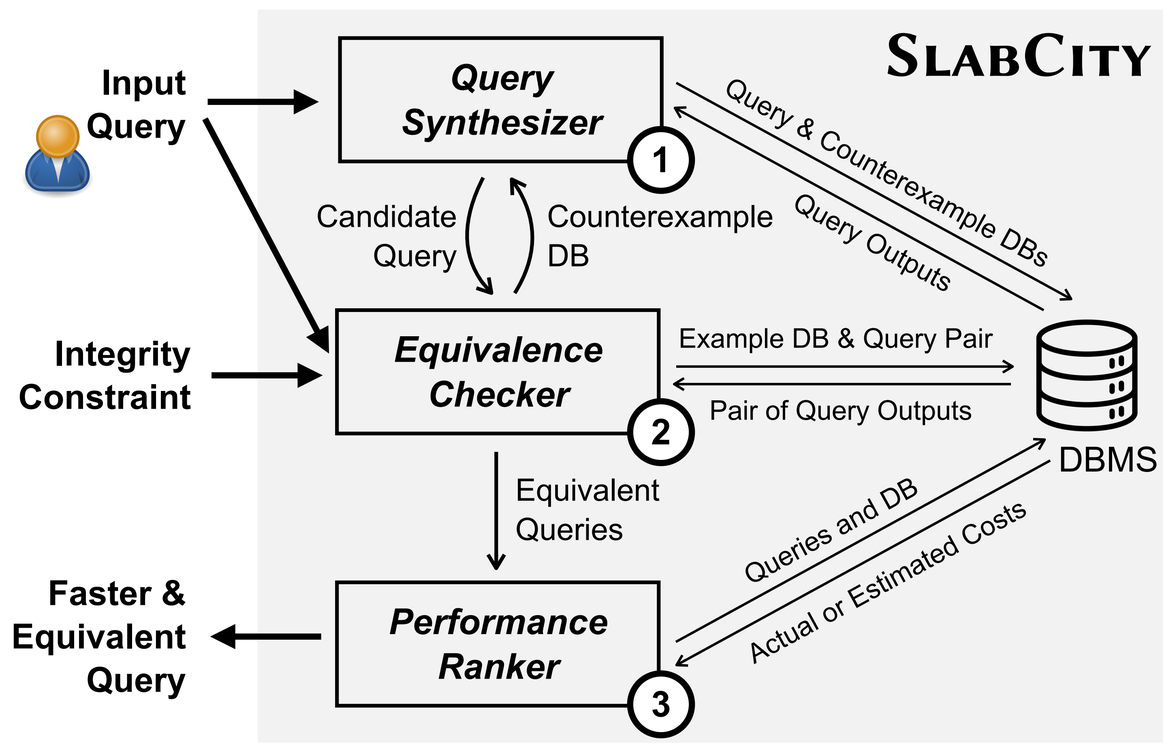
SlabCity: Whole-Query Optimization using Program Synthesis

Rui Dong*, Jie Liu*, Yuxuan Zhu, Cong Yan, Barzan Mozafari, Xinyu Wang

University of Michigan & Microsoft Research

Workflow of SlabCity



Motivation

- Poorly-written queries are a major problem in the industry.
- **Query rewriting** transforms a query into another that is semantically equivalent but faster.
- Current query rewriting solutions usually rely on **rewrite rules**.
- However, it is impossible to enumerate every slow query pattern and design a rewrite rule for each of them.
- **If there is a slow query pattern not covered by existing rules, how can we optimize it?**

Let us look at two queries both written by real LeetCode users for **calculating running total**.

Q1

```
SELECT DISTINCT s1.gender,
                s1.day,
                SUM(s2.score)
FROM scores AS s1 JOIN scores AS s2 ON s1.gender = s2.gender
WHERE s2.day <= s1.day
GROUP BY s1.gender, s1.day
```

Q2

```
SELECT gender,
       day,
       SUM(score) OVER PARTITION BY gender ORDER BY day
FROM scores
```

- **Q2** is much faster than **Q1**.
- People who wrote **Q1** may not realize **Q1** is very inefficient.
- People who wrote **Q2** may not understand why someone would write a query like **Q1**.
- It's hard to design a rule rewriting **Q1** to **Q2** before you see **Q1**.

Key Contributions

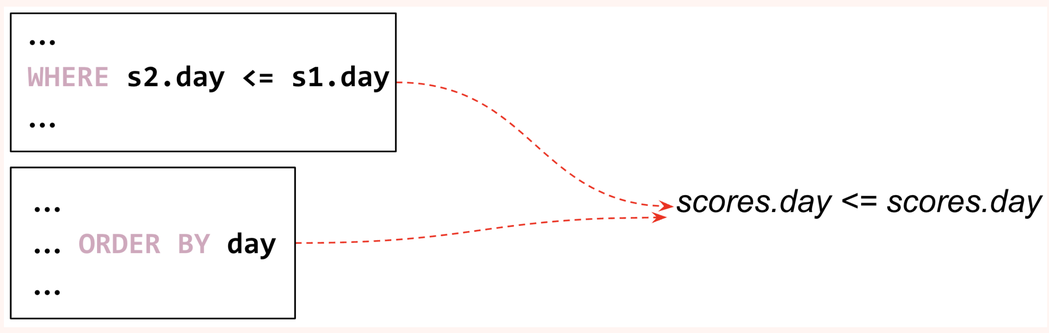
- Propose the first synthesis-based query rewriting technique capable of whole-query optimization **without requiring predefined rewrite rules**.
- Define **dataflows** for SQL queries and exploit them for efficient query synthesis.
- Contribute a new benchmark by curating more than 1000 real-life queries from LeetCode participants.

Approach

Dataflows capture how data is computed and used

scores.gender	*scores.gender = scores.gender*
scores.day	*scores.day <= scores.day*
scores.score -> SUM	*GroupBy(scores.gender, scores.day)*

Equivalent queries often shares dataflows

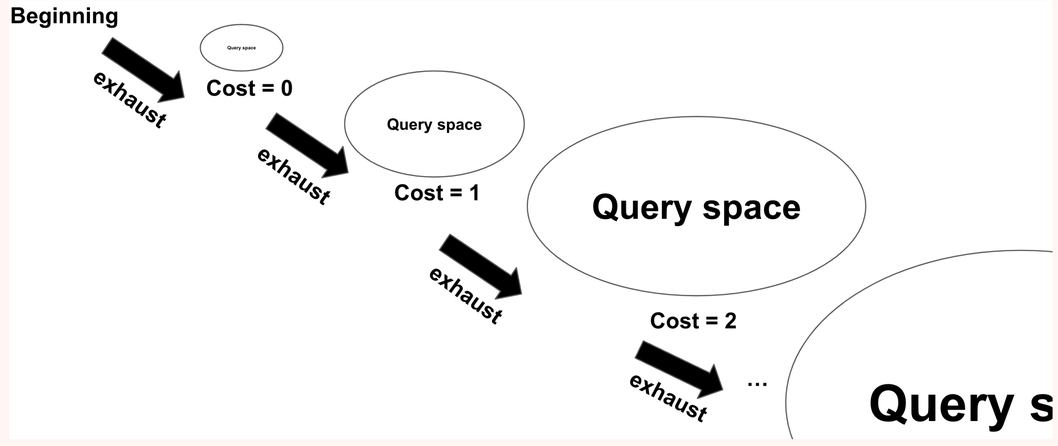


Derive query cost using dataflows

Given a input query **Q**, for any query component **C** in search space,

cost(**C**) = # of dataflows in **C** but not in **Q**

Search queries in ascending order based on cost



Evaluation

Four **workloads**: LeetCode user submissions, Calcite rule testing pairs, TPC-H, TPC-DS

Two **baselines**: WeTune[1], LearnedRewrite[2]

- **Optimization Coverage**: SlabCity can optimize more queries.
- **Latency Reduction**: SlabCity can find faster queries.

Acknowledgements

We thank Lin Ma, and the anonymous reviewers for the helpful feedback. This work was supported by the National Science Foundation under Grant No. CCF-2210832

References

[1] Zhaoguo Wang, Zhou Zhou, Yicun Yang, Haoran Ding, Gansen Hu, Ding Ding, Chuzhe Tang, Haibo Chen, and Jinyang Li. Wetune: Automatic discovery and verification of query rewrite rules. In *Proceedings of the 2022 International Conference on Management of Data*, pages 94–107, 2022.

[2] Xuanhe Zhou, Guoliang Li, Chengliang Chai, and Jianhua Feng. A learned query rewrite system using monte carlo tree search. *Proceedings of the VLDB Endowment*, 15(1):46–58, 2021.