Physical Data Warehousing Design

Ladjel Bellatreche
LISI/ENSMA, France

Mukesh Mohania
IBM India Research Lab, India

INTRODUCTION

Recently, organizations have increasingly emphasized applications in which current and historical data are analyzed and explored comprehensively, identifying useful trends and creating summaries of the data in order to support high-level decision making. Every organization keeps accumulating data from different functional units, so that they can be analyzed (after integration), and important decisions can be made from the analytical results. Conceptually, a data warehouse is extremely simple. As popularized by Inmon (1992), it is a "subject-oriented, integrated, time-invariant, non-updatable collection of data used to support management decision-making processes and business intelligence". A data warehouse is a repository into which are placed all data relevant to the management of an organization and from which emerge the information and knowledge needed to effectively manage the organization. This management can be done using data-mining techniques, comparisons of historical data, and trend analysis. For such analysis, it is vital that (1) data should be accurate, complete, consistent, well defined, and time-stamped for informational purposes; and (2) data should follow business rules and satisfy integrity constraints. Designing a data warehouse is a lengthy, time-consuming, and iterative process. Due to the interactive nature of a data warehouse application, having fast query response time is a critical performance goal. Therefore, the physical design of a warehouse gets the lion's part of research done in the data warehousing area. Several techniques have been developed to meet the performance requirement of such an application, including materialized views, indexing techniques, partitioning and parallel processing, and so forth. Next, we briefly outline the architecture of a data warehousing system.

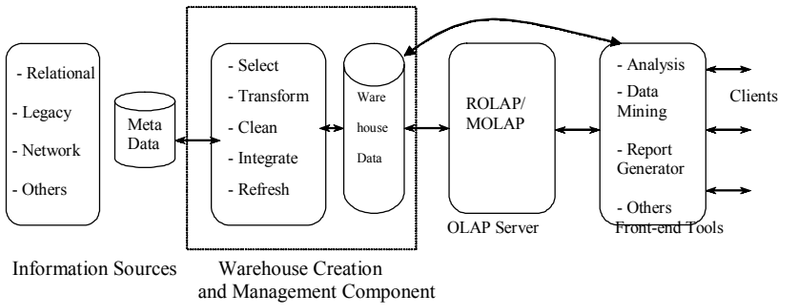
BACKGROUND

The conceptual architecture of a data warehousing system is shown in Figure 1. Data of a warehouse is extracted from operational databases (relational, object-oriented, or relational-object) and external sources (legacy data, other files formats) that may be distributed, autonomous, and heterogeneous. Before integrating this data into a warehouse, it should be cleaned to minimize errors and to fill in missing information, when possible, and transformed to reconcile semantic conflicts that can be found in the various sources. The cleaned and transformed data are integrated finally into a warehouse. Since the sources are updated periodically, it is necessary to refresh the warehouse. This component also is responsible for managing the warehouse data, creating indices on data tables, partitioning data, and updating meta-data. The warehouse data contain the detail data, summary data, consolidated data, and/or multi-dimensional data. The data typically are accessed and analyzed using tools, including OLAP query engines, data mining algorithms, information, visualization tools, statistical packages, and report generators.

The meta-data generally is held in a separate repository. The meta-data contain the informational data about the creation, management, and usage of tools (e.g., analytical tools, report writers, spreadsheets and data-mining tools) for analysis and informational purposes. Basically, the OLAP server interprets client queries (the client interacts with the front-end tools and passes these queries to the OLAP server) and converts them into complex SQL queries required to access the warehouse data. It also might access the data warehouse. It serves as a bridge between the users of the warehouse and the data contained in it. The warehouse data also are accessed by the OLAP server to present the data in a multi-dimensional way to the front-end tools. Finally, the OLAP server passes the multi-dimensional views of data to the front-end tools, which format the data according to the client's requirements.

The warehouse data are typically modeled multi-dimensionally. The multi-dimensional data model has been proved to be the most suitable for OLAP applications. OLAP tools provide an environment for decision making and business modeling activities by supporting ad-hoc queries. There are two ways to implement a multi-dimensional data model: (1) by using the underlying relational architecture (star schemas, snowflake schemas)

Figure 1. A data warehousing architecture



to project a pseudo-multi-dimensional model (example includes Informix Red Brick Warehouse); and (2) by using true multi-dimensional data structures such as, arrays (example includes Hyperion Essbase OLAP Server Hyperion). The advantage of MOLAP architecture is that it provides a direct multi-dimensional view of the data whereas the ROLAP architecture is just a multi-dimensional interface to relational data. On the other hand, the ROLAP architecture has two major advantages: (i) it can be used and easily integrated into other existing relational database systems; and (ii) relational data can be stored more efficiently than multi-dimensional data.

Data warehousing query operations include standard SQL operations, such as selection, projection, and join. In addition, it supports various extensions to aggregate functions, such as percentile functions (e.g., top 20th percentile of all products), rank functions (e.g., top 10 products), mean, mode, and median. One of the important extensions to the existing query language is to support multiple group-by, by defining roll-up, drill-down, and cube operators. Roll-up corresponds to doing further group-by on the same data object. Note that roll-up operator is order sensitive; that is, when it is defined in the extended SQL, the order of columns (attributes) matters. The function of a drill-down operation is the opposite of roll-up.

OLTP vs. OLAP

Relational database systems (RDBMS) are designed to record, retrieve, and manage large amounts of real-time transaction data and to keep the organization running by supporting daily business transactions (e.g., update transactions). These systems generally are tuned for a large community of users, and the user typically knows what is needed and generally accesses a small number of rows in a single transaction. Indeed, relational database systems are suited for robust and efficient Online Transaction Processing (OLTP) on operational data. Such OLTP applications have driven the growth of the DBMS industry in the past three decades and will doubtless continue to be important. One of the main objectives of relational systems is to maximize transaction throughput and minimize concurrency conflicts. However, these systems generally have limited decision support functions and do not extract all the necessary information required for faster, better, and intelligent decision making for the growth of an organization. For example, it is hard for an RDBMS to answer the following query: What are the supply patterns for product ABC in New Delhi in 2003, and how were they different from the year 2002? Therefore, it has become important to support analytical processing capabilities in organizations for (1) the efficient management of organizations, (2) effective marketing strategies, and (3) efficient and intelligent decision making. OLAP tools are well suited for complex data analysis, such as multi-dimensional data analysis and to assist in decision support activities that access data from a separate repository called a data warehouse, which selects data from many operational legacies, and possibly heterogeneous data sources. The following table summarizes the differences between OLTP and OLAP.

MAIN THRUST

Decision-support systems demand speedy access to data, no matter how complex the query. To satisfy this objective, many optimization techniques exist in the literature. Most of these techniques are inherited from traditional relational database systems. Among them are materialized views (Bellatreche et al., 2000; Jixue et al., 2003; Mohania et al., 2000; Sanjay et al., 2000, 2001), indexing methods (Chaudhuri et al., 1999; Jügens et al., 2001; Stockinger et al., 2002), data partitioning (Bellatreche et al., 2000, 2002, 2004; Gopalkrishnan et al., 2000; Kalnis et al., 2002; Oracle, 2000; Sanjay et al., 2004), and parallel processing (Datta et al., 1998).