

Data-Agnostic Cardinality Learning from Imperfect Workloads

Peizhi Wu^{*}
University of Pennsylvania
pagewu@cis.upenn.edu

Rong Kang
ByteDance
kangrong.cn@bytedance.com

Tieying Zhang[†]
ByteDance
tieying.zhang@bytedance.com

Jianjun Chen
ByteDance
jianjun.chen@bytedance.com

Ryan Marcus
University of Pennsylvania
rcmarcus@cis.upenn.edu

Zachary G. Ives
University of Pennsylvania
zives@cis.upenn.edu

ABSTRACT

Cardinality estimation (CardEst) is a critical aspect of query optimization. Traditionally, it leverages statistics built directly over the data. However, organizational policies (e.g., regulatory compliance) may restrict global data access. Fortunately, *query-driven* cardinality estimation can learn CardEst models using query workloads. However, existing query-driven models often require access to data or summaries for best performance, and they assume *perfect* training workloads with complete and balanced join templates (or join graphs). Such assumptions rarely hold in real-world scenarios, in which join templates are incomplete and imbalanced.

We present GRASP, a *data-agnostic* cardinality learning system designed to work under these real-world constraints. GRASP’s *compositional design* generalizes to unseen join templates and is robust to join template imbalance. It also introduces a new per-table CardEst model that handles value distribution shifts for range predicates, and a novel *learned count sketch* model that captures join correlations across base relations. Across three database instances, we demonstrate that GRASP consistently outperforms existing query-driven models on imperfect workloads, both in terms of estimation accuracy and query latency. Remarkably, GRASP achieves performance comparable to, or even surpassing, traditional approaches built over the underlying data on the complex CEB-IMDb-full benchmark — despite operating without any data access and using only 10% of all possible join templates.

PVLDB Reference Format:

Peizhi Wu, Rong Kang, Tieying Zhang, Jianjun Chen, Ryan Marcus, and Zachary G. Ives. Data-Agnostic Cardinality Learning from Imperfect Workloads. PVLDB, 18(8): XXX-XXX, 2025.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at <https://github.com/shoupzwu/GRASP>.

1 INTRODUCTION

At enterprise scale and beyond, data management and access may be *compartmentalized*, due to (1) organizational structure and implementation (e.g., different groups adopt incompatible database platforms or access control mechanisms), (2) limited-access agreements between groups (e.g., one organization only grants access through restricted query APIs), and/or (3) regulatory policy and the principle of least privilege (e.g., due to HIPAA or FERPA, certain data fields may be protected). In such settings, query optimization remains essential for performance, despite the fact some data is kept private¹: the organization may need to execute federated queries [22], provide a service that abstracts over alternative DBMS platforms [20], or even develop query optimization-as-a-service [32]. Consider our motivating use case involving two real-world scenarios at ByteDance. First, within the organization, numerous internal business units require query optimization services to enhance performance. However, these units often handle sensitive data, such as TikTok user profiles and e-commerce transactions, which are subject to strict data privacy constraints. Consequently, sharing this data across departments is restricted. Additionally, ByteDance’s cloud-native services provide query services, but many users are hesitant to allow access to their personal data.

Traditional query optimization relies on cardinality estimation (CardEst) that leverages statistics built directly over base relations. A potentially promising alternative is to instrument *query workloads* and their cardinality results, which typically raise fewer privacy concerns than direct data access, and are often shared more readily by users with database vendors for optimizing query performance. Importantly, pairs of SQL queries and their resultant cardinalities can be collected with minimal impact on system performance. This facilitates the feasibility of developing learned *query-driven* CardEst models based primarily on query workloads, circumventing the need for direct access to underlying data. Recent studies [44, 56] also demonstrate that machine learning (ML)-based query-driven approaches, which learn a regression model that predicts the cardinality for an input query, have the potential to consistently outperform traditional histogram-based methods. This is because traditional methods make assumptions like independence and uniformity of data distributions, whereas learning-based techniques can model data correlations and skewness. These lessons motivate us to adopt query-driven CardEst techniques, which do not access the underlying data while enhancing estimation accuracy.

^{*}Most of the work was done during Peizhi Wu’s internship at ByteDance.

[†]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. 18, No. 8 ISSN 2150-8097.
doi:XX.XX/XXX.XX

¹Note that by “privacy”, we refer to constraints on what data can be seen by other users or systems. The restrictions do not guarantee differential privacy [18].

Table 1: Problem settings for query-driven cardinality estimation in the literature versus production environments.

Problem Setting	Data Access	Coverage of Join Templates	Distribution of Join Templates	Distribution of Predicate Values
in the literature	Yes	Complete	Balanced	Static
in production	No	Incomplete	Imbalanced	Shifting

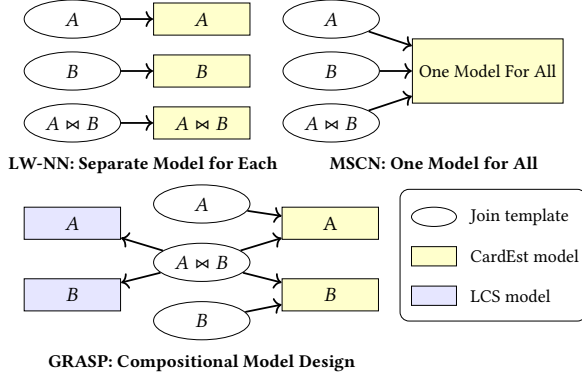


Figure 1: Join handling in existing query-driven methods

Limitations of existing approaches. However, it is challenging to apply existing ML-based query-driven CardEst methods to our real-world settings for three main reasons.

First, most of them [16, 35] are *not* purely query-driven as they still require data-derived metadata (e.g., samples and histogram estimates) for better generalization performance.

Second, they assume *perfect training workloads* in which the join templates (i.e., join graphs that connect base tables²) are complete and balanced. As demonstrated in § 2.2, these assumptions may not hold in real-world environments. Specifically, apart from data inaccessibility, as shown in Table 1, production workloads are inherently *imperfect*, characterized by *incomplete and imbalanced join templates*. This issue is especially problematic in business workloads, which may introduce new join templates over time (Figure 2c). Even if join templates remain constant, for a query from seen join templates, cost models must estimate cardinalities for all possible subqueries — including those with join templates not seen in the workload³ — to make accurate cost predictions. Unfortunately, existing query-driven models suffer in these scenarios. Specifically, for **join template incompleteness**, as shown in Figure 1, they handle joins by either encoding each join in the query encoding individually [35] or by training separate models for each join template [16]. However, neither approach generalizes well to unseen join templates. While join bitmaps [44] improve their generalizability to unseen join templates, constructing join bitmaps requires data access. Furthermore, for **join template imbalance**, the lack of consistency across join templates in existing methods means that knowledge learned from the “majority” join templates

(with more training queries) cannot be transferred to the “minority” join templates (with very few training queries).

Third, even for the same query templates (with fixed join graph and queried columns), production workloads often experience shifts (Figure 2d) in value/literal distributions (aka., *value distribution shifts*). However, existing query-driven approaches rely heavily on data information (e.g., samples, histograms) to handle value distribution shifts [35, 44]. Nevertheless, without such data information (a key constraint in this paper), these models are prone to overfitting to the training query distribution, resulting in poor performance when tested on queries from different distributions [44, 55].

Contributions. Motivated by these challenges, this paper makes contributions as follows.

- We summarize **real query workloads in production**, leading to a **new problem setting** for CardEst: *data-agnostic cardinality learning from imperfect workloads* (§ 3.1).
- We develop GRASP, a *truly data-agnostic* CardEst system that does not require data access and provides robust and generalizable estimates over imperfect workloads. GRASP develops a design innovation (**D1**) with two components (**C1**, **C2**):
 - D1. A general and compositional design** that handles incomplete and imbalanced join templates, using the notion of **compositional generalization** (§ 4).
 - C1. A new query-driven per-table CardEst model** (ARCDF) that is robust to changes in value distributions for range predicates (§ 5). ARCDF is inspired by NEUROCDF, which introduces the CDF modeling paradigm for CardEst. However, NEUROCDF, as a framework, does not specify the CDF prediction model; and existing attempts, as discussed in [56], may face issues with negative estimates. ARCDF mitigates these challenges by introducing a *new CDF prediction model* that utilizes a deep autoregressive model and enforces monotonicity through *monotonic piecewise splines*.
 - C2. A novel query-driven learned count sketch (LCS) model** that captures join correlations across base relations (§ 6). Instead of prior count sketches built over the data, the LCS model learns from queries to output *low-dimensional representations* that effectively approximate the dot products of join key distributions in the results of per-table subqueries.
- We validate that GRASP, with no data or statistics, achieves both **generalizable and robust CardEst accuracy** and **reduced query latency** over imperfect training workloads. Notably, on the complex CEB-IMDb-full benchmark [43] with up to 16-way joins, GRASP achieves comparable or even superior performance to traditional methods with access to the data (§ 7). We use *only* 10% of all possible join templates.

GRASP performs well with perfect workloads. However, since we developed it based on the challenges of real-world scenarios, we focus on evaluating its performance under those conditions to highlight the *practical utility* of GRASP.

Data updates/shifts. In production environments, we observe that user data typically remains static throughout a week, and production systems often accumulate a significant number of queries weekly. For example, businesses we can access in ByteDance generates, on average, over 2 million queries per workload (each corresponding to a database instance) each week. Therefore, we adopt a

²Join templates and join graphs are interchangeable throughout this paper.

³In real-world workloads, such as those collected at ByteDance, we only observe the ultimate cardinality of each query without access to the cardinalities of its subqueries.

pragmatic approach to manage data distribution shifts: we retrain GRASP weekly using the queries collected during that period.

2 PRELIMINARIES

This section first introduces notations and concepts, and then summarizes real-world production workloads we collected at ByteDance.

2.1 Definitions

Definition 2.1 (Database Instance). A database instance \mathcal{DB} comprises a set of base relations/tables $\{T_j\}_{j=1}^m$, where m denotes the number of tables in \mathcal{DB} . Each table T_j includes a set of columns or attributes $\{A_i\}_{i=1}^n$, where n represents the number of attributes per table. We define the cardinality $|T_j|$ of each table T_j as the number of total tuples in T_j . We also define the domain of an attribute A , denoted as $\text{dom}(A)$, as the set of all distinct values in A .

Definition 2.2 (Query). A query is a structured request to retrieve data from a database based on specified operations and conditions. This paper focuses on SPJ queries with *inner equi-joins*, following most learned CardEst work [28, 33, 57, 58]. GRASP is capable of handling chain joins, star joins, and self-joins, all of which we evaluate in the paper. While GRASP can also be extended to handle cyclic joins under the join key independence assumption (as in [57]), we did not evaluate cyclic joins, as most benchmarks focus on acyclic joins. For supported predicates, we focus on equality ($=$), range ($<$, \leq , $>$, \geq), string matching (LIKE), containment (IN), and null-checking (NULL) predicates. These supported predicates are consistent with most existing learned CardEst work and are included in the benchmarks we evaluate in this paper.

Other features *e.g.*, Group-By, Distinct) also impacts query cardinalities. While existing techniques [34] could be integrated with GRASP to address additional features, they fall outside the scope of this work. Moreover, these features are not prevalent in the real-world queries we collected from ByteDance: 82% of queries we collected fit within the scope of this paper.

Definition 2.3 (Join Template/Join Graph). A join template, also referred to as a join graph, connects base relations through their join keys. It consists of nodes and edges, where each node represents a base relation and each edge represents a join operation between two relations based on join keys. Here, a join key is defined as a pair of attributes from two relations that are used to establish a condition for joining tuples from those relations. This paper considers both foreign key-primary key (FK-PK) and FK-FK equi-joins.

Definition 2.4 (Query Template). A query template is a parameterized query pattern that maintains consistent or fixed join template as well as other predicates outside of the joining conditions, varying only in the values or literals specified within non-join predicates. This paper does not consider nested queries, following most of learned CardEst work [28, 33, 57, 58].

Definition 2.5 (Cardinality Estimation). Given a database instance \mathcal{DB} and query q over \mathcal{DB} , the goal of cardinality estimation is to predict the cardinality $c(q)$, *i.e.*, the number of tuples that satisfy q . Another equivalent term, selectivity, is the ratio $c(q)/|T|$, where $|T|$ is the table cardinality or the join size if q is a join query. In query optimization, the optimizer needs to estimate the cardinality for each subquery of q , as each subquery can be considered a query.

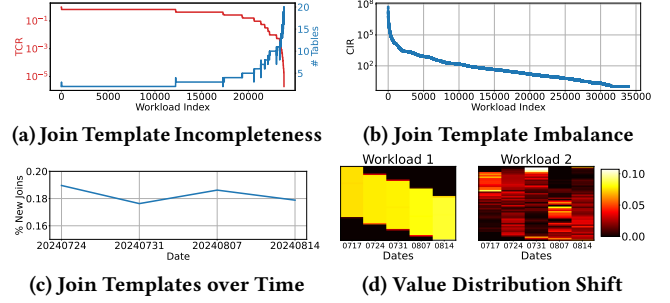


Figure 2: Production Workload Analysis at ByteDance

Definition 2.6 (Template Coverage Ratio). Let \mathcal{T} be the set of all possible join templates of a schema, and $\mathcal{T}_{\text{train}} \subset \mathcal{T}$ be the set of templates observed during training. We define **Template Coverage Ratio (TCR)** as $\text{TCR} = \frac{|\mathcal{T}_{\text{train}}|}{|\mathcal{T}|}$. A lower TCR indicates that more join templates are missing from the training workload.

Definition 2.7 (Class Imbalance Ratio). Let n_t be the number of training queries for a join template $t \in \mathcal{T}_{\text{train}}$. We define **Class Imbalance Ratio (CIR)**: $\text{CIR} = \frac{\max_{t \in \mathcal{T}_{\text{train}}} n_t}{\min_{t \in \mathcal{T}_{\text{train}}} n_t}$. A higher CIR indicates a greater imbalance between join templates.

Definition 2.8 (Granularity). We define **Granularity** as the *range size* for a query q on a specific *numerical* attribute A_i that supports *range predicates/filters*. For example, let ℓ_i and r_i be the lower and upper bounds of q 's filter over A_i . Then: $\text{Granularity}(q) = r_i - \ell_i$.

2.2 Analysis of Production Workloads

This section uses real workloads collected from ByteDance, a technology company operating a range of social media platforms, to illustrate the characteristics (Figure 2) of RDBMS workloads over 30 days with more than 35,000 applications. In Figure 2a 2b, a workload index corresponds to the query workload of an internal business. The key features of these workloads are detailed as follows:

In Figure 2a, the red line represents the TCR of join templates during the period; the blue line represents the number of tables involved in the workload. Only $\sim 0.03\%$ of workloads' query templates combine all possible table combinations, implying that **production workloads seldom contain complete join templates**.

Figure 2b shows the imbalance in join templates within each workload. We use the Class Imbalance Ratio (CIR) to measure the imbalance of join templates in a workload, defined as the ratio of the most and least common join templates. A higher CIR suggests a greater imbalance. In Figure 2b, $\sim 60\%$ of the workloads have a CIR > 10 , and 20% have a CIR > 1000 , indicating a **significant presence of imbalanced workloads in production environments**.

As illustrated in Figure 2c, each week, each workload exhibits $\sim 18\%$ new query templates on average that were unseen in the previous week. This indicates that **in business scenarios, the appearance of new join templates is common**.

Figure 2d uses two business workloads to illustrate weekly variations in predicate values from fixed templates. The x-axis denotes dates, while the y-axis represents normalized value distributions of a predicate. The left shows a systematic shift due to time-increasing IDs focused on the latest week, whereas the right depicts irregular

shifts from random user IDs. Both figures highlight that **predicate values within the same query template can change over time**.

3 THE GRASP SYSTEM

In this section, we define the problem, discuss the key design choices of GRASP, and provide an overview of the GRASP system.

3.1 Problem Overview

Problem: *Data-Agnostic Cardinality Learning from Imperfect Query Workloads (DACL).*

Inputs: We are provided with a set of SPJ queries $Q = \{q_i\}_{i=1}^n$ and their ultimate cardinalities $C = \{c(q_i)\}_{i=1}^n$ collected from a database instance \mathcal{DB} , along with the database schema for \mathcal{DB} and the table cardinalities (i.e., numbers of total rows) $\{|T_j|\}_{j=1}^m$, where m is the number of tables in \mathcal{DB} . Training queries do not contain predicates on join keys, since we observe only 2% of real-world queries in production environments apply filters on join keys.

Constraints: 1). **No Data Access:** Direct access to the database \mathcal{DB} is prohibited. This means no samples, histograms, or direct statistics about the data are available. 2). **Incomplete Join Templates:** The training queries do not cover all possible join templates, leaving most unseen join templates untrained. We use TCR as the measure for incompleteness, e.g., $\text{TCR} = 0.3$. 3). **Imbalanced Join Templates:** The distribution of training queries across different join templates is skewed, with some templates having very few queries available for training. We use CIR as the measure for imbalance, e.g., $\text{CIR} = 100$. 4). **Value Distribution Shifts:** The value distributions for *range predicates* in incoming queries may differ significantly from those seen during training. We focus on *granularity shifts*, i.e., changes in the distribution of granularity defined in Definition 2.8.

Goals: The objective is to develop a model/system that can accurately predict the cardinality (or selectivity) for any incoming query q , under the following conditions:

- (1) The query might involve a join template that was not seen during training.
- (2) The query may come from a join template that has very few examples in the training set.
- (3) The query may contain predicate values with distributions that differ from the training queries.

Although our primary focus is on imperfect workloads - motivated by real challenges in Bytedance - a desired model should also apply to *perfect* workloads with complete and balanced join templates. Since we developed GRASP primarily to address real challenges (which are likely to arise in other enterprises as well), our evaluations focus on imperfect workloads to underscore the practical utility of our approach in real-world settings. Nonetheless, in principle GRASP requires no modification to handle perfect workloads.

3.2 Key Design Choices

Unfortunately, to the best of our knowledge, no existing cardinality estimation approach can effectively achieve the goals simultaneously under the constraints in the DACL problem. To achieve this,

we present GRASP: Generalizable, and Robust, data-Agnostic cardinality Prediction system. Below, we provide an overview of GRASP, detailing the desiderate (D) and our solutions.

- **D1: Generalization to unseen join templates despite no access to the underlying data.**

Our Solution: GRASP achieves this goal through the notion of *compositionality*. The core idea is that, instead of training a single model to handle all join templates [35] or training separate models for each join template [16], we only learn models for *necessary primitives*. First, we learn query-driven CardEst models only for base relations. Second, without data access, it is challenging to model join correlations. GRASP borrows the concept of *count sketches* in the literature [3, 48]. However, instead of existing count sketches that are built from data, GRASP introduces *learned count sketch (LCS)* models to learn *low-dimensional* count sketches that capture join correlations across base relations, *solely from queries*. Put it all together, GRASP allows unseen join templates to be addressed by composing the corresponding primitives. For a taste of the compositional design of GRASP, consider the simple example:

EXAMPLE 3.1. Consider a scenario illustrated in Figure 1 where base relations A and B are joined using a key x with a large domain size (e.g., $|\text{dom}(x)| = 10^6$). To compute the cardinality $c(q^{A \bowtie B})$ of the join query $q^{A \bowtie B}$, it is necessary to consider not only the cardinalities $c(q^A)$ and $c(q^B)$ of the subqueries on base relations A and B , but also the distribution of the join keys within these subqueries. Let \mathbf{f}_{q^A} and \mathbf{f}_{q^B} represent the probability distributions of the join key x in the two subqueries q^A and q^B , respectively. The cardinality of the $q^{A \bowtie B}$ can be computed as follows,

$$c(q^{A \bowtie B}) = c(q^A) \cdot c(q^B) \cdot (\mathbf{f}_{q^A} \cdot \mathbf{f}_{q^B}) \quad (1)$$

$$\approx c(q^A) \cdot c(q^B) \cdot (\mathbf{v}_{q^A} \cdot \mathbf{v}_{q^B}) \quad (2)$$

$$\text{where } \underbrace{\mathbf{f}_{q^A} \cdot \mathbf{f}_{q^B}}_{\text{high-dimensional}} \approx \underbrace{\mathbf{v}_{q^A} \cdot \mathbf{v}_{q^B}}_{\text{low-dimensional}}$$

where $c(q^A)$ and $c(q^B)$ are computed by per-table CardEst models for A and B , respectively. \mathbf{v}_{q^A} and \mathbf{v}_{q^B} are low-dimensional (e.g., $|\mathbf{v}_{q^A}| = |\mathbf{v}_{q^B}| = 500$) count sketch computed by LCS models for A and B . In this computation, (1) is according to the definition of joins; (2) is based on the assumption that LCS models approximate the scalar dot product (i.e., $\mathbf{f}_{q^A} \cdot \mathbf{f}_{q^B}$) through their learned count sketches for queries q^A and q^B , thus efficiently capturing join correlations between A and B .

- **D2: Robustness to join template imbalance despite no access to the underlying data.**

Our Solution: GRASP naturally achieves this goal through its compositional system design. By leveraging compositionality, GRASP ensures that model predictions are consistent across different join templates. This allows knowledge learned from the "majority" join templates to be effectively transferred to the "minority" join templates, which addresses the issue of join template

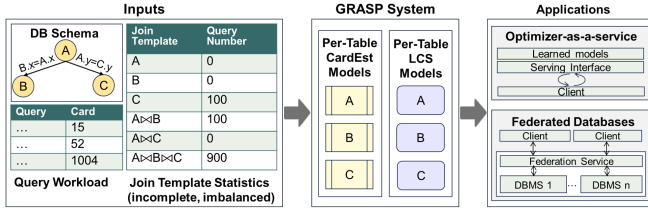


Figure 3: Workflow of GRASP.

imbalance and improves prediction accuracy for underrepresented join templates.

D3: Robustness to value distribution shifts despite no access to the underlying data.

Our Solution: GRASP builds on the NEUROCDF framework from [56], which learns models to predict cumulative distribution functions (CDFs) rather than direct query selectivities. This approach enhances out-of-distribution generalization for range predicates by ensuring predictions are induced from signed measures, enforcing the additivity constraint of selectivity/cardinality functions. Additionally, GRASP improves on NEUROCDF by introducing a novel CDF prediction model, ARCDF, that addresses a key limitation of the original NEUROCDF framework.

Sample Complexity. Compared to non-DNN models [9, 45, 51], the use of DNNs in GRASP improves the prediction accuracy due to higher model capacity [34, 56], at the cost of increasing the need for training queries (e.g., higher sample complexity). However, the sample complexity remains modest. For instance, in our experiments, the maximum number of queries used (on CEB-IMDb-full) is $\sim 260K$, significantly lower than the 2 million queries generated weekly per workload at ByteDance. The sample complexity of GRASP and compared models will be further evaluated in § 7.2.2. However, the sample complexity is evaluated based on the query type outlined in Definition 2.2, and introducing additional features such as Group-By would increase the sample complexity, which is a limitation of this work.

3.3 GRASP Overview

Recall that GRASP does not have direct access to DBMS data, but it does have access to schema information and a query workload with ultimate cardinality outcomes.

GRASP Workflow. Combining the design choices introduced before, GRASP operates in two main stages: the *building stage* and the *serving stage*. The workflow of GRASP is depicted in Figure 3. In the building stage, GRASP ingests the schema of the DB instance along with a collected query workload, which may be incomplete or imbalanced. The goal of this stage is to train the primitive models (i.e., the CardEst model and the LCS model for each base relation) in GRASP using the query workload. Since all computational steps in CardEst with GRASP (will be introduced in § 4.3) are fully differentiable, we can use modern stochastic optimization methods, such as backpropagation and stochastic gradient descent (SGD), to efficiently learn the model parameters θ of all primitive models. The primitive models (both CardEst and LCS models) for each base relation T will be trained on queries in which T is involved.

Specifically, training is performed by minimizing the following loss function over the training workload (N training queries):

$$\mathcal{L} = \left(\sum_{i=1}^N \text{Error}(\hat{c}(q_i), c(q_i)) \right) / N, \quad (3)$$

where Error is the error function measuring the discrepancy between the estimated cardinality $\hat{c}(q_i)$ from GRASP and the true cardinality $c(q)$. Users can specify this error function, with common choices including Square Error. i.e., $SE = (\hat{c}(q) - c(q))^2$, Qerror: $(\max_{\hat{c}(q)} \frac{\hat{c}(q)}{c(q)}, \frac{c(q)}{\hat{c}(q)})$ or Squared Logarithmic Error, i.e., $SLE = (\log \hat{c}(q) - \log c(q))^2$. Optimizing SLE is equivalent to optimizing Qerror [43] (resulting in consistent outcomes between SLE and Qerror measurements). While GRASP is compatible with any error functions, we adopt SLE as it yields the best overall performance.

Once built, GRASP can serve as the underlying cardinality estimator for federated databases [22], services that abstract over alternative DBMS platforms [20], and for query optimization-as-a-service [32]. Note that all these applications may not have access to the underlying data.

Optimizing GRASP. Queries from different join templates may require distinct sets of primitive models. This enables the application of *distributed training* with multiple GPUs, where join templates with non-overlapping primitive models can be trained concurrently across multiple computing nodes. By partitioning the training tasks based on their *primitive model dependencies*, we can assign each group of non-overlapping join templates to separate workers by using a *greedy packing algorithm* [10] for scheduling. Due to space constraints, we omit the details in this paper.

To perform query optimization for a join query, we need to estimate the cardinalities of all subqueries. Leveraging a GPU can accelerate this process through batched inference. However, even with CPU-based inference, *progressive query inference* can significantly reduce GRASP’s computational overhead, due to its compositional design. This approach estimates subqueries in a bottom-up fashion, starting with those containing fewer joins. By saving and reusing results from these smaller joins, we avoid redundant computations in estimations of larger joins, enhancing overall efficiency.

4 COMPOSITIONAL GENERALIZATION

4.1 Motivating Examples

Consider the database instance in figure 4 with three base relations $\{A, B, D\}$, each of cardinality 5. They can be joined via join keys $A.x, B.x, D.x$ (with no restrictions on primary or foreign keys), which belong to a join group x with domain: $\text{dom}(x) = \{1, 2\}$. Figure 4b shows four possible join templates: $A \bowtie B, A \bowtie D, B \bowtie D$, and $A \bowtie B \bowtie D$.

Inferring cardinalities for queries over a “hidden” table. We begin by examining the scenario where a specific table is absent from any single-table queries within the training workload. Instead, the workload comprises single-table queries on other tables and join queries that involve this “hidden” table. Our objective is to explore the feasibility of inferring cardinality estimates for the hidden table under these conditions.

We first focus on deriving the **unknown** cardinalities of queries q^B on relation B by leveraging **known** cardinalities from queries

q^A on relation A and queries $q^{A \bowtie B}$ on join template $A \bowtie B$. Similar to Example 3.1, calculating the cardinality of a join query $q^{A \bowtie B}$ needs the cardinalities of the individual subqueries q^A, q^B on base relations A, B , and the distributions of join keys in both. Here we simplify the problem by assuming a small domain size for x , $|\text{dom}(x)| = 2$. Specifically, assume that the join key distributions for $x \in \{1, 2\}$ in q^A, q^B are $\mathbf{p}^A = [p_1^A, p_2^A]^\top, \mathbf{p}^B = [p_1^B, p_2^B]^\top$. Let $\mathbf{L} = [c(q^A), c(q^B)]$ and $\mathbf{P} = [\mathbf{p}^A, \mathbf{p}^B]^\top$, by definition of joins, the cardinality of the $q^{A \bowtie B}$ is

$$c(q^{A \bowtie B}) = \|\mathbf{LP}\|_1 = \underbrace{c(q^A)p_1^A c(q^B)p_1^B}_{\text{join key } x=1} + \underbrace{c(q^A)p_2^A c(q^B)p_2^B}_{\text{join key } x=2}. \quad (4)$$

Eq. 4 can be easily generalized to joins (via x) with > 2 relations.

Apparently, given the join key distributions of queries from tables A and B (i.e., \mathbf{P}), we can estimate the cardinality $c(q^B)$ of any query q^B using Eq. 4. However, these distributions are typically *hidden* and not directly available. Therefore, in order to infer $c(q^B)$, we also need to infer these join key distributions from known cardinalities, $c(q^A)$ and $c(q^{A \bowtie B})$. This approach is viable provided we have a sufficient number of queries for q^A and $q^{A \bowtie B}$.

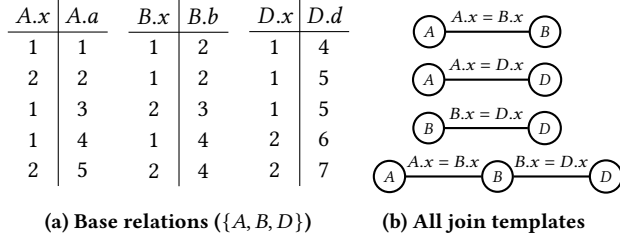


Figure 4: The database instance for motivating examples.

Table 2: Queries on table A

Query	Predicate	Card
q_1^A	$\{a \leq 2\}$	2
q_2^A	$\{2 < a \leq 4\}$	2
q_3^A	$\{a > 4\}$	1

Table 3: Queries on table B

Query	Predicate	Card
q_1^B	$\{b \leq 3\}$?
q_2^B	$\{b > 4\}$	$5 - c(q_1^B)$

Table 4: Queries on join $A \bowtie B$

Query	Predicate	Card
$q_1^{A \bowtie B}$	$\{a \leq 2 \wedge b \leq 3\}$	3
$q_2^{A \bowtie B}$	$\{a \leq 2 \wedge b > 3\}$	2
$q_3^{A \bowtie B}$	$\{2 < a \leq 4 \wedge b \leq 3\}$	4
$q_4^{A \bowtie B}$	$\{2 < a \leq 4 \wedge b > 3\}$	2
$q_5^{A \bowtie B}$	$\{a > 4 \wedge b \leq 3\}$	1
$q_6^{A \bowtie B}$	$\{a > 4 \wedge b > 3\}$	1

EXAMPLE 4.1. In our query workload, we have three queries on relation A (as listed in Table 2) and six join queries on $A \bowtie B$ (detailed in Table 4). Each query is represented by a tuple consisting of (the query name, predicate, and cardinality). For notation, we define p_i^T as the proportion of tuples in the query q_i^T on relation T with join key $x = 1$; thus, the remaining proportion with $x = 2$ is $1 - p_i^T$. For instance, p_1^A denotes the proportion of tuples in the query q_1^A on A with join key $x = 1$. Our objective is to deduce the cardinalities of queries on the “hidden” relation B , specifically for queries q_1^B and q_2^B as outlined in Table 3. Thus, the primary unknown is $c(q_1^B)$.

The challenge lies in the absence of join key distributions for queries on A and B , which include five unknowns: $p_1^A, p_2^A, p_3^A, p_1^B$, and p_2^B .

However, the six join queries from $q_1^{A \bowtie B}$ to $q_6^{A \bowtie B}$ provide a basis to establish six linearly dependent equations using Eq. 4. Combining these with the three known cardinalities from queries on A (i.e., $c(q_1^A), c(q_2^A), c(q_3^A)$) and employing linear algebra techniques, we can solve these equations to determine $c(q_1^B) = 3$ along with other unknowns.

Note that the approach of “decomposing large queries into smaller queries” applies to queries of any size (e.g., queries with more than two joins). In other words, small join templates (e.g., involving hidden tables) can be effectively learned from the larger join templates they are part of.

Composing a novel join template. Now, we explore composing queries of a novel join template that was unseen in the workload. We use information from the query workload, as well as inferred query cardinalities and join correlations. Before moving forward, we note that Eq. 4 can be generalized to accommodate join templates involving multiple tables (i.e., number of tables $n_d > 2$) that belong to the same join group, say x . In this generalized case, \mathbf{L} is a vector of length n_d , representing the cardinalities of single-table subqueries on each base relation. Additionally, \mathbf{P} is an $n_d \times n_k$ matrix, where n_k is the number of distinct values of join key x . Each row of \mathbf{P} is the join key distribution for the corresponding single-table subquery.

EXAMPLE 4.2. Assume that apart from the queries discussed in Example 4.1, the query workload includes an additional query on table D , $q_1^D : \{d \leq 5\}$ with $c(q_1^D) = 3$; and one query on the join template $B \bowtie D$, $q_1^{B \bowtie D} : \{b \leq 3 \wedge d \leq 5\}$ with $c(q_1^{B \bowtie D}) = 6$. The goal is now to compose queries on a larger join template $A \bowtie B \bowtie D$ that has not been observed in the query workload.

From Example 4.1, we already determined that $c(q_1^B) = 3$ and $p_1^B = \frac{2}{3}$. Therefore, we can establish the following equation for $q_1^{B \bowtie D}$ with only one unknown, p_1^D , using Eq. 4:

$$3 \cdot \frac{2}{3} \cdot 3 \cdot p_1^D + 3 \cdot (1 - \frac{2}{3}) \cdot 3 \cdot (1 - p_1^D) = 6$$

Solving this equation, we have $p_1^D = 1$. Then we can derive the cardinality of the query $q_1^{A \bowtie B \bowtie C} : \{a \leq 2 \wedge b \leq 3 \wedge d \leq 5\}$ by using the generalized version of Eq. 4:

$$c(q_1^{A \bowtie B \bowtie C}) = 2 \cdot \underbrace{\frac{1}{2} \cdot 3 \cdot \frac{2}{3} \cdot 3 \cdot 1}_{\text{join key } x=1} + 2 \cdot \underbrace{\frac{1}{2} \cdot 3 \cdot \frac{1}{3} \cdot 3 \cdot 0}_{\text{join key } x=2} = 6$$

Other queries such as $q_2^{A \bowtie B \bowtie C} : \{2 < a \leq 4 \wedge b \leq 3 \wedge d \leq 5\}$ and $q_3^{A \bowtie B \bowtie C} : \{a > 4 \wedge b \leq 3 \wedge d \leq 5\}$ can be composed similarly as we have derived necessary elements (e.g., cardinalities of single-table subqueries and corresponding join key distributions).

4.2 The Compositional Design

From the previous examples, we draw two key takeaways. **First**, smaller join templates not present in the query workload could be inferred from larger ones. **Second**, by combining explicit workload information (e.g., cardinalities of training queries) with inferred information (e.g., cardinalities of hidden relations and join key distributions), we can reliably estimate the cardinalities of queries from unseen join templates. Therefore, a key question arises: how can we more effectively extract and utilize both the explicit and implicit information from the query workload to perform accurate cardinality

estimation for as many join templates as possible? This informs the compositional design of GRASP.

Compositionality. Compositionality as compositional generalization, refers to the capacity for systematic generalization to new, combined examples from a specific distribution, after training on a different distribution that introduced the necessary *components/primitives*. This concept emphasizes the ability to understand and apply combinations of learned primitives in novel contexts. For example, Pavel et.al [53] trains an image classification model that decomposes concepts into parts and allows it to generalize to novel categories with fewer examples.

Primitives. With regard to the DACL problem, we introduce two types of primitive models for each base relation: 1) Per-table CardEst models and 2) Per-table Join Key (JK) models, both receiving single-table queries on the associated relation as inputs. Per-table CardEst models predict the cardinality of the query, whereas JK models output the distribution of the join key in the query results (thus capturing the correlations among different base relations via join keys). If a base relation includes multiple join keys, the JK models output a set of distributions, one for each join key. We formally define their abstracts as follows.

- **Per-table CardEst Models** M_{CE} : Query \rightarrow Cardinality
- **Join Key (JK) Models** M_{JK} : Query \rightarrow {Join Key Distribution}

GRASP builds these primitive models from queries, enabling generalizable and consistent cardinality estimation across join templates, which is the key to achieving **D1** and **D2** in § 3.3. Since these primitive models can be effectively trained on any relevant queries, they do not require specific single-table or two-table queries.

The use of join key models, M_{JK} , has exhibited certain limitations. Consequently, we will revisit and revise the design of M_{JK} by replacing it with the *learned count sketch* models, M_{LCS} , as detailed in § 6. For the purpose of our current discussion, we will provisionally consider M_{JK} and M_{LCS} to be interchangeable.

Model Choices for M_{CE} . Note that the compositional design of GRASP is general, which means any query-driven model architectures can be used for M_{CE} . For tables involving range predicates, GRASP uses the ArCDF model as it is more robust to value distribution shifts for range predicates (will be introduced in § 5). For tables involving other types of predicates (such as LIKE, IN, and more), GRASP employs the Multi-Set Convolutional Networks (MSCN) [35] for M_{CE} , as the MSCN encoding mechanism is flexible enough to incorporate complex operators (e.g., LIKE) [43]. For details about the MSCN encoding, refer to [35]. We will discuss the model choices for M_{LCS} in § 6.

4.3 Cardinality Estimation with Primitives

To estimate the cardinality $\hat{c}(q)$ of a query q , we use the CardEst models $\{M_{CE}\}$ and join key models $\{M_{JK}\}$ via GRASP’s main function: $\hat{c}(q) = \text{Est}(q, \{M_{CE}\}, \{M_{JK}\})$.

4.3.1 Single-table Queries. Answering queries over base relations is trivial — GRASP uses the corresponding per-table CardEst models to estimate the cardinality results.

4.3.2 Join Queries. Estimating join queries involves the use of both per-table CardEst models and join key models that are related to the join queries. First, we define an *equivalence join key group* as

the set of keys that are joinable according to the database schema. For example, if $A.x = B.x$ and $B.x = D.x$ are specified in the schema, then $A.x$, $B.x$, and $D.x$ are considered to belong to the same join key group x . The full set of join key groups and the table ordering within each key group can be easily precomputed from the database schema [33]. Now, we begin with an easier case of join estimation.

Case 1: One join key group. Consider a join query q that involve a set \mathcal{T} of base relations. They are joined via join keys in the same join key group. Similar to Example 4.2. GRASP employs four steps to calculate the query cardinality \hat{c} .

- (1) Calculating the cardinality estimates for single-table subqueries (i.e., $\hat{c}(q^T)$ for $T \in \mathcal{T}$), using associated M_{CE} .
- (2) Calculating the join key distributions (i.e., $\hat{f}(q^T)$ for $T \in \mathcal{T}$) for single-table subqueries, using associated M_{JK} .
- (3) Calculating the frequencies of join keys in each single-table query: $\widehat{freq}(q^T) = \hat{c}(q^T) \cdot \hat{f}(q^T)$ for $T \in \mathcal{T}$.
- (4) Multiplying the frequencies for every single query and then summing over the result: $\hat{c} = \|\prod_{T \in \mathcal{T}} \widehat{freq}(q^T)\|_1$.

Algorithm 1: Estimating a join query with GRASP

Input: Per-table CardEst models $\{M_{CE}\}$ and JK models $\{M_{JK}\}$;
Query q with involved tables \mathcal{T} and join key groups $\mathcal{G} = \{G\}$.
Output: GRASP’s cardinality estimate of q : $\hat{c}(q)$.

```

1: # compute cardinalities for per-table subqueries
2: for  $T \in \mathcal{T}$  do
3:    $\hat{c}(q^T) = M_{CE}^T(q^T)$  # associated CardEst model
4: # computing join estimates
5:  $\mathcal{T}_{curr} \leftarrow \{\}$ ,  $\hat{c} \leftarrow 1$ 
6: while  $\mathcal{G}$  is not empty do
7:    $G = \text{FindNextGroup}(\mathcal{T}_{curr}, \mathcal{G})$ 
8:    $\text{key}_G \leftarrow \text{join key of the group } G$ 
9:    $\hat{f} \leftarrow \mathbf{1}_{|\text{dom}(\text{key}_G)|}$  # Init. distrib. vector
10:  for  $T \in G.\text{tables}$  do
11:     $\hat{f} = \hat{f} \odot M_{JK}^T(q^T, \text{key}_G)$  # associated JK model
12:    if not  $T \in \mathcal{T}_{curr}$  then
13:       $\hat{c} = \hat{c} * \hat{c}(q^T)$ 
14:      add  $T$  to  $\mathcal{T}_{curr}$ 
15:     $\hat{c} = \hat{c} * \|\hat{f}\|_1$ 
16:  Remove  $G$  from  $\mathcal{G}$ 
17: return  $\hat{c}$ 
```

Case 2: Multiple join key groups. GRASP can be extended to handle the case when the join query q contains join keys from a set of multiple join key groups, $\mathcal{G} = \{G\}$. The estimation procedure is shown in Algorithm 1.

The algorithm estimates the cardinality of a q by leveraging both per-table CardEst models M_{CE} and join key models M_{JK} . It begins by calculating cardinality estimates for single-table queries on each relation using corresponding M_{CE} (lines 1-3). Then, it recursively processes groups of join keys by finding joinable groups (the implementation of FindNextGroup is straightforward using the database schema and can be pre-computed) that connect tables already considered (lines 19-27). For each group, it initializes a distribution vector (lines 9) and iteratively refines it by multiplying with the join key distribution estimate from the associated JK model

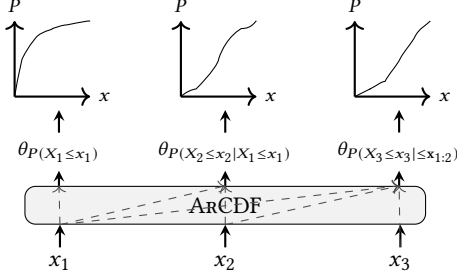


Figure 5: The ARCDF model: Autoregressively outputs parameters θ for monotonic rational-quadratic splines.

M_{JK} (lines 10-11, where $M_{JK}^T(q^T, \text{key}_G)$ represents the predicted distribution of key_G in query q^T by model M_{JK}^T). The overall cardinality estimate is updated by multiplying with the sizes of newly included tables and the *sum* of the refined distribution vector (lines 12-15). This process continues until all join key groups have been incorporated, ultimately yielding the estimated cardinality of q .

4.3.3 Remark. Acute readers might notice that Algorithm 1 assumes that join keys *within a table* are independent. Despite this simplification, it works well in practice without requiring much computation. Although GRASP could be naturally extended to account for these correlations, our tests show that this only slightly improves accuracy while significantly increasing computational costs. Therefore, GRASP keeps the independence assumption to balance efficiency and accuracy for real-world use.

5 BUILDING ROBUST CARDEST MODELS

Recall that in D3 of § 3.3, we outline the key goal for per-table CardEst models: robustness to value distribution shifts for range predicates. This section presents our solution — ARCDF.

5.1 Overview

The NEUROCDF Framework. ARCDF builds upon the NEUROCDF framework. The central idea of NEUROCDF [56] is to construct query-driven neural network (NN) models that predict the underlying cumulative distribution functions (CDFs) for cardinality estimation, rather than directly estimating query selectivities/cardinalities. Then, the selectivity of any rectangular query can be computed as a linear combination of the CDFs evaluated at its vertices [15]. Formally speaking, the CDF prediction model in NEUROCDF parameterizes the underlying CDFs: $F(\mathbf{x}) = P(X \leq \mathbf{x})$, where $\mathbf{x} = [x_1, x_2, \dots, x_d]$ is a real value variable \mathbf{x} , and d is the dimension (*i.e.*, the number of attributes of the table).

Compared to the common modeling paradigm that targets query cardinalities directly, NEUROCDF *provably* provides robust cardinality estimates for out-of-distribution (OOD) queries. Specifically, it ensures the *additivity constraint* or *containment property* of cardinality functions: $\hat{c}(q_1) = \hat{c}(q_2) + \hat{c}(q_3)$ whenever $q_1 = q_2 \cup q_3$ and $q_2 \cap q_3 = \emptyset$. This property enhances GRASP’s robustness to value distribution shifts for range predicates.

Limitation. Despite these advantages, NEUROCDF has a limitation. As noted in [56], NEUROCDF is not compatible with Q-error or SLE

due to potential negative cardinality estimates. This issue arises because the multilayer perceptron (MLP) model used in NEUROCDF may fail to learn a monotonically increasing function — necessary property for CDFs. This restricts the practical applicability of NEUROCDF, as Q-error is a commonly used error metric in cardinality estimation due to their emphasis on highly selective queries [41].

Our Solution. We enhance the NEUROCDF framework by introducing an Autoregressive CDF prediction model. We refer to our solution as ARCDF: NEUROCDF with AutoRegressive CDF modeling, which is illustrated in Figure 5. The key idea of ARCDF is to ensure that the CDF prediction model learns a monotonically increasing function *by model design*, which alleviates negative cardinality estimates. This is achieved by: (1) employing a deep AR model to parameterize the CDFs, and (2) enforcing the monotonicity property along each attribute using *monotonic piecewise splines*.

5.2 Autoregressive CDF Modeling

ARCDF decomposes the joint CDF $F(\mathbf{x}) = P(X \leq \mathbf{x})$ in an autoregressive manner:

$$F(\mathbf{x}) = \prod_{i=1}^d P(X_i \leq x_i | X_{i-1} \leq x_{i-1}, \dots, X_1 \leq x_1), \quad (5)$$

where $P(X_i \leq x_i | X_{i-1} \leq x_{i-1}, \dots, X_1 \leq x_1)$ is a *conditional CDF* given the event: $X_{i-1} \leq x_{i-1}, \dots, X_1 \leq x_1$. For simplicity, we denote this as $P(X_i \leq x_i | \leq \mathbf{x}_{1:i-1})$ hereafter. This conditional CDF depends only on the values of $x_{1:i-1}$. Unlike the conventional definition of conditional CDFs that condition on exact values (*i.e.*, $X_{i-1} = x_{i-1}, \dots, X_1 = x_1$), our definition conditions on *inequalities*, which ensures the correctness of decomposition.

The motivation behind this autoregressive decomposition is twofold. First, it allows us to leverage recent advances in deep autoregressive models [21] for efficient and accurate (conditional) CDFs estimation. Second, by decomposing the joint CDF into a product of conditional CDFs, we can conveniently enforce the monotonicity constraint along each dimension.

Modeling CDFs with Deep Autoregressive Models. To parameterize the sequence of CDFs, we employ modern deep AR models such as MADE [21]. These models have proven to be efficient and powerful for cardinality estimation by parameterizing joint probability density functions (PDFs) [58, 59]. To the best of our knowledge, we are the *first* to leverage deep AR models to parameterize joint CDFs, enabling efficient and accurate CDF modeling.

Typically, deep AR models output a categorical distribution that represents the conditional probability distribution over the attribute domain at each dimension. Unfortunately, in the DACL problem, the domain values of each attribute are *not* available.

5.3 Parameterizing Conditional CDFs

To address this challenge, ARCDF uses deep AR models to parameterize *piecewise spline functions* along each attribute x_i , representing the conditional CDF, $P(X_i \leq x_i | \leq \mathbf{x}_{1:i-1})$. This approach offers two main advantages: (1) it allows for a flexible trade-off computational efficiency and model expressiveness without requiring knowledge of the attribute domain values; and (2) ensuring monotonicity in each conditional CDF becomes straightforward by leveraging the rich literature in monotonic piecewise spline functions [23, 50].

Monotonic Rational-Quadratic Splines. Common choices for monotonic piecewise spline functions include linear polynomial splines [47], cubic splines [19], and rational-quadratic splines [50]. Among these, rational-quadratic splines offer the most expressive modeling capability while maintaining computational efficiency, as demonstrated in applications such as density estimation and image generative modeling in recent ML research [14]. Therefore, ARCDF employs rational-quadratic splines to model the conditional CDFs for each attribute. Specifically, these splines are defined on one-dimensional data x and consist of K consecutive rational-quadratic functions, connected at $K + 1$ monotonically increasing knots, denoted as $\{(x^k, y^k)\}_{k=0}^K$. The parameters for each segment are the outputs of the deep AR model at each dimension. Due to space constraints, we omit the exact mathematical expression of the rational-quadratic function; interested readers may refer to [23].

Validity of Learned Joint CDFs. ARCDF ensures the conditional CDFs are monotonically increasing because all knots are monotonically increasing and the derivatives δ are positive. However, in principle, this does not guarantee the validity of the learned joint CDFs. Monotonicity along each one-dimensional conditional CDF does not ensure that the cardinality estimate of the query over the multi-dimensional data is always valid (i.e., non-negative).

However, in our experiments, we find that ARCDF *automatically* learns better CDFs, with only a small portion of “outlier” queries yielding negative estimates (GRASP switches to square error loss in this case). This improves on previous CDF model architectures (e.g., MLP). We attribute this improvement to the autoregressive decomposition combined with the enforcement of monotonic conditional CDFs, allowing ARCDF to effectively learn valid joint CDFs from training queries. Although we cannot provide theoretical guarantees, our empirical results demonstrate that ARCDF achieves a significantly lower Q-error than other model choices in NEUROCDF. One could also add a penalty term to loss function to enforce the global monotonicity, but we do not explore this in the paper.

6 CAPTURING JOIN CORRELATIONS

In this section, we outline the practical challenges associated with using join key models M_{JK} in GRASP. We then present our proposed solution, which effectively mitigates these issues to ensure robust and efficient join correlation modeling.

6.1 Challenges and the Idea

Challenges. First, in real-world datasets, join keys often possess **large domain sizes**, with millions of unique values. This makes it highly impractical to build join key prediction models defined in § 3.3. Second, as mentioned in the definition of the DACL problem, both the **join key domain and its size are unknown** due to no data access. This further complicates the task of constructing models that accurately capture join correlations.

Key Idea. GRASP leverages the notion of *count sketches* [3, 48] to effectively approximate join sizes while accounting for join correlations. The fundamental concept behind count sketches involves creating *low-dimensional* vector representations for data streams. For instance, consider two data streams S_1 and S_2 where each stream consists of tuples in the form (key, frequency), and the number of

distinct key values is denoted as $|\text{dom}(\text{key})|$. By constructing compact vector representations \mathbf{v}_1 and \mathbf{v}_2 for S_1 and S_2 respectively (where $|\mathbf{v}_1| \ll |\text{dom}(\text{key})|$), we can efficiently estimate the join size $|S_1 \bowtie S_2|$ by computing the scalar dot product $\mathbf{v}_1 \cdot \mathbf{v}_2$.

6.2 Query-Driven Learned Count Sketches

To estimate the join size of two query results, q_1 and q_2 , with corresponding join key distributions \mathbf{f}_1 and \mathbf{f}_2 , count sketches construct low-dimensional vectors \mathbf{v}_1 and \mathbf{v}_2 such that $\|\mathbf{v}_1 \cdot \mathbf{v}_2\|_1 \approx \|\mathbf{f}_1 \cdot \mathbf{f}_2\|_1$. This approximation allows the join size to be estimated as $c(q_1) \cdot c(q_2) \cdot \|\mathbf{v}_1 \cdot \mathbf{v}_2\|_1$. The approach naturally extends to joins involving more than two query results.

However, constructing count sketches typically requires scanning the underlying data, which is not permissible in the DACL problem. To overcome this limitation, we propose training machine learning models to generate *learned* count sketches directly from input queries. These learned count sketch (LCS) models, denoted as M_{LCS} , can seamlessly replace the join key models described in § 4. The training process for M_{LCS} follows the same procedure as that for M_{JK} without accessing the underlying data. Therefore, we will use M_{LCS} to replace M_{JK} for each base relation, hereafter.

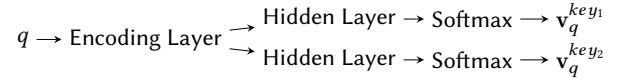


Figure 6: The Learned Count Sketch (LCS) model.

Model Architecture. For a single-table query q , the learned count sketch \mathbf{v}_q is a normalized vector. The LCS model (Figure 6) encodes the query q into an embedding/vector, processes it through a hidden layer, and outputs the learned count sketch \mathbf{v}_q (which is a n_{lcs} -dimensional normalized distribution) using a softmax activation function, where n_{lcs} is a hyperparameter. Existing query encoding techniques (e.g., MSCN [35] and flatten encoding [17]) can serve as the encoding layer. Again, GRASP chooses the MSCN encoding due to its flexibility in handling complex filters. If the relation includes multiple join keys from different groups, the LCS models output a set of count sketches via *different hidden layers*, all of which share a *common encoding layer*.

EXAMPLE 6.1. Consider the LCS model in Figure 6 where the table T includes two different join keys, key_1 and key_2 . Each count sketch is derived from the same query embedding but processed via different hidden layers for each key. Let’s define the hidden layer size as n_h (256 in our experiments). For each key, a hidden layer consists of a weight matrix $W \in \mathbb{R}^{n_h \times n_{lcs}}$ and a bias vector $b \in \mathbb{R}^{n_{lcs}}$. Thus, the model first produces an n_h -dimensional embedding E_q from the encoding layer, and applies the transformation for the count sketch as $\mathbf{v}_q = \text{softmax}(E_q W + b)$. This transformation is executed separately for each key using distinct sets of W and b .

6.3 Comparison to Join Histograms

One may find that GRASP along with the LCS models share a similar high-level idea with join histograms [31] (or FactorJoin [57] that extends the join histograms) — *learning per-table CardEst models and join correlation models separately*.

However, they are *conceptually different* in three key ways. **First**, join histograms are built from the data, whereas LCS models are learned exclusively from queries. **Second**, they rely on *different assumptions*: join histograms assume that join keys within each bin are uniformly distributed, while LCS models assume that the dot products of join key distributions can be approximated using low-dimensional count sketches. **Third**, unlike the join histograms (or FactorJoin) framework which uses *correlated* join histograms and per-table CardEst models for *bin-specific* cardinality estimates, we employ a different approach due to production queries often lacking filters on join keys. Consequently, accurate bin-specific models are inaccessible. Instead, we utilize separate per-table CardEst models to estimate the *overall* cardinality for joined tables without relying on binning. Additionally, LCS models are used to address the correlations between the results of per-table subqueries, *disentangling* the estimation of per-table cardinalities and join correlations.

7 EXPERIMENTAL EVALUATION

In this section, we primarily answer: despite having no access to the data, **RQ1**: Can GRASP generalize to unseen join templates? **RQ2**: Can GRASP maintain robustness to join template imbalance? **RQ3**: Can GRASP achieve robustness to value distribution shifts in range predicates? Moreover, we also explore **RQ4**: Can GRASP lead to better query end-to-end performance?

Table 5: Statistics of data and workloads of DB instances

DB Instance	CEB-IMDb-full	DSB	BD
# Tables	21	25	7
Max # tales in joins	16	5	4
# All possible join templates	3219	20	28
Template Coverage Ratio (TCR)	%10	%25	%28
Class Imbalance Ratio (CIR)	108	7.4	17

7.1 Experimental Setup

Datasets and Workloads. We conducted experiments using three DB instances: CEB-IMDb-full [43], DSB [13], an internal business unit (query logging system) at ByteDance (**BD**). This system stores information (e.g., count and running time) about each SQL template, and allows users to issue queries for monitoring and diagnosing slow queries. Table 5 presents the statistics of data and workloads of these DB instances. The CEB-IMDb-full benchmark is derived from the JOB [36] benchmark, whose data distribution is highly correlated and skewed [36]. Moreover, CEB-IMDb-full provides hundreds of queries per handcrafted base template⁴ with real-world interpretations [43] as opposed to five in the JOB benchmark. Moreover, CEB-IMDb-full contains various join types (e.g., star, chain, self-joins) and complex predicates (e.g., LIKE, IN). This variety makes it an ideal benchmark for our experiments. We primarily utilize it to evaluate **RQ1** and **RQ2** (due to its complexity), and **RQ4** (due to its large number of joins). The DSB dataset is an extension of the TPC-DS benchmark [46] with more complex data distributions. Following [56], we populate the DSB dataset with a scale factor of 50 and use the workload generated by [56] mainly for evaluating **RQ3** because its queries are range predicate-focused (though we

also evaluate it for other RQs). For BD, we utilize real data and queries (over a day) directly collected from the internal business.

Baselines. We are not aware of any deep learning-based query-driven approaches that operate without data access; non-deep learning approaches [29, 45] do not require data access, but they generally show poorer empirical performance [56] and do not support joins. Therefore, we employ two representative deep-learning-based query-driven models, LW-NN [16] and MSCN [35], as our baselines, modifying their query encodings to exclude data information. We refer to the variants as LW-NN_{w/oData} and MSCN_{w/oData}. We implemented LW-NN [17]. For MSCN, we utilized the available code from [1]. Note that LW-NN’s encoding is not flexible to handle LIKE predicates; in such cases, we adopt MSCN’s encoding while retaining LW-NN’s approach to handling joins — using a separate model for each join template. All query-driven models use a batch size of 128. For GRASP, the dimensions (n_{lcs}) of the LCS models are set to 2000, 100 and 3000 for CEB-IMDb-full, DSB and BD, respectively.

Moreover, we include two unfair approaches that require data access for comparison: the histogram-based estimator in PostgreSQL (PG) to demonstrate the potential of query-driven models, and the original version of MSCN that incorporates sample bitmaps in its query encoding (MSCN_{w/Data}), to illustrate the superiority of GRASP. We do not include LW-NN_{w/Data} as MSCN_{w/oData} generally outperforms LW-NN_{w/oData} in all experiments.

Evaluation Metrics. For prediction accuracy (**RQ1**, **RQ2**, and **RQ3**), we use Qerror. For query end-to-end performance (**RQ4**), we report the query running latency/time, which includes the query execution time and the inference time. The latter includes the total time the model takes to estimate the cardinalities for all subqueries. **Environment.** All model training was performed on a Nvidia Tesla V100 32GB GPU (although all models consume < 10% of the GPU memory). All query latency experiments were performed on a Debian 9 Linux machine. The hardware included an Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40GHz with 8 cores and a clock speed of 2.394 GHz, along with 16 GB of RAM. We utilize a modified version [26] of PostgreSQL 13.1 for our query end-to-end performance evaluations. During query optimization, the PostgreSQL optimizer employs injected cardinality estimates for query planning.

7.2 Generalization to Unseen Join Templates

This subsection evaluates **RQ1**. To construct the training workloads, we first retain queries from base templates, as these typically include the largest join templates observed in real-world workloads. From all the subqueries of these base templates, we randomly sample 10% – 20% of the join templates, and incorporate their queries into the training workload. For BD, we directly used real queries collected as the training workload. The statistics of data and workloads (including TCR and CIR) are shown in Table 5.

Table 6 presents the accuracy of compared approaches for seen and unseen join templates, respectively. MSCN consistently outperforms LW-NN, demonstrating that MSCN’s join handling approach provides a degree of generalizability superior to that of LW-NN. However, despite slight improvements from sample bitmaps (which are independently sampled per table and lack information on join correlations), MSCN_{w/Data}’s performance *significantly* deteriorates when moving from seen to unseen join templates. Notably, it even

⁴The term “base template” refers to a predefined query template that generates various specific queries by substituting parameters within that template.

Table 6: Accuracy (Qerror) on seen/unseen join templates, where unseen join templates account for the majority of all possible join templates. The first two methods require data access, which is infeasible in the real-world setting of this paper.

Method	Information Needed	DSB			CEB-IMDb-full			BD (real workload)		
		Median	%95	Mean	Median	%95	Mean	Median	%95	Mean
Postgres	Data	1.5/1.3	9.6/8.1	3.6/3.2	1781/6.1	$1 \times 10^6/702$	$4 \times 10^5/717$	276/3.1	$1 \times 10^6/184$	$9 \times 10^6/4 \times 10^4$
MSCN _{w/Data}	Queries + Data	1.1/5.2	2.3/5533	2.0/5915	1.3/1.8	3.4/328	3.5/2188	1.3/32	3.2/1251	2.0/2048
LW-NN _{w/oData}	Queries	2.3/n/a	37/n/a	7.2/n/a	3.1/n/a	42/n/a	17/n/a	1.8/n/a	4.8/n/a	3.5/n/a
MSCN _{w/oData}	Queries	1.5/1.9	$5.3/2 \times 10^6$	$2.1/8 \times 10^4$	1.9/2.5	7.1/663	3.7/2671	1.6/75	4.0/1865	2.7/2904
GRASP	Queries	1.8/1.8	8.7/15	6.7/9.5	2.0/1.7	9.4/23	5.7/41	1.5/5.5	4.2/149	3.2/320

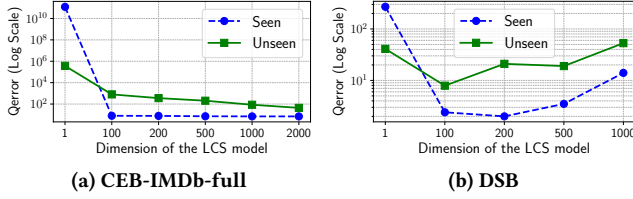


Figure 7: Mean Qerror with varying dimensions of the learned count sketch model across two datasets.

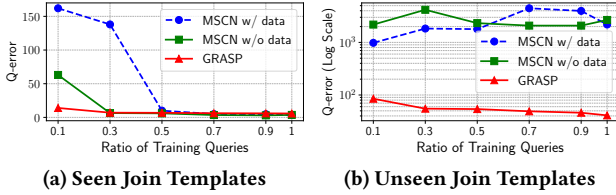


Figure 8: Mean Q-error with varying ratios of training queries on CEB-IMDb-full.

underperforms PG in terms of mean Q-error on unseen join templates within the CEB-IMDb-full benchmark. Note that PG is data-driven and does not require training queries. This is why its Qerror is worse for ‘seen’ versus ‘unseen’ queries, as the seen set contains the largest join templates, which typically pose challenges for PG.

Importantly, we observe GRASP can achieve very consistent and robust accuracy on both seen and unseen join templates, across the three datasets. Moreover, it even surpasses MSCN_{w/Data} and PG that are built over the data. This demonstrates GRASP’s superior generalization performance, attributed to its *compositional design*.

7.2.1 Ablation Study. We also perform an ablation study to validate the effectiveness of the proposed learned count sketch model. Figure 7 presents the estimation accuracy of GRASP for seen and unseen join templates across different dimensions (n_{lcs}) of LCS models. The significant improvement from $n_{lcs} = 1$ (representing independent join assumptions) to $n_{lcs} = 100$ confirms that *LCS models effectively capture join correlations among relations*.

7.2.2 Sample Complexity. We then evaluate the sample complexity of GRASP and two MSCN variants. We replicate the experiments in § 7.2, varying the training query ratios. Figure 8 shows the

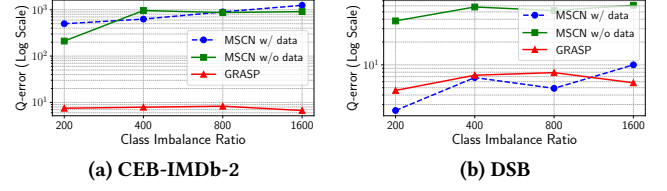


Figure 9: Accuracy with varying class imbalance ratios (CIR).

mean Q-errors for seen and unseen join templates at different ratios. Notably, GRASP achieves good accuracy with only 10% (25,701) of the original training queries for both seen and unseen join templates, suggesting its *moderate* sample complexity. Additionally, GRASP consistently outperforms both MSCN variants across all training query ratios for unseen queries.

7.3 Robustness to Join Template Imbalance

Now, we assess the robustness of the compared methods to join template imbalances (RQ2). We employ CIR [52] (as defined in Definition 2.7) to quantify the imbalance scale. To create imbalanced workloads for a specified CIR, we first sort the join templates in descending order based on their query counts. Starting with the largest count (n_l), we iteratively reduce each subsequent template’s query count by a decay factor (e.g., 1.5) until the condition $n_l/n_s \geq \text{CIR}$ is satisfied, where n_s is the current query count. All remaining join templates are then assigned a query count of n_s . We evaluate their performance on test workloads that contain an *equal* number of queries for each join template to assess robustness to join template imbalance. For CEB-IMDb-full, we create imbalanced workloads by combining base templates that share the same prefix ID (e.g., 3a, 3b) with their respective subqueries. This enables a more fine-grained evaluation of robustness due to the extensive variety of join templates in CEB-IMDb-full. For DSB, we directly construct imbalanced workloads from all available join templates, as it contains a considerably smaller number of join templates.

Figure 9 presents the accuracy of query-driven approaches, excluding LW-NN since it is consistently outperformed by MSCN. CEB-IMDb-2 is constructed using base templates with prefix IDs 2 from CEB-IMDb-full. As we can see, on DSB, MSCN_{w/Data} improves upon MSCN_{w/oData} and is comparable to GRASP. However, when evaluating on CEB-IMDb-2, which features more complex

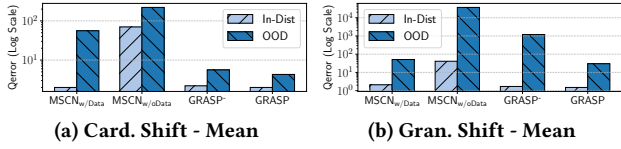


Figure 10: Accuracy under value distribution shifts for DSB.

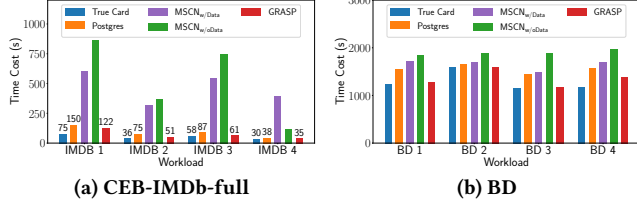


Figure 11: Overall time cost (running time).

join graphs, GRASP significantly outperforms both MSCN variants. This demonstrates that GRASP is robust to join template imbalance.

7.4 Robustness to Value Distribution Shifts

This subsection addresses RQ3 using the DSB dataset. Following [56], we introduce granularity (Gran.) shifts, where the granularity (g) of range predicates changes (from $g \neq 0.5$ to $g = 0.5$). Additionally, we evaluate cardinality (Card.) shifts, referring to changes in query cardinalities (e.g., from small to large values).

Figure 10 presents the mean Q-errors of query-driven approaches under both shift types. Here, GRASP⁺ represents GRASP using the original NEUROCDF as the CDF prediction model (i.e., GRASP without ARCDF). We use GRASP⁺ to evaluate the accuracy of NEUROCDF and to determine if ARCDF improves on NEUROCDF.

As shown in the figure, MSCN_{w/Data} improves MSCN’s robustness to value distribution shifts, especially for granularity shifts. Furthermore, GRASP⁺ consistently outperforms MSCN_{w/oData}, indicating that NEUROCDF is more robust to value distribution shifts compared to MSCN_{w/oData}. However, for granularity shifts, GRASP⁺ is less effective than MSCN_{w/Data} due to latter’s additional data information. The gap can be bridged by ARCDF (used in GRASP), which demonstrates ARCDF’s advantages over NEUROCDF.

7.5 Impact on End-to-End Performance

This subsection evaluates the benefits of GRASP in improving query latency performance (RQ4). We focus on the more complex CEB-IMDb-full (with up to 16-way joins) and real queries in BD. For the test workloads, we randomly sample four workloads from all base templates (e.g., those with larger joins), consisting of 10 queries each for CEB-IMDb-full and 20 queries each for BD. We also include True Cardinalities (True-Card) as Oracle. Note that compared approaches use the same runtime, differing only in the cardinality estimates.

Figure 11 presents the overall running time for both datasets. We report the ratios of the total running time (summing all four workloads) of the evaluated models relative to True-Card, which serves as the baseline and is set at 100%. On CEB-IMDb-full, GRASP incurs a running time ratio of 135%, significantly outperforming MSCN_{w/oData} (1044%) and MSCN_{w/Data} (930%), while achieving

performance better than PG (176%). It is worth noting that both PG and MSCN_{w/Data} have access to full data statistics. These results demonstrate the strong performance of GRASP in enhancing query optimization. On BD, GRASP achieves running time performance comparable to True-Card, while significantly outperforming MSCN_{w/Data} and MSCN_{w/oData} (126%). This suggests that query-driven models may perform worse than simple baselines like PG when they fail to accurately estimate the majority of subqueries.

Figure 12 further presents the per-query running time on CEB-IMDb-full, with a timeout of 300 seconds. We observe that both MSCN_{w/Data} and MSCN_{w/oData} time out on some queries. Additionally, MSCN_{w/oData} exhibits significant degradation on a large number of queries. In contrast, GRASP achieves running times close to True-Card for the vast majority of queries without noticeable query regression. This demonstrates that GRASP effectively generalize to various join templates, despite being trained on only 10% of the all possible join templates.

7.6 Efficiency

Training Time. Training GRASP takes 198s, 4.9s, and 51s per epoch on the IMDb-CEB-full, DSB, and BD datasets, respectively. Moreover, GRASP converges within 50 epochs for all datasets.

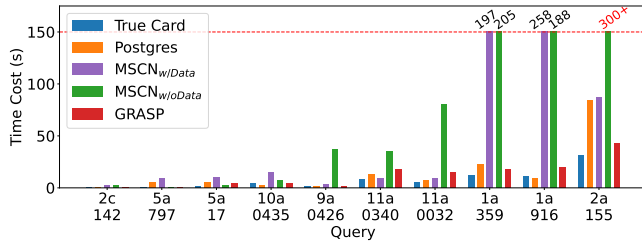
Query Inference. We focus on the query inference of the largest join (16-way join) on CEB-IMDb-full. For query optimization of such a 16-way join query, GRASP completes the estimation of all subqueries (including itself) in less than 0.5s using batch inference on a GPU. On a CPU, GRASP achieves 1.6s inference time, which is not costly compared to the query running time. Specifically, although a 16-way join query involves estimating 6092 subqueries, only 32 calls to GRASP’s primitive models (CardEst and LCS models) are needed — one call to each model for each of the 16 tables. This step only takes 0.15s. After obtaining these model outputs, the cardinality of each subquery is estimated using these outputs (Algorithm 1). Here, the computations for the 6092 subqueries can be efficiently handled through progressive query inference. For instance, the cardinality estimate for $t \bowtie m_i$ is cached and subsequently reused in the estimation of $t \bowtie m_i \bowtie c_i$.

8 RELATED WORK

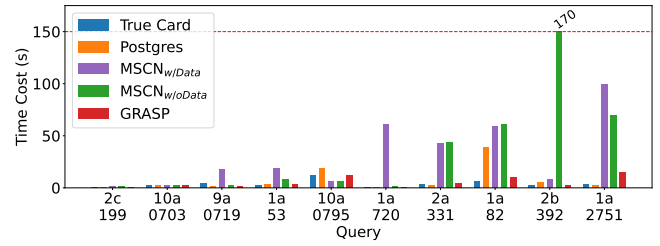
CardEst dates back to the early days of query optimization [39, 49]. Early methods relied on data statistics, such as histograms [12, 24, 25, 42], assuming uniformity within buckets and independence across columns. However, it often results in significant estimation errors [30, 36]. Such techniques were refined to use queries themselves to compute histograms [2, 8, 38], query expression statistics [7] and adjustments to correlated predicates [40]

Recently, CardEst has been approached as an ML problem. ML-based CardEst methods are categorized into *data-driven* and *query-driven* models (with a few exceptions of hybrid approaches [37, 54, 60]). Data-driven techniques [11, 27, 28, 33, 58, 59] build models of the data distribution by scanning the underlying data.

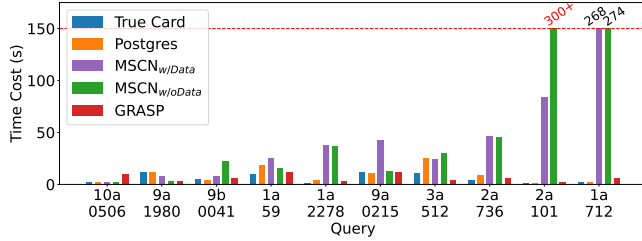
Query-driven techniques fall into two main categories. The first constructs a *data model* based on observed queries and their cardinalities [2, 29, 45]. Although these methods generally do not require data access, they often perform worse than regression-based models due to limited capacity. The second category, initiated by [4–6]



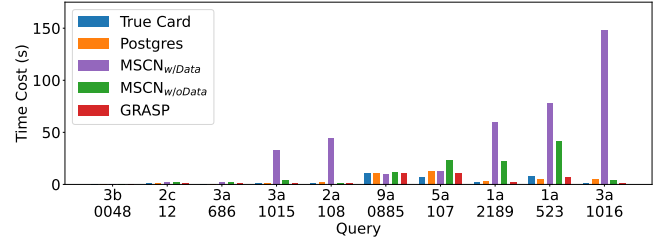
(a) CEB-IMDb-full - Workload 1



(b) CEB-IMDb-full - Workload 2



(c) CEB-IMDb-full - Workload 3



(d) CEB-IMDb-full - Workload 4

Figure 12: Per-query running time performance on CEB-IMDb-full

(without join support), learns a *regression model* from query features to cardinality. Recent deep learning-based approaches [16, 35] in this category support joins and demonstrate impressive empirical performance. However, they require data access for improved generalizability and assume training workloads with complete and balanced join templates.

9 CONCLUSIONS

We introduced a *new problem setting* for CardEst: *data-agnostic cardinality learning from imperfect workloads*, grounded in real-world scenarios and a detailed analysis of production workloads. To solve this challenging problem, we developed GRASP, a truly *data-agnostic* CardEst system that handles incomplete and imbalanced join templates through its *compositional design*. Additionally, we proposed a query-driven CardEst model to address value distribution shifts for range predicates, and a novel *learned count sketch* model that efficiently captures join correlations across base relations. We empirically validated the generalizability and robustness of GRASP using three database instances.

ACKNOWLEDGMENTS

Components of this work were funded by NSF grant DBI-2400135.

REFERENCES

- [1] MSCN code. <https://github.com/andreaskipf/learnedcardinalities>. Last Accessed Date: 2025-04-05.
- [2] ABOULNAGA, A., AND CHAUDHURI, S. Self-tuning histograms: building histograms without looking at data. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data* (1999), ACM, pp. 181–192.
- [3] ALON, N., GIBBONS, P. B., MATIAS, Y., AND SZEGEDY, M. Tracking join and self-join sizes in limited storage. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (1999), pp. 10–20.
- [4] ANAGNOSTOPOULOS, C., AND TRIANTAFILLOU, P. Learning Set Cardinality in Distance Nearest Neighbours. In *2015 IEEE International Conference on Data Mining* (Atlantic City, NJ, USA, 2015), IEEE, pp. 691–696.
- [5] ANAGNOSTOPOULOS, C., AND TRIANTAFILLOU, P. Learning to accurately COUNT with query-driven predictive analytics. In *2015 IEEE International Conference on Big Data (Big Data)* (2015), IEEE, pp. 14–23.
- [6] ANAGNOSTOPOULOS, C., AND TRIANTAFILLOU, P. Query-Driven Learning for Predictive Analytics of Data Subspace Cardinality. *ACM Transactions on Knowledge Discovery from Data* 11, 4 (2017), 1–46.
- [7] BRUNO, N., AND CHAUDHURI, S. Exploiting statistics on query expressions for optimization. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (2002), ACM, pp. 263–274.
- [8] BRUNO, N., CHAUDHURI, S., AND GRAVANO, L. STHoles: a multidimensional workload-aware histogram. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data* (2001), ACM, pp. 211–222.
- [9] CHEN, C. M., AND ROUSSOPOULOS, N. Adaptive selectivity estimation using query feedback. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data* (1994), ACM, pp. 161–172.
- [10] COFFMAN, E. G., GALAMBOS, G., MARTELLO, S., AND VIGO, D. Bin packing approximation algorithms: Combinatorial analysis. *Handbook of Combinatorial Optimization: Supplement Volume A* (1999), 151–207.
- [11] DEEDS, K. B., SUCIU, D., AND BALAZINSKA, M. SafeBound: A Practical System for Generating Cardinality Bounds. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.
- [12] DESHPANDE, A., GAROFALAKIS, M., AND RASTOGI, R. Independence is good: Dependency-based histogram synopses for high-dimensional data. *ACM SIGMOD Record* 30, 2 (2001), 199–210.
- [13] DING, B., CHAUDHURI, S., GEHRKE, J., AND NARASAYYA, V. DSB: a decision support benchmark for workload-driven and traditional database systems. *Proceedings of the VLDB Endowment* 14, 13 (2021), 3376–3388.
- [14] DURKAN, C., BEKASOV, A., MURRAY, I., AND PAPAMAKARIOS, G. Neural spline flows. In *Advances in Neural Information Processing Systems* 32 (2019), pp. 7509–7520.
- [15] DURRETT, R. *Probability: theory and examples*, fifth edition ed. No. 49 in Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, Cambridge ; New York, NY, 2019.
- [16] DUTT, A., WANG, C., NARASAYYA, V., AND CHAUDHURI, S. Efficiently approximating selectivity functions using low overhead regression models. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2215–2228.
- [17] DUTT, A., WANG, C., NAZI, A., KANDULA, S., NARASAYYA, V., AND CHAUDHURI, S. Selectivity estimation for range predicates using lightweight models. *Proceedings of the VLDB Endowment* 12, 9 (2019), 1044–1057.
- [18] DWORC, C. Differential privacy. In *International colloquium on automata, languages, and programming* (2006), Springer, pp. 1–12.
- [19] FRITSCH, F. N., AND CARLSON, R. E. Monotone Piecewise Cubic Interpolation. *SIAM Journal on Numerical Analysis* 17, 2 (1980), 238–246.
- [20] GADEPALLY, V., CHEN, P., DUGGAN, J., ELMORE, A., HAYNES, B., KEPNER, J., MADDEN, S., MATTSO, T., AND STONEBRAKER, M. The bigdawg polystore system and architecture. In *2016 IEEE High Performance Extreme Computing Conference*

- (HPEC) (2016), IEEE, pp. 1–6.
- [21] GERMAIN, M., GREGOR, K., MURRAY, I., AND LAROCHELLE, H. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning* (2015), PMLR, pp. 881–889.
 - [22] GIANNAKOURIS, V. Building learned federated query optimizers. In *Proceedings of the VLDB 2022 PhD Workshop co-located with the 48th International Conference on Very Large Databases* (2022), vol. 3186 of *CEUR Workshop Proceedings*.
 - [23] GREGORY, J. A., AND DELBOURGO, R. Piecewise Rational Quadratic Interpolation to Monotonic Data. *IMA Journal of Numerical Analysis* 2, 2 (1982), 123–130.
 - [24] GUNOPULOS, D., KOLLIOS, G., TSOTRAS, V. J., AND DOMENICONI, C. Approximating multi-dimensional aggregate range queries over real attributes. *ACM SIGMOD Record* 29, 2 (2000), 463–474.
 - [25] GUNOPULOS, D., KOLLIOS, G., TSOTRAS, V. J., AND DOMENICONI, C. Selectivity estimators for multidimensional range queries over real attributes. *The VLDB Journal* 14, 2 (2005), 137–154.
 - [26] HAN, Y., WU, Z., WU, P., ZHU, R., YANG, J., TAN, L. W., ZENG, K., CONG, G., QIN, Y., PRADLER, A., QIAN, Z., ZHOU, J., LI, J., AND CUI, B. Cardinality estimation in DBMS: a comprehensive benchmark evaluation. *Proceedings of the VLDB Endowment* 15, 4 (2021), 752–765.
 - [27] HEDDES, M., NUNES, I., GIVARGIS, T., AND NICOLAU, A. Convolution and Cross-Correlation of Count Sketches Enables Fast Cardinality Estimation of Multi-Join Queries. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–26.
 - [28] HILPRECHT, B., SCHMIDT, A., KULESSA, M., MOLINA, A., KERSTING, K., AND BINNING, C. DeepDB: learn from data, not from queries! *Proceedings of the VLDB Endowment* 13, 7 (2020), 992–1005.
 - [29] HU, X., LIU, Y., XIU, H., AGARWAL, P. K., PANIGRAHI, D., ROY, S., AND YANG, J. Selectivity Functions of Range Queries are Learnable. In *Proceedings of the 2022 ACM SIGMOD International Conference on Management of Data* (2022), ACM, pp. 959–972.
 - [30] IOANNIDIS, Y. E., AND CHRISTODOULAKIS, S. On the propagation of errors in the size of join results. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of data* (1991), pp. 268–277.
 - [31] IOANNIDIS, Y. E., AND CHRISTODOULAKIS, S. Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM Transactions on Database Systems* 18, 4 (1993), 709–748.
 - [32] JINDAL, A., AND LEEKA, J. Query Optimizer as a Service: An Idea Whose Time Has Come! *ACM SIGMOD Record* 51, 3 (2022), 49–55.
 - [33] KIM, K., LEE, S., KIM, I., AND HAN, W.-S. ASM: Harmonizing Autoregressive Model, Sampling, and Multi-dimensional Statistics Merging for Cardinality Estimation. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–27.
 - [34] KIPF, A., FREITAG, M., VORONA, D., BONCZ, P., NEUMANN, T., AND KEMPER, A. Estimating filtered group-by queries is hard: Deep learning to the rescue. In *1st International Workshop on Applied AI for Database Systems and Applications* (2019).
 - [35] KIPF, A., KIPF, T., RADKE, B., LEIS, V., BONCZ, P. A., AND KEMPER, A. Learned cardinalities: Estimating correlated joins with deep learning. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019* (2019).
 - [36] LEIS, V., GUBICHEV, A., MIRCHEV, A., BONCZ, P., KEMPER, A., AND NEUMANN, T. How good are query optimizers, really? *Proceedings of the VLDB Endowment* 9, 3 (2015), 204–215.
 - [37] LI, P., WEI, W., ZHU, R., DING, B., ZHOU, J., AND LU, H. ALECE: An Attention-based Learned Cardinality Estimator for SPJ Queries on Dynamic Workloads. *Proceedings of the VLDB Endowment* 17, 2 (2023), 197–210.
 - [38] LIM, L., WANG, M., AND VITTER, J. S. SASH: A self-adaptive histogram set for dynamically changing workloads. In *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003* (2003), pp. 369–380.
 - [39] LYNCH, C. A. Selectivity estimation and query optimization in large databases with highly skewed distribution of column values. In *Proceedings of the VLDB Endowment* (1988), pp. 240–251.
 - [40] MARKL, V., LOHMAN, G. M., AND RAMAN, V. LEO: An autonomic query optimizer for DB2. *IBM Systems Journal* 42, 1 (2003), 98–106.
 - [41] MOERKOTTE, G., NEUMANN, T., AND STEIDL, G. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proceedings of the VLDB Endowment* 2, 1 (2009), 982–993.
 - [42] MURALIKRISHNA, M., AND DEWITT, D. J. Equi-depth multidimensional histograms. In *SIGMOD* (1988), pp. 28–36.
 - [43] NEGI, P., MARCUS, R., KIPF, A., MAO, H., TATBUL, N., KRASKA, T., AND ALIZADEH, M. Flow-loss: learning cardinality estimates that matter. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2019–2032.
 - [44] NEGI, P., WU, Z., KIPF, A., TATBUL, N., MARCUS, R., MADDEN, S., KRASKA, T., AND ALIZADEH, M. Robust Query Driven Cardinality Estimation under Changing Workloads. *Proceedings of the VLDB Endowment* 16, 6 (2023), 1520–1533.
 - [45] PARK, Y., ZHONG, S., AND MOZAFARI, B. QuickSel: Quick Selectivity Learning with Mixture Models. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (2020), ACM, pp. 1017–1033.
 - [46] POESS, M., SMITH, B., KOLLAR, L., AND LARSON, P. TPC-DS, taking decision support benchmarking to the next level. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (2002), ACM, pp. 582–587.
 - [47] RIVLIN, T. J. *An introduction to the approximation of functions*, unabridged republ. of the 1981 dover reprint ed. Dover phoenix editions. Dover Publ, 2003.
 - [48] RUSU, F., AND DOBRA, A. Sketches for size of join estimation. *ACM Transactions on Database Systems* 33, 3 (2008), 1–46.
 - [49] SELINGER, P. G., ASTRAHAN, M. M., CHAMBERLIN, D. D., LORIE, R. A., AND PRICE, T. G. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD international conference on Management of data* (1979), ACM Press, p. 23.
 - [50] STEFFEN, M. A simple method for monotonic interpolation in one dimension. *Astronomy and Astrophysics, Vol. 239, NO. NOV (II), P. 443, 1990* 239 (1990), 443.
 - [51] STILLGER, M., LOHMAN, G. M., MARKL, V., AND KANDIL, M. LEO - db2’s learning optimizer. In *Proceedings of the VLDB Endowment* (2001), pp. 19–28.
 - [52] THABTAH, F., HAMMOUD, S., KAMALOV, F., AND GONSALVES, A. Data imbalance in classification: Experimental evaluation. *Information Sciences* 513 (2020), 429–441.
 - [53] TOKMAKOV, P., WANG, Y.-X., AND HEBERT, M. Learning Compositional Representations for Few-Shot Recognition. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), IEEE, pp. 6371–6380.
 - [54] WU, P., AND CONG, G. A Unified Deep Model of Learning from both Data and Queries for Cardinality Estimation. In *Proceedings of the 2021 International Conference on Management of Data* (2021), ACM, pp. 2009–2022.
 - [55] WU, P., AND IVES, Z. G. Modeling Shifting Workloads for Learned Database Systems. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–27.
 - [56] WU, P., XU, H., MARCUS, R., AND IVES, Z. G. A Practical Theory of Generalization in Selectivity Learning. *Proceedings of the VLDB Endowment* 18, 6 (2025).
 - [57] WU, Z., NEGI, P., ALIZADEH, M., KRASKA, T., AND MADDEN, S. FactorJoin: A New Cardinality Estimation Framework for Join Queries. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–27.
 - [58] YANG, Z., KAMSETTY, A., LUAN, S., LIANG, E., DUAN, Y., CHEN, X., AND STOICA, I. NeuroCard: one cardinality estimator for all tables. *Proceedings of the VLDB Endowment* 14, 1 (2020), 61–73.
 - [59] YANG, Z., LIANG, E., KAMSETTY, A., WU, C., DUAN, Y., CHEN, X., ABBEEL, P., HELLERSTEIN, J. M., KRISHNAN, S., AND STOICA, I. Deep Unsupervised Cardinality Estimation. *Proceedings of the VLDB Endowment* 13, 3 (2020), 279–292.
 - [60] ZHANG, K., WANG, H., LU, Y., LI, Z., SHU, C., YAN, Y., AND YANG, D. Duet: Efficient and Scalable Hybrid Neural Relation Understanding. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)* (2024), IEEE, pp. 56–69.