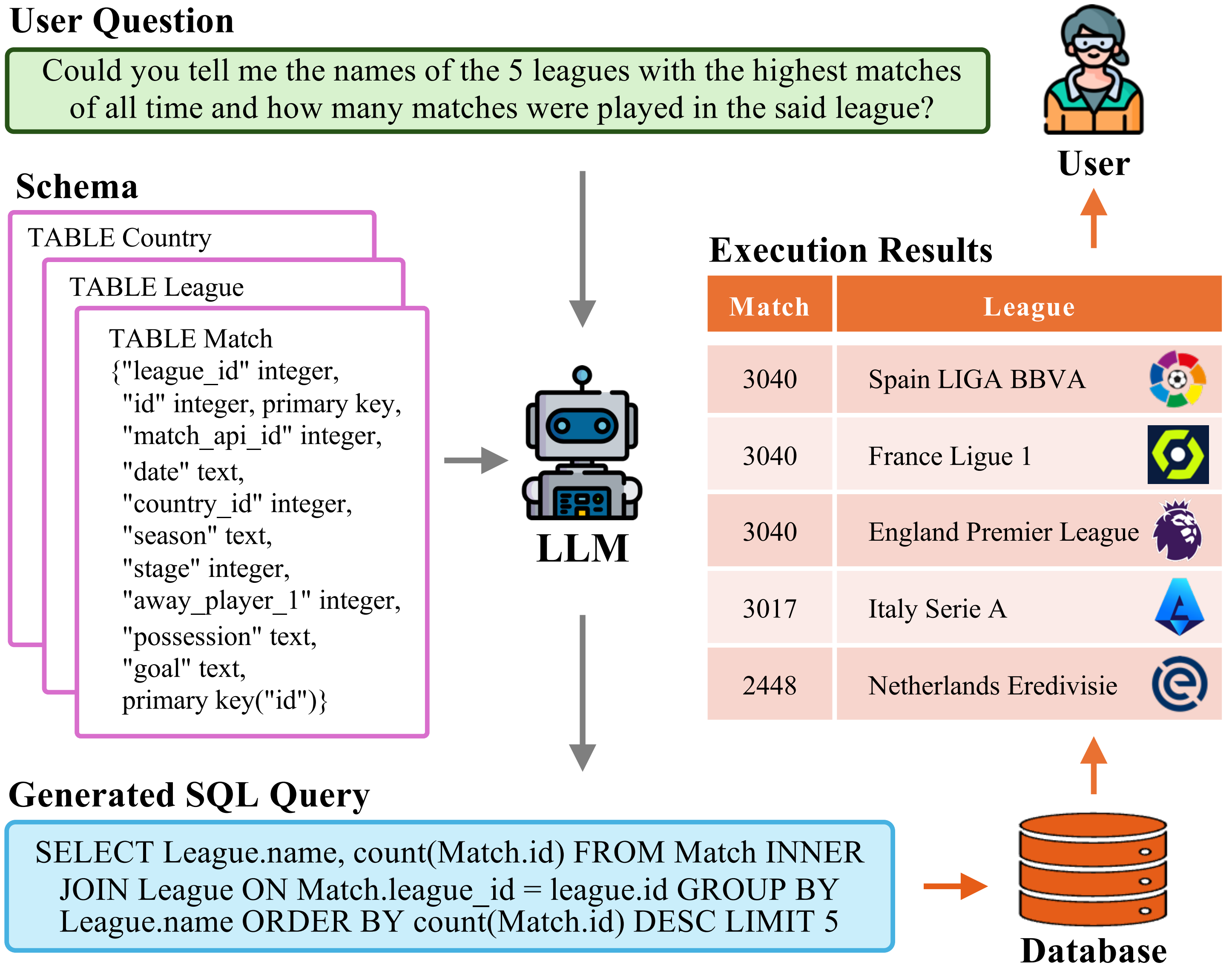
What is Text-to-SQL

- (2023) Proficiency in SQL remains essential for 51.52% of professional developers, who depend on it for database interactions.

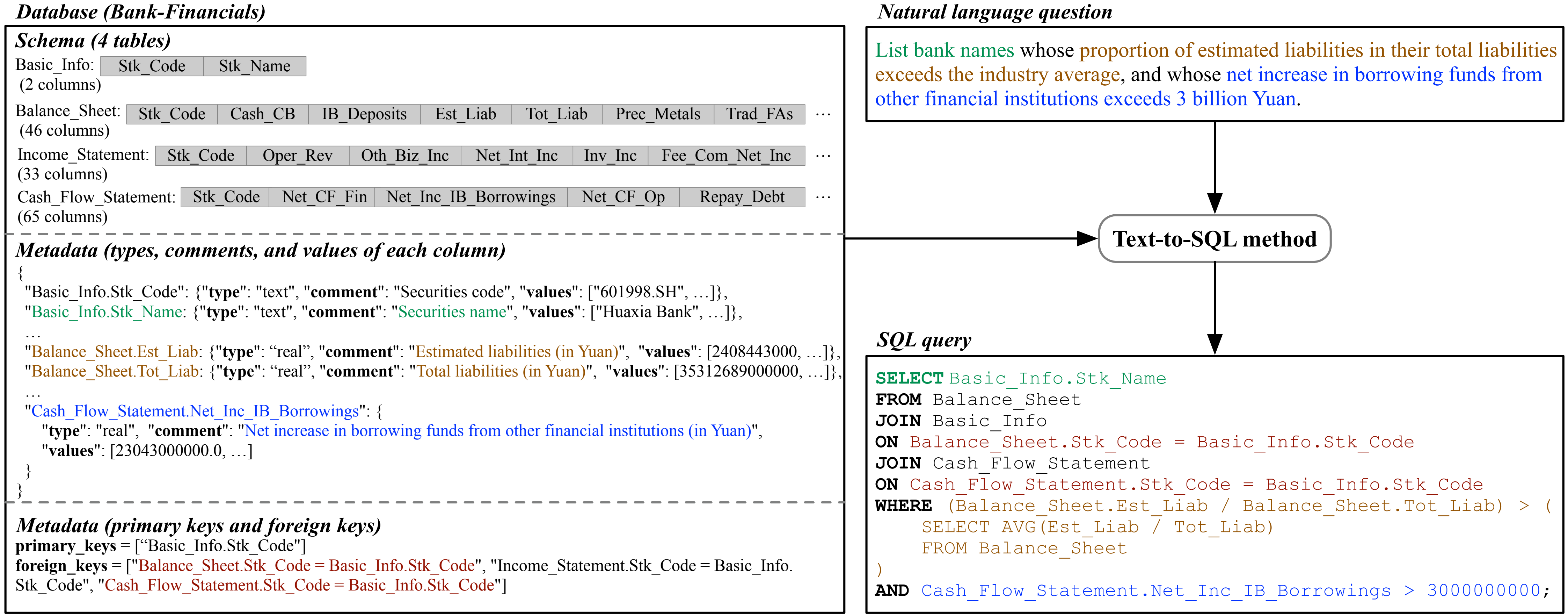
Text-to-SQL:

- automate the transformation of natural language questions into SQL.
- understand the intent of question & structure of database.
- generate SQL corresponding to the question. The execution result can answer the question.



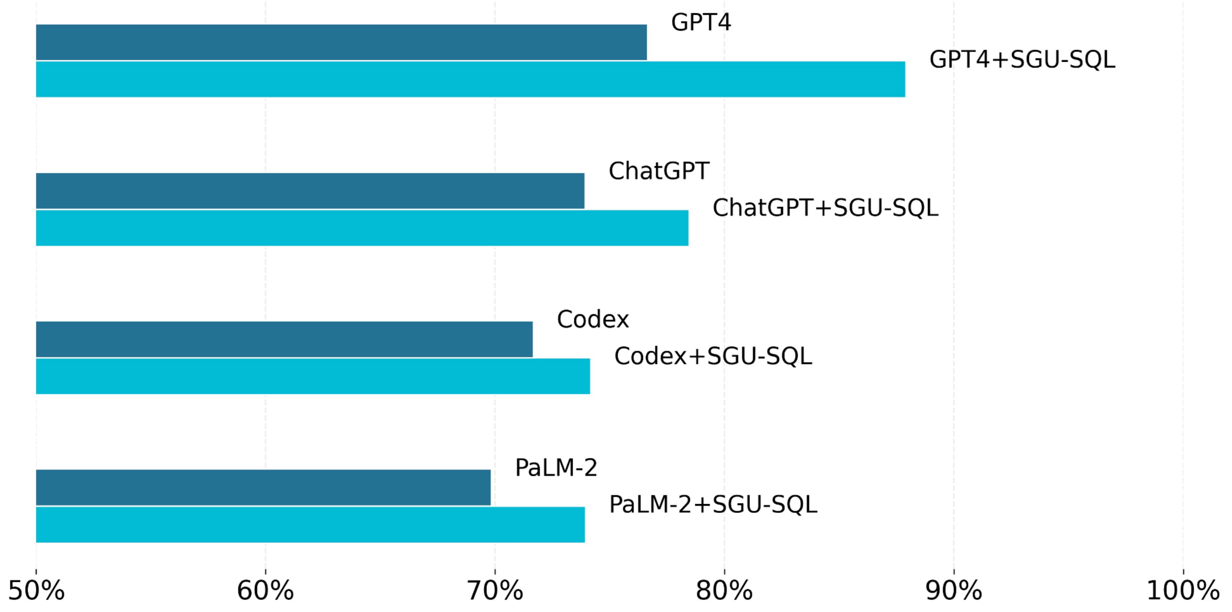
Applications of Text-to-SQL

- Operation & maintenance
- Real-time information querying for public users
  - "When's the next 103 bus from Hung Hom?"
  - "Taxi fare to Central from here?"
- Business intelligence & analytics
- Internal company tools



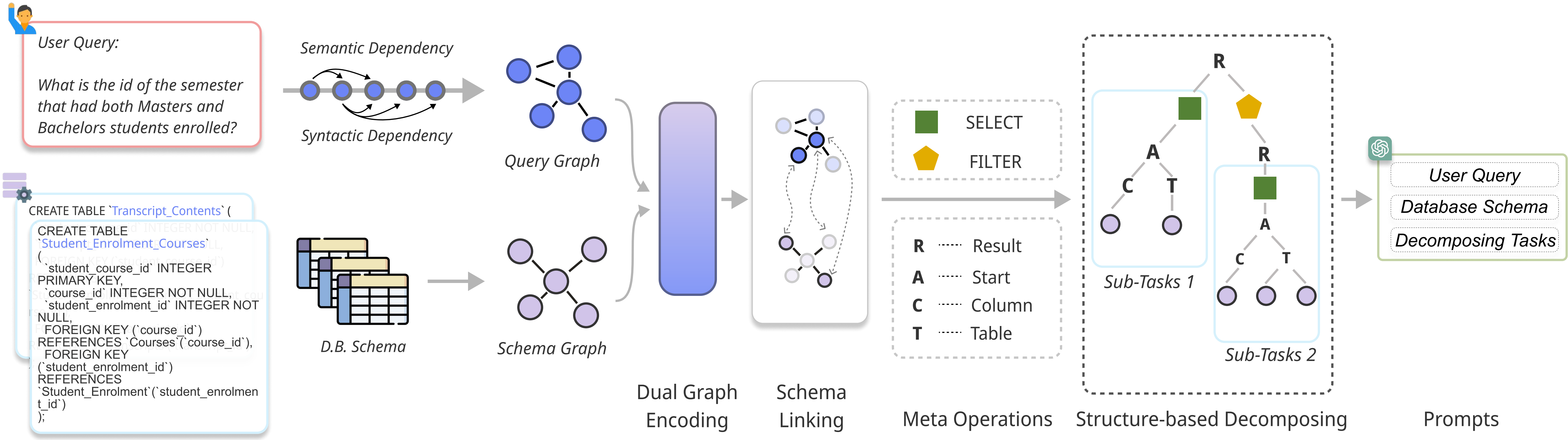
Challenges of LLM-based Text-to-SQL

- 10,181 questions on 200 databases with multiple tables covering 138 different domains.
- 8,659 instances in the training split and 1,034 instances in the development split.
- Performance when directly use LLMs:



- LLMs often struggle to fully comprehend user intention:
  - users are lazy and may not be familiar with the database schema.
- Sophisticated database architecture:
  - complex schemas with interrelated tables; non-intuitive naming conventions.
  - large or poorly documented databases: similar column names across different tables cause confusion.
- Complex Syntax Structure of SQL:
  - intricate connections between query concepts and database elements.
  - SQL requires precise clause arrangement, correct operator usage, & adherence to grammatical rules.
  - complex queries: nested subqueries, aggregate functions, window operations demands high precision.
    - often beyond the capabilities of current LLMs

Structure-GUided text-to-SQL framework (SGU-SQL)



- Establishes structure-aware links between user queries and database schema.
  - graph-based structure construction for both user query and database understanding;
  - tailored structure linking method: map the query to the relevant database elements.
- Recursively decomposes the complex generation task using syntax-based prompting to guide LLMs in constructing target SQLs.
  - adhering to the syntax structure

Step 1: Revisit Query and Database via Graph

Context-free grammar:

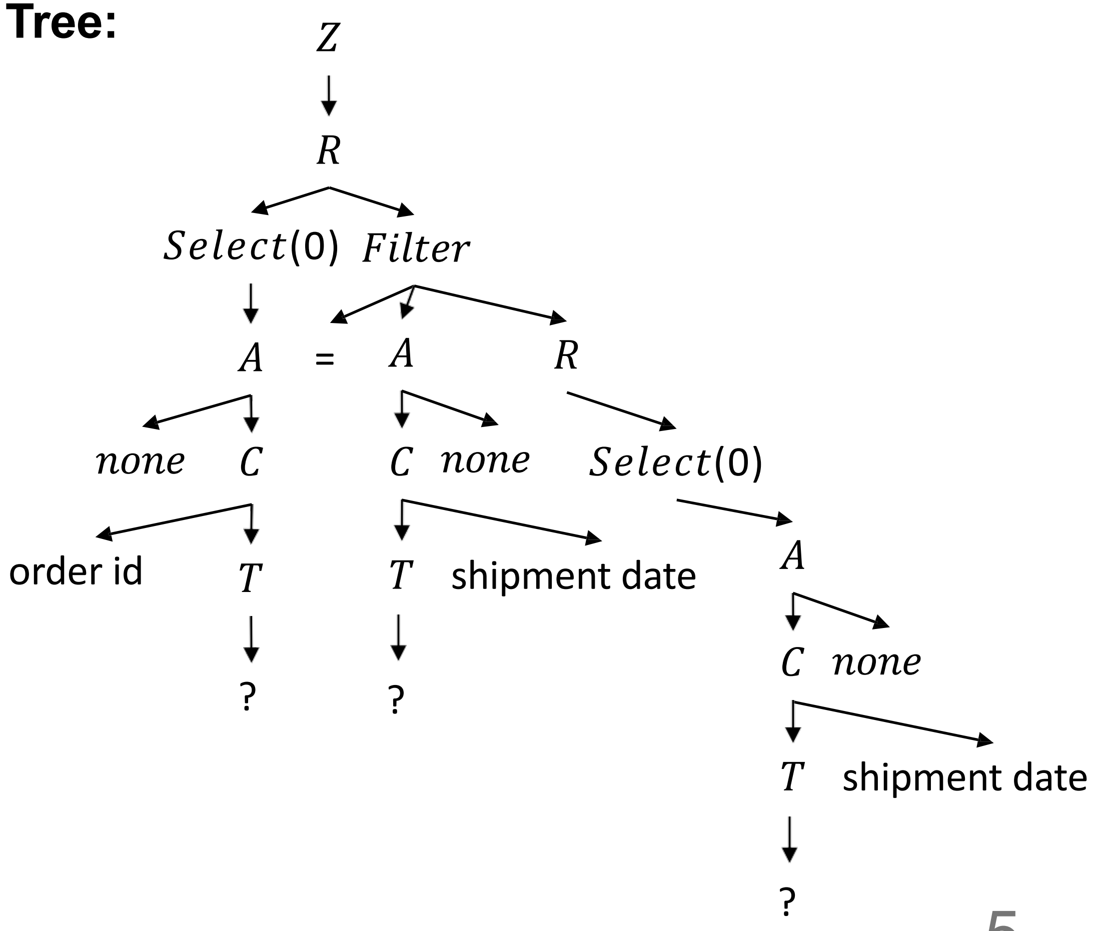
$$Z ::= intersect\ R\ R\ union\ R\ R\ except\ R\ R \mid R$$
$$R ::= Select(n)$$
$$\mid Select(n)\ Filter \mid Select\ (n)\ Order(d)$$
$$\mid Select(n)\ Sup(p) \mid Select\ (n)\ Order(d)\ Filter$$
$$\mid Select(n)\ Sup(p)\ Filter$$
$$Select(n) ::= \mathrm{A_0 A_1 \ldots A_{n-1}}$$
$$Order(d) ::= A$$
$$Sup(p) ::= A$$
$$A ::= \max C \mid \min C \mid count\ C \mid sum\ C \mid avg\ C \mid none\ C$$
$$C ::= c\ T$$
$$T ::= t$$
$$Filter ::= and\ Filter\ Filter \mid or\ Filter\ Filter$$
$$\mid > A \mid > A\ R \mid < A \mid < A\ R$$
$$\mid \geq A \mid \geq A\ R \mid = A \mid = A\ R$$
$$\mid \neq A \mid \neq A\ R \mid between\ A$$
$$\mid like\ A \mid not\ like\ A \mid in\ A\ R \mid not\ in\ A\ R$$
$$n \in \{0, 1, 2, 3, \ldots\}$$
$$d \in \{asc, desc\}$$
$$p \in \{most, least\}$$
c *ranges over distinct column names*

t *ranges over table names*

Query parsing:

NL: Which order has the most recent shipment? Give me the order id.

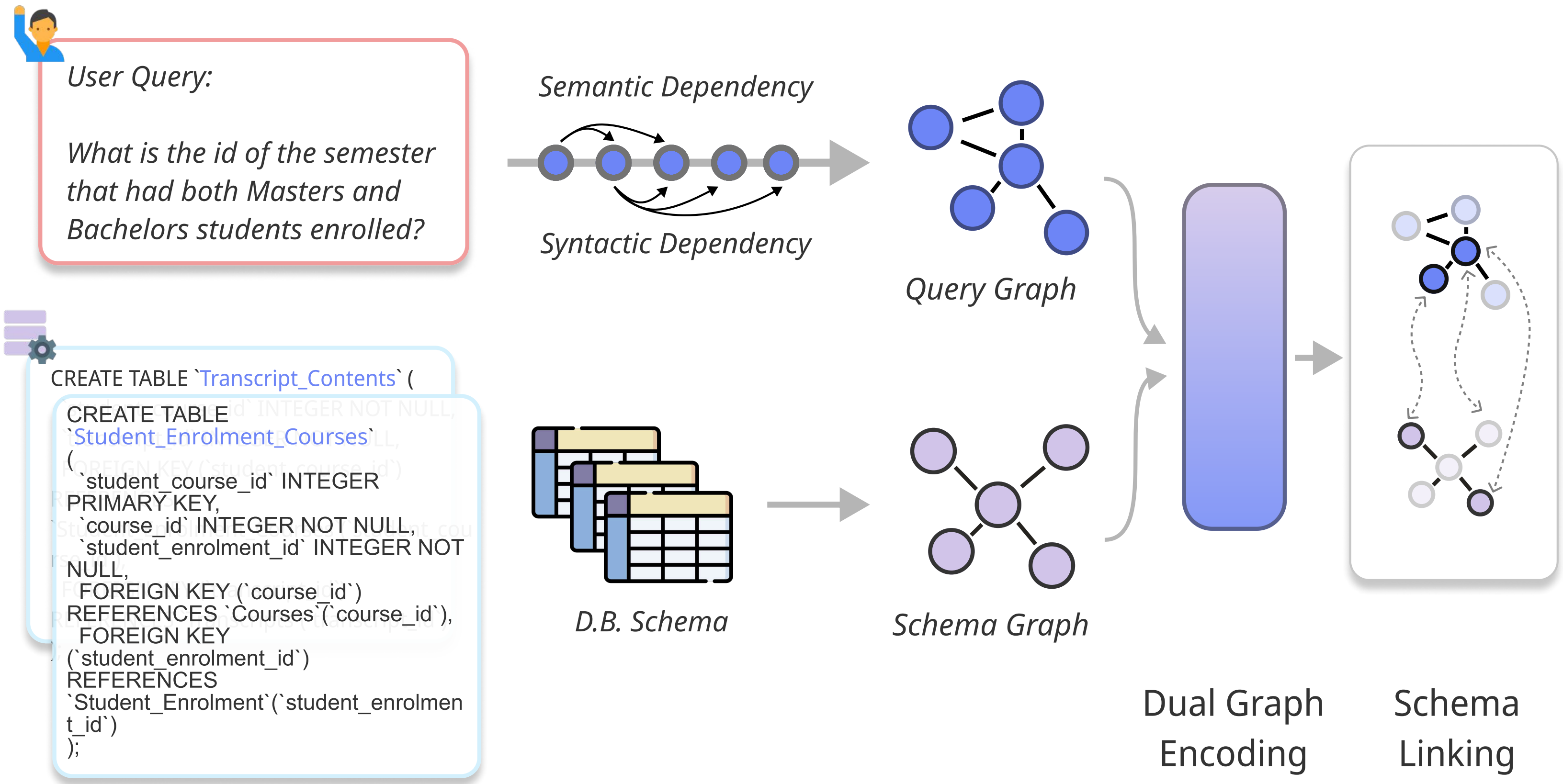
Syntax Tree:



Step 2: Structure Linking with Dual Graph Encoding

Schema Graph:

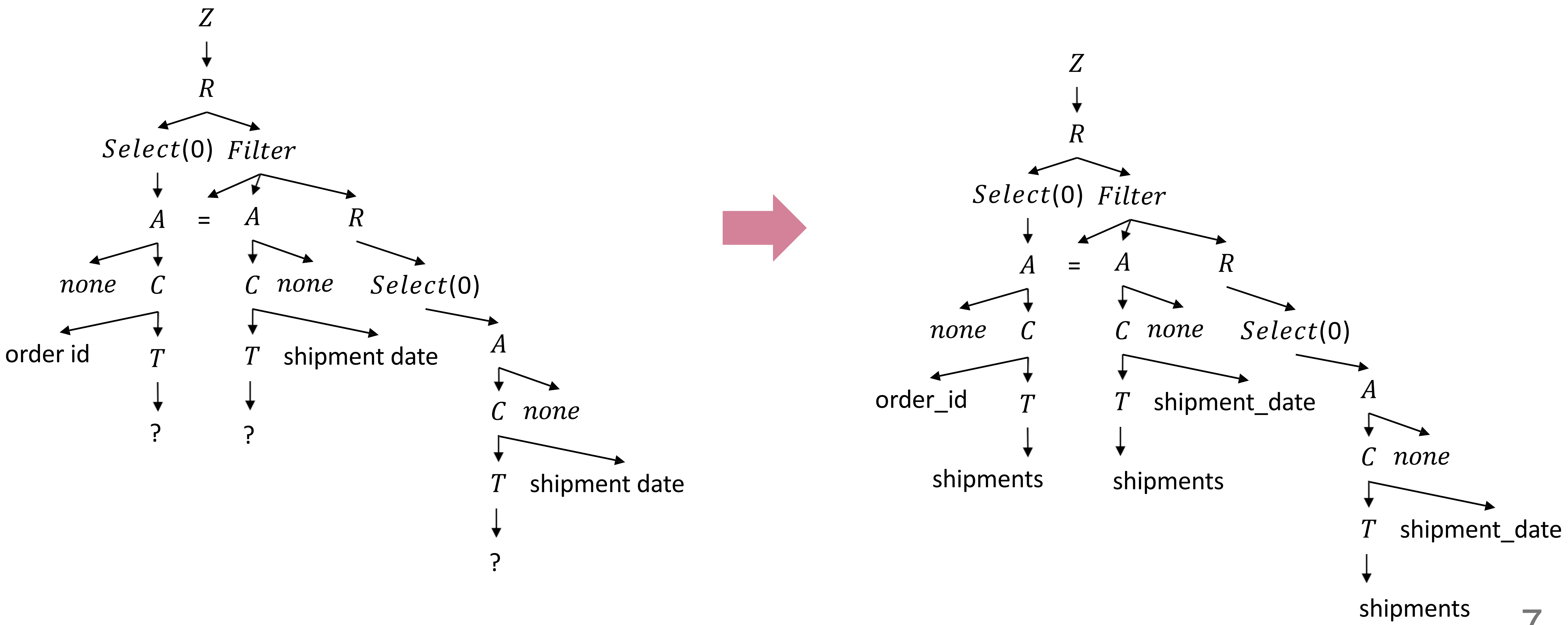
- Nodes: tables & columns
- Table-column edges, primary-key edges, foreign-key edges



- Subgraphs: link nodes in the *Query Graph* and the candidate tables/columns in *Schema Graph.*
- Learn embeddings of Subgraphs based on relational graph attention network.
  - capture the compatibility between natural language concepts and database elements
  - use negative sampling for training

Example of Syntax Tree after Linking

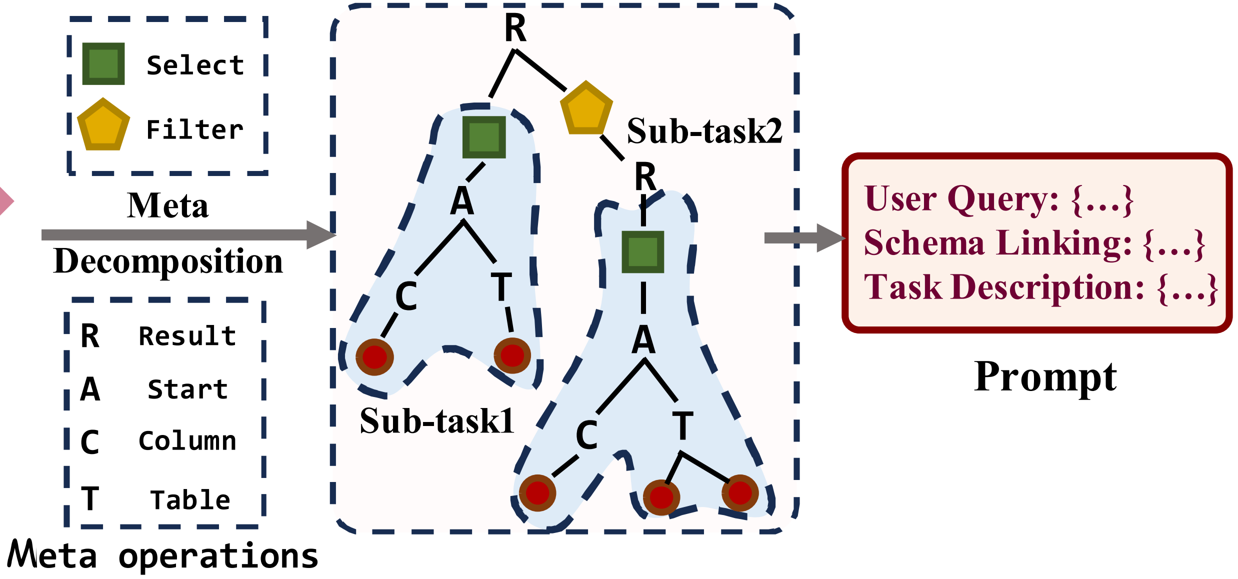
NL: Which order has the most recent shipment?
Give me the order id.



Step 3: Structure-guided Decomposition

NL Question:

Which order has the most recent shipment? Give me the order id.



SQL:

```
SELECT order_id
FROM shipments
WHERE shipment_date = (
    SELECT max(shipment_date)
    FROM shipments
)
```

- Divide the user query into several subtasks.
- Map each non-terminal node to its corresponding SQL component.
- Final SQL: combine the SQL components generated for all non-terminal nodes.

Spider dataset

- 10,181 questions on 200 databases with multiple tables covering 138 different domains.
- 8,659 instances in the training split and 1,034 instances in the development split.

Easy

What is the number of cars with more than 4 cylinders?

```
SELECT COUNT(*)
FROM cars_data
WHERE cylinders > 4
```

Meidum

For each stadium, how many concerts are there?

```
SELECT T2.name, COUNT(*)
FROM concert AS T1 JOIN stadium AS T2
ON T1.stadium_id = T2.stadium_id
GROUP BY T1.stadium_id
```

Hard

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name
FROM countries AS T1 JOIN continents
AS T2 ON T1.continent = T2.cont_id
JOIN car_makers AS T3 ON
T1.country_id = T3.country
WHERE T2.continent = 'Europe'
GROUP BY T1.country_name
HAVING COUNT(*) >= 3
```

Extra Hard

What is the average life expectancy in the countries where English is not the official language?

```
SELECT AVG(life_expectancy)
FROM country
WHERE name NOT IN
    (SELECT T1.name
     FROM country AS T1 JOIN
     country_language AS T2
     ON T1.code = T2.country_code
     WHERE T2.language = "English"
       AND T2.is_official = "T")
```

Execution Accuracy on Spider

Text-to-SQL Method	Backbone LM/LLM	Finetuning	Structure Information	Prompt Strategy	SPIDER Easy	Medium	Hard	Extra	Overall
LlaMA2	LlaMA2-7B	LoRA	✘	✘	0.8868±0.0016	0.6410 ±0.0041	0.4892±0.0030	0.3311±0.0017	0.6259±0.0022
		QLoRA	✘	✘	0.8472±0.0025	0.6234±0.0032	0.4658±0.0021	0.3309±0.0027	0.6083±0.0035
	LlaMA2-13B	LoRA	✘	✘	0.9066±0.0037	0.7292±0.0045	0.5517±0.0029	0.3430±0.0055	0.6809±0.0030
		QLoRA	✘	✘	0.9110±0.0043	0.7004±0.0059	0.5523±0.0032	0.3190±0.0061	0.6648±0.0045
	LlaMA2-70B	SFT	✘	✘	0.4110±0.0093	0.2293±0.0075	0.1906±0.0081	0.0725±0.0090	0.2414±0.0108
		LoRA	✘	✘	0.9151±0.0069	0.7323±0.0080	0.5575±0.0049	0.3921±0.0035	0.6869±0.0040
CodeLlama	CodeLlama-7B	SFT	✘	✘	0.2136±0.0150	0.1769±0.0161	0.0921±0.0169	0.0363±0.0144	0.1487±0.0163
		LoRA	✘	✘	0.9228±0.0105	0.7562±0.0134	0.5863±0.0096	0.3485±0.0126	0.7018±0.0108
		QLoRA	✘	✘	0.9115±0.0127	0.7506±0.0142	0.5982±0.0120	0.3310±0.0085	0.6961±0.0104
	CodeLlama-13B	SFT	✘	✘	0.6980±0.0115	0.6015±0.0121	0.4073±0.0109	0.2708±0.0145	0.5288±0.0140
		LoRA	✘	✘	0.9414±0.0086	0.7885±0.0073	0.6842±0.0081	0.4041±0.0069	0.7462±0.0092
		QLoRA	✘	✘	0.9402±0.0053	0.7445±0.0066	0.6263±0.0085	0.3915±0.0061	0.7270±0.0085
	CodeLlama-70B	SFT	✘	✘	0.7223±0.0143	0.6245±0.0120	0.4432±0.0131	0.3028±0.0147	0.5675±0.0144
		LoRA	✘	✘	**0.9621±0.0053**	0.8122±0.0069	0.7167±0.0055	0.4324±0.0069	0.7710±0.0061
CodeS	CodeLlama-13B	SFT	✘	✔	0.9274±0.0084	0.8789±0.0052	0.7069±0.0079	0.5904±0.0038	0.8150±0.0070
C^3-SQL	GPT-3.5	✘	✘	✔	0.9136±0.0068	0.8402±0.0094	0.7731±0.0064	0.6153±0.0080	0.8108±0.0095
DIN-SQL	GPT-4	✘	✘	✔	0.9234±0.0059	0.8744±0.0080	0.7644±0.0091	0.6265±0.0103	0.8279±0.0098
DAIL-SQL	GPT-4	✘	✘	✔	0.9153±0.0103	0.8924±0.0125	0.7701±0.0098	0.6024±0.0107	0.8308±0.0110
EPI-SQL	GPT-4	✘	✘	✔	0.9310±0.0121	0.9053±0.0085	0.8178±0.0108	0.6189±0.0097	0.8511±0.0114
SuperSQL	GPT-4	✘	✘	✔	0.9435±0.0074	0.9126±0.0050	0.8333±0.0062	0.6867±0.0055	0.8682±0.0068
PURPLE	GPT-4	✘	✘	✔	0.9404±0.0086	**0.9206±0.0041**	0.8268±0.0055	0.6715±0.0080	0.8670±0.0072
SGU-SQL	GPT-4	✘	✔	✔	0.9352±0.0061	0.9190±0.0043	**0.8437±0.0045**	**0.7213±0.0067**	**0.8795±0.0063**

Performance on BIRD

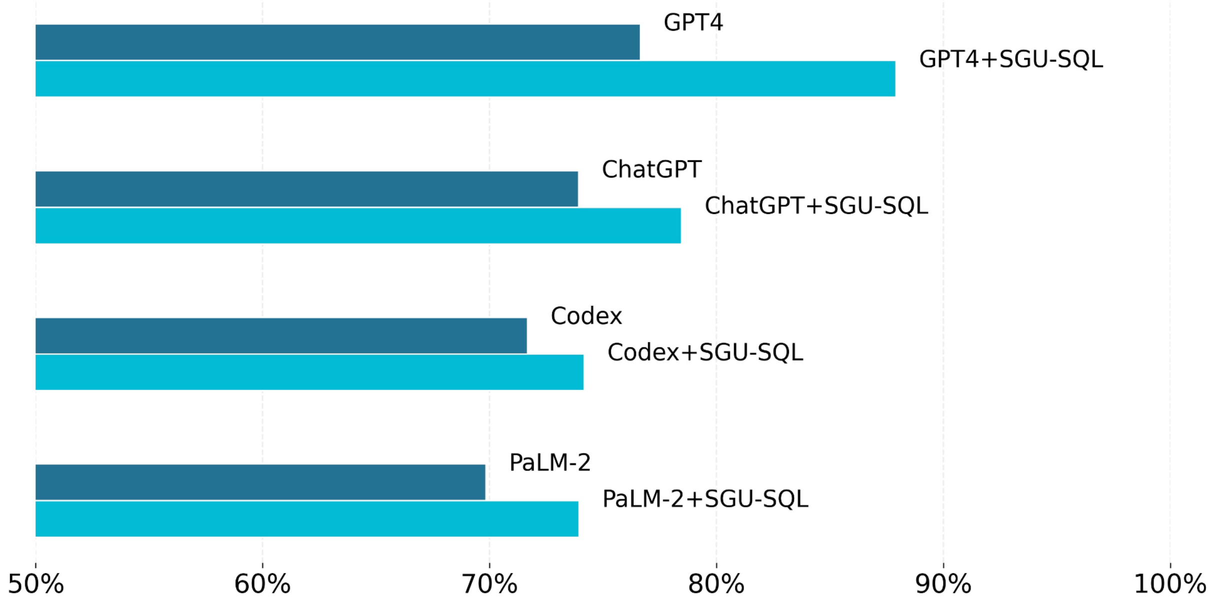
BIRD features over 12,751 unique question-SQL pairs:

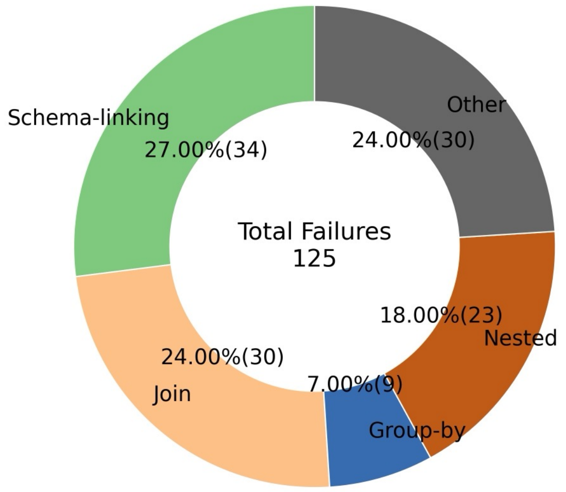
- encompass 95 large databases with a total size of 33.4 GB;
- encompass 37 professional domains.

	Dataset	Spider			BIRD		
	Metric	EX Acc	EM Acc	VES	EX Acc	EM Acc	VES
In-Context Learning	GPT-3.5	0.7394	0.5327	0.7457	0.3562	0.3041	0.3415
	GPT-4	0.7665	0.5892	0.7390	0.4633	0.4255	0.4794
	PaLM-2	0.6985	0.4438	0.7148	0.2735	0.2543	0.3061
	CodeX	0.7167	0.4905	0.7011	0.3438	0.3019	0.3496
	C^3-GPT	0.8108	0.7036	0.8009	0.5020	0.4143	0.5077
	DIN-SQL	0.8279	0.7187	0.8173	0.5072	0.4398	0.5879
	DAIL-SQL	0.8308	0.7443	0.8317	0.5434	0.4581	0.5576
	DTS-SQL	0.8269	0.7260	0.8163	0.5581	0.4825	0.6038
	CodeS	0.8150	0.7069	0.8092	0.5714	0.4893	0.6120
	SuperSQL	0.8682	0.7589	0.8410	0.5860	0.4745	0.6067
	MAC-SQL	0.8635	0.7545	0.8541	0.5759	0.4906	0.5872
	SGU-SQL	**0.8795**	**0.7826**	**0.8652**	**0.6180**	**0.5144**	**0.6393**

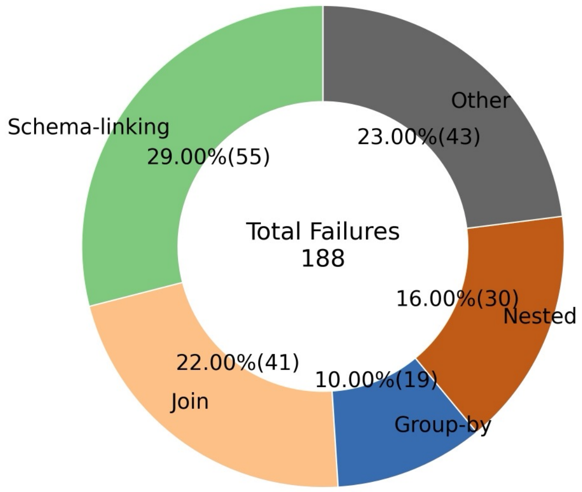
- EX Acc: execution accuracy
- EM Acc: exact-set-match accuracy
- VES: valid efficiency score

Performance of SGU-SQL with Different LLMs





Ours(SGU-SQL+GPT4)



C3-GPT

- LLMs with stronger reasoning abilities exhibit greater improvement.

- Error analysis:
  - SGU-SQL has 125 failures
  - Baseline C3-GPT has 188 failures

Future Work of LLM-based Text-to-SQL

- Users are lazy and have random questions.
- Robustness on poorly documented databases.
- Enterprise databases: SQL queries with complex multi-layer nested structures and an average token count exceeding 100.
  - including full schema may exceed LLMs' maximum token length.
  - High API cost & long SQL generation time.
- Inference speed of LLM-based text-to-SQL methods is slow.
- Data privacy & interpretability
  - calling proprietary APIs to handle local databases with confidentiality pose a risk of data leakage.
  - fine-tuning methods are costly.

- Text to API coding & text-to-code.
- Extensions on multilingual and multi-modal scenarios.