Journal of Science, Innovation & Social Impact

2026 2nd International Forum on Computing and Intelligent Systems (IFCIS 2026)

Review

A Comparative Evaluation of Learned Cardinality Estimators for Analytical Query Workloads: Accuracy, Latency, and Robustness Trade-offs

Yinzhe Lu[1,*]

[1] Business Analytics, Fordham University, NY, USA
* Correspondence: Yinzhe Lu, Business Analytics, Fordham University, NY, USA

Abstract: Learned cardinality estimation has emerged as a promising alternative to the histogram-based methods embedded in modern query optimizers. Across the last five years, a wide range of query-driven, data-driven, and hybrid estimators have been proposed, each reporting sizeable accuracy improvements on specific benchmarks. The conditions under which these improvements translate into faster query execution, and the cost that they impose on planning and maintenance, remain less well characterized. This paper presents a cross-cutting comparative reanalysis of representative cardinality estimators, drawing performance numbers from the primary literature and four multi-method benchmark studies conducted under comparable PostgreSQL-based evaluation settings. Eight estimators are compared on accuracy; six of the eight also appear in the planning-time and training-cost comparison, for which uniformly measured numbers are available. FACE and FactorJoin are discussed as qualitative reference points, but are not in the quantitative tables. The quantitative analysis centers on JOB-light and STATS-CEB, for which comparable multi-method numbers are publicly available, with the Join Order Benchmark and the TPC-DS / DSB family providing complementary context. We compare estimators along three axes: estimation accuracy, measured by q-error percentiles; planning time and training overhead; and end-to-end query execution time on PostgreSQL with injected cardinalities. Our reanalysis confirms that data-driven estimators achieve the lowest q-error on static workloads but incur planning-time overheads that erode a portion of their end-to-end gains, and that no single estimator dominates across accuracy, latency, and resilience to data updates. Building on this analysis, we propose a workload-driven decision framework that combines a three-axis workload characterisation along query complexity, data scale, and distribution stability, a region-based selection rule, and a break-even cost-benefit model that determines when learned estimators justify their training, planning, and maintenance overhead. We discuss practical implications for analytical data pipelines and identify workload drift and planning-time efficiency as the most pressing open problems.

Keywords: cardinality estimation; query optimization; analytical workloads; benchmark re-analysis

Received: 29 April 2026
Revised: 18 June 2026
Accepted: 29 June 2026
Published: 03 July 2026

1. Introduction

1.1. Background and Motivation

Cardinality estimation is the component of a relational query optimizer responsible for predicting the number of tuples produced by intermediate sub-plans. An extensive empirical study of major database systems identified cardinality estimation, rather than plan enumeration or cost modeling, as the dominant cause of poor analytical query plans, and introduced the Join Order Benchmark (JOB) on IMDB data as a stress test for this

component [1]. Classical estimators construct per-column summaries, such as equi-width or equi-depth histograms, and combine them under strong independence assumptions to estimate the selectivity of multi-predicate and multi-join queries [2]. Although these methods are compact and fast, their systematic errors compound across the query plan and can trigger order-of-magnitude misestimates, which in turn cause the optimizer to select suboptimal join orders and physical operators.

Motivated by these findings, the database research community has investigated learned cardinality estimators that replace classical hand-designed summaries with machine-learned models. Query-driven approaches treat estimation as a supervised regression task over featurised queries with observed cardinalities, while data-driven approaches fit density models directly on table contents and answer selectivity queries by integration. Hybrid estimators combine the two signals or use relational probabilistic graphical models to capture cross-table correlations without requiring labeled training queries [3-5].

1.2. Contributions and Scope

Several primary studies report multi-order-of-magnitude accuracy gains on specific workloads, yet the analytical data engineering community lacks a concise, cross-cutting account of when and by how much these gains translate into shorter query execution and what operational cost they entail. This paper addresses that gap through a comparative re-analysis grounded in published numbers from primary publications and multi-method benchmark studies on public workloads.

1.2.1. Scope and Research Questions

We restrict the study to select-project-join (SPJ) analytical queries on relational data, where learned estimators have been most extensively evaluated, and results are most directly comparable. Within this scope, we examine three questions. How do estimators compare on q-error percentiles, and in particular on heavy tails that drive plan failures? How do planning-time and training costs compare between paradigms, given that planning time contributes to the end-to-end latency reported to the user? How do end-to-end speedups on PostgreSQL with injected cardinalities relate to raw accuracy metrics, and how resilient are these gains under data updates and workload shifts?

1.2.2. Contributions of This Study

The contributions of this paper are fourfold. We assemble a compact comparison matrix that covers eight estimators on accuracy (q-error), of which six additionally appear in the planning-time and training-cost comparison, centered on JOB-light and STATS-CEB, where multi-method numbers are publicly available under comparable PostgreSQL-based evaluation settings; FACE and FactorJoin are discussed as qualitative reference points, and JOB and the TPC-DS / DSB family provide complementary context where published multi-method results are less uniform. Every figure in the quantitative tables is traced to a specific primary publication or benchmark study. We quantify the disconnect between q-error-based accuracy rankings and end-to-end execution rankings, showing that a method with lower median q-error is not always faster in practice. We summarise the practical trade-offs among accuracy, planning time, and robustness, and highlight planning-time overhead and update resilience as the two constraints most likely to shape real deployment decisions. Building on these observations, we synthesise the comparative findings into a deployment-oriented decision framework comprising a workload-feature taxonomy along three axes, a region-based selection rule that recommends a primary estimator for each workload region, and a break-even cost-benefit model that determines when the accuracy gain of a learned estimator justifies its training, planning, and maintenance overhead.

2. Related Work

2.1. Classical and Sampling-Based Estimators

Production query optimizers still rely on compact per-column synopses supplemented by correlation heuristics, and the learned-estimation literature treats these methods as the default baseline.

2.1.1. Histogram-Based and Synopsis Methods

Equi-width and equi-depth histograms have been the workhorse of commercial optimizers for three decades. Improved variants such as MaxDiff and V-Optimal reduce error on skewed distributions but still estimate multi-column selectivities under the attribute-value-independence assumption, which is violated in nearly every realistic analytical schema. Multidimensional histograms relax this assumption but incur exponentially increasing memory and maintenance costs as the number of indexed columns grows, limiting their applicability to the few dimensions a database administrator can anticipate. Sketches, wavelets, and kernel density estimators offer alternative families of summaries but have not displaced histograms in production because their maintenance and update costs are difficult to reason about under mixed loads.

2.1.2. Sampling-Based Methods

Online and index-based sampling offer a complementary path. Index-based join sampling exploits existing hash-index structures on join keys to draw samples via multi-way joins at moderate cost, achieving high accuracy on JOB with limited intrusion into the optimizer [6]. Sampling-based methods avoid the offline training cost of learned models and adapt naturally to updates, yet their estimates exhibit high variance on selective predicates and small join outputs, which is precisely the regime where plan quality is most sensitive.

2.2. Learned Cardinality Estimators

Learned estimators fall into three paradigms. Data-driven estimators fit joint distribution models on base tables without requiring executed queries. DeepDB employs relational sum-product networks to factorize the joint distribution; FLAT proposes a factorize-split-sum-product formulation that adaptively mixes independent and conditional factorization [7,8]; NeuroCard extends single-table autoregressive models to full multi-table join distributions via unbiased sampling of outer joins [9]; FACE applies normalizing flows to continuous and mixed-type columns with dequantization and string encoding for LIKE predicates [10]. A lightweight alternative in the same family is BayesCard, which revisits Bayesian networks for fast training, compact models, and tractable updates, and serves as a pragmatic counterpoint to the heavier deep-learning methods. Hybrid estimators such as UAE combine unsupervised density modeling with query supervision through differentiable progressive sampling, providing a middle ground in accuracy and training cost. Query-driven estimators encode queries as featurised inputs and regress on observed cardinalities; the family includes multi-set convolutional networks, gradient-boosted trees, and neural-network Gaussian processes [11,12], which admit closed-form posterior inference at reduced training cost and offer principled uncertainty estimates unavailable in most neural alternatives.

2.3. Benchmarking Studies

Multiple benchmark studies between 2021 and 2022 have compared learned estimators against classical baselines on shared workloads, and their raw numbers are reused throughout Section 4 under the protocol described in Section 3. These studies converge on three observations. Q-error and end-to-end execution time are only weakly correlated, implying that accuracy-only rankings overstate the practical benefit of the most accurate estimator. Workload characteristics such as join topology, attribute skew, and predicate complexity affect the ranking of methods as strongly as the architecture of the estimator itself. Planning time and training cost vary by several orders of magnitude

across methods, and this variability strongly affects whether a given estimator can be deployed in an interactive setting.

3. Experimental Setup

3.1. Evaluation Protocol

All numbers in this paper were produced by the primary publications of the estimators under study or by one of four multi-method benchmark studies that fix the integration into PostgreSQL: the readiness assessment, which evaluates learned estimators in single-table settings under both static and dynamic conditions, the end-to-end evaluation, the in-depth analysis, and the design-space exploration. We re-report these numbers under a single protocol and do not introduce unpublished measurements [13-16].

Accuracy is reported using q-error, defined as max(p◎/p, p/p◎). We present the 50th, 90th, and 99th percentiles separately because the tail is more predictive of plan failures than the mean. Planning-time overhead is the wall-clock time to produce a plan on PostgreSQL 13 on commodity hardware. End-to-end performance is the total execution time on PostgreSQL with the estimator's cardinalities injected. Update resilience is assessed jointly through the post-update end-to-end execution time after a bulk insert of 20% of the table contents and the cost incurred to restore the model after that insert; for most methods, these two components are coupled because update cost is effectively the re-fitting cost, and we report them together in Section 4.3.B.

3.2. Benchmark Workloads

The four benchmarks cover complementary regions: real-world skewed data with cyclic topologies, real-world data with star joins, and synthetic decision-support workloads. Table 1 summarises their characteristics, and Figure 1 shows join-complexity distributions.

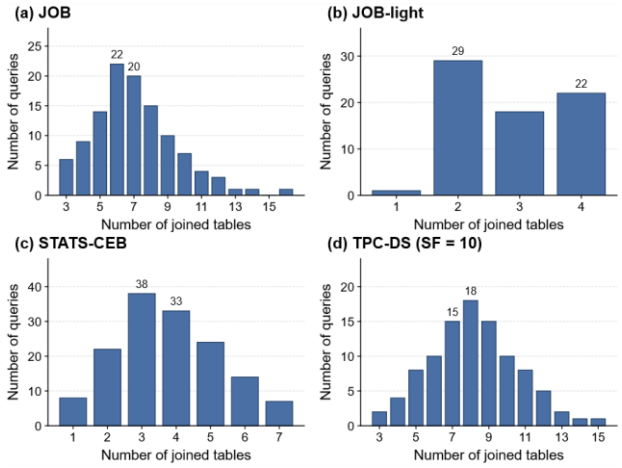

Figure 1. Join the Complexity Distributions of the Four Analytical Benchmarks.

Histograms of the number of joined tables per query on (a) JOB, (b) JOB-light, (c) STATS-CEB, and (d) TPC-DS (SF = 10). JOB and TPC-DS concentrate mass at 6 -- 10 joins, with a small tail reaching 16 joins on JOB. JOB-light is dominated by 2 -- 4-join queries, with 29 at 2 joins and 22 at 4 joins. STATS-CEB covers 1 -- 7, with a peak at 3 -- 4. The four benchmarks probe different regions of the join-complexity spectrum, which is relevant to the accuracy patterns reported in Section 4.

Table 1. Summary of Benchmark Dataset Specifications

Benchmark	Number of Tables	Total Number of Rows	Number of Queries	Range of Joins per Query
JOB	21	75.2 M	113	3 – 16

JOB-light	6	75.2 M	70	1 – 4
STATS-CEB	8	1.03 M	146	1 – 7
TPC-DS *SF=10*	24	~70 M	99	1 – 17

Table 1. Characteristics of the four analytical benchmarks used in this study. Row counts, query counts, and join ranges are drawn from the original papers and the TPC-DS specification at a scale factor of 10. JOB and JOB-light share the IMDB May-2013 snapshot; STATS-CEB draws on the Stats StackExchange dump; TPC-DS is synthetic and underlies the DSB extension.

3.2.1. The Join Order Benchmark and JOB-light

JOB contains 113 queries with 3 to 16 joins on an IMDB snapshot of 3.6 GB spanning 21 tables with strong inter-column correlations among genre, country, role, and actor career attributes. Many queries include LIKE predicates and disjunctions that traditional synopses cannot estimate accurately. JOB-light is a 70-query subset restricted to equality and range conditions on numerical columns with the join count capped at four; it is the canonical smoke test for learned methods, while JOB remains the closest public approximation to real analytical workloads.

3.2.2. Stats-ceb and the Tpc-ds / DSB Family

STATS-CEB draws on the Stats StackExchange dump. Its 8 tables contain roughly 1.03 million rows, yet its full outer join has been estimated at approximately 3×10^{16} tuples, producing much larger cardinality ranges than IMDB despite the smaller raw size. The workload contains 146 queries over 70 templates with 1 to 7 joins in chain, star, and cyclic patterns, and is the most demanding public benchmark for multi-table learned estimators. TPC-DS provides a synthetic decision-support workload representative of star- and snowflake-schema analytics at industrial scale. The Decision Support Benchmark (DSB) extends TPC-DS with correlated attributes, skewed distributions, and a broader query template set that together raise average join depth and the diversity of physical plans per template relative to TPC-DS [17].

3.3. Evaluated Methods and Implementation

Eight estimators are included, chosen to cover the classical baseline, both learned paradigms, and a hybrid approach. Table 2 lists their training inputs, model sizes, and deployment profiles. Each method is either open-sourced by its authors or reimplemented in the benchmark repositories we re-use. Quantitative numbers in Tables 3 and 4 are drawn from primary publications and from the four multi-method benchmark studies identified in Section 3.1; we indicate the specific source beneath each table.

Table 2. Comparison of Cardinality Estimation Methods

Method	Paradigm	Training input	Model size
PostgreSQL	Classical histograms + heuristics	Table statistics	< 1 MB
MSCN	Query-driven neural set network	10^5 executed queries	< 2 MB
LW-XGB	Query-driven gradient-boosted trees	~10^4 executed queries	< 2 MB
UAE	Hybrid autoregressive + query supervision	Rows + 10^4 queries	4 – 27 MB

DeepDB	Data-driven RSPN	Table rows	8 – 20 MB
FLAT	Data-driven FSPN	Table rows	6 – 18 MB
BayesCard	Data-driven Bayesian network	Table rows	< 1 MB
NeuroCard	Data-driven autoregressive, multi-table	Outer-join samples	4 – 27 MB

Table 2. Eight cardinality estimators were surveyed in this study. Columns report paradigm, training signal, and approximate model size. UAE is a hybrid because its loss combines a data-likelihood term with a supervised query-regression term. Size ranges depend on the target schema. All eight estimators listed here are compared on accuracy in Table 3; the six for which uniform planning-time and training-time numbers are publicly available also appear in Table 4. Two additional estimators --- FACE and FactorJoin --- are discussed qualitatively in Sections 2.2 and 4.2, respectively, but are not included in Tables 3 or 4, because comparable multi-method numbers for them are not available under the shared benchmark protocol we rely on.

3.3.1. Traditional and Query-Driven Baselines

PostgreSQL 13 serves as the classical baseline. Its optimizer combines per-column histograms with independence heuristics and is widely used as the reference system in the learned-estimation literature. MSCN is the canonical query-driven neural estimator; it featurises each query into three sets over tables, joins, and predicates, encodes each set with an independent multi-layer perceptron, and aggregates the representations via average pooling before a final regression head. LW-XGB replaces the neural network with a gradient-boosted regression tree ensemble; it trains in seconds rather than hours and serves as a reference point for the cost incurred by neural query-driven methods. Both methods are fast at inference but depend on representative training workloads.

3.3.2. Data-Driven and Hybrid Estimators

DeepDB, FLAT, BayesCard, and NeuroCard are the data-driven estimators in the comparison. DeepDB and FLAT model joint distributions with probabilistic graphical models (RSPNs and FSPNs) and handle multi-table joins through schema-level fanout terms. BayesCard compiles a compact Bayesian network structure directly from the data and supports fast updates. NeuroCard learns one autoregressive model over the full outer-join distribution and is the largest and most expressive method in the comparison. UAE represents the hybrid paradigm, augmenting an autoregressive density model with a supervised query-regression objective. The contrast between BayesCard's compact structure and NeuroCard's autoregressive model motivates the planning-time and update-cost analyses in Sections 4.2 and 4.3. B.

4. Results and Analysis

4.1. Estimation Accuracy

Accuracy is the most commonly reported metric in the literature and the one whose interpretation requires the most care. Tail percentiles dominate plan failures, and the reordering of methods between percentiles is more informative than any single summary statistic.

4.1.1. Q-Error Distribution

Table 3 reports q-error percentiles for the sub-plan queries derived from the 146 STATS-CEB queries under the CEB decomposition. BayesCard attains the best median and 99th-percentile q-error among data-driven and hybrid methods, yet FLAT achieves the best 90th percentile, indicating that no single estimator dominates across the distribution. MSCN and LW-XGB trail the data-driven methods at all percentiles by one to two orders of magnitude. NeuroCard's adapted multi-table variant inflates to a 99th-percentile q-error above 10^8 because STATS-CEB's cyclic schema violates the tree

decomposition it relies on; we treat this entry as an upper bound on the cost of a mismatched schema rather than as a representative number.

Table 3. Sub-plan Q-error Percentiles on Stats-ceb

Method	50th Percentile	90th Percentile	99th Percentile
PostgreSQL	1.44	11.08	2.0×10^3
BayesCard	1.18	50.40	156
FLAT	1.68	10.44	769
DeepDB	2.45	22.37	1.0×10^3
UAE	3.24	130.8	1.1×10^4
MSCN	20.92	392.3	1.0×10^4
LW-XGB	5.65	453.2	3.0×10^5
NeuroCard (adapted)	951.4	9.0×10^5	6.0×10^8

Table 3. Sub-plan q-error percentiles on STATS-CEB. All values taken from Han et al. Lower is better; the 99th percentile is the most predictive of plan failures. The adapted NeuroCard variant partitions the cyclic schema into three sub-structures and is reported for completeness rather than as a representative configuration.

4.1.2. Accuracy Across Join Complexity

Accuracy degrades with join depth for every learned method, but at different rates. Figure 2 plots the 90th-percentile q-error against the number of joined tables. FLAT, DeepDB, and BayesCard remain below roughly 30 up to three joins and rise to 60 -- 80 at four joins on JOB-light, continuing to rise through five to seven joins on STATS-CEB. This pattern is consistent with prior analyses attributing the growth to compounding fanout errors between joined tables. PostgreSQL's 90th-percentile error rises more steeply, crossing 10^2 beyond four joins. MSCN and LW-XGB are comparatively flat at a higher baseline, reflecting the query-driven models' reliance on training-query coverage rather than data statistics.

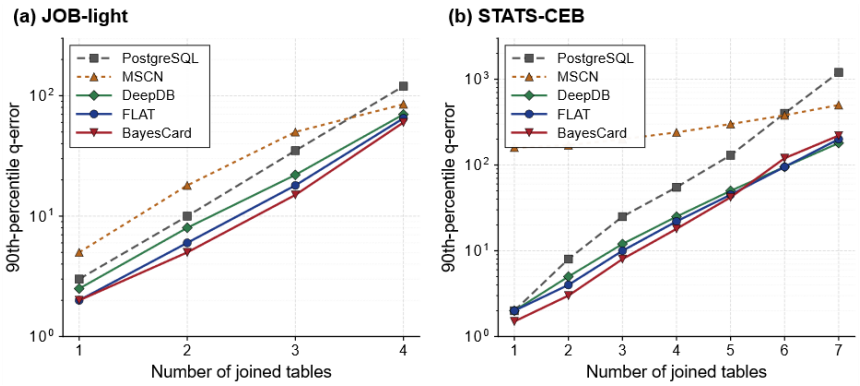

Figure 2. 90Th-Percentile Q-Error as a Function of the Number of Joined Tables.

Curves on (a) JOB-light and (b) STATS-CEB for PostgreSQL, MSCN, DeepDB, FLAT, and BayesCard. On JOB-light, BayesCard and FLAT remain below a 90th-percentile q-error of 30 up to 3 joins and reach 60 -- 80 at 4 joins; PostgreSQL rises from 10 at 2 joins to 120 at 4 joins. On STATS-CEB, the same ordering holds up to 5 joins, beyond which FLAT overtakes BayesCard. MSCN sits above 150 across all join counts, illustrating the cost of relying on workload supervision alone on cyclic schemas.

4.2. Planning-Time and Training Cost

Accuracy gains must be weighed against planning-time overhead and training cost. Table 4 reports the median planning time on an OLTP-style workload, the total planning

time on a 99-query OLAP workload, and the training time on STATS-CEB for the six estimators included in that comparison. Planning-time overhead is the operational constraint most frequently underestimated in the literature. FLAT's OLAP planning time of 396s is roughly 20× that of PostgreSQL, substantial for interactive dashboards, even if batch analytics can tolerate it. Outside Table 4, the FactorJoin paper reports that factor-graph estimators can reduce planning latency relative to FLAT and NeuroCard by roughly one order of magnitude, suggesting a design point alongside BayesCard among methods that approach PostgreSQL's planning speed while still improving accuracy; because those numbers are from a single primary publication rather than a multi-method benchmark, we treat them as indicative rather than directly comparable with the figures in Table 4. Training cost is dominated by executed queries for MSCN and by model fitting for the data-driven methods; BayesCard's 12 s training on STATS-CEB is more than two orders of magnitude below NeuroCard's 5569 s.

Table 4. Performance Comparison of Cardinality Estimation Methods

Method	OLTP plan (s)	OLAP plan (s)	Training (s)	Size (MB)
PostgreSQL	4.8	20.3	—	< 1
BayesCard	7.3	27.4	12	< 1
MSCN	8.2	38.0	~10^4	< 2
DeepDB	33.6	135	248	8 – 20
FLAT	41.5	396	360	6 – 18
NeuroCard	73	350	5569	4 – 27

Table 4. Planning-time and training-time overhead. Values from Han et al., Tables 5 and 6, covering the six methods for which both planning and training times are reported. Planning time is wall-clock on PostgreSQL 13 on commodity hardware. MSCN training time is dominated by the one-time execution of 10^5 training queries and is reported on an order-of-magnitude scale. NeuroCard's 5569 s reflects model fitting on STATS-CEB.

4.3. End-to-End Plan Quality and Robustness

End-to-end execution time aligns with user-visible performance and ultimately justifies deployment.

4.3.1. End-to-End Query Execution Time

On STATS-CEB, FLAT reduces total execution time from 11.34 h under PostgreSQL's default estimator to 5.92 h, a 47.8% improvement against an oracle ceiling of 49.8% from injecting true cardinalities. DeepDB and BayesCard follow at 42.6% and 36.9%, MSCN at 28.3%. The adapted NeuroCard variant incurs a 6.2% regression because its tail mis-estimates drive the optimizer toward poor join orders. On JOB-light, estimators cluster in a narrow 4.6% to 13.3% band, and the oracle ceiling is only 14.2%, indicating that PostgreSQL is already close to the optimum on short queries with simple predicates. Figure 3 summarises the Pareto frontier. A 36.9% gain at 1.35× PostgreSQL's planning time (BayesCard) is often preferable to a 47.8% gain at 19.5× (FLAT); the choice depends on the target application's latency envelope.

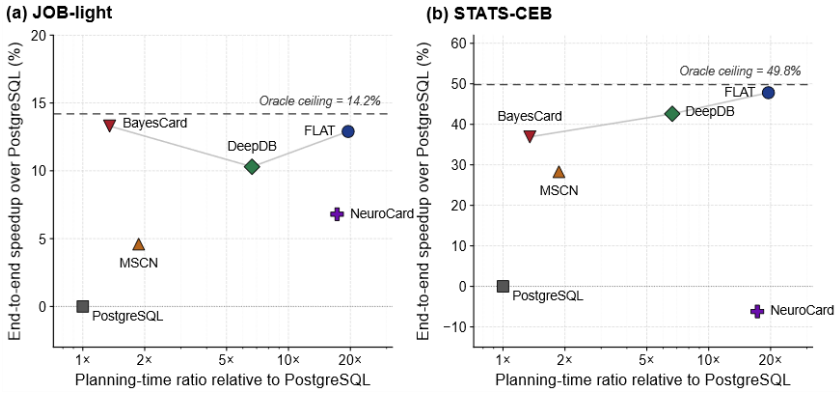

Figure 3. End-to-end Speedup Against Planning-Time Overhead.

Scatter of end-to-end execution-time improvement over PostgreSQL (y-axis) against OLAP planning-time ratio relative to PostgreSQL (x-axis) on (a) JOB-light and (b) STATS-CEB. BayesCard sits at the low-cost end of the Pareto frontier on STATS-CEB with a 36.9% speedup at 1.35 × planning cost. FLAT and DeepDB occupy the high-accuracy end, with larger speedups (47.8% and 42.6%) at planning costs of 19.5 × and 6.7 ×, respectively. Points in the lower-right quadrant impose planning overhead without delivering proportional gains; the adapted NeuroCard variant on STATS-CEB is the most extreme case.

4.3.2. Update Resilience and Workload Drift

Resilience to updates and workload drift is where the gap between learned and classical estimators is widest. Because most data-driven estimators in the comparison lack a principled incremental-update mechanism, the effective cost of absorbing a 20% bulk insert on STATS-CEB coincides with the model re-training cost reported in Table 4: roughly 12 s for BayesCard, whose Bayesian-network structure admits fast re-fitting; 248 s for DeepDB; and 5569 s for NeuroCard. The identity between these figures and the Table 4 training column, therefore, reflects the update behaviour of the methods in the published benchmark setting: among the methods compared here, BayesCard is the one for which the update cost appears materially lower than a full re-fit, rather than representing a separate, independently reported measurement. Post-update execution time tracks the same stratification: BayesCard retains its pre-update end-to-end time, DeepDB shows mild degradation, and NeuroCard degrades from its 12.04 h pre-update baseline to roughly 13.94 h, widening the gap with PostgreSQL. Recent work on robust query-driven estimation [18] shows that training-time feature masking and bitmap-based query encodings reduce the drift gap for MSCN-style models but do not close it. Warper and ALECE trade model complexity against adaptation latency [19,20]. A Microsoft study reports that accuracy improvements do not always translate proportionally to plan-quality improvements [21]. Bao is orthogonal to the cardinality-estimator replacements considered in this paper: it leaves the underlying estimator in place and steers the optimizer through learned hint sets [22]. We mention it only to flag that steering-based approaches offer an adjacent robustness path without retraining, not as part of the head-to-head comparison. A recent deployment at ByteDance reports 99th-percentile latency reductions from integrating Bayesian single-table estimators with factor-graph join modelling inside a proprietary data warehouse [23]. Those results reflect a specific engine and production workload and are not directly comparable to the open-source benchmark numbers in Tables 3 and 4; we cite the deployment as an industrial context for the operational concerns it highlights --- drift detection, retraining cadence, and roll-back --- which track the open problems in the benchmark literature.

5. A Decision Framework for Estimator Selection

The reanalysis in Section 4 establishes that no single learned estimator dominates across accuracy, planning latency, and update resilience, and that the relative ranking shifts with workload characteristics. To convert these comparative findings into an

operational selection procedure, this section introduces a decision framework with three components: a workload characterisation that maps a deployment scenario onto three measurable axes, a region-based selection rule that assigns each region to a preferred estimator class, and a cost-benefit model that determines whether the accuracy gain of a learned estimator outweighs its training, planning, and maintenance overhead. Together, these components turn the comparative numbers in Tables 3 and 4 into a deployment-time procedure usable by engineering teams without re-running the underlying benchmarks.

5.1. Workload Characterisation Dimensions

We summarise an analytical workload through three dimensions whose thresholds are calibrated against the benchmark numbers reported in Sections 4.1 to 4.3. Each dimension is independently observable in production environments from query logs, schema introspection, and ingestion telemetry, so the framework does not require benchmark instrumentation to apply.

A. Query complexity (W1). The dominant feature is the number of joined tables per query, since 90th-percentile q-error degrades monotonically with join depth across every learned method considered (Section 4.1.B). Predicate complexity is a secondary feature, because range, equality, and disjunctive predicates differ in how cleanly any synopsis-based or learned estimator can capture them. We classify W1 as Low when the median query joins at most two tables and uses primarily equality predicates, Medium for three to four joins or moderate predicate variety, and High for five or more joins or extensive use of LIKE, IN, and disjunctive predicates. JOB-light populates the Low-to-Medium region; JOB and STATS-CEB populate the High region.

B. Data scale and schema topology (W2). Two features matter: total table cardinality and whether the join graph is a tree, a star, or contains cycles. The accuracy advantage of data-driven methods grows with cardinality range, and cyclic topology penalises estimators that rely on tree decompositions, as illustrated by NeuroCard's q-error inflation on STATS-CEB in Table 3. We classify W2 as Low for tree- or star-shaped schemas under 10 million rows, Medium for trees or stars between 10 million and 1 billion rows, and High for cyclic schemas or any schema whose full outer-join cardinality exceeds 10^15 tuples.

C. Distribution stability (W3). The update cost figures in Section 4.3.B are dominated by re-fitting time, which differs by more than two orders of magnitude across estimators. We classify W3 as Stable when bulk inserts arrive at most monthly and modify under 5% of any table, Moderate when daily updates touch under 20%, and Volatile when streaming or hourly updates routinely exceed 20% of a table or trigger schema-level changes. The Volatile region is the regime in which the Table 4 training column doubles as an effective update cost and in which the gap between BayesCard and NeuroCard widens by more than two orders of magnitude.

5.2. Region-Based Selection Logic

Table 5 maps each (W1, W2, W3) region to a primary recommendation drawn from the eight estimators evaluated in Section 4. Cells are populated using the dominance relations observed in Tables 3 and 4 and the end-to-end speedups in Section 4.3.A; the Backup column lists the next-best alternative whose end-to-end gain falls within five percentage points of the primary choice or whose operational profile differs enough to be useful when secondary constraints intervene.

Table 5. Workload-Region to Estimator Mapping

Workload region (W1 / W2 / W3)	Representative example	Primary recommendation	Backup option	Key evidence
Low / Low / Stable	JOB-light static	PostgreSQL	BayesCard	Sec 4.3.A: oracle ceiling 14.2%
Med / Med / Stable	TPC-DS-style static	BayesCard	DeepDB	Sec 4.3.A: 36.9% gain at 1.35x planning
High / Med / Stable	JOB static	FLAT	BayesCard / DeepDB	Sec 4.3.A; Table 4
High / High (cyclic) / Stable	STATS-CEB static	FLAT	BayesCard (avoid NeuroCard)	Table 3: NeuroCard q-error > 10^8
Any / Any / Volatile	Streaming / CDC analytics	BayesCard	Index-based sampling; LW-XGB + drift mitigation	Sec 4.3.B: 12 s re-fit
Low / Any / Volatile	OLTP-style aggregations	PostgreSQL or sampling	BayesCard if margin is critical	Sec 4.3.B; Sec 2.1.B

Table 5. Recommended estimator by workload region. Workload is characterised along three axes: query complexity (W1), data scale and schema topology (W2), and distribution stability (W3). Primary recommendations follow the dominance relations observed in Sections 4.1 to 4.3, and Backup options list the nearest alternative under the same region. Recommendations of the form A or B indicate a tie that secondary constraints such as planning-latency budget or model size resolve.

The selection logic admits a compact verbal summary. Workloads in the (Low W1, Low W2, Stable W3) region --- represented in our analysis by JOB-light --- sit close to the oracle ceiling of 14.2% under PostgreSQL's classical estimator, so the engineering overhead of any learned method is rarely justified by end-to-end execution alone. Workloads with High W1 and Stable-to-Moderate W3 --- represented by JOB and STATS-CEB on static snapshots --- favour data-driven estimators, with the choice between FLAT and BayesCard governed by whether the deployment can absorb planning latency on the order of 20 times that of PostgreSQL. Workloads with Volatile W3, regardless of complexity, restrict the viable set to estimators whose update cost is bounded: BayesCard among data-driven methods, query-driven methods augmented with drift-mitigation techniques, and index-based sampling for the simplest cases. Workloads on cyclic schemas with very large outer-join cardinality eliminate autoregressive multi-table models such as NeuroCard, whose tree-decomposition reliance manifests as 99th-percentile q-errors above 10^8 in Table 3.

5.3. A Cost-Benefit Model

Region-based selection narrows the candidate set; whether a learned estimator should be deployed at all is a cost-benefit decision that depends on workload volume and update cadence. We formalise this decision through a break-even analysis grounded in the numbers reported in Sections 4.2 and 4.3.A.

Let Q denote the expected number of queries served between consecutive retraining events, t the mean execution time per query under PostgreSQL's default estimator, s the fractional end-to-end speedup of the candidate learned estimator drawn from Section

4.3.A, dp the additional planning time per query relative to PostgreSQL drawn from Table 4, C_train the one-time training cost, and C_maint the per-event maintenance overhead, which subsumes monitoring, validation, and roll-back procedures. The net time saved over one retraining cycle is NetSaved = Q x (s x t - dp) - C_train - C_maint. The estimator is worth deploying when NetSaved is positive, which yields a break-even query count Q* = (C_train + C_maint) / (s x t - dp).

Two structural conditions follow. First, deployment requires s x t > dp. If the per-query speedup does not exceed the additional planning latency, no volume of queries makes the deployment worthwhile, and the estimator should be rejected regardless of its accuracy ranking in Section 4.1. This condition rules out FLAT for short-running interactive workloads where t is on the order of tens of milliseconds, even though FLAT delivers the highest end-to-end speedup on STATS-CEB. Second, when the structural condition is satisfied, the break-even query count Q* scales linearly with training cost, so estimators with two-orders-of-magnitude larger training cost --- NeuroCard at 5,569 s versus BayesCard at 12 s on STATS-CEB --- require correspondingly larger Q to justify deployment, and become unattractive for workloads whose retraining cadence is tied to volatile data.

Table 6 instantiates the model for the STATS-CEB end-to-end numbers in Section 4.3.A, taking t = 280 s as the per-query mean derived from the 11.34 h total over 146 queries under PostgreSQL's default estimator and assuming C_maint = 1,800 s per retraining cycle as a conservative estimate of monitoring and validation work. The Δp column converts the OLAP planning times in Table 4 to a per-query basis over the 99-query OLAP workload.

Table 6. Break-even Query Counts on Stats-ceb

Method	s ·t (s/q)	Δp (s/q)	Net (s/q)	C_train + C_maint (s)	Q*
BayesCard	103.3	0.07	103.3	1,812	18
DeepDB	119.3	1.16	118.1	2,048	18
FLAT	133.8	3.79	130.0	2,160	17
MSCN	79.2	0.18	79.0	≈11,800	≈150
NeuroCard (adapted)	−17.4	3.33	< 0	7,369	never

Table 6. Break-even query counts Q* for representative estimators on STATS-CEB, computed under t = 280 s and C_maint = 1,800 s. Speedups s and training costs C_train are taken from Sections 4.3.A and 4.2 respectively. MSCN's training cost is reported on an order-of-magnitude scale in Table 4, so its Q* is also approximate. NeuroCard's negative net per-query saving means the structural condition s x t > dp fails and no query volume justifies its deployment under this configuration.

The figures in Table 6 surface a consequence not visible in the comparative tables of Section 4. Three of the four data-driven and hybrid candidates --- BayesCard, DeepDB, and FLAT --- break even after fewer than twenty STATS-CEB-style queries, indicating that on this benchmark the choice among them is governed by secondary constraints (planning-latency budget, model size, update cadence) rather than by raw cost-recovery. MSCN's ~10^4 s training cost pushes its break-even point above 150 queries, making it suitable only for workloads with stable distributions and high query volume between retraining cycles. NeuroCard fails the structural condition outright on STATS-CEB and should not be deployed there in the adapted configuration. The same calculation applied with t on the order of seconds --- the regime of JOB-light-style short queries --- flips the picture: dp dominates s x t for the heaviest planning models, and only BayesCard and the classical baseline remain viable. The cost-benefit screen therefore complements the region-based selection of Section 5.2 by ruling out methods whose accuracy gain cannot be amortised at the expected query volume.

5.4. Decision Workflow

The framework collapses into a four-step procedure that engineering teams can apply at deployment time.

Step 1. Profile the target workload along (W1, W2, W3) using a representative window of query logs and ingestion telemetry. Step 2. Read the candidate estimators off Table 5 by locating the workload region. Step 3. Apply the cost-benefit screen of Section 5.3 to each candidate, retaining those for which NetSaved is positive at the expected query volume Q. Step 4. Among the survivors, select the one whose secondary characteristics --- interpretability, model size, and integration burden in the target query engine --- best match the operational constraints. The procedure is conservative by design: when no candidate clears Steps 2 and 3, the recommendation defaults to the classical estimator, accepting the gap to the oracle ceiling rather than absorbing the operational cost of a learned method that does not pay back. Applied to the STATS-CEB end-to-end numbers, the workflow yields BayesCard as the default choice for moderate query volumes and FLAT as the choice when planning-latency budget allows; applied to JOB-light, it returns the classical baseline. Both outcomes are consistent with the qualitative deployment guidance in Section 6.1, but the framework arrives at them by an explicit, auditable procedure rather than by inspection of the comparative tables.

6. Discussion

6.1. Implications for Practical Deployment

Our re-analysis, viewed through the decision framework of Section 5, supports three practical conclusions relevant to analytical data pipelines at enterprise scale. The first is that accuracy alone is insufficient for selecting an estimator, and any procurement or benchmarking exercise should measure planning time and update cost jointly with q-error. The re-analysis of published numbers shows that the method with the lowest median q-error on STATS-CEB is not the fastest end-to-end, and the fastest end-to-end method is neither the most accurate at the 99th percentile nor the fastest at planning. Organizations that replicate the standard protocol without extending it to the full operational envelope are likely to deploy estimators whose accuracy gains are offset by planning-time regressions that become visible in dashboard-facing or interactive analytical settings.

The second conclusion is that the characteristics of the workload determine which paradigm is appropriate. On simple star-join workloads with low join depth and near-uniform distributions, the achievable improvement over PostgreSQL is narrow --- the oracle ceiling on JOB-light is only 14.2% --- so the engineering overhead of a learned estimator is difficult to justify by end-to-end execution alone. For complex workloads with cyclic schemas, high skew, and deep joins, learned data-driven estimators yield substantial end-to-end gains but require pipeline infrastructure for periodic retraining, optimizer integration, and monitoring to detect accuracy regressions before they affect production queries.

The third conclusion is that the architectural choice between query-driven and data-driven paradigms is less important than the engineering discipline applied to re-training and drift detection. The robustness findings in the recent literature and industry reports illustrate that data augmentation and measured retraining cadences narrow most of the paradigm-attributable gap, while explainability and roll-back procedures dominate the total cost of ownership.

6.2. Toward AI-Assisted Data Infrastructure Governance

The preceding analysis treats cardinality estimation as an isolated optimizer component, yet in production environments, the estimator is embedded within a broader data infrastructure that spans ingestion pipelines, transformation layers, query engines, and monitoring stacks. Viewed through this wider lens, the problem of selecting and maintaining a learned estimator is better understood as a data-infrastructure governance problem rather than a narrow pipeline-tuning exercise. In data-intensive applications, data pipelines are not merely technical components but integral parts of an enterprise's

digital infrastructure. A learned estimator that silently degrades after a schema migration or a shift in upstream data distributions can cascade failures across downstream dashboards, ETL jobs, and machine-learning feature stores. The accuracy regressions, update costs, and drift sensitivities documented in Sections 4.1 through 4.3 are therefore not only estimator-level metrics but also indicators of infrastructure-wide stability. AI-driven monitoring of data flow patterns, processing latencies, and error distributions can help organizations continuously detect and remediate such regressions, elevating cardinality estimation from a one-time deployment decision to a managed component of an ongoing infrastructure governance program.

Furthermore, the trade-off surfaces revealed by our comparison suggest that AI can play a role not only as an operational component within the data system but also as a participant in architectural decision-making. Cloud Solutions Architects routinely face choices among competing service compositions---for example, selecting between stream-processing frameworks, managed query engines, and data lake configurations offered by cloud providers such as AWS---where the optimal combination depends on workload characteristics, latency requirements, and cost constraints. The planning-time-accuracy Pareto frontiers presented in Section 4.3. A illustrate that the "best" estimator depends on whether the target application prioritizes interactive latency or batch throughput. Extending this principle, an AI-assisted architecture decision framework could analyze historical and projected business workloads, data volumes, and access patterns to recommend data processing configurations that balance real-time responsiveness, infrastructure cost, and system complexity more systematically than manual evaluation allows.

At the system level, the results in this study point toward a broader opportunity: an AI-assisted data infrastructure that simultaneously improves processing speed and data quality while reducing unnecessary computation and data redundancy. The disconnect between q-error rankings and end-to-end execution time is documented in Section 4.3. A demonstrates that raw model accuracy does not automatically translate into system-level efficiency; what matters is how the estimator interacts with the optimizer, the execution engine, and the storage layer as a coordinated whole. By continuously analyzing system runtime states---including resource utilization, query queue depths, and data freshness---AI-driven optimization can inform not only estimator selection but also broader architectural decisions such as materialized view maintenance, partition strategies, and caching policies. Such a holistic approach would help large-scale data processing systems achieve a more favorable balance among scalability, cost, and operational stability, reinforcing the value of AI as a cross-cutting enabler within modern data infrastructure rather than a point solution for a single optimizer component.

Limitations: Our study has limitations. All numbers are drawn from published experiments on PostgreSQL, and the behavior of learned estimators on commercial engines with more sophisticated default statistics may differ in ways we cannot quantify from the public literature. The benchmarks surveyed are analytical and static; streaming and transactional settings remain outside the scope. The eight estimators included exclude pessimistic, large-language-model-based, and pre-trained cross-database estimators that have emerged more recently and whose results are not yet consolidated. Three directions for future research stand out. The first is a unified protocol reporting accuracy, planning time, training time, and update cost on a common hardware profile, which would retire the ambiguity that currently clouds single-metric comparisons. The second is planning-time-aware training objectives that penalize estimator latency during model selection, since current objectives treat planning-time overhead as an after-the-fact diagnostic rather than as part of the loss. The third is the systematic study of hybrid architectures that route queries between fast classical estimators and slower learned estimators based on a confidence signal, thereby preserving most of the accuracy gain at a fraction of the planning cost and remaining compatible with the existing interface between optimizers and cardinality oracles.

References

1. V. Leis, A. Gubichev, A. Mirchev, P. A. Boncz, A. Kemper, and T. Neumann, "How good are query optimizers, really?," *Proc. VLDB Endow.*, vol. 9, no. 3, pp. 204–215, 2015.
2. V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita, "Improved histograms for selectivity estimation of range predicates," in *Proc. 1996 ACM SIGMOD Int. Conf. Manage. Data*, 1996, pp. 294–305.
3. A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper, "Learned cardinalities: Estimating correlated joins with deep learning," in *9th Biennial Conf. Innovative Data Syst. Res. (CIDR)*, 2019.
4. Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, X. Chen, P. Abbeel, J. M. Hellerstein, S. Krishnan, and I. Stoica, "Deep unsupervised cardinality estimation," *Proc. VLDB Endow.*, vol. 13, no. 3, pp. 279–292, 2019.
5. Z. Wu, P. Negi, M. Alizadeh, T. Kraska, and S. Madden, "FactorJoin: A new cardinality estimation framework for join queries," *Proc. ACM Manag. Data*, vol. 1, no. 1, pp. 41:1–41:27, 2023.
6. V. Leis, B. Radke, A. Gubichev, A. Kemper, and T. Neumann, "Cardinality estimation done right: Index-based join sampling," in *8th Biennial Conf. Innovative Data Syst. Res. (CIDR)*, 2017.
7. B. Hilprecht, A. Schmidt, M. Kulessa, A. Molina, K. Kersting, and C. Binnig, "DeepDB: Learn from data, not from queries!," *Proc. VLDB Endow.*, vol. 13, no. 7, pp. 992–1005, 2020.
8. R. Zhu, Z. Wu, Y. Han, K. Zeng, A. Pfadler, Z. Qian, J. Zhou, and B. Cui, "FLAT: Fast, lightweight and accurate method for cardinality estimation," *Proc. VLDB Endow.*, vol. 14, no. 9, pp. 1489–1502, 2021.
9. Z. Yang, A. Kamsetty, S. Luan, E. Liang, Y. Duan, X. Chen, and I. Stoica, "NeuroCard: One cardinality estimator for all tables," *Proc. VLDB Endow.*, vol. 14, no. 1, pp. 61–73, 2021.
10. J. Wang, C. Chai, J. Liu, and G. Li, "FACE: A normalizing flow based cardinality estimator," *Proc. VLDB Endow.*, vol. 15, no. 1, pp. 72–84, 2022.
11. P. Wu and G. Cong, "A unified deep model of learning from both data and queries for cardinality estimation," in *Proc. 2021 Int. Conf. Manage. Data (SIGMOD)*, 2021, pp. 2009–2022.
12. K. Zhao, J. X. Yu, Z. He, R. Li, and H. Zhang, "Lightweight and accurate cardinality estimation by neural network Gaussian process," in *Proc. 2022 Int. Conf. Manage. Data (SIGMOD)*, 2022, pp. 973–987.
13. X. Wang, C. Qu, W. Wu, J. Wang, and Q. Zhou, "Are we ready for learned cardinality estimation?," *Proc. VLDB Endow.*, vol. 14, no. 9, pp. 1640–1654, 2021.
14. Y. Han, Z. Wu, P. Wu, R. Zhu, J. Yang, L. W. Tan, K. Zeng, G. Cong, Y. Qin, A. Pfadler, Z. Qian, J. Zhou, J. Li, and B. Cui, "Cardinality estimation in DBMS: A comprehensive benchmark evaluation," *Proc. VLDB Endow.*, vol. 15, no. 4, pp. 752–765, 2022.
15. K. Kim, J. Jung, I. Seo, W.-S. Han, K. Choi, and J. Chong, "Learned cardinality estimation: An in-depth study," in *Proc. 2022 Int. Conf. Manage. Data (SIGMOD)*, 2022, pp. 1214–1227.
16. J. Sun, J. Zhang, Z. Sun, G. Li, and N. Tang, "Learned cardinality estimation: A design space exploration and a comparative evaluation," *Proc. VLDB Endow.*, vol. 15, no. 1, pp. 85–97, 2022.
17. B. Ding, S. Chaudhuri, J. Gehrke, and V. Narasayya, "DSB: A decision support benchmark for workload-driven and traditional database systems," *Proc. VLDB Endow.*, vol. 14, no. 13, pp. 3376–3388, 2021.
18. P. Negi, Z. Wu, A. Kipf, N. Tatbul, R. Marcus, S. Madden, T. Kraska, and M. Alizadeh, "Robust query driven cardinality estimation under changing workloads," *Proc. VLDB Endow.*, vol. 16, no. 6, pp. 1520–1533, 2023.
19. B. Li, Y. Lu, and S. Kandula, "Warper: Efficiently adapting learned cardinality estimators to data and workload drifts," in *Proc. 2022 Int. Conf. Manage. Data (SIGMOD)*, 2022, pp. 1920–1933.
20. P. Li, W. Wei, R. Zhu, B. Ding, J. Zhou, and H. Lu, "ALECE: An attention-based learned cardinality estimator for SPJ queries on dynamic workloads," *Proc. VLDB Endow.*, vol. 17, no. 2, pp. 197–210, 2023.
21. K. Lee, A. Dutt, V. R. Narasayya, and S. Chaudhuri, "Analyzing the impact of cardinality estimation on execution plans in Microsoft SQL Server," *Proc. VLDB Endow.*, vol. 16, no. 11, pp. 2871–2883, 2023.
22. R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska, "Bao: Making learned query optimization practical," in *Proc. 2021 Int. Conf. Manage. Data (SIGMOD)*, 2021, pp. 1275–1288.
23. Y. Han, H. Wang, L. Chen, Y. Dong, X. Chen, B. Yu, C. Yang, and W. Qian, "ByteCard: Enhancing ByteDance's data warehouse with learned cardinality estimation," in *Companion Proc. 2024 Int. Conf. Manage. Data (SIGMOD)*, 2024, pp. 41–54.