Heuristic and Cost-based Optimization for Provenance Computation

Xing Niu
Illinois Institute Of Technology
xniu7@hawk.iit.edu

Raghav Kapoor
Illinois Institute Of Technology
rkapoor7@hawk.iit.edu

Boris Glavic
Illinois Institute Of Technology
bglavic@iit.edu

Motivation. Data Provenance, information about the origin of data and the transformations used to produce it, has attracted attention by the database community in the recent years and has proven to be essential for a wide range of use cases including debugging of data and queries, auditing, and establishing authorship. The de-facto standard for database provenance is to model provenance as annotations on data and compute the provenance for the outputs of an operation by propagating these annotations. Provenance systems use query rewrite techniques to compile provenance computations into declarative queries (e.g., SQL [1]). This approach has the important advantage that standard relational databases can be used to compute provenance. However, these techniques generate queries with unusual access patterns and operator sequences. Even sophisticated databases are not capable of producing efficient plans for such queries. Thus, while query rewrite techniques enable easy implementation of provenance support for databases without the need to modify the database system itself, their performance is often far from optimal.

Our Approach. We address this problem though the development of novel heuristic and cost-based optimization techniques and their implementation in the GProM system [2]. We have developed a suite of heuristic rewrites including both well-known textbook equivalences as well as specific rules that are typically not explored by database optimizers. These rewrites help us avoid unnecessary computations in many cases and to restructure the query to make it digestible by database optimizers. To enable cost-based comparison between alternative options for rewriting queries, we have developed a novel extensible optimization algorithm that is agnostic to the plan space shape and uses the backend database for cost estimation. Our initial experimental evaluation confirms that these techniques are highly effective, resulting in a performance improvement of several orders of magnitude in some cases.

Poster Description. In this poster we will explain our approach for heuristic and cost-based optimization for provenance computation and its implementation in GProM. In particular, we will 1) showcase some of the unique heuristic rules that have been implemented in GProM; 2) explain our cost-based optimization algorithm that is plan-space agnostic and leverages the backend database's optimizer; 3) showcase some example code that illustrates how new optimization choices can be retrofitted into existing rewrite code by adding a few LOC; and 4) present preliminary experimental results which demonstrate the effectiveness of the approach for diverse provenance tasks ranging from traditional provenance computation for queries, over computing the provenance for updates and transactions [3], to computing game provenance.

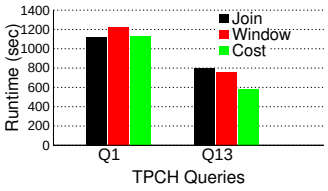

Fig. 1. Cost-based Optimization - TPCH Q1 + Q13 - 10GB

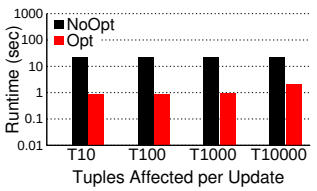

Fig. 2. Heuristic Optimization - Transaction Provenance

Experiments. In an initial experimental evaluation we have studied 1) the effectiveness of cost-based optimization in choosing the best alternative and 2) the effectiveness of heuristic optimization. Figures 1 and 2 show the results of two exemplary experiments. Figure 1 shows the performance of provenance computation for TPC-H benchmark queries 1 and 13 over a 10GB database instance. We compare two alternative methods of rewriting aggregations (one using joins and the other using SQL window functions) with a provenance computation generated by letting the cost-based optimizer decide which method to apply for each aggregation in a query. Notably, for query 13 which contains two aggregations, the optimizer has chosen the superior combination of both methods. Figure 2 shows results for computing the provenance of transactions with 10 updates each varying the number of affected tuples per update (from 10 to 10,000). All transactions were updating a relation with 1 million tuples. We compared the performance of provenance computation with and without heuristic optimizations. For the given workload, our heuristic optimizations result in a factor $\sim$10 speedup.

References

[1] B. Glavic and G. Alonso, "Perm: Processing provenance and data on the same data model through query rewriting," in *ICDE*, 2009, pp. 174–185.
[2] B. Arab, D. Gawlick, V. Radhakrishnan, H. Guo, and B. Glavic, "A generic provenance middleware for database queries, updates, and transactions," *TaPP*, 2014.
[3] B. Arab, D. Gawlick, V. Krishnaswamy, V. Radhakrishnan, and B. Glavic, "Reenacting transactions to compute their provenance," Illinois Institute of Technology, Tech. Rep., 2014.