Referential Horizontal Partitioning Selection Problem in Data Warehouses: Hardness Study and Selection Algorithms

Ladjel Bellatreche, University of Poitiers, France
Kamel Boukhalfa, University of Poitiers, France
Pascal Richard, University of Poitiers, France
Komla Yamavo Woameno, University of Poitiers, France

ABSTRACT

Horizontal Partitioning has been largely adopted by the database community, where it took a significant part in the physical design process. Actually, it is supported by most commercial database systems (DBMS), where a native Data Definition Language for decomposing tables/materialized views using various modes is proposed. In traditional databases, horizontal partitioning has been largely studied, where several fragmentation algorithms were proposed to partition tables in isolation. In the relational data warehouse environment, horizontal partitioning consists in decomposing the whole warehouse schema into sub schemas, where each schema contains fragments of dimension and fact tables. Dimension tables are fragmented using the primary partitioning mode, whereas the fact table is divided using referential mode. In this article, the authors first focus on the evolution of horizontal partitioning in commercial DBMS motivated by decision support applications. Secondly, they give a formalization of the referential fragmentation schema selection problem in the data warehouse and they study its hardness to select an optimal solution. Due to its high complexity, they develop two algorithms: hill climbing and simulated annealing with several variants to select a near optimal partitioning schema. Finally, extensive experimental studies are conducted using the data set of APB1 benchmark to compare the quality the proposed algorithms using a mathematical cost model. Based on these experiments, some recommendations are given to advise database administrator for well using horizontal partitioning.

Keywords: Fragmentation Schema Selection Algorithm, Partitioning, Referential Horizontal Partitioning, Primary Star Join Query Optimization

INTRODUCTION

Data warehouse applications store large amounts of data in various tables, where each instance is described using several attributes. Usually, these applications are modeled with relational database schemas like star or snowflake schemas. A star schema contains a large fact table and various dimension tables. A star schema is usually queried in various combina-
tions involving many tables. The most used operations are joins, aggregations and selections (Papadomanolakis & Ailamaki, 2004). Joins are well known to be expensive operations, especially when the involved relations are substantially larger than the size of the main memory (Lei & Ross, 1999) which is usually the case of business intelligence applications. The typical queries defined on the star schema are commonly referred to as star join queries, and exhibit the following two characteristics: (1) there is a multi-table join among the large fact table and multiple smaller dimension tables, and (2) each of the dimension tables involved in the join operation has multiple selection predicates on its descriptive attributes. To speed up star join queries, many optimization techniques were proposed. In (Bellatreche, Moussaoui, Necir & Drias, 2008), we classified them into two main categories: redundant techniques such as materialized views, advanced index schemes, vertical partitioning, parallel processing with replication and non redundant techniques like ad-hoc joins, where joins are performed without additional data structures like indexes (hash join, nested loop, sort merge, etc.), horizontal partitioning, parallel processing without replication. In this article, we concentrate only on horizontal partitioning, since it is more adapted to reduce the cost of star join queries.

Horizontal partitioning has been mainly used in logical distributed and parallel databases design in last decades (Özsu & Valduriez, 1999; Saecia & Wiederhold, 1985). Recently, it becomes a crucial part of physical database design (Sanjay, Narasayya & Yang, 2004; Papadomanolakis & Ailamaki, 2004; Eadon et al., 2008), where most of today’s commercial DBMS offer native DDL (data definition language) support for defining horizontal partitions (fragments) of a table (Sanjay et al., 2004). In context of relational warehouses, horizontal partitioning allows tables, indexes and materialized views to be partitioned into disjoint sets of rows that are physically stored and accessed separately. Contrary to materialized views and indexes, horizontal data partitioning does not replicate data, thereby reducing space requirements and minimizing the update overhead. The main characteristic of data partitioning is its ability to be combined with some redundant optimization techniques such as indexes and materialized views. It also affect positively (1) query performance, by performing partition elimination: if a query includes a partition key as a predicate in the WHERE clause, the query optimizer will automatically route the query to only relevant partitions and (2) database manageability, for instance by allocating partitions in different machines (Stöhr, Märtens & Rahm, 2000).

Two versions of horizontal partitioning exist (Ceri, Negri & Pelagatti, 1982; Özsu & Valduriez, 1999): primary and derived (known as referential partitioning in Oracle 11g (Eadon et al., 2008)). Primary horizontal partitioning of a table is performed using attributes defined on that table. Derived horizontal partitioning, on the other hand, is the fragmentation of a table using attribute(s) defined on another table(s). In other word, the derived horizontal partitioning of a table is based on the fragmentation schema of another table (the fragmentation schema is the result of the partitioning process of a given table). The derived partitioning of a table $R$ according a fragmentation schema of $S$ is feasible if and only if there is a join link between $R$ and $S$ ($R$ contains a foreigner key of $S$). It has been used to optimize data transfer when executing queries in the distributed database environment.

In traditional databases (relational and object oriented), tables/classes are horizontally partitioned in isolation. Consequently, the problem of selecting horizontal fragments of a given table $T$ of a database may be formulated as follows:

Given a representative workload $W$ defined on the table $T$, find a partitioning schema $FS$ of $T$, such that the overall query processing cost

$$
\left( \sum_{Q \in W} f_Q \times \text{Cost}(Q, FS) \right),
$$

where $f_Q$ represents the access frequency of the query $Q$ is minimized. According this formal-
Related Content

White Patch Detection in Brain MRI Image Using Evolutionary Clustering Algorithm
[www.igi-global.com/chapter/white-patch-detection-in-brain-mri-image-using-evolutionary-clustering-algorithm/139410?camid=4v1a](www.igi-global.com/chapter/white-patch-detection-in-brain-mri-image-using-evolutionary-clustering-algorithm/139410?camid=4v1a)

A Cached-Based Stream-Relation Join Operator for Semi-Stream Data Processing
A Methodology for Building XML Data Warehouses
www.igi-global.com/article/methodology-building-xml-data-warehouses/1750?camid=4v1a

Online Prediction of Blood Glucose Levels using Genetic Algorithm
Khaled Eskaf, Tim Ritchings and Osama Bedawy (2014). *Biologically-Inspired Techniques for Knowledge Discovery and Data Mining* (pp. 299-310).
www.igi-global.com/chapter/online-prediction-of-blood-glucose-levels-using-genetic-algorithm/110465?camid=4v1a