Symbolic Reasoning for SQL Queries

Chenglong Wang, Kaiyuan Zhang and Shumo Chu

Abstract

SQL is a commonly used data query language for database systems. Due to the rich and complex nature of SQL queries, understanding SQL queries is not easy, and this difficulty leads to challenges in the following two scenarios: 1) checking correctness of a SQL optimization rule and 2) debugging a partially corrected SQL. Concretely, these challenges present because SQL is a highly abstract declarative language, and the semantics of a complex SQL query is hard to be inferred directly from its syntax.

To address these challenges, we consider building the formal semantics of SQL in Rosette and employing automatic verification techniques to reason about it. Particularly, we consider the following two applications: 1) bounded checking equivalence of two SQL queries to help verify the correctness of SQL rewriting rules (or provide counter examples when exist), and 2) SQL synthesis based on partially correct queries and counter examples, i.e. help user to correct their wrong queries using counter examples.

1 Introduction

1.1 Verifying query rewriting

Rewriting SQL query to equivalent queries is an essential part of query optimizer. Reasoning the correctness of complicated rewriting rigorously requires a lot of effort and error prone. Using the evaluation semantic that we developed, we can build tool to help verify the correctness of query rewriting. Given Q_1 and Q_2, we can use the SMT solver to verify the validity of $Q_1 \leftrightarrow Q_2$. For complex query rewriting, we may need to bound the size and arity of the relation. Then the verification will not be complete but not sound. However, we believe that being able to find counter example in bounded case is still helpful compared with manually reasoning.

2 Related Work

Symbolic SQL Reasoning Qex [3, 4] is a tool to generate tables based on parameterized SQL queries for database system test purpose. Concretely, Qex takes an parameterized SQL query as well as a SQL property as inputs, it will then generate a full SQL query as well as a table instance satisfying the property for database unit test. Qex has build the background theory for SQL queries and is able to translate a SQL query into SMT formulas. Our project uses the same reasoning method as Qex do, but for different application scenario, i.e. SQL semantics checking.

Equivalence of SQL Queries Chirkova et al. [2] presents a method to evaluate SQL query equivalence with presence of embedded dependencies. However, their technique addresses only a subset of SQl grammar (conjunctive queries plus aggregation), it is different from our target of equivalence checking for SQL queries with nested subqueries.

Chakravarthy et al. [1] presents a logic based approach for semantics quey optimization. Optimization rules in their systems are generated by the system so that the optimization process is semantics preserving, differently, we are able to bounded-check the whether two SQL optimization rules are equivalent or not, which is more general than their approach..

References

[1] Upen S. Chakravarthy, John Grant, and Jack Minker. Logic-based approach to semantic query optimization. *ACM Trans. Database Syst.*, 15(2):162–207, June 1990.

[2] Rada Chirkova and Michael R Genesereth. Equivalence of sql queries in presence of embedded dependencies. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 217–226. ACM, 2009.

[3] Margus Veanes, Pavel Grigorenko, Peli De Halleux, and Nikolai Tillmann. Symbolic query exploration. In *Formal Methods and Software Engineering*, pages 49–68. Springer, 2009.

[4] Margus Veanes, Nikolai Tillmann, and Jonathan De Halleux. Qex: Symbolic sql query explorer. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 425–446. Springer, 2010.