A Notations and Symbols

The notations and symbols adopted in this paper are summarized in Table 1

Table 1: Notations

Notation	Explanation
	Common Variables
L	Language Model
Q	User-issued natural language question
D	Database, including its schema information
S	Target SQL query
D	The task-related annotated dataset contains N input-output pairs (x_i, y_i)
θ	Model parameters
d	Embedding dimension
b	Training batch
	schema-related Variables
T	The set of tables in the database
C	The set of columns in the database
S_k	The output of the relevant schemas predicted by the SchemaLinker model
P_s	The predicted set of schema links
T_s	The ground truth set of schema links
	SchemaLinker-related Variables
CoT	CoT information
I	The instruction for specific task t
m_t	Sampling weight for task t
N_t	The number of tasks t in the sample
p_n	The probability of a single task t being sampled
$E[samples_t]$	The expected number of samples per task
A_i	The group-relative advantage for each output
π_{ref}	The reference policy
π_θ	Current policy model
R_k	Reward for schema linking tasks
	SAR-related Variables
E	The final embedding representation enhanced by contrastive learning
E_Q	The embedding representation of question Q
E_S	The embedding representation of SQL query
τ	The temperature coefficient in contrastive learning
W_{sim}	Similarity loss weight
M	Attention mask in the Transformer
$\hat{S}$	Schema embedding generated by the SAR model

B Performance Evaluation of the Schema Linking Module

This section presents a comprehensive performance evaluation of our proposed **SchemaLinker**, assessing it from the dual perspectives of task effectiveness and computational efficiency during inference. To this end, we first introduce the implementation details and evaluation metrics. Subsequently, we conduct an in-depth analysis of the model's linking accuracy on the Spider and BIRD datasets across a progression of methods. Finally, by comparing the inference token consumption of different models, we further validate that our ultimate SchemaLinker method achieves both conciseness and high efficiency while maintaining superior performance.

B.1 Implementation Details

The entire training pipeline was conducted on a computational setup consisting of two NVIDIA A800 (80GB) GPUs. We employed BF-16 mixed-precision training, facilitated by the `Accelerate` framework. The model was optimized using the AdamW optimizer with the following hyperparameter configuration:

- $\beta_1 = 0.9$
- $\beta_2 = 0.99$
- $\epsilon = 10^{-8}$
- Learning Rate = 5×10^{-5}

B.2 Evaluation Metrics

For performance assessment, we evaluate table and column predictions using different metrics:

- **Tables**: Performance on table selection is measured using the **F1 score** (F_1).
- **Columns**: Due to the significant data imbalance (i.e., SQL queries typically use only a small subset of available columns), we evaluate column selection performance using the **Matthews Correlation Coefficient**(MCC), as it provides a more reliable measure for imbalanced classification tasks.

B.3 Results and Analysis

To validate the efficacy of the training methodologies of this study, we conducted evaluations on the development sets of the Spider and BIRD datasets subsequent to each training iteration. The results, as shown in Figure 1, demonstrate a consistent and progressive enhancement in the schema linking capabilities of the model at both the table and column levels across the evolution of the proposed methods—from Supervised Fine-Tuning and Distillation with Chain-of-Thought (CoT), to the integration of multi-task learning in Distillation CoT_MTL, and culminating in the Reinforcement Learning-based SchemaLinker.

Specifically, on the Spider dataset, the SchemaLinker method achieved competitive performance, attaining a table-level F1-score of 97.38% and Recall of 99.69%, and a column-level MCC of 90.81% and Recall of 94.29%.

Specifically, on the Spider dataset, the SchemaLinker method achieved competitive performance, attaining a table-level F1-score of 97.38% and a recall of 99.69%. For column-level metrics, it achieved an MCC of 90.81% and a recall of 94.29%. These results represent substantial improvements of approximately 10.9%, 12.1%, 17.1%, and 18.8% respectively, over the baseline supervised fine-tuning approach.

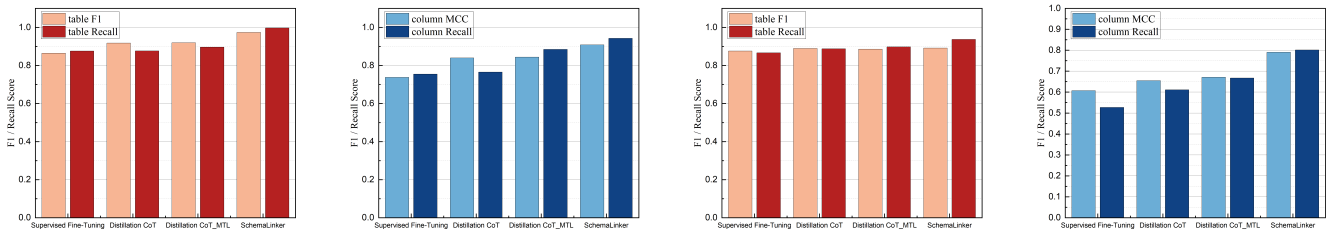
A similar trend was observed on the more challenging BIRD development set. Although the overall performance metrics were marginally lower due to the inherent complexity of the dataset, the **SchemaLinker** method continued to demonstrate superior performance. This evidence strongly suggests that leveraging the capabilities of large models in conjunction with knowledge distillation and reinforcement learning strategies is a crucial pathway for augmenting the generalization and semantic understanding of a model regarding schema elements, thereby significantly elevating schema linking performance.

To further evaluate the computational efficiency of the SchemaLinker model, we compared its token consumption during the inference process, as presented in Table 2. The experimental results clearly demonstrate that SchemaLinker exhibits significant efficiency in both input and output token usage. Specifically, SchemaLinker achieves an average output token count of 175.7 ± 36.9, which is substantially lower than that of the base Qwen model (368 ± 96.2) and the Qwen model fine-tuned with Chain-of-Thought and multi-task learning (CoT_MTL, 205.6 ± 38.6).

This efficiency stems from SchemaLinker's multi-stage optimization process, which refines the model's verbosity without sacrificing its reasoning capabilities. The model is first taught a detailed reasoning process through knowledge distillation with CoT data. Subsequently, the final reinforcement learning stage, using GRPO, optimizes the model based on a precise reward function. This incentivizes the model to produce the most accurate and concise schema linking results, effectively pruning the verbose intermediate steps learned during the CoT phase. This process ensures that while the model internalizes a complex reasoning framework, its final output is distilled down to the essential, structured information required for the downstream task, thereby significantly reducing token consumption.

Table 2: Comparison of average Token Consumption on 300 samples from the BIRD dataset.

Model	Input Token	Output Token
Qwen		368.0 ± 96.2
Qwen-Fintune(without CoT)		62.3 ± 22.6
Qwen-Fintune(CoT)	1886.9 ± 423.6	165.6 ± 32.4
Qwen-Fintune(CoT_MTL)		205.6 ± 38.6
SchemaLinker		175.7 ± 36.9



(a) table F1 and table recall on Spider (b) column MCC and column recall on Spider (c) table F1 and table recall on BIRD (d) column MCC and column recall on BIRD

Figure 1: The four models (from left to right): supervised fine-tuning; knowledge distillation with CoT; multi-task CoT distillation; and distillation with reinforcement learning.

C Dataset Details

We selected two widely adopted Text-to-SQL benchmark datasets: Spider and BIRD. As shown in Table 3, Spider is a large-scale cross-domain dataset, while BIRD focuses on complex SQL queries in big data environments. To simulate real-world scenarios where Retrieval-Augmented Generation (RAG) is used to enhance model performance, we extended the official development sets of both datasets.

In our implementation, we employ the GPT-4o model to generate example "Question-SQL-Schema" triples with structural similarity for each original triple in the validation sets of the BIRD and Spider datasets. After a manual selection process, the resulting datasets are curated and denoted as RAG_Spider and RAG_BIRD, respectively. These datasets not only serve as core evaluation benchmarks for assessing the performance of the Schema-Augmented Retriever(SAR), but also provide supervised signals enriched with explicit retrieval cues to facilitate model learning.

As shown in Table 4, we collected a large number of Question–Schema–SQL triples from various public software development communities, data science forums, and code hosting platforms. To ensure data quality, we removed erroneous pairs where the SQL statements failed to execute successfully, as well as pairs with overly simplistic formats. The resulting cleaned and curated data was then structured into Question–Schema–SQL triples, which were used to train the Schema-aware module.

The training data for the Contrastive Learning module was generated based on the original training sets of BIRD and Spider, using the same generation methodology as that employed for the Spider_RAG and BIRD_RAG datasets.

Table 3: Comparison of Spider and BIRD dataset properties

Feature	Spider	BIRD
Training Set	8,659	9,428
Development Set	1,034	1,534
RAG Set	3,102	4,176
Number of Databases	200	95
Domain Coverage	138	37
Average Database Scale	~2,000 rows	~549,000 rows
Total Data Size	Small-scale (N/A)	33.4 GB

Table 4: Shared RAG training set

Training Method	Number of Examples
Schema-aware	31,171
Contrastive Learning	54,261

D Evaluation Metrics Details

The valid efficiency score (VES) is designed to evaluate the performance of correctly predicted queries. Unlike execution accuracy (EX), where a correct query receives a binary score of 1, VES provides a more nuanced measure of efficiency. It is calculated as the ratio of the execution time of the ground-truth query to the execution time of the predicted query:

$$\text{VES} = \frac{\text{Execution Time}_{\text{ground-truth}}}{\text{Execution Time}_{\text{predicted}}} \tag{1}$$

Consequently, if the execution times of the predicted and ground-truth queries are identical, the VES score equals 1. If the predicted query executes faster than the ground-truth query, the VES score will be greater than 1. In practice, to ensure stable results, each correct predicted query and its corresponding ground-truth query are executed 100 times to record an average runtime.

Despite this averaging process, VES remains highly sensitive to variations in hardware, software, and system state, which is why EX is considered the primary and more dependable metric for the BIRD benchmark.

E Model Details

The Large Language Models(LLMs) employed in this study, along with the comparative baseline models, are detailed in Table 5.

Table 5: Descriptions of all the involved models

Model	Description
	large language model
CodeS [7]*PACMMOD 2024*	A text-to-sql model based on StarCoder, developed using incremental pre-training techniques.
XiYanSQL[4]*Alibaba 2025*	A text-to-sql model based on QwenCoder, trained using both fine-tuning and GRPO.
Qwen 2.5[10]*Alibaba 2025*	An open-source, general-purpose LLM from the Tongyi Qianwen series, developed by Alibaba Cloud.
Llama 3[6]*Meta AI 2024*	An open-source, general-purpose LLM from the Llama series, developed by Meta AI.
GPT-4o-mini[1]*OpenAI 2024*	A lightweight, distilled LLM from the GPT-4o series, developed by OpenAI.
GPT-4o[1]*OpenAI 2024*	A proprietary, general-purpose LLM from the GPT series, developed by OpenAI.
DeepSeek-V3[2]*DeepSeek 2024*	A general-purpose LLM from the DeepSeek series, based on the MoE architecture, developed by DeepSeek.
GLM-4-Plus[5]*Zhipu AI 2024*	A proprietary, general-purpose LLM from the GLM series, developed by Zhipu AI.
	TEXT-to-SQL method
TA-SQL[9]*ACL 2024*	A Text-to-SQL method that mitigates hallucination in large language models through a task alignment strategy.
Din-SQL[8]*NeurIPS 2023*	A Text-to-SQL method that addresses complex problems by employing a divide-and-conquer strategy.
ACT-SQL[11]*EMNLP 2023*	A Text-to-SQL method that enhances the reasoning of large models via automatically generated Chain-of-Thought.
C3-SQL[3]*arXiv 2023*	A Text-to-SQL method that improves zero-shot generation quality through prompt engineering and calibration.

F Detailed Experimental Results

This section provides the complete and detailed experimental results, which are presented in a summarized form in the main body of the paper.

Table 6 presents a comprehensive breakdown of model performance on the Spider dataset. It evaluates EX and EM across four difficulty levels (Easy, Medium, Hard, Extra). Table 7 details the performance on the more challenging BIRD dataset. It measures EX and the VES across three complexity levels (SIM, MOD, CHALL).

In both tables, results are shown for each framework (e.g., TA-SQL, SchemaRAG) when paired with different backbone LLMs. Values highlighted in bold or marked with a • indicate the top-performing model in each category.

Table 6: Comparison of EX and EM on the Spider dataset

Model	LLM	Metric	Easy	Medium	Hard	Extra	All
TA-SQL	GPT-4o-mini	EX	93.5●	90.8●	77.6	64.5	85.0
		EM	87.9●	68.8●	63.2	45.2●	68.7●
	Deepseek-v3	EX	91.9●	89.0	75.9	71.1	84.6
		EM	85.5	81.4	63.2	59.6●	75.8●
	Qwen-7B	EX	90.3	88.5●	54.8	45.9	76.4
		EM	31.0	21.0	16.7	2.4	19.7
	XiYanSQL-14B	EX	91.4	85.2	59.3	49.7	76.6
		EM	40.7	38.4	25.5	18.8	33.6
	Llama-8B	EX	86.3	78.5	61.8	43.6	72.0
		EM	18.8	12.9	14.6	0.6	12.6
DIN-SQL	GPT-4o-mini	EX	91.1	79.8	64.9	43.4	74.2
		EM	82.7	65.5	42.0	30.7	60.1
	Deepseek-v3	EX	89.5	88.3	69.0	59.0	80.6
		EM	76.2	42.2	33.3	10.0	43.7
	Qwen-7B	EX	82.3	81.4	59.8	55.4	73.8
		EM	36.7	34.5	17.8	3.6	27.3
	XiYanSQL-14B	EX	94.8	89.5	76.4	68.1	85.1
		EM	91.5	82.7	63.2	54.2	77.0
	Llama-8B	EX	82.3	80.9	59.2	55.4	73.5
		EM	35.8	35.3	20.5	3.7	27.9
ACT-SQL	GPT-4o-mini	EX	87.9	81.4	61.5	42.8	73.4
		EM	75.8	42.6	37.9	13.3	45.1
	Deepseek-v3	EX	91.1	83.6	56.9	50.0	75.5
		EM	33.5	30.3	15.5	9.0	25.2
	Qwen-7B	EX	89.9	86.3	56.9	51.2	76.6
		EM	32.3	29.1	17.8	1.2	23.5
	XiYanSQL-14B	EX	90.3	84.3	58.6	49.4	75.8
		EM	33.1	30.0	16.7	0.0	23.7
	Llama-8B	EX	89.9	85.4	61.5	48.2	76.5
		EM	33.9	28.3	13.2	1.8	22.8
C3-SQL	GPT-4o-mini	EX	92.7	85.0	77.6	62.0	81.9
		EM	80.2	43.5	35.6	18.1	46.9
	Deepseek-v3	EX	89.9	85.4	61.5	48.2	76.5
		EM	33.5	27.8	14.9	6.0	23.5
	Qwen-7B	EX	81.9	76.7	70.7	47.6	72.3
		EM	75.0●	41.7	42.0	10.2	44.7
	XiYanSQL-14B	EX	91.9	81.6	76.4	51.2	78.3
		EM	83.9	44.4	46.0	11.4	48.8
	Llama-8B	EX	82.6	76.2	69.8	45.7	71.8
		EM	74.2	41.5	41.9	10.2	44.4
SchemaRAG	GPT-4o-mini	EX	91.1	88.1	**83.3**	**72.9**	**85.6**
		EM	86.7	65.1	**64.0**	44.4	66.8
	Deepseek-v3	EX	91.2	**94.3**	**83.7**	**80.6**	**89.6**
		EM	**87.1**	**82.7**	**63.2**	54.2	75.9
	Qwen-7B	EX	**90.7**	87.4	**71.8**	**55.4**	**80.4**
		EM	74.2	**56.1**	**46.0**	**22.3**	**53.3**
	XiYanSQL-14B	EX	**98.0**	**95.7**	**87.4**	**86.1**	**93.3**
		EM	**97.6**	**89.7**	**82.2**	**75.3**	**88.0**
	Llama-8B	EX	**88.7**	**87.0**	**71.8**	**62.0**	**80.8**
		EM	**82.3**	**68.8**	**56.3**	**34.9**	**64.5**

G Detailed Ablation Study

This section presents the ablation studies conducted to validate the contribution of the individual components of the SchemaRAG framework. We analyze the performance degradation when the SchemaLinker, SAR and POSG modules are individually removed.

Table 8 shows the results of this study on the Spider dataset. It compares the full SchemaRAG model against variants without SchemaLinker (w/o SchemaLinker), without SAR (w/o SAR) and without POSG(w/o POSG), demonstrating the impact of each component on EX and EM.

Table 9 provides the corresponding ablation results on the BIRD dataset. The same model variants are tested to show the importance of each component in this more challenging environment. Bolded values indicate the highest score among the three variants (full and ablated) for EX and VES.

H SchemaRAG Prompt

This section provides a detailed explanation of the prompts used at each core stage of our SchemaRAG framework.

H.1 PromptSchema

The schema used by our study is organized as shown in Figure 2.

Table 7: Comparison of EX and VES with evidence on the BIRD dataset

Model	LLM	Metric	SIM	MOD	CHALL	TOTAL
TA-SQL	GPT-4o	EX	60.54	40.86	38.19	52.43
		VES	25.28	20.33	19.13	23.21
	Deepseek-v3	EX	62.92	49.03	36.81	56.24
		VES	27.67	25.39	13.86	25.66
	Qwen-7B	EX	43.14	19.57	12.50	33.11
		VES	16.96	7.50	4.56	12.92
	XiYanSQL-14B	EX	47.03	24.95	19.44	37.74
		VES	20.42	13.97	9.34	17.42
	Llama-8B	EX	41.62	19.14	12.50	32.06
		VES	15.12	7.75	4.61	11.89
DIN-SQL	GPT-4o	EX	55.78	35.27	27.08	46.86
		VES	22.21	19.15	9.90	20.11
	Deepseek-v3	EX	54.70	38.28	27.08	47.11
		VES	21.15	17.05	9.60	18.81
	Qwen-7B	EX	36.86	14.84	8.33	27.50
		VES	12.63	5.83	3.11	9.67
	XiYanSQL-14B	EX	52.43	34.62	22.22	44.18
		VES	26.60	18.90	8.86	22.58
	Llama-8B	EX	32.32	20.22	14.58	26.98
		VES	12.76	8.61	4.00	10.67
ACT-SQL	GPT-4o	EX	56.86	35.70	25.69	47.51
		VES	25.77	17.14	15.14	22.15
	Deepseek-v3	EX	53.68	36.72	26.57	45.98
		VES	21.06	17.12	9.83	18.80
	Qwen-7B	EX	35.42	13.68	8.21	26.27
		VES	11.97	6.01	4.21	9.43
	XiYanSQL-14B	EX	53.51	34.84	27.08	43.16
		VES	24.05	15.62	11.86	20.35
	Llama-8B	EX	32.32	20.22	14.58	26.98
		VES	10.65	7.63	4.22	9.12
C3-SQL	GPT-4o	EX	54.78	36.27	26.08	44.46
		VES	20.21	19.15	9.50	18.86
	Deepseek-v3	EX	52.76	35.28	25.08	44.85
		VES	21.25	16.15	8.60	18.50
	Qwen-7B	EX	37.97	15.95	8.43	28.51
		VES	12.63	5.83	3.11	9.67
	XiYanSQL-14B	EX	51.63	30.25	21.77	42.34
		VES	26.50	17.80	7.96	22.10
	Llama-8B	EX	32.42	20.42	14.39	27.08
		VES	11.37	8.15	3.85	9.66
SchemaRAG	GPT-4o	EX	**76.54**	**60.43**	**47.22**	**68.90**
		VES	**34.76**	**29.74**	**21.40**	**31.98**
	Deepseek-v3	EX	**76.11**	**60.00**	**45.83**	**68.38**
		VES	**33.81**	**29.43**	**18.51**	**31.05**
	Qwen-7B	EX	**64.86**	**43.01**	**25.00**	**54.50**
		VES	**29.43**	**21.95**	**7.43**	**25.10**
	XiYanSQL-14B	EX	**73.30**	**56.13**	**43.06**	**65.25**
		VES	**34.37**	**38.96**	**21.31**	**34.54**
	Llama-8B	EX	**71.24**	**47.31**	**32.64**	**60.37**
		VES	**33.06**	**27.67**	**16.99**	**29.92**

H.2 SchemaLinker

In the first stage of fine-tuning, the prompt-output pairs were formatted as shown in Figure 3. During the second stage of multi-task fine-tuning, for the error identification task, the prompt-output pairs were formatted as shown in Figure 4.

Table 8: Ablation study of SchemaRAG on the Spider dataset

Model	LLM	Metric	Easy	Medium	Hard	Extra	All
w/o SchemaLinker	GPT-4o-mini	EX	91.1	81.1	83.3●	72.9	85.6
		EM	86.7	65.1●	64.0●	44.4	66.9●
	Deepseek-v3	EX	92.3●	89.7	77.6	71.1	85.3
		EM	87.1	84.8●	66.1●	59.6●	78.1●
	Qwen-7B	EX	87.9	81.4	74.1●	59.0●	78.1
		EM	72.6	52.9	51.7●	25.9●	53.1
	XiYanSQL-14B	EX	97.2	94.6	87.9●	81.9	92.1
		EM	97.2	92.6●	85.6●	74.7	89.7●
	Llama-8B	EX	87.1	80.7	69.0	64.5●	77.7
		EM	83.5●	61.0	50.6	35.5●	60.5
w/o SAR	GPT-4o-mini	EX	88.7	84.4	69.7	61.6	79.4
		EM	32.2	33.0	22.5	4.1	26.7
	Deepseek-v3	EX	89.6	89.7	65.2	58.9	80.8
		EM	27.8	27.2	5.6	4.1	20.2
	Qwen-7B	EX	86.3	82.5	51.7	47.0	72.5
		EM	37.1	26.5	9.2	6.0	22.0
	XiYanSQL-14B	EX	97.6	94.8	85.6	86.1	92.6
		EM	97.6	90.8	79.3	78.3●	88.5
	Llama-8B	EX	75.4	61.4	47.7	39.2	58.9
		EM	27.0	13.7	6.3	5.4	14.3
w/ POSG	GPT-4o-mini	EX	92.1	81.1	82.6	76.6	86.5
		EM	85.6	64.3	62.2	45.3	65.0
	Deepseek-v3	EX	91.6	90.7	78.6	72.5	86.6
		EM	86.9	82.2	62.6	53.3	74.2
	Qwen-7B	EX	87.8	82.5	70.9	54.2	79.5
		EM	73.2	53.1	45.2	21.2	53.2
	XiYanSQL-14B	EX	97.2	94.6	87.1	82.3	92.6
		EM	96.4	88.5	81.7	75.2	86.5
	Llama-8B	EX	87.9	82.1	69.2	61.4	79.8
		EM	81.5	64.3	54.4	32.2	62.2
SchemaRAG	GPT-4o-mini	EX	**92.2**	**91.1**	81.1	**81.7**	**87.7**
		EM	**92.2**	62.9	62.3	**48.3**	66.1
	Deepseek-v3	EX	91.2	**94.3**	**83.7**	**80.6**	**89.6**
		EM	**87.1**	82.7	63.2	54.2	75.4
	Qwen-7B	EX	**90.7**	**87.4**	71.8	55.4	**80.5**
		EM	**74.2**	**56.1**	46.0	22.3	**53.3**
	XiYanSQL-14B	EX	**98.0**	**95.7**	87.4	**86.1**	**93.3**
		EM	**97.6**	89.7	82.2	75.3	88.0
	Llama-8B	EX	**88.7**	**87.0**	**71.8**	62.0	**80.9**
		EM	82.3	**68.8**	**56.3**	34.9	**64.5**

Question: How many heads of the departments are older than 56?

database .....

<think>

1. Understand the key concepts in the question:
   - `heads of the departments`: Find the head of each department (head_ID in head table)
   - `older than 56`: corresponds to the age column in the head table
   - `How many`: Count the number of department heads meeting the criteria

2. Analyze database table relationships:
   - Head information stored in the head table
   - Head-department relationship in management table (management.head_ID = head.head_ID)
   - Department info in department table (management.department_ID = department.Department_ID)

3. Key field for filtering: **head.age** (determines if > 56)
   The head table stores personal information directly related to age filtering.

</think>

The key field for determining whether the person in charge is older than 56 is head.age, as the head table stores the relevant personal information. The management table links the person in charge with the department, but it does not directly provide the age information. Therefore, head.age is the most directly related field for answering this question. The key field matching the question is: [head.age].

Figure 3: An example of a prompt-response pair used during the first stage of fine-tuning.

```
// Database definition
DATABASE thrombosis_prediction {

  // Examination table - Stores patient examination information
  TABLE examination {
    id: INTEGER // Examination ID → [14872, 48473]
    examination_date: DATE // Examination date → [1997-05-27, 1992-12-21]
    acl_igg: REAL // Anticardiolipin antibody (IgG) → [1.3, 4.3]
    acl_igm: REAL // Anticardiolipin antibody (IgM) → [1.6, 4.6]
    ana: INTEGER // Antinuclear antibody → [256, 0]
    ana_pattern: TEXT // ANA test pattern → ["P", "P,S"]
    acl_iga: INTEGER // Anticardiolipin antibody (IgA) concentration → [0, 3]
    diagnosis: TEXT // Diagnosis result → ["MCTD", "AMI", "SLE"]
    kct: TEXT // Coagulation measurement → ["-", "+"]
    rvvt: TEXT // Coagulation measurement → ["-", "+"]
    lac: TEXT // Coagulation measurement → ["-", "+"]
    symptoms: TEXT // Symptoms → ["AMI", "CNS lupus"]
    thrombosis: INTEGER // Thrombosis formation indicator → [1, 0]
  }

  // Patient table - Stores basic patient information
  TABLE patient {
    id: INTEGER PRIMARY_KEY // Patient ID → [14872, 48473]
    sex: TEXT // Gender → ["F", "M"]
    birthday: DATE // Date of birth → [1934-02-13, 1937-05-02]
    description: TEXT // Description information → [ ... ]
    first_date: DATE // First visit date → [1994-02-14, 1996-12-01]
    admission: TEXT // Admission information → ["+", "-"]
    diagnosis: TEXT // Diagnosis → ["RA susp.", "PSS"]
  }

  // Foreign key constraints definition
  FOREIGN_KEYS {
    examination.id → patient.id
  }

} // END DATABASE thrombosis_prediction
```

Figure 2: An example of PromptSchema.

Question: How many heads of the departments are older than 56?

database .....

Error Analysis

Step 1 Error: think_pre incorrectly mapped 'heads of the departments' directly to head.name, when it should have focused on head_ID and its associated relationships, rather than the name.

Step 3 Error: think_pre mistakenly used head.name as a retrieval field, but the question only required a count, so COUNT(*) would suffice without needing head.name. So 'head.name' should not be included in 'schema_links_pred'.

Figure 4: An example of a prompt-response pair used during the second stage of multi-task fine-tuning.

Table 9: Ablation study of SchemaRAG on the BIRD dataset

Model	LLM	Metric	SIM	MOD	CHALL	TOTAL
w/o SchemaLinker	GPT-4o	EX	72.22	56.77	47.22	65.19
		VES	32.63	29.52	23.20●	30.80
	Deepseek-v3	EX	72.11	56.13	45.14	64.73
		VES	32.92	29.89●	21.55●	30.93
	Qwen-7B	EX	64.54	43.66●	32.64●	55.22
		VES	31.03●	26.07●	17.06●	28.22●
	XiYanSQL-14B	EX	70.27	56.13	41.67	63.30
		VES	31.04	34.85	21.65●	31.32
	Llama-8B	EX	65.08	43.87	29.86	55.35
		VES	30.79	29.59●	15.06	28.95
w/o SAR	GPT-4o	EX	61.41	41.29	31.94	52.54
		VES	27.26	19.47	16.50	23.89
	Deepseek-v3	EX	60.86	40.86	31.94	52.09
		VES	25.00	18.09	15.35	22.00
	Qwen-7B	EX	55.78	35.27	25	46.68
		VES	23.07	21.22	9.01	21.19
	XiYanSQL-14B	EX	69.30	53.76	41.67	61.99
		VES	30.47	29.25	17.45	28.88
	Llama-8B	EX	55.14	35.27	26.39	46.41
		VES	21.62	20.65	8.78	20.12
w/o POSG	GPT-4o	EX	73.24	58.68	47.02	66.80
		VES	32.52	29.38	21.23	30.88
	Deepseek-v3	EX	74.31	58.73	45.33	66.28
		VES	31.92	29.11	17.50	31.00
	Qwen-7B	EX	65.22	42.66	26.78	54.12
		VES	29.98	19.88	9.24	26.23
	XiYanSQL-14B	EX	71.54	55.23	42.57	65.04
		VES	32.24	35.75	20.42	33.63
	Llama-8B	EX	66.24	43.87	31.86	59.26
		VES	32.22	26.55	15.72	29.82
SchemaRAG	GPT-4o	EX	**76.54**	**60.43**	**47.22**	**68.90**
		VES	**34.76**	**29.74**	21.40	**31.98**
	Deepseek-v3	EX	**76.11**	**60.00**	**45.83**	**68.38**
		VES	**33.81**	29.43	18.51	**31.05**
	Qwen-7B	EX	**65.95**	43.01	27.08	**55.35**
		VES	30.58	20.08	9.58	25.42
	XiYanSQL-14B	EX	**73.3**	**56.13**	**43.06**	**65.25**
		VES	**34.37**	**38.96**	21.31	**34.54**
	Llama-8B	EX	**71.24**	**47.31**	**32.64**	**60.37**
		VES	**33.06**	27.67	**16.99**	**29.92**

H.3 SQL Generation

```
Question: How many singers do we have?
database .....
schema used: [singer]

📋 In-Context Learning Examples

Question: How many employees are currently working?
SQL: SELECT count(*) FROM Employees

Question: What is the total number of employees?
SQL: SELECT count(*) FROM EMPLOYEES

Question: What is the total number of students enrolled?
SQL: SELECT count(*) FROM Students
```

Figure 5: An example of the prompt used for the final Text-to-SQL task.

The prompt used in the final Text-to-SQL task is shown in the Figure 5.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, et al. 2024. GPT-4 Technical Report. arXiv:2303.08774
[2] DeepSeek-AI, Aixin Liu, Bei Feng, et al. 2025. DeepSeek-V3 Technical Report. arXiv:2412.19437
[3] Xuemei Dong, Chao Zhang, Yuhang Ge, et al. 2023. C3: Zero-shot Text-to-SQL with ChatGPT. arXiv:2307.07306
[4] Yingqi Gao, Yifu Liu, Xiaoxia Li, et al. 2025. A Preview of XiYan-SQL: A Multi-Generator Ensemble Framework for Text-to-SQL. arXiv:2411.08599
[5] Team GLM, Aohan Zeng, Bin Xu, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. arXiv:2406.12793
[6] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783
[7] Haoyang Li, Jing Zhang, Hanbing Liu, et al. 2024. CodeS: Towards Building Open-source Language Models for Text-to-SQL. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 127.
[8] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. In *NeurIPS*, Vol. 36.
[9] Ge Qu, Jinyang Li, Bowen Li, et al. 2024. Before Generation, Align it! A Novel and Effective Strategy for Mitigating Hallucinations in Text-to-SQL Generation. In *Findings of ACL*. 5456–5471.
[10] An Yang, Baosong Yang, Beichen Zhang, et al. 2025. Qwen2.5 Technical Report. arXiv:2412.15115
[11] Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023. ACT-SQL: In-Context Learning for Text-to-SQL with Automatically-Generated Chain-of-Thought. In *Findings of EMNLP*. 3501–3532.