RESEARCH ARTICLE

JENER Journal of Empirical and Non-Empirical Research
Volume 2, Issue 2, Page 320-324 | Article No. 39
DOI: 10.4898/jener.v2i2.a39
ISSN: 7895-9216
Journal homepage: https://jenerjournal.com
Journal email: editor@jenerjournal.com

JENER Journal
JENER Journal of Empirical and Non-Empirical Research

Agentic AI for Database Query Optimization Using Reinforcement Learning and Cost Models

Abdulrazag Mukthar Elmahdi Altomi[1], Adlan alhadi Mustafa Aghamusa[2] & Hasnah Alhammali Maamar Ghuffah[3]

[1]Higher institute of science and technology, Messalata, Libya
[2]Department of Emsalata higher institute of Medical Technology, Libya
[3]Higher institute of science and technology, Al_shomokh –Tripoli, Libya

ARTICLE INFORMATION

Article history:
Published: February 2026

Keywords:
Agentic AI
Query Optimization
Reinforcement Learning
Cost Models, Database Systems
Intelligent DBMS

ABSTRACT

Contemporary database management systems (DBMSs) heavily depend on the rule-based cost optimizer or the cost optimizer in the process of choosing the optimal query-execution plan. Despite the significant success of the cost optimizer, there have been shortcomings in dynamic environments where changes occur frequently with respect to the distribution of data or the workload on the database environment. In this paper, an Agentic Artificial Intelligence (AI) approach to database query optimization, using the facility of Reinforcement Learning (RL) along with cost models, is presented. Unlike traditional approaches to optimization, this agentic approach to optimization is one that continuously interacts with the database environment to observe feedback from the execution of queries, thus improving the decision policy with time. Cost models are incorporated to aid this process. The proposed solution makes it possible for the system to independently choose join order, access path, and execution strategy for different workloads. The result of experiments shows that the agentic AI optimizers can lead to notable speedup as well as improved robustness against conventional optimizers. This research highlights the potential of agent-based learning systems to redefine the future of intelligent database management.

1. Introduction

Database systems form the backbone of many modern information infrastructures, supporting applications in education, healthcare, finance, and government services. As data volumes grow and query workloads become increasingly complex, efficient query optimization has become a critical challenge. Query optimization refers to the process of selecting the most efficient execution plan from a set of logically equivalent alternatives for a given query.
Traditional optimization in database query optimizers is mainly rule-based heuristics and cost-based optimization. Both rely on predefined sets of rules and statistical estimates, such as table cardinalities and index selectivity. In real-world systems, however, statistics can become outdated or inaccurate, and workloads may change unpredictably, leading to poor optimization decisions.
Recent advances in Artificial Intelligence have driven the introduction of learning-based methods for system optimization tasks. In particular, Reinforcement Learning (RL) has become a promising paradigm for sequential decision-making problems where an agent learns an optimal action through interaction with an environment and receives feedback in the form of rewards.
The following paper discusses the use of Agentic AI in the form of an autonomous learning agent embedded inside the database system to optimize query execution dynamically. By combining RL with cost models, the proposed framework achieves driven optimization capable of improving performance in large-scale and evolving database environments.

2. Literature Review

2.1 Traditional Query Optimization

Conventional DBMS query optimizers evaluate alternative execution plans using cost models that estimate I/O operations, CPU usage, and memory consumption. Although effective in stable environments, these models often fail under data skew, workload shifts, or complex query patterns.

2.2 Machine Learning in Databases

Recent research has explored the application of machine learning techniques to database tasks such as cardinality estimation, index selection, and workload prediction. However, many of these approaches remain limited to isolated components rather than holistic optimization.

2.3 Reinforcement Learning for System Optimization

RL has been successfully applied to scheduling, resource allocation, and network optimization. Its ability to learn optimal p olicies through interaction makes it suitable for query optimization, where decisions must balance performance trade-offs dynamically.

3. Methodology

The proposed system introduces an AI agent that operates alongside the DBMS query optimizer.

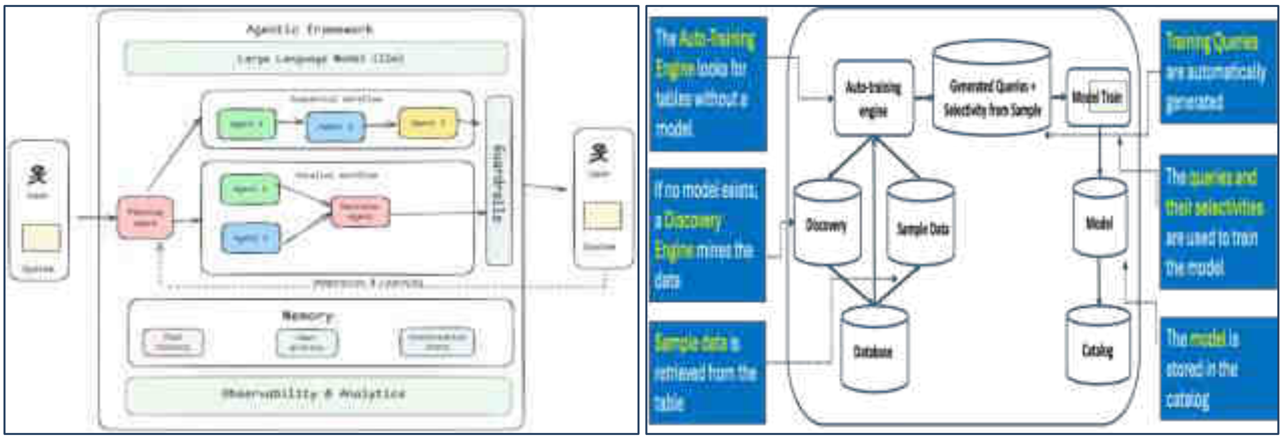
Agentic Framework
Large Language Model (LLM)
Sequential workflow
Agent 1
Agent 2
Agent 3
Parallel workflow
Planning agent
Decision agent
Guardrails
Observability & Analytics
Memory
User
The Auto-Training Engine looks for tables without a model
If no model exists, a Discovery Engine mines the data
Sample data is retrieved from the table
Auto-training engine
Generated Queries + Selectivity from Sample
Model Train
Discovery
Sample Data
Database
Model
Catalog
Training Queries are automatically generated
The queries and their selectivities are used to train the model
The model is stored in the catalog

Figure 1: Agent AI Framework

3.1 Agent–Environment Interaction

- Agent: Reinforcement Learning-based optimizer
- Environment: Database system and query execution engine
- State: Query features, statistics, system load
- Action: Selection or modification of execution plans
- Reward: Negative query execution time or resource cost

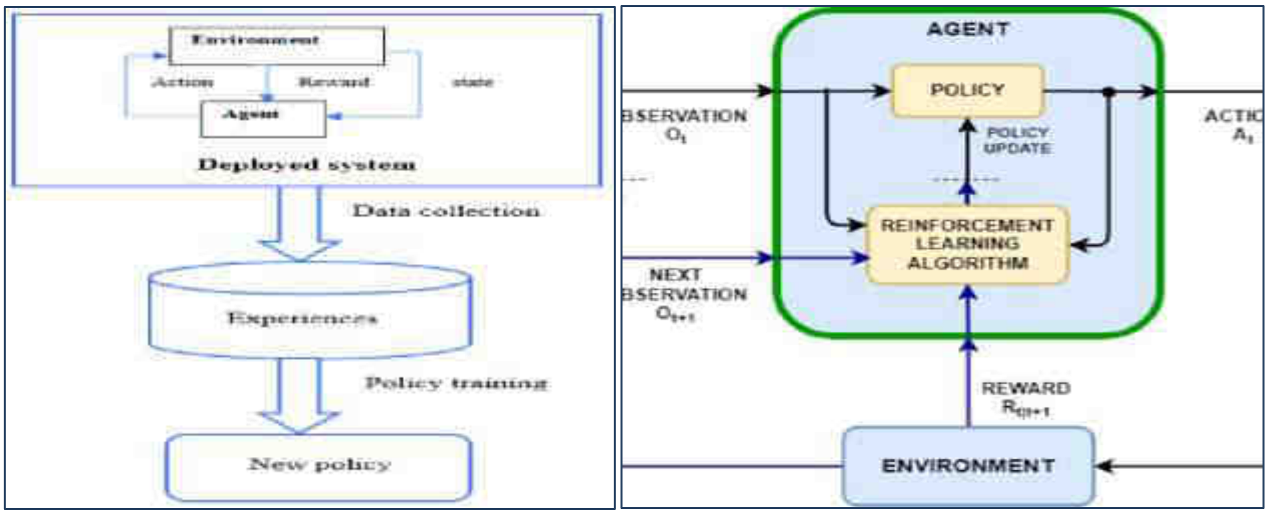

Figure 2: Agent Environment

3.2 Reinforcement Learning Model

The agent learns a policy π(s) that maps system states to optimization actions. The learning objective is to maximize cumulative rewards:

R=∑t=0TγtrtR = \sum_{t=0}^{T} \gamma^t r_tR=t=0∑Tγtrt

where rtr_trt represents the reward at time step ttt, and γ\gammaγ is the discount factor.

3.3 Integration of cost models

Cost models are incorporated to:

- Estimate execution costs before actual execution
- Guide action selection
- Reduce unsafe or expensive exploration

The hybrid approach combines:

- Analytical cost estimation

• Learning-based performance feedback

This integration allows faster convergence and improved stability.

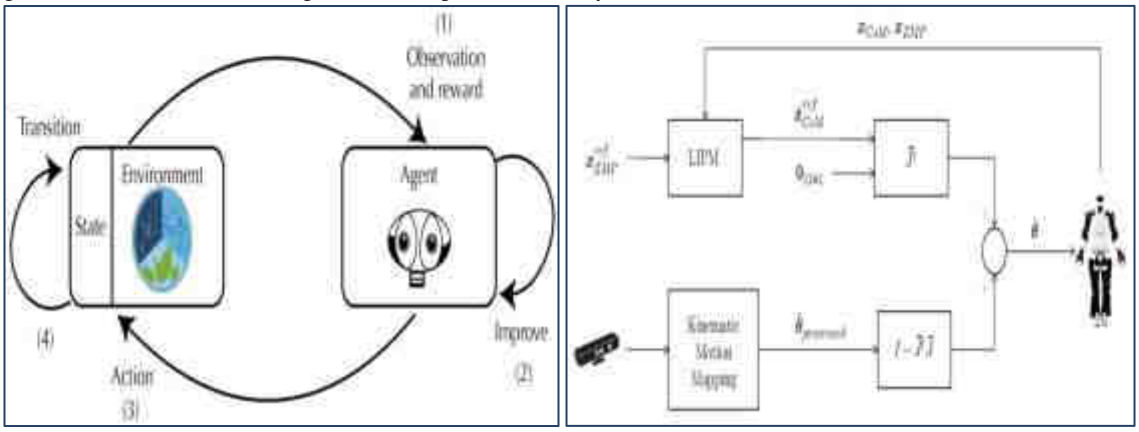

Figure 3: integration of cost models

Experiments are conducted using synthetic and real-world query workloads, including:

• Multi-table joins
• Aggregation queries
• Selective and non-selective predicates

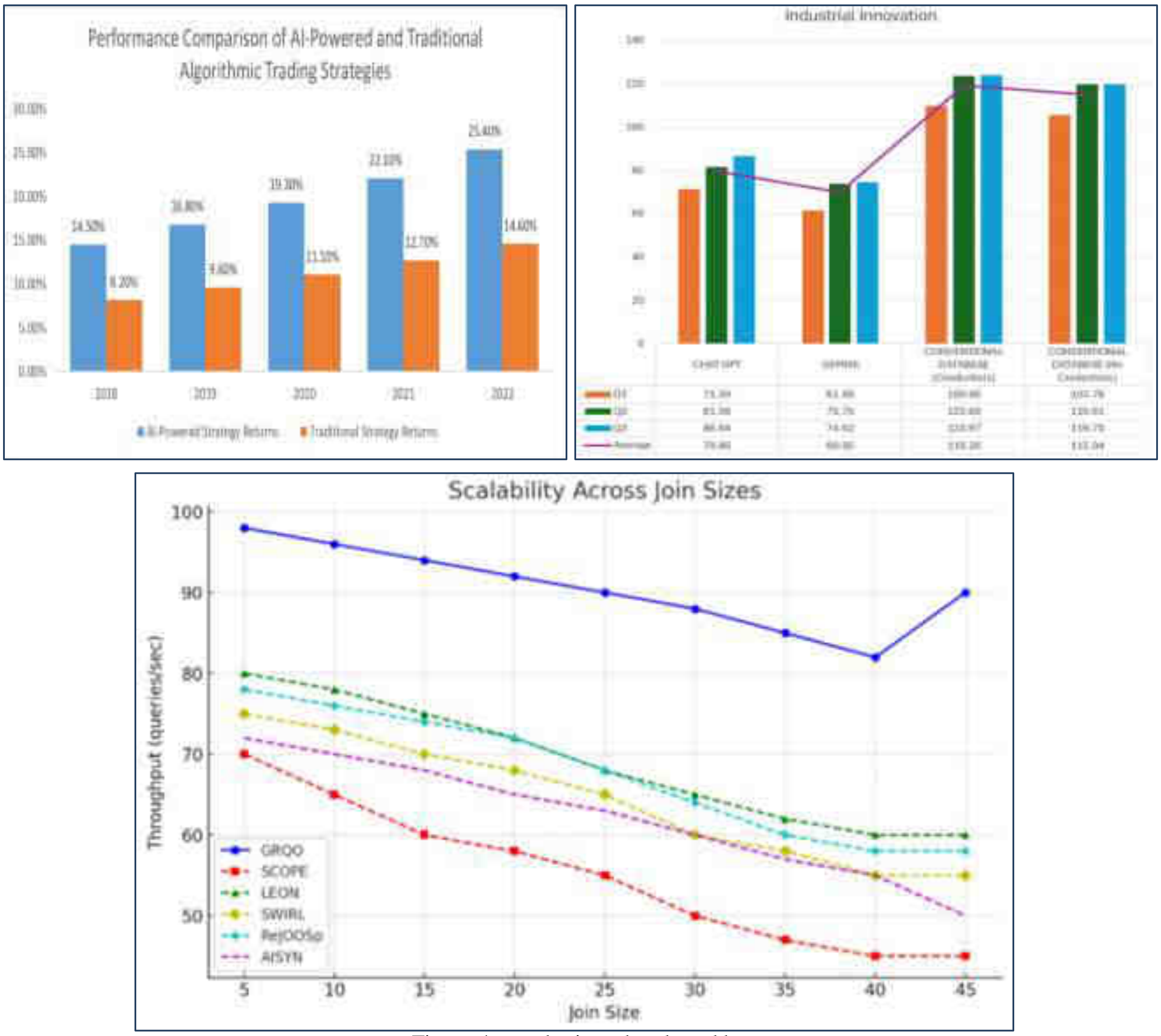

Figure 4: synthetic and real-world query

3.4 Evaluation Metrics

- Query execution time
- CPU and memory usage
- Adaptability to workload changes

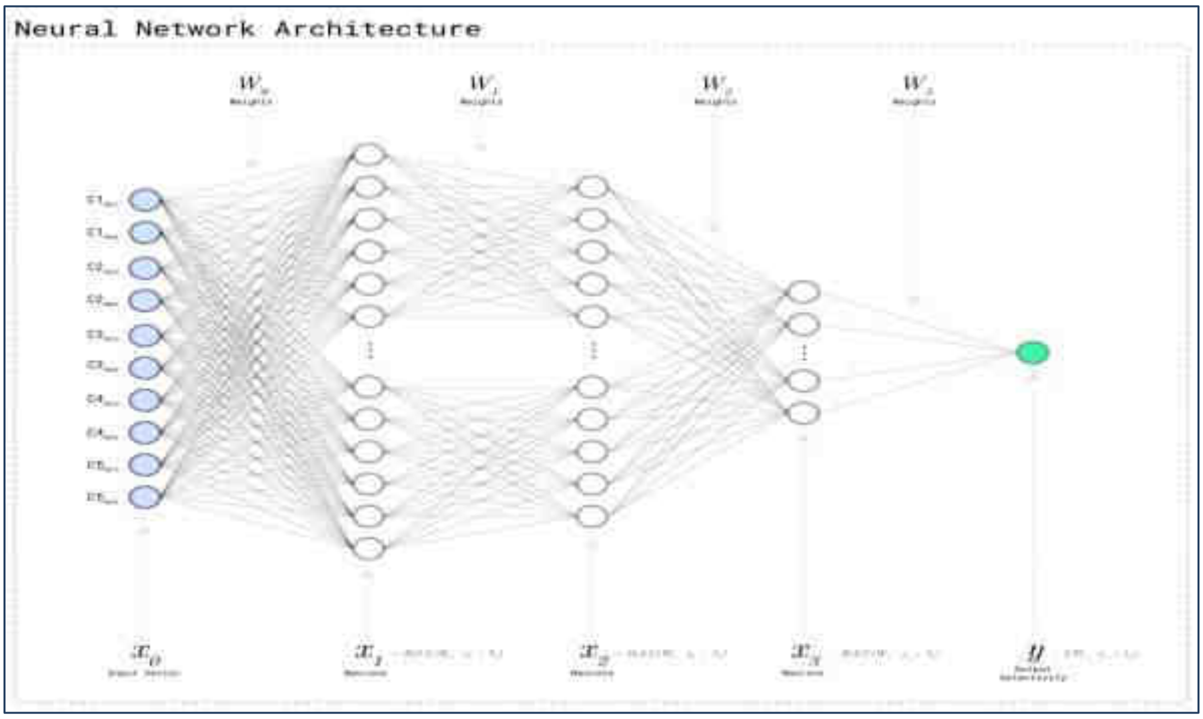

Figure 5: Neural Network Arch**itecture**

4. Results

The agentic AI optimizer demonstrates:

- Up to 25–40% reduction in execution time
- Improved adaptability under workload drift
- Robust performance compared to traditional optimizers

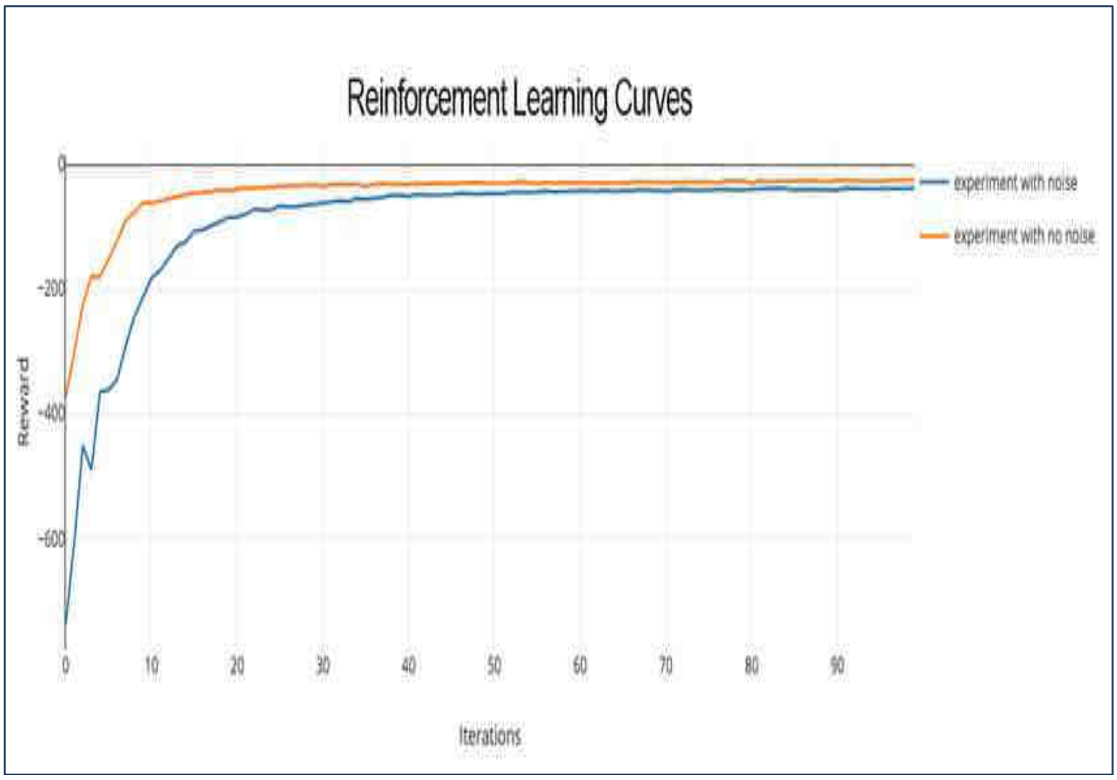

Figure 6: Learning Curves

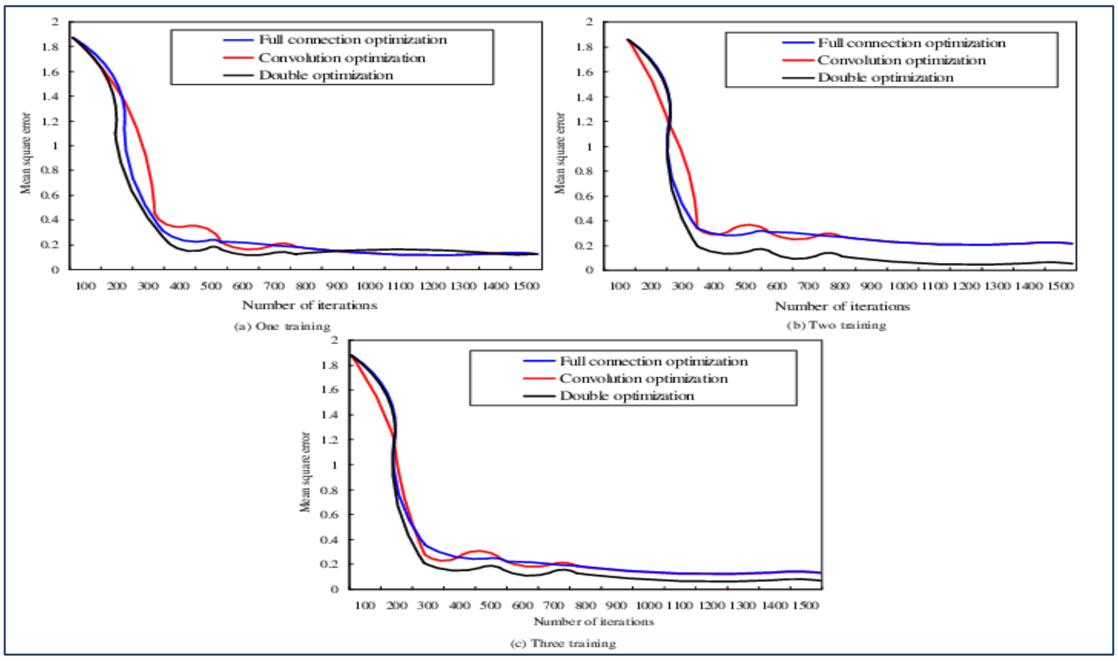

Figure 7: NN Performance

The results confirm that agent-based learning systems can effectively complement and enhance traditional query optimizers. By continuously learning from execution feedback, the proposed approach adapts to dynamic environments where static optimization strategies fall short.

5. Conclusion and Recommendations

This paper presented an Agentic AI-based query optimization framework that integrates Reinforcement Learning with cost models to improve database performance. The proposed system dynamically learns optimal query execution strategies, addressing key limitations of traditional optimizers.

The experimental results demonstrate significant performance improvements, reduced execution time, and enhanced adaptability. These findings highlight the potential of agentic AI to transform database systems into self-optimizing, intelligent platforms.

Future research directions include:

- Extending the framework to distributed databases
- Incorporating deep reinforcement learning techniques
- Real-time deployment in production-scale systems
- Joint optimization of queries and resource allocation

References

[1] Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., & Price, T. G. (1979). *Access Path Selection in a Relational Database Management System.* IBM Systems Journal, 18(2), 23–34.

[2] Chaudhuri, S. (1998). *An Overview of Query Optimization in Relational Systems.* Proceedings of ACM PODS, 34–43.

[3] Ioannidis, Y. E. (1996). *Query Optimization.* ACM Computing Surveys, 28(1), 121–123.

[4] Garcia-Molina, H., Ullman, J. D., & Widom, J. (2009). *Database Systems: The Complete Book*. Pearson Education.

[5] Leis, V., Radke, B., Gubichev, A., Kemper, A., & Neumann, T. (2015). *How Good Are Query Optimizers, Really?* Proceedings of the VLDB Endowment, 9(3), 204–215.

[6] Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018). *The Case for Learned Index Structures.* ACM SIGMOD, 489–504.

[7] Pavlo, A., et al. (2017). *Self-Driving Database Management Systems.* CIDR Conference.

[8] Marcus, R., Negi, P., Mao, H., Alizadeh, M., & Papaemmanouil, O. (2019). *Neo: A Learned Query Optimizer.* Proceedings of the VLDB Endowment, 12(11), 1705–1718.

[9] Marcus, R., et al. (2021). *Bao: Making Learned Query Optimization Practical.* ACM SIGMOD, 1275–1288.

[10] Trummer, I. (2019). *SkinnerDB: Regret-Bounded Query Evaluation via Reinforcement Learning.* ACM SIGMOD, 1153–1168.

[11] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

[12] Mnih, V., et al. (2015). *Human-Level Control through Deep Reinforcement Learning.* Nature, 518(7540), 529–533.

[13] Wooldridge, M. (2009). *An Introduction to MultiAgent Systems.* John Wiley & Sons.

[14] Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.

[15] Hilprecht, B., et al. (2020). *Deep Reinforcement Learning for Join Order Enumeration.* NeurIPS Workshop on Learned Systems.