \models (P_1 = P_2)\).
In the most naïve attempt, the generator could blindly generate candidates \( P_2, P'_2, P''_2, \ldots \), until one is accepted by the verifier; this is hopelessly inefficient. A first optimization in CEGIS is that the verifier returns a small counterexample database instance \( X \) for each unsuccessful candidate \( P_2 \), i.e., \( P_1(X) \neq P_2(X) \). When considering a new candidate \( P_2 \), the generator checks that \( P_1(X_i) = P_2(X_i) \) holds for all previous counterexamples \( X_1, X_2, \ldots \), by simply evaluating the queries \( P_1, P_2 \) on the small instance \( X_i \). This significantly reduces the search space of the generator. A second optimization is to use the SMT solver itself to generate the next candidate \( P_2 \), as follows. We assume that the grammar \( \Sigma \) is non-recursive, and associate a symbolic Boolean variable \( b_1, b_2, \ldots \) to each choice of the grammar. The grammar \( \Sigma \) can be viewed now as a Binary Decision Diagram, where each node is labeled by a choice variable \( b_j \), and each leaf by a completely specified program \( P_2 \). The search space of the generator is completely defined by the choice variables \( b_j \), and Rosette uses the SMT solver to generate values for these Boolean variables such that the corresponding program \( P_2 \) satisfies \( P_1(X_i) = P_2(X_i) \), for all previous counterexample instances \( X_i \). This significantly speeds up the choice of the next candidate \( P_2 \).

Using Rosette To use Rosette, we need to define the specification and the grammar. A first attempt is to simply provide the specification \( \Gamma = (G(F(X)) = H(G(X))) \) and supply the grammar of Datalog\textsuperscript{5}. This does not work, since Rosette uses the SMT solver to check the identity, and modern SMT solvers have limitations that require us to first normalize \( G(F(X)) \) and \( H(G(X)) \) before checking their equivalence. Even if we could modify Rosette to normalize \( H(G(X)) \) during verification, there is still no obvious way to incorporate normalization into the program generator driven by the SMT solver. Instead, we define a grammar for \( \text{normalize}(H(G(X))) \) rather than \( H \), and then specify:

\[
\Gamma \models \text{normalize}(G(F(X))) = \text{normalize}(H(G(X)))
\]

Then, we denormalize the result returned by Rosette, in order to extract \( H \). In summary, our CEGIS-approach for FGH-optimization can be visualized as follows:

\[
P_1 \xrightarrow{\text{normalize}} P'_1 \xrightarrow{\text{CEGIS}} P'_2 \xrightarrow{\text{denormalize}} P_2
\]

(25)

The choice of the grammar \( \Sigma \) is critical for the FGH-optimizer. If it is too restricted, then the optimizer will be limited too; if it is too general, then the optimizer will take a prohibitive amount of time to explore the entire space. We refer the reader to [47] for details on how we constructed the grammar \( \Sigma \).

4.4 Experimental results

We implemented an optimizer for Datalog\textsuperscript{7} programs. The input is a program \( \Pi_1 \), given by \( F, G \), and a database constraint \( \Gamma \), and the output is an optimized program \( H \). We evaluated it on three Datalog systems, and several programs from benchmarks proposed by prior research [41, 13]; in [47] we also propose new benchmarks that perform standard data analysis tasks. We did not modify any of the three Datalog engines. Our experiments aim to answer the question: How effective is our source-to-source optimization, given that each system already supports a range of optimizations?

Setup There is a great number of commercial and open-source Datalog engines in the wild, but only a few support aggregates in recursion. We chose 2 open source research systems, BigDatalog [41] and RecStep [13], and an unreleased commercial system \( X \) for our experiments. Both BigDatalog and RecStep are multi-core systems. The commercial system \( X \) is single core. As we shall discuss, \( X \) is the only one that supports all features for our benchmarks. In this paper we cover 3 out of the 7 benchmark programs used in [47]: Ex. 4.4 (BM), Ex. 4.1 (CC), and Single-source Shortest Path (SSSP) from [41]. The real-world datasets twitter, epinions, and wiki are from the popular SNAP collection [30].

Run Time Measurement For each program-dataset pair we measure the runtime of three programs: the original, with the FGH-optimization, and with the FGH-optimization and the generalized semi-naïve transformation (GSN). We report the speeds relative to the original program in Fig. 5. Where the original program timed out after 3 hours, we report the speedup against 3 hours. In some other cases the original program ran out of memory and we mark them with “o.o.m.” in the figure. All three systems already perform semi-naïve evaluation on the original program expressed over the Boolean semiring. But the FGH-optimized program is over a different semiring (except for BM), and GSN has non-stratifiable rules with negation, which are supported only by system \( X \); we report GSN only for system \( X \).

Findings. Figure 5 shows the results of the first group of benchmarks optimized by the rule-based synthesizer. The optimizer provides significant (up to 4 orders of magnitude) speedup across systems and datasets. In a few cases, for BM and CC on wiki under BigDatalog, and SSSP on wiki under \( X \), the optimization has little effect. This is due to the small size of the wiki dataset: both the optimized and unoptimized programs finish instantly, so the run time is dominated by optimization overhead. We also note that (under \( X \)) GSN speeds up SSSP but slows down CC (note the log scale). The latter occurs because the \( \Delta \)-relations for CC are very large, and as a result the semi-naïve evaluation has the same complexity as the naïve evaluation; but the semi-naïve program is more complex and incurs a constant slowdown. GSN has no effect on BM because the program is in the
5. CONCLUSIONS

We presented Datalog\(^5\), a recursive language which combines the elegant syntax and semantics of Datalog, with the power of semiring abstraction. This combination allows Datalog\(^5\) to express iterative computations prevalent in modern data analytics, yet retain the declarative least fixpoint semantics of Datalog. We also presented a novel optimization rule called the FGH-rule, and techniques for optimizing Datalog\(^5\) by program synthesis using the FGH-rule. Experimental results were presented to validate the theory. There are interesting open problems relating to Datalog\(^5\) convergence properties and its optimization; we refer to [26, 47] for details.

6. REFERENCES


[41] **Shkapsky, A., Yang, M., Interlandi, M., Chiu, H., Condie, T., and Zaniolo, C.** Big data analytics with Datalog queries on spark. In *SIGMOD* (2016).


Management of Implicit Requirements Data in Large SRS Documents: Taxonomy and Techniques

Dev Dave¹, Angelica Celestino¹, Aparna S. Varde¹,², Vaibhav Anu¹
1. Department of Computer Science, Montclair State University, NJ, USA
2. Visiting Researcher, Max Planck Institute for Informatics, Saarbrücken, Germany
(daved2 | celestinoa1 | vardea | anuv)@montclair.edu

ABSTRACT
Implicit Requirements (IMR) identification is part of the Requirements Engineering (RE) phase in Software Engineering during which data is gathered to create SRS (Software Requirements Specifications) documents. As opposed to explicit requirements clearly stated, IMRs constitute subtle data and need to be inferred. Research has shown that IMRs are crucial to the success of software development. Many software systems can encounter failures due to lack of IMR data management. SRS documents are large, often hundreds of pages, due to which manually identifying IMRs by human software engineers is not feasible. Moreover, such data is ever-growing due to the expansion of software systems. It is thus important to address the crucial issue of IMR data management. This article presents a survey on IMRs in SRS documents with the definition and overview of IMR data, detailed taxonomy of IMRs with explanation and examples, practices in managing IMR data, and tools for IMR identification. In addition to reviewing classical and state-of-the-art approaches, we highlight trends and challenges and point out open issues for future research. This survey article is interesting based on data quality, hidden information retrieval, veracity and salience, and knowledge discovery from large textual documents with complex heterogeneous data.

1. INTRODUCTION
The importance of high data quality in requirements engineering (RE) has long been recognized and well documented. Requirements elicitation is a critical RE activity that entails gathering data on system requirements from a multitude of sources including stakeholders, the existing system and its documentation, and various problem owners. Requirements explicitly expressed by the sources get recorded in requirements artifacts that are typically large Software Requirements Specifications (SRS) documents that could range to hundreds of pages with thousands of requirements. It is widely accepted that the industry is still striving to establish and apply practices to help identify crucial data on requirements that are often hidden or incomplete. Such requirements, also referred to as “Implicit requirements (IMRs)”, are a known root cause of software project failure [1], [2], [3], [4]. Since SRS documents often run into 100s of pages, it is not feasible for human software engineers to detect IMR data manually. In the world of big data, SRS documents are growing as per the Vs of volume, velocity, variety, etc. For example, the volume of SRS documents can range to Terabytes, their velocity can be of the order of a huge SRS document generated per week, and their variety entails heterogeneous data including plain text, structured text, images, tables and other infographics. This makes it even more significant to dwell upon the issue of IMR management, clearly a non-trivial issue. IMR data deserves attention with respect to its classification as well as techniques, practices, and tools.

Hidden requirements data with detrimental impacts on data quality and software development was recently highlighted in seminal research, NaPiRE (Naming the Pain in Requirements Engineering) [1]. This is a family of surveys conducted by leading RE researchers to understand the state of the practice in RE data and the most critical RE problems. The NaPiRe survey, which obtained data from RE practitioners from across 10 countries and 228 companies, highlighted hidden requirements data, i.e., IMR-based data, as the most frequently cited RE problem (48% of survey respondents cited IMR data as a problem frequently affecting their project results). The NaPiRe survey also provided a probabilistic cause-effect model with major reasons on requirements data being hidden or implicit during RE. Practitioners noted that hidden requirements are caused by team members' lack of experience and knowledge about elicitation practices, inadequate use of data management techniques such as completeness checking, and not using available RE tools. Given that hidden requirements data (aka IMRs) are recognized as an important RE problem, this area calls for further study. Classical work by Spanoudakis [5] focuses on analogical reuse of SRS data, propounding a paradigm to compute similarities between SRS documents. Though modeling such analogies is a useful aspect of SRS documents, this study does not focus on the actual identification or management of IMRs.

Research on practices, tools, and techniques available for IMR data identification and management has been sporadic, being performed by a select few, e.g., [6], [7]. A tool SRElicitor [6] has been developed as a prototype
in the form of an Eclipse plugin, usable in RE to convert SRS data to plain text and then to the required format via adaptation of a Semantic of Business Vocabulary and Rules (SBVR). Other tools, NAI and ARUgen, surveyed in [7] have been developed for ambiguity resolution, helpful in addressing IMR data. A tool COTIR has been proposed in [8] and enhanced in [9] to integrate commonsense knowledge with textual data mining and ontology for early detection of IMR. While fundamental COTIR deals with plain textual data in SRS documents, its enhanced version aims to address finding IMR from images and tables via approaches such as CNN (convolutional neural networks). Likewise, there are interesting advancements that deserve attention in IMR identification.

The data science community on a broad scale has addressed the importance of data on implicit requirements and related terms such as hidden / ambiguous/vague requirements through works such as [10], [9], [11], [12]. These works bridge various perspectives, e.g., data quality, business management, machine learning, commonsense knowledge, ontology, fuzzy logic, etc. Such research has often been published in multidisciplinary venues. However, data scientists have not dwelt substantially on IMR data management. Yet, this is a significant paradigm in the world of big data today where huge complex SRS documents are generated regularly and need to be effectively harnessed for high-quality information. Big data on IMRs is ever-growing and mandates considerable attention. Moreover, IMR data management relates broadly to topics such as data cleaning [13] due to connection with missing information; data veracity [14] due to emphasis on authenticity and salience of information [15]; hidden data discovery [16], [17] due to hidden requirements, complex text data analytics [18] due to heterogeneous textual data involved, and machine learning for data science [19] due to automation of knowledge discovery from large documents, all of which are of much interest to the data science community. Hence, this inspires a detailed study in the area and motivates our survey article.

To that end, this article is our attempt at presenting the state-of-the-research and state-of-the-practice in IMR identification and management through a review and analysis of the relevant literature. We focus on delineating a taxonomy of IMR in several categories accompanied by suitable examples to clarify the concepts. Additionally, we aim to inform the RE community of the techniques, practices, and tools used in IMR identification and management while also discussing the open issues in the area with pointers to future research directions.

2. IMR OVERVIEW & TAXONOMY
This section provides a brief introduction to software requirements engineering and IMRs followed by an IMR classification system that we have proposed.

2.1 Background on SRS and IMR Data
The software development life cycle (SDLC) typically begins with gathering data on requirements, followed by the requirements analysis phase that involves a detailed study of the needs the software is supposed to address. Often, some requirements related data is not well documented, and the burden of visualizing such data is left on the developer. Such vague requirements often result in software solutions not addressing the clients’ needs completely. Most requirements not captured initially are discovered inadvertently in user acceptance stages [2]. Clients think that such requirements would be captured by software automatically. It is difficult to identify who is at fault for not capturing IMRs. To avoid such incidents, resulting in dissatisfied customers, it is important to focus on identifying IMR data.

Figure 1 adapted and redrawn from [21], gives a generic depiction of Requirements Engineering as a whole.
Once the client presents the software problem to the developers, requirements data is collected before embarking on analysis. Many requirements are indeed explicitly captured. However, customers have other expectations: system security, availability, usability, performance, hardly defined in SRS documents. By default, and without writing, users expect that the system will always be available, be essentially secure, and perform its intended tasks well, without necessarily stating that. IMRs often look too simple due to which the customer views them as obvious, or too complicated such that the customer cannot visualize them.

Software requirements fall into 2 broad categories: functional and non-functional. Functional requirements define the functions of a system or its components, while non-functional requirements specify criteria that can be used to judge its operation, rather than specific behavior. Non-functional requirements are introduced during development [22]. Functional requirements are specified before the development begins. Functional development constitutes the earliest stage in the SDLC. The distinction between functional and non-functional requirements informs how each is handled during elicitation, documentation, and validation [22]. Often, developers ignore the assessment of non-functional requirements, which if identified can aid early detection and mitigation of risks. Speech tag parts constitute highly informative parts of the non-functional requirements. Based on this general background, we proceed with a thorough classification of IMR data.

2.2 Classification of IMR Data

In Table 1, we present a taxonomy of implicit requirements (IMR) data. This taxonomy is the main contribution of our survey and was developed based on this literature study and our expertise in the area. We are proposing that there are 5 categories of requirement data that are frequently missed (i.e., remain implicit). As shown in Table 1, our main IMR categories are: Security, Accessibility, Maintainability, Sustainability, and Usability. We describe the five categories below.

Data on security requirements can be explicitly stated but can also imply additional requirements (which are not overtly stated by the clients). Riaz et al. propose to identify security requirements data. Security categories in their study are: identification and authentication, availability, accountability, and privacy [23].

Accessibility requirements constitute another category of data often missed. This is due to a lack of proper development methods and authoring tools [24]. Web Content Accessibility Guidelines (WCAG) 2.0 and ISO 9241-171:2008 guidance on software accessibility provides a wide range of recommendations to make the web more accessible for individuals with disabilities.

Fundamental categories in accessibility guidelines are: perceivable, operable, understandable and, robust [24].

Data on maintainability requirements is important during software design and implementation. Improving software processes can therefore enhance software maintainability [25]. Problems during development can lead to lower standards of maintainability. The ISO/IEC (International Organization for Standardization and International Electrotechnical Commission) 9126-1 2001 standard defines maintainability as the capability of a software to be modified. Software maintainability has 4 main categories: analyzability, changeability, stability, and traceability [25].

Sustainability requirements data is generally not supported by traditional software engineering methods. Yet it is very important, especially today with much focus on developing sustainable and environment-friendly systems. Sustainable development is defined as meeting the needs of the present without compromising the ability of future generations to meet their own needs [26]. Sustainability is mentioned as per 3 variables: time, function, and system. Time refers to the actual amount of time required to maintain or develop software; function refers to satisfying the task-based requirements, and system refers to humanity in its ecosystem. Note that the system usage aspect can be further divided into 5 sustainability dimensions, namely: economic, technical, social, individual, and environmental. The 5 dimensions serve to highlight the various potential impacts of the system.

Usability data affects the design of software and should be considered during the requirements phase. Usability is an important requirement as it improves productivity and customer satisfaction while reducing training and documentation costs as mentioned in an interesting piece of research. This research aims to embody HCI (Human Computer Interaction) principles. The author defines Functional Usability Features (FUF) based on HCI to make recommendations that the software should provide to the user. Usability requirements are related to the software’s user interface and often remain hidden during RE. We found eight (8) sub-categories of usability requirements / features, namely: feedback features, undo features, cancel features, form or field validation, Wizard requirements, user expertise features, different languages and alert features.

Table 1 provides descriptions of the various categories in IMR taxonomy with their sub-categories, definitions and examples of software requirements. Hence, we have presented a detailed taxonomy of IMR data as widely accepted in the literature. We now proceed with tools and techniques used in the literature to address IMRs.
### Table 1. Taxonomy of IMR Data: Categories, Definitions, and Examples

<table>
<thead>
<tr>
<th>Category</th>
<th>Sub-category</th>
<th>Definition</th>
<th>Example of Software Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Identification and</td>
<td>Claimed identity of user must be valid for the user, process, or device</td>
<td>The system shall authenticate the user before any access to Protected Resources (e.g., PHI) is allowed, including when not connected to a network e.g., mobile devices.</td>
</tr>
<tr>
<td></td>
<td>Authentication</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Availability</td>
<td>The system or component must be available to a certain degree</td>
<td>The system shall provide business continuity in the situation where the Electronic Health Record (EHR) system is not available by providing access to the last available clinically relevant patient data in the EHR.</td>
</tr>
<tr>
<td></td>
<td>Accountability</td>
<td>Any action taken that affects the system can be traced back to the user responsible for the action</td>
<td>System shall keep track of every entry in the health record. Each entry will be identified with the author and should not be made/signed by someone other than the author.</td>
</tr>
<tr>
<td></td>
<td>Privacy</td>
<td>The user can understand and control how their information is used in the system</td>
<td>The system shall allow nurses to provide legitimate care in crisis situations that may go against prior patient consent directives (&quot;break the glass&quot; situations).</td>
</tr>
<tr>
<td>Accessibility</td>
<td>Perceivable</td>
<td>Information and UI components must be presentable to users in ways they can easily discern the information</td>
<td>The system shall provide text alternatives for any non-text content so it can be transformed into other forms people need, such as large print, speech, symbols, or simpler language.</td>
</tr>
<tr>
<td></td>
<td>Operable</td>
<td>User interface components and navigation must be usable</td>
<td>All functionality that is available by keyboard.</td>
</tr>
<tr>
<td></td>
<td>Understandable</td>
<td>Information and the operation of the user interface must be comprehensible</td>
<td>Significant changes on a web page do not happen without the consent of the user.</td>
</tr>
<tr>
<td></td>
<td>Robust</td>
<td>Content must be adaptable enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies</td>
<td>The system shall rely interpret markup by ensuring it is valid.</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Analyzability</td>
<td>The software system is easy to diagnose or examine</td>
<td>All states, including fault conditions, are known.</td>
</tr>
<tr>
<td></td>
<td>Changeability</td>
<td>The software system is easy to change or modify</td>
<td>The system shall be easily modifiable so it can be compatible with new hardware.</td>
</tr>
<tr>
<td></td>
<td>Stability</td>
<td>The software system performs as expected and does not have any unexpected effects</td>
<td>During a ‘sale/festival’ season many people purchase commodities from web portals. The performance of a website must satisfy user expectations.</td>
</tr>
<tr>
<td></td>
<td>Traceability</td>
<td>The ability to describe and follow the life of a requirement</td>
<td>The system shall have a functional audit trail by ensuring that a record is kept of all the changes made to the system.</td>
</tr>
<tr>
<td>Sustainability</td>
<td>Software Evolution Aspect</td>
<td>Sustainability of a software during its upkeep period by continuous monitoring of quality and knowledge management until it is replaced by a new system</td>
<td>The system shall be able to show what equipment is available, where it is located and in which state. This will avoid buying superfluous equipment and maximize the expected lifetime of equipment by doing maintenance when needed.</td>
</tr>
<tr>
<td></td>
<td>System Production Aspect</td>
<td>Sustainability of a software system as product with respect to its use of resources for production is achieved. This can be achieved by using green IT principles</td>
<td>The system shall enable the Event Manager to select Event Parts that require low energy and emit low CO2 amounts.</td>
</tr>
<tr>
<td></td>
<td>System Usage Aspect</td>
<td>Sustainability of a software system in the usage process takes into account responsibility for the environmental impact and designing green business processes.</td>
<td>The system shall support the quality manager in efficiently assessing the sustainability of an event, in order to enhance firm’s practices.</td>
</tr>
<tr>
<td>Usability</td>
<td>Feedback</td>
<td>The software informs the user about what is happening to the system</td>
<td>The system should always keep users informed about what is going on through appropriate feedback within reasonable time</td>
</tr>
<tr>
<td></td>
<td>Undo</td>
<td>The software allows the user to Undo an action at several levels</td>
<td>If the user cancels, the system will go back to the first window listing what theatres there are.</td>
</tr>
<tr>
<td></td>
<td>Cancel</td>
<td>The software allows the user to Cancel the execution of a command or an application</td>
<td>The user will be given the chance to cancel the operation, and the system will again display the selected theatre show times.</td>
</tr>
<tr>
<td></td>
<td>Form/Field Validation</td>
<td>Improve data input for users and software correction as soon as possible</td>
<td>Form validation should be consistent and non-obtrusive in styling, location, and tone relevant to its</td>
</tr>
<tr>
<td></td>
<td>Wizard</td>
<td>Assist users with tasks that require various steps and user input</td>
<td>An installer will be used to unpack the required libraries for the program via an install wizard.</td>
</tr>
<tr>
<td></td>
<td>User Expertise</td>
<td>Allows adapting system functionality to users’ expertise</td>
<td>The application enables users to select tutorials based on their level of expertise.</td>
</tr>
<tr>
<td></td>
<td>Different Languages</td>
<td>Allows users to work with their language, currency, ZIP code etc.</td>
<td>The system requires an Internet connection and uses Google Translate to perform the text translation</td>
</tr>
<tr>
<td></td>
<td>Alert</td>
<td>Warns users of actions with important consequences</td>
<td>At the end of the booking process, the system will display a window reporting whether the operation was a success or failure</td>
</tr>
</tbody>
</table>
3. IMR TECHNIQUES AND TOOLS

This section provides a comprehensive overview of existing IMR identification techniques and tools. Our literature selection criteria was: “tools and techniques used for IMR detection and classification”.

3.1 Ontology, Semantics, and Pragmatics

Technological performances on management of IMR data are evaluated by various metrics and approaches. Various frameworks are used in detecting and managing IMR data, including case-based reasoning, ontology-based reasoning, and analogy-based reasoning, [2]. Some of these are illustrated in Figure 2 pertaining to ontology (redrawn based on [27]), and in Figure 3 about analogy (reconstructed from [28]). A reuse-based implicit requirements model (RM) is vital in facilitating the reuse of IMR data across projects when substantial similarities can be established between existing documented requirements and new ones [2].

Semantic matching is used to check for the similarity between requirements [2]. This deploys ontology to improve syntactic matches by exploring the relationship between semantics. Detecting IMR data is a collective responsibility of various team members. Analyzing risks of defects can save time and resources. Things that seem common sense can hurt system functionality, e.g., ignoring the fact that users can delete their accounts. Giving users control to delete their accounts can hurt an organization, especially by users parting with the company acrimoniously. Various tests require different tools and methods. Among the IMR data tested, an interesting method is the Implicit Priming Test (IPT) designed to determine connections between attributes and objects. The IPT further attempts to detect the strength of such connections. The speed of a response to an external stimulus is determined while factoring in the effects of priming [2]. By design, the IPT ignores explicit responses but returns values for implicit connections, e.g., emotional connection of a product itself and customers’ thoughts about the given product.

The pragmatism of quality assurance (QA) teams should span beyond the benefits of the team to the entire company [29]. Challenging defects, regardless of their priorities, results in a decrease of unresolved issues. Technical knowledge is a critical requirement in analyzing and detecting IMRs, especially in complex systems. Various aspects of software system data might not be documented, mandating deep technical knowledge for testing. This includes testing for data integrity, application properties, flooding and draining queues, circuit breakers, and deadlocks. The deeper is the QA knowledge in a given domain, the more effective is the IMR data management. Automating and incorporating functional testing in the SDLC brings QA and development teams together, freeing more time for exploratory tests [22]. QA teams working closely with developers help in fault-finding and fixing issues early.

3.2 IMR Architectural Framework

Researchers address IMR data identification in different ways. While some prefer an unorthodox means of obtaining real-life views from the user perspective,
others opt for utilizing theoretical and conceptual frameworks [30]. An IMR architectural framework is shown in Figure 4 [31]. This provides support for managing IMR data as follows: (1) IMRs already documented, can be reused in new projects; (2) New IMRs previously overlooked can be discovered and stored thereby mitigating additional costs; (3) IMR data can be ranked based on established organizational standards, assigning the right priority level and scope to specific requirements.

This framework uses analogy-based reasoning for the reuse of previous requirements specifications, ontology to represent relevant domain knowledge crucial for managing IMRs, and natural language processing (NLP) to facilitate the analysis of textual data. This research aims to evolve a process framework for managing IMR within an organization. However, the system lacks human reasoning. Likewise, the ranking deployed also presents the scope for further research, interesting in data science. Ranking is a problem that is of interest to data scientists. Conducting ranking in this context would help to prioritize the requirements, so as to guide the software development. Including human reasoning in this process (potentially through machine learning), would help simulate the manner in which human software engineers can identify IMRs. This can be beneficial in software development processes.

3.3 The PROMIRAR Tool

PROMIRAR is a tool that facilitates the reuse of previously documented specifications to establish new requirements via analogy-based reasoning [27]. To identify the basis for analogy, understand similarities, and discover IMRs, NLP is used to analyze and extract important information, as shown in Figure 5 [27].

PROMIRAR takes preprocessed SRS documents as input data, uses NLP that empowers its feature extractor, and has an ontology library for knowledge representation of domain ontologies (specific purposes / general business rules). The Java Protege 4.1 ontology API is used to build the ontology library. Its feature extractor provides essential rules for classifying possible sources of IMR data in SRS documents. It has a heuristic classifier responsible for classifying the actual requirements based on intermediate outputs. Its analogy-based reasoner comprises 3 types of knowledge: domain, solution, and goal. Steps to use the PROMIRAR tool based on its architecture are: preprocess, import, analyze, identify and manage.

This tool addresses IMR data management by adapting analogy-based reasoning and provides good results. Yet, there is the potential for augmentation via research on advanced data management techniques in areas combining NLP, image mining, and data extraction from infographics, embodying domain knowledge and software engineering concepts. Such research can help improve IMR data management because many SRS documents contain complex data such as images, tables and other infographics. Extracting valuable information from these is a non-trivial process. While human software engineers can do this on a small scale, it is hard to achieve for huge volumes of complex data in SRS documents. Hence, discovering knowledge on IMRs from these documents in a seamless manner using data science approaches mentioned here would contribute to the RE phase, thus enhancing software development.

Figure 5. The PROMIRAR Tool
3.4 Using Templates for IMR Data Detection

An interesting practice used for detecting IMR data is the concept of “templates” to elicit implied security requirements [23]. The goal of this work is to determine whether automatically suggested security requirements templates help in efficient and effective requirements elicitation compared to a manual approach based on personal expertise. This tool aids the visualization of IMR data by providing a template as a checklist for developers to reference.

Dealing specifically with security, the process takes requirements-related artifacts as inputs. These include requirement specifications, feature requests, use case scenarios, etc. Based on these, it generates security requirements. This tool is a good visual aid and can be useful in conjunction with other IMR-related research, especially dealing with data security and privacy issues. It can propel further research in these areas and can be subjected to enhancement based on research outcomes.

3.5 InfoVis: IMR Data Visualization

A tool-supported approach is proposed to identify IMR data based on ambiguity and incompleteness [4]. This uses NLP techniques combined with visualization techniques to extract and interpret IMR data. It involves taking in user story requirements processed by a novel algorithm that deploys the Semantic Folding Theory (SFT) to calculate the semantic distance between 2 words to produce a similarity score. The algorithm also calculates the ambiguity score computed as a linear combination of term and context similarity. In this research, a visualization approach called InfoVis is developed that enables analysts to explore multiple viewpoints and extract IMR data. It harnesses Venn diagrams to simplify highlighting IMR data.

In their evaluation, term pairs in each category (low ambiguity, medium ambiguity, high ambiguity scores) are used from the WebCompany dataset with 98 user story requirements [4]. Students are presented with selected term pairs and asked to rate ambiguity. Results indicate a strong correlation between the score calculated by the algorithm and that given by students, the latter being the source of ground truth. Hence, this tool serves as an effective means for managing IMR data. This can serve as a benchmark for comparison further studies on IMR data management.

3.6 Machine Learning for IMR data

Binkhonain and Zhao [32] present a review of various machine learning approaches to identify and classify non-functional requirements. They present an overview of 16 machine learning approaches that utilize 5 Supervised, 5 Semi-Supervised, and 4 Unsupervised machine learning algorithms as shown in Table 2.

<table>
<thead>
<tr>
<th>Supervised</th>
<th>Semi-Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Support Vector Machines (SVMs)</td>
<td>• Expectation-Maximization (EM)</td>
<td>• Latent Dirichlet Allocation (LDA)</td>
</tr>
<tr>
<td>• Naïve Bayes (NB)</td>
<td>• Self-training</td>
<td>• K-means</td>
</tr>
<tr>
<td>• Decision Tree (DT)</td>
<td>• Active learning</td>
<td>• Hierarchical</td>
</tr>
<tr>
<td>• K-Nearest Neighbors (K-NN)</td>
<td>• Random Subspace Method for Co-training(RAS-CO)</td>
<td>Agglomerative</td>
</tr>
<tr>
<td>• Multinomial Naïve Bayes (MNB)</td>
<td>• Relevant Random Subspace Method for Co-training (Rel-RASCO)</td>
<td>• Biterm Topic Modelling (BTM)</td>
</tr>
</tbody>
</table>

All the machine learning approaches followed the same general process consisting of text preparation, learning, and evaluation. The text preparation phase consisted of text preparation and feature selection [32]. Text preparation involved NLP techniques such as Stemming, Stop word removal, tokenization, etc. For feature selection, the text document is converted into a numeric matrix using methods such as Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). The important features are selected from the matrix using methods such as information gain and Chi-square [32]. The learning phase consists of training and testing the machine learning algorithms on the preprocessed text dataset. The evaluation phase consisted of measuring the performance of the machine learning algorithms by using various metrics such as precision, recall, F-Score [32]. The key findings of this research were that ML-approaches generally perform well and achieve an accuracy of more than 70% when identifying and classifying NFRS, Supervised algorithms perform better than Semi-Supervised and Unsupervised algorithms with SVM and NB having the best overall performance, ML algorithms produce better results when individual words are used rather than phrases and without text preprocessing such as stemming and lemmatization [32]. Some of the challenges in this research area highlighted by Binkhonain and Zhao are the lack of shared datasets to train machine learning algorithms, lack of a standard definition of NFRs, and feature identification and selection [32].

3.7 Other Approaches in IMR Research

An interesting approach in IMR data management is COTIR, i.e., Common Sense Knowledge, Ontology, and Text Mining for Implicit Requirements. It consists of 6 steps: (1) Preprocess source documents to get requirements into text file format devoid of graphics,
images, and tables; (2) Select existing CSKB (Common Sense Knowledge Base) to be used for identification of IMR data; (3) Import SRS documents and domain ontology into the COTIR environment; (4) Click on the “analyze” function to allow the feature extractor to identify potential sources of IMR data in SRS; (5) See the potential IMRs identified and their recommended possible explicit requirements; (6) Seek expert opinion on IMR data, experts could approve/disapprove recommendations by adding/removing using the tool.

Emebo et al. envisage an enhancement of COTIR where a convolutional autoencoder can be used. This enables detecting IMRs in complex data (as opposed to plain text only in fundamental COTIR), e.g., deciphering text within images [9]. Facets of the enhancement are as follows: (1) SRS documents supply requirements data from which IMRs need to be identified. Data cleaning removes noise in the RE data. This step is performed in its NLP component. (2) The requirements author selects relevant knowledge from the CSKB and relevant domain ontology from the ontology library. Previously cleaned RE data with the selected knowledge and domain ontology are transferred to the CNN-based autoencoder component. (3) The autoencoder's input construction transforms the data into vectors for deep learning models. In an autoencoder, parameters are trained by minimizing differences between input and output layers in an unsupervised manner. (4) The trained model is applied to solve new IMR problems. This model is capable of encoding the word frequency vector of a new IMR feature into the hidden states. In this enhanced version, COTIR has fewer steps than its predecessor and takes in more input, while catering to more complex data.

Further, Emebo et al. revisit COTIR via a demo paper. They focus on explaining how commonsense concepts, ontological aspects, and mining of textual data help identify areas of explicit requirements where relevant IMRs may be hiding [33] therefore, making it an IMR-source localization tool. As per their claims, this is the first tool embodying commonsense knowledge for the detection of IMR data by aiming to simulate human reasoning. In the demo, the researchers use a course management system (CMS) example where the SRS document is based on explicit requirements available for the CMS. Possible IMR sources are analyzed by the feature extractor, with suitable recommendations being prompted for the CMS. The demo concludes that the COTIR tool reduces software defects by around 10% and enhances overall software quality by around 20% on an average [33].

Other approaches exist in the overall paradigm of IMR data management [34] that could refer to IMRs by different names such as hidden, vague, missing, ambiguous, incomplete, derived, or assumed requirements instead of the specific term implicit requirements used herewith. Regardless of terminology and nomenclature, the detection and management of IMR data is of the utmost importance for good software

<table>
<thead>
<tr>
<th>Techniques/Tools</th>
<th>Taxonomy</th>
<th>Data Model</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implicit Priming Test (IPT) [2]</td>
<td>General</td>
<td>Ontology, Semantics and Pragmatics</td>
<td>• Analyzing risks of defects &lt;br&gt; • Determining connections between attributes and objects</td>
<td>• Relies heavily on retrieving previous IMR classifications results to classify future IMRs</td>
</tr>
<tr>
<td>IMR Architectural Framework [31]</td>
<td>General</td>
<td>Analogy-based reasoning</td>
<td>• Reuse of previous requirements, ontology and domain knowledge and NLP for textual analysis</td>
<td>• The system lacks human reasoning, new IMRs may be overlooked</td>
</tr>
<tr>
<td>The PROMIRAR Tool [27]</td>
<td>General</td>
<td>Analogy-based reasoning</td>
<td>• Feature extractor using NLP &lt;br&gt; • Heuristic classifier to classify IMRs</td>
<td>• Can be improved by augmentation using techniques in NLP, image mining and data extraction</td>
</tr>
<tr>
<td>Using Templates for IMR Data Detection [23]</td>
<td>Security</td>
<td>Templates</td>
<td>• Aids visualization of IMR by providing a template as a checklist for developers</td>
<td>• Good visual aid but works better when used with other IMR tools</td>
</tr>
<tr>
<td>InfoVis: IMR Data Visualization &amp; NLP [4]</td>
<td>Maintainability – Defect Detection</td>
<td>NLP combined with semantic similarity techniques</td>
<td>• Enables analysts to explore multiple viewpoints and extract IMR Data &lt;br&gt; • Uses Venn diagrams to simplify highlighting IMR Data</td>
<td>• Effectiveness of tool needs to be tested at a larger scale &lt;br&gt; • Algorithm for detecting ambiguity can be improved and tuned while avoiding over-fitting</td>
</tr>
<tr>
<td>Machine Learning Classification techniques [32]</td>
<td>General</td>
<td>Supervised ML &lt;br&gt; Semi-Supervised ML &lt;br&gt; Unsupervised ML</td>
<td>• Average accuracy of above 70% when identifying and classifying NFRs &lt;br&gt; • Supervised ML algorithms, specifically, SVM and NB perform the best on average</td>
<td>• Lack of shared datasets to train ML algorithms &lt;br&gt; • Lack of standard definition for NFRs &lt;br&gt; • Feature identification and selection</td>
</tr>
</tbody>
</table>
development with user satisfaction and is a good practice that should be emphasized in data science, software engineering, and related areas.

3.8 Comparison of IMR Tools, Techniques

Table 3 above presents a comparison of the various IMR tools and techniques discussed in the previous sections. The table highlights the data model used for each tool and technique along with advantages and disadvantages.

4. TRENDS IN IMR MANAGEMENT

Existing gaps in IMR data management are likely to propel further research trends in this area. Some works such as [12] observe gaps in the literature regarding the specification of data-flow requirements. IMR data can be represented as either structured, semi-structured, or unstructured data as requirements can be encoded without a standard format having text, figures, and infographics [35]. Future research in IMR management can focus on developing intelligent, interactive, user-friendly tools to identify, analyze, and specify IMRs. It can also focus on more automation in the RE phase, and on improving data verification to ensure identification of vital system features.

Trends in IMR management include several research issues, among which we can potentially address the following avenues that would be of interest to data scientists:

- Relationships between data on defects in the RE phase and actual sources errors causing those defects are often ignored. Tracing precise causes of defects can result in SRS documents with enhanced data quality which would lead to better IMR data management. This can possibly be explored with data science techniques such as association rule mining. It would help to find relationships of the type “X implies Y”, where X can be the actual source error causing defect Y in the RE phase, hence rules of the type “Error implies Defect” can be discovered. Such knowledge discovery can help fix the root cause of problems leading to IMR related issues in SRS documents. Hence, this is a justifiable piece of research since it would augment the RE phase in software engineering, leading to better outcomes in software development as a whole.

- Thorough studies can be conducted with tools such as COTIR, NAI, SR-Elicitor, and ARUgen, used in contexts related to IMRs [33] addressing efficiency, accuracy, complexity etc. with respect to usefulness in facets such as ontology, knowledge bases, and other relevant data science concepts. This can be justified as follows. Considering ontology, it would be interesting to explore whether standards such as RDF (Resource Description Format) and OWL (Web Ontology Language) can be useful in IMR specification tasks, since that would create a streamlined manner of data exchange usable globally by software engineers and data scientists across industry and academia. Currently, there are no such ontological standards. Nor are there such existing studies relating IMR with data science paradigms. Likewise, creating knowledge bases, e.g. domain-specific KBs (finance, law, healthcare) with respect to IMR data can guide the RE phase of software development processes. Such research would help to use data science concepts within software development to resolve issues on IMRs.

- IMR practices should be integrated into education and training for students and working professionals, e.g., in courses such as Human-Computer Interaction, Database Management Systems, Machine Learning, and special topics courses in areas such as Data Quality, and Requirements Engineering. We should strive towards making this commonplace in academia and industry, as potentially paradigm-shifting best practices [36], [37]. The justification for this entails the constant growth of education and training to keep abreast with the latest technology. This is required by academic boards such as ABET (Accreditation Board for Engineering and Technology) and CAC (Computing Accreditation Commission). Such growth and advancement is also preferred by the corporate world when they hire fresh graduates, e.g. as software engineers, data scientists, full-stack developers. Hence, embedding IMR research within relevant courses in the realms of data science and software engineering is beneficial to education.

Some further research in the avenues mentioned here could be orthogonal to efforts of interest to the data science community. A few appealing works in line with such insights include: [38] that deals with data quality monitoring for constantly evolving big data focusing particularly on data veracity (in addition to volume, variety, etc.); [39] that addresses ontology compliance for query processing with enrichment; [16], [17] that entail hidden information discovery; [40] that intends to conduct query optimization with robustness and reduced complexity with emphasis on real-life workloads; [13] that encompass various perspectives of data cleaning; [14], [15] that address topics such as veracity and salience of information; [19] that addresses machine learning within database management, and [41] that propels database education in conjunction with natural language aiming to make this widespread.

Other interesting works include those in commonsense knowledge (CSK), e.g. [42] that presents a short survey on the usefulness of CSK, incorporating its derivation, knowledge base construction, as well as benefits in data
management and machine learning; and [43] that presents a tutorial on CSK extraction, compilation, and evaluation, explaining where CSK is significant and how it can be supplementary/complementary to deep learning. Since we aim to incorporate both deep learning and CSK in some of our future work on IMR data management, many studies surveyed in these articles [42], [43] provide useful references.

There is the potential for future research in IMR data management and related avenues that have heretofore remained areas of less substantial focus. We have outlined trends as envisaged by us upon a survey of related topics. Our work on enhanced automation in early IMR detection with usage in suitable applications [33], as well as further investigation with active practice in studies and training [36], [37] would contribute the two cents to the paradigm of IMR data management. This would help augment R&D and education in data science and software engineering.

5. CONCLUSION
Summary: Missed user requirements, also known as implicit requirements (IMRs), are a major contributor to software project failures. These can have other nomenclature such as hidden / vague / ambiguous / derived requirements. In this survey article, we have presented a background of IMR data along with a detailed taxonomy of terms, definitions, and examples associated with this paradigm, mainly encompassing the categories of security, accessibility, maintainability, sustainability, and usability.

IMR data is primarily available for software development projects that use the Waterfall model as SRS documents are not created for projects using the Agile model. Existing research has not focused on Agile requirements engineering and more specifically, on identifying hidden requirements from user stories. Even though many software development projects use the Agile model, not much research has been done on identifying hidden requirements. While this can be an issue of concern and calls for further research, we have not focused on that in this study. Our survey in this paper caters much more to the Waterfall model, and the IMR data management there.

We have provided an insight into classical as well as state-of-the-art tools, techniques, and practices in the overall context of IMR data management. We have outlined projects such as PROMIRAR, InfoVis, and COTIR. Suitable explanations with illustrations and discussions have been provided. The works surveyed in this article indicate that there is much emphasis on IMR data with its importance being realized in successful software development from user standpoints, yet there is scope for further research. We have listed open issues for future work, in our survey of the respective tools as well as in our section on trends in IMR management. These would contribute further to enhancing IMR detection and improving software development.

In short, this survey article provides novel insights for the data science community with respect to the challenging and non-trivial issue of IMR data management from huge SRS documents. It is interesting from the angles of data quality, hidden information retrieval, knowledge discovery, textual heterogeneous data, ontology, and semantics, all of which are avenues of interest to data scientists.

Roadmap: In our own future work, we aim to overcome a few limitations of the current survey and address some further challenges. First, we intend to extend this study by performing a more systematic literature review to help address more research studies (possibly not outlined in the current survey) that focus on various facets of IMR data.

Second, we intend to investigate in detail specific works of the literature in areas such as data quality, veracity, and hidden information retrieval for IMR management to outline specific research sub-problems that would be of joint interest to the data science and software engineering communities. These would provide the potential for MS Theses and Ph.D. Dissertations in the common areas, along with the scope for implementing useful software tools that cater to the interests of both communities, as well as publications in both venues.

Third, we aim to further automate the early detection of IMR data by leveraging deep learning techniques with the potential use of commonsense knowledge to localize the source of IMRs and assist their management. An important goal of our work here is to develop a large-scale tool for the automation of early IMR identification. We envisage that such a tool will be useful to software developers and will also help to train students and practitioners on adequate IMR detection in the requirements engineering phase.

Finally, we envisage making early IMR detection a common practice via dissemination and usage of our research and development efforts as well as active deployment of various IMR data management tools in real-world applications. We would reach out to our collaborators in academia to include IMR-related concepts and practices in their course syllabus, and to our industry collaborators to help spread the awareness of IMR data management in software developmental efforts. We anticipate that such a practice will enhance work in data science and software engineering.
6. ACKNOWLEDGMENTS
Dev Dave has been funded by a Graduate Assistantship in Computer Science at Montclair State University. Aparna Varde acknowledges the NSF grants MRI: Acquisition of a High-Performance GPU Cluster for Research and Education Award # 2018575, and MRI: Acquisition of a Multimodal Collaborative Robot System (MCROS) to Support Cross-Disciplinary Human-Centered Research and Education at Montclair State University, Award # 2117308. She is a visiting researcher at Max Planck Institute for Informatics, Saarbrücken, Germany in the research group of Gerhard Weikum, during her sabbatical. Vaibhav Anu hereby acknowledges the NSF grant, REU Site: Enhancing Undergraduate Research Experiences in Cyber-security and Privacy-Enhanced Technologies (NSF Award # 2050548).

7. REFERENCES


S. Rzazewski, N. Tandon, and A. S. Varde. Information to wisdom: Commonsense knowledge extraction and compilation. In ACM WSDM Intl. Conf. on Web Search and Data Mining, 2021.
ABSTRACT

The recent success of machine learning (ML) has led to an explosive growth of systems and applications built by an ever-growing community of system builders and data science (DS) practitioners. This quickly shifting panorama, however, is challenging for system builders and practitioners alike to follow. In this paper, we set out to capture this panorama through a wide-angle lens, performing the largest analysis of DS projects to date, focusing on questions that can advance our understanding of the field and determine investments. Specifically, we download and analyze (a) over 8M notebooks publicly available on GitHub and (b) over 2M enterprise ML pipelines developed within Microsoft. Our analysis includes coarse-grained statistical characterizations, fine-grained analysis of libraries and pipelines, and comparative studies across datasets and time. We report a large number of measurements for our readers to interpret and draw actionable conclusions on (a) what system builders should focus on to better serve practitioners and (b) what technologies should practitioners rely on.

1. INTRODUCTION

The ascent of machine learning (ML) to mainstream technology is in full swing: from academic curiosity in the 80s and 90s to core technology enabling large-scale Web applications in the 90s and 2000s to ubiquitous technology for the masses today. We expect that in the next decade, most applications will be “ML-infused” [2]. This massive commercial success and academic interest are powering an unprecedented amount of engineering and research efforts—in the last five years alone, we have seen over 75K papers in a leading public archive (https://bit.ly/3KZFla5) and millions of publicly shared data science (DS) notebooks corresponding to billions of dollars of development cost (per COCOMO model [5]).

As our team at Microsoft’s Gray Systems Lab began to invest heavily both in building systems to support DS and leveraging DS to build applications, we were faced with many open questions due to the speed of evolution of ML. As system builders, we were uncertain about what DS practitioners needed (e.g., are practitioners shifting to using only DNNs?). As DS practitioners, we were equally puzzled on which technologies to learn and build upon (e.g., shall we use TENSORFLOW [1, 34] or PYTORCH [24]?). Discussing with experts in the field led to rather inconsistent views, too.

We thus embarked on a (costly) fact-finding mission, consisting of large data collection and analysis, to better understand this shifting panorama. In particular, we analyzed (a) 8M notebooks, publicly shared on GitHub—representative of OSS, educational, and self-learning activities; and (b) 2M ML pipelines professionally authored in ML.NET within Microsoft. DS encompasses a wide range of operations (e.g., wrangling, visualization, ML, collaboration), and collecting datasets representative of them all is a herculean task. As we will see, the datasets we use here are a first step towards this end, and skewed towards ML, visualization, and data pre-processing.

Over the past few years, we have used the results of this analysis to educate several decisions across Microsoft in a data-driven fashion (§6). Given the significant internal impact, we realized that this knowledge could serve the community at large. Hence, we summarize our key results in this paper. To our knowledge, this is the largest analysis of DS projects made public.

The diversity and sheer size of these datasets enable multiple dimensions of analysis. In this paper, we focus on extracting insights from dimensions that are most urgent for the development of systems for DS and for practitioners to interpret adoption and support trends:

- **Landscape** (§3) provides a bird’s-eye view on the volume, shape, and authors of DS code.
- **Library analysis** (§4) provides a finer-grained view of this landscape. As such, it (a) sheds light on the functionality that data scientists rely on and systems for DS should focus on, and (b) informs prioritization of efforts based on the relative usage of libraries.
- **Pipeline analysis** (§5) provides an even finer-grained view by analyzing operators (e.g., learners, transformers) and the shape (e.g., #operators) of ML pipelines, both explicit (i.e., SKIT-Learn and ML.NET pipelines) and implicit ones (i.e., pipelines defined using one or more libraries).

In addition to reporting objective measures in isolation, throughout the paper we perform comparative analysis to better understand trends of usage. In particular, we compare: (a) statistics for Python notebooks, libraries, and DS pipelines across four years in GitHub datasets;
and (b) ML pipelines from GitHub with ML.NET ones. Furthermore, throughout the paper we draw actionable interpretations of our results to better inform practitioners and system builders.

Finally, to better highlight the importance of DS code analytics in practice, we conclude our discussion with scenarios from multiple teams within Microsoft and maintainers of DS libraries on how they have used the results of our analysis over the years (§6).

2. CORPORAS AND CODE ANALYTICS

For our analysis, we leverage all publicly available notebooks from GitHub and a large dataset of ML.NET pipelines from within Microsoft. These datasets cover a broad spectrum of users—from inexperienced to somewhat experienced to experts. Although the majority of the notebooks include ML operations, at least 34% of them involve data processing and visualization but no ML operations. Next, we briefly describe each data source and our purpose-built system for code analytics.

GitHub. In our analysis, we use three corpora of publicly available notebooks on GitHub spanning four years (2017 to 2020): GH17; GH19; and GH20, indicating the corresponding download year. Each consists of notebooks available at the HEAD of the default (e.g., main or master) branch of all public repositories at the time of download: 1.2M notebooks (0.3TB compressed) in July 2017 for GH17 [26], 4.6M notebooks (1.5TB compressed) in July 2019 for GH19, and 8.7M notebooks (3.2TB compressed) in July 2020 for GH20. Based on our analysis, this dataset is skewed towards inexperienced or somewhat-experienced users.

ML.NET. The underlying system, ML.NET [3], has been used in production for over a decade. We obtained access to a telemetry database from 2015 to 2019, containing over 88M events. While many users opted out of reporting telemetry, this sample is representative of ML activities within Microsoft, providing an enterprise-centric vantage point. This dataset is also representative of an ML community within Microsoft that supports mostly large-scale enterprise applications.

Code Analysis System. Analyzing DS code at the scale outlined above involves multiple challenges (e.g., distributed downloading, parsing notebooks and ML.NET pipelines to extract meaningful information, analyzing DS code efficiently, supporting extensible interfaces for an ever-increasing need for analytics). To address these challenges, we designed and implemented a purpose-built system—here, we only summarize the main steps for analyzing code. We employ a distributed crawler for downloading GitHub notebooks based on [26, 27]. Upon downloading, we parse notebooks based on the nbformat format, discard malformed notebooks, and upload extracted information from valid ones to a backend database. We include all metadata (kernel, language, and nbformat versions) and cell-level information (type, order, source code). Along with notebooks, we also keep track of related information provided by GitHub (e.g., repository and owner). ML.NET pipelines are similarly processed. We then perform several extraction passes using sophisticated extraction subsystems (e.g., extract and semantically annotate data flows to identify training pipelines), and store results back in the same database for composition purposes (e.g., count #users that train a Scikit-Learn learner using remote CSVs).

3. LANDSCAPE

We start our analysis with a bird’s-eye view of the landscape of DS through coarse-grained statistics from GitHub notebooks. In particular, our analysis in this section aims to reveal the volume of (a) notebooks and cells; (b) languages associated with them; (c) users; and (d) characteristics of code shape. Table 1 presents the overall statistics and drives our discussion.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Metric</th>
<th>GH17</th>
<th>GH19</th>
<th>GH20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notebooks</td>
<td>Total</td>
<td>1.23M</td>
<td>4.6M</td>
<td>8.7M</td>
</tr>
<tr>
<td></td>
<td>Deduped</td>
<td>66.0%</td>
<td>65.5%</td>
<td>65.7%</td>
</tr>
<tr>
<td></td>
<td>Linear</td>
<td>26.4%</td>
<td>29.1%</td>
<td>30.3%</td>
</tr>
<tr>
<td></td>
<td>Completely Linear</td>
<td>21.2%</td>
<td>23.3%</td>
<td>24.6%</td>
</tr>
<tr>
<td>Languages</td>
<td>Python</td>
<td>81.7%</td>
<td>91.7%</td>
<td>91.1%</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>18.3%</td>
<td>8.3%</td>
<td>8.9%</td>
</tr>
<tr>
<td>Code Cells</td>
<td>Total</td>
<td>34.6M</td>
<td>143.1M</td>
<td>261.2M</td>
</tr>
<tr>
<td></td>
<td>Deduped</td>
<td>64.5%</td>
<td>66.4%</td>
<td>66.9%</td>
</tr>
<tr>
<td></td>
<td>Linear</td>
<td>41.0%</td>
<td>38.6%</td>
<td>38.5%</td>
</tr>
<tr>
<td></td>
<td>Completely Linear</td>
<td>72.1%</td>
<td>80.2%</td>
<td>79.3%</td>
</tr>
<tr>
<td>Users</td>
<td>Total</td>
<td>100K</td>
<td>400K</td>
<td>697K</td>
</tr>
</tbody>
</table>

Table 1: Overall statistics for GitHub. Average yearly growth is ~2× for most metrics (except “Other languages” growing at 1.4×). Percentages for each metric are computed over the closer “Total” upward in the table, and remain relatively similar across years.
all other languages combined grew by roughly $1.4 \times$ on average, below Python’s $2 \times$; growth is increasing in recent years, but absolute volumes remain relatively low. These results suggest that Python is emerging as a de-facto standard for DS; system builders and practitioners should focus primarily on Python.

**Users.** User growth is similarly paced at $\sim 2 \times$ yearly growth, reaching an impressive 697K unique users in GH20. The top-100 most prolific users have each authored at least 432 notebooks in GH17 (693 in GH19, 1052 in GH20), yet no individual user is a “heavy-hitter.” This is best-seen in Figure 1, which shows the coverage of users on notebooks: note the absence of spikes on the left-hand side of the figure. Interestingly, the average number of notebooks per user has remained roughly the same across years (i.e., 12.1 per user per year).

**Code shape.** To better understand the code complexity of Python notebooks, we focused on how many notebooks or cells are linear (contain no conditional statements) or completely linear (contain neither conditional statements nor classes or functions). To do so, we analyzed every code cell by first parsing it to its AST representation using parso [8]. This resulted in an immense 12.9B AST nodes for GH20. At the notebook level for GH20, 25% are completely linear while 30% are linear. At the cell level, however, 76% are completely linear while 80% are linear. (Observations are similar for GH17 and GH19). Overall, this analysis shows that DS code is a mostly linear orchestration of libraries. It is thus amenable to transformation into declarative dataflows, leveraging compiler/database optimizations (e.g., lazy evaluation or cross-optimization between SQL and ML [13, 3, 2]).

**Takeaways:** Notebooks are emerging as a widely adopted programming medium, with a roughly $\sim 2 \times$ yearly growth across our metrics (see Table 1). Python is emerging as a de-facto standard language for authoring notebooks, with $\sim 91\%$ of GitHub notebooks being authored in Python today. Moreover, Python code in notebooks appears mostly as a linear orchestration of libraries ($80\%$ of notebook cells are linear).

**4. LIBRARIES**

We continue our analysis on GitHub notebooks by focusing on which libraries are used more prominently and in which combinations. This will implicitly help us characterize what practitioners code for in such notebooks (e.g., data processing, DNNs, or classical ML).

We look at this through Python notebooks (due to their dominance across years in the GitHub datasets) and the lens of import statements (i.e., `import...` or `from...import...`). Through imports, we observe a large number of unique libraries: 41.3K, 116.4K, and 175.6K unique libraries in GH17, GH19, and GH20, respectively (1.5× yearly growth).

In our analysis, next, we identify important libraries (§4.1), analyze the coverage of libraries on Python notebooks (§4.2), and conclude with statistical correlations (positive and negative) between libraries (§4.3).

**4.1 Important Libraries**

We now focus our analysis on identifying what we informally refer to as “important” libraries.

**Most used libraries.** Figure 2 shows the top-10 libraries according to the percentage of Python notebooks that each library is imported. We first confirm a key assumption of our study: notebooks are used primarily for DS activities—the top-10 libraries focus on common DS tasks (e.g., communicating with external sources, processing data, ML modeling, exploring and visualizing datasets, and scientific computations). Second, we verify our intuition that Numpy [18], Matplotlib [15], Pandas [20], and Scikit-Learn [32] are quite popular. However, their frequencies exceed our expectations (e.g., Numpy is used in $> 60\%$ of the notebooks). Third, by comparing usage across years, we observe that “big” (i.e., most used) libraries are becoming “bigger”, with several libraries losing in popularity (e.g., Scipy [31]); indicating a consolidation around a core set of libraries.

Overall, we believe these results suggest systems builders can focus efforts on a few frequently used libraries (e.g., Numpy or Pandas), but must also provide mechanisms to support a growing tail of less frequently used libraries.

**Highest ranking differentials in usage.** Comparing GH17 to GH20, the ranking in terms of usage changed substantially for a few libraries. Figure 4a shows libraries that increased their usage ranking the most over the last three years. We observe a popularity increase of ML frameworks (e.g., +38 positions for PyTorch [24]; Keras, XGBoost [37], and TensorFlow are in top
10). Furthermore, we observe an interest increase in image and text processing (PIFLOW [21], OPENCV [19], and GENSEM [10]), while the increase of PLOTLY [22] indicates an interest increase for (interactive) visualization. Finally, the increase of TQDM [36] indicates a growing interest for showing progress bars (which, in turn, indicates long-running computations), while the increase for SQLALCHEMY [33] and REQUESTS [25] suggest a need to access data stored in databases and other endpoints.

**Most increased in usage.** We complement the ranking differential analysis with the percentage increase in absolute terms. The top-10 libraries by highest percentage increase are shown in Figure 4b. Interestingly, we observe that “big” libraries are getting “bigger” at a faster rate than average. We observe a similar pattern for libraries related to deep learning (e.g., KERAS, PYTORCH, and TENSORFLOW), that we analyze in more detail next.

**Usage among deep learning libraries.** Figure 3 shows the percentage of notebooks that use TENSORFLOW, KERAS, THEANO [35], CAFFE [7], and PYTORCH. We observe that PYTORCH has increased the most, followed by TENSORFLOW and KERAS (with the usage of the latter two slightly decreasing in GH20). Furthermore, for both THEANO and CAFFE the usage rates have dropped considerably. Overall, deep learning is becoming more popular, yet accounts for less than 20% of DS today.

Overall, we believe that the above library usage changes can be mainly attributed to (a) shifts in user interests and operations (e.g., increased interest in image processing) and (b) social and community trends (e.g., companies or classes focusing on specific toolkits). The latter cannot be unveiled solely by data.

### 4.2 Coverage

How to prioritize implementation efforts and what is the impact of supporting a library are important questions during the development of systems for DS. In this direction, we perform a coverage analysis of libraries on notebooks (i.e., if we only support K libraries, how many notebooks would be fully covered?).

Figure 5 shows the cumulative percentage of notebooks covered (y-axis) while varying the number of libraries (x-axis). We sort libraries from the most to the least used and pick a prefix of size K. Our main observation is that by including just the top-10 most used libraries (i.e., the ones shown in Figure 2), we can reach a coverage of ~40% across years, while a coverage of 75% can be achieved by including the top-100 most used libraries. The increase in coverage, however, is diminishing as less used libraries are added in. More interestingly, a coverage of 100% is much harder to achieve in GH20 and GH19 than in GH17—suggesting that the DS field is expanding both in size and tail complexity.

### 4.3 Correlation

We conclude this section with an analysis of the co-occurrence (or correlation) of libraries. Figure 6a projects top-10 positively and negatively Pearson correlated libraries. (We defer a detailed discussion on individual correlations to [23]). Our main observation is that negative correlations highlight incompatibilities between supported data types and focus areas, whereas positive correlations validate common wisdom and indicate (a) to practitioners the need for expertise in certain combinations of libraries, and (b) to system builders which libraries to focus on co-optimizing.

**Takeaways:** Python notebooks on GITHUB emerge as large collections of DS-related activities, with all top-10 most used libraries focusing on DS. In particular, we observe an increased consolidation around a core set of highly used libraries: NUMPY, MATPLOTLIB, PANDAS, SCIKIT-LEARN are used in more than 23% of notebooks in GH20. The adoption of DL is increasing, yet it accounts for less than 20% of DS. At the same time, an increase in #used libraries indicates that DS is still an expanding field. Finally, our correlation analysis ranks library connections (Figure 6) to help system builders and practitioners decide what to co-optimize or get expertise on.
Table 2: #Extracted pipelines and distinct operators.

<table>
<thead>
<tr>
<th></th>
<th>GH17</th>
<th>GH19</th>
<th>GH20</th>
<th>ML.NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Pipelines</td>
<td>Implicit</td>
<td>164K</td>
<td>415K</td>
<td>1.4M</td>
</tr>
<tr>
<td></td>
<td>Explicit</td>
<td>10K</td>
<td>129K</td>
<td>252K</td>
</tr>
<tr>
<td>#Distinct Ops</td>
<td>Implicit</td>
<td>668K</td>
<td>1.8M</td>
<td>2.6M</td>
</tr>
<tr>
<td></td>
<td>Explicit</td>
<td>584</td>
<td>3.4K</td>
<td>5.5K</td>
</tr>
</tbody>
</table>

5. PIPELINES

So far, we have focused on understanding DS projects based on the libraries they are using (Section 4). In this section, we dive deeper into (primarily training) ML pipelines to provide an even finer-grained view of DS logic, that is also optimizable and manageable [2, 29].

Pipelines. We focus analysis on explicit and implicit pipelines. We refer to the former as pipelines defined declaratively: for our analysis, we use pipelines constructed using “sklearn.pipeline” [30] in notebooks, and the method chaining pattern in ML.NET. We refer to the latter as pipelines defined without such constructs but rather imperatively, using functions from different toolkits. To enable comparisons among explicit and implicit pipelines, we require implicit ones to be used for training, and have operators semantically close to SCIKIT-LEARN ones (which are close to ML.NET ones). As such, and given that PANDAS is the predominant way to read data (Section 4), we model implicit pipelines as data flows in notebooks with PANDAS reads and SCIKIT-LEARN learners being source and sink nodes, respectively.

Volumes. Table 2 outlines the overall volumes of extracted pipelines across datasets. Regarding GITHUB, we observe a steady increase (>1.9× growth), for both implicit and explicit pipelines. Furthermore, explicit pipelines have started increasing in size and stabilizing their growth. Moreover, implicit pipelines are larger in volume than explicit ones (e.g., 5× larger in GH20). Note, however, that from our extraction process, explicit SCIKIT-LEARN pipelines are subsets of implicit ones because a SCIKIT-LEARN pipeline can be a subpipeline of an implicit pipeline. Finally, from ML.NET we extracted 29.7M pipelines (2M unique after dedup), highlighting the overall importance of explicit pipelines in enterprises.

5.1 Pipeline Length

Figures 7a and 7b show the #pipelines per pipeline length, for both explicit and implicit pipelines, as a proxy for the complexity of problems tackled by different pipelines. (Intuitively, problems requiring more steps correspond to pipelines with more operators).

Explicit pipelines are right-skewed, with most having a length of 1 to 4. Among them, SCIKIT-LEARN pipelines have a smaller length than the ML.NET ones. Given that the operators used in both are of similar expressiveness, we believe this is related to the type of users behind each of them. As also discussed in Section 2, expert users (represented by ML.NET) tend to address more complicated problems that require more steps, resulting in longer pipelines. Note, however, that SCIKIT-LEARN pipelines are increasing in length over the years, which might indicate that corresponding users are addressing increasingly more complex problems. Implicit pipelines are also right-skewed, but their length is much larger than the one of explicit pipelines: most implicit pipelines have a length of 4 to 100, reaching a max length of 4K in GH20. We believe this is because explicit pipelines tend to encompass a much narrower functionality than the implicit ones (e.g., data cleaning and visualization steps are uncommon in explicit pipelines), and enable expressing operations in a more succinct way.

5.2 Operators

To study the type of operations that users perform using ML pipelines, we discuss #distinct operators and then we rank individual operators by frequency to identify important ones. Trends on operator frequencies are similar across years (we omit drill-downs to avoid duplication).

Table 2 shows that the #distinct pipeline operators across datasets has increased substantially for both implicit and explicit pipelines. Furthermore, it is evident that practitioners need the flexibility for introducing their own functionality: #operators of explicit SCIKIT-LEARN pipelines has increased due to user-defined transformers and learners, ML.NET contains 23K user-defined operators (and 536 system-defined ones), and #operators in implicit pipelines is much higher than the explicit ones due to the unconstrained way in encoding user logic.

Explicit Pipelines. Table 3 shows the top learners and transformers in explicit pipelines. An interesting observation is that normalizers are not within the top transformers for ML.NET while they are popular in SCIKIT-LEARN pipelines. This is because ML.NET adds these automatically based on needs of downstream operators or because data is normalized beforehand. Furthermore, regarding learners, Gradient Boosting and Random Forest are more popular in ML.NET than SCIKIT-LEARN. We believe this is due to their relative quality and the tasks observed in Microsoft. Finally, the importance of Poisson Regression in ML.NET indicates that data scientists in enterprises routinely deal with count data, and data like that is rarely released and made available publicly.
Implicit Pipelines. The trends on transformers and learners of explicit pipelines follow the implicit ones (modulo hyperparameter tuning being more prominent in implicit ones). Importantly, however, implicit pipelines are primarily dominated by operators for data wrangling (e.g., project, select, group by, concat, aggregations, and merge) and data visualization (e.g., print, plot, and head).

5.3 Coverage

We conclude our discussion on pipelines with a coverage analysis of operators on pipelines (to better help system builders realize prioritization opportunities). Figures 7c and 7d show the coverage of pipelines while increasing operators in descending order of operator frequency. Regarding explicit pipelines, top-100 and top-10 operators in `scikit-learn` and `ml.net` pipelines, respectively, can cover more than 80% of corresponding pipelines, highlighting optimization opportunities. In contrast, implicit pipelines are much harder to cover in full, with >50% coverage requiring >10⁴ operators.

Takeaways: Explicit pipelines are gaining considerable traction in GitHub notebooks (>1.9× yearly growth), and they are an established practice in an enterprise setting. Nevertheless, implicit pipelines continue to be the dominant way (by 5× in GH20) to specify DS logic in GitHub notebooks. While explicit pipelines appear to focus more on feature transformations, implicit ones contain more data preprocessing and visualization operations. Finally, explicit pipelines contain a core set of operators that system builders can prioritize support for, and practitioners casually introduce their own operators indicating a need for additional functionality.

6. INTERNAL IMPACT OF ANALYSIS

The analysis we presented here has been used to guide decisions on the DS space throughout Microsoft. Its impact has exceeded our expectations, leading to a continuous stream of drill-down asks from several teams.

Operationally, insights have already been used to (a) inform decision making on resource allocation and feature enhancements (e.g., what libraries to support and optimize in Azure services); and (b) verify the correctness of prior decisions (e.g., first class support of notebooks in Synapse, VSCode, and Azure Data Studio). Furthermore, this analysis has motivated decisions on several projects: (a) holistic optimization of DS pipelines [13]; (b) constructing KBs of Python libraries for data-flow analysis [17]; (c) tracking provenance from DS scripts [17]; and (d) overall shape visions and inform research agendas in the space [2]. Finally, ongoing collaborations triggered by this line of work include furthering the analysis of notebooks with analysis of context (i.e., output and markdown cells), supporting finer-grained analysis of DS code (e.g., understanding dataset types), and constructing ML models for code auto-completion and synthesis.

7. RELATED WORK

Understanding DS-related activities is crucial as most applications are becoming ML-infused [2]. Our work pushes the envelope on the topic by performing an extensive analysis of millions of Python notebooks and `ml.net` pipelines. Other studies reveal interesting insights on complementary DS aspects through discussions with data scientists about their engineering and collaboration practices [4, 12, 16, 39, 28, 11, 14]. Furthermore, the work in [6] presents a coarse-grained analysis on PyPI. Our work targets the Python language as well but with a special focus on DS projects. The work in [9] compares the package dependency graphs of CRAN, PyPI, and NPM. This work targets various languages and thus does not contain a detailed analysis of Python libraries. The study in [26] performs an analysis of GitHub notebooks with a focus on interactions between exploration and explanation of results. Our work incorporates the dataset used in this study (GH17) but focuses on analyzing the DS code structure instead. Finally, the study in [38] performs dynamic analysis of notebooks (to recommend data preprocessing operators) and provides interesting insights on DS operators, albeit on limited #notebooks due to the requirement on executing notebooks.

8. CONCLUSION

In this work, we amassed and analyzed approximately 40M DS projects, both publicly available at GitHub and Microsoft-internal ones. With this paper, we share the key findings of our analysis with the community, provide actionable interpretations, and share how this analysis has been used internally at Microsoft to inform decisions in the DS space. Finally, we believe this analysis is pragmatically useful to better inform investments by practitioners and systems builders—and a step towards a thorough, data-driven understanding of DS as a field.
9. REFERENCES


A Case for Enrichment in Data Management Systems

Dhrubajyoti Ghosh1, Peeyush Gupta1, Sharad Mehrotra1, and Shantanu Sharma2

1University of California, Irvine, USA. 2New Jersey Institute of Technology, USA.

ABSTRACT

We describe NRICHDB, a new DBMS technology designed for emerging domains (e.g., sensor-driven smart spaces and social media analytics) that require incoming data to be enriched using expensive functions prior to its usage. To support online processing, today, such enrichment is performed outside of DBMSs, as a static data processing workflow prior to its ingestion into a DBMS. Such a strategy could result in a significant delay from the time when data arrives and when it is enriched and ingested into the DBMS, especially when the enrichment complexity is high. Also, enriching at ingestion could result in wastage of resources if applications do not use/require all data to be enriched. NRICHDB’s design represents a significant departure from the above, where we explore seamless integration of data enrichment all through the data processing pipeline — at ingestion, triggered based on events in the background, and progressively during query processing. The cornerstone of NRICHDB is a powerful enrichment data and query model that encapsulates enrichment as an operator inside a DBMS enabling it to co-optimize enrichment with query processing. This paper describes this data model and provides a summary of the system implementation.

1. INTRODUCTION

This paper envisions a new type of data management technology that seamlessly integrates data enrichment in the data analysis pipeline. Data analysis pipeline refers to the process of acquiring data from data sources, potentially enhancing the data, ingesting it into a database system, and running queries on the enhanced data. Today, organizations have access to potentially limitless data sources in the form of web data repositories, social media posts, and continuously generated sensory data. Such data is often low-level/raw and needs to be enriched to be useful for analysis. Functions used to enrich data (referred to as enrichment functions in the paper) could consist of (a combination of) custom-compiled code, declarative queries, and/or expensive machine learning techniques. Examples of enrichment functions include sensor interpretation and fusion over sensory inputs, mechanisms for sentiment analysis over social media posts, and named entity extraction in text.

Traditionally, data enrichment is performed offline as part of a periodic Extract-Transform-Load (ETL) process. This process is performed inside a separate system and the enriched data is stored in a data warehouse for analysis. This approach adds significant latency between the time data arrives (or is created) and when it is available for analysis. [14] has highlighted the limitations of the traditional data warehouse approach in analyzing the recent data (as it arrives) for online business applications. It has led to the emergence of Hybrid Transaction/Analytical Processing (HTAP) systems that support both transactional and analytical workloads. A warehouse strategy (of periodic enrichment as part of ETL) exhibits similar limitations in application contexts, where enrichment is part of the data processing pipeline. One possibility to overcome this limitation is enriching the data as it arrives. Systems (e.g., Spark Streaming [20] often used for scalable ingestion) are capable of executing enrichment functions on newly arriving data prior to its storage in a DBMS. Recently, [17] has explored ways to optimize enrichment during ingestion by batching such operations.

Enriching data at arrival is only feasible when enrichment functions are simple. Complex functions (e.g., Multi-layer Perceptron and Random Forest), often, used to classify/interpret incoming data, may take several hundred milliseconds to execute on a single core of a modern server.1 Applying such functions at ingestion will allow a system to ingest only tens of events per second per core which is very low. An alternate strategy is to restrict ETL process to selectively enrich only a part of the data (based on expected usage) at ingestion. However, predicting usage is difficult, especially in an online setting where an analyst can pose any adhoc query. If the prediction underestimates the need of enrichment, it may not support certain queries and overestimation leads to wasted enrichment and resources.

Motivating Example. A quintessential example domain for which NRICHDB is designed, is a sensor-driven smart space environment. Such an environment is often instrumented with a large number of sensors producing data, which is stored in databases. Such data consists of videos, images, data from motion sensors, as well as connectivity data of user’s mobile devices with WiFi access points. Such data needs to be processed before it can be used by applications. E.g., [12] uses connectivity data of user’s mobile devices with WiFi access points to localize users inside a building. Furthermore, one can use surveillance camera images to localize users more accurately. Localization based on WiFi connectivity data or images can be expensive, e.g., analyzing a single WiFi

1E.g., a server of 64 core Intel Xeon CPU E5-4640, 2.40GHz, and 128GB memory.
connectivity event takes $\approx 200$ms, and analyzing a single image takes $\approx 1s$. If we consider a campus environment with hundreds of Wi-Fi access points and cameras (where $\approx 1,000$ Wi-Fi events/sec and $\approx 100$ images/sec are produced by the sensors), we will need $\approx 5$ minutes of processing time for locating person using the data that has been generated in one second, and such a processing time is not feasible.

Instead, we need to process such data during query execution in an adaptive manner. Queries on such data can be ad-hoc: for example, a visitor planning to attend an event at a location may wish to know the attendees already arrived (or the count) apriori to avoid crowded regions. Another example will be exploring suspicious activities that may create a timeline of events at different parts of a building using Wi-Fi connectivity data and then performing detailed analysis using camera images. To answer such ad-hoc queries, if a system enriches the required data at query time, it can still result in high latency depending on the query selectivity.

Motivated by the above limitations, we design ENRICHDB — an adaptive data management technology that allows enrichment to be performed all through the data processing pipeline. i.e., during ingestion, triggered based on events, or during query processing. ENRICHDB is designed based on the following criteria:

**Semantic Abstraction and Transparency of Enrichment.** ENRICHDB supports a declarative interface to specify and to link enrichment functions with higher-level observations that the functions generate from raw data. Users may associate one or more such functions that differ in terms of quality (e.g., uncertainty in the enriched value) and cost (e.g., execution time of the function).

In ENRICHDB, developers do not have to deal with raw data directly — applications can be fully developed based on higher-level semantic observation. Furthermore, developers do not have to be concerned about what data has to be enriched, using which functions, and at what stage of data processing. ENRICHDB maintains the state of enrichment of objects and performs enrichment automatically based on the current state of objects.

**Optimization of Enrichment.** ENRICHDB allows enrichment all through the data processing pipeline. ENRICHDB makes sure that enrichment of objects is performed optimally. At query time enrichment, ENRICHDB exploits query optimizer to prune away enrichment of objects that do not influence the query results. Furthermore, ENRICHDB allows enrichment of data closer to where the data resides resulting in a low data movement.

**Progressive Computation.** When ENRICHDB executes enrichment functions during query processing, it produces query answers progressively. A progressive query answering technique (motivated by Approximate Query Processing systems [10] that provided progressive query answering for aggregation queries) produces an initial set of answers that are improved over time as data is further enriched.

The cornerstone of ENRICHDB is *Enrichment Data and Query Model (EDQM)* that integrates enrichment as a first-class operator in the database system. This paper describes both data and query models in §2 and briefly describes the implementation of ENRICHDB in §3. The codebase and detailed discussion on design decisions are presented in [2].

## 2. DATA AND QUERY MODEL

In this section, we develop a new data and query model, called Enrichment Data and Query Model (EDQM).

### 2.1 Data Model

In EDQM, the data is modeled using relations where a relation can have two types of attributes: (i) *derived* attributes that require enrichment and (ii) *fixed* attributes that do not require enrichment. Each derived attribute is optionally associated with a domain size. If the domain size is not specified, then that attribute is considered to have a value from a continuous range. The command for specifying a relation in ENRICHDB is shown below.

```
CREATE TABLE wifi(id int, user_id char(30),
timestamp time, wifi_ap char(30),
location int derived:304);
```

The value of a derived attribute is determined using one or more enrichment functions associated with it.

**Enrichment functions.** EDQM supports a general class of enrichment functions (frequently used in real world). The input to an enrichment function is a tuple and the output is either a single value, multiple values, or a probability distribution, as described below.

We categorize enrichment functions based on the output cardinality: (i) *single-valued*: outputting a single value, e.g., a binary classifier [16], (ii) *multi-valued*: outputting a set of values, e.g., top-k classifiers [11], (iii) *probabilistic*: outputting a probability distribution over the possible values of a label, e.g., probabilistic classifiers [6]. Also, enrichment functions can be categorized based on the size of output domain: (i) *categorical*: predicts outputs from a finite set of possible values, e.g., sentiment of positive/negative, and (ii) *continuous*: outputs a real number, e.g., a weather of 72.8°F.

An enrichment function is associated with two parameters: (i) *cost*: the average execution time/tuple, and (ii) *quality*: a metric of goodness (i.e., accuracy) of enrichment function in determining the correct value of the derived attribute.

**Training of enrichment functions.** EDQM supports training procedures for enrichment functions that internally uses machine learning models to predict the value of derived attributes. Often such models use a supervised learning method [5] that learns a mapping function between a set of input and output pairs from a ground truth data set (often referred to as training data). A user needs to

---

2The derived attributes cannot be updated directly by the user.
### Table 1: The wifi table (location is derived).

<table>
<thead>
<tr>
<th>id</th>
<th>user_id</th>
<th>time</th>
<th>wifi_ap</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>24</td>
<td>09:14</td>
<td>56</td>
<td>L1</td>
</tr>
<tr>
<td>t2</td>
<td>22</td>
<td>10:26</td>
<td>110</td>
<td>NULL</td>
</tr>
<tr>
<td>t3</td>
<td>108</td>
<td>14:10</td>
<td>116</td>
<td>L4</td>
</tr>
</tbody>
</table>

### Table 2: State output for derived attributes.

<table>
<thead>
<tr>
<th>tid</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>L1:0.54, L2:0.35, L3: 0.11</td>
</tr>
<tr>
<td>t2</td>
<td>L1: 0.1, L2: 0.1, L3: 0.1, L4: 0.1</td>
</tr>
<tr>
<td>t3</td>
<td>L4: 0.3, L5: 0.15, L6: 0.08</td>
</tr>
</tbody>
</table>

The cost and quality of enrichment functions can either be specified by user or can be determined automatically by using several methods, e.g., train/test split and k-fold cross-validation during the training phase.

#### 2.2.1 Query Language

The query language of `ENRICH DB` is an extended

Table 1: The wifi table (location is derived).

specify the table that stores the training data for the model. Below, we show an example where a machine learning model of Multi-Layer Perceptron (MLP) is learned using a training procedure of `model_train`. The training data is stored in `wifi_train` and the name of the model is `location_mlp`. It uses the attributes values of `feature` as input to the model and outputs the prediction for `location` attribute. The model-specific parameters are passed as a string in `model_params`. ¹

```sql
SELECT db.model_train('wifi', 
  'location_mlp', 'mlp', 'location', 
  'feature', model_params);
```

The `cost` and `quality` of enrichment functions can either be specified by user or can be determined automatically by using several methods, e.g., train/test split and k-fold cross-validation during the training phase.

In real scenarios, often multiple enrichment functions are used to perform a particular analysis. To localize a person, one can use multiple ML functions, e.g., decision tree, random forest, and multi-layered perceptron models. `ENRICHDB` supports specification of such functions using a function-family. Formally, the set of enrichment functions for a derived attribute `A_i` are called function-family of `A_i`. (We use calligraphic font for derived attributes.) Outputs of enrichment functions in a function-family are combined using a combiner function. One can use weighted-average, majority-voting, or stacking-based [19] combiner functions. Below we show, creation of function-family for `location` attribute consisting of multiple functions along with their cost (seconds/tuple) and quality (measured in AUC) respectively, using the assigned enrichment functions command.

```sql
SELECT db.assign_enrichment_functions('wifi', 
  [['location',3,'location_dt',0.8,0.7],
   ['location',4,'location_fo',0.6,0.8],
   ['location',1,'location_mlp',0.95,0.9]]);
```

#### State of a Derived Attribute

Enrichment state or state of a derived attribute `A_i` in tuple `t_k` (denoted by `state(t_k,A_i)`) is the information about enrichment functions that have been executed on `t_k` to derive `A_i`. The state has two components: state-bitmap that stores the list of enrichment functions already executed on `t_k` and state-output that stores the output of executed enrichment functions on `t_k`-`A_i`. E.g., consider that there are three enrichment functions `f_1, f_2, f_3` and out of which `f_1, f_3` have been executed on `t_k`-`A_i`. Also, assume that the domain of `A_i` contains three possible values: `d_1, d_2, d_3`. Thus, the state-bitmap for `t_k`-`A_i` contains `{101}`, i.e., only first and third functions are executed and the state-output of `t_k`-`A_i` contains: `{[0.7,0.3],[0.8,0.1],[0.5]}`, i.e., the output of the first and third enrichment functions (remaining arrays are left empty). The state-output stores a list of probability distributions when the enrichment functions are probabilistic. For single/multi-valued functions and continuous functions, the state-output attribute stores the actual output of the function instead of a probability distribution, e.g., `{[72.4],[76.8]}`.

#### State of Tuples and Relations

The notion of state of derived attributes is generalized to the state of tuples and relations in a straightforward way. The state of a tuple `t_k` is the concatenation of the state of all derived attributes of `t_k`, e.g., the state of a tuple `t_k` of a relation `R` with three derived attributes `A_p, A_q, A_r` is denoted by `state(t_k) = (state(t_k,A_p)||state(t_k,A_q)||state(t_k,A_r))`.

#### Relative Ordering of Enrichment Functions

In `EDQM`, the user can specify (or can be learned by `ENRICHDB` using a training dataset) the relative order in which enrichment functions need to be executed. This order is specified using the state of tuples for each derived attribute. Such relative ordering is important for selecting different enrichment functions to be executed on a tuple. This ordering is stored in a table called `DecisionTable` (see Table 3).

This table, for each derived attribute of a relation, stores a map that — given the current state of a tuple with respect to the attribute — specifies the next function that should be executed to further enrich the attribute, as well as (optionally) the expected improvement in quality (denoted as `benefit`) that will result from enriching the attribute of the tuple. `ENRICHDB` uses benefit and the cost of enrichment functions to order the enrichment of tuples.

In Table 3, each row stores a map containing (state bitmap, entropy range) as keys and the corresponding (next best function, benefit) pair as values. Consider the tuple `t_1` of `wifi` table (see Table 1) and assume that the state output bitmap of `t_1` is `[1,0,0]` and the location state output of `t_1` is `[0.54,0.35,0.11],[0,0,0],[0,0,0]`. The entropy of `t_1` is `(-0.54 × log_3(0.54) − 0.35 × log_3(0.35) − 0.11 × log_3(0.11)) = 0.85`. From first row of Table 3, since entropy of `t_1` is in the range (0.75-1), the decision table specifies that the next best function to execute is `f_2` and its benefit as 0.22.

#### 2.2.2 Query Model

This section describes the query language (§2.2.1), query semantics (§2.2.2), and the goal of enrichment (§2.2.3).

##### 2.2.2 Query Language

The query language of `ENRICHDB` is an extended
version of SQL. Queries in ENRICHDB are associated with a query semantics (which are required to deal with probabilistic values of derived attributes) and a (optional) quality parameter for the quality of the query results.

Two types of query semantics for probabilistic data have been proposed in the past: (i) determination-based semantics [7] and (ii) possible world (PW) semantics [15]. The determination-based semantics converts probabilistic representation to a single or a small set of deterministic worlds. The query is executed in each of these worlds and a single deterministic answer is produced. In contrast, in PW semantics, all possible worlds are generated (implicitly/explicitly) from probabilistic representation and the query is executed in each world. The result consists of all possible tuples along with their probability of being part of the result in at least one world. The choice of one semantics over the other depends on the application scenarios. In some scenarios, applications can make good decisions by using the most probable answers, whereas in some scenarios, they require analysis of all possible answers along with their probability distribution. Due to simplicity, we have implemented the determination-based query semantics in ENRICHDB (the implementation of PW semantics is under development).

An example query in ENRICHDB that requires a minimum quality of 0.9 is shown below:

```sql
SELECT wifi.location as p_location, wifi.timestamp as p_time FROM wifi
WHERE p_location = 'L1'
AND p_time BETWEEN ('10:00','12:00')
AND QUALITY 0.9;
```

### 2.2.2 Query Semantics

In determination-based query semantics, tuples of all participating relations in a query are determined first before evaluating the query. The process of converting a probabilistic data representation, i.e., the output of probabilistic enrichment functions, to a deterministic representation is referred to as the determination process.

Consider a derived attribute $A_i$ and a tuple $t_k$. The value of tuple $t_k$ in attribute $A_k$ (i.e., $t_k.A_k$) is determined using a determination function ($DET(.)$) based on tuple’s state. $DET(state(t_k.A_i))$ returns a single or multiple values for $t_k.A_i$ or a NULL value, representing a situation when state of the attribute does not provide enough evidence to assign any value for $t_k.A_i$. Determinization concept naturally extends to a tuple and a relation. The determinized representation of a relation $R$ is denoted by:

$$DET(R) = DET(state(t_i.A_j)) \forall t_i \in R, \forall A_j \text{ of } R.$$
to $P$. When both expressions evaluate to either $T$, $F$, or $U$, we follow the same evaluation logic as in standard SQL.

**Aggregation.** Aggregation functions on fixed attributes are evaluated as in SQL, while, on a derived attribute, return a range of values $[l, u]$, denoting the lower and upper bounds of aggregated value. An aggregation function (e.g., `count`, `sum`, `min`, `max`) applied to all $T$ tuples of a set produces the lower bound $l$, while applied to all $T$ and $P$ tuples produces the upper bound $u$. E.g., consider a query on Table 1 that counts the occupancy of location $L1$, and assume that the table has 250 tuples of which 100 tuples evaluate to $T$, while 20 of the remaining 150 tuples evaluate to $P$. Hence, the condition evaluation logic returns a range of $[100, 120]$. Likewise, group-by aggregation results in such a range per group.

**Top-k Aggregation.** ENRICHDB first evaluates aggregation functions for each group-by key (as described above), and then ranks their outputs by a ranking function. The query result consists of a set of group-by keys with the top-k ranks. The purpose of the ranking function is to return a minimal answer set $A$, such that the real top-k groups are guaranteed to be part of $A$. ENRICHDB sorts the group-by keys based on the lower bounds in a descending order and selects the first $n$ (where $n \geq k$) group-by keys as the minimal answer set $A$ such that the upper bound of $(n+1)$-th key is lower than the lower bound of the $n$-th key. This ensures that the $(n+1)$-th group-by key cannot be part of the top-k answer set.

Consider a query that returns top-2 locations with highest occupancy from Table 1. Suppose after applying `count()`, the locations had following bounds for occupancy: $L1$: $[100, 150]$, $L2$: $[110, 120]$, $L3$: $[100, 115]$, and $L4$: $[80, 95]$. The results returned are locations $\{L1, L2, L3\}$ that guarantees that the actual top-2 locations (i.e., $L1, L2$) are part of the result. $L4$ is excluded as the upper bound of occupancy (i.e., 95) is lower than the lower bounds of locations in the answer.

Based on the definition of determinization function and the predicate evaluation logic as described above, we define the query semantics as follows:

$q(R_1, R_2, ..., R_n) = q'(DET(R_1), DET(R_2), ..., DET(R_n))$

Here, $q(R_1, R_2, ..., R_n)$ is a query on relations $R_1, ..., R_n$, $DET(R_i)$ is the determined representation of the $i^{th}$ relation. Query $q$ is rewritten as $q'$ to be executed on the determined representations of relations using the four valued logic as described above.

**2.2.3 Quality Measure of Query Results**

In ENRICHDB, we measure the quality of answers to (i) set based queries using Jaccard’s similarity or expected F-score measure, (ii) aggregation queries using the root-mean-square error, mean absolute error, or the half-interval length of query answer, and (iii) group-by and top-k queries using the summation of half-interval lengths of all group by keys.

**Progressive Score.** Since ENRICHDB allows users to stop query evaluation at any instance of time (even before the quality requirement is met), performing enrichments impacting answer quality as early as possible is needed. ENRICHDB’s effectiveness is measured using the following progressive score (similar to [13, 4]):

$$\mathcal{PS}(Ans(q,E)) = \sum_{i=1}^{|E|} W(e_i) \cdot [Q(Ans(q,e_i)) - Q(Ans(q,e_{i-1}))]$$

The query execution time is discretized into sub-intervals, called epochs $\{(e_1, e_2, ..., e_z)\}$, $W(e_i) \in [0, 1]$ is the weight allotted to the epoch $e_i$, $W(e_i) > W(e_{i+1})$, $Q$ is the quality of answers, and $[Q(Ans(q,e_i)) - Q(Ans(q,e_{i-1}))]$ is the improvement in the quality of answers occurred in the epoch $e_i$. The quality $Q$ is measured according to the type and semantics of the query as discussed above. Given a query, a quality metric, and a set of weights assigned to each epoch, ENRICHDB’s goal is to achieve maximum progressive score for the query, if query execution is stopped early.

### 3. ENRICHDB IMPLEMENTATION

There are two possible ways of implementing the above data model as shown in Figure 1: (i) a loosely coupled (LC) approach, wherein an enrichment module is implemented separately from the DBMS, and (ii) a tightly coupled (TC) approach, wherein an enrichment module is tightly integrated with the query processing module of the DBMS. ENRICHDB follows TC approach on top of PostgreSQL as it uses the query context to eliminate redundant enrichment.

Consider a query with two selection conditions on derived attributes $A_1$ and $A_2$, connected using `AND`, the LC approach will enrich the tuples for both $A_1$ and $A_2$. In contrast, in TC, after enriching $A_1$ of a tuple, if it does not satisfy the condition on $A_1$, then attribute $A_2$ is not enriched. Such a pruning strategy can be very effective, when queries are complex and selective. Furthermore, the TC approach executes the enrichment functions closer to the data, in the database engine.

An ENRICHDB query is wrapped in a stored procedure that internally executes appropriate SQL queries on top of PostgreSQL tables during multiple epochs. The query results are maintained using Incremental Materialized Views (IMV) [3] to reduce the overhead of executing queries multiple times. Enrichment functions are implemented as user-defined functions (UDFs), and their execution is orchestrated by a special UDF that executes enrichment functions as UDFs by taking them as arguments. For implementation details, please check [2].

Figure 1: Loosely coupled system versus ENRICHDB.
4. USE CASE OF ENRICHDB

This section describes how ENRICHDB can be used to
develop the application described in §1 that finds out location
of attendees already arrived for an event. It requires fine-grained localization of people using WiFi connectivity
data inside a building using multiple predictive models with
different cost and quality [12]. The application poses queries
to find out attendees at a location between two time intervals.
Ease of Application Development. To develop this application,
the steps to take in ENRICHDB are presented below.
ENRICHDB-based implementation is much simpler (∼26 lines of code) as compared to any loosely coupled implementation,
where enrichment is performed outside of DBMS
and requires much more lines of code (∼130 lines [2]).

```
1. CREATE TABLE wifi(id int, user_id char(30),
   2. time_timestamp, wifi_ap char(30),
   3. location int derived:304)
4. -- Training ML Models
5. SELECT db.model_train('wifi_train',
6.  'location', 'feature()', model_params);
7. -- Associating functions with 'location'
8. SELECT db.assign_enrichment_functions(
9.  'wifi', ['location', 'loc_fo', 0.9, 0.8]);
10. -- Setting up decision table
11. SELECT db.learn_decision_table('wifi',
12.  'location', 'WifiValidation');
13. -- Adding data
14. SELECT db.enriched_insert('INSERT INTO wifi
15. VALUES (1,1051,"10:02",12, NULL)');
16. -- Executing Queries
17. SELECT call db.exec_driver('SELECT location, time
18. FROM wifi WHERE id<100 AND location = "11"
19. AND time BETWEEN ("10:00","12:00")');
```

Performance Evaluation. Figure 2 shows the quality of results achieved by ENRICHDB with respect to time
for the query described above (Line 20). The results are produced at the end of each epoch, where the epoch duration
is set to 5 seconds. The quality is measured using normalized $F_1$ measure, i.e., $F_1/F_{1\text{max}}$, where $F_{1\text{max}}$
is the maximum $F_1$ measure achieved during query execution.
Figure 2 highlights that ENRICHDB provides high-quality query
results within the first few epochs of a query execution as compared
to the strategy of eager enrichment that enriches the tuples completely
and then executes the query.

5. RELATED SYSTEMS

ENRICHDB can be viewed as a system similar to Extract-
Load-Transform (ELT) based systems [1], where the data is
extracted and loaded to a data warehouse/lake system and
enrichment is performed at the analysis time. In contrast, ENRICHDB provides a powerful data model to developers that make application programming very easy. Query-driven approaches of data cleaning has been studied significantly [18, 8]. However, such works were restricted to only data cleaning
algorithms of duplicate detection, duplication elimination,
and entity resolution, whereas ENRICHDB supports a
general class of enrichment functions such as classification,
clustering, and regression functions. Systems for supporting ML using databases (e.g., Apache MADlib [9], RIOT [21]) are designed to learn ML models inside or on top of
database systems; however, such systems do not support semantic abstraction of specifying enrichment functions and linking them to higher-level observation generated by them
as supported by ENRICHDB.

6. CONCLUSION

In this paper, we proposed ENRICHDB — a new system
for supporting data enrichment inside a single data management
system. The cornerstone of ENRICHDB is a powerful enrichment data model that encapsulates enrichment as an operator inside a DBMS enabling it to co-optimize enrichment
with query processing. Furthermore, ENRICHDB provides semantic abstraction, transparency of enrichment,
and progressive computation of queries to make application programming very simple for the developers.

Acknowledgements. This material was partially funded by the research
sponsored by DARPA under agreement number FA8750-16-2-0021 and
NSF Grants No. 1527536, 1545071, 2032525, and 2008993.

7. REFERENCES

[9] J. M. Hellestein et al. The madlib analytics library or MAD skills, the SQL. VLDB 2012.
This column was established by Richard Snodgrass in 1998 and was continued by Ken Ross from 1999 to 2005. It celebrated one of the key aspects that makes us grow as a research community: the papers that influence us. At each issue, different members of the data management community wrote anecdotes about a paper that had a unique impact in their career. The anecdotes highlighted that impact can come in many forms. A paper’s value is not only in its citation count, but also in the way it influences individuals who in turn influence other individuals that make up our community. Such impact is not countable.

When the SIGMOD Record’s editor-in-chief Rada Chirkova approached me to revive this column last year, I was immediately excited. I would like to thank Rada Chirkova, Richard Snodgrass, and Ken Ross for this opportunity. I am delighted to present the three invited contributions for this issue. Hope you enjoy reading them as much as I did.

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.

M. Tamer Özsu
University of Waterloo, Canada

tamer.ozsu@uwaterloo.ca

When you have been in business as long as I have, it is hard to select one paper that influenced me - so I cheat and select two: one that influenced how I approach new topics and the second that influenced my career in a fundamental way.

Jim Gray.

Notes on Data Base Operating Systems.

This paper was first published as an IBM San Jose Laboratory Technical Report in 1977, which is the form I first read it. At that time, we used to subscribe to technical reports from a few database pioneer research places and IBM San Jose was one of them. This is an exceedingly well-constructed paper that properly frames the issues and walks through them systematically. At the time, I was doing my master’s and transitioning to computer science from an undergrad degree in industrial engineering (with a whole bunch of undergrad CS courses to annoy advisors who insisted that I gain some breadth). But I did not have sufficient foundational knowledge. As I read materials, I kept trying to develop a reference model in my mind as to what the important pieces were and how they fit together. Jim’s paper was eye opening to me in how systematically he framed the issues and developed the arguments. I have used it as a mental template when I tackle a new area and try to write a framework, at least for my own understanding.

Incidentally, the paper contains the following sentence: “If one tries to implement such an application on top of a general purpose operating system it quickly becomes clear that many necessary functions are absent from the operating system.” After 45 years and tremendous development, it is interesting that DBMS-OS co-design people would probably still say the same thing.

Michael Stonebraker and Eric Neuhold.

A Distributed Database Version of INGRES.
Berkeley Workshops, as they were known, ran for a few years and were sources of great insight into new directions. This paper originally appeared as a Berkeley technical report in 1976 and then was published in this workshop’s proceedings which is where I read it. This was when INGRES and System R were being built as first relational DBMSs at Berkeley and IBM San Jose Laboratory, respectively. Thus, this paper appeared very early during INGRES development. It describes the extension of the system as one that runs on a single UNIX machine to run on multiple UNIX machines assuming “the existence of the UNIX to UNIX communication facility being constructed by the UNIX designers.” This was early in the development of both database and computer network technologies, and I was new to both. I was desperately trying to wrap my head around how to reconcile the “integration” objective of database systems with “decentralization” objective of computer networks. The design of the multi-machine version of INGRES was eye opening to me and I decided to do my PhD in distributed databases. The rest is history.

Alberto Lerner
University of Fribourg, Switzerland
alberto.lerner@unifr.ch

Donovan A. Schneider and David J. DeWitt.

The paper dates back to 1989 and compares the performance of best-known hash-based join algorithms when executed in a parallel setting. We might call it an Experiment & Analysis paper nowadays, but it was much more than that. It proposed parallel formulations that did not exist before for centralized join algorithms and analyzed their performance over a rich set of scenarios. There were several important insights in the paper. It found that hash-based join algorithms were quite amenable to parallelization. It showed how the availability of main memory could be crucial to performance. It also called attention to the fact that skewed data affects the algorithms. The authors stopped short of suggesting skew handling techniques. This came at a later paper. Interestingly, the authors also performed experiments with diskless workstations in a setting that resembles what we call serverless today.

The paper represents the period when databases were essentially being redesigned for parallel execution. At the time, parallelism came from using a set of networked workstations. Little did we know that ten years or so after the paper was published, we would be ushered into the multicore era. Many of the paper’s techniques are still applicable today, and the paper (and a cluster of related ones) have been regularly cited ever since.

I came upon the topic of parallel databases when I was looking for problems to work on for my master’s degree. The paper was part of a compilation book much like the “Red Book,” only this time it focused specifically on parallel query processing papers, and it was green. The compilation appeared in 1994, edited by Hongjun Lu, Beng-Chin Ooi, and Kian-Lee Tan. Its opening paper is Gray’s and DeWitt’s presentation of parallel database systems as the future of high performance. When I found the paper, clusters of ethernet-connected workstations were quite common, which allowed me to reproduce the results. I still remember the thrill of writing parallel software and engaging multiple machines to attack increasingly large data sets. The experience sparked an interest in query execution techniques that I keep to this day.

Tianzheng Wang
Simon Fraser University, Canada
tzwang@sfu.ca

Ryan Johnson, Ippokratis Pandis, Radu Stoica, Manos Athanassoulis, and Anastasia Ailamaki.
Aether: A Scalable Approach to Logging.

In fall 2012 I arrived at the University of Toronto as a new grad student hoping to work on systems. Database systems, of course, is one of these areas, yet I wasn’t quite sure about pursuing it coming from a storage and embedded systems background. It was the Aether paper that settled it. Although the title says it’s about logging, there you find all kinds of delicate interactions and design issues touching various components inside and beyond the DBMS kernel to make logging (and hence the whole system) scalable: concurrency control, synchronization, OS CPU and I/O scheduling and more! I was in particular fascinated by the ideas of
commit pipelining and early lock release, where one can get throughput like asynchronous commit yet without losing correctness. Of course, the fineprint was that we are trading latency for throughput, a classic tradeoff in high-performance storage engines. It was also fun to think about the twist on group vs. pipelined commit, although in many cases we take pipelining for granted when we say group commit.

The bulk of my PhD thesis got inspirations from this paper when I explored persistent memory and RDMA for scalability and reliability at even bigger scales than Aether could handle. To this day, the fundamental principles in this paper still find their way into my own research group’s latest work. This paper remains my favourite recommendation for anyone who would like to get a taste on what and how to think about when building a transactional storage manager.
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

Deciding What Not to Do

David Maier, Portland State University

I recall an early conversation with my advisor, a couple years after I completed my PhD. I was worried about not having been invited onto any program committees when others in my cohort were getting such opportunities. He assured me that it would come in time, though I was still anxious. He was right; after another year or so, the invitations started coming. At this point, I need to decline most of them, or I’d spend all my time reviewing.

It’s pretty much guaranteed that anyone with a productive research program will reach a “tipping point” where the professional opportunities — PCs, conference organization, editorships, professional society offices, agenda-setting panels, paper and proposal reviewing — exceed his or her capacity to contribute. Some people get in a jam when opportunities start presenting themselves, saying yes to most requests, then being swamped with professional activities. While as a responsible member of the research community, you should expect to contribute your fair share of service, as your career progresses, choosing activities wisely becomes increasingly important.

Requests for professional service are often accompanied by reasons why you should accept the request, not all of which you should accept at face value. These are not reasons to say yes:

- Because you’d be the best person. (Or someone tells you you’d be the ideal person to do it.) The second-, third- or tenth-best choice might well do a wholly adequate job.
- Because it won’t happen if you don’t do it (or so you’re told). Maybe such a lack of candidates for the task means it’s time to phase out that activity.
- It’s your turn. Perhaps it’s true that you’ve never organized a particular conference before, but you may currently be contributing to the community in large ways that others aren’t, such as editing a journal or heading a professional society.
- Your participation is sought to represent an underrepresented group. Obviously, diversity in professional activities is valuable, but there is a danger of overloading members of such groups precisely because there are relatively fewer of them. Moreover, in Computer Science, members of these groups are concentrated in the more junior segments, and such people need to focus on establishing their research and academic careers. If you are a member of an underrepresented group, remember that the imbalance is not your doing, and you can’t be expected to represent the group in every opportunity that comes along. (If you know other people who might benefit from participation in the activity, ask them if it’s okay that you suggest them as alternate choices.)
- Because everyone else seems to be taking part — all the “cool kids” are involved. First of all, you are
probably subject to observation bias: no one is pointing out to you the people who aren’t involved. Further, you should have a more principled reason to say yes (see below).

You should figure out what are your valid reasons for saying “yes” — it shouldn’t simply be to “pay your dues.” You should be as strategic in choosing your service activities as you are in deciding on research directions. Your scarcest resources are your time and attention; you should invest them carefully. Think about how activities might benefit you and where you could make a unique contribution. What do you find enjoyable, what are you particularly effective at? Try to avoid what has caused you pain and anxiety in the past. Consider even writing down your reasons for saying yes (and revising them periodically), so you can consult them when an opportunity arises. My current list of reasons to say yes includes:

- There will be an opportunity to meet others outside my immediate field.

- It will help get me spun up in an area I’m interested in moving into.

- To repay help from someone else. I see this aspect more as gratitude than obligation. You contributed to my special issue — I could buy you a beer, or I could help you organize tutorials.

- To push policy changes. You might not agree with current policies of an organization — the activity offers a chance to advocate for changing it.

- To influence policymakers, such as agenda-setting workshops for funding agencies and government study panels.

- The activity offers a high return on time invested, such as organizing a panel.

I also try to have limits per category of task, for example, only one program committee at a time.

Some closing thoughts on service. It’s better to do really well at a few things than make a minimal contribution to many. The latter may pad your resume, but won’t enhance your reputation. Also, you get a better reputation doing 100% of a moderate task than a moderate amount of a massive task. If you agree to do something, do it gladly, not grumpily. Don’t say yes then complain constantly about the task. Do watch out for bait-and-switch, or ambiguity about an activity. Put bounds up front about what you are willing to do — hours per week, number of plane trips. Talk to someone who’s done the job in the past about the effort required, and the kinds of activities. (How frequent are meetings? How much work between meetings?) Don’t be shy about asking for paid support, such as a conference manager if you are General Chair or handling local arrangements for a conference. While some departments and companies may be willing to let you use staff for coordination and clerical activities, many organizations, especially in academia, are on tight budgets and short-staffed. You want to focus your time on tasks that can’t be offloaded. Finally, if you are going to decline, do so quickly, and suggest alternative names, if you can think of any.

And remember to make time for your family. If you have children at home, don’t neglect investing in them.
PERSPECTIVES ON DATA(BASE/SCIENCE) EDUCATION

With the surge of interest in all things “data”, enrollments in traditional data-oriented courses are at an all time high. The rise of Data Science as a discipline has also led to the creation of new courses whose content significantly overlaps that of an introductory database course. This column presents a series of perspectives on data (base/science) education to help educators think about what we should be teaching in our courses, and what resources we should use to teach them.

This episode contains two contributions. The first discusses a recent effort called OpenDS4All, whose goal is to accelerate the creation of Data Science curricula by providing a “starter set” of open source data science training materials, including PowerPoint slides and Jupyter Notebooks with Python code. The second presents a process for making pedagogical decisions of how to construct a data management class given the wealth of content that can be included.

Susan B. Davidson
University of Pennsylvania

OpenDS4All: Accelerating the Creation of Data Science Curricula at Academic Institutions

Andre de Waal
IBM
andre.dewaal@ibm.com

Ana Echeverri
IBM
ana.echeverri@us.ibm.com

Background and motivation

Over the past decade, the interest in data science careers has been unwavering. In 2001, Linkedin ranked data science specialists as one of its top jobs, and noted that hiring for these roles have grown nearly 46 percent since 2019. Yet there is still a significant data science skills gap in the market and part of the problem is education focused. Research conducted by the University of California, Riverside shows that fewer than a third of the US News & World Report’s Top 100 Global Universities offer degrees in Data Science, and most programs are still taught at a graduate or PhD level.

The bottom line is that the growth in demand for data science skills currently outpaces the ability of academic institutions around the world to build data science programs. Although Data Science as a field has existed for several decades, the rapid growth of the last decade, the current skills shortage, and the interdisciplinary nature of the field, have contributed to the difficulty in building new programs.

To help academic institutions overcome these challenges, IBM - working with the University of Pennsylvania and the Linux Foundation - brought to the market a “starter set” of data science training materials launched as an open source project: Open Data Science for All (OpenDS4All). The goal of OpenDS4All is to accelerate the development of data science curricula at academic institutions, making it easier for academic institutions to leverage existing education modules built by professors for professors without the need to build every single course from scratch.

Starting a data science program from scratch is incredibly difficult, as building a curriculum requires significant resources. By making a “starter set” of training materials available containing sets of
PowerPoint slides and Jupyter Notebooks with Python code, we aim to help accelerate the availability of data science skills building programs around the world. By making it open source, we can leverage the open source communities for growth of the available educational content with contributions from other experts in the community.

Current and future content

There are currently 16 educational modules in the OpenDS4All repository on GitHub, and we are constantly striving to add new modules covering topics of immediate relevance and interest. The original content covers the core of data science and includes modules on:

- **Overview** (What is Data Science?)
- **Foundations** (How your computer works and graph theory)
- **Data and Knowledge Modeling** (Representing and codifying knowledge)
- **Data Wrangling and Integration** (Getting Started in Data Science: Data Acquisition and Wrangling)
- **Exploratory Data Analysis** (Information Visualization, aka Visual Analytics)
- **Machine Learning** (Building unsupervised and supervised machine learning models)
- **Model Assessment** (Training, validating and tuning robust models)
- **Scalable Data Processing** (Efficient and cluster-based processing with graph data)
- **Ethics** (Ethics, privacy and fairness in data and algorithms)

The latest module that has been added to the repository covers the basics of deep learning on a well-known image classification data set. It was designed to be a “soft introduction” to image processing using convolutional neural networks (CNNs).

With AI Trust being one of the most pertinent topics currently being discussed in AI and Data Science, two new educational modules based on two of the pillars of AI Trust are in the works. The first module under development is focused on AI Fairness (are certain groups at a systematic disadvantage compared to others) using information from the AI Fairness 360 (AIF360) extensible open source toolkit. The second module is on AI Explainability (the ability to provide a clear and relevant explanation of a model's decision) and is based on the AI Explainability 360 (AIX360) toolkit (also open source). We hope to have these modules incorporated into the repository in the not-too-distant future, and several professors have already expressed their interest in the content.

There are many new developments in Data Science such as the increase in use of natural language processing, federated learning, and automation to name a few. Contributions on any of these topics as well as other relevant topics are welcomed and will be reviewed by OpenDS4All’s technical steering committee (TSC) for inclusion into the repository.

Although the Jupyter Notebooks can be run in nearly any local and cloud environment, OpenDS4All provides access to a readily available binder environment so that faculty and students with limited computing resources can experiment with the notebooks. The binder link is available from the GitHub repository. This may be especially useful for students who do not have access to a local installation of Python or a virtual lab where they can execute the code.

Usage

The reception of the developed modules has been overwhelmingly positive, with many academics from all over the world commenting positively on the clear structure, engaging content, and applicability to undergraduate and graduate programs from all disciplines.

One of the most successful ways of promoting the content has been through bootcamps and workshops. These workshops and bootcamps are usually structured as follows:

- **Introduction to Data Science and Data Analytics and job opportunities (15 minutes)**
- **Presentation of one or two of the lectures from the repository (45 – 90 minutes)**
Demonstration and hands-on exercise on how to import a Jupyter notebook and how to run it at least one cloud environment (45 minutes)

Students and professors who attended these workshops and bootcamps loved the hands-on aspects of the modules, experimenting with Python code and running the Jupyter Notebooks, in particular those involving data wrangling, acquisition and web scraping. A good example is the bootcamp we did with the University of Liverpool in the United Kingdom during 2021 where 82 students attended, even though attendance was optional. This is written up as a success story that is available from the GitHub repository.

OpenDS4All has also been adopted as a core curriculum component of the National Student Data Corps. The National Student Data Corps (NSDC) is a community-developed initiative that teaches data science fundamentals to students across the United States, with a special focus on underserved institutions and students.

Lessons learned

During conversations with more than 50 academic institutions on OpenDS4All, we learned that universities are still trying to figure out the best way to teach Data Science. Questions that came up included:

- Should Data Science be taught at the undergraduate level or at the graduate level?
- Should programming in Python or R (as well as mathematics and statistics) be a prerequisite for doing Data Science?
- Should we cater for students preferring low-code environments?

We also underestimated the difficulty of incorporating OpenDS4All content into existing data science programs, due to the somewhat lengthy approval process for new courses. Many academics are also not familiar with open source, and as OpenDS4All is run as an open source software (OSS) project this represents a barrier to adoption.

Although the initial goal of OpenDS4All was to accelerate the development of (new) data science programs worldwide, we also learned that existing programs could benefit greatly from the content by supplementing existing content with some of the modules from the repository. For example, big data is becoming more pervasive, and existing data science programs can augment or enrich their content with novel content from the repository on scalable data processing with Apache Spark.

We also learned that language may be a barrier. To make OpenDS4All more inclusive, we have started translating the hosted educational modules into Spanish and Portuguese. These modules will be incorporated into OpenDS4All during the upcoming months.

How to become involved

There are many ways to interact with this repository:

- Browse and download content (PowerPoint slides, Jupyter notebooks, etc.)
- Contribute content (become a contributor to the project)
- Become involved in the day-to-day management of the project (become a maintainer or committer)
- Provide overall direction and leadership to the project (become a Technical Steering Committee member)

Please visit the GitHub repository at [https://github.com/odpi/OpenDS4All](https://github.com/odpi/OpenDS4All), fork it to download content, and star the repository to help us improve the content and keep it relevant.

Acknowledgements

Ana Echeverri, who designed and launched OpenDS4All, reached the final shortlist for the AI Innovator of the Year Awards at AI Summit London 2022 (Solutions Provider award) for OpenDS4All, confirming that OpenDS4All presents a novel and innovative approach to reducing the data science skills gap.

Zachary Ives, who designed much of the original content of OpenDS4All, received the IEEE 2022 IEEE TCDE Education Award for fundamental contributions to Data Science education.
ABSTRACT

The panel on data(base) education at VLDB2021 [13] drew attention to important challenges in choosing how database classes are constructed for students in a world where data is being used in novel and impactful settings. This paper aims to present one view of a process for making these pedagogy decisions. We don’t aim to present a best-possible design of the subject, rather we want to illuminate the space of possibilities, to encourage reasoned choices rather than simply teaching the subject as it was previously offered, or spending time on the latest innovations without considering the “opportunity cost” of doing so. We hope to guide the perplexed instructor or departmental curriculum committee.

1. THE CHALLENGE

Any computing education will include material about data management, but there are many alternative ways a class on this topic might be structured. There are so many topics already that seem valuable, and new ones are gaining importance, but a degree would not be long enough to learn it all. The standard textbooks like [30, 25, 12, 23, 20] are each over 800 pages. Class time and student attention are scarce resources. So there are hard choices facing the instructor or curriculum committee who want to decide what to teach, and how to teach it. If we include the alternative approaches to computing distributed joins, what would we leave out to make room? Which of the sophisticated aspects of SQL will be taught and assessed? Which (if any) normal forms are crucial? Which (if any) concurrency control mechanisms are covered? Should we replace XML by graph data and its query language? Do students need to know how to code insertion into a B+-tree, or to use an object-relational mapping framework? Is blockchain ready to be a core topic?

This paper aims to characterize some of the design space of data management teaching, and we advocate making principled decisions within the space. Our approach comes from constructive alignment, where the learning objectives are chosen (and announced to students), and then one picks activities for students to carry out, and especially the assessment tasks, so these will help the students achieve the intended outcomes. We go a step further, and argue that the choice of detailed learning outcomes, should be guided by an overall purpose for the class, which takes account of career paths students will be prepared for, and other wider curricular goals.

2. CHOOSE THE PURPOSE(S)

Why would a student take a data-related class, and why would a department offer it? Just as there is too much known about databases, to fit it all in one class (or even two), so there are too many topics in computer science, for a student to learn them all. One justification of devoting some of the scarce teaching attention to data management may simply be accreditation: some coverage of this is required for all students by standard curricula such as ACM Computer Science [7] and ACM Data Science [28]. But the curricula aren’t detailed enough determine the focus of a data-related subject.

For most faculty, their teaching is driven by the goal to contribute to students’ intellectual growth. This view would tend to favor using the available time for fewer topics in depth, rather than a broad and shallow introduction, but again, how should we choose which aspect to explore deeply? One consideration might be if there is a crucial style of thinking that is neglected elsewhere in the curriculum, and we can then pick the topics within data management that develop students’ aptitudes in this important degree-wide outcome. For example, the database subject has often served as an exemplar of larger-scale software design, teaching students the way a complex system can be structured in layers and components with clean interfaces. A quite different role for the database subject, can be as the main exposure for technically-oriented students, to awareness of business needs and organisational is-
sues. Coverage of normalisation can be a vehicle demonstrating the capacity of theory to have practical impact. In some curricula, SQL may be the best hope to awaken students to the power of declarative programming.

While faculty can take a larger view of their purpose, students often engage better when they see how the skills and knowledge contribute to their professional future. Many career paths are available from a computing education, and knowing which paths are most common for the students in a class can help identify which topics are most worth including in the limited time available for a subject. Even for the traditional relational database class, some topics were most natural for the future developer of database-backed applications, others for someone who will become a database administrator, and it is different again for an aspiring data analyst, or for the potential implementor of a platform. Now, with the spread of data science, there are extra paths such as data engineer (creating and tuning the analysis pipelines), or even the domain expert from other disciplines who will need to manage their own datasets and explore them. Because many smaller organisations can’t afford a whole team for data science, so a single student may graduate into a role where they have to clean and integrate datasets, write data analysis programs, install and tune data storage, explain the results, etc. As we discuss in the next section, the students’ intended path can shape the amount to content on different topics, as well as the approach taken to lectures and assessments. The choice of career focus in a subject may be influenced by local industry needs, and also by the alternatives offered nearby: for example, if a Business School is producing strong Information Systems graduates for the local region, then a CS department has less reason to stress data modelling for its students.

A final consideration at this level of the design process, is how the subject fits into the overall curriculum, and any institutional constraints. If a subject is counted in several majors, there is usually a struggle to ensure that it works equally well for the different cohorts. One important issue is the expected background knowledge. If a class is intended to accept data science students, we should probably assume less experience with computer organisation, and perhaps they haven’t knowledge of object-oriented programming. If all students have already learned first-order logic, the way we teach query language can exploit it; similarly conceptual data model topics might be abbreviated or eliminated when students already have done object-oriented domain models and UML.

Of course, some departments have the resources to offer more than one database course; this raises really interesting choices in how to divide the material. One common pattern is to have a lower division subject focused for application developers, and follow in higher years with a class on system internals, where the future database implementer, or data engineer is the target. At some institutions, there are different introductory classes in different study paths, for example, one with a business focus in the information systems curriculum, a more technical one for computer science, and/or one where data science students learn how to scale analytic tasks for data beyond the in-memory processing of their favorite language.

Our intention is not to argue that one purpose is better or worse, merely that the instructor needs to understand the purpose and ensure that the way they teach is well-matched to it. It is motivating for students if they understand why a subject is offered, and where different topics fit into their growth as a professional, providing both career skills and also more general aptitudes.

3. SELECT THE TOPICS AND DEPTH OF TREATMENT

There are strongly held views among faculty and employers, on what students need to know; unfortunately, there is insufficient time in a degree to meet all the requests for content. We look here at several of the main facets of the subject, within which choices must be made. In each case, we expect that any class will cover the concepts at some level, but there is large variation possible: for example, a learning outcome “Students will recognize when a schema involves redundancy and possible inconsistency in the data storage” could be covered in much less time (and with much less assumed math background) than “Students can use inference rules to deduce consequences of a set of functional and multi-valued dependencies”; in between might be “Students can produce a BCNF schema for a collection of attributes related by functional dependencies”. Deeper coverage of one direction will inevitably leave less time for another.

3.1 Diverse data models and data types

The clean, 1NF, relational model has a magnificent theory and is a joy to teach. It is naturally at the core of any data management class no matter the expected career path: much traditional enterprise data is structured this way, and many data science datasets come in csv format with a tabu-
lar structure, and are processed with dataframes for which the relational theory fits fairly well. But there are many other NoSQL platforms; at the least every graduate needs to know enough to have sensible conversations with enthusiasts for graph databases, document stores, and key-access column-family stores. In some settings, some students will need to work with, or develop the internals of, one of these style of system. Whether to merely show the concepts, or to give practical experience with systems, is a hard decision for the course designer, as reaching real understanding the strengths and weaknesses of each approach takes a lot of class time.

Future platform developers may well work on systems supporting these diverse models, but the main ideas of system internals seem likely to transfer well from relational systems, so these careers may not need much class focus on the new models. In contrast, data engineer and science careers are likely to need skills with stores that support these models, while for enterprise application developers the focus may instead be on how to shoehorn data like this into a relational representation. Many relational platforms have added support for columns declared as json or graph. Unfortunately, these extensions are often proprietary, and so education about the details may not transfer well across the diverse platforms students will need to use in later careers, and also the query facilities for a typed column are usually much weaker than in a dedicated store for the type. Thus, it is worth discussing how to shred the data into a standard 1NF relation, and write SQL queries to recover some useful information.

As well as json and graph, there are many significant data types beyond string, integer, float! Working with time and date data is crucial for any career path, and any likely data set of interest will include temporal aspects, so covering these seems inevitable. Students need to know the distinction between time instants and time periods, different ways of dealing with current data (a period whose end has not yet been determined), how to query the state as-of a given time, and a variety of temporal joins.

A class aimed at data science careers will also want good coverage of geo-spatial data. For example, in our university, the data science curriculum core class (DATA2001) presents conceptual material on spatial data. We stress the differences between points, regions, paths, and trajectories (and different approaches to represent them), and also the importance of the coordinate system and of topological relationships between spatial objects. We introduce some of the operations and functions which are meaningful on spatial types, especially drawing attention to the high query costs even when indices are present, and the distinction between spatial join and a join on non-spatial attributes. The students get practice in labs, working with both PostGIS and Python geopandas; their project work also involves spatial data (Section 4.2)

### 3.2 Query languages

No matter what career path students will follow, they should have learnt enough about SQL or a similar declarative way to express queries. Even for data science students who may not be keeping data in a relational platform, most of the concepts of where-conditions, joins, group-by, etc, are found in dataframe processing in Python, or in systems like Apache Hive above HDFS for distributed analytics. However, even standard SQL has much more complexity than the basic SELECT-FROM-WHERE-GROUPBY-HAVING, INSERT, DELETE, UPDATE, handling of NULL (including outer join variants), and nested subqueries, that all textbooks cover. Beyond the standard, each platform’s SQL variant has its own extensions and restrictions. An instructor needs to decide how much of this students see, and how much of that to stress in assessment. Future application developers will need to know many of the tricks covered in books like [5], such as how to do running totals or recursive aggregation for bill-of-materials; but these may not be important for other careers. Other parts of SQL such as table declaration and modification, constraints, triggers and access control, are often introduced to various depth, in connection with topics on schema design, security etc.

Data science frameworks like pandas have many of the concepts taught in SQL, such as filtering by a boolean predicate, grouping, joins. While the style of code, with an explicitly sequence of operators, is less declarative than SQL, it is close enough that students seem likely to transfer SQL skills to these frameworks without difficulty. Alternatively, students can be taught to program with a dataframe library directly, either a standard one or a simplified one for novices, as done in the famous Berkeley Data8 class [1] see [http://data8.org](http://data8.org). Another category of query language topics is for non-relational data models and advanced data types, as mentioned above.

What of the theoretical query languages? Relational algebra is a powerful notation for discussing even the simplest ideas of query processing, so we expect most classes to introduce this. It is hard to see career utility for relational calculus, but this
may be valued by instructors who want to stress the connections between theory and practice (for example, it can motivate some of the SQL tricks for universal quantifiers via nested NOT EXISTS clauses), or as a way to teach some logic in a curriculum where other subjects do not.

3.3 Internals

In many departments, the database class is presented as a systems experience, with lots of coverage of the algorithms and structures inside the engine, such as indexing, query processing and optimization, transaction management, replication. For students who may work as platform developers, database administrators, data engineers, this is vital content, and it may be covered as a way of building general skills in this direction (if the rest of the curriculum does not do so). However, for students aiming to become application developers, this material may feel unmotivated; also it usually requires knowledge of low-level coding (e.g., C or C++), a skill which may not be produced in previous subjects. In our experience, one can find a nice subset of topics about internals, which are meaningful for application programmer careers, by looking for cases where the user’s satisfaction is shaped by choices available at the application. For example, declaring an index can greatly affect application performance. The choice of class orientation shapes how time is spent and especially how the students are assessed. Teaching index structures for platform developers may lead to exam questions to trace the execution of an insertion, or estimate storage used in a structure. But for application developers, one might ask which index to create to improve performance of a given workload.

Concurrency control and recovery algorithms are sometimes excluded entirely from a first data class with a focus for application developers (e.g. in [34]). However, some knowledge of transaction concepts must be provided for these students, because enforcing isolation impacts the tradeoff between integrity and throughput [9]. We believe that a crucial skill for application coders, is identifying what steps need to be combined into a transaction. Developers need to be aware of the kinds of integrity violation that can arise when their code runs at default isolation (which is Read Committed rather than Serializable, in most engines).

Data science work often does not use a platform where an index or isolation level can be set declaratively, and where query optimization is done by the engine; instead the programmer will code support for these aspects in the application. For these careers, it will be important to know at least the simplest techniques and how they influence performance at scale. Replica coordination and distributed processing are of particular importance in a class for these students.

3.4 Application structure

If a class is a path to application developer careers, it needs to give students awareness and practice of some ways that a data-backed application can be coded and deployed. An important skill for these graduates will be to recognize the tradeoffs between different architectural approaches, such as the explicit writing of SQL or having it generated through an ORM such as Rails, separation of the code into tiers, placement of tiers across a network (including whether to use cloud-hosting, and differences between self-managed databases versus cloud-provided data services), and where and how to maintain state. This is a topic where the technology changes rapidly (and often a programmer’s options are constrained by organisational imperatives). This makes it hard to choose core concepts that will still hold their value for the students when they reach the workforce. It is a challenge to find teaching resources, as textbooks are often obsolete, and online resources are rarely written with a broad and platform-independent viewpoint. It may be that the best approach is to provide students with a sample codebase to explore, and to discuss some alternatives as a way to model the architecture decision.

3.5 Conceptual models and schema design

Learning to create an E-R diagram for a domain, and how to achieve normal forms for schema, have long been core topics in a database class. The relevance to many future enterprise-centred careers for these seems hard to see in current times, when little work is done in a greenfields setting; rather, application developers and DBAs and data scientists typically work with data whose storage and schema are already established. One approach we have found useful, is to motivate in the context of expanding an existing design to include extra data for extra functionality. Normalisation can also be a wonderful topic for building students skills with logic and discrete math, though perhaps one might limit discussion to one or two normal forms.

In a class for data science, there are topics related to logical design that are important, but are not part of classic relational design. One issue that is crucial for working with charting libraries, and machine learning, is the different ways to treat related
attributes. In standard theory, attribute names are atomic, but real-world data often arrives with attributes whose name includes the year, location, category etc (e.g. "temperature-Sydney" or "responses-2020"). For a dataframe, there are alternatives between wide schema (with a family of related attribute names) or long schema (where the varying aspect is made a separate attribute), and libraries like pandas have functions to transform data between these formats.

3.6 Security, integrity, and resilience

The traditional database curriculum includes a scattering of topics such as views, access control, integrity constraints, triggers, recovery, SQL injection threats; more recently, differential privacy and blockchain have emerged as headline topics. These can feel to students like minor and unrelated technical details. We suggest that they can be motivated when brought together under the heading of data quality and use, and combined with discussion of the ethical decisions. One important distinction when teaching students whose career may lie in data science, is that the management of these properties is not typically provided by a platform, but instead needs careful implementation in code and file settings, etc.

4. CREATE LEARNING EXPERIENCES

Pedagogy theory tells us that active learning is effective. So even a traditional class with content-delivery lectures, will include time for tasks where students perform actively, perhaps during small group discussions with a TA, or as homework. The choice of which activities to assign, has a huge impact on what the students learn in the subject. We next discuss two categories of active tasks: writing queries, and completing a semester-scale project.

4.1 Query writing

As mentioned above, a first data management subject will teach SQL or a similar declarative approach to queries. As when mastering any programming skill, there is no effective substitute for hands-on experience. A very common style of task which students are told to do for practice, and for assessment, is to write the SQL to calculate some result, given the natural language description of the purpose. For example, “Here is a schema . . . Write a SQL query to show the employees by name, and their department, where the employee’s salary is more than 10% of the highest salary paid in the department.”

It is natural to set query-writing tasks in domains that are relevant to students, and also where they have background knowledge to help them understand the data and the intention behind each query. So a remarkably common choice is a student-record domain, with tables of student, subject, grade, instructor, etc. However, a class centered on business applications can use tables like Employer, Customer, etc; or data science students may work with real datasets (weather, census, movies, etc).

For effective pedagogy, the query-writing tasks should be ordered over at least several weeks, so that students are gradually exposed to more complex SQL constructs and concepts, after they have mastered easier ideas. Experiences from teaching programming shows that combining constructs is itself a complex step, and is better done after the students are comfortable with each of the sub-techniques being combined. So, one should give practice with simple aggregates over a table, and practice with simple filtering of a table to find the rows that satisfy a condition, before students are asked to write queries that aggregate over a subset of rows. Natural joins, and select from a single table, should both come before queries that join and also select. There are however constructs which seem independent of each other, and whose comparative difficulty is not so different, and thus there is a choice in the ordering. Should aggregation over a table come before, or after, selecting rows from a table based on a simple condition? What is the better sequence, for learning correlated subqueries and grouping?

A central challenge in giving students programming practice, is how to provide feedback. Rapid indication of their mistakes is invaluable, and students also gain from helpful advice to undo any misconceptions revealed by their errors. In smaller classes, a teaching assistant can provide this face-to-face, but support from teaching software has advantages of accessibility on-demand, allows students to practice as much as they wish, and scales well as student numbers grow. We now discuss some of the types of tool that have been proposed in both database and computing education communities.

Early work [22, 27] was based on the concept of an Intelligent Tutoring System which takes the student through a sequence of tasks for which they write SQL queries. It provides guidance, with error messages that help a novice understand why their answer is wrong (e.g. “This task requires a GROUP BY clause” or “you do not mention the correct attribute”). The next task given to a user is often personalized based on a learner model, built up from the previous mistakes and correct answers, that reflects which topics need practice. Another kind of
tool derives from algorithm animation. Here the goal is to help a student develop their understanding of SQL semantics, but showing step-by-step, how the output of a query is determined from the database contents [15]. The pedagogy assumption is that students will write queries more sensibly, if they understand the “notional machine” and therefore can predict what a query text is calculating. A similar system https://pandastutor.com has been produced for pandas dataframe manipulations.

Another type of tool, especially driven by MOOCs, has built systems for automated grading of SQL queries. These can check a query by comparing the output to what is required; often the query is run against a variety of database states, to pick up cases where an incorrect formulation happens to give the same answer in a particular state [18, 26]. Our experience though is that it is actually quite hard to phrase a question so the meaning is unambiguous, and no reasonable interpretation of the wording will be marked incorrect. An interesting proposal [8] combines automated checks with peer review. Some systems such as Ullman’s Gradiance https://www.gradiance.com/STwelcomeAWS.html generate personalised question wording for each student (but testing the same constructs), by filling in a template with randomly chosen elements. An interesting recent enhancement of grading systems has been to automatically generate input database instances, specifically chosen so that mutated queries will give different output than the correct query text [4].

4.2 Project work

Due to strong practical relevance, it is widespread that each computing class (except the most theoretical) will have a significant project as a large component in the assessment. For a data-management class, there are two quite different ways this happens: project work in the platform implementation, or project work in building a data-backed application. While some very intensive subjects may give both projects, more often, an instructor chooses between these directions, and the decision naturally would impact the overall career focus of the subject. However, there are plenty of axes of choice within each style of project.

For a project looking at platform internals, it is usual to work with an existing platform, and in a sequence of tasks, students either rewrite some component, or extend that component (e.g. provide an alternative page replacement policy, or include a new index type or join algorithm). This style of project was famous from the Carey and Ramakrishnan Minibase code from Wisconsin, written in C++ and supporting a subset of SQL; this was provided with [25] (and inspired by an older Minirel system from Dave DeWitt). Alternative simple pedagogic platforms have been produced in different languages, with different query interfaces [24, 32, 29]. Ailamaki and Hellerstein [2] argued for instead having project work that uses a full-power open-source DBMS such as PostgreSQL. Broadly speaking, projects based on a simplified pedagogic platform pose less cognitive load, with cleaner interfaces (and less interdependency) between components, but they can be less motivating, and also they provide less preparation for future internship and career activities. A different approach has students implement all components of a simple dbms from scratch [31].

With an application development project, one essential difference is between a greenfields work, doing the whole application from scratch, versus extending a code-base that is provided to the students (for example, creating code for additional use-cases, which involve modifying the schema, creating appropriate extra tables and indices, and writing new transactions). In either case, there are choices for an instructor, in the technology used for the development. There are of course different options for the data management platform, and also choices for the application development tools, such as an object-relational mapping layer, a focus on putting the business logic in stored procedures, or use of explicit code working with cursors. The instructor also needs to decide which stages of the life-cycle to emphasise in a project. If the class has a focus to application development careers, more emphasis is often given to requirements elicitation; while educating future DBAs might devote significant work for conceptual modelling and schema improvement, then physical design.

What would be suitable project work in a class where data science is an important focus? It is likely to focus more on complex analytics, and less on transactions and updates. Also, a lot depends on the decisions about specific content topics that are covered. For example, if the class has looked at a variety of data types, then it is natural to include those in the project work. Here is one way, that we use at our institution. Our Data Science major contains a (compulsory) second-year course on Big Data and Data Variety (DATA2001). A central part of this course is a project-based assignment where students develop a complete Data Science pipeline from data ingestion, data transformation and cleaning, to data analysis and visualisation. We provide students with a real-world scenario including
a spatial-temporal dataset which students have to combine with other datasets, typically at least one CSV and some JSON/XML data via a Web API. The scenarios are inspired by local circumstances, so for example in our case a bushfire risk analysis for Greater Sydney, or the computation of a cyclability score for different suburbs of the city. In order to teach students the declarative power of SQL, and also the scalability benefit of appropriate indexes, these scenarios cover at least one spatial join. Students develop their pipeline with Jupiter notebooks in Python, but are explicitly encouraged to keep the spatial data in PostgreSQL and to use its PostGIS extension for the spatial join(s). With a spatial index, this approach is typically much more scalable than using a pure in-memory Python implementation with GeoPandas.

As explained in Section 3.2, a background on databases and declarative data processing with SQL is also very helpful for understanding the challenges of big data processing. We found this enables us teach high-achieving students to scale-out their data analysis using distributed data processing platforms, such as Spark, and to gain some first experiences with scalability evaluations.

Whichever choices are made, it is vital that the project work be well supported, as any problems with server capacity, security restrictions on access, etc, will distract the students from the learning goals, and also add a lot of unnecessary time demands on them.

5. RELATED WORK

The central conferences of our community such as SIGMOD, VLDB, do sometimes include work about teaching [33, 4, 26]. Another source for ideas and information comes from the Computing Education community, and its main conferences such as SIGCSE and ITiCSE. Some papers here are in the style of education research, but the majority are experience papers that contribute to a “community of practice”, where the author describes an innovation in teaching, and reflects on lessons learned that others could benefit from. Papers about database and data science education have appeared regularly in these forums. Here we draw attention to a few recent ones, beyond those we already cited. In [10] we show one way data management concepts can be taught in a sequence of subjects in a data science major. Including NoSQL concepts into a database class is discussed [16]. Using data about errors made in student tasks, SQL language topics were characterised [21], and similarly for document store queries [3] and graph data queries [6]. A tool was created for grading data designs (in UML) [11]. The impact of automated assessment tools on student learning has been studied [17]. A flipped approach to teaching an upper-division class on internals was evaluated [19], as was use of group exams in an introductory class [14].

6. REFERENCES


[31] B. Sotomayor and A. Shaw. chidb: Building a


“How would you archive databases for the next 60 years such that they incur no migration cost, and they remain usable in 2080?” This was an open challenge raised by digital preservation experts from the Landesarchiv of Baden-Württemberg [12], who, similar to other memory institutions (archives, museums, libraries, etc.), have faced several challenges in archiving culturally significant, historic data stored in digital databases since early 1960s.

On the hardware front, all current media technologies suffer from media decay and have a limited lifetime of few decades at best. Further, current media technologies tightly couple the storage medium, with the technology to read data off the medium. This leads to media obsolescence, where data stored in an older medium is no longer readable by new readers. Memory institutions, in contrast, are tasked with preserving databases for much longer duration requiring expensive, periodic data migration. On the software front, database engines store data in proprietary file formats that evolve rapidly. This leads to format obsolescence, as data archived in an engine’s native format becomes unreadable by even a newer version of the same engine. In addition to archiving data, it is also necessary to archive the application logic expressed in stored procedures, SQL queries, and views, as they provide the context in which data is accessed. Unfortunately, even SQL statement archiving is not standardized due to deviations from ANSI/ISO SQL caused by vendor-specific extensions [15].

These challenges are not specific to memory institutions; the growth of data fueled by AI and BI, coupled with new compliance requirements, has created the perfect breeding ground for long-term archival challenges in today’s data-driven enterprises. It is time to revisit the archival hardware–software stack, especially given recent advances in the development of storage technologies based on novel media. One such medium that has received a lot of attention recently is synthetic DNA [2, 4, 7, 9, 14].

DNA possesses several advantages over current media. First, its theoretical density is at least eight orders of magnitude higher than contemporary magnetic media [5]. Second, DNA is very durable and can last millennia at ambient temperature [6]. Third, DNA decouples the medium (biological molecules) from read technology (sequencing). Thus, data stored in DNA can be left untouched without migration as it does not suffer from media decay or obsolescence.

Designing read and write pipelines for DNA-based database archival opens up several opportunities for data management research. For instance, in project OligoArchive [13], we are developing database-aware encoding [2], decoding [10], and bootstrap [1] techniques for reliably archiving databases on DNA [11]. In addition to these challenges, several others, like designing indexed access paths over DNA [14], exploiting rather than masking DNA errors by using it as an approximate storage medium [8], or enabling near-molecule computation by executing search [3] or query [2] operations using biochemical reactions over DNA, remain open for further research.

While DNA solves issues at the medium level, it does not solve format obsolescence issues. Digital preservation experts have standardized text-based, archival file formats (SIARD [15]) and rely on exporting data out of a database into a software independent archival file to overcome format obsolescence. However, text-based archival leads to data bloat, and tooling support is limited to a few relational databases. In contrast, the rise of open-source binary file formats like Arrow and Parquet is a step towards eliminating format obsolescence. But more work is required to understand their conformance to SQL and their ability to archive non-relational data and contextual application logic.

With archival problems affecting memory institutions and enterprises alike, it is time for database vendors and researchers to tackle the Landesarchiv challenge with solutions and standards that can make low-cost, migration-free data archival feasible.
REFERENCES

The main goal of the Social Technology and Research Laboratory (STAR Lab) in the University of Hong Kong (https://star.hku.hk) is to develop novel IT technologies for serving the society. Our team has more than three years of experience in project development, web, app, and game design, photography, and video production. We are interested in “Data Science for Social Good”, researching data-driven approaches that can benefit the public, NGOs, and the government.

As of Fall 2021, the STAR lab is comprised of four professors, five postdoc researchers, ten PhD students, and more than twenty software developers. The lab are working on different aspects towards “Data Science for Social Good”:

1. research – interdisciplinary research on data and social science;
2. technology – developing technologies for social services;
3. synergy – connecting with the public, NGOs, and the government;
4. teaching – educating NGOs and the public on social technologies; and
5. impact – bringing benefit to the public, NGOs, and the government.

To achieve the above goals, the STAR lab has been working on two frontiers: fundamental research (Section 2) and social and legal applications (Section 3). For fundamental research, we tackle the challenge of the huge volume and complexity of graph data, and develop efficient, scalable, and efficient algorithms on different kinds of graphs. We also develop novel graph-based recommender systems. We collaborate with universities including the University of Illinois at Urbana-Champaign and the University of British Columbia on these efforts. As for social and legal applications, we have been collaborating with more than 20 organizations in universities, governments, NGOs, and commercial organizations. The lab has acquired more than HKD $35M funding. We have recently received a SIGMOD Research Highlights Award, two industry awards, and one university knowledge-exchange award. A PhD graduate of the lab has been selected by Baidu Scholar as one of the 2021 Global Top 100 Chinese Rising Stars in Artificial Intelligence.

2. FUNDAMENTAL RESEARCH

Graph data are prevalent in different social applications. For example, as we will discuss in Section 3.1, one of our projects, called HINCare, utilize big graph technologies to facilitate the matching of volunteers to elders. Hence, the STAR lab has been recently engaged in various of fundamental research activities in graph mining, with the goal of advancing “data science in social good”. In Section 2.1, we will discuss the main work that we have done in the past few years, namely densest subgraph discovery, motif analysis, and community search. We then describe the research problems we address regarding the heterogeneous information networks (HINs), in Section 2.2. These work can be useful for social applications, as we will describe in Section 3.

2.1 Big graph technologies

1. Densest subgraph discovery is fundamental to a wide range of applications, such as fraud
detection, community mining, and graph compression. We have examined this problem on undirected and directed graphs, as described below.

- **Undirected densest subgraph (UDS):** Given an undirected graph $G$, UDS aims to find a subgraph $D$ of $G$ with the highest density (e.g., the number of edges over the number of vertices in $D$). Because UDS is difficult to solve, we propose a new solution paradigm [10]. Our main observation is that the densest subgraph can be accurately found through a $k$-core (a kind of dense subgraph of $G$), with theoretical guarantees. Based on this intuition, we develop efficient exact and approximation solutions for UDS. Moreover, our solutions support a wide range of graph density definitions, including clique-based and general pattern-based density. Extensive experimental evaluation shows that our algorithms are up to four orders of magnitude faster than existing approaches.

- **Directed densest subgraph (DDS):** Given a directed graph $G$, DDS refers to the finding of a subgraph from $G$, whose “directed density” is the highest among all the subgraphs of $G$. Essentially, we aim to find two sets of vertices, $S^*$ and $T^*$, from $G$, where (1) vertices in $S^*$ have a large proportion of outgoing edges to those in $T^*$, and (2) vertices in $T^*$ receive a large proportion of edges from those in $S^*$. Existing DDS solutions suffer from efficiency and scalability problems. Hence, we develop an efficient and scalable DDS solution [19]. We introduce the notion of $[x, y]$-core, which is a dense subgraph for $G$, and show that the densest subgraph can be accurately located through the $[x, y]$-core with theoretical guarantees. Extensive experiments show that our proposed solutions are up to six orders of magnitude faster than the state-of-the-art. This year, we have a follow-up DDS work based on convex programming [18].

Our work in DDS [20] received the SIGMOD Research Highlight Award 2021, and its journal extension has been recently accepted by TODS as one of the Best of SIGMOD 2020 papers [21]. Recently, we have been collaborating with the HK Applied Science and Technology Research Institute on using the densest subgraphs found on their user-website-click graphs to find fraudulent clicks. We also plan to extract densest subgraphs from the volunteer-elderly graph (details in Section 3.1), and examine how these subgraphs can be used to provide recommendations of volunteers to elders.

2. **Motif-based graph analysis.** This kind of analysis has recently emerged as an important tool for discovering insight from graphs, e.g., biological and social networks. A motif, or a small graph with a few nodes, is a fundamental building block of large and complex networks [17, 14, 15, 16]. Motif-based graph analysis enables “higher-order semantics” analysis, and performs better than traditional “edge-based” solutions in a range of graph analytics tasks, such as link prediction [16], graph clustering [15], and node ranking [14]. These tasks are often important for extracting insights and patterns from the graphs collected in our lab, as well as predicting user behaviors. We studied two problems about motifs, and developed a system prototype, as detailed below.

- **Counting motifs on uncertain graphs.** Given a graph $G$ and a pattern $m$, e.g., a “triangle”, a fundamental task in motif analysis is to count the number of instances (or frequency) of $m$ on $G$; if $m$ occurs frequently, then $m$ can be considered as a motif of $G$. Motif counting enables the understanding of the characteristics of the graph, and also the discovery of the right motifs for motif-based graph analysis and visualization. We recently examine how to count motifs on uncertain graphs, whose edges exist probabilistically [17]. Although researchers have developed several fast motif counting solutions, they assume that graphs are deterministic, i.e., the graph edges are certain to exist. However, this assumption may not always hold. For example, in a social network, a link between two nodes (representing two users), which represent the friendship between these users, may only exist probabilistically. Ignoring this issues can lead to a wrong counting result, and affect motif-based graph analysis. We propose a solution framework, called LINC, which can be used by existing deterministic motif counting algorithms. Extensive experiments on real datasets show that LINC is more effective and efficient than existing motif-counting solutions for uncertain graphs.

- **Motif-paths.** We propose a new notion of motifs, known as motif-path, which is conceptually a concatenation of one or more motif instances between two given nodes on $G$. We use motif-paths to develop algorithms for three graph mining tasks, namely link prediction, local graph clustering and node ranking [14]. These tasks are important to the analysis of the graphs collected in our lab, and enables the performance of various tasks (e.g., graph integration and cleaning). Our experiments show that motif-paths are more effective than traditional motif-based analysis and “path-based solutions”, i.e.,
those that use shortest path distance as a dissimilarity metric. We also develop a motif-path-based drug analysis algorithm based on a COVID-19 knowledge graph [16]. The algorithm can be used to trace the origins of COVID-19 variant strains.

- **System prototype.** Existing graph database systems are not designed to support queries that involve motifs. We develop M-Cypher, which is a system prototype designed to enable expression and execution of motif-related queries through a user-friendly graph query language [15].

3. Community search. A fundamental component of big graphs is the network community. Essentially, a community is a group of vertices which are densely connected. Community retrieval can be used in many real applications, such as event organization, friend recommendation, and network analysis. How to effectively and efficiently find high-quality communities from big graphs is an important research topic in the era of big data. A large group of research works, called community search (CS), have been proposed, which aim to provide efficient solutions for searching high-quality communities from large networks in real time. Nevertheless, earlier CS solutions mainly focused on simple undirected graphs, so they could not be applied to perform CS on more complicated graphs such as attributed graphs, directed graphs, and Heterogeneous Information Networks (HINs).

To overcome the above limitations, we extensively study the problems of CS over these graphs. Specifically, for attributed graphs, we consider several kinds of vertex attributes such as keywords, spatial locations, and profile information, and for each of them, we formulate novel models of communities by considering both link relationship and vertex attributes, propose efficient CS solutions, and experimentally evaluate them on real large attributed graphs [5, 4, 2]. For directed graphs, we formulate a novel community model by carefully considering the vertices’ in-degrees and out-degrees, develop efficient both online and index-based CS algorithms, and evaluate the proposed solutions on billion-scale directed graphs [8, 3]. For HINs, we propose three community models and develop efficient algorithms to perform CS [9]. To the best of our knowledge, our works of CS on keyword-based attributed graphs [5], directed graphs [8], and HINs [9] are the first works in these problems.

We also develop a system, called C-Explorer [6], to assist users in extracting, visualizing, and analyzing communities. C-Explorer implements several state-of-the-art CS and community detection solutions, and various functions for analyzing the effectiveness of the communities of these solutions. Recently, we conduct a survey of existing CS works, compare the quality of communities with different cohesive subgraph models, and point out promising research directions [7].

2.2 Heterogeneous information networks

1. Meta path discovery. A heterogeneous information network (HIN) is a graph whose nodes and edges are tagged with “type labels” to express their meanings [13, 28, 25, 22]. Given two HIN nodes s and t, and a set S of example node pairs (e.g., pairs of nodes representing celebrity star couples), in [22] we developed a machine-learning model for discovering meta paths [25], which is essentially a sequence of node types and edge types that characterize the important relationships between node pairs in S. The meta paths found can be used to support graph-based applications such as friend search, product recommendation, anomaly detection, and graph clustering. More recently, we developed efficient algorithms for discovering the k most important meta paths in real-time, based on the occurrence frequency and rarity of meta-paths [28].

2. Meta structure relevance. An important problem in HIN is the computation of closeness, or relevance, between two HIN objects. We propose to use meta structure, which is a directed acyclic graph of object types connected by edge types, to measure the proximity between objects in [13]. The strength of meta structure is that it can describe complex relationships between two HIN objects (e.g., two papers in DBLP share the same authors and topics). We develop three relevance measures based on meta structure, and an efficient algorithm proposed to support the relevance evaluation.

3. Web query recommendation. A web query is a string of keywords posted by users for finding information in the Internet. Typically, web search engines provide alternative query formulations, which can be more articulate and interesting to users. A long-tail query is an uncommon request that rarely occurs in query logs. Traditional approaches, which rely solely on query logs, could perform poorly on long-tail queries because they rarely occur in query logs. However, it is relatively easy to extract HIN entities from long-tail queries. We have studied how to utilize HIN entity relationship information effectively to provide a recommendation solution for long-tail queries. In [11, 12], we study the use of meta-paths, a form of HIN-entity relationships, for query recommendation. Next, we will examine the use of HIN-based algorithms in social applications.
3. SOCIAL APPLICATIONS

We now discuss three social-related projects being done in the STAR lab: (1) data-driven recommendation for elderly care; (2) big data analytics for social services; and (3) legal data analysis.

3.1 Data-driven elderly care

Many metropolitan cities are facing sharp increase in aging population. In Hong Kong, for instance, the number of elderly citizens is estimated to rise to one third of the population, or 2.37 million, in year 2037. About 13% and 24% of these people are living alone or with their spouses only respectively. As they age and become more frail, the demand for formal support services will increase exponentially in the coming years. However, there is a severe lack of manpower to meet these needs: in HK, on average, each NGO employee needs to manage 10 elderly people at the same time. Some elderly-care homes also reported a 70% shortage in staff. There is thus a strong need of helpers for taking care of elderly people on a full-time, part-time, or voluntary basis.

We have been working on HINCare, a HKD $4M project supported by HK Innovation and Technology Commission. The HINCare is a volunteer management system with timebanking facilities (i.e., each person, after providing a service, can earn a time credit. The time credits are stored in the person’s time bank account. He/she can later use the earned time credits to purchase other services.) We have designed an elderly-user-friendly mobile app. The system backend, designed for NGO administrators, is cloud-based and generic. Essentially, any organization can use our system to support their voluntary work services easily. Instead of having sophisticated software installation, only a few customization steps are needed. The platform supports multiple organizations, which can enable more sharing of data and collaboration. Currently, HINCare has been serving 5000 elders in 6 NGOs. We won one local (HKICT) and one international information technology awards (Asia Smart Apps) and a HKU Faculty Knowledge Exchange Award. Recently, the HK government’s Community Investment and Inclusion Fund (CIIF) has provided 4 contract research projects to our team to further support their associated NGOs.

**HIN-based matching.** The core of HINCare employs novel heterogeneous information network (HIN) and AI technologies to recommend helpers to elders. Here, the HIN stores the relationship information among elders, helpers, and NGOs. It originates from various Big Data sources, such as social networks and senior citizen’s profiles. We use the HIN to find out the best helpers for assisting elders. For example, a living-alone elder may want someone to repair a light bulb; the HIN reveals that a certain helper living close to the elder has the expertise and availability to do so, and the system will recommend the helper to the elder. In detail, we use the HIN built to develop a recommender system [1, 27, 24]. These meta-path-based solutions leverage semantic relationship among graph nodes. They provide rich information of interaction among users and items, and help to comprehend a user’s interest. We remark that this is the first time that HIN is used to support elderly care. Experiment results tested on two collected NGO datasets show good performance compared with other systems.

3.2 Big data analytics for social services

Applying big data and artificial intelligence in behavioral and social science are promising but limited currently [23]. Family is the core in shaping individual behavior and influencing social capital, particularly in Chinese societies. Family services provide an important source to understand the influences of family on individual and society by using the large volume of data regularly collected in territory-wide family services, including the information on service users, groups and programmes, and counselling case recordings. These data are multi-dimensional, containing text, numeric, audio, or video forms.

The Hong Kong Jockey Club SMART Family-Link (JCSFL) Project, initiated and funded by the Hong Kong Jockey Club Charities Trust in 2018, is a large scale (HKD $80M), 4-year cross-sectoral collaboration among (1) the School of Public Health, and (2) Technology-Enriched Learning Initiative of HKU, and (3) STAR lab, with 26 Integrated Family Service Centers and Integrated Service Centers for advance the use of Information and Communications Technology and big data analytics for enhancing family services in Hong Kong. Being the first in Chinese population, we are collaborating with family services providers to analyze aggregated anonymous data of a large number of IFSCs/ISCs. The findings will be useful for informing family services and policy in the future.

**i-Connect.** A key element of JCSFL is i-Connect – a software service management platform developed by the STAR lab to support the service operations of the NGO-operated IFSCs. The system enables the users, who are social workers and clerical staff in IFSCs, to perform their operational pro-
cesses and workflow on a centralised platform. Different kinds of data about clients, staff, services, workflow, and operations are processed and stored through this system. The system has been deployed on a secure cloud platform. Industrial standards and various security measures are also adopted to ensure data security and data privacy. The system was reviewed by different professional parties including Office of the Government Chief Information Officer of Hong Kong SAR, the Information Security Team of Hong Kong Jockey Club, and a third-party security audit company.

3.3 Legal data analysis in collaboration with LawTech Centre

We collaborate with HKU Law Faculty in joint projects that study the problems of machine-assisted extraction and modeling of legal knowledge from legal texts, leveraging domain knowledge provided by law experts. Our knowledge models have led to the development of a number of essential legal applications that facilitate legal studies and research. For example, we developed a prediction model for illegal drug trafficking sentencing, which can be accessed online by the public (http://wwwnew2.hklii.hk/predictor). The sentencing predictor is used by some NGOs in youth education and crime prevention programs. Moreover, our sentencing prediction model helps address a number of interesting issues in legal information processing, which include judgment recommendation, fairness and explainability in machine predictions [26]. We also develop an AI-driven conversational system that helps people who have not received any legal training to effectively locate relevant legal information on our Community Legal Information Centre (CLIC) website. CLIC is an online information source covering 32 legal topics with contents such as FAQs, reading guides and explanatory notes on illustrative court cases, and short videos with hypothetical illustrative stories. The CLIC project is part of our continuing efforts in promoting free public legal education.

4. EDUCATION AND ENGAGEMENT

The STAR lab has dedicated a lot of effort in education. We have been producing video clips and conducting seminars, in order to train social workers to use our systems. We have organized press conferences and international symposiums for sharing our knowledge and experience with the public. Some materials related to the projects have also been used in courses taught by the lab leader, to educate students about how data science can be used in the social domain. We plan to recruit students to assist NGOs in our projects, in order to enrich their social awareness and practical experience.

5. BUILDING THE LAB

Finally, we share our experience of building the capabilities of our lab. We have built a core team of professors, who have worked with each other, and have expertise in data science, gerontology, and social science. We assembled a competent software development team, through fundings provided by government and charity organizations. It is very important for the NGOs to participate actively and provide their data for our projects. This requires a huge effort in understanding their needs and establishing trust. We have also organized public seminars, conducted interviews in newspapers and radio stations, and participated in exhibitions.

Acknowledgement

This research is supported by the University of Hong Kong (Projects 104005858, 104005994, KE award 2100048), HKU Innovation Wing Two Research Fund, the Innovation and Technology Commission of Hong Kong (ITF project MRP/029/18), the Hong Kong Jockey Club Charities Trust (HKJC project no. 2018-0025-001), and HKU-TCL Joint Research Center for Artificial Intelligence (Project no. 200009430). This research is supported by the WYNG Foundation; the Innovation and Technology Commission fund ITS/234/20; and The University of Hong Kong Knowledge Exchange Fund KE-ID-2018/19-21. Yixiang Fang was supported by NSFC under Grant 62102341 and CUHK-SZ grant UDF01002139. The Hong Kong Jockey Club Smart Family-Link Project was funded by the Hong Kong Jockey Club Charities Trust.

6. REFERENCES


Diversity and Inclusion Activities in Database Conferences: A 2021 Report

https://dbdni.github.io


1. THE D&I@DB INITIATIVE

Diversity and Inclusion (D&I) are core to fostering innovative thinking. Existing theories demonstrate that to facilitate inclusion, multiple types of exclusionary dynamics, such as self-segregation, communication apprehension, and stereotyping and stigmatizing, must be overcome [11]. A diverse group of people tends to surface different perspectives, which help to understand and address D&I. Fostering D&I in research communities must address issues related to inclusive interpersonal and small group dynamics, rules and codes of conduct, increasing diversity in under-represented groups and disciplines, and organizing D&I events, and long-term efforts to champion change [15].

Why D&I@DB? It is our responsibility as a community to [ensure that attendees of DB conferences feel included irrespective of their scientific perspective and personal background.] The D&I team is dedicated to providing leadership to help our community achieve that goal by [advising D&I chairs at individual DB conferences, serve as a memory and buffer between DB conferences, build an agreed-upon vision, and commit to working together to devise a set of measures for achieving diversity and inclusion.] That is pursued via actions led by our core members (Figure 1): REACH OUT collects data and experiences from our community. INCLUDE monitors and recommends inclusion efforts. ORGANIZE focuses on in-conference organization efforts, such as adopting a code of conduct. INFORM communicates through various channels. SUPPORT coordinates D&I support from executive bodies and sponsors. SCOUT collates D&I efforts from other communities. COORDINATE manages all actions. Everyone in the DB community is welcome to join the initiative.

What did we achieve this year? [This year’s focus was on raising awareness through the introduction of special D&I sessions and collecting statistics and wishes through surveys. Some active actions were also taken at some of our conferences, i.e., making deliberate efforts to address diversity and gender balance of conference officers, student volunteers, and invited speakers, and drafting and displaying a Code of Conduct.] We also published tips on inclusive writing and on video captioning and developed and deployed tools that facilitate our D&I efforts.

D&I statistics. Table 1 reports the results of a voluntary survey made available to conference attendees. In 2022, we will advertise the survey earlier and link it to registration (as in MDM) to make it available to all.

<table>
<thead>
<tr>
<th>Conference</th>
<th># Attendees</th>
<th># Respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDBT 2021</td>
<td>457</td>
<td>77</td>
</tr>
<tr>
<td>ICDE 2021</td>
<td>612</td>
<td>39</td>
</tr>
<tr>
<td>SIGMOD 2021</td>
<td>1819</td>
<td>86</td>
</tr>
<tr>
<td>VLDB 2021</td>
<td>1045</td>
<td>142</td>
</tr>
<tr>
<td>MDM 2021</td>
<td>332</td>
<td>332</td>
</tr>
</tbody>
</table>

Figures 3, 4, 5, and 6 show the distribution of age, gender, race/ethnicity, and professional experience of respondents. We also asked the respondents to tell us
the top three topics they wanted to see discussed at D&I sessions. Figure 7 shows an overwhelming convergence on five topics, with work-life balance, research (finding topics, setting goals, making progress, determining success), and career paths being selected by the most number of respondents. Other important topics (selected by about 20% of the respondents) included challenges of remote work, writing good research proposals, and inclusive culture in data management. We will continue to encourage participation in the survey, collect more data and use it to track how we are improving on D&I. For instance, PC chairs will be encouraged to present these stats and their evolution from year to year in the same way as paper submission stats. Our aim is to integrate our tools such as the gender analysis tool described in Section 3 to enrich the stats.

2. **D&I@DB CONFERENCES 2021**

**D&I@EDBT/ICDT.** EDBT/ICDT 2021 organized the inaugural event of the D&I@DB initiative with a debrief aiming to *Inform and Engage*. The debrief took place in a plenary session and was well-attended. It introduced the general goals, plans, and actions of D&I@DB in general and at EDBT/ICDT 2021 in particular. Two specific actions were piloted, namely pre- and post-conference questionnaires, and a tutorial on preparing inclusive videos. In the debrief, we explained the necessity of gathering data and experiences from the community to drive our actions. We also provided guidelines to session chairs on how to conduct D&I sessions.

**D&I@ICDE.** The D&I chairs of ICDE 2021 invited Geraldine Fitzpatrick, a Professor of Technology Design and Assessment who leads the Human Computer Interaction group at TU Wien, Austria. The talk entitled “Diversity matters matter” touched upon different notions and perceptions of diversity and bias, and highlighted their complexities. An introduction to the D&I initiative was also given. The talk explained the mission and actions and then focused on the automatic detection of COIs [12] that was performed this year at ICDE.

**D&I@MDM.** The D&I chairs of MDM 2021 view peer education as an important step towards achieving Diversity and Inclusion at conferences. With this in mind, they organized Education Day that began and ended with D&I events. The first event was a keynote talk on Federated Learning for Mobile Sensing Data. Cristian Borcea (Associate Dean for Strategic Initiatives, New Jersey Institute of Technology) provided tips for writing inclusive presentations using his own slides as examples. The second event was a D&I debrief, which discussed the two MDM-specific questions on the pre-conference D&I questionnaire, (1) Does research in mobile data management & IoT consider D&I aspects? and (2) How would you assess the current climate for women and under-represented minorities? Two invited talks followed the debriefing and closed the Education Day. The first talk was on Bias and Representation in Sociotechnical Systems (Danaë Metaxa, Stanford University) and the second on Disrupting Models of D&I in Undergraduate Research: Inside the iSchool Inclusion Institute (Kayla Booth, University of Pittsburgh).

The authors were provided with guidelines for preparing inclusive camera-ready papers, talks and videos. To recognize students’ efforts, MDM 2021 granted two *Best D&I Presentation* awards sponsored by the IEEE TCDE. Presentations were judged based on gender neutrality (in narrative and in figures), avoiding exclusionary examples, and illustrating color-blind inclusive graphs.

**D&I@SIGMOD.** SIGMOD 2021 D&I efforts were centered around educating authors and reviewers on inclusive writing, and hearing the perspectives of under-represented groups. The D&I chairs provided detailed instructions [6] on how to create inclusive papers and presentations and asked PC members to evaluate whether submitted papers followed inclusive writing instructions.

We organized round tables targeting under-represented groups: (1) a student/post-doc panel where the new generation of DB researchers provided their perspectives on academia, challenges during PhD and experiences at DB conferences, (2) a Women-in-DB panel where women shared their opinions on how to make conferences more inclusive, (3) two round-tables featuring senior and junior members of our community from both academia and industry discussing their experiences with failures and impostor syndrome, and (4) a panel discussing the D&I@DB initiative.

**D&I@VLDB.** VLDB 2021 D&I events were as follows. The first was a keynote titled “What relationship between an author’s ethics and their work?” by Gisèle Sapiro, research director at Centre national de la recherche scientifique (CNRS) and professor at the École des hautes études en sciences sociales. Gisèle provided an in-depth
analysis of the controversy following the Ullman Turing Award, discussed some cases of scandals, from Heidegger to Polanski and Handke, and recommended drafting a Code of Ethics for our community.

The second event was a panel titled “Are we D&I?”. The panelists debated the question of conducting research on D&I. The audience was polled during the panel with three questions: (1) In your opinion, what is and what is not D&I? Answers were Inclusive paper writing, Video captioning, Fund under-represented communities, Research on data ethics; (2) Do you think we are more/less D&I than other research communities? to which the majority answered “less”, and (3) How can we be more D&I? Answers included Session on personal experiences (good and not so good ones), Actions toward students, and Modifying review forms.

The third event was a “Women in Databases: What does it mean?” round table. The coffee style session invited participants to discuss anything from the impact of Covid-19 on our lives, to how to choose a promising research topic or establish a fruitful collaboration.

**D&I@ADBIS.** ADBIS 2021 promoted the D&I Code of Conduct [5] as part of the effort to bring awareness to the goals and actions of the D&I@DB initiative. The conference has been organized as a hybrid event and, to make it more inclusive, the organizers provided a room for participants with children.

**D&I@WISE.** WISE 2021 first action was to enforce, whenever possible, a balance of gender representation as well as country of origin in conference officers: PC selection, session chairs, student volunteers, and award sessions. The approach taken was being more inclusive rather than quota-based, such that to ensure that we remain fair in the process. This was a deliberate effort made throughout the whole planning and preparation of the conference by the organizing team, and it would be a continuous improvement process where diversity and gender balance will play a key role in the organisation of upcoming WISE conferences. The program included a special session on raising awareness of D&I, and invited suggestions for the D&I agenda at future editions. One specific outcome is the need for a plenary speaker to discuss the D&I in our research community and the role every researcher can play to promote D&I.

**D&I@SoCC.** A key action at SoCC’21 was the involvement of the D&I chair in all conference-related decisions, including selecting PC chair, PC members, and keynote speakers with an effort to select conference officers from under-represented groups, e.g, females. The program for SoCC also worked towards ensuring that keynotes, and paper talks are scheduled at the best feasible times for as many attendees as possible.

We posted D&I guidelines in writing and presentations as part of camera-ready instructions. We also posted the standard ACM policy against harassment and ACM processes for reporting unacceptable behavior. We enabled closed captioning on the client side with support from Microsoft Teams. To help students feel more connected and included, we ran a mentorship program offering them explicit help to interact with senior attendees. Additionally, the D&I chair executed the students travel scholarship program with the support from ACM. Lastly, the D&I chair organized and moderated a panel about “Research environment and Diversity in Cloud Computing” that focused on the topics that were ranked the highest in the D&I survey taken by the attendees at registration. The topics ranged from finding research problems, choosing career paths, handling failures and roadblocks to women in cloud computing.

### 3. TOOLS

There is a continuous ongoing effort towards building data-driven tools with the broad aim of improving con-
ference attendance and review quality for our venues. We described two tools deployed this year.

**COI tool.** Our professional societies, such as ACM, clearly dictate that conflicts of interest (COI) in the publication process are to be avoided in our venues [9]. Analysis of review data over multiple venues reveals that a reviewer who has COI with an author (referred to as **COI reviewer** and **COI author**, respectively) of a submission often tends to ignore or gloss over in his/her review critical issues with the submission that may be raised by his/her non-COI counterparts. Clearly, this adversely impacts review quality. Furthermore, our analysis shows that some COI reviewers accept only submissions of COI authors assigned to him/her while rejecting all other submissions. In particular, when COI authors are from the same country or ethnicity as the COI reviewer and the authors of rejected submissions have high diversity w.r.t these attributes, such reviewing behavior may collide with our D&I goals.

Double-blind reviewing is not designed to address these concerns. The goal of double-blind is to mitigate bias that may arise due to the ‘stature’ of authors or institutions, not to address COI [17]. Additionally, anecdotal evidence from PC chairs [16] indicates the existence of collusion rings that subvert the double-blindness by creating covert communication channels between COI reviewers and COI authors.

To facilitate detection and management of COIs during a review process, we have built CLOSET to automatically detects unreported/overlooked COIs [12]. CLOSET has been deployed for SIGMOD, VLDB, ICDE since 2020. Our initial analysis reveals that more than 20% of the submissions (conservatively) in some venues have one or more unreported COIs. That is, the problem of unreported COI is not insignificant in our venues.

**Gender analysis tool.** As an extension of the CMTStat tool, which automatically analyzes and visualizes conference statistics, such as acceptance rates and author country distribution [1], we are developing tools to import gender annotations of authors and perform analytics including growth in diversity for different Computer Science fields, acceptance/rejection rates based on gender, and cross-gender collaboration frequency. We have initial results on published papers across CS conferences and communities [13]. With GAP (Gender Analysis for Publications), we are currently leveraging the gender annotation on Gender API [4], a commercial tool already used for gender distribution in CS [18], and integrating manual annotations. The aim is to empower PC chairs to present enriched statistics at conferences [3].

4. **GOING FORWARD**

A number of important matters are being framed. We are currently discussing our bylaws with the aim of involving as many people as possible in this initiative. We also need to position D&I with respect to DBCares [2]—a joint effort by the VLDB Endowment and the SIGMOD Executive Committee whose purpose is to foster zero tolerance for abuse, discrimination, or harassment. ACM has a policy, including a definition of abuse, discrimination and harassment, a mechanism for reporting grievances, and consequences for validated grievances. Both SIGMOD and VLDB follow this policy. DBCares provides advice from respected and approachable members of the community with the goal of making reporting easier and less intimidating.

We have been scouting for Codes of Ethics from venues in other fields [10, 14] to identify unique aspects from each of them to integrate into our community. Additionally, we are looking into organizing committees in other conferences [7, 8]—particularly those that are targeted towards D&I efforts such as Accessibility Committee, Family Committee, and Sustainability Committee.

Reaching out to traditionally under-represented groups is an essential step in strengthening the influence of our community and attracting young scientists to leading conferences. Giving access to courses, material, as planned by SIGMOD, will be an efficient way to connect with universities and institutions that do not usually participate to major DB conferences. Our initial analysis showed that the Hispanic and Latin American community is particularly under-represented both as participants and authors. We have started to identify agencies in Latin American who can relay calls and funding opportunities for attending conferences. We will enhance these efforts by reaching out to (i) the Hispanic and Latin American community in the US and Canada through local associations; (ii) education managers of DB companies that have access to talented assets; (iii) under-represented countries in other regions, for example, the Caribbean, Asia Pacific and Africa.

Further, the D&I chairs of EDBT 2022 wish to promote a global perspective and understanding of Diversity and Inclusion, which includes promoting discussions on the variations in underrepresented groups in different countries and regions. Towards this end, the D&I chairs are planning a panel on “Diversity and Inclusion Perspectives Around the World” at EDBT/ICDT 2022.

The response from the community in various events in the 2021 conferences has been extremely positive and experiences shared across DB conferences has already enabled the scaling up of D&I activities in 2022 conferences. D&I@VLDB 2022 will focus on reaching out to other scientific communities (Web and IR, Distributed Systems, CHI) and organizing an event on what we learned so far and how we can do better at conducting D&I research. D&I@SIGMOD 2022, MDM 2022, AD-BIS 2022 and WISE 2022 plan to focus on mentoring activities that will extend beyond the main conferences.

We will encourage the wider adoption of automatic tools, such as for COI and Gender Analysis, along with extensive consent-based data collection to be combined with these tools 1. CLOSET is currently being extended to support data-driven PC formation with the goal of optimizing diversity, topic-related expertise, and COI-related risks. CLOSET was recently deployed in ACM/IEEE LICS 2022 (a premium venue in theoretical computer science community). There is also ongoing discussions with ACM on further development and deployment in venues beyond DB conferences.

---

1Data collection has to comply with data protection rules, such as the European GDPR and alike.
5. REFERENCES

Artifacts Availability & Reproducibility
(VLDB 2021 Round Table)

Manos Athanassoulis
Boston University
Rajesh Bordawekar
IBM Research
Ioana Manolescu
Inria and Institut Polytechnique de Paris

Peter Triantafillou
University of Warwick
Badrish Chandramouli
Microsoft Research
Yannis Papakonstantinou
Databricks and UCSD

Raja Appuswamy
EURECOM
Xuntao Cheng
Alibaba Cloud
Nesime Tatbul
Intel Labs and MIT

ABSTRACT

In the last few years, SIGMOD and VLDB have intensified efforts to encourage, facilitate, and establish reproducibility as a key process for accepted research papers, awarding them with the Reproducibility badge. In addition, complementary efforts have focused on increasing the sharing of accompanying artifacts of published work (code, scripts, data), independently of reproducibility, awarding them the Artifacts Available badge. In this short note, we summarize the discussion of a panel held during VLDB 2021 titled “Artifacts, Availability & Reproducibility”. We first present a more detailed summary of the recent efforts. Then, we present the discussion and the contributed key points that were made, aiming to assess the reproducibility of data management research and to propose changes moving forward.

1. SUMMARY OF RECENT EFFORTS

Reproducibility. In the recent past, the goal of reproducing scientific results has been adopted by increasingly many research communities. By indicating that a published research paper is reproducible, we refer to a result that obtained with stated precision by a different team using the same experimental setup and the artifacts provided by the author, following the current ACM terminology.\(^1\)

SIGMOD Reproducibility 2008-2012. In 2008, SIGMOD was the first data management conference that had an optional reproducibility test of published papers.\(^2\) In 2009, the effort was amended to test Workability as well, that is, whether the results can be generalized to broader experimental setups.\(^3\) This effort continued until 2012 where the goal was to make it part of the main conference and award to papers an official Reproducible badge for papers that have been independently reproduced and a Shareable badge for papers that have their artifacts available via a public URL.\(^4\)

SIGMOD Reproducibility 2016-2021. This reproducibility effort stopped between 2012 and 2015, and was resumed in 2016.\(^5\) The effort built on the previous instances of SIGMOD Reproducibility to create a standardized process for authors of accepted SIGMOD papers to follow. The SIGMOD Reproducibility Committee (RC) was established (renewed periodically) and calls for papers to be submitted for reproducibility were issued. Each submission was assigned to a member of RC, which evaluated the extent to which presented results could be adequately reproduced by them, using the artifacts (scripts, code, and data) provided by authors. Since 2017, reproducibility awards have been awarded to the easiest to reproduce artifacts,\(^6\) which are presented during the conference event.

PVLDB Joins the Ranks in 2018. In September 2018, PVLDB launched its own reproducibility effort.\(^7\) Similar to the SIGMOD effort, an RC was created and periodically renewed and authors of accepted papers were invited to undergo a reproducibility process, akin to that of SIGMOD. Also, awards were established and handed out to the authors of the easiest to reproduce papers.

\(^1\)https://www.acm.org/publications/policies/artifact-review-and-badging-current
\(^2\)Initially, the term Repeatability was used, however, since 2016, multiple communities converged to Reproducibility. Eventually, ACM adopted Reproducibility formally in 2020, which we use throughout this report to avoid confusion.
\(^4\)https://www.sigmod.org/2012/reproducibility.shtml
\(^5\)https://reproducibility.sigmod.org/history.html
\(^6\)https://sigmod.org/sigmod-awards/sigmod-most-reproducible-paper-award/
\(^7\)https://vldb.org/pvldb/reproducibility/
Currently, PVLDB mandates that, in addition to the voluntary nature of reproducibility for research papers, all experimental, analysis, and benchmark (EAB) papers must undergo reproducibility review.

**Availability and Reproducibility 2022.** Starting from 2022, the SIGMOD effort is renamed to **Availability and Reproducibility** and the committee renamed to **SIGMOD ARC**. Similarly, the VLDB RC is also conducting artifact availability checks.

**Artifacts Availability as a Goal.** Both VLDB and SIGMOD have been recognizing the increasing importance of artifact availability as a means to (1) increase the impact of database research papers, (2) enable easy dissemination and understanding of research results, and (3) enable easy sharing and uptake of data, code and experimentation set-ups. As a result, both conferences are strongly suggesting that all papers make their artifacts available, and the authors are asked to provide a justification when they do not submit links to their code and data. VLDB invites all papers to share a public URL as part of the paper metadata and the VLDB RC is carrying out checks for such links before awarding the badge. Starting from Volume 16, artifact availability for accepted papers in VLDB is considered part of the review process. SIGMOD invites all papers to share an archive with the artifacts which is also checked by SIGMOD ARC. Note that artifacts may include code, scripts, and experimental set-up on one hand, and also data sets, on the other hand.

Currently, the vetted artifacts along with deployment documentation, are made available through the ACM digital library as supplemental material. The data management community is also considering creating citable artifacts for code and data using independent DOIs. Similar efforts take place in sibling communities like systems, where conferences have an optional call-for-artifacts (e.g., SOSP\(^8\), OSDI\(^9\)), machine learning with their reproducibility challenge\(^10\), information retrieval, where the flag-ship conference, SIGIR, has a track for reproducibility papers\(^11\) and they assign badges and unique DOIs to code and data\(^12\), and data mining, where reproducibility is part of the review process\(^13\).

**Participation.** The participation of accepted pa-

\(^8\)https://sysartifacts.github.io/sosp2021/call.html
\(^9\)https://www.usenix.org/conference/osdi21/call-for-artifacts
\(^10\)https://paperswithcode.com/rc2021
\(^11\)https://sigir.org/sigir2022/call-for-reproducibility-track-papers/
\(^12\)https://sigir.org/general-information/acm-sigir-artifact-badging/
\(^13\)https://kdd.org/kdd2022/cfpResearch.html

pers in the reproducibility process has been fluctuating with mostly increasing trends over the last few years. For example, SIGMOD received 9 reproducibility submissions of 2019 papers, 16 for 2020, and 25 for 2021 (that are currently under reproducibility review). VLDB RC is on track to receive about 20 papers in this submission cycle. In fact, a key point brought up in the VLDB 2021 panel was how to increase participation. Since the effort to make artifacts available just started, we look forward to report on those in future reports and panels. But, there are encouraging signs: from the VLDB side, more than 120 papers published in Volume 14 carried the **Artifacts Available** badge. In the current issue of Volume 15, VLDB RC observed that about 70% of all published papers carried the **Artifacts Available** badge in their published pdf.

### 2. PANEL QUESTIONS

**Is what we are doing at present science?**

The panel had a lively discussion that highlighted that our community’s research has a strong practical component, hence, availability of artifacts helps to increase the impact of published papers. The panel further discussed the importance for the community to test reproducibility in, ideally, all papers, as a clear signal that we value impact over quantity. Yannis Papakonstantinou highlighted that a large fraction of the data management papers are essentially technological contributions aiming to build new artifacts, hence artifact availability is key.

**Should published results be reproduced?**

Panelists agreed that reproducibility reviewing is desired for all research results, however, it was noted that there are artifacts that cannot be shared mostly for legal reasons (e.g., industrial closed-source systems and sensitive data sets). This can be mitigated if reproducibility uses the artifacts without publicizing them for these cases. For example, Xuntao Cheng pointed out that it should be an option to release the binary only and not to open source the code. Badrish Chandramouli highlighted that while availability and reproducibility are crucial, the community accepts claims at face value and should have a degree of trust in the authors.

**Should we require availability?**

The panel argued that availability and reproducibility are both crucial, and in fact, Rajesh Bordawekar, pointed out that code availability is absolutely necessary for showing impact on practical problems and accelerating research progress. Further, Rajesh proposed to elevate the artifacts to a separate citable entity with its own DBLP entry and potentially DOI, and hold sessions to highlight avail-
able and reproduced code artifacts. Badrish Chandramouli and Yannis Papakonstantinou pointed out that availability is not the same as usability, in the sense that publicly available artifacts should also have detailed instructions on how to use them to have impact. In fact, in order for a VLDB paper to receive the Artifact Available badge, the submission to the RC must be accompanied by an easy-to-read set of instructions describing the code and data sets and their use. Further, Badrish highlighted that industrial research might involve (modifications to existing) closed-source systems (e.g., the Microsoft SQL Server optimizer), which might be harder to be made available. He further pointed out that with some extra effort, such contributions could also validate the ideas using open-source alternatives, thereby releasing such artifacts. Currently, the authors of VLDB and SIGMOD submissions are asked to justify it when artifacts are not available.

Even then, is availability enough?
The panel agreed that availability is paramount and should be as wide as possible. Further, the panelists agreed that verifying reproducibility is also crucial for three reasons: (i) to reward well-developed, easy to use/adopt tools that help accelerate the progress of our field, (ii) send the signal that impact and applicability is valued by our community, and (iii) discourage poor scholarship and/or malicious publication efforts. However, both Raja Appuswamy and Xuntao Cheng pointed out that in addition to availability and reproducibility, we need cloud-based availability that can readily offer a library of prior artifacts in an easy-to-deploy setup. Badrish highlighted that one trend seen in industrial labs is to incorporate reproducibility in the development process, making it almost automatically reproducible. The main challenge is that sometimes new industrial research results are based on new proprietary hardware making it very hard to reproduce without access to that hardware.

Please comment on the status quo.
In this question, the panel discussed diverse points of view. Xuntao Cheng highlighted that publishing for industry is a means to advertise high-quality work and build technical reputation for the company. Further, he pointed out that our community needs to work on delivering more benchmarks that more closely match modern applications. Rajesh Bordawekar proposed to make the review process open (similar to NeurIPS) and double-blind and focus on code-reuse and availability. Raja Appuswamy pointed out that in other communities (e.g., Bioinformatics) the source code and the data are required for the paper to be accepted, and the authors have to provide usable material in order for the paper to get published. Nesime Tatbul shared her view on making the process easier by giving incentives to everyone to participate (e.g., more visibility for the authors, recognition and education opportunities for reviewers, faster innovation for adopters, and sharing success stories from all fronts), and also work towards having a better time management for the authors through up-to-date best practices and shared infrastructure. This resembles the point raised earlier for a common research cloud with a library of approaches. Ioana Manolescu pointed out that currently the authors only perceive a small benefit from making sure their code is reproducible, while the effort to make it available and package it for reproducibility is frequently perceived as much higher. This resonates with Nesime’s point about giving incentives for researchers (especially junior ones) to participate both as submitting authors and as reviewers.

Overall, the panelists agreed that reproducibility has made progress in our community, however, more steps still need to be taken.

How would you suggest to change the status quo?
The panelists came with a wealth of ideas on how to change the status quo, focusing mostly on making the process easier and smoother for the authors, the reviewers, and the (future) researchers that will be able to benefit from the available artifacts.

Incentivize. Virtually all the panelists agreed that it is important to give additional incentives to authors and reviewers to participate. The incentives can be both rewards and penalties. For example, Ioana supported that the extra recognition of the reproducibility and availability badges is not always enough. Her proposal is to ensure that at training and education level, all graduate students learn good coding practices early on and use the available state-of-the-art tools. In order to further reward artifacts, Ioana also proposed to give higher acceptance chances to papers that verify. Badrish agreed on providing incentives, but does not want to punish authors that do not share their code, which may be perfectly acceptable in many circumstances. Nesime proposed to further incentivize through more publicity of success stories and role models.

Use good practices. Building on Ioana’s point above, there was a unanimous proposal to make sure that all authors should use good coding practices, code sharing tools (e.g., git), and if possible, build a repository of accepted solutions that researchers can quickly access artifacts (verified or not). The idea of a common repository (or collection of repos-
itories) got wide support in the discussion. In addition, Raja Appuswamy suggested that building a common data management dev cloud that will allow researchers to have access to virtual machines with high-end capabilities would significantly help this process. Further, tools like docker that help make execution and cloud deployment easy should also be employed. Yannis Papakonstantinou further highlighted that it is very important to ensure that artifacts are always accompanied by the setup, the parameters, and all the necessary details to use and reproduce an artifact.

Learn from other communities. Several of the panelists highlighted that there is expertise in other communities that can be leveraged. For example, Nesime and Rajesh mentioned the NeurIPS efforts in reproducibility, highlighting the open access journal ReScience\(^{14}\) that publishes reproducibility reports, and the reproducibility challenge as a tool to crowd-source the effort and educate young researchers. Similarly, the systems community has integrated artifact evaluation in all of its conferences as an optional but strongly suggested step.

Increase experimental analysis papers. Yannis Papakonstantinou highlights that from the papers published in our community only a small fraction is evaluation papers (in the form of experimental analysis and benchmarking papers), while in other disciplines (e.g., Medicine) it is frequent that the evaluation papers are a much larger fraction of all published papers. One actionable item is to welcome more such evaluation papers, as VLDB has pioneered for a few years now.

A verifiability metric. Lastly, Yannis proposed the addition of a new review metric called verifiability to capture the reviewer’s confidence that the available details and artifacts are enough to reproduce and verify the presented results.

3. CONCLUSION & FUTURE PLANS

The efforts surrounding reproducibility and artifact availability are now maturing steadily within our community, as evidenced by the SIGMOD and VLDB efforts. Artifact availability and reproducibility are increasingly being sought after, which is shown by the increasing participation in the SIGMOD and VLDB reproducibility processes. Regarding artifact availability, both SIGMOD and VLDB publish artifacts as supplemental material, while creating dedicated entries for software and code with unique DOIs is also considered. This approach would allow artifacts to be directly citable. However, storing a snapshot of the artifacts (the one associated with the published paper) as a separate entity makes it harder to cite new versions of the artifacts. Further, by having two citable entities, the research paper that explains intricacies and details of the artifact might be less accessed or cited.

A plan under consideration moving forward is to continue to attach software artifacts to the paper, and create unique DOIs for artifacts that contain new data sets. Ultimately, our goal is for all published papers to have their code, scripts and data available, and ideally to have been reproduced.

However, there are obstacles along the way; the primary one being the significant additional workload imposed on authors and reviewers, especially to carry out reproducibility checks. The panel concluded that more effort on rewarding this workload is needed to ensure that the reproducibility process will further expand and eventually become an integral part of the publication process. We are far from the ideal scenario where every single published paper in our community has successfully undergone a reproducibility process and is accompanied by artifacts (when possible). Similar to the recommendations of the National Academies report\(^{15}\) [3], to reach our ideal state, we need (i) to inform researchers about the need and associated benefits, (ii) to blend artifact generation as a first-class citizen within the education and training of our future colleagues, and (iii) create the necessary operational infrastructures. Overall, our efforts are part of a larger effort at a higher level, exemplified by the new ACM Reproducibility Emerging Interest Group\(^{16}\) that hosts an annual workshop dedicated on practical reproducibility. The data management research community is determined to continue and deepen these efforts.

For any questions and suggestions, please contact the SIGMOD ARC and VLDB RC chairs.

4. REFERENCES


\(^{14}\)http://rescience.github.io/

\(^{15}\)https://www.nationalacademies.org/our-work/reproducibility-and-replicability-in-science

\(^{16}\)https://reproducibility.acm.org/

\(^{17}\)https://p-recs.github.io/2022/