Data Warehouse Performance

Beixin (Betsy) Lin
Montclair State University, USA

Yu Hong
BearingPoint Inc., USA

Zu-Hsu Lee
Montclair State University, USA

INTRODUCTION

A data warehouse is a large electronic repository of information that is generated and updated in a structured manner by an enterprise over time to aid business intelligence and to support decision making. Data stored in a data warehouse is non-volatile and time variant and is organized by subjects in a manner to support decision making (Inmon, Rudin, Buss, & Sousa, 1998). Data warehousing has been increasingly adopted by enterprises as the backbone technology for business intelligence reporting and query performance has become the key to the successful implementation of data warehouses. According to a survey of 358 businesses on reporting and end-user query tools, conducted by Appfluent Technology, data warehouse performance significantly affects the Return on Investment (ROI) on Business Intelligence (BI) systems and directly impacts the bottom line of the systems (Appfluent Technology, 2002). Even though in some circumstances it is very difficult to measure the benefits of BI projects in terms of ROI or dollar figures, management teams are still eager to have a "single version of the truth," better information for strategic and tactical decision making, and more efficient business processes by using BI solutions (Eckerson, 2003).

Dramatic increases in data volumes over time and the mixed quality of data can adversely affect the performance of a data warehouse. Some data may become outdated over time and can be mixed with data that are still valid for decision making. In addition, data are often collected to meet potential requirements, but may never be used. Data warehouses also contain external data (e.g. demographic, psychographic, etc.) to support a variety of predictive data mining activities. All these factors contribute to the massive growth of data volume. As a result, even a simple query may become burdensome to process and cause overflowing system indices (Inmon, Rudin, Buss & Sousa, 2001). Thus, exploring the techniques of performance tuning becomes an important subject in data warehouse management.

BACKGROUND

There are inherent differences between a traditional database system and a data warehouse system, though to a certain extent, all databases are similarly designed to serve a basic administrative purpose, e.g., to deliver a quick response to transactional data processes such as entry, update, query and retrieval. For many conventional databases, this objective has been achieved by online transactional processing (OLTP) systems (e.g. Oracle Corp, 2004; Winter & Auerbach, 2004). In contrast, data warehouses deal with a huge volume of data that are more historical in nature. Moreover, data warehouse designs are strongly organized for decision making by subject matter rather than by defined access or system privileges. As a result, a dimension model is usually adopted in a data warehouse to meet these needs, whereas an Entity-Relationship model is commonly used in an OLTP system. Due to these differences, an OLTP query usually requires much shorter processing time than a data warehouse query (Raden, 2003). Performance enhancement techniques are, therefore, especially critical in the arena of data warehousing.

Despite the differences, these two types of database systems share some common characteristics. Some techniques used in a data warehouse to achieve a better performance are similar to those used in OLTP, while some are only developed in relation to data warehousing. For example, as in an OLTP system, an index is also used in a data warehouse system, though a data warehouse might have different kinds of indexing mechanisms based on its granularity. Partitioning is a technique which can be used in data warehouse systems as well (Silberstein, Eacrett, Mayer, & Lo, 2003).

On the other hand, some techniques are developed specifically to improve the performance of data warehouses. For example, aggregates can be built to provide a quick response time for summary information (e.g. Eacrett, 2003; Silberstein, 2003). Query parallelism can be implemented to speed up the query when data are queried from

several tables (Silberstein, et al., 2003). Caching and query statistics are unique for data warehouses since the statistics will help to build a smart cache for better performance. Also, pre-calculated reports are useful to certain groups of users who are only interested in seeing static reports (Eacrett, 2003). Periodic data compression and archiving helps to cleanse the data warehouse environment. Keeping only the necessary data online will allow faster access (e.g. Kimball, 1996).

MAIN THRUST

As discussed earlier, performance issues play a crucial role in a data warehouse environment. This chapter describes ways to design, build, and manage data warehouses for optimum performance. The techniques of tuning and refining the data warehouse discussed below have been developed in recent years to reduce operating and maintenance costs and to substantially improve the performance of new and existing data warehouses.

Performance Optimization at the Data Model Design Stage

Adopting a good data model design is a proactive way to enhance future performance. In the data warehouse design phase, the following factors should be taken into consideration.

- **Granularity:** Granularity is the main issue that needs to be investigated carefully before the data warehouse is built. For example, does the report need the data at the level of store keeping units (SKUs), or just at the brand level? These are size questions that should be asked of business users before designing the model. Since a data warehouse is a decision support system, rather than a transactional system, the level of detail required is usually not as deep as the latter. For instance, a data warehouse does not need data at the document level such as sales orders, purchase orders, which are usually needed in a transactional system. In such a case, data should be summarized before they are loaded into the system. Defining the data that are needed – no more and no less – will determine the performance in the future. In some cases, the Operational Data Stores (ODS) will be a good place to store the most detailed granular level data and those data can be provided on the jump query basis.
- **Cardinality:** Cardinality means the number of possible entries of the table. By collecting business requirements, the cardinality of the table can be decided. Given a table's cardinality, an appropriate indexing method can then be chosen.
- **Dimensional Models:** Most data warehouse designs use dimensional models, such as Star-Schema, Snow-Flake, and Star-Flake. A star-schema is a dimensional model with fully denormalized hierarchies, whereas a snowflake schema is a dimensional model with fully normalized hierarchies. A star-flake schema represents a combination of a star schema and a snow-flake schema (e.g. Moody & Kortink, 2003). Data warehouse architects should consider the pros and cons of each dimensional model before making a choice.

Aggregates

Aggregates are the subsets of the fact table data (Eacrett, 2003). The data from the fact table are summarized into aggregates and stored physically in a different table than the fact table. Aggregates can significantly increase the performance of the OLAP query since the query will read fewer data from the aggregates than from the fact table. Database read time is the major factor in query execution time. Being able to reduce the database read time will help the query performance a great deal since fewer data are being read. However, the disadvantage of using aggregates is its loading performance. The data loaded into the fact table have to be rolled up to the aggregates, which means any newly updated records will have to be updated in the aggregates as well to make the data in the aggregates consistent with those in the fact table. Keeping the data as current as possible has presented a real challenge to data warehousing (e.g. Bruckner & Tjoa, 2002). The ratio of the database records transferred to the database records read is a good indicator of whether or not to use the aggregate technique. In practice, if the ratio is 1/10 or less, building aggregates will definitely help performance (Silberstein, 2003).

Database Partitioning

Logical partitioning means using year, planned/actual data, and business regions as criteria to partition the database into smaller data sets. After logical partitioning, a database view is created to include all the partitioned tables. In this case, no extra storage is needed and each partitioned table will be smaller to accelerate the query (Silberstein, Eacrett, Mayer & Lo, 2003). Take a multinational company as an example. It is better to put the data from different countries into different data targets, such as cubes or data marts, than to put the data from all the countries into one data target. By logical partitioning (splitting the data into smaller cubes), query can read the