DB-Explore: Automated Database Exploration and Instruction Synthesis for Text-to-SQL

Haoyuan Ma[1], Yongliang Shen[1†], Hengwei Liu[1], Wenqi Zhang[1], Haolei Xu[1], Qiuying Peng[2] Jun Wang[2], Weiming Lu[1†]
[1]Zhejiang University, [2] OPPO Research Institute
{mahaoyuan, syl, luwm}@zju.edu.cn
{pengqiuying, wangjun7}@oppo.com

Abstract

Recent text-to-SQL systems powered by large language models (LLMs) have demonstrated remarkable performance in translating natural language queries into SQL. However, these systems often struggle with complex database structures and domain-specific queries, as they primarily focus on enhancing logical reasoning and SQL syntax while overlooking the critical need for comprehensive database understanding. To address this limitation, we propose DB-Explore, a novel framework that systematically aligns LLMs with database knowledge through automated exploration and instruction synthesis. DB-Explore constructs database graphs to capture complex relational schemas, leverages GPT-4 to systematically mine structural patterns and semantic knowledge, and synthesizes instructions to distill this knowledge for efficient fine-tuning of LLMs. Our framework enables comprehensive database understanding through diverse sampling strategies and automated instruction generation, bridging the gap between database structures and language models. Experiments conducted on the SPIDER and BIRD benchmarks validate the effectiveness of DB-Explore, achieving an execution accuracy of 67.0% on BIRD and 87.8% on SPIDER. Notably, our open-source implementation based on Qwen2.5-Coder-7B achieves state-of-the-art results at minimal computational cost, outperforming several GPT-4-driven Text-to-SQL systems.

1 Introduction

Text-to-SQL systems have emerged as a crucial bridge between humans and structured data, enabling natural language interaction with databases without requiring SQL expertise. While recent advances in large language models (LLMs), including ChatGPT (Liu et al., 2023), GPT-4 (Achiam et al., 2023), and Qwen2 (Hui et al., 2024), have significantly improved these systems' capabilities, demonstrating enhanced comprehension of user queries and the ability to generate complex SQL queries, they still face fundamental challenges in understanding complex database structures and generating accurate SQL queries, particularly for domain-specific applications (Zhang et al., 2024; Liu et al., 2023).

A key limitation of current approaches is their reliance on direct query translation without deep understanding of the underlying database structure and semantics. Existing methods primarily focus improving prompting strategies (Dong et al., 2023; Rajkumar et al., 2022; Liu et al., 2023; Sun et al., 2023) or enhancing the reasoning ability of LLMs through supervised fine-tuning (Li et al., 2023a; Scholak et al., 2021; Wang et al., 2020), but often overlook the critical role of database comprehension in query generation. While techniques like schema linking (Wang et al., 2020; Li et al., 2023a), question decomposition (Pourreza and Rafiei, 2023) and self-correction (Wang et al., 2023) have attempted to address this gap, they typically treat the database as a static structure rather than a rich source of semantic knowledge that can inform the query generation process.

We argue that effective Text-to-SQL systems must first develop a comprehensive understanding of the database through systematic exploration before attempting query generation. This exploration should capture not only the structural relationships between tables but also the semantic patterns and domain-specific knowledge embedded within the schema. However, automated database exploration presents several challenges: (1) *Complex structural relationships:* Real-world databases often contain intricate networks of table relationships that are difficult to capture and represent effectively. (2) *Hidden semantic patterns:* Domain-specific knowledge embedded in schema elements (table names,

†Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 18969–18979
November 4-9, 2025

column names, etc.) is often implicit and requires careful analysis to uncover. (3) *Query complexity progression:* Understanding how simple queries can be systematically extended to handle more complex scenarios requires a structured approach to knowledge acquisition.

To address these challenges, we propose DB-Explore, a novel framework that transforms Text-to-SQL training through systematic database exploration and instruction synthesis. At the core of our approach is a graph-based database representation technique called DB Graph, which captures both structural relationships and semantic patterns within the database. This representation serves as a foundation for three key components: (1) *Semantic Knowledge Extraction:* We develop techniques to uncover domain-specific knowledge embedded in schema elements, enabling more contextually appropriate query generation. (2) *Structural Pattern Mining:* By analyzing the DB Graph, our system identifies common join patterns, table relationships, and query templates that reflect the database's architectural design. (3) *Progressive Instruction Synthesis:* Using the extracted knowledge, we automatically generate training examples that progress from simple to complex queries, allowing models to gradually master increasingly sophisticated query patterns.

Our framework leverages these components to create comprehensive training datasets that capture both the structural and semantic complexity of the target database. This approach fundamentally differs from existing methods by treating database exploration as a crucial prerequisite for effective query generation, rather than relying solely on input-output pairs or schema information. The main contributions of this work are:

1. We introduce DB-Explore, a framework that revolutionizes Text-to-SQL training by incorporating systematic database exploration and knowledge extraction.

2. We develop DB Graph, a novel representation that captures both structural relationships and semantic patterns within databases, enabling more effective exploration and learning.

3. We present techniques for automated instruction synthesis that leverage extracted database knowledge to create progressively complex training examples.

4. We demonstrate significant improvements in query generation accuracy and complexity handling across multiple benchmark datasets with minimal computational cost.

2 Related Work

Early Text-to-SQL systems relied on handcrafted, rule-based parsers to map natural language into SQL (Li and Jagadish, 2014; Popescu et al., 2003). The advent of large language models (LLMs) such as GPT-4 (Achiam et al., 2023) shifted focus to leveraging rich language priors for SQL generation (Li et al., 2023b). Building on this paradigm, schema-focused methods such as RESD-SQL, DTS-SQL, and RSL-SQL enhance schema linking and selection to ground queries in database structures (Li et al., 2023a; Pourreza and Rafiei, 2024; Cao et al., 2024), while decomposition-based techniques like DIN-SQL, PTD-SQL, and QPL break complex queries into simpler sub-tasks (Pourreza and Rafiei, 2023; Luo et al., 2024; Eyal et al., 2023). MSC-SQL, SQL-PaLM, Distillery, and XiYan improve robustness through multi-candidate generation and ranking (Gorti et al., 2024; Sun et al., 2023; Maamari et al., 2024; Gao et al., 2024b), and skeleton-aware selection in DAIL-SQL refines few-shot demonstrations (Gao et al., 2024a). Self-correction frameworks in MAC and EPI-SQL further reduce errors via iterative feedback (Wang et al., 2023; Liu and Tan, 2024).

In parallel, competitive open-source LLMs have spurred fine-tuning frameworks that exploit synthetic or task-specific data. CODES incrementally pre-trains StarCoder on SQL corpora and integrates external plugins for enhanced performance (Li et al., 2024), whereas DTS-SQL and SENSE employ schema-aware fine-tuning and direct preference optimization (DPO) (Rafailov et al., 2024) to improve accuracy (Pourreza and Rafiei, 2024; Yang et al., 2024b). ROUTE unifies multi-task tuning through targeted data synthesis (Qin et al., 2024). Recently, reinforcement learning (RL) methods have emerged as a promising direction in the Text-to-SQL field. SQL-o1 (Lyu et al., 2025) employs Monte Carlo Tree Search (MCTS) (Świechowski et al., 2023) to enhance the reasoning process during SQL generation. SQL-R1 (Ma et al., 2025) and Reasoning-SQL (Pourreza et al., 2025) leverage the GRPO (Liu et al., 2024) framework and incorporate domain-specific reward functions, improving the accuracy and robustness of Text-to-SQL systems. Despite these advances, most work remains

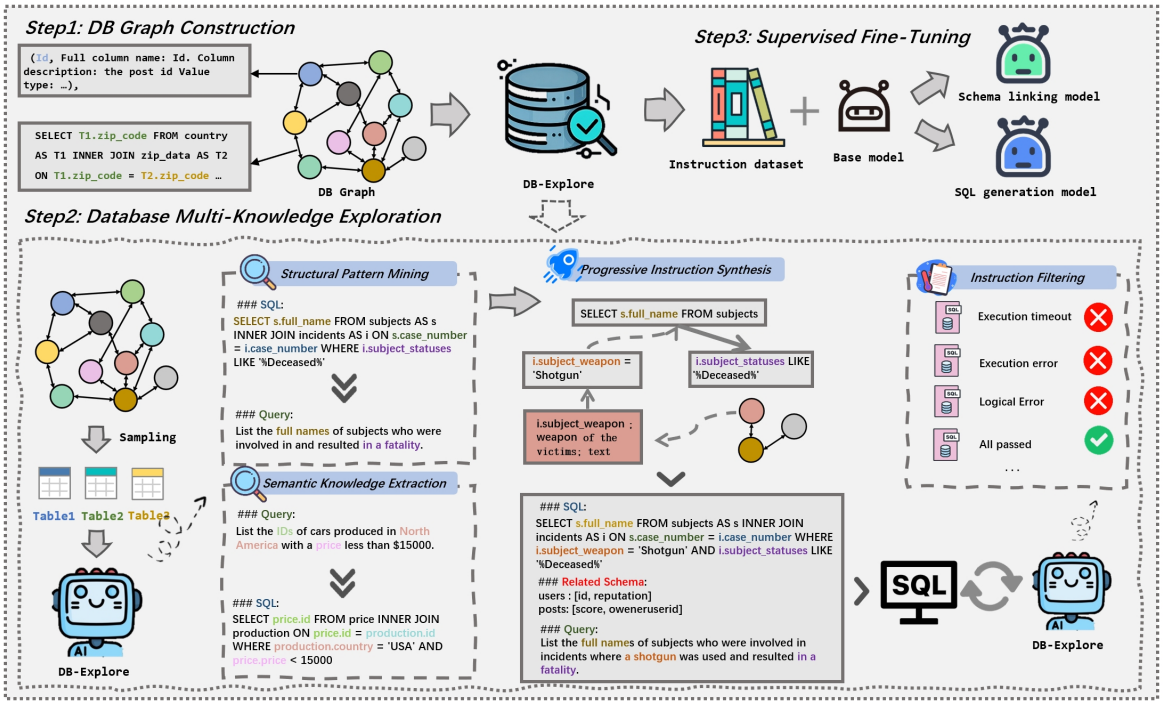

Figure 1: Overall framework of DB-Explore. Our framework operates through three principal phases: (1) Database Graph Construction, (2) Database Multi-Knowledge Exploration: semantic knowledge extraction, structural pattern mining, progressive instruction synthesis, and filtering mechanisms, and (3) Supervised Fine-Tuning executing database-adaptive model training. The synthesis phase systematically produces database-consistent training corpora through multi-stratum knowledge alignment.

focused on direct SQL generation, often overlooking holistic database understanding.

3 Methodology

In this section, we present a detailed explanation of DB-Explore, a fully automated framework that leverages LLMs to explore databases, generate data, and perform supervised fine-tuning. As illustrated in Figure 1, the framework comprises three core components: 1) Database Graph Construction, 2) Database Multi-Knowledge Exploration, and 3) Supervised Fine-Tuning.

DB-Explore initiates with the construction of DB Graph, which captures the intricate topological relationships within the database. By applying various sampling strategies, we extract subgraphs containing diverse information. DB-Explore then converts this database knowledge into a instruction dataset through semantic knowledge extraction, structural pattern mining, and progressive instruction synthesis. To ensure the quality of the generated instructions, an instruction filtering mechanism is employed. Finally, the extracted knowledge is injected into the base model via supervised fine-tuning. In the following subsections, we present a detailed description of each component of DB-Explore.

3.1 Database Graph Construction and Sampling

To effectively model both structural and semantic relationships within databases, we formalize database schemas as DB graph, as illustrated in Step 1 of Figure 1. The graph comprises nodes representing database columns and edges encoding relational constraints. Two edge types are defined: Intra-table edges connect columns within the same table, enabling direct access without JOIN operations; Inter-table edges represent foreign key relationships across tables, bridging subgraphs into a unified structure. Edge weights are assigned based on co-occurrence frequency of connected columns in seed queries, prioritizing frequently accessed schema patterns that reflect real-world query behaviors.

To enable more effective exploration and learning, we propose two complementary sampling strategies: 1) **Random Walk Sampling**: Starting from selected nodes, the walker iteratively traverses adjacent nodes via randomly chosen edges until reaching a predefined sampling size. This method ensures comprehensive schema coverage

but may overlook high-priority relationships. 2) **Empirical Weighted Sampling**: Normalized edge weights guide traversal probabilities, biasing selection toward high-weight edges. This strategy emphasizes schema subgraphs aligned with user query patterns, enhancing instruction relevance. Both strategies generate subgraphs as input for subsequent database multi-knowledge exploration, balancing semantic diversity via random walks and user intent alignment via weighted sampling.

3.2 Database Multi-Knowledge Exploration

We propose a three-pronged exploration and data synthesis framework to explore database by instruction synthesis while minimizing computational overhead: 1) Semantic Knowledge Extraction, 2) Structural Pattern Mining, and 3) Progressive Instruction Synthesis.

3.2.1 Semantic Knowledge Extraction

Semantic knowledge extraction mechanism leverages LLMs' internal knowledge to discover latent schema relationships through a self-guided paradigm. The process initiates by sampling schema subgraphs S from the DB Graph via random walk sampling, containing table/column metadata and structural relationships.

Self-instruction has been shown to be an effective way to synthesize data (Wang et al., 2022). The instruction synthesis pipeline operates through a self-instruct framework initialized using manually-crafted template. We prompt the LLM with S and randomly sampled seed instructions as demonstrations to synthesize novel queries Q through schema-constrained synthesis. The generated $\langle Q, S \rangle$ pairs are subsequently feedback to the LLM to produce executable SQL responses R.

To achieve optimal semantic diversity while preserving logical validity, we implement temperature scaling (τ=0.8) during generation. The iterative process progressively enriches the instruction pool through random sampling, thereby expanding coverage of potential user intents.

3.2.2 Structural Pattern Mining

Unlike semantic knowledge extraction, structural pattern mining is designed to generate schema-grounded SQL queries through explicit modeling of relational topologies derived from the DB Graph. It prioritizes foreign-key dependencies and latent connectivity patterns observed in seed instructions, while strategically balancing single-table operations with multi-tables queries through weighted edge sampling. We propose a SQL-first generation paradigm to ensure structural accuracy by initially producing executable SQL statements that rigorously adhere to schema constraints. These syntactically valid queries are subsequently translated into natural language instructions via inverse parsing, effectively reducing hallucination risks compared to open-ended synthesis.

As shown in Figure 1, we first apply empirical weighted sampling to select schema subgraphs S, giving priority to high-frequency foreign-key edges. For each selected subgraph, SQL statements R are automatically generated by traversing all nodes in the subgraph, incorporating JOIN operations based on foreign keys, filter conditions derived from column data types, and aggregation functions applied to numerical columns. The resulting SQL statements are then translated into natural language queries using the back-instruct method (Shen et al., 2023), where an LLM rewrites each SQL query R into a structurally faithful natural language instruction Q.

3.2.3 Progressive Instruction Synthesis

In this section, we introduce a progressive instruction synthesis mechanism that progressively enriches query complexity to address the challenge of generating complex queries from simple user inputs. As illustrated in Figure 1, SQL queries are formalized as syntax trees where the root represents the core SELECT statement and leaf nodes correspond to schema-specific constraints. Each leaf node maps to a modular condition that can be independently translated into natural language, establishing a bidirectional bridge between SQL semantics and user intent.

The synthesis process initiates with a base query sampled from the DB Graph. Through iterative refinement: 1. Condition Augmentation: Schema-derived nodes (foreign keys, constraints) are recursively appended to the syntax tree via DB Graph-guided sampling. 2. Complexity Scaling: The LLM orchestrates tree expansion by injecting compound conditions such as nested WHERE clauses and multi-table JOIN while preserving syntactic validity. 3. Instruction Rewriting: Each evolved SQL tree is verbalized into diverse natural language expressions through LLM rewriting.

Each iteration introduces new constraints and conditions to the instruction, progressively increasing its complexity. We set the number of iterations

to 3, corresponding to user instructions of varying difficulty levels: simple, moderate, and challenging. This progressive paradigm generates instruction pairs ranging from atomic queries to multi-hop reasoning tasks, effectively simulating the evolutionary patterns of real-world user queries.

3.3 Instruction Filtering

Progressive instruction synthesis often generates a significant amount of noisy data due to hallucinations of LLMs. To mitigate noise data, we implement a dual-stage validation pipeline:

All generated SQL candidates R are executed against the target database, and queries are discarded if they contain syntax errors, reference invalid columns or tables, or exceed an execution time limit of 25 seconds. The surviving $<Q, R>$ pairs are then subjected to LLM-based consistency verification to ensure faithful alignment between the natural language intent and the SQL logic, as well as strict adherence to schema constraints defined in S. Validated instances are formatted as augmented tuples $<Q, D, R, S>$, where D denotes the database context. These high-quality instances are reintegrated into the seed dataset, forming a self-improving feedback loop that drives iterative synthesis and database exploration.

3.4 Supervised Fine-Tuning and Inference

To inject the database knowledge into base model, we implement a two-stage supervised fine-tuning framework using the synthesized instruction dataset $<Q, D, R, S>$. Addressing the challenge of schema complexity versus limited context windows, user queries undergo schema linking to extract critical schema elements S before SQL generation. This cascaded approach reduces irrelevant schema noise and minimizes token consumption.

To enhance the model's comprehension of database structures during inference, we integrates a Value Retrieval Module which implements a coarse-to-fine matching mechanism via BM25 indexing and Longest Common Subsequence (LCS) algorithms (Li et al., 2024), resolving token mismatches. Additionally, metadata augmentation is utilized, embedding database descriptions, column types, foreign keys, and domain knowledge into prompts.

4 Experiments

Benchmarks (1) **SPIDER** The SPIDER dataset (Yu et al., 2018) is a large-scale, cross-domain benchmark for Text-to-SQL tasks, designed to evaluate the generalization ability of models across diverse database structures. It consists of 8,659 training samples, and 1,034 development samples, spanning 200 databases and 138 domains. (2) **BIRD** The BIRD dataset (Li et al., 2023b) is a more challenging Text-to-SQL benchmark compared to SPIDER, featuring 95 real-world databases across 37 professional domains, with a total storage size of 33.4GB.

Evaluation Metrics In our experiments, we use three key evaluation metrics to assess the performance of Text-to-SQL models. Execution Accuracy (EX) (Yu et al., 2018) measures the proportion of queries where the predicted SQL query produces the exact output as the ground truth, serving as a direct indicator of correctness. Test-Suite Accuracy (TS) (Zhong et al., 2020) stands out as a more trustworthy metric than EX. For BIRD, we report EX along with the Valid Efficiency Score (VES) (Li et al., 2023b), which is used to evaluate the execution efficiency of accurately generated SQL queries. Unlike EX, in VES the score for an accurately generated SQL query is no longer fixed at 1. Instead, it is determined by the ratio of the execution time of the ground truth SQL query to that of the predicted SQL query.

Implementation Details We employ GPT-4o (Hurst et al., 2024) for data synthesis and fine-tune Qwen2.5-Coder-7B (Yang et al., 2024a) as our base model. A total of 26,000 and 22,000 initial synthetic query-SQL pairs are first constructed for the SPIDER and BIRD datasets; after filtering, we obtain 21,273 and 16,457 valid query-SQL pairs for the two datasets, as shown in Table 2, corresponding to a success rate of 81.8% for SPIDER and 74.8% for BIRD. All experiments are conducted on four NVIDIA A6000 GPUs with a batch size of 32. We utilize the Llama-Factory framework (Zheng et al., 2024) for parameter-efficient fine-tuning. Both schema linking and text generation modules are trained for 3 epochs using AdamW optimization, with an initial learning rate of $1e$-5 and a context window of 4,096 tokens. The learning rate follows a cosine decay schedule. For self-consistency decoding, we sample 8 SQL candidates with a temperature of 0.8.

Baselines We evaluate DB-Explore against state-of-the-art Text-to-SQL methods across two major paradigms. The first includes approaches **Prompt-**

Methods	SPIDER		BIRD	
	Dev-EX	Dev-TS	Dev-EX	Dev-VES
Prompting with Closed-Source LLMs				
MCS-SQL + GPT-4 (Lee et al., 2025)	89.5	-	63.4	64.8
XIYAN (Gao et al., 2024b)	89.7	-	73.3	-
CHASE-SQL + Gemini-1.5 (Pourreza et al., 2024)	87.6	-	73.1	-
GPT-4 (Achiam et al., 2023)	72.9	64.9	46.4	49.8
DIN-SQL + GPT-4 (Pourreza and Rafiei, 2023)	82.8	74.2	50.7	58.8
DAIL-SQL + GPT-4 (Gao et al., 2024a)	83.5	76.2	54.8	56.1
MAC-SQL + GPT-4 (Wang et al., 2023)	86.8	82.8	59.4	66.2
TA-SQL + GPT-4 (Qu et al., 2024)	85.0	-	56.2	-
MAG-SQL + GPT-4 (Xie et al., 2024)	85.3	-	61.1	-
Fine-Tuning with Open-Source LLMs 7B				
DTS-SQL + DeepSeek-7B (Pourreza and Rafiei, 2024)	82.7*	78.4*	55.8	60.3
CODES-7B (Li et al., 2024)	85.4	80.3	57.2	58.8
SENSE-7B (Yang et al., 2024b)	83.2	81.7	51.8	-
ROUTE + Qwen2.5-7B (Qin et al., 2024)	83.6	77.5	55.9	57.4
OmniSQL-7B (Li et al., 2025)	81.6	-	63.9	-
SQL-o1 + Qwen2.5-7B (Lyu et al., 2025)	84.7	78.5	66.7	70.4
SQL-R1(self-consistency@8) (Ma et al., 2025)	87.6	-	66.6	-
Reasoning-SQL + Qwen2.5-Coder-7B(Pourreza et al., 2025)	-	-	64.0	-
Ours: DB-Explore + Qwen2.5-Coder-7B	87.2	80.2	65.2	68.9
Ours: DB-Explore + Qwen2.5-Coder-7B + self-consistency@8	**87.8**	**81.9**	**67.0**	**71.2**

Table 1: Experimental results on SPIDER and BIRD benchmarks. Asterisks '*' denote performance re-evaluated through the official open-source implementation by ROUTE. Within the Fine-Tuning with Open-Source LLMs 7B group, top-performing entries are **bold** with runner-up scores underlined.

ing with Closed-Source LLMs, such as MCS-SQL (Lee et al., 2025) and Chase-SQL (Guo et al., 2021), which leverage multi-prompt strategies and result selection. MAC-SQL (Wang et al., 2023) and MAG-SQL (Xie et al., 2024) adopt multi-agent frameworks for error correction, while TA-SQL (Qu et al., 2024), DIN-SQL (Pourreza and Rafiei, 2023), and DAIL-SQL (Gao et al., 2024a) improve generation through task alignment, decomposition, and demonstration selection. The second group involves **Fine-Tuning Open-Source 7B LLMs**. DTS-SQL (Pourreza and Rafiei, 2024) employs a two-stage pipeline, SENSE (Yang et al., 2024b) leverages preference learning, and CODES (Li et al., 2024) incorporates extensive pretraining and plugins. ROUTE (Qin et al., 2024) introduces multi-task learning, while SQL-o1 (Lyu et al., 2025), SQL-R1 (Ma et al., 2025), and Reasoning-SQL (Pourreza et al., 2025) integrate reinforcement learning to enhance SQL generation.

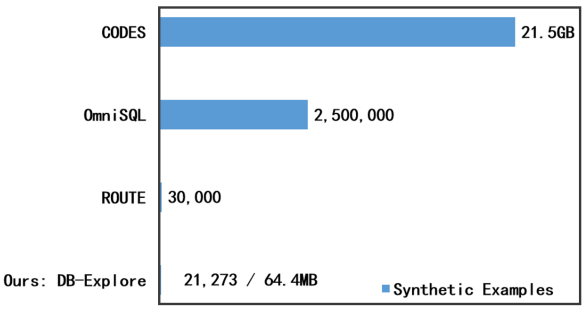

Figure 2: Data volume analysis of different data synthesis methods on BIRD, CODES only reports the storage size for pre-training data

5 Results and Analysis

5.1 Main Results

Performance on Main Benchmarks As shown in Table 1, DB-Explore achieves strong performance, attaining execution accuracies of 87.8% on the SPIDER development set and 67.0% on the BIRD development set—setting a new state-of-the-art among methods based on open-source 7B models. Compared to the baseline using Qwen2.5-Coder-7B, our approach yields substantial improvements. Notably, DB-Explore outperforms reinforcement learning-based methods such as SQL-R1 (Ma et al., 2025) and Reasoning-SQL (Pourreza et al., 2025), despite relying only on limited supervised fine-tuning (SFT), significantly reducing training costs. Moreover, DB-Explore surpasses

Datasets	Initial Synthetic Pairs	Total Valid Pairs	Semantic Knowledge Extraction Rate	Structural Pattern Mining Rate	Success Rate
SPIDER	26000	21273	36%	64%	81.8%
BIRD	22000	16457	32%	68%	74.8%

Table 2: Performance of synthetic data on SPIDER and BIRD Datasets.

	SPIDER		BIRD	
	DB-Explore	Base Model	DB-Explore	Base Model
Full-Schema EX	83.9	80.0	63.5	53.7
Filtered-Schema EX	87.2	-	65.2	-
Gold-Schema EX	89.1	83.4	69.9	63.8

Table 3: The performance of the SQL generation model for different schema inputs, where the filtered-schema is the result of the DB-Explore schema linking model.

many Text-to-SQL methods that rely on closed-source models, including GPT-4 and GPT-4o. On the BIRD benchmark, it also achieves a VES score of 71.2%, underscoring that our database-specific exploration strategy can yield highly efficient query generation through capturing complex schema patterns.

Analysis of Data Volume Unlike existing approaches that focus on enhancing LLMs' generalization capabilities for SQL generation, DB-Explore adopts a database-centric data synthesis paradigm that prioritizes mining database-specific knowledge. This fundamental distinction enables our method to achieve competitive performance with significantly reduced data requirements. As illustrated in Figure 2, our approach achieves the lowest computational cost among all data synthesis-based Text-to-SQL methods. Specifically, DB-Explore demonstrates superior data efficiency on BIRD benchmark, achieving a significantly higher EX accuracy than other data synthesis methods while using only 70.9% of the data used by ROUTE, and less than 1% of that used by OmniSQL and CODES.

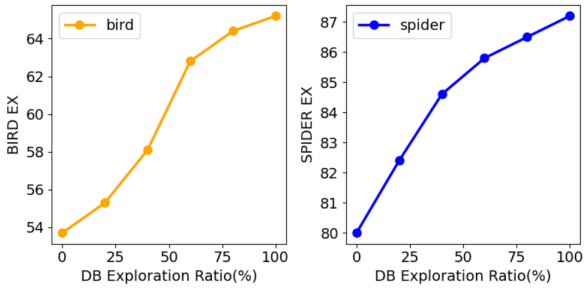

Figure 3: DB-Explore performance trend under different database exploration ratios.

Analysis of DB Exploration Ratio We analyze the impact of database exploration levels on model performance across the SPIDER and BIRD development sets. Different exploration levels are achieved by constraining the sampling scope of the DB Graph: 0% corresponds to the Qwen2.5-Coder-7B baseline without exploration, while 100% denotes full exploration as performed by DB-Explore. As shown in Figure 3, due to the relatively uniform distribution of queries across databases in both datasets, model performance increases almost linearly with the exploration ratio. Notably, database exploration not only enriches the model's schema understanding but also enhances its overall Text-to-SQL capabilities through fine-tuning. As a result, performance gains are slightly more pronounced in the early stages of exploration compared to the later stages.

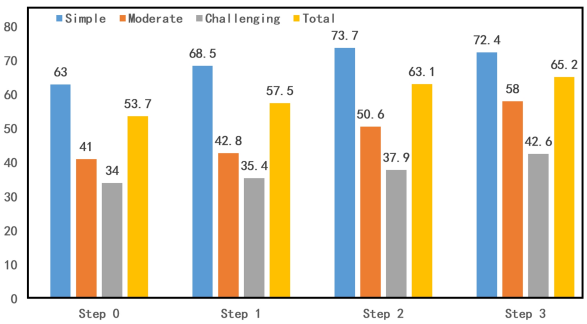

Figure 4: Performance comparison of DB-Explore at different instruction generation stages on BIRD development set.

Analysis of Progressive Instruction Synthesis To investigate the effect of progressive instruction generation on DB-Explore's ability to handle complex queries, we analyze its performance across

Methods	SPIDER Dev-EX	BIRD Dev-EX
Ours: DB-Explore + Qwen2.5-Coder-7B	**87.2**	**65.2**
- w/o Semantic Knowledge Extraction	$86.1_{\downarrow 1.1}$	$64.8_{\downarrow 0.4}$
- w/o Structural Pattern Mining	$83.9_{\downarrow 3.3}$	$62.5_{\downarrow 2.7}$
- w/o Progressive Instruction Synthesis	$84.8_{\downarrow 2.4}$	$62.2_{\downarrow 3.0}$
- w/o Instruction Filtering	$83.7_{\downarrow 3.5}$	$61.4_{\downarrow 3.8}$

Table 4: Ablation study on DB-Explore, complete method of DB-Explore is **bold**

different instruction synthesis stages on the BIRD development set. Each step corresponds to one iteration of evolving the training instructions; to prevent excessive distributional drift, we retain all simpler instructions from prior steps at every iteration. As shown in Figure 4, DB-Explore exhibits a significant improvement in handling simple queries at Step 1, where the synthesized instructions remain relatively basic. However, performance on moderate and challenging queries shows little change at this stage. In Steps 2 and 3, as the synthesized data becomes increasingly complex through iterative evolution, the model's performance on moderate and challenging queries improves rapidly. As iterations accumulate, the synthesized instructions at Step 3, which now encompass queries involving four tables and five tables, are substantially more complex than those in the evaluation set. Although we retain all simpler instructions from Step 1 and Step 2, the overall data distribution at Step 3 still shifts slightly toward the more difficult examples, leading to a modest decline in performance on simple queries.

Analysis of Schema Linking As shown in Table 3, the schema linking module not only alleviates the context length limitations of LLMs but also effectively focuses their attention on relevant schema elements. Incorporating Filtered-Schema leads to a 3.3% and 1.7% execution accuracy improvement on the SPIDER and BIRD datasets. Moreover, DB-Explore achieves strong performance when provided with the gold schema, further demonstrating its effectiveness on the Text-to-SQL task.

5.2 Ablation Study

Study on Semantic Knowledge Extraction and Structural Pattern Mining We conducted ablation experiments to quantify the individual and joint effects of Semantic Knowledge Extraction and Structural Pattern Mining on DB-Explore's performance. As shown in Table 4, when the Semantic Knowledge Extraction module is removed, accuracy decreases by 1.1% on SPIDER and 0.4% on BIRD, indicating its role in extracting entity categories and predicate relationships that guide precise column and value mapping. By contrast, disabling Structural Pattern Mining which captures foreign-key dependencies and the global topology of the database schema results in a much larger 3.3% drop on SPIDER and 2.7% drop on BIRD, primarily due to increased join-path selection failures and ambiguous column predictions. Due to the inherent semantic information present in the synthesized data, Structural Pattern Mining brings even larger gains than Semantic Knowledge Extraction.

Study on Progressive Instruction Synthesis To assess the impact of gradually increasing query complexity, we evaluated DB-Explore with and without the Progressive Instruction Synthesis module. Incorporation of this module yields a 2.4% improvement on SPIDER and a 3.0% gain on BIRD, with the majority of benefits observed on moderate and challenging queries that require multi-table joins and nested filters.

Study on Instruction Filtering Finally, we examined the necessity of our Instruction Filtering component, which prunes hallucinated or incomplete instructions produced during synthesis. Without this filter, performance on SPIDER declines by 3.5% and on BIRD by 3.8%, reflecting the disruptive influence of low-quality examples. This ablation confirms that rigorous filtering is indispensable for maintaining instruction quality and preventing performance degradation caused by spurious or malformed queries.

6 Conclusion

In this paper, we propose DB-Explore, a framework for database exploration, instruction synthesis, and fine-tuning that enhances the LLMs' understanding of databases by extracting dynamic

database knowledge, thereby unlocking the potential of LLMs in Text-to-SQL tasks. Our approach, through database multi-knowledge exploration, effectively minimizes the risks associated with insufficient database knowledge in SQL generation. We demonstrate the efficacy and superiority of our method on recent LLMs across several benchmark tests. The results show that our approach achieves state-of-the-art performance in Text-to-SQL with minimal computational cost. In the future, we aim to explore more database knowledge, larger LLMs, and more efficient fine-tuning frameworks to achieve robust and efficient Text-to-SQL solutions.

Limitations

While our exploration demonstrates promising results, several important limitations warrant acknowledgment. First, the framework introduces modest computational cost and GPT-4o API consumption. Although these costs remain substantially lower than comparable methods, we encourage future work to explore more efficient data generation and fine-tuning paradigms. Second, our upper-bound analysis reveals a persistent performance gap between our approach and inference frameworks directly leveraging closed-source models. Finally, due to computational constraints, the framework's effectiveness on larger language models (e.g., Qwen2-72B (Hui et al., 2024)) remains unverified, leaving its scalability to state-of-the-art architectures as open research questions.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 62376245), the Key Research and Development Program of Zhejiang Province, China (No. 2024C01034), the Fundamental Research Funds for the Central Universities (226-2024-00170), National Key Research and Development Project of China (No. 2018AAA0101900)and MOE Engineering Research Center of Digital Library.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Zhenbiao Cao, Yuanlei Zheng, Zhihao Fan, Xiaojin Zhang, Wei Chen, and Xiang Bai. 2024. RSL-SQL: Robust Schema Linking in Text-to-SQL Generation. *arXiv preprint*. ArXiv:2411.00073 [cs].

Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, lu Chen, Jinshu Lin, and Dongfang Lou. 2023. C3: Zero-shot Text-to-SQL with ChatGPT. *arXiv preprint*. ArXiv:2307.07306 [cs].

Ben Eyal, Amir Bachar, Ophir Haroche, Moran Mahabi, and Michael Elhadad. 2023. Semantic decomposition of question and sql for text-to-sql parsing. *arXiv preprint arXiv:2310.13575*.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024a. Text-to-sql empowered by large language models: A benchmark evaluation. *Proc. VLDB Endow.*, 17(5):1132–1145.

Yingqi Gao, Yifu Liu, Xiaoxia Li, Xiaorong Shi, Yin Zhu, Yiming Wang, Shiqi Li, Wei Li, Yuntao Hong, Zhiling Luo, et al. 2024b. Xiyan-sql: A multi-generator ensemble framework for text-to-sql. *arXiv preprint arXiv:2411.08599*.

Satya Krishna Gorti, Ilan Gofman, Zhaoyan Liu, Jiapeng Wu, Noël Vouitsis, Guangwei Yu, Jesse C. Cresswell, and Rasa Hosseinzadeh. 2024. MSc-SQL: Multi-Sample Critiquing Small Language Models For Text-To-SQL Translation. *arXiv preprint*. ArXiv:2410.12916.

Jiaqi Guo, Ziliang Si, Yu Wang, Qian Liu, Ming Fan, Jian-Guang Lou, Zijiang Yang, and Ting Liu. 2021. Chase: A Large-Scale and Pragmatic Chinese Dataset for Cross-Database Context-Dependent Text-to-SQL. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2316–2331, Online. Association for Computational Linguistics.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. 2025. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation. In *COLING*, pages 337–353. Association for Computational Linguistics.

Fei Li and Hosagrahar V Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, et al. 2025. Omnisql: Synthesizing high-quality text-to-sql data at scale. *arXiv preprint arXiv:2503.02240*.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13067–13075. Number: 11.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024. CodeS: Towards Building Open-source Language Models for Text-to-SQL. *arXiv preprint*. ArXiv:2402.16347 [cs].

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023b. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. *arXiv preprint*. ArXiv:2305.03111 [cs].

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. 2023. A comprehensive evaluation of ChatGPT's zero-shot Text-to-SQL capability. *arXiv preprint*. ArXiv:2303.13547 [cs].

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Xiping Liu and Zhao Tan. 2024. Epi-sql: Enhancing text-to-sql translation with error-prevention instructions. *arXiv preprint arXiv:2404.14453*.

Ruilin Luo, Liyuan Wang, Binghuai Lin, Zicheng Lin, and Yujiu Yang. 2024. Ptd-sql: Partitioning and targeted drilling with llms in text-to-sql. *arXiv preprint arXiv:2409.14082*.

Shuai Lyu, Haoran Luo, Zhonghong Ou, Yifan Zhu, Xiaoran Shang, Yang Qin, and Meina Song. 2025. Sql-o1: A self-reward heuristic dynamic search method for text-to-sql. *arXiv preprint arXiv:2502.11741*.

Peixian Ma, Xialie Zhuang, Chengjin Xu, Xuhui Jiang, Ran Chen, and Jian Guo. 2025. Sql-r1: Training natural language to sql reasoning model by reinforcement learning. *arXiv preprint arXiv:2504.08600*.

Karime Maamari, Fadhil Abubaker, Daniel Jaroslawicz, and Amine Mhedhbi. 2024. The Death of Schema Linking? Text-to-SQL in the Age of Well-Reasoned Language Models. *arXiv preprint*. ArXiv:2408.07702 [cs].

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157.

Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan Ö. Arik. 2024. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql. *CoRR*, abs/2410.01943.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. *arXiv preprint*. ArXiv:2304.11015 [cs].

Mohammadreza Pourreza and Davood Rafiei. 2024. DTS-SQL: Decomposed Text-to-SQL with Small Large Language Models. *arXiv preprint*. ArXiv:2402.01117 [cs].

Mohammadreza Pourreza, Shayan Talaei, Ruoxi Sun, Xingchen Wan, Hailong Li, Azalia Mirhoseini, Amin Saberi, Sercan Arik, et al. 2025. Reasoning-sql: Reinforcement learning with sql tailored partial rewards for reasoning-enhanced text-to-sql. *arXiv preprint arXiv:2503.23157*.

Yang Qin, Chao Chen, Zhihang Fu, Ze Chen, Dezhong Peng, Peng Hu, and Jieping Ye. 2024. ROUTE: Robust Multitask Tuning and Collaboration for Text-to-SQL. *arXiv preprint*. ArXiv:2412.10138 [cs].

Ge Qu, Jinyang Li, Bowen Li, Bowen Qin, Nan Huo, Chenhao Ma, and Reynold Cheng. 2024. Before generation, align it! a novel and effective strategy for mitigating hallucinations in text-to-sql generation. In *ACL (Findings)*, pages 5456–5471. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the Text-to-SQL Capabilities of Large Language Models. *arXiv preprint*. ArXiv:2204.00498 [cs].

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. *arXiv preprint*. ArXiv:2109.05093 [cs].

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. 2023. TaskBench: Benchmarking Large Language Models for Task Automation. *arXiv preprint*. ArXiv:2311.18760 [cs].

Ruoxi Sun, Sercan Ö Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, et al. 2023. Sql-palm: Improved large language model adaptation for text-to-sql (extended). *arXiv preprint arXiv:2306.00739*.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. 2023. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. 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*, pages 7567–7578, Online. Association for Computational Linguistics.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Qian-Wen Zhang, Zhao Yan, and Zhoujun Li. 2023. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. *arXiv preprint*. ArXiv:2312.11242 [cs].

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Wenxuan Xie, Gaochen Wu, and Bowen Zhou. 2024. Mag-sql: Multi-agent generative approach with soft schema linking and iterative sub-sql refinement for text-to-sql. *CoRR*, abs/2408.07930.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Jiaxi Yang, Binyuan Hui, Min Yang, Jian Yang, Junyang Lin, and Chang Zhou. 2024b. Synthesizing Text-to-SQL Data from Weak and Strong LLMs. *arXiv preprint*. ArXiv:2408.03256 [cs].

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2024. A Survey of Table Reasoning with Large Language Models. *arXiv preprint*. ArXiv:2402.08259 [cs].

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-sql with distilled test suites. *arXiv preprint arXiv:2010.02840*.