End Term Report

Karvy Mohnot and Raj Shah

GitHub Repo: https://github.com/RajShah-1/db-optimizers/tree/main/CardEst/

Introduction

Query optimization is the process of selecting the most efficient query-evaluation plan from among the many strategies usually possible for processing a given query. The system is expected to construct a plan that minimizes the cost of query evaluation, where cost is typically measured in terms of I/O, CPU time, or memory usage.

A key component of this process is cardinality estimation—predicting the number of tuples that will be produced at various stages of query execution. These estimates directly influence the choice of access paths, join orders, and algorithms. Since the difference in cost between a good strategy and a bad one can be several orders of magnitude, accurate cardinality estimation becomes critical to avoid suboptimal plans. Therefore, it is worthwhile for the optimizer to invest effort into both selecting a good plan and producing reliable cardinality estimates, even if the query is executed only once.

The project's goal was to thoroughly understand, implement, and evaluate traditional CardEst models, analyze their limitations, and explore advanced techniques, including hybrid approaches guided by machine learning, to improve estimation accuracy. Our work involved establishing a robust benchmarking environment using the JOB benchmark on the IMDB dataset, implementing core traditional estimators (histogram-based and sample-based), analyzing their performance using the Q-error metric, and investigating state-of-the-art methods for accurate join cardinality estimation. A key aspect of our recent efforts has been the development of a feedback-driven refinement loop aimed at improving traditional estimates using insights gained from performance analysis.

Background & Related Work

Traditional cardinality estimation techniques largely rely on pre-computed statistics:

Histogram-Based Methods

These techniques summarize the distribution of values in columns using histograms, estimating selectivity based on how predicates align with histogram buckets. They are widely used due to their efficiency but suffer from inaccuracies when dealing with correlated data or complex predicates.

Sampling-Based Methods

These methods estimate cardinalities by running queries on a small sample of the data and scaling the results. While they can capture some data correlations, their accuracy is sensitive to sample size and strategy, and variance can be high for complex queries with low selectivity.

HyperLogLog-Based Methods

These probabilistic techniques estimate the cardinality of a multiset (i.e., the number of distinct values) using fixed-size sketches. HyperLogLog (HLL) is particularly space-efficient and scalable, making it suitable for large datasets and streaming scenarios. It provides fast approximate distinct counts with a small memory footprint and predictable error bounds. However, HLL focuses on estimating distinct counts rather than full result cardinalities, and its integration into general-purpose query optimizers requires careful handling, especially when combining with filters or joins.

Limitations of Traditional Methods

The biggest limitation of traditional estimators is their reliance on simplifying assumptions. They assume data independence between columns and uniform distribution within histogram bins. These assumptions often lead to substantial errors, especially in estimating the cardinality of multi-join queries, which is a known bottleneck for optimizer performance.

Literature Review of Modern Approaches

Recent advances in cardinality estimation have increasingly leveraged machine learning to overcome the limitations of traditional statistical techniques. Deep learning models, such as those proposed in *Fauce* and *Deep Learning Models for Selectivity Estimation of Multi-Attribute Queries*, aim to capture complex attribute correlations across relations. *Fauce* introduces a deep ensemble model that estimates not only cardinality but also predictive uncertainty, which helps in reducing the optimizer's overconfidence in inaccurate predictions. *FactorJoin* presents a hybrid framework that constructs factor graphs from traditional single-table statistics to estimate join cardinalities more effectively, offering a balance between classical and learned techniques. However, as highlighted in *PACE*, learned estimators are vulnerable to adversarial poisoning attacks that can significantly degrade performance, raising concerns about robustness in deployment. Additionally, *Is Your Learned Query Optimizer Behaving As You Expect?* critiques the current evaluation practices of learned query optimizers and emphasizes the need for standardized, rigorous benchmarking, noting that traditional optimizers can still outperform learned ones under certain conditions. Together, these works reflect an evolving landscape where learning-based estimators offer promising improvements in accuracy and generalization, but also introduce new challenges in robustness, interpretability, and integration.

Project Objectives

This project aimed to explore and improve traditional cardinality estimation techniques. The initial focus was on understanding core CardEst principles and implementing standard estimators such as histograms and sampling-based methods. These were evaluated using benchmark datasets, and the resulting estimation errors were analyzed to identify key weaknesses. Based on these findings, the project investigated advanced traditional and hybrid approaches, and began designing a feedback-based refinement mechanism to improve estimator accuracy over time.

System Design

To support the development and evaluation of cardinality estimators, we built a custom Python framework (`db-optimizers/CardEst/`). This framework provides support for schema definition, data loading (using the IMDB dataset), query parsing, statistical structure creation, modular estimator integration, and benchmarking with Q-error metrics. Instrumentation was added to monitor and log estimation errors at various stages of query execution. Alongside implementing the estimators themselves, significant effort went into building the surrounding infrastructure for robust benchmarking and error analysis.

Implementation Details

Traditional Estimators

We implemented two foundational cardinality estimation techniques:

- **Histogram-Based Estimator**: Raj developed the core logic for constructing histograms and estimating predicate selectivity based on bucket alignment.
- **Sample-Based Estimator**: Karvy implemented a sampling-based estimator, incorporating various

sampling strategies and scaling mechanisms to approximate result cardinalities.

Joint Improvement Efforts

After the initial implementation, our efforts shifted toward improving estimation accuracy and diagnosing failure modes.

- **Q-error:**
  Q-error is a common metric used in cardinality estimation to quantify the inaccuracy of estimates. It measures the multiplicative error between the estimated cardinality (Est) and the actual cardinality (Act). The Q-error for a given query is typically defined as max(Est/Act, Act/Est). This metric provides a symmetric and scale-invariant way to assess the quality of cardinality estimators.
- **Benchmarking and Error Analysis**: We integrated the JOB benchmark suite, comprising 70 queries over the IMDB dataset. Evaluation of the histogram-based estimator revealed a median Q-error of 7.964 and a maximum Q-error of 2498.712, with major inaccuracies observed in join-heavy queries. Instrumentation confirmed that underestimation in join selectivity was a key source of error.
- **Correlation Summaries**: To improve estimation over correlated columns, we extended the framework to include correlation-aware selectivity estimation.
- **Advanced Techniques**: We studied and partially incorporated several advanced methods, including HyperLogLog for distinct count estimation, Most Common Values (MCVs) for handling skew, and concepts from *FactorJoin* such as factor graphs and learned joint distributions for multi-table cardinality estimation.
- **Feedback Loop Design**: A feedback-driven refinement mechanism was designed to analyze stage-level Q-errors from benchmark runs. The proposed system includes a guiding component—potentially ML-based in future iterations—that suggests targeted estimator refinements (e.g., histogram splitting on join keys) based on past errors. To validate the potential of this approach, we currently apply all enhancements proactively per query, without relying on learning-based feedback, thereby demonstrating its effectiveness while preserving interpretability.

Evaluation

The evaluation was conducted using the Join Order Benchmark (JOB), a standard suite of 70 real-world queries over the IMDB dataset. Estimation quality was measured using **Q-error**, a widely used metric defined as the maximum of the ratio between estimated and true cardinality.

Initial results with the baseline `HistogramEstimator` revealed limitations in handling join-heavy and correlated queries, with a **median Q-error of 7.964** and a **maximum Q-error exceeding 2400**. After integrating the feedback-based enhancements, we observed consistent improvements across all metrics. As shown in the table below, the feedback loop led to reductions in mean, median, and tail Q-errors, particularly improving robustness in outlier cases.

Metric	**With Feedback Loop**	**Histogram Estimator**
median_q_error	7.371	7.964
mean_q_error	92.430	106.598
max_q_error	1852.115	2498.712
90th_percentile	210.589	255.238
95th_percentile	312.987	347.777
99th_percentile	1123.456	1354.009

These results demonstrate that even simple feedback-driven refinements, such as query-aware histogram splits and join-selectivity adjustments, can yield meaningful improvements. Future work will focus on automating the refinement loop and conducting broader evaluations across synthetic and skewed datasets.

Challenges and Learnings

Several challenges arose over the course of the project. Accurately estimating join cardinalities using traditional methods proved difficult, particularly due to correlated data selectivity. Developing a flexible and extensible benchmarking framework required careful design to support a wide range of queries, error metrics, and estimator configurations. Debugging the estimator logic, especially in multi-stage queries, was non-trivial, requiring fine-grained instrumentation and query tracing.

Through these efforts, we developed a deeper understanding of the nuances in query optimization, the limitations of existing cardinality estimators, and the importance of balancing interpretability and accuracy in cost estimation strategies.

Conclusion and Future Work

This project involved the implementation and evaluation of traditional cardinality estimation models, along with a systematic benchmarking effort to uncover their strengths and limitations. Through empirical analysis using the JOB benchmark, we identified significant estimation errors and used these insights to initiate the design of a feedback-driven refinement mechanism. Early results suggest that even targeted improvements to traditional estimators can lead to meaningful gains in accuracy.

Looking ahead, future work will focus on:

- Fully integrating advanced techniques such as **HyperLogLog** and **Most Common Values (MCV)** into the estimator pipeline.
- Exploring the use of **Factor Graphs** for modeling multi-table correlations, including the application of the feedback loop within this framework.
- Completing the design and evaluation of a **machine learning-guided feedback loop** for adaptive refinement.
- Conducting **comprehensive performance comparisons** against both classical and learned cardinality estimation methods.
- **Improving robustness against poisoning attacks**, particularly as ML-based components become more central to estimation and plan selection.

Overall, our findings suggest that enhancing traditional estimators with selective learning-based guidance offers a promising direction for building robust and interpretable cardinality estimation systems.

References

1. Liu, X., Li, Y., Du, Y., & Kraska, T. (2021). *Fauce: Fast and Accurate Deep Ensembles with Uncertainty for Cardinality Estimation*. Proceedings of the VLDB Endowment (PVLDB), 14(11), 1950–1962.
2. Yang, J., Tang, J., & Zhao, B. (2023). *PACE: Poisoning Attacks on Learned Cardinality Estimation*. Proceedings of the ACM SIGMOD International Conference on Management of Data.
3. Kipf, A., Kipf, M., Hilprecht, B., Kemper, A., & Neumann, T. (2022). *Learned Cardinality Estimation: A Design Space Exploration and A Comparative Evaluation*. Proceedings of the VLDB Endowment (PVLDB), 15(9), 1929–1942.
4. Trummer, I., & Yao, J. (2023). *Is Your Learned Query Optimizer Behaving As You Expect? A Machine Learning Perspective*. Proceedings of the VLDB Endowment (PVLDB), 16(8), 1789–1801.
5. Hasan, M. K., et al. (2020). *Deep Learning Models for Selectivity Estimation of Multi-Attribute Queries*. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data.

6. Wu, Z., Huang, X., Feng, Y., Xu, J., & Ma, K. (2022). *FactorJoin: A New Cardinality Estimation Framework for Join Queries*. arXiv preprint arXiv:2212.05526.
7. Moerkotte, G. (2019). *Fundamentals of Cardinality Estimation*. Now Foundations and Trends in Databases, 10(2–3), 130–282.
8. Hellerstein, J. M., Haas, P. J., & Wang, H. J. (1997). *Online Aggregation*. SIGMOD Record, 26(2), 171–182.
9. Tao, Y., Papadias, D., & Faloutsos, C. (2007). *Issa: Iceberg Query Processing with Approximation*. VLDB Journal, 15(4), 367–387.
10. Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., & Neumann, T. (2018). *The JOB Benchmark*. http://www.cs.cornell.edu/~jrg/imdb-benchmark/
11. CardBench Repository. *CardBench: Benchmarking Framework for Learned Cardinality Estimation*. GitHub. https://github.com/learnedsystems/CardBench