Using SQL for Efficient Generation and Querying of Provenance Information

Boris Glavic¹, Renée J. Miller², and Gustavo Alonso³

¹ Illinois Institute of Technology
bglavic@iit.edu
² University of Toronto
miller@cs.toronto.edu
³ ETH Zurich
alonso@inf.ethz.ch

Abstract. In applications such as data warehousing or data exchange, the ability to efficiently generate and query provenance information is crucial to understand the origin of data. In this chapter, we review some of the main contributions of Perm, a DBMS that generates different types of provenance information for complex SQL queries (including nested and correlated subqueries and aggregation). The two key ideas behind Perm are representing data and its provenance together in a single relation and relying on query rewrites to generate this representation. Through this, Perm supports fully integrated, on-demand provenance generation and querying using SQL. Since Perm rewrites a query requesting provenance into a regular SQL query and generates easily optimizable SQL code, its performance greatly benefits from the query optimization techniques provided by the underlying DBMS.

1 Introduction

Peter Buneman was one of the first to recognize the importance of data provenance. With co-authors Khanna and Tan, he introduced the seminal models of Why- and Where-provenance [7]. Provenance, information about the creation process or the origin of data, can be used to debug queries and clean data in data warehouses, to understand and correct complex data integration transformations, for auditing, and to understand the value of data in curated databases. Provenance generation has also been used as a supporting technology for exchanging updates between heterogeneous databases [21], to provide access control based on the origin of data [31], and in modeling uncertainty in databases [35].

While provenance has many applications, these applications often place very high requirements on a provenance management system to be useful in practice. In this chapter, we overview the contributions of the Perm provenance management system [17]. Perm was designed as a scalable system for the generation and querying of provenance information over relational data. To understand the requirements for such a system, we begin with an example and then consider the foundations in provenance research on which Perm builds.
Example 1 (Running Example). The example database shown in Figure 1 stores credit card information: customers, their credit cards, purchases made with credit cards (Purchase), and from which external database (recorded in the company attribute) a batch of purchase tuples was imported, when and by whom (Imports). For convenience, we show an identifier for each tuple in the instance (e.g., p2). The query q shown in Figure 2 returns the months during which customers with at least two credit cards exceeded their credit limit on some card. To understand from which inputs of q the result tuple t2 (Joe, Feb) is derived, a user needs access to the data provenance of the query and the ability to query this information. For example, a user may be interested in knowing if some of these over-drafts are caused by suspiciously low credit card limits. This question can be answered by running a query over the provenance of q to retrieve tuples in the result of q that depend on credit card tuples with low limit values (i.e., these credit card tuples belong to the data provenance of the tuples to be returned). Alternatively, if the user realizes that some names are spelled incorrectly in the query result, she needs to understand where the name attribute values in the query result have been copied from to trace this error. This requires access to a different type of provenance that tracks the copying of information instead of which inputs caused a tuple to appear in the query result.

1.1 Requirements for Provenance Systems

The example and discussion above motivates four requirements for relational provenance systems. (Requirement 1) Support different types of provenance with sound semantics. Information from different provenance types is often needed to best understand the data and how it has been transformed. We would consider a provenance type to have sound semantics, if it provably captures our intuitive understanding of provenance. For example, the provenance of a query...