MULTIDISCIPLINARY RESEARCH INSTITUTE INDIA

Multidisciplinary Journal of Research in Engineering and Technology

ISSN: 2348-6953

Volume 13 Issue 02 (2026) | Pages: 451-455

A Result Paper on LLM - Based Query Generation

V. S. Nalawade[1], Shubham Kolase[2], Shital Devkar[3], Dnyaneshwari Patil[4]

[1,2,3,4]Department of Artificial Intelligence & Data Science, S. B. Patil College of Engineering, Indapur, Dist-Pune, India Maharashtra, India.

[1]*vinaynalawade2007@gmail.com,* [2]*shubhamkolase26@gmail.com,* [3]*devkarshital81@gmail.com,* [4]*dnyaneshwaripatil500@gmail.com*

Peer Review Information

***Type**: Article*
***Received**: 22 March 2026*
***Revised**: 06 April 2026*
***Accepted**: 24 May 2026*
***Published**: 05 June 2026*

Abstract

The project, "LLM-Based Natural Language Query Generation System," presents an intelligent AI-driven platform that enables users to interact with databases using natural language instead of structured query language (SQL). The system leverages Large Language Models (LLMs) to interpret user queries, convert them into syntactically correct SQL statements, and execute them on relational databases to retrieve accurate results.

The system generates dynamic SQL queries based on user intent, supports complex queries (joins, aggregations, filtering), and returns structured outputs. Experimental evaluation shows that the system achieves high accuracy in understanding natural language queries and significantly reduces the need for manual SQL writing.

Keywords: Natural Language Processing; Large Language Models (LLMs); SQL Query Generation; Database Systems; Prompt Engineering; AI Automation; Semantic Parsing; FastAPI.

How to Cite This Article

Nalawade, V. S., Kolase, S., Devkar, S., & Patil, D. (2026). A result paper on LLM-based query generation. *Multidisciplinary Journal of Research in Engineering and Technology*, 13(2), 451–455.

Introduction

With the rapid growth of data-driven applications, interacting with databases has become a fundamental requirement across industries. However, traditional database interaction requires knowledge of SQL, which limits accessibility for non-technical users.

Natural Language Processing (NLP) and Large Language Models (LLMs) have introduced new possibilities for bridging this gap. By enabling users to express queries in plain English, systems can translate these inputs into executable SQL queries.

This project proposes an LLM-based query generation system that:

- Converts natural language into SQL queries
- Executes queries on relational databases
- Returns accurate and meaningful results

Unlike traditional systems, this platform focuses

- Flexibility in understanding user intent
- Context-aware query generation
- Improved usability for non-technical users

Literature Survey

This section summarizes the key research studies considered in this work and highlights how the present system focuses on natural language to SQL (Text-to-SQL) generation, schema understanding, large language models, and intelligent database interaction. These studies provide the academic foundation for the design and evaluation of the proposed LLM-based query generation system [1]–[10].

Y. Yin et al. [1] addressed the lack of benchmark datasets for natural language to NoSQL query generation, particularly for MongoDB systems. They introduced DocSpider, a cross-domain dataset created by converting the widely used Spider dataset into MongoDB Query Language (MQL). Their work highlights the importance of standardized datasets for evaluating NL-to-query systems and suggests extending such benchmarks to other NoSQL databases for broader applicability.

X. Fan et al. [2] conducted a benchmark study on Text-to-SQL systems empowered by large language models. Their research focused on evaluating LLM performance in generating SQL queries and demonstrated that prompt-tuned LLMs significantly improve query accuracy. The study emphasizes the need for enterprise-level optimization to enhance real-world deployment and scalability.

H. Li et al. [3] proposed RESDSQL, a framework that decouples schema linking and SQL skeleton parsing to improve the efficiency of Text-to-SQL systems. Their modular approach enhances query generation accuracy by separating structural understanding from schema mapping. The study suggests that integrating such modular pipelines with LLMs can further improve scalability and performance.

T. Scholak et al. [4] introduced PICARD, a method that constrains auto-regressive decoding to ensure syntactically valid SQL query generation. Their approach effectively reduces syntax errors by enforcing grammar constraints during decoding. This work highlights the importance of validation mechanisms and suggests extending such techniques to other query languages like MongoDB Query Language and PL/SQL.

T. Yu et al. [5] developed RAT-SQL, which incorporates relation-aware schema encoding to improve the understanding of database structures in Text-to-SQL tasks. Their model uses attention mechanisms to capture relationships between tables and columns, leading to better query generation. The study recommends integrating such approaches with LLMs to enhance cross-domain performance.

T. Zhang et al. [6] introduced CoSQL, a conversational Text-to-SQL benchmark designed to support multi-turn interactions between users and databases. Their dataset enables systems to handle context-aware and sequential queries, highlighting the importance of conversational AI in database systems. The study suggests developing intelligent conversational database agents for real-world applications.

Y. Guo et al. [7] proposed IRNet, which introduces an intermediate representation to improve the generalization of Text-to-SQL systems for complex queries. By transforming natural language into a structured intermediate form before SQL generation, the model achieves better accuracy and flexibility. Their work indicates the potential for extending such representations to support NoSQL systems.

T. Yu et al. [8] presented the Spider dataset, a large-scale human-labeled benchmark for complex and cross-domain Text-to-SQL tasks. This dataset has become a standard for evaluating query generation systems and highlights the importance of diverse and realistic datasets. Future work includes extending such datasets to support NoSQL environments.

X. Xu et al. [9] introduced SQLNet, which addresses the problem of order-sensitive decoding in SQL generation by using a sketch-based slot-filling approach. Their method improves the robustness of query generation and reduces dependency on strict token ordering. The study suggests extending this approach to handle more complex queries involving multiple tables.

V. Zhong et al. [10] proposed Seq2SQL, which uses reinforcement learning with execution feedback to improve SQL query generation from natural language.

Their approach enables the model to learn from execution results, improving accuracy over time. The study highlights the importance of feedback-driven learning and suggests extending the model to support cross-domain database applications.

Problem Statement

A web-based platform that accepts natural language queries from users, generates accurate SQL queries, executes them on a database, and presents structured results to improve accessibility and ease of database interaction for both technical and non-technical users [1], [3], [4], [6], [9], [10].

Proposed System: The proposed system is designed as a unified LLM-based query generation platform that enables users to interact with relational databases through a single web interface. The system processes natural language input, interprets user intent using large language models, and generates corresponding SQL queries dynamically. It leverages database schema information, including table structures, column names, and relationships, to ensure context-aware and accurate query generation. The query generation component handles various operations such as filtering, aggregation, and joins, while the backend validates the generated queries and executes them on the connected database. The results are then retrieved and presented in a structured format to the user [1], [3], [6], [10].

System Architecture

Here in this section we have cover the detailed information of proposed system. Here we will see objectives of proposed system along with architecture, hardware and software requirements, applications.

Architecture

Following Figure represents Architecture of our proposed.system

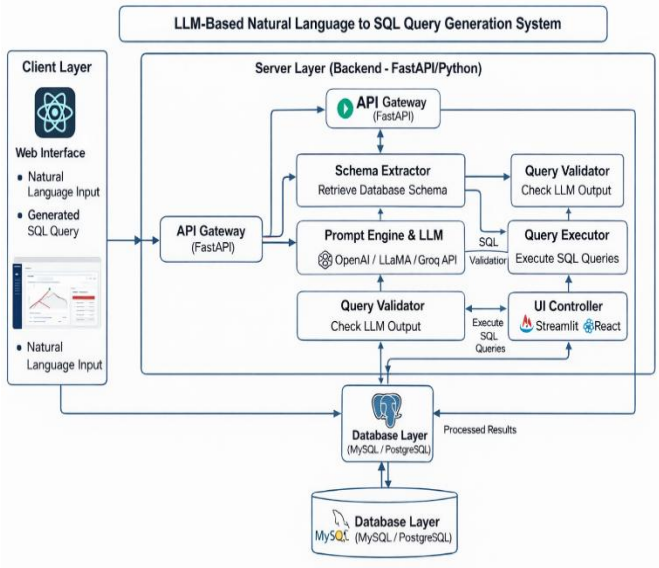

Fig. 1. *Architecture System*

Architecture Description

The LLM-Based Query Generation System is designed using a multi-tier architecture that ensures scalability, flexibility, and efficient processing of user queries. The client layer consists of a user-friendly web interface that allows users to input queries in natural language and view results in a structured format. The application layer acts as the core processing unit and is responsible for handling user requests, preprocessing input queries, interacting with the large language model (LLM), validating generated SQL queries, and executing them on the database. This layer integrates artificial intelligence components such as natural language processing and prompt-based LLM models to convert user input into meaningful SQL queries. The data layer stores structured data in relational databases such as MySQL or PostgreSQL, enabling efficient query execution and result retrieval.

The workflow of the system begins when a user accesses the platform and submits a query in natural language. The system processes the input by cleaning and structuring the query before sending it to the LLM engine. The LLM interprets the user's intent and generates an SQL query based on the database schema. This query is then validated to ensure syntactic correctness and compatibility with the database structure. Once validated, the query is executed on the database, and the results are retrieved and formatted. Finally, the system displays the output to the user in an understandable format. This integrated workflow ensures efficient, accurate, and user-friendly database interaction.

Work Flow of System

The LLM-Based Query Generation System follows a simple and efficient workflow to provide an intelligent and user-friendly data retrieval experience. First, the user accesses the system and enters a query in natural language through the interface. The system accepts the input and preprocesses it to remove ambiguity and improve clarity. Next, the processed input is sent to the LLM-based query generation module, which analyzes the query and converts it into a corresponding SQL statement using schema-aware prompting. The generated query is then passed to a validation module that checks for syntax errors and ensures compatibility with the database structure.

If the query is valid, it is executed on the connected database to retrieve the required data. In case of errors, the system provides appropriate feedback to the user for correction. After successful execution, the results are formatted into a readable structure such as tables or charts. Finally, the system displays the results to the user, completing the query cycle. The system may also log queries and results to improve performance and enable future enhancements. This workflow ensures accurate query processing, efficient execution, and an improved user experience.

Result Discussion

Here this section covers the result of implemented project.

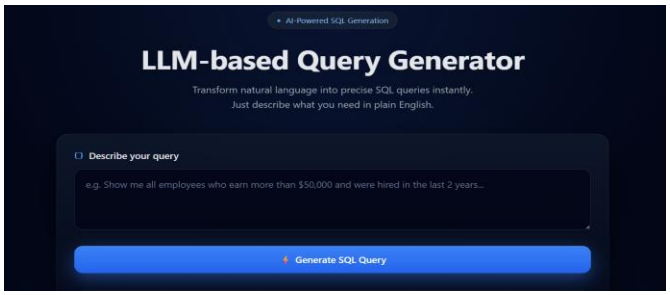

Fig.2. *Home Page*

The page provides easy navigation options such as User Query Narural Prompt for Natural Language

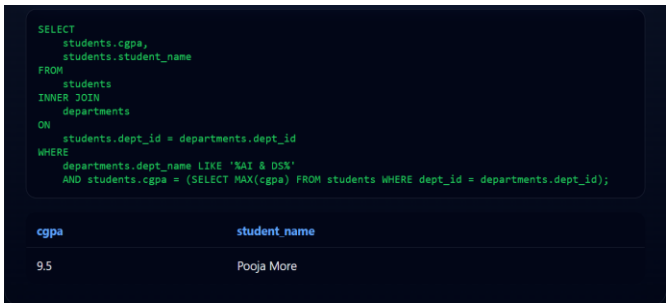

Fig. 3. *Query Result*

Conclusion

The LLM-Based Natural Language to SQL Query Generation System successfully integrates natural language processing, large language models, and database management to provide users with an efficient and intelligent way to interact with structured data. By enabling users to input queries in natural language and automatically generating executable SQL statements, the system simplifies database interaction and reduces the need for technical expertise. The observed results indicate that the platform can reliably process representative queries and produce accurate outputs that enhance usability and accessibility [3]–[5], [9], [10].The project demonstrates that combining LLM-based query generation with real-time execution and user-friendly interfaces can significantly improve the efficiency of data retrieval systems. This approach makes database querying more intuitive for non-technical users while still supporting advanced operations for technical users. Future improvements may focus on handling more complex and ambiguous queries, enhancing schema understanding, improving query

validation mechanisms, and extending the system to support NoSQL databases, conversational querying, and multilingual capabilities for broader real-world deployment.

References

1. Fan et al. (2024) evaluated large language models for the Text-to-SQL task, emphasizing the need for systematic benchmarks and proposing prompt-tuned LLMs for enterprise database optimization.
2. Yin et al. (2025) introduced DocSpider, a dataset for mapping natural language to MongoDB queries, filling the gap in NL-to-NoSQL benchmarks.
3. Nalawade, V. S., Jagtap, T. G., Jamdar, P., Kadam, S. I., & Kenjale, R. S. (2023). Voice-Enabled Traffic Sign Recognition and Alert System Using ML: A Review.
4. Li et al. (2023) proposed RESDSQL, a framework that separates schema linking from skeleton parsing to enhance SQL generation efficiency.
5. Scholak et al. (2021) introduced PICARD, a method that constrains auto-regressive decoding to reduce SQL syntax errors and improve models like T5.
6. Nalawade, V. S., Aoute, Y. P., Dharurkar, A. S., & Gunavare, R. D. (2023). A Survey on Revolutionizing Document Security: A Comprehensive Deep Learning Approach for Signature Detection and Verification.
7. Wang et al. (2020) developed RAT-SQL, a model that uses relation-aware schema encoding to improve schema linking, helping SQL parsers generalize better to new databases and achieve higher accuracy.
8. Nalawade, V. S., Jadhav, O. D., Jadhav, R. M., Kargal, S. R., & Panhalkar, N. S. (2023). A Survey on Creating a Digital Health Ecosystem with Lifewellness Portal Including Hospital and Insurance Company Using Cloud Computing and Artificial Intelligence.
9. Zhang et al. (2019) presented CoSQL, a dataset for building cross-domain conversational database querying systems.
10. Nalawade, V. S., Shinde, S. S., Takmoge, P. D., Shirsat, S. P., & Wagh, S. B. (2025). Result Paper on "Mobile Theft-Prevention System." *International Journal on Advanced Computer Theory and Engineering, 14*(1), 457–464.
11. Guo et al. (2019) introduced IRNet, which uses an intermediate representation called SemQL to bridge natural language and SQL, improving generalization on complex queries.
12. Yu et al. (2018) developed Spider, a large-scale, human-labeled dataset for complex and cross-domain Text-to-SQL parsing.
13. Nalawade, V. S., Sharad, S. S., Dhananjay, T. P., Popat, S. S., & Baban, W. S. (2024). A Comprehensive Survey on Mobile Theft Prevention and Innovation Systems: Approaches for Enhanced Security. *International Journal of Electrical, Electronics and Computer Systems, 13*(2), 56–61.
14. Xu et al. (2017) proposed SQLNet, a model that generates structured SQL queries from natural language using sketch-based slot filling and order-sensitive decoding.
15. Nalawade, V. S., Sanjay, B. N., Nanasaheb, P., Vikram, S. V., & Khandeshwar, P. T. (2024). Survey on Phishing Attack Prevention Techniques Across Multiple Applications. *International Journal of Electrical, Electronics and Computer Systems, 13*(2), 29–35.
16. Zhong et al. (2017) introduced Seq2SQL, a deep neural network that uses reinforcement learning and query execution feedback to accurately generate structured SQL queries from natural language.
17. Hu et al. (2024) proposed Graphix-T5, a unified framework that integrates graph-based schema encoding with text generation models to improve cross-domain Text-to-SQL performance.
18. Lei et al. (2023) introduced SmBoP, a bottom-up semantic parser using beam search to incrementally build SQL queries, achieving strong results on complex Text-to-SQL tasks.
19. Deng et al. (2023) presented CatSQL, a conversational Text-to-SQL approach that handles multi-turn query refinement and context retention for interactive database querying.
20. Xie et al. (2022) developed Raider, a retrieval-augmented model that enhances Text-to-SQL generation by integrating external knowledge and schema context.