Chapter 12
On Querying Data and Metadata in Multiversion Data Warehouse *

Wojciech Leja
Poznań University of Technology, Poland

Robert Wrembel
Poznań University of Technology, Poland

Robert Ziembicki
Poznań University of Technology, Poland

ABSTRACT
Methods of designing a data warehouse (DW) usually assume that its structure is static. In practice, however, a DW structure changes among others as the result of the evolution of external data sources, the changes of the real world represented in a DW, and new user requirements. The most advanced research approaches to managing the evolution of DWs are based on temporal extensions and versioning techniques. An important feature of a DW system supporting evolution is its ability to query different DW states. Such querying is challenging since different DW states may differ with respect to their schemas. As a consequence, a system may not be able to execute a query for some DW states. Our approach to managing the evolution of DWs is based on the so-called Multiversion Data Warehouse (MVDW) that is composed of the sequence of DW versions. In this chapter, we contribute a query language called MVDWQL for querying the MVDW. The MVDWQL supports two types of queries, namely content queries and metadata queries. A content query is used for analyzing the content (i.e., data) of multiple DW versions. A metadata query is used for analyzing the history of evolution of the MVDW. The results of both types of queries are graphically visualized in a user interface.

INTRODUCTION
Contemporary manner of managing enterprises is based on knowledge. Typically, knowledge is gained from the advanced analysis of various types of data processed and collected during the lifetime of an enterprise. In practice, within the same enterprise data are stored in multiple heterogeneous
On Querying Data and Metadata in Multiversion Data Warehouse *

and autonomous storage systems that often are geographically distributed. In order to provide means for the analysis of data coming from such systems, a data warehouse architecture has been developed (Jarke et al., 2003; Widom, 1995). The data warehouse architecture, firstly, offers techniques for the integration of multiple data sources in one central repository, called a data warehouse (DW). Secondly, it offers means for advanced, complex, and efficient analysis of integrated data.

Data in a DW are organized according to a specific conceptual model (Gyssens & Lakshmanan, 1997; Letz, Henn, & Vossen, 2002). In this model, an elementary information being the subject of analysis is called a fact. It contains numerical features, called measures (e.g., quantity, income, duration time) that quantify the fact and allow to compare different facts. Values of measures depend on a context set up by dimensions. A dimension is composed of levels that form a hierarchy. A lower level is connected to its direct parent level by a relation, further denoted as $\rightarrow$. Every level $l_i$ has associated a domain of values. The finite subset of domain values constitutes the set of level instances. The instances of levels in a given dimension are related to each other, so that they form a hierarchy, called a dimension instance. A typical example of a dimension, is Location. It may be composed, for example, of three hierarchically connected levels, i.e., $\text{Shops} \rightarrow \text{Cities} \rightarrow \text{Regions}$. An example instance of dimension Location may include: $\{\text{Macy's} \rightarrow \text{New Orleans} \rightarrow \text{Louisiana}\}$, $\{\text{Timberland} \rightarrow \text{Houston} \rightarrow \text{Texas}\}$.

In practice, this conceptual model of a DW can be implemented either in multidimensional OLAP servers (MOLAP) or in relational OLAP servers (ROLAP). In a MOLAP implementation, data are stored in specialized multidimensional data structures whereas in a ROLAP implementation, data are stored in relational tables. Some of the tables represent levels and are called level tables, while others store values of measures, and are called fact tables. Level and fact tables are typically organized into a star schema or a snowflake schema (Chaudhuri & Dayal, 1997).

**DW Evolution**

For a long period of time, research concepts, prototypes, and commercial DW systems have assumed that the structure of a deployed DW is time invariant. This assumption turned out to be false. In practice, a DW structure may evolve (change) among others as the result of the evolution of external data sources, the changes of the real world represented by a DW, new user requirements, as well as the creation of simulation environments (Mendelzon & Vaisman, 2000; Rundenstein, Koeller, & Zhang, 2000; Wrembel, 2009).

The most advanced research approaches to managing the evolution of DWs are based on temporal extensions (Bruckner & Tjoa, 2002; Chamoni & Stock, 1999; Eder & Koncilia, 2001; Eder, Koncilia, & Morzy, 2002; Letz et al., 2002; Malinowski & Zimányi, 2008; Schlesinger et al., 2001), and versioning extensions (Body et al., 2002; Golfarelli et al., 2004; Mendelzon & Vaisman, 2000; Ravat, Teste, & Zurfluh, 2006; Rizzi & Golfarelli, 2007; Vaisman & Mendelzon, 2001). Concepts from the first category use timestamps on modified data in order to create temporal versions. In versioning extensions, a DW evolution is managed partially by means of schema versions and partially by data versions. These concepts solve the DW evolution problem partially. Firstly, they do not offer a clear separation between different DW states. Secondly, they do not support modeling alternative, hypothetical DW states required for simulations and predictions within the so-called ‘what-if’ analysis.

In order to eliminate the limitations of the aforementioned approaches, we proposed the so-called Multiversion Data Warehouse (MVDW). The MVDW is composed of the sequence of DW versions, each of which represents either the real-world state within a certain period of time or a ‘what-if’ simulation scenario (Bębel et al., 2004).