CorrBound: Cardinality Estimation Accounting for Inter- and Intra-relation Correlations

CHRISTOPH MAYER, University of Zurich, Switzerland
HAOZHE ZHANG, University of Zurich, Switzerland
MAHMOUD ABO KHAMIS, RelationalAI, USA
KYLE DEEDS, Boston University, USA
DAN OLTEANU, University of Zurich, Switzerland
DAN SUCIU, University of Washington, USA

This paper introduces CorrBound, a cardinality estimator that exploits the correlations between the join columns across and within relations. The state-of-the-art estimators do not observe such correlations and can therefore yield poor estimates. By explicitly accounting for correlations, CorrBound can significantly improve the accuracy of cardinality estimation.

CorrBound supports multi-join queries (acyclic or cyclic) with conjunctions of equality and range predicates and group-by clauses. Its estimate is the optimal solution of a linear program whose constraints encode data statistics and Shannon inequalities. It uses a new information inequality that captures the intra- and inter-relation correlations of join columns and uses the generalized inner product of degree vectors of join columns. This inequality also captures the ℓ_p-norm inequality used by the state-of-the-art estimator LpBound.

The inequality comes with a high-dimensionality challenge: managing the cardinality tensor defined by the generalized inner products of many large degree vectors. To keep the estimation time feasible, CorrBound uses two compression techniques that retain the accuracy of the inner products: sketches of degree vectors and low-rank decomposition of the cardinality tensor.

We experimentally evaluate CorrBound against traditional, pessimistic, and machine learning-based estimators on the JOBlight and STATS, and subgraph matching benchmarks. Our main finding is that CorrBound can be more accurate than the state of the art while maintaining a low estimation time.

CCS Concepts: • **Information systems → Query optimization**.

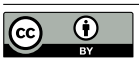
Additional Key Words and Phrases: Cardinality Estimation, Degree Sequence, Information Theory

ACM Reference Format:
Christoph Mayer, Haozhe Zhang, Mahmoud Abo Khamis, Kyle Deeds, Dan Olteanu, and Dan Suciu. 2026. CorrBound: Cardinality Estimation Accounting for Inter- and Intra-relation Correlations. *Proc. ACM Manag. Data* 4, 1 (SIGMOD), Article 19 (February 2026), 27 pages. https://doi.org/10.1145/3786633

1 Introduction

The goal of Cardinality Estimation is to provide an accurate estimate of the query output size, using only precomputed statistics on the database. This estimate is the central metric that guides cost-based query optimization. The optimizer uses cardinality estimates for a variety of purposes, ranging

Authors' Contact Information: 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; Kyle Deeds, kdeeds@bu.edu, Boston University, Department of Computer Science, Boston, MA, 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.



ACM 2836-6573/2026/2-ART19
https://doi.org/10.1145/3786633

from plan selection to decisions on memory allocation. Unfortunately, cardinality estimation is notoriously difficult. Most database systems in use today rely on density-based estimators, pioneered by System R [42], but these have been shown to often underestimate significantly due to their strong reliance on assumptions like data uniformity and independence [32]. Alternative approaches come with their own limitations: sampling-based estimators require expensive data access at estimation time, while learned estimators suffer from large memory footprints and long training time [12, 28, 50].

Pessimistic cardinality estimation (PCE) [7, 11, 12, 54] computes a guaranteed *upper bound* on the cardinality. The challenge of pessimistic estimators is to come as close as possible to the true cardinality, while always guaranteeing an upper bound. They rely on information theory to compute the upper bound, following techniques introduced in the theory community. The first such bound, called the AGM bound [6], uses only the size of the input tables. But this information is insufficient to produce an accurate bound, for example the AGM bound on the size of a join $R \bowtie S$ is the product of the sizes of the two relations, $|R| \cdot |S|$. Subsequent work improved the accuracy of the estimate by incorporating increasingly detailed statistics into the information inequalities: functional dependencies [2, 19], maximum degrees [3, 7], and ℓ_p-norms of degree sequences [1, 27, 54]. While the theory of pessimistic estimators is concerned only with the information inequalities leading to the upper bound, the systems that implement them refine the statistics to support queries with selection predicates, e.g. by using histograms and most common values (MCVs). In addition to statistics on the full relation R, these systems compute for instance statistics on subsets of the form $\sigma_{X=a}(R)$, where a is an MCV value.

However, while existing pessimistic estimators capture skew effectively, none of them account for correlations in the data. These correlations can occur both *intra relation*, e.g. when the degree sequences of $R.X$ and $R.Y$ are correlated, and *inter relations*, when the degree sequences of $R.X$ and $S.X$ are correlated. Lacking information about correlations, pessimistic estimators must assume the worst possible correlations when computing the upper bound.

In this paper, we introduce CorrBound, a pessimistic cardinality estimator for (acyclic and cyclic) queries with equi-joins, equality and range predicates, and group-by clauses. A defining feature of CorrBound is that it accounts for both intra- and inter-relation correlations and incorporates them naturally into the information-theoretic cardinality estimation framework. At its core lies a novel information inequality that upper bounds an entropic expression in terms of simple statistics that capture these correlations and are efficient to compute. This inequality is introduced in Sec. 3. It strictly generalizes all previously known inequalities, including the degree constraints [3] and the ℓ_p-norms [1, 54]. To give some intuition on the new inequality, we illustrate with two examples.

Example 1.1 (Intra-relation Correlations). Consider some complex query that involves a relation $R(X, Y, Z)$. The state-of-the-art estimator LpBound [54] would store the ℓ_2-norms $||\mathsf{deg}_R(YZ|X)||_2$ and $||\mathsf{deg}_R(XZ|Y)||_2$ (reviewed in Sec. 2) and use them to compute a bound on the query's cardinality. However, the estimator must make the worst case assumption about the correlation between the values of $R.X$ and $R.Y$. It must assume that the most frequent value $X = a$ co-occurs with the most frequent value $Y = b$, etc. In our new approach, we precompute a new statistic, $\sum_{(a,b,c)\in R} \mathsf{deg}_R(YZ|X = a) \cdot \mathsf{deg}_R(XZ|Y = b)$, that captures the correlation between the frequencies of X and Y. The question is how to use this new statistic in the general information-theoretic framework to compute an upper bound. The answer is given by our novel inequality, described in Sec. 3: in this example, the inequality states that the logarithm of this statistic upper bounds the entropic term $h(XYZ) + h(YZ|X) + h(XZ|Y)$. By using this additional inequality, the information-theoretic framework can compute an improved (smaller) upper bound, since it has more refined information on the correlation between $R.X$ and $R.Y$.

Example 1.2 (Inter-relation Correlations). Consider some complex query that involves, among other things, the natural join of k relations $R_1(X, Y_1), \ldots, R_k(X, Y_k)$. To estimate the output size, LpBound [54] uses statistics of the form $||\boldsymbol{d}^{(j)}||_p$, where $\boldsymbol{d}^{(j)} \stackrel{\text{def}}{=} \deg_{R_j}(Y_j|X)$ is the *degree vector* of the join column $R_j.X$, i.e., the vector of degrees of the X-values in relation R_j, for $j \in [k]$. Again, the estimator must make the worst case assumption on the correlation of the values X, e.g. it will assume that the most frequent value $X = i$ is the same in all relations. CorrBound computes a new statistic, which is the generalized inner product:

$$\left\langle \boldsymbol{d}^{(1)}, \ldots, \boldsymbol{d}^{(k)} \right\rangle \stackrel{\text{def}}{=} \sum_{i=1}^{n} \prod_{j=1}^{k} \boldsymbol{d}^{(j)}[i], \tag{1}$$

Our novel inequality proves that the logarithm of this inner product upper bounds the entropic term $h(X) + \sum_{i=1}^{k} h(Y_i|X)$, and this allows the quantity in Eq. (1) to be used by the information-theoretic framework to improve the cardinality upper bound.

We note that the intra-relation correlations captured by our new formula are different from the correlations traditionally captured with 2-dimensional histograms and more sophisticated graphical models [13, 18, 46]. In that setting, the challenge is to capture the correlation between traditional statistics (cardinalities), given selections on two or more attributes. For example, capturing the correlation between $|\sigma_{A=a}(R)|$ and $|\sigma_{B=b}(R)|$ for various values a, b requires constructing a 2-dimensional histogram on the attributes A, B. In contrast, we address correlations between foreign keys, say $R.X, R.Y$, and this requires computing a single scalar quantity, as opposed to an entire histogram. Our technique is orthogonal to the use of 1- or 2-dimensional histograms.

Challenges Incorporating our novel inequality into a practical system, however, faces two important feasibility challenges.

Challenge 1. *How to efficiently use the inter-relation statistics for queries with selection predicates?*

Precomputing and storing the quantity in Eq. (1) is not a problem. However, in order to support effectively queries with equality and range predicates, we need to refine the statistics on each relation R to subsets of R defined by a Most Common Value (MCV) $A = a$ and by a histogram bucket $\ell \le A \le r$. For each relation R_j, the number s_j of such refined statistics is typically $s_j = 10 - 5000$. Doing this for all k relations in Eq. (1) is prohibitive, because it requires storing $O(\prod_{j=1}^{k} s_j)$ separate statistics, which is exponential in k.

Challenge 2. *How to achieve a low estimation time?*

In its most general setting, the information-theoretic estimate requires solving a linear program with a number of unknowns that is exponential in the size of the query. This is prohibitive for the estimation time. Special cases have been identified in the literature that reduce the number of unknowns to quadratic, or even linear in the size of the query. The challenge is to extend those approaches to the new information inequality.

Our Solutions We explored two approaches to Challenge 1.

First, we explore using Fast-AMS sketches [4, 10] to approximate the inner products: we discuss this in Sec. 4. The advantage is that for each relation R_j we need to pre-compute and store only s_j sketches: this reduces the memory footprint from $O(\prod_j s_j)$ to $O(\sum_j s_j)$. At estimation time, the system retrieves the k sketches corresponding to the selection predicates, and uses them to estimate the value in Eq. (1), which further feeds into the information-theoretic framework. Although this first solution shows that sketches can be naturally incorporated into our information-theoretic estimation framework, we found that it has two disadvantages: the space/accuracy tradeoff of the

Fast-AMS sketch requires a large memory footprint to achieve our desired accuracy. Second, we lose the strong upper bounds guarantee.

Our second solution, described in Sec. 5, is to store the degree vectors themselves, separately for each relation R_j; there will be one separate degree vector, $\boldsymbol{d}_1^{(j)}, \ldots, \boldsymbol{d}_{s_j}^{(j)}$ for each of the s_j refinements of R_j. At estimation time, the estimator knows the selection predicates, fetches the corresponding degree vector for each R_j, and computes the value in Eq. (1). However, degree vectors are large, and it is not practical to store them completely. Our solution is to compress them to significantly lower the memory footprint and estimation time. For this purpose, we consider the tensor defined by all combinations of refinements of $R_1, \ldots, R_k$:

$$\boldsymbol{T}[\ell_1, \ldots, \ell_k] \stackrel{\text{def}}{=} \left\langle \boldsymbol{d}_{\ell_1}^{(1)}, \ldots, \boldsymbol{d}_{\ell_k}^{(k)} \right\rangle, \text{ for } \ell_j \in [s_j], j \in [k] \tag{2}$$

We use low-rank tensor decomposition to compress this cardinality tensor in Eq. (2), while still allowing to estimate efficiently its entries. We describe a simple extension of the tensor decomposition that guarantees our estimates are upper bounds, thus allowing the general information-theoretic framework to continue to guarantee an upper bound on the cardinality of the query output.

Example 1.3. The JOBlight benchmark [33] has queries that join up to $k = 5$ relations on movie_id and have selection predicates on each relation. There are 2.5M distinct movie identifiers in the database. The 5-way cardinality tensor has more than 10^{13} entries[1], so it is not feasible to compute and store it. We achieve almost six orders of magnitude compression through three steps:

(1) **Lossless tensor decomposition:** We reformulate this 5-way tensor as a summation of 2.5M rank-1 tensors. This reformulation essentially stores separately the degree vectors for each relation and only has 2.67×10^{10} entries (380× compression).
(2) **Truncation:** We further keep the 250,000 most important rank-1 tensors in the decomposition, compressing the tensor to 2.64×10^9 entries (10× further compression).
(3) **Sparse storage:** Since the degree vectors are sparse, we only need to store the 1.52×10^7 non-zero entries (174× further compression).

We thus only store a lossy decomposition of the cardinality tensor that has 6.7×10^5 times fewer entries than the tensor itself. The experiments in Sec. 7 show that this compression comes at a modest cost: the median estimation error increases from 1.37x to 2.5x; for context, LpBound's median estimation error is 6.5x.

To address Challenge 2, we describe in Sec. 6 our solution for reducing the size of the linear program. We show that, if the query is Berge-acyclic, then the number of unknowns in the linear program can be decreased from exponential, to linear in the size of the query (number of relations and attributes). More precisely, we show that the method described in [54] continues to apply even when we use the new inequality. For general queries, the number of unknowns can be reduced to quadratic in the size of the query, by replacing the linear program with a network flow graph. This extends a prior approach [25, 54] to support our new inequality.

Finally, in Sec. 7, we report on experiments with real and synthetic workloads. CorrBound can be more accurate than the state-of-the-art LpBound pessimistic cardinality estimator at the expense of additional storage for compressed degree vectors. The reason for the accuracy improvement and extra storage is CorrBound's account of correlations of join columns, which is missing in LpBound. CorrBound takes a few milliseconds to estimate the cardinality of all connected subqueries of a given query. We also report on experiments with the state-of-the-art sketch-based cardinality estimator [21], traditional estimators used in Postgres, DuckDB, and a commercial database

[1]This calculation is based on the following numbers of MCVs for five relations in JOBlight, as used in prior experiments with LpBound [54] and also in Sec. 7: 12, 5001, 72, 5001, and 473.

system, as well as with variants of CorrBound that either compute the exact inner products, use sketches, or use tensor decomposition.

Related Work Beyond pessimistic cardinality estimation, there are three modern approaches to cardinality estimation: sketching, sampling, and machine-learning. Prior work has used sketches as an end-to-end solution for cardinality estimation, primarily in the streaming setting [4, 21, 26, 40, 45]. These approaches support high update throughput and often provide confidence intervals. However, they typically assume prior knowledge of the query. This allows them to filter tuples before inserting them into sketches, avoiding the use of histograms and most-common-value lists [21, 26]. Further, existing sketching techniques lack support for cyclic queries and queries with group-by clauses.

Work on sampling can be divided into offline and online methods. In the former, each table is sampled once before estimating many queries [9, 16, 17, 47, 49]. While accurate for single-table queries, offline approaches struggle with joins because the table samples often fail to produce any join results. Online sampling improves join estimation by increasing the odds of sampling join results [23, 29, 32, 35]. However, this requires accessing indices on join columns at estimation time, introducing engineering and latency issues.

Machine learning approaches learn a model to predict the cardinality of queries. Query-based techniques leverage query-size pairs to directly perform this estimation [30, 34, 37]. Data-based techniques instead model the data distribution directly then use this to estimate cardinalities [14, 22, 51, 53]. These approaches are promising but often struggle with long training times, large memory footprints, and lack support for group-by and cyclic queries [28, 50].

2 Background

Queries. We review briefly background material from recent work [7, 11, 12, 54]. We fix an ordered domain Dom, and consider a relational schema where each relation R has a set of attributes, denoted $\text{attrs}(R)$. We consider multi-join queries (acyclic and cyclic) with equi-joins, equality and range selection predicates, and group-by clauses. Equivalently, these are single-block SQL queries:

```
SELECT [groupby-attrs]
FROM R_1, R_2, ..., R_m
WHERE [join-and-selection-predicates]
GROUP BY [groupby-attrs]
```

where the `WHERE` clause consists of equi-joins and conjunctions of equality and range predicates. We do not support subqueries. For the purpose of cardinality estimation, aggregates in the `SELECT` clause do not matter and we do not discuss them. A *full query* is a query where the [groupby-attrs] are all attributes; in that case we omit the `GROUP BY` clause, and write the query equivalently as `SELECT *` ... For the purpose of cardinality estimation, a query under bag semantics is equivalent to a full query; they are covered by our framework and we will not discuss them separately.

To reduce clutter, we will write the query using the syntax from Conjunctive Queries:

$$Q(\text{groupby-attrs}) = R_1(\text{attrs}_1) \wedge \cdots \wedge R_m(\text{attrs}_m) \tag{3}$$

Degree Vectors. Fix a relation R and two sets of attributes $X, Y \subseteq \text{attrs}(R)$. The *degree from X to Y* of a value $X = a$ is the number of distinct Y-values that occur with a, i.e. $\deg_R(Y|X = a) \stackrel{\text{def}}{=} |\Pi_Y(\sigma_{A=a}(R))|$. Let $a_1, \ldots, a_n$ be all values in the active domain of X, sorted using the domain order, and let $d_i \stackrel{\text{def}}{=} \deg_R(Y|X = a_i)$. Then, the *degree vector from X to Y* is $\deg_R(Y|X) = (d_1, \ldots, d_n)$. For example, if the functional dependency $X \rightarrow Y$ holds in R, then $\deg_R(Y|X) = (1, 1, \ldots, 1)$, and if all tuples in R have the same value $R.X$ (and $n = 1$), then $\deg_R(Y|X) = (d)$, where $d = |\Pi_Y(R)|$. Notice that the definition of $\deg_R(Y|X)$ remains unchanged if we include X in the set Y, i.e. $\deg_R(Y|X) =$

$\mathsf{deg}_R(XY|X)$. If Y is the set of all attributes of R then we denote $\mathsf{deg}_R(Y|X)$ by $\mathsf{deg}_R(*|X)$. Finally, assume Z is another attribute, or set of attributes of R, and let $b \in R.Z$ be one of its values. Then, the degree vector from X to Y *conditioned* on a value $Z = b$ is $\mathsf{deg}_R(Y|X, Z = b) \stackrel{\text{def}}{=} \mathsf{deg}_{\sigma_{Z=b}(R)}(Y|X)$.

Both traditional and newer cardinality estimators collect statistics on the input tables that can be described in terms of ℓ_p-norms of degree vectors. Recall that, if $\boldsymbol{d} = (d_1, \ldots, d_n)$ is an n-dimensional vector, then its ℓ_p norm is $||\boldsymbol{d}||_p \stackrel{\text{def}}{=} \left(\sum_{i\in[n]} d_i^p\right)^{1/p}$. Existing pessimistic cardinality estimators use the following statistics: the *cardinality* of a relation R, which is $||\mathsf{deg}_R(*|X)||_1$; the maximum degree of the attribute X (i.e. the largest frequency of any value $X = a$ in R), which is $||\mathsf{deg}_R(*|X)||_\infty$; the domain size of $R.X$, which is $||\mathsf{deg}_R(\emptyset|X)||_1$ and also equal to $||\mathsf{deg}_R(X|\emptyset)||_1$; other ℓ_p-norms give insights into the distribution of the values of X. For example, the system may precompute the quantity $||\mathsf{deg}_R(*|X)||_2^2$; if $R.X$ is a key, then this quantity is equal to $|R|$; if $R.X$ has a single value, then this quantity is equal to $|R|^2$; anything in between gives an indication of the degree of skewness of $R.X$.

In this paper we assume that the system has a fixed set of statistics on the input database:

$$N_1 = ||\mathsf{deg}_{R_{j_1}}(Y_1|X_1)||_{p_1}^{p_1} \cdots N_s = ||\mathsf{deg}_{R_{j_s}}(Y_s|X_s)||_{p_s}^{p_s} \tag{4}$$

We denote the set of conditionals and their norms by $\Delta \stackrel{\text{def}}{=} \{(Y_1, X_1, p_1), \ldots, (Y_s, X_s, p_s)\}$, and by $\boldsymbol{N} \stackrel{\text{def}}{=} (N_1, \ldots, N_s)$, and refer to the pair $(\Delta, \boldsymbol{N})$ as the *statistics on the input database*.

Example 2.1. Consider the relation R below:

$R =$

X	Y	Z
a	1	10
a	1	20
b	1	10
b	1	20
b	2	20
c	1	10

$\mathsf{deg}_R(*|X) = (2, 3, 1)$ $\quad$ $\mathsf{deg}_R(Y|X) = (1, 2, 1)$

$||\mathsf{deg}_R(*|X)||_1 = 6$ $\quad$ $||\mathsf{deg}_R(Y|X)||_1 = 4$

$||\mathsf{deg}_R(*|X)||_\infty = 3$ $\quad$ $||\mathsf{deg}_R(Y|X)||_\infty = 2$

$||\mathsf{deg}_R(*|X)||_2^2 = 14$ $\quad$ $||\mathsf{deg}_R(Y|X)||_2^2 = 6$

The database statistics are $(\Delta, \boldsymbol{N})$, where $\Delta = \{(*, X, 1), (Y, X, 1), (*, X, \infty), (Y, X, \infty), (*, X, 2), (Y, X, 2)\}$ and $\boldsymbol{N} = (6, 4, 3, 2, 14, 6)$.

We note that degree vectors are different from the *degree sequences* used in prior work, which are ordered in decreasing order of the degrees rather than the domain order. While degree sequences compress very effectively [12], the advantage of degree vectors is that they capture inter-relation correlations. We demonstrate this with a simple example.

Example 2.2. Consider the join of two relations $R \bowtie_{R.X=S.X} S$. Suppose the domain of X is $\{1, 2, 3\}$, and the degree vectors of $R.X$ and $S.X$ are as follows:

i	1	2	3
$\mathsf{deg}_R(*\|X = i)$	5	1	10
$\mathsf{deg}_S(*\|X = i)$	2	4	0

Then the size of the join is exactly the inner product of the degree vectors, $\langle \mathsf{deg}_R(*|X), \mathsf{deg}_S(*|X)\rangle = 5 \cdot 2 + 1 \cdot 4 + 10 \cdot 0 = 14$, because the degree vectors capture perfectly the inter-relation correlations. In contrast, the degree sequences are sorted in decreasing order of the degrees, but their dot product is only an upper bound on the join size. In our example the degree sequences are $(10, 5, 1)$ and $(4, 2, 0)$ and their inner product is $10 \cdot 4 + 5 \cdot 2 = 50$.

Information Theory. Our work is based on information inequalities that can be used to derive upper bounds on the query output size. Consider a random variable X with n outcomes $a_1, \ldots, a_n$ and probabilities $p_1, \ldots, p_N$. The entropy of X is:

$$h(X) \stackrel{\text{def}}{=} -\sum_{i\in[n]} p_i \log p_i$$

where all logarithms are in base 2. We consider multiple, joint random variables, e.g., X, Y, Z, and the entropy of each subset of the random variables, e.g., $h(X), h(Y), h(XY)$. Shannon [43] proved two basic inequalities satisfied by the entropies of joint random variables, called *monotonicity* and *submodularity*:

$$h(\boldsymbol{UV}) \geq h(\boldsymbol{U}) \qquad h(\boldsymbol{UV}) + h(\boldsymbol{UW}) \geq h(\boldsymbol{U}) + h(\boldsymbol{UVW}) \tag{5}$$

where $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ are three sets of random variables. The *conditional entropy* is defined as $h(\boldsymbol{V}|\boldsymbol{U}) \stackrel{\text{def}}{=} h(\boldsymbol{UV}) - h(\boldsymbol{U})$. The basic Shannon inequalities (5) can be rewritten as:

$$h(\boldsymbol{V}|\boldsymbol{U}) \geq 0 \qquad h(\boldsymbol{V}|\boldsymbol{U}) \geq h(\boldsymbol{V}|\boldsymbol{UW})$$

Statistics Constraints. Consider a relation R with attribute set $\boldsymbol{V}$, and assume any probability distribution on its tuples. Then the following inequalities hold between the entropic values and statistics on the relation R, where $\boldsymbol{X}, \boldsymbol{Y} \subseteq \boldsymbol{V}$ are sets of attributes:

$$\begin{aligned} h(\boldsymbol{V}) &\leq \log|R| \\ h(\boldsymbol{Y}|\boldsymbol{X}) &\leq \log\left(||\mathsf{deg}_R(\boldsymbol{Y}|\boldsymbol{X})||_\infty\right) \\ h(\boldsymbol{X}) + p\cdot h(\boldsymbol{Y}|\boldsymbol{X}) &\leq \log\left(||\mathsf{deg}_R(\boldsymbol{Y}|\boldsymbol{X})||_p^p\right) \end{aligned} \tag{6}$$

This connection between information-theoretic terms and database statistics can lead to tight upper bounds on the query output size, as captured by the following theorem:

Theorem 2.3. *[1] Consider the conjunctive query Q given by* (3)*, and denote its output attributes by $\boldsymbol{Z}$. Suppose we have stored the statistics $N_1, \ldots, N_s$ shown in* (4)*. Then, if there exist non-negative numbers $w_1, \ldots, w_s \geq 0$ such that the following is a valid information-theoretic inequality implied by Shannon's basic inequalities:*

$$h(\boldsymbol{Z}) \leq \sum_{\ell\in[s]} w_\ell \left(h(\boldsymbol{X}_\ell) + p_\ell h(\boldsymbol{Y}_\ell|\boldsymbol{X}_\ell)\right) \tag{7}$$

then the following upper bound holds on the size of the output to the query on the database satisfying these statistics:

$$|Q| \leq N_1^{w_1} N_2^{w_2} \cdots N_s^{w_s}$$

Example 2.4. Consider a single join $Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z)$. Assume we have the statistics:

$$N_1 = ||\mathsf{deg}_R(X|Y)||_2 \qquad N_2 = ||\mathsf{deg}_S(Z|Y)||_2$$

We use the following valid information inequality:[2]

$$2h(XYZ) \leq (h(Y) + 2h(X|Y)) + (h(Y) + 2h(Z|Y))$$

If we divide by 2, we then obtain an inequality of the form (7). This implies the upper bound $|Q| \leq N_1 N_2$, which makes implicitly a worst-case assumption on the correlation between $R.Y$ and $S.Y$. To see this, denote the degree sequences $\mathsf{deg}_R(X|Y) = (d_1, \ldots, d_n)$, $\mathsf{deg}_S(Z|Y) = (e_1, \ldots, e_n)$. The upper bound is the Cauchy-Schwartz inequality: $|Q| = \sum_i d_i e_i \leq \sqrt{\sum_i d_i^2} \cdot \sqrt{\sum_i e_i^2} = N_1 N_2$.

[2]Proof of validity: $2h(Y) + 2h(X|Y) + 2h(Z|Y) = 2h(XY) + 2h(Z|Y) \geq 2h(XY) + 2h(Z|XY) = 2h(XYZ)$.

This is an equality only when the degree vectors are proportional, otherwise the upper bound is a large over-estimate.

Computing the Upper Bound. Since Theorem 2.3 gives a guaranteed upper bound on the query output size, we would like to find the best choice of the coefficients $w_1, \ldots, w_s$ that lead to the smallest bound for the query Q. The smallest upper bound is obtained by solving a linear program, constructed as follows. Let $X_1, \ldots, X_n$ be all variables occurring in Q and $\boldsymbol{Z}$ the output variables. Our linear program has 2^n unknowns, one for each entropic value $h(\boldsymbol{V})$, for $\boldsymbol{V} \subseteq \{X_1, \ldots, X_n\}$. The linear program is the following:

$$\text{maximize } h(\boldsymbol{Z}) \text{ where:} \tag{8}$$
$$h(\boldsymbol{X}_\ell) + p_\ell h(\boldsymbol{Y}_\ell | \boldsymbol{X}_\ell) \le \log N_\ell \text{ for } \ell \in [s] \tag{9}$$
$$h(\boldsymbol{UV}) \ge h(\boldsymbol{V}) \text{ for all sets } \boldsymbol{U}, \boldsymbol{V} \subseteq \{X_1, \ldots, X_n\} \tag{10}$$
$$h(\boldsymbol{UV}) + h(\boldsymbol{UW}) \ge h(\boldsymbol{U}) + h(\boldsymbol{UVW}) \text{ for all sets } \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W} \subseteq \{X_1, \ldots, X_n\} \tag{11}$$

The program has one constraint (9) for each statistics collected on the database, and one inequality (10), (11) for every basic Shannon inequality over the query variables $X_1, \ldots, X_n$. It was shown that the optimal value of this linear program is equal to the best (smallest) upper bound that can be proven using an inequality of the form (7).

The linear program has 2^n unknowns, where n is the number of variables in the query (3). It was shown recently that there are equivalent linear programs with linear (for full Berge acyclic queries) and quadratic (for arbirary conjunctive queries) number of unknowns [25, 54]. In Sec. 6, we refine these equivalent programs to the new information inequality introduced in Sec. 3.

3 Information Inequality capturing Inter- and Intra-relation Correlations of Join Columns

Different degree vectors can be correlated with one another in different ways, and taking these correlations into consideration could potentially allow us to improve the accuracy of cardinality estimation. Correlation can happen among degree vectors of different relations that join over the same variable, in which case we call it *inter-relation correlation*. For example, given two atoms $R(X, Y)$ and $S(X, Z)$, we can consider the correlation between the degree vectors $\deg_R(Y|X)$ and $\deg_S(Z|X)$. Correlation can also happen among degree vectors in the same relation, in which case we call it *intra-relation correlation*. For example, we can consider the correlation among the two degree vectors $\deg_R(Y|X)$ and $\deg_R(X|Y)$ within the same relation $R(X, Y)$. In addition, correlation can happen in complicated ways that mix the inter- and intra-relation ones. We introduce below a new general information inequality that translates correlations among different degree vectors (over potentially different relations and different join variables) into the language of information theory. This inequality allows us to capture both inter- and intra-relation correlations (and more) in the information-theoretic framework for cardinality estimation, thus allowing us to derive tighter upper bounds on the cardinality of a join-project query. We present our general inequality below.

Lemma 3.1. *Consider a full query Q. Let* $\text{atoms}(Q)$ *denote the set of atoms in Q, and* $\text{attrs}(Q)$ *denote the set of all variables that appear in Q, where the variables take values from a domain Dom. Let k be a natural number, and for each $i \in [k]$, let $R_i(\boldsymbol{Z}_i)$ be an atom in* $\text{atoms}(Q)$ *with variables $\boldsymbol{Z}_i$ (where the atoms $R_i(\boldsymbol{Z}_i)$ are not necessarily distinct for different i values), $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ be two subsets of $\boldsymbol{Z}_i$, and p_i be a positive real number. Moreover, let $\boldsymbol{X}$ be any superset of $\boldsymbol{X}_1 \cup \cdots \cup \boldsymbol{X}_k$ and $p \stackrel{def}{=} \max(p_1, \ldots, p_k)$. Given any probability distribution over the output of Q, let $h : 2^{\text{attrs}(Q)} \to \mathbb{R}_+$ be the corresponding entropy function. Then, the following inequality holds:*

$$\log\left(\sum_{\boldsymbol{x}\in Dom^{\boldsymbol{X}}}\left(\deg_{R_1}(\boldsymbol{Y}_1|\boldsymbol{X}_1=\pi_{\boldsymbol{X}_1}(\boldsymbol{x}))\right)^{p_1}\cdot\ldots\cdot\left(\deg_{R_k}(\boldsymbol{Y}_k|\boldsymbol{X}_k=\pi_{\boldsymbol{X}_k}(\boldsymbol{x}))\right)^{p_k}\right)^{1/p} \geq \frac{h(\boldsymbol{X})+p_1h(\boldsymbol{Y}_1|\boldsymbol{X}_1)+\cdots+p_kh(\boldsymbol{Y}_k|\boldsymbol{X}_k)}{p} \tag{12}$$

Inter-relation correlation as a special case. Suppose we have k relations $R_1(\boldsymbol{Z}_1),\ldots,R_k(\boldsymbol{Z}_k)$ that join on a common variable X. In this case, it is natural to consider the correlation among the degree vectors $\deg_{R_i}(*|X) \stackrel{\text{def}}{=} \deg_{R_i}(\boldsymbol{Z}_i|X)$ for $i\in[k]$. This corresponds to the special case of Eq. (12) where we set $\boldsymbol{X}_i=\{X\}$ and $\boldsymbol{Y}_i=\boldsymbol{Z}_i$ for $i\in[k]$, leading to the following inequality:

$$\log\left(\sum_{x\in\mathrm{Dom}^{X}}\left(\deg_{R_1}(\boldsymbol{Z}_1|X=x)\right)^{p_1}\cdot\ldots\cdot\left(\deg_{R_k}(\boldsymbol{Z}_k|X=x)\right)^{p_k}\right)^{1/p} \geq \frac{h(X)+p_1h(\boldsymbol{Z}_1|X)+\cdots+p_kh(\boldsymbol{Z}_k|X)}{p} \tag{13}$$

Intra-relation correlation as a special case. Given a relation $R(\boldsymbol{Z})$, let $\{X_1,\ldots,X_k\}\subseteq\boldsymbol{X}\subseteq\boldsymbol{Z}$. To capture the correlation among the degree vectors $\deg_R(*|X_1),\ldots,\deg_R(*|X_k)$, we can use the special case of Eq. (12) where we set $R_i=R$, $\boldsymbol{X}_i=\{X_i\}$ and $\boldsymbol{Y}_i=\boldsymbol{Z}$ for $i\in[k]$, leading to the following inequality:

$$\log\left(\sum_{\boldsymbol{x}\in\mathrm{Dom}^{\boldsymbol{X}}}\left(\deg_R(\boldsymbol{Z}|X_1=x_1)\right)^{p_1}\cdot\ldots\cdot\left(\deg_R(\boldsymbol{Z}|X_k=x_k)\right)^{p_k}\right)^{1/p} \geq \frac{h(\boldsymbol{X})+p_1h(\boldsymbol{Z}|X_1)+\cdots+p_kh(\boldsymbol{Z}|X_k)}{p} \tag{14}$$

ℓ_p-norm bound as a special case. The *ℓ_p-norm bound* [1, 54] is a special case of both Eq. (13) and (14), and by extension Eq. (12). In particular, it is the special case where we only have a single degree vector, leading to Inequality (6), which translates the ℓ_p-norm of the degree vector into an upper bound on an entropic expression.

In order to incorporate Inequality (12) (including all its special cases) into the information-theoretic framework for cardinality estimation, we can use the same linear program from Eq. (8) where we replace the constraints from Eq. (9) with constraints of the form of Eq. (12) whenever the quantity on the LHS of Eq. (12) is an available statistic. In its most general form, this quantity is not necessarily computable in linear time in the size of the input relations because it involves an arbitrary join over a set of variables $\boldsymbol{X}$. However, what the above special cases have in common is that they make the LHS computable in linear time, thus making them suitable for cardinality estimation. In particular, the LHS of Eq. (13) is a sum over the values of a *single variable* X, whereas the LHS of Eq. (14) is a sum over the projection of a *single relation* $R(\boldsymbol{Z})$. Both statistics can be maintained efficiently.

The following example shows that both inter- and intra-relation correlations can be used to derive bounds that are strictly tighter than the ℓ_p-norm bound [1, 54]. Moreover, when used in isolation, the two types of correlation can result in bounds that are incomparable to one another. Future examples will show how they can complement one another when used together.

Example 3.2. Consider the triangle query $Q_\triangle$:

$$Q_\triangle(X,Y,Z)=R(X,Y)\wedge S(Y,Z)\wedge T(Z,X) \tag{15}$$

There are 6 degree vectors, two per relation, e.g., R has $\deg_R(X|Y)$ and $\deg_R(Y|X)$. Depending on available statistics, the ℓ_p-norm bound from Eq. (8) can give the following upper bound:

$$|Q_\triangle| \leq \Big(|| \deg_R(Y|X)||_2 \cdot || \deg_R(X|Y)||_2 \cdot || \deg_S(Z|Y)||_2 \cdot \\ || \deg_S(Y|Z)||_2 \cdot || \deg_T(X|Z)||_2 \cdot || \deg_T(Z|X)||_2 \Big)^{1/3} \tag{16}$$

In contrast, the inter-relation correlation can give the following better upper bound:

$$|Q_\triangle| \leq \Big(\big\langle \deg_R(Y|X), \deg_T(Z|X) \big\rangle \cdot \big\langle \deg_R(X|Y), \deg_S(Z|Y) \big\rangle \cdot \big\langle \deg_S(Y|Z), \deg_T(X|Z) \big\rangle \Big)^{1/3} \tag{17}$$

The bound from Eq. (17) is never worse than the bound from Eq. (16) and is strictly better in most cases. This is because by Cauchy-Schwarz, we have the inequality (and two similar others):

$$\big\langle \deg_R(Y|X), \deg_T(Z|X) \big\rangle \leq || \deg_R(Y|X)||_2 \cdot || \deg_T(Z|X)||_2$$

Moreover, the above inequality only becomes an equality in the very special case where the two degree vectors $\deg_R(Y|X)$ and $\deg_T(Z|X)$ are perfectly correlated, which is rarely the case.

In contrast, the intra-relation correlation gives the upper bound:

$$|Q_\triangle| \leq \Big(\Big(\sum_{(x,y)\in R} \sqrt{\deg_R(Y|X=x) \cdot \deg_R(X|Y=y)} \Big) \cdot \Big(\sum_{(y,z)\in S} \sqrt{\deg_S(Z|Y=y) \cdot \deg_S(Y|Z=z)} \Big) \cdot \\ \Big(\sum_{(z,x)\in T} \sqrt{\deg_T(X|Z=z) \cdot \deg_T(Z|X=x)} \Big) \Big)^{1/3} \tag{18}$$

The bound from Eq. (18) is also never worse than the bound from Eq. (16) and is strictly better in most cases. In particular, less obviously, the following inequality is also a consequence of Cauchy-Schwarz (alongside two other similar inequalities):

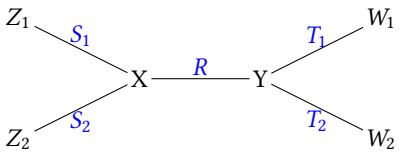
$$\sum_{(x,y)\in R} \sqrt{\deg_R(Y|X=x) \cdot \deg_R(X|Y=y)} \quad \leq \quad || \deg_R(Y|X)||_2 \cdot || \deg_R(X|Y)||_2 \tag{19}$$

However, the bounds from Eq. (17) and Eq. (18) are incomparable to one another.

The following example illustrates how the inter- and intra-relation correlations can complement one another to derive highly non-trivial cardinality upper bounds.

Example 3.3. Consider the following full query:

$$Q(X, Y, Z_1, Z_2, W_1, W_2) = R(X, Y) \wedge S_1(X, Z_1) \wedge S_2(X, Z_2) \wedge T_1(Y, W_1) \wedge T_2(Y, W_2) \tag{20}$$



This query has the structure of a STATS query that we use in the experiments in Sec. 7. The inter-relation correlation among the degree vectors $\deg_{S_1}(Z_1|X)$ and $\deg_{S_2}(Z_2|X)$ can give us a good idea about the join size of $S_1 \bowtie S_2$. In particular, the size of $S_1 \bowtie S_2$ is exactly the inner product of these two degree vectors. Similarly, the size of the join $T_1 \bowtie T_2$ is exactly the inner product of the degree vectors $\deg_{T_1}(W_1|Y)$ and $\deg_{T_2}(W_2|Y)$. But the question is: How can we stitch together

these two inter-relation correlations to get an upper bound on the size of the entire query Q? An obvious upper bound is to multiply the two:

$$|Q| \leq \left\langle \deg_{S_1}(Z_1|X), \deg_{S_2}(Z_2|X) \right\rangle \cdot \left\langle \deg_{T_1}(W_1|Y), \deg_{T_2}(W_2|Y) \right\rangle$$

But this is a very loose upper bound because it assumes that X and Y are completely independent, which is unlikely considering that they occur in the same relation R. To tighten this bound, we need to consider the intra-relation correlation of X and Y inside R. One bound that can be derived this way is as follows:

$$|Q| \leq \left[\left(\sum_{x \in \text{Dom}^X} (\deg_{S_1}(Z_1|X=x) \cdot \deg_{S_2}(Z_2|X=x))^3 \right) \cdot \left(\sum_{y \in \text{Dom}^Y} (\deg_{T_1}(W_1|Y=y) \cdot \deg_{T_2}(W_2|Y=y))^3 \right) \cdot \left(\sum_{(x,y) \in R} \deg_R(Y|X=x) \cdot \deg_R(X|Y=y) \right) \right]^{1/3} \tag{21}$$

The above bound is proved using the following Shannon inequality:

$$3h(XYZ_1Z_2W_1W_2) \leq [h(X) + 3h(Z_1|X) + 3h(Z_2|X)] + [h(Y) + 3h(W_1|Y) + 3h(W_2|Y)] + [h(XY) + h(Y|X) + h(X|Y)] \tag{22}$$

4 Approximating Inner Products using Sketches

In Sec.3, we introduced a new information inequality that leverages the generalized inner products of degree vectors to capture inter- and intra-relation correlations. The exact computation of these inner products can be computationally expensive and memory-intensive for large datasets, especially when needing to compute them for relations refined by predicates using MCVs and histograms.

This section introduces an extension of CorrBound that uses sketches to efficiently and accurately approximate these inner products. It uses the Fast-AMS sketch [10], which improves on the original AMS sketch [4, 5] by using a hashing technique previously introduced for Count sketch [8]. This sketching approach is also used by a state-of-the-art cardinality estimator for join queries [21]. In the following, we first recall the application of the AMS and Fast-AMS sketches to inner products and then discuss how they are incorporated in CorrBound.

Basic AMS sketch. We first introduce sketching for the inner product of two degree vectors $\langle \boldsymbol{d}_1, \boldsymbol{d}_2 \rangle = \sum_{i=1}^{n} \boldsymbol{d}_1[i] \cdot \boldsymbol{d}_2[i]$. The core idea is to compress large degree vectors into fixed-size summaries from which we can estimate the inner product of the original degree vectors. The basic AMS sketch summarizes each degree vector $\boldsymbol{d}_i$ to a single counter C_i using a *sign hash function* $\sigma : [n] \to \{-1, 1\}$, which is 4-wise independent: $C_j = \sum_{i=1}^{n} \boldsymbol{d}_j[i] \cdot \sigma(i)$ for $j \in [2]$. The estimate of the inner product is then given by the product, $X = C_1 \cdot C_2$. This estimator is unbiased, i.e., its expectation is equal to the inner product: $\mathbb{E}[X] = \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{d}_1[i] \cdot \boldsymbol{d}_2[j] \cdot \mathbb{E}[\sigma(i) \cdot \sigma(j)]$. Due to the pairwise independence of σ, we have $\mathbb{E}[\sigma(i) \cdot \sigma(j)] = 1$ for $i = j$ and $\mathbb{E}[\sigma(i) \cdot \sigma(j)] = 0$ otherwise. Thus, the expectation simplifies to the true inner product. While unbiased, this single-counter sketch has a high variance [5]: $\text{Var}(X) = \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2 \leq 2||\boldsymbol{d}_1||_2^2 \cdot ||\boldsymbol{d}_2||_2^2$. The variance can be large, especially when the degree vectors $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ have large values, which can lead to significant errors in the estimate of the inner product.

Fast-AMS sketch. The Fast-AMS sketch brings two improvements to the basic AMS sketch and provides more accurate estimates with higher confidence [10].

First, instead of using one counter, the sketch distributes the computation across a vector of b counters ("buckets") using a hash function $h : [n] \to [b]$ that maps each index to one of the

counters [8, 10]. The sketches for $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are now vectors of size b: $\boldsymbol{C}_j[k] = \sum_{i \in [n] \text{ s.t. } h(i)=k} \boldsymbol{d}_j[i] \cdot \sigma(i)$, for $k \in [b], j \in [2]$. The estimate is the inner product of the two sketch vectors: $X = \langle \boldsymbol{C}_1, \boldsymbol{C}_2 \rangle = \sum_{k=1}^{b} \boldsymbol{C}_1[k] \cdot \boldsymbol{C}_2[k]$. While the estimate remains unbiased, the variance is reduced by a factor of b. This also has the practical advantage of improving the update time, as only one counter needs to be updated for each update, rather than updating all counters as in the basic AMS sketch.

The second improvement is the use of the "median trick" to boost the probability of getting a good estimate [8]. We create m independent instances of the b-bucket sketch, where each instance $j \in [m]$ uses its own independent hash functions, σ_j and h_j. This process yields m separate estimates, $X_1, \ldots, X_m$. The final estimate is the median of these values. By taking the median, the probability of obtaining a poor estimate decreases exponentially as the number m of instances increases. The combination of b counters (for accuracy) and m independent instances (for success probability) constitutes the Fast-AMS sketch, which can be visualized as a matrix with b columns and m rows.

Sketching for generalized inner products. The aforementioned methodology can be extended to estimate the generalized inner products of arbitrarily many degree vectors. We next explain this for three degree vectors [4]: $\langle \boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3 \rangle = \sum_{i=1}^{n} d_1[i] \cdot d_2[i] \cdot d_3[i]$. We use two independent sign hash functions, σ_1 and σ_2, and construct three sketch counters: $C_j = \sum_{i=1}^{n} \boldsymbol{d}_j[i] \cdot \sigma_j(i)$ for $j \in [2]$, and $C_3 = \sum_{i=1}^{n} \boldsymbol{d}_3[i] \cdot \sigma_1(i) \cdot \sigma_2(i)$. The estimate for the 3-way inner product is then given by $X = C_1 \cdot C_2 \cdot C_3$. The expectation of this estimate is the true inner product. The variance of the estimate X for the inner product of k degree vectors $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_k$ can be upper bounded [5, 10] as $\text{Var}(X) \leq 3^{k-1} \prod_{j=1}^{k} ||\boldsymbol{d}_j||_2^2$. This bound grows exponentially with k. Larger sketches (i.e., larger b and m) are needed to maintain the same level of accuracy as k increases.

Fast-AMS sketch in CorrBound. CorrBound uses Fast-AMS sketch to approximate the inner products of 2 and 3 degree vectors, so as to support subqueries that are 2-way and 3-way joins on the same variable. The Fast-AMS sketch is implemented as $m \times b$ matrices of sketch counters, where m is the number of independent instances and b is the number of counters per instance. It uses two families of m independent, 4-wise independent sign hash functions, $\Sigma^{(1)} = \{\sigma_1^{(1)}, \ldots, \sigma_m^{(1)}\}$ and $\Sigma^{(2)} = \{\sigma_1^{(2)}, \ldots, \sigma_m^{(2)}\}$. We also use a third family by taking the element-wise product of the first two families, $\Sigma^{(1,2)} = \{\sigma_1^{(1)}\sigma_1^{(2)}, \ldots, \sigma_m^{(1)}\sigma_m^{(2)}\}$. It also uses one family of m independent, pairwise independent hash functions, $H = \{h_1, \ldots, h_m\}$, where $h_i : [n] \to [b]$.

For each degree vector $\boldsymbol{d}$, we construct and store three sketch matrices in a precomputation phase. These matrices share the same bucket hash functions H but differ in the sign hash functions used: The sketch matrices $S^{(1)}(\boldsymbol{d})$, $S^{(2)}(\boldsymbol{d})$, and $S^{(1,2)}(\boldsymbol{d})$, are constructed using the sign hash functions $\Sigma^{(1)}$, $\Sigma^{(2)}$, and respectively $\Sigma^{(1,2)}$, and the same bucket hash functions H. This setup allows us to estimate different types of inner products at estimation time.

We estimate the inner product of two degree vectors $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ using sketches built with the same sign hash functions $S^{(1)}(\boldsymbol{d}_1)$ and $S^{(1)}(\boldsymbol{d}_2)$. For the inner product of three degree vectors $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, and $\boldsymbol{d}_3$, we use sketches from different families $S^{(1)}(\boldsymbol{d}_1)$, $S^{(2)}(\boldsymbol{d}_2)$, and $S^{(1,2)}(\boldsymbol{d}_3)$. The consistent use of the hash functions H across all sketches ensures that the values are correctly aligned.

Handling Predicates. Since selection predicates are not known in advance, we cannot pre-filter the relations and compute sketches only for the relevant data. Instead, we pre-compute the sketches for the degree vectors of the different MCVs (and histogram buckets). At estimation time, we identify the relevant MCVs for the selection predicates in the query and use the sketch matrices corresponding to the degree vectors in the MCVs to approximate the inner products. If a relation

has multiple predicates, we use the sketch matrices for the predicate with the smallest estimated selectivity (ℓ_1-norm).

Space Optimizations. We employ two optimizations to reduce the memory footprint, reminiscent of prior work [39, 48]. (1) The sketch matrix for one MCV/bucket is often sparse, with most entries being zero. We use sparse data structures to store only the non-zero counters. (2) The majority of non-zero counters have small integer values. If all counters in a matrix are under 256, we use an 8-bit integer type to represent them.

5 Approximating the Cardinality Tensor

The CorrBound's information inequality uses as data statistics a cardinality tensor of generalized inner products of degree vectors for each join variable in the query. Eq. (2) defines this tensor for k-dimensions: The point $\boldsymbol{T}[\ell_1, \ldots, \ell_k]$ represents (an upper bound on) the cardinality of the k-way join where we consider each relation R_j restricted to the fragment defined by its ℓ_j-th Most Common Value or histogram bucket for $j \in [k]$. Equivalently, the tensor defines (upper bounds on) the cardinalities of all possible queries with the same k-way join but arbitrary (equality or range) predicates.[3] In this section, we will refer to this class of queries as $\boldsymbol{Q}$.

This tensor can be extremely large and cannot be materialized. In this section, we explain how we can compute efficiently an accurate approximation of it in time linear in: the number k of dimensions, the number n of elements in a degree vector, and the numbers $s_1, \ldots, s_k$ of degree vectors for the relations $R_1, \ldots, R_k$.

Reformulating the tensor. Our first insight is that the tensor in Eq. (2) can be reformulated as a sum of rank-1 tensors. We exemplify this in case $\boldsymbol{T}$ is a matrix ($k = 2$):

$$\boldsymbol{T} = \begin{bmatrix} \left\langle \boldsymbol{d}_1^{(1)}, \boldsymbol{d}_1^{(2)} \right\rangle & \ldots & \left\langle \boldsymbol{d}_1^{(1)}, \boldsymbol{d}_{s_2}^{(2)} \right\rangle \\ \vdots & \vdots & \vdots \\ \left\langle \boldsymbol{d}_{s_1}^{(1)}, \boldsymbol{d}_1^{(2)} \right\rangle & \ldots & \left\langle \boldsymbol{d}_{s_1}^{(1)}, \boldsymbol{d}_{s_2}^{(2)} \right\rangle \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} \boldsymbol{d}_1^{(1)}[i] \\ \vdots \\ \boldsymbol{d}_{s_1}^{(1)}[i] \end{bmatrix} \otimes \begin{bmatrix} \boldsymbol{d}_1^{(2)}[i] \\ \vdots \\ \boldsymbol{d}_{s_2}^{(2)}[i] \end{bmatrix} \tag{23}$$

That is, the matrix $\boldsymbol{T}$ is a sum of n outer products of vectors. The i-th outer product i corresponds to the i-th value in the active domain of the join variable (in some predefined order), its first (second) vector consists of the degrees of the i-th join value in each of the s_1 (s_2) degree vectors for relation R_1 (R_2). This reformulation immediately generalizes to arbitrary k: The k-dimensional tensor $\boldsymbol{T}$ is the sum of n outer products of k vectors, where the j-th vector in the i-th outer product consists of the degrees of the i-th join value in the s_j degree vectors for relation R_j:

$$\boldsymbol{T} = \sum_{i=1}^{n} \begin{bmatrix} \boldsymbol{d}_1^{(1)}[i] \\ \vdots \\ \boldsymbol{d}_{s_1}^{(1)}[i] \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \boldsymbol{d}_1^{(k)}[i] \\ \vdots \\ \boldsymbol{d}_{s_k}^{(k)}[i] \end{bmatrix} \tag{24}$$

This reformulation is the CP decomposition of the tensor [31]. In our setting, it can be achieved *without the need to first materialize the tensor and then decompose it.* This is possible due to (1) the multi-way join structure of queries in $\boldsymbol{Q}$, where the degree vectors for a relation are stored separately from those for the other relations, and to (2) the symbolic, non-materialized form of the generalized inner products. It is easy to see that the reformulation can be achieved in time linear in k, n, and $s_1, \ldots, s_k$.

[3] Given the s_j MCVs and histogram buckets per relation R_j, CorrBound can only distinguish up to $\prod_{j=1}^{k} s_j$ such queries, since many predicates may be satisfied by the same MCV or histogram bucket and therefore yield the same cardinality.

The reformulation has a rank-1 tensor per join value. To compute $\boldsymbol{T}[\ell_1, \ldots, \ell_k]$, we would therefore need to loop over all join values (or equivalently, over all rank-1 tensors) and for each of them take the ℓ_j-th value from the j-th vector in each of the outer products:

$$\boldsymbol{T}[\ell_1, \ldots, \ell_k] = \sum_{i=1}^{n} \boldsymbol{d}_{\ell_1}^{(1)}[i] \cdot \ldots \cdot \boldsymbol{d}_{\ell_k}^{(k)}[i]$$

This computation remains expensive, even if we process the rank-1 tensors in parallel. We next consider the alternative of selecting the top-m most relevant rank-1 tensors, for $m \ll n$, and then approximating the remaining $n - m$ rank-1 tensors.

Keeping the most relevant m rank-1 tensors in the reformulation. To understand the relevance of each of the rank-1 tensors in the cardinality tensor, we recall that each of them contributes to the cardinality of each of the queries in $\boldsymbol{Q}$. Therefore, a natural measure of relevance of the i-th rank-1 tensor is its ℓ_1-norm ω_i, which sums up its contributions to the cardinalities of all such queries:

$$\omega_i = \left\| \begin{bmatrix} \boldsymbol{d}_1^{(1)}[i] \\ \vdots \\ \boldsymbol{d}_{s_1}^{(1)}[i] \end{bmatrix} \otimes \ldots \otimes \begin{bmatrix} \boldsymbol{d}_1^{(k)}[i] \\ \vdots \\ \boldsymbol{d}_{s_k}^{(k)}[i] \end{bmatrix} \right\|_1 = \left\| \begin{bmatrix} \boldsymbol{d}_1^{(1)}[i] \\ \vdots \\ \boldsymbol{d}_{s_1}^{(1)}[i] \end{bmatrix} \right\|_1 \cdot \ldots \cdot \left\| \begin{bmatrix} \boldsymbol{d}_1^{(k)}[i] \\ \vdots \\ \boldsymbol{d}_{s_k}^{(k)}[i] \end{bmatrix} \right\|_1 = \left(\sum_{p_1=1}^{s_1} \boldsymbol{d}_{p_1}^{(1)}[i] \right) \cdot \ldots \cdot \left(\sum_{p_k=1}^{s_k} \boldsymbol{d}_{p_k}^{(k)}[i] \right)$$

The ℓ_1-norm ω_i can be computed in time $O(\sum_{j=1}^{k} s_j)$, and so the computation of all n ℓ_1-norms takes time $O(n \cdot \sum_{j=1}^{k} s_j)$.

To reduce the estimation time, we can approximate the cardinality $\boldsymbol{T}[\ell_1, \ldots, \ell_k]$ by keeping the $m \ll n$ rank-1 tensors with the largest ℓ_1-norms. This can lead to an under-approximation of the cardinality for each of the queries in $\boldsymbol{Q}$. To find an appropriate value for m, we can plot the approximation error, the average time to compute the approximate cardinality for queries in $\boldsymbol{Q}$, and the extra space needed to store the m rank-1 tensors as a function of the tensor rank m, and then select m so that the time and space are within the desired budget. This approach has a shortcoming: It under-approximates the cardinalities. In contrast, CorrBound's information-theoretic framework introduced in Sec. 3 over-approximates the cardinalities. By combining the two components, CorrBound cannot guarantee anymore an upper bound on the cardinality of the query output.[4] We show next how to extend this approach to yield an over-approximation and therefore to guarantee an upper bound.

Upper-bounding the discarded rank-1 tensors. Given the tensor $\boldsymbol{T}$ expressed as a sum of n rank-1 tensors as in Eq. (24), we want to find a tensor $\hat{\boldsymbol{T}}$ expressed as a sum of $m + 1$ rank-1 tensors such that $\hat{\boldsymbol{T}}[\ell_1, \ldots, \ell_k] \geq \boldsymbol{T}[\ell_1, \ldots, \ell_k], \forall (\ell_1, \ldots, \ell_k) \in ([s_1] \times \cdots \times [s_k])$. We define $\hat{\boldsymbol{T}}$ as the sum of the most relevant m rank-1 tensors and one extra rank-1 tensor, which upper bounds the sum of the discarded $n - m$ rank-1 tensors. We next explain how to define this extra rank-1 tensor. Consider again the 2-dimensional tensor $\boldsymbol{T}$ in Eq. (23). To upper bound the contribution of the discarded rank-1 tensors (indices $m + 1$ to n), we apply the Cauchy-Schwarz inequality: Given two vectors $\boldsymbol{d}_1, \boldsymbol{d}_2 \in \mathbb{N}^{n-m}$, it holds that $\langle \boldsymbol{u}, \boldsymbol{v} \rangle \leq ||\boldsymbol{u}||_2 \cdot ||\boldsymbol{v}||_2 = \left(\sum_{j=1}^{n-m} (u[j])^2 \right)^{1/2} \cdot \left(\sum_{j=1}^{n-m} (v[j])^2 \right)^{1/2}$. We apply this inequality for pairs of degree vectors $\boldsymbol{d}_{\ell_1}^{(1)}$ and $\boldsymbol{d}_{\ell_2}^{(2)}$, restricted to their values used in the discarded rank-1 tensors:

$$\sum_{i=m+1}^{n} \boldsymbol{d}_{\ell_1}^{(1)}[i] \cdot \boldsymbol{d}_{\ell_2}^{(2)}[i] \leq \left(\sum_{i=m+1}^{n} \left(\boldsymbol{d}_{\ell_1}^{(1)}[i] \right)^2 \right)^{1/2} \cdot \left(\sum_{i=m+1}^{n} \left(\boldsymbol{d}_{\ell_2}^{(2)}[i] \right)^2 \right)^{1/2}$$

[4] We verified experimentally in Sec. 7 that the under-approximation due to tensor truncation can offset some of the over-approximation due to the information theoretic framework and therefore may provide an estimate closer to the true cardinality than the latter over-approximation.

The upper bound for the discarded tail of the tensor is then:

$$\sum_{i=m+1}^{n} \begin{bmatrix} \boldsymbol{d}_1^{(1)}[i] \\ \vdots \\ \boldsymbol{d}_{s_1}^{(1)}[i] \end{bmatrix} \otimes \begin{bmatrix} \boldsymbol{d}_1^{(2)}[i] \\ \vdots \\ \boldsymbol{d}_{s_2}^{(2)}[i] \end{bmatrix} \leq \boldsymbol{t}^{(1)} \otimes \boldsymbol{t}^{(2)}, \text{ where } \boldsymbol{t}^{(j)} = \begin{bmatrix} \left(\sum_{i=m+1}^{n} \left(\boldsymbol{d}_1^{(j)}[i]\right)^2\right)^{1/2} \\ \vdots \\ \left(\sum_{i=m+1}^{n} \left(\boldsymbol{d}_{s_j}^{(j)}[i]\right)^2\right)^{1/2} \end{bmatrix} \text{ for } j \in [2]. \tag{25}$$

The above inequality generalizes from 2 to k dimensions thanks to Hölder's inequality [38], where the ℓ_2-norm is replaced by the ℓ_k-norm.

An alternative lossless decomposition. In case of 2-way joins, we use an alternative, more effective approach based on low-rank factorization using singular value decomposition (SVD). Even in case of a k-way join, CorrBound also considers the possible 2-way joins with the remaining $k-2$ relations excluded from the inter-relation correlation data statistics and subject to the ℓ_p-norm data statistics of LpBound. We next explain our approach for $k = 2$.

Consider the matrix $\boldsymbol{M} \in \mathbb{N}^{n \times s}$, which collects all $s = \sum_{j=1}^{k} s_j$ degree vectors from relations $R_1, \ldots, R_k$. This matrix $\boldsymbol{M}$ is a 2-dimensional flattening of the k-dimensional cardinality tensor $\boldsymbol{T}$. For cardinality estimation, we need the inner products of the degree vectors, which are the values in the matrix $\boldsymbol{M}^\top \boldsymbol{M}$. In our setting, $s \ll n$: there are much less MCVs and histogram buckets for all six relations joining on movie_id ($s \approx 10$K) than the number of movie identifiers ($n \approx 2.5$M). The size of the matrix $\boldsymbol{M}^\top \boldsymbol{M}$ remains very large ($s^2 \approx 100$M) and it is prohibitive to precompute and store it. Instead, we compress $\boldsymbol{M}$ into another matrix $\hat{\boldsymbol{M}} \in \mathbb{N}^{s \times s}$ such that $\hat{\boldsymbol{M}}^\top \hat{\boldsymbol{M}} = \boldsymbol{M}^\top \boldsymbol{M}$. In other words, we compress the degree vectors from n to s values while preserving their exact pairwise inner product. This compression is fully justified in our setting: The rank of $\boldsymbol{M}$ is $\min(n, s) = s$, so $n - s$ rows in $\boldsymbol{M}$ are linear combinations of other rows and do not add new information. To retrieve the cardinality of the 2-way join of two relations at estimation time, each restricted by an MCV/bucket, we then only need to look up the corresponding compressed degree vectors in $\hat{\boldsymbol{M}}$ and compute their inner product.

Let the SVD decomposition of our matrix: $\boldsymbol{M} = \boldsymbol{U}\Sigma_s \boldsymbol{V}_s^\top$, where $\boldsymbol{U} \in \mathbb{R}^{n \times s}$ and $\Sigma_s, \boldsymbol{V}_s \in \mathbb{R}^{s \times s}$. Then, $\boldsymbol{M}^\top \boldsymbol{M} = \boldsymbol{V}_s \Sigma_s^\top \boldsymbol{U}^\top \boldsymbol{U} \Sigma_s \boldsymbol{V}_s^\top = \boldsymbol{V}_s \Sigma_s^2 \boldsymbol{V}_s^\top$, since Σ_s is square diagonal and $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_s$. There is another matrix $\hat{\boldsymbol{M}}$ such that $\hat{\boldsymbol{M}}^\top \hat{\boldsymbol{M}} = \boldsymbol{V}_s \Sigma_s^2 \boldsymbol{V}_s^\top$: Namely, $\hat{\boldsymbol{M}} = \Sigma_s \boldsymbol{V}_s^\top$. Thus, $\hat{\boldsymbol{M}}^\top \hat{\boldsymbol{M}} = \boldsymbol{M}^\top \boldsymbol{M}$ and $\hat{\boldsymbol{M}} \in \mathbb{R}^{s \times s}$.

If s is too large, we can settle for a lossy compression. Say we want to compute a low-rank decomposition of $\boldsymbol{M}$ of rank $s' < \min(s, n)$. Using SVD, we can obtain the best rank-s' approximation $\hat{\boldsymbol{M}}$ of $\boldsymbol{M}$ among all possible rank-s' matrices, by the Eckart-Young-Mirsky's theorem [15]; here, best means the matrix $\hat{\boldsymbol{M}}$ that is closest to $\boldsymbol{M}$ under Frobenius norm.

6 Computing the Cardinality Estimates

The linear program from Eq. (8) to (11) has an exponential number of unknowns and constraints in the size of the query Q, and this continues to hold for our enhanced linear program where we replace the constraints from Eq. (9) with more general constraints of Eq. (12). Prior work [25, 54] describes how to reformulate the linear program into a compact one with a linear (or quadratic) number of unknowns and constraints in two cases that are most relevant in practice. The first case considers queries that are *full* (i.e., no projections) and *Berge-acyclic* (i.e., acyclic where any two atoms share at most one variable) where all degree vectors are *simple* and *full*, i.e., have the form $\deg_R(*|X)$ where X is *a single* variable and $*$ denotes *all* remaining variables in R. The second case considers all queries (acyclic or cyclic, with or without projections) where all degree vectors are *simple* (but not necessarily full). We describe below how to extend the first case to handle our general correlation constraints from Eq. (12) (the second case is omitted for lack of space).

The case of full Berge-acyclic queries. The original linear program from Eq. (8) uses one unknown $h(\boldsymbol{X})$ for every subset $\boldsymbol{X}$ of query variables. However in case of full Berge-acyclic queries with full and simple degree vectors, prior work [54] provide an equivalent but compact linear program, where we only use unknowns $h(\boldsymbol{X})$ where $\boldsymbol{X}$ either consists of a single query variable or all the query variables of a single relation $R(\boldsymbol{X})$. The objective is to maximize the sum of the joint entropies of the variables in each relation and the negative entropy of each occurrence of a join variable.

The entropic terms in our generalized constraints from Eq. (12) are of this form, and we can therefore use such compact linear programs for upper bounding the output size of full Berge acyclic queries. In particular, the RHS of Eq. (12) can be written as follows:

$$\frac{h(\boldsymbol{X}) + p_1 h(\boldsymbol{Y}_1\boldsymbol{X}_1) - p_1 h(\boldsymbol{X}_1) + \cdots + p_k h(\boldsymbol{Y}_k\boldsymbol{X}_k) - p_k h(\boldsymbol{X}_k)}{p}$$

If all degree vectors are full and simple, then for every term $h(\boldsymbol{X}_i)$, the set $\boldsymbol{X}_i$ consists of a single variable, and for every term $h(\boldsymbol{Y}_i\boldsymbol{X}_i)$, the set $\boldsymbol{Y}_i\boldsymbol{X}_i$ consists of all the variables of a relation $R(\boldsymbol{Y}_i\boldsymbol{X}_i)$. We still need to ensure that $\boldsymbol{X}$ in $h(\boldsymbol{X})$ either consists of a single variable or of all the variables of a relation. The former is true in the inter-relation correlation from Eq. (13), while the latter can be enforced in the intra-relation correlation from Eq. (14) by choosing $\boldsymbol{X} \stackrel{\text{def}}{=} \boldsymbol{Z}$.

Example 6.1. Consider the query Q from Example 3.3. The compact linear program has the following form, where all the unknowns appear in the objective function. Constraints (26), (27), and (28) capture the inter- and intra-relation correlations from Eq. (21):

$$\text{maximize } h(XY) + h(XZ_1) + h(XZ_2) + h(YW_1) + h(YW_2) - 2h(X) - 2h(Y)$$

where:

$$3h(XZ_1) + 3h(XZ_2) - 5h(X) \le \log \sum_{x \in \text{Dom}^X} (\deg_{S_1}(Z_1|X=x) \cdot \deg_{S_2}(Z_2|X=x))^3 \tag{26}$$

$$3h(YW_1) + 3h(YW_2) - 5h(Y) \le \log \sum_{y \in \text{Dom}^Y} (\deg_{T_1}(W_1|Y=y) \cdot \deg_{T_2}(W_2|Y=y))^3 \tag{27}$$

$$3h(XY) - h(X) - h(Y) \le \log \sum_{(x,y) \in R} \deg_R(Y|X=x) \cdot \deg_R(X|Y=y) \tag{28}$$

$\cdots$ (other data statistics)

$\max(h(X), h(Y)) \le h(XY)$ (monotonicity for $R(X,Y)$)

$h(XY) \le h(X) + h(Y)$ (subadditivity for $R(X,Y)$)

$\cdots$ (similarly for other relations)

7 Experimental Evaluation

In this section, we experimentally show that by observing the correlations across the join columns we can improve the estimation accuracy at a modest price in extra storage and time for the additional data statistics. Our main findings are as follows.

1. CorrBound-d, the tensor-decomposition variant of CorrBound (Sec. 5), is our best pick: Its estimates are guaranteed cardinality upper bounds and have a small error range (in contrast to the sketch-based variant, Sec. 4) and takes small space and time (in contrast to the variant without compression of degree vectors).

2. CorrBound-d is at least as accurate as the state-of-the art pessimistic cardinality estimator LpBound [54] by design, as it extends the latter with correlation-aware inequalities. It can be up to 3.8 (2) orders of magnitude more accurate than LpBound for the JOBlight (STATS) acyclic queries. For the SM cyclic queries, which join up to 28 copies of the edge relation, CorrBound reduces the estimation error of LpBound by 6.7 – 16.3 times and has a median error of only 6.23. For the

SM queries, CorrBound can be up to 10 orders of magnitude more accurate than the traditional estimators from Postgres, DuckDB, and a commercial database system.

3. The time to estimate all connected subqueries of a query in the JOBlight and STATS benchmarks is on average below ten milliseconds for CorrBound-d. This is more than 65 (13) times faster than the sketch-based SOTA estimator UCISketch [21] and 6.1 (2.5) times slower than LpBound for JOBlight (STATS).

7.1 Experimental Setup

We used the LpBound experimental framework from prior work [54].

Estimators. We analyze the performance of three CorrBound variants. CorrBound-s uses sketching to compress the degree vectors (Sec. 4), CorrBound-d uses tensor decomposition to compress the degree vectors (Sec. 5), and CorrBound-e uses the exact degree vectors without any compression. Further, we compare with a variety of recent estimators: (1) the SOTA pessimistic cardinality estimators LpBound [54] and SafeBound [12]; (2) the SOTA sketch-based estimator UCISketch [21]; (3) the built-in traditional estimators from Postgres 13.14 and DuckDB 0.10.1 and a commercial system DbX; and (4) the probabilistic graph model-based estimators BayesCard [52], DeepDB [22], and FactorJoin [51]. CorrBound is implemented as an extension of LpBound. To ensure fair comparison between the two estimators, CorrBound uses the same configuration as LpBound (MCVs and histogram buckets, LP solver, code base to create the linear programs).

Benchmarks. JOBlight [33] has 70 full Berge acyclic queries, which are star equi-joins of 2-5 relations from the IMDB dataset (3.7GB) and have 1-4 equality predicates. STATS [36] has 146 full Berge acyclic queries, which are snowflake joins of 2-8 relations from the Stats Stack Exchange network dataset (38MB) and have 2-16 equality and range predicates. SM (Subgraph Matching) [44] has 400 full cyclic queries over the DBLP dataset (26.8MB edge relation and 3.5MB vertex relation). The SM queries use 11-28 copies of the edge relation and 2 vertex relation copies per edge relation copy, with one equality predicate per vertex relation copy.

Metrics. We report the estimation error: $\frac{\max(1,\text{Estimated Cardinality})}{\max(1,\text{True Cardinality})}$. It is greater (less) than 1 in case of over (under)-estimation.

System configuration: Intel Xeon Silver 4214, 48 cores/193GB memory, Debian GNU/Linux 10 (buster). Default configuration for data statistics for each estimator. Postgres configuration [33]: 4GB shared memory, 2GB work memory, 32GB implicit OS cache, and 6 max parallel workers. Indices are enabled on primary/foreign keys. CorrBound and LpBound use DuckDB to compute the statistics and HiGHS 1.7.2 [24] to solve the linear programs.

7.2 Estimation Errors for Acyclic Queries

Fig. 1 gives the estimation errors for the JOBlight and STATS benchmarks. CorrBound-d has up to 3.8 (2) orders of magnitude (OOM) better accuracy than the pessimistic estimators SafeBound and LpBound on JOBlight (STATS). The median error is however only up to one OOM smaller, which means that for some queries the errors of CorrBound-d are almost the same as for LpBound.

CorrBound-d's error can be up to 6.4/6.0/4.6 (2.9/5.3/1.7) times lower than for Postgres/DuckDB/DbX for STATS (JOBlight). At the other extreme, CorrBound-d's error can be up to 4.2/2.8/3.3 (2.2/0.5/2.2) times higher than for Postgres/DuckDB/DbX for STATS (JOBlight). The traditional estimators can return both under- and over-estimates and their error range is much wider than of CorrBound variants (even more so for STATS).

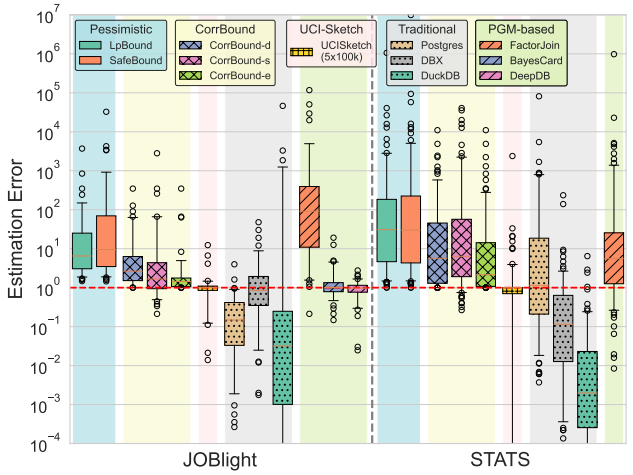


Fig. 1. Estimation errors for acyclic queries. CorrBound-d uses tensor rank $250k$ ($15k$) for JOBlight (STATS). CorrBound-s uses sketch matrices of dimension 5×4096 (5×512) for JOBlight (STATS). UCISketch uses a $5 \times 100k$ sketch matrix for both JOBlight and STATS. UCISketch gives 4/70 (27/146) negative estimates for JOBlight (STATS). These are adjusted to 1. BayesCard and DeepDB do not support STATS.

CorrBound-e has the best accuracy among the CorrBound variants (median error of only 1.5× on JOBLight). Yet it is not feasible for datasets with very large domain sizes due its excessive memory use (Table 2: 0.5GB for JOBlight and 155MB for STATS). We included it here to better understand the impact on accuracy brought by the compression of the degree vectors. CorrBound-s uses sketches. While its estimation time and space requirements are less than of CorrBound-d, it may occasionally yield underestimates and its error range is larger.

UCISketch produces accurate estimates by building sketches *online* for each query by filtering the tables according to the predicates and building sketches on the result. This avoids estimating the selectivity of filter predicates, improving the accuracy. However, this increases estimation time significantly (Table 1), and it introduces engineering challenges by mixing query optimization and execution. To achieve the good accuracy reported in Fig. 1, UCISketch needs 25x larger sketch matrices than CorrBound-s. This is due to the k-way joins in the benchmark queries, for which the cardinality variance degrades exponentially with k. For smaller sketch matrices, we verified experimentally that the variance increases significantly (figure not included for lack of space).

The PGM-based estimators are designed to capture the correlations in the data, as they are trained over samples from the full join of all relations in the dataset; their training time can be non-trivial, however (Table 2). BayesCard and DeepDB incur relatively low estimation errors on JOBlight (also reported in prior work [54]). They do not work on the other two benchmarks. In contrast, FactorJoin incurs high errors; the reasons are specific to the choice in its implementation to enforce independence between the data columns (cf. prior work [54]). It is not a limitation of its approach, which can build high-dimensional probability distributions over the attributes of each relation to capture their correlation. Keeping such distributions and working with them would incur a high penalty at estimation time and require large space. Generally, learned estimators are known to poorly generalize to new datasets and have large training times and extra space [20].

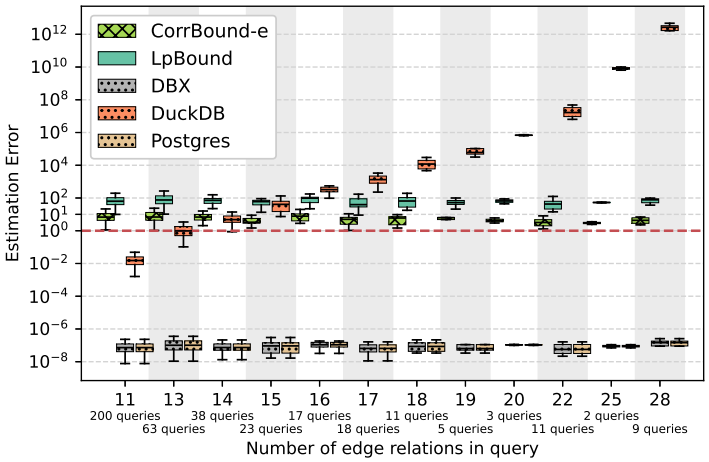


Fig. 2. Estimation errors for the SM cyclic queries, categorized by the number of copies of the edge relation in the query. All missing estimators cannot handle cyclic queries.

7.3 Estimation Errors for Cyclic Queries

Fig. 2 shows that CorrBound-e outperforms LpBound and traditional estimators for the SM cyclic queries. For this benchmark, CorrBound-d and CorrBound-e coincide as we did not compress the degree vectors. The cardinality tensor (and therefore its exact decomposition) only requires 7.3 MB, since: (1) all queries use, regardless of their number of joins, the same one edge relation and one vertex relation; (2) there are only 255 MCVs; and (3) the domain size (and thus the length of the full degree vector) is 317K, approximately 8x smaller than JOBlight's 2.5M.

By incorporating the intra- and inter-relation inequalities in addition to the ℓ_p-norm inequalities used by LpBound, CorrBound-e improves the estimation error of LpBound by a factor between 6.7 and 16.3 times. This brings the median estimation error of CorrBound-e to 6.23, whereas for LpBound is 65.35.

The traditional estimators of Postgres and DBX systematically overestimate join selectivity, producing an estimate of 1 for all SM queries, as also observed in prior work [54]. The estimation error of DuckDB increases exponentially with increasing subgraph pattern connectivity. This is because DuckDB ignores most constraints imposed by edge relations in case of highly cyclic queries, leading to progressively larger overestimates for queries with many edge relations. An in-depth explanation of this behavior is given in prior work [54].

7.4 Estimation Times

Query optimizers typically require the cardinality estimates for all connected subqueries of a query. By estimation time, we mean the time to compute all these estimates. Following prior work [20, 51, 54], we use the subqueries produced by Postgres's planner for a given query. The range (min-max) of the number of subqueries is 1-26 (1-75) for JOBlight (STATS). Table 1 gives the average estimation times over all queries in each of the two benchmarks.

We break down the estimation of the CorrBound variants into the computation of the inner products and building and solving the linear programs. The degree vectors are large for JOBlight and relatively smaller for STATS: This is reflected in the time to compute their inner products. CorrBound-d and CorrBound-s take 7 and respectively 31 times less than CorrBound-e for this

Estimator	JOBlight	STATS
LpBound	1.5	1.6
SafeBound	13.0	5.6
CorrBound-d	8.48 + 0.74	3.57 + 0.96
CorrBound-s	1.9 + 0.74	1.08 + 0.94
CorrBound-e	59.13 + 0.75	5.97 + 0.96
UCISketch	602.7	54.47
FactorJoin	166.5	626
BayesCard	21.7	-
DeepDB	28.6	-

Table 1. Wall-clock times (ms) to estimate for a query and all of its connected subqueries, averaged over all queries of a benchmark. For CorrBound variants, this is separated into the time to compute the generalized inner products plus and the time to build and solve the linear program. For UCISketch, this includes the time to filter the relation using the query predicates, then build the sketch matrices. (-) means unavailable data or unsupported workload.

Estimator	JOBlight		STATS		SM	
	Time	Space	Time	Space	Time	Space
LpBound	12.88	1.25	27.84	4.76	10.74	0.34
SafeBound	162.09	1.75	32.04	5.94	-	-
CorrBound-d	155.71	132.33	24.81	85.48	19.73	7.28
CorrBound-s	64.10	57.24	90.38	71.41	19.73	7.28
CorrBound-e	122.69	555.50	19.98	155.72	19.73	7.28
UCISketch	0	0	0	0	-	-
FactorJoin	4,990.9	22.8	360.92	8.2	-	-
BayesCard	493.36	1.6	-	-	-	-
DeepDB	1,191.2	34.0	-	-	-	-

Table 2. Precompute time (sec) and extra space (MB) for statistics. For CorrBound-d, we store the tensor decomposition of rank $250k$ ($15k$) and for CorrBound-s, we store 5×4096 (5×512) sketch matrices for JOBlight (STATS). For CorrBound-e we store all degree vectors uncompressed. The reported numbers for CorrBound variants are in addition to those for LpBound, as the former also uses the statistics of the latter. (-) means unsupported workload.

computation for JOBlight due to their effective compressions. For STATS, the time difference is much less since the degree vectors are already small before compression. The degree vectors are stored in a sparse format so as to take less space. Yet for computing the inner products at estimation time, they are first converted back to the dense format. This conversion sped up the inner product computation by a 2x factor. The time to build and solve the LPs is relatively insignificant as they are massively parallelized. This is about half the time needed by LpBound, since our implementation is more efficient.

BayesCard and DeepDB take about the same time as CorrBound-d. The other competitors take significantly more time. UCISketch's time includes the times to: filter the relations based on the predicates at runtime, construct the sketches, and multiply the sketches. Therefore, its time is significantly larger. STATS is much smaller than JOB, so the time to filter the relations is much smaller. FactorJoin takes significant time: it computes the estimates for all subqueries of a query in a sequential (not parallelized) bottom-up traversal of a left-deep query plan.

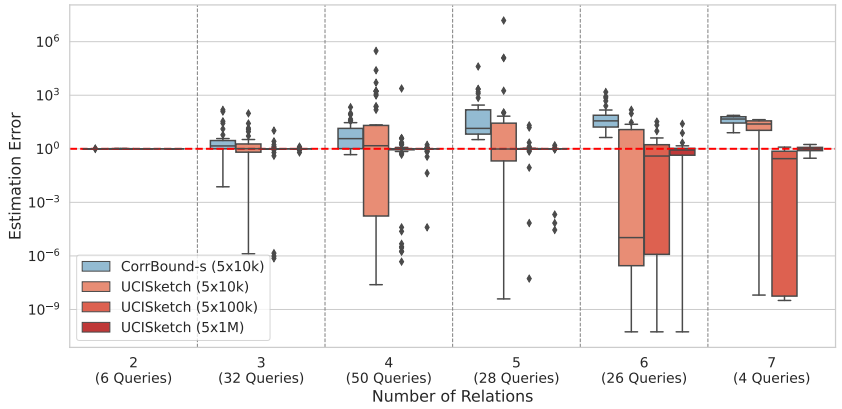


Fig. 3. Estimation error of CorrBound-s (blue) and UCI-sketch (red) for STATS in case the input relations are already filtered with the selection predicates. The queries are grouped by the number of joined relations.

7.5 Space and Precompute Time for Statistics

Table 2 gives the time to compute the statistics and the space to store them. The time of CorrBound remains much lower than of PGM-based estimators, which require model training. UCISketch computes the sketches on the fly at estimation time, so it has no precomputation. When configured with $5 \times 1M$ sketch matrices, UCISketch needs 137MB of space on average for each query at estimation time [21].

CorrBound-d computes and stores the SVD of the matrix of degree vectors and the top-m rank-1 tensors in case of k-dimensional cardinality tensors for $k > 2$. The SVD alone takes 40MB of the reported space. CorrBound-s is the most space efficient out of the CorrBound variants because of the highly compact nature of sketches and of only keeping sketches to support 2-way and 3-way joins. In contrast, CorrBound-d takes a bit more than 2x space and time for JOBlight since the degree vectors are very large in that dataset. The degree vectors in STATS are much smaller, yet there are many more degree vectors since there are many more predicates per relation in the STATS queries and therefore more MCVs and histogram buckets, each with their own degree vectors.

For SM, the statistics are only for the edge relation, so they can be computed quickly and require little space. The degree vectors are also small so there is no need for compression (all CorrBound variants use the same space and precompute time).

7.6 Deep Dive in CorrBound-s and UCISketch

The dimensions of the sketch matrix impacts the estimation error of CorrBound-s: large sketch matrices, especially those with more columns, reduce estimation variance and make underestimation less frequent. The ideal size is data-dependent. STATS, which has smaller attribute domains, requires small 5×512 sketch matrices for CorrBound-s to achieve good accuracy, while JOBlight requires 5×4096 matrices.

We next consider the default setting of UCISketch, where the predicates are filtered in a preprocessing step and the estimator is called for a join query without predicates (this contrasts with the common setting reported in Fig. 1, where predicates are not known in a preprocessing step). In this default setting, we build the sketches for CorrBound-s and UCISketch on the pre-filtered data. For CorrBound-s, all other statistics are computed as before.

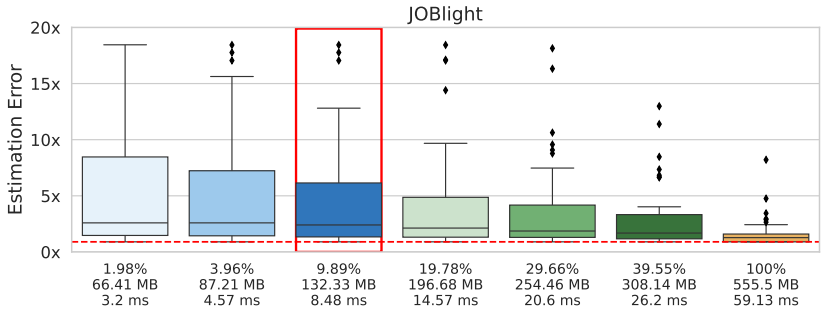


Fig. 4. Estimation error of CorrBound-d for different levels of compression. The percentage is the fraction of rank-1 tensors retained in the decomposition. The memory and time costs are displayed below the x-axis. Due to sparse storage of degree vectors, space savings do not scale linearly with compression ratio: the retained rank-1 tensors correspond to heavy hitters with many non-zero elements, while discarded tensors have sparser representations. The red box marks the hyperparameters selected for benchmarking against other systems.

Fig. 3 plots the estimation errors of CorrBound-s and UCISketch on STATS in this default setting. CorrBound-s provides accurate estimates even with small sketch matrices ($5 \times 1k$ and $5 \times 10k$). In contrast, UCISketch is less accurate when using sketch matrices of the same size and produces many negative estimates. UCISketch's accuracy becomes competitive only with significantly larger sketch matrices. For queries with up to 5 relations, it requires 10x larger sketch matrices ($5 \times 100k$). For queries with more than 5 relations, UCISketch needs 100x larger sketch matrices ($5 \times 1M$) to achieve accurate estimates. We observed a similar behavior for JOBlight (not included here).

7.7 Deep Dive in CorrBound-d

Fig. 4 analyzes the relationship between the tensor rank of the decomposition of the cardinality tensor, the extra space needed to keep the statistics, and the time to estimate the cardinalities of all connected subqueries of a query, averaged over all queries in JOBlight. The third setting, i.e., 9.89% of the rank-1 tensors are kept and the sum of the remaining ones are upper bounded by one rank-1 tensor, is the one used in Fig. 1. The space saving is not proportional to the fraction of the retained rank-1 tensors: We are selecting the rank-1 tensors that contribute most of the mass to the cardinality tensor, so they more likely have many non-zero entries. The discarded rank-1 tensors likely have many more zero values and consume less memory when stored in a sparse vector format. We observed a similar takeaway for STATS (not included here).

7.8 End-to-End Runtime Performance

We evaluate the impact of the cardinality estimators on query execution by measuring the end-to-end evaluation time of Postgres when we inject the estimates from CorrBound-d, LpBound, and Postgres's own estimator for all connected subqueries of each query. We find that, whereas the prior pessimistic cardinality estimators (SafeBound [12] and LpBound [54]) improved the runtime for long-running queries, CorrBound also improves the runtime for short-running queries.

Fig. 5 shows Postgres's evaluation time for the JOBlight queries when using various estimators relative to its time when provided the true cardinalities (TrueCard), with parallelism turned on. CorrBound-d is more accurate than the alternatives, resulting in execution times that approach

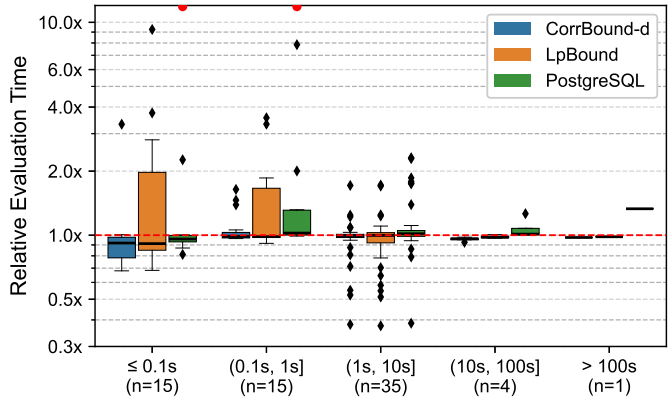


Fig. 5. Postgres evaluation times for JOBlight queries when using CorrBound-d, LpBound, and Postgres estimates relative to its times when using true cardinalities. The queries are grouped based on their evaluation times. The two red dots represent outliers for Postgres: 22.0x and 20.7x worse evaluation times than using true cardinalities.

Estimator	Paral.	Join Order	Join Operators	Time (s)
TrueCard	off	`mk,t,ci,mi,mc`	NL, NL, NL, NL	0.186
CorrBound-d	off	`mk,t,ci,mi,mc`	NL, NL, NL, HASH(mk,t,ci,mi)	0.409
LpBound	off	`mk,t,ci,mi,mc`	NL, NL, NL, HASH(mc)	0.759
Postgres	off	`mk,mc,t,mi,ci`	NL, NL, NL, NL	4.291
TrueCard	on	`mk,t,ci,mi,mc`	NL, NL, NL, NL	0.194
CorrBound-d	on	`mk,t,ci,mi,mc`	NL, NL, NL, pHASH(mk,t,ci,mi)	0.459
LpBound	on	`mk,t,ci,mi,mc`	NL, NL, NL, pHASH(mc)	0.675
Postgres	on	`mk,mc,t,mi,ci`	NL, NL, NL, NL	4.268

Table 3. Query plans and execution times for JOBlight-Q68 under different cardinality estimators. Blue text marks deviations from the best plan (TrueCard). NL denotes a nested loops join. (p)HASH(mc) denotes a (parallel) hash join building the hash table on relation mc.

those achieved using TrueCard. CorrBound-d leads to speed-ups of up to 4.1× for short-running queries (≤ 0.1 sec) and up to 1.8× for longer-running queries (> 1 sec) over LpBound. Overall, CorrBound-d outperforms LpBound on 13/70 queries, while LpBound is better on 5/70 queries; they show negligible differences (within 5% or 5 ms) for the remaining queries. For STATS (not depicted), the improvements are more pronounced: CorrBound-d leads to faster execution on 44/146 queries compared to LpBound (25 in favor of LpBound, 77 with negligible differences). The Postgres estimates lead to very high performance variability.

To understand how estimate quality affects execution times, we analyzed the query plans generated by Postgres using the different estimators for JOBlight-Q68 (see Table 3).

We first consider single-threaded plans. TrueCard gives the best plan: a left-deep join order `mk,t,ci,mi,mc` using nested-loops (NL) joins. Using CorrBound-d, the join order remains unchanged, but it overestimates the size of the intermediate result `(mk,t,ci,mi)`. This causes Postgres to replace the last NL join with a hash join (build side `mk,t,ci,mi`; probe side `mc`). Since the actual intermediate is small, the hash join incurs unnecessary overhead (slowdown 2.2×).

Using LpBound, the overestimation of the size of `(mk,t,ci,mi)` exceeds that of `mc`. Consequently, Postgres builds the hash table on `mc` (swapping the build/probe sides), which increases the cost of the last join (slowdown 4.1×). Postgres's native estimates lead to the worst plan (slowdown 23×): It joins the large `mc` relation early, producing large intermediate results that are processed using NL joins.

Enabling parallelism does not change the plans generated with TrueCard and Postgres's native estimates. In both cases, the estimated data size is too small to justify the overhead of parallelism. CorrBound-d's overestimates lead to a parallel hash join; however, this plan is worse since the actual data is small. Similarly, LpBound's overestimates trigger a parallel hash join, but this time building the hash table on `mc` in parallel helps reduce the overhead, making it faster than its single-threaded version. For other queries, we observed that the parallel plans prompted by cardinality overestimates can be faster than the single-threaded plans prompted by TrueCard.

8 Conclusion and Future Work

We introduced CorrBound, an information-theoretic pessimistic cardinality estimator whose key innovation is a new information inequality that relates joint entropies of join variables in the query to statistics computable over the data. This informs CorrBound about the correlations between the foreign keys, both intra-relation, and inter-relation. To make this approach practical, we proposed methods based on low-rank tensor decompositions and extended the state-of-the-art algorithms for computing the upper bound on the query cardinality. Our experiments showed that the correlation information leads to significantly better upper bounds.

We have only exploited simple correlations with our new inequality. Future work can exploit its full potential. For example, the inequality allows us, in theory, to capture statistics of a longer chain of joins $R_1(X_1, X_2), \ldots, R_n(X_n, X_{n+1})$. The challenge is to integrate it with selection predicates. In CorrBound, we used compressed degree vectors to compute these statistics at runtime, but only when all joins are on the same variable. Future work is needed to enable statistics over more complex expressions.

Future work can also address the selective computation of intra-relation correlation statistics. Among the benchmarks used in this work, only STATS and SM have relations with two join variables (JOBlight only has one join variable per relation). As the number of join variables per relation increases, the number of pairwise intra-relation statistics grows quadratically. To address this scalability challenge, one can use Chow-Liu trees to select pairs of join variables that capture most of the information about the joint distribution, following prior approaches [46, 52].

We would also like to understand the sensitivity of the optimal solutions of the linear programs that compute the cardinality estimates to the updates (inserts and deletes) to the underlying database. We expect that the linear programs only need to be recomputed infrequently, as local updates would most likely change the values of the statistics but not the weights of the inequalities (dual values) used for the optimal solution.

A stress test to consider for CorrBound is a recently published comprehensive benchmark called SQLStorm [41]. This requires extensions of the supported query language to include window functions, outer-joins, and having-clauses.

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, 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. doi:10.1145/3651597

[2] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2016. Computing Join Queries with Functional Dependencies. In *PODS*. 327–342. doi:10.1145/2902251.2902289

[3] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2017. What Do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog Have to Do with One Another?. In *PODS*. 429–444. doi:10.1145/3034786.3056105 Extended version available at http://arxiv.org/abs/1612.02503.

[4] Noga Alon, Phillip B. Gibbons, Yossi Matias, and Mario Szegedy. 2002. Tracking Join and Self-Join Sizes in Limited Storage. *J. Comput. Syst. Sci.* 64, 3 (2002), 719–747. doi:10.1006/JCSS.2001.1813

[5] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.* 58, 1 (1999), 137–147. doi:10.1006/JCSS.1997.1545

[6] Albert Atserias, Martin Grohe, and Dániel Marx. 2013. Size Bounds and Query Plans for Relational Joins. *SIAM J. Comput.* 42, 4 (2013), 1737–1767. doi:10.1137/110859440

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

[8] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. 2004. Finding frequent items in data streams. *Theor. Comput. Sci.* 312, 1 (2004), 3–15. doi:10.1016/S0304-3975(03)00400-6

[9] Yu Chen and Ke Yi. 2017. Two-Level Sampling for Join Size Estimation. In *SIGMOD*. 759–774. doi:10.1145/3035918.3035921

[10] G. Cormode and K. Yi. 2020. *Small Summaries for Big Data*. Cambridge University Press. https://books.google.com/books?id=6in-DwAAQBAJ

[11] Kyle Deeds, Dan Suciu, Magda Balazinska, and Walter Cai. 2023. Degree Sequence Bound for Join Cardinality Estimation. In *ICDT*. 8:1–8:18. doi:10.4230/LIPICS.ICDT.2023.8

[12] 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. doi:10.1145/3588907

[13] Amol Deshpande, Minos N. Garofalakis, and Rajeev Rastogi. 2001. Independence is Good: Dependency-Based Histogram Synopses for High-Dimensional Data. In *SIGMOD*. 199–210. doi:10.1145/375663.375685

[14] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek R. Narasayya, and Surajit Chaudhuri. 2019. Selectivity Estimation for Range Predicates using Lightweight Models. *Proc. VLDB Endow.* 12, 9 (2019), 1044–1057. doi:10.14778/3329772.3329780

[15] Carl Eckart and Gale Young. 1936. The Approximation of One Matrix by Another of Lower Rank. *Psychometrika* 1, 3 (1936), 211–218. doi:10.1007/BF02288367

[16] Cristian Estan and Jeffrey F. Naughton. 2006. End-biased Samples for Join Cardinality Estimation. In *ICDE*. doi:10.1109/ICDE.2006.61

[17] Sumit Ganguly, Phillip B. Gibbons, Yossi Matias, and Abraham Silberschatz. 1996. Bifocal Sampling for Skew-Resistant Join Size Estimation. In *SIGMOD*. 271–281. doi:10.1145/233269.233340

[18] Lise Getoor, Benjamin Taskar, and Daphne Koller. 2001. Selectivity Estimation using Probabilistic Models. In *SIGMOD*. 461–472. doi:10.1145/375663.375727

[19] Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, and Paul Valiant. 2012. Size and Treewidth Bounds for Conjunctive Queries. *J. ACM* 59, 3 (2012), 16:1–16:35. doi:10.1145/2220357.2220363

[20] 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. doi:10.14778/3503585.3503586

[21] Mike Heddes, Igor Nunes, Tony Givargis, and Alex Nicolau. 2024. Convolution and Cross-Correlation of Count Sketches Enables Fast Cardinality Estimation of Multi-Join Queries. *Proc. ACM Manag. Data* 2, 3 (2024), 129. doi:10.1145/3654932

[22] 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. doi:10.14778/3384345.3384349

[23] Pan Hu and Boris Motik. 2024. Accurate Sampling-Based Cardinality Estimation for Complex Graph Queries. *ACM Trans. Database Syst.* 49, 3 (2024), 12:1–12:46. doi:10.1145/3689209

[24] Qi Huangfu and J. A. J. Hall. 2018. Parallelizing the Dual Revised Simplex Method. *Math. Program. Comput.* 10, 1 (2018), 119–142. doi:10.1007/S12532-017-0130-5

[25] Sungjin Im, Benjamin Moseley, Hung Ngo, and Kirk Pruhs. 2025. Efficient Algorithms for Cardinality Estimation and Conjunctive Query Evaluation With Simple Degree Constraints. *Proc. ACM Manag. Data* 3, 2, Article 96 (2025). doi:10.1145/3725233

[26] Yesdaulet Izenov, Asoke Datta, Florin Rusu, and Jun Hyung Shin. 2021. COMPASS: Online Sketch-based Query Optimization for In-Memory Databases. In *SIGMOD*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava

(Eds.). 804–816. doi:10.1145/3448016.3452840
[27] Sai Vikneshwar Mani Jayaraman, Corey Ropell, and Atri Rudra. 2021. Worst-case Optimal Binary Join Algorithms under General ℓ_p Constraints. arXiv:2112.01003 [cs.DB] https://arxiv.org/abs/2112.01003
[28] Kyoungmin Kim, Jisung Jung, In Seo, Wook-Shin Han, Kangwoo Choi, and Jaehyok Chong. 2022. Learned Cardinality Estimation: An In-depth Study. In *SIGMOD*. 1214–1227. doi:10.1145/3514221.3526154
[29] Kyoungmin Kim, Hyeonji Kim, George Fletcher, and Wook-Shin Han. 2021. Combining Sampling and Synopses with Worst-Case Optimal Runtime and Quality Guarantees for Graph Pattern Cardinality Estimation. In *SIGMOD*. 964–976. doi:10.1145/3448016.3457246
[30] 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*. http://cidrdb.org/cidr2019/papers/p101-kipf-cidr19.pdf
[31] Tamara G. Kolda and Brett W. Bader. 2009. Tensor Decompositions and Applications. *SIAM Rev.* 51, 3 (2009), 455–500. doi:10.1137/07070111X
[32] Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. 2017. Cardinality Estimation Done Right: Index-Based Join Sampling. In *CIDR*. http://cidrdb.org/cidr2017/papers/p9-leis-cidr17.pdf
[33] 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. doi:10.1007/s00778-017-0480-7
[34] Beibin Li, Yao Lu, and Srikanth Kandula. 2022. Warper: Efficiently Adapting Learned Cardinality Estimators to Data and Workload Drifts. In *SIGMOD*. 1920–1933. doi:10.1145/3514221.3526179
[35] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. 2019. Wander Join and XDB: Online Aggregation via Random Walks. *ACM Trans. Datab. Syst.* 44, 1 (2019), 2:1–2:41. doi:10.1145/3284551
[36] Jan Motl and Oliver Schulte. 2024. Stats Stack Exchange Network Dataset. https://relational.fel.cvut.cz/dataset/Stats The CTU Prague Relational Learning Repository.
[37] Silvan Reiner and Michael Grossniklaus. 2023. Sample-Efficient Cardinality Estimation Using Geometric Deep Learning. *Proc. VLDB Endow.* 17, 4 (2023), 740–752. doi:10.14778/3636218.3636229
[38] L. J. Rogers. 1888. An extension of a certain theorem in inequalities. *Messenger of Mathematics, New Series* XVII, 10 (1888), 145—-150.
[39] Pratanu Roy, Arijit Khan, and Gustavo Alonso. 2016. Augmented Sketch: Faster and More Accurate Stream Processing. In *SIGMOD*. 1449–1463. doi:10.1145/2882903.2882948
[40] Florin Rusu and Alin Dobra. 2008. Sketches for size of join estimation. *ACM Trans. Database Syst.* 33, 3 (2008), 15:1–15:46. doi:10.1145/1386118.1386121
[41] Tobias Schmidt, Viktor Leis, Peter Boncz, and Thomas Neumann. 2025. SQLStorm: Taking Database Benchmarking into the LLM Era. *Proc. VLDB Endow.* 18, 11 (2025), 4144–4157. doi:10.14778/3749646.3749683
[42] 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*. 23–34. doi:10.1145/582095.582099
[43] Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.* 27, 3 (1948), 379–423. doi:10.1002/J.1538-7305.1948.TB01338.X
[44] Shixuan Sun and Qiong Luo. 2020. In-Memory Subgraph Matching: An In-depth Study. In *SIGMOD*. 1083–1098. doi:10.1145/3318464.3380581
[45] Brian Tsan, Asoke Datta, Yesdaulet Izenov, and Florin Rusu. 2024. Approximate Sketches. *Proc. ACM Manag. Data* 2, 1 (2024), 66:1–66:24. doi:10.1145/3639321
[46] Kostas Tzoumas, Amol Deshpande, and Christian S. Jensen. 2011. Lightweight Graphical Models for Selectivity Estimation Without Independence Assumptions. *Proc. VLDB Endow.* 4, 11 (2011), 852–863. http://www.vldb.org/pvldb/vol4/p852-tzoumas.pdf
[47] David Vengerov, Andre Cavalheiro Menck, Mohamed Zaït, and Sunil Chakkappen. 2015. Join Size Estimation Subject to Filter Conditions. *Proc. VLDB Endow.* 8, 12 (2015), 1530–1541. doi:10.14778/2824032.2824051
[48] Feiyu Wang, Qizhi Chen, Yuanpeng Li, Tong Yang, Yaofeng Tu, Lian Yu, and Bin Cui. 2023. JoinSketch: A Sketch Algorithm for Accurate and Unbiased Inner-Product Estimation. *Proc. ACM Manag. Data (SIGMOD)* 1, Article 81 (2023). doi:10.1145/3588935
[49] TaiNing Wang and Chee-Yong Chan. 2020. Improved Correlated Sampling for Join Size Estimation. In *ICDE*. 325–336. doi:10.1109/ICDE48307.2020.00035
[50] 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. doi:10.14778/3461535.3461552
[51] 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 (SIGMOD) (2023), 41:1–41:27. doi:10.1145/3588721
[52] Ziniu Wu and Amir Shaikhha. 2020. BayesCard: A Unified Bayesian Framework for Cardinality Estimation. *CoRR* abs/2012.14743 (2020). arXiv:2012.14743 https://arxiv.org/abs/2012.14743

[53] 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. doi:10.14778/3421424.3421432

[54] 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, Article 184 (2025). doi:10.1145/3725321

Received July 2025; revised October 2025; accepted November 2025