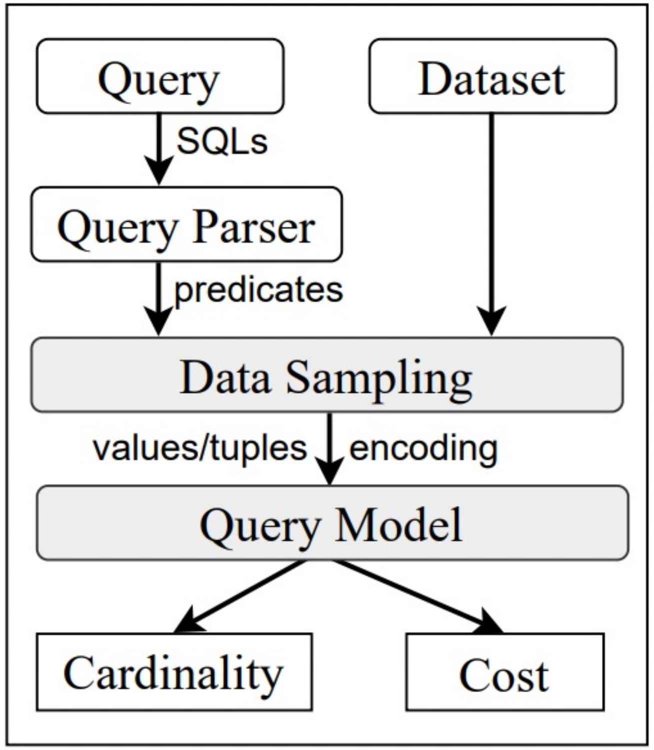
PreQR: Pre-training Representation for SQL Understanding

Xiu Tang, Sai Wu*, Mingli Song,
Shanshan Ying, Feifei Li, Gang Chen

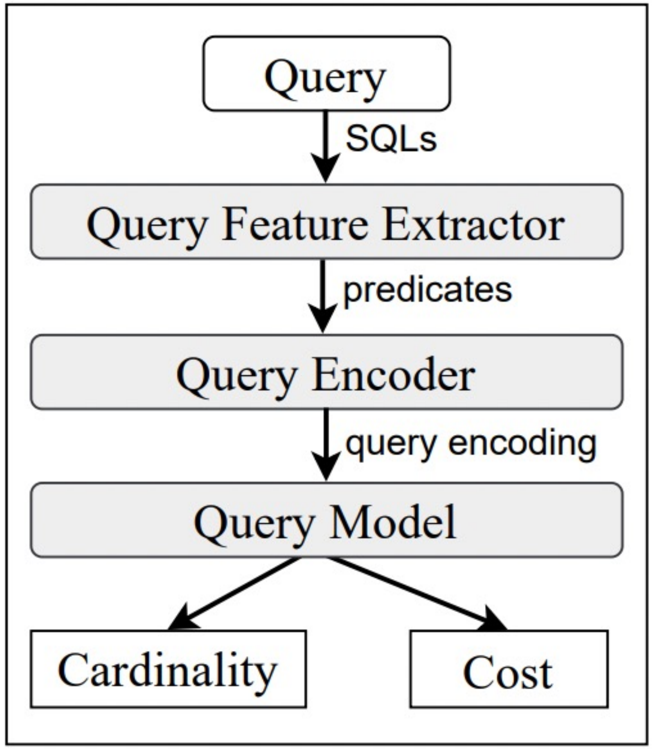
Zhejiang University & Alibaba Group

AZFT (Alibaba-Zhejiang University)

Learning-based Database Optimization

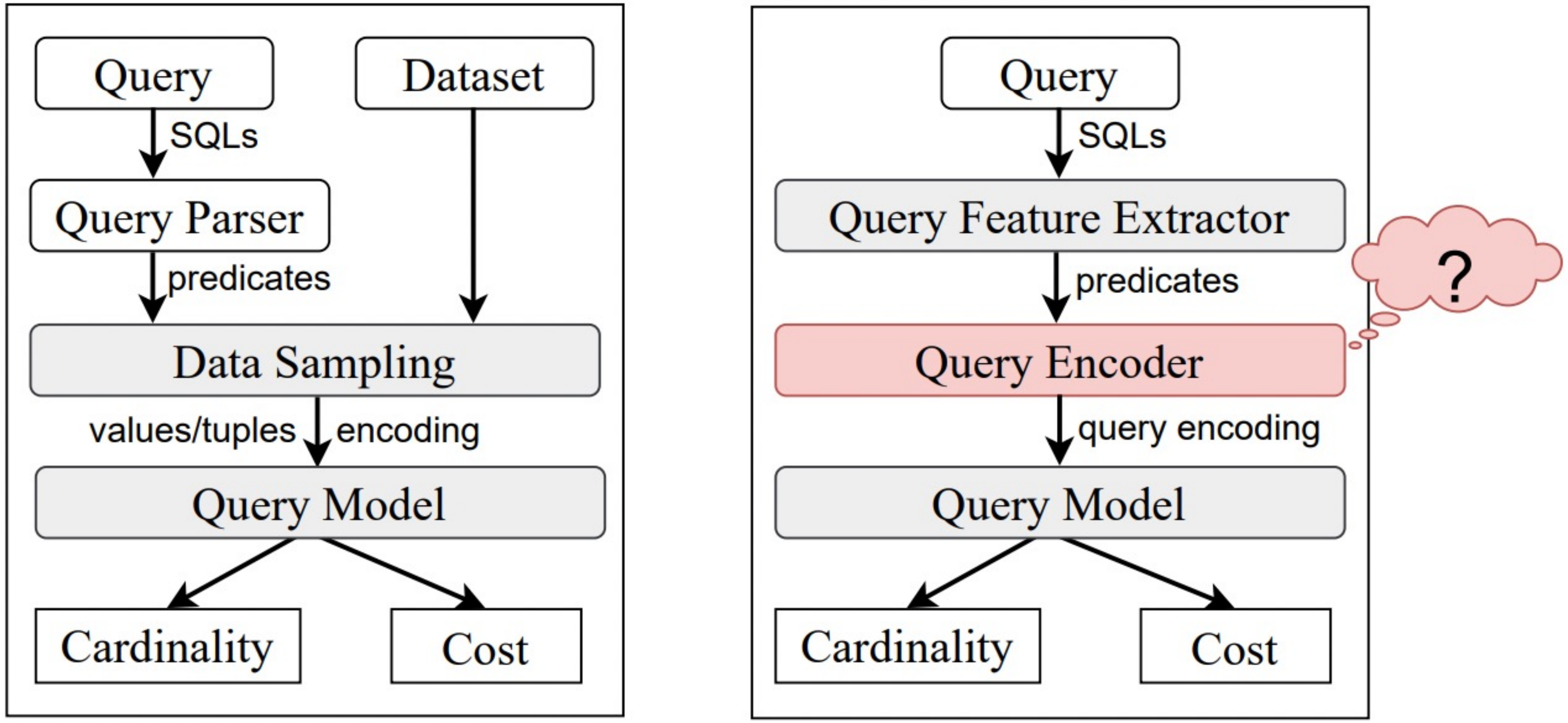


(a) Data Model



(b) Query Model

Learning-based Database Optimization



(a) Data Model (b) Query Model

Previous Approach: One-hot Encoding

- **SQL structure information:**

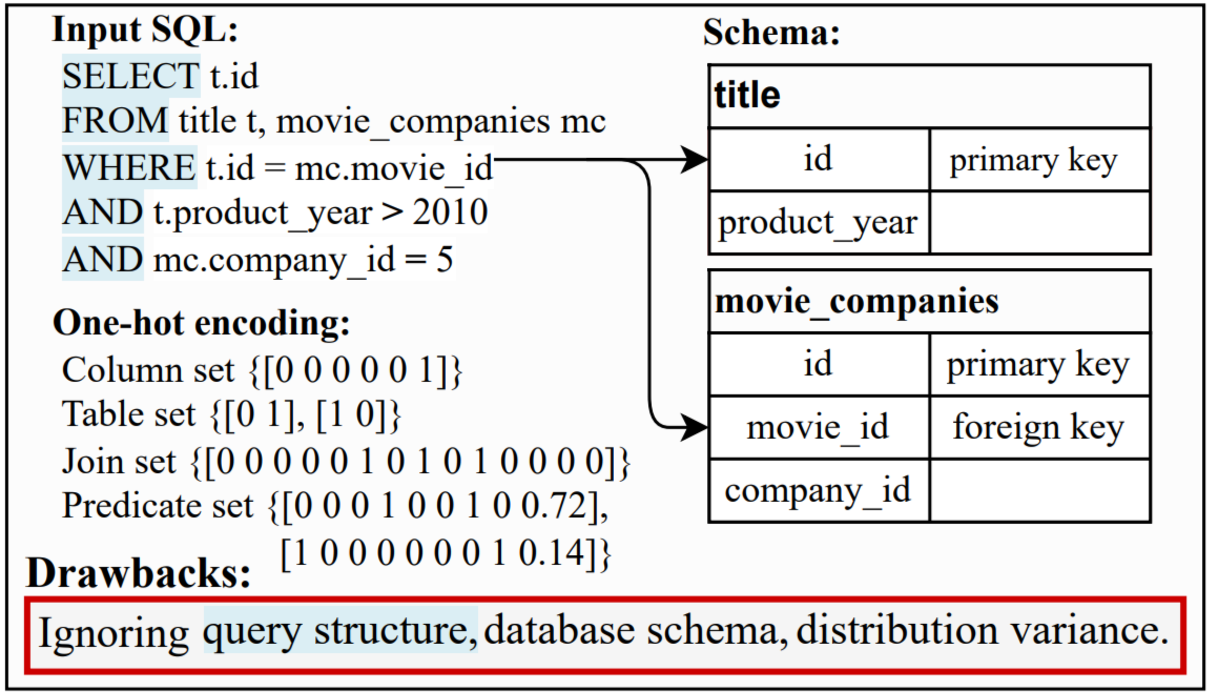
Encoding simply concatenates the encoding of all clauses in the query.

- Database schema information:

All tables and columns use an independent one-hot encoding.

- Database column value distribution information:

All values in SQL are normalized to [0,1].



Previous Approach: One-hot Encoding

- SQL structure information:

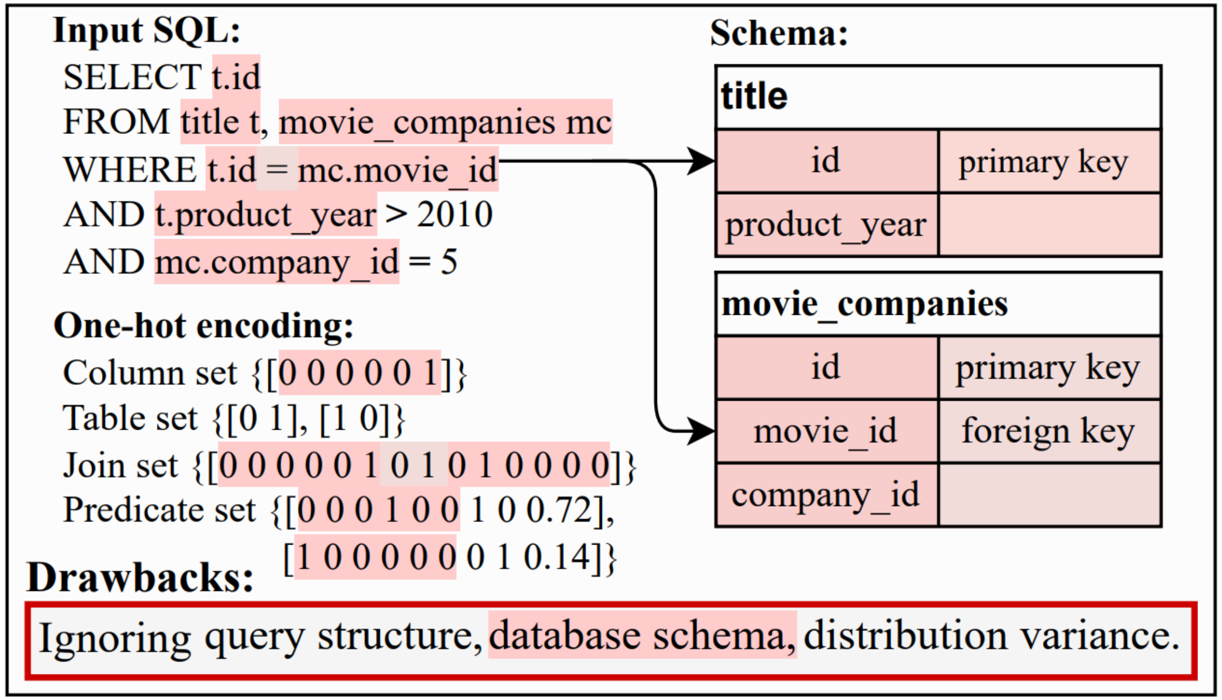
Encoding simply concatenates the encoding of all clauses in the query.

- **Database schema information:**

All tables and columns use an independent one-hot encoding.

- Database column value distribution information:

All values in SQL are normalized to [0,1].



Previous Approach: One-hot Encoding

- **SQL structure information:**

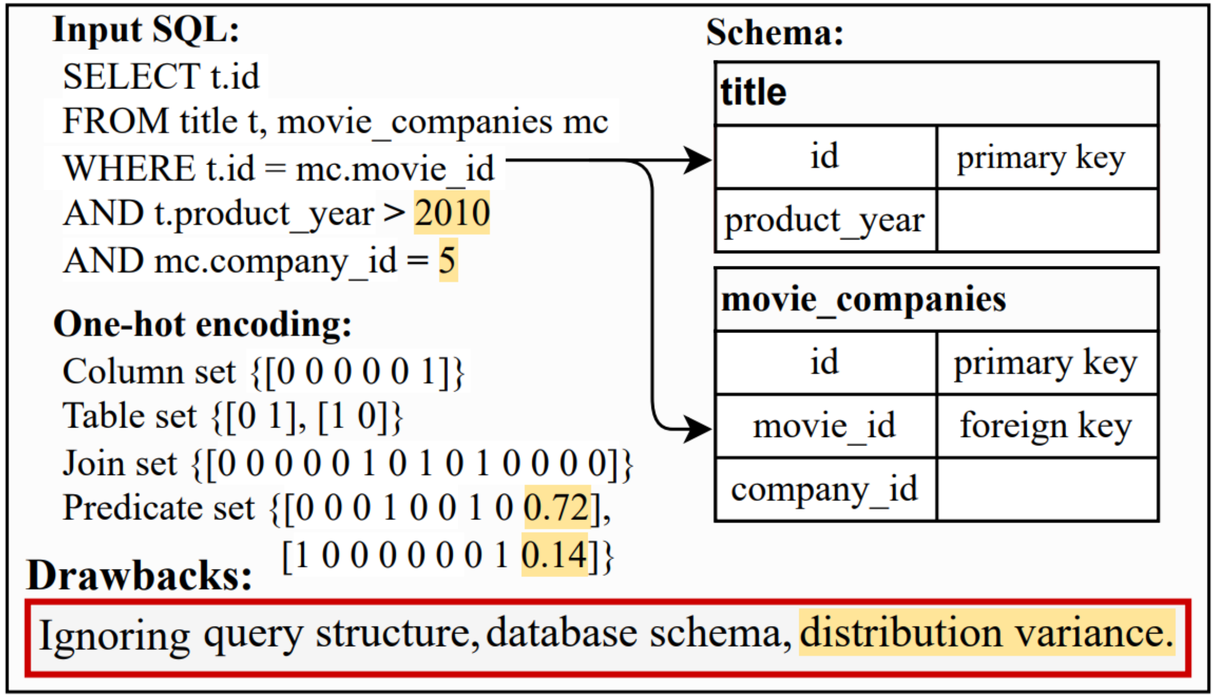
Encoding simply concatenates the encoding of all clauses in the query.

- **Database schema information:**

All tables and columns use an independent one-hot encoding.

- **Database column value distribution information:**

All values in SQL are normalized to [0,1].



Previous Approach: One-hot Encoding

- **SQL structure information:**

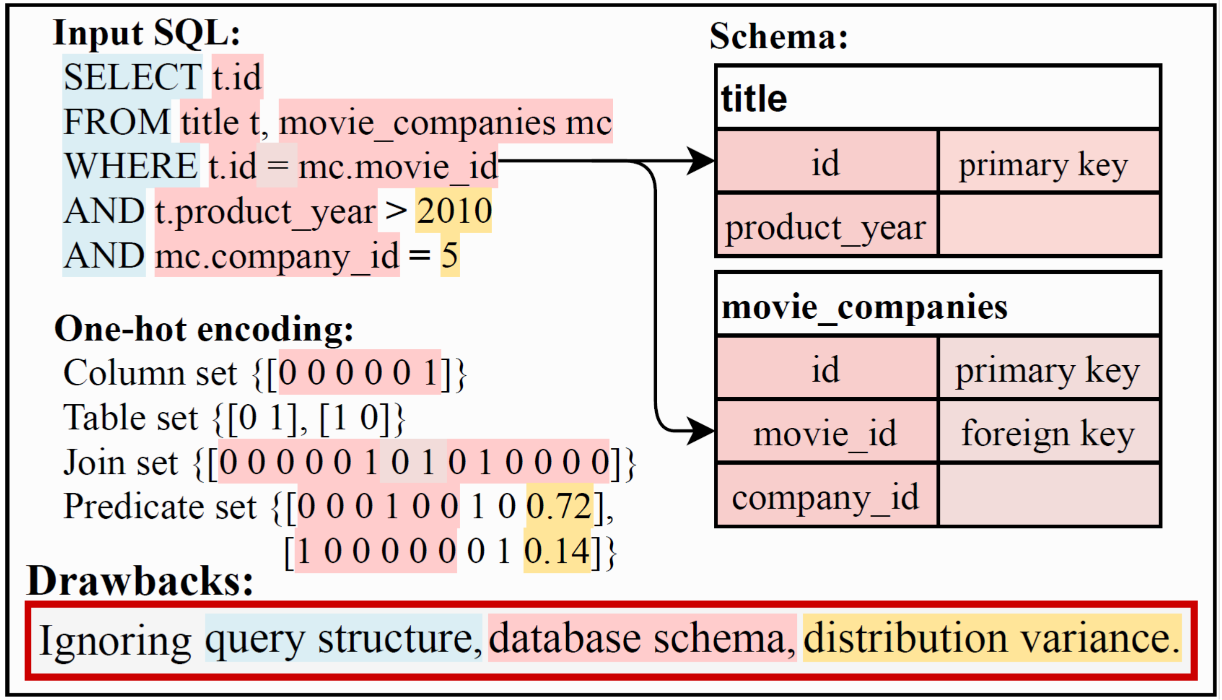
Encoding simply concatenates the encoding of all clauses in the query.

- **Database schema information:**

All tables and columns use an independent one-hot encoding.

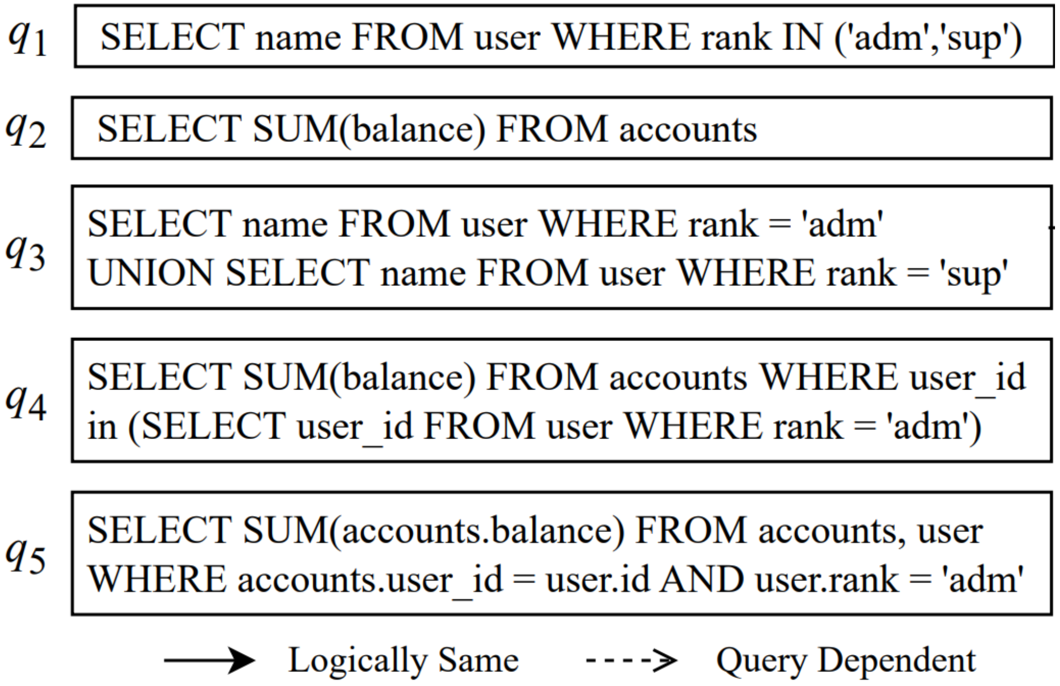
- **Database column value distribution information:**

All values in SQL are normalized to [0,1].



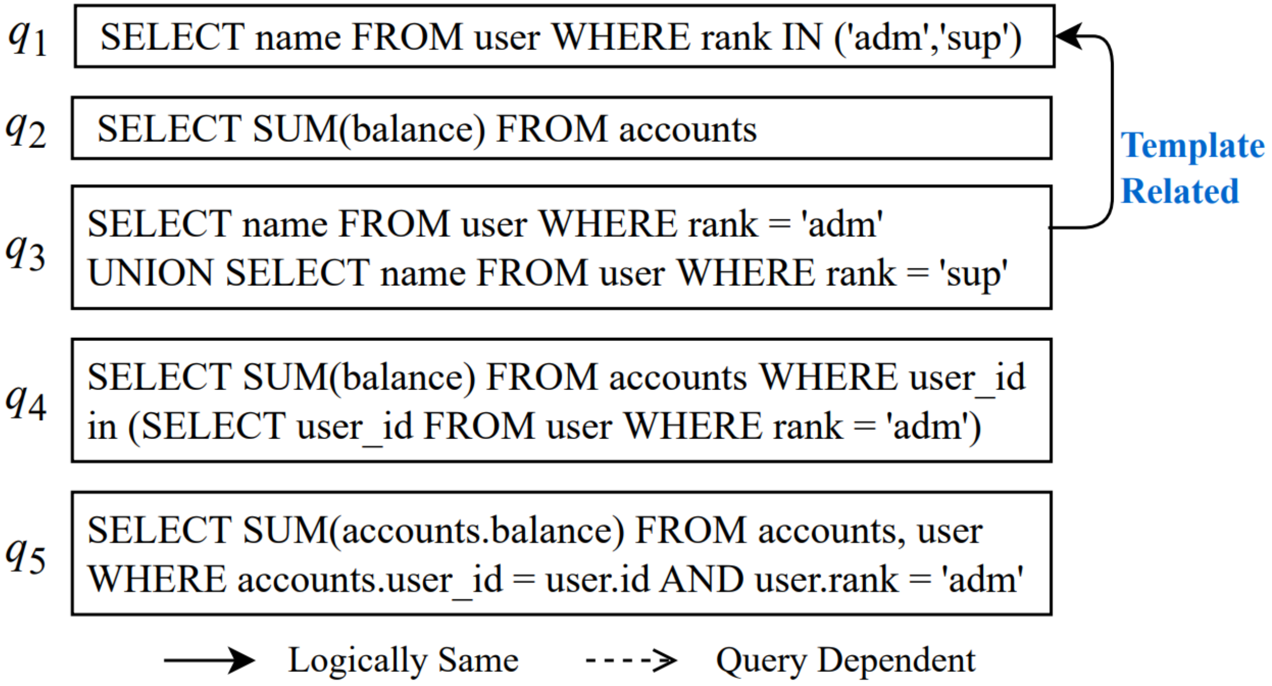
Previous Approach: Pretrained Language Model

- The language representation has been well studied by work on the NLP.
- However, SQL incurs **new challenges:**
  - **Semantically equivalent:**
    - query q_3 and q_1, which can be easily identified by their query structures;
    - query q_5 and q_4, which can be discovered via involved schema information.



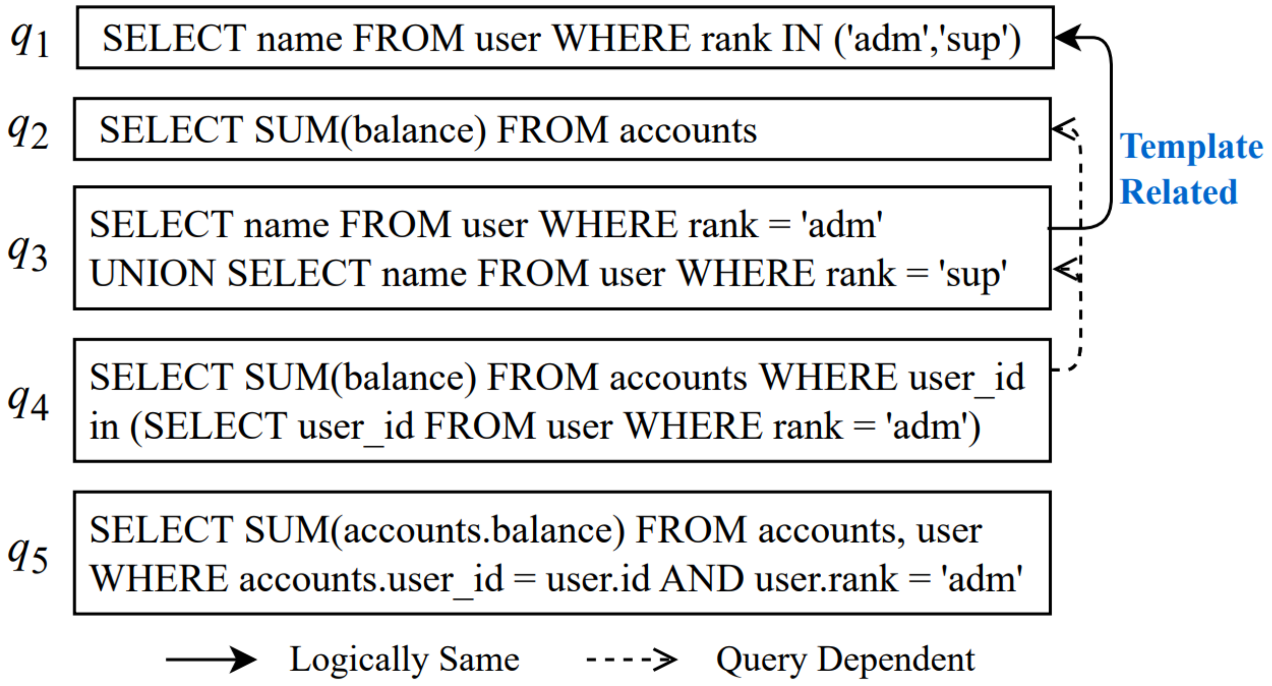
Previous Approach: Pretrained Language Model

- The language representation has been well studied by work on the NLP.
- However, SQL incurs **new challenges:**
  - **Semantically equivalent:**
    - query q_3 and q_1, which can be easily identified by their query structures;
    - query q_5 and q_4, which can be discovered via involved schema information.



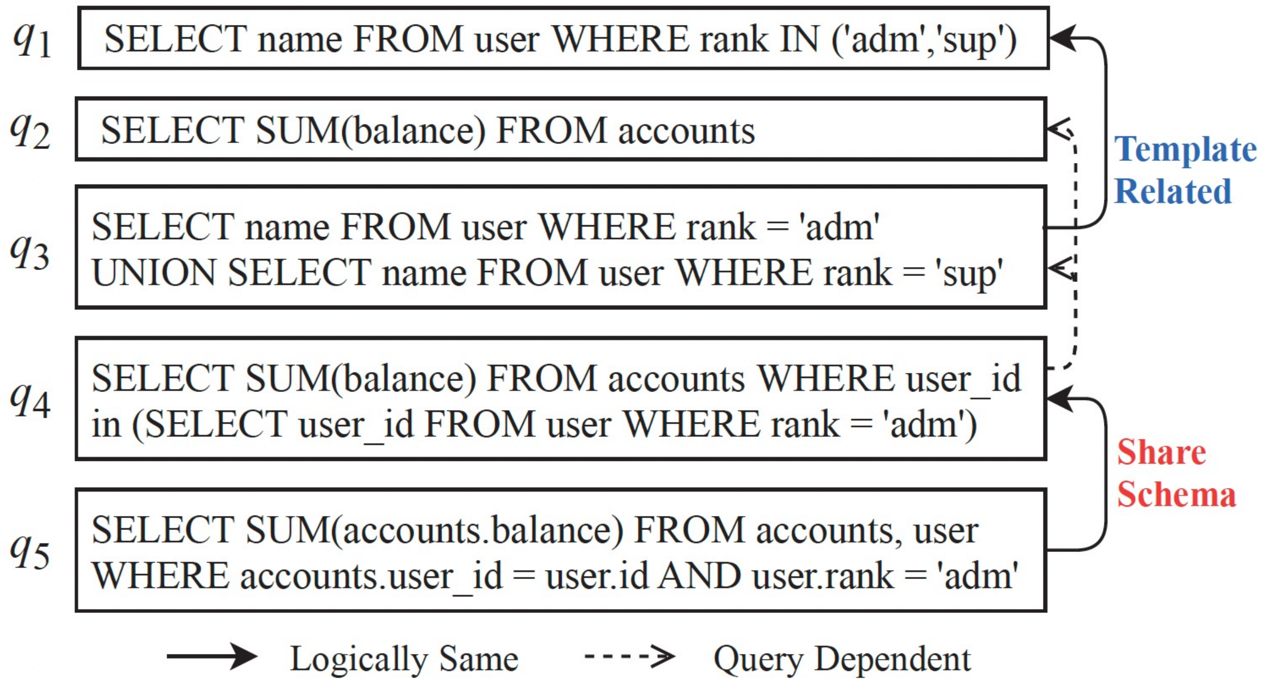
Previous Approach: Pretrained Language Model

- The language representation has been well studied by work on the NLP.
- However, SQL incurs **new challenges:**
  - **Semantically equivalent:**
    - query q_3 and q_1, which can be easily identified by their query structures;
    - query q_5 and q_4, which can be discovered via involved schema information.



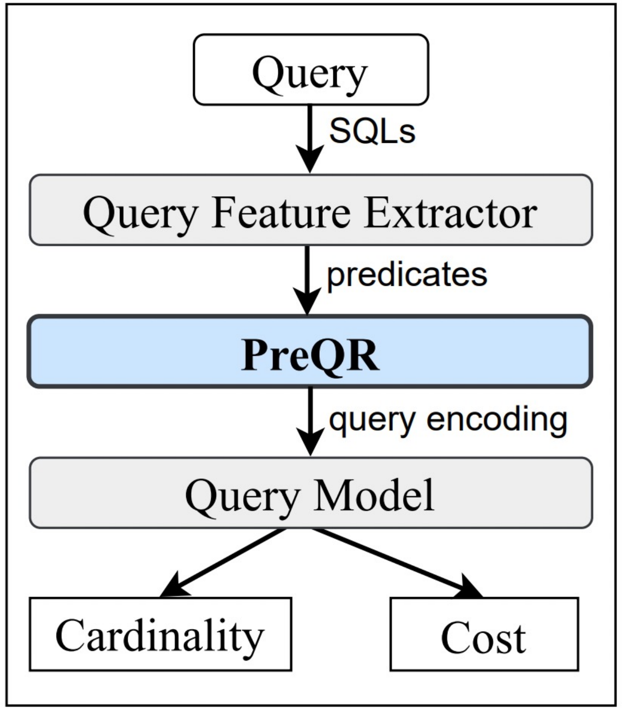
Previous Approach: Pretrained Language Model

- The language representation has been well studied by work on the NLP.
- However, SQL incurs **new challenges:**
  - **Semantically equivalent:**
    - query q_3 and q_1, which can be easily identified by their query structures;
    - query q_5 and q_4, which can be discovered via involved schema information.



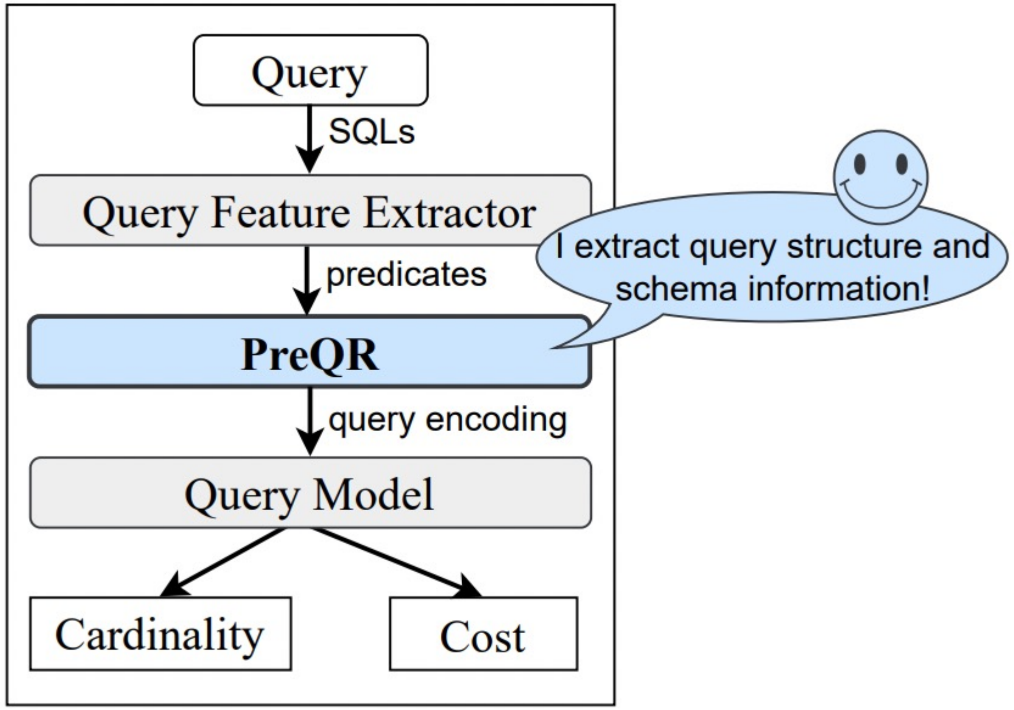
Introducing PreQR

- **PreQR:** Pretraining Query Representation.
- By pretraining query representation, **PreQR:**
  - integrates the database schema, query structure and content knowledge.
  - only needs to be trained once for a database and can be used in various learning tasks.
  - performances on various database tasks obtain a significant improvement.

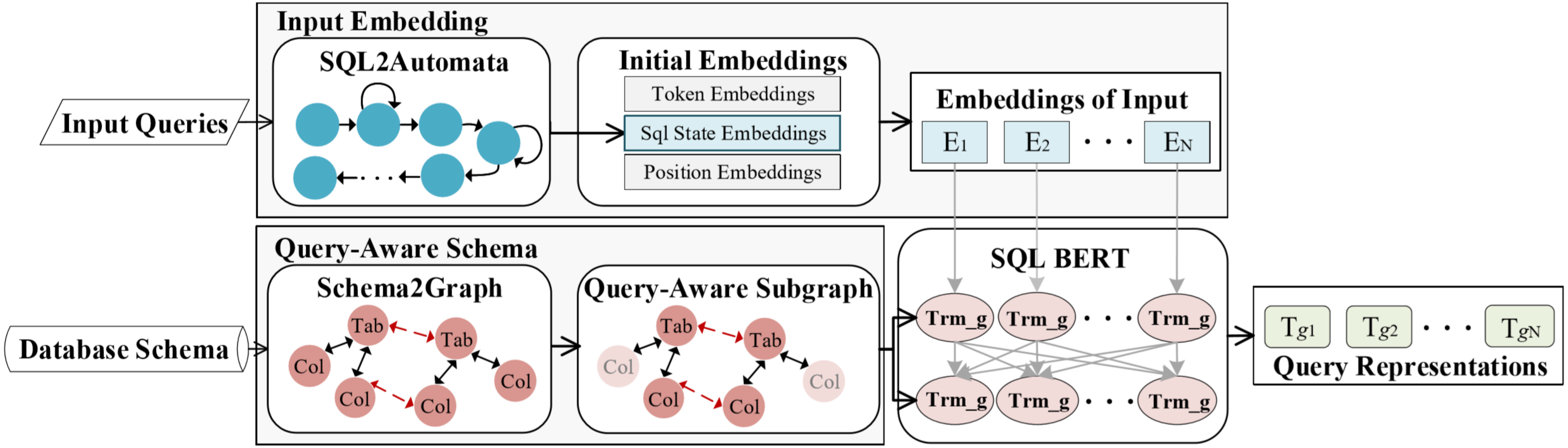


Introducing PreQR

- **PreQR:** Pretraining Query Representation.
- By pretraining query representation, **PreQR:**
    - integrates the database schema, query structure and content knowledge.
    - only needs to be trained once for a database and can be used in various learning tasks.
    - performances on various database tasks obtain a significant improvement.

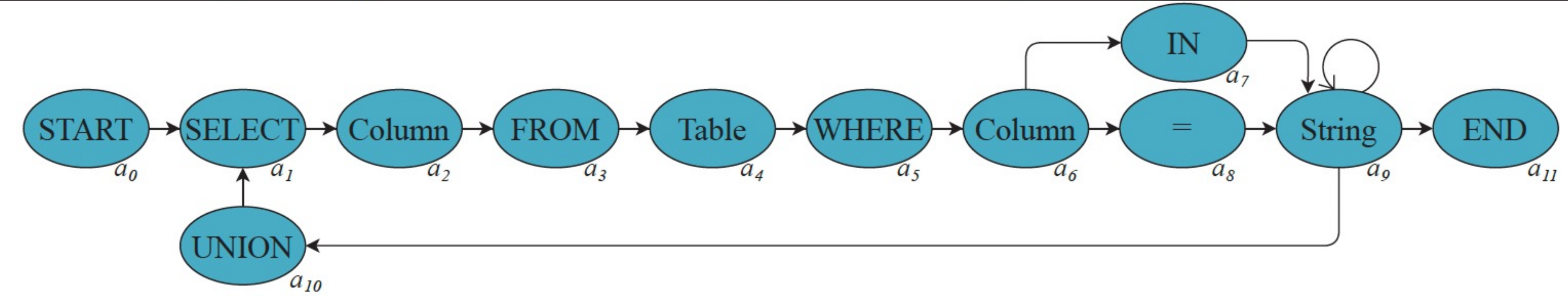


PreQR



- The **input embedding** represents the query structure via matching automaton states.
- The **query-aware schema** use a graph-structured model to encode SQL-related schema information.
- The **SQL BERT encoder** leverages the attention mechanism to identify the query-aware structural and schema information in an ad-hoc way.

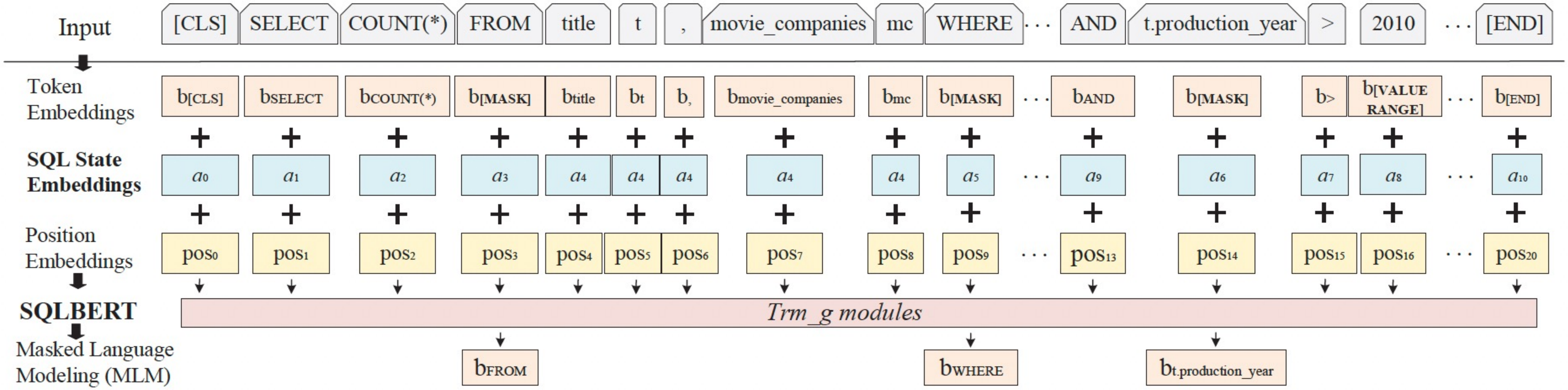
SQL2Automaton

Input	Queries q_1 in Section 1	Queries q_3 in Section 1
Automaton Matching		
SQL State Embedding	$a = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_9, a_9, a_{11})$	$a = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_8, a_9, a_{10},$ $a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_8, a_9, a_{11})$

- PreQR transforms the query structure into a finite-state automaton (FA), which is a machine with a finite number of states.

- Automata can recognize syntactically well-formed strings to represent the semantic structure of SQL.

PreQR Input Representation



Schema2Graph

Schema

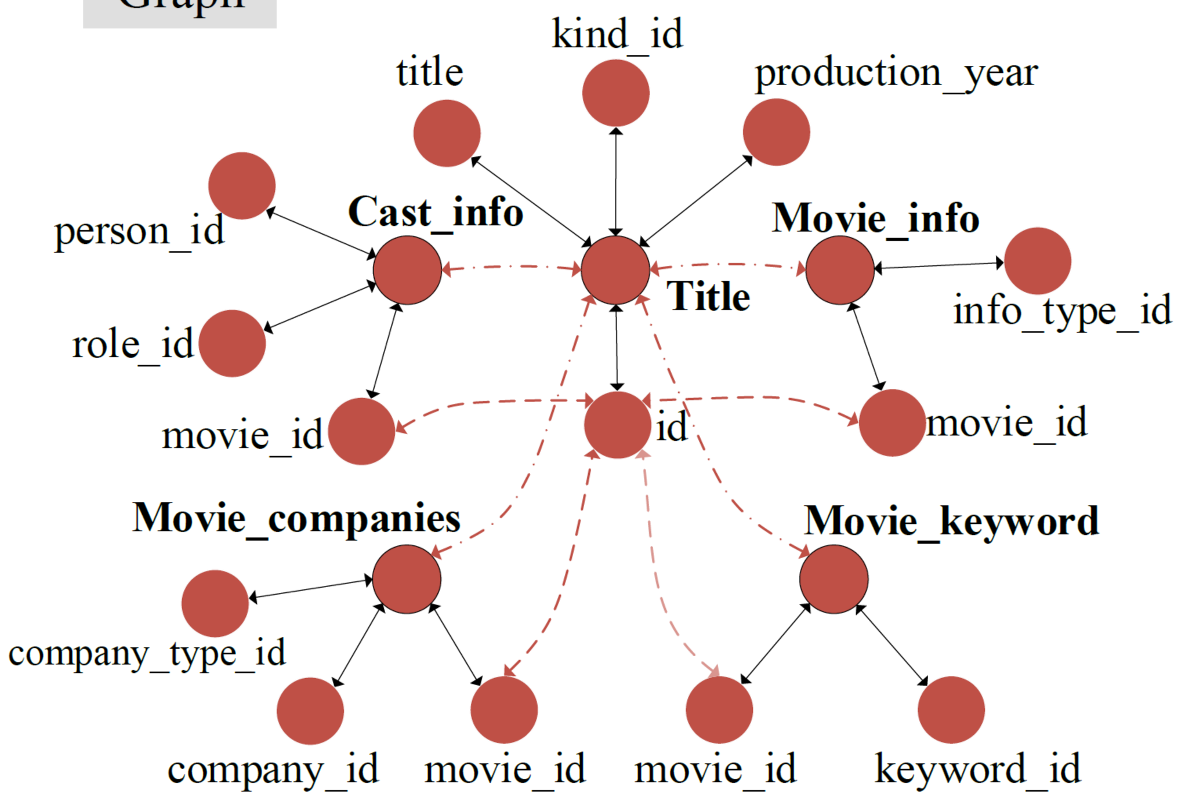
Tables: $\boldsymbol{T}$ = (*Title, Movie_keyword, Cast_info, Movie_info, Movie_companies, ...*)

Columns: $\boldsymbol{C_{title}}$ = {*id, title, kind_id, production_year, ...* }

$\boldsymbol{C_{movie\_companies}}$ = {*movie_id, company_id, company_type_id, ...* }

Foreign: $\boldsymbol{F}$ = {(*title.id, movie_companies.movie_id*), (*title.id, movie_info.movie_id*), ... }

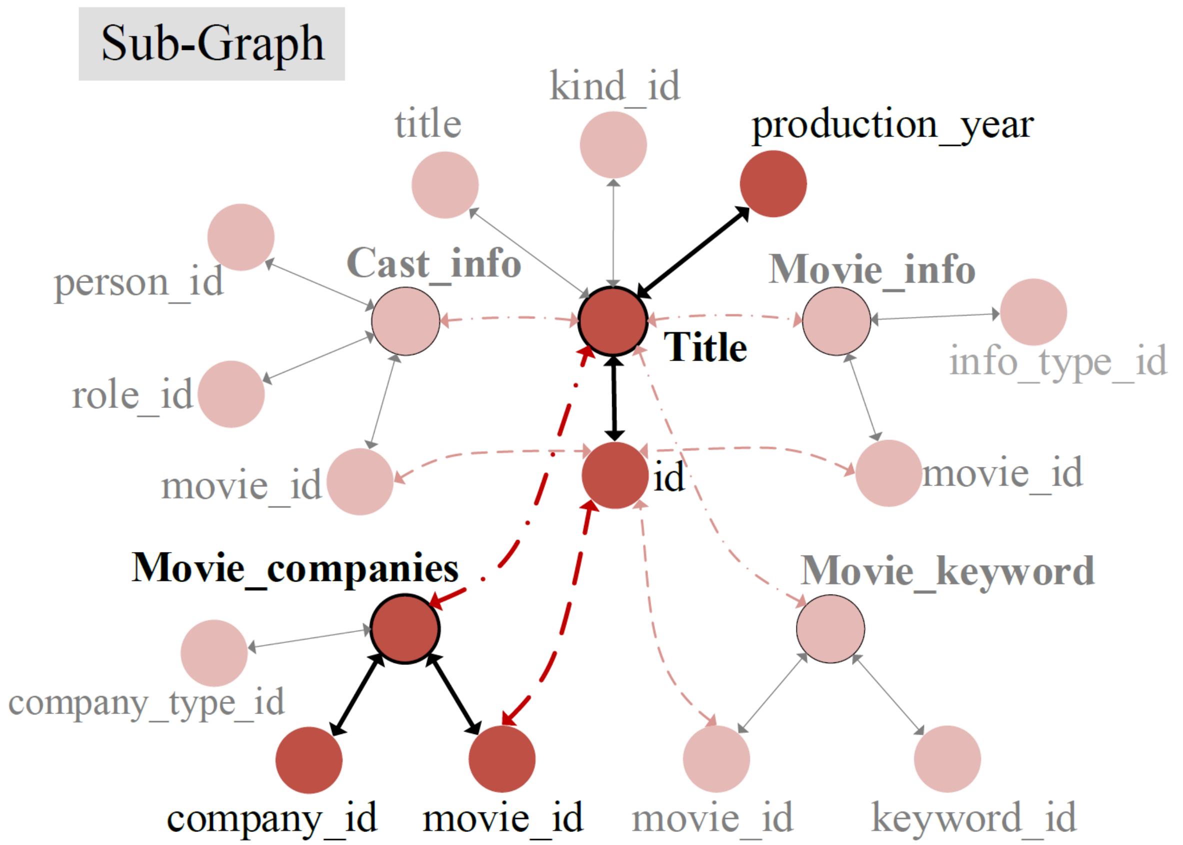
Graph



Query-Aware Schema

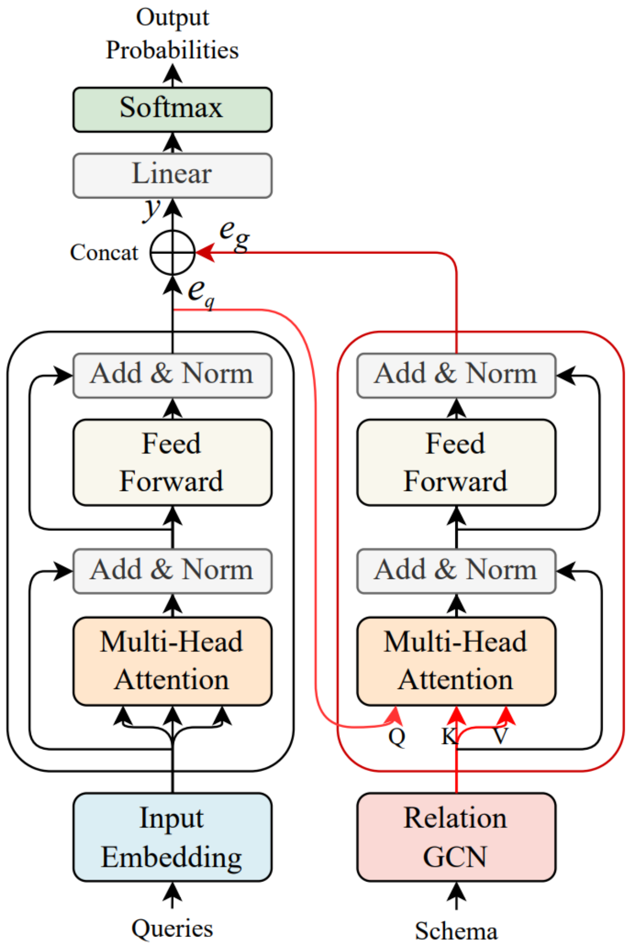
Input

Query: ***q*** = *"SELECT COUNT(*) FROM title t, movie_companies mc WHERE t.id = mc.movie_id AND t.production_year > 2010 AND mc.company_id = 5 "*



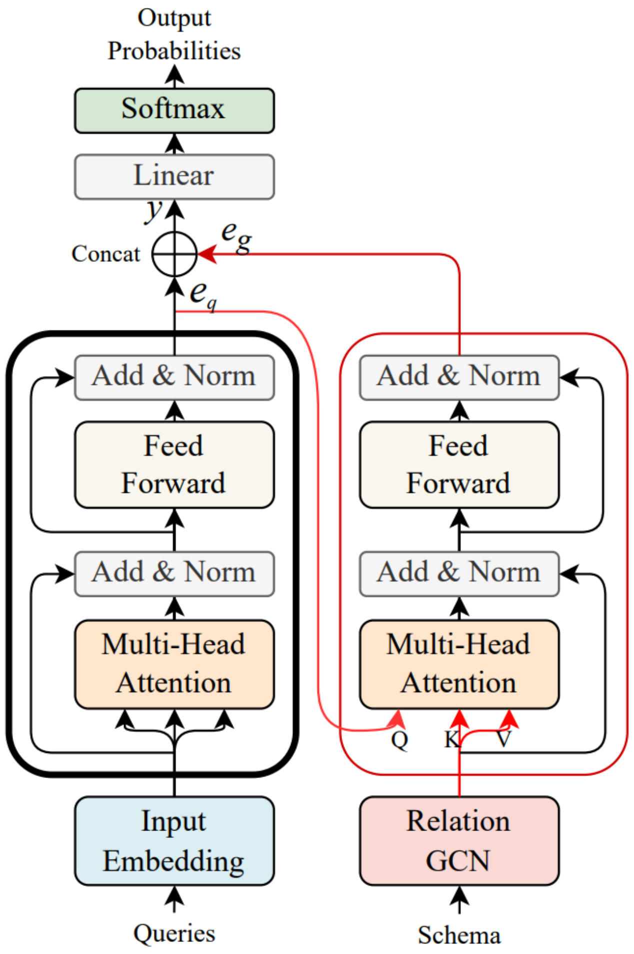
Trm_g Module in PreQR

- *Trm_g* architecture is a variant of the Transformer from BERT.
- The *Trm_g* model includes the original Transformer *Trm* (black rectangle) and our query-aware sub-graph Transformer *Trm'* (red rectangle).
- PreQR augments each word with the graph structure of the schema items that it is linked to.



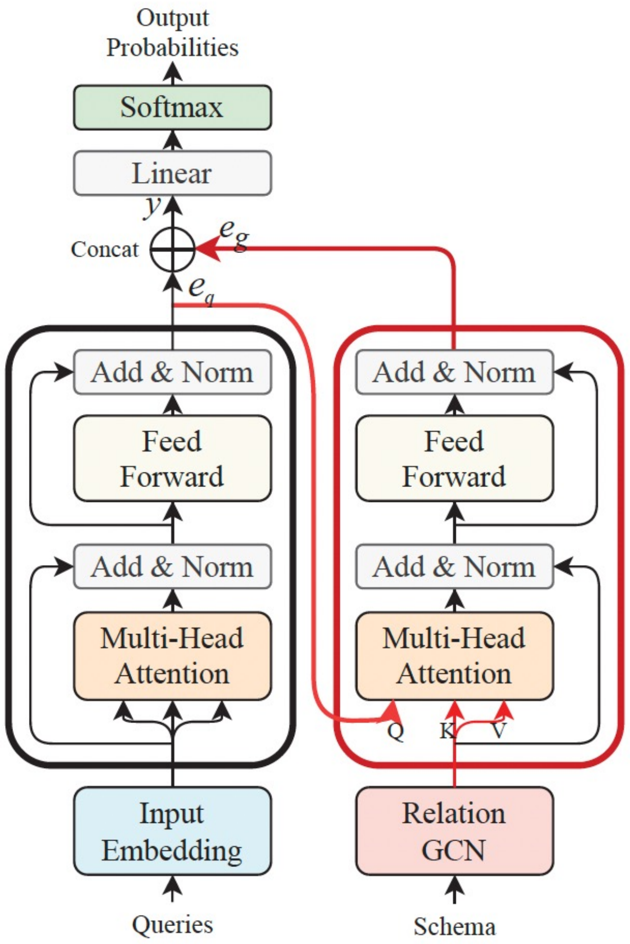
Trm_g Module in PreQR

- *Trm_g* architecture is a variant of the Transformer from BERT.
- The *Trm_g* model includes the original Transformer *Trm* (black rectangle) and our query-aware sub-graph Transformer *Trm'* (red rectangle).
- PreQR augments each word with the graph structure of the schema items that it is linked to.



Trm_g Module in PreQR

- *Trm_g* architecture is a variant of the Transformer from BERT.
- The *Trm_g* model includes the original Transformer *Trm* (black rectangle) and our query-aware sub-graph Transformer *Trm'* (red rectangle).
- PreQR augments each word with the graph structure of the schema items that it is linked to.



Extensibility

- Case 1: The distribution of data changes significantly.
- Case 2: If the database schema is updated, we need to update the schema graph model G_s.
- Case 3: When query patterns change, we may need to update the FA to handle new queries.
- Case 4: Training a new embedding model for a database from scratch.

Case	Description	Time
Case 1	Incremental learning for the last layer of *SQLBERT*	15min
Case 2	Incremental Learning for the *Schema2Graph* part	3.5h
Case 3	Incremental learning for the *Input Embedding* module	6.7h
Case 4	Train from scratch	18.3h

Experiment Highlight

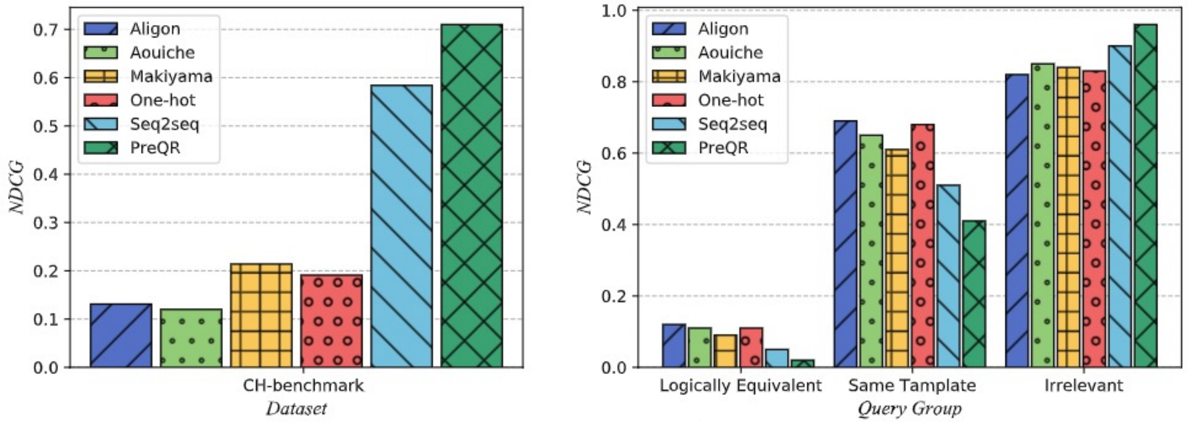
PreQR handles various downstream tasks:

- Query Clustering:

  Comparing with five approaches to measure pairwise similarity between queries.

- SQL-to-Text Generation:

  Comparing the encoding of PreQR model against the Seq2Seq, Tree2Seq and Graph2Seq.



(a) Similarity ranking validation (b) Query group distance

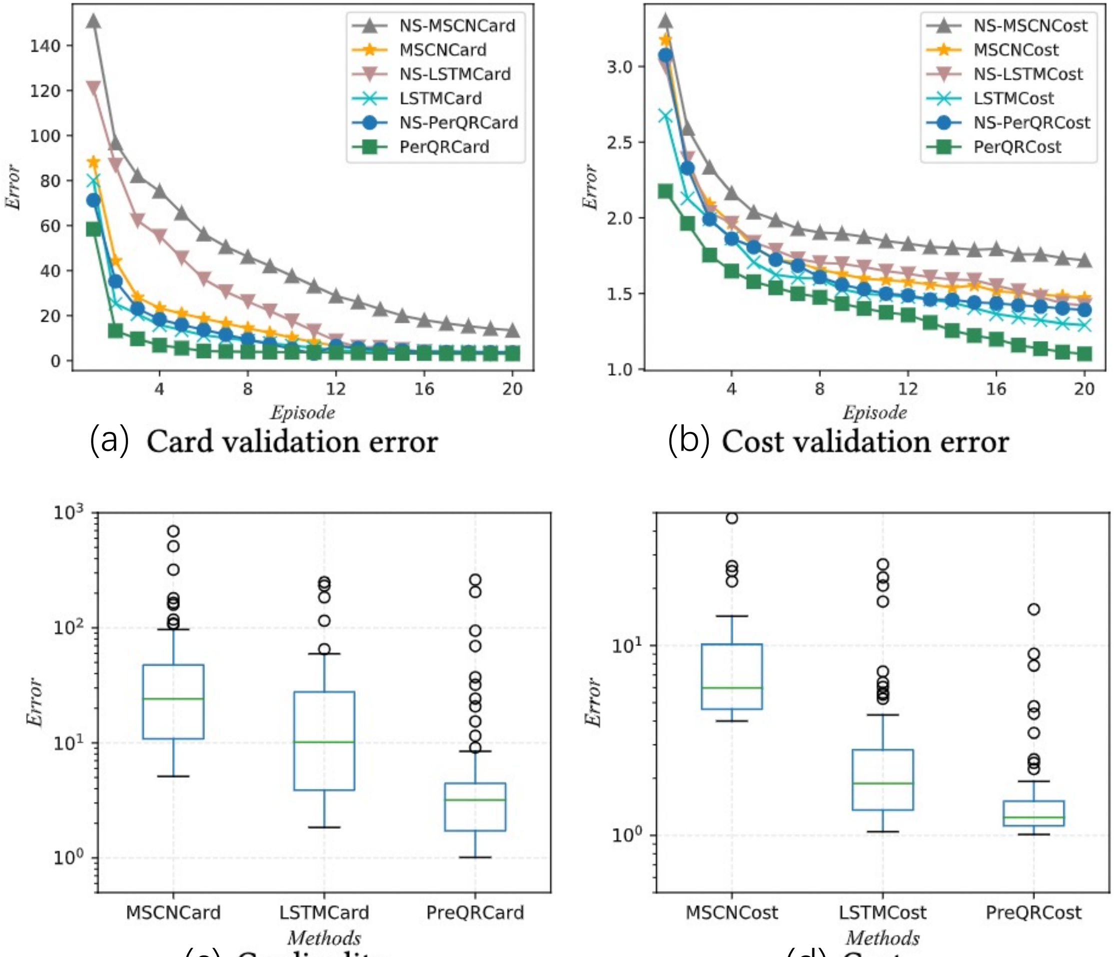
SQL	SELECT opponent WHERE points < 18 AND November > 11;
Seq2Seq	What is the opponent when the points are less than 18 with the November is more than 11 ?
PreQR	Which opponent has the points less than 18, and the November more than 11 ?

Experiment Highlight

- Query Cardinality and Cost Estimation:

  Comparing with a conventional method (PostgreSQL), the query-based learning models (MSCN and LSTM), and a data-based learning model (NeuroCard).

- The experimental results showed that by replacing the encoders of existing models with PreQR encoding, performances on various database tasks obtain a significant improvement.



(a) Card validation error (b) Cost validation error

(c) Cardinality (d) Cost

PreQR

- PreQR: towards pre-training SQL embedding.

Xiu Tang

- Email: tangxiu@zju.edu.cn