How Out-of-Bounds Are Your Cardinality Estimates?

Mihail Stoian
University of Technology Nuremberg
Nuremberg, Germany
mihail.stoian@utn.de

Tiemo Bang
Microsoft Gray Systems Lab
Barcelona, Spain
tiemobang@microsoft.com

Hangdong Zhao
Microsoft Gray Systems Lab
Redmond, WA, USA
hangdongzhao@microsoft.com

Jesús Camacho-Rodríguez
Microsoft Fabric DW
Mountain View, CA, USA
jesusca@microsoft.com

Yuanyuan Tian
Microsoft Gray Systems Lab
Mountain View, CA, USA
yuanyuantian@microsoft.com

Andreas Kipf
University of Technology Nuremberg
Nuremberg, Germany
andreas.kipf@utn.de

ABSTRACT

We introduce a soundness checker for database cardinality estimators powered by the recent frameworks for cardinality bounds, xBound and LpBound. Traditional optimizers do not provide any guarantees about their estimates, whereas frameworks such as xBound and LpBound yield provable lower and upper bounds on an operator's cardinality. Our demonstration lets users understand how out-of-bounds cardinality estimators of well-known database systems are, by comparing their cardinality estimates with the corresponding bounds; severe violations indicate an unsound cardinality estimator. Additionally, users can issue ad-hoc queries from open benchmarks and understand the parameter space of xBound along with its effect on the quality of the lower bounds.

PVLDB Reference Format:
Mihail Stoian, Tiemo Bang, Hangdong Zhao, Jesús Camacho-Rodríguez, Yuanyuan Tian, and Andreas Kipf. How Out-of-Bounds Are Your Cardinality Estimates?. PVLDB, 19(12): 4510 - 4513, 2026.
doi:10.14778/3827998.3828052

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/xbound.

1 INTRODUCTION

The last decade in database research has witnessed a proliferation of new techniques for cardinality estimation and the broader challenge of query optimization [2, 3, 5–7, 9, 13–15]. This resurgence was largely sparked by a seminal paper by Leis et al. [8], which showed that estimating relational operator cardinalities remains challenging even for industrial-strength systems. Their work revealed that all tested systems routinely produce large estimation errors, with join size estimation being particularly fragile.

In this demonstration, we do not focus on highlighting Q-errors, i.e., the symmetric ratio between estimated and actual operator cardinalities, which have already been extensively demonstrated in prior literature. Instead, we expose a more fundamental property of modern cardinality estimators: their *soundness*.

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828052

Soundness. We define a cardinality estimator as *sound* if its estimates strictly adhere to mathematically verified bounds. Assume that the true cardinality of a join $A \bowtie B$ lies within a provable range $[\texttt{lb}, \texttt{ub}]$, i.e., $\texttt{lb} \leq |A \bowtie B| \leq \texttt{ub}$. Then if the estimate, $\texttt{est}$, falls within the provided bounds, i.e., $\texttt{est} \in [\texttt{lb}, \texttt{ub}]$, the estimate is *safe* and the estimator is sound for this particular join. Otherwise, the provided estimate is mathematically invalid and thus unsafe.

From a system-wide perspective, the impact of unsafe cardinality estimates extends well beyond the query optimizer. Indeed, unsafe estimates lead to poor plan selection and also severely disrupt downstream, runtime operations [12, §1]. Additionally, an out-of-bounds estimate misleads query schedulers and resource managers into severe under- or over-provisioning of CPU and memory. Guaranteeing the estimator's soundness establishes a reliable foundation across the entire system, ensuring that both plan selection and runtime execution strategies are driven by valid, provable bounds.

Provable Cardinality Bounds. This soundness analysis is only possible due to the line of research on provable cardinality bounds. In this paper, we consider two complementary frameworks: (i) our recently developed framework for lower bounds, xBound [12], and (ii) the state-of-the-art work for upper bounds, LpBound [15].[1] At their core, both frameworks rely on ℓ_p norms on degree sequences. Because deriving tight lower bounds is inherently more challenging, xBound introduces a suite of advanced optimizations to refine them. Illustrating the empirical effect of these optimizations is another key aspect of this demonstration.

Setup. We will test the soundness of both open and closed systems: (i) DuckDB, (ii) PostgreSQL, and (iii) Microsoft's Fabric DW. Our interactive demo includes a suite of open benchmarks, which are supported by both frameworks: (a) JOBlight [7], (b) STATS-CEB [4], and (c) SO-CEB, which is the STATS-CEB benchmark run on the 220 GB StackOverflow dump provided in SQLStorm [11]. On each benchmark, we will also rank the query engines by their soundness. Additionally, users can also inspect the Q-errors when they activate the *bounded* counterparts of the systems, i.e., we clip the system's estimates with the corresponding lower and upper bounds. Indeed, as shown in our previous work [12], this reduces the overall Q-error and leads to measurable query speedups.

The demo, along with the bounds of both frameworks on the benchmarks, is available at `xbound.github.io`. The webpage

[1] The demonstration of xBound is new; LpBound already had a demonstration [10]. We employ LpBound's bounds so that users can *visually* see the range formed by the corresponding lower and upper bounds.

is a read-only version, while our local demonstration allows users to interactively issue custom queries on the open systems and the two provable frameworks. The tool also allows users to add a custom system, enabling them to quantify (and visualize) its soundness.

Contributions. Our demonstration empowers users to:

(1) quantify the extent to which current database systems produce unsound cardinality estimates, measuring the severity of these out-of-bounds violations.
(2) explore the parameter space of xBound, the first framework designed for cardinality lower bounds, to understand how different configurations tighten the resulting bounds.

We next outline the structure of our demonstration, starting with an outline of our framework for lower bounds, xBound, and continuing with the description of both the dashboard where users can issue queries and the leaderboard of the systems regarding their soundness and (improved) Q-errors.

2 CARDINALITY BOUNDS

We first overview how provable frameworks for cardinality bounds work, focusing on our framework for lower bounds, xBound.

Status Quo. The existing literature on provable cardinality bounds, which recently culminated in the LpBound framework [15], has had a major limitation: such estimators can only yield *upper bounds*. While upper bounds are highly effective at fixing *overestimation* by safely capping the optimizer's estimates, they offer no protection against the opposite problem. Consequently, safeguarding query execution against severe *underestimates* remains an open challenge.

In fact, underestimation can lead to catastrophic outcomes. It compromises both plan selection, tricking optimizers into fragile small-data choices like nested-loop or broadcast joins, and runtime execution, triggering out-of-memory errors and massive disk spilling. The severity of this issue is evident in the heavy-tailed distribution of CPU estimation errors in Fabric DW. While >99% of stages face no underestimation (cardinality underestimation does not always lead to CPU underestimation), 1 in 10'000 stages suffers from >70x CPU underallocation, with extreme outliers starved by over a thousandfold [12].

xBound. To bridge this critical gap, in prior work we introduced xBound, the first framework for cardinality lower bounds [12]. xBound uses lightweight base table statistics and dot-product reverse inequalities to first bound the number of distinct keys that will join, boosting this number further with prefixed ℓ_p norms. Like traditional cardinality estimators, all its statistics are built once and can be re-used across the whole workload.

Before describing xBound's internals, let us understand how such size bounds can be inferred from the base tables.

Size Bounds: Intuition. We provide the key idea behind join size bounds. Consider the toy example in Fig. 1, where we show two join columns, $A(X)$ and $B(X)$, with their corresponding degree sequences on the side. A degree sequence consists of the degrees of the key in the respective column, sorted in non-decreasing order. For the sake of presentation, we assume that each key has a join partner in the other table. The join cardinality can be computed as: 2×2 (□) + 1×3 (○) + 1×1 (◇) = 8.

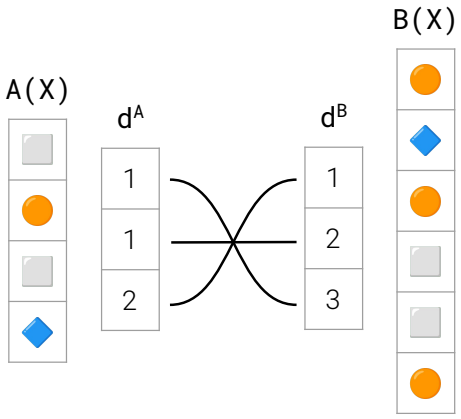


Figure 1: Obtaining a lower bound on the join size, assuming all keys join, by multiplying *cross-wise* the values of the (non-decreasing) degree sequences. This follows from the rearrangement inequality.

The degree sequences depicted on the side do not capture the keys for which the frequencies are attained. The question is thus: What is a valid (non-zero) lower bound in this case? A moment of thought reveals that multiplying *cross-wise* the values in the degree sequences yields the best we can hope for.[2] In this case, the lower bound is $1 \times 3 + 1 \times 2 + 2 \times 1 = 7$. This does have a mathematical argument behind it, namely the *rearrangement inequality*. On the other hand, SafeBound [2, 3] (and, extrapolated, LpBound) uses element-wise multiplication to upper bound the join size. In this example, the upper bound reads: $1 \times 1 + 1 \times 2 + 2 \times 3 = 9$.

Lower Bounds. To lower bound the number of distinct joining keys, xBound stores a `ThetaSketch` [1] per join column and performs a set intersection at optimization time; xBound also supports *hard* lower bounds, which do not rely on sketches [12, Sec. 3.1]. This estimate is boosted by prefixed ℓ_p norms, i.e., statistics such as the Euclidean norm or the largest degree in the join column.

Predicate Support. Currently, our framework comes with support for basic predicate types such as equality, range, conjunction, and disjunction predicates, including the text data type. Equality predicates are supported via storing statistics related to the most common values (MCVs) in a predicate column. Statistics for the range predicates, on the other hand, use equi-width histograms.

Frequent Keys. Since lower bounds assume the worst possible case, we also maintain a heavy partition of frequent keys. Namely, we store statistics related to the heavy hitters of a join column, such as the degree of a given heavy key. To enable support for multiple table predicates in this case, we store per MCV and range bucket a `ThetaSketch` on the row ids. At optimization time, we intersect the sketches of the predicate columns to get a lower bound on the degrees of the heavy keys.

We refer the reader to Ref. [12] for a detailed formalization and further technical details of xBound.

[2] See the visualization; this also inspired our estimator's acronym.

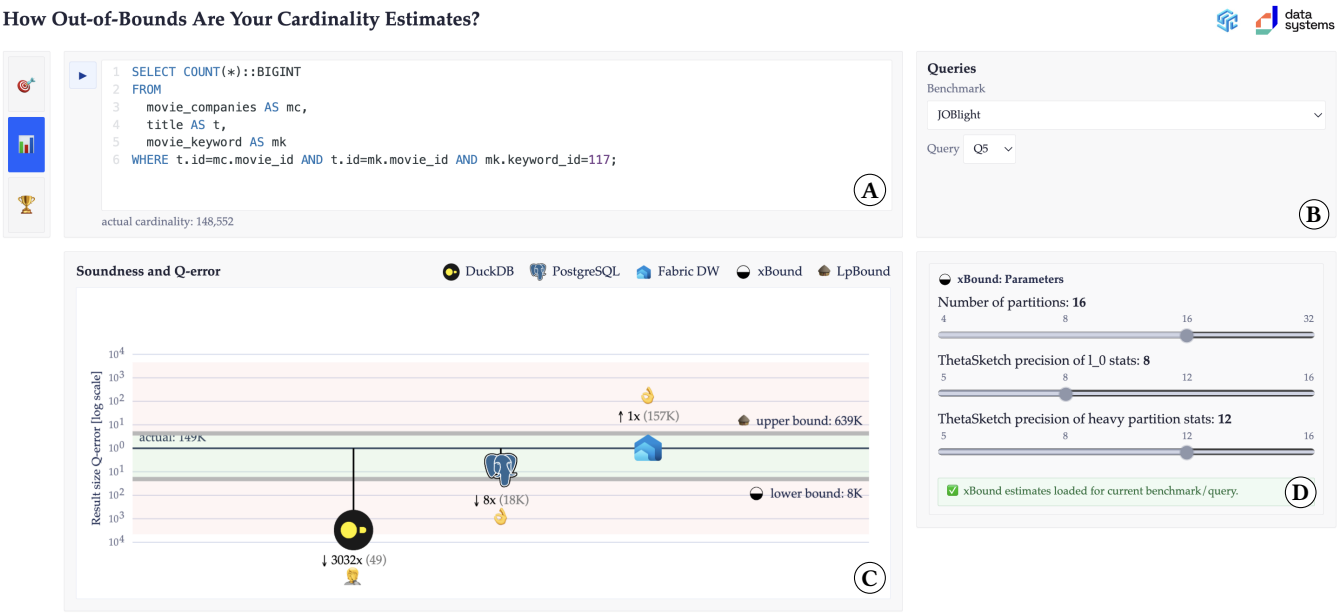


Figure 2: Visualizing the soundness of the systems' estimates with respect to xBound's lower bounds. Users can choose from the benchmark, run a query, and understand the effect of xBound's components on the quality of the lower bounds.

The demo, explained in the following, has the role of both showing the quality of the bounds (for different parameters of the components outlined above) and quantifying how much out-of-bounds the estimates of modern query engines are.

3 DEMONSTRATION

Our demo features (i) an interactive dashboard that allows participants to issue custom SQL queries and check the systems' estimates against the provable bounds provided by xBound and LpBound, and (ii) a live leaderboard that ranks the systems based on the soundness of their cardinality estimates, scoring them according to the severity and frequency of their bound violations.

3.1 Dashboard

The main dashboard helps users visualize the soundness of the systems' estimators. For ease of explanation, the dashboard is split into four areas (A)-(D): The upper part focuses on the SQL input given to the systems, while the bottom part visualizes the result-size Q-errors of the input query ((C)), taking also into account the bounds provided by both xBound and LpBound; xBound also comes with a configurable parameter regime ((D)).

Input SQL. The dashboard provides a set of well-studied benchmarks in area (A): JOBlight [7] (70 queries), STATS-CEB [4] (146 queries), and StackOverflow-CEB [4] (146 queries on the 220 GB dump) from which users can either select from pre-defined benchmark queries or issue a custom one. In the latter case, the dashboard will fetch in real time the estimates from the open systems and xBound; notably, this feature is only provided in the local version.

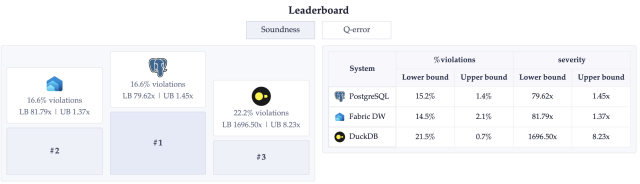
Estimates & Parameter Space. The lower part of the dashboard ((C)) visualizes the systems' (traditional) cardinality estimates, and the cardinality bounds of the two provable frameworks (shown as horizontal lines in the plot). We mark the safe zone and whether the traditional estimates were safe or not for that particular query.

In addition, users can interact with xBound's parameter space ((D)). They can vary three parameters: (a) the number of partitions for the lightweight partition, (b) the precision of the `ThetaSketch` for the ℓ_0 statistics (higher precision also means more space), and (c) the precision of the `ThetaSketch` for the heavy partition. This will update in real time both the lower bound shown in the plot and the space usage of xBound's statistics.

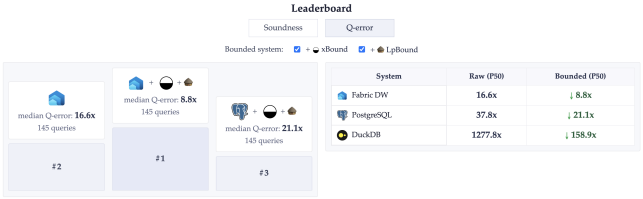
3.2 Systems Leaderboard

A cardinality estimator that frequently produces out-of-bounds estimates can lead to detrimental outcomes at runtime. Using the computed bounds, we can now rank the systems based on how sound their estimators are. This ranking is shown in the second page of the demo, as visualized in Fig. 3a. The ranking incorporates two key metrics: (i) the *violation frequency*, representing the absolute number of queries where a system produced out-of-bounds estimates, and (ii) a *severity score*, quantifying the magnitude of these violations. The severity score is computed as the average relative error between the invalid estimates and the provable bounds. Furthermore, maintaining continuity with the query dashboard, this dedicated ranking view also includes an interactive parameter-space tuner (omitted from the screenshot).

Bounded Systems. As shown in our previous work [12], clipping the system's estimates with the lower bound reduces the magnitude of underestimation. Therefore, we also show, in Fig. 3b, the improvement in the overall Q-error of the systems' estimates when enabling what we term xBound-ed systems; the results are for the StackOverflow-CEB benchmark. For instance, while the median



(a) The soundness of cardinality estimators of different systems.



(b) Effect of clipping with bounds on the overall Q-error.

Figure 3: Systems leaderboard page showing (a) the soundness of their cardinality estimators and (b) the effect of clipping with lower and upper bounds on the benchmark's median Q-error (in this case StackOverflow-CEB).

Q-error of Fabric DW is 16.8x, its bounded counterpart reduces it to 8.8x, making it to the top of the leaderboard.

3.3 Demo Session

We outline how we will interact with demo session participants.

Motivation. We will show a plot with production insights from Fabric Data Warehouse (as shown on `xbound.github.io`), which shows the severity of cardinality *underestimation* for the resource estimator. This serves as the motivation for our work on lower bounds.

Demo. We will then invite them to try out the actual demo and see how their favorite database system performs, i.e., how out-of-bounds it is. As described in §3.1, they can choose a benchmark and a query, and inspect the gap between the system's estimate and the corresponding lower and upper bounds. After exploring two queries, participants are redirected to the leaderboard to understand the effect of clipping with lower bounds, which is a lightweight technique to reduce underestimation without replacing the systems' cardinality estimator.

4 CONCLUSION

We introduced an interactive demo that quantifies the soundness of cardinality estimators with respect to cardinality bounds. Powered by the recent frameworks xBound and LpBound, our demo shows that, after decades of research, we (finally) have guaranteed ranges for operator cardinalities. We anticipate a lively demo session where we hope to convince the community that robust query optimizers are just around the corner.

REFERENCES

[1] Anirban Dasgupta, Kevin J. Lang, Lee Rhodes, and Justin Thaler. 2016. A Framework for Estimating Stream Expression Cardinalities. In *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016 (LIPIcs)*, Wim Martens and Thomas Zeume (Eds.), Vol. 48. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:17. https://doi.org/10.4230/LIPICS.ICDT.2016.6

[2] Kyle Deeds, Dan Suciu, Magda Balazinska, and Walter Cai. 2023. Degree Sequence Bound for Join Cardinality Estimation. In *26th International Conference on Database Theory, ICDT 2023, Ioannina, Greece, March 28-31, 2023 (LIPIcs)*, Floris Geerts and Brecht Vandevoort (Eds.), Vol. 255. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 8:1–8:18. https://doi.org/10.4230/LIPICS.ICDT.2023.8

[3] 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. https://doi.org/10.1145/3588907

[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] Roman Heinrich, Manisha Luthra, Johannes Wehrstein, Harald Kornmayer, and Carsten Binnig. 2025. How Good are Learned Cost Models, Really? Insights from Query Optimization Tasks. *Proc. ACM Manag. Data* 3, 3 (2025), 172:1–172:27. https://doi.org/10.1145/3725309

[6] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2020. DeepDB: Learn from Data, not from Queries! *Proc. VLDB Endow.* 13, 7 (2020), 992–1005. https://doi.org/10.14778/3384345.3384349

[7] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. 2019. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2019/papers/p101-kipf-cidr19.pdf

[8] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter 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

[9] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A Learned Query Optimizer. *CoRR* abs/1904.03711 (2019). arXiv:1904.03711 http://arxiv.org/abs/1904.03711

[10] 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/PODS 2025, Berlin, Germany, June 22-27, 2025*, Volker Markl, Joseph M. Hellerstein, and Azza Abouzied (Eds.). ACM, 187–190. https://doi.org/10.1145/3722212.3725114

[11] Tobias Schmidt, Viktor Leis, Peter Boncz, and Thomas Neumann. 2025. SQLStorm: Taking Database Benchmarking into the LLM Era. *Proceedings of the VLDB Endowment* 18, 11 (2025), 4144–4157.

[12] Mihail Stoian, Tiemo Bang, Hangdong Zhao, Jesús Camacho-Rodríguez, Yuanyuan Tian, and Andreas Kipf. 2026. The Case for Cardinality Lower Bounds. arXiv:2601.13117 [cs.DB] https://arxiv.org/abs/2601.13117

[13] 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, Article 41 (May 2023), 27 pages. https://doi.org/10.1145/3588721

[14] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep Unsupervised Cardinality Estimation. *Proc. VLDB Endow.* 13, 3 (2019), 279–292. https://doi.org/10.14778/3368289.3368294

[15] 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 (2025), 184:1–184:27. https://doi.org/10.1145/3725321