
## 2025 하계 세미나

# Applying Reasoning in Text2SQL task

NLP&AI 강명훈

Natural Language Processing & Artificial Intelligence

고려대학교 KOREA UNIVERSITY

# What is Text2SQL?

- **Leveraging structural reasoning capability with LM**

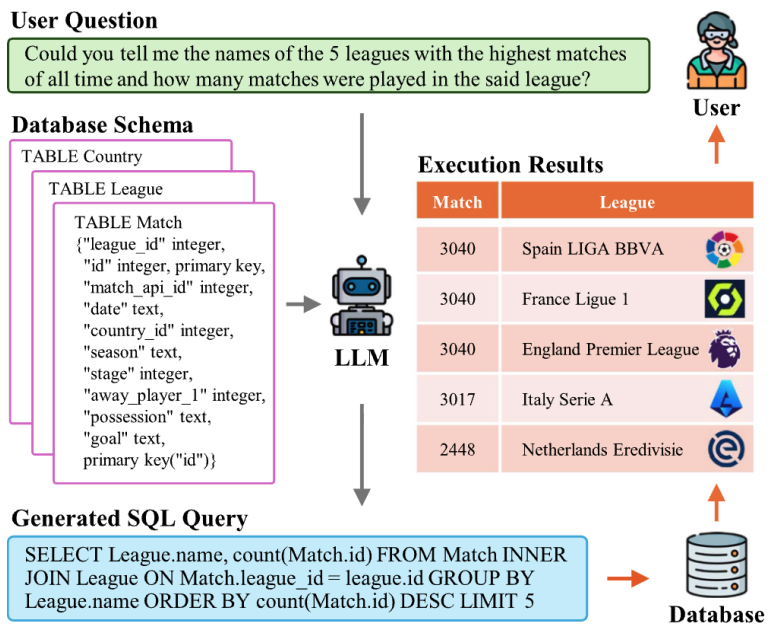



Fig. 1. An example of LLM-based text-to-SQL is selected from the BIRD [1] dataset. A user asks a question about football leagues. The LLM takes this question along with the schema of the corresponding database as input and generates an SQL query as output. The generated SQL query can be executed in the database, retrieves the content *"The 5 leagues with the highest matches"*, providing the answer to the user's question.

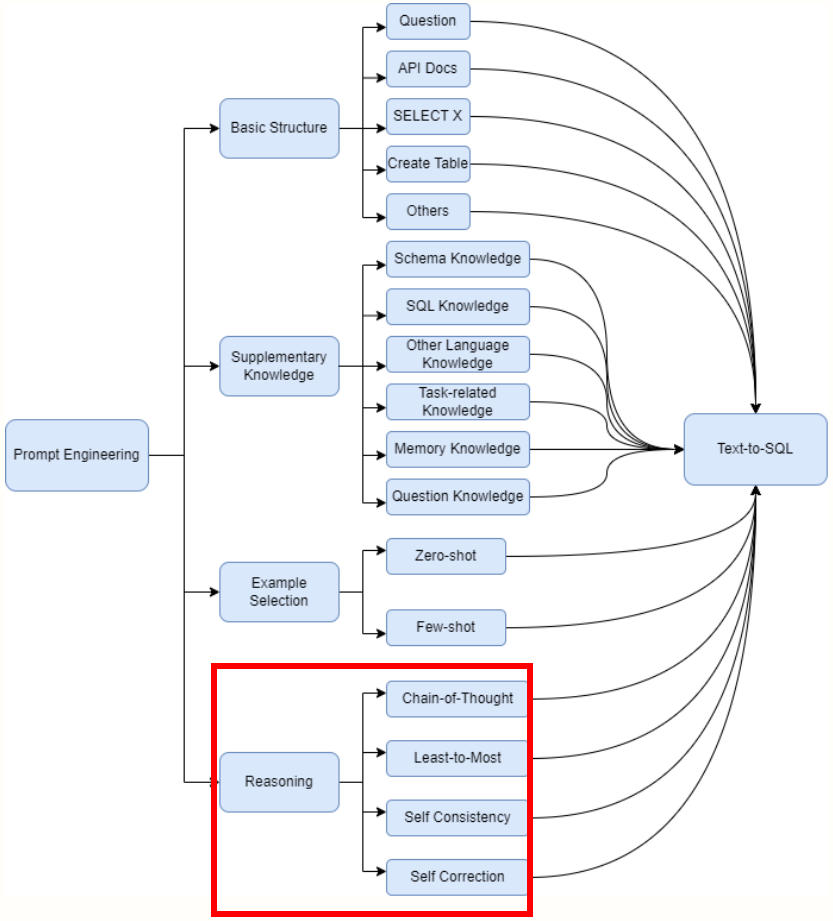
- Text2SQL (NL2SQL): 사용자가 자연어로 물어본 질의 (Text)의 맥락과 의도에 부합하는 근거 답변을 구조화된 자료 DB로부터 추출할 수 있는 SQL 질의문을 생성하는 과업

- 정확한 Text2SQL 과업을 수행하기 위해서는 다음의 challenge (sub-task) 수행이 필수적
  - Linguistic Complexity and Ambiguity: 자연어로 표현된 사용자 질의 속 맥락과 의도를 정확하게 파악하는 능력 필요
  - Schema Understanding and Representation: 질의에 부합하는 답변을 생성하기 위해 참조하는 DB의 구조, 내용을 이해할 수 있어야 함. 이러한 이해는 단일 Table 구조, 내용 이해부터 Table간 관계 파악 등을 포함
  - Rare and Complex SQL Operations: 사용자의 질의 해결을 위해 여러 sub-query를 생성하는 Nested-query, Window function 등과 같이 복잡한 SQL 구문 생성 능력
  - Cross-Domain Generalization: System이 학습하거나 추론하는 DB의 domain에 관계없이 일반화된 성능을 발휘할 수 있어야 함

Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., & Huang, X. (2025). Next-generation database interfaces: A survey of llm-based text-to-sql. IEEE Transactions on Knowledge and Data Engineering.

# What is Text2SQL?

- **How to resolve these challenges for successful Text2SQL performance?**




- TextSQL에서 요구되는 4가지 challenge를 달성하기 위한 다양한 methodology가 존재할 수 있음
  - Basic Structure: SQL과 관련된 기초 지식을 주입 및 전달하는 방법
  - Supplementary Knowledge: SQL과 관련된 보조 지식을 주입 및 전달하여 SQL 구문을 정확하게 생성하게 하는 방법
  - Example Selection: ICL 방식으로 SQL 구문을 생성함에 있어서 일반화된 생성을 도모하는 example sampling 방법
  - Reasoning: 정확한 SQL 구문 생성을 위한 fine-grained 추론 과정 정립에 관한 방법

- 최근 Reasoning trend와 맞물려 Text2SQL 과업에서 LLM, LRM을 활용하여 어떻게 정확한 SQL 구문을 생성할 것인가와 관련된 연구가 활발하게 진행중

Shi, L., Tang, Z., Zhang, N., Zhang, X., & Yang, Z. (2025). A survey on employing large language models for text-to-sql tasks. ACM Computing Surveys, 58(2), 1-37.

# Clue for applying reasoning in Text2SQL

**LinkAlign: Scalable Schema Linking for Real-World Large-Scale Multi-Database Text-to-SQL**

**Yihan Wang**[1,2]* **Peiyu Liu**[3]* **Xin Yang**[1]†
[1]China Academy of Information and Communications Technology
[2]Renmin University of China [3]University of International Business and Economics
yihan3123@gmail.com liupeiyustu@163.com yangxincps@163.com

Agentic 방식으로 schema linking 과업에서 reasoning 능력을 향상시켜 Tex2SQL 성능을 높여보자!

**SQL-R1: Training Natural Language to SQL Reasoning Model By Reinforcement Learning**

**Peixian Ma**[1,2], **Xialie Zhuang**[1,3], **Chengjin Xu**[1,4,*], **Xuhui Jiang**[1,4], **Ran Chen**[1], **Jian Guo**[1]
[1]IDEA Research, International Digital Economy Academy
[2]The Hong Kong University of Science and Technology (Guangzhou)
[3]University of Chinese Academy of Sciences
[4]DataArc Tech Ltd.
pma929@connect.hkust-gz.edu.cn, xuchengjin@idea.edu.cn

Text2SQL 특화 강화학습을 설계하여 LLM의 structural reasoning 능력을 향상시켜보자

# Reasoning Text2SQL

## LinkAlign: Scalable Schema Linking for Real-World Large-Scale Multi-Database Text-to-SQL

**Yihan Wang**[1,2]* **Peiyu Liu**[3]* **Xin Yang**[1]†
[1]China Academy of Information and Communications Technology
[2]Renmin University of China [3]University of International Business and Economics
yihan3123@gmail.com liupeiyustu@163.com yangxincps@163.com

**EMNLP 2025 Main poster**

# LinkAlign

- **Motivations: Schema linking is a critical bottleneck for applying Text2SQL in real-world**

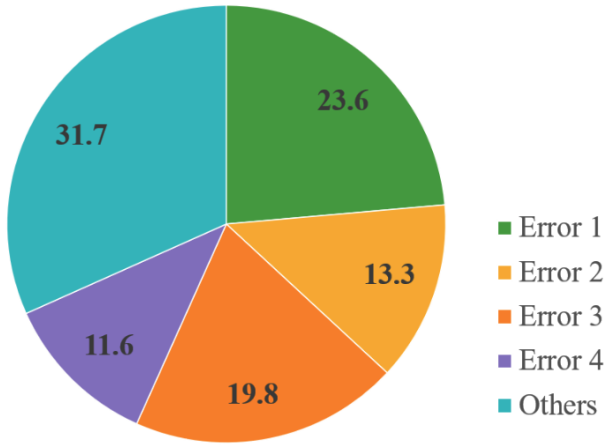



Figure 4: Error Distribution in Failed Cases.

- 기존 방법은 Multi-database 상황에서 사용자 의도에 부합하는 SQL 구문을 생성하는 능력이 부족함을 주장

- 사전 실험에서 Schema linking의 error로 인한 SQL query 생성 실패가 60%에 달함을 확인

  **Fine-grained Schema linking error analysis**
  - Error1: 질의와 관련 있는 DB schema 검색 실패
  - Error2: 질의와 관련 없는 DB schema를 검색
  - Error3: 질의와 관련 없는 Table을 linking
  - Error4: 질의와 관련 없는 Column을 linking

← DB linking error가 36.9%에 달하는 만큼 real-world 상황에서 Text2SQL을 수행하기 위해서는 정확한 Schema linking이 선행되어야 함

# LinkAlign - Method

- **Resolving Schema linking bottleneck with Agentic**

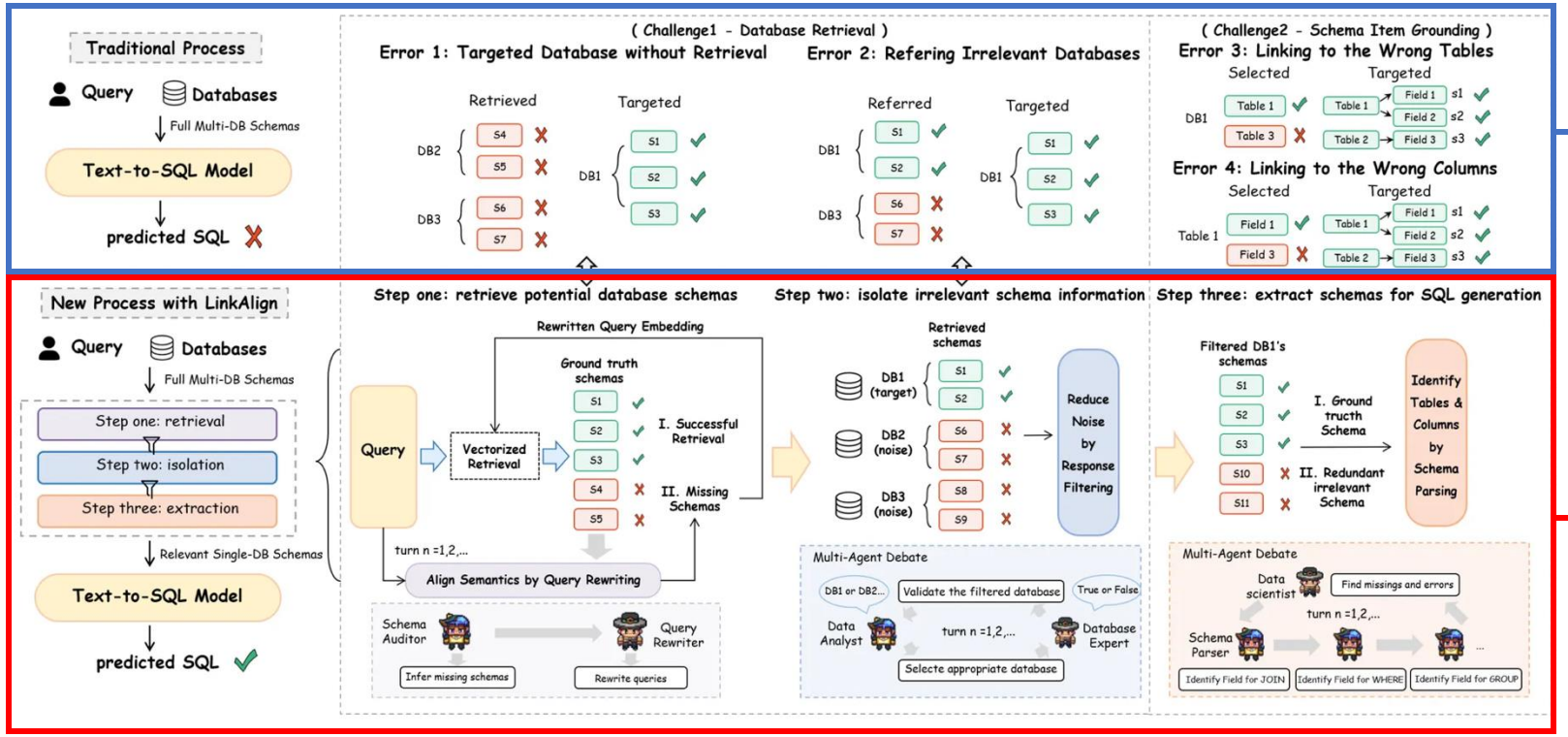



Figure 2: Overview of the LinkAlign framework including three core components.

Multi-database 상황에 대응하지 못하는 기존 방법의 erroneous schema linking

Retrieval, Isolation, Extraction의 3가지 step을 Multi-agent를 적용하여 Multi-database 상황에서 schema linking을 수행하는 LinkAlign 제안

# LinkAlign - Method

- **Problem Definition**

Multi-database 상황에서 Schema linking을 수행하기 위한 problem definition은 아래와 같음

$D = \{D_1, D_2, \ldots, D_N\}$: N개의 DB로 구성된 DB의 집합, Multi-database
$S = \{S_1, S_2, \ldots, S_N\}$: DB별 meta data를 담은 schema의 집합
- $S_i = \{T_i, C_i\}$ where

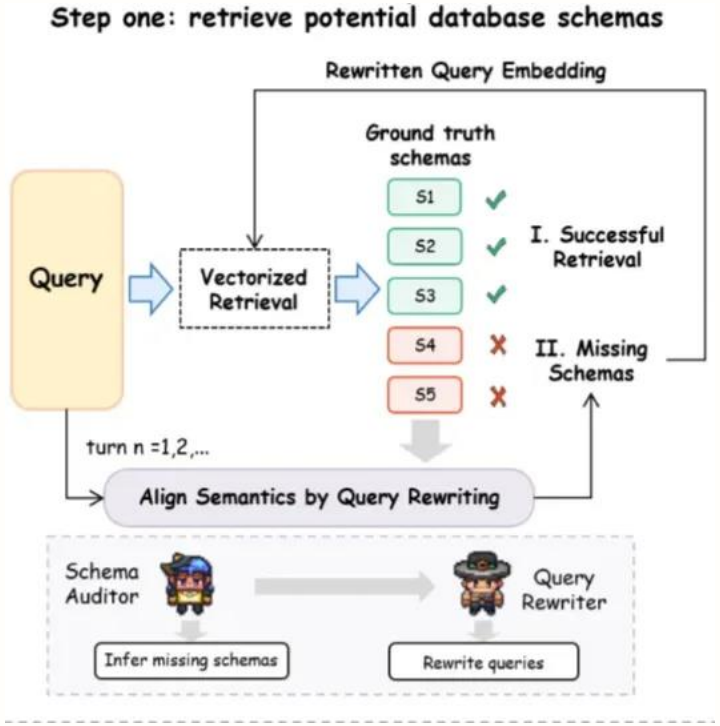
$T_i = \{T_1^i, T_2^i, \ldots, T_{|T_i|}^i\}$: $|T_i|$개의 table로 구성된 Table 집합
$C_i = \{C_1^i, C_2^i, \ldots, C_{|C_i|}^i\}$: $|C_i|$개의 column으로 구성된 Column 집합

System은 Embedding model $E$와 $LLM$ 을 사용해서 사용자 질의 $Q$, schema 집합 $S$, 보조 정보 $c$ 가 주어질 때 fine-grained schema subset $\acute{S}$를 추출해야 함

$$S' = f_{\text{parser}}(S, Q, c \mid E, LLM), \qquad (1)$$

# LinkAlign - Method

- **Step one: retrieve potential database schemas (Retrieval)**



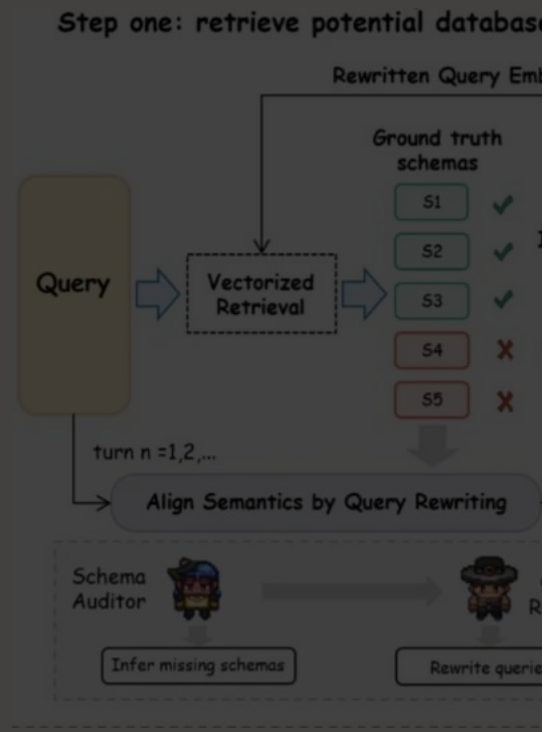

$$Z = \bigcup_{t=0}^{T} f_{\text{retriever}}\left(S, Q_t, c \mid E\right), \qquad (2)$$

- 사용자 질의를 해결할 수 있는 DB의 집합 $Z$를 얻을 수 있도록 **반복적으로 검색을 수행**하는 단계
- Embedding model $E$를 활용해 Top-k schema를 우선 검색
- Top-k schema, Q를 입력으로 받아서 **Schema Auditor Agent**는 missing schema가 있는지를 판별
- Missing schema가 존재하는 **경우 Query Rewriter Agent**는 $E$가 missing schema를 검색할 수 있도록 query를 rewriting
- Agent가 missing schema가 없다고 판단할 때 까지 반복적인 검색을 수행하여 질의 해결에 필요한 DB가 무조건 포함 되도록 하는 greedy한 방법

# LinkAlign - Method

- **Step one: retrieve potential database schemas (Retrieval)**

**Step 수행에 따른 Query가 rewrite된 예시**




➢ **User Query** $Q_0$: Which semester the master and the bachelor both got enrolled in?
**Missing Schema**: degree_programs (degree_type)
[1] **Rewrite** $Q_1$: In a database with degree_programs, how to find semesters where both master's degree_type and bachelor's degree_type programs exist? Group by enrollment_semester with checks for both program types.

---

**Missing Schema**: enrollment_records (semester)
[2] **Rewrite** $Q_2$: In a database with enrollment_records, how to find semesters where both master and bachelor students enrolled? Group by semester and filter for overlapping enrollments.

(2)

있도록 **반복적으로 검색을 수행**하는

검색

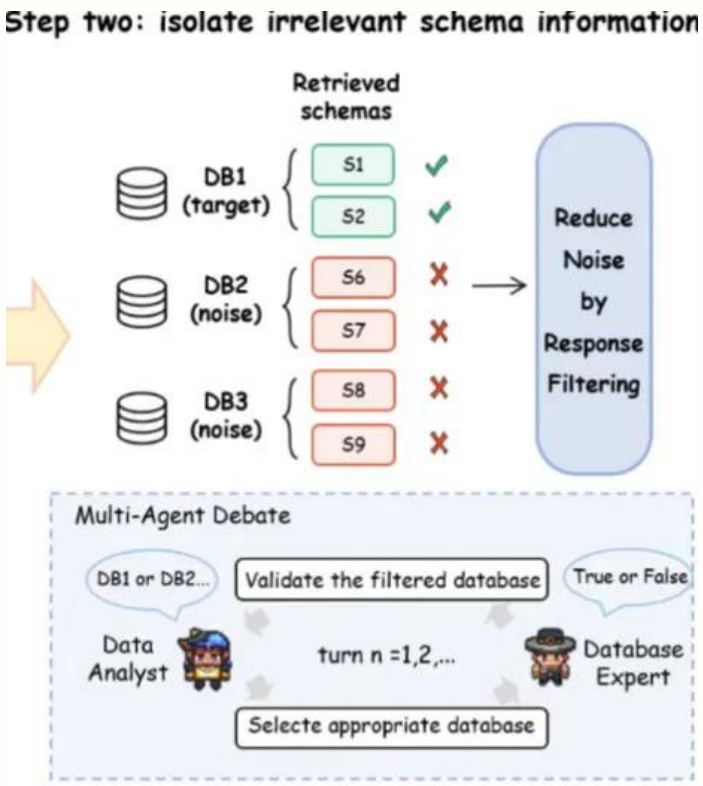
**Agent**는 missing schema가

t는 $E$가 missing schema를 검색할

복적인 검색을 수행하여 질의 해결에

# LinkAlign - Method

- **Step two: isolate irrelevant schema information (Isolation)**

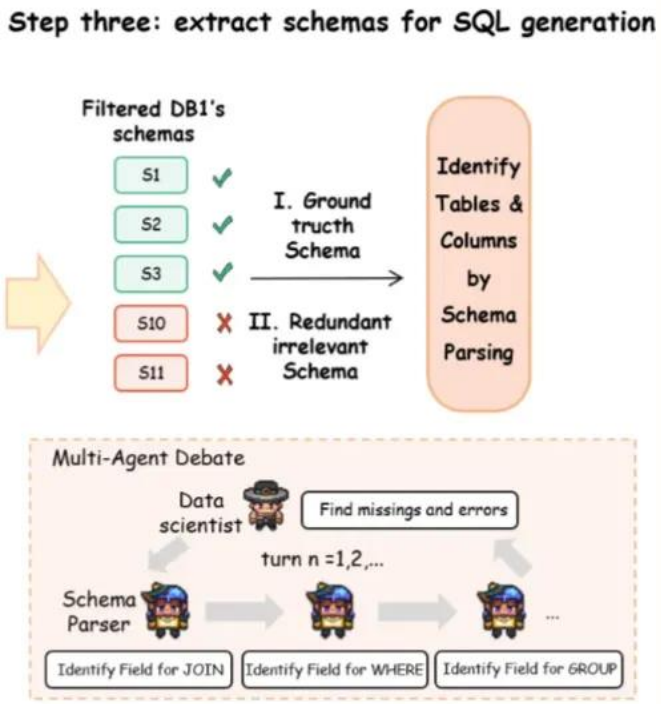



$$D_t = \arg \max_{1<i<N} P_M(D_i \mid Q_0, Z), \qquad (3)$$

- 사용자 질의와 관련된 DB 집합 $Z$에서 실제 질의를 해결할 수 있는 단 하나의 DB $D_t$를 추출
- LLM $M$으로 구현된 Data Analyst, Database Expert와의 Iterative Debate를 통해 $D_t$를 추출하도록 함
- **Data Analyst Agent**는 먼저 $Z$에 속한 Database별 schema와 query를 보고 query와 관련 있는 Database를 선정
- 선정된 Database는 **Database Expert Agent**는 해당 DB의 schema를 보고 질의와 관련이 있는지 검수를 진행
- 사용자가 설정한 turn동안 debate를 진행하며 debate가 종료된 뒤에는 debate를 summarization하여 concensus를 얻고 해당 concensus를 바탕으로 핵심 Database $D_t$를 추출

# LinkAlign - Method

- **Step three: extract schemas for SQL generation.**




$$S'_{\hat{u}} = \{T_i^{\hat{u}}, C_i^{\hat{u}} | \mathbb{I}(Q_0, C_i^{\hat{u}}) = 1\}\}, \qquad (4)$$

- 질의 관련 핵심 DB $D_t$ 에서 실제 질의를 해결할 수 있는 Table, column을 추출하는 작업
- LLM $M$으로 구현된 Schema Parser, Data scientist 와의 Iterative Debate를 통해 salient subset schema $\acute{S}_{\tilde{u}}$ 를 추출
- **Schema Parser Agent**는 Query decomposition, Schema harvesting, Validation의 과정을 통해서 주어진 schema내에 관련 있는 Table, Column을 확인
- **Data scientist Agent**는 Schema Parser Agent의 schema linking 결과를 평가하여 linking된 table, column의 query 관련성 여부를 1, 0으로 이진 분류
- 사용자가 설정한 turn에 도달하면 debate를 종료하고 최종 schema linking 결과를 추출

# LinkAlign – Experimental Setups

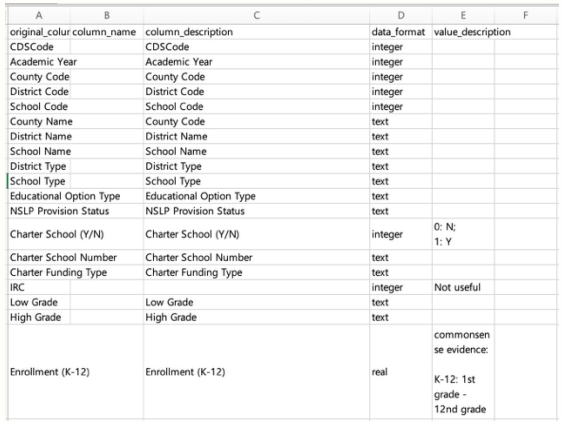
- **Experimental dataset**

Spider

| Segment | Size |
|---|---|
| Train | 8,659 |
| Validation | 1,034 |

BIRD

| Segment | Size |
|---|---|
| Train | 9,078 |
| Validation | 1,534 |

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | original_colur | column_name | column_description | data_format | value_description | |
| 2 | CDSCode | | CDSCode | integer | | |
| 3 | Academic Year | | Academic Year | integer | | |
| 4 | County Code | | County Code | integer | | |
| 5 | District Code | | District Code | integer | | |
| 6 | School Code | | School Code | integer | | |
| 7 | County Name | | County Code | text | | |
| 8 | District Name | | District Name | text | | |
| 9 | School Name | | School Name | text | | |
| 10 | District Type | | District Type | text | | |
| 11 | School Type | | School Type | text | | |
| 12 | Educational Option Type | | Educational Option Type | text | | |
| 13 | NSLP Provision Status | | NSLP Provision Status | text | | |
| 14 | Charter School (Y/N) | | Charter School (Y/N) | integer | 0: N;<br>1: Y | |
| 15 | Charter School Number | | Charter School Number | text | | |
| 16 | Charter Funding Type | | Charter Funding Type | text | | |
| 17 | IRC | | | integer | Not useful | |
| 18 | Low Grade | | Low Grade | text | | |
| 19 | High Grade | | High Grade | text | | |
| 20 | Enrollment (K-12) | | Enrollment (K-12) | real | commonsense evidence:<br><br>K-12: 1st grade - 12nd grade | |

Metadata

{'question_id': 0,
'db_id': 'california_schools',
'question': 'What is the highest eligible free rate for K-12 students in the schools in Alameda County?',
'evidence': 'Eligible free rate for K-12 = `Free Meal Count (K-12)` / `Enrollment (K-12)`',
'SQL': "SELECT `Free Meal Count (K-12)` / `Enrollment (K-12)` FROM frpm WHERE `County Name` = 'Alameda' ORDER BY (CAST(`Free Meal Count (K-12)` AS REAL) / `Enrollment (K-12)`) DESC LIMIT 1",
'difficulty': 'simple'}

# LinkAlign– Experimental Setups

- **Implementation details**

- Retrieval: bge-large-en-v1.5
  - top-k: 5
- Schema linking (Step 1 ~ Step 3): GLM-4-air
- SQL generation: DeepSeek-V3, DeepSeeek-R1, Qwen-72B
  - DIN-SQL 방법론을 적용해서 SQL generation을 수행

- **Metric**

Schema Linking Evaluation Metrics

- **Locate Accuracy (LA):** Schema linking 추론 결과 정확한 database가 추출되었는지 여부 확인
- **Exact Matching (EM):** Schema linking 추론 결과가 GT schema linking결과와 동일한지 여부 확인 (Column까지 모두 정확하게 맞춘 경우)
- **Recall:** Schema linking 추론 결과가 GT schema linking 결과와 overlap 되는 비율

SQL Query generation Evaluation Metric

- **Execution Accuracy (EX)**

$$\mathrm{EX} = \frac{\sum_{i=1}^{N} \mathbb{1}(\hat{V}_i, V_i)}{N}$$

$$\mathbb{1}(\hat{V}_i, V_i) = \begin{cases} 1 & \text{if} \quad \hat{V}_i = V_i \\ 0 & \text{if} \quad \hat{V}_i \neq V_i \end{cases}$$

# LinkAlign– Experimental Setups

- **Baselines**

DIN-SQL

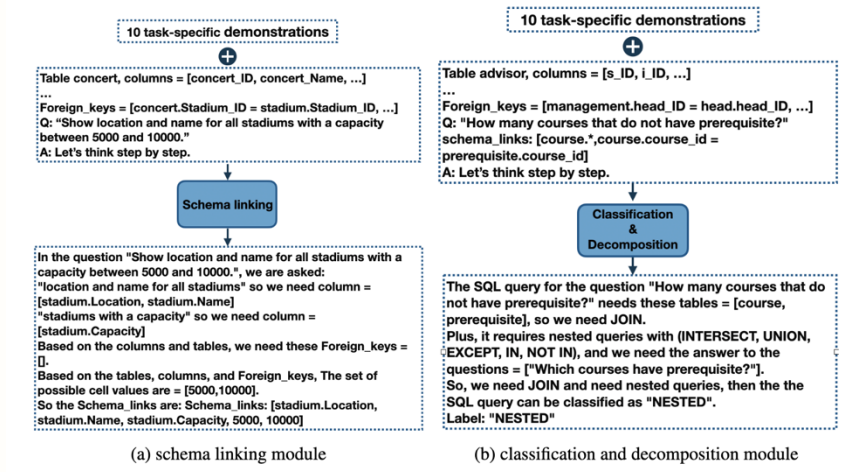



Figure 3: Examples showing the input and output of schema linking (left) and classification and decomposition (right)

- In-context learning 방식으로 Schema linking을 수행
- Schema linking 결과와 질의를 참조하여 생성할 SQL의 수준을 Easy / Non-nested complex / Nested complex 로 분류
- 분류 수준별 few-shot example을 활용하여 SQL 구문 생성

MCS-SQL

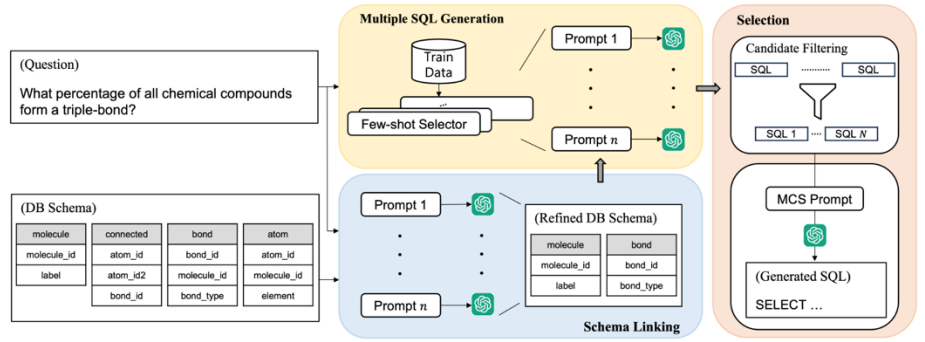



Figure 1: Overview of the proposed methodology, including three steps: schema linking, multiple SQL generation, and selection.

- Schema linking, SQL generation을 수행할 때 **입력의 형태를 다양화해서 투과한 multiple prompt** 결과를 활용
- 이를 통해 보다 일관적인 Schema linking, SQL query 생성 결과를 얻고자 함

# LinkAlign – Result

- **Main result: Schema linking performance**

| Approach | Spider | | | Bird | | | AmbiDB | | |
|---|---|---|---|---|---|---|---|---|---|
| | LA | EM | Recall | LA | EM | Recall | LA | EM | Recall |
| LlamaIndex | | | | | | | | | |
| DIN-SQL | 80.0 | 26.8 | 62.4 | 68.8 | 5.1 | 31.3 | 59.7 | 13.3 | 44.2 |
| PET-SQL | 84.1 | 38.6 | 67.2 | 77.1 | 8.2 | 39.7 | 66.4 | 22.0 | 50.2 |
| MAC-SQL | 82.3 | 17.3 | 42.8 | 75.0 | 5.7 | 34.5 | 65.1 | 9.7 | 30.8 |
| MCS-SQL | 81.0 | 24.3 | 73.2 | 73.7 | 13.9 | 56.1 | 61.9 | 13.7 | 54.8 |
| RSL-SQL | 74.8 | 29.1 | 76.1 | 80.0 | 16.1 | 61.8 | 62.4 | 17.9 | **59.6** |
| Pipeline(ours) | 85.4 | 37.4 | 65.9 | 66.8 | 8.6 | 38.1 | 69.4 | 20.3 | 50.4 |
| Agent(ours) | **86.4** | **47.7** | **80.7** | **83.4** | **22.1** | **64.9** | **67.6** | **22.4** | 56.9 |

Table 2: Comparison of LA, EM and Recall across different methods in multi-database scenario.

| Approach | Spider-dev | | | Bird-dev | | | AmbiDB | | |
|---|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | EM | Precision | Recall | EM | Precision | Recall | EM |
| DIN-SQL | 83.9 | 73.2 | 40.4 | 79.9 | 55.7 | 13.1 | 86.6 | 76.9 | 44.2 |
| PET-SQL | **84.8** | 73.9 | 33.4 | **81.6** | 64.9 | 25.9 | **90.2** | 78.3 | 39.5 |
| MAC-SQL | 75.0 | 66.8 | 24.4 | 76.3 | 56.2 | 9.1 | 79.8 | 69.6 | 30.1 |
| MCS-SQL | 66.7 | 85.0 | 29.8 | 79.6 | 76.9 | 25.5 | 71.5 | **88.5** | 34.1 |
| RSL-SQL | 74.8 | 84.3 | 37.6 | 78.1 | 77.5 | 27.7 | 80.7 | 88.3 | 42.2 |
| Agent (ours) | 80.2 | **87.3** | **48.1** | 77.1 | **79.4** | **29.0** | 86.7 | 85.8 | **51.5** |

Table 3: Comparison of Precision, Recall and EM across different methods in single-database scenario.

- 제안한 LinkAlign을 multi-database, single-database setting 모두에서 진행
- 제안한 LinkAlign은 baselines 대비 우수한 LA. recall, EM score를 보임
- 이는 LinkAlign이 실제 SQL 구문 생성에 필요한 DB에 대한 누락을 덜 하면서 (LA) 적절한 DB를 찾았을 경우 더 정확한 Schema linking을 수행하는 것을 의미 (Recall, EM)

# LinkAlign – Result

- **Main result: SQL query generation performance**

| Approach | EX (%) |
|---|---|
| DIN-SQL + GPT-4 | 82.8 |
| MAC-SQL + GPT-4 | 86.8 |
| DAIL-SQL + GPT-4 | 86.6 |
| MCS-SQL + GPT-4 | 89.5 |
| LinkAlign* + GPT-4 | 91.2 |
| LinkAlign* + DeepSeek-V3(671B) | 88.9 |
| LinkAlign* + Qwen(72B) | 86.8 |

Table 4: Comparison of different methods on Spider-dev dataset. * indicates method using a simplified LinkAlign framework without Step One and Step Two.

| Approach | EX (%) |
|---|---|
| DIN-SQL + GPT-4 | 50.7 |
| MAC-SQL + GPT-4 | 59.4 |
| DAIL-SQL + GPT-4 | 54.8 |
| MCS-SQL + GPT-4 | 63.4 |
| RSL-SQL + GPT-4 | 67.2 |
| LinkAlign* + GPT-4 | 61.6 |
| LinkAlign* + DeepSeek-V3(671B) | 57.5 |
| LinkAlign* + Qwen(72B) | 53.4 |

Table 5: Comparison of different methods on Bird-dev dataset. * indicates method using a simplified LinkAlign framework without Step One and Step Two.

- 제안한 LinkAlign의 schema linking 결과를 사용해서 DIN-SQL 방법론에 적용하여 SQL 구문 생성능력을 측정
- SPIDER 에서는 BASELINE 대비 SOTA 성능을 달성, BIRD 에서는 중간 정도의 성능 달성

# LinkAlign – Result

- **Ablation study: Validating Query rewriting & Response filtering strategy**

| Model variant | Spider | | | AmbiDB | | |
|---|---|---|---|---|---|---|
| | LA | EM | Recall | LA | EM | Recall |
| **Pipeline** | 85.4 | 37.5 | 66.1 | 69.4 | 20.3 | 50.4 |
| **w/o que. rew.** | 85.3 | 37.7 | 72.3 | 63.1 | 14.5 | 52.8 |
| **w/o res. fil.** | 81.9 | 26.0 | 62.0 | 66.2 | 15.3 | 48.5 |
| **w/o both** | 80.0 | 26.8 | 62.4 | 59.5 | 11.4 | 38.7 |
| **Agent** | 86.4 | 47.7 | 80.7 | 67.6 | 22.4 | 56.9 |
| **w/o que. rew.** | 83.6 | 30.6 | 73.0 | 65.3 | 15.1 | 57.0 |
| **w/o res. fil.** | 66.7 | 27.8 | 54.8 | 58.5 | 14.5 | 60.6 |
| **w/o both** | 73.6 | 32.9 | 61.1 | 58.0 | 17.6 | 47.8 |

Table 7: Performance comparison of model variants on Spider and AmbiDB datasets. "que. rew." indicates query rewriting and "res. fil." denotes the response filtering.

- Retrieval 단계에서 Query Rewriting과 Schema linking 단계에서 Response filtering 방법을 제거했을 때의 schema linking 성능 측정
- Query rewriting, Response filtering 제거할 때 모두 schema linking 성능이 하락함
- 또한 Response filtering을 제거했을 때 Query rewriting 대비 성능 하락이 두드러짐
- 이는 Step two의 실패가 정답 SQL 구문 생성을 위한 참조 DB 실패로 이어져서 성능 하락이 두드러지는 것으로 추정
- 반대로 초기 관련 DB 집합을 구성하는 step one의 실패를 step two에서 방지해준다고도 볼 수 있음

# Reasoning Text2SQL

## SQL-R1: Training Natural Language to SQL Reasoning Model By Reinforcement Learning

**Peixian Ma**[1,2,], **Xialie Zhuang**[1,3], **Chengjin Xu**[1,4,*], **Xuhui Jiang**[1,4], **Ran Chen**[1], **Jian Guo**[1]
[1]IDEA Research, International Digital Economy Academy
[2]The Hong Kong University of Science and Technology (Guangzhou)
[3]University of Chinese Academy of Sciences
[4]DataArc Tech Ltd.
pma929@connect.hkust-gz.edu.cn, xuchengjin@idea.edu.cn

**NeurIPS 2025 poster**

**score:5 / 4 / 2 / 4**

# SQL-R1

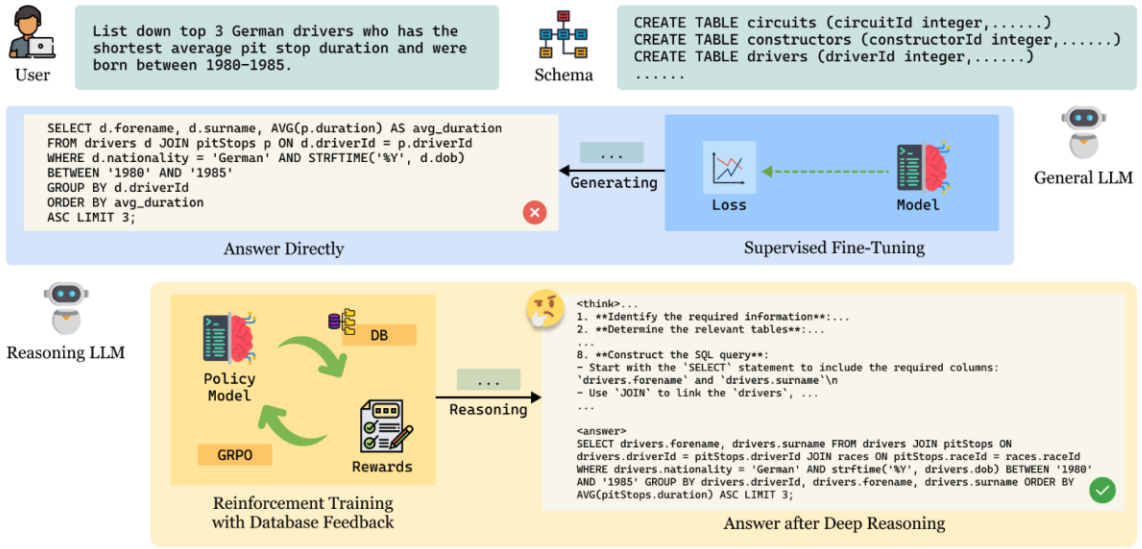



Figure 1: Demonstration of our work. Previous work on NL2SQL primarily relies on supervised fine-tuning to enable the model to learn how to generate SQL. However, in the case of complex database schema or ambiguous semantics, the fine-tuned model may struggle to produce SQL that does not align with the user's intentions, as it depends on a fixed generation strategy and previous data. By introducing reinforcement learning algorithms, the model can receive intuitive feedback from the database during the training process. This feedback encourages the model to independently explore various SQL generation reasoning approaches, ultimately enhancing the accuracy of its output.

- 기존의 Text2SQL 모델은 SFT 방식을 주로 활용하여 SQL 생성 능력을 향상시키고자 함
- 그러나 SFT 방식으로 학습할 경우 특정 domain에 치중되거나 complex question에 대응되는 complex query를 생성하기 어려움
- 또한 SFT로 학습된 기존 모델은 단순 SQL query만 생성하므로 SQL query문을 생성하기 까지의 추론 과정을 이해하기 어려움 (Interpretability)

← 일반화 능력이 높으면서 추론의 과정을 설명할 수 있는 Reasoning 기반 Text2SQL 모델 필요

→ 강화학습 (RL) 방식으로 학습된 Text2SQL 특화 모델 SQL-R1을 제안

# SQL-R1 – Method

- **Step 1: SFT (cold start)**

**Table 1: Overall statistics of different datasets. Note: † denotes that each database in WikiSQL consists of a single table. * indicates that the number of unique SQL queries for BIRD cannot be calculated due to the inaccessibility of its test set.**

| Dataset | Source | # Example | # Unique SQL | # DB | Lang. Styles | Knowledge | CoT Solution |
|---|---|---|---|---|---|---|---|
| WIKISQL [92] | Human+Template | 80,654 | 80,257 | 26,531† | ✘ | ✘ | ✘ |
| Spider [87] | Human | 10,181 | 4,489 | 200 | ✘ | ✘ | ✘ |
| BIRD [45] | Human | 12,751 | -* | 95 | ✘ | ✓ | ✘ |
| ScienceBenchmark [88] | LLM-Gen+Human+Template | 5,031 | 3,652 | 3 | ✘ | ✘ | ✘ |
| EHRSQL [38] | Human+Template | 20,108 | 18,253 | 2 | ✘ | ✘ | ✘ |
| SynSQL-2.5M | LLM-Gen | 2,544,390 | 2,412,915 | 16,583 | ✓ | ✓ | ✓ |

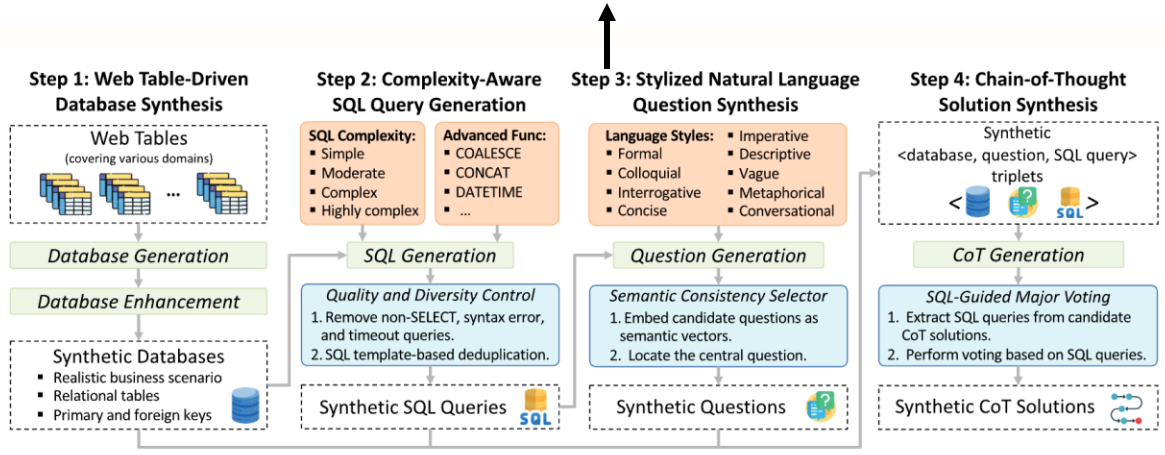



**Figure 1: Illustration of the proposed text-to-SQL data synthesis framework.**

- 강화학습 기반의 Reasoning Text2SQL 모델 학습을 할 때 어떤 Base 모델로 RL을 시작할지에 관한 Cold-start 문제를 확인하기 위한 실험을 진행

- 기본 Base 모델은 250만여 개의 합성 데이터셋으로 구성된 SynSQL-2.5M을 바탕으로 학습된 Qwen2.5-Coder 기반 OmniSQL 모델을 사용

- Cold-start 문제를 검증하기 위한 추가 SFT 데이터셋으로 SynSQL-2.5M 태깅된 4 개의 난이도별 5만 건씩 샘플링한 **SynSQL-200K**를
  SFT 학습 데이터셋을 구성

$$v = (x, t, y^*)$$

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. 2025. OmniSQL: Synthesizing High-Quality Text-to-SQL Data at Scale. Proc. VLDB Endow. 18, 11 (July 2025), 4695–4709. https://doi.org/10.14778/3749646.3749723

# SQL-R1 – Method

- **Step 2: RL**

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\mathbf{v}\sim P(\mathbf{V}),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|\mathbf{v})} \left[ \frac{1}{G}\sum_{i=1}^{G} \left( \min\left(r_i^{\text{ratio}} A_i,\ \text{clip}\left(r_i^{\text{ratio}}, 1-\epsilon, 1+\epsilon\right) A_i\right) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \right) \right], \quad (1)$$

$$\hat{A}_{i,t} = \sum_{index(j)\geq t} \tilde{r}_i^{index(j)} \qquad \tilde{r}_i^{index(j)} = \frac{r_i^{index(j)} - \text{mean}(\mathbf{R})}{\text{std}(\mathbf{R})}.$$

- 학습된 SFT 모델을 활용하여 Text2SQL에서 Reasoning을 강화하기 위한 강화학습을 수행

- SQL-R1 강화학습 알고리즘으로 GRPO를 사용, GRPO에서 Advantage를 계산하기 위한 verifiable reward function을 Text2SQL 상황에 맞게 4가지 Reward를 제안

- RL 학습 데이터셋은 SynSQL-2.5M에서 Complex에 해당하는 난이도 데이터를 5k개 sampling한 **SynSQL-Complex-5K** 데이터를 학습 데이터로 사용

$$v = (x, y^*)$$

# SQL-R1 – Method

- **Step 2: RL**

Format Reward

$$S_f = \begin{cases} 1, & \text{if format is correct} \\ -1, & \text{if format is incorrect} \end{cases}$$

- 강화학습 과정에서 학습하는 모델의 생성물이 Reasoning을 담은 부분은 <think>...</think> 안에 담기고, 최종 SQL 구문 생성은 <answer> ... </answer> 부분에 담기도록 Format Reward를 부여
- 생성된 SQL 구문의 경우 ```sql...``` 형식을 준수해야함

Execution Reward

$$S_e = \begin{cases} 2, & \text{if SQL candidate is executable} \\ 0, & \text{if format is incorrect} \\ -2, & \text{if SQL candidate is not executable} \end{cases}$$

- 생성된 SQL 구문이 실행 가능한 구문인지에 관한 Reward
- 실행 불가능한 SQL 구문일 경우 다른 reward에서 모두 penalty를 받도록 함
- 또한 생성된 SQL 구문의 실행시간이 너무 길 경우에 한해서도 penalty를 받도록 함

# SQL-R1 – Method

- **Step 2: RL**

Result Reward

$$S_r = \begin{cases} 3, & \text{if query result is correct} \\ 0, & \text{if format is incorrect or SQL candidate is not executable} \\ -3, & \text{if query result is incorrect} \end{cases}$$

- 생성한 SQL 구문이 정답 SQL 구문과 동일한 결과를 반환할 경우, 즉 EX가 1일 경우에 높은 Reward를 부여

- 다른 Reward대비 가장 scale이 크므로 저자들은 모델이 정확한 SQL 구문을 생성하도록 Reward를 설계 했다는 것을 알 수 있음

Length Reward

$$S_l = \begin{cases} 0.5 \times S_{tl} + S_{al}, & \text{if query result is correct and } len_{response} <= \texttt{MAX LENGTH} \\ 0.5 + S_{al}, & \text{if query result is correct and } len_{response} > \texttt{MAX LENGTH} \\ 0, & \text{other cases} \end{cases}$$

where $S_{tl} = (len_{think} + len_{answer}) \,/\, \texttt{MAX LENGTH}$ and $S_{al} = len_{sql} \,/\, len_{answer}$.

- 생성된 SQL 구문이 정답에 해당할 때 모델의 정답 생성 과정을 최적화 하기 위하여 생성 길이 관점의 reward를 부여

- 위 reward에서 이상적인 상황은 think length, answer length의 합이 MAX LENGTH와 비슷하면서 answer length내에 SQL length 비율이 높은 경우가 가장 높은 reward를 받을 수 있음

- 만약 thinking path가 너무 길어질 경우 penalty를 부여

# SQL-R1 – Experimental Setups

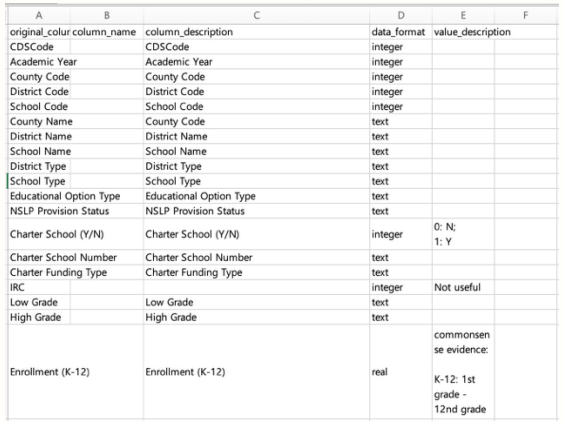
- **Experimental dataset**

Spider

| Segment | Size |
|---|---|
| Train | 8,659 |
| Validation | 1,034 |

BIRD

| Segment | Size |
|---|---|
| Train | 9,078 |
| Validation | 1,534 |

| original_colur | column_name | column_description | data_format | value_description |
|---|---|---|---|---|
| CDSCode | | CDSCode | integer | |
| Academic Year | | Academic Year | integer | |
| County Code | | County Code | integer | |
| District Code | | District Code | integer | |
| School Code | | School Code | integer | |
| County Name | | County Code | text | |
| District Name | | District Name | text | |
| School Name | | School Name | text | |
| District Type | | District Type | text | |
| School Type | | School Type | text | |
| Educational Option Type | | Educational Option Type | text | |
| NSLP Provision Status | | NSLP Provision Status | text | |
| Charter School (Y/N) | | Charter School (Y/N) | integer | 0: N;<br>1: Y |
| Charter School Number | | Charter School Number | text | |
| Charter Funding Type | | Charter Funding Type | text | |
| IRC | | | integer | Not useful |
| Low Grade | | Low Grade | text | |
| High Grade | | High Grade | text | |
| Enrollment (K-12) | | Enrollment (K-12) | real | commonsense evidence:<br><br>K-12: 1st grade - 12nd grade |

Metadata

{'question_id': 0,
'db_id': 'california_schools',
'question': 'What is the highest eligible free rate for K-12 students in the schools in Alameda County?',
'evidence': 'Eligible free rate for K-12 = `Free Meal Count (K-12)` / `Enrollment (K-12)`',
'SQL': "SELECT `Free Meal Count (K-12)` / `Enrollment (K-12)` FROM frpm WHERE `County Name` = 'Alameda' ORDER BY (CAST(`Free Meal Count (K-12)` AS REAL) / `Enrollment (K-12)`) DESC LIMIT 1",
'difficulty': 'simple'}

# SQL-R1 – Experimental Setups

- **Metric**

**Execution Accuracy (EX)**

$$\mathrm{EX} = \frac{\sum_{i=1}^{N} \mathbb{1}(\hat{V}_i, V_i)}{N}$$

$$\mathbb{1}(\hat{V}_i, V_i) = \begin{cases} 1 & \text{if} \quad \hat{V}_i = V_i \\ 0 & \text{if} \quad \hat{V}_i \neq V_i \end{cases}$$

# SQL-R1 – Experimental Setups

- **Baselines**

SQL-o1

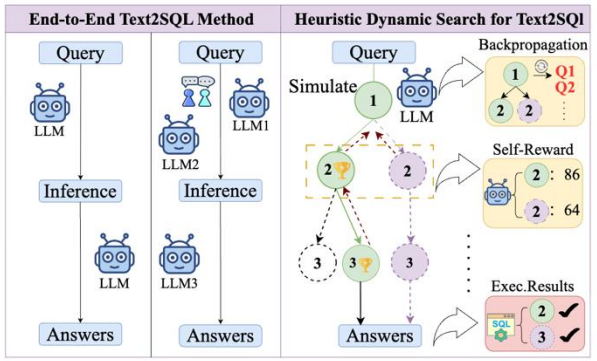



Figure 1: The illustrations of the differences among end-to-end Text2SQL method and Heuristic Dynamic Search with Self-Reward.

- Text2SQL reasoning을 Monte Carlo Tree Search (Selection, Expansion, Simulation, Backpropagation) 방법을 통해 접근

- 모델을 SFT 방식으로 학습한 뒤 학습된 모델의 reasoning path 생성 과정에 MCTS 개념을 구현한 reasoning 방법론 제안

Reasoning-SQL

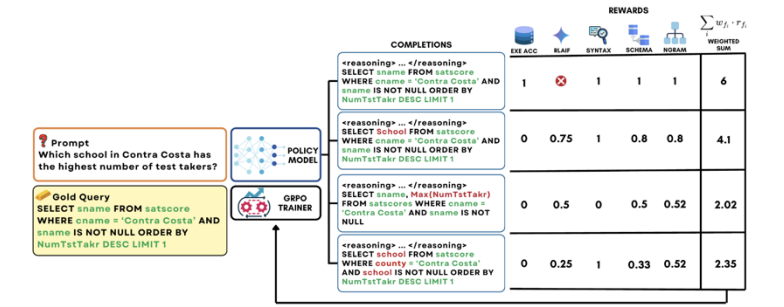



Figure 1: Overview of the GRPO-based Text-to-SQL training pipeline. For each natural language prompt q and its associated database schema, the policy model $\pi_\theta$ generates a group of candidate SQL queries. Each candidate is evaluated using a suite of reward functions to produce a composite reward. These rewards are then used to compute advantages and update the policy via GRPO.

- SQL-R1과 비슷하게 GRPO와 Text2SQL을 위한 Reward를 설계하여 RL로 학습된 Reasoning-RL 제안

- Execution accuracy reward, LLM-as-a-judge reward, Syntax check reward, Schema linking reward, N-gram similarity reward, Format reward를 사용

# SQL-R1 – Result

- **Main result**

Table 1: Execution accuracy (%) of different NL2SQL methods on Spider and BIRD benchmark.

| NL2SQL Method | Base Model | Candidate Selection | Spider (Dev) | Spider (Test) | BIRD (Dev) |
|---|---|---|---|---|---|
| CodeS [14] | CodeS-15B | - | 84.9 | 79.4 | 57.0 |
| DTS-SQL [13] | Deepseek-Coder-7B | - | 85.5 | 84.4 | 55.8 |
| CHESS [7] | Deepseek-Coder-33B | - | - | 87.2 | 61.5 |
| Alpha-SQL [29] | Qwen2.5-Coder-7B | Self-Consistency | 84.0 | - | 66.8 |
| SQL-o1 [30] | Qwen2.5-Coder-7B | Self-Consistency | 84.7 | 85.1 | 66.7 |
| OmniSQL [22] | Qwen2.5-Coder-7B | Self-Consistency | 85.5 | 88.9 | 66.1 |
| DeepRetrieval [31] | Qwen2.5-Coder-7B | - | - | 76.1 | 56.0 |
| Reasoning-SQL [32] | Qwen2.5-Coder-14B | Self-Consistency | 81.4 | - | 65.3 |
| C3-SQL [10] | GPT-3.5-Turbo | Self-Consistency | 82.0 | 82.3 | - |
| DIN-SQL [8] | GPT-4 | - | 82.8 | 85.3 | - |
| DAIL-SQL [16] | GPT-4 | Self-Consistency | 83.6 | 86.2 | 54.8 |
| MAC-SQL [9] | GPT-4 | Self-Consistency | 86.8 | 82.8 | 59.4 |
| SuperSQL [33] | GPT-4 | Self-Consistency | 84.0 | 87.0 | 58.5 |
| MCTS-SQL [34] | GPT-4o | - | 88.7 | 86.6 | 69.4 |
| OpenSearch-SQL [35] | GPT-4o | Self-Consistency | - | 87.1 | 69.3 |
| CHASE-SQL [36] | Gemimi-1.5-Pro | - | - | 87.6 | 73.0 |
| **SQL-R1 (Ours)** | **Qwen2.5-Coder-3B** | **Self-Consistency** | **78.1** | **78.9** | **54.6** |
| **SQL-R1 (Ours)** | **Qwen2.5-Coder-7B** | **Self-Consistency** | **87.6** | **88.7** | **66.6** |
| **SQL-R1 (Ours)** | **Qwen2.5-Coder-14B** | **Self-Consistency** | **86.7** | **88.1** | **67.1** |

- 제안 SLQ-R1은 Baseline 대비 우수한 SQL 생성 능력을 보임
- 동일 Base Model, 동일 parameter를 쓰는 경우에 SQL-R1이 가장 우수한 성능을 달성
- GPT-4, Gemini 1.5-Pro와 같은 Proprietary model을 사용하는 방법론과 비교할 때도 우수한 성능을 보이며 3B 모델의 경우 BIRD에서 comparable한 성능을 달성함

# SQL-R1 – Result

- **Main result by complexity level**

Table 3: Execution accuracy (%) of different complexity levels on BIRD-Dev dataset.

| NL2SQL Method | Base Model | Simple | Moderate | Challenging | All |
|---|---|---|---|---|---|
| CodeS [14] | Codes-15B | 65.8 | 48.8 | 42.4 | 58.5 |
| DAIL-SQL [16] | GPT-4 | 63.0 | 45.6 | 43.1 | 55.9 |
| SuperSQL [33] | GPT-4 | 66.9 | 46.5 | 43.8 | 58.5 |
| **SQL-R1 (Ours)** | **Qwen2.5-Coder-7B** | **72.1** | **60.8** | **51.0** | **66.6** |
| **SQL-R1 (Ours)** | **Qwen2.5-Coder-14B** | **72.4** | **59.7** | **56.5** | **67.1** |

- Text2SQL 난이도에 따른 실험에서 SQL-R1은 Baselines 대비 모든 난이도에서 우수한 성능을 달성

- 특히 다른 Baselines 대비 난이도 상승에 따른 성능 감소폭이 적은 것을 볼 때 난이도 변화에도 robust한 성능을 보이는 것을 확인

# SQL-R1 – Result

- **Main result by cold start strategy**

Table 4: Execution accuracy (%) of models with different cold start strategy. The *Reasoning Instruction* column is applied represents the instruction applied SFT process.

| Model | SFT Data | Reasoning Instruction | Spider (Dev) | Spider (Test) | BIRD (Dev) |
|---|---|---|---|---|---|
| Qwen2.5-Coder-7B | - | ✗ | 77.4 | 79.4 | 58.2 |
| Qwen2.5-Coder-7B | SynSQL-200K | ✓ | 82.7 | 83.3 | 57.0 |
| Qwen2.5-Coder-14B | - | ✗ | 87.0 | 88.0 | 66.1 |
| OmniSQL-7B [22] | SynSQL-2.5M | ✗ | 85.5 | 88.9 | 66.1 |
| OmniSQL-14B [22] | SynSQL-2.5M | ✗ | 86.2 | 88.3 | 65.9 |
| SQL-R1 + Qwen2.5-Coder-7B | - | ✗ | 84.5 | 86.1 | **63.1** |
| SQL-R1 + Qwen2.5-Coder-7B | SynSQL-200K | ✓ | 84.7 | 86.4 | 59.2 |
| SQL-R1 + Qwen2.5-Coder-14B | - | ✗ | 86.7 | 88.1 | **67.1** |
| SQL-R1 + OmniSQL-7B | SynSQL-2.5M | ✗ | 87.6 | 88.7 | **66.6** |
| SQL-R1 + OmniSQL-14B | SynSQL-2.5M | ✗ | 86.4 | 87.6 | **66.6** |

- Text2SQL task에서 RL 기반 reasoning model을 학습할 때 Base모델 선정 및 데이터셋 선정에 따른 SFT 모델이 RL 성능에 미치는 영향을 알아보고자

  추가 분석을 실시

  - 데이터셋의 양, 데이터셋의 구성 방식 등에 따른 최종 Reasoning 성능 변화 측정
- 분석 결과, 데이터셋의 양이 많을수록 Reasoning 성능이 높아졌으며, 이는 7B 모델을 SynSQL-200K 학습한 성능과 OminSQL-7B 모델의 성능 비교결과를 통해 할 수 있음
- 한편, 데이터가 적은 상태에서 Thinking path가 추가된 Reasoning Instruction이 가미된 형태의 SFT는 성능 향상이 미미했음

# SQL-R1 – Result

- **Ablation study: Validating reward design**

Table 5: Ablation study of reward components on BIRD-Dev dataset.

| Reward Function | Accuracy (%) |
|---|---|
| Qwen2.5-Coder-7B | 58.2 |
| $S_f + S_e + S_r + S_l$ | **63.1** |
| - w/o $S_f$ (Format Score) | 60.4 ($\downarrow$ 2.7) |
| - w/o $S_e$ (Execution Score) | 60.7 ($\downarrow$ 2.4) |
| - w/o $S_r$ (Result Score) | 62.4 ($\downarrow$ 0.7) |
| - w/o $S_l$ (Length Score) | 61.0 ($\downarrow$ 2.1) |

- Reward design에 관한 타당성 검증을 위해 Ablation Study를 진행

- 모든 Reward를 제거함에 따른 성능 저하가 나타나는 것으로 볼 때, 각 reward는 SQL-R1의 reasoning 능력 향상에 기여를 하는 것을 알 수 있음

- 한편, Execution Reward, Format Reward를 제거했을 때 성능 하락이 두드러지는 것을 볼 때 SQL구문의 Syntax를 정확히 따르도록 학습을 유도하는 방법이 실제 정확한 SQL 구문 생성에 유효함을 알 수 있음

# Thank you

atural Language
rocessing
Artificial Intelligence


고려대학교
KOREA UNIVERSITY