Intelligent Database Query Assistant: A Conversational AI for SQL Queries

B.V. Dinesh Krishna, P. Yogendrasai, S. Rajeev, Ms Pooja Gowda
Department of Computer Science and Engineering,
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India
bl.en.u4cse22210@bl.students.amrita.edu, bl.en.u4cse22246@bl.students.amrita.edu,
bl.en.u4cse22251@bl.students.amrita.edu, g_pooja@blr.amrita.edu

***Abstract*—A knowledgeable conversational system has been developed to offer natural language access for MySQL databases. The system directs its functions towards users without technical expertise while using LangChain framework along with OpenAI models to convert user queries into SQL commands. The solution grants stakeholders including business analysts along with healthcare professionals and public sector users to access accurate database information through its intuitive interface without requiring technical skills. The development followed an agile method which permitted continuous feedback and guarantee of translation precision and also maintained secure execution of queries. The auxiliary component comes with a user-friendly interface that includes error management technologies enabling flexible user interaction alongside enterprise scalability.**

***Index Terms*—Conversational AI, SQL Query Generation, Large Language Models (LLMs), Context-Aware Querying, Relational Databases, Human-Computer Interaction, Database Accessibility, AI-driven Interfaces.**

I. Introduction

Organizations within every industry in present times rely on relational databases to handle the large structured information collections they manage. Most users find it difficult to access database information because they need Structured Query Language (SQL) expertise which creates an entry hurdle for non-technical individuals. Business analysts along with human resources specialists and marketers and healthcare professionals encounter performance limitations when they depend on IT departments to obtain valuable information and reports. The dependency delays time-sensitive choices for decision-making while reducing operational effectiveness. Systems require improved interfaces which enable users to access databases through their own spoken words instead of complex query languages.

The research community has tackled the challenge by developing multiple forms of natural language interfaces to databases (NLIDBs) as well as conversational agents to resolve this matter. The research team at Murali et al. created ChEMBL Bot to serve as a chatbot that permits user-friendly queries of chemical databases [1]. The Thai People Map established an information system with a conversational interface that Simud et al. created to serve large-scale population databases [2]. The educational database learning system ChatbotSQL accepts Perez-Mercado et al proposes to teach SQL using conversational discourse for educational purposes [3]. Parthasarathi et al. conducted a thorough survey to explain the existing situation of Text-to-SQL conversational systems while identifying their limitations with multi-turn dialogues and domain-specific challenges [4]. The technical viability of natural language database querying appears in works by Galitsky [6] and Owda et al. [5][10] as well as Koutrika et al. [7] because these studies prove its practical benefits in academic and industrial applications.

Currently deployed systems show various significant drawbacks according to current standards. Most systems fail to execute sophisticated multi-table queries as well as transform unclear natural language inputs into valid SQL statements. Several translation applications do not give users adequate clarification about the translated SQL code which results in reduced understanding and decreased user confidence. The system does not scale well and has specific domain restrictions while displaying user interface problems which prevent enterprise-wide adoption. An intelligent database query assistant called "Intelligent Database Query Assistant" serves as our main contribution through its development as a conversational AI tool which turns natural language to SQL syntax through LangChain and OpenAI framework operations and provides readable informative output. Users who lack technical skills can work with this system since it employs strong error detection functionality coupled with interactive feedback that helps users perfect their queries while enhancing their overall experience.

Key Features of the Proposed System:

- Through this system non-expert users can interact with relational databases by entering natural language requests.
- Leverages LangChain + OpenAI for real-time SQL translation and result interpretation.
- The system places emphasis on security measures alongside scalability features and error handling protocols for deployment at an enterprise scale.
- The system incorporates a feedback-based conversational system that permits ongoing learning as well as feedback-based refinement.

Identify applicable funding agency here. If none, delete this.

II. Literature Survey

Recent literature has thoroughly discussed the development history of conversational interfaces in communication with a database. These systems are designed to help close the gap between language users and structured query languages with its capability to connect data in the fields that are non-technical.

Murali et al [1] proposed ChEMBL Bot, a domain-specific conversational coal take that allows query interface based on ChEMBL, one of the early concepts of chatbot based query interface. Simud et al. [2] developed Thai-language agent for querying geographic analytics platforms hence extending such an approach to support multilanguage capabilities. Building on the theoretical foundation further, Parthasarathi et al. [3] provided a detailed analysis of conversational Text-to-SQL systems pinning down fundamental issues associated with schema alignment, multi-turn dialogue management, and the generation of complex nested SQL queries.

Alonso et al. [4] suggested a knowledge plug-in framework for integrating external structured data, in the form of O*NET, to conversational systems thereby elating its use to employability and career analysis tools. Owda et al. [5] did foundational work establishing the ground for natural language interfaces (NLIs) to relational databases, whose current iterations can be seen as being influenced by rule-based paradigms. Galitsky [6] designed a theoretical model concentrating on linguistic structures for intelligent chat bot response, whereas Koutrika et al[7] investigated query explanation mechanisms for better user comprehension of query results in a structured form.

A great contribution to the field of education was provided by Pérez-Mercado et al. [8] with ChatbotSQL, which helped the students to understand the syntax of SQL using this dialoguing mode. Similarly, Shabaz et al. [9] presented Aneesah, enhanced contextual query understanding based on machine learned semantic parsing. Carrying on their effort, Owda et al. [10]improved upon the conversational system C-BIRD that brings information extraction with automatic SQL generation together.

The latest developments have made use of AI at its cutting edge. Kumar et al. [11] further improved database interactions by streamlining integration of OpenAI GPT models and Stanford Core NLP thereby improving both accuracy and contextual understanding. Reshmi and Balakrishnan [12] have developed an inquisitive chatbot that takes part in clarification dialogues in an attempt to increase response accuracy. On deployment after deployment, Colas et al. [13] managed the scalability and deployment challenges, providing practical solutions for the real-time performance.

Jamil [14] developed ConSQL, Context aware conversational database query language while Quamar et al [15] integrated ontology based reasoning in their dialogue mechanisms. Yu et al. [16] developed CoSQL, a benchmark dataset to compare learning to cross-domain natural language queries, to which generalizability can be compared. Gassen et al. [17] introduced CAT – a toolchain, which synthesizes data-aware agents to perform transactional operations.

While beyond further technical developments, Jiang et al. [18] have presented an example of conversational AI being applied to the enterprise business convergence, Lewandowski et al. [19] have provided a systematic review on the adoption of AI conversational agents to organizations. Khan et al. [20] studied the impact that generative AI models have on library systems and Cai et al [21] suggested natural conversational interfaces that are designed for geospatial databases, thereby broadening the scope of use of such systems to include spatial analytics.

Over all, the literature highlights a consistent development from rule based interfaces to sophisticated neural conversational agents. Nevertheless, restrictions remain in relation to schema generalization, multi-domain adaption, and instant user feedback incorporation. Our project solves these problems by creating a schema-aware, feedback-driven AI assistant built on top of LangChain and OpenAI LLMs for generating correct SQL queries and natural language replies at various domains.

III. Methodology

An intelligent natural language interface for relational databases using Large Language Models (LLMs) receives its methodology description within this section. The system architecture consists of four distinct phases as shown in Fig. 1, which are input processing, SQL query creation, database execution, and result generation. The system operates in independent components which work together for delivering contextualized natural language output regarding complex database queries throughout the complete intelligent processing pipeline.

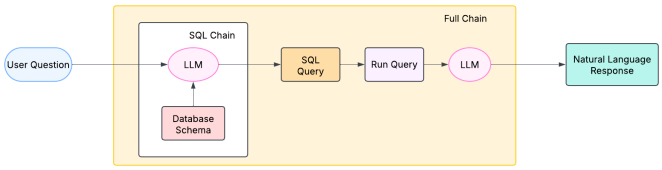

Fig. 1. Architecture Diagram

A. User Query Ingestion

The system starts its process by accepting questions uttered by users in natural spoken language. The system receives questions from users in natural colloquial or domain-specific language when users ask for example “What were the average sales for last month?” This marks the beginning of data processing operations. Users enjoy a natural interaction experience in this system instead of the traditional SQL requirement which diminishes the learning process and enhances user accessibility.

The user-entered data moves to the SQL Chain module that performs the transformation process from unstructured language to structured database queries. The semantic interpreter

functions as the semantic interpreter between human language and machine-executable commands.

1) SQL Chain: Semantic Parsing and Query Formulation: A unique fine-tuned Large Language Model (LLM) serves as the backbone of SQL Chain which uses both database schema information and user input to understand their requirements. The essential elements that build the sub-system include various components.

2) Schema-Aware Contextualization: The LLM requires database metadata from the connected relational database before starting the SQL generation task. The database system supplies the sub-system with complete metadata about table structures including column types together with foreign key constraints and entity relationships. The schema-aware prompting mechanism enables the model to create database queries that meet both syntactic requirements and structural rules of the relational database system.

3) Intent Extraction and Mapping: The LLM analyzes text queries to detect user goals before translating them into an SQL statement. The LLM detects query requirements for aggregation operations (e.g., AVG, COUNT) alongside filtering with (e.g., WHERE clauses), sorting and join execution. The model connects intent expressions to SQL operations by making use of prompt-engineering methods to produce the appropriate SQL command.

4) SQL Generation: The LLM uses extracted intent and schema data to develop a SQL query. The processed request maintains an optimized condition that balances syntactical precision with semantic accuracy. The user request "List all employees who joined in the last 6 months" would get transformed into a parameterized SQL query that employs date functions and table filtering techniques.

B. Query Execution Engine

After the generation process succeeds the SQL query proceeds to the database execution engine. This phase consists of:

- The Connection Handler creates secure connections to target databases (MySQL) through the use of SQLAlchemy or comparable ORMs.
- The Execution Layer implements both SQL execution and fetches results from the database systems.
- The Error Management Module operates as a component that tracks and records both syntax along with runtime errors. An error message produced by the LLM involving faulty queries or missing fields undergoes analysis for use as feedback to adjust either the query or request for clarification within the processing chain.

C. Result Interpretation and Natural Language Generation

The end-user does not access the raw output data which originates from the database system. The LLM receives the database output before generating descriptive prose from it. An example response generated by the LLM for tabulated sales numbers might be

- The total sales recorded in the second quarter of 2023 were $1.2 million, which shows a 15% increase compared to the previous quarter.

D. Full Chain Integration and Feedback Loop

The user receives the natural language response as the last element of the system. The pathway that includes intake processing and understanding runs as an automated distinct system. User clarification becomes necessary when the system detects ambiguous queries or responses. It activates a feedback feature to request clarification from the user. User experience improves progressively due to dialogue-based feedback operations that minimize confusion and decrease mental strain for users.

E. SQL Query Transparency and User Access

For the sake of transparency and to facilitate learning, the system has an optional component that allows the SQL query generated to be presented next to the natural language response. This is of special use to the users who wish to:

- Understand how their question was translated in SQL.
- Reuse the query in other application or tool to query databases.
- Learn the SQL syntax from reverse mapping of their natural language input.

When the query is executed successfully, users are allowed to view the associate SQL statement or copy it. For example, to an inquiry question such as "Show me the highest-grossing products in 2023", the assistant will return:

The highest-grossing products in 2023 were Product A, Product B, and Product C.

Upon request, the system displays:

SELECT PRODUCT_NAME FROM SALES WHERE YEAR = 2023 ORDER BY REVENUE;

This feature corresponds to explainable AI principles, and this advantage nurtures people's trust especially in business analytics, education and semi-technical settings. Besides that it also supports debugging, promotes SQL for non-technical users and is also a good learning tool.

F. User Stories

To make sure that the system meets real-world needs, a set of user stories was defined during the design phase. These stories reflect typical scenarios encountered by target users across various domains, including business analytics, healthcare, and public sector data retrieval. Each story is written from the user's perspective to guide development and testing priorities.

- **User Story 1:** Being a business analyst, I seek to pose sales-related questions, simply in English, so that I can interact with the data without the SQL knowledge.
- **User Story 2:** As an HR manager, I want to get back employee records in natural language, so that I can gather a quick report during meetings.

- **User Story 3:** As a health worker, I need to search test records for patients using common phrases so that I am in a position to make better clinical decisions more speedily.
- **User Story 4:** As a government official, I wish to query about the data on city services in spoken terms so as to examine the performance of the public sector without even depending on IT services.

G. Functional and Non-Functional Requirements

1) Functional Requirements: The below key functions describe the behavior of the Intelligent Database Query Assistant:

- **FR1:** The system shall then take queries from the users in natural language through a web- based interface.
- **FR2:** The system will convert natural language queries into valid SQL commands with the use of LangChain and OpenAI.
- **FR3:** The system shall process SQL queries with a connected MySQL database.
- **FR4:** The system will present results in natural language and formatted in an easy to use manner.
- **FR5:** The system will offer an option to show the generated SQL to the user in terms of clarity and learning.
- **FR6:** The system will store the query history on users for the session tracking and future reuse.
- **FR7:** The system should find and process ambiguous or invalid queries by requesting the user for clarification.

2) Non-Functional Requirements: Besides the core functionality, the system should satisfy a number of the performance, usability, and security criteria.

- **NFR1 – Usability:** The interface should be easy-to-use and intuitive in order to cater for those that are not technical in nature.
- **NFR2 – Reliability:** The system will ensure that it has stable connections to the database and should be able to handle errors gracefully.
- **NFR3 – Scalability:** The architecture should be capable of handling several concurrent users without the degradation of performance.
- **NFR4 – Security:** The system will have to employ secure connections and will not allow a SQL injection/ unauthorized access.
- **NFR5 – Responsiveness:** The system shall reply to user inquiries in an average time of 3 seconds with normal load.
- **NFR6 – Portability:** The solution will be web-based and will be capable of deployment in different platforms (Windows, Linux).

These requirements make the system robust and user-centric while being prepared for use in real, enterprise-level settings.

IV. Results and Discussion

A MySQL database-based, interactive online system was used to evaluate the Intelligent Database Query Assistant. To evaluate how well the system could handle natural inquiries, create applicable SQL commands, maintain secure database connections and submit its findings, it was tested in many test scenarios. This section presents a detailed description of the system's performance supported by annotated figures demonstrating its main features and results.

A. Application Interface and Database Connectivity

Upon launching the application, users experience a simple design that is designed to increase accessibility to non technical users. Fig. 2 shows the actual application's interface gathering database credentials such as host, user, password, and database name, sets up AI model configuration in an API key and provides the chat panel for users to interact with the assistant.

When valid credentials are entered and connection to the MySQL database is made, officers view the changed connection status displayed in real time in the system. It is assured that the users will know that their backend connection is alive before running any queries. Along with the interface, the left panel offers tools to build new chats and obtain access to previous query histories to achieve extended operation and maintain auditable logs.

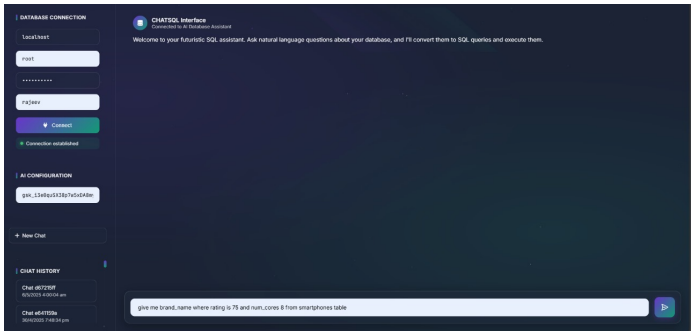

Fig. 2. Main interface for database connection and AI assistant access.

B. Natural Language Query Input

The assistant enables users to ask questions in Natural language- English. In Fig. 3, the user inputs:

GIVE ME BRAND_NAME WHERE RATING IS 75 AND NUM_CORES 8 FROM SMARTPHONES TABLE

The system captures the request, extracts relevant attributes (brand_name, rating, num_cores), identifies which table needs to be used (smartphones), interprets conditional aspects of the request. Real-time syntax feedback is provided in the input field where users can send their query by pressing-on an easy action button.

Fig. 3. Text input box for entering natural language queries.

C. AI Response and Data Presentation

After the assistant gets the query and writes the necessary SQL, it calculates the results and shows it to the user in an easy to read format. Fig. 4 illustrates the assistant's response: list of smartphone brands fulfilling the requirements rating = 75 and num_cores = 8. In the results, there is prominent brands such as Samsung, Realme, Motorola, Vivo, and iQOO.

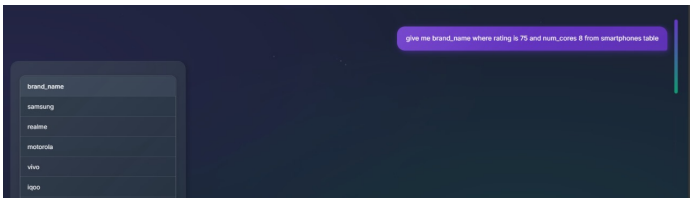

Fig. 4. Results show matching smartphone brands based on user input.

This approach will bring forth answers to non-technical users at once thus enabling rapid insight building without making them vulnerable to complex SQL demands or manual database use. Conversational interface of its approach finds particular applications among business analysis, sales reporting and market research experts.

D. SQL Transparency and Learning Support

To allow users who are reviewing or utilising the original SQL queries, the system provides a supplementary optional feature of SQL visibility. Naturally upon clicking the "Show SQL Query" which is displayed in Fig. 5, the following becomes visible to the assistant's real SQL Query which is:

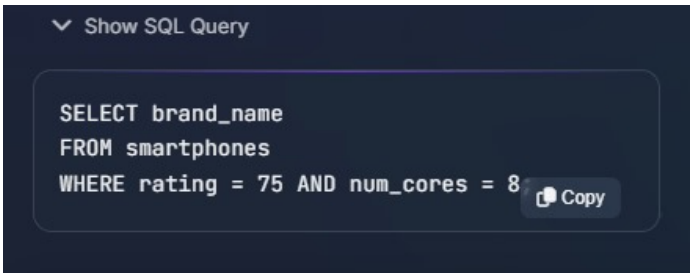

Fig. 5. SQL query shown for clarity and learning.

Apart from the technical validation, this feature also contributes to the learning process. This tool allows novice users to experience learning SQL gradually because they see how everyday wordings behave in formal Database language. Moreover, a one click copy feature enables reuse of produced queries.

E. Backend SQL Validation

AI-generated queries were tested for accuracy by having them run through backend checks in MySQL Workbench. Fig. 6 shows the assistant generated query which is executed. The results obtained by implementing SQL queries straight to the database were the same as the assistant generated, thereby showing that the transition from the natural language to SQL was accurate and complete.

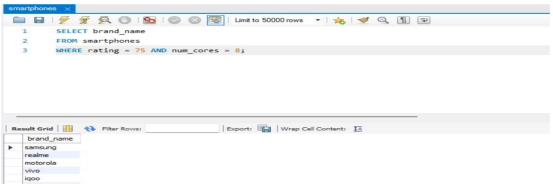

Fig. 6. Query verification in MySQL Workbench.

This validation is showing the systems reliability when accuracy is of paramount importance such as healthcare, finance, and enterprise reporting.

F. Session Management and Chat History

A major aspect of user-friendliness is the way session management is performed. The diagram in Fig. 7 shows that all user sessions and their respective queries all have their timestamps recorded. Users can retrieve previous queries, update them as suitable or save them as templates for future use. This maintenance of the user interactions enhances the app's functionality because it enables real-time data analysis and progressive query refinement and even expanded analytical operations.

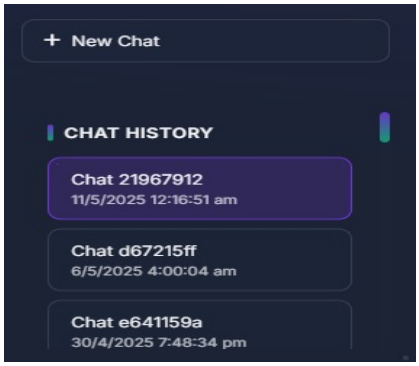

Fig. 7. Chat history panel for quick access and reuse of past queries.

G. Analysis and Observations

The evaluation yielded the following key observations:

- **Accuracy:** The assistant understood and processed complex queries with various conditional aspects.
- **Usability:** Outcomes driven by results were available for the users without having to engage the SQL.
- **Transparency:** The automatically visible query reveal aid increased user confidence and enabled the absorption of knowledge.
- **Validation:** The backend processes confirmed that AI generated SQL aligned with manual queries in content and output.
- **Productivity:** The use of chat history and real-time feedback helped in processing similar and better queries.

V. Conclusion and Future Work

As a result, this research gave rise to the Intelligent Database Query Assistant, which allows users to speak with relational databases in natural language regardless of SQL literacy. The use of integrations with LangChain and OpenAI's language models allows the platform to convert user guidelines into implementable SQL commands and deliver results in a simple, conversational interface. Adding SQL transparency, the session history and backend validation will ensure that the system is easy to use and accurate. In the future the assistant might be developed to support multi-turn conversations, accept voice commands, integrate with multiple databases, and support data visualization, hence expanding real world use to business, education and publics.

References

[1] V. Murali, R. J. Sarma, P. A. Sukanya and P. Athri, "ChEMBL Bot - A Chat Bot for ChEMBL database," 2018 Fourteenth International Conference on Information Processing (ICINPRO), Bangalore, India, 2018, pp. 1-6, doi: 10.1109/ICINPRO43533.2018.9096710.

[2] T. Simud, S. Ruengittinun, N. Surasvadi, N. Sanglerdsinlapachai and A. Plangprasopchok, "A Conversational Agent for Database Query: A Use Case for Thai People Map and Analytics Platform," 2020 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), Bangkok, Thailand, 2020, pp. 1-6, doi: 10.1109/iSAI-NLP51646.2020.9376833.

[3] S. Hari Krishnan Parthasarathi, L. Zeng and D. Hakkani-Tür, "Conversational Text-to-SQL: An Odyssey into State-of-the-Art and Challenges Ahead," ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 2023, pp. 1-5, doi: 10.1109/ICASSP49357.2023.10096170.

[4] R. Alonso, D. Dessí, A. Meloni and D. Reforgiato Recupero, "A Novel Knowledge Plug-In for Incorporating Information About Employability From the O*NET Database Into Conversational Agents," in IEEE Access, vol. 12, pp. 92663-92681, 2024, doi: 10.1109/ACCESS.2024.3423359.

[5] M. Owda, Z. Bandar and K. Crockett, "Conversation-Based Natural Language Interface to Relational Databases," 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, Silicon Valley, CA, USA, 2007, pp. 363-367, doi: 10.1109/WI-IATW.2007.60.

[6] Galitsky, B., & Galitsky, B. (2019). Developing Conversational Natural Language Interface to a Database. Developing Enterprise Chatbots: Learning Linguistic Structures, 85-120.

[7] Koutrika, G., Simitsis, A., & Ioannidis, Y. (2009). Conversational databases: explaining structured queries to users. Stanford InfoLab.

[8] Pérez-Mercado, R., Balderas, A., Muñoz, A., Cabrera, J. F., Palomo-Duarte, M., & Dodero, J. M. (2023). ChatbotSQL: Conversational agent to support relational database query language learning. SoftwareX, 22, 101346.

[9] Shabaz, K., O'Shea, J. D., Crockett, K. A., & Latham, A. (2015, July). Aneesah: A conversational natural language interface to databases. In Proceedings of the World Congress on Engineering (Vol. 1).

[10] Owda, M., Bandar, Z., & Crockett, K. (2011). Information extraction for SQL query generation in the conversation-based interfaces to relational databases (C-BIRD). In Agent and Multi-Agent Systems: Technologies and Applications: 5th KES International Conference, KES-AMSTA 2011, Manchester, UK, June 29–July 1, 2011. Proceedings 5 (pp. 44-53). Springer Berlin Heidelberg.

[11] S. Kumar, M. S. Alam, Z. Khursheed, S. Bashar and N. Kalam, "Enhancing Relational Database Interaction through Open AI and Stanford Core NLP-Based on Natural Language Interface," 2024 5th International Conference on Recent Trends in Computer Science and Technology (ICRTCST), Jamshedpur, India, 2024, pp. 589-602, doi: 10.1109/ICRTCST61793.2024.10578418.

[12] Reshmi, S., & Balakrishnan, K. (2016). Implementation of an inquisitive chatbot for database supported knowledge bases. sādhanā, 41, 1173-1178.

[13] Colas, A., Bui, T., Dernoncourt, F., Sinha, M., & Kim, D. S. (2020). Efficient deployment of conversational natural language interfaces over databases. arXiv preprint arXiv:2006.00591.

[14] Jamil, H. (2023). ConSQL: A Database Language for Contex-aware Conversational Cooperative Query Answering. Research Square.

[15] Quamar, A., Lei, C., Miller, D., Ozcan, F., Kreulen, J., Moore, R. J., & Efthymiou, V. (2020, June). An ontology-based conversation system for knowledge bases. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (pp. 361-376).

[16] Yu, T., Zhang, R., Er, H. Y., Li, S., Xue, E., Pang, B., ... & Radev, D. (2019). Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. arXiv preprint arXiv:1909.05378.

[17] Gassen, M., Hättasch, B., Hilprecht, B., Geisler, N., Fraser, A., & Binnig, C. (2022). Demonstrating CAT: synthesizing data-aware conversational agents for transactional databases. arXiv preprint arXiv:2203.14144.

[18] Jiang, J., Ye, S., & Zeng, L. (2024, September). Research on key technology of business convergence based on conversational AI. In Proceedings of the International Conference on Image Processing, Machine Learning and Pattern Recognition (pp. 621-626).

[19] Lewandowski, T., Delling, J., Grotherr, C., & Böhmann, T. (2021). State-of-the-Art Analysis of Adopting AI-based Conversational Agents in Organizations: A Systematic Literature Review. PACIS, 167.

[20] Khan, R., Gupta, N., Sinhababu, A., & Chakravarty, R. (2024). Impact of conversational and generative AI systems on libraries: A use case large language model (LLM). Science & technology libraries, 43(4), 319-333.

[21] Cai, G., Wang, H., MacEachren, A. M., & Fuhrmann, S. (2005). Natural conversational interfaces to geospatial databases. Transactions in GIS, 9(2), 199-221.