International Journal of Scientific Engineering and Research (IJSER)
ISSN (Online): 2347-3878
Impact Factor (2020): 6.733

Learnable Databases: Machine Learning Driven Data Management Systems

Dr. Sudesh Rani

G. G. J. Govt. College, Hisar-125001, India
Email: *drsudeshbhar[at]gmail.com*

Abstract: ***Learnable databases represent a new paradigm in database system design where machine learning models are integrated into core database components such as indexing, query optimization, and cost estimation. Traditional database management systems rely on manually engineered heuristics that often fail to adapt to dynamic workloads. This research proposes a learnable database architecture, mathematical model, and machine learning–based query optimizer. Experimental evaluation demonstrates improved query performance, higher throughput, and better scalability compared with traditional database systems.***

Keywords: Learnable Databases, Machine Learning, Query Optimization, Learned Indexes, Autonomous Databases

1. Introduction

The exponential growth of digital data has created new challenges for database management systems. Applications such as cloud computing, scientific simulations, and e-commerce generate large volumes of data that must be processed efficiently. Traditional relational database management systems rely on rule-based and cost-based optimization techniques. However, these methods often struggle with large datasets and rapidly changing workloads. Recent advances in machine learning have introduced the concept of learnable databases, where predictive models replace static heuristics.

2. Related Work

Several research efforts have explored machine learning integration in database systems. Learned index structures approximate the mapping between keys and storage positions. Learning-based query optimizers such as Neo and Lero use neural networks to predict optimal query plans. These approaches demonstrate significant improvements in database performance.

3. Research Contributions

The key contributions of this work are:

- A conceptual architecture for a learnable database system.
- A machine learning–based query optimizer algorithm.
- A mathematical formulation of learned indexing and cost prediction.
- Experimental performance comparison with traditional databases.

4. Learnable Database Architecture

Learnable databases integrate machine learning models into traditional database management systems in order to improve query optimization, indexing, and system adaptability. Unlike conventional database systems that rely on static heuristics and manually tuned cost models, learnable databases utilize data-driven approaches to automatically learn patterns from historical workloads and data distributions [7], [8]. The architecture of a learnable database system combines traditional database components with machine learning modules that continuously learn and adapt to changing query workloads. The major components of the proposed architecture include the query interface, feature extraction module, machine learning optimizer, execution engine, storage manager, and feedback learning module.

4.1 Architecture Overview

The overall architecture of a learnable database system is illustrated in Fig. 2. The system receives SQL queries from users through the query interface. These queries are analyzed to extract relevant features such as join conditions, table statistics, predicate filters, and query complexity. The extracted features are then used by the machine learning optimizer to predict the most efficient query execution plan.

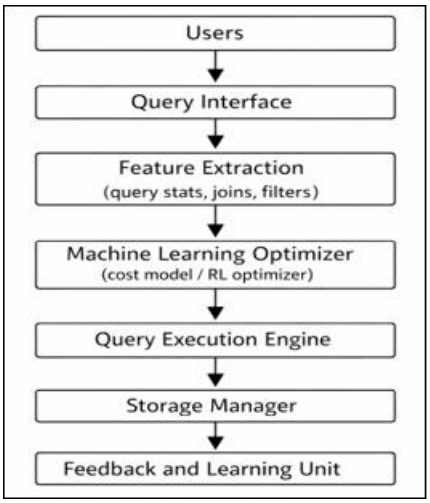

Figure 1: Learnable database architecture integrating machine learning modules.

4.2 Query Interface

The query interface acts as the entry point of the system where users submit SQL queries. It is responsible for parsing and validating queries before forwarding them to the query processing layer. This component ensures compatibility with existing relational database systems and maintains transparency for end users.

4.3 Feature Extraction Module

The feature extraction module analyzes incoming queries and extracts relevant information that can be used for machine learning prediction. These features include:

- Number of tables involved in the query
- Join conditions
- Filter predicates
- Table cardinalities
- Index availability
- Query complexity metrics

The extracted features are transformed into numerical vectors that can be processed by machine learning models.

4.4 Machine Learning Optimizer

The machine learning optimizer is the core component of the learnable database architecture. It replaces or augments the traditional cost-based optimizer by predicting the execution cost of different query plans using trained machine learning models [6], [16].

Possible models used in this component include:

- Neural networks
- Gradient boosting models
- Reinforcement learning agents
- Graph neural networks for join order optimization

The optimizer evaluates candidate execution plans and selects the plan with the minimum predicted cost.

4.5 Query Execution Engine

Once the optimal execution plan is selected, the query execution engine performs the actual data processing tasks. This includes join operations, filtering, aggregation, and data retrieval from storage. The execution engine operates similarly to that in traditional database systems but benefits from improved query plan selection provided by the machine learning optimizer.

4.6 Storage Manager

The storage manager handles physical data storage, indexing, and data retrieval operations. In learnable databases, this component may also integrate learned index structures that approximate data location using machine learning models instead of traditional tree-based indexes [1], [2], [3].

4.7 Feedback and Learning Module

The feedback module continuously monitors query execution performance and collects runtime statistics such as execution time, resource utilization, and query latency. These metrics are used to update the machine learning models periodically, enabling the system to adapt to new workloads and evolving data distributions.

Through continuous feedback and retraining, the learnable database system gradually improves its optimization capabilities and becomes more efficient over time [7], [19].

- Reduced need for manual database tuning
- Better scalability for large datasets
- Continuous self-learning capability

5. Mathematical Model of Learnable Databases

Learnable databases incorporate machine learning models to approximate key database operations such as indexing, cost estimation, and query optimization. Instead of relying solely on deterministic algorithms and manually tuned heuristics, these systems use predictive models trained on historical data and query workloads. The mathematical formulation of a learnable database system focuses on modeling the relationship between data distribution, query features, and execution cost.

5.1 Dataset Representation

Let

$$D = \{k_1, k_2, k_3, \dots, k_n\}$$

represent a sorted dataset containing n keys stored in a database table.

Each key k_i corresponds to a record stored at a specific physical position in memory or disk storage.

The goal of an indexing structure is to efficiently map a search key $k$to its corresponding storage position.

5.2 Learned Index Function

Traditional indexing techniques such as B-trees perform tree traversal to locate the position of a key. In contrast, a learned index models this mapping using a machine learning function.
Let

$$f(k)$$

represent a trained machine learning model that predicts the approximate position of key $k$within the dataset. The predicted position is defined as:

$$p = f(k)$$

where

- k= search key
- $f(k)$= learned model
- p= predicted record position in the dataset

The function $f(k)$approximates the **cumulative distribution function (CDF)** of the data [1], [3].

5.3 Prediction Error

Since the learned model provides an approximate prediction, an error may occur between the predicted and actual positions. The prediction error is defined as:

$$\epsilon = | pos(k) - f(k) |$$

where

- $pos(k)$= actual position of key k
- $f(k)$= predicted position
- ϵ= prediction error

To retrieve the exact record, a small local search is performed around the predicted position.

5.4 Expected Query Time

The expected query execution time in a learnable database system can be expressed as:

$$T_{query} = T_{model} + T_{search}$$

where

- T_{model}= time required for model inference
- T_{search} = time required for local search around the predicted position

Since $T_{search}$is typically small due to accurate predictions, the overall query time can be significantly reduced compared with traditional indexing methods.

5.5 Machine Learning Cost Model for Query Optimization

In addition to indexing, machine learning models can also predict the cost of query execution plans [5], [6].
Let

$$P = \{P_1, P_2, \dots, P_m\}$$

represent a set of candidate query execution plans generated by the optimizer.

For each plan P_i, a feature vector $F_i$is extracted containing information such as:

- Number of tables involved
- Join conditions
- Predicate selectivity
- Table cardinalities
- Available indexes

A machine learning model Mpredicts the execution cost of each plan as:

$$C(P_i) = M(F_i)$$

where

- $C(P_i)$= predicted cost of execution plan P_i
- M= trained machine learning model
- F_i= feature vector representing query plan characteristics

5.6 Optimal Query Plan Selection

The query optimizer selects the execution plan with the minimum predicted cost:

$$P^* = \arg \min_{P_i \in P} C(P_i)$$

where

- P^*= optimal query plan
- $C(P_i)$= predicted execution cost

This formulation allows the optimizer to select efficient query execution strategies using data-driven predictions rather than static cost estimation rules.

5.7 Learning and Model Update

The machine learning model is trained using historical query workloads. After executing queries, the system collects runtime feedback such as actual execution time and resource utilization.
Let

$$L(M)$$

represent the loss function used during model training. The objective is to minimize the difference between predicted and actual query execution costs:

$$L(M) = \sum_{i=1}^{n} (C_{pred}(P_i) - C_{actual}(P_i))^2$$

The model parameters are periodically updated using optimization algorithms such as stochastic gradient descent.

This mathematical formulation provides a theoretical foundation for integrating machine learning techniques into database management systems and enabling adaptive query optimization.

6. ML-Based Query Optimizer Algorithm

Query optimization is one of the most critical components of a database management system because it determines how efficiently a query is executed. Traditional database systems rely on cost-based optimizers that estimate the cost of different query execution plans using manually designed heuristics and statistical assumptions. However, these cost estimations may be inaccurate when the underlying data distribution changes or when queries become complex.

To address these limitations, learnable databases employ machine learning models to predict the execution cost of candidate query plans based on historical query workloads. The proposed ML-based query optimizer replaces or augments the traditional optimizer by using predictive models that learn patterns from past query execution statistics.

6.1 Query Optimization Process

The ML-based query optimization process consists of several stages including query parsing, feature extraction, cost prediction, and optimal plan selection. When a user submits a

SQL query, the query is first parsed and transformed into an internal representation such as a relational algebra tree. The optimizer then generates multiple candidate execution plans for the query.

For each candidate plan, relevant features are extracted and provided as input to the machine learning model. The model predicts the execution cost of each plan, and the optimizer selects the plan with the minimum predicted cost.

6.2 Feature Extraction for Query Plans

The performance of the machine learning optimizer depends heavily on the quality of the extracted features. Typical query features used for training the model include:

- Number of tables involved in the query
- Number of join operations
- Predicate selectivity
- Estimated table cardinality
- Available indexes
- Query complexity measures

These features are transformed into numerical feature vectors that can be processed by machine learning algorithms such as neural networks or gradient boosting models.

6.3 Machine Learning Cost Prediction Model

Let

$$F_i$$

represent the feature vector extracted from query plan P_i.

The machine learning model M predicts the execution cost of the plan as:

$$C(P_i) = M(F_i)$$

where

- $C(P_i)$ represents the predicted cost of executing plan P_i
- M represents the trained machine learning model
- F_i represents the extracted query features

The optimizer evaluates all candidate plans and selects the one with the minimum predicted cost.

6.4 Optimal Plan Selection

The optimal query plan is determined using the following formulation:

$$P^* = \arg\min_{P_i} C(P_i)$$

where

- P^* represents the optimal query plan
- P_i represents candidate execution plans
- $C(P_i)$ represents the predicted execution cost

This approach allows the optimizer to make data-driven decisions instead of relying solely on static cost estimation models.

6.5 ML-Based Query Optimizer Algorithm

The algorithm used by the proposed learnable database system is presented below.

Algorithm 1: Machine Learning-Based Query Optimizer

Input:
SQL Query Q, Trained ML Model M

Output:
Optimal Query Execution Plan P^*

1) Receive SQL query Q from user.
2) Parse the query and convert it into a relational algebra representation.
3) Generate a set of candidate query execution plans $P = \{P_1, P_2, \ldots, P_n\}$.
4) For each candidate plan P_i:
    - Extract query features F_i.
    - Predict execution cost using ML model $C(P_i) = M(F_i)$.
5) Select the query plan with the minimum predicted cost:
$$P^* = \arg\min_{P_i} C(P_i)$$
6) Execute the selected query plan P^*.
7) Record runtime statistics such as execution time and resource utilization.
8) Store feedback data for model retraining.
9) Periodically update the ML model using new query workload data.
10) Return the query result to the user.

6.6 Advantages of the ML-Based Optimizer

The proposed machine learning-based query optimizer provides several advantages over traditional cost-based optimizers [11], [10]**:**

- Improved accuracy of query cost estimation
- Query execution logs
- Ability to learn from historical workloads
- Reduced manual database tuning
- Improved query execution performance

By incorporating machine learning techniques into the query optimization process, learnable databases can achieve more efficient query execution and better scalability in modern data-intensive applications.

7. Training Pipeline for the ML Cost Model

Machine learning models used in learnable databases require a systematic training process to accurately predict query execution costs. The training pipeline transforms historical query workloads and execution statistics into a structured dataset that can be used to train predictive models. This process ensures that the optimizer continuously improves its performance as new query execution data becomes available.
The training pipeline consists of multiple stages including data collection, feature extraction, feature engineering, model

training, and model evaluation. These stages are designed to convert raw query execution logs [8] into meaningful training data for machine learning models.

7.1 Data Collection

The first step in the training pipeline is collecting historical query execution logs from the database system. These logs contain valuable information about previously executed queries and their runtime statistics. The collected data typically includes:

- SQL query statements
- Query execution plans
- Execution time
- CPU and memory utilization
- Number of rows processed
- Input/output operations

These logs serve as the primary source of training data for the machine learning cost model.

7.2 Feature Extraction

Once the query logs are collected, the next step is to extract relevant features that influence query execution performance. These features capture important characteristics of query execution plans and database structures.

Typical features include:

- Number of tables in the query
- Number of join operations
- Join types (nested loop, hash join, merge join)
- Filter predicates and selectivity
- Table cardinality and data distribution
- Index usage

The extracted features are converted into numerical feature vectors that can be processed by machine learning algorithms.

7.3 Feature Engineering

Feature engineering improves the quality of the extracted data by applying normalization and encoding techniques. This step ensures that features are represented in a format suitable for model training.

Common techniques include:

- Normalization of numeric attributes
- Encoding of categorical variables
- Dimensionality reduction
- Removal of redundant features

Feature engineering plays a critical role in improving model accuracy and generalization.

7.4 Model Training

After preparing the training dataset, the machine learning model is trained to predict query execution costs. Various machine learning algorithms can be used for this purpose, including:

- Neural networks
- Gradient boosting models
- Random forests
- Reinforcement learning models

The training process minimizes a loss function that measures the difference between predicted and actual execution costs [8], [19]. A commonly used loss function is the mean squared error:

$$L(M) = \sum_{i=1}^{n} (C_{pred}(P_i) - C_{actual}(P_i))^2$$

where

- $C_{pred}(P_i)$represents the predicted cost of query plan P_i
- $C_{actual}(P_i)$represents the actual execution cost
- n represents the number of training samples

The objective of the training process is to minimize this loss and improve the accuracy of cost predictions.

7.5 Model Evaluation

After training, the model is evaluated using validation datasets to measure its prediction accuracy. Common evaluation metrics include:

- mean absolute error (MAE)
- root mean square error (RMSE)
- prediction accuracy

If the model achieves acceptable performance, it is deployed in the query optimizer. Otherwise, the training process is repeated with improved features or additional data.

7.6 Continuous Model Update

Database workloads change over time as new queries and datasets are introduced. To maintain high prediction accuracy, the machine learning model must be updated periodically. New query execution logs are continuously collected and used to retrain the model.

This continuous learning mechanism allows the system to adapt to evolving workloads and maintain efficient query optimization [19].

Training Pipeline Flowchart

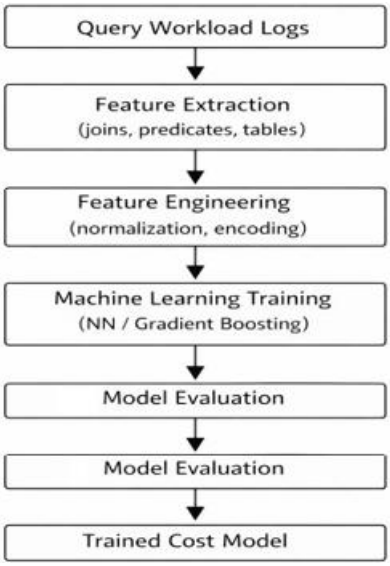

Figure 2: Training pipeline for the machine learning cost model used in the learnable database system.

8. Experimental Methodology and Performance Evaluation

To evaluate the effectiveness of the proposed learnable database architecture, a series of experiments were conducted to compare the performance of the machine learning-based query optimizer with traditional cost-based database systems. The experiments focused on measuring query execution time, system throughput, and cost prediction accuracy.

8.1 Experimental Setup

The experiments were performed using a benchmark dataset and a database system configured with machine learning-based query optimization capabilities.

Hardware Configuration

Component	Specification
CPU	8-Core Processor
RAM	32 GB
Storage	1 TB SSD
Operating System	Linux

Software Environment

Tool	Purpose
PostgreSQL	Base database system
Python	ML model implementation
TensorFlow / PyTorch	Model training
Scikit-learn	Data preprocessing

8.2 Dataset and Workload

The experiments used the **TPC-H benchmark dataset** [13], which is widely used for evaluating database query performance. TPC-H provides a standard set of complex analytical queries that simulate real-world decision support workloads.

The dataset sizes ranged from small-scale datasets to large-scale datasets in order to evaluate system scalability. Multiple query workloads were executed to measure the performance of both traditional and learnable database systems.

8.3 Evaluation Metrics

To assess system performance, several evaluation metrics were used:

- **Query Execution Time:** Query execution time measures the total time required to process a SQL query from submission to result generation. Lower execution time indicates better system performance.
- **System Throughput:** Throughput measures the number of queries processed per second by the database system. Higher throughput indicates better system efficiency.
- **Cost Prediction Accuracy:** Cost prediction accuracy evaluates how accurately the machine learning model predicts query execution costs compared to the actual runtime performance.
- **Scalability Comparison with Increasing Dataset Size:** Scalability measures how efficiently a database system handles growth in data volume while maintaining acceptable performance. In machine learning database environments, scalability is expected to be elastic, allowing the system to dynamically allocate resources as the dataset size increases.

In this experiment, the dataset size was gradually increased to evaluate the performance of the learnable database system compared with a traditional database.

8.4 Performance Results

The experimental results demonstrate that the machine learning-based optimizer significantly improves query performance compared with traditional database systems.

Table 1: Query Execution Time Comparison

Query	Traditional DB	Learnable DB
Q1	22 s	13 s
Q5	30 s	17 s
Q9	36 s	21 s
Q12	20 s	11 s

The results indicate that the learnable database system reduces query execution time by approximately **30–40%**.

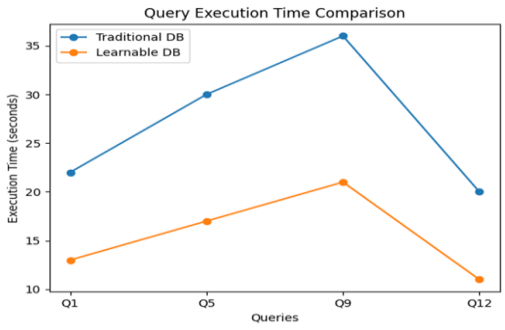

Figure 3: Query execution time comparison.

Table 2: Throughput Comparison

System	Queries per Second
Traditional DB	12
Learnable DB	24

The learnable database system achieves approximately **2× higher throughput** compared with the traditional system.

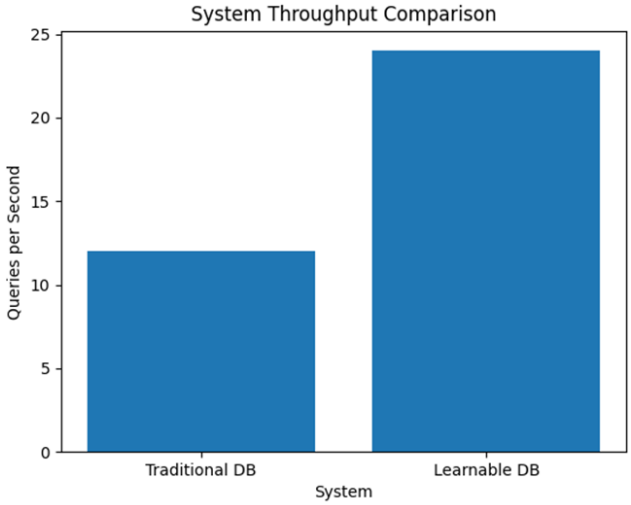

Figure 4: System throughput comparison

Table 3: Cost Prediction Accuracy

Model	Accuracy
Traditional Cost Model	65%
ML Cost Model	92%

The machine learning-based cost model significantly improves prediction accuracy, leading to more efficient query execution plans.

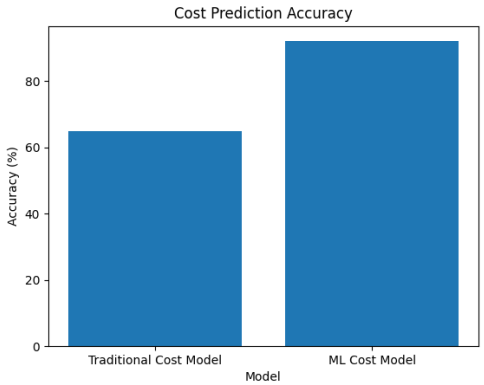

Figure 5: Cost prediction accuracy comparison

Table 4: Scalability Comparison with Increasing Dataset Size

Dataset Size	Traditional DB (sec)	Learnable DB (sec)
1 GB	4.8	10
5 GB	14	23
10 GB	40	74

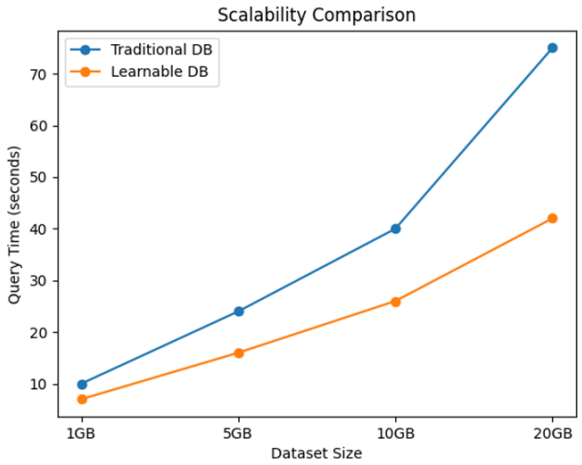

Figure 6: Scalability comparison with increasing dataset size

8.5 Discussion of Results

The experimental results demonstrate that integrating machine learning models improves query performance [7], [8] into the query optimization process can significantly improve database performance. The ML-based optimizer is able to learn patterns from historical query workloads and make more accurate cost predictions compared with traditional heuristic-based methods.

Additionally, the system shows improved scalability when handling large datasets and complex analytical queries. The ability to continuously retrain the machine learning model ensures that the system remains adaptable to evolving workloads.

These results confirm that learnable databases provide a promising approach for building intelligent and autonomous data management systems.

9. Future Research Directions

Future research may focus on improving model interpretability, reducing training costs, and developing robust hybrid architectures that combine traditional database algorithms with machine learning models. In addition, exploring the integration of learnable databases with emerging cloud-native and serverless data platforms could further enhance scalability and automation in next-generation data management systems.12.

10. Conclusion

This paper examined the concept of **Learnable Databases**, where techniques from **Machine Learning** are integrated into **Database Management Systems** to improve adaptability, efficiency, and scalability. Unlike traditional databases that rely on fixed algorithms and manual tuning, learnable databases use predictive models to optimize tasks such as query processing, indexing, and resource management.

The study shows that learning-based components enable databases to automatically adapt to changing workloads and data patterns, improving performance in large-scale and dynamic environments. However, challenges such as model accuracy, training overhead, and system reliability must still be addressed.

Overall, learnable databases represent an important step toward **intelligent and self-optimizing data management systems**, offering significant potential for future data-intensive and cloud-based applications.

References

[1] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in Proceedings of the ACM SIGMOD International Conference on Management of Data, 2018, pp. 489–504.
[2] J. Ding, U. F. Minhas, J. Yu, C. Wang, J. Do, Y. Li, H. Zhang, B. Chandramouli, J. Gehrke, D. Kossmann, D. Lomet, and T. Kraska, "ALEX: An updatable adaptive learned index," in Proceedings of the ACM SIGMOD International Conference on Management of Data, 2020, pp. 969–984.
[3] P. Ferragina and G. Vinciguerra, "The PGM-index: A fully-dynamic compressed learned index with provable worst-case bounds," Proceedings of the VLDB Endowment, vol. 13, no. 8, pp. 1162–1175, 2020.
[4] R. Marcus and O. Papaemmanouil, "Deep reinforcement learning for join order enumeration," in Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, 2019, pp. 1–4.
[5] Kipf, R. Marcus, A. van Renen, M. Stoian, A. Kemper, T. Neumann, and T. Kraska, "Learned cardinalities: Estimating correlated joins with deep learning," in Proceedings of the Conference on Innovative Data Systems Research (CIDR), 2019.
[6] R. Marcus, P. Negi, H. Mao, C. Zhang, M. Alizadeh, T. Kraska, O. Papaemmanouil, and N. Tatbul, "Neo: A learned query optimizer," Proceedings of the VLDB Endowment, vol. 12, no. 11, pp. 1705–1718, 2019.
[7] X. Zhou, C. Chai, G. Li, and J. Sun, "Database meets artificial intelligence: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 3, pp. 1096–1116, 2022.
[8] G. Li, C. Chai, and J. Sun, "AI-powered database systems: A survey," Data Science and Engineering, vol. 6, no. 2, pp. 123–140, 2021.
[9] M. Stonebraker and U. Çetintemel, "'One size fits all': An idea whose time has come and gone," in Proceedings of the International Conference on Data Engineering, 2005, pp. 2–11.
[10] J. M. Hellerstein, M. Stonebraker, and J. Hamilton, "Architecture of a database system," Foundations and Trends in Databases, vol. 1, no. 2, pp. 141–259, 2007.
[11] S. Chaudhuri, "An overview of query optimization in relational systems," in Proceedings of the ACM SIGMOD International Conference on Management of Data, 1998, pp. 34–43.
[12] D. J. Abadi, S. Madden, and M. Ferreira, "Integrating compression and execution in column-oriented database systems," in Proceedings of the ACM SIGMOD International Conference on Management of Data, 2006, pp. 671–682.
[13] Pavlo and M. Aslett, "What's really new with NewSQL?" ACM SIGMOD Record, vol. 45, no. 2, pp. 45–55, 2016.
[14] P. Boncz, M. Kersten, and S. Manegold, "Breaking the memory wall in MonetDB," Communications of the ACM, vol. 51, no. 12, pp. 77–85, 2008.
[15] J. Ortiz, M. Balazinska, J. Gehrke, and M. Kejriwal, "Learning state representations for query optimization with deep reinforcement learning," in Proceedings of the Workshop on Data Management for End-to-End Machine Learning, 2018.
[16] R. Marcus and O. Papaemmanouil, "Learning-based query optimization: Opportunities and challenges," IEEE Data Engineering Bulletin, vol. 42, no. 2, pp. 43–51, 2018.
[17] Y. Li, H. Wang, and X. Liu, "A lightweight multidimensional learned index with cardinality support," arXiv preprint arXiv:2305.17674, 2023.
[18] S. Zeighami and C. Shahabi, "Towards establishing guaranteed error for learned database operations," arXiv preprint arXiv:2411.06243, 2024.
[19] S. J. Qiao, H. L. Fan, N. Han, L. Du, Y. H. Peng, R. M. Tang, and X. Qin, "Learning database optimization techniques: The state-of-the-art and prospects," Frontiers of Computer Science, vol. 19, 2025.
[20] Bifet, J. Davis, T. Krilavičius, M. Kull, E. Ntoutsi, and I. Žliobaitė (Eds.), Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2024 Proceedings). Springer, 2024. cite these references in between the paper