



# Still Asking: How Good Are Query Optimizers, Really?

Viktor Leis  
Technische Universität München  
Germany  
leis@in.tum.de

Andrey Gubichev  
Databricks  
USA  
andrey.gubichev@databricks.com

Atanas Mirchev  
Volkswagen Group  
Germany  
atanas.mirchev@volkswagen.de

Peter Boncz  
CWI  
Netherlands  
p.boncz@cwi.nl

Alfons Kemper  
Technische Universität München  
Germany  
kemper@in.tum.de

Thomas Neumann  
Technische Universität München  
Germany  
neumann@in.tum.de

## ABSTRACT

This retrospective revisits our 2015 PVLDB paper *How Good Are Query Optimizers, Really?*, which challenged the prevailing notion that query optimization was a solved problem. By designing the Join Order Benchmark (JOB) and conducting a series of systematic experiments, we empirically disentangled the contributions of plan enumeration, cost modeling, and cardinality estimation. Our findings showed that cardinality estimation errors are widespread and often the dominant factor behind poor query plans, while cost models and enumeration strategies matter comparatively less. The benchmark and methodology helped refocus the community’s attention on cardinality estimation and led to a resurgence of research in this area, including learned and AI-based approaches. We reflect on the role of experiments and benchmarking in database research, survey developments in query optimization over the past decade, and discuss open challenges around robustness, adaptive execution, and realistic workloads.

### PVLDB Reference Format:

Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. *Still Asking: How Good Are Query Optimizers, Really?*. PVLDB, 18(12): 5531 - 5536, 2025.  
doi:10.14778/3750601.3760521

## 1 INTRODUCTION

**Is query optimization magic?** Compared to other critical internal components – such as the query executor, storage manager, and transaction manager – the query optimizer is often seen as mysterious, even esoteric. It has a reputation as a kind of black art, typically mastered only by those who have spent decades working on it. The optimizer’s primary goal is to select an efficient execution plan for a query expressed in a declarative language like SQL. While the basic ideas behind cost-based, Selinger-style [4, 32] optimizers are easy enough to understand, real-world optimizers rely on a broad range of techniques, including advanced plan enumeration and optimization algorithms, rule-based transformations, sophisticated cost models, statistical information about the data, and even crude

heuristics. The complex interplay among these techniques makes it difficult to predict the impact of any single change on the overall behavior of the optimizer.

**Query optimization solved?** Our goal with the 2015 PVLDB paper *How Good Are Query Optimizers, Really?* was to try to demystify query optimization and to empirically disentangle the interactions among plan enumeration, cost models, and cardinality estimation. At that time, query optimization research had become somewhat of a niche area within the database research community. While interesting papers were still being published, the area as a whole appeared somewhat stagnant – and from the prevailing topics at the time one might have (incorrectly) inferred that in its core, query optimization was a solved problem. The few new papers that did appear focused primarily on optimization algorithms for very large join queries. Most of these sophisticated algorithms guarantee optimality in the sense that, if the cost model and cardinality estimates were correct, the algorithm will find the cheapest plan. However, this raises a more practical question: how large is the benefit of fully exploring the plan space, and how accurate are the cost model and cardinality estimates in practice?

**The Achilles Heel.** In 2014, toward the end of his long and successful career in query optimization at IBM, Guy Lohman wrote:

*“The root of all evil, the Achilles Heel of query optimization, is the estimation of the size of intermediate results, known as cardinalities. Everything in cost estimation depends upon how many rows will be processed, so the entire cost model is predicated upon the cardinality model. In my experience, the cost model may introduce errors of at most 30% for a given cardinality, but the cardinality model can quite easily introduce errors of many orders of magnitude!” [20]*

So there seemed to be a disconnect between the academic community and industry on the state of query optimization, and specifically, cardinality estimation. In industry, DBMS development teams get confronted with a continuous stream of bugs and issues filed by users, often the result of sub-optimal optimization of SQL queries running on real-life customer datasets. In contrast, in academia, before the JOB paper, workloads would be evaluated on the TPC

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing [info@vldb.org](mailto:info@vldb.org). Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.  
Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.  
doi:10.14778/3750601.3760521

benchmarks, which use data that is synthetically-generated and devoid of correlation or heavy skew. On these synthetic benchmarks, query optimization is a much less pressing problem.

**Join Order Benchmark.** The biggest hurdle in our study therefore was designing a suitable workload. We realized that standard benchmarks such as TPC-H and TPC-DS – though highly successful for evaluating query execution – are not well-suited for assessing query optimization. Ideally, we would have had access to a real-world dataset paired with real-world queries. Instead, we settled for a real-world dataset (IMDB) and 113 manually crafted queries. We intentionally kept the query structure very simple: each query consisted of a single Select-Project-Join block, such as to focus on the join ordering problem, which we considered the core problem for query optimization. The resulting workload is called the Join Order Benchmark (JOB).

**Real systems have large cardinality errors.** Answering our research questions required not only a suitable benchmark, but also several methodological innovations. To compare the quality of cardinality estimation across different systems, we developed a setup to extract cardinality estimates from several relational database systems. By obtaining estimates for all intermediate results in JOB, and comparing them to the true cardinalities obtained through execution, we showed that estimation errors are large and grow rapidly with query complexity. Significant errors – often one order of magnitude or more for larger expressions – occur routinely across all systems.

**Cardinality errors cause large slowdowns.** The next step was to determine the impact of these estimation errors on query execution times. To do this, we modified PostgreSQL to allow the injection of cardinalities into its optimizer. We found that, while true cardinalities do not significantly affect runtimes for most queries, a substantial fraction benefit dramatically. In fact, in PostgreSQL 9.4, roughly 10% of the JOB queries failed to complete in any reasonable time frame due to cardinality estimation errors. Our results also showed that the performance degradation caused by misestimation is more pronounced when more indexes are available – a finding confirmed in a recent experimental study using Microsoft SQL Server [17].

**The cost model does not matter much.** To investigate the impact of the cost model, we compared PostgreSQL’s default cost model with both a tuned version and a trivial one. Overall, we found that the impact of the cost model is dwarfed by errors in cardinality estimation.

**Join enumeration space size matters (somewhat).** Finally, we investigated the impact of join enumeration space size on query performance. Unsurprisingly, we found that a larger search space (e.g., including bushy trees) improves performance compared to more restricted spaces (e.g., only left-deep trees), and that full enumeration via dynamic programming outperforms heuristic approaches such as greedy operator ordering. However, we again observed that these benefits are much smaller than the improvements gained from more accurate cardinality estimates. Note that the results reported in the conference version of the paper were incorrect

due to a data-handling issue; they were corrected in the journal version [19].

**Impact.** We are pleased that our paper has contributed to the growth of research in query optimization and cardinality estimation. Anecdotally, junior researchers have told us that the paper served as their introduction to the area and motivated them to pursue research in query optimization. According to Google Scholar, the yearly number of citations for the conference version of the paper increased from 35 in 2017 to 176 in 2024. This humbling number reflects a growing interest in query optimization, particularly in learned and AI-based approaches. Today, the conference paper [18] and its extended journal version [19] are usually cited either to motivate the need for better cardinality estimates and robust query execution strategies, or because JOB is used as a benchmark. The paper and JOB helped operationalize the problem of cardinality estimation – transforming it from a vague concern into a measurable and actionable optimization target.

**Outline.** In this retrospective, we first reflect on the role of experiments and benchmarking in the database research community. We then survey key developments in query optimization over the past decade. Finally, we conclude with a discussion of open questions and promising research directions.

## 2 THE STORY OF THE JOIN ORDER BENCHMARK

**A happy accident.** Much of the impact of our paper stems from the wide adoption of the Join Order Benchmark – despite the fact that we did not initially set out to propose a new benchmark. Instead, designing a custom workload was simply a necessity to answer the research questions we had posed. It was only relatively late in the project that we realized that packaging and promoting our workload as a benchmark could benefit other researchers. This turned out to be a fortunate decision: the paper provided both a compelling motivation (via experimental evidence) and a practical tool (via the JOB artifact) for advancing research in cardinality estimation.

**Synthetic data generators are too easy.** Standard benchmarks such as TPC-H, TPC-DS, and the Star Schema Benchmark include synthetic data generators. To enable scalability without altering query behavior, these generators rely on relatively simple data distributions and assumptions. For example, in TPC-H, each order contains between 1 and 7 lineitems, with the number chosen uniformly at random. In contrast, real-world data often exhibits skewed distributions, long tails, and is dominated by strings [37]. In real-world datasets one observes value, frequency and join-hit-rate skew, as well as correlations between values in different columns and even tables. For instance, the amount of tuples selected by a (column=constant) can differ by orders of magnitude, depending on the constant. A conjunction of predicates can even be completely correlated, e.g., in *model='accord' AND make='honda'* the second predicate adds nothing [24]. Correlations also span connected tables, e.g., the join hit ratio between movies produced in France and actors born in Paris is much higher than on average. Overall, synthetic data generators are overly sanitized and tend to “bake in”

many of the assumptions – such as uniformity, independence, and the principle of inclusion – that cardinality estimators themselves rely on. This makes synthetic benchmarks significantly easier than real-world workloads.

**Dataset.** To make cardinality estimation a challenging and meaningful task, we needed a real-world dataset. Although many interesting datasets are available online, most are denormalized. Because our goal was to study join ordering, we required a relational, normalized schema with multiple tables that would yield non-trivial join queries. We also wanted the dataset to be of reasonable size to support meaningful runtime experiments. Eventually, we settled on the IMDB database. After transforming it using the *imdbpy* package, we obtained a schema with 21 relations.

**Queries.** In addition to the dataset, we needed queries – which proved even more challenging to obtain. One of the authors, then an undergraduate student, was tasked with manually creating them. The final workload consists of 113 queries based on 33 query templates that differ in their base table predicates. A small number of queries are intentionally designed to return empty results, reflecting a common occurrence in real workloads but one rarely seen in synthetic benchmarks. All attributes in the SELECT clause are wrapped in a MIN aggregate to avoid copying large result sets to the client, which could otherwise become a performance bottleneck [30].

**Is JOB realistic?** Given the origins of the JOB queries, we do not claim that the benchmark is “realistic” or representative of real-world workloads. However, we would argue that the workload is reasonable, and there is no compelling reason why a database system should perform poorly on it.

**Why JOB was widely adopted.** JOB was explicitly designed to evaluate cardinality estimation, cost models, and join enumeration. Its query structure is deliberately simple, consisting solely of inner joins and base table selections. All joins are on PK/FK or FK/FK integer attributes. This simplicity – combined with its manageable size – makes JOB easy to implement in prototype systems and well-suited for exploring and comparing different optimization approaches.

**Limitations of JOB.** The very features that make JOB appealing – its basis in a real-world dataset and the structural simplicity of its queries – also constitute its primary limitations. Unlike synthetic benchmarks, which can be scaled to arbitrary sizes, the IMDB dataset is simply too small to serve as a meaningful benchmark for large-scale, distributed query processing systems. Moreover, JOB lacks many common and challenging SQL features, such as GROUP BY, outer joins, subqueries, and complex expressions<sup>1</sup>. Finally, the specific way in which the JOB queries were constructed may also influence some of the findings reported in our paper. For example, we recently observed that while query optimizers tend to underestimate cardinalities as the number of joins increases on JOB, the opposite trend (overestimation) occurs on the SQLStorm benchmark [31].

<sup>1</sup>We were fortunate that one of the VLDB reviewers rightly criticized the name we had originally chosen, Query Optimizer Benchmark, which prompted us to adopt the more appropriate name Join Order Benchmark.

### 3 ON EXPERIMENTS, ANALYSES, AND BENCHMARKS

We submitted our paper to the *Experiments and Analyses* (E&A) track of PVLDB 2015. A Test of Time Award for an E&A paper is a fitting opportunity to reflect on the history and future of experimental science in the database community.

**History of the E&A track.** The E&A track has its roots in a “special topic” experiment at VLDB 2008, which aimed to “meet needs for consolidation of a maturing research area by providing a prestigious forum for in-depth analytical or empirical studies and comparisons of existing techniques” [1]. The driving force behind this initiative – and chair of the first two iterations – was Volker Markl. Originally, E&A papers were reviewed by a separate program committee and presented in a dedicated session of the conference. Today, they are reviewed by the general VLDB program committee and integrated into the main conference program. During the review process, E&A papers are marked, and program committee members are instructed to evaluate them using appropriate, special criteria.

**E&A impact.** We believe the E&A initiative has been highly successful and has significantly enriched the database literature. Similar initiatives have been adopted by EDBT (since 2022) and SIGMOD (2026). While experimental papers were occasionally published at prestigious database venues even before the introduction of explicit E&A categories, it is likely that many influential papers – including our own – would not have been written without this dedicated paper type. This success has also been supported by making E&A papers as visible and prestigious as regular VLDB papers. Comparing existing algorithms, designing meaningful benchmarks, and carefully analyzing experimental results often requires an inordinate amount of effort – but this work benefits the entire community. It is encouraging that the database community has found a way to properly incentivize this kind of contribution. Other community-wide efforts, such as vision papers, reproducibility initiatives, and a stronger emphasis on encouraging authors to publish source code, are also promising.

**EA&B.** It is altogether fitting and proper that, starting with VLDB 2021, the E&A category was renamed to *Experiments, Analysis & Benchmarks* (EA&B). After all, good experiments require well-designed and meaningful benchmarks, and many E&A papers begin with the design of such benchmarks. For better or worse, benchmarks shape the research landscape. Good benchmarks can drive progress by providing objective targets and can thus contribute to the development of better systems. Unrealistic benchmarks, on the other hand, can hinder progress by misleading researchers and encouraging work on irrelevant aspects of a problem. While there is value in having a small number of established and well-understood benchmarks such as TPC-C and TPC-H these should not be the sole gold standard for determining whether a research idea is worthwhile. After all, benchmarks are never perfect – JOB certainly has many weaknesses – and should therefore be updated, revised, or superseded, based on lessons learned from their adoption and emerging real-world challenges.

## 4 2015-2025: A DECADE OF QUERY OPTIMIZATION RESEARCH

Over the past ten years, we have witnessed a resurgence in query optimization research. Notably, two monographs on query optimizer implementation have been published: one by Ding et al. [7], and another by Moerkotte [26]. While not exhaustive, this section provides a brief overview of the major research themes, highlighting representative papers.

### 4.1 Learning-Based Approaches

Since 2015, deep learning has disrupted an increasing number of areas within Computer Science. Unsurprisingly, learning-based techniques have also played a prominent role in recent query optimization research.

**Learning cardinalities.** Cardinality estimation is a natural fit for statistical learning methods: it is a data-driven, inherently uncertain problem with a clear optimization objective – yet one for which we still lack a robust solution. Two main approaches can be distinguished: query-driven cardinality estimation (supervised learning) and data-driven techniques (unsupervised learning). As the names suggest, the former learns a statistical mapping from queries observed in a particular workload to cardinality estimates, while the latter models the data distribution directly. MSCN [16] is an example of a query-driven approach. In contrast, DeepDB [12] and NeuroCard [46] exemplify data-driven approaches, modeling the joint probability distribution of a relation or database. There are also hybrid methods, such as UAE [42].

**Learning to find better plans.** Rather than merely improving cardinality estimation through statistical methods and feeding these estimates into traditional optimizers, one can also learn query plans directly. Neo [22] uses an existing optimizer to bootstrap a model that generates full-fledged execution plans. Bao [21] and FASTgres [41] steer an existing optimizer through optimizer hints by learning how these affect the performance of specific queries. SkinnerDB [35] employs reinforcement learning during query execution to explore different join orders.

**Learned approaches have not yet been widely adopted.** Several independent experimental studies [11, 15, 38] confirm that learning-based methods can improve estimation quality (e.g., q-error) over traditional approaches on benchmarks such as JOB. However, these methods also present notable drawbacks, including high training and inference costs, difficulty adapting to dynamic environments, challenges in obtaining high-quality training data, and unpredictability due to their black-box nature [38]. For a Bao-style plan-based approach, Microsoft reported limited performance gains in production and cited challenges such as noisy and expensive performance measurements [13]. Thus, despite promising experimental results in academic settings, learning-based techniques have yet to see widespread adoption in industry.

### 4.2 Theory to the Rescue?

**Worst-case bounds.** An alternative paradigm in cardinality estimation relies on theoretical worst-case bounds derived from data statistics, as exemplified by SafeBound [5] and LpBound [48]. A

key challenge, however, is that practical heuristics often yield estimates that are closer to the true cardinality than these worst-case bounds. Despite this, recent theoretical work is making progress toward significantly tightening such bounds. Looking ahead, these approaches could inspire new sketching techniques or be combined with traditional or learned estimators – for example, to limit the impact of large estimation errors (outliers) when other methods fail.

**Worst-case optimal joins.** The theory community successfully developed Worst-Case Optimal Join (WCOJ) algorithms [28, 29], and these have also been adopted in some systems [9, 36]. WCOJ algorithms achieve run-time complexity superior to traditional binary joins (that join two relations at-a-time) on “diamond shaped join” plans [3]. In such plans, the inputs and final query result are much smaller than the intermediate results generated by traditional joins. Examples of such query plans are cyclic join patterns over many-to-many (n:m) relationships, as found in graph exploration, e.g., triangle queries. An advantage of WCOJs is that it makes determining a join order less relevant, since they execute multiple joins at once. However, the cost of constructing the – typically – trie-shaped data structures needed for WCOJs, which requires materialization of all inputs, mean that such algorithms can be slower when the query does not have strong diamond characteristics, which is the case in JOB, and when the inputs do not fit in memory [39]. Therefore, a query optimizer still needs to decide when to use a WCOJ or not [25], but estimating the cost of such explosive/graph joins is an area where cardinality estimation is still weak.

### 4.3 Better Benchmarks

While many of the papers discussed above use JOB for evaluation, several new benchmark proposals have also emerged.

**Synthetic benchmarks.** LDDB created the Social Network Benchmark (SNB) with two workloads: Interactive [8] and Business Intelligence [34]. Both operate on a synthetically generated graph dataset, which, when represented relationally, features “growing” many-to-many (n:m) joins. The dataset is further enriched with structural and value correlations, and the benchmark queries are parameterized to explicitly exploit these correlations [10].

**Manually-crafted workloads.** Despite the relatively simple structure of the JOB benchmark, many early learning-based approaches used *JOB-light*, an even more simplified variant introduced in the MSCN paper [16]. At the opposite end of the spectrum, *JOB-complex* [40] was recently proposed to increase the benchmark’s difficulty, for example, by introducing joins on non-PK/FK and string attributes. The Cardinality Estimation Benchmark (CEB), introduced in the Flow-Loss [27] paper, programmatically generates a large number of queries on the IMDB and Stack Exchange datasets. Another benchmark based on the Stack Exchange dataset is STATS-CEB [11], which consists of 146 hand-curated queries.

### 4.4 Runtime Approaches

**Runtime feedback.** Since decades of work on cardinality estimation failed to result in a decisive breakthrough, an alternative

avenue is to learn cardinalities during run-time, e.g., by observing the intermediate result sizes of queries as they get executed. This had been proposed long before JOB, specifically in IBM’s Learning Optimizer (LEO) project [33]. An important lesson from this approach was that having exact knowledge on the cost of some query plans does not necessarily lead to improved decision making. Specifically, if the cost model underestimates the size of alternative (not yet executed) plans, these will appear to be very attractive compared to the real cardinality of the executed plan. This can lead to poor decisions and is called “fleeing from knowledge to ignorance” [24]. The discovery of this phenomenon led to the identification of the wider problem of how to combine estimates that come from different estimators, in a probabilistically consistent way. Maximum Entropy [23] is a solution to this problem, but it is computationally very expensive and difficult to apply to arbitrary join queries, limiting its practical use. We suspect that issues similar to “fleeing from knowledge to ignorance” haunt many learning-based approaches as well.

**Limited runtime adaptivity.** There also exist more limited, but practical, forms of runtime feedback. One is to dynamically reorder certain kinds of joins in a query pipeline based on observed runtime behavior [14, 51]. Another approach is to dynamically adapt the distributed joins between broadcasting and shuffling [43]. These techniques operate at the level of physical operators and have become increasingly widespread, improving system robustness without relying on cardinality estimates.

## 5 LESSONS, CHALLENGES, AND FUTURE DIRECTIONS

**Overfitting.** JOB has been used in a wide array of (mostly) learning-based inspired approaches to query optimization. While a lot of these research efforts have reported positive results, we note that learned query optimization is vulnerable to overfitting. This certainly is a danger in JOB, because of its small dataset and limited query workload. To guard against this, benchmarks on large real-life datasets and diverse query workloads are needed. In this VLDB conference, we introduce the SQLStorm benchmark [31], which leverages LLMs to generate large and complex workloads. Its methodology enables workload scales far beyond JOB and improves robustness against overfitting. We believe that there is significant opportunity for further research in this space, e.g., into targeted benchmark development that model real-world query logs [6]. It could also be the case that such large-scale benchmarking efforts eventually will show limits of learned query optimization. As mentioned, in real-world systems learned methods have not replaced classical methods yet.

**Regressions can prevent innovation.** We should acknowledge that even traditional query optimizers in many cases already work so well that even limited overheads become show-stoppers. Performance regressions are the bane of industry efforts in query optimization: even if a particular technique noticeably improves the average performance of a large workload, the presence of regressions can prevent its adoption. In practice, users rarely notice queries that become faster, but are quick to report regressions when

queries slow down. As a result, a central challenge in query optimization is to identify and improve the negative outliers without degrading performance for queries that are already well-optimized.

**Yannakakis revival.** In recent years, there has been a revival of run-time techniques for executing complex queries. Specifically, the idea to make the Yannakakis algorithm [47] practical, is now spurring a lot of research, in which JOB and other query optimization benchmarks get used. The Yannakakis join processing algorithm has complexity that is linear in the input tables and query result, and hence is optimal. It uses semi-joins to first reduce all input tables to those subsets of rows that actually participate in the query result. However, in doing so it reads the input twice, which in practice makes it slower on many queries than a traditional join plan with correct join order. The new approaches [2, 44, 45, 49, 50] optimize its semi-join reduction strategy using cost models, heuristics, or run-time feedback, and enhance its raw speed by replacing semi-joins with probabilistic filters and adaptive query execution. This wave of work could be seen as an alternative path forward, partially by-passing the limitations of cardinality estimation errors.

**Data lake challenges.** While run-time techniques can dampen the effects of wrong join orders [50], they still obtain best results if the join order is optimal. We note that the availability of meta-data has become scarcer – even just key constraints, let alone statistics or sketches – in modern cloud-based environments, compared to classical on-premise data warehouses, common at the start of the millennium. Run-time techniques rely less on models and statistics, and thus are at an advantage in such environments. But, efforts to enhance Data Lakes by introducing semantic layers/views, which add meta-data beyond the realm of traditional database schemata, could also start playing a role in enhancing query optimization.

**Conclusion.** Overall, we foresee a future where combination of static and run-time optimization is used, enhanced by novel kinds of metadata and where robust benchmarking will remain a crucial technique to evaluate the progress of this field.

## REFERENCES

- [1] 2008. 34th International Conference on Very Large Data: Calls. [https://www.cs.auckland.ac.nz/research/conferences/vldb\\_site/calls.html#125](https://www.cs.auckland.ac.nz/research/conferences/vldb_site/calls.html#125)
- [2] Liese Bekkers, Frank Neven, Stijn Vansummeren, and Yisu Remy Wang. 2024. Instance-Optimal Acyclic Join Processing Without Regret: Engineering the Yannakakis Algorithm in Column Stores. *CoRR abs/2411.04042* (2024).
- [3] Altan Birlir, Alfons Kemper, and Thomas Neumann. 2024. Robust Join Processing with Diamond Hardened Joins. *Proc. VLDB Endow.* 17, 11 (2024), 3215–3228.
- [4] Surajit Chaudhuri. 1998. An Overview of Query Optimization in Relational Systems. In *PODS*. 34–43.
- [5] Kyle B. Deeds, Dan Suciu, and Magdalena Balazinska. 2023. SafeBound: A Practical System for Generating Cardinality Bounds. *Proc. ACM Manag. Data* 1, 1 (2023), 53:1–53:26.
- [6] Shaleen Deep, Anja Gruenheid, Kruthi Nagaraj, Hiro Naito, Jeffrey F. Naughton, and Stratis Viglas. 2022. DIAMETRICS: benchmarking query engines at scale. *Commun. ACM* 65, 12 (2022), 105–112.
- [7] Bailu Ding, Vivek R. Narasayya, and Surajit Chaudhuri. 2024. Extensible Query Optimizers in Practice. *Found. Trends Databases* 14, 3-4 (2024), 186–402.
- [8] Orri Erling, Alex Averbuch, Josep Lluis Larriba-Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat-Pérez, Minh-Duc Pham, and Peter A. Boncz. 2015. The LDBC Social Network Benchmark: Interactive Workload. In *Proc. SIGMOD Conference*. ACM, 619–630.
- [9] Michael J. Freitag, Maximilian Bandle, Tobias Schmidt, Alfons Kemper, and Thomas Neumann. 2020. Adopting Worst-Case Optimal Joins in Relational Database Systems. *Proc. VLDB Endow.* 13, 11 (2020), 1891–1904.
- [10] Andrey Gubichev and Peter A. Boncz. 2014. Parameter Curation for Benchmark Queries. In *Proc. TPCTC (LNCS, Vol. 8904)*. Springer, 113–129.

- [11] Yuxing Han, Ziniu Wu, Peizhi Wu, Rong Zhu, Jingyi Yang, Liang Wei Tan, Kai Zeng, Gao Cong, Yanzhao Qin, Andreas Pfadler, Zhengping Qian, Jingren Zhou, Jiangneng Li, and Bin Cui. 2021. Cardinality Estimation in DBMS: A Comprehensive Benchmark Evaluation. *Proc. VLDB Endow.* 15, 4 (2021), 752–765.
- [12] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulesa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2020. DeepDB: Learn from Data, not from Queries! *Proc. VLDB Endow.* 13, 7 (2020), 992–1005.
- [13] Matteo Interlandi. 2024. Running a Query Optimizer Advisor in Production: What we Learned (and What the Model didn't). [https://hpts.ws/papers/2024/2024\\_session9\\_interlandi.pdf](https://hpts.ws/papers/2024/2024_session9_interlandi.pdf)
- [14] David Justen, Daniel Ritter, Campbell Fraser, Andrew Lamb, Nga Tran, Allison Lee, Thomas Bodner, Mhd Yamen Haddad, Steffen Zeuch, Volker Markl, and Matthias Boehm. 2024. POLAR: Adaptive and Non-invasive Join Order Selection via Plans of Least Resistance. *Proc. VLDB Endow.* 17, 6 (2024), 1350–1363.
- [15] Kyoungmin Kim, Jisung Jung, In Seo, Wook-Shin Han, Kangwoo Choi, and Jaehyok Chong. 2022. Learned Cardinality Estimation: An In-depth Study. In *SIGMOD Conference*. 1214–1227.
- [16] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. 2019. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *CIDR*.
- [17] Kukjin Lee, Anshuman Dutt, Vivek R. Narasayya, and Surajit Chaudhuri. 2023. Analyzing the Impact of Cardinality Estimation on Execution Plans in Microsoft SQL Server. *Proc. VLDB Endow.* 16, 11 (2023), 2871–2883.
- [18] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proc. VLDB Endow.* 9, 3 (2015), 204–215.
- [19] Viktor Leis, Bernhard Radke, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2018. Query optimization through the looking glass, and what we found running the Join Order Benchmark. *VLDB J.* 27, 5 (2018), 643–668.
- [20] Guy Lohman. 2014. Is Query Optimization a “Solved” Problem? <https://wp.sigmod.org/?p=1075>
- [21] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2021. Bao: Making Learned Query Optimization Practical. In *SIGMOD Conference*. 1275–1288.
- [22] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A Learned Query Optimizer. *Proc. VLDB Endow.* 12, 11 (2019), 1705–1718.
- [23] Volker Markl, Peter J. Haas, Marcel Kutsch, Nimrod Megiddo, Utkarsh Srivastava, and Tam Minh Tran. 2007. Consistent selectivity estimation via maximum entropy. *VLDB J.* 16, 1 (2007), 55–76.
- [24] Volker Markl, Nimrod Megiddo, Marcel Kutsch, Tam Minh Tran, Peter J. Haas, and Utkarsh Srivastava. 2005. Consistently Estimating the Selectivity of Conjunctions of Predicates. In *VLDB*. 373–384.
- [25] Amine Mhedhbi and Semih Salihoglu. 2019. Optimizing Subgraph Queries by Combining Binary and Worst-Case Optimal Joins. *Proc. VLDB Endow.* 12, 11 (2019), 1692–1704.
- [26] Guido Moerkotte. 2023. *Building Query Compilers (Draft / Under Construction)*.
- [27] Parimarjan Negi, Ryan Marcus, Andreas Kipf, Hongzi Mao, Nesime Tatbul, Tim Kraska, and Mohammad Alizadeh. 2021. Flow-Loss: Learning Cardinality Estimates That Matter. *Proc. VLDB Endow.* 14, 11 (2021), 2019–2032.
- [28] Hung Q. Ngo. 2018. Worst-Case Optimal Join Algorithms: Techniques, Results, and Open Problems. In *PODS*, Jan Van den Bussche and Marcelo Arenas (Eds.). ACM, 111–124.
- [29] Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. 2018. Worst-case Optimal Join Algorithms. *J. ACM* 65, 3 (2018), 16:1–16:40.
- [30] Mark Raasveldt and Hannes Mühleisen. 2017. Don't Hold My Data Hostage - A Case For Client Protocol Redesign. *Proc. VLDB Endow.* 10, 10 (2017), 1022–1033.
- [31] Tobias Schmidt, Viktor Leis, Peter Boncz, and Thomas Neumann. 2025. SQLStorm: Taking Database Benchmarking into the LLM Era. *Proc. VLDB Endow.* 18, 11 (2025).
- [32] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. 1979. Access Path Selection in a Relational Database Management System. In *SIGMOD Conference*. 23–34.
- [33] Michael Stillger, Guy M. Lohman, Volker Markl, and Mokhtar Kandil. 2001. LEO - DB2's LEarning Optimizer. In *VLDB*. 19–28.
- [34] Gábor Szárnyas, Jack Waudby, Benjamin A. Steer, Dávid Szakállas, Altan Birler, Mingxi Wu, Yuchen Zhang, and Peter A. Boncz. 2022. The LDDB Social Network Benchmark: Business Intelligence Workload. *Proc. VLDB Endow.* 16, 4 (2022), 877–890.
- [35] Immanuel Trummer, Samuel Moseley, Deepak Maram, Saehan Jo, and Joseph Antonakakis. 2018. SkinnerDB: Regret-Bounded Query Evaluation via Reinforcement Learning. *Proc. VLDB Endow.* 11, 12 (2018), 2074–2077.
- [36] Todd L. Veldhuizen. 2012. Leapfrog Triejoin: a worst-case optimal join algorithm. *CoRR* abs/1210.0481 (2012). <http://arxiv.org/abs/1210.0481>
- [37] Adrian Vogelsgesang, Michael Haubenschild, Jan Finis, Alfons Kemper, Viktor Leis, Tobias Mühlbauer, Thomas Neumann, and Manuel Then. 2018. Get Real: How Benchmarks Fail to Represent the Real World. In *DBTest@SIGMOD*. 1:1–1:6.
- [38] Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. 2021. Are We Ready For Learned Cardinality Estimation? *Proc. VLDB Endow.* 14, 9 (2021), 1640–1654.
- [39] Yisu Remy Wang, Max Willsey, and Dan Suciu. 2024. From Binary Join to Free Join. *SIGMOD Rec.* 53, 1 (2024), 25–31.
- [40] Johannes Wehrstein, Timo Eckmann, Roman Heinrich, and Carsten Binnig. 2025. JOB-Complex: A Challenging Benchmark for Traditional & Learned Query Optimization. *CoRR* abs/2507.07471 (2025).
- [41] Lucas Woltmann, Jerome Thiessat, Claudio Hartmann, Dirk Habich, and Wolfgang Lehner. 2023. FASTgres: Making Learned Query Optimizer Hinting Effective. *Proc. VLDB Endow.* 16, 11 (2023), 3310–3322.
- [42] Peizhi Wu and Gao Cong. 2021. A Unified Deep Model of Learning from both Data and Queries for Cardinality Estimation. In *SIGMOD Conference*. 2009–2022.
- [43] Maryann Xue, Yingyi Bu, Abhishek Somani, Wenchen Fan, Ziqi Liu, Steven Chen, Herman Van Hovell, Bart Samwel, Mostafa Mokhtar, Rk Korlapati, Andy Lam, Yunxiao Ma, Vuk Ercegovic, Jiexing Li, Alexander Behm, Yuanjian Li, Xiao Li, Sriram Krishnamurthy, Amit Shukla, Michalis Petropoulos, Sameer Paranjpye, Reynold Xin, and Matei Zaharia. 2024. Adaptive and Robust Query Execution for Lakehouses At Scale. *Proc. VLDB Endow.* 17, 12 (2024), 3947–3959.
- [44] Yifei Yang and Xiangyao Yu. 2025. Accelerate Distributed Joins with Predicate Transfer. *Proc. ACM Manag. Data* 3, 3 (2025), 122:1–122:27.
- [45] Yifei Yang, Hangdong Zhao, Xiangyao Yu, and Paraschos Koutris. 2024. Predicate Transfer: Efficient Pre-Filtering on Multi-Join Queries. In *CIDR*.
- [46] Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. 2020. NeuroCard: One Cardinality Estimator for All Tables. *Proc. VLDB Endow.* 14, 1 (2020), 61–73.
- [47] Mihalís Yannakakis. 1981. Algorithms for acyclic database schemes. In *Proc. VLDB Conference*. VLDB Endowment, 82–94.
- [48] Haozhe Zhang, Christoph Mayer, Mahmoud Abo Khamis, Dan Olteanu, and Dan Suciu. 2025. LpBound: Pessimistic Cardinality Estimation Using  $\ell_p$ -Norms of Degree Sequences. *Proc. ACM Manag. Data* 3, 3 (2025), 184:1–184:27.
- [49] Yunjia Zhang, Yannis Chronis, Jignesh M. Patel, and Theodoros Rekatsinas. 2023. Simple Adaptive Query Processing vs. Learned Query Optimizers: Observations and Analysis. *Proc. VLDB Endow.* 16, 11 (2023), 2962–2975.
- [50] Junyi Zhao, Kai Su, Yifei Yang, Xiangyao Yu, Paraschos Koutris, and Huanchen Zhang. 2025. Debunking the Myth of Join Ordering: Toward Robust SQL Analytics. *Proc. ACM Manag. Data* 3, 3 (2025), 146:1–146:28.
- [51] Jianqiao Zhu, Navneet Potti, Saket Saurabh, and Jignesh M. Patel. 2017. Looking Ahead Makes Query Plans Robust. *Proc. VLDB Endow.* 10, 8 (2017), 889–900.