SHORT-PAPER

LpBound in Action: Cardinality Estimation with One-Sided Guarantees

CHRISTOPH MAYER, University of Zurich, Zurich, ZH, Switzerland

HAOZHE ZHANG, University of Zurich, Zurich, ZH, Switzerland

MAHMOUD ABO-KHAMIS, RelationalAI, Inc., Berkeley, CA, United States

DAN OLTEANU, University of Zurich, Zurich, ZH, Switzerland

DAN SUCIU, University of Washington, Seattle, WA, United States

Open Access Support provided by:

University of Zurich

RelationalAI, Inc.

University of Washington

Published: 22 June 2025

Citation in BibTeX format

SIGMOD/PODS '25: International Conference on Management of Data
June 22 - 27, 2025
Berlin, Germany

Conference Sponsors:
SIGMOD

https://doi.org/10.1145/3722212.3725114
ISBN: 9798400715648

LpBound in Action: Cardinality Estimation with One-Sided Guarantees

Christoph Mayer
christoph.mayer2@uzh.ch
University of Zurich
Department of Informatics
Zurich, Switzerland

Haozhe Zhang
haozhe.zhang@uzh.ch
University of Zurich
Department of Informatics
Zurich, Switzerland

Mahmoud Abo Khamis
mahmoud.abokhamis@relational.ai
RelationalAI
Berkeley, CA, USA

Dan Olteanu
dan.olteanu@uzh.ch
University of Zurich
Department of Informatics
Zurich, Switzerland

Dan Suciu
suciu@cs.washington.edu
University of Washington
Department of Computer Science & Engineering
Seattle, WA, USA

Abstract

We demonstrate LpBound, a cardinality estimator that computes guaranteed upper bounds on the output size of a given query. Among the wealth of traditional, learned, and pessimistic estimators, LpBound's uniqueness lies in its use of two key ingredients: (1) data statistics based on ℓ_p-norms of degree sequences of the join columns, and (2) a linear program formulation of the cardinality estimation problem, whose constraints are the Shannon inequalities and new information inequalities derived from data statistics.

LpBound comes with a visual interface accessible in the browser. The users can interact with the interface by choosing a query from the JOB, STATS, and Subgraph Matching benchmarks and the range of ℓ_p-norms available to LpBound. Within a few milliseconds, LpBound computes an upper bound on the query output size. This bound is explained by a closed-form formula using the available ℓ_p-norms. The users can also inspect the estimation errors of LpBound and a variety of other estimators.

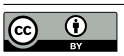
CCS Concepts

• **Information systems → Query optimization**.

Keywords

Cardinality Estimation; Degree Sequence; Lp-norms

ACM Reference Format:
Christoph Mayer, Haozhe Zhang, Mahmoud Abo Khamis, Dan Olteanu, and Dan Suciu. 2025. LpBound in Action: Cardinality Estimation with One-Sided Guarantees . In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3722212.3725114



SIGMOD-Companion '25, Berlin, Germany

ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3725114

1 Introduction

The cardinality estimation problem is to estimate the output size of a query without evaluating the query and using simple, precomputed statistics on the input database [1]. It is a critical component of any database system, as it informs the query optimizer on which evaluation plan to choose for the given query. When these estimates are wrong, the estimation errors may cascade through query plans, potentially leading to orders of magnitude slower evaluation than with a good query plan. Despite decades of work on this problem, it remains the Achilles' heel of query optimizers.

The traditional cardinality estimators are widely used in database systems and go back to System R [9]. They make unrealistic assumptions about the distribution of the data (uniformity, independence, containment of values, and preservation of values). These assumptions allow for simple and efficient estimation yet they often lead to gross underestimates or overestimates of query output sizes, particularly for joins over multiple tables with correlated data [7]. Lack of any theoretical guarantees about their estimation errors is their significant drawback and the main culprit for the degradation in the performance of database systems [6, 7].

The learned cardinality estimators, e.g., [4, 10–12, 14], avoid the unrealistic assumptions of the traditional estimators. Their goal is to learn an accurate model of the underlying distribution in a specific database. This is a promising direction that deserves further innovation to overcome the lack of explainability and updateability, the slow training time, the large model size, and the difficulty to transfer with comparable accuracy to unseen workloads.

First-generation pessimistic cardinality estimators, e.g., [2, 3, 8], compute an upper bound on the query output size. So far, they support a limited class of (acyclic) queries and limited types of data statistics: relation sizes and maximum degree of join attributes. By using only limited input statistics, the existing pessimistic estimators yield significant overestimates and can have worse accuracy than traditional estimators.

In recent work [5, 13], we introduced LpBound, a pessimistic cardinality estimator that provides accurate and guaranteed upper bounds on query output size. Its guarantee avoids the risk of catastrophic cardinality underestimation. LpBound supports a comprehensive range of real-world queries, featuring: acyclic and cyclic

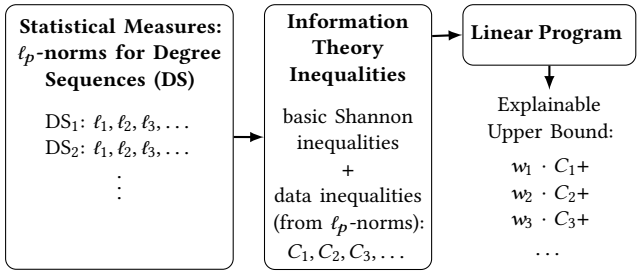


Figure 1: Pipeline of LpBound: statistics on degree sequences are transformed into information-theoretic constraints, which are then compiled into a linear program whose optimal solution is an upper bound on the query output size. The LP solver also provides the weights of each data statistics constraint, which helps explain the effect of the data statistics on the obtained upper bound.

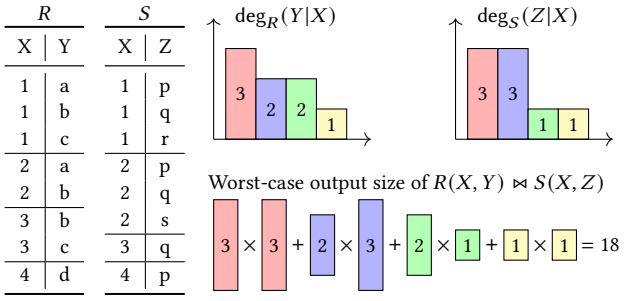


Figure 2: The degree sequences on the columns X in two relations R and S. A guaranteed upper bound on the output size of the join query $Q(X,Y,Z) = R(X,Y) \bowtie S(X,Z)$, is given by the dot product of the degree sequences ordered descendingly.

equi-joins, group-by clauses, and equality and range predicates. Unlike machine learning approaches and like traditional estimators, LpBound's estimates are interpretable, allowing database administrators to understand and validate its estimates. Besides common statistics such as domain sizes, sizes of input relations, most common values and histograms for predicate columns, LpBound critically uses ℓ_p-norms on degree sequences of the join columns.

We demonstrate LpBound using an online interactive system[1]. The users can inspect the outcome of LpBound's *live estimation* for thousands of queries from the established benchmarks JOB, STATS, and Subgraph Matching. For each query, the interface explains the LpBound's upper bound as a formula that is a product of several ℓ_p-norms on degree sequences of join columns. *The users can interact with the interface by choosing which ℓ_p-norms are available to LpBound and observe the effect their choice has on the estimates and how the linear program adapts to the availability of different ℓ_p-norms.* To understand the relationship between ℓ_p-norm values of the degree sequence of a join column, the interface also presents the plot of such values as a function of p, with the p-value picked by LpBound highlighted. To support the comparison of the accuracy of the different estimators, the interface also shows the estimation errors of a variety of estimators, including: pessimistic estimators (LpBound and SafeBound); traditional estimators (Postgres, DuckDB, and a commercial system dubbed DbX); learned estimators based on probabilistic graphical models (DeepDB, BayesCard, and FactorJoin) and on neural networks (NeuroCard and Flat). Whereas the estimates of LpBound are computed on the fly, the estimates of all other estimators were pre-computed and are as reported in a full paper [13]. This is unavoidable, as the estimation time of learned estimators can be 1-2 orders of magnitude slower than LpBound.

2 LpBound in a Nutshell

Fig. 1 shows the three-stage pipeline of LpBound.

In the first stage, LpBound precomputes statistics on the input database: most common values (MCVs) to support equality predicates and histograms to support range predicates. For each MCV, histogram bucket, and full relation, it then computes degree sequences on the join columns and several ℓ_p-norms on these sequences. Such norms are the novelty behind LpBound's estimation. This precomputation is entirely expressed in SQL. It is executed and can be maintained under data updates by the query engine of an underlying database system (LpBound uses DuckDB).

In the second stage, the computed statistics are turned into information inequalities [5]. Together with the Shannon (submodularity and monotonicity) inequalities, they are constraints on the joint entropies of the (subsets of the sets of) variables in the query.

In the third and final stage, these inequalities are compiled into a linear program whose objective is to maximize the joint entropy of the group-by attributes (free variables) of the query. The optimal solution of the linear program is (i) the exact query output size for a "worst-case database" that satisfies the data statistics and (ii) an upper bound on the query output size for any other database satisfying the data statistics, including the input database. The weights of the constraints in the linear program provide an explanation for how the characteristics of the input data influence the estimate.

In the following, we briefly recall degree sequences and their ℓ_p-norms, which are central to LpBound. We then go through one simple join of two relations to showcase LpBound.

2.1 Degree Sequences and ℓ_p-Norms

The *degree* of a value in a column X of a relation R is the number of tuples in R with that X-value. The *degree sequence* of a column X is the list of the degrees of the distinct X-values, $\boldsymbol{d} = (d_1 \geq \cdots \geq d_m)$, where d_1 and d_m are the largest and respectively smallest degrees. The ℓ_p-norm of a degree sequence $\boldsymbol{d}$ is defined as $||\boldsymbol{d}||_p = (d_1^p + \cdots + d_m^p)^{1/p}$. In particular, the ℓ_1-norm is $||\boldsymbol{d}||_1 = \sum_{i=1}^m d_i$, i.e., the relation size, while the ℓ_∞-norm is $||\boldsymbol{d}||_\infty = d_1$, i.e., the maximum degree. Fig. 2 shows two relations $R(X,Y)$ and $S(X,Z)$ and degree sequences $\deg_R(Y|X) = (3,2,2,1)$ and $\deg_S(Z|X) = (3,3,1,1)$ on the join column X in each of the two relations. They count the number of Y-values and respectively Z-values that are paired with each X-value in R and respectively S. The ℓ_1-norm of the degree sequence $\deg_R(Y|X)$ is $||\deg_R(Y|X)||_1 = 3+2+2+1 = 8$, while the ℓ_∞-norm is $||\deg_R(Y|X)||_\infty = \max(3,2,2,1) = 3$.

[1]Demo available at https://www.ifi.uzh.ch/en/dast/research/LpBound.html.

2.2 Upper Bounds on Query Output Size

Consider the join $Q(X,Y,Z) = R(X,Y) \bowtie S(X,Z)$. An upper bound on its output size is given by the dot product of the two degree sequences $\deg_R(Y|X)$ and $\deg_S(Z|X)$, as shown in Fig. 2, so pairing the i-th largest degrees from the two sequences. Yet using at estimation time the full degree sequences can be expensive, especially for large relations. To address this challenge, LpBound approximates the degree sequences by some of their ℓ_p-norms and thereby avoids the computational cost of working with full degree sequences.

Below are some of the bounds that can be produced by LpBound:

$$|Q| \le ||\deg_R(Y|X)||_1 \cdot ||\deg_S(Z|X)||_1$$
$$|Q| \le ||\deg_R(Y|X)||_1 \cdot ||\deg_S(Z|X)||_\infty$$
$$|Q| \le ||\deg_R(Y|X)||_\infty \cdot ||\deg_S(Z|X)||_1$$
$$|Q| \le ||\deg_R(Y|X)||_2 \cdot ||\deg_S(Z|X)||_2$$
$$|Q| \le ||\deg_R(Y|X)||_p \cdot ||\deg_S(Z|X)||_q^{\frac{q}{p(q-1)}} \cdot ||\deg_S(Z|X)||_1^{1-\frac{q}{p(q-1)}} \quad (\tfrac{1}{p} + \tfrac{1}{q} \le 1)$$

The upper bound chosen by LpBound is the minimum among the possible upper bounds and depends on the actual values of the available ℓ_p-norms. Instead of solving a minimization problem, which would potentially require to go over very many possible bounds, LpBound solves instead a dual formulation as a maximization problem via the following linear program:

$$\max\ h(X,Y,Z)$$
$$\text{// basic Shannon inequalities} \qquad (\forall \boldsymbol{U}, \boldsymbol{V} \subseteq \{X,Y,Z\})$$
$$h(\boldsymbol{U}) \le h(\boldsymbol{U} \cup \boldsymbol{V})$$
$$h(\boldsymbol{U} \cup \boldsymbol{V}) + h(\boldsymbol{U} \cap \boldsymbol{V}) \le h(\boldsymbol{U}) + h(\boldsymbol{V})$$
$$h(\emptyset) = 0$$
$$\text{// data statistics inequalities}$$
$$\frac{1}{1} h(X) + h(Y|X) \le \log ||\deg_R(Y|X)||_1$$
$$\frac{1}{2} h(X) + h(Y|X) \le \log ||\deg_R(Y|X)||_2$$
$$\frac{1}{3} h(X) + h(Y|X) \le \log ||\deg_R(Y|X)||_3$$
$$\dots$$
$$\frac{1}{\infty} h(X) + h(Y|X) \le \log ||\deg_R(Y|X)||_\infty$$
$$\text{Similarly for relation } S \text{ using } ||\deg_S(Z|X)||_p$$

The unknowns of the above linear program are the joint entropies $h(\boldsymbol{U})$ of the subsets of the set of query variables $\boldsymbol{U} \subseteq \{X,Y,Z\}$. The entropy function $h : 2^{\{X,Y,Z\}} \to \mathbb{R}_+$ is defined as $h(\boldsymbol{U}) = -\sum_{\boldsymbol{u} \in \mathrm{Dom}(\boldsymbol{U})} P(\boldsymbol{u}) \cdot \log P(\boldsymbol{u})$, where $\boldsymbol{u}$ ranges over the possible outcomes of the tuple $\boldsymbol{U}$ of query variables and $P(\cdot)$ is a finite probability distribution over such possible outcomes.

There are two types of constraints in the above linear program: the basic Shannon inequalities (monotonicity, submodularity, and the entropy of the empty set) and the data statistics inequalities. LpBound connects the data statistics using ℓ_p-norms and the entropies of query variables [5]. This connection is exemplified in the above data statistics inequalities. For any relation $R(X,Y)$, the general form of such inequalities is:

$$\frac{1}{p} h(X) + h(Y|X) \le \log ||\deg_R(Y|X)||_p, \text{ for } p = 1, 2, \dots, \infty.$$

The objective of the above linear program is to maximize the joint entropy of all (free or group-by) query variables, i.e., $h(X,Y,Z)$.

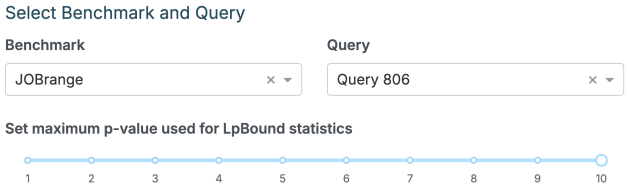


Figure 3: Input panel for selecting a benchmark query and specifying the ℓ_p-norms to include in the estimation.

2.3 Brief Account of Experiments

Experimental evaluation [13] demonstrated LpBound's effectiveness across a wide range of queries and datasets (JOBlight, JOBrange, STATS, Subgraph Matching benchmarks with a few thousand queries). For acyclic queries, LpBound consistently outperforms traditional estimators, which often underestimate due to the uniform distribution assumption. Learned estimators achieve high accuracy on queries and datasets that they have been tuned on (JOBlight), but have portability difficulties or accuracy degradation when used on new benchmarks. For cyclic queries with up to 84 copies of the edge and vertex relations from graph datasets, traditional estimators either underestimate by as much as the true cardinality (e.g., PostgreSQL) or produce bounds that grow exponentially with the number of edge relations, leading to orders of magnitude overestimations (e.g., DuckDB). In contrast, LpBound maintains consistent accuracy across a large set of cyclic queries. When integrated into PostgreSQL's query optimizer, LpBound's estimates lead to execution plans whose performance is close to that of the plans chosen using true cardinalities, particularly for the most expensive queries in the benchmarks. LpBound achieves these improvements while maintaining minimal overhead, with estimation times of just a few milliseconds and a memory footprint of only a few MBs, making it practical for production deployment.

3 User Interaction

LpBound's user interface has two main components: an input panel and a visualization panel for results and explanations.

The input panel allows users to choose from benchmark datasets and queries for cardinality estimation, as shown in Fig. 3 (top). We next exemplify with Query 806 from the JOBrange benchmark:

```
SELECT * FROM movie_companies AS mc, movie_keyword AS mk,
    movie_info_idx AS mi_idx, movie_info AS mi, title AS t
WHERE t.id=mi.movie_id AND t.id=mk.movie_id
    AND t.id=mi_idx.movie_id AND t.id=mc.movie_id
    AND t.production_year<=2012 AND t.kind_id=1
    AND t.phonetic_code>='E4153';
```

This is a star query that joins five relations on `movie_id` and has three range and equality predicates.

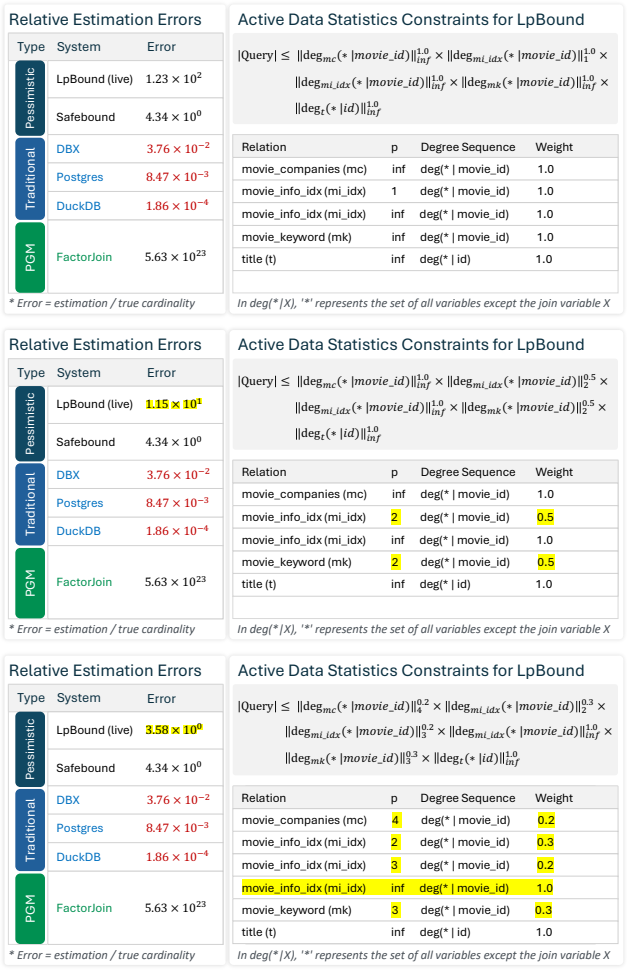


Figure 4: Estimation errors, upper bounds and the active data constraints when different ℓ_p-norms are used. From top to bottom: (1) $\ell_1 + \ell_\infty$, (2) $\ell_1 + \ell_2 + \ell_\infty$, and (3) $\ell_1 + \ldots + \ell_{10} + \ell_\infty$. The changes from (1) to (2) and from (2) to (3) are highlighted.

The input panel also provides a slider in Fig. 3 (bottom) for users to select the ℓ_p-norms from ℓ_1 to ℓ_{10} available to LpBound; ℓ_1 and ℓ_∞ are always included, so that LpBound can always compute an upper bound. By default, the slider is set to ℓ_{10}, meaning LpBound will use $\ell_1, \ldots, \ell_{10}$ and ℓ_∞ norms for computing the upper bound. This interface feature allows users to explore how the selection of ℓ_p-norms affects the estimation accuracy of LpBound.

Once the query and ℓ_p-norms are selected, the backend runs LpBound to compute the cardinality estimation in real time. The visualization panel presents both LpBound's estimates and explanatory insights, alongside precomputed estimates for other estimators for comparison. The estimation errors[2] for Query JOBrange-806 are shown in Fig. 4 (left column) for three configurations from top to bottom: (1) ℓ_1 and ℓ_∞, (2) ℓ_1, ℓ_2, and ℓ_∞, and (3) $\ell_1, \ldots, \ell_{10}, \ell_\infty$. The bound tightens as more norms are used, with the error decreasing from 123x to 11.5x and finally to 3.58x. In contrast, traditional estimators underestimate by 2-4 orders of magnitude, while the PGM-based FactorJoin overestimates by 23 orders of magnitude.

The interface also visualizes how the upper bound is derived from the data statistics constraints, cf. Fig. 4 (right column). When few norms are available (e.g., ℓ_1 and ℓ_∞), the LpBound's options are limited. However, as more ℓ_p-norms are enabled, the linear program combines data constraints with varying weights, leading to progressively tighter and more accurate estimations.

[2]The estimation error is the estimated cardinality divided by the true cardinality. If the true cardinality is 0, then the estimation error is the estimated cardinality.

Acknowledgments

This work was partially supported by NSF-BSF 2109922, NSF IIS 2314527, NSF SHF 2312195, SNSF 200021-231956, and RelationalAI.

References

[1] Mahmoud Abo Khamis, Kyle Deeds, Dan Olteanu, and Dan Suciu. 2025. Pessimistic Cardinality Estimation. *SIGMOD Rec.* 4 (2025), 1–17. https://doi.org/10.1145/3712311.3712313

[2] Walter Cai, Magdalena Balazinska, and Dan Suciu. 2019. Pessimistic Cardinality Estimation: Tighter Upper Bounds for Intermediate Join Cardinalities. In *SIGMOD*. 18–35. https://doi.org/10.1145/3299869.3319894

[3] Jeremy Chen, Yuqing Huang, Mushi Wang, Semih Salihoglu, and Kenneth Salem. 2022. Accurate Summary-based Cardinality Estimation Through the Lens of Cardinality Estimation Graphs. *Proc. VLDB Endow.* 15, 8 (2022), 1533–1545.

[4] 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. https://doi.org/10.14778/3503585.3503586

[5] Mahmoud Abo Khamis, Vasileios Nakos, Dan Olteanu, and Dan Suciu. 2024. Join Size Bounds using ℓ_p-Norms on Degree Sequences. *Proc. ACM Manag. Data* 2, 2 (PODS) (2024), 96. https://doi.org/10.1145/3651597

[6] 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. https://doi.org/10.14778/3611479.3611494

[7] 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. https://doi.org/10.14778/2850583.2850594

[8] Amine Mhedhbi, Chathura Kankanamge, and Semih Salihoglu. 2021. Optimizing One-time and Continuous Subgraph Queries using Worst-case Optimal Joins. *ACM Trans. Datab. Syst.* 46, 2 (2021), 6:1–6:45. https://doi.org/10.1145/3446980

[9] Patricia Selinger, Morton Astrahan, Donald Chamberlin, Raymond Lorie, and Thomas Price. 1979. Access Path Selection in a Relational Database Management System. In *SIGMOD*. 23–34. https://doi.org/10.1145/582095.582099

[10] Ziniu Wu, Parimarjan Negi, Mohammad Alizadeh, Tim Kraska, and Samuel Madden. 2023. FactorJoin: A New Cardinality Estimation Framework for Join Queries. *Proc. ACM Manag. Data* 1, 1 (2023), 41:1–41:27. https://doi.org/10.1145/3588721

[11] Ziniu Wu and Amir Shaikhha. 2020. BayesCard: A Unified Bayesian Framework for Cardinality Estimation. *CoRR* abs/2012.14743 (2020).

[12] 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. https://doi.org/10.14778/3421424.3421432

[13] Haozhe Zhang, Christoph Mayer, Mahmoud Abo Khamis, Dan Olteanu, and Dan Suciu. 2025. LpBound: Pessimistic Cardinality Estimation Using ℓ_p-Norms of Degree Sequences. *Proc. ACM Manag. Data* 3, 3 (SIGMOD) (2025), 184. https://doi.org/10.1145/3725321

[14] Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, and Bin Cui. 2021. FLAT: Fast, Lightweight and Accurate Method for Cardinality Estimation. *Proc. VLDB Endow.* 14, 9 (2021), 1489–1502. https://doi.org/10.14778/3461535.3461539