GenLink: Generation-Driven Schema-Linking via Multi-Model Learning for Text-to-SQL

Zhifeng Hao[1,2], **Junqi Huang**[1], **Shaobin Shi**[1], **Ruichu Cai**[1,3], **Boyan Xu**[1*]
[1]School of Computer Science, Guangdong University of Technology
[2]College of Science, Shantou University
[3]Peng Cheng Laboratory
haozhifeng@stu.edu.cn
{jqhuang018,d7inshi,cairuichu,hpakyim}@gmail.com

Abstract

Schema linking is widely recognized as a key factor in improving text-to-SQL performance. Supervised fine-tuning approaches enhance SQL generation quality by explicitly fine-tuning schema linking as an extraction task. However, they suffer from two major limitations: (i) The training corpus of small language models restricts their cross-domain generalization ability. (ii) The extraction-based fine-tuning process struggles to capture complex linking patterns. To address these issues, we propose **GenLink**, a generation-driven schema-linking framework based on multi-model learning. Instead of explicitly extracting schema elements, GenLink enhances linking through a generation-based learning process, effectively capturing implicit schema relationships. By integrating multiple small language models, GenLink improves schema-linking recall rate and ensures robust cross-domain adaptability. Experimental results on the BIRD and Spider benchmarks validate the effectiveness of GenLink, achieving execution accuracies of 67.34% (BIRD), 89.7% (Spider development set), and 87.8% (Spider test set), demonstrating its superiority in handling diverse and complex database schemas. Our implementation will be open-sourced at https://github.com/DMIRLAB-Group/GenLink.

1 Introduction

Text-to-SQL aims to convert natural language queries into SQL based on a given database schema(Gao et al., 2023; Qin et al., 2022; Deng et al., 2022). Benchmarks like Spider(Yu et al., 2019) and BIRD(Li et al., 2023b) evaluate cross-domain generalization, where training and test databases are disjoint. Schema-linking is widely recognized as key to improving generalization by adaptively associating queries with database items.

[*]Corresponding author, hpakyim@gmail.com

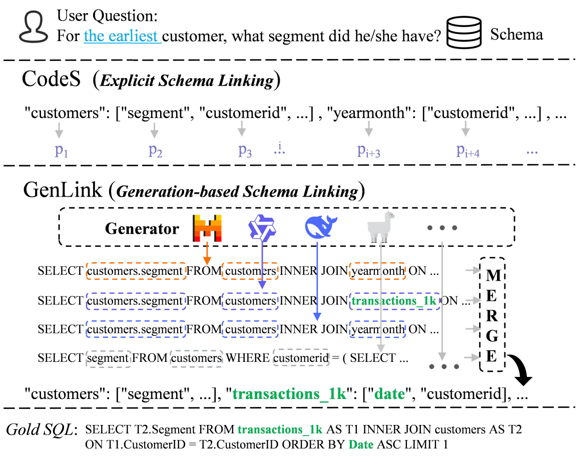

User Question:
For the earliest customer, what segment did he/she have?
Schema
CodeS (Explicit Schema Linking)
"customers": ["segment", "customerid", ...] , "yearmonth": ["customerid", ...] , ...
p1
p2
p3
pi+3
pi+4
...
GenLink (Generation-based Schema Linking)
Generator
SELECT customers.segment FROM customers INNER JOIN yearmonth ON ...
SELECT customers.segment FROM customers INNER JOIN transactions_1k ON ...
SELECT customers.segment FROM customers INNER JOIN yearmonth ON ...
SELECT segment FROM customers WHERE customerid = (SELECT ...
MERGE
"customers": ["segment", ...], "transactions_1k": ["date", "customerid], ...
Gold SQL: SELECT T2.Segment FROM transactions_1k AS T1 INNER JOIN customers AS T2 ON T1.CustomerID = T2.CustomerID ORDER BY Date ASC LIMIT 1

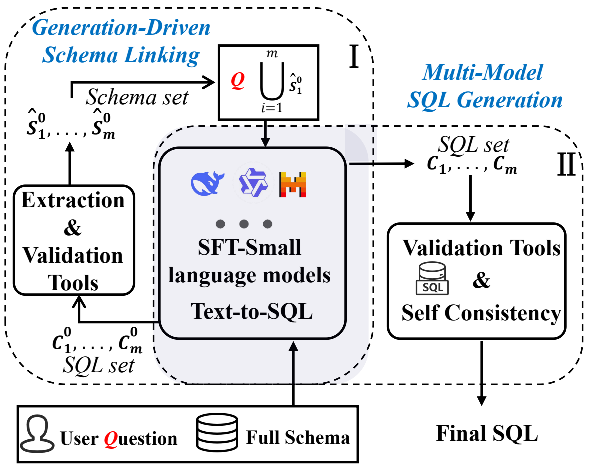
Figure 1: An example of extracting implicit schema on the BIRD dataset.

Leveraging the reasoning capabilities of large language models (LLMs)(Zhu et al., 2024), they have become the dominant approach in text-to-SQL. In-context learning-based methods, such as CHESS (Talaei et al., 2024) and MCS-SQL (Lee et al., 2024), explicitly incorporate schema-linking by extracting database items via LLMs, significantly improving SQL generation quality. However, their high token cost remains a major limitation.

To overcome this, supervised fine-tuning (SFT) approaches train LLMs on labeled Text-to-SQL data. Similarly, explicitly training schema linking in SFT methods, as seen in DTS-SQL (Pourreza and Rafiei, 2024) and CodeS (Li et al., 2024), has proven effective in enhancing performance. However, these solutions face two major limitations: (i) Their schema-linking extraction performance is constrained by model size and training corpus, making it difficult to handle samples with significant domain differences. (ii) The explicit extraction and fine-tuning process limits the model's ability to capture complex linking patterns.

To address these limitations, we propose GenLink, a generation-driven schema-linking framework based on multi-model learning. Instead of relying on explicit schema extraction, GenLink enhances schema linking through generation-based learning, enabling more robust alignment between queries and database schemas. As illustrated in Figure 1, GenLink successfully links the implicit schema "*transactions_1k.date*". By integrating multiple small language models, GenLink has strong cross-domain adaptation capabilities, ensuring a high recall rate for schema linking. Experimental results on two authoritative benchmarks, BIRD and Spider, validate the effectiveness of GenLink, achieving execution accuracies of 67.34% (BIRD), 89.7% (Spider development set), and 87.8% (Spider test set), substantiating its effectiveness across diverse database domains.

2 Methodology



Generation-Driven Schema Linking
I
Schema set
$Q \bigcup_{i=1}^{m} \hat{S}_i^0$
$\hat{S}_1^0, \ldots, \hat{S}_m^0$
Extraction & Validation Tools
SFT-Small language models Text-to-SQL
$C_1^0, \ldots, C_m^0$ SQL set
Multi-Model SQL Generation
SQL set $C_1, \ldots, C_m$
II
Validation Tools & Self Consistency
User Question
Full Schema
Final SQL

Figure 2: GenLink pipeline.

GenLink is a multi-stage, multi-model learning framework based on supervised fine-tuning. It aims to achieve implicit schema linking and SQL generation by integrating various generative language models. As shown in Figure 2, the framework comprises two key modules: *Generation-Driven Schema Linking* and *Multi-Model SQL Generation*. Specifically, as the core component, the Generation-Driven Schema Linking module strategically integrates multiple generative language models to independently generate SQL queries and extract relevant schema information. Following this, it systematically consolidates the verified information using an advanced merging mechanism, ensuring the accuracy and robustness of schema linking. The Multi-Model SQL Generation module introduces a self-consistency(Wang et al., 2023) mechanism, further optimizing SQL generation performance based on the merged schema representation generated by the Generation-Driven Schema Linking module. This hierarchical architecture makes full use of the diversity in training corpora across different language models, significantly enhancing the framework's ability to extract implicit schema links and effectively addressing domain generalization issues.

2.1 Generation-Driven Schema Linking

Generation-Driven Schema Linking(GDSL) module significantly enhances the accuracy of schema linking by integrating the strengths of multiple small generative language models and leveraging the SQL generation task to indirectly infer implicit schema linking. Specifically, although these generative language models are all based on similar Decoder architectures and have acquired language comprehension capabilities through training on large-scale datasets, there are notable differences in their understanding of schema semantics and their ability to handle complex queries in SQL generation tasks due to variations in pre-training corpora. By harmonizing the consistency and diversity among these models, the module effectively improves both the recall and precision of schema extraction.

In terms of implementation methodology, given a user question $\mathcal{Q}$, a complete database schema $\mathcal{S}_{full}$, and the corresponding ground truth SQL statement $\mathcal{C}^{Gold}$, we first train a series of SQL generation models $LM_i(i \in [1, ..., m])$ using the prepared training set $(\mathcal{Q}, \mathcal{S}_{full}) \rightarrow \mathcal{C}^{Gold}$. During the inference phase, the trained models generate an initial set of candidate SQL queries:

$$\mathcal{C}_i^0 = LM_i(\mathcal{Q}, \mathcal{S}_{full})$$

Subsequently, a SQL parsing tool is used to extract the set of schema items $\mathcal{S}_i^0$ from $\mathcal{C}_i^0$. Each schema item in $\mathcal{S}_i^0$ is then validated to determine whether it exists in $\mathcal{S}_{full}$, a process denoted as $\mathcal{V}$. The verified schema set is then given by $\hat{\mathcal{S}}_i^0 = \mathcal{V}(\mathcal{S}_{full}, \mathcal{S}_i^0)$. The aforementioned steps are applied to all language models respectively, resulting in multiple validated sets of schema items. Finally, these sets are merged into a unified set of schema items $\bar{\mathcal{S}}$, expressed as:

$$\bar{\mathcal{S}} = \bigcup_{i=1}^{m} \hat{\mathcal{S}}_i^0$$

2.2 Multi-Model SQL Generation

After GDSL module extracts both explicit and implicit schema items related to the natural language question, the core task of the Multi-Model SQL Generation(MMSG) module is to generate a set of high-quality SQL queries and filter out the final result. The diversity among multiple language models is reflected in their different utilization of the schema, thereby generating diverse SQL structures, especially in complex SQL generation scenarios. When combined with the self-consistency technique, this approach demonstrates significant potential. This strategy not only enhances the robustness of the framework but also significantly improves the accuracy of SQL generation.

In the specific implementation, we feed $\mathcal{Q}$ and $\bar{\mathcal{S}}$ into multiple generation models to obtain SQL queries $\mathcal{C}_i$ again:

$$\mathcal{C}_i = LM_i(\mathcal{Q}, \bar{\mathcal{S}})$$

Subsequently, we further refine the query result sets by applying the self-consistency method for voting-based filtering, obtaining the verified query set $\hat{\mathcal{C}}(\hat{\mathcal{C}} \subseteq \mathcal{C})$. We define $\mathcal{G} = \{G_1, ..., G_n\}$, where each G_k represents a group with the same query result set. For each $G_k \in \mathcal{G}$, the number of query results it contains is represented by $|G_k|$. Therefore, the group with the largest number of results is $G_{max} = \arg\max_{G_k \in \mathcal{G}} |G_k|$. Then, from G_{max}, we can obtain the corresponding query set $\bar{\mathcal{C}}(\bar{\mathcal{C}} \subseteq \hat{\mathcal{C}})$, from which we select the SQL query with the fastest execution speed through the optimization criterion: $\mathcal{C}_F = \arg\min_{\mathcal{C}_z \in \bar{\mathcal{C}}} T(\mathcal{C}_z)$ as the final output SQL query.

In addition, to optimize models selection under computational constraints, we consider the trade-off between execution accuracy and inference time across multiple models: for a given model LM_i, the inference time is defined as $\mathcal{T}_i = T(LM_i(\mathcal{Q}, S_{full}) + T(LM_i(\mathcal{Q}, \bar{\mathcal{S}}))$. During parallelized reasoning, the overall multiple models reasoning time $\mathcal{T}_{multi}$ is determined by the slowest model: $\mathcal{T}_{multi} = \max_{i \in \{1,...,m\}} \mathcal{T}_i$. To standardize efficiency measurements, we define inference time per sample (ITS) $t_{its} = \frac{\mathcal{T}_{multi}}{\mathcal{N}}$, where $\mathcal{N}$ represents the number of processed samples.

3 Experiments

3.1 Experimental Setup

Datasets and Evaluation Metrics In our experiments, we employed two cross-domain Text-to-SQL benchmark datasets, Spider (Yu et al., 2019) and BIRD (Li et al., 2023b), to evaluate the performance of GenLink framework.

For the evaluation of SQL generation, we adopt execution accuracy (EX) and Inference Time per Sample (ITS) as the primary metric, which takes into account both the execution accuracy and the reasoning efficiency simultaneously. In evaluating schema linking, we utilize a combination of recall and precision metrics. Specifically, recall is further broken down into Count-based Table Recall (TR), Count-based Column Recall (CR) and Sample-based Recall (SR); precision is subdivided into Count-based Table Precision (TP) and Count-based Column Precision (CP). This multidimensional evaluation system comprehensively reflects the model's performance in schema linking tasks.

Models Our experiments used the Llama-3.1-8B-Instruct, Qwen2.5-Coder-7B-Instruct, Qwen2.5-7B-Instruct, deepseek-coder-6.7b-instruct, and Mistral-7B-Instruct-v0.3 models, denoted by L, QC, Q, D, and M, respectively. And the model combinations used by GenLink default to $\{L, QC, Q, D, M\}$ unless otherwise specified.

3.2 Experimental Analysis

Main Results This experiment systematically evaluates the GenLink framework against in-context learning methods and supervised fine-tuning approaches. As shown in Table 1, on the BIRD dev set, GenLink demonstrates a 2.34% advantage in EX over the GPT-4-turbo-based ICL method CHESS$_{IR+SS+CG}$, while achieving more precise SQL generation with significantly fewer parameters, demonstrating the superior parameter efficiency of our approach. Regarding cross-domain generalization capability, GenLink outperforms the SFT baseline DTS-SQL + DeepSeek-7B by 4.2% and 3.4% EX on Spider validation and test sets respectively. Compared with SFT CodeS-15B, GenLink achieves a notable 8.87% EX performance gain on the BIRD dev set. In addition, under resource constraints, the use of lightweight model combinations $\{L,Q,D\}$ can significantly reduce the inference time with a slight reduction in EX. This performance breakthrough stems from the framework's innovative multi-model collaboration mechanism: Through stage-wise integration of specialized capabilities from different models in schema linking and SQL generation tasks, it effectively enhances the capture of implicit semantic relation-

Method	Bird Dev		Spider Dev		Spider Test	
	EX	ITS	EX	ITS	EX	ITS
In-Context Learning Methods						
DAIL-SQL + GPT-4(Gao et al., 2023)	54.76	-	84.4	-	86.6	-
MCS-SQL + GPT-4(Lee et al., 2024)	63.36	-	89.5	-	89.6	-
CHESS$_{IR+SS+CG}$ + GPT-4-turbo(Talaei et al., 2024)	65.0	-	-	-	87.2	-
Supervised Fine-Tuning Approaches						
DTS-SQL + DeepSeek 7B (Pourreza and Rafiei, 2024)	55.8	1.78	85.5	1.41	84.4	1.56
SFT CodeS-7B(Li et al., 2024)	57.17	0.86	85.4	0.62	83.3	0.70
SFT CodeS-15B(Li et al., 2024)	58.47	0.92	84.9	0.82	83.7	0.88
GenLink + $\{QC\}$	**59.32** (↑**0.85**)	2.02	**87.3** (↑**1.8**)	1.78	**86.6** (↑**2.2**)	1.80
GenLink + $\{L, Q, D\}$	**65.58** (↑**7.11**)	2.24	**88.8** (↑**3.3**)	2.11	**87.5** (↑**3.1**)	2.16
GenLink + $\{L, QC, Q, D, M\}$	**67.34** (↑**8.87**)	3.24	**89.7** (↑**4.2**)	2.85	**87.8** (↑**3.4**)	3.15

Table 1: Performance and computational overhead of GenLink on BIRD and Spider dataset.

ships between questions and database schemas, while simultaneously reducing the schema linking learning burden on SQL generation models.

Ablation Studies We conducted an ablation study on the BIRD development set to systematically evaluate the contributions of three key components within the GenLink framework: Schema Linking(SL), GDSL and MMSG. Table 2 illustrates the impact of each component on the execution accuracy of GenLink. The results demonstrate that both our proposed GDSL and MMSG strategies yield significant performance improvements. Specifically, compared to employing only traditional schema linking methods, GDSL enhances execution accuracy by 2.22%, while the Multi-Model SQL Generation strategy contributes a more substantial increase of 5.80%. These findings highlight the importance of each component in enhancing the overall performance of Text-to-SQL translation.

Model	Simple	Mod.	Chall.	All
Baseline	64.00	49.68	38.19	57.24
+ SL	67.68	51.61	43.75	59.32(+2.08)
+ GDSL	68.97	52.90	41.67	61.54(+2.22)
+ MMSG	73.19	60.00	53.47	**67.34(+5.80)**

Table 2: Experiments on GenLink + $\{L, QC, Q, D, M\}$ components using EX based on BIRD development set, with **Baseline** representing Full Schema strategy.

Effect of Different Numbers of Models To evaluate the impact of different numbers of models on the GenLink framework, we gradually increased the number of models from 1 to 5. The results, which are the average of all possible combinations as shown in Figure 3, indicate that as the number

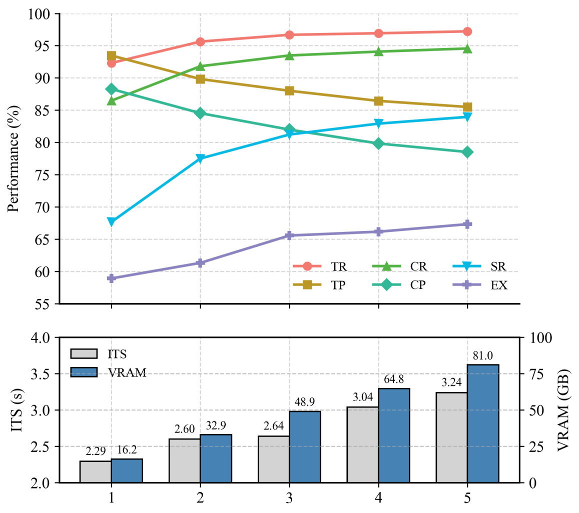


Figure 3: Performance and computational overhead of different numbers of models on BIRD development set. VRAM is the GPU memory usage of GenLink during the inference stage.

of models increases, there is a consistent upward trend in TR, CR, SR, and EX. This demonstrates that incorporating a greater number of diverse models enhances the framework's ability to identify and link relevant schema elements, thereby improving overall schema linking accuracy. The improvement can be attributed to the complementary strengths of individual models, which, when combined, reduce the likelihood of missing valid schema entries and increase robustness against model-specific biases or errors.

However, due to the adoption of a multi-model extraction and merging strategy, the increase in the number of models inevitably leads to redundant schema entries, resulting in a downward trend in TP and CP. Additionally, the computational overhead increases accordingly.

Dataset	Method	TR	TP	CR	CP	SR	EX
BIRD-dev	SFT CodeS-15B	99.76	36.40	98.91	13.26	95.89	58.47†
	CHESS$_{IR+SS+CG}$	96†	90†	94†	71†	-	65.0†
	GenLink (ours)	97.22(↑1.22)	85.50	94.56(↑0.56)	**78.52**(**↑7.52**)	83.96	**67.34** (**↑2.34**)
Spider-dev	SFT CodeS-15B	100	38.05	99.76	13.92	95.84	84.9
	GenLink (ours)	99.61	**92.56** (**↑54.51**)	98.33	**88.93** (**↑75.01**)	93.42	**89.7** (**↑4.8**)
Spider-test	SFT CodeS-15B	99.97	37.40	99.98	16.69	96.60	83.7
	GenLink (ours)	99.54	**90.53** (**↑53.13**)	98.44	**89.29** (**↑72.60**)	92.41	**87.8** (**↑4.1**)

Table 3: Performance of Generation-Driven Schema Linking Module on BIRD and Spider datasets. Results marked with † are from the original paper, others are reproduced by us.

Effect of Generation-Driven Schema Linking Module To comprehensively evaluate the effectiveness of the Generation-Driven Schema Linking (GDSL) module, we conducted systematic experimental assessments using TR, TP, CR, CP, SR and EX on the BIRD and Spider datasets (Table 3). CodeS-15B, which adopts a probability-based prediction method by selecting the top 6 tables and top 10 columns per table, achieved high recall but low precision. In contrast, CHESS$_{IR+SS+CG}$, by incorporating GPT-4-turbo technology, achieved a better balance between recall and precision, although its high token cost severely limited its practical application scenarios. Notably, our GDSL module, also a generative approach, achieved superior schema linking with far fewer parameters.

On BIRD-dev, our GDSL module improved CP by 7.52% over CHESS$_{IR+SS+CG}$, and benefiting from a higher recall rate, the model also enhanced SQL generation performance (EX) by 2.34% compared to CHESS$_{IR+SS+CG}$. On Spider-dev and Spider-test, where CHESS results were unavailable, we only conducted comparative experiments with SFT CodeS-15B. The results in Table 3 demonstrate that GDSL consistently improves both precision (TP, CP) and execution accuracy (EX), showing large gains in CP (up to +75.01%) and notable improvements in EX (+4.8% on dev and +4.1% on test).

The Role of Model Diversity under Matched Compute Budgets To rigorously evaluate whether GenLink's performance gains stem from increased computational budget or the proposed multi-model diversity, we conduct a controlled ablation study under matched compute conditions.

Specifically, we replace the five distinct models in GenLink with five independent inference runs from a single model (e.g., Mistral-7B-Instruct-v0.3 or Qwen2.5-Coder-7B-Instruct), each initialized with a different random seed. All other settings (e.g., temperature=0 for deterministic generation, self-consistency voting) remain identical to the original GenLink setup. This ensures that the total computational cost (number of forward passes) is equivalent to the full GenLink ensemble.

As shown in Table 4, the single-model ensemble strategy yields only marginal improvements over the base model (e.g., 56.45% $\rightarrow$ 57.30% for Mistral). In stark contrast, the full GenLink ensemble, which leverages diverse model families, achieves a significantly higher execution accuracy of 67.34%. This substantial performance gap demonstrates that GenLink's effectiveness is not merely a byproduct of increased computation or sampling variance. Instead, it critically depends on the complementarity and diversity across different model architectures and pre-training corpora, which enables more robust schema linking and SQL generation.

Method	EX
M (single model)	56.45
Single-Model Ensemble (M)	57.30
QC (single model)	57.24
Single-Model Ensemble (QC)	59.39
GenLink (Full Ensemble)	**67.34**

Table 4: Execution accuracy (EX) on BIRD development set under matched compute budget.

4 Conclusion

In this work, we introduced **GenLink**, a generation-driven schema-linking framework for Text-to-SQL. The core idea of GenLink is to leverage multi-model generation to drive a more stable schema-linking process. Experimental results on the BIRD and Spider benchmarks demonstrate the effectiveness of GenLink, achieving high execution accuracy across diverse database schemas. In future work, we aim to further refine GenLink by incorporating more advanced reasoning mechanisms.

Limitations

Our work has the following limitations: first, due to the limitation of computational resources, we only conducted experiments on 3B, 8B and 14B scale models. Based on the analysis of the existing experimental results, we speculate that the model performance is expected to be further improved if a larger scale model (e.g., 32B) is used. Second, in terms of model architecture, the multi-model approach does have the problem of larger computational cost compared to traditional single-model approaches (e.g., DTS-SQL, CodeS), but GenLink has significant advantages in solving the cross-domain problem and handling implicit patterns. And we can also use a more lightweight model combination such as $\{L, Q, D\}$ that takes care of both computational overhead and performance.

Acknowledgements

This research was supported in part by the National Science and Technology Major Project (2021ZD0111501), the National Science Fund for Excellent Young Scholars (62122022), the Natural Science Foundation of China (62206064, 62406078), the Guangzhou Basic and Applied Basic Research Foundation (2024A04J4384), the major key project of PCL (PCL2021A12), the Guangdong Basic and Applied Basic Research Foundation (2023B1515120020), and the Jihua laboratory scientific project (X210101UZ210).

References

Hasan Alp Caferoğlu and Özgür Ulusoy. 2025. E-sql: Direct schema linking via question enrichment in text-to-sql. *Preprint*, arXiv:2409.16751.

Naihao Deng, Yulong Chen, and Yue Zhang. 2022. Recent advances in text-to-sql: A survey of what we have and what we expect. *Preprint*, arXiv:2208.10099.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-SQL. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1337–1350, Online. Association for Computational Linguistics.

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. *Preprint*, arXiv:2307.07306.

Zhen Dong, Shizhao Sun, Hongzhi Liu, Jian-Guang Lou, and Dongmei Zhang. 2019. Data-anonymous encoding for text-to-SQL generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5405–5414, Hong Kong, China. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021. Towards robustness of text-to-SQL models against synonym substitution. 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 2505–2515, Online. Association for Computational Linguistics.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *Preprint*, arXiv:2308.15363.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. 2024. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation. *Preprint*, arXiv:2405.07467.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954, Online. Association for Computational Linguistics.

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.

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. *Preprint*, arXiv:2402.16347.

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. *Preprint*, arXiv:2305.03111.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, and Ilge Akkay. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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

Mohammadreza Pourreza and Davood Rafiei. 2024. Dts-sql: Decomposed text-to-sql with small large language models. *Preprint*, arXiv:2402.01117.

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *Preprint*, arXiv:2208.13629.

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. *Preprint*, arXiv:2405.15307.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. Chess: Contextual harnessing for efficient sql synthesis. *Preprint*, arXiv:2405.16755.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2025. Mac-sql: A multi-agent collaborative framework for text-to-sql. *Preprint*, arXiv:2312.11242.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *Preprint*, arXiv:1809.08887.

Chao Zhang, Yuren Mao, Yijiang Fan, Yu Mi, Yunjun Gao, Lu Chen, Dongfang Lou, and Jinshu Lin. 2024. Finsql: Model-agnostic llms-based text-to-sql framework for financial analysis. *Preprint*, arXiv:2401.10506.

Xiaohu Zhu, Qian Li, Lizhen Cui, and Yongkang Liu. 2024. Large language model enhanced text-to-sql generation: A survey. *Preprint*, arXiv:2410.06011.

A Appendix A

A.1 Experiment Settings

Hyperparameters Since the pre-training corpus for different language models is different, we use the Low-Rank Adaptation (LoRA)(Hu et al., 2021) technique on NVIDIA A800 to fine tune the model with a specific epoch, where the epoch is different for different models, to make the model converge. The detailed information of the experimental hyperparameters is shown in Table 5, which can help readers better reproduce our work.

Hyper-parameter	Value
lora_rank	32
lora_alpha	128
lora_dropout	0.05
learning_rate	5e-05
per_device_train_batch_size	2
gradient_accumulation_steps	8

Table 5: Model Hyper-parameter Configuration.

A.2 Further Experiments and Analysis

A.2.1 Generalization of Generation-Driven Schema Linking Module

To demonstrate the generalization of the Generation-Driven Schema Linking (GDSL) module, we conducted an experiment in which the GDSL module was treated as a plug-in and integrated into single SQL generation baseline models with varying parameter sizes, such as 3B, 7B, and 14B. As shown in Table 6, Full Schema EX represents the performance without our Generation-Driven Schema Linking module, while GDSL Schema EX represents the performance with it. The Qwen2.5-Coder-3B-Instruct model achieved an EX of 60.63%, a significant improvement of 6.33% over the full schema approach. This performance advantage expanded as the model size increased: the Qwen2.5-7B-Instruct model achieved an EX of 62.58% (3.45% higher than the full schema), and the Qwen2.5-14B-Instruct model reached an EX of 64.60% (3.84% higher than the full schema). These performance improvements can be attributed to the inherent superior SQL generation capabilities of the Qwen model series. This architectural advantage enables us to adopt a single-model inference strategy during the SQL generation phase, achieving an optimal balance between computational cost and performance.

Params	SQL Generation model	Full Schema EX	GDSL Schema EX
3B	Qwen2.5-3B-Instruct	50.26	58.54 (↑8.28)
	Qwen2.5-Coder-3B-Instruct	54.30	**60.63 (↑6.33)**
6.7~8B	Llama-3.1-8B-Instruct	58.47	61.21 (↑2.74)
	Qwen2.5-Coder-7B-Instruct	57.24	61.54 (↑4.30)
	Mistral-7B-Instruct-v0.3	56.45	61.15 (↑4.70)
	deepseek-coder-6.7b-instruct	57.76	61.15 (↑3.39)
	Qwen2.5-7B-Instruct	59.13	**62.58 (↑3.45)**
14B	Qwen2.5-Coder-14B-Instruct	60.95	64.15 (↑3.20)
	Qwen2.5-14B-Instruct	60.76	**64.60 (↑3.84)**

Table 6: Execution Accuracy of GDSL on BIRD development set using single-model SQL generation across different parameter sizes.

Method	Easy	Medium	Hard	Extra	Total
Baseline	96.8	91.7	74.1	62.0	85.2
+SL	96.0	92.6	75.9	66.9	86.5(+1.3)
+GDSL	95.6	92.4	82.8	66.3	87.3(+0.8)
+MMSG	96.0	94.8	82.8	74.1	**89.7 (+2.4)**

Table 7: The execution accuracy (EX) of the GenLink pipeline by removing each component on the Spider development set.

Method	Easy	Medium	Hard	Extra	Total
Baseline	93.4	87.0	79.9	71.4	84.3
+SL	93.8	88.1	80.6	74.8	85.5(+1.2)
+GDSL	94.0	89.0	82.5	76.2	86.6(+1.1)
+MMSG	93.8	89.1	87.3	77.6	**87.8 (+1.2)**

Table 8: The execution accuracy (EX) of the GenLink pipeline by removing each component on the Spider test set.

A.2.2 Effect of Different modules

To comprehensively evaluate the performance contributions of each module within GenLink, we provided a detailed modular analysis on the BIRD dataset in the main text. To further validate the generalization capability and module effectiveness of GenLink, we conducted systematic experiments on the development and test sets of the Spider benchmark. The experimental results indicate that the introduction of the Generation-Driven Schema Linking (GDSL) module led to a consistent improvement in the ability to generate correct SQL compared to traditional Schema Linking (SL), although the extent of improvement was relatively modest. This phenomenon primarily stems from the inherent limitations of single-model SQL generation in terms of domain adaptability. To overcome this bottleneck, we adopted a Multi-Model SQL Generation (MMSG) strategy, which not only fully leveraged the potential of the GDSL module but also significantly enhanced the model's execution accuracy (EX), achieving more substantial performance improvements. As shown in Table 7 and Table 8, MMSG contributed to execution accuracy improvements of 2.4% and 1.2%, respectively.

A.2.3 Effect of Different Numbers of Models

In the main text, we introduced Figure 3 to show the average performance of different number of model combinations on the BIRD development set, providing a visual comparison of the overall trend. Table 9, on the other hand, lists in detail the precise values of each specific model combination on each metric (e.g., TR, TP, CR, CP, SR, EX, etc.), and labels the optimal EX results (bolded) and the sub-optimal EX results (underlined), which facilitates an in-depth analysis of the differences in the strengths and weaknesses of different combinations. In short, Figure 3 focuses on macro trends, while Table 9 provides micro details. To measure the resource overhead, we also provide the Inference Time per Sample (ITS) for each model combination and the GPU memory usage (VRAM) for reasoning on each model combination. In the case of limited resources, we can choose the appropriate model combination as needed to balance the model performance and inference efficiency. In order to ensure the reliability of the experimental results, under each set of model number settings (except for the case of 5 models), we have conducted tests with at least 5 different model combinations to fully evaluate the performance under various configurations. The experimental results indicate that: (1) under a fixed number of models, the performance variation across different model combinations is minimal, demonstrating good stability; (2) with the increase in the number of models, although the computational overhead increases, the execution

Number of models	Combination of models	TR	TP	CR	CP	SR	EX	ITS	VRAM
1	L	92.31	93.48	86.47	88.29	67.67	58.93	2.03	17.06
	QC	93.02	93.56	87.09	89.45	67.21	<u>59.32</u>	2.02	16.95
	Q	93.94	92.68	88.06	89.45	67.21	**60.56**	2.24	16.95
	D	91.67	93.57	86.37	89.15	67.21	58.67	1.94	14.47
	M	91.57	93.02	85.01	89.16	64.99	57.17	3.24	15.57
2	L, Q	95.63	89.82	91.81	84.54	77.51	**61.34**	2.24	34.01
	QC, Q	95.63	90.48	91.33	85.80	76.14	61.15	2.24	33.90
	L, QC	95.26	90.89	91.45	84.73	76.53	**61.34**	2.03	34.01
	QC, M	95.56	90.42	91.35	85.25	75.49	<u>61.21</u>	3.24	32.52
	D, M	95.23	90.51	91.39	85.27	76.86	61.02	3.24	30.04
3	QC, Q, D	96.61	88.71	93.21	83.25	80.57	64.99	2.24	48.37
	L, Q, D	96.68	88.01	93.48	81.99	81.23	**65.58**	2.24	48.48
	L, D, M	96.17	88.42	92.95	81.81	80.05	<u>65.45</u>	3.24	47.10
	L, Q, M	96.11	87.76	92.97	81.79	80.38	64.21	3.24	49.58
	L, QC, Q	96.34	88.49	93.07	82.25	80.31	63.89	2.24	50.96
4	L, Q, D, M	96.92	86.44	94.08	79.84	82.92	**66.17**	3.24	64.05
	L, QC, Q, D	96.99	86.97	94.05	80.27	82.59	<u>65.97</u>	2.24	65.43
	QC, Q, D, M	96.99	86.87	94.00	80.72	82.53	65.78	3.24	63.94
	L, QC, Q, M	96.68	86.78	93.77	79.99	81.88	64.28	3.24	66.53
	L, QC, D, M	96.75	86.97	93.75	79.76	81.81	64.54	3.24	64.05
5	L, QC, Q, D, M	97.22	85.50	94.56	78.52	83.96	**67.34**	3.24	81.00

Table 9: Performance and computational overhead of different model combinations on BIRD development set.

accuracy EX and recall rates TR and CR for tables and columns exhibits a clear monotonic upward trend. Specifically, when the number of models increased from 2 to 3, the optimal EX improved by 4.24% and both TR and CR increased by over 1%. This phenomenon fully validates the effectiveness of the multi-model collaboration mechanism in the GenLink method, indicating that integrating decisions from multiple models can significantly enhance the overall performance of the system.

A.2.4 Effect on Robustness Benchmarks

In addition to evaluations on BIRD-dev, Spider-dev, and Spider-test, we have supplemented experiments on Spider-SYN(Gan et al., 2021) (a human-currated synonym-substitution benchmark) and Spider-Realistic(Deng et al., 2021) (where explicit column mentions are removed) to further validate the robustness and effectiveness of our method. Table 10 shows the execution accuracy performance of GenLink. GenLink achieves better performance on both Spider-SYN (83.5%) and Spider-Realistic (86.6%), surpassing all compared methods by significant margins. Notably, GenLink improves upon the strongest baseline (SFT CodeS-15B) by 6.5% and 3.5% on these benchmarks, respectively. These results demonstrate GenLink's robustness against real-world linguistic variations and schema perturbations.

Method	Spider-SYN	Spider-Realistic
DAIL-SQL	-	76.0
SFT CodeS-7B	76.9	82.9
SFT CodeS-15B	77.0	83.1
GenLink(ours)	**83.5** (↑6.5)	**86.6** (↑3.5)

Table 10: Execution accuracy (EX) of GenLink on the Spider-SYN and Spider-Realistic datasets.

A.2.5 Ablation Study on the Interplay between GDSL and MMSG

To rigorously evaluate the individual and synergistic contributions of the Generation-Driven Schema Linking (GDSL) and Multi-Model SQL Generation (MMSG) modules, we conduct a comprehensive ablation study by introducing two critical variants: (1) SL + MMSG: MMSG using a schema linker trained via supervised fine-tuning (QwenCoder). (2) MMSG only (Full Schema + Voting): MMSG with full database schema input and no schema linking.

As shown in Table 11, GenLink (GDSL + MMSG) achieves 67.34% EX, outperforming SL +

Method	Simple	Moderate	Challenging	All
SL + MMSG	71.14	57.20	45.14	64.47
MMSG only (Full Schema + Voting)	71.89	57.42	45.14	64.99
GenLink (GDSL + MMSG)	**73.19**	**60.00**	**53.47**	**67.34**

Table 11: Component-wise execution accuracy (EX) by difficulty level on the BIRD development set.

Method	TR	TP	CR	CP	SR	Dev EX
SL + MMSG	93.02	93.56	87.09	89.45	67.21	64.47
MMSG only (Full Schema + Voting)	100.00	27.20	100.00	5.91	100.00	64.99
GenLink (GDSL + MMSG)	97.22	85.50	94.56	78.52	83.96	**67.34**

Table 12: Schema linking quality and execution accuracy (EX) on the BIRD development set.

MMSG (64.47%) and MMSG only (64.99%). GenLink provides the most substantial gains on Moderate and Challenging questions, where complex schema reasoning is paramount. This highlights GDSL's strength in handling intricate, implicit relationships that are difficult for explicit extractors to capture. A deeper analysis of schema quality (Table 12) reveals the reasons for this gap: MMSG only has perfect recall (CR=100%) but very low precision (CP=5.91%), as the full schema introduces noise and harms generation. SL + MMSG has high precision (CP=89.45%) but lower recall (CR=87.09%), indicating the explicit linker may miss critical columns. GenLink strikes the optimal balance, achieving high recall (CR=94.56%) and high precision (CP=78.52%).

This demonstrates that GDSL is not redundant but essential. It provides a high-quality, focused schema that maximizes the effectiveness of the MMSG ensemble, enabling superior performance compared to both explicit linking and no linking.

Comparison Pair	Similarity Score
Q vs QC	0.944
Q vs L	0.944
Q vs D	0.936
Q vs M	0.945
QC vs L	0.947
QC vs D	0.940
QC vs M	0.947
L vs D	0.942
L vs M	0.951
D vs M	0.943
Average	**0.944**

Table 13: Comparison of semantic similarity of SQL generated by different generative language models.

Number of Models	Jaccard Similarity	TR	CR	EX
2	0.81	95.46	91.47	61.21
3	0.74	96.38	93.14	64.82
4	0.69	96.87	93.93	65.35
5	0.65	97.22	94.56	67.34

Table 14: Model Ensemble Performance Metrics.

A.2.6 Analysis of Different Generative Language Models

We analyzed five models used in the experiment: Llama-3.1-8B-Instruct, Qwen2.5-Coder-7B-Instruct, Qwen2.5- 7B-Instruct, deepseek-coder-6.7b-instruct and Mistral-7B-Instruct-v0.3. While different generative language models generate SQL with similar basic skeletons due to their pretrained SQL generation capabilities, their linked semantic schemas may vary owing to biases introduced by their respective pre-training corpora. To validate this observation, we conducted the following experiments (1)Semantic Similarity of Generated SQL and (2)Schema Linking Diversity via Jaccard Similarity on the BIRD-dev dataset.

The specific details of Experiment (1) are as follows: we used BERT to vector SQL and measured the semantic similarity between SQL queries generated by different generative models on BIRD-dev. As shown in Table 13, experimental results revealed an average similarity of 94.4%, confirming near identical SQL skeletons. The remaining 5.6% divergence stemmed from variations in linked schemas.

To further prove the difference comes from the schema linking, we designed Experiment (2). We quantified schema linking differences using Jaccard similarity in Table 14. To be specific, we carry out Jaccard Similarity calculation for different schema sets. The schema sets are linked by different models. And we take an average value among 1534 BIRD samples. As the number of models increased, Jaccard similarity decreased, indicating greater diversity in linked schemas. This diversity enables GenLink to explore a broader space of potential schemas, thereby improving Table Recall (TR) and Column Recall (CR). The superior schema linking performance directly contributes to GenLink's 67.34% EX on BIRD-dev, demonstrating its effectiveness.

B Appendix B

B.1 Related work

Schema Linking-Centered Research Schema linking is a core preliminary step in the text-to-SQL task, whose primary goal is to establish accurate associations between natural language queries and database schemas. It directly determines the structural accuracy of subsequent SQL generation. Existing research focuses on "how to efficiently capture the matching relationships between NL queries and schema elements" and can be mainly categorized into two core approaches: neural network-based methods and in-context learning (ICL)-based methods. These two approaches exhibit significant differences in matching granularity and semantic understanding capabilities.

Neural network-based schema linking leverages deep learning to capture the semantic relationships between queries and schemas. DAE (Dong et al., 2019) treats schema linking as a sequence labeling task, while SLSQL (Lei et al., 2020) improves performance by annotating schema linking data in benchmarks such as Spider (Yu et al., 2019). RESDSQL (Li et al., 2023a) introduces a cross-encoder with enhanced ranking to prioritize relevant schema elements, and FinSQL (Zhang et al., 2024) accelerates the linking process through parallel cross-encoders. Despite the advantages of these methods, they face generalization challenges across different schemas, especially when training data is limited.

Schema linking methods based on ICL, relying on the semantic understanding and reasoning capabilities of LLMs, break through the surface-level limitations of string matching and are suitable for dynamic and complex schema association scenarios. TA-SQL (Qu et al., 2024) utilizes the task alignment capability of LLMs (e.g., GPT-4 (OpenAI et al., 2024)) to link the schema linking and SQL generation stages. By embedding schema information in prompts, it reduces "hallucinatory linking". To address the schema filtering challenge in large-scale databases, MAC-SQL (Wang et al., 2025) designs a multi-agent architecture, where a "selector agent" is specifically responsible for the initial filtering of schema elements, and a "generator agent" then completes fine-grained linking to improve linking efficiency. E-SQL (Caferoğlu and Özgür Ulusoy, 2025) further optimizes semantic associations by integrating key values and conditional constraints in the database into NL query reconstruction, thereby strengthening the semantic binding between queries and schemas. Building on this, CHESS$_{IR+SS+CG}$ (Talaei et al., 2024) adds keyword extraction from external evidence and combines a "three-stage schema pruning strategy" (coarse filtering - fine ranking - verification) to reduce redundant linking. Although ICL methods have significant advantages in semantic matching, they suffer from a prominent high token cost issue: descriptions of large-scale schemas occupy a large amount of prompt length, leading to reduced inference efficiency and increased costs in practical deployment.

End-to-End Text-to-SQL Models End-to-end text-to-SQL models aim to skip the explicit schema linking step and directly map NL queries to complete SQL statements. Their core design logic is to "integrate schema understanding and SQL generation into a single learning task", which is mainly implemented through two technical approaches: neural network-based semantic modeling and supervised fine-tuning (SFT)-based generative models. While simplifying the process, these models also face challenges in schema generalization capabilities.

Neural network-based end-to-end models directly capture the joint semantics of NL queries, SQL structures, and database schemas through deep learning architectures. DAE (Dong et al., 2019) pioneered the transformation of the end-to-end task into a sequence labeling problem, treating table names and column names as "annotation labels". It performs joint encoding of NL queries and schema sequences using a BiLSTM encoder, directly outputting SQL token sequences containing schema information. Addressing the scarcity of annotated data, SLSQL (Lei et al., 2020) constructs a dedicated schema linking annotation dataset within the Spider (Yu et al., 2019) benchmark. By increasing "schema-query" aligned samples, it enhances the model's implicit recognition ability of schema elements.

SFT-based generative end-to-end models optimize the parameters of pre-trained language models (PLMs) through gradient descent on labeled (NL query, SQL) pairs, internalizing the mapping rules of "schema understanding - SQL generation" into model parameters without relying on schema demonstrations in prompts. DTS-SQL (Pourreza and Rafiei, 2024) proposes a two-stage fine-tuning strategy: the first stage only trains the "schema element recognition" subtask (e.g., predicting table names and column names in SQL), and the second stage trains complete SQL generation. By decomposing the task, it reduces the learning difficulty of schema understanding and indirectly improves the accuracy of schema association. From the perspective of data augmentation, CodeS (Li et al., 2024) optimizes SFT performance. It constructs diverse training samples through "bidirectional data augmentation" and combines "strategic prompt construction" to strengthen the model's memory of schemas. However, the "implicit schema understanding" design formed by such methods also makes it difficult for the model to learn complex linking patterns, resulting in weak interpretability and error correction capabilities.

Utilization of Self-Consistency in SQL Generation Traditional ensemble learning in machine learning shows that combining multiple models can improve overall performance and robustness. Common techniques include majority voting, averaging, or selecting the most confident prediction. In the context of LLMs and generative models, basic ensemble methods may involve generating multiple outputs and selecting the most frequent response, which is the strategy adopted in self-consistency (Wang et al., 2023). C3 (Dong et al., 2023) samples multiple reasoning paths to generate diverse SQL answers and applies a voting mechanism to the execution results of SQL. MCS-SQL (Lee et al., 2024) generates various SQL queries based on different prompts, which differ in the selection method and order of few-shot examples. Driven by Gemini 1.5, CHASE-SQL (Pourreza et al., 2024) adopts various chain-of-thought prompting techniques to generate candidates, and then selects from the candidates through a binary voting mechanism. However, our method, GenLink, effectively improves the recall rate of schema linking by integrating the variability of different generative language models in complex schema understanding, thereby solving the problem of domain generalization.

B.2 Prompt Template

BIRD prompt template

Given the following database schema, question and evidence, your task is to write a valid SQLite SQL query whose execution results can accurately answer the question. Evidence can help you to comprehend the database values.

Database Schema:
{schema}
{content_sequence}

Question:
{question}

Evidence:
{evidence}

Please only provide valid SQLite SQL query and do not provide unnecessary explanations.

SQL:

Spider prompt template

Given the following database schema and question, your task is to write a valid SQLite SQL query whose execution results can accurately answer the question.

Database Schema:
{schema}
{content_sequence}

Question:
{question}

Please only provide valid SQLite SQL query and do not provide unnecessary explanations.

SQL:

Existing approaches typically rely on complex instructions crafted by human experts, which not only require significant domain expertise but also consume a lot of time for optimization. In contrast, we use a simple and generalized prompt template. Experimental results show that GenLink achieves better performance than in-context learning methods such as CHESS$_{IR+SS+CG}$ on both the BIRD and Spider benchmark datasets, which validates that our method can achieve excellent performance even on simple prompts.

B.3 Case study

In this section, through a systematic analysis of typical cases from the BIRD dataset, we compared the performance differences between GenLink and SFT CodeS-15B in schema linking task. The experimental results (as shown in Table 15 and Table 16) indicate that when CodeS employs a single-model explicit extraction approach, it incorrectly links *schools.school* and *foreign_data.name*. This error primarily stems from two key factors: (1) the inherent limitations of explicit extraction methods in handling complex schemas, making it difficult to capture deep semantic relationships; and (2) the lack of sufficient generalization capability in language models trained in a single training corpus environment, which struggle to adapt to cross-domain scenarios. In contrast, GenLink, by integrating multiple small language models and adopting a generation-based schema linking strategy, successfully achieves implicit schema linking (e.g., *satscores.cds* and *cards.name*). This result demonstrates that the multi-model strategy can effectively enhance the robustness and accuracy of schema linking, particularly showcasing significant advantages in handling complex semantic scenarios.

Question	What is the total number of schools whose total SAT scores are greater or equal to 1500 whose mailing city is Lakeport?
Evidence	Total SAT scores can be computed by avgscrread + avgscrmath + avgscrwrite.
Schema Linking Variants	
Qwen	{"satscores": ["cds", "avgscrread", "avgscrmath", "avgscrwrite"], "schools": ["cdscode", "mailcity"]}
QwenCoder	{"satscores": ["cds", "avgscrread", "avgscrmath", "avgscrwrite"], "schools": ["cdscode", "mailcity"]}
Llama	{"satscores": ["cds", "numge1500"], "schools": ["cdscode", "mailcity"]}
DeepSeek	{"schools": ["cdscode", "mailcity"], "satscores": ["cds", "avgscrread", "avgscrmath", "avgscrwrite"]}
Mistral	{"satscores": ["cds", "numtsttakr"], "schools": ["cdscode", "mailcity"]}
Merged Schema	{"satscores": ["cds", "avgscrread", "avgscrmath", "avgscrwrite", "numge1500", "numtsttakr"], "schools": ["cdscode", "mailcity"]}
GenLink Predicted SQL	SELECT count(satscores.cds) FROM schools INNER JOIN satscores ON schools.cdscode = satscores.cds WHERE schools.mailcity = 'Lakeport' AND satscores.avgscrread + satscores.avgscrmath + satscores.avgscrwrite >= 1500
CodeS Explicitly Schema	{"schools": ["mailcity", "cdscode", "school", "city", ...], "satscores": ["cds", "avgscrread", "avgscrmath", "avgscrwrite", ...], ...}
CodeS Predicted SQL	SELECT count(schools.school) FROM schools INNER JOIN satscores ON schools.cdscode = satscores.cds WHERE schools.mailcity = 'Lakeport' AND satscores.avgscrread + satscores.avgscrmath + satscores.avgscrwrite >= 1500
Gold SQL	SELECT count(satscores.cds) FROM satscores INNER JOIN schools ON satscores.cds = schools.cdscode WHERE schools.mailcity = 'Lakeport' AND (satscores.avgscrread + satscores.avgscrmath + satscores.avgscrwrite) >= 1500

Table 15: Case Study 1 of GenLink compared to CodeS on the BIRD Dataset.

Question	Among the Artifact cards, which are black color and comes with foreign language translation?
Evidence	Artifact card refers to originaltype = 'Artifact'; black color refers to colors = 'B'; foreign language refers to language in foreign_data;
Schema Linking Variants	
Qwen	{"foreign_data": ["name", "uuid"], "cards": ["uuid", "originaltype", "colors"]}
QwenCoder	{"foreign_data": ["uuid", "language"], "cards": ["name", "uuid", "originaltype", "colors"]}
Llama	{"foreign_data": ["uuid"], "cards": ["name", "uuid", "originaltype", "colors"]}
DeepSeek	{"foreign_data": ["uuid"], "cards": ["name", "uuid", "originaltype", "colors"]}
Mistral	{"foreign_data": ["uuid"], "cards": ["name", "uuid", "originaltype", "colors"]}
Merged Schema	{"foreign_data": ["name", "uuid", "language"], "cards": ["name", "uuid", "originaltype", "colors"]}
GenLink Predicted SQL	SELECT cards.name FROM cards INNER JOIN foreign_data ON cards.uuid = foreign_data.uuid WHERE cards.colors = 'B' AND cards.originaltype = 'Artifact'
CodeS Explicitly Schema	{"foreign_data": ["name", "uuid", ...], "cards": ["uuid", "originaltype", "colors", ...], ...}
CodeS Predicted SQL	SELECT foreign_data.name FROM cards INNER JOIN foreign_data ON cards.uuid = foreign_data.uuid WHERE cards.originaltype = 'Artifact' AND cards.colors = 'B'
Gold SQL	SELECT DISTINCT cards.name FROM cards INNER JOIN foreign_data ON cards.uuid = foreign_data.uuid WHERE cards.originaltype = 'Artifact' AND cards.colors = 'B'

Table 16: Case Study 2 of GenLink compared to CodeS on the BIRD Dataset.