Chapter 7

View Selection and Materialization

Zohra Bellahsene
LIRMM-CNRS/Université Montpellier 2, France

ABSTRACT

There are many motivations for investigating the view selection problem. At first, materialized views are increasingly being supported by commercial database systems and are used to speed up query response time. Therefore, the problem of choosing an appropriate set of views to materialize in the database is crucial in order to improve query processing cost. Another application of the view selection issue is selecting views to materialize in data warehousing systems to answer decision support queries. The problem addressed in this paper is similar to that of deciding which views to materialize in data warehousing. However, most existing view selection methods are static. Moreover, none of these methods have considered the problem of de-materializing the already materialized views. Yet it is a very important issue since the size of storage space is usually restricted. This chapter deals with the problem of dynamic view selection and with the pending issue of removing materialized views in order to replace less beneficial views with more beneficial ones. We propose a view selection method for deciding which views to materialize according to statistic metadata. More precisely, we have designed and implemented our view selection method, including a polynomial algorithm, to decide which views to materialize.

INTRODUCTION

Nowadays, materialized views are increasingly being supported by a variety of commercial DBMS to speed up query response time. This technique is also very useful in data warehousing for optimizing OLAP queries. In such systems, data are extracted in advance and stored in a repository. Then, user queries are addressed directly to the data warehouse system and processed without needing to access the data sources. At an abstract level, a data warehouse can be seen as a set of materialized views. Furthermore, new applications of the problem of view selection arise namely in data placement in
distributed databases and in peer to peer data sharing systems.

The problem addressed in this paper is similar to that of deciding which views to materialize in data warehousing. However, most existing view selection methods are static. Moreover, none of these methods have considered the problem of dematerializing the already materialized views. Yet it is a very important issue since the size of storage space is usually restricted.

Many database systems support creation and use of materialized views. The presence of appropriate materialized views can significantly improve performance and speed up the processing of queries by several orders of magnitude. For this reason the problem of view selection has received significant attention in recent literature. The majority of these works (Yang, 1997; Kotidis, 1999; Theodoratos, 2001; Baril, 2003) presents the solution for data warehousing environments that are used for On-Line Analytical Processing (OLAP) and Decision Support System applications. The problem in this context is the following: given a database scheme R, a storage space B, and a workload of queries Q, choose a set of views V over R to materialize, whose combined size is at most B. The goal of view selection process is to find a set of views that minimizes the expected cost of evaluating the queries in Q. Traditionally, view selection has been carried out statically. With static view selection, a system administrator decides what kinds of queries might be carried out in the system. Several models and tools have been designed to take the expected query workload and choose set of views to materialize; e.g., (Yang, 1997; Gupta, 1999; Baril, 2003). Obviously, static selection of views has several weaknesses: (i) the query workload is often not predictable; (ii) even if the workload can be predicted, the workload is likely to change, and the workload might change so quickly that the system administrator cannot adjust the view selection quickly enough so the static view selection might very quickly become outdated. This means that the administrator should monitor the query pattern and periodically “recalibrate” the materialized views by rerunning these algorithms. This task for a large warehouse where many users with different profiles submit their queries is rather complicated and time consuming.

Once the views are selected and are materialized, another problem arises. Each time a base table is changed, the materialized views and indexes built on it have to be updated (or at least have to be checked whether some changes have to be propagated or not). The problem of updating the views is known as the view maintenance problem. In most cases it is wasteful to maintain a view by re-computing it from scratch. Often, it is cheaper to use the heuristic of inertia (only a part of the view changes in response to changes in the base relations) and thus compute only the changes in the view to update its materialization (Gupta, 1995). This technique is called incremental view maintenance. However, not always the incremental maintenance is a right choice. For example, if an entire base relation is deleted, it may be cheaper to re-compute a view that depends on the deleted relation (if the new view will quickly evaluate to an empty relation) than to compute the changes to the view. The view maintenance cost constraint is very important for the view selection problem and cannot be ignored. Another kind of view maintenance, called view adaptation is the one occurring after schema evolution or after direct view query changes. The problem is that of propagating the schema change arising at the data sources or on the view schema to the materialized views (Belbahsene, 2004).

This chapter deals with the problem of dynamic view selection and with the pending issue of removing materialized views in order to replace less beneficial views with more beneficial ones. We propose a view selection method for deciding which views to materialize according to statistic metadata.
Related Content

Automatic Music Timbre Indexing
[www.igi-global.com/chapter/automatic-music-timbre-indexing/10809?camid=4v1a](www.igi-global.com/chapter/automatic-music-timbre-indexing/10809?camid=4v1a)

Traversal Pattern Mining in Web Usage Data
[www.igi-global.com/chapter/traversal-pattern-mining-web-usage/7745?camid=4v1a](www.igi-global.com/chapter/traversal-pattern-mining-web-usage/7745?camid=4v1a)

A Multidimensional Pattern Based Approach for the Design of Data Marts
[www.igi-global.com/chapter/multidimensional-pattern-based-approach-design/28167?camid=4v1a](www.igi-global.com/chapter/multidimensional-pattern-based-approach-design/28167?camid=4v1a)

Discretization for Data Mining
[www.igi-global.com/chapter/discretization-data-mining/10629?camid=4v1a](www.igi-global.com/chapter/discretization-data-mining/10629?camid=4v1a)