

Metis: Workload-Aware Multi-Dimensional Histograms for Cardinality Estimation

Xuhua Huang
East China Normal University
xhhuang@stu.ecnu.edu.cn

Xinyu Xiang
East China Normal University
xyxiang@stu.ecnu.edu.cn

Zirui Hu
East China Normal University
zrhu@stu.ecnu.edu.cn

Zhicheng Pan
East China Normal University
zcpan@stu.ecnu.edu.cn

Siyang Weng
East China Normal University
syweng@stu.ecnu.edu.cn

Rong Zhang*
East China Normal University
rzhang@dase.ecnu.edu.cn

Chengcheng Yang
East China Normal University
ccyang@dase.ecnu.edu.cn

Xuan Zhou
East China Normal University
xzhou@dase.ecnu.edu.cn

Quanqing Xu
OceanBase, Ant Group
xuquanqing.xqq@oceanbase.com

Chuanhui Yang
OceanBase, Ant Group
rizhao.ych@oceanbase.com

ABSTRACT

Cardinality estimation (*CardEst*) is a long-standing cornerstone of query optimizers for selecting high-quality execution plans. Although learned *CardEst* methods have been extensively studied, they struggle to make trade-offs among accuracy, efficiency, and deployability. Consequently, mainstream database systems still employ traditional histogram-based estimators due to their practicality. However, managing histograms remains challenging in the following three aspects: 1) histogram *preparation* (effectively identifying correlated attributes); 2) histogram *construction* (precisely modeling joint distributions); 3) histogram *application* (elaborately combining multiple histograms). To address these challenges, we propose Metis, which (i) employs a feedback-driven paradigm to analyze workload-imperative correlations and determine appropriate attribute subsets; (ii) derives rich probabilistic constraints to construct histograms that precisely capture joint distributions; and (iii) proposes shared-attribute-driven key histogram selection and probabilistic-graph-based multi-histogram inference for accurate estimation. Extensive experiments on well-established datasets and diverse queries demonstrate the superiority of Metis.

PVLDB Reference Format:
Xuhua Huang, Xinyu Xiang, Zirui Hu, Zhicheng Pan, Siyang Weng, Rong Zhang, Chengcheng Yang, Xuan Zhou, Quanqing Xu, and Chuanhui Yang. Metis: Workload-Aware Multi-Dimensional Histograms for Cardinality Estimation. PVLDB, 20(2): 122 - 135, 2026.
doi:10.14778/3849423.3849428

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

*Rong Zhang is the corresponding author.
This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 20, No. 2 ISSN 2150-8097.
doi:10.14778/3849423.3849428

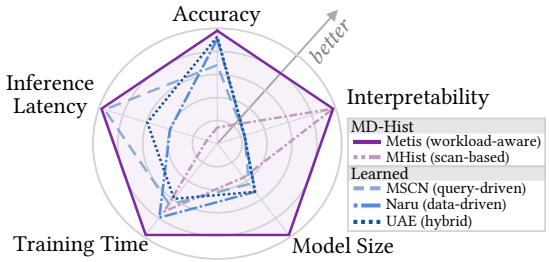


Figure 1: Relative performance comparison on the DMV dataset. Outer rings indicate better performance. For each metric, results are normalized relative to the best-performing method, and each concentric interval represents 2×.

1 INTRODUCTION

Cardinality estimation (*CardEst*) predicts the output size of query operators, providing a key signal for cost-based optimizers [12, 23, 29, 30, 70]. In particular, single-table *CardEst* estimates the number of tuples satisfying selection predicates over a base relation. It directly influences early optimization decisions such as access path selection and serves as the foundation for multi-table *CardEst* [59]. Thus, even small errors in single-table *CardEst* can be amplified as optimization proceeds, distorting the estimated costs of downstream operators and leading to suboptimal plans [3, 41, 42].

In the literature, an ideal *CardEst* should be *accurate* for high-quality query plan generation, *efficient* with negligible inference overhead, and *deployable* in industrial query optimizers. Moreover, *deployability* attracts increasing attention in commercial DBMSs [71], and it requires the following characteristics: ❶ Compactness, as estimators are memory-resident and frequently invoked during query optimization. ❷ Fast adaptation, enabling rapid rebuilding after data updates to maintain accuracy. ❸ Interpretability, which refers to satisfying *fundamental statistical properties*, including stability, monotonicity, consistency, and fidelity [71]. For example, stability requires identical estimates for the same query across repeated runs. Without them, inconsistent or unstable *CardEst* would mislead the cost model, causing unpredictable plan selection and limiting practical deployment [24, 66].

Recent years have witnessed extensive efforts on learned methods for single-table *CardEst*. They either learn mappings from query features to cardinalities (*query-driven* [8, 26, 36]), learn joint data distributions from the underlying data (*data-driven* [27, 76, 79]), or combine both strategies (*hybrid* [72]). However, query-driven methods are inherently limited by the coverage of training workload, while data-driven and hybrid methods often impose substantial training cost (e.g., frequent data access) and non-trivial inference overhead (e.g., runtime sampling) [25]. Moreover, most learned estimators do not provide formal guarantees on fundamental statistical properties due to the opaque nature of their learned models [71, 78].

As summarized in Fig. 1, our comprehensive evaluation demonstrates that existing learned estimators achieve accuracy largely by sacrificing efficiency and deployability, limiting their practical applications. Most database systems continue to employ traditional histogram-based estimators, as they have been extensively validated in practice [18, 50, 52, 64]. However, one-dimensional histograms (1D-Hists) estimate multi-attribute predicates under the attribute value independence (AVI) assumption. When attributes are correlated, this assumption is often violated, leading to estimation errors of up to 10^4–10^8 in real-world workloads [4, 41]. Thus, multi-dimensional histograms (MD-Hists) are proposed to capture multi-attribute joint distributions. Nevertheless, constructing an MD-Hist over all attributes is infeasible due to exponential bucket growth with dimensionality [38]. Accordingly, prior work explores constructing multiple MD-Hists over selected attribute subsets and combining them to estimate predicates that are not covered by any single histogram [6, 31, 44, 45, 68]. However, three key challenges remain in achieving accurate, efficient, and deployable *CardEst*:

Histogram Preparation (C1). Under a limited statistics budget, histogram preparation aims to select attribute subsets with high *CardEst* utility, i.e., whose joint modeling yields large reductions in *CardEst* errors for the target workload. However, the number of candidate subsets grows exponentially with the number of attributes, making exhaustive evaluation computationally prohibitive. Existing estimators therefore either require attribute subsets to be specified *a priori* [2, 19, 56], limiting their practical applicability, or rely on intrinsic data correlations [6, 11, 68], which ignore workload access patterns and may prioritize strongly correlated subsets with low *CardEst* utility. To assess subset utility, SASH [44] constructs and evaluates candidate histograms using query feedback, but such candidate-wise evaluation incurs substantial overhead.

Histogram Construction (C2). Scan-based methods [6, 19, 20, 56] typically evaluate and compare candidate bucket partitions iteratively, requiring repeated scans of the underlying data to compute statistics for newly generated buckets and thereby incurring substantial construction overhead. To avoid this, feedback-driven methods [1, 2, 34, 44, 61, 65] construct histograms from historical query predicates and observed cardinalities. However, they face two key limitations. First, they cannot accurately exploit feedback from *partially-contained* queries whose predicate attributes are not fully contained in the histogram attributes, as the observed cardinalities of such queries do not directly yield equality constraints on bucket probabilities. This leaves substantial feedback unexploited, even though such queries are prevalent in practice due to the mismatch between limited histogram coverage and diverse query predicates [25], and they are informative about the underlying joint data distribution. Second, they typically split existing buckets to align them with newly observed query predicates. As a result, the bucket number grows exponentially with the number of observed queries, leading to prohibitive construction overhead and preventing these methods from benefiting from sufficient feedback [54].

Histogram Application (C3). When the predicate attributes of a query are not covered by any single MD-Hist, the *CardEst* process must perform joint inference by combining multiple histograms. Existing methods [6, 44, 68] typically organize multiple MD-Hists into a globally decomposable structure, usually a junction tree, and combine them according to the corresponding global factorization. However, this structural requirement may prevent incompatible yet high-utility MD-Hists from being constructed. For example, suppose that MD-Hists over $\{A_1, A_2\}$, $\{A_2, A_3\}$, and $\{A_3, A_4\}$ have been organized into such a structure. A subsequently identified high-utility MD-Hist over $\{A_1, A_4, A_5\}$ cannot be incorporated because it is incompatible with the existing structure, forcing the dependency between A_1 and A_4 to be inferred indirectly along the fixed path $\{A_1, A_2\}$–$\{A_2, A_3\}$–$\{A_3, A_4\}$ and potentially propagating local errors. *CLTree* [69] stores only pairwise marginals and dynamically organizes them into a query-specific Chow–Liu tree for each incoming query. However, for a high-dimensional query, it needs to combine multiple pairwise factors along a relatively long inference path, introducing more conditional-independence approximations and potentially missing higher-order local dependencies.

To address these challenges, we propose Metis, designed for seamless integration into industrial query optimizers. To address **C1**, we propose a *feedback-driven attribute-subset selection strategy* that directly quantifies subset utility from historical *CardEst* errors. Then, we formulate subset selection as a storage-constrained 0–1 integer linear program (ILP) to identify high-utility subsets without costly candidate-histogram construction and evaluation. To address **C2**, we introduce a *projection-aware constraint derivation strategy* to derive rich bucket-probability constraints from partially-contained queries. Furthermore, we adopt an *a priori* bucket partitioning based on the Cartesian product of single-attribute intervals, preventing bucket growth as more feedback is incorporated. To address **C3**, we propose a *flexible inference framework* that supports the direct construction of higher-dimensional MD-Hists over high-utility attribute combinations (i.e., $\{A_1, A_4, A_5\}$ in the above example) without the constraints of a single global structure. At inference time, to improve accuracy while preserving interpretability, Metis deterministically selects query-relevant MD-Hists using a shared-attribute-driven strategy, which prioritizes larger overlaps to provide a richer conditioning context and leave fewer residual dependencies unmodeled. Then, it performs marginal inference over the resulting query-specific factor graph. For example, given a query whose predicate attributes are $\{A_1, A_2, A_4, A_5\}$, Metis directly combines the MD-Hists over $\{A_1, A_4, A_5\}$ and $\{A_1, A_2\}$ through their shared attribute A_1 to estimate the query cardinality.

In summary, we make the following contributions:

- We propose Metis, a novel workload-aware method for constructing and utilizing multi-dimensional histograms.
- We open-source the implementation of Metis, supporting efficient deployment and seamless integration with DBMSs [5].
- Extensive experiments demonstrate that Metis achieves a good balance between accuracy, efficiency, and deployability.

2 PRELIMINARIES

In this paper, we target single-table cardinality estimation for selection queries with conjunctive predicates, following the majority of histogram-based methods [2, 6, 19, 20, 44, 56, 65] and learned methods under single-table settings [8, 36, 59, 72, 76]. Each predicate is defined over a single attribute, imposing either an equality or a range constraint. Note, METIS can be extended to support negations and disjunctions [54, 76] with selectivities computed using the inclusion–exclusion principle [9].

2.1 Cardinality Estimation

Data, Queries and Query Feedback. Given a table T with $|T|$ tuples and n attributes $\{A_1, A_2, \ldots, A_n\}$, a selection query q specifies a predicate σ over an attribute subset $\mathcal{A}_\sigma$. Formally, the predicate is a conjunction of single-attribute predicates, i.e., $\sigma = \bigwedge_{A_i \in \mathcal{A}_\sigma} \sigma_{A_i}$. Moreover, each σ_{A_i} corresponds to a region $R(\sigma_{A_i})$, and the overall query region is $R(\sigma) = \prod_{A_i \in \mathcal{A}_\sigma} R(\sigma_{A_i})$. We denote query feedback as $\mathcal{Q}_F = \{(q_i, Card(q_i))\}$, where each pair consists of a query q_i and its observed cardinality $Card(q_i)$. Such feedback can be obtained from query execution logs with negligible overhead [45, 65].

Cardinality Estimation. Given a query q with predicate σ, cardinality estimation aims to predict the number of tuples in T satisfying σ before execution [12]. Let $Sel(\sigma)$ denote the selectivity of σ, defined as the fraction of tuples in T that fall within the query region. The cardinality of the query is then calculated as $Card(q) = Sel(\sigma) \cdot |T|$. In general, the estimation error is measured by *Q-error* [48], defined as $\max(\frac{Card(q)}{\widehat{Card}(q)}, \frac{\widehat{Card}(q)}{Card(q)})$, where $Card(q)$ and $\widehat{Card}(q)$ are true and estimated cardinalities, respectively.

2.2 Statistics

Single-attribute Statistics. A single-attribute statistic S_i for an attribute A_i partitions its domain $D(A_i)$ into disjoint buckets $\mathcal{B}_{A_i} = \{B^1_{A_i}, B^2_{A_i}, \ldots\}$. Each bucket $B^k_{A_i}$ represents an interval $(lb^k, rb^k]_{A_i}$, where lb^k/rb^k denote the left/right boundaries of the k-th bucket. Then, the probability mass $P(B^k_{A_i})$ is defined as the fraction of tuples falling into $(lb^k, rb^k]_{A_i}$. In practice, DBMSs automatically maintain such statistics (e.g., equi-depth histograms and most common values (MCV) [15, 18]), forming a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$.

Multi-dimensional Histograms. A multi-dimensional histogram (MD-Hist) H summarizes the joint distribution over an attribute subset $\mathcal{A}_H$ by partitioning the joint domain $D(\mathcal{A}_H) = \prod_{A_i \in \mathcal{A}_H} D(A_i)$ into disjoint buckets $\mathcal{B}_H = \{B^1_H, B^2_H, \ldots\}$. Each bucket B^k_H corresponds to a hyper-rectangle region $R(B^k_H) = \prod_{A_i \in \mathcal{A}_H} (lb^k, rb^k]_{A_i}$ and is associated with a probability mass $P(B^k_H)$.

2.3 Cardinality Estimation with MD-Hists

Histogram-based cardinality estimation relies on the uniformity assumption, where tuples are uniformly distributed within each histogram bucket [33]. Accordingly, the probability mass contributed by a bucket is proportional to its *overlap ratio* with query region.

Definition 2.1 (Overlap Ratio between region R and bucket B^i_H). Given a region R and a bucket B^i_H in an MD-Hist H, the *overlap ratio* of B^i_H with respect to R is defined as $\alpha^i_H(R) = |R \cap R(B^i_H)|/|R(B^i_H)|$, i.e., the fraction of the bucket region $R(B^i_H)$ covered by R.

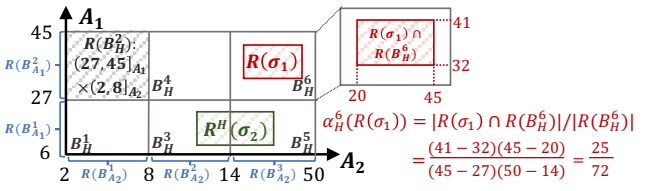


Figure 2: Notations of an MD-Hist H over $\mathcal{A}_H = \{A_1, A_2\}$.

Given a query q with predicate σ, we first consider that an MD-Hist H covers all predicate attributes, i.e., $\mathcal{A}_\sigma \subseteq \mathcal{A}_H$. In this case, the selectivity is estimated by aggregating the probability mass of intersections between $R(\sigma)$ and the buckets of H, given by $\widehat{Sel}(\sigma) = \sum_{B^i_H \in \mathcal{B}_H} P(B^i_H) \cdot \alpha^i_H(R(\sigma))$. For example, in Fig. 2, the 2D-Hist H has 6 buckets, i.e., $B^1_H, \ldots, B^6_H$. Consider the predicate $\sigma_1 : 32 < A_1 \leq 41 \wedge 20 < A_2 \leq 45$, whose query region $R(\sigma_1)$ (the red region) intersects a single bucket of H, i.e., B^6_H. Thus, we have $\widehat{Sel}(\sigma_1) = P(B^6_H) \cdot \alpha^6_H(R(\sigma_1))$, where $\alpha^6_H(R(\sigma_1)) = \frac{25}{72}$. We next consider the general case where H covers only a subset of predicate attributes, i.e., $\mathcal{A}_\sigma \cap \mathcal{A}_H \neq \emptyset$ and $\mathcal{A}_\sigma \not\subseteq \mathcal{A}_H$. In this case, H can only provide probability mass over the projection of the query region onto the attribute subset $\mathcal{A}_H$, referred to as the *projected query region*.

Definition 2.2 (Projected Query Region). Given a query predicate σ and an MD-Hist H, the *projected query region* is defined as $R^H(\sigma) = \prod_{A_i \in \mathcal{A}_\sigma \cap \mathcal{A}_H} R(\sigma_{A_i})$, which corresponds to projecting the query region $R(\sigma)$ onto the attribute subset $\mathcal{A}_H$.

For example, consider the predicate $\sigma_2 : 9 < A_1 \leq 23 \wedge 10 < A_2 \leq 31 \wedge A_3 = 9$. Its projected query region on H is $R^H(\sigma_2) = (9, 23]_{A_1} \times (10, 31]_{A_2}$ (the green region in Fig. 2). Since the predicate on A_3 in σ_2 is not reflected in $R^H(\sigma_2)$, estimating $\widehat{Sel}(\sigma_2)$ then requires combining H with additional statistics that cover A_3.

3 OVERVIEW

As shown in Fig. 3, the *offline* phase of METIS incrementally enriches the statistics library with high-utility histograms. This process is *iterative*: ❶ METIS evaluates the estimation errors of historical queries under the *current* statistics library and uses these errors to guide the attribute subset selection. ❷ METIS derives comprehensive probabilistic constraints from query feedback and statistics. Then, it jointly solves these constraints to determine bucket probabilities. The resulting histogram is materialized and added to the statistics library. Note, this cycle ❶–❷ repeats until the estimation accuracy on the historical queries converges. In the *online* phase, for each incoming query, ❸ METIS selects relevant MD-Hists based on its predicate attributes and estimates its cardinality through inference over the resulting query-specific factor graph.

Histogram Preparation (§4). This step selects a high-utility attribute subset for constructing a new MD-Hist. Existing approaches either rely on intrinsic data correlations [6, 11, 68], which may allocate limited statistics budget to subsets with low *CardEst* utility, or incur substantial overhead by constructing and evaluating candidate histograms (i.e., SASH) [44]. METIS directly quantifies subset utility from historical *CardEst* errors. Specifically, we shift the focus to *workload-imperative correlations*, defined as attribute correlations whose lack of joint modeling consistently leads to relatively large *CardEst* errors under the target workload. Based on these errors, we construct an *error-based correlation graph* (*CoG*), whose edge

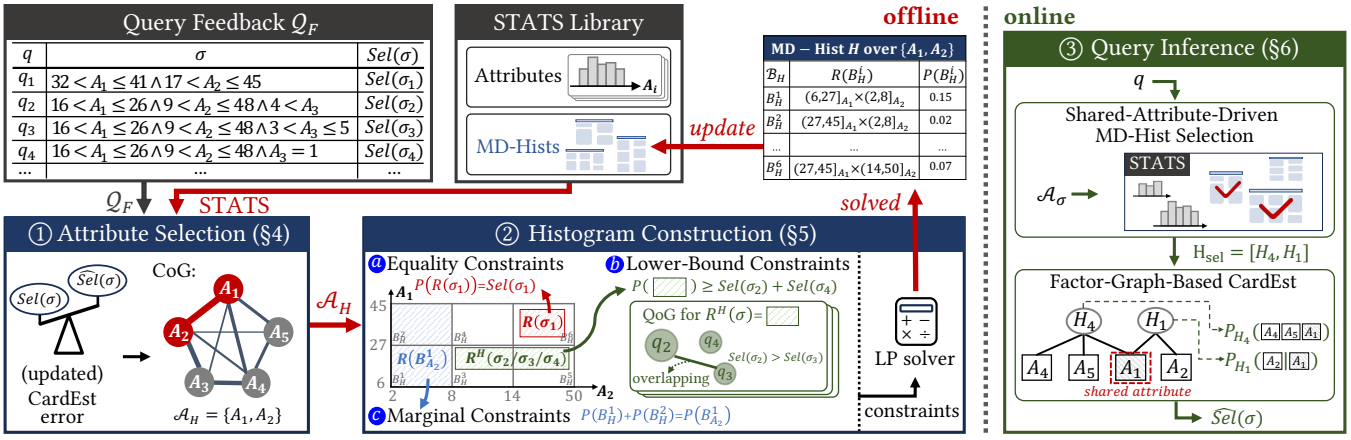


Figure 3: The framework of METIS.

weights quantify the potential error reduction from jointly modeling each workload-imperative correlation. Attribute-subset selection is then formulated as a storage-constrained 0–1 integer linear program (ILP) that maximizes the captured workload-imperative correlations. For example, for the five-attribute relation in Fig. 3❶, SASH must construct and evaluate all 10 pairwise candidates even in the first iteration, whereas METIS selects $\mathcal{A}_H = \{A_1, A_2\}$ via the ILP and constructs only the selected MD-Hist H.

Histogram Construction (§5). Given a selected attribute subset, METIS constructs an MD-Hist H by inferring its bucket probability masses. Generally, scan-based methods [19, 20, 56] may require repeated data scans for iterative bucket refinement, incurring substantial construction overhead. Alternatively, feedback-driven methods [2, 34, 65] avoid these scans by deriving constraints from historical query feedback. However, they suffer from two primary limitations that hinder the construction of high-quality histograms. First, they cannot accurately exploit feedback from *partially-contained* queries, leaving valuable constraints unexploited. Second, repeatedly splitting buckets to align with newly observed query predicates makes incorporating sufficient feedback prohibitively expensive [65]. METIS addresses these limitations using three types of constraints. First, for *fully-contained* queries, we impose their observed selectivities as *equality constraints* (ⓐ) on the probability masses of the corresponding query regions. For example, since the predicate attributes of q_1 are fully contained in $\mathcal{A}_H = \{A_1, A_2\}$, its observed selectivity yields the equality constraint $P(R(\sigma_1)) = Sel(\sigma_1)$. Next, to further exploit *partially-contained queries*, we consider their *projected query regions* on H (see Definition 2.2). Since each predicate is conjunctive, projecting it onto an attribute subset would relax the constraints and yield a projected region that *contains* the original query region. Consequently, its observed selectivity provides a valid *lower bound* on the probability mass of the projected region. Moreover, if multiple queries share the same projected region and their original query regions are pairwise disjoint, their observed selectivities can be further *aggregated* to derive a tighter lower-bound constraint (ⓑ). For example, as shown in Fig. 3❷, q_2, q_3, and q_4 share the same projected region g (i.e., $16 < A_1 \le 26 \wedge 9 < A_2 \le 48$) on H. Since the original query regions of q_2 and q_4 are disjoint, their observed selectivities can be aggregated to derive $P(g) \ge Sel(\sigma_2) + Sel(\sigma_4)$, whereas the overlapping q_3 is excluded to avoid double counting. To maximize this lower bound, METIS constructs a *query overlap graph (QoG)* for each projected region and formulates the selection of aggregable queries as a maximum-weight independent set (MWIS) problem on *QoG* [13]. Finally, METIS incorporates these equality and lower-bound constraints together with marginal constraints (ⓒ) derived from DBMS-maintained single-attribute statistics into a unified linear program (LP). Thus, we can leverage an off-the-shelf solver [21] to infer the probability masses of the histogram buckets.

Histogram Application (§6). In the *online* phase, METIS estimates query selectivities using the statistics library. Existing methods either require all MD-Hists to conform to a single globally decomposable structure [6, 44, 68], which may exclude high-utility attribute combinations, or rely only on pairwise marginals [69], which may require long inference paths for high-dimensional queries. In contrast, METIS removes the global-structure restriction on histogram construction, allowing higher-dimensional MD-Hists over high-utility attribute combinations to be constructed directly. At inference time, to improve accuracy while preserving interpretability, METIS deterministically selects query-relevant MD-Hists by prioritizing candidates that share more attributes with those already selected. The shared attributes serve as conditioning variables for combining the non-overlapping attributes of the selected MD-Hists, thereby leaving fewer residual dependencies to be approximated as conditionally independent. For example, since H_4 and H_1 share A_1, METIS approximates $\widehat{P}(A_1, A_2, A_4, A_5) \approx P_{H_4}(A_1, A_4, A_5) P_{H_1}(A_2 \mid A_1)$. The selected MD-Hists, e.g., $\mathbf{H}_{\text{sel}} = [H_4, H_1]$ in Fig. 3 ❸, are then represented as factors in a query-specific factor graph [39], over which METIS performs marginal inference for *CardEst*.

Integration with PostgreSQL. Fig. 4 shows how a PostgreSQL extension registers a planner hook [14] that invokes METIS for *CardEst* during query optimization. We provide two implementations with different flexibility–latency trade-offs. ❶ The *out-of-process* Java service supports independent development and debugging but incurs inter-process communication and serialization overhead. ❷ The *in-process* C implementation runs within PostgreSQL and thus

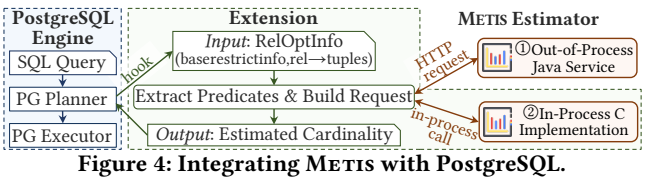


Figure 4: Integrating METIS with PostgreSQL.

Table 1: A subset of queries in Q_F and their *CardEst* results before and after constructing H over $\mathcal{A}_H = \{A_1, A_2\}$.

	Query information		$\mathcal{S}$ only	$\mathcal{S} + H$
q	σ	$Sel(\sigma)$	$\widehat{Sel}(\sigma)$	$\widehat{Sel}(\sigma)$
q_1	$0 < A_1 \le 15 \wedge 40 < A_2 \le 95$	0.02	0.107	0.02
q_2	$30 < A_1 \le 40 \wedge 50 < A_2 \le 75 \wedge A_3 = 1$	0.079	0.025	0.078
q_3	$A_1 > 32 \wedge A_5 \le 10$	0.10	0.078	0.078

eliminates this overhead, but it couples development and debugging to the server process. More details are in our technical report [5].

4 ATTRIBUTE SELECTION

4.1 Error-Based Correlation Graph

Large *CardEst* errors indicate that certain attribute correlations required by the target workload are not adequately captured under the *current* statistics. One straightforward approach is to select an attribute subset based on query-level estimation errors, which groups historical queries with the same predicate attributes and chooses the one with the largest cumulative error. Under the AVI assumption, queries involving more attributes tend to accumulate larger errors. However, in practice, these errors are often dominated by a small subset of the predicate attributes [45]. Thus, query-level estimation errors provide only coarse-grained signals and cannot identify the attribute correlations that dominate the estimation errors, leading to suboptimal subset selection. To address this issue, Metis *decomposes* query-level estimation errors into fine-grained signals associated with attribute correlations. This enables the quantification of *workload-imperative correlations*, i.e., correlations whose lack of joint modeling consistently leads to large estimation errors under the target workload. Metis constructs an *error-based correlation graph (CoG)*, which is a weighted undirected graph $\mathcal{G} = (V, E, W)$. Each node $v_i \in V$ corresponds to an attribute A_i, and an edge $e_{ij} \in E$ exists if attributes A_i and A_j co-occur in the predicate of at least one query in Q_F. Moreover, each edge e_{ij} is assigned a weight w_{ij} to quantify the workload-imperative correlation strength between A_i and A_j. Specifically, for each query q, we measure its estimation error using the widely adopted *Q-error* metric [48], as defined in §2.1. Then, the edge weight w_{ij} is defined as:

$$w_{ij} = \sum_{q \in Q_{ij}} \frac{Q\text{-}error(q) - 1}{\binom{|\mathcal{A}_\sigma|}{2}}, \tag{1}$$

where Q_{ij} denotes the set of queries in Q_F whose predicates involve both A_i and A_j. Since query-level estimation errors do not reveal which attribute pairs dominate the error, a precise decomposition would require additional modified query executions or data-correlation analysis. Therefore, Eq. 1 uniformly allocates each query-level error among the involved attribute pairs as a lightweight approximation. Although this may bias the edge weights, Metis mitigates it in two ways. First, the weight of each edge in *CoG* is accumulated across the workload, enabling attribute pairs that repeatedly contribute to high-error queries to receive larger weights. Second, uniform allocation may inflate the weights of other attribute pairs when they co-occur with high-error pairs in the same query. Thus, it may obscure other important correlations. To address this, after constructing each new MD-Hist, Metis re-estimates the historical queries and refreshes the *CoG*, reducing inflated contributions once the dominant high-error correlations are captured. Then, it makes workload-imperative correlations that still lead to residual errors more prominent in later iterations.

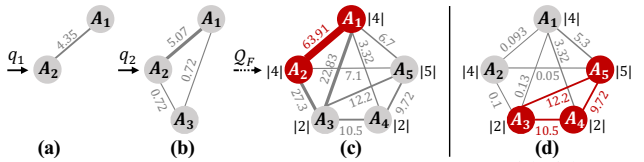


Figure 5: *CoG* construction and subset selection. (a)–(b) show the *CoG* constructed from $\{q_1\}$ and $\{q_1, q_2\}$, respectively; (c) shows the *CoG* over all queries in Q_F. After constructing H over $\mathcal{A}_H = \{A_1, A_2\}$, *CoG* is updated as (d). The label $|\mathcal{B}_{A_i}|$ next to each node is the bucket count of the single-attribute statistic S_i, and red nodes indicate the selected subset.

4.2 Optimal Subset Selection

Given the *CoG*, Metis selects an attribute subset $\mathcal{A}_H$ that captures the strongest workload-imperative correlations under a per-histogram bucket budget L. As each *CoG* edge weight quantifies pairwise imperative correlation, a natural heuristic is to greedily select high-weight edges. However, this strategy only focuses on pairwise correlations and fails to maximize the global correlations.

To address this issue, we formulate the subset selection as a 0–1 integer linear program (ILP) on the *CoG* (see Eq. 2). Specifically, for each attribute node $v_i \in V$, a binary variable x_i indicates whether A_i is selected; for each edge $e_{ij} \in E$, y_{ij} indicates whether the correlation between A_i and A_j is captured. The consistency constraints (Eq. 2b) enforce $y_{ij} = 1$ only if $x_i = x_j = 1$, while the objective (Eq. 2a) maximizes the total selected edge weight. However, maximizing the edge-weight objective alone might select an attribute subset containing attribute pairs that never co-occur in the workload, wasting storage by jointly modeling such attributes. Since each edge e_{ij} in the *CoG* represents the co-occurrence of A_i and A_j in at least one query, we require every pair of selected attributes to be connected in the *CoG*, forming a fully connected subgraph (Eq. 2c). Moreover, the number of buckets for each built MD-Hist, given by $\prod_{A_i \in \mathcal{A}_H} |\mathcal{B}_{A_i}|^{x_i}$, should not exceed the storage limit L (Eq. 2d), where $|\mathcal{B}_{A_i}|^{x_i}$ equals $|\mathcal{B}_{A_i}|$ if A_i is selected (i.e., $x_i = 1$).

$$\max_{\substack{x_i \in \{0,1\}, \forall i \\ y_{ij} \in \{0,1\}, \forall e_{ij} \in E}} \sum_{e_{ij} \in E} w_{ij} y_{ij} \tag{2a}$$

$$\text{s.t.} \quad y_{ij} \le x_i, \; y_{ij} \le x_j, \; y_{ij} \ge x_i + x_j - 1, \; \forall e_{ij} \in E, \tag{2b}$$

$$x_i + x_j \le 1, \quad \forall e_{ij} \notin E, \tag{2c}$$

$$\sum_i x_i \log |\mathcal{B}_{A_i}| \le \log L. \tag{2d}$$

Remark. For each historical query with k predicate attributes, Metis adds all $\binom{k}{2}$ associated edges to *CoG*. Thus, a clique need not originate from a single high-dimensional query: multiple lower-dimensional queries could collectively cover all attribute pairs in a clique. Conversely, a non-clique contains at least one pair that has never co-occurred in the observed workload. Since future query access patterns cannot be known in advance, Metis allocates the limited statistics budget according to observed error reduction rather than unvalidated latent correlations. This strategy is particularly important as an MD-Hist contains the Cartesian product of its per-attribute buckets. Thus, materializing an MD-Hist over an attribute set containing unobserved attribute pairs would substantially increase the storage cost, without clear benefit to the target workload. Note, if we observe that an absent combination repeatedly leads to large estimation errors in future workloads, the corresponding

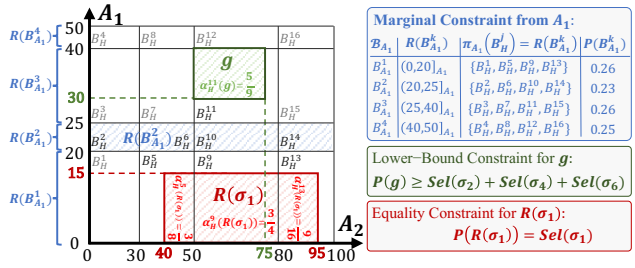


Figure 6: Equality, lower-bound, and marginal constraints derived from $\mathcal{Q}_F$ and $\mathcal{S}$. Here, $R(\sigma_1)$ corresponds to q_1 in Table 1, and g denotes the projected region of queries in Fig. 7.

edges will be created and accumulate error signals in the *CoG*, allowing these attribute subsets to be reconsidered subsequently.

Example 4.1. We demonstrate how METIS (i) constructs the *CoG* from $\mathcal{Q}_F$, (ii) selects an attribute subset $\mathcal{A}_H$ via ILP, and (iii) updates the *CoG* after adding a new MD-Hist H. Table 1 reports the estimated selectivities before and after adding H, while Fig. 5 visualizes the corresponding *CoG*. We set the storage budget for an MD-Hist $L = 20$, with $|\mathcal{B}_{A_1}| = |\mathcal{B}_{A_2}| = 4$, $|\mathcal{B}_{A_3}| = |\mathcal{B}_{A_4}| = 2$, and $|\mathcal{B}_{A_5}| = 5$.

Step 1: Constructing the *CoG* from $\mathcal{Q}_F$. Fig. 5a–b show the *CoG* incrementally constructed from $\{q_1\}$ to $\{q_1, q_2\}$. Using only $\mathcal{S}$, q_2 yields $Q\text{-}error(q_2) = \max(\frac{0.079}{0.025}, \frac{0.025}{0.079}) = 3.16$. As shown in Fig. 5b, Eq. 1 adds $(Q\text{-}error(q_2) - 1)/\binom{3}{2} \approx 0.72$ to each of $w_{1,2}$, $w_{1,3}$, and $w_{2,3}$. Aggregating all queries in $\mathcal{Q}_F$ produces the *CoG* in Fig. 5c.

Step 2: Selecting $\mathcal{A}_H$ via ILP. Based on the *CoG*, we solve the ILP (i.e., Eq. 2) to select $\mathcal{A}_H$. Although selecting $\{A_1, A_2, A_3\}$ attains $w_{1,2} + w_{1,3} + w_{2,3} = 114.04$, it requires $4 \times 4 \times 2 = 32 > L$ buckets and is infeasible. Solving the ILP yields $x_1 = x_2 = 1$ and $x_3 = x_4 = x_5 = 0$. Accordingly, $y_{1,2} = 1$ and all other $y_{ij} = 0$. The selected subset is $\mathcal{A}_H = \{A_1, A_2\}$, achieving an objective value of $w_{1,2} = 63.91$ with $4 \times 4 = 16 \leq L$ buckets, as highlighted in Fig. 5c.

Step 3: Iterative Updates. METIS constructs H over $\{A_1, A_2\}$ as described in §5. Table 1 shows that H reduces errors for queries involving both attributes, e.g., $Q\text{-}error(q_2)$ decreases from 3.16 to 1.013. This indicates that the missing correlation between A_1 and A_2 dominates the error for q_2 without requiring a joint dependency with A_3. METIS then re-estimates $\mathcal{Q}_F$ and refreshes the *CoG* (Fig. 5d).

5 MD-HIST CONSTRUCTION

METIS builds an MD-Hist by inferring its bucket probability masses from feedback $\mathcal{Q}_F$ of both *fully-* and *partially-contained* queries.

5.1 Equality Constraints

Let $\mathcal{Q}^{FC}$ denote the set of fully-contained queries in $\mathcal{Q}_F$. For any query in $\mathcal{Q}^{FC}$, its selectivity can be derived by aggregating the probability mass of the intersections between the query region $R(\sigma)$ and the buckets $\mathcal{B}_H$ within the histogram H, yielding the following *equality constraint*: $\sum_{B_H^i \in \mathcal{B}_H} P(B_H^i) \cdot \alpha_H^i(R(\sigma)) = Sel(\sigma)$, where $P(B_H^i)$ is the probability mass of bucket B_H^i, and $\alpha_H^i(R(\sigma))$ denotes the overlap ratio between $R(\sigma)$ and $R(B_H^i)$.

Example 5.1. Consider the query q_1 (from Table 1) fully-contained by $\mathcal{A}_H = \{A_1, A_2\}$. As indicated by the red shaded region in Fig. 6, the query region $R(\sigma_1)$ intersects the buckets B_H^5, B_H^9, and B_H^{13}, with overlap ratios $\frac{3}{8}$, $\frac{3}{4}$, and $\frac{9}{16}$, respectively. The resulting equality constraint is $P(B_H^5) \cdot \frac{3}{8} + P(B_H^9) \cdot \frac{3}{4} + P(B_H^{13}) \cdot \frac{9}{16} = 0.02$.

q	σ	$Sel(\sigma)$
q_2	$\sigma_{A_1,A_2} \wedge A_3 = 1$	0.079
q_4	$\sigma_{A_1,A_2} \wedge A_3 = 0 \wedge A_5 > 7$	0.023
q_5	$\sigma_{A_1,A_2} \wedge A_5 > 24$	0.072
q_6	$\sigma_{A_1,A_2} \wedge A_3 = 0 \wedge A_5 = 3$	0.010

σ_{A_1,A_2}: $30 < A_1 \leq 40 \wedge 50 < A_2 \leq 75$.

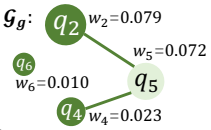


Figure 7: Partially-contained queries $\mathcal{Q}_g^{PC} = \{q_2, q_4, q_5, q_6\}$ correspond to the projected region $g = (30, 40]_{A_1} \times (50, 75]_{A_2}$, as shown by the green shaded region in Fig. 6, together with their query overlap graph $\mathcal{G}_g$. In $\mathcal{G}_g$, vertices represent queries weighted by $Sel(\sigma)$, and an edge connects two queries whose regions overlap. Dark green nodes indicate the disjoint subset $\mathcal{Q}_g^\star$ that achieves the maximum aggregated selectivity.

5.2 Lower-Bound Constraints

Partially-contained queries are prevalent in practice and contain valuable information about bucket probabilities. To exploit them, we project the query region onto the attribute subset $\mathcal{A}_H$, resulting in the projected query region $R^H(\sigma)$ (see Definition 2.2). Since σ is conjunctive, this projection removes query predicates on attributes outside $\mathcal{A}_H$, thereby relaxing the query constraints. That is, the query selectivity $Sel(\sigma)$ is upper-bounded by the probability of its projected region. Thus, each partially-contained query helps induce a lower-bound constraint on $R^H(\sigma)$, i.e., $Sel(\sigma) \leq P(R^H(\sigma))$.

Let $\mathcal{Q}^{PC}$ denote the set of partially-contained queries in $\mathcal{Q}_F$, and let $\mathcal{Q}_g^{PC} \subseteq \mathcal{Q}^{PC}$ denote the subset of these queries that share a common projected region g. A subset $\mathcal{Q}_g' \subseteq \mathcal{Q}_g^{PC}$ is *aggregable* if the query regions of its queries are pairwise disjoint. In this case, the corresponding query regions form disjoint subregions of the projected query region g, and their probability mass is aggregable. Thus, aggregating the selectivities of queries in $\mathcal{Q}_g'$ yields a tighter lower bound on $P(g)$, i.e., $\sum_{q \in \mathcal{Q}_g'} Sel(\sigma) \leq P(g)$, which in turn provides more informative constraints on bucket probabilities. To obtain the tightest lower bound on $P(g)$, we aim to identify an aggregable subset $\mathcal{Q}_g^\star \subseteq \mathcal{Q}_g^{PC}$ that maximizes the total selectivity. Then, the *tightest lower-bound constraint* for the projected query region g is

$$\sum_{q \in \mathcal{Q}_g^\star} Sel(\sigma) \;\leq\; \sum_{B_H^i \in \mathcal{B}_H} P(B_H^i) \cdot \alpha_H^i(g).$$

Here, $\alpha_H^i(g)$ denotes the overlap ratio between g and $R(B_H^i)$. To obtain $\mathcal{Q}_g^\star$, we construct a *query overlap graph (QoG)* $\mathcal{G}_g = (V_g, E_g)$ for each projected query region g. Each vertex $v_i \in V_g$ corresponds to a query $q_i \in \mathcal{Q}_g^{PC}$ and is assigned a weight $w(v_i) = Sel(\sigma_i)$. An edge $(v_i, v_j) \in E_g$ is added if their associated query regions overlap, i.e., $R(\sigma_i) \cap R(\sigma_j) \neq \emptyset$. Selecting $\mathcal{Q}_g^\star$ is equivalent to finding a maximum-weight independent set (MWIS) [13] on $\mathcal{G}_g$. Since MWIS is NP-hard and no efficient approximation algorithm with provable guarantees has been proposed [67], METIS adopts a greedy heuristic that iteratively selects the query with the largest selectivity while maintaining pairwise disjointness with the queries already in $\mathcal{Q}_g^\star$.

Example 5.2. Figure 7 shows the query overlap graph $\mathcal{G}_g$. In this example, q_2 and q_5 are connected in $\mathcal{G}_g$ because their query regions overlap on the predicates $A_3 = 1 \wedge A_5 > 24$. Solving the MWIS problem on $\mathcal{G}_g$ yields the disjoint subset $\mathcal{Q}_g^\star = \{q_2, q_4, q_6\}$ with maximum aggregated selectivity $Sel(\sigma_2) + Sel(\sigma_4) + Sel(\sigma_6) = 0.112$. As shown by the green shaded region in Fig. 6, g overlaps only with the bucket B_H^{11} with an overlap ratio $\frac{5}{9}$. Therefore, the lower-bound constraint for g is $P(B_H^{11}) \cdot \frac{5}{9} \geq 0.112$.

5.3 MD-Hist Construction

In addition to query feedback, we leverage single-attribute statistics $\mathcal{S}$ maintained by traditional DBMSs, which provide marginal distributions. Note, for each MD-Hist H, Metis initializes its buckets as the Cartesian product of single-attribute intervals over attributes in the histogram attribute set $\mathcal{A}_H$. As a result, the projection of each histogram bucket onto any attribute in $\mathcal{A}_H$ coincides with exactly one single-attribute interval. Therefore, we impose marginal constraints such that, for each attribute A_i in $\mathcal{A}_H$ and each single-attribute bucket $B^k_{A_i}$ in $\mathcal{B}_{A_i}$ (i.e., the set of buckets on A_i), the probability mass of $B^k_{A_i}$ equals the total probability mass of all histogram buckets whose projection onto A_i equals $R(B^k_{A_i})$. Formally,

$$\sum\nolimits_{\pi_{A_i}(B^j_H)=R(B^k_{A_i})} P(B^j_H) = P(B^k_{A_i}), \quad \forall A_i \in \mathcal{A}_H,\ \forall B^k_{A_i} \in \mathcal{B}_{A_i}.$$

Example 5.3. As shown in Fig. 6, the single-attribute statistic $S_1 \in \mathcal{S}$ on A_1 contains four buckets. Consider the second bucket $B^2_{A_1}$ highlighted by the blue shaded region. Its region is $R(B^2_{A_1}) = (20, 25]_{A_1}$ and its probability is $P(B^2_{A_1}) = 0.23$. The marginal probability of $B^2_{A_1}$ equals the sum of probabilities of all histogram buckets whose projection onto A_1 is $R(B^2_{A_1})$, which are B^2_H, B^6_H, B^{10}_H, and B^{14}_H. Therefore, $P(B^2_H)+P(B^6_H)+P(B^{10}_H)+P(B^{14}_H) = P(B^2_{A_1}) = 0.23$.

Finally, we integrate the aforementioned *equality* (Eq. 3b), *lower-bound* (Eq. 3c) and *marginal* (Eq. 3d) constraints into a unified constraint set C, and formulate a linear program (LP) over C. To tolerate deviations caused by the uniformity assumption within each bucket, we introduce, for each constraint instance $c_i \in C$, a pair of non-negative slack variables $(\xi^+_{c_i}, \xi^-_{c_i})$. The objective minimizes total slack Ξ (Eq. 3a), yielding an MD-Hist that closely adheres to the constraints. We leverage an off-the-shelf solver [21] to infer the probability masses of histogram buckets by solving the LP:

$$\min_{\{P(B^i_H)\},\,\Xi} \sum_{c_l \in C} (\xi^+_{c_l} + \xi^-_{c_l}), \tag{3a}$$

$$\text{s.t.} \sum\nolimits_{B^i_H \in \mathcal{B}_H} P(B^i_H) \cdot \alpha^i_H(R(\sigma)) + \xi^+_{c_l} - \xi^-_{c_l} = Sel(\sigma),\ \forall q \in Q^{FC}, \tag{3b}$$

$$\sum\nolimits_{B^i_H \in \mathcal{B}_H} P(B^i_H) \cdot \alpha^i_H(g) + \xi^+_{c_l} - \xi^-_{c_l} \geq \sum\nolimits_{q \in Q^\star_g} Sel(\sigma), \forall g \in \{R^H(\sigma) | q \in Q^{PC}\}, \tag{3c}$$

$$\sum\nolimits_{\pi_{A_i}(B^j_H)=R(B^k_{A_i})} P(B^j_H) + \xi^+_{c_l} - \xi^-_{c_l} = P(B^k_{A_i}),\ \forall A_i \in \mathcal{A}_H, \forall B^k_{A_i} \in \mathcal{B}_{A_i}, \tag{3d}$$

$$\sum\nolimits_{B^i_H \in \mathcal{B}_H} P(B^i_H) = 1,\ 0 \leq P(B^i_H) \leq 1,\ \forall B^i_H \in \mathcal{B}_H,\ \ \xi^+_{c_l}, \xi^-_{c_l} \geq 0,\ \forall\, c_l \in C.$$

Compact Bucket Representation. For an MD-Hist H, each bucket can be uniquely identified by an integer index corresponding to its enumeration order in the Cartesian product of the single-attribute intervals for each attribute in $\mathcal{A}_H$. This reduces the space complexity from $O(d \cdot |\mathcal{B}_H|)$ to $O(|\mathcal{B}_H|)$ for a d-dimensional histogram.

6 SELECTIVITY INFERENCE FROM MD-HISTS

This section describes how Metis selects and combines multiple materialized MD-Hists for *CardEst*.

6.1 Shared-Attribute-Driven Selection

Existing methods [6, 44, 68, 69] that reconstruct a high-dimensional joint distribution by combining multiple low-dimensional MD-Hists face an *inherent limitation*: combining local distributions typically relies on conditional-independence assumptions, which may inevitably introduce estimation errors. To mitigate this, we design a *shared-attribute-driven selection strategy*, where the *shared attributes* of a candidate MD-Hist are defined as the attributes it shares with those already covered by the selected MD-Hists. These attributes provide the conditioning context to incorporate the candidate's local joint distribution. A larger overlap provides a richer conditioning context and leaves fewer residual dependencies unmodeled, thereby reducing estimation bias. When a query arrives, we yield an ordered MD-Hist list $\mathbf{H}_{\text{sel}} = [H^{(1)}, \ldots, H^{(k)}]$. Specifically, we first select the MD-Hist covering the largest number of predicate attributes, and then iteratively select an MD-Hist that covers at least one uncovered predicate attribute while maximizing overlap with the attributes already covered. This process continues until all predicate attributes are covered. Note that the order of MD-Hists in $\mathbf{H}_{\text{sel}}$ affects the estimation, as the shared attributes at each step depend on the previously selected MD-Hists. To ensure deterministic estimates for identical queries under the same MD-Hist library, we break ties between candidates with the same overlap size according to their construction order.

Remark. Maximizing overlap reduces but does not eliminate conditional independence errors. Metis further mitigates them in two ways. First, without a global decomposability constraint, it constructs MD-Hists for high-utility attribute subsets within the statistics budget, directly capturing important dependencies. Second, after constructing each MD-Hist, it re-estimates historical queries and refreshes the *CoG*, thereby prioritizing attribute combinations whose residual dependencies still cause large errors.

6.2 Factor Graph-Based Inference

Since $\mathbf{H}_{\text{sel}}$ covers the predicate-attribute set $\mathcal{A}_\sigma$, the query selectivity can be estimated from the reconstructed joint distribution over the *attribute union* $\mathcal{A}_{\text{U}} = \bigcup_{H^{(i)} \in \mathbf{H}_{\text{sel}}} \mathcal{A}_{H^{(i)}}$. Reconstructing such a joint distribution from multiple local joint distributions is a classical problem in probabilistic modeling [37, 40]. Factor graphs [39] provide a standard formulation by representing a global distribution as the product of local factors over shared variables. Following this formulation, Metis treats each selected histogram $H^{(i)} \in \mathbf{H}_{\text{sel}}$ as a local factor and combines them sequentially.

Formally, let $\mathcal{B}_{\text{U}} = \prod_{A_i \in \mathcal{A}_{\text{U}}} \mathcal{B}_{A_i}$ denote the set of multi-dimensional buckets over the attribute union $\mathcal{A}_{\text{U}}$, where $\mathcal{B}_{A_i}$ is the set of one-dimensional buckets for attribute A_i. Since each histogram $H^{(i)}$ models the joint distribution over its attribute set $\mathcal{A}_{H^{(i)}}$, for any multi-dimensional bucket $\mathbf{g} \in \mathcal{B}_{\text{U}}$, $H^{(i)}$ provides a local factor $P_{H^{(i)}}\big(\pi_{\mathcal{A}_{H^{(i)}}}(\mathbf{g})\big)$. Here, $\pi_{\mathcal{A}_{H^{(i)}}}(\mathbf{g})$ denotes the projection of $\mathbf{g}$ onto $\mathcal{A}_{H^{(i)}}$, and $P_{H^{(i)}}(\cdot)$ denotes the probability mass of the corresponding region in $H^{(i)}$. However, for each histogram $H^{(i)}$ selected after the first ($i \geq 2$), its local factor cannot be incorporated directly, since the probability mass over the attributes it shares with previously incorporated histograms has already been taken into account. For each such $H^{(i)}$, let $O_i = \mathcal{A}_{H^{(i)}} \cap \bigcup_{j<i} \mathcal{A}_{H^{(j)}}$ denote the attributes shared with the previously incorporated histograms. To avoid repeatedly counting the probability mass of O_i, Metis incorporates $H^{(i)}$ through the conditional factor $\frac{P_{H^{(i)}}\left(\pi_{\mathcal{A}_{H^{(i)}}}(\mathbf{g})\right)}{P_{H^{(i)}}\left(\pi_{O_i}(\mathbf{g})\right)}$, where the denominator is the marginal probability of $H^{(i)}$ on O_i, evaluated at $\pi_{O_i}(\mathbf{g})$. This factor gives the contribution of $H^{(i)}$ conditioned on O_i. Combining it with the previously incorporated factors assumes that, conditioned on O_i, the attributes in $\mathcal{A}_{H^{(i)}} \setminus O_i$ are independent of the

previously incorporated attributes outside O_i. The resulting factorization is exact when this assumption holds and otherwise provides an approximation. Accordingly, the reconstructed probability mass of a bucket $\mathbf{g} \in \mathcal{B}_U$ is $P(\mathbf{g}) = P_{H^{(1)}}(\pi_{\mathcal{A}_{H^{(1)}}}(\mathbf{g})) \cdot \prod_{i=2}^{k} \frac{P_{H^{(i)}}(\pi_{\mathcal{A}_{H^{(i)}}}(\mathbf{g}))}{P_{H^{(i)}}(\pi_{O_i}(\mathbf{g}))}$.

Since $\mathcal{A}_\sigma \subseteq \mathcal{A}_U$, the selectivity estimation reduces to the same case considered in §2.3, where the predicate attributes are fully covered. Thus, $\widehat{Sel}(\sigma) = \sum_{\mathbf{g} \in \mathcal{B}_U} P(\mathbf{g}) \cdot \alpha_{\mathbf{g}}(R(\sigma))$, where $\alpha_{\mathbf{g}}(R(\sigma))$ denotes the overlap ratio between $\mathbf{g}$ and the query region $R(\sigma)$.

6.3 Inference Efficiency Optimization

Eliminating Query-Irrelevant Attributes. The selectivity of σ depends only on the predicates $\mathcal{A}_\sigma$ and attributes required for the *alignment* of overlapping histograms. Let $\mathcal{A}_C = \bigcup_{1 \le i < j \le k} (\mathcal{A}_{H^{(i)}} \cap \mathcal{A}_{H^{(j)}})$ denote the set of attributes shared among the selected histograms, and define the necessary attribute set as $\mathcal{A}_N = \mathcal{A}_\sigma \cup \mathcal{A}_C$. All attributes $\mathcal{A}_U \setminus \mathcal{A}_N$ can be marginalized out in advance. Then, the complexity is reduced from $O\left(\prod_{A \in \mathcal{A}_U} |\mathcal{B}_A|\right)$ to $O\left(\prod_{A \in \mathcal{A}_N} |\mathcal{B}_A|\right)$.

Merging Joint Buckets. A key observation is that joint buckets having the same overlap ratio with respect to the query predicate would contribute to $\widehat{Sel}(\sigma)$ with the same multiplicative factor. That is, the only difference lies in their associated probability mass. Formally, consider any two joint buckets $\mathbf{g}, \mathbf{g}' \in \mathcal{B}_N$, where $\mathcal{B}_N = \prod_{A \in \mathcal{A}_N} \mathcal{B}_A$. If $\alpha_{\mathbf{g}}(R(\sigma)) = \alpha_{\mathbf{g}'}(R(\sigma))$, then their contributions to the selectivity estimation differ only in $P(\mathbf{g})$ and $P(\mathbf{g}')$. This helps induce an equivalence relation over $\mathcal{B}_N$, where each equivalence class Γ consists of joint buckets sharing a common overlap ratio α_Γ. Furthermore, the selectivity estimation can be rewritten as $\sum_{\mathbf{g} \in \mathcal{B}_N} P(\mathbf{g}) \cdot \alpha_{\mathbf{g}}(R(\sigma)) = \sum_\Gamma \alpha_\Gamma \cdot \sum_{\mathbf{g} \in \Gamma} P(\mathbf{g})$, which aggregates buckets by overlap ratio. Note, bucket partitions on shared attributes $A \in \mathcal{A}_C$ are preserved to maintain factor alignment. Then, for each attribute $A \in \mathcal{A}_\sigma \setminus \mathcal{A}_C$, assume that this merging reduces the number of buckets from $|\mathcal{B}_A|$ to $|\widetilde{\mathcal{B}}_A|$, the enumeration complexity is reduced from $O(\prod_{A \in \mathcal{A}_N} |\mathcal{B}_A|)$ to $O\left(\prod_{A \in \mathcal{A}_\sigma \setminus \mathcal{A}_C} |\widetilde{\mathcal{B}}_A| \cdot \prod_{A \in \mathcal{A}_C} |\mathcal{B}_A|\right)$.

7 EXPERIMENTAL EVALUATION

In this section, we aim to answer the following questions. **RQ1:** How does Metis perform compared with the state-of-the-art methods (§7.2)? **RQ2:** How do the key designs (§4,5,6) contribute to its overall performance improvements (§7.3)? **RQ3:** Does Metis demonstrate strong practical applicability in terms of scalability, robustness, and improvements in end-to-end query performance (§7.4)?

7.1 Experimental Setup

Datasets. We use three real-world datasets with diverse data characteristics, shown in Table 2, which are widely adopted in prior work on single-table *CardEst* [28, 54, 59, 71, 72]. Note, *Instacart* [22] consists of three tables. For single-table *CardEst*, we join them through their primary–foreign key (PK–FK) relationships to construct a large-scale wide table with non-trivial attribute correlations.

Workloads. Following Hilprecht et al. [27], for each dataset, we define the query-template universe as all 2–5-attribute combinations. For each template, we generate 10^2 pilot queries and use their average *Q-error* under PostgreSQL single-column statistics as its correlation score, where a larger score suggests stronger uncaptured inter-attribute correlations. Then, we sort the templates by this score and divide them into four quartile groups, i.e., the top G_T, upper-middle G_{UM}, lower-middle G_{LM}, and bottom G_B. Table 3 defines four workload scenarios. Specifically, W_0 uniformly accesses all templates, whereas W_1–W_3 follow different skewed access patterns. For W_1–W_3, we concatenate the four quartile groups according to the group orders shown in Table 3 and assign Zipfian access probabilities with $\alpha = 0.75$ over the resulting template sequence, where templates appearing earlier in the sequence are assigned higher access frequencies. Consequently, W_1, W_2, and W_3 are dominated by high-, low-, and medium-correlation templates, respectively. Moreover, following prior work [59, 71, 76], we instantiate equality or range predicates over each attribute in the sampled template. Finally, for each workload scenario, we generate 10^5 training queries and $5 \cdot 10^3$ unseen test queries.

Table 2: Dataset characteristics. "Cols/Cat" is the number of (categorical) attributes; "Domain" is the product of NDVs.

Dataset	Size(MB)	Rows	Cols/Cat	NDV	Domain
Census [7]	4.8	49K	14/8	2~10^2	10^{17}
DMV [51]	685.4	11.6M	11/10	2~10^3	10^{16}
Instacart [22]	2130.9	33.8M	10/4	2~10^5	10^{17}

Table 3: Workload scenarios with different access skews.

Scenario	Group Order	Access Pattern
W_0	N/A	Uniform access over all templates
W_1	G_T, G_{UM}, G_{LM}, G_B	Dominated by high-score templates
W_2	G_B, G_{LM}, G_{UM}, G_T	Dominated by low-score templates
W_3	G_{UM}, G_{LM}, G_T, G_B	Dominated by medium-score templates

Baselines. We compare Metis with representative *CardEst* methods in prior evaluation [58, 71] or conceptually related to Metis. Unless otherwise stated, we follow their reported configurations.

Traditional Estimators. 1) *PG* denotes PostgreSQL's native estimator [18], which uses single-column statistics and the attribute value independence (AVI) assumption. *PG-Ext-CORDS* and *PG-Ext-Metis* additionally build PG extended statistics [16] over user-defined attribute subsets, including functional dependencies, multivariate N-distinct counts, and most-common-value (MCV) lists. Both use *CoG* for attribute-subset selection but differ in edge weights: *PG-Ext-CORDS* uses mean-square contingency scores computed by CORDS [32] to capture pairwise data correlations, whereas *PG-Ext-Metis* uses historical query errors aggregated by Eq. 1, representing data-correlation-driven and workload-feedback-driven selections. For all three PostgreSQL-based estimators, we set the statistics target to its maximum value of 10^4 to improve estimation accuracy. 2) *MHIST* [56] constructs a single MD-Hist over all attributes. Following Wang et al. [71], we use the MaxDiff-2 configuration and limit its size to approximately 1.5% of the data size. 3) *QuickSel* [54] approximates the data distribution using a uniform mixture model and reportedly outperforms earlier feedback-driven histograms, including *STHoles* [2] and *ISOMER* [65].

Probabilistic Graphical Model-Based Estimators. 1) *SASH* [44] leverages workload errors to search over junction-tree structures and bucket allocations. We set its storage budget to 1.5% of the dataset size and its training budget to 12 hours, while all other baselines complete training within this budget. 2) *CLTree* [69] stores all pairwise marginals and constructs a Chow–Liu tree over each query's predicate attributes during inference. 3) *BayesCard* [74] learns a Bayesian network from the underlying data.

Table 4: *Q-error* of different *CardEst* methods across four workload scenarios.

Dataset	Methods	W_0				W_1				W_2				W_3			
		50^{th}	95^{th}	99^{th}	99.9^{th}	50^{th}	95^{th}	99^{th}	99.9^{th}	50^{th}	95^{th}	99^{th}	99.9^{th}	50^{th}	95^{th}	99^{th}	99.9^{th}
Census	PG	1.50	6.41	20.00	$2.3\cdot10^3$	1.72	11.51	$5.0\cdot10^2$	$6.4\cdot10^3$	1.21	3.85	10.92	58.27	1.56	7.91	22.50	$1.6\cdot10^2$
	PG-Ext-CORDS	1.18	4.00	9.00	54.67	1.12	4.50	14.67	67.84	1.09	3.00	5.86	15.39	1.21	5.00	13.09	53.53
	PG-Ext-Metis	1.17	3.57	7.50	29.53	1.12	3.67	8.27	24.00	1.07	2.80	5.34	12.50	1.21	4.00	10.50	44.33
	MHIST	6.01	$2.0\cdot10^2$	$8.4\cdot10^2$	$2.9\cdot10^3$	5.00	$1.5\cdot10^2$	$6.1\cdot10^2$	$2.2\cdot10^3$	7.00	$2.4\cdot10^2$	$7.4\cdot10^2$	$2.4\cdot10^3$	6.04	$1.7\cdot10^2$	$7.5\cdot10^2$	$3.8\cdot10^3$
	QuickSel	5.24	$1.9\cdot10^2$	$7.0\cdot10^2$	$2.1\cdot10^3$	5.00	$1.7\cdot10^2$	$6.0\cdot10^2$	$1.5\cdot10^3$	6.00	$3.1\cdot10^2$	$1.3\cdot10^3$	$3.3\cdot10^3$	5.28	$1.6\cdot10^2$	$5.4\cdot10^2$	$1.6\cdot10^3$
	SASH	7.19	$1.9\cdot10^2$	$7.5\cdot10^2$	$2.1\cdot10^3$	6.56	$2.0\cdot10^2$	$1.1\cdot10^3$	$3.5\cdot10^3$	4.50	$1.3\cdot10^2$	$5.1\cdot10^2$	$1.4\cdot10^3$	6.24	$1.7\cdot10^2$	$6.3\cdot10^2$	$1.6\cdot10^3$
	CLTree	1.16	3.31	10.71	48.90	1.16	3.34	10.15	$1.5\cdot10^2$	1.08	3.31	14.51	$2.9\cdot10^2$	1.20	3.65	14.14	$2.8\cdot10^2$
	BayesCard	1.24	3.30	6.37	17.52	1.25	3.43	6.67	13.48	1.16	2.70	5.02	28.51	1.27	3.56	6.33	20.92
	MSCN	1.33	3.99	8.00	24.48	1.25	4.00	7.28	17.67	1.24	4.00	7.00	19.00	1.35	4.00	8.00	25.15
	LW-NN	1.29	3.33	6.67	16.00	1.27	3.00	5.42	10.00	1.18	3.00	5.00	13.00	1.29	3.00	6.33	15.67
	Naru	1.19	2.75	5.86	30.00	1.21	3.00	15.68	31.00	1.16	2.67	5.00	10.00	1.20	2.80	5.00	12.00
	DeepDB	1.25	3.42	7.45	$7.0\cdot10^2$	1.28	4.24	$2.3\cdot10^2$	$1.5\cdot10^3$	1.15	2.84	5.36	12.96	1.27	3.56	6.30	59.35
	UAE	1.25	4.00	13.00	51.02	1.29	5.00	37.33	$1.1\cdot10^2$	1.20	3.67	12.50	39.00	1.29	4.00	9.50	48.00
	CoLSE	1.36	6.00	57.00	$7.7\cdot10^2$	1.33	7.21	$2.9\cdot10^2$	$9.6\cdot10^2$	1.20	4.08	15.01	$3.5\cdot10^2$	1.35	6.00	51.01	$8.7\cdot10^2$
	Metis	**1.10**	**2.00**	**3.00**	**5.50**	**1.03**	**1.74**	**2.86**	**5.08**	**1.04**	**1.67**	**2.57**	**6.00**	**1.11**	**2.00**	**3.00**	**6.70**
DMV	PG	1.51	50.42	$5.4\cdot10^2$	$4.2\cdot10^4$	1.85	$1.4\cdot10^2$	$2.7\cdot10^4$	$2.0\cdot10^5$	1.32	25.89	$2.2\cdot10^2$	$2.7\cdot10^4$	1.61	66.00	$1.2\cdot10^3$	$3.8\cdot10^4$
	PG-Ext-CORDS	1.36	32.69	$3.8\cdot10^2$	$8.6\cdot10^4$	1.56	79.48	$8.5\cdot10^3$	$1.1\cdot10^5$	1.24	18.50	$2.1\cdot10^2$	$1.6\cdot10^4$	1.42	36.87	$5.9\cdot10^2$	$1.1\cdot10^4$
	PG-Ext-Metis	1.38	26.34	$3.3\cdot10^2$	$8.5\cdot10^3$	1.40	48.75	$6.1\cdot10^2$	$2.7\cdot10^4$	1.24	17.45	$1.9\cdot10^2$	$4.4\cdot10^3$	1.40	35.58	$5.0\cdot10^2$	$5.3\cdot10^3$
	MHIST	1.17	$7.3\cdot10^2$	$1.1\cdot10^4$	$1.3\cdot10^5$	1.07	$2.9\cdot10^2$	$9.0\cdot10^3$	$1.2\cdot10^5$	1.08	$4.6\cdot10^2$	$1.1\cdot10^4$	$1.5\cdot10^5$	1.37	$7.0\cdot10^2$	$1.2\cdot10^4$	$1.4\cdot10^5$
	QuickSel	37.19	$4.4\cdot10^4$	$1.9\cdot10^5$	$4.5\cdot10^5$	47.16	$4.4\cdot10^4$	$1.9\cdot10^5$	$3.8\cdot10^5$	30.47	$5.0\cdot10^4$	$2.1\cdot10^5$	$4.5\cdot10^5$	38.22	$4.4\cdot10^4$	$1.9\cdot10^5$	$4.4\cdot10^5$
	SASH	17.71	$4.3\cdot10^3$	$2.5\cdot10^4$	$1.6\cdot10^5$	28.54	$2.1\cdot10^4$	$1.9\cdot10^5$	$1.1\cdot10^6$	15.04	$4.9\cdot10^3$	$3.3\cdot10^4$	$1.3\cdot10^5$	30.35	$1.0\cdot10^4$	$5.4\cdot10^4$	$1.7\cdot10^5$
	CLTree	1.04	6.99	$1.2\cdot10^2$	$2.5\cdot10^3$	1.04	8.59	$1.8\cdot10^2$	$2.2\cdot10^3$	1.02	6.62	91.71	$2.2\cdot10^3$	1.04	7.00	90.02	$7.7\cdot10^2$
	BayesCard	1.08	5.13	24.84	$5.2\cdot10^2$	1.08	8.77	$4.1\cdot10^2$	$2.2\cdot10^3$	1.05	4.81	15.86	$4.2\cdot10^2$	1.08	5.63	18.13	$4.2\cdot10^2$
	MSCN	1.50	17.00	69.02	$2.4\cdot10^2$	1.50	16.97	50.63	$2.1\cdot10^2$	1.41	14.87	58.00	$3.4\cdot10^2$	1.50	15.54	52.49	$2.2\cdot10^2$
	LW-NN	1.60	8.61	35.00	$3.2\cdot10^2$	1.43	8.58	37.00	$3.7\cdot10^2$	1.39	10.26	41.10	$7.2\cdot10^2$	1.50	10.50	66.02	$1.6\cdot10^3$
	Naru	1.20	3.57	9.00	55.01	1.22	4.00	11.50	$1.0\cdot10^2$	1.16	3.50	11.17	79.01	1.20	4.00	11.32	$1.1\cdot10^2$
	DeepDB	1.25	27.97	$3.7\cdot10^2$	$2.6\cdot10^4$	1.39	54.00	$8.0\cdot10^3$	$5.7\cdot10^4$	1.17	15.70	$1.9\cdot10^2$	$8.1\cdot10^3$	1.25	29.88	$8.5\cdot10^2$	$7.9\cdot10^3$
	UAE	1.12	3.09	9.00	27.61	1.15	3.90	9.00	65.52	1.09	3.00	7.15	28.50	1.13	3.50	9.25	30.00
	CoLSE	1.30	49.00	$5.5\cdot10^3$	$1.2\cdot10^5$	1.33	69.04	$7.1\cdot10^3$	$5.5\cdot10^4$	1.20	21.00	$1.2\cdot10^3$	$1.1\cdot10^5$	1.39	57.72	$9.5\cdot10^3$	$2.2\cdot10^5$
	Metis	**1.03**	**2.16**	**2.63**	**9.00**	**1.01**	**2.00**	**3.50**	**7.83**	**1.02**	**2.00**	**2.28**	**9.67**	1.06	**2.40**	**3.00**	**8.33**
Instacart	PG	1.24	42.34	$1.5\cdot10^2$	$6.6\cdot10^2$	1.67	93.43	$4.0\cdot10^2$	$1.1\cdot10^3$	1.08	11.43	65.80	$3.5\cdot10^2$	1.22	20.27	90.34	$3.7\cdot10^2$
	PG-Ext-CORDS	1.14	3.60	29.89	$2.7\cdot10^2$	1.14	10.20	56.00	$2.9\cdot10^2$	1.07	2.51	12.19	$1.1\cdot10^2$	1.16	3.34	24.77	$1.7\cdot10^2$
	PG-Ext-Metis	1.11	2.62	10.00	56.22	**1.11**	2.45	9.00	22.00	1.06	2.12	5.00	52.43	1.13	2.71	17.27	97.83
	MHIST	3.06	$3.1\cdot10^3$	$5.5\cdot10^4$	$5.1\cdot10^5$	2.85	$7.4\cdot10^3$	$1.6\cdot10^5$	$6.3\cdot10^5$	1.80	$6.4\cdot10^2$	$3.3\cdot10^4$	$4.0\cdot10^5$	4.33	$5.2\cdot10^3$	$7.5\cdot10^4$	$4.9\cdot10^5$
	QuickSel	8.10	$1.6\cdot10^4$	$8.0\cdot10^4$	$1.6\cdot10^5$	9.00	$2.1\cdot10^4$	$1.3\cdot10^5$	$2.7\cdot10^5$	2.72	$1.9\cdot10^4$	$1.1\cdot10^5$	$3.9\cdot10^5$	8.39	$1.4\cdot10^4$	$8.4\cdot10^4$	$3.0\cdot10^5$
	SASH	3.76	98.54	$3.5\cdot10^2$	$1.9\cdot10^3$	3.87	51.76	$1.7\cdot10^2$	$4.5\cdot10^2$	2.17	99.07	$9.9\cdot10^2$	$3.9\cdot10^3$	4.09	$2.0\cdot10^2$	$9.3\cdot10^2$	$9.7\cdot10^3$
	CLTree	1.12	3.40	13.03	38.93	1.36	3.51	11.35	36.43	**1.03**	2.71	9.86	47.54	**1.09**	3.01	10.93	37.48
	BayesCard	1.09	3.42	11.22	23.41	1.14	7.14	16.43	42.40	1.05	2.10	6.14	16.94	**1.09**	2.64	8.27	22.63
	MSCN	1.41	6.08	13.40	32.97	1.45	6.24	12.43	33.63	1.24	4.91	10.60	21.80	1.33	5.05	11.55	42.36
	LW-NN	1.17	2.11	4.46	10.83	1.19	2.44	5.00	16.00	1.15	2.00	3.80	8.00	1.18	2.28	4.15	13.33
	Naru	1.16	3.00	10.41	$2.1\cdot10^3$	1.20	3.16	11.00	$6.5\cdot10^2$	1.09	2.50	7.19	$5.2\cdot10^3$	1.14	3.00	9.00	$4.0\cdot10^3$
	DeepDB	1.16	10.06	33.40	72.62	1.46	25.50	55.10	$2.5\cdot10^2$	1.07	3.89	20.41	66.38	1.14	5.54	23.13	71.79
	UAE	1.12	2.50	8.00	32.80	1.14	3.04	9.33	31.00	1.08	2.02	4.64	14.50	1.11	2.62	6.00	26.94
	CoLSE	1.16	3.43	64.02	$6.8\cdot10^3$	1.22	3.85	31.00	$1.5\cdot10^4$	1.07	2.31	11.00	$2.4\cdot10^3$	1.14	3.72	$1.1\cdot10^2$	$1.5\cdot10^4$
	Metis	**1.07**	**1.71**	**3.00**	**9.55**	**1.11**	**2.01**	**3.90**	**14.67**	1.04	**1.61**	**2.67**	**4.28**	1.11	**2.00**	**4.00**	**11.50**

Learned Estimators. 1) *MSCN* [36] is a query-driven estimator based on multi-set convolutional networks that encode query predicates and bitmaps over sampled tuples. 2) *LW-NN* [8] is a lightweight query-driven neural estimator that augments predicate encodings with handcrafted statistical features, such as AVI estimates. 3) *Naru* [76] is a data-driven autoregressive estimator that models the data distribution and performs inference via progressive sampling. 4) *DeepDB* [27] is a data-driven estimator that models the data distribution using relational sum-product networks. 5) *UAE* [72] is a hybrid autoregressive estimator that extends *Naru* with supervision from query workloads. 6) *CoLSE* [59] is a hybrid estimator that combines a copula-based model learned from data with a lightweight neural network trained on query workloads to correct residual estimation errors. Note, recent learned methods [43, 73, 75] for multi-table and dynamic scenarios generally extend state-of-the-art single-table estimators while preserving the core single-table estimation methodology. Thus, the baselines we select remain state-of-the-art and representative.

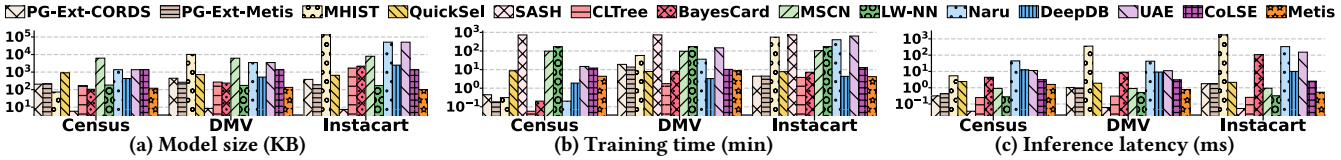
Real-World Application Scenario. We use a real-world entertainment ticketing application workload. With the assistance of OceanBase engineers, we conducted experiments in their internal environment using privacy-preserving techniques. We extract query accesses over three main tables with 16, 13, and 11 attributes. The three tables contain approximately 100K, 10M, and 30M tuples, with corresponding domain sizes of 4×10^{21}, 9×10^{19}, and 9×10^{16}, respectively. Next, we extract $1.05\cdot10^5$ queries from the workload, with 10^5 and $5\cdot10^3$ used for training and testing. These queries cover 300 query templates, among which 5-predicate templates are the most frequent, accounting for 38% of all queries. We also observe clear access skewness in the workload, with approximately 39.3% of hot templates accounting for 67.3% of query accesses.

Table 5: Comparison on the real-world application scenario.

Methods	*Q-error* 50^{th}	95^{th}	99^{th}	99.9^{th}	Size (KB)	Training Time (min)	Latency (ms)
PG	2.44	$1.9\cdot10^2$	$1.6\cdot10^4$	$1.1\cdot10^5$	–	**1.05**	0.38
PG-Ext-CORDS	1.21	41.54	$9.2\cdot10^2$	$8.7\cdot10^4$	1027.6	22.93	0.53
PG-Ext-Metis	1.17	44.33	$7.5\cdot10^2$	$9.0\cdot10^3$	836.5	16.07	0.62
MHIST	1.27	$4.6\cdot10^2$	$3.8\cdot10^4$	$4.6\cdot10^5$	$1.7\cdot10^5$	775.5	835.6
QuickSel	20.05	$4.2\cdot10^4$	$1.6\cdot10^5$	$3.5\cdot10^5$	2281	8.95	2.1
SASH	12.55	$5.7\cdot10^3$	$4.0\cdot10^4$	$2.9\cdot10^5$	**38.2**	2180	**0.07**
CLTree	1.12	5.14	57.10	$4.1\cdot10^2$	2147.8	5.57	0.3
BayesCard	1.16	12.08	$1.9\cdot10^2$	$1.9\cdot10^3$	2513	124.6	42.92
MSCN	1.49	8.38	29.50	$1.3\cdot10^2$	$2.0\cdot10^4$	99.02	0.97
LW-NN	1.36	5.18	27.00	$2.4\cdot10^2$	528	162.7	0.39
Naru	1.25	4.00	15.00	80.03	$5.5\cdot10^4$	397.7	326.2
DeepDB	1.68	46.45	$6.2\cdot10^3$	$4.0\cdot10^4$	3224	9.34	10.69
UAE	1.16	3.76	19.00	$1.1\cdot10^2$	$5.5\cdot10^4$	679.1	65.91
CoLSE	1.33	32.02	$7.7\cdot10^3$	$2.2\cdot10^4$	4050	11.21	3.26
Metis	**1.01**	**2.00**	**4.01**	**19.01**	198.6	7.09	0.37

Metrics. 1) *Q-error*, as defined in §2.1, is a widely adopted metric to measure the estimation accuracy; 2) *End-to-End (E2E) Time* measures the total time to generate a query plan using estimated cardinalities and execute the physical plan [25].

Experimental Settings. We construct single-attribute statistics $\mathcal{S}$ from PostgreSQL's defaults with up to 10^2 most common values (MCVs) and 10^2 equi-depth buckets per attribute. The bucket budget for each MD-Hist is $L=2\times10^4$, determined via experimental tuning, and construction stops when the relative reduction in *Q-error* on the training queries falls below 1% for two consecutive rounds. Note, all experiments are conducted on a server with two Intel(R) Xeon(R) Gold 6240R CPUs; learned estimators that benefit from GPU acceleration are trained on an NVIDIA RTX 6000 Ada GPU.



(a) Model size (KB) (b) Training time (min) (c) Inference latency (ms)

Figure 8: Model size, training time, and inference latency of different *CardEst* estimators.

7.2 Comparison with State-of-the-Art

Estimation Accuracy. Table 4 reports the *Q-error* results under the four workload scenarios. Metis achieves the lowest or near-lowest errors across all datasets and workloads, especially on tail errors. This is because Metis always effectively identifies high-utility attribute combinations despite workload-dependent correlations. Traditional estimators exhibit substantially higher *Q-error*. AVI used in PG ignores inter-attribute correlations, while *MHIST* and *QuickSel* are limited by a global partitioning or a small set of subpopulations, both failing to capture sparse high-dimensional distributions. *PG-Ext-CORDS* and *PG-Ext-Metis* use the same *PG extended statistics* but different attribute-selection strategies. The latter's lower tail errors indicate the benefit of feedback-driven selection, although both provide only partial joint-distribution coverage. *SASH* also uses feedback, but its expensive candidate evaluation limits exploration under a fixed training budget. *MSCN* and *LW-NN* directly learn an opaque query-to-cardinality mapping and show similar performance across different workload scenarios. For example, under each dataset, the 99.9^{th} *Q-error* of *MSCN* remains within the same order of magnitude across W_0–W_3 scenarios. However, they are still limited by the coverage of the training workload and might produce unreliable estimates for underrepresented queries. Workload-agnostic methods such as *BayesCard*, *Naru*, and *DeepDB* sometimes achieve competitive median errors but show greater tail variation. For example, the 99.9^{th} error of *DeepDB* under W_1 is 3.8–115.7× that under W_2, since such methods do not prioritize workload-imperative correlations. Among hybrid methods, *UAE* is competitive but costly to train, whereas *CoLSE* suffers large tail errors since CDF factorization can amplify estimation errors.

Model Size. Fig. 8a reports the average model size across the four workload scenarios, where Metis remains below 150 KB on all datasets while achieving competitive accuracy. On the large-scale Instacart dataset, all other methods except *SASH* are 1.7–1307.5× larger than that of Metis. This compactness comes from using integer bucket identifiers instead of explicit multi-attribute ranges.

Training Time. As shown in Fig. 8b, Metis completes histogram construction within 4.2–8.6 minutes, with each MD-Hist taking approximately 42 seconds to construct on average. However, on the small Census dataset, Metis incurs slightly longer training time than several methods, including *MHIST*, *CLTree*, *BayesCard*, and *Naru*. This is because Metis incrementally constructs workload-aware MD-Hists using query feedback, whereas these methods build their models directly from the underlying data and thus benefit from the small dataset size. As the dataset size grows, however, these methods require substantially longer training time, whereas Metis maintains relatively stable training time across datasets.

Inference Latency. As shown in Fig. 8c, Metis consistently achieves millisecond-level inference latency across all datasets. Metis is slightly slower than lightweight estimators such as *CLTree* and *LW-NN*. This is because Metis dynamically selects and combines multiple histograms during inference to capture complex attribute correlations, achieving a better trade-off between estimation accuracy and inference efficiency. In contrast, *CLTree* models only pairwise attribute dependencies, while *LW-NN* adopts a lightweight neural network architecture. These lightweight designs reduce inference latency but might compromise estimation accuracy. Moreover, compared with data-driven and hybrid learned estimators, Metis achieves approximately 7.1–627.6× lower inference latency.

Real-World Application Scenario. Table 5 reports the results on the real-world workload. We observe that Metis still achieves the lowest *Q-error* across all percentiles, together with a compact statistics footprint and low training and inference overhead. This advantage mainly stems from the highly skewed query accesses observed in the workload, which allow Metis to accumulate stronger feedback for frequently accessed, high-error attribute combinations and prioritize the corresponding high-utility MD-Hists.

Underestimation and Overestimation. For each test query q, we compute the estimated-to-true cardinality ratio, where values below/above 1 indicate underestimation/overestimation. Fig. 9 shows that Metis consistently yields lower errors in both directions, as its *CoG* is built upon historical *Q-error* values, which measure error magnitude symmetrically. *PG* substantially underestimates as its AVI ignores correlations. *PG-Ext-CORDS* and *PG-Ext-Metis* partially mitigate this problem, but their extended statistics offer only partial coverage of the joint distribution. *MHIST*, *QuickSel*, and *SASH* suffer larger errors in both directions due to global partitioning, limited subpopulations, or expensive candidate histogram-set evaluation. *MSCN*, *BayesCard*, *Naru*, *DeepDB*, and *UAE* rely on sampling during model construction or query inference, but finite samples might underrepresent sparse and strongly correlated data regions. *CLTree* and *CoLSE* mainly rely on pairwise statistics, which might substantially underestimate the cardinalities of queries over dense regions with higher-order correlations. *LW-NN* incorporates data-distribution features into its query encoding, but some of these

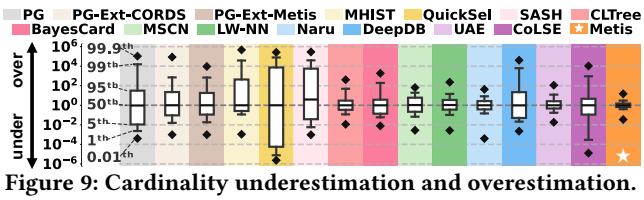


Figure 9: Cardinality underestimation and overestimation.

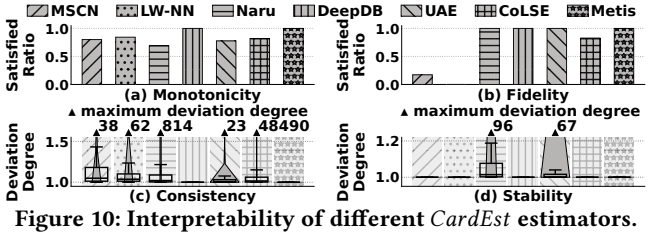


Figure 10: Interpretability of different *CardEst* estimators.

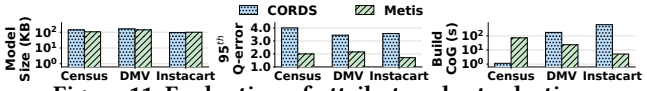


Figure 11: Evaluation of attribute subset selection.

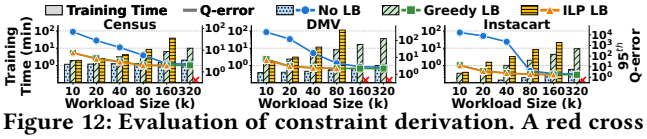


Figure 12: Evaluation of constraint derivation. A red cross (×) indicates that MD-Hist construction exceeds 2 hours.

features are computed under the AVI assumption and might lead to underestimation when attributes are correlated.

Interpretability. We evaluate *Monotonicity*, *Consistency*, *Stability*, and *Fidelity* (see §1) for Metis and competitive learned estimators. *Monotonicity* and *Fidelity* are measured by the Satisfied Ratio; *Consistency* and *Stability* are measured by the Deviation Degree, where larger values indicate more severe violations. Specifically, *Monotonicity* requires tighter predicates to have no larger estimates. For predicate pairs $\mathcal{P}$ in which σ_2 is more restrictive than σ_1, its Satisfied Ratio is $\frac{1}{|\mathcal{P}|}\sum_{(\sigma_1,\sigma_2)\in\mathcal{P}}\mathbb{I}\left[\widehat{Sel}(\sigma_2) \le \widehat{Sel}(\sigma_1)\right]$. *Consistency* requires additivity over disjoint sub-ranges $\{\sigma_i\}_{i=1}^{k}$ whose union is σ, with Deviation Degree $= \max\left(\frac{\widehat{Sel}(\sigma)}{\sum_{i=1}^{k}\widehat{Sel}(\sigma_i)}, \frac{\sum_{i=1}^{k}\widehat{Sel}(\sigma_i)}{\widehat{Sel}(\sigma)}\right)$. *Stability* requires identical estimates, with Deviation Degree $= \max\left(\frac{\widehat{Sel}^{(j)}(\sigma)}{\widehat{Sel}(\sigma)}, \frac{\widehat{Sel}(\sigma)}{\widehat{Sel}^{(j)}(\sigma)}\right)$, where $\widehat{Sel}^{(j)}(\sigma)$ is the estimate of σ in the j-th evaluation and $\widehat{Sel}(\sigma) = \frac{1}{m}\sum_{j=1}^{m}\widehat{Sel}^{(j)}(\sigma)$. *Fidelity* requires selectivity 1 for a universal predicate ($\top$) and 0 for an invalid predicate ($\bot$), with Satisfied Ratio $= \frac{1}{|\mathcal{Q}|}\sum_{\sigma\in\mathcal{Q}}\mathbb{I}\left[(\sigma=\top \wedge \widehat{Sel}(\sigma)=1) \vee (\sigma=\bot \wedge \widehat{Sel}(\sigma)=0)\right]$.

Figure 10 shows that only Metis and *DeepDB* satisfy all four properties since they explicitly model distributions and deterministically aggregate probability masses. *MSCN* and *LW-NN* lack globally consistent distributions, while *CoLSE* independently corrects each query, causing all three to violate monotonicity, consistency, and fidelity. *Naru* and *UAE* exhibit severe violations of stability due to the inherent randomness in their sampling-based inference.

7.3 Key Designs of Metis

Attribute Subset Selection. We evaluate whether *CoG* helps identify high-utility attribute subsets. Following Lu et al. [45], we use *CORDS* [32] to compute mean-square contingency scores for all attribute pairs; these scores are directly used as *CoG* edge weights. As shown in Fig. 11, compared with *CORDS*, Metis consistently achieves lower 95^{th} *Q-error* across all datasets while maintaining comparable model sizes. This is because Metis constructs and updates the *CoG* based on estimation errors, continuously identifying workload-imperative correlations under the current statistics. Moreover, *CORDS* incurs much higher *CoG* build time on large datasets (up to 119.1× slower) as it relies on expensive data scans.

Constraint Derivation for MD-Hist Construction. We evaluate three lower-bound constraint derivation strategies: *No LB* uses no lower-bound constraints; *Greedy LB* (used in Metis) greedily selects high-selectivity queries under pairwise disjointness; and

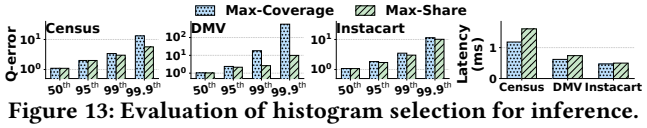


Figure 13: Evaluation of histogram selection for inference.

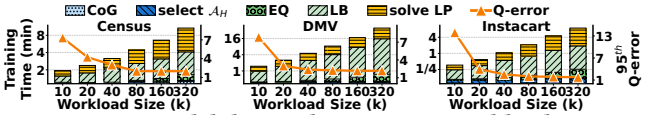


Figure 14: Scalability with increasing workload size.

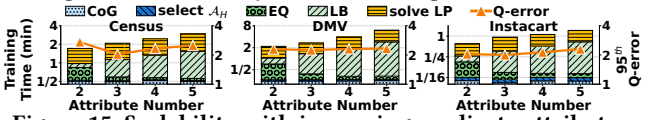


Figure 15: Scalability with increasing predicate attributes.

ILP LB computes optimal constraints via an ILP formulation. Figure 12 reports the training time and the 95^{th} *Q-error* under varying workload sizes. We observe that *Greedy LB* provides substantial accuracy improvements over *No LB* for small workloads (10k and 20k), demonstrating the effectiveness of incorporating lower-bound constraints. In contrast, *ILP LB* does not yield noticeable accuracy improvements over *Greedy LB* while incurring significantly higher construction cost because it exactly solves the MWIS problem.

Histogram Selection. We compare Metis's shared-attribute strategy (*Max-Share*) with *Max-Coverage*, which greedily maximizes query-attribute coverage. As shown in Fig. 13, *Max-Share* consistently reduces *Q-error* by enlarging the conditioning context, including a 98% reduction at the 99.9^{th} percentile on DMV. Although *Max-Share* might select more histograms and incur slightly higher latency, inference remains below 1.7 ms across all datasets.

7.4 Applicability of Metis

Scalability Analysis. We evaluate how Metis scales with training-workload size and the number of predicate attributes.

(a) Workload Size. Figure 14 reports the decomposition of training time and the 95^{th} *Q-error*. The training time is decomposed into correlation graph construction (*CoG*), attribute subset selection (*select* $\mathcal{A}_H$), equality constraint derivation (*EQ*), lower-bound constraint derivation (*LB*), and LP solving (*solve LP*). As the workload size increases, training time grows linearly, indicating proportional overhead from incorporating more query feedback. Meanwhile, the 95^{th} *Q-error* decreases rapidly and gradually converges toward 1 as additional feedback better captures the data distribution.

(b) Number of Predicate Attributes. We generate four 50K-query training workloads, each containing 40K queries with a fixed number (i.e., 2, 3, 4, or 5) of predicate attributes and 10K queries uniformly distributed across the remaining numbers. The test workload contains 5K queries with 2–5 predicate attributes. Fig. 15 shows that the training time of Metis scales well with the number of predicate attributes. The additional training time mainly comes from *LB*, as higher-dimensional queries more often partially overlap $\mathcal{A}_H$, increasing the projected regions and aggregable subsets to process. Nevertheless, Metis mitigates this overhead by using an efficient greedy heuristic. In terms of accuracy, Metis always maintains low and stable 95^{th} *Q-error*. Note, 2-attribute workloads sometimes exhibit slightly worse estimation accuracy. This is because queries with 2 predicate attributes mainly provide low-dimensional equality constraints, which may offer relatively limited information about the higher-order attribute correlations required by test queries.

Robustness Analysis. We analyze the robustness of Metis under incremental data updates and workload shifts.

(a) Data Updates. We insert 5% new tuples per step for four steps (20% in total) and measure the resulting *Q-error*. We consider two

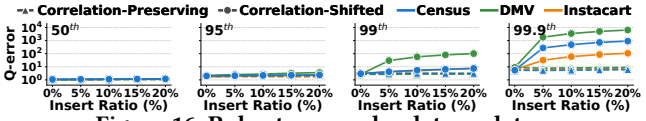


Figure 16: Robustness under data updates.

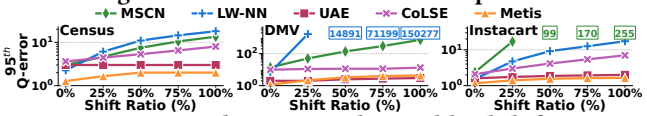


Figure 17: Robustness under workload shifts.

insertion scenarios: in the *correlation-preserving* setting, new tuples are randomly sampled from the original dataset, preserving inter-attribute correlations; in the *correlation-shifted* setting, following Wang et al. [71], we independently sort each column of a dataset copy and randomly sample tuples from it to append to the original dataset, thereby imposing different inter-attribute correlations from the original data. Fig. 16 shows that Metis remains accurate under correlation-preserving updates but degrades under correlation shifts, particularly at the 99.9^{th} percentile. Nevertheless, Metis supports efficient rebuilding, requiring only modest feedback and low construction cost to achieve high accuracy (Figs. 8 and 14).

(b) Workload Shifts. We first split query templates into seen templates and unseen templates. Then, we generate training queries only from the seen templates and gradually increase the fraction of unseen-template queries during testing (denoted as shift ratio). To evaluate robustness under workload shifts, we compare Metis with competitive estimators that leverage query workloads during training. As shown in Fig. 17, the 95^{th} *Q-error* of Metis increases with the query shift ratio. This is because the shifted workload requires workload-imperative correlations that are not captured by feedback from training templates. Nevertheless, Metis degrades relatively mildly and retains competitive 95^{th} *Q-error* under most shift ratios. This robustness mainly comes from the reuse of constructed MD-Hists when performing multi-histogram inference, as they usually still cover subsets of the unseen-template query's predicate attributes. *UAE* performs well because it learns from both data and workloads, but at the cost of substantially higher training overhead (i.e., 3.4×–153.2× of Metis). In contrast, query-driven estimators (e.g., *MSCN* and *LW-NN*) exhibit significant error increases, indicating stronger sensitivity to the coverage of the training workload.

End-to-End Query Performance Improvement. We evaluate whether improved cardinality estimates benefit end-to-end (E2E) performance on multi-table joins. We extend `pg_hint_plan` [17] to inject estimates during planning, and following Park et al. [54], generate 5,000 three-table joins on Instacart. We compare *True*, which provides true cardinalities for all plan nodes, and *PRICE*, a state-of-the-art join estimator [59, 77] using its released pre-trained model [57]. Figure 18 shows that Metis outperforms *PRICE* at all E2E speedup percentiles and approaches *True*'s performance. Since *PRICE* does not estimate single-table selection predicates, access-path decisions still rely on DBMS estimates, limiting its E2E gains. In addition, *PRICE* requires 218× the storage of Metis. Figure 19 further illustrates the distribution of E2E performance for queries whose plans change under Metis. Of 5,000 queries, 3,918 (78%) change plans, and 99.6% achieve speedups. Fewer than 0.3% of the queries regress slightly since the plan quality also depends on factors such as the empirically tuned cost model, which might select suboptimal plans even with accurate cardinalities [41, 53, 63].

		True	PRICE	Metis
E2E Speedup	Min	0.36	0.007	0.34
	50^{th}	11.07	1.01	10.3
	95^{th}	18.81	3.89	14.4
	99^{th}	22.27	4.07	21.62
	99.9^{th}	274.78	4.34	253.17
	Max	12929.1	4.46	11911.45
Latency (ms)		/	3.36	0.62
Model Size (KB)		/	42102	193

Figure 18: Performance comparison for join queries.

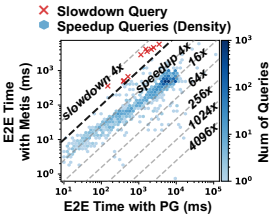


Figure 19: Metis's impact on PostgreSQL.

8 RELATED WORK

The first line of work on MD-Hist based methods assumes that the attribute subset is given *a priori*, and can be categorized as scan-based or feedback-driven. Scan-based methods [19, 20, 49, 55, 56, 62] partition the joint attribute space using recursive [19, 56] and grid-based schemes [49, 62]. While they could capture joint distributions, the expensive data scans limit their scalability. Feedback-driven methods [1, 65] exploit historical query feedback. STHoles-based methods [2, 10, 34] incrementally refine buckets per query. However, such independent processing lacks workload-wide accuracy guarantees. Next, maximum-entropy-based methods [34, 46, 47, 60, 61, 65] jointly incorporate all constraints derived from query feedback and infer bucket probabilities. However, their predicate-based partitioning incurs exponential bucket growth, limiting the number of incorporable queries during construction. Furthermore, their constraints are derived only from fully-contained queries.

The second line of work [6, 11, 44, 68, 69] maintains and combines collections of low-dimensional statistics. DBHist [6] and SASH [44] both adopt junction-tree-based decomposable graphical models to construct MD-Hists. DBHist derives the structure from intrinsic data correlations, which may not align with the target workload. SASH searches over candidate MD-Hists and evaluates their utility by constructing the corresponding histograms and measuring estimation errors using query feedback, incurring substantial overhead. Moreover, they rely on decomposable structures, which restrict the correlations that can be modeled and enforce a fixed factorization structure during inference, thereby limiting estimation accuracy.

This section reviews prior work on single-table *CardEst* using MD-Hists. Due to space constraints and the scope of this paper, learned *CardEst* methods [8, 26, 27, 36, 72, 76, 79] are only briefly discussed in §1, and we refer readers to recent surveys [25, 35, 66, 71] for a broader review.

9 CONCLUSION

In this paper, we present Metis, a workload-aware framework for constructing and utilizing multi-dimensional histograms. Experiments show that Metis achieves accurate and efficient estimation, while preserving interpretability and enabling seamless integration. In future work, we plan to investigate histogram maintenance under dynamic data updates and workload shifts, with an online decision mechanism that chooses between incremental refinement and full reconstruction.

ACKNOWLEDGMENTS

This work is supported by NSFC (92270202, 62461146205), National Key R&D Program of China (2024YFC3308102), and OceanBase, Ant Group Research Fund.

REFERENCES

[1] Ashraf Aboulnaga and Surajit Chaudhuri. 1999. Self-tuning histograms: Building histograms without looking at data. In *SIGMOD*, Vol. 28. 181–192.

[2] Nicolas Bruno, Surajit Chaudhuri, and Luis Gravano. 2001. STHoles: a multidimensional workload-aware histogram. In *SIGMOD*, Vol. 30. 211–222.

[3] Lixiang Chen, Yuxing Han, Yu Chen, Xing Chen, Chengcheng Yang, and Weining Qian. 2025. AQETuner: Reliable Query-level Configuration Tuning for Analytical Query Engines. In *VLDB*, Vol. 18. 2709–2721.

[4] S. Christodoulakis. 1984. Implications of certain assumptions in database performance evaluation. In *TODS*, Vol. 9. 163–186.

[5] DBHammer. 2026. The Source Code, Data, and Technical Report of Metis. https://github.com/DBHammer/Metis. Accessed: Sep. 2026.

[6] Amol Deshpande, Minos Garofalakis, and Rajeev Rastogi. 2001. Independence is good: Dependency-based histogram synopses for high-dimensional data. In *SIGMOD*, Vol. 30. 199–210.

[7] Dheeru Dua and Casey Graff. 1996. UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/20/census+income. Accessed: Sep. 2026.

[8] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. 2019. Selectivity estimation for range predicates using lightweight models. In *VLDB*, Vol. 12. 1044–1057.

[9] William Feller. 1991. *An introduction to probability theory and its applications*. Vol. 2. John Wiley & Sons.

[10] Dennis Fuchs, Zhen He, and Byung Suk Lee. 2007. Compressed histograms with arbitrary bucket layouts for selectivity estimation. In *Information Sciences*, Vol. 177. 680–702.

[11] Lise Getoor, Benjamin Taskar, and Daphne Koller. 2001. Selectivity estimation using probabilistic models. In *SIGMOD*, Vol. 30. 461–472.

[12] Goetz Graefe and William J McKenna. 1993. The volcano optimizer generator: Extensibility and efficient search. In *ICDE*. 209–218.

[13] Martin Grötschel, László Lovász, and Alexander Schrijver. 2012. *Geometric algorithms and combinatorial optimization*. Vol. 2. Springer Science & Business Media.

[14] PostgreSQL Global Development Group. 2026. Creating Custom Scan Paths. https://www.postgresql.org/docs/current/custom-scan-path.html. Accessed: Sep. 2026.

[15] PostgreSQL Global Development Group. 2026. Extended Statistics: Multivariate MCV Lists and Functional Dependencies. https://www.postgresql.org/docs/current/multivariate-statistics-examples.html. Accessed: Sep. 2026.

[16] PostgreSQL Global Development Group. 2026. Multivariate Statistics Examples. https://www.postgresql.org/docs/current/multivariate-statistics-examples.html. Accessed: Sep. 2026.

[17] PostgreSQL Global Development Group. 2026. pg_hint_plan: Hinting Extension for PostgreSQL. https://www.postgresql.org/about/news/pg_hint_plan-v180-released-3119/. Accessed: Sep. 2026.

[18] PostgreSQL Global Development Group. 2026. Statistics Used by the Planner. https://www.postgresql.org/docs/current/planner-stats.html. Accessed: Sep. 2026.

[19] Dimitrios Gunopulos, George Kollios, Vassilis J Tsotras, and Carlotta Domeniconi. 2000. Approximating multi-dimensional aggregate range queries over real attributes. In *SIGMOD*, Vol. 29. 463–474.

[20] Dimitrios Gunopulos, George Kollios, Vassilis J Tsotras, and Carlotta Domeniconi. 2005. Selectivity estimators for multidimensional range queries over real attributes. In *VLDBJ*, Vol. 14. 137–154.

[21] Gurobi Optimization, LLC. 2026. Gurobi Optimizer. https://www.gurobi.com. Accessed: Sep. 2026.

[22] M Yasser H. 2026. InstaCart Online Grocery Basket Analysis Dataset. https://www.kaggle.com/datasets/yasserh/instacart-online-grocery-basket-analysis-dataset. Accessed: Sep. 2026.

[23] Yiwen Han, Zhicheng Pan, Lixiang Chen, Chengcheng Yang, Rong Zhang, and Xuan Zhou. 2026. Q-Doctor: Retrieval-Augmented Diagnosis and Multi-agent Correction for Query Performance Anomalies. In *DASFAA*. 526–542.

[24] Yuxing Han, Haoyu Wang, Lixiang Chen, Yifeng Dong, Xing Chen, Benquan Yu, Chengcheng Yang, and Weining Qian. 2024. ByteCard: Enhancing ByteDance's Data Warehouse with Learned Cardinality Estimation. In *SIGMOD Companion*. 41–54.

[25] 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. In *VLDB*, Vol. 15. 752–765.

[26] Shohedul Hasan, Saravanan Thirumuruganathan, Jees Augustine, Nick Koudas, and Gautam Das. 2020. Deep learning models for selectivity estimation of multi-attribute queries. In *SIGMOD*. 1035–1050.

[27] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2020. DeepDB: learn from data, not from queries!. In *VLDB*, Vol. 13. 992–1005.

[28] Xiao Hu, Yuxi Liu, Haibo Xiu, Pankaj K Agarwal, Debmalya Panigrahi, Sudeepa Roy, and Jun Yang. 2022. Selectivity functions of range queries are learnable. In *SIGMOD*. 959–972.

[29] Zirui Hu, Rong Zhang, Chengcheng Yang, Xuan Zhou, Quanqing Xu, and Chuanhui Yang. 2025. Artemis: A Customizable Workload Generation Toolkit for Benchmarking Cardinality Estimation. In *ICDE*. 4628–4631.

[30] Xuhua Huang, Zirui Hu, Siyang Weng, Rong Zhang, Chengcheng Yang, Xuan Zhou, Weining Qian, Chuanhui Yang, and Quanqing Xu. 2025. A Query-Aware Enormous Database Generator For System Performance Evaluation. In *SIGMOD Companion*. 131–134.

[31] Aaron Hurst, Daniel E. Lucani, and Qi Zhang. 2024. PairwiseHist: Fast, Accurate and Space-Efficient Approximate Query Processing with Data Compression. In *VLDB*, Vol. 17. 1432–1445.

[32] Ihab F Ilyas, Volker Markl, Peter Haas, Paul Brown, and Ashraf Aboulnaga. 2004. CORDS: Automatic discovery of correlations and soft functional dependencies. In *SIGMOD*. 647–658.

[33] Yannis Ioannidis. 2003. The history of histograms (abridged). In *VLDB*. 19–30.

[34] Raghav Kaushik and Dan Suciu. 2009. Consistent histograms in the presence of distinct value counts. In *VLDB*, Vol. 2. 850–861.

[35] 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.

[36] 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*. 1–8.

[37] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT Press.

[38] Mario Köppen. 2000. The curse of dimensionality. In *WSC5*, Vol. 1. 4–8.

[39] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47, 2 (2001), 498–519.

[40] Steffen L Lauritzen. 1996. *Graphical models*. Vol. 17. Clarendon Press.

[41] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How good are query optimizers, really?. In *VLDB*, Vol. 9. 204–215.

[42] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2025. Still Asking: How Good Are Query Optimizers, Really?. In *VLDB*, Vol. 18. 5531–5536.

[43] Beibin Li, Yao Lu, and Srikanth Kandula. 2022. Warper: Efficiently adapting learned cardinality estimators to data and workload drifts. In *SIGMOD*. 1920–1933.

[44] Lipyeow Lim, Min Wang, and Jeffrey Scott Vitter. 2003. SASH: A self-adaptive histogram set for dynamically changing workloads. In *VLDB*. 369–380.

[45] Yao Lu, Srikanth Kandula, Arnd Christian König, and Surajit Chaudhuri. 2021. Pre-training summarization models of structured datasets for cardinality estimation. In *VLDB*, Vol. 15. 414–426.

[46] Volker Markl, Peter J Haas, Marcel Kutsch, Nimrod Megiddo, Utkarsh Srivastava, and Tam Minh Tran. 2007. Consistent selectivity estimation via maximum entropy. In *VLDBJ*, Vol. 16. 55–76.

[47] Volker Markl, Nimrod Megiddo, Marcel Kutsch, Tam Minh Tran, Peter J Haas, and Utkarsh Srivastava. 2005. Consistently estimating the selectivity of conjuncts of predicates. In *VLDB*. 373–384.

[48] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. 2009. Preventing bad plans by bounding the impact of cardinality estimation errors. In *VLDB*, Vol. 2. 982–993.

[49] M Muralikrishna and David J DeWitt. 1988. Equi-depth multidimensional histograms. In *SIGMOD*. 28–36.

[50] MySQL. 2025. Histogram Statistics in MySQL. https://dev.mysql.com/doc/refman/8.4/en/optimizer-statistics.html. Accessed: Sep. 2026.

[51] State of New York. 2020. Vehicle, Snowmobile, and Boat Registrations. https://catalog.data.gov/dataset/vehicle-snowmobile-and-boat-registrations. Accessed: Sep. 2026.

[52] Oracle. 2019. Histograms. https://docs.oracle.com/en/database/oracle/oracle-database/19/tgsql/histograms.html. Accessed: Sep. 2026.

[53] Zhicheng Pan, Wenwen Sun, Yuanjia Zhang, Terence Purcell, Yu Dong, Chengcheng Yang, Rong Zhang, Xuan Zhou, and Jianliang Xu. 2026. OBELISK: Efficient Offline Query Planning with Bayesian Optimization-Informed Language Model Reasoning. In *VLDB*, Vol. 19. 1674–1687.

[54] Yongjoo Park, Shucheng Zhong, and Barzan Mozafari. 2020. QuickSel: Quick Selectivity Learning with Mixture Models. In *SIGMOD*. 1017–1033.

[55] Viswanath Poosala, Peter J Haas, Yannis E Ioannidis, and Eugene J Shekita. 1996. Improved histograms for selectivity estimation of range predicates. In *SIGMOD*, Vol. 25. 294–305.

[56] Viswanath Poosala and Yannis E Ioannidis. 1997. Selectivity estimation without the attribute value independence assumption. In *VLDB*. 486–495.

[57] PRICE. 2024. PRICE GitHub Repository. https://github.com/StCarmen/PRICE. Accessed: Sep. 2026.

[58] Nazanin Rashedi and Guido Moerkotte. 2025. The Accuracy of Cardinality Estimators: Unraveling the Evaluation Result Conundrum. In *VLDB*, Vol. 18. 3744–3756.

[59] Lankadinee Rathuwadu, Guanli Liu, Christopher Leckie, and Renata Borovica-Gajic. 2026. CoLSE: A Lightweight and Robust Hybrid Learned Model for Single-Table Cardinality Estimation using Joint CDF. In *ICDE*. 905–918.

[60] Christopher Ré and Dan Suciu. 2010. Understanding cardinality estimation using entropy maximization. In *PODS*. 53–64.

[61] Christopher Ré and Dan Suciu. 2012. Understanding cardinality estimation using entropy maximization. In *TODS*, Vol. 37. 1–31.

[62] Michael Shekelyan, Anton Dignös, and Johann Gamper. 2017. Digithist: a histogram-based data summary with tight error bounds. In *VLDB*, Vol. 10. 1514–1525.

[63] Tarique Siddiqui, Alekh Jindal, Shi Qiao, Hiren Patel, and Wangchao Le. 2020. Cost Models for Big Data Query Processing: Learning, Retrofitting, and Our Findings. In *SIGMOD*. 99–113.

[64] SQL Server. 2022. Statistics. https://learn.microsoft.com/en-us/sql/relational-databases/statistics. Accessed: Sep. 2026.

[65] Utkarsh Srivastava, Peter J Haas, Volker Markl, Marcel Kutsch, and Tam Minh Tran. 2006. Isomer: Consistent histogram construction using query feedback. In *ICDE*. 39–39.

[66] Ji Sun, Jintao Zhang, Zhaoyan Sun, Guoliang Li, and Nan Tang. 2021. Learned cardinality estimation: A design space exploration and a comparative evaluation. In *VLDB*, Vol. 15. 85–97.

[67] Luca Trevisan. 2014. Inapproximability of combinatorial optimization problems. In *Paradigms of Combinatorial Optimization: Problems and New Approaches*. 381–434.

[68] Kostas Tzoumas, Amol Deshpande, and Christian S Jensen. 2011. Lightweight graphical models for selectivity estimation without independence assumptions. In *VLDB*, Vol. 4. 852–863.

[69] Kostas Tzoumas, Amol Deshpande, and Christian S Jensen. 2013. Efficiently adapting graphical models for selectivity estimation. In *VLDBJ*, Vol. 22. 3–27.

[70] Qingshuai Wang, Hao Li, Zirui Hu, Rong Zhang, Chengcheng Yang, Peng Cai, Xuan Zhou, and Aoying Zhou. 2024. Mirage: Generating enormous databases for complex workloads. In *ICDE*. 3989–4001.

[71] Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. 2021. Are We Ready For Learned Cardinality Estimation?. In *VLDB*, Vol. 14. 1640–1654.

[72] Peizhi Wu and Gao Cong. 2021. A unified deep model of learning from both data and queries for cardinality estimation. In *SIGMOD*. 2009–2022.

[73] Ziniu Wu, Parimarjan Negi, Mohammad Alizadeh, Tim Kraska, and Samuel Madden. 2023. FactorJoin: a new cardinality estimation framework for join queries. In *SIGMOD*, Vol. 1. 1–27.

[74] Ziniu Wu, Amir Shaikhha, Rong Zhu, Kai Zeng, Yuxing Han, and Jingren Zhou. 2020. BayesCard: Revitilizing Bayesian Frameworks for Cardinality Estimation. *arXiv preprint arXiv:2012.14743* (2020).

[75] Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. 2020. NeuroCard: one cardinality estimator for all tables. In *VLDB*, Vol. 14. 61–73.

[76] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep Unsupervised Cardinality Estimation. In *VLDB*, Vol. 13. 279–292.

[77] Tianjing Zeng, Junwei Lan, Jiahong Ma, Wenqing Wei, Rong Zhu, Pengfei Li, Bolin Ding, Defu Lian, Zhewei Wei, and Jingren Zhou. 2024. PRICE: A Pretrained Model for Cross-Database Cardinality Estimation. In *VLDB*, Vol. 18. 637–650.

[78] Haozhe Zhang, Christoph Mayer, Mahmoud Abo Khamis, Dan Olteanu, and Dan Suciu. 2025. LpBound: Pessimistic Cardinality Estimation Using ℓp-Norms of Degree Sequences. In *SIGMOD*, Vol. 3. 1–27.

[79] Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, and Bin Cui. 2021. FLAT: Fast, Lightweight and Accurate Method for Cardinality Estimation. In *VLDB*, Vol. 14. 1489–1502.