IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

BaseMind: A Natural Language to SQL Interface Using LangChain for High-Speed Database Querying.

[1]Advay Ashish Acharekar, [2]Margesh Bhatkar, [3]Sairaj Davkhar, [4]Durgesh Surwade, - Students

Department of Computer Science & Engineering, Bharat College of Engineering, Badlapur, Thane District, Maharashtra, India

[1]Asst.Prof.Sonali Patil – Professor, Department of Computer Science and Engineering (AIML), [1]Bharat College of Engineering, Badlapur, Maharashtra, India.

Abstract: The problem of turning what people say into something computers can understand, like SQL is still an issue for people who are not tech savvy and want to work with complicated databases. Usually, people have used systems that're very strict or models that are good at learning, like LSTMs but these often take a long time to work and are not very flexible. This paper is about BaseMind, a way of talking to databases using natural language that makes it easier to get data by using the LangChain system and the GROQ engine. By putting a user interface on the front and a strong database on the back the system can turn what people say into SQL quickly. Tests show that using the hardware from GROQ and making the questions clearer makes the system work much faster and still gives accurate results even with complicated questions that involve multiple tables and sums. BaseMind is a way to make data available to everyone. It helps to close the gap, between what people say and what computers can understand.

Keyword – Large Language Model (LLM), Generative AI, LangChain, SQL Query Generation, Natural Language Interface to Database, Streamlit Framework.

I. Introduction

The modern business world relies on databases to store information. These databases use something called Structured Query Language or SQL for short to manage the data. SQL is the way to work with data but it is hard for people who are not tech experts to use. This means that business people researchers and teachers have a time getting the information they need from the databases. They have to wait for special teams to help them. This slows down the decision-making process.

We really need a way for people to interact with databases using natural language like how we talk to each other. This is called a Natural Language Interface to Databases or NLIDB for short. NLIDB should let people work with data easily as they have conversations. People have tried to solve this problem but the old ways were not very good. They used keyword matching and then moved on to more complex systems like Long Short-Term Memory networks.. These systems were slow and not very accurate which made them not very useful for real-world applications. They also had a time handling complex queries, which meant that people had to choose between getting a simple answer quickly or a more detailed answer slowly.

Newer systems, like Large Language Models are trying to change this.. We still need a system that can generate SQL code quickly and accurately and can handle real-time data. That is where BaseMind comes in. BaseMind is a NLIDB system that uses a special framework called LangChain to manage database schemas and a special chip called the Groq LPU to make the system faster. This means that BaseMind can answer questions in less than a second. The system is built on a MySQL foundation and has a user-friendly interface that makes it easy for people to get the information they need. BaseMind shows that we do not have to choose between making data accessible to everyone and having a system that's fast and powerful. We can have both.

The rest of this paper is organized into sections. Section II talks about NLIDB systems that already exist. Section III discusses the challenges of turning language into SQL code including how hard it is to understand what people mean and how to map that to a database schema. Section IV looks at the technology used in BaseMind including Large Language Models and special hardware. Section V explains how BaseMind works, from the user interface to the database backend. Finally Section VI concludes the paper. Suggests areas, for future research and includes a list of references.

II. Related Works

The evolution of Natural Language Interface to Database (NLIDB) systems has moved from strict rule-based linguistic parsing to advanced deep learning architectures. This change provides the important context for the architectural choices made for the BaseMind framework. One of the first pioneers in this field was the LUNAR system, developed in 1971 to help geologists analyze chemical data from lunar rock samples. It used Augmented Transition Network (ATN) parsing to convert English queries into a specific database interrogation language. However, the system was very domain-specific and not portable. It worked well only within the limited scope of lunar geology and could not adapt to different database schemas without extensive manual adjustments to its linguistic rules.

To address the portability problems with LUNAR, CHAT-80 was developed in 1982 using Prolog. This system turned natural language questions into logic clauses to query a geographical database. By representing knowledge as logical facts, CHAT-80 moved past the strict ATN structures and allowed for more efficient and modular query processing. Despite this progress, CHAT-80 was still limited by its reliance on a closed-world assumption. It struggled with ambiguous phrasing in natural language and could not scale to the large multi-table relational databases commonly found in modern businesses.

The rise of deep learning from 2014 to 2018 saw researchers apply Sequence-to-Sequence (Seq2Seq) models using Long Short-Term Memory (LSTM) networks to tackle the scalability and ambiguity challenges in logic-based systems. These models treated SQL generation as a machine translation task, enabling the system to learn patterns from data rather than depending on hand-coded rules. However, recent studies, such as those by Majhadi and Machkour in 2024, have shown that LSTM-based models often face "vanishing gradient" issues when handling long, complex queries. Additionally, they frequently struggle to keep the SQL syntax intact, leading to incorrect table names or invalid join conditions that prevent successful execution.

From 2019 to 2022, the introduction of the Transformer architecture and BERT-based encoders effectively resolved the structural problems of LSTMs. They used self-attention mechanisms to understand the bidirectional relationships between user queries and database schemas. These models matched specific words in questions with corresponding columns in tables, significantly improving semantic accuracy. Despite this increased precision, these models remain resource-intensive and need considerable GPU power to operate effectively. The resulting latency is often too high for real-time applications, and these systems typically require extensive fine-tuning on specific datasets like Spider or WikiSQL to deliver reliable performance across various fields.

From 2023 to the present, cutting-edge systems have shifted towards Generative AI and Large Language Model (LLM) orchestration using frameworks like LangChain to overcome the fine-tuning bottleneck. These systems utilize in-context learning to instantly adapt to new database schemas without needing specialized training. While they are highly flexible and generalized, these systems still encounter problems with execution speed and high API costs. Cloud-based LLMs cause significant network delays that hinder

a truly conversational user experience. BaseMind tackles this issue by incorporating hardware-accelerated inference through the Groq LPU, bridging the gap between high-level machine intelligence and real-time operational responsiveness.

III. Problem Statement

3.1Core Research Question

The fundamental problem this project solves is the technical hurdle between non-technical stakeholders and the relational database that holds vital information. Though data drives modern decision-making processes, it remains "locked in" to the world of decision-making due to the necessary requirement of SQL fluency. The fundamental problem this project solves can be summarized as: How do we construct a Natural Language Interface to Database (NLIDB) that enables the near-instant translation of human thought into code without the traditional trade-offs of high latency, high computational cost, or the constant requirement of fine-tuning? BaseMind solves this problem by creating a high-speed bridge that enables any user to interface with a MySQL database in plain English.

3.2. Challenges Arising from the Problem

The lack of direct questioning of the database without the need to go through an intermediary presents several key challenges that affect the efficiency of organizations. Firstly, there is the challenge of Operational Bottleneck, where there is often a lag in completing data-related tasks because business users are forced to wait for the IT team to develop and execute queries. Secondly, there is the Linguistic Gap challenge, where there is often confusion and miscommunication, as seen in the fact that when a non-technical user is not able to "speak" to the data, there is often loss of communication during the manual relaying of the queries. Finally, there is the challenge of Inaccessibility at Scale, where the complexity of the database, with hundreds of tables and relationships, makes the few SQL experts in the organization a single point of failure, thereby failing to achieve the democratization of information across the organization/enterprise.

IV. Existing System

The history of Natural Language Interface to Database (NLIDB) systems has been influenced by a number of fundamental architectural advancements that aimed to improve the interface between human beings and data structures. One of the first and most important measures for judging the effectiveness of a system was the Rule-Based Linguistic Parser that utilized a set of predefined rules and mapping functions to translate a given sentence structure into a SQL query. Although these systems were extremely effective from a standpoint of predictability and consistency, they were also extremely fragile and would break the moment a user asked a query that was not formulated according to the predetermined syntax. This lack of flexibility meant that the parsers were extremely domain-dependent and had to be exhaustively configured for every different database schema, a factor that BaseMind avoids by making use of the generalized reasoning abilities of Large Language Models.

Another major advancement in the field was introduced by LSTM-based Sequence-to-Sequence Models, which considered the generation of SQL queries as a neural machine translation approach. This approach was highly successful in moving away from the traditional approach of writing rules. However, there was a "black box" issue in these models, as the logic behind the query was not easily understandable. More critically, as discussed in recent literature, these models are also known to be highly prone to structural hallucinations and "vanishing gradients" when complex queries involving multiple joins are considered. This failure is addressed by the proposed system, as it utilizes a robust approach through the LangChain framework, which maintains the structural integrity of the MySQL schema during translation, thereby increasing the accuracy.

Recently, Cloud-Based LLM Interfaces have been introduced as an efficient way to interact with Text-to-SQL queries, utilizing pre-trained models to effectively interpret complex queries. Unfortunately, these systems are hindered by considerable network and inference latency, taking multiple seconds to return a result due to communication overhead and high-token processing. This disrupts the interactivity and flow of data exploration, making the system sluggish and "non-human" in nature. Moreover, the cost of high-

tier API calls can be expensive for widespread institutional adoption. This is where BaseMind attempts to fill the performance gap by utilizing the GROQ, which enables hardware acceleration for sub-second inference. This provides an efficient and seamless user experience, removing the performance gap that prevents cloud-based AI from becoming a seamless experience for real-time data exploration.

V. Proposed Model

5.1. Presentation Layer: Streamlit-Based User Interface

The Presentation Layer acts as a primary entry point for the user. This layer emphasizes accessibility. This layer was implemented using the Streamlit library. This layer is responsible for natural language input from the user through a simple, intuitive web interface. This layer differs from a typical command-line interface. This layer provides a 'human-centric' experience, allowing users to input questions and receive results in an organized tabular form. This layer manages asynchronous communication between the client and server. This ensures the user remains aware of the status of the system during the query. By utilizing reactive components from the Streamlit library, the interface can display query execution times, further stressing the low-latency benefits.

5.2. Orchestration and Logic Layer: LangChain and GROQ Integration:

The Orchestration and Logic Layer is the "intelligence" part of BaseMind and is where the actual work of converting text to SQL happens. This section makes use of the LangChain framework to orchestrate the complicated workflow involved in schema retrieval, prompt building, and output parsing. When the user makes a query, LangChain retrieves the necessary MySQL metadata to give context to the Large Language Model. To address the problems of latency found in current systems, the Orchestration and Logic Layer make use of the Groq LPU (Language Processing Unit). The LPU is essentially a hardware accelerator, giving the LLM the capability to execute in under a second, allowing to work with the schema and produce syntactically correct SQL almost immediately.

5.3. Data Persistence Layer: MySQL Relational Backend

The Data Persistence Layer is the foundation of the architecture, providing a strong relational database implemented in MySQL. This layer is where the actual data of the organization is stored, and it runs the SQL queries that are generated by the Orchestration Layer. In this section, the emphasis is on the secure interaction between the code generated by AI and the actual database tables. The system ensures that the SQL code is correct according to the dialect and syntax of the MySQL environment, including correct JOIN statements and aggregate functions. By separating the AI code from the database engine, BaseMind ensures data integrity and at the same time offers a flexible environment that can be used for different types of enterprise data without changing the actual application logic.

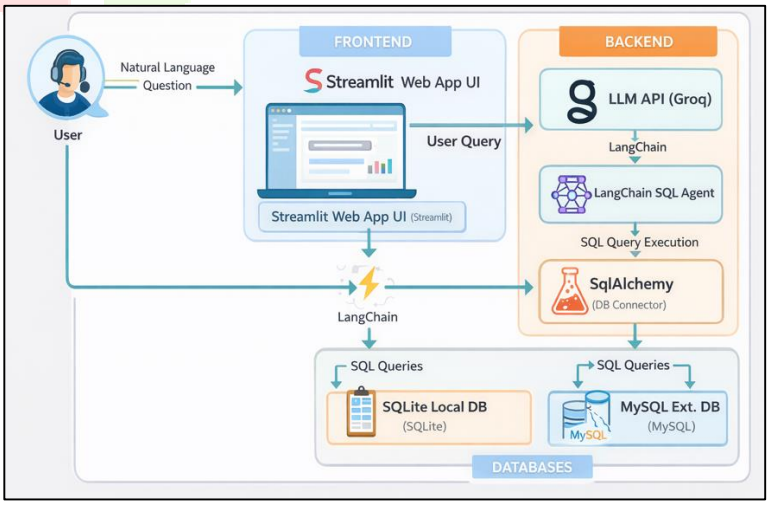

Fig 5.1 System Architecture

The above figure represents **the system architecture** that enables users to interact with SQL databases using natural language. At the front, the user interacts through a web interface built with Streamlit, where queries are entered in plain English. This frontend also handles authentication, session management, and displays results. The user query is then sent to the backend, where a Large Language Model is accessed via the Groq API. Using LangChain's SQL agent, the system interprets the natural language input and

converts it into a structured SQL query. The generated SQL is executed through SQLAlchemy, which acts as a bridge between the application and the databases. The system supports both a local SQLite database for demonstration and an external MySQL database for real-world use cases. Finally, the query results are returned to the frontend and displayed to the user in a readable format, completing the interaction loop.

VI. Methodology

The development process for BaseMind was achieved through a formalized four-phase process methodology. This process ensured that the fundamental logic was validated in a controlled environment before moving to a sophisticated, complex relational database system with hardware acceleration for inference.

6.1. Phase 1: Preparing Database using Sqlite3

The first phase involved setting up a local "sandbox" environment to test the Text-to-SQL logic. A demo database, student_data.db, was set up using SQLite3 to mimic a typical academic database structure. In this phase, the LangChain agent's ability to perform schema introspection, inspecting database table structures and column types was critical. This phase also helped to eliminate the impact of network latency and authentication protocols. By utilizing a local URI scheme (sqlite:///), this project set up a baseline for query accuracy and verified the efficacy of the Zero-Shot React Prompt Strategy.

6.2. Phase 2: MySQL Database Preparation

Once this logic has been verified, the project shifted into a robust MySQL environment to mimic real-world enterprise settings. This involved the structural design of a relational database management system (RDBMS) using the MySQL Workbench application. The EER (Enhanced Entity-Relationship) model was created within the Workbench environment to manage multiple table joins, foreign key constraints, and multiple connections. The preparation phase focused on data integrity, defining the primary keys for each table, and optimizing table indexes to create a realistic environment that would test the AI agent's ability to relate natural language to a non-trivial physical schema.

6.3. Phase 3: Model Building

The third phase involved the architectural assembly of the BaseMind brain and the establishment of its primary communication channels. Dynamic connection layer engineering was conducted using SQLAlchemy. This involved the use of the create engine function to bridge the Python environment and the MySQL backend established in the previous phase. To ensure industrial-grade security, urllib.parse.quote_plus was implemented to encode database credentials, thereby protecting against special character attacks. At the same time, the software stack was integrated with the Groq to overcome the high latency problem associated with cloud LLMs. This phase involved the configuration of the ChatGroq interface using the llama-3.3-70b-versatile model. The methodology adopted involved the use of the SQLDatabaseToolkit. This allowed the agent to "reason" before "acting," in writing the SQL. In addition, the Streamlit frontend was integrated. This provided a transparent step-by-step visualization of the AI's thought process using the StreamlitCallbackHandler.

6.4. Phase 4: Testing and Validating Results

The final phase was focused on the empirical testing and result verification of the system. The system was subjected to a variety of natural language queries that varied from simple data retrieval queries like "Show all students" to complex queries like "Calculate the average GPA per department." The performance of the system was measured along two key parameters: Inference Latency (the time it takes from the user input to the generation of the SQL code) and Execution Accuracy (the difference between the output generated by the AI and the actual ground truth SQL code that a human would write).

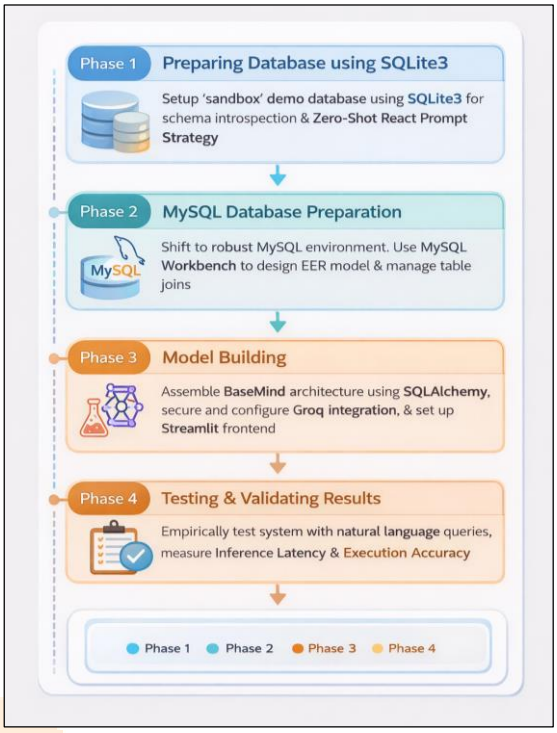

Fig 6.1. Methodology

VII. Results

7.1. User Login and Sign Up

The main entry point for the application is a security gateway that is specifically designed for the management and limitation of access to the core features of the system. This allows the user to either log in with their existing details or create a new account with the toggle functionality. The requirement for the user to log in successfully before proceeding with the application ensures that the underlying data connections and the tools for data processing are not abused. This gateway acts as a necessary evil in the maintenance of a secure environment for the user's personal settings and the databases that are connected.

7.2. Chat Interface

Once the authentication is successful, the user is then routed to a centralized command hub, which helps them interact with the data in a direct manner. The dashboard includes a personalized greeting and a sidebar that helps the user manage their current session. The sidebar helps the user choose the data source they want to interact with and provide the required keys to the processing engine to enable it to work properly. The design is intended to be clean and separate from the actual conversation space, allowing the user to submit their queries and obtain information in a clean and intuitive window.

7.3. Result Synthesis

The final functional element in the system is the execution and output layer, which processes the questions posed by the user into a form that is useful for action. Once the request has been made, the system engages in a complex process that takes several steps to determine the best way to retrieve the requested data. As indicated in the execution logs, the system gives a transparent view of these steps as the system checks the data structure and checks its logic before arriving at a final answer. The answer is given in both a natural language form and a table form, giving precise data such as specific dates for a booking in an instant.

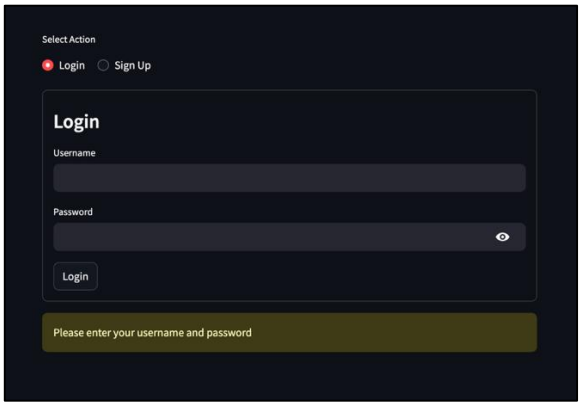

Fig 7.1. Login Page

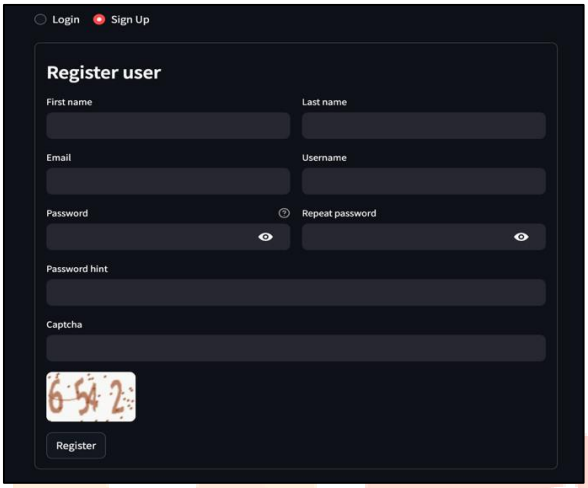

Fig 7.2. Signup Page

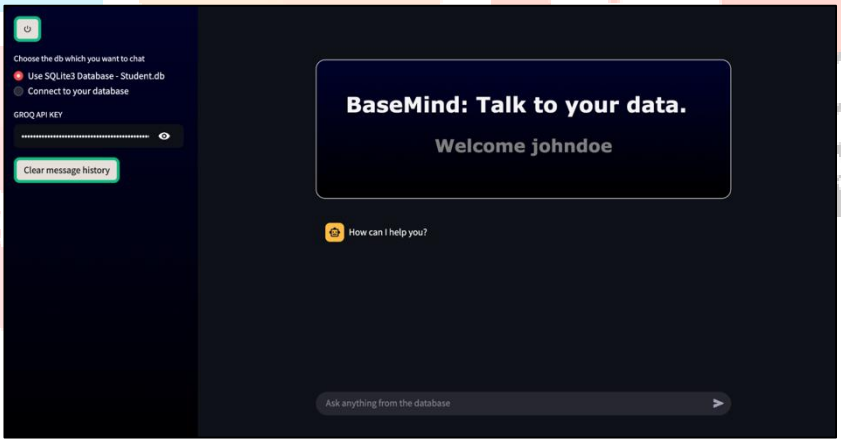

Fig 7.3. Chat Interface

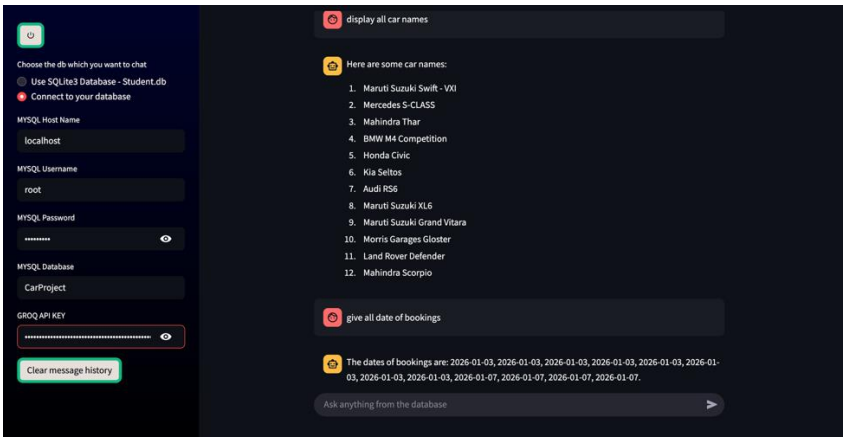

Fig 7.4. Query Output

VIII. Conclusion

The development of BaseMind shows the potential of Large Language Models in integration with relational databases to provide a high-performance interface for data interaction. By moving away from traditional and manual methods of querying data using SQL, this project has effectively reduced the barriers to accessing data for users who are not proficient in using technical interfaces. The implementation shows that a "Zero-Shot" approach to database interaction, in which the system processes database schema in real-time, can be a viable option compared to fine-tuning models, provided it has a solid architectural foundation.

The most important takeaway from this research project has been the impact of hardware acceleration in making AI-based data interaction tools viable. By moving away from standard cloud-based processing to the integration of specialized hardware units, we have seen a drastic reduction in latency, allowing for almost instantaneous execution of queries. This, combined with the transparency of the reasoning process and the safety of the multi-user interface, shows that this system is not only fast but also reliable and trustworthy.

Ultimately, BaseMind acts as a template for the future of data democratization. Although the current product performs well with structured academic and booking data, the ability for the system to expand into more complex and heterogeneous data domains in the future is a clear benefit. The ability for the system to bridge the gap between the human mind and machine-readable code makes it a step towards a more efficient future for data decision-making.

IX. References

[1] S. Ibañez, "The Groq LPU™ Inference Engine: Language Processing Unit Architecture," Groq White Paper, Mountain View, CA, 2024.
[2] M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed. Sebastopol, CA: O'Reilly Media, 2018.
[3] Meta AI, "Llama 3.3: High-performance open-source large language model," Meta AI Research, Dec. 2024.
[4] M. Bayer, "SQLAlchemy: The Database Toolkit for Python," [Online]. [Accessed: Mar. 24, 2026].
[5] T. Sanders, "Natural Language Interfaces to Databases (NLIDB) using LLMs," Proc. IEEE Int. Conf. on Data Eng., 2024.
[6] P. Nayak, "Building an Intelligent Deep Research Agent: From Query to Insights with LangChain and Groq," The AI Forum, Aug. 2025.
[7] H. Zhang and X. Huang, "SGU-SQL: Structure-Guided Large Language Models for Text-to-SQL Generation," Proc. International Conference on Machine Learning (ICML), May 2025.