OPEN ACCESS

Int. J. Creative Res. Publ. *Peer-Reviewed Journal* *Volume 1 Issue 4 [Jul-Aug] 2026*

International Journal of Creative Research and Publications

Double Blind Peer Reviewed Journal

Research Article

AI-Based SQL Query Optimization Using Large Language Models: A Framework for Intelligent Database Query Performance Enhancement

Tripti Sharma

Assistant Professor, IFTM University, Moradabad, Uttar Pradesh, India

Corresponding Author: *Tripti Sharma

DOI: https://doi.org/10.5281/zenodo.21618939

Abstract

Structured Query Language (SQL) is the standard language used to manage and retrieve data from relational databases. As modern applications generate massive volumes of data, optimising SQL queries has become increasingly important for improving database performance and reducing execution time. Traditional query optimisation techniques rely on rule-based and cost-based optimisers, which often struggle with complex queries, dynamic workloads, and evolving database environments.

Recent advancements in Artificial Intelligence (AI), particularly Large Language Models (LLMs), provide new opportunities for intelligent SQL query optimisation. LLMs can understand natural language, analyse SQL syntax, identify inefficient query patterns, and recommend optimised query structures. They can also assist developers in writing efficient SQL statements and improving query execution plans.

This paper presents a conceptual framework for AI-based SQL query optimisation using Large Language Models. The proposed approach integrates an LLM with a traditional Database Management System (DBMS) to enhance query rewriting, indexing recommendations, join optimisation, and execution plan analysis. The framework aims to improve query performance, reduce execution cost, and support database administrators in optimising complex SQL queries. The study also discusses the potential benefits, challenges, and future research directions in AI-assisted database optimisation.

Manuscript Information

- **ISSN No:** 3139-6313
- **Received:** 13-06-2026
- **Accepted:** 23-07-2026
- **Published:** 27-07-2026
- **IJCRP:**1(4); 2026: 09-13

- **Plagiarism Checked:** Yes
- **Peer Review Process:** Yes

How to Cite this Article

Sharma T. AI-Based SQL Query Optimization Using Large Language Models: A Framework for Intelligent Database Query Performance Enhancement. Int. J. Creative Res. Publ. 2026;1(4):09-13.

Access this Article Online

www.jcrp.in

KEYWORDS: SQL Query Optimisation, Artificial Intelligence, Large Language Models, Database Management System, Query Execution Plan, Cost-Based Optimisation, Natural Language Processing.

1. INTRODUCTION

Relational Database Management Systems (RDBMS) are widely used in industries such as banking, healthcare, education, e-commerce, and cloud computing. These systems manage enormous amounts of structured data, making efficient data retrieval an essential requirement. SQL (Structured Query Language) serves as the primary language for interacting with relational databases. However, poorly written SQL queries often lead to high execution costs, increased response time, excessive CPU utilisation, and unnecessary memory consumption.

Traditional database systems employ rule-based and cost-based query optimisers to determine efficient execution plans. While these optimisers perform well for many standard queries, they may struggle with highly complex joins, nested subqueries, dynamic workloads, and rapidly.

changing data distributions. As databases continue to grow in size and complexity, intelligent optimisation techniques are becoming increasingly necessary.

Artificial Intelligence has significantly transformed various fields of computer science, including software engineering, cybersecurity, and database management. Among AI technologies, Large Language Models (LLMs) such as GPT-based models have demonstrated remarkable capabilities in understanding programming languages, generating code, debugging software, and analysing structured queries. These capabilities make LLMs promising tools for assisting SQL query optimisation.

Unlike traditional optimisers that depend mainly on predefined rules and statistical information, LLMs can analyse SQL statements semantically, detect inefficient coding practices, recommend optimised query structures, and even explain execution plans in human-readable language. They can assist developers by suggesting better indexing strategies, reducing unnecessary joins, eliminating redundant conditions, and rewriting inefficient SQL statements.

The integration of AI with database systems introduces a new paradigm of intelligent query optimisation. Instead of replacing existing query optimisers, LLMs can function as intelligent assistants that complement traditional optimisation techniques. This collaborative approach can improve query execution efficiency while reducing the manual effort required by database administrators and software developers.

This paper proposes a conceptual AI-based framework that combines Large Language Models with traditional database query optimisation techniques. The framework aims to enhance SQL performance through intelligent query analysis, optimisation recommendations, and execution plan evaluation. The study also highlights current challenges, potential applications, and future research opportunities in AI-assisted database optimisation.

2. LITERATURE REVIEW

SQL query optimisation has been an active research area for several decades because database performance directly depends on the efficiency of query execution. Traditional query optimizers used in Relational Database Management Systems (RDBMS) rely on **rule-based** and **cost-based optimisation** techniques. These optimisers generate multiple execution plans and select the one with the lowest estimated execution cost. Although effective for many applications, they often face challenges when dealing with complex queries, changing workloads, and inaccurate database statistics.

Early research focused on heuristic optimisation methods, where predefined rules were applied to improve query execution. Examples include predicate pushdown, join reordering, projection optimisation, and index utilisation. Later, cost-based optimisation became the standard approach in modern database systems such as MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.

Recent advances in Artificial Intelligence have introduced new possibilities for intelligent database management. Machine Learning techniques have been applied to estimate query execution costs, predict resource utilisation, recommend indexes, and optimise execution plans.

These AI-based approaches have shown promising improvements over traditional optimisation methods in dynamic database environments.

Large Language Models (LLMs), including GPT-based models and other transformer-based architectures, have recently demonstrated exceptional capabilities in understanding programming languages and generating high-quality code. Researchers have explored their applications in SQL generation, SQL debugging, query explanation, and database education. LLMs can understand the semantic meaning of SQL statements and identify inefficient query structures that may not be detected through conventional optimisation techniques.

Several recent studies suggest that AI-assisted query optimisation can reduce execution time by recommending efficient JOIN orders, eliminating redundant conditions, improving indexing strategies, and rewriting SQL statements into more optimised forms. LLMs also provide natural language explanations of execution plans, making database optimisation easier for developers with varying levels of expertise.

Despite these developments, most existing research primarily focuses on SQL generation from natural language or automated code completion rather than comprehensive query optimisation. The integration of LLMs with traditional cost-based query optimisers remains an emerging research area requiring further investigation.

Therefore, combining Large Language Models with existing database optimisation mechanisms offers a promising direction for improving SQL performance while maintaining compatibility with current Database Management Systems.

3. Research Gap

Although significant progress has been made in SQL query optimisation, several research gaps still exist:

- Traditional cost-based optimisers depend heavily on database statistics, which may become outdated or inaccurate.

- Existing optimisation techniques have limited capability to understand the semantic intent of SQL queries.
- Most AI-based research concentrates on SQL generation rather than optimisation.
- Current database systems rarely integrate Large Language Models directly into the optimisation process.
- Developers still spend considerable time manually rewriting inefficient SQL queries.
- Few studies provide intelligent explanations for query execution plans using AI.
- Limited research exists on combining semantic understanding from LLMs with traditional execution-cost estimation.

These gaps indicate the need for an AI-assisted framework capable of providing intelligent query rewriting, optimisation recommendations, and execution plan analysis.

4. RESEARCH OBJECTIVES

The primary objectives of this research are:

1. To study the limitations of traditional SQL query optimisation techniques.
2. To investigate the role of Large Language Models in SQL optimisation.
3. To propose an AI-based framework for intelligent SQL query optimisation.
4. To improve query execution performance through semantic query analysis.
5. To reduce execution cost and response time.
6. To provide optimisation recommendations for developers and database administrators.
7. To evaluate the effectiveness of AI-assisted optimisation using standard database workloads.

5. Proposed AI-Based SQL Query Optimisation Framework

5.1 Overview

This research proposes an AI-Based SQL Query Optimization Framework that integrates a Large Language Model (LLM) with a traditional Relational Database Management System (RDBMS). The framework is designed to analyse SQL queries, detect inefficient patterns, recommend optimised alternatives, and assist database administrators in improving query performance.

Unlike conventional query optimisers, which rely mainly on cost estimation and database statistics, the proposed framework incorporates semantic understanding provided by LLMs. This enables the system to identify logical inefficiencies that may not be detected by traditional optimisation methods.

The framework does not replace the existing DBMS optimiser; instead, it acts as an intelligent assistant that enhances optimisation by providing additional recommendations before query execution.

5.2 Architecture of the Proposed Framework

The proposed framework consists of the following components:

1. User Query Input

The user submits an SQL query to the database system.

Example:

```
SELECT *
FROM Orders
WHERE YEAR(OrderDate)=2025;
```

2. SQL Query Analyser

The query analyser performs:

- Syntax checking
- Query parsing
- Identification of SQL clauses
- Detection of joins
- Detection of subqueries
- Identification of aggregate functions

This module prepares the query for AI analysis.

3. Large Language Model (LLM) Module

The LLM receives the parsed SQL query and performs semantic analysis.

Its responsibilities include:

- Detecting inefficient SQL patterns
- Suggesting query rewriting
- Removing unnecessary conditions
- Recommending JOIN improvements
- Suggesting index creation
- Identifying redundant operations
- Explaining execution plans in natural language

Example Recommendation:

Instead of

```
WHERE YEAR(OrderDate)=2025
```

recommend

```
WHERE OrderDate
BETWEEN '2025-01-01'
AND '2025-12-31'
```

This allows indexes on **OrderDate** to be used more effectively.

4. Query Optimisation Engine

The optimisation engine combines:

- Traditional Cost-Based Optimisation
- Rule-Based Optimisation
- AI Recommendations from the LLM

The engine selects the most efficient execution plan.

5. Execution Plan Generator

The optimised query is converted into an execution plan.

The plan includes:

- Table Scan
- Index Scan
- Join Order
- Estimated Cost
- CPU Usage
- Memory Consumption
- Execution Time

6. Query Execution
The optimized query is executed by the DBMS.
The optimised results are returned to the user with reduced execution time.

6. Methodology
The proposed research follows a conceptual and experimental methodology.
Step 1: Dataset Selection
A relational database containing sample business data will be created.
Example tables:
- Customers
- Orders
- Products
- Employees
- Sales

Step 2: Query Collection
Different SQL queries of varying complexity will be prepared.
Examples include:
- Simple SELECT queries
- JOIN queries
- Nested queries
- Aggregate queries
- GROUP BY queries
- ORDER BY queries
- Subqueries

Step 3: Baseline Performance Measurement
Each query will first be executed using the traditional database optimiser.
The following metrics will be recorded:
- Execution Time
- CPU Utilisation
- Memory Usage
- Query Cost
- Number of Rows Processed

Step 4: AI-Based Optimisation
The same SQL queries will then be analysed using the proposed LLM framework.
The LLM will provide recommendations such as:
- Query rewriting
- Index suggestions
- Join optimization
- Predicate optimization
- Removal of redundant clauses

Step 5: Performance Comparison
Both approaches will be compared using the collected performance metrics.
The evaluation will determine whether AI-assisted optimisation improves efficiency over traditional optimisation techniques.

Expected Advantages
The proposed framework is expected to:
- Reduce SQL query execution time.
- Improve database performance.
- Decrease CPU and memory utilisation.
- Assist developers in writing optimised SQL queries.
- Enhance understanding of execution plans.
- Reduce manual optimisation effort.
- Improve scalability for large databases.

7. Experimental Setup
7.1 Software Environment
The proposed framework can be evaluated using the following environment:

Component	Specification
Operating System	Windows 11 / Ubuntu 22.04
Database	PostgreSQL 16 / MySQL 8.0
Programming Language	Python 3.12
AI Model	Large Language Model (GPT-based or open-source LLM)
IDE	Visual Studio Code / PyCharm
Hardware	Intel Core i5 or higher, 16 GB RAM

7.2 Evaluation Metrics
The performance of the proposed framework can be evaluated using:
- Query Execution Time (ms)
- Query Cost
- CPU Utilisation (%)
- Memory Usage (MB)
- Number of Logical Reads
- Optimisation Accuracy
- User Satisfaction (optional)

8. RESULTS AND DISCUSSION
To evaluate the effectiveness of the proposed framework, identical SQL queries should be executed using:
1. Traditional DBMS Query Optimiser
2. AI-Based SQL Query Optimisation Framework

The following comparisons should be analysed:
- Execution time before and after AI optimisation.
- Query execution cost.
- CPU and memory consumption.
- Improvements achieved through query rewriting.
- Impact of index recommendations.
- Efficiency of optimised JOIN operations.

The experimental results should demonstrate whether the proposed framework improves SQL query performance while maintaining correctness.

9. CONCLUSION
SQL query optimisation is an essential aspect of database performance. Traditional optimisation techniques perform well for many workloads but may be limited when handling complex queries and dynamic environments.
This paper proposed an AI-Based SQL Query Optimisation Framework that integrates Large Language Models with

traditional database optimisation techniques. The framework enables semantic analysis of SQL queries, intelligent query rewriting, indexing recommendations, and execution plan interpretation.

The proposed approach is expected to improve query execution efficiency, reduce computational cost, and assist developers and database administrators in optimising SQL queries. As AI technology continues to evolve, integrating LLMs into DBMS optimisation has significant potential for future intelligent database systems.

10. Future Work

Future research may include:

- Integration with cloud-native database platforms.
- Fine-tuning LLMs specifically for SQL optimisation tasks.
- Support for NoSQL and distributed databases.
- Real-time adaptive query optimisation.
- Reinforcement Learning for continuous optimiser improvement.
- Integration with autonomous database management systems.
- Evaluation on large-scale enterprise datasets.

REFERENCES

1. Silberschatz A, Korth HF, Sudarshan S. *Database System Concepts*. 7th ed. New York: McGraw-Hill Education; 2019.
2. Elmasri R, Navathe SB. *Fundamentals of Database Systems*. 7th ed. Boston: Pearson; 2016.
3. Chaudhuri S. An overview of query optimisation in relational systems. In: *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*. New York: ACM; 1998. p. 34–43.
4. Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Minneapolis, MN: ACL; 2019. p. 4171–4186.
5. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: *Advances in Neural Information Processing Systems*. Vol. 30. Red Hook (NY): Curran Associates Inc.; 2017.
6. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. Language models are few-shot learners. In: *Advances in Neural Information Processing Systems*. Vol. 33. Red Hook (NY): Curran Associates Inc.; 2020. p. 1877–1901.
7. OpenAI. *GPT models technical documentation*. San Francisco (CA): OpenAI; 2025.
8. Recent peer-reviewed papers (2024–2026) on AI-assisted SQL optimisation and large language models from IEEE Xplore, ACM Digital Library, Springer Nature, and Elsevier.
9. Yao Z, Li H, Zhang J, Li C, Chen H. A query optimisation method utilizing large language models. *arXiv*. 2025; arXiv:2503.06902. doi:10.48550/arXiv.2503.06902.
10. Song M, Zheng M. A survey of query optimization in large language models. *CoRR*. 2024;abs/2412.17558. doi:10.48550/arXiv.2412.17558.
11. Bagui S, et al. Large language models in database management system optimization: a survey. *ACM Transactions on Intelligent Systems and Technology*. 2026. doi:10.1145/3829078.
12. Milicevic B, Babovic Z. A systematic review of deep learning applications in database query execution. Journal of Big Data. 2024;11(1):173. doi:10.1186/s40537-024-01025-1.

About the author

Tripti Sharma is an Assistant Professor at IFTM University, Moradabad, Uttar Pradesh, India. Her academic interests include computer science, database management systems, artificial intelligence, machine learning, and emerging technologies. She is actively involved in teaching, research, and scholarly publications, with a focus on advancing innovative and practical solutions in computing and information technology.