Chapter 2.7
Applying UML for Modeling the Physical Design of Data Warehouses

Sergio Luján-Mora
Universidad de Alicante, Spain

Juan Trujillo
Universidad de Alicante, Spain

ABSTRACT

In previous work, we have shown how to use unified modeling language (UML) as the primary representation mechanism to model conceptual design, logical design, modeling of extraction, transformation, loading (ETL) processes, and defining online analytical processing (OLAP) requirements of data warehouses (DW). Continuing our work on using UML throughout the DW development lifecycle, in this chapter, we present our modeling techniques of physical design of DW using component diagrams and deployment diagrams of UML. Our approach allows the DW designer to anticipate important physical design decisions that may reduce the overall development time of a DW such as replicating dimension tables, vertical and horizontal partitioning of a fact table, and the use of particular servers for certain ETL processes. We illustrate our techniques with a case study.

INTRODUCTION

In the early 90s, Bill Inmon (Inmon, 2002) coined the term *data warehouse* (DW): "A data warehouse is a subject-oriented, integrated, time-variant, non-volatile collection of data in support of management's decisions" (p. 33). This definition contains four key elements that deserve a detailed explanation:

- **Subject orientation** means that the development of the DW will be done in order to satisfy the analytical requirements of managers that will query the DW. The topics of analysis differ and depend on the kind of

business activities; for example, it can be product sales focusing on client interests in some sales company, the client behavior in utilization of different banking services, the insurance history of the clients, the railroad system utilization or changes in structure, and so forth.

- **Integration** relates to the problem that data from different operational and external systems have to be joined. In this process, some problems have to be resolved: differences in data format, data codification, synonyms (fields with different names but the same data), homonyms (fields with the same name but different meaning), multiplicity of data occurrences, nulls presence, default values selection, and so forth.
- **Non-volatility** implies data durability: Data can neither be modified nor removed.
- **Time-variation** indicates the possibility to count on different values of the same object according to its changes in time. For example, in a banking DW, the average balances of client's account during different months for the period of several years.

DWs provide organizations with historical information to support a decision. It is widely accepted that these systems are based on multidimensional (MD) modeling. Thus, research on the design of a DW has been mainly addressed from the conceptual and logical point of view through multidimensional (MD) data models (Blaschka, Sapia, Höfling, & Dinter, 1998, Abelló, Samos, & Saltor, 2001). During the few last years, few efforts have been dedicated to the modeling of the physical design (e.g., the physical structures that will host data together with their corresponding implementations) of a DW from the early stages of a DW project.

Nevertheless, the physical design of a DW is vitally important and highly influences the overall performance of the DW (Nicola & Rizvi, 2003) and the following maintenance; even more, a well-structured physical design policy can provide the perfect roadmap for implementing the whole warehouse architecture (Triantafillakis, Kanellis, & Martakos, 2004).

In some companies, the same employee may take on both the role of DW designer and DW administrator; other organizations may have separate people working on each task. Regardless of the situation, modeling the storage of the data and how it will be deployed across different components (servers, drives, and so forth) helps in the implementation and maintenance of a DW. In traditional software products or transactional databases, physical design or implementation issues are not considered until the latest stages of a software project. Then, if the final product does not satisfy user requirements, designers do a feedback taking into consideration (or at least bearing in mind) some final implementation issues.

Nevertheless, due to the specific characteristics of DWs, we can address several decisions regarding the physical design of a DW from the early stages of a DW project, with no need to leave them until the final implementation stage. DWs, mainly built for analytical reasons, are queried by final users trying to analyze historical data on which they can base their strategy decisions. Thus, the performance measure for DWs is the amount of queries that can be executed instead of the amount of processes or transactions that it supports. Moreover, the kinds of queries on DWs are demonstrated to be much more complex than the queries normally posed in transactional databases (Kimball, 2002, Poe, Klauer, & Brobst, 1998). Therefore, poor performance of queries has a worse impact in DWs than in transactional databases. Furthermore, the set of online analytical processing (OLAP) operations that users can execute with OLAP tools on DWs depends so much on the design of the DW, that is, on the multidimensional model underneath (Sapia, 1999, Trujillo, Palomar, Gómez, & Song, 2001).

Based on our experience in real world DW projects, physical storage and query performance