QA-SQL: query-augmented SQL generation using few-shot prompting with data augmentation

Sokheang Chan[1], Vungsovanreach Kong[1], Anand Nayyar[2] and Tae-Kyung Kim[3]

[1] Big Data Department, Chungbuk National University, Cheongju-si, Chungcheongbuk-do, Republic of South Korea
[2] School of Computer Science, Duy Tan University, Da Nang, Vietnam
[3] Department of Management Information Systems, Chungbuk National University, Cheongju-si, Chungcheongbuk-do, Republic of South Korea

ABSTRACT

Natural Language to Structured Query Language (NL-to-SQL) models make it easier to retrieve data from structured databases by converting plain language queries into Structured Query Language (SQL) commands. However, current methods often struggle with complex database structures, unclear user questions, and generating efficient SQL queries. These issues make it hard to use them effectively in real-world, large-scale databases. To tackle these problems, a novel framework titled "Query-Augmented SQL Generation (QA-SQL)", is introduced that combines retrieval augmented generation (RAG) with data augmentation with advanced techniques to enhance the accuracy and efficiency of natural language to SQL (NL-to-SQL) generation. The proposed approach uses Schema Linking, RAG, and Dynamic Prompt Engineering along with few-shot learning and vector database retrieval, to enhance SQL query generation through query augmentation. To test and validate the proposed model, extensive experimentations were conducted on the BIg Bench for LaRge-scale Database Grounded Text-to-SQL Evaluation (BIRD) benchmark, which includes 1,534 natural language queries across various relational databases and the results highlighted a execution accuracy of 46.68% and a valid efficiency score of 87.66%, outperforming existing methods. The proposed research introduces a context-aware retrieval system that significantly enhances SQL accuracy, paving the way for more scalable and reliable NL-to-SQL solutions.

Subjects Artificial Intelligence, Databases
Keywords Text-to-SQL, Retrieval-augmented generation, Data augmentation, Large language models

Submitted 11 April 2025
Accepted 4 December 2025
Published 11 February 2026

Corresponding author
Tae-Kyung Kim,
kimtk@chungbuk.ac.kr

Academic editor
Ankit Vishnoi

Additional Information and Declarations can be found on page 19

DOI 10.7717/peerj-cs.3525

INTRODUCTION

Structured Query Language (SQL) remains the standard for querying relational databases, yet it poses a significant barrier to non-technical users due to its complexity. In business settings, this limitation creates obstacles to direct data access, slows down decision-making, and increases operational costs. Studies indicate that 65% of businesses struggle to access data from their Enterprise Resource Planning systems (*Wire, 2022*), and 68% of enterprise data remains unused, highlighting the severe accessibility challenges organizations face without intuitive query solutions (*Secoda, 2024*).

To bridge this gap, Natural Language to SQL (NL-to-SQL) models have been developed, allowing users to interact with databases using everyday language. These systems aim to eliminate technical barriers, making data access more efficient and user-friendly. However, despite advancements, current solutions often fall short in several key areas: accurately aligning user queries with database schema, generating precise SQL statements, and adapting to diverse database structures. Furthermore, while state-of-the-art NL-to-SQL models achieve high accuracy on benchmarks such as Spider (*Yu et al., 2019b*), their real-world performance can drop significantly, with accuracy ranging between 25% and 60% on complex datasets (*Zhang et al., 2023*).

Given the increasing demand for accessible data solutions, the global Natural Language Processing (NLP) market is projected to reach $9.71 billion by the end of 2024 (*Coherent Solutions, 2024*), emphasizing the growing role of intelligent query systems in modern business intelligence. Addressing these limitations is essential for unlocking the full potential of enterprise data, improving decision-making efficiency, and reducing the dependency on technical expertise.

Despite recent progress in NL-to-SQL techniques such as schema linking, self-correction, and prompting, current models still face challenges—particularly when handling complex schemas, multi-table queries, and context-aware reasoning. These limitations often result in incorrect or inefficient SQL outputs (*Zhu et al., 2024*; *Deng, Chen & Zhang, 2022*). Many existing models fail to properly match natural language inputs with database structures, resulting in incorrect queries, missing join conditions, and execution errors (*Hong et al., 2025*). This research explores how retrieval augmented generation (RAG) and data augmentation, combined with prompting in a few shots, can strengthen the accuracy of SQL generation, schema alignment, and adaptability across various relational databases. Specifically, this study seeks to answer the following questions: (1) How does synthetic prompt generation impact schema comprehension? (2) To what extent does data augmentation mitigate hallucination errors in SQL generation? (3) How effectively does QA-SQL generalize across varying and unseen database schemas?

At the same time, *Lewis et al. (2020)* had shown great potential of RAG in increasing language model performance by retrieving relevant external information during query generation. Although RAG has traditionally been used to incorporate historical patterns and domain knowledge to improve natural language understanding, its application in text-to-SQL systems remains largely untapped. This research aims to explore how RAG can be used to enhance NL-to-SQL systems, offering a more robust and scalable solution for real-world database querying.

In this study, the capabilities of RAG are extended by integrating context-aware retrieval and data augmentation to dynamically generate contextually relevant few-shot prompts. Instead of relying on static historical query logs, RAG is used to retrieve synthetic query examples (SQL-NL pairs), enabling more accurate SQL synthesis and improved adaptability to diverse database schema. Existing text-to-SQL systems often struggle with schema alignment, complex query generation, and context-aware reasoning, leading to inaccurate SQL translations. To address these challenges "QA-SQL", a scheme aware retrieval and data augmentation model is proposed that improves query synthesis.

The objectives of the article are:

- To introduce a query-augmented approach that dynamically generates synthetic few-shot prompts based on the database schema, eliminating the need for pre-existing query examples;
- To develop a retrieval mechanism that selects contextually relevant SQL-NL pairs, ensuring improved SQL synthesis and query alignment;
- To evaluate the proposed system on the BIRD benchmark, using different Large Language Models (LLMs) to demonstrate their capability in Text-to-SQL tasks;
- And, to analyze the impact of synthetic query generation in mitigating hallucination errors and improving SQL synthesis accuracy in various database environments;

In this research, it is hypothesized that synthetic query retrieval combined with few-shot prompting will enhance SQL generation accuracy by improving query synthesis and reducing query errors. The experiments aim to validate whether synthetic retrieval prompts provide comparable or superior contextual support compared to traditional retrieval-based models. Through rigorous experimental evaluation, the main objective is to demonstrate how this integrated approach can significantly advance the state-of-the-art in natural language database interfaces.

BACKGROUND

The evolution of NL-to-SQL systems

Text-to-SQL models have gained significant traction in recent years, aiming to bridge the gap between human language understanding and structured database querying. Early systems relied on rule-based approaches, requiring extensive manual feature engineering to map user queries to SQL structures. With the advent of deep learning and large language models (LLMs), modern text-to-SQL systems have achieved remarkable improvements by learning query patterns, database relationships, and schema structures from large-scale datasets.

Text-to-SQL, a specialized domain of semantic parsing, has demonstrated substantial improvements in performance metrics, consistently exceeding predecessor methodologies in recent BIRD benchmarks (*Li et al., 2023*). This advancement has gained significant traction in industry applications, with leading data visualization platforms such as Tableau and Power BI implementing natural language interfaces to facilitate dashboard construction. These commercial applications primarily handle relatively simple structured queries, driving high industry demand for text-to-SQL capabilities across various sectors, including *Salesforce (2020)* and *Uber (2025)*.

Despite significant advances, text-to-SQL models continue to face persistent limitations, especially when applied to real-world databases. First, schema alignment errors often occur due to ambiguous column names, missing foreign keys, or insufficient linking mechanisms (*Qin et al., 2022*; *Deng, Chen & Zhang, 2022*). Second, model performance degrades significantly when transitioning from academic benchmarks to large or complex databases (*Li et al., 2023*; *Rajkumar, Li & Bahdanau, 2022*; *Zhang et al., 2023*). Third, LLM-based systems can hallucinate SQL structures, produce inefficient queries, or fail to generalize in multi-table or nested SQL scenarios (*Zhu et al., 2024*; *Coelho et al., 2024*). These challenges highlight the need for schema-aware retrieval, dynamic few-shot prompting, and synthetic data augmentation key components of the QA-SQL framework presented in this study.

Although deep learning-based text-to-SQL models have achieved strong performance in structured query generation, they still face persistent limitations such as schema misalignment, multi-table joins, and weak contextual reasoning across domains (*Zhang et al., 2023*; *Rossi et al., 2024*). These challenges persist because real-world database schemas are highly variable and often contain ambiguities in table relationships and column naming conventions. For example, benchmarks like BIRD emphasize syntactic correctness but do not evaluate adaptability to unseen schema structures, limiting their real-world applicability. Despite its limitations, BIRD remains one of the most widely used and standardized benchmarks for evaluating text-to-SQL models. Its large-scale, diverse set of real-world database schemas provides a strong basis for controlled comparison across different methods. Using BIRD allows us to evaluate our framework under the same conditions as prior work, while our future work aims to extend to more complex and dynamic database scenarios. Addressing these issues requires retrieval mechanisms that incorporate schema semantics and database context, which motivates the RAG-based strategies explored in this work.

The role of RAG in SQL querying

In the context of NL-to-SQL, (*Guo et al., 2023a*) enhanced query accuracy by retrieving schema-aligned contextual knowledge and structured augmentations that improve SQL synthesis. Rather than relying on static schema retrieval, which may be incomplete or ambiguous, recent studies had proposed data augmentation-driven few-shot prompting to construct synthetic but contextually relevant schema examples. By storing augmented prompts in a vector database, the retrieval process dynamically adapts to the given database schema, enabling more precise SQL generation.

The effectiveness of RAG in SQL querying depends on its ability to retrieve contextually relevant synthetic query examples that align with database structures, relationships, and domain-specific rules. Using a vector database populated with LLM-driven data augmentation, synthetic few-shot prompts can be retrieved to dynamically adapt to the given NL query. Unlike static schema-base retrieval, which may be incomplete or ambiguous, RAG enhances SQL synthesis by retrieving synthetic, query-aligned examples to improve accuracy. This adaptive retrieval process is shown to help handle complex multi-table queries, generate correct joins, and reduce SQL syntax errors, leading to more reliable NL-to-SQL transformation.

RELATED WORKS

Data augmentation

In *Dyk & Meng (2001)*, data augmentation was a technique where new data points are artificially created from existing data by applying various transformations. This method is widely used in machine learning and NLP to enhance model generalization, reduce over-

Table 1 **The statistics and analysis of well-known datasets of text-to-SQL ordered by release time.**

Benchmark dataset	#Example	#DB	#Table/DB	#Row/DB	Characteristics
BIRD (*Li et al., 2023*)	12,751	95	7.3	549 K	CD, KA
KaggleDBQA (*Lee, Polozov & Richardson, 2021*)	272	8	2.3	280 K	CD
DuSQL (*Wang et al., 2020*)	23,797	200	4.1	–	CD, CL
SQUALL (*Shi et al., 2020*)	11,468	1,679	1	–	KA
CoSQL (*Yu et al., 2019a*)	15,598	200	–	–	CD, CDep
Spider (*Shi et al., 2020*)	10,181	200	5.1	2 K	CD
WikiSQL (*Zhong, Xiong & Socher, 2017*)	80,654	26,521	1	17	CD

fitting, and improve performance on real-world datasets. Data augmentation has emerged as a pivotal technique for enhancing model performance by diversifying training examples without the need for additional data collection (*Ding et al., 2024*).

Benchmarks

The BIRD (BIg Bench for LaRge-scale Database Grounded text-to-SQL Evaluation) benchmark, introduced by *Li et al. (2023)*, was designed to advance text-to-SQL generalization in realistic application settings. It has been widely adopted for evaluating model performance on complex, large-scale databases spanning multiple domains. In the proposed work, BIRD is used as primary benchmark because it provides diverse relational database schema and complex query structures, making it well suited for evaluating query-augmented retrieval and few-shot prompting in NL-to-SQL tasks.

Table 1 summarizes widely used text-to-SQL datasets, capturing their diversity in terms of example volume, schema complexity, and domain characteristics. These benchmarks vary in the number of databases, table structures, and data types, which influence the generalization performance of text-to-SQL models. The benchmark dataset indicates that the dataset is designed with a corresponding database, while post-annotated datasets involve annotating new components within existing datasets and databases rather than releasing a new database (*Hong et al., 2024*). These are typically constructed by first generating SQL queries using an automated model, then manually verifying or correcting them to ensure quality and correctness. The dataset characteristics are abbreviated as follows: CD (Cross-Domain), KA (Knowledge-Augmented), CL (Cross-Lingual), CDep (Context-Dependent). These terms describe the diversity and complexity of the datasets used in text-to-SQL benchmarks.

Text-to-SQL and few-shot prompting

Sequence-to-sequence worked well for transforming text into code, including text-to-SQL. The main approach was to combine the user's question and database structure in the first step, then use a second step to predict the target SQL code (*Sutskever, Vinyals & Le, 2014*).

The latest development in this progression was the use of LLMs under zero-shot and few-shot prompting (*Rajkumar, Li & Bahdanau, 2022*; *Liu et al., 2023*). It was shown that LLMs provided strong baselines using only a few demonstrations and no fine-tuning (*Brown et al., 2020*; *Chen et al., 2021*; *Liu et al., 2021*). According to those studies,

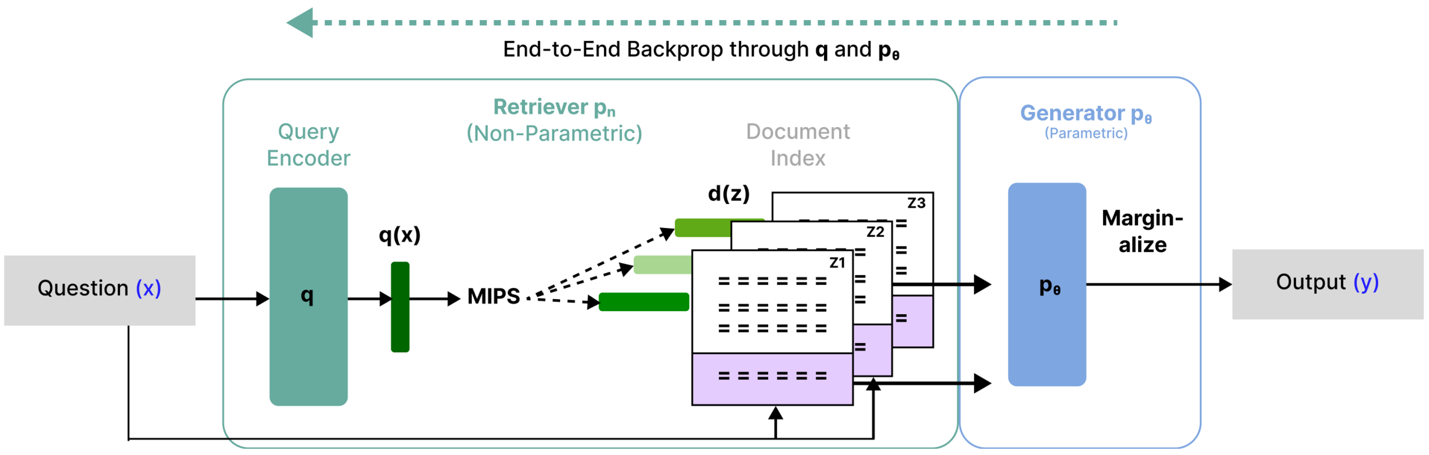
End-to-End Backprop through q and p_θ
Query Encoder
Retriever p_η (Non-Parametric)
Document Index
Generator p_θ (Parametric)
Question (x)
q
q(x)
MIPS
d(z)
z3
z2
z1
p_θ
Margin-alize
Output (y)

Figure 1 **An overview of RAG.** Full-size DOI: 10.7717/peerj-cs.3525/fig-1

pre-trained and fine-tuned models had a high potential on text-to-SQL evaluation on Spider benchmark (*Yu et al., 2019b*) by using only zero-shot prompting. LLMs conducted a remarkable result with prompting by just using one or a few examples. Prompting played a vital role in guiding LLMs effectively with minimal examples in processing structured data task such understanding database schema, Table-to-Text Generation, and table reasoning (*Guo et al., 2023b*; *Chen, 2022*).

Coelho et al. (2024) proposed a method to enhance the accuracy of LLM-based text-to-SQL systems on real-world relational database (RW-RDB) using RAG framework. The approach involved two steps. Using the RW-RDB schema, the first step generated a synthetic dataset E of pairs (Q_N, Q_S), where Q_N was an NL question and Q_S was the corresponding SQL translation. The core contribution of the article was an algorithm that implemented this first step. The proposed approach similarly adopts this idea by extending the first-step generation of schema-aligned synthetic (Q_N, Q_S) pairs, which are then used for context retrieval during inference to enhance SQL generation. Given an input NL question Q_I, the second step retrieved pairs (Q_N, Q_S) from E based on the similarity of Q_I and Q_N, and as context in prompts to the LLM to enhance SQL generation accuracy.

Retrieval-augmented generation (RAG)

Lewis et al. (2020) introduced RAG, a novel approach that addressed the limitations of traditional language models in knowledge-intensive tasks. This approach helped to improve the accuracy and comprehensiveness of responses from external knowledge. In this study, it has demonstrated significant improvements in NL processing tasks by incorporating retrieval-based mechanisms into generative models. In Fig. 1 shows the combination of a pre-trained retriever with a pre-trained encoder-decoder (Generator) and fine-tune end-to-end. In the article, retriever was used to find the top-k most relevant documents of all documents z_i to make the final prediction y.

Gao et al. (2023) explored the evolution of RAG framework, consisting of three major stages of development. Naive RAG implementation was operated through a simple process of splitting documents into chunks, indexing them, retrieving relevant information, and generating responses. Advanced RAG built upon this foundation by implementing sophisticated retrieval optimization techniques, including pre-retrieval query refinement, enhanced indexing strategies, and post-retrieval methods such as re-ranking and context compression to ensure the most relevant and concise information is prioritized in prompts. Modular RAG further improves system performance by refining specific components. In particular, it incorporates similarity-based search features and specialized retriever fine-tuning methods (*Nascimento et al., 2025*).

Guo et al. (2023a) proposed a retrieval-augmented prompting approach within an LLM-based text-to-SQL framework, introducing sample-aware prompting and a dynamic revision chain. They proposed two strategies to assist with retrieval. The authors initially used LLMs to simplify the original user questions, unifying the syntax and thus clarifying the intentions of the users. Next, the method incorporated a dynamic revision chain to generate executable and accurate SQL, which iteratively adapted fine-grained feedback from the previously generated SQL queries without human intervention.

PROPOSED METHODOLOGY: QA-SQL

This section discusses the overall architecture of QA-SQL, followed by subsections with a detailed breakdown of each component.

System model

The proposed QA-SQL framework is designed to bridge the gap between natural language queries and structured database retrieval by integrating Schema Linking, Retrieval-Augmentation, and Prompt Engineering. Unlike conventional NL-to-SQL approaches that rely on static query examples, QA-SQL dynamically retrieves SQL-NL pairs to improve SQL synthesis accuracy. The main aim to evaluate QA-SQL using a complex and diverse database benchmark that reflects real-world variability in schema design (*Li et al., 2023*).

This approach enhances SQL query generation by improving schema understanding and contextual retrieval of relevant data. The system model consists of three key stages: Schema Linking, Retrieval-Augmentation, and Prompt Engineering.

Given a natural language query Q_{NL}, the QA-SQL system model can be represented as:

$$Q_{SQL} = F(Q_{NL}, C, D)$$

where Q_{NL} represents the user's natural language question. C denotes the columns and tables linked to the retrieved schema. D consists of SQL-NL pairs retrieved by using a vector similarity search. F is the function that constructs the SQL query using prompt engineering.

System architecture

The architecture of QA-SQL is designed to efficiently process NL queries, retrieve query augmented example, and generate high-accuracy SQL queries. It consists of three primary components, each interacting through a structured retrieval and query synthesis pipeline.

Within the system component, Schema Linking is used to identify and extract relevant columns from a large database schema while filtering out unrelated columns and enabling

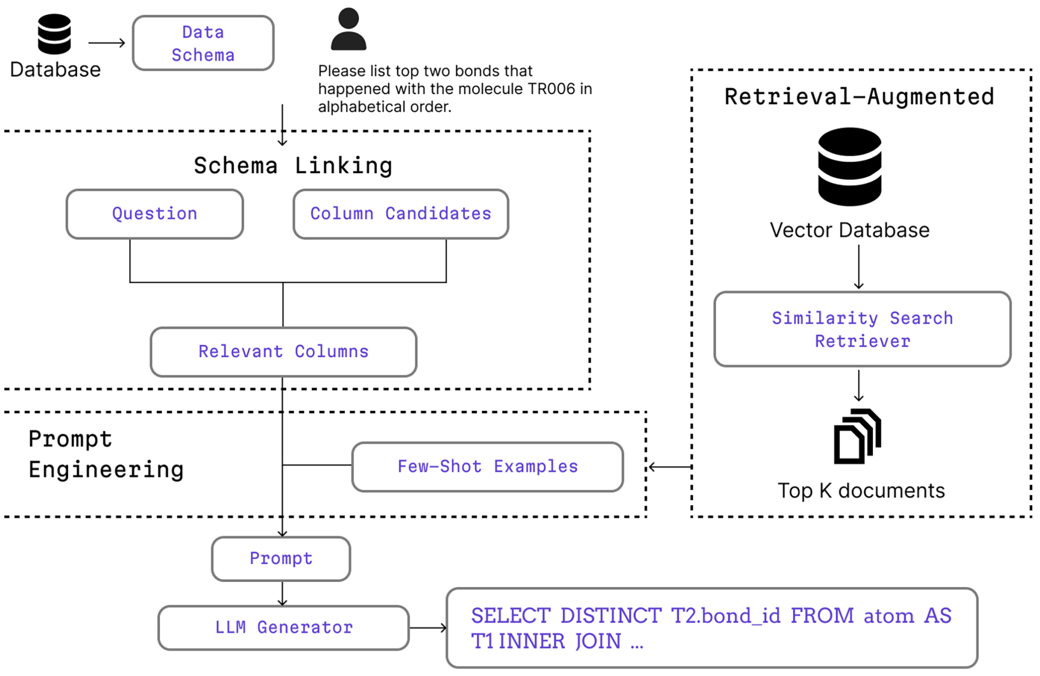

Figure 2 A high-level workflow of the proposed QA-SQL framework, showing how schema linking, retrieval-augmentation, and prompt engineering interact to generate SQL queries. The Retrieval-Augmented module accesses a vector database populated by the synthetic data generation process. Image Source Credit: Iconify, licensed under Apache 2.0, MIT, CC BY 4.0.
Full-size DOI: 10.7717/peerj-cs.3525/fig-2

more focused and efficient data operations. Retrieval-Augmentation is used to retrieve the most relevant data from the vector database based on similarity scores from vector search. Both types of context information are then fed into the Prompt Engineering module as a few shot prompt to learn from and to improve its performance on similar context before generating a response. Figure 2 shows the process begins with a user query and database schema, followed by schema linking to identify relevant columns. Retrieved few-shot examples assist in query formulation before generating SQL using an LLM.

Generating synthetic dataset

In this section, data augmentation technique is applied for the generating synthetic dataset, enhancing schema linking and improving generalization to unseen database structures. Specifically, it generates synthetic NL-SQL pairs from the database schema, which are embedded into vector database to support few-shot prompting. These pairs serve as additional training examples to enhance the model's adaptability to diverse schema. Only 270 pairs are generated as synthetic dataset using all tables and columns in the database, following the triplet generation algorithm shown in Algorithm 1.

Figure 3 illustrates the end-to-end process of generating synthetic SQL-NL pairs using a large language model (LLM) and embedding them into a vector database for use in retrieval-augmented prompting. This synthetic dataset is an essential support component of the Retrieval-Augmentation module illustrated in Fig. 2. The resulting SQL-NL pairs are

Algorithm 1 **Generating synthetic (Q_N, Q_S, Q_E) dataset for text-to-SQL.**

```
Data: Database schema S, pre-trained language model LM
Result: Synthetic dataset (Q_N, Q_S, Q_E) for training text-to-SQL models.
 1: function GENERATESCHEMA (DB)
 2:     $T \leftarrow$ SelectTables($DB$)
 3:     $C \leftarrow$ SelectColumns($T, DB$)
 4:     $S \leftarrow$ ConstructSchema($T, C$)
 5:     return $S$
 6: function GENERATEDATASET (S, LM)
 7:     $Q_E \leftarrow$ ExtractEvidence($LM, S, Q_N$)
 8:     $Q_N \leftarrow$ CreateQuestion($LM, S$)
 9:     $Q_S \leftarrow$ GenerateSQL($LM, S, Q_N, Q_E$)
10:     return ($\mathbf{Q}_N, \mathbf{Q}_S, \mathbf{Q}_E$)
```

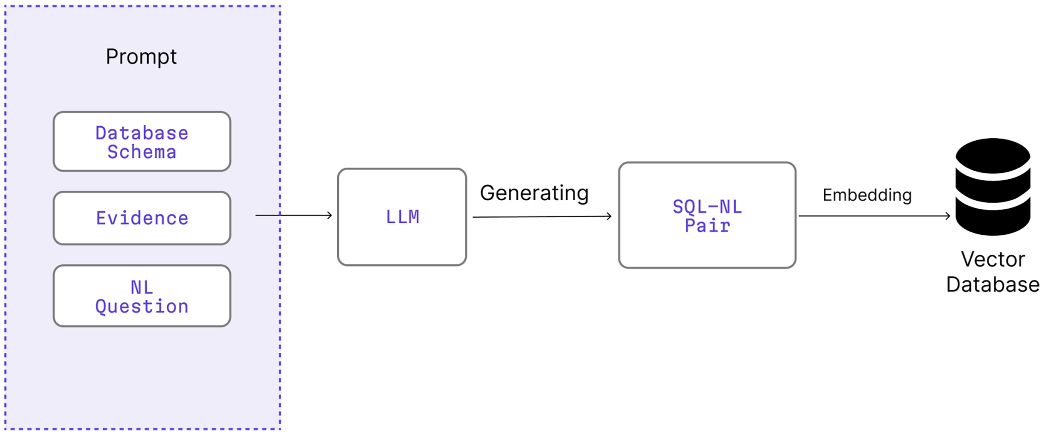

Figure 3 **Process for generating synthetic SQL–NL pairs using LLMs and embedding them into a vector database that supports the retrieval-augmented stage of QA-SQL.** Image source credit: Iconify, licensed under Apache 2.0, MIT, CC BY 4.0. Full-size DOI: 10.7717/peerj-cs.3525/fig-3

embedded into a vector database, which is later queried during inference to retrieve relevant few-shot examples for prompt construction.

Component definition: In Fig. 3, BIRD is used as a basis as a basis for generating query augmented samples. By extracting relational structures from BIRD's diverse databases, the data augmentation strategy ensures that synthetic examples reflect real-world schema variability, enhancing generalization across unseen databases.

The generating synthetic dataset begins with GENERATESCHEMA (DB) in Algorithm 1 by filtering the database to select only the relevant tables using the T component. C identifies and extracts the necessary columns from the tables selected previously. S integrates the selected tables and columns to form a structured schema, providing a clear blueprint of the database's design.

The GENERATEDATASET (S, LM) function on Line 6 consists of three components. First, EXTRACTEVIDENCE Q_E provides the reasoning behind each NL question by extracting relevant details from the database schema. CREATEQUESTION Q_N is the generated NL question that simulate realistic user inquiries, covering with three difficulty levels: simple, moderate, and challenging. For every question in Q_N, a corresponding SQL Q_S is formulated based on the database schema, ensuring the queries are both syntactically correct and semantically aligned with the schema's structure and constraints.

To ensure that synthetic dataset being high-quality and diverse, a systematic generation and validation strategy was employed. The goal was to create a balanced collection of queries with varying complexity. To achieve this, the LLM was prompted to generate examples across four distinct categories: aggregation, comparison, ranking, and complex reasoning. The final dataset size of 270 samples is a result of this structured approach across the various databases equally in the BIRD benchmark.

We selected 270 as the optimal dataset size after empirical testing. While larger sets were initially considered, we observed that they introduced redundant or noisy samples and led to slower retrieval times during inference without significant performance gains. Therefore, 270 samples provided the best balance between retrieval efficiency and coverage diversity.

Furthermore, to automate quality control and establish clear acceptance criteria, every generated SQL query (Q_S) underwent an automated executability check. Each query was executed against its corresponding database, and only those that ran without errors were accepted into the final dataset. This entire pipeline, from targeted query generation to executability check, is fully automated, requiring no human intervention and confirming the scalability of our approach while ensuring all samples are of high quality.

Data generation process: In the $\textsc{GenerateSchema}\ (DB)$, the database schema S is constructed by first selecting the relevant tables T from the database, followed by extracting their corresponding columns C. Finally, the schema S is formed by combining T and C, ensuring a well-structured representation of the database for downstream text-to-SQL processing. This process iterates through all tables in the database, ensuring that at least one NL question is generated for each table.

Q_E plays a crucial role in helping LM generate SQL queries by providing contextual and structural information about the database. Given the NL question Q_N, "*Under whose administration is the school with the highest number of students scoring 1,500 or more on the SAT? Indicate their full names*," the function extracts schema-aware evidence Q_E through $\textsc{ExtractEvidence}\ (LM, S, Q_N)$. This step incorporates metadata, column relationships, constraints, and semantic hints that refine the query generation process.

To extract $\boldsymbol{Q_E}$, LM first identifies key attributes by aligning the terms in $\boldsymbol{Q_N}$ with column names in the database schema $\boldsymbol{S}$. It recognizes that "*students scoring 1,500 or more on the SAT*" corresponds to the `NumGE1500` column in the `satscores` table and the phrase of "*school with the highest number*" indicate the need for aggregation and ranking, prompting the model to identify `CDSCode` from `schools` referencing to `satscores`. To determine the name of the school administrator, the LM understands the term of `AdmFName1` and `AdmLName1` and need of concatenation between these key terms in `schools` table.

The synthetic dataset is generated using a design prompt that incorporates three core components: Q_N, Q_S, and Q_E. The NL question Q_N is created based on the database schema S, which is retrieved using the GENERATESCHEMA (DB) prompt. The SQL query Q_S is generated to match Q_N and S, using the top five rows of each table that ensure table names, column names, and value are included. These steps are performed with the GENERATEDATASET (S, LM) prompt, powered by a the pre-trained language model LM. The GENERATEDATASET (S, LM) prompt includes specific instructions to incorporate the database schema S, NL question Q_S, corresponding SQL query Q_S and generating evidence Q_E.

Example: Given a database schema S containing school information Database Schema S:

```
CREATE TABLE students (
      student_id INT PRIMARY KEY,
      first_name VARCHAR(50),
      last_name VARCHAR(50),
      school_id VARCHAR(10)
  );

  CREATE TABLE satscores (
      student_id INT,
      cds VARCHAR(10),
      NumGE1500 INT,
      FOREIGN KEY (student_id) REFERENCES students
      (student_id),
      FOREIGN KEY (cds) REFERENCES schools(CDSCode)
  );

  CREATE TABLE schools (
      CDSCode VARCHAR(10) PRIMARY KEY,
      school_name VARCHAR(100),
      AdmFName1 VARCHAR(50),
      AdmLName1 VARCHAR(50)
  );
```

The LM generates an NL question Q_N:
"*Under whose administration is the school with the highest number of students scoring 1,500 or more on the SAT? Indicate their full names.*"
Generated Evidence Q_E:
"SAT Scores above 1,500 refers to `NumGE1500` column in the satscores table."
"The school with the highest number refers to the one with the largest number of matched records in satscores."
Generated SQL Query Q_S:

```
SELECT T2.AdmFName1, T2.AdmLName1
FROM satscores AS T1
INNER JOIN schools AS T2 ON T1.cds = T2.CDSCode
```

```
ORDER BY T1.NumGE1500 DESC
LIMIT 1;
```

This highlights the challenge of generating synthetic (Q_N, Q_S, Q_E) dataset, ensuring dynamic and adaptable knowledge return to enhance LM generalization in context-aware SQL query generation.

Schema linking

Schema linking is a crucial step in identifying relevant database attributes that correspond to entities following the user's question. In our Schema Linking implementation, we parse the NL question and extracting the potential column candidates using LLM-Based Column Extraction from given user's query and schema which are represented by mathematic equation. In the below equation, for a given NL query Q_{NL}, the schema linking function f_s maps Q_{NL} to the most relevant schema columns

$$C = f_s(Q_{NL}, S)$$

where S represents the full set of database schema columns, and C is the filtered subset used for SQL generation.

Retrieval-augmentation

RAG improves the performance of the system by incorporating external contextual knowledge from vector databases. This ensures that relevant schema-specific or domain-specific knowledge is integrated into the SQL generation process. The NL question Q_{NL} is first converted into a vector embedding V_Q using OpenAI's text-embedding-ada-002 model. A similarity search function f_r retrieves the top-k most relevant SQL-NL pair D from a precomputed vector database V_D, structured using ChromaDB:

$$D = f_r(V_Q, V_D)$$

where V_D represents the vector embeddings of all documents stored in ChromaDB, an open-source vector database designed for efficient similarity search over dense embeddings. ChromaDB enables fast retrieval of top-k relevant documents and is widely used in retrieval-augmented generation (RAG) applications. We use cosine similarity to measure the closeness between the query embedding V_Q and the stored document embeddings V_D, ensuring that the retrieved examples are contextually aligned with the NL query:

$$\text{cosine\_sim}\,(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

where A and B represent the normalized embedding vectors of the query and the stored documents, respectively. This similarity measure indicates the semantic relevance between the query and the stored document embeddings.

Vector similarity search: By enriching the SQL generation process with synthetic query retrieval, QA-SQL ensures that queries requiring additional context are properly structured and aligned with user intent.

1. The system receives an NL question Q_{NL} and converts it into a dense vector representation V_D and retrieves the most relevant synthetic query examples using OpenAI's text-embedding-ada-002 model.
2. A cosine similarity search is performed in the Chroma vector database V_D
3. The top-5 most relevant SQL-NL pairs are retrieved based on similarity ranking.
4. The selected contextual examples D.

Prompt engineering

To enhance the SQL generation, dynamic few-shot prompting mechanism was used. Instead of relying on static example, the most relevant examples from D and C into dynamic prompt component for contextual relevance was retrieved, which improved the accuracy of column extraction and overall SQL synthesis.

Unlike traditional static few-shot prompting, where predefined examples may not align with new queries, our approach dynamically retrieved examples contextually that aligned with the database structure, reducing schema mismatch errors and improving SQL accuracy.

Dynamic few-shot prompting: Dynamic few-shot prompting mechanism was used to ensure that relevant contexts are retrieved for SQL synthetsis.

1. Receives as input an NL question Q_{NL}.
2. Retrieve the top-k most relevant examples from the vector database D, where stored NL questions Q_{NL} are ranked using cosine similarity.
3. Retrieve schema-aligned columns and table references C from Schema Linking, ensuring that retrieved examples align with the database structure.
4. Construct a structured prompt using Q_{NL}, C, D, and task-specific instructions.
5. Return the final structured prompt that use for SQL generation Q_{SQL}, ensuring that the query is contextually accurate and schema-aware.

EXPERIMENTAL SETUP

Technologies involved

To implement the proposed QA-SQL framework, the system was developed using Python and the Langchain framework to build the RAG pipeline, manage component chaining, and integrate dynamic few-shot prompting. The SQLite engine was used to host the relational databases for SQL execution and testing. For LLMs, we used API access to models from OpenAI (GPT-4o), Anthropic (Claude 3.7 Sonnet), and Groq (LLaMA 3 70B Versatile), each configured with a temperature setting of 0.7 to maintain response consistency while preserving a degree of generation flexibility. This environment allows scalable experimentation across diverse database schema and query structures.

Dataset

To evaluate the performance of proposed QA-SQL model, BIRD benchmark is used, which includes 1,533 NL questions paired with ground-truth SQL queries across multiple

Table 2 **Distribution of test questions across difficulty levels in the BIRD benchmark.**

Difficulty level	Number of questions
Simple	925
Moderate	464
Challenging	145

relational databases. Each database contains multiple tables with well-defined schema, attributes, and relationships. The Mini-Dev dataset, a curated subset of the BIRD benchmark, is used to test our method. It is designed to facilitate efficient and cost-effective development cycles, especially for testing and refining SQL query generation models. Table 2 shows that the NL question set, being classified as simple, medium, or complex based on the complexity of its corresponding ground-truth SQL query.

Evaluation metrics

In this evaluation procedure, two evaluation methods are used from BIRD, execution accuracy (EX) and valid efficiency score (VES) to evaluate text-to-SQL parsers confronted with large real-world database values (*Li et al., 2023*).

Execution accuracy (EX) measures the proportion of test cases where the execution results of both the predicted and ground-truth SQL queries match exactly, normalized by the total number of queries N (*Qin et al., 2022*). Let V_n represent the result set from the n^{th} ground-truth SQL Y_n, and $\hat{V}_n$ the result set from the corresponding predicted SQL. The EX score is computed as:

$$EX = \frac{\sum_{n=1}^{N} \quad (V_n = \hat{V}_n)}{N},$$

where $1(\cdot)$ is an indicator function:

$$1(V, \hat{V}) = \begin{cases} 1, & \text{if } V = \hat{V}, \\ 0, & \text{otherwise.} \end{cases}$$

The valid efficiency score (VES) evaluates the efficiency of valid predicted SQL queries—those whose execution results match the ground truth. Queries producing mismatched results are marked as invalid since they cannot satisfy user requirements, regardless of execution time. VES combines both accuracy and efficiency by incorporating a relative efficiency measure:

$$VES = \frac{\sum_{n=1}^{N} \quad (V_n, \hat{V}_n) \cdot \mathbf{R}(Y_n, \hat{Y}_n)}{N}$$

where $R(Y_n, \hat{Y}_n)$ represents the relative efficiency of executing the predicted SQL compared to the ground truth. In BIRD, runtime in milliseconds is primarily used (*Li et al., 2023*).

Table 3 **Accuracy of QA-SQL models across different query difficulty levels, evaluated using EX.** Higher accuracy indicates better SQL generation performance.

Model	EX performance of different models %			
	Simple	Moderate	Challenging	Total
Claude 3.7 Sonnet	48.43	30.82	26.90	41.07
GPT-4o	55.35	32.97	23.45	45.57
Groq LLaMA 3 70B Versatile	54.05	36.85	31.03	46.68

The relative efficiency R is calculated as:

$$\mathbf{R}(Y_n, \hat{Y}_n) = \sqrt{\frac{\mathbb{E}(Y_n)}{\mathbb{E}}(\hat{Y}_n)}.$$

$\mathbb{E}(Y_n)$, is termed as the total execution time (latency) in milliseconds for each SQL query.

RESULTS

Execution accuracy

To evaluate the effectiveness of our QA-SQL model, its execution accuracy is assessed which measures the percentage of generated SQL queries that produce the same results as their ground-truth counterparts when executed. This evaluation provides insights into how well our model translates natural language (NL) questions into executable SQL queries. To better understand the QA-SQL model's effectiveness in handling various query complexities, we evaluate its execution accuracy across simple, moderate, and challenging queries. Table 3 summarizes the comparative EX performance across different LLMs.

As a first step, three LLM backbones (Claude 3.7 Sonnet, GPT-4o, and Groq LLaMA 3 70B Versatile) was evaluated within the proposed QA-SQL framework on BIRD queries (Table 3). This comparison is used solely to determine the most suitable LLM for downstream evaluation. Groq LLaMA 3 70B, achieving the highest EX (46.68%), is selected as the backbone model for all subsequent analysis.

Valid efficiency score

While execution accuracy evaluates the correctness of generated SQL queries, efficiency metrics provide insights into how well these queries perform when executed. Therefore, we assess the VES to measure the computational efficiency of the generated SQLs. In addition to execution accuracy, we assess the VES to measure the efficiency of correctly executed SQL queries. The VES metric evaluates how well the generated SQL queries perform compared to ground-truth SQL queries in terms of execution speed and resource efficiency. To complement accuracy analysis, we assess the computational efficiency of each model's generated SQL queries using the VES metric. Table 4 presents these results, highlighting differences in execution optimization across models. As shown in Table 4,

Table 4 **Evaluation of SQL execution efficiency across different models using the VES.** The columns VES-1, VES-2, and VES-3 show the scores from three independent experimental runs, with the final column presenting the average (Avg) score. This approach is used to ensure reliable results and account for variations in execution time. Higher scores indicate more efficient SQL query execution in terms of computational speed and resource utilization.

Model	**VES of SQL execution performance %**			
	VES-1	**VES-2**	**VES-3**	**Avg**
Claude 3.7 Sonnet	49.36	33.59	27.03	42.48
GPT-4o	57.32	35.25	23.97	47.49
Groq LLaMA 3 70B Versatile	88.65	85.28	88.93	87.66

Groq LLaMA 3 70B also demonstrates the highest VES (87.66%), further justifying its selection based on both accuracy and efficiency.

While one might intuitively expect the ground-truth SQLs to be the most efficient, in practice, predicted SQLs can exhibit higher efficiency scores. This may occur when the model learns to generate semantically equivalent but structurally simpler or more optimized SQLs (*e.g.*, avoiding unnecessary joins or subqueries). Such differences can lead to shorter execution times even when query outputs remain correct, thereby resulting in VES is greater than EX in some cases.

Table 5 presents a direct comparison of the entire QA-SQL framework (as described in Fig. 2) against recent state-of-the-art text-to-SQL baselines on the BIRD benchmark. ChatGPT, using vanilla prompting, achieves 46.35% EX (*Li et al., 2023*), while DIN-SQL (*Pourreza & Rafiei, 2023*), which incorporates self-correction and multi-step schema linking, reaches 55.9% EX and 59.44% VES. In contrast, our QA-SQL framework achieves 46.68% EX and 87.66% VES, as shown in Table 5. The results are discussed in detail as follows:

Table 5 summarizes the comparative performance between QA-SQL and recent Text-to-SQL frameworks, including DIN-SQL (*Pourreza & Rafiei, 2023*), CHASE-SQL (*Pourreza et al., 2024*), ROUTE (*Qin et al., 2024*), and OpenSearch-SQL (*Xie et al., 2025*). DIN-SQL (*Pourreza & Rafiei, 2023*) was selected as the principal benchmark because it is methodologically closest to QA-SQL, employing schema-aware reasoning and few-shot prompting without fine-tuning, which allows a fair, model-agnostic comparison. In contrast, QA-SQL expands this structure with a retrieval-augmented few-shot entails with synthetic data augmentation to dynamically retrieve query-aligned examples without the need for manual curation.

Recent study including CHASE-SQL (*Pourreza et al., 2024*), ROUTE (*Qin et al., 2024*), and OpenSearch-SQL (*Xie et al., 2025*) achieve higher execution accuracies using fine-tuned, multi-agent, or multitask supervision approaches. CHASE-SQL achieved state-of-the-art accuracy using multi-path chain-of-thought reasoning, in combination with a fine-tuned selection module, ROUTE demonstrates improved consistency results through using multitask training and uncertainty-based ranking, while OpenSearch-SQL uses

Table 5 **Comparison of QA-SQL with prior text-to-SQL models on BIRD benchmark in terms of execution accuracy (EX) and efficiency (VES).**

Model	Year	Dev-EX (%)	VES (%)	Notes
ChatGPT (*Li et al., 2023*)	2023	46.35	–	Zero-shot baseline
DIN-SQL + GPT-4 (*Pourreza & Rafiei, 2023*)	2023	50.7	58.8	Self-correction + schema linking
CHASE-SQL + Gemini 1.5 (*Pourreza et al., 2024*)	2024	73.01	–	Multi-path reasoning with chain-of-thought and fine-tuned selection agent
ROUTE (*Qin et al., 2024*)	2025	60.9	65.2	Multitask supervised fine-tuning with schema linking, noise correction, and continuation writing
OpenSearch-SQL + GPT-4o (*Xie et al., 2025*)	2025	69.3	69.36	Dynamic few-shot generation with multi-agent consistency alignment
QA-SQL (Groq LLaMA 3 70B)	2025	46.68	87.66	Schema-aware retrieval + synthetic data

GPT-4o with multi-agent consistency alignment for dynamic few-shot generation. While these systems are the highest accuracy systems, they also depend on either closed-source or fine-tuned models with considerable computational expense.

QA-SQL demonstrates a complementary form of excellence with a focus on efficiency, productivity, and scalability. By combining schema-aware retrieval with LLM-driven synthetic augmentation, QA-SQL can dynamically adapt few-shot examples for unseen schemas without fine-tuning, and balance the trade-offs between accuracy and runtime cost. While QA-SQL achieves a moderate execution accuracy (46.68%), QA-SQL achieves the highest Valid Efficiency Score VES = 87.66% demonstrating the superior practicality of a lightweight and deployable framework for real-world usage overall.

DISCUSSION

The experimental analysis is presented in Tables 3 and 4, which reveals notable differences across models in both EX and VES metrics. These results underscore the superior balance of accuracy and efficiency achieved by Groq LLaMA 3 70B Versatile. The evaluation results demonstrate (*Groq, 2024*) superior balance between execution accuracy and computational efficiency. As shown in Table 3, Groq LLaMA 3 70B maintains stable accuracy across all query complexities, achieving 54.05% for simple queries and 31.03% for challenging ones, making it more reliable than GPT-4o for complex SQL tasks.

Table 4 further highlights the advantage of Groq's model in execution efficiency, achieving the highest VES score of 87.66%, significantly outperforming GPT-4o (47.49%) and Claude 3.7 Sonnet (42.48%). This suggests that Groq LLaMA 3 70B generates optimized SQL queries that executed efficiently. Several factors contribute to this performance:

- **Inference speed**: Unlike traditional transformer-based architectures, Groq uses a Tensor Streaming Architecture (TSA), which drastically reduces latency and processing overhead. This enables faster token generation, leading to more computationally efficient SQL execution.

- **Optimization for retrieval-augmented tasks**: Groq's fine-tuning optimizations align well with retrieval-augmented prompting, ensuring schema-aware SQL generation. This reduces redundant joins and unnecessary query steps, which can cause inefficiencies in the output from the GPT-4o and Claude models.

These findings suggest that retrieval-augmented few-shot prompting significantly benefits models with optimized inference speeds, reinforcing Groq LLaMA 3 70B's efficiency for real-world SQL execution. The integration of synthetic SQL-NL pairs and schema-aware retrieval reduces hallucination errors, as reflected in the improved EX performance shown in Table 3 and the efficiency gains in Table 4. QA-SQL demonstrates consistent performance across simple, moderate, and challenging queries (Table 3), which indicates its ability to generalize to unseen database schemas and varying query complexities.

Limitations of embedding-based retrieval

While our retrieval mechanism based on vector embedding and cosine similarity contributes to improved SQL synthesis, these approaches have known limitations. As noted in *Rossi et al. (2024)* "relying on top-K or cosine-similarity cutoff is often insufficient to filter out irrelevant results effectively." Furthermore, embedding heterogeneous content, such as NL queries, database schemas, and SQL statements, can introduce noise and irrelevant dimensions in the embedding space. To mitigate these challenges, QA-SQL integrates schema linking to pre-filter database attributes, generates schema-aware synthetic SQL-NL pairs for consistent semantic contexts, and restricts retrieval to the top-5 examples, which has shown to balance diversity and relevance (Tables 3 and 4). These strategies help reduce retrieval noise and improve the precision of context augmentation.

Role and risks of augmented data

Augmented SQL-NL pairs are critical to QA-SQL's improved accuracy, as they provide schema-specific context and reduce hallucination errors during SQL synthesis. Tables 3 and 4 indicates that models benefit from the inclusion of these augmented examples, as seen in the higher EX and VES scores compared to retrieval without augmentation. However, it is acknowledged that augmented data may, in rare cases, introduce noise or misleading patterns, particularly if the synthetic SQL queries do not fully align with the schema. In Schema Linking, the filtering of irrelevant database columns ensures that embeddings are focused on the most contextually relevant attributes, reducing the risk of introducing noise. While we have not explicitly shown an ablation comparing performance with and without column filtering, our pipeline is designed to discard low-confidence matches and use dynamic few-shot examples that adapt to schema variations. It is also planned to explore formal ablation studies in future work to quantify the direct impact of augmented data and Schema Linking, and to investigate additional safeguards (*e.g.*, LLM-based semantic validation of synthetic queries) to mitigate the risk of degraded accuracy.

CONCLUSION AND FUTURE WORK

In this research article, a novel framework titled "QA-SQL", a schema-aware retrieval and data augmentation-driven approach to NL to-SQL translation is presented. The experimental results indicate that Groq LLaMA 3 70B Versatile effectively balances query accuracy and execution efficiency, making it a strong candidate for real-world SQL generation tasks. Unlike GPT-4o, which excels in accuracy for simple queries but struggles with efficiency, *Groq*'s *(2024)* model maintains stable accuracy across all difficulty levels while achieving the highest VES score (87.66%).

These findings highlight the importance of retrieval-augmented few-shot prompting in reducing schema mismatch errors and improving SQL synthesis. The efficiency gains observed in Groq's model suggest that low-latency inference architectures significantly impact retrieval-based query generation. These results confirm that synthetic data augmentation effectively reduces query errors and that QA-SQL generalizes well across diverse database structures. While QA-SQL's EX is slightly lower than the state-of-the-art DIN-SQL (55.9%), its higher execution efficiency (VES = 87.66%) demonstrates a distinct trade-off between accuracy and query runtime, which we consider critical for real-world SQL deployment.

The system shows solid evidence of improving NL-to-SQL by introducing RAG, but further validation is required across a wider range of real-world databases. Additionally, the current retrieval mechanism relies on static similarity ranking, which may lead to the selection of less relevant few-shot examples, particularly for complex queries.

In the near future, it is planned to expand experiments to directly compare QA-SQL (with the best-performing backbone) against existing state-of-the-art methods on BIRD and other benchmarks, further validating its generality.

Additionally, we acknowledge the value of an ablation study for isolating the effect of synthetic data generation. Although we were unable to include it in the current version due to resource constraints and the complexity of disentangling evidence-based retrieval, we plan to conduct a controlled ablation experiment in future work. This will help quantify the specific contribution of the synthetic evidence module to the accuracy and efficiency of SQL generation.

Also, we intend to extend the evaluation of QA-SQL on various NL-to-SQL benchmarks beyond BIRD, incorporating databases with more complex schema variations. In addition, the work will be carried out in another promising direction to explore reinforcement learning-based retrieval optimization, dynamically adjusting retrieval rankings based on query complexity, schema constraints, and execution feedback to further improve SQL synthesis quality.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by Chungbuk National University BK21 program (2025). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
Chungbuk National University BK21 Program: 2025.

Competing Interests

Anand Nayyar is an Academic Editor for PeerJ.

Author Contributions

- Sokheang Chan conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Vungsovanreach Kong conceived and designed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, and approved the final draft.
- Anand Nayyar performed the experiments, analyzed the data, prepared figures and/or tables, and approved the final draft.
- Tae-Kyung Kim conceived and designed the experiments, performed the experiments, prepared figures and/or tables, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The data is available at Zenodo: Kong, V. (2025). Synthetic_Dataset [Data set]. Zenodo. https://doi.org/10.5281/zenodo.16809493.

REFERENCES

Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh R, Ziegler D, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D. 2020. Language models are few-shot learners. In: *34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada*.

Chen W. 2022. Large language models are few (1)-shot table reasoners. ArXiv DOI 10.48550/arXiv.2210.06710.

Chen M, Tworek J, Jun H, Yuan Q, Pinto HPDO, Kaplan J, Edwards H, Burda Y, Joseph N, Brockman G, Ray A, Puri R, Krueger G, Petrov M, Khlaaf H, Sastry G, Mishkin P, Chan B, Gray S, Ryder N, Pavlov M, Power A, Kaiser L, Bavarian M, Winter C, Tillet P, Petroski F, Cummings D, Plappert M, Chantzis F, Barnes E, Herbert-Voss A, Hebgen Guss W, Nichol A, Paino A, Tezak N, Tang J, Babuschkin I, Balaji S, Jain S, Saunders W, Hesse C, Carr AN, Leike J, Achiam J, Misra V, Morikawa E, Radford A, Knight M, Brundage M, Murati M, Mayer K, Welinder P, McGrew B, Amodei D, McCandlish S, Sutskever I, Zaremba W. 2021. Evaluating large language models trained on code. ArXiv DOI 10.48550/arXiv.2107.03374.

Coelho GM, Nascimento ER, Izquierdo YT, García GM, Feijó L, Lemos M, Garcia RL, de Oliveira AR, Pinheiro JP, Casanova MA. 2024. Improving the accuracy of text-to-SQL tools based on large language models for real-world relational databases. In: *International Conference on Database and Expert Systems Applications*. Cham: Springer, 93–107.

Deng N, Chen Y, Zhang Y. 2022. Recent advances in text-to-SQL: a survey of what we have and what we expect. ArXiv DOI 10.48550/arXiv.2208.10099.

Ding B, Qin C, Zhao R, Luo T, Li X, Chen G, Xia W, Hu J, Luu AT, Joty S. 2024. Data augmentation using large language models: data perspectives, learning paradigms and challenges. ArXiv DOI 10.48550/arXiv.2403.02990.

Dyk DAV, Meng X-L. 2001. The art of data augmentation. *Journal of Computational and Graphical Statistics* **10(1)**:1–50 DOI 10.1198/10618600152418584.

Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, Dai Y, Sun J, Wang H, Wang H. 2023. Retrieval-augmented generation for large language models: a survey. ArXiv DOI 10.48550/arXiv.2312.10997.

Groq. 2024. Groq-accelerating AI with the world's fastest language processing. *Available at https://groq.com* (accessed 10 January 2025).

Guo C, Tian Z, Tang J, Li S, Wen Z, Wang K, Wang T. 2023a. Retrieval-augmented GPT-3.5-based text-to-SQL framework with sample-aware prompting and dynamic revision chain. In: *International Conference on Neural Information Processing*. Cham: Springer, 341–356.

Guo Z, Yan M, Qi J, Zhou J, He Z, Lin Z, Zheng G, Wang X. 2023b. Few-shot table-to-text generation with prompt planning and knowledge memorization. ArXiv DOI 10.48550/arXiv.2302.04415.

Hong Z, Yuan Z, Zhang Q, Chen H, Dong J, Huang F, Huang X. 2024. Next-generation database interfaces: a survey of LLM-based text-to-SQL. ArXiv DOI 10.1109/tkde.2025.3609486.

Hong Z, Yuan Z, Zhang Q, Chen H, Dong J, Huang F, Huang X. 2025. Next-generation database interfaces: a survey of LLM-based text-to-SQL. ArXiv DOI 10.48550/arXiv.2406.08426.

Lee C-H, Polozov O, Richardson M. 2021. KaggleDBQA: realistic evaluation of Text-to-SQL parsers. In: *Annual Meeting of the Association for Computational Linguistics*. Stroudsburg: ACL.

Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih WT, Rocktäschel T, Riedel S, Kiela D. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: *34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada*.

Li J, Hui B, Qu G, Yang J, Li B, Li B, Wang B, Qin B, Cao R, Huo N, Zhou X, Chenhao M, Li G, Chang K, Huang F, Cheng R, Li Y. 2023. Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs. In: *NIPS '23: Proceedings of the 37th International Conference on Neural Information Processing Systems*.

Liu A, Hu X, Wen L, Yu PS. 2023. A comprehensive evaluation of ChatGPT's zero-shot text-to-SQL capability. ArXiv DOI 10.48550/arXiv.2303.13547.

Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G. 2021. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. *ACM Computing Surveys* **55(9)**:1–35 DOI 10.1145/3560815.

Nascimento ER, Avila CVS, Izquierdo YT, García GM, Andrade LFL, Facina MSP, Lemos M, Casanova MA. 2025. Text-to-SQL based on large language models and database keyword search. ArXiv DOI 10.48550/arXiv.2501.13594.

Pourreza M, Li H, Sun R, Chung Y, Talaei S, Kakkar GT, Gan Y, Saberi A, Ozcan F, Arik SO. 2024. Chase-SQL: multi-path reasoning and preference optimized candidate selection in text-to-SQL. ArXiv DOI 10.48550/arXiv.2410.01943.

Pourreza MR, Rafiei D. 2023. Din-SQL: decomposed in-context learning of text-to-SQL with self-correction. ArXiv DOI 10.48550/arXiv.2304.11015.

Qin Y, Chen C, Fu Z, Chen Z, Peng D, Hu P, Ye J. 2024. Route: robust multitask tuning and collaboration for text-to-SQL. ArXiv DOI 10.48550/arXiv.2412.10138.

Qin B, Hui B, Wang L, Yang M, Li J, Li B, Geng R, Cao R, Sun J, Si L, Huang F, Li Y. 2022. A survey on text-to-SQL parsing: concepts, methods, and future directions. ArXiv DOI 10.48550/arXiv.2208.13629.

Rajkumar N, Li R, Bahdanau D. 2022. Evaluating the text-to-SQL capabilities of large language models. ArXiv DOI 10.48550/arXiv.2204.00498.

Rossi N, Lin J, Liu F, Yang Z, Lee T, Magnani A, Liao C. 2024. Relevance filtering for embedding-based retrieval. In: *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM'24*. New York: ACM, 4828–4835.

Salesforce. 2020. Talk to your data: one model, any database. *Available at https://www.salesforce.com/blog/talk-to-your-data-one-model-any-database* (accessed 16 February 2025).

Secoda. 2024. Unleashing the power of data: study shows 2/3 of company data goes unused. *Available at https://www.secoda.co/learn/unleashing-the-power-of-data-study-shows-2-3-of-company-data-goes-unused* (accessed 14 March 2025).

Shi T, Zhao C, Boyd-Graber JL, Daum'e H, Lee L. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. ArXiv DOI 10.48550/arXiv.2010.11246.

Coherent Solutions. 2024. NLP in business intelligence: 7 success stories, benefits, and future trends. *Available at https://www.coherentsolutions.com/insights/nlp-in-business-intelligence-7-success-stories-benefits-and-future-trends* (accessed 16 February 2025).

Sutskever I, Vinyals O, Le QV. 2014. Sequence to sequence learning with neural networks. ArXiv DOI 10.48550/arXiv.1409.3215.

Uber. 2025. How Uber is saving 140,000 hours each month using text-to-SQL and how you can harness the same. *Available at https://medium.com/wrenai/how-uber-is-saving-140-000-hours-each-month-using-text-to-SQL-and-how-you-can-harness-the-same-fb4818ae4ea3* (accessed 16 February 2025).

Wang L, Zhang A, Wu K, Sun K, Li Z, Wu H, Zhang M, Wang H. 2020. ChiTeSQL: a large-scale and pragmatic Chinese text-to-SQL dataset. In: *Conference on Empirical Methods in Natural Language Processing*.

Wire B. 2022. Over 80 percent of companies rely on stale data for decision making. *Available at https://www.businesswire.com/news/home/20220511005403/en/Over-80-Percent-of-Companies-Rely-on-Stale-Data-for-Decision-Making* (accessed 16 February 2025).

Xie X, Xu G, Zhao L, Guo R. 2025. Opensearch-SQL: enhancing text-to-SQL with dynamic few-shot and consistency alignment. *Proceedings of the ACM on Management of Data* **3(3)**:1–24 DOI 10.1145/3725331.

Yu T, Zhang R, Er HY, Li S, Xue E, Pang B, Lin XV, Tan YC, Shi T, Li Z, Jiang Y, Yasunaga M, Shim S, Chen T, Fabbri A, Li Z, Chen L, Zhang Y, Dixit S, Zhang V, Xiong C, Socher R, Lasecki W, Radev D. 2019a. CoSQL: a conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. ArXiv DOI 10.48550/arXiv.1909.05378.

Yu T, Zhang R, Yang K, Yasunaga M, Wang D, Li Z, Ma J, Li I, Yao Q, Roman S, Zhang Z, Radev D. 2019b. Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. ArXiv DOI 10.48550/arXiv.1809.08887.

Zhang Y, Deriu J, Katsogiannis-Meimarakis G, Kosten C, Koutrika G, Stockinger K. 2023. ScienceBenchmark: a complex real-world benchmark for evaluating natural language to SQL systems. *Proceedings of the VLDB Endowment* **17(4)**:685–698 DOI 10.14778/3636218.3636225.

Zhong V, Xiong C, Socher R. 2017. Seq2SQL: generating structured queries from natural language using reinforcement learning. ArXiv DOI 10.48550/arXiv.1709.00103.

Zhu X, Li Q, Cui L, Liu Y. 2024. Large language model enhanced text-to-SQL generation: a survey. ArXiv DOI 10.48550/arXiv.2410.06011.