Available online at website: https://jurnal.iaii.or.id/index.php/RESTI

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 10 No. 4 (2026) 949 - 957 e-ISSN: 2580-0760

Schema-Guided Prompt Strategies for Text-to-SQL over Relational Databases Using Local LLMs

Nurjayanti[1], Adiwijaya[2], Ade Romadhony[3], Alfian Akbar Gozali[4]
[1]Department of Software Engineering, School of Computing, Telkom University, Bandung, Indonesia
[2,3]Department of Informatics, School of Computing, Telkom University, Bandung, Indonesia
[4]Department of Application Software Engineering, School of Applied Science, Telkom University, Bandung, Indonesia

*jayanti@telkomuniversity.ac.id

Abstract

Text-to-SQL systems translate natural language questions into executable SQL queries, allowing users without SQL expertise to access structured data stored in relational databases. Although Large Language Models (LLMs) have substantially improved SQL generation capabilities, many state-of-the-art Text-to-SQL approaches continue to rely on cloud-based models with high computational requirements. Such dependence limits their deployment in environments with limited computing resources. This study addresses this limitation by proposing schema-guided prompting strategies for Text-to-SQL generation using local LLMs. A chat-based application was developed using the Django Web Framework, while model inference was performed through the Ollama platform to enable the deployment of local LLMs. The proposed framework incorporates database schema information, including table structures and column attributes, into structured prompts to improve the alignment between natural language questions and SQL generation. Experiment results across multiple databases demonstrate that schema-guided prompting significantly improves Text-to-SQL performance. The highest accuracy was achieved by LLaMA 3 (8B) with objective-aware prompting, reaching an Exact Matching (EM) accuracy of 71.96%. These findings suggest that structured prompt engineering provides a practical alternative to model fine-tuning for locally deployed LLMs, offering an effective balance between SQL generation accuracy, computational efficiency, and data privacy. Future work will investigate fine-tuning strategies, example selection methods, and cross-domain evaluation to enhance SQL generation.

Keywords: databases; generation; large language models; prompt engineering; schema-guided prompting; SQL relational; Text-to-SQL

How to Cite: N. Nurjayanti, A. Adiwijaya, A. Romadhony, and A. A. Gozali, "Schema-Guided Prompt Strategies for Text-to-SQL over Relational Databases Using Local LLMs", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 10, no. 4, pp. 949 - 957, Jul. 2026.
Permalink/DOI: https://doi.org/10.29207/resti.v10i4.6966

Received: July 16, 2025
Accepted: July 7, 2026
Available Online: July 30, 2026

This is an open-access article under the CC BY 4.0 License
Published by Ikatan Ahli Informatika Indonesia

1. Introduction

Relational databases are widely used to store and manage structured data throughout various domains [1], [2], [3]. Accessing information stored in these databases typically requires Structured Query Language (SQL), the standard language for querying and manipulating relational data. However, writing SQL queries demands technical expertise that many end users do not possess, making database access difficult for users without programming experience [4], [5]. This limitation has motivated the development of Text-to-SQL systems that allow users to interact with databases in natural language by translating natural language questions into executable SQL queries.

Early approaches relied on rule-based models that utilized handcrafted linguistic rules and predefined query templates [5]. These methods later evolved into keyword-based [6] and pattern-based [7], [8] querying approaches that attempted to map natural language expressions to SQL components using manually defined rules. Grammar-based methods were introduced to enforce syntactic constraints on SQL query generation and improve query correctness [9]. With advances in machine learning, recent studies have also adopted neural network models such as sequence-to-sequence architectures based on Long Short-Term Memory (LSTM) networks, which have demonstrated the ability to learn mappings between natural language questions and SQL queries from annotated datasets [10]. Transformer-based architectures have also

achieved significant advances by capturing complex relationships between natural language questions and database schemas [11]. Representative models include IRNet [12], which combined a transformer encoder with an LSTM decoder guided by abstract syntax trees, and RAT-SQL [13], which introduced schema-aware self-attention mechanisms and achieved 65.6% exact-match (EM) accuracy on the Spider development benchmark when augmented with BERT.

The development of large-scale datasets has further accelerated research in these tasks. Public benchmarks such as Spider[14], BIRD[15], KaggleDBQA[16], and WikiSQL[10] offer challenging cross-domain evaluation that includes complex SQL operators such as *where*, *join*, *group by*, as well as aggregation functions and nested subqueries. These benchmarks require models to generalize over different database schemas and application domains, reflecting real-world database scenarios. Beyond these cross-domain benchmarks, several studies have explored domain-specific applications such as finance [17] and healthcare [18], demonstrating their potential to support natural language interaction with domain-specific databases.

Recent advances in large language models (LLMs) have led to notable improvements in the state of the art of natural language processing (NLP) tasks, including code generation, question answering, and Text-to-SQL [19], [20], [21]. Early LLM-based approaches use pre-trained models (PLMs) such as GPT-4 and LLaMA to translate natural language questions directly into SQL queries. Instead of requiring extensive task-specific training, many recent methods employ prompt engineering, where task instructions, schema descriptions, and example queries are embedded into prompts to guide SQL generation [22]. This paradigm enables flexible adaptation across different models, reduces the need for expensive model training, and benefits from continuous improvements in PLMs [4].

Prompt engineering in LLM-based approaches typically involves two main components, namely question representation and example selection. In question representation, database schema information such as table and column names and their relationships is converted into textual descriptions and incorporated into prompts alongside task instructions [23], [24], [25]. Meanwhile, in example selection, representative question–SQL pairs are entered into the prompt to guide the model in generating queries for new inputs [26], [27]. Previous studies have shown that selecting diverse and relevant examples can greatly improve SQL generation performance [26]. However, formulating efficient prompts that include extensive schema information and many examples remains challenging. Large prompts may increase computational cost, inference latency, and token consumption, especially when using API-based LLM services [22], [28].

Several recent studies have explored additional strategies to improve Text-to-SQL performance. These include contextual prompting [29], [30], knowledge graph integration [1], [2], [31], structured reasoning mechanisms [4], multi-generator ensemble approaches [25], and ranking-enhanced encoders with skeleton parsing [32]. Although these approaches demonstrate promising results, many recent LLM-based approaches remain computationally expensive and difficult to deploy in local environments. Furthermore, they found it challenging to handle complex database schemas, multi-table queries, and diverse natural language expressions, especially when applied to unseen databases [33]. For example, GPT-4-based decomposition methods such as DIN-SQL [25] and PET-SQL [23] report execution accuracies of 85.3% and 87.6%, respectively, on the Spider benchmark, but this level of performance depends on repeated calls to large, cloud-hosted proprietary models. These challenges call for efficient prompt strategies that improve SQL generation performance while maintaining computational efficiency and deployment flexibility.

To address these challenges, this study proposes a schema-guided prompt strategy for Text-to-SQL generation using locally deployed LLMs. The proposed approach organizes database schema information, task instructions, and representative examples into structured prompts to improve alignment between natural language questions and relational database schemas. A chat-based database interaction system was implemented, enabling users to query relational databases in a unified interface through natural language. The system uses locally deployed LLMs via the Ollama platform, enabling SQL queries to be generated without relying on external cloud-based services. In this study, LLaMA 3 and SQLCoder are evaluated for SQL generation without additional fine-tuning. We hypothesize that explicitly structuring schema and query-intent information within the prompt improves SQL generation accuracy relative to a minimal-instruction baseline, without requiring model fine-tuning.

Accordingly, there are three main contributions of this study: (1) we propose and empirically compare three schema-guided prompting strategies for Text-to-SQL generation: basic prompt, objective-aware, and few-shot; (2) we develop a chat-based application using Django and the Ollama platform that operates entirely without external API calls, addressing privacy, cost, and connectivity limitations associated with cloud-based solutions; and (3) we provide an error analysis characterizing the specific failure modes, including table/column ambiguity, incorrect *join* identification, and example-selection bias that persist even under schema-guided prompting, informing directions for future local-LLM Text-to-SQL research.

The remainder of this paper is organized as follows. Section 2 describes the system architecture and methodology. Section 3 presents the results and discusses the findings. Finally, Section 4 presents the conclusion and ideas for future research.

2. Methods

This section describes the methodology adopted to develop and evaluate the proposed Text-to-SQL system. The proposed framework integrates a chat-based application, schema-guided prompt construction, locally deployed large language models, and SQL execution over relational databases. The methodology covers the overall system architecture, dataset configuration, prompt engineering strategies, and the experimental setup used to evaluate the proposed approach.

2.1 System Architecture

The proposed Text-to-SQL system architecture is illustrated in Figure 1. The architecture consisted of three layers: user interaction, schema-guided prompt generation, and SQL execution.

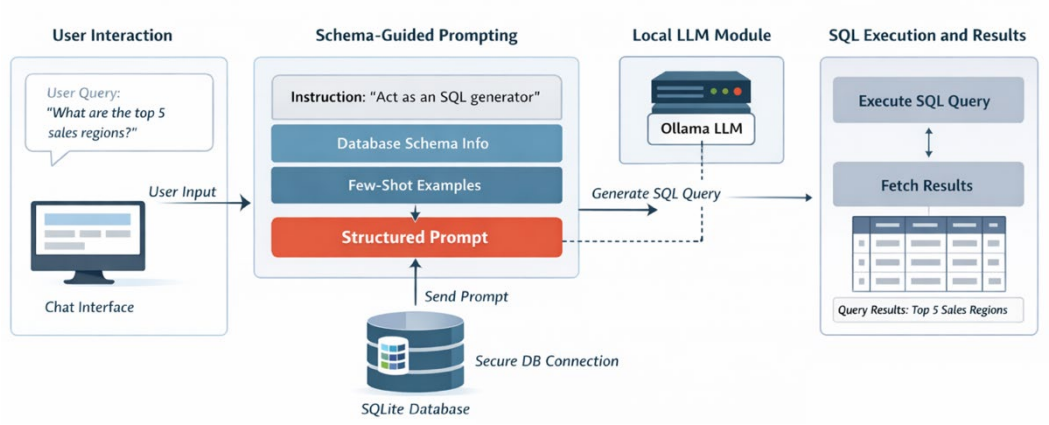


Figure 1. System Architecture for Text-to-SQL using Local LLMs

The user interaction layer is implemented as a chat-based application developed using the Django Web Framework. Through this interface, users can submit natural language questions about the database. The interface also provides schema information, allowing users to explore available database tables and columns before submitting queries.

Once a query is submitted, the schema-guided prompt construction layer retrieves information from the relevant database schema. This information, along with the user's question and task instructions, is used to create a structured prompt. Depending on the selected prompting strategy, the prompt may also include objective-aware guidance or representative few-shot examples to provide additional contextual information for SQL generation.

The constructed prompt is then processed by the local LLM component, where LLaMA 3 or SQLCoder is deployed through the Ollama platform. Running the models locally enables secure and low-latency SQL generation while eliminating the need for external cloud-based services.

In the final stage, the generated SQL query is forwarded to the query execution layer. The query results alongside the generated SQL are displayed through the application interface, allowing users to verify how their natural language questions are interpreted.

2.2 Dataset and Database Configuration

The proposed framework was evaluated using the Spider development dataset [14], a widely adopted benchmark for Text-to-SQL research. The Spider dataset contains natural language questions paired with ground-truth SQL queries across multiple relational database schemas; making it suitable for evaluating cross-domain SQL generation.

For the experiments, four representative databases from the Spider development set were selected: Singer, Concert_Singer, Real_Estate_Properties, and Course_Teach. Each evaluation instance consists of three components: a user question, the corresponding database schema, and the expected SQL query. Together, these components provide a reliable basis for assessing the generated SQL statements across databases with different schema characteristics.

Figure 2 illustrates the relationship between a user question, the underlying database schema, and the corresponding ground-truth SQL query. This example demonstrates how the proposed framework maps user questions to executable SQL statements by leveraging structural information in the database schema.

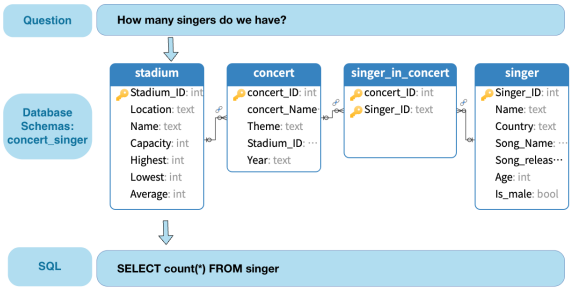


Figure 2. Text-to-SQL task illustration

2.3 Prompt Engineering Strategy

Prompt engineering plays a central role in guiding LLMs to generate accurate and executable SQL queries. In the baseline configuration, the prompt consisted of

three main components: a task instruction, user question, and the corresponding database schema information. This basic prompt structure provides the model with sufficient information to generate SQL queries. An example of the basic prompt configuration and the corresponding response is shown in Figure 3.

Prompt
You are a SQL generator. Convert the question into a valid SQL query using the provided schema. Table: head Columns: head_ID, name, born_state, age Question: How many heads of the departments are older than 56 ? SQL:
LLM Response SQL: SELECT COUNT(h.head_ID) FROM ...

Figure 3. Prompt and LLM Response for Initial Setup

Inspired by previous work [4], we further experimented with an objective-aware prompting strategy. In this approach, the prompt explicitly identifies key elements of the user query, including the objective of the inquiry, the expected output data, and the key entities involved in the query. Organizing these elements within the structured prompt helps reduce ambiguity and provides clearer guidance for SQL generation. Examples of prompts and model responses for user question analysis and SQL composition are shown in Figure 4 and 5.

Prompt
Instruction: Read the question to identify the required data to query and return. Assess if subqueries are needed. Table: head Columns: head_ID, name, born_state, age Question: How many heads of the departments are older than 56 ?
LLM Response To Be Queried: ['heads of departments'] To Return: ['number of heads of departments'] Table Needed: ['head'] Subquery Required:['no']

Figure 4. Prompt and LLM Response for Question Analysis

Prompt
Instruction: Analyze the intent of SQL clauses and subqueries, then formulate a SQL query. Table: head Columns: head_ID, name, born_state, age Question: How many heads of the departments are older than 56 ? Keyword Indicator: The SQL query is likely to need: FROM, WHERE, SELECT, COUNT
LLM Response Clause Analysis: FROM clause for retrieving … WHERE clause for filtering … Subquery Analysis: - SQL: SELECT count(head_ID) FROM …

Figure 5. Prompt and LLM Response for SQL Composition

In addition to instruction-based prompting above, we also evaluated few-shot prompting where several examples of question-SQL pairs were included in the prompt. This strategy uses in-context learning of the model's capability, allowing it to infer patterns in SQL generation from the provided examples.

These examples can help the model better understand query structures, especially for complex database operations involving joins, filtering conditions, and aggregation functions. An example of a few-shot prompt configuration is presented in Figure 6.

Prompt
You are a Text-to-SQL expert. Instruction: Given a database schema and a question, generate the correct SQL query. Examples: Schema: Table: head Columns: head_ID, name, born_state, age Question: How many heads of the departments are older than 56 ? SQL: select count(head_id) from heads where age>= 56; Schema: Table: head Columns: head_ID, name, born_state, age Question: How many heads of the departments are older than 56 ?
LLM Response SQL: SELECT count(head_ID) FROM …

Figure 6. Prompt and LLM for Few-Shot Examples

2.4 Experimental Setup

Experiments were conducted using LLaMA 3 (8B) and SQLCoder (7.2B) deployed locally through the Ollama platform with each model's default quantization and inference settings. All experiments were performed on an Apple M4 processor, 16 GB unified memory, and macOS Sequoia.

For the few-shot prompting strategy, three example question–SQL pairs (3-shot) were randomly selected once for each database and kept fixed throughout all evaluations to ensure consistent comparisons across prompting strategies. Model performance was evaluated using Exact Matching (EM) accuracy. This metric measures whether the generated SQL query exactly matched its ground-truth counterpart [14]. Prior to evaluation, generated SQL queries were post-processed to remove non-SQL formatting resulting from the LLMs, such as Markdown code fences.

3. Results and Discussions

This section presents the experimental results and discusses the proposed prompt strategies and system components in supporting natural language interaction with relational databases.

3.1 System Implementation and Query Processing

The proposed system implements an end-to-end pipeline that generates executable SQL queries from natural language questions. When users submit a question through the interface, the application constructs a prompt by combining it with the corresponding database schema and task instructions. This constructed prompt is then forwarded to the Ollama service, where the SQL query is generated.

Before execution, the generated SQL is post-processed to remove non-SQL formatting, such as markdown code blocks, ensuring that only valid SQL statements are executed. The cleaned query is then submitted to the

SQLite database, where the execution module retrieves both the query results and the associated column metadata. These outputs are formatted into a structured table and presented through the chat interface.

The implementation demonstrates the seamless integration of prompt construction, local LLM inference, SQL execution, and result presentation within a single workflow. By automating the entire Text-to-SQL pipeline, the system allows users to interact with relational databases using natural language without requiring SQL expertise. Furthermore, executing through the Ollama platform enhances data privacy and reduces inference latency. This makes the proposed system suitable for environments where cloud-based services are impractical or restricted.

3.2 Schema Extraction and Context Generation

Schema extraction is a key component of the proposed Text-to-SQL system because it provides database schema information needed for accurate SQL generation. The system automatically retrieves metadata from the SQLite database, including table names and column attributes, and incorporates this information into the prompt provided to the LLM. By explicitly providing the database schema, the model can better relate natural language questions to the underlying tables and columns, reducing ambiguity during query generation.

The extracted schema serves two main purposes. First, it enriches the prompt with contextual information that helps the model identify the relevant database entities when generating SQL queries. This additional context is particularly beneficial for questions involving multiple tables, filtering conditions, or aggregation operations, where a clear understanding of the schema is essential. Similar observations have been reported in previous Text-to-SQL studies, demonstrating that schema-aware prompting improves the alignment between natural language queries and relational database structures.

Second, the extracted schema information is presented through the interface, allowing users to browse available tables before submitting their questions. This feature helps users become familiar with the existing database structure and formulate more specific questions, thereby improving the system's overall usability.

By automatically integrating schema information into both the prompt generation process and the user interface, the proposed system can adapt automatically to changes in the database schema without manual reconfiguration. This design not only simplifies system maintenance but also improves scalability, which is suitable for deployment across different relational database applications.

3.3 Evaluation of Schema-Guided Prompt Strategies

To evaluate the effectiveness of different prompt designs in guiding SQL generation, three prompting strategies were investigated: a basic prompt configuration, an objective-aware prompt strategy, and an example-based (few-shot) prompting approach. These strategies provide different levels of contextual guidance to examine their impact on the ability of locally deployed LLMs to generate accurate SQL queries from natural language questions.

The basic prompting strategy served as the baseline configuration. The prompt consisted of a task instruction, user question, and the corresponding database schema. In comparison, this configuration provided sufficient information for generating relatively simple queries. Its performance declined when handling more complex requests involving multiple tables, aggregation functions, or ambiguous language. Using this baseline, LLaMA 3 (8B) achieved an average EM accuracy of 65.1%, indicating that schema information alone is insufficient for reliably generating complex SQL queries.

To improve user question interpretation, an objective-aware prompting strategy was introduced based on the structured reasoning approach proposed in previous Text-to-SQL studies [4]. In addition to the database schema, the prompt explicitly describes the objective of the question and highlights the key entities and attributes involved in the expected SQL query. This additional semantic guidance reduces ambiguity and helps the model better map natural language expressions to the corresponding database schema. As shown in Table 1, objective-aware prompting achieved the best overall performance, with LLaMA 3 (8B) reaching an average EM accuracy of 71.96%. The consistent improvement across the evaluated databases suggests that explicitly representing user intent is particularly beneficial for queries involving filtering conditions, aggregation operations, and multi-table relationships.

The third experiment investigated an example-based prompting strategy in which several question–SQL pairs were included in the prompt. This approach leverages in-context learning, enabling the model to infer SQL query structures from provided examples. Even though the examples in this study were randomly selected without considering semantic similarity or schema alignment, this approach still improved performance over the baseline. Under this configuration, the LLaMA 3 model achieved an average EM accuracy of 68.91%, indicating that providing example queries can help guide the model toward more structured SQL outputs.

These findings support our initial hypothesis that explicitly structuring schema and query-intent information within the prompt improves SQL generation accuracy relative to a minimal-instruction baseline. Without requiring model fine-tuning, objective-aware prompting raised LLaMA 3's average EM accuracy from 65.1% (basic prompt) to 71.96%, a gain of nearly 7 percentage points achieved through prompt design alone. This improvement is notable

given that comparable gains reported elsewhere in the Text-to-SQL literature typically rely on decomposed multi-step inference pipelines or large proprietary cloud models; for example, DIN-SQL [25] and PET-SQL [23] report execution accuracies of 85.3% and 87.6%, respectively, on Spider using GPT-4. Our best configuration, using an 8B-parameter model deployed entirely on local hardware, remains below these cloud-hosted results in absolute terms, reflecting the expected trade-off between the privacy-preserving, resource-efficient deployment this study targets and the raw capability of much larger proprietary models. This suggests that in privacy-, cost-, or connectivity-constrained settings, schema-guided prompting on local 8B-scale models offers a practical middle ground rather than a direct replacement for cloud-hosted LLM pipelines.

These findings highlight the importance of prompt engineering in LLM-based Text-to-SQL systems. Among the evaluated strategies, objective-aware prompting produced the most uniform performance improvements, particularly when combined with models that possess strong natural language reasoning capabilities, such as LLaMA 3. These results indicate that carefully designed prompt structures can act as an effective alternative to costly model fine-tuning for real-world database applications.

3.4 Model Results Comparison

The experiments were performed using two open-source LLMs available through the Ollama platform: LLaMA 3 (8B) and SQLCoder (7.2B). Model performance was measured using Exact Matching (EM) [14] accuracy, which measures the percentage of generated SQL queries that exactly match the ground-truth query in terms of structure and syntax. Following the standard Spider evaluation protocol [14], EM is calculated as shown in Equation 1.

$$EM = \frac{\sum_{i=1}^{N} \mathbb{1}\left(\wedge_{C_k \in \mathbb{C}} Y_i^{C_k} = \hat{Y}_i^{C_k}\right)}{N} \quad (1)$$

N denotes the number of evaluated queries, C represents the set of SQL components, Y_i denotes the ground-truth SQL query, and $\hat{Y}_i$ represents the generated SQL query. A query is considered correct only when all SQL components match the ground-truth query.

Table 1 presents the detailed experimental results, while Figure 7 provides a visual comparison of model performance across the three prompting strategies. Both indicate that LLaMA 3 consistently outperformed SQLCoder across most experimental configurations, despite SQLCoder being specifically designed for SQL generation tasks. This finding suggests that LLaMA 3 has stronger semantic reasoning capabilities, enabling more effective alignment between user question and database schema. In addition, example-based prompting achieved its highest single-database performance on DB3 (87.5%), considerably higher than its average EM accuracy of 68.91%. Since this improvement was not consistently observed across the remaining databases, the results suggest that randomly selected examples are sensitive to database schema characteristics. This observation highlights the need for more systematic example retrieval or selection strategies.

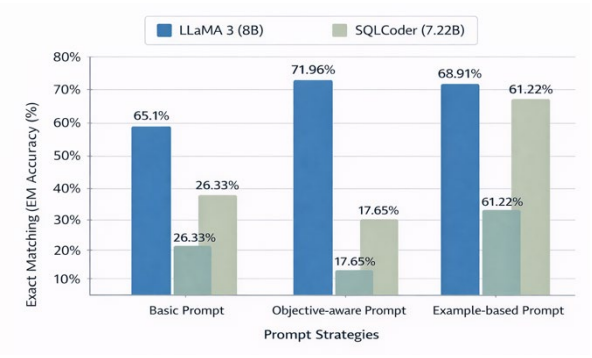


Figure 7. Comparison of Exact Matching (EM) accuracy for LLaMA 3 and SQLCoder under different prompting strategies. Objective-aware prompting achieves the best performance for LLaMA 3, while example-based prompting significantly improves SQLCoder performance.

Table 1. Exact Matching Accuracy (EM) on the Spider Development Dataset

Methods	DB 1	DB 2	DB 3	DB 4	Avg EM
Basic Prompt + LLaMA 3 8B	71.5	63.53	58.75	65.07	65.1
Basic Prompt + SQLCoder 7.2B	38.66	17.42	26.78	27.30	26.33
Objective-aware Prompt + LLaMA 3 8B	75.97	72.94	60.83	69.85	71.96
Objective-aware Prompt + SQLCoder 7.2B	24.32	11.51	25.21	19.19	17.65
Example-based Prompt + LLaMA 3 8B	71.95	65.88	87.5	67.85	68.91
Example-based Prompt + SQLCoder 7.2B	66.02	61.67	41.87	58.32	61.22

DB 1: singer; DB 2: concert_singer; DB 3: real_estate_properties; DB 4: course_teach

The relatively lower performance of SQLCoder in this study may be explained by the absence of domain-specific fine-tuning and its limited ability to generalize across complex database schemas in zero-shot scenarios. Nevertheless, SQLCoder demonstrated improved performance when evaluated using example-based prompting, achieving an average EM accuracy of 61.22%. This finding suggests that in-context examples can partially compensate for the lack of task-specific training.

Among the evaluated prompt strategies, objective-aware prompting, particularly for LLaMA 3, produced the best overall performance. This finding underscores the importance of guiding LLMs with structured representations of user intent and schema semantics. Explicitly defining the query's objective helps the model reduce ambiguity and generate more accurate SQL structures.

3.5 Error Analysis

Several error types were observed during the experiments. Although the proposed schema-guided

prompting strategies improved the accuracy of SQL generation, these errors primarily occurred in queries involving complex database schemas and ambiguous natural language expressions.

One common type of error involved the incorrect selection of tables or columns. In some cases, the model generated SQL queries that referenced semantically related but incorrect columns. These happened when multiple attributes with similar meanings were present in the schema. This issue was particularly noticeable in databases with large schemas or tables containing overlapping column names.

Another frequent error occurred during *join* operations in multi-table queries. Although database schema information was included in the prompt, the model occasionally failed to correctly identify the relationships between tables, resulting in incomplete or incorrect *join* clauses. This limitation suggests the need for additional reasoning mechanisms or explicit relationship representations to improve query synthesis in complex relational schemas.

Errors were also observed in queries involving aggregation functions and nested subqueries. The models generated inaccurate SQL queries in several cases: they correctly identified the relevant tables but applied incorrect aggregation functions or filtering conditions. These errors indicate that the model struggles with the deeper logical reasoning required for complex SQL structures; it may correctly interpret the entities mentioned in the natural language question.

The few-shot prompting strategy sometimes introduced structural bias when randomly selected examples were not closely related to the user question or database schema. As a result, the models occasionally produced SQL queries that followed patterns from unrelated examples rather than the intended query structure. In this strategy, a more effective example selection strategy based on semantic similarity or schema relevance is needed.

Finally, the error analysis indicates that schema-guided prompts significantly improve SQL generation; further gains can be achieved by refining example selection strategies and incorporating additional reasoning mechanisms into the prompt design.

3.6 Limitations and Future Works

Although the experimental results demonstrate the effectiveness of schema-guided prompt strategies for improving SQL generation accuracy, several limitations should be considered when interpreting the findings.

First, the experiments were conducted using four databases from the Spider development dataset; these cannot fully capture the diversity and complexity of real-world database environments, which often include larger schemas, domain-specific terminology, and continuously evolving database structures. Consequently, the findings should be interpreted as being specific to the evaluated datasets. Further validation is needed to confirm their broader applicability to more diverse real-world databases.

Second, the evaluated models were used in zero-shot scenarios without additional fine-tuning and might constrain their performance. Though this configuration reflects realistic deployment scenarios where fine-tuning resources may be limited, domain-specific fine-tuning could improve SQL generation accuracy. In particular, this is true for complex queries involving multiple tables, nested subqueries, or advanced SQL operations.

Third, the performance of the example-based prompting strategy observed in this study may underestimate the likely improvements achievable through more advanced example retrieval techniques. Previous studies suggest that example-selection methods based on semantic similarity or schema relevance can improve in-context learning performance of language models.

Finally, model performance was evaluated primarily using Exact Matching (EM) accuracy. Although EM is the standard metric for Text-to-SQL evaluation, it does not account for semantically equivalent SQL queries that differ syntactically but produce identical execution results. In addition, the observed performance differences should be interpreted as indicative rather than statistically conclusive; the study did not include significance testing (e.g., paired comparisons across databases) due to the limited number of evaluation databases.

Future work may incorporate additional evaluation metrics to deliver a more comprehensive assessment of SQL generation in Text-to-SQL, such as execution accuracy (EX).

3.7 Text-to-SQL Application for Database Interaction

Our developed application shows how the proposed schema-guided prompting can be applied to support natural language interaction with relational databases. As shown in Figure8, users can submit questions and receive both the generated SQL query and its execution results through a single interface.

One advantage of the proposed system is that the LLMs are deployed locally using the Ollama platform. Unlike cloud-based services, local deployment allows database queries to be processed entirely within the local environment, therefore eliminating the need to transmit potentially sensitive information to external servers. This approach not only helps protect data privacy but also reduces network latency, making the system suitable for organizations where security, confidentiality, and reliable offline operation are important considerations.

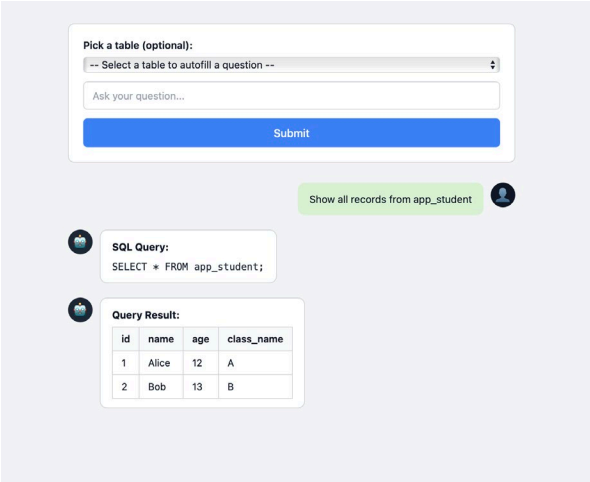


Figure8. Chat Application Interface

The interface was designed to make the query generation process more transparent to users. Along with the query results, the system displays the SQL statement generated from the user's natural language question. Therefore, users can understand how their requests are interpreted and verify the generated query before using the results. Such transparency can increase user reliance while also helping students and novice users learn how natural language questions are translated into SQL queries.

By combining structured prompts, local LLM deployment, and an interactive chat interface, the proposed system offers a practical solution for improving database accessibility while maintaining privacy and operational efficiency. Overall, the application demonstrates that schema-guided prompting can be integrated into a practical database interface without requiring users to have prior SQL knowledge.

4. Conclusions

This study presents a Text-to-SQL system that enables database interaction using natural language. The proposed approach adopted a schema-guided prompt strategy that combines database schema information, task instructions, and query examples into structured prompts to improve SQL generation.

The results of experiments confirm that prompt design has a significant impact on SQL generation performance. Specifically, objective-aware prompting raised average EM accuracy from 65.1% with the basic prompt to 71.96%, a gain of nearly 7 percentage points achieved through prompt design alone. Among the evaluated strategies, objective-aware prompting and the LLaMA 3 (8B) model achieved the best performance with an average EM accuracy of 71.96%. The results also show that general-purpose language models (i.e., LLaMA 3) can outperform specialized models like SQLCoder in zero-shot scenarios due to their stronger natural language reasoning capabilities.

For the broader Text-to-SQL research community, these results suggest that prompt engineering can meaningfully close part of the gap for locally deployable LLMs, rather than costly fine-tuning or reliance on cloud models. Thus, database access is more attainable for organizations facing data-privacy, budget, or connectivity constraints, such as small enterprises, government agencies, or healthcare providers operating under strict data-residency requirements. Beyond Text-to-SQL, the schema-guided prompting evaluated in this study may also apply to other structured generation tasks. By explicitly incorporating user intent and structured schema information into the prompt, the same principles could support applications such as natural language-to-API translation, structured code generation, and document-grounded question answering over domain-specific knowledge bases.

The findings underscore the importance of structured prompt engineering in enabling efficient Text-to-SQL systems. This is especially important in environments where computational resources, privacy, or network connectivity limit the use of cloud-based LLMs. Future work will investigate fine-tuning strategies, improved example selection methods, and cross-domain evaluation to enhance SQL generation performance further across diverse database schemas.

Acknowledgements

The authors declare no conflicts of interest regarding the publication of this paper. This work was supported in part by the Directorate of Research, Technology, and Community Service.

References

[1] J. Sequeda, D. Allemang, and B. Jacob, “A Benchmark to Understand the Role of Knowledge Graphs on Large Language Model’s Accuracy for Question Answering on Enterprise SQL Databases,” in *Proceedings of the 7th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences and Systems, (GRADES) and Network Data Analytics, (NDA), GRADES-NDA 2024*, 2024. doi: 10.1145/3661304.3661901.

[2] J. Sequeda, D. Allemang, and B. Jacob, “Knowledge Graphs as a source of trust for LLM-powered enterprise question answering,” *Journal of Web Semantics*, vol. 85, 2025, doi: 10.1016/j.websem.2024.100858.

[3] F. Lei *et al.*, “Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows,” *arXiv preprint arXiv:2411.07763*, Nov. 2024, doi: 10.48550/arXiv.2411.07763.

[4] H. Yuan, X. Tang, K. Chen, L. Shou, G. Chen, and H. Li, “COGSQL: A Cognitive Framework for Enhancing Large Language Models in Text-to-SQL Translation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025, pp. 25778–25786. doi: 10.1609/aaai.v39i24.34770.

[5] A.-M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates, “Modern natural language interfaces to databases: composing statistical parsing with semantic tractability,” in *Proceedings of the 20th International Conference on Computational Linguistics*, in COLING ’04. USA: Association for Computational Linguistics, 2004, pp. 141–es. doi: 10.3115/1220355.1220376.

[6] H. Bast and E. Haussmann, “More Accurate Question Answering on Freebase,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, in CIKM ’15. New York, NY, USA: Association

for Computing Machinery, 2015, pp. 1431–1440. doi: 10.1145/2806416.2806472.
[7] W. Zheng, H. Cheng, L. Zou, J. X. Yu, and K. Zhao, "Natural Language Question/Answering: Let Users Talk With The Knowledge Graph," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, in CIKM '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 217–226. doi: 10.1145/3132847.3132977.
[8] M. and C. H. Damljanovic Danica and Agatonovic, "Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-Based Lookup through the User Interaction," in *The Semantic Web: Research and Applications*, G. and H. E. and ten T. A. and S. H. and C. L. and T. T. Aroyo Lora and Antoniou, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 106–120. doi: 10.1007/978-3-642-13486-9_8.
[9] F. and S. C. and B. C. and Z. T. and B. H. and M. R. and D. C. and D. J. and M. T. and H. J. Song Dezhao and Schilder, "TR Discover: A Natural Language Interface for Querying and Analyzing Interlinked Datasets," in *The Semantic Web - ISWC 2015*, O. and S. E. and S. M. and d'Aquin M. and S. K. and G. P. and D. M. and H. J. and T. K. and S. S. Arenas Marcelo and Corcho, Ed., Cham: Springer International Publishing, 2015, pp. 21–37. doi: 10.1007/978-3-319-25010-62.
[10] V. Zhong, C. Xiong, and R. Socher, "Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning," *arXiv preprint arXiv:1709.00103*, Aug. 2017, doi: 10.48550/arXiv.1709.00103.
[11] W. Hwang, J. Yim, S. Park, and M. Seo, "A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization," *CoRR*, vol. abs/1902.01069, 2019, doi: 10.48550/arXiv.1902.01069.
[12] J. Guo *et al.*, "Towards complex text-to-SQL in cross-domain database with intermediate representation," in *Acl 2019 57th Annual Meeting of the Association for Computational Linguistics Proceedings of the Conference*, 2020, pp. 4524–4535. doi: 10.18653/v1/P19-1444.
[13] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2020, pp. 7567–7578. doi: 10.18653/v1/2020.acl-main.677.
[14] T. Yu *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, 2018, pp. 3911–3921. doi: 10.18653/v1/D18-1425.
[15] J. Li *et al.*, "Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs," in *Advances in Neural Information Processing Systems*, 2023. doi: 10.48550/arXiv.2305.03111.
[16] C.-H. Lee, O. Polozov, and M. Richardson, "KaggleDBQA: Realistic evaluation of text-to-SQL parsers," in *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 2021, pp. 2261–2273. doi: 10.18653/v1/2021.acl-long.176.
[17] C. Zhang *et al.*, "FinSQL: Model-Agnostic LLMs-based Text-to-SQL Framework for Financial Analysis," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Santiago: Association for Computing Machinery, Jun. 2024, pp. 93–105. doi: 10.1145/3626246.3653375.
[18] A. Marshan, A. N. Almutairi, A. Ioannou, D. Bell, M. Arzoky, and A. Monaghan, "MedT5SQL: a transformers-based large language model for text-to-SQL conversion in the healthcare domain," *Front. Big Data*, vol. 7, 2024, doi: 10.3389/fdata.2024.1371680.
[19] OpenAI *et al.*, "GPT-4 Technical Report," Mar. 2024, doi: 10.48550/arXiv.2303.08774.
[20] H. Touvron *et al.*, "LLaMA: Open and Efficient Foundation Language Models," Feb. 2023, doi: 10.48550/arXiv.2302.13971.
[21] B. Li, Y. Luo, C. Chai, G. Li, and N. Tang, "The Dawn of Natural Language to SQL: Are We Fully Ready?," *Proceedings of the VLDB Endowment*, vol. 17, no. 11, pp. 3318–3331, Jul. 2024, doi: 10.14778/3681954.3682003.
[22] D. Gao *et al.*, "Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation," *Proceedings of the VLDB Endowment*, vol. 17, no. 5, pp. 1132–1145, 2024, doi: 10.14778/3641204.3641221.
[23] Z. Li *et al.*, "PET-SQL: A Prompt-Enhanced Two-Round Refinement of Text-to-SQL with Cross-consistency," *arXiv preprint arXiv:2403.09732*, Mar. 2024, doi: 10.48550/arXiv.2403.09732.
[24] 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," Nov. 2023, doi: 10.48550/arXiv.2305.11853.
[25] M. Pourreza and D. Rafiei, "DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction," in *Advances in Neural Information Processing Systems*, 2023. doi: 10.5555/3666122.3667699.
[26] L. Nan *et al.*, "Enhancing Text-to-SQL Capabilities of Large Language Models: A Study on Prompt Design Strategies," 2023. doi: 10.18653/v1/2023.findings-emnlp.996.
[27] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen, "What Makes Good In-Context Examples for GPT-3?," in *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2022, pp. 100–114. doi: 10.18653/v1/2022.deelio-1.10.
[28] I. Trummer, "Generating Succinct Descriptions of Database Schemata for Cost-Efficient Prompting of Large Language Models," *Proceedings of the VLDB Endowment*, vol. 17, no. 11, pp. 3511–3523, 2024, doi: 10.14778/3681954.3682017.
[29] J. Byun, B. Kim, K.-A. Cha, and E. Lee, "Design and Implementation of an Interactive Question-Answering System with Retrieval-Augmented Generation for Personalized Databases," *Applied Sciences (Switzerland)*, vol. 14, no. 17, 2024, doi: 10.3390/app14177995.
[30] A. Mansurova, A. Mansurova, and A. Nugumanova, "QA-RAG: Exploring LLM Reliance on External Knowledge," *Big Data and Cognitive Computing*, vol. 8, no. 9, 2024, doi: 10.3390/bdcc8090115.
[31] P. Wang, T. Shi, and C. K. Reddy, "Text-to-SQL Generation for Question Answering on Electronic Medical Records," in *The Web Conference 2020 - Proceedings of the World Wide Web Conference, WWW 2020*, 2020, pp. 350–361. doi: 10.1145/3366423.3380120.
[32] H. Li, J. Zhang, C. Li, and H. Chen, "RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL," in *Proceedings of the 37th Aaai Conference on Artificial Intelligence Aaai 2023*, 2023, pp. 13067–13075. doi: 10.1609/aaai.v37i11.26535.
[33] Z. Zhan, E. Haihong, and M. Song, "Leveraging Large Language Model for Enhanced Text-to-SQL Parsing," *IEEE Access*, vol. 13, pp. 30497–30504, 2025, doi: 10.1109/ACCESS.2025.3540072.