Graph query optimization using semi-join rewritings

JELLE HELLINGS

Hasselt University, Martelarenlaan 42, 3500 Hasselt, Belgium

The graph data model is a versatile and natural data model for representing various sources of big data. To query such graph data, many graph query languages rely on graph navigation to select nodes of interest. In graph query languages such as XPath, GXPath, the (nested) regular path queries, and the calculus of relations, this navigation is provided by using *composition* ($\circ$). To see this, consider the following query:

$$\pi_1[\mathit{parentOf} \circ \mathit{parentOf} \circ \mathit{parentOf}] \circ \mathit{friendOf}.$$

This query returns pairs of great-grandparents and their friends, and illustrates how the composition operator expresses the intent of graph navigation in a simple and intuitive way. Especially in the setting of big data, this usage of compositions has a major drawback: evaluating compositions can be very costly. Luckily, the need for composition can be reduced by manually rewriting (parts of) queries using *semi-joins* instead. We can, for example, rewrite the above query as follows:

$$\pi_1[\mathit{parentOf} \ltimes (\mathit{parentOf} \ltimes \mathit{parentOf})] \rtimes \mathit{friendOf}.$$

Due to replacing compositions by semi-joins, straightforward evaluation of the resulting query is much faster than evaluation of the original query, even in the worst-case. This simple example does, however, illustrate the main drawback of introducing semi-joins: query writing becomes more complicated and the resultant queries are less intuitive and harder to understand.

As an alternative to such manually rewriting—from queries that use composition to queries that use semi-joins—we study techniques that recognize (parts of) queries that can be optimized in this way and enable their automatic optimization. Concretely, we study optimization of expressions in the calculus of relations augmented with transitive closure by eliminating composition and transitive closure, and instead use semi-joins and a restricted form of fixpoint iteration, respectively. Resulting is a set of sound term-based rewrite rules for the calculus of relations and these rewrite rules guarantee that the steps needed to evaluate the resulting optimized expression is linearly upper-bounded by the size of the original expression. For full optimization towards only using semi-joins and fixpoints, the rewrite rules provided place only minimal restrictions on the usage of composition, intersection, and difference, and these restrictions are shown to be necessary. Moreover, the rewriting is, in a sense, complete: every expression expressible by only using semi-joins and fixpoints is also expressible by a fully-optimizable expression in the calculus of relations.

To extend the applicability of our results, we are currently investigating ways to apply automatic semi-join rewriting in cases not covered by our results, this, for example, by using heuristics on the data. We are also looking to generalize our results to the setting of relational databases, relational algebra, and SQL.

E-mail address: `jelle.hellins@uhasselt.be`.

This material is based upon joint work with Catherine L. Pilachowski, Dirk Van Gucht, Marc Gyssens, and Yuqing Wu.