IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511145

Literature Survey: Latency and Cost Optimization of a Compound AI System for Text-to-SQL Generation

Aditya Dudhagi[1], Pranav Mali[2], Naren Manekari[3], Pushkar Takale[4], Pranjali Joshi[5]

Department of Computer Engineering, PICT Pune, India[1-5]

Abstract: Text-to-SQL translation has advanced rapidly with the integration of Large Language Models (LLMs). However, as enterprise databases scale in size and structural complexity, single-pass query generation consistently hits three bottlenecks: context window saturation from excessive schema text, high inference latency, and the hallucination of non-existent tables or columns.

This survey examines recent architectural solutions to these bottlenecks, categorizing them into schema pruning, task decomposition, and execution-guided refinement. We trace the field's evolution from early graph-based schema encoders to modern adaptive pipelines that route queries based on their computational difficulty. Using DecoSearch as a baseline, we also analyze an extended framework that integrates confidence-gated schema pruning and graph-based join path validation, demonstrating how these mechanisms lower inference costs and reduce execution delay in production-scale environments.

Text-to-SQL, Large Language Models, Schema Pruning, Task Decomposition, Complexity-Aware Routing

1. INTRODUCTION

Interfaces that let people query databases in plain language have developed rapidly, helped in large part by public benchmarks such as Spider (**b1?**), BIRD (**b2?**), and Spider 2.0 (**b3?**). Recent surveys organize the growing body of LLM-based Text-to-SQL research and identify persistent challenges in grounding, evaluation, and deployment (**b24?**; **b25?**). LLMs perform well when translating questions to SQL with few or no examples, but their accuracy tends to fall when the target database has hundreds of tables. Placing an entire schema in the context window spreads the model's attention over a great deal of irrelevant text, slows responses, and encourages queries that refer to tables or joins that are not really there (**b6?**), (**b8?**).

To address these bottlenecks, current research has diverged into several specialized optimization vectors: schema linking and pruning, complexity-based query routing, task decomposition, and execution-guided feedback.

Parallel efforts emphasize the context layer. Studies like DAIL-SQL demonstrate how query representation and in-context example selection directly drive model performance (**b20?**). Meanwhile, multi-agent architectures like MAC-SQL distribute the workload, orchestrating decomposition, schema retrieval, and iterative SQL refinement across concurrent agents (**b21?**).

Because this domain evolves rapidly, several systems analyzed here—including DecoSearch (**b9?**)—are recent preprints. To maintain a rigorous perspective, this survey grounds these emerging frameworks alongside established methodologies from peer-reviewed NLP and database venues. This approach clearly maps the trajectory from accepted baselines to cutting-edge architectural proposals.

2. SCHEMA LINKING AND PRUNING STRATEGIES

The initial methods of establishing a connection between the schema were based on the semantic parsing and the graph-based neural encoder that is used to match question words with DB elements. The RAT-SQL (**b4?**) proposed a relation aware schema encoder and IGSQL (**b5?**) introduced an interaction graph to make it possible to interpret the schema elements in the context of the previous dialogue.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511145

With the rise of LLM as the standard backbones, the focus shifted from schema encoding to pruning what the model can see. Enterprise-focused benchmarks such as BEAVER emphasize that realistic warehouse schemas and business questions remain difficult even when standard prompting and retrieval-augmented generation are used (**b22?**). GenLink further explores generation-driven schema linking with multi-model learning (**b23?**). CHESS (**b6?**) introduced LLM-based schema pruning as a standalone step in its agent pipeline. RSL-SQL (**b7?**) applied pruning in both directions, aiming to cut a large share of the input tokens while keeping recall high. Agrawal and Indukuri (**b8?**) took a retrieval view with Schema-First Retrieval, which builds its index over typed catalog entries such as tables, metrics, and relationships, not over raw warehouse rows. Their position is that picking the right catalog entries is a basic retrieval problem.

Other recent work seeks to place pruning on a more principled footing through mathematical and graph-based constraints. In this direction, our approach extends the DecoSearch design (**b9?**) with a confidence-gated routing engine, aimed at the difficulty of pruning a schema before it is known which tables a question will need. The engine runs BM25 (**bm25?**) and dense bi-encoder retrieval (**dpr?**) in parallel and merges their ranked lists with Reciprocal Rank Fusion (RRF) (**rrf?**). Because pruning can drop tables that the question never mentions but that are needed to connect other tables, the engine restores such bridge tables by building a Steiner tree over the foreign-key graph, using the Kou–Markowsky–Berman (KMB) 2-approximation (**kmb?**). It then scores competing join paths and treats the gap between the best candidates, called the subgraph margin, as a rule-based confidence signal: a large margin leads to direct SQL generation, whereas a small margin escalates the question to an interactive recovery agent. The intended effect is to keep prompts compact and to reserve the added latency of recovery for ambiguous questions, which should lower token cost and delay, although this benefit still has to be confirmed experimentally.

Fig. 1 gives a high-level view of the resulting pipeline. In Stage 1, the pruner narrows the schema to a compact subgraph S_1 and measures its confidence through the subgraph margin Δ. If Δ reaches the threshold τ and the subgraph is sufficiently confident, S_1 is passed to DecoSearch, which the figure treats as a single generation-and-verification module so that the diagram stays focused on the added components. Otherwise, the question is handed to Spider-Agent (**b3?**), an interactive agent for exploring the database, with the Stage-1 tables retained as its starting context.
High-level architecture of the proposed extension.

3. TASK DECOMPOSITION AND COMPLEXITY-AWARE ROUTING

Queries that need several joins or nested logic often fail when the model must write them in a single step. Decomposition-based approaches handle this by splitting the question into sub-problems that are easier to solve. DIN-SQL (**b11?**) was an early example of decomposed in-context learning, assigning schema linking, classification of the query, and SQL writing to separate stages. DEA-SQL (**b12?**) likewise arranges generation as an ordered workflow, which keeps the model from losing focus across a long prompt.
More recently, attention has turned to complexity-aware routing, in which the compute spent on a question grows with its difficulty. Guo et al. (**b13?**) evaluated test-time scaling strategies for Text-to-SQL and found that approaches such as routing and decomposition generally beat plain parallel sampling. EllieSQL (**b14?**) predicts how hard a question is and sends it to a generation pipeline of matching cost, which the authors report brings sizable efficiency gains.
DecoSearch (**b9?**) brings these ideas together with an escalation strategy led by a judger. A light classifier decides whether a question is simple enough for a direct path or whether it should be expressed as a directed acyclic graph (DAG) of sub-queries. In our extension, this judger is paired with the confidence-gated pruner described in Section II, so that the extra latency of DAG decomposition and common table expression (CTE) assembly is paid only for the hardest and most ambiguous questions.

4. EXECUTION-GUIDED REFINEMENT

the result of running a query back to the model has become a common way to let Text-to-SQL systems correct themselves. Self-Refine (**b15?**) and CRITIC (**b17?**) showed that LLMs can improve their own outputs over several rounds when given feedback such as error traces or tool outputs. MAGIC (**b16?**) focuses on Text-to-SQL and automatically produces guidelines for self-correction that are supplied as in-context examples.
Execution results are also useful for picking among candidates. Borchmann and Wydmuch (**b18?**) argue in Query and Conquer that this lets smaller open-source models approach the accuracy of larger proprietary ones. ExCoT (**b19?**) improves reasoning by fine-tuning with supervision from execution outcomes.
DecoSearch (**b9?**) adds plan-level repair on top of these ideas. When a sub-query keeps failing, a Topology Refiner rebuilds the reasoning DAG, instead of the system merely correcting syntax. Our extension additionally verifies candidate queries in a local SQLite environment, which keeps each feedback round inexpensive and gives a closed loop

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511145

in which generation, execution, and correction feed one another. If the retry budget is exhausted before a query executes and returns rows, our extension falls back to Spider-Agent (**b3?**), as shown in Fig. 1.

5. COMPARATIVE ANALYSIS

The studies reviewed above address related but distinct sources of Text-to-SQL error and inefficiency. Schema-pruning methods aim to reduce prompt size while preserving the tables and relationships needed to form a valid query. Complexity-aware routing allocates more expensive reasoning to difficult questions, while execution-guided refinement uses database feedback to identify and repair mistakes. Tables [tab:comparison-pruning]–[tab:comparison-accuracy] summarize selected reported results across these areas. The studies use different models, benchmark splits, and evaluation settings; the values should therefore be interpreted as reported results rather than as a controlled ranking. Benchmark quality is another consideration: analyses of BIRD-Bench have shown that noise in reference SQL can affect conclusions drawn from evaluation (**b26?**). Context-dependent datasets such as MultiSQL also broaden evaluation across diverse SQL operations and conversational schema use (**b27?**). Schema recall, token reduction, and execution accuracy measure different properties and should not be treated as interchangeable metrics.
@>
>
p0.22 >
p0.32 X@
Method & **Reported result** & **Comparative interpretation** CHESS (**b6?**) & Approximately 5 × fewer LLM input tokens; an accuracy improvement of about 2% is reported. & Suggests that schema pruning can reduce prompt size while maintaining or improving performance. The exact baseline and whether the change is relative or in percentage points should be checked in the original paper.
RSL-SQL (**b7?**) & 94% strict schema-linking recall and 83% fewer input columns. & Reports high recall while substantially reducing the number of columns supplied to the model.
RSL-SQL (**b7?**) & Reported execution accuracy of 67.2% on BIRD and 87.9% on Spider in the cited GPT-4o setup. & These are end-to-end SQL results, not schema-recall measurements; benchmark split and evaluation protocol matter.
CHESS (**b6?**) & Reported execution accuracy of 65.0% on BIRD development, 66.69% on BIRD test, and 87.2% on Spider test in the cited setup. & Results reflect a particular model configuration and are not a controlled comparison with RSL-SQL.

Table [tab:comparison-pruning] includes both schema-level measures and end-to-end execution accuracy. High schema recall is useful, but does not by itself guarantee that generated SQL is correct. Similarly, a high execution score does not directly indicate how much schema context was removed.
@>
>
p0.24 >
p0.38 X@
Method & **Reported result** & **Comparative interpretation**
EllieSQL (**b14?**) & Reports more than 40% fewer tokens while maintaining performance, compared with always using its most advanced generation pipeline. & The claim is relative to EllieSQL's stated baseline; it is not a direct token-cost comparison with DecoSearch.
DecoSearch (**b9?**) & Reports 70.53% execution accuracy on BIRD and 88.31% on Spider with the stated DeepSeek-based setup; the paper also describes reduced token use relative to its baselines. & Accuracy and token savings depend on the model, benchmark, and baseline. The original paper should be consulted for the exact token-reduction comparison.
Rethinking Agentic Workflows (**b13?**) & Evaluates routing, decomposition, and other test-time scaling strategies for Text-to-SQL. & A numeric comparison should be included only when the metric, benchmark, and baseline are clearly specified.

The reported results in Table [tab:comparison-routing] suggest that routing can reduce unnecessary computation. Token savings are meaningful only when the reference baseline is clear. For a fair comparison, latency, token use, and execution accuracy should ideally be measured on the same questions using the same underlying model and evaluation procedure.
@>
>
p0.26 >p0.18 X@

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511145

Method & **Reported Spider test accuracy** & **Setup and interpretation**
DIN-SQL (**b11?**) & 85.3% & Reported with GPT-4. The method decomposes generation into stages and includes self-correction.
CHESS (**b6?**) & 87.2% & Reported for the configuration cited in the survey; CHESS is a multi-stage system, not a refinement-only baseline.
RSL-SQL (**b7?**) & 87.9% & Reported with GPT-4o in the cited setup. Its main focus is robust schema linking, so this is an end-to-end reference point rather than a pure refinement result.

Table [tab:comparison-accuracy] provides context for reported Spider test results, but the methods do not isolate the same component. The scores should not be taken as proof that one method is inherently better. A more direct evaluation of refinement would report accuracy before and after feedback, the number of repair attempts, and the additional tokens or latency required.

Paper / approach	Main focus	Research gap / opportunity
Schema-First Retrieval (**b8?**) (2026)	Retrieval over structured catalog objects such as tables, metrics, and relationships.	Further study is needed on how catalog retrieval integrates with complexity-aware routing, SQL generation, and execution-based correction in one end-to-end evaluation.
DecoSearch (**b9?**) (2026)	Complexity-aware routing, query decomposition, and plan-level repair.	Its overall benefits should be assessed through reproducible comparisons that jointly measure execution accuracy, latency, and token cost against clearly defined baselines.
EllieSQL (**b14?**) (2025)	Routes questions to generation pipelines according to estimated complexity.	The trade-off between routing overhead, token savings, response latency, and accuracy should be measured under the same evaluation setup as alternative methods.
Rethinking Agentic Workflows (**b13?**) (2025)	Evaluates routing, decomposition, and test-time scaling strategies.	Results across different strategies can be difficult to compare when models, benchmarks, and cost measures differ; standardized evaluation would strengthen conclusions.
ExCoT (**b19?**) (2025)	Uses execution feedback to supervise Text-to-SQL reasoning.	Further evaluation can quantify the additional training and inference requirements and determine how well the approach generalizes across database schemas.
Query and Conquer (**b18?**) (2025)	Uses execution signals to guide SQL candidate generation and selection.	The cost of executing and selecting among candidates should be compared with the accuracy gains, including total latency and resource use.
MAGIC (**b16?**) (2025)	Generates self-correction guidelines for in-context Text-to-SQL.	The added cost of producing and using correction guidelines should be evaluated alongside the reliability of the resulting SQL across different query types.
RSL-SQL (**b7?**) (2024)	Robust schema linking and pruning to reduce irrelevant schema context.	Schema reduction must preserve columns and relationships needed for valid joins; token reduction, schema recall, latency, and execution accuracy should be considered together.
CHESS (**b6?**) (2024)	LLM-based schema pruning in a multi-stage Text-to-SQL pipeline.	Its reported efficiency and accuracy gains should be compared with other methods under consistent models, benchmark splits, and end-to-end latency measurements.
DEA-SQL (**b12?**) (2024)	Structures SQL generation as a staged workflow.	Additional stages may improve reasoning but can increase latency; the benefit of decomposition should be measured against its total inference cost.
CRITIC (**b17?**) (2024)	Uses tool-interactive critique to support self-correction.	The number of feedback rounds and their contribution to accuracy should be weighed against extra calls, latency, and token consumption.
DIN-SQL (**b11?**) (2023)	Decomposes Text-to-SQL into stages and incorporates self-correction.	Multi-stage generation may add inference overhead; further controlled evaluation can isolate the contribution of each stage and its cost.
Self-Refine (**b15?**) (2023)	Iteratively improves outputs using model-generated feedback.	The effectiveness and cost of repeated refinement need to be established for database-grounded SQL tasks, where execution correctness is essential.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511145

Paper / approach	Main focus	Research gap / opportunity
IGSQL (**b5?**) (2020)	Uses schema interaction graphs for context-dependent Text-to-SQL.	Modeling schema relationships does not directly address the prompt-size and inference-cost challenges of large LLM-based enterprise workflows.
RAT-SQL (**b4?**) (2020)	Relation-aware schema encoding and linking.	Graph-based schema linking provides structural information but does not, by itself, optimize LLM context selection, adaptive routing, or execution-guided repair.

6. CONCLUSION

The literature shows a gradual move away from fixed graph embeddings of the schema and toward adaptive, multi-agent reasoning pipelines. Across the published work, a common message is that schema linking, task decomposition, and execution-based correction work better when designed together than when treated as separate problems, particularly for enterprise databases. Systems that combine complexity-aware routing with graph-constrained schema pruning, such as the confidence-gated extension of DecoSearch discussed here, are a promising direction for balancing the structural correctness of the database graph against the cost of LLM inference. Because several of these systems are recent preprints, their reported gains still need independent evaluation under shared benchmarks and settings.

REFERENCES

[1]. T. Yu, R. Zhang, K. Yang, et al., "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.
[2]. J. Li, B. Hui, G. Qu, et al., "Can LLM already serve as a database interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023.
[3]. F. Lei, J. Chen, Y. Ye, et al., "Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows," in *International Conference on Learning Representations (ICLR)*, 2025.
[4]. B. Wang, R. Shin, X. Liu, et al., "RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.
[5]. Y. Cai and X. Wan, "IGSQL: Database Schema Interaction Graph Based Neural Model for Context-Dependent Text-to-SQL Generation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
[6]. S. Talaei, M. Pourreza, Y.-C. Chang, et al., "CHESS: Contextual Harnessing for Efficient SQL Synthesis," *arXiv preprint arXiv:2405.16755*, 2024.
[7]. Z. Cao, Y. Zheng, Z. Fan, et al., "RSL-SQL: Robust Schema Linking in Text-to-SQL Generation," *arXiv preprint arXiv:2411.00073*, 2024. A. Agrawal and S. Indukuri, "Schema-First Retrieval: Embedding Catalogs for Natural Language Analytics," *arXiv preprint arXiv:2606.28387*, 2026.
[8]. E. Schafir, X. Zheng, H. A. Salehi, et al., "DecoSearch: Complexity-Aware Routing and Plan-Level Repair for Text-to-SQL," *arXiv preprint arXiv:2606.17821*, 2026.
[9]. S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.
[10]. V. Karpukhin, B. Oğuz, S. Min, et al., "Dense Passage Retrieval for Open-Domain Question Answering," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
[11]. G. V. Cormack, C. L. A. Clarke, and S. Büttcher, "Reciprocal rank fusion outperforms Condorcet and individual rank learning methods," in *Proc. ACM SIGIR*, 2009, pp. 758–759.
[12]. L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, no. 2, pp. 141–145, 1981.
[13]. M. Pourreza and D. Rafiei, "DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Y. Xie, X. Jin, T. Xie, et al., "Decomposition for Enhancing Attention: Improving LLM-based Text-to-SQL through Workflow Paradigm," *arXiv preprint arXiv:2402.10671*, 2024.
[14]. J. Guo, K. Patel, J. P. Ono, et al., "Rethinking Agentic Workflows: Evaluating Inference-Based Test-Time Scaling Strategies in Text-to-SQL Tasks," *arXiv preprint arXiv:2510.10885*, 2025.
[15]. Y. Zhu, R. Jiang, B. Li, N. Tang, and Y. Luo, "EllieSQL: Cost-Efficient Text-to-SQL with Complexity-Aware Routing," *arXiv preprint arXiv:2503.22402*, 2025.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511145

A. Madaan, N. Tandon, P. Gupta, et al., "Self-refine: Iterative refinement with self-feedback," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023.

[16]. Askari, C. Poelitz, and X. Tang, "MAGIC: Generating Self-Correction Guideline for In-Context Text-to-SQL," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

[17]. Z. Gou, Z. Shao, Y. Gong, et al., "Critic: Large language models can self-correct with tool-interactive critiquing," in *International Conference on Learning Representations (ICLR)*, 2024.

[18]. Ł. Borchmann and M. Wydmuch, "Query and Conquer: Execution-Guided SQL Generation," *arXiv preprint arXiv:2503.24364*, 2025. B. Zhai, C. Xu, Y. He, and Z. Yao, "ExCoT: Optimizing Reasoning for Text-to-SQL with Execution Feedback," *arXiv preprint arXiv:2503.19988*, 2025.

[19]. L. Nan, Y. Zhao, W. Zou, R. Ni, J. Tae, E. Zhang, A. Cohan, and D. Radev, "Enhancing Text-to-SQL Capabilities of Large Language Models: A Study on Prompt Design Strategies," in *Findings of the Association for Computational Linguistics: EMNLP*, pp. 14935–14956, 2023.

[20]. Wang, C. Ren, J. Yang, X. Liang, J. Bai, L. Chai, Z. Yan, Q.-W. Zhang, D. Yin, X. Sun, and Z. Li, "MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL," in *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, pp. 540–557, 2025.

[21]. P. B. Chen, F. Wenz, Y. Zhang, D. Yang, J. Choi, N. Tatbul, M. Cafarella, C. Demiralp, and M. Stonebraker, "BEAVER: An Enterprise Benchmark for Text-to-SQL," *arXiv preprint arXiv:2409.02038*, 2024.

[22]. Z. Hao, J. Huang, S. Shi, R. Cai, and B. Xu, "GenLink: Generation-Driven Schema-Linking via Multi-Model Learning for Text-to-SQL," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 29892–29905, 2025.

[23]. X. Liu, S. Shen, B. Li, P. Ma, R. Jiang, Y. Zhang, J. Fan, G. Li, N. Tang, and Y. Luo, "A Survey of Text-to-SQL in the Era of LLMs: Where Are We, and Where Are We Going?" *IEEE Transactions on Knowledge and Data Engineering*, vol. 37, no. 10, pp. 5735–5754, 2025, doi: 10.1109/TKDE.2025.3592032.

[24]. L. Shi, Z. Tang, N. Zhang, X. Zhang, and Z. Yang, "A Survey on Employing Large Language Models for Text-to-SQL Tasks," *ACM Computing Surveys*, vol. 58, no. 2, Art. no. 54, 2025, doi: 10.1145/3737873.

[25]. N. Wretblad, F. Riseby, R. Biswas, A. Ahmadi, and O. Holmström, "Understanding the Effects of Noise in Text-to-SQL: An Examination of the BIRD-Bench Benchmark," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 356–369, 2024, doi: 10.18653/v1/2024.acl-short.34.

[26]. Li, Y. Wang, Z. Wu, Z. Yu, F. Zhao, S. Huang, and X. Dai, "MultiSQL: A Schema-Integrated Context-Dependent Text2SQL Dataset with Diverse SQL Operations," in *Findings of the Association for Computational Linguistics: ACL*, pp. 13857–13867, 2024.

 This work is licensed under a Creative Commons Attribution 4.0 International License