View Maintenance Using Conditional Tables*

Hua Shu

Institut für Informatik
Universität Hannover
Lange Laube 22, D-30159 Hannover, Germany
hs@informatik.uni-hannover.de

Abstract. This paper presents a new approach to maintaining materialized views without accessing the underlying base relations. Views can be made self-maintainable using additional data together with the views. For instance, one can replicate auxiliary views of the base relations at the site where the views are materialized to ensure self-maintenance of the views. However, the previous approaches often lead to the replication of the entire base relations, which is not acceptable from the data protection point of view. We propose to represent the base data using tables with variables and to materialize auxiliary views of such tables in form of conditional tables. Modeling updates of the base data as changes of the assignment to the variables, we can compute the updated views by evaluating the conditional tables with respect to the new assignment of the variables. Our approach avoids the replication of the base data and allows active self-maintenance of views triggered by identified updates of the base data.

1 Introduction

Views are versions of data that are restructured and possibly restricted images of a database. Materialized views are physical copies of views that are stored and maintained. The view maintenance problem is about how to efficiently update a view that is materialized in response to updates of the base relations. Suppose that there is a view defined by a relational algebra expression \( q \) over a database schema. A state of the view with respect to a set \( I \) of base relations is denoted by \( q(I) \). Consider an update operation \( \mu \) against \( I \). Let \( I' \) denote the updated state of the base relations. The simplest way to update the view is to compute \( q(I') \) from scratch. Sometimes it is more efficient to incrementally maintain the view, i.e. to compute only the changes in the view [6].

We are particularly concerned with the situations where the access to the base data may be slow, expensive or even periodically unavailable, and it is desirable to be able to incrementally maintain views without additional queries over the base data. Views that can be maintained only based on information about the

* This work was supported by Swedish Research Council for Engineering Sciences (TFR) under grant 282-95-966.
views and the updates against the base data, without additional queries over the base data, are said to be self-maintainable [4].

Self-maintainability of views has been recognized as one of the main optimization problems in, for instance, data warehouses. At a sufficiently abstract level, a data warehouse can be seen as a collection of materialized views over base data residing at external information sources. An important task of warehouse management is to perform materialized view maintenance [10, 12]. In order to integrate the change of the base data into the warehouse, it may be needed to fetch additional data from the external sources. Issuing such queries to the sources may lead to a processing delay. The queries can be expensive or may not be permitted at all for security reasons. Thus it is often required to minimize the additional queries to the external sources.

Previous studies have mainly been focus on identifying the class of self-maintainable views. It has been shown that only a very restricted subclass of SPJ views is self-maintainable [11, 3, 5, 6, 10, 12]. Views involving joins, for instance, are generally not self-maintainable in response to insertions into a component relation, and are self-maintainable in response to deletions and modifications only under certain conditions [3, 4].

Suppose that views are stored at sites different from where the base relations are stored. A site where base relations are stored is called base-relation site. A site where views are stored is called view site. A simple way to achieve self-maintenance of views is to replicate (a subset of) the base relations at view site. However, the cost of replicating large base relations may become prohibitive. Moreover, the replication may not be acceptable for applications where the very purpose of defining views is data protection, i.e. to limit the access to the entire base relations.

Another approach is to materialize auxiliary views of the base relations at view sites [7, 10]. In [7], views are made self-maintainable by storing, at the view sites, the results of pushing down selections and projections to the base relations. The cost of storing the results is usually lower than the cost of replicating the base relations in their entirety. For example, consider a view defined by $\pi_{r_1, r_2, s_1, s_2}(\sigma_{E_1} R \land_{r_2=s_1} \sigma_{E_2} S)$, where $r_1, r_2, r_3$ are some attributes of $R$, $s_1, s_2$ are some attributes of $S$ and $E_1, E_2$ are selection conditions. The idea is to materialize, at view site, the auxiliary views $\pi_{r_1, r_2, s_1, s_2}(\sigma_{E_1} R \land_{r_2=s_1} \sigma_{E_2} S)$ of the base relations $R$ and $S$. These views are smaller than the base relations. Based on the materialized, auxiliary views, the view $\pi_{r_1, r_3, s_1, s_2}(\sigma_{E_1} R \land_{r_2=s_1} \sigma_{E_2} S)$ can be computed without access to the base relations. However, when no selections and projections can be pushed down to the base relations, this approach degenerates into the replication of the entire base relations.

The updates of the base relations that are relevant to a view are sometimes known to be restricted to certain part of the base relations. For instance, in a personnel database, it may be that only the records of the temporarily employed or guests can be removed from the database, and only the fields of salaries are frequently modified. It is then desirable to have some active mechanism of view self-maintenance triggered by such expected updates.