Handling Very Large Databases with Informix Extended Parallel Server

Andreas Weininger
Informix Software
andreasw@informix.com

Abstract

In this paper, we investigate which problems exist in very large real databases and describe which mechanisms are provided by Informix Extended Parallel Server (XPS) for dealing with these problems. Currently the largest customer XPS database contains 27 TB of data. A database server that has to handle such an amount of data has to provide mechanisms which allow achieving adequate performance and easing the usability. We will present mechanisms which address both of these issues and illustrate them with examples from real customer systems.

1 Introduction

Most large databases are data warehouses or operational data stores. Therefore, we will concentrate in this paper on these types of database systems. Two different phases can be distinguished for data warehouses:

- A management phase during which data are inserted, updated, or deleted from the database and auxiliary structures like indices are created, and
- a query phase.

Traditionally, the management phase happned often during the night, and the queries were running during the day. However, for databases with web access and for databases in global companies, which have users in offices all around the globe, these two phases are more or less overlapping in time. Therefore, a database system must support this.

The following section discusses problems specific to the management phase of a very large data warehouse, and how these are solved by the Informix Extented Parallel Server (XPS). XPS is a shared nothing DBMS with support for SMP machines, clusters, and MPP systems.

The next section considers problems and solutions for the query phase. After this, we look at a concrete example from a telecommunications company.

MOD 2000, Dallas, TX USA

2 Problems and Solutions for the Management Phase

Loading and incremental loading of data is an important operation for data warehouses: It is the way how all the data gets into data warehouse. In some cases unloading/loading is used as a backup/restore solution.

Specific problems associated with loading are:

- Loads can be frequent and involve a large amount of data. For instance, one telecom company is loading daily 30 GB of data in a data warehouse.
- Normal queries are running during the load.
- The load often involved a lot of transformations and checks. For instance, unique and foreign key constraints have to be checked. Doing these checks via indices is usually too expensive.
- The queries executed by load jobs are often very complex. It is not untypical that these queries are the most expensive queries run against the data warehouse

Other problems during the management phase are:

- If old data is removed by delete statements, data is no longer stored as compact as before.
- Delete and update operations involve a large amount of data.

Some of these problems do also exist for smaller database but because of the amount of data, they become much more difficult to handle in very large databases.

There are several features in XPS which allow to solve the above problems efficiently:

- External tables provide an easy way of loading and unloading tables. They are created similar to regular database tables but describe the load/unload files. They can be used in the same way as internal database tables in insert and select statements. It is even possible to join internal and external tables.
- Hybrid fragmentation allows a two-dimensional partitioning of tables. Partitioning in one dimension (usually by hash) allows full use of parallelism while the second dimension provides a second criteria for reducing the amount of work per query by fragment elimination. The example in section 4 illustrates this in more detail.
- Hybrid fragmentation provides an mechanism for efficient incremental loading and deleting: A new table fragmented by hash in the same way as the target table can be loaded while the target table is still in use by other queries. After this, new table can be attached to the target table without any need for data movement. Old data can be removed by detaching the corresponding fragments.
- Update and delete joins provide statements, which handle mass updates and deletes efficiently.
- Constraint checks can be done by queries which use hash-based implementaions for operations. Thus random disk accesses are avoided.

3 Problems and Solutions for the Query Phase

There are two types of queries: Queries involving a large amount of data, which should be executed with intra-query parallelism and queries, which access only a small amount of data, which should get executed serially. For large queries to scale all operations must be executed in parallel: Insert, update, delete, select, update statistics, index builds, but even index checks have to be done in parallel in a very large database. In addition, the use of several coservers (which are logical nodes) on one SMP node in XPS allows to have several parallel instances of resources like the database log which would otherwise not exist in parallel.

For putting complex logic in SQL statements which are executed with intra-query parallelism by the engine, it is important to have a conditional expression like the SQL-92 CASE statements which can be used in any place where a column expression can be used.

4 Example

This examples shows how hybrid fragmentation is used in a data warehouse at a telecommunications company for the physical design of the largest table of the warehouse which contains the call detail records.

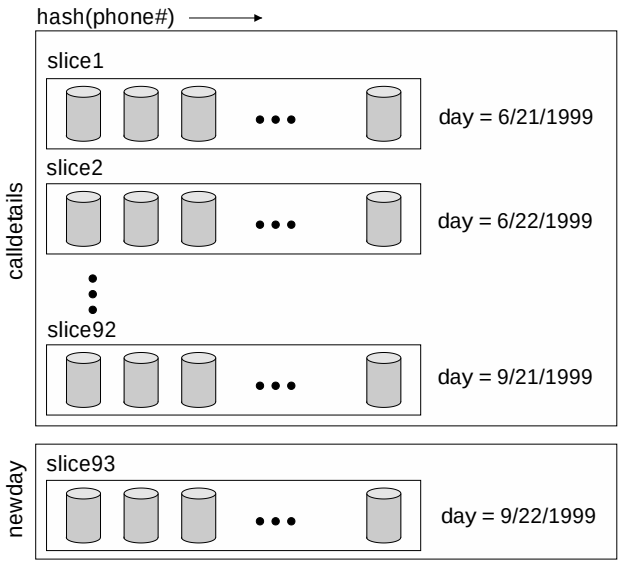

Figure 1: Fragmentation of calldetails table

This data warehouse contains three month of call detail records in a table called calldetails. Hybrid fragmentation is used for this table. By this way, the table is partitioned in so-called dbslices, which are a set of dbspaces, which are the database equivalent of disks. Within one dbslice, the data is distributed by a hash function on the column phonenumber. Therefore, queries involving all customers i.e. all phone numbers can use all the available parallelism while at the same time, the amount of work done by a query scanning calldetails can be reduced by fragment elimination, if the query restricts itself to a specific time interval.

Once a day, new call detail recoreds are loaded: First, these data are loaded into the table newday. Then, this table can be attached to the calldetails table without any effort. Deleting the oldest data from day 6/21/1999 can be done by detaching the corresponding dbslice from the table calldetails. Therefore, no delete has to run to achieve this.

5 Summary

This paper concentrated the discussion on topics related to the physical design of very large databases and the support provided by XPS. There are of course many other aspects which have to be considered when implementing large databases, like an efficient and scalable implementation of operations like join and group. However, these topics are beyond the scope of this paper.