Chapter LXXIV
Index and Materialized View Selection in Data Warehouses

Kamel Aouiche
Université de Québec à Montréal, Canada

Jérôme Darmont
University of Lyon (ERIC Lyon 2), France

INTRODUCTION

Database management systems (DBMSs) require an administrator whose principal tasks are data management, both at the logical and physical levels, as well as performance optimization. With the wide development of databases and data warehouses, minimizing the administration function is crucial. This function includes the selection of suitable physical structures to improve system performance.

View materialization and indexing are presumably some of the most effective optimization techniques adopted in relational implementations of data warehouses. Materialized views are physical structures that improve data access time by precomputing intermediary results. Therefore, end-user queries can be efficiently processed through data stored in views and do not need to access the original data. Indexes are also physical structures that allow direct data access. They avoid sequential scans and thereby reduce query response time. Nevertheless, these solutions require additional storage space and entail maintenance overhead. The issue is then to select an appropriate configuration of materialized views and indexes that minimizes both query response time and maintenance cost given a limited storage space. This problem is NP hard (Gupta & Mumick, 2005).

The aim of this article is to present an overview of the major families of state-of-the-art index and materialized view selection methods, and to discuss the issues and future trends in data warehouse performance optimization. We particularly focus on data-mining-based heuristics we developed to reduce the selection problem complexity and target the most pertinent candidate indexes and materialized views.

BACKGROUND

Today's commercial relational DBMSs provide integrated tools for automatic physical design. For a given workload, they automatically recom-

mend configurations of indexes and materialized views (Dageville, Das, Dias, Yagoub, Zaït, & Ziauddin, 2004) coupled with data partitioning (Agrawal, Chaudhuri, Kollàr, Marathe, Narasayya, & Syamala, 2004) or table clustering (Zilio et al., 2004). However, these tools depend on the query optimizer and therefore the host DBMS, which renders their adaptation onto other systems intricate. In the remainder of this section, we detail published research about index and materialized view selection.

Index Selection Problem

The index selection problem has been studied for many years in databases (Chaudhuri, Datar, & Narasayya, 2004; Finkelstein, Schkolnick, & Tiberio, 1988), but adaptations to data warehouses are few. In this particular context, research studies may be clustered into two families: algorithms that optimize maintenance cost (Labio, Quass, & Adelberg, 1997) and algorithms that optimize query response time. In both cases, optimization is realized under the storage space constraint. We particularly focus on the second family of approaches, which may be classified depending on how the set of candidate indexes and the final configuration of indexes are built.

The set of candidate indexes may be built manually by the administrator according to his or her expertise of the workload (Choenni, Blanken, & Chang, 1993a, 1993b; Frank, Omiecinski, & Navathe, 1992). This is both subjective and quite hard to achieve when the number of queries is large. In opposition, candidate indexes may also be extracted automatically through a syntactic analysis of the workload (Chaudhuri & Narasayya, 1997; Golfarelli, Rizzi, & Saltarelli, 2002; Valentin, Zuliani, Zilio, Lohman, & Skelley, 2000).

There are several methods for building the final index configuration from the candidate indexes. Ascending methods start from an empty set of indexes (Chaudhuri & Narasayya, 1997; Choenni et al., 1993b; Frank et al., 1992; Kyu-Young, 1987). They increasingly select indexes minimizing workload cost until it does not decrease anymore. Descending methods start with the whole set of candidate indexes and prune indexes until workload cost increases (Choenni et al., 1993a; Kyu-Young, 1987). Classical optimization algorithms have also been used to solve this problem, such as knapsack resolution (Feldman & Reouven, 2003; Gündem, 1999; Ip, Saxton, & Raghavan, 1983; Valentin et al., 2000) and genetic algorithms (Kratika, Ljubic, & Tosic, 2003).

Materialized View Selection Problem

The classical papers about materialized view selection in data warehouses introduce a lattice framework that models and captures ancestor-descendent dependency among aggregate views in a multidimensional context (Baralis, Paraboschi, & Teniente, 1997; Harinarayan, Rajaraman, & Ullman, 1996; Kotidis & Roussopoulos, 1999; Uchiyama, Runapongsa, & Teorey, 1999). This lattice is greedily browsed with the help of cost models to select the best views to materialize. This problem has first been addressed in one data cube, and then extended to multiple cubes (Shukla, Deshpande, & Naughton, 2000). Another theoretical framework, called the and-or view graph, may also be used to capture the relationships between materialized views (Chan, Li, & Feng, 1999; Gupta & Mumick, 2005; Theodoratos & Bouzeghoub, 2000; Valluri, Vadapalli, & Karlapalem, 2002). However, the majority of these solutions are theoretical and not truly scalable.

Another method decomposes data cubes into an indexed hierarchy of wavelet view elements and selects those that minimize the average processing cost of the queries defined on the data cubes (Smith, Li, & Jhingran, 2004). Similarly, the dwarf structure (Sismanis, Deligiannakis, Roussopoulos, & Kotidis, 2002) compresses data cubes, thereby suppressing redundancy to improve maintenance and interrogation costs. These approaches are very interesting, but they mainly focus on computing efficient data cubes by changing their physical design, which is not always convenient in practice.