Chapter XXIII
Horizontal Data Partitioning: Past, Present and Future

Ladjel Bellatreche
LISI/ENSMA - University of Poitiers, France

INTRODUCTION

Horizontal data partitioning is the process of splitting access objects into set of disjoint rows. It was first introduced in the end of 70's and beginning of the 80's (Ceri et al., 1982) for logically designing databases in order to improve the query performance by eliminating unnecessary accesses to non-relevant data. It knew a large success (in the beginning of the 80's) in designing homogeneous distributed databases (Ceri et al., 1982; Ceri et al., 1984; Özsu et al., 1999) and parallel databases (DeWitt et al., 1992; Valduriez, 1993). In distributed environment, horizontal partitioning decomposes global tables into horizontal fragments, where each partition may be spread over multiple nodes. End users at the node can perform local queries/transactions on the partition *transparently* (the fragmentation of data across multiple sites/processors is not visible to the users.). This increases performance for sites that have regular transactions involving certain views of data, whilst maintaining availability and security. In parallel database context (Rao et al., 2002), horizontal partitioning has been used in order to speed up query performance in a shared-nothing parallel database system (DeWitt et al., 1992). This will be done by both intra-query and intra-query parallelisms (Valduriez, 1993). It also facilitates the exploitation of the inputs/outputs bandwidth of the disks by reading and writing data in parallel. In this paper, we use fragmentation and partitioning words interchangeably.

There are two versions of horizontal partitioning (Özsu et al., 1999): *primary* and *derived*. Primary horizontal partitioning of a relation is performed using selection predicates that are defined on that relation. Note that a simple predicate (selection predicate) has the following form: A θ value, where A is an attribute of the table to be fragmented, θ is one of six comparison operators $\{=, <, >, \leq, \geq\}$ and value is the predicate constant

belonging to the domain of the attribute A. *Derived horizontal partitioning*, on the other hand, is the fragmentation of a relation that results from predicates being defined on another relation. 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. The relation at the link is called the owner of the link (the case of the table S) and the relation at the head is called member (Ceri et al., 1982) (the case of the relation R).

In the context of data warehousing, the horizontal partitioning becomes an important aspect of physical design. Note that in relational data warehouses, two types of tables exist: dimension tables and fact table. A dimension table is a table containing the data for one dimension within a warehouse schema. The primary key is used to link to the fact table, and each level in the dimension has a corresponding field in the dimension table. The fact table is the central table in a star schema, containing the basic facts or measures of interest. Dimension fields are also included (as foreign keys) to link to each dimension table. In this context, horizontal partitioning allows to partition tables (fact and dimension tables) (Bellatreche et al., 2006) and materialized views and indexes (Sanjay et al., 2004). It has also been used designing parallel data warehouses. Due to its importance in the database fields, most of today's commercial database systems offer native DDL (data definition language) support for defining horizontal partitions of a table (Sanjay et al., 2004), where many partitioning methods are available: range, hash, list and composite (hash list, range hash). These methods map a given row in an object to a key partition. All rows of the object with the same partition number are stored in the same partition. These partitions may be assigned either in a single-node partitioning (single machine) or in multi-node partitioning (parallel machine). A range partitioning method is defined by a tuple (c, V), where c is a column type and V is an ordered sequence of values from the domain of c. This partitioning mode is often used when there is a logical range (examples: starting date, transaction date, close date, etc.).

Example 1

```
CREATE TABLE PARTITION_BY_RANGE
(. . . , BIRTH_MM INT NOT NULL, BIRTH_DD
INT NOT NULL, BIRTH_YYYY INT NOT
NULL)
PARTITION BY RANGE (BIRTH_YYYY,
BIRTH_MM, BIRTH_DD)
(PARTITION P_01 VALUES LESS THAN (1970,
01, 01) TABLESPACE TS01, . . .
PARTITION P_N VALUES LESS THAN
(MAXVALUE, MAXVALUE, MAXVALUE)
TABLESPACE TS05)
ENABLE ROW MOVEMENT;
```

The hash mode decomposes the data according to a hash function (provided by the system) applied to the values of the partitioning columns.

Example 2

```
CREATE TABLE PARTITION_BY_HASH
(FIRST_NAME VARCHAR2(10), MIDDLE_
INIT VARCHAR2(1), LAST_NAME VAR-
CHAR2(10), AGE INT NOT NULL)
PARTITION BY HASH (AGE)
(PARTITION P1TABLESPACE TS01,
 PARTITION P2 TABLESPACE TS02,
 PARTITION P3 TABLESPACE TS03,
 PARTITION P4 TABLESPACE TS04)
ENABLE ROW MOVEMENT;
```

The list partitioning splits a table according list values of a column. These methods can be combined to generate composite partitioning (range-hash, range-list). These methods are available in Oracle.