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 Motivation

2 Model

3 Application

3.1 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.

3.2