Materialized View Selection for Data Warehouse Design

Dimitri Theodoratos
New Jersey Institute of Technology, USA

Alkis Simitsis
National Technical University of Athens, Greece

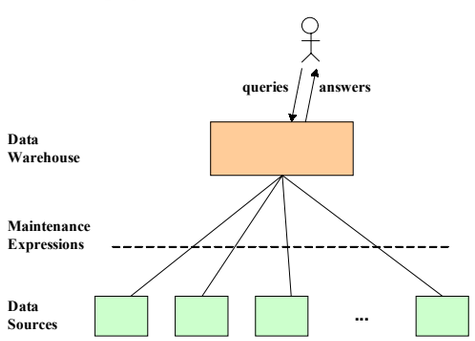
INTRODUCTION

A data warehouse (DW) is a repository of information retrieved from multiple, possibly heterogeneous, autonomous, distributed databases and other information sources for the purpose of complex querying, analysis, and decision support. Data in the DW are selectively collected from the sources, processed in order to resolve inconsistencies, and integrated in advance (at design time) before data loading. DW data are usually organized multi-dimensionally to support online analytical processing (OLAP). A DW can be seen abstractly as a set of materialized views defined over the source relations. During the initial design of a DW, the DW designer faces the problem of deciding which views to materialize in the DW. This problem has been addressed in the literature for different classes of queries and views, and with different design goals.

BACKGROUND

Figure 1 shows a simplified DW architecture. The DW contains a set of materialized views. The users address their queries to the DW. The materialized views are used partially or completely for the evaluation of the user queries. This is achieved through partial or complete rewritings of the queries using the materialized views.

Figure 1. A simplified DW architecture



When the source relations change, the materialized views need to be updated. The materialized views are usually maintained using an incremental strategy. In such a strategy, the changes to the source relations are propagated to the DW. The changes to the materialized views are computed using the changes of the source relations and are eventually applied to the materialized views. The expressions used to compute the view changes involve the changes of the source relations and are called maintenance expressions. Maintenance expressions are issued by the DW against the data sources, and the answers are sent back to the DW. When the source relation changes affect more than one materialized view, multiple maintenance expressions need to be evaluated. The techniques of multi-query optimization can be used to detect common subexpressions among maintenance expressions in order to derive an efficient global evaluation plan for all the maintenance expressions.

MAIN THRUST

When selecting views to materialize in a DW, one attempts to satisfy one or more design goals. A design goal is either the minimization of a cost function or a constraint. A constraint can be classified as user-oriented or system-oriented. Attempting to satisfy the constraints can result in no feasible solution to the view selection problem. The design goals determine the design of the algorithms that select views to materialize from the space of alternative view sets.

Minimization of Cost Functions

Most approaches comprise in their design goals the minimization of a cost function.

- **Query Evaluation Cost:** Often, the queries that the DW has to satisfy are given as input to the view selection problem. The overall query evaluation cost is the sum of the cost of evaluating each input

query rewritten (partially or completely) over the materialized views. This sum also can be weighted, each weight indicating the frequency or importance of the corresponding query. Several approaches aim at minimizing the query evaluation cost (Gupta & Mumick, 1999; Harinarayan et al., 1996; Shukla et al., 1998).

- **View Maintenance Cost:** The view maintenance cost is the sum of the cost of propagating each source relation change to the materialized views. This sum can be weighted, each weight indicating the frequency of propagation of the changes of the corresponding source relation. The maintenance expressions can be evaluated more efficiently if they can be partially rewritten over views already materialized at the DW; the evaluation of parts of the maintenance expression is avoided since their materializations are present at the DW. Moreover, access of the remote data sources and expensive data transmissions are reduced. Materialized views that are added to the DW for reducing the view maintenance cost are called *auxiliary views* (Ross et al., 1996; Theodoratos & Sellis, 1999). Obviously, maintaining the auxiliary views incurs additional maintenance cost. However, if this cost is less than the reduction to the maintenance cost of the initially materialized views, it is worth keeping the auxiliary views in the DW. Ross, et al. (1996) derive auxiliary views to materialize in order to minimize the view maintenance cost.
- **Operational Cost:** Minimizing the query evaluation cost and the view maintenance cost are conflicting requirements. Low view maintenance cost can be obtained by replicating source relations at the DW. In this case, though, the query evaluation cost is high, since queries need to be computed from the replicas of the source relations. Low query evaluation cost can be obtained by materializing at the DW all the input queries. In this case, all the input queries can be answered by a simple lookup, but the view maintenance cost is high, since complex maintenance expressions over the source relations need to be computed. The input queries may overlap; that is, they may share many common subexpressions. By materializing common subexpressions and other views over the source relations, it is possible, in general, to reduce the view maintenance cost. These savings must be balanced against higher query evaluation cost. For this reason, one can choose to minimize a linear combination of the query evaluation and view maintenance cost, which is called *operational cost*. Most approaches endeavor to minimize the operational cost (Baralis et al., 1997; Gupta, 1997; Theodoratos & Sellis, 1999; Yang et al., 1997).

System-Oriented Constraints

System-oriented constraints are dictated by the restrictions of the system and are transparent to the users.

- **Space Constraint:** Although the degradation of the cost of disk space allows for massive storage of data, one cannot consider that the disk space is unlimited. The space constraint restricts the space occupied by the selected materialized views not to exceed the space allocated to the DW for this end. Space constraints are adopted in many works (Gupta, 1997; Golfarelli & Rizzi, 2000; Harinarayan et al., 1996, Theodoratos & Sellis, 1999).
- **View Maintenance Cost Constraint:** In many practical cases, the refraining factor in materializing all the views in the DW is not the space constraint but the view maintenance cost. Usually, DWs are updated periodically (e.g., at nighttime) in a large batch update transaction. Therefore, the update window must be sufficiently short so that the DW is available for querying and analysis during the daytime. The view maintenance cost constraint states that the total view maintenance cost should be less than a given amount of view maintenance time. Gupta and Mumick (1999), Golfareli and Rizzi (2000), and Lee and Hammer (2001) consider a view maintenance cost constraint in selecting materialized views.
- **Self Maintainability:** A materialized view is self-maintainable if it can be maintained for any instance of the source relations over which it is defined and for all source relation changes, using only these changes, the view definition, and the view materialization. The notion is extended to a set of views in a straightforward manner. By adding auxiliary views to a set of materialized views, one can make the whole view set self-maintainable. There are different reasons for making a view set self-maintainable: (a) the remote source relations need not be contacted in turn for evaluating maintenance expressions during view updating; (b) anomalies due to concurrent changes are eliminated, and the view maintenance process is simplified; (c) the materialized views can be maintained efficiently even if the sources are not able to answer queries (e.g., legacy systems), or if they are temporarily unavailable (e.g., in mobile systems). By adding auxiliary views to a set of materialized views, the whole view set can be made self-maintainable. Self-maintainability can be trivially achieved by replicating at the DW all the source relations used in the view definitions. Self-maintainability viewed as a constraint requires that the set of materialized views taken together is self-maintainable. Quass, et al., (1996), Akinde, et al.,