SQL Query evaluation on Incomplete Databases with correctness guarantees

Etienne Toussaint, Leonid Libkin
Laboratory for Foundations of Computer Science, The University of Edinburgh

21st August 2016

The general context

It is a fact of life that data we need to handle on an everyday basis is rarely complete. Handling incomplete information is one of the oldest topics in database research, we have a good theoretical understanding of what is needed and what it takes to produce correct results over incomplete databases. However most of those results have limited application due to their high complexity. The standard way of answering queries on incomplete databases is to compute certain answer: those that do not depend on the interpretation of unknown data.

Computing certain answers is coNP-hard for most reasonable semantics, however it is not yet a reason for undesirable behavior. Indeed one could expect from an evaluation over incomplete databases to either miss some certain answers, or returning some that are not, but not both at the same time. The problem with SQL is that it produces both kinds of errors as it have been shown recently in . The paper also offers a translation on relational algebra in order to ensure that only one kind of errors are produce.

The research problem

The translation on relational algebra in order to have correctness guarantee has proven to be efficient, however DBMS do not translate queries in relational algebra. Moreover relational algebra is based on sets semantic while SQL databases are based on bags. The question that naturally arises is can we find a direct translation from SQL to SQL which ensure correctness regarding bags semantic.

The aim of my internship is to propose a direct translation as efficient as possible which has correctness guarantee, and implement it.

Your contribution

The main idea is that order to compute a strict subset of certain answers, we also compute an over-approximation of those certain-answer due to the presence of negation in the queries. However computing such a set can be really expensive, so i tried to avoid it as much as possible and to make it fast when we had to compute it.

The contributions of my internship are the following: a semantic of SQL evaluation (section 2); the actual SQL to SQL translation (section 3); methods to help the planner while computing disjunctive queries answers (section 5); a Postgres extension implementing the translation.

Arguments supporting its validity

The translation is proven to have correctness guarantee on a semantic of SQL evaluation, however such semantic has to be prove right (code analysis due to lack of literature). Indeed work have to be done to .... however we did not find a counter example.

Benchmarks show that the various methods propose increase performances, and it's proven that those transformation does not change answers. Many of the methods can be applied to optimize more general query with only a few precautions.

Summary and future work

During my internship, i have proposed a direct translation from SQL to SQL with correctness guarantee, and i was able to implement a proof concept. More fundamental efforts to improve direct SQL translation will be to formalize SQL evaluation semantic. On a practical point of view, the way Postgres (or any other DBMS) compute disjunctive queries should be improve directly in the software (Hash Join with disjunction), and work should be done to ensure that the translated query behave well with parallelizationn. A very interesting ?next question?

Notes and Acknowledgements