A Genetic Algorithm for Selecting Horizontal Fragments

Ladjel Bellatreche
Poitiers University, France

INTRODUCTION

Decision support applications require complex queries, e.g., multi way joins defining on huge warehouses usually modelled using star schemas, i.e., a fact table and a set of data dimensions (Papadomanolakis & Ailamaki, 2004). Star schemas have an important property in terms of join operations between dimensions tables and the fact table (i.e., the fact table contains foreign keys for each dimension). None join operations between dimension tables. Joins in data warehouses (called star join queries) are particularly expensive because the fact table (the largest table in the warehouse by far) participates in every join and multiple dimensions are likely to participate in each join.

To speed up star join queries, many optimization structures were proposed: redundant structures (materialized views and advanced index schemes) and non redundant structures (data partitioning and parallel processing). Recently, data partitioning is known as an important aspect of physical database design (Sanjay, Narasayya & Yang, 2004; Papadomanolakis & Ailamaki, 2004). Two types of data partitioning are available (Özsu & Valduriez, 1999): vertical and horizontal partitioning. Vertical partitioning allows tables to be decomposed into disjoint sets of columns. Horizontal partitioning allows tables, materialized views and indexes to be partitioned into disjoint sets of rows that are physically stored and usually accessed separately. Contrary to redundant structures, data partitioning does not replicate data, thereby reducing storage requirement and minimizing maintenance overhead. In this paper, we concentrate only on horizontal data partitioning (HP).

HP may 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 or by splitting any access paths: tables, materialized views, indexes, etc. Most of database systems allow three methods to perform the HP using PARTITION statement: RANGE, HASH and LIST (Sanjay, Narasayya & Yang, 2004). In the range partitioning, an access path (table, view, and index) is split according to a range of values of a given set of columns. The hash mode decomposes the data according to a hash function (provided by the system) applied to the values of the partitioning columns. The list partitioning splits a table according to the listed values of a column. These methods can be combined to generate composite partitioning. Oracle currently supports range-hash and range-list composite partitioning using PARTITION - SUBPARTITION statement. The following SQL statement shows an example of fragmenting a table Student using range partitioning.

CREATE TABLE student (Student_ID Number(6), Student_FName VARCHAR(25), Student-LName VARCHAR(25), PRIMARY KEY (Student_ID) PARTITION BY *RANGE* (student_LN) (PARTITION student_ae VALUES LESS THAN ('F%') TABLESPACE part1, PARTITION student_fl VALUES LESS THAN ('M%') TABLESPACE part2, PARTITION student_mr VALUES LESS THAN ('S%') TABLESPACE part3, PARTITION student_sz VALUES LESS THAN (MAXVALUE) TABLESPACE part4);

HP can also be combined with others optimization structures like indexes, materialized views, and parallel processing. Several work and commercial systems show its utility and impact in optimizing OLAP queries (Noaman & Barker, 1999, Sanjay, Narasayya & Yang, 2004; Stöhr, Märtens & Rahm, 2000). But none of these studies has formalized the problem of selecting a horizontal partitioning schema to speed up a set of queries as optimization problem with constraint and proposed selection algorithms.

Logically, two types of HP are distinguished (Ceri, Negri & Pelagatti, 1982; Özsu & Valduriez, 1999): (1) *primary HP* and (2) *derived HP*. The primary HP consists in fragmenting a table T based only on its attribute(s). The derived HP consists in splitting a table S (e.g., the fact table of a given star schema) based on fragmentation schemas of other tables (e.g., dimension tables). This type has been used in (Stöhr, Märtens & Rahm, 2000). The primary HP may be used in optimizing selection operations, while the second in optimizing join and selection operations since it pre-computes them.

BACKGROUND

The HP in relational data warehouses is more challenging compared to that in relational and object databases. Several work and commercial systems show its utility and impact in optimizing OLAP queries (Sanjay, Narasayya & Yang, 2004; Stöhr, Märtens & Rahm,2000). But none study has formalized the problem of selecting a horizontal partitioning schema to speed up a set of queries as an optimization problem and proposed selection algorithms.

This challenge is due to the several choices of partitioning schema (a fragmentation schema is the result of the data partitioning process) of a star or snowflake schemas:

1. Partition only the dimension tables using simple predicates defined on these tables (a simple predicate p is defined by: p: A_i θ Value; where A_i is an attribute, θ ∈{=, <, >, ≥, ≤}, and Value ∈ Domain(A_i)). This scenario is not suitable for OLAP queries, because the sizes of dimension tables are generally small compare to the fact table. Most of OLAP queries access the fact table, which is huge Therefore, any partitioning that does not take into account the fact table is discarded.
2. Partition only the fact table using simple predicates defined on this table because it normally contains millions of rows and is highly normalized. The fact relation stores time-series factual data. The fact table is composed of foreign keys and raw data. Each foreign key references a primary key on one of the dimension relations. These dimension relations could be time, product, customer, etc. The raw data represent the numerical measurement of the organization such as sales amount, number of units sold, prices and so forth. The raw data usually never contain descriptive (textual) attributes because the fact relation is designed to perform arithmetic operations such as summarization, aggregation, average and so forth on such data. In a data warehouse modelled by a star schema, most of OLAP queries access dimension tables first and after that to the fact table. This choice is also discarded.
3. Partition some/all dimension tables using their predicates, and then partition the fact table based on the fragmentation schemas of dimension tables. This approach is best in applying partitioning in data warehouses. Because it takes into consideration the star join queries requirements and the relationship between the fact table and dimension tables. In our study, we recommend the use of this scenario.

Horizontal Partitioning Selection Problem

Suppose a relational warehouse modelled by a star schema with d dimension tables and a fact table F. Among these dimension tables, we consider that g tables are fragmented (g ≤ d), where each table D_i (1 ≤ i ≤ g) is partitioned into m_i fragments: {D_{i1}, D_{i2}, …, D_{imi}}, such as: $D_{ij} = \sigma_{cl_{ji}}(D_i)$, where cl_{ji} and σ represent a conjunction of simple predicates and the selection predicate, respectively. Thus, the fragmentation schema of the fact table F is defined as follows:

$F_i = F \lceil D_{1i} \lceil D_{2i} \lceil \ldots \lceil D_{gi}$, $(1 \le i \le m_i)$, where $\lceil$ represents the semi join operation.

Example

To illustrate the procedure of fragmenting the fact table based on the fragmentation schemas of dimension tables, let's consider a star schema with three dimension tables: Customer, Time and Product and one fact table Sales. Suppose that the dimension table Customer is fragmented into two fragments *CustFemale* and *CustMale* defined by the following clauses: