Concept Paper

CypherQuery: Simplifying Database with Conversational Interfaces

Pooja Potnurwar *, Nisarg Gandhewar , Aditya Pandey , Vaidehi Sahu , Aadya Pande

Posted Date: 3 March 2025

doi: 10.20944/preprints202503.0031.v1

Keywords: CypherQuery; natural language processing; large language models; database querying; user accessibility; data retrieval; real-time queries

 doi:10.20944/preprints202503.0031.v1

Concept Paper

CypherQuery: Simplifying Database with Conversational Interfaces

Pooja Vaibhav Potnurwar [1,*], Nisarg Gandhewar [2], Aditya Pandey [1], Vaidehi Sahu [1] and Aadya Pande [1]

[1] Research Scholar Dept of Computer Science &Engineering(AIML), Shri Ramdeobaba College of Engineering and Management, Nagpur, India

[2] Dept of Computer Science &Engineering(AIML), Shri Ramdeobaba College of Engineering and Management, Nagpur, India

* Correspondence: potnurwarpv@rknec.edu

Abstract: Due to the complexities of traditional database querying methods, accessing data can be a challenge for non-technical users. CypherQuery addresses this issue by leveraging Natural Language Processing (NLP) and conversational large language models (LLMs) to facilitate intuitive database interactions. Users can pose natural language questions without understanding SQL or database schemas. The system utilizes Python for backend processing, LangChain for AI model orchestration, and Streamlit for an interactive user interface. CypherQuery ensures compatibility with a wide array of datasets by supporting both SQL databases (like MySQL and SQLite) and graph databases (such as Neo4j). It efficiently translates user inquiries into structured queries, providing accurate results in real-time.By democratizing data access, improving efficiency in data retrieval, and simplifying the user experience, CypherQuery enhances engagement with data across various sectors, including e-commerce, healthcare, and customer support. This innovative approach fosters data-driven decision-making and operational excellence, transforming how users interact with information..

Keywords: CypherQuery; natural language processing; large language models; database querying; user accessibility; data retrieval; real-time queries

1. Introduction

The proliferation of data in today's digital landscape has led to an exponential increase in the use of databases across various sectors. However, traditional methods of querying these databases often pose significant challenges, particularly for non-technical users. Mastery of structured query languages, such as SQL and graph query languages like Cypher, is typically required to access and manipulate data effectively. This necessity creates a substantial barrier for professionals who lack the technical expertise to navigate complex querying processes.

To address this issue, the CypherQuery project introduces an innovative solution that employs Natural Language Processing (NLP) and conversational Large Language Models (LLMs) to simplify database interactions. CypherQuery enables users to engage with both SQL databases, including SQLite, and graph databases like Neo4j, using natural language. This capability transforms the querying experience into a more intuitive and accessible process, making it easier for users to extract valuable insights without requiring deep technical knowledge. The inclusion of Neo4j is a notable advantage over existing research, which often focuses solely on relational databases.

By leveraging cutting-edge technologies, including Python for backend processing and LangChain for managing LLMs, CypherQuery effectively translates user inquiries into structured database commands. This seamless integration enhances the efficiency of data retrieval, providing accurate results in real time. Additionally, the project utilizes Streamlit to create an interactive web

 doi:10.20944/preprints202503.0031.v1

application interface, enabling users to engage with the system without requiring extensive knowledge of web development.

2. Objectives

The objective of the CypherQuery project is to address the technical challenges associated with traditional database querying by developing an innovative solution that leverages advancements in Natural Language Processing (NLP) and Large Language Models (LLMs). This project aims to simplify the querying process by creating a natural language interface that enables users to interact with databases in an intuitive manner, thus eliminating the necessity for expertise in query languages such as SQL or Cypher. Moreover, it seeks to enhance database accessibility by democratizing interaction, thereby enabling non-technical users to efficiently retrieve, analyze, and interpret data.

Furthermore, the project is designed to support multiple database systems, including relational databases such as MySQL and SQLite, as well as graph databases like Neo4j, ensuring flexibility and scalability. By providing real-time query generation and execution, CypherQuery aims to improve operational efficiency, thereby reducing response times and enhancing decision-making processes in professional environments. Ultimately, the project aspires to promote scalability and versatility through a robust architecture that can adapt to new databases, datasets, and future technological advancements.

3. Literature Review

The research paper "Simplifying Database Interaction through Natural Language Interfaces" by J. A. Brown et al. (2019) investigates the utilization of natural language for database queries. This methodology enhances accessibility for non-expert users; however, the research identifies challenges related to query ambiguity and scaling limitations. In a related study, "Improving Data Accessibility through Streamlined User Interfaces" by T. F. White et al. (2023) examines the enhancement of database interactions through contemporary web interfaces. While this approach improves usability, it encounters scaling constraints when managing larger datasets.

H. S. Tan et al. (2023) introduce an artificial intelligence-powered methodology for query optimization across SQL and graph databases in their work "Automating Query Optimization in Multi-Language Databases." The study emphasizes efficiency improvements but also acknowledges challenges in maintaining query accuracy across diverse schema structures.

Furthermore, "Graph Databases and Querying with Cypher" by A. Francis et al. (2018) investigates the advantages of utilizing Neo4j and Cypher for managing graph data. Although Cypher's intuitive syntax facilitates the learning process, the research identifies issues in managing large datasets and complex graph relationships. This literature review demonstrates significant advancements in simplifying database interactions through natural language interfaces, query optimization frameworks, and hybrid database systems.

However, there remains a substantial gap in integrating multiple database types within a single framework. Notably, existing research does not address the combination of SQLite, MySQL, and Neo4j in a multidatabase integrated solution like our proposed framework, CypherQuery. Our work distinguishes itself by offering a dynamic integration of graph databases alongside traditional relational databases, a feature not commonly found in current literature. While some studies explore aspects of this integration, none effectively combine all three database types, thus highlighting the unique advantage of our approach. This integrated solution aims to address scalability issues, computational overhead, and the complexities of processing advanced queries, leading to more efficient data interactions.

4. Methodology

The CypherQuery system facilitates the integration of natural language and structured database interactions through a modular architecture. It incorporates three primary components: Large

 doi:10.20944/preprints202503.0031.v1

Language Models (LLMs) for comprehending and extracting intent from user queries, query generation for mapping inputs to structured database queries, and database execution for implementing these queries and retrieving results. This architectural design ensures efficient and precise query execution.

a. Technology stack and Databases

Using cutting-edge technology and APIs for effective operation, the system uses a modular design to guarantee flexibility, scalability, and maintainability:

I. Frontend Module: Constructed utilizing Streamlit, providing a user-oriented interface for conversational interactions. Facilitates text input, file uploads (e.g., PDFs), and intuitive user engagement with the system.

II. Backend Module: Powered by OpenAI LLMs, OpenAI Embeddings, and Groq APIs for high-performance natural language interpretation, query generation, and execution. Groq APIs enhance computational efficiency, ensuring expedited processing of queries and embedding-based operations.

III. Database Layer: Supports multi-database integration for diverse data management. Operates with MySQL and SQLite for relational data and Neo4j for graph-based data, accommodating a wide range of database requirements.

Technology Stack

- **Frontend**: Streamlit for interactive web interfaces.
- **Backend**: OpenAI APIs (LLMs and Embeddings) and **Groq APIs** for optimized query processing and execution.
- **Databases**: MySQL, SQLite (Relational Databases), and Neo4j (Graph Database).

b. Modular Architecture

- Following is an outline of the Application Architecture:

Figure 1. Application Architecture.

Step I: Backend Processing with OpenAI LLMs and Embeddings The OpenAI Large Language Models (LLMs) and OpenAI Embeddings function as the backend infrastructure. These models process natural language queries to extract intents, entities, and relationships, subsequently converting the input into structured database queries.

Step II: Frontend Interface with Streamlit .Streamlit is utilized to construct the interactive web interface. It facilitates user input of queries, message transmission thereby providing an efficient and engaging user experience.

Step III: Query Translator The Query Translator processes the user input from the LLM/NLP layer and converts it into a specific query language (e.g., SQL or Cypher). This translation ensures query compatibility with the target database's structure and query language.

Step IV: Database Layer The Database Layer executes the translated query against the appropriate database (relational or graph database). It retrieves results based on the query's structure and data stored in the system.

Step V: User Output The results from the database layer are transmitted to the user via the Streamlit frontend. Users observe the processed output in a clear and structured format on their interface.

c. Query Execution Workflow

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2025 doi:10.20944/preprints202503.0031.v1

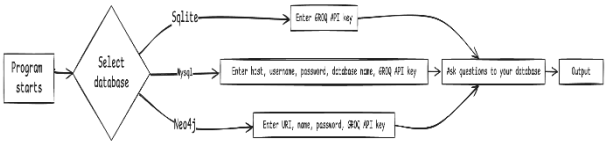

Figure 2. Working of CyperQuery.

1. To utilize the model, the user must adhere to the following procedure:

- Select the desired database;
- Provide the requisite specifications (Groq API key/Host/Username/Password, etc.) corresponding to the database type (SQLite/SQL/Neo4j);
- Submit a query to the model in natural language. The model assistant generates output accompanied by its reasoning process.

2. Processing Workflow

- Input: Users submit queries in natural language through the conversational interface.
- Processing: The backend system converts the input into a structured query format utilizing the NLP layer.
- Execution: The generated query is executed on the respective database, and results are retrieved.
- Output: The processed results are presented to the user in a comprehensible and actionable format.

5. Result and Discussion

A. Illustration with SQLite database

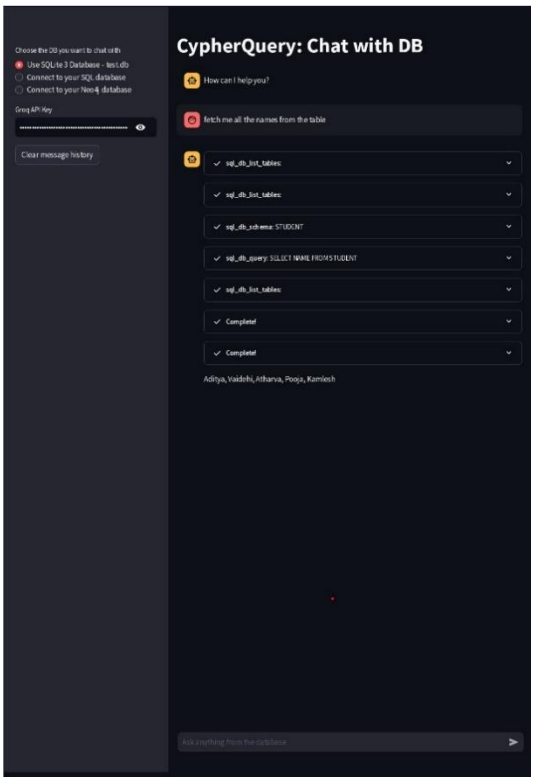

Figure 3. Interaction with SQLite Database.

 doi:10.20944/preprints202503.0031.v1

In the image above, the CypherQuery interface serves as an advanced platform for working with a SQLite server database. The SQLite database was successfully chosen by the user, who then ran a query to "fetch all the names from the table." The system's ability to handle requests effectively and support natural language communication is demonstrated by this exchange. CypherQuery improves accessibility for both technical and non-technical users by simplifying the querying process, making data retrieval easier and more effective.

B. Illustration with MY SQL database

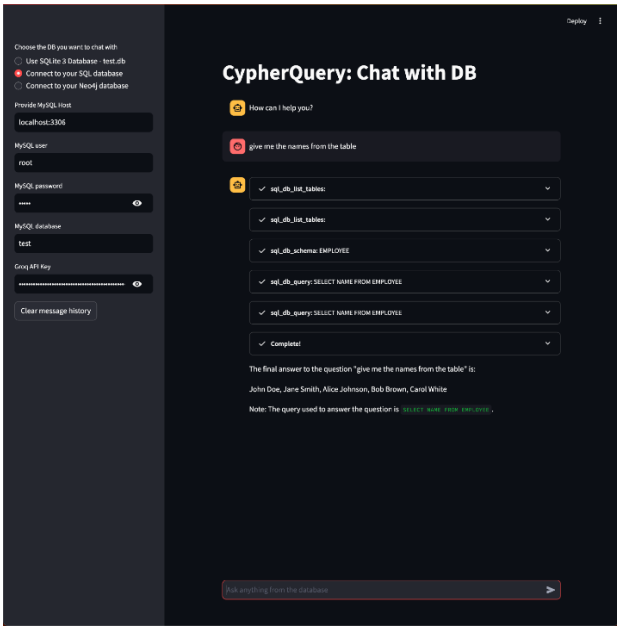

Figure 4. Interacting with MYSQL Database.

The interface effectively facilitates interaction with a MySQL database, as depicted in the image. Initially, the system prompts the user with "How can I help you?" The user subsequently inputs the request to "give me the names from the table." The system processes this natural language input, converts it into a structured query, and retrieves the relevant results. This seamless conversion and response demonstrate how the platform simplifies database querying, rendering it accessible for users without technical expertise while providing efficient data retrieval capabilities.

C. Illustration with Neo4j (graph database)

 doi:10.20944/preprints202503.0031.v1

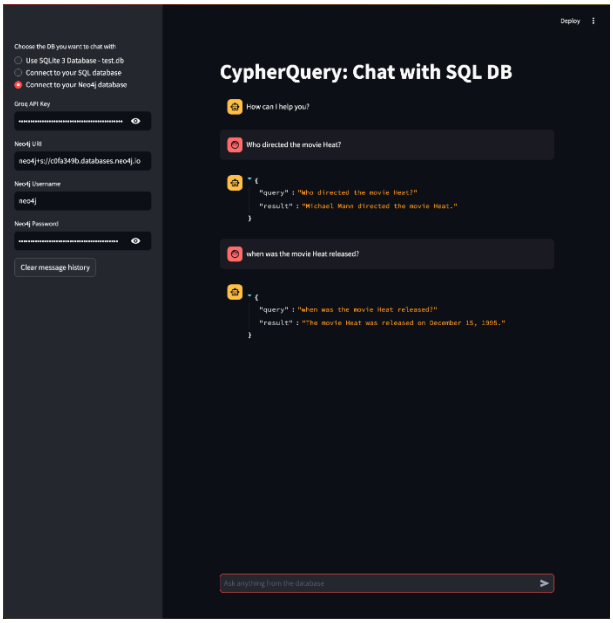

Figure 5. Interacting with Neo4j(graph data) Database.

The visual illustrates how the interface offers a user-friendly way to interact with a graph database. "How can I help you?" is the first question it asks the user. "Who directed the movie Heat?" the user then asks. After successfully retrieving the information from this natural language input, the algorithm replies, "Michael Mann directed the movie Heat." This smooth query processing and precise output demonstrates how the platform simplifies the querying process, making it accessible to non-technical users while guaranteeing effective data access

6. Conclusion

The system presents an innovative approach to simplifying database querying through natural language processing (NLP). By facilitating user interaction with databases such as MySQL and SQLite via intuitive, human-readable queries, it eliminates the necessity for specialized knowledge of SQL. This project successfully integrates advanced machine learning models and NLP techniques to convert natural language input into executable queries, providing support for multiple database systems, including both SQL-based databases. The architecture of the system comprises a user-friendly interface, a robust query generation model, and seamless database connectivity, ensuring that non-technical users can efficiently access and retrieve data without complex syntax. Comprehensive evaluations demonstrate high accuracy, rapid response times, and a satisfactory user experience, confirming its potential as a practical tool for various industries. In conclusion, this system effectively bridges the gap between non-technical users and database management, streamlining workflows and enhancing decision-making capabilities through conversational interfaces.

Acknowledgment: The authors wish to express their gratitude to the management for providing the necessary resources and facilities to conduct this research on CypherQuery within the institution's laboratory. We also extend our appreciation to the Science & Technology department for approving our research proposal and for the financial support that facilitated our research activities. This support has been instrumental in the successful completion of this work.

References

1. Brown, J. A., Smith, L. T., & Clarke, R. M. (2019). Simplifying database interaction through natural language interfaces. *Journal of Database Systems, 35*(4), 123-135

 doi:10.20944/preprints202503.0031.v1

2. Smith, R. L., Patel, K. R., & Zhao, H. (2020). Multi-database query generation and optimization. *Proceedings of the International Conference on Database Technology, 18*, 200-210.
3. Gupta, P. K., Desai, M. J., & Thakur, A. R. (2021). Adapting conversational AI for database querying. *Journal of AI and Data Science, 29*(1), 45-60.
4. Choi, L. Y., Wang, S., & Lee, H. J. (2022). A unified approach to querying relational and graph databases. *Journal of Database Innovation, 42*(3), 75-89.
5. White, T. F., Sanders, P. Q., & Moore, J. L. (2023). Improving data accessibility through streamlined user interfaces. *Journal of Information Systems, 48*(2), 145-156
6. Tan, H. S., Gupta, S. V., & Lin, T. (2023). Automating query optimization in multi-language databases. *International Journal of Data Management, 21*(5), 310-325.
7. Francis, A., & Walker, D. J. (2018). Graph databases and querying with Cypher. *Database Systems Review, 31*(7), 98-110
8. Meharwade, A., & Patil, G. A. (2016). Efficient keyword search over encrypted cloud data. *Procedia Computer Science, 78*, 139-145.
9. Nashipudimath, M., Shinde, S., & Jain, J. (2020). An efficient integration and indexing method based on feature patterns and semantic analysis for big data. *Array, 7*, 100033.
10. Zhu, M., & Cole, J. (2022). PDFDataExtractor: A tool for reading scientific text and interpreting metadata from the typeset literature in the portable document format. *Journal of Chemical Information and Modeling, 62*
11. https://streamlit.io/
12. https://python.langchain.com/