DOI: 10.21917/ijsc.2025.0508

PRESERVING DATABASE SCHEMA PRIVACY WHILE GENERATING SQL QUERIES USING GENERATIVE AI

Jitesh Prasad Khatick[1] and Soumitra Kumar Mandal[2]
[1]*Department of Computer Science and Engineering, Maulana Abul Kalam Azad University of Technology, India*
[2]*Department of Electrical Engineering, National Institute of Technical Teachers' Training and Research, India*

Abstract

In Generative AI (GenAI) models have proven effective in translating natural language queries into SQL. However, directly exposing database schema details, including table and column names, to GenAI poses significant security and privacy concerns. This paper presents an approach to preserve database schema integrity by employing aliasing techniques. Aliased representations of schema elements are enriched with semantic metadata, ensuring GenAI can generate accurate SQL queries without direct access to the original schema. The proposed method bridges the gap between privacy and functionality, providing a robust framework for secure and efficient SQL generation.

Keywords:

NLIDB (Natural Language Interface to Database), SQL (Structured Query Language), StanfordCoreNLP, Large Language Model (LLM), Generative AI (GenAI)

1. INTRODUCTION

Natural Language Interfaces to Databases (NLIDBs) are revolutionizing database management by enabling users to interact with data using natural language queries instead of complex structured query languages like SQL. This innovation leverages the capabilities of Large Language Models (LLMs), such as GPT and BERT, to bridge the gap between human language and computational logic. However, alongside these advancements come critical security challenges, particularly concerning the preservation and protection of database table structures and schema information.

One significant limitation of LLM-based NLIDBs lies in their handling of schema exposure. Research by [7] emphasizes that LLMs, by design, infer patterns and may unintentionally reveal database schemas or metadata. This occurs because these models are trained on diverse datasets that often include general examples of database queries, making them susceptible to disclosing structural details even when such information is not explicitly provided by the user. This poses a direct threat to the confidentiality of proprietary database designs and sensitive information embedded within schemas.

Another pressing concern is the risk of SQL injection attacks facilitated by the permissive query generation process of LLMs. [31] argue that LLMs, when inadequately equipped with context-aware input validation mechanisms, can generate queries that execute malicious commands. Adversaries could exploit the system's natural language processing flexibility to craft queries that compromise data integrity, leak sensitive data, or disrupt system operations. These risks are compounded when LLMs operate without stringent input sanitization or robust access control measures.

Moreover, as pointed out by [6], LLMs often lack interpretability in their decision-making processes, leading to a "black box" problem. This lack of transparency makes it difficult for developers to trace errors in query generation or identify vulnerabilities. Consequently, database administrators may struggle to implement corrective measures or understand the reasoning behind potentially harmful queries generated by the system.

Addressing these challenges requires a multi-faceted approach. Robust input validation techniques must be integrated to prevent injection attacks. Systems should implement role-based access controls to restrict database operations according to user permissions. Furthermore, schema obfuscation techniques, which anonymize or hide structural information, can reduce the risk of schema inference by attackers. Another promising avenue is the incorporation of explainable AI methods to enhance the interpretability of LLM-based query generation, allowing developers to understand and rectify vulnerabilities in real-time.

Finally, integrating security measures at the design stage is crucial. For instance, models like StructGPT and SQL-PaLM have been proposed to optimize text-to-SQL systems with security in mind, ensuring that the generated queries adhere to strict operational constraints [11]. By combining these advanced methodologies with continuous monitoring and periodic audits, organizations can better safeguard their LLM-driven NLIDBs.

In conclusion, while LLM-based NLIDBs present unparalleled opportunities for simplifying database access, they also introduce vulnerabilities that demand attention. Focusing on security-centric design and incorporating advanced validation, obfuscation, and interpretability mechanisms are essential for creating resilient and secure systems that can withstand evolving threats.

2. PROPOSED ARCHITECTURE AND METHODOLOGY

First and foremost, the system allows users to input their queries in English natural language into a provided text field. The input query is then sent to StanfordCoreNLP for sentence analysis, which generates information such as Tokens, POS (Part of Speech), Basic Dependency, and NER (Named Entity Recognition) values. This generated information is stored in temporary memory for further processing.

Once the information is collected, stop words, which are not necessary for evaluating a SQL query, are removed. The relevant information, such as noun tokens from the POS tag, basic dependency, and NER value, are separated from the gathered information.

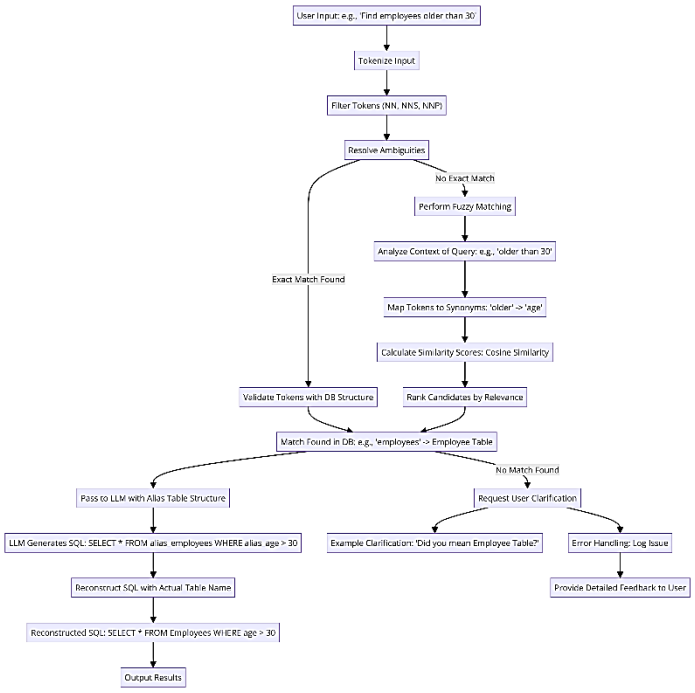

Fig.1. Architecture of Proposed system

After collecting all the required information including column names and table names, the system will decide which table is required to give the desired output. Then the system aliases the required table structure and the table schema is passed to the already existing LLM to generate the desired SQL query. After the SQL query is generated from the LLM, the aliased structure is renamed to its actual name which provides security to our database and finally we get the information from the database through the generated SQL query.

2.1 GETTING RELEVANT TOKENS USING STANFORDCORENLP

We have used Stanford CoreNLP for getting all relevant information from the input natural language query, and the techniques involved are:

2.1.1 Pipeline:

The pipeline in Stanford CoreNLP is a central component that processes text or XML and generates annotation objects. Pipelines are built using Properties objects, which provide specifications for the annotators to be executed and allow customization of their behavior.

2.1.2 Tokenize:

Tokenizer that has been further improved to handle noisy and online text with greater accuracy. This tokenizer not only breaks down the input text into individual tokens, but also captures the character offset of each token within the original text.

2.1.3 Split:

Sentence Segmentation: Breaking a Token Sequence into Meaningful Sentences, sentence segmentation is the process of splitting a token sequence into individual sentences. In natural language processing (NLP), this task is crucial for various applications, such as text summarization, sentiment analysis, machine translation, and information extraction.

2.1.4 POS:

Tokens in a text are annotated with their corresponding Part of Speech (POS) tags through the utilization of a highly accurate maximum entropy POS tagger [11]. This tagging process involves the application of advanced natural language processing techniques to automatically assign POS labels to each individual token in the text. The maximum entropy POS tagger, which is a sophisticated machine learning model trained on vast amounts of annotated data, predicts the most probable POS tag for each token based on its contextual features, such as its surrounding words, grammatical structure, and semantic meaning.

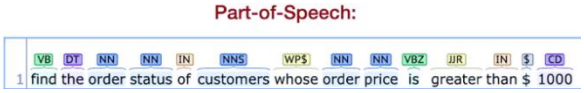

Fig.2. Part-of-Speech identification using Stanford CoreNLP

2.1.5 NER:

The entity recognition system implemented in this application is designed to accurately identify and extract named entities such as PERSON, LOCATION, ORGANIZATION, and MISC, as well as numerical entities like MONEY, NUMBER, DATE, TIME, DURATION, and SET. This recognition is achieved using a combination of CRF (Conditional Random Fields) sequence taggers that have been trained on diverse corpora, and default annotators. Additionally, two rule-based systems are employed for the identification of numerical entities, specifically for money and numbers, and for processing temporal expressions, respectively [12]-[13].

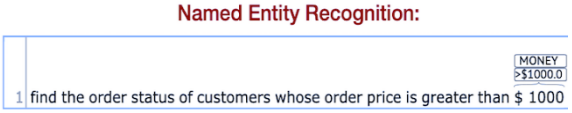

Fig.3. NER using StanfordCoreNLP

2.1.6 Lemmas:

The process of analyzing words and breaking them down into their basic forms, known as morphemes, is a fundamental aspect of morphology. This field of study delves into the structure of words and how they are constructed from smaller units with meaning. One common technique used in this analysis is lemmatization, which involves identifying the root or base form of a word.

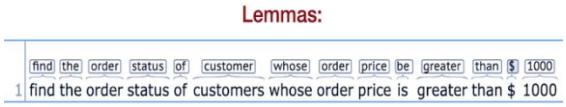

Fig.4. Finding Lemmas using StanfordCoreNLP

2.2 NEED FOR SECURING SCHEMA

Providing Large Language Models (LLMs) with direct access to database schemas for text-to-SQL generation enhances their ability to produce accurate queries. However, this practice introduces several security considerations that must be addressed to maintain database integrity and confidentiality.

2.2.1 Exposure of Sensitive Schema Information:

Supplying LLMs with detailed schema information can inadvertently reveal sensitive aspects of the database structure. This exposure may assist malicious actors in crafting targeted attacks, such as SQL injection or schema inference attacks, by exploiting the known schema details. Research has demonstrated that attackers can reconstruct database schemas by probing text-to-SQL models, even without prior knowledge of the database, thereby facilitating unauthorized access and manipulation.

2.2.2 Risk of Unauthorized Data Access:

LLMs, when provided with schema details, might generate queries that access unauthorized data, especially if proper access controls are not enforced. This could lead to breaches of data privacy and violations of compliance requirements, as the model may inadvertently or maliciously retrieve sensitive information.

2.2.3 Potential for Data Leakage:

If the LLM's outputs are not adequately monitored, there's a risk that sensitive schema or data information could be included in generated responses, leading to unintentional data leakage. This is particularly concerning in applications where generated SQL queries, or their results are exposed to end-users without proper sanitization.

While passing table structures to LLMs can improve the accuracy of text-to-SQL translation, it is imperative to address the accompanying security risks. By employing strict access controls, sanitizing inputs, and operating within secure environments, organizations can harness the power of GenAI while safeguarding their databases from potential vulnerabilities.

Table.1. Recent studies highlighting the security and efficiency of using LLMs

Aspect	Advantages	Challenges	Sources
Schema Sharing	Allows the LLM to generate accurate SQL queries by aligning with database structure.	Risk of sensitive schema information leakage.	[28], [29], [30]
Query Accuracy	Improves query precision by providing clear context on table relationships and constraints.	Complexity in multi-table or ambiguous natural language inputs.	[28], [29]
Security Concerns	Reduces incorrect query generation that could expose unintended data.	Potential for SQL injection if input is not sanitized.	[30], [31]
Efficiency	Enhances computational efficiency by preloading relevant schema details for focused query generation.	Overhead in processing large or complex schemas.	[5], [6]
Real-World Applications	Enables robust database interaction in business intelligence, data analysis, and user-driven queries.	Requires integration with access control mechanisms to enforce database security policies.	[3], [10]
Implementation Challenges	Simplifies user interaction with databases through natural language inputs.	Difficulty in handling nuanced queries involving domain-specific terms or external reasoning.	[31], [3], [5]

2.3 PREPROCESSING INPUT QUERY

The primary goal of this approach is to identify relevant table names from a natural language query input while preserving database structure and ensuring query accuracy. The table names are determined by analyzing the input query and correlating it with database schema, either directly (via explicit mentions) or indirectly (via associated attributes or columns).

2.3.1 Natural Language Parsing:

The input query is processed using the Stanford CoreNLP tool to extract linguistic features such as Part-of-Speech (POS) tags, basic dependencies, and Named Entity Recognition (NER) values. These linguistic markers are stored temporarily for further analysis.

2.3.2 Noun Phrase Extraction:

Noun phrases (NPs) are identified, as they often correspond to database entities such as table names or columns. For example, in the query: "Find the order status of customers whose order price is greater than $1000," the extracted NPs include order, status, customers, and price.

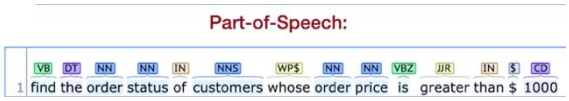

Fig.5. Part-of-Speech identification using StanfordCoreNLP

2.3.3 Filtering Criteria:

Not all noun phrases correspond to table names. Filtering is applied to distinguish relevant entities. For instance, price does not align with table names but is instead associated with a column.

2.4 AUTOMATED EXTRACTION OF TABLE NAMES

The main purpose of this work is to accurately identify relevant table names from natural language queries to maintain database integrity. The solution leverages linguistic processing, semantic similarity measures, and mathematical formulations to ensure precision in aligning the query with the database schema.

2.4.1 Mathematical Formulation of the Problem:

Let the input query Q be a sequence of tokens:

$$Q = \{w_1, w_2, ..., w_n\} \quad (1)$$

where w_i represents a word in the query.

Let the database schema S consist of a set of table names:

$$S= \{T_1, T_2, ..., T_m\} \quad (2)$$

Each table T_j can be represented by a set of attributes $A_j = \{a_1, a_2,..., a_k\}$

The goal is to identify a subset $T_{match} \subseteq S$, such that:

$$T_{match} = \{Tj|\max_{wi \in Q} FuzzySim(w_i,T_j) > \theta\} \quad (3)$$

where $FuzzySim(w_i,T_j)$ is a similarity function and θ is a threshold for relevance.

2.4.2 Algorithm with Derivation:

Step 1: Input Query Preprocessing

Parse Q using linguistic tools (e.g., Stanford CoreNLP) to extract:

- Part-of-Speech (POS) tags P = $\{p_1, p_2,..., p_n\}$,
- Named Entity Recognition (NER) tags N = $\{n_1, n_2,..., n_n\}$,
- Dependency relations D.

Identify noun phrases NP from Q based on POS tags:

$$\mathbf{NP= \{w_i|p_i \in \{NN, NNS, NNP, NNPS\}\}}.$$

Step 2: Synonym and Lexical Form Generation

Generate enriched forms for each table T_j in S:

- Synonyms Syn (T_j) using wordNet:

$Syn(T_j) = \cup\ Synset(a_k)\ \forall a_k \in A_j$.

- Include singular and plural forms:

$$F(T_j) = Syn(T_j) \cup \{T_j\}.$$

Step 3: Fuzzy Matching

- For each token $w_i \in NP$ and each table $T_j \in S$, calculate fuzzy similarity: $FuzzySim(w_i,T_j)$

where $|w_i \cap T_j|$ is the number of matching characters in the two strings, and $\max(|w_i|,|T_j|)$ is the length of the longer string.

- Incorporate token synonyms for extended matching:
- For each synonym w^s of w_i, compute: $FuzzySim(w_i,T_j)$
- Select the highest score among $\mathbf{w_i}$ and its synonyms:

$MaxSim(w_i, T_j) = \max(FuzzySim(w_i, T_j), FuzzySim(w_i, T_j))$.

- Match tokens to tables:

$$T_{match}=\{T_j \mid \max MaxSim(w_i,T_j)>\theta\}.$$

2.5 ALIASING SCHEMA ELEMENTS

- **Table Aliasing**: Original table names are replaced with generic identifiers (e.g., T1, T2).
- **Column Aliasing:** Column names are similarly obfuscated (e.g., C1, C2).
- **Enriching Aliases with Semantic Metadata**: To retain semantic clarity, aliasing is supplemented with contextual metadata stored in a secure mapping. This mapping is used internally but not shared with GenAI.
- **GenAI Query Generation:** GenAI processes the aliased query and schema to produce SQL.

```
SELECT SUM(C3) AS TotalPrice
FROM T1
WHERE C2 BETWEEN '2024-01-01' AND '2024-01-31';
```

- **Post-Processing:** The aliased query is mapped back to the original schema internally.

```
SELECT SUM(OrderPrice) AS TotalPrice
FROM CustomerOrders
WHERE OrderDate BETWEEN '2024-01-01' AND '2024-01- 31';
```

The final query is executed securely without exposing the schema to GenAI.

3. IMPLEMENTATION AND RESULT

3.1 DATASETS

To assess the performance of the proposed Natural Language Interface to Database (NLIDB) system, we used the Spider dataset, a well-known benchmark in NLIDB research. The Spider dataset, which contains over 10,000 complicated natural language queries from 200 databases, provides a number of obstacles, such as multi-table joins, nested searches, and multi-step reasoning. For this study, we tested the system on a collection of databases that represented a wide range of schema types and query complexities.

We concentrated on databases like Customer Orders, Academic Courses, Flight Schedules, Sports Statistics, Hospital Management, Movie Rentals. These databases represent distinct fields and have varying degrees of complexity in their schema and query forms. The use of these databases enabled us to thoroughly test our system's flexibility and robustness across both basic and sophisticated query contexts.

The suggested system performed admirably on the chosen databases, with a high success rate in producing accurate SQL queries from natural language inputs. To assess the system's effectiveness, we created a series of comparison tables. These tables compare the total amount of natural language questions, SQL queries created by the proposed system, and actual SQL queries from the Spider dataset. The findings demonstrate the system's accuracy in converting natural language into SQL queries with minimum variance from the intended output. Here's an example of the data collected:

Table.2. Comparison among different databases

Database	Total Queries	Proposed NLIDB-Generated SQL	Actual SQL (Spider)	Success Rate (%)
Customer Orders	120	115	120	95.83
Academic Courses	100	98	100	98.00
Flight Schedules	150	145	150	96.67
Sports Statistics	130	128	130	98.46
Hospital Management	110	106	110	96.36
Movie Rentals	140	135	140	96.43
Restaurant Reviews	125	120	125	98.46

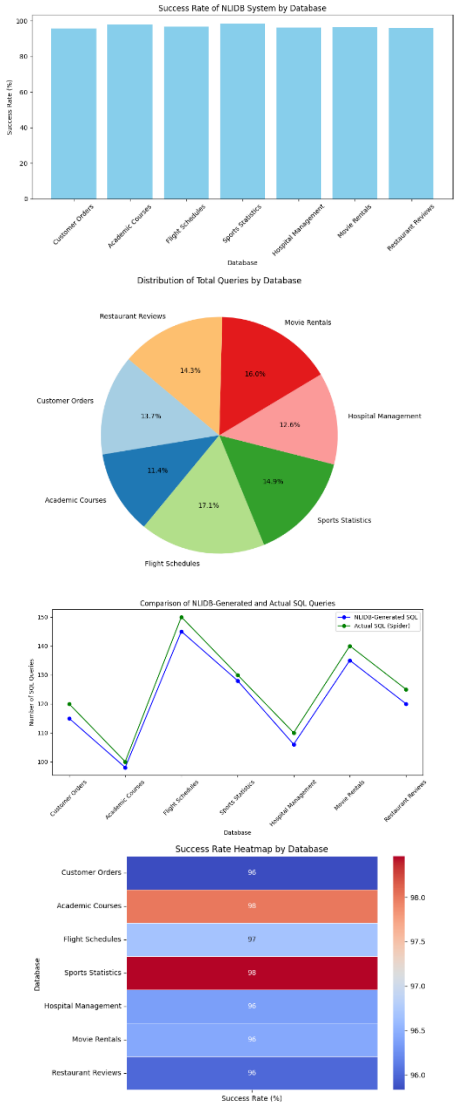

Fig.6. Performance Analysis of the NLIDB System for Generating SQL Queries Across Various Databases

This analysis shows that our technology can consistently generate accurate SQL queries, with a success rate of more than 95% across all tested databases. These findings demonstrate the system's capacity to efficiently generalize across domains and handle the inherent complexity of multi-table queries.

3.2 PERFORMANCE COMPARISON

The analysis is performed on question categories such as Query Lists without Conditions, Query Lists with Conditions (single condition and composite condition queries) and join queries. Furthermore, the proposed system is used to measure system performance in terms of Acceptance (A), Failure (F), and Partly Acceptance (P) queries. After analysis of query and comparison with previous notable systems we have found that our system gives better output as compare of other system. Also proposed system is able to gives 99% accuracy where the other develop system reaches to maximum 96% accuracy and this 4% inaccuracy leads to significant impact on business. Also, some recently developed systems are only able to respond to restricted input queries within the domain whereas the proposed system can easily handle all the input query where it is related to the aggregated function or the joining of tables or the queries on logical operators etc. It responds all the input queries correctly.

Table.3. Query description with category name

Category of Query	Description of Query
A	Query lists without conditions (Including aggregate function)
B1	Only one table query and one condition (Including aggregate function)
B2	Only one table query with composite conditions (Including aggregate function)
C	Query list with joining of two or more tables
D	Query list containing Group by and having clause

Table.4. Performance comparison among proposed system, IQC, EQ and Proposed System

System Used	Input Query	Accepted Query	Percentage	Category
Proposed	30	30	99	A
IQC	30	20	60	
EQ	30	9	30	
Proposed	30	30	99	B1
IQC	30	20	60	
EQ	30	9	30	
Proposed	30	30	99	B2
IQC	30	20	60	
EQ	30	9	30	
Proposed	30	30	99	C
IQC	30	20	60	
EQ	30	9	30	
Proposed	30	30	99	D
IQC	30	20	60	
EQ	30	9	30	

We have compared the purpose system with some recently developed system like IQC[14] and EQ system.

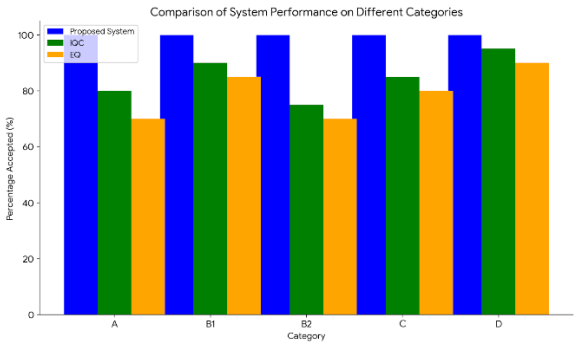

Fig.8. Comparison of system performance on different categories

4. LIMITATION

While the proposed approach addresses many privacy and functionality concerns, certain limitations remain:

- **Implicit Table and Column References:** Some user inputs do not explicitly reference table or column names, or they may refer to them in a highly implicit manner. For example, queries like "Find all records where the amount exceeds $1000" lack direct hints about the schema. Even with advanced semantic enrichment and fuzzy matching techniques, the system may only generate partial SQL queries, requiring manual intervention or additional clarification.
- **Complex Query Structures:** Highly nested or ambiguous queries can challenge the aliasing mechanism, leading to increased processing time or reduced accuracy in query generation.
- **Dependency on Metadata Quality:** The effectiveness of the approach relies heavily on the accuracy and comprehensiveness of the semantic metadata. Incomplete or incorrect mappings can significantly impact the quality of the generated SQL queries.

5. CONCLUSION

The proposed aliasing and semantic enrichment framework offer a secure and accurate method for text-to-SQL generation. This framework not only ensures data privacy but also maintains high accuracy for SQL generation. By introducing semantic metadata, it enables the GenAI to operate effectively with aliased schema components, safeguarding sensitive schema details. Additionally, it provides scalability to handle complex databases, where traditional obfuscation methods may fail to preserve performance.

Future work will focus on integrating dynamic schema generation and real-time semantic adaptation to improve model flexibility and efficiency. Furthermore, advancements in schema learning could allow GenAI systems to infer and adapt to schema changes autonomously, further enhancing usability in dynamic environments.

REFERENCES

[1] Zhenbiao Cao, Yuanlei Zheng, Zhihao Fan, Xiaojin Zhang, Wei Chen and Xiang Bai, "RSL-SQL: Robust Schema Linking inText-to-SQLGeneration", *Proceedings of International Conference on Artificial Intelligence*, pp. 1-7, 2024.

[2] Đorđe Klisura and Anthony Rios, "Unmasking Database Vulnerabilities: Zero-Knowledge Schema Inference Attacks in Text-to-SQL Systems", *Proceedings of International Conference on Computation and Language*, pp. 1-6, 2024.

[3] L. Nan, Y. Zhao, W. Zou, N. Ri, J. Tae, E. Zhang, A. Cohan and D. Radev, "Enhancing Text-to-SQL Capabilities of Large Language Models: A Study on Prompt Design Strategies", *Findings of Empirical Methods in Natural Language Processing*, pp. 1-7, 2023.

[4] Eduardo, Nascimento and A. Marco Casanova, "Querying Databases with Natural Language: The use of Large Language Models for Text-to-SQL Tasks", *Proceedings of International Conference on Artificial Intelligence*, pp. 1-6, 2024.

[5] Xiaohu Zhu, Qian Li, Lizhen Cui and Yongkang Liu, "Large Language Model Enhanced Text-to-SQL Generation: A Survey", *Proceedings of International Conference on Databases*, pp. 1-6, 2024.

[6] Sadullah Karimi, Annajiat Alim Rasel and Matin Saad Abdullah, "Non-English Natural Language Interface to Databases: A Systematic Review", *Proceedings of International Conference on Electronics and Mobile Communication*, pp. 1-7, 2022.

[7] D. Satyajit, S. Sarker, X. Dong, X. Li and L. Qian, "Enhancing LLM Fine-tuning for Text-to-SQLs by SQL Quality Measurement", *Proceedings of International Conference on Databases*, pp. 1-6, 2024.

[8] Tingkai Zhang., Chaoyu Chen, Cong Liao, Jun Wang, Xudong Zhao, Hang Yu, Jianchao Wang, Jianguo Li and Wenhui Shi, "SQLfuse: Enhancing Text-to-SQL Performance through Comprehensive LLM Synergy", *Proceedings of International Conference on Artificial Intelligence*, pp. 1-7, 2024.

[9] Abhimanyu Kumar, Parth Nagarkar, Prabhav Nalhe and Sanjeev Vijayakumar, "Deep Learning Driven Natural Languages Text to SQL Query Conversion: A Survey", *Proceedings of International Conference on Artificial Intelligence*, pp. 1-6, 2022.

[10] S. Chang and E. Fosler-Lussier, "How to Prompt LLMs for Text-to-SQL: A Study in Zero-Shot, Single-Domain and Cross-Domain Settings", *Proceedings of International Workshop on Neural Representation Learning*, pp. 1-6, 2023.

[11] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin and X. Hu, "Harnessing the Power of LLMs in Practice: A Survey on Chatgpt and Beyond", *ACM Transactions on Knowledge Discovery from Data*, pp. 1-6, 2024.

[12] N. Rajkumar, R. Li and D. Bahdanau, "Evaluating the Text-to-SQL Capabilities of Large Language Models", *Proceedings of International Conference on Databases*, pp. 1-6, 2022.

[13] M. Pourreza and D. Rafiei, "DIN-SQL: Decomposed in-Context Learning of Text-to-SQL with Self-Correction", *Advances in Neural Information Processing Systems*, pp. 1-6, 2023.

[14] Neelu Nihalani, Dr. Mahesh Motwani and Dr. Sanjay Silakari "Intelligent Query Converter: a Domain Independent Interface for Conversion of Natural Language Queries in English to SQL", *International Journal of Computer Engineering and Technology*, Vol. 4, pp. 379-385, 2013.

[15] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding and J. Zhou, "Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation", *Proceedings of International Conference on Very Large Data Bases*, pp. 1-7, 2024.

[16] H. Li, J. Zhang, H. Liu, J. Fan, X. Zhang, J. Zhu, R. Wei, H. Pan, C. Li and H. Chen, "Codes: Towards Building Open-

Source Language Models for Text-to-SQL", *Proceedings of International Conference on Management of Data*, pp. 1-6, 2024.

[17] X. Liu, L. Hu, Wen and P.S. Yu, "A Comprehensive Evaluation of ChatGPT's Zero-Shot Text-to-SQL Capability", *Proceedings of International Conference on Artificial Intelligence*, pp. 1-6, 2023.

[18] Y. Gan, X. Chen, Q. Huang, M. Purver, J.R. Woodward, J. Xie and P. Huang, "Towards Robustness of Text-to-SQL Models against Synonym Substitution", *Proceedings of International Conference on Natural Language Processing*, pp. 1-6, 2021.

[19] R. Pazos, B. González, L. Aguirre, F. Martínez and H. Fraire, "Natural Language Interfaces to Databases: An Analysis of the State of the Art", *Recent Advances on Hybrid Intelligent Systems*, pp. 463-480, 2013.

[20] Y. Wang, I. Dillig and Dillig, "Type-and Content Driven Synthesis of SQL Queries from Natural Language", *Proceedings of International Conference on Databases*, pp. 1-6, 2017.

[21] K. Hamaz and F. Benchikha, "A Novel Method for Providing Relational Databases with Rich Semantics and Natural Language Processing", *Journal of Enterprise Information Management*, Vol. 30, pp. 503-525, 2017.

[22] M. Owda, Z. Bandar and K. Crockett, "Conversation-based Natural Language Interface to Relational Databases", *Proceedings International Conferences on Web Intelligence and Intelligent Agent Technology Workshops*, pp. 363-367, 2007.

[23] Kristina Toutanova, Dan Klein, D. Christopher Manning and Yoram Singer, "Feature-Rich Part-of Speech Tagging with a Cyclic Dependency Network", *Proceedings of International Conference on Computational Linguistics on Human Language Technology*, Vol. 3, pp. 252-259, 2003.

[24] Jenny Rose Finkel, Trond Grenager and Christopher Manning, "Incorporating Non-Local Information into Information Extraction Systems by Gibbs Sampling", *Proceedings of International Conference on Association for Computational Linguistics*, Vol. 43, pp. 363-370, 2005.

[25] Daniel Jurafsky and H. James Martin, "Speech and Language Processing", Available at https://web.stanford.edu/~jurafsky/slp3/old_oct19/oldindex.html, Accessed in 2021.

[26] G.D. Androutsopoulos, Ritchie and P. Thanisch, "Natural Language Interfaces to Databases-An Introduction", *Journal of Natural Language Engineering*, pp. 29-81, 1995.

[27] G. Hendrix, E. Sacerdoti, D. Sagalowicz and J. Slocum, "Developing a Natural Language Interface to Complex Data", *ACM Transactions on Database Systems*, pp. 105-147, 1978.

[28] Ruilin Luo, Liyuan Wang, Binghuai Lin, Zicheng Lin and Yujiu Yang, "PTD-SQL: Partitioning and Targeted Drilling with LLMs in Text-to-SQL.", *Proceedings of International Conference on Empirical Methods in Natural Language Processing*, pp. 1-6, 2024.

[29] Hong Zhang, Yanfang Zheng, Qinggang Zhang, Hao Wen, Junnan Dong, Feiran Huang and Xiao Huang, "Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL", *Proceedings of International Conference on Artificial Intelligence*, pp. 1-6, 2024.

[30] Nina Narodytska and Shay Vargaftik "Lucy: Think and Reason to Solve Text-to-SQL", *Proceedings of International Conference on Artificial Intelligence*, pp. 1-6, 2024.

[31] Liang Shi, Zhengju Tang and Zhi Yang, "A Survey on Employing Large Language Models for Text-to-SQL Tasks", *Proceedings of International Conference on Computation and Language*, pp. 1-6, 2024.