Chapter XIV
Schema Evolution Models and Languages for Multidimensional Data Warehouses

Edgard Benítez-Guerrero
Laboratorio Nacional de Informática Avanzada, Mexico

Ericka-Janet Rechy-Ramírez
Laboratorio Nacional de Informática Avanzada, Mexico

INTRODUCTION

A Data Warehouse (DW) is a collection of historical data, built by gathering and integrating data from several sources, which supports decision-making processes (Inmon, 1992). On-Line Analytical Processing (OLAP) applications provide users with a multidimensional view of the DW and the tools to manipulate it (Codd, 1993). In this view, a DW is seen as a set of dimensions and cubes (Torlone, 2003). A dimension represents a business perspective under which data analysis is performed and organized in a hierarchy of levels that correspond to different ways to group its elements (e.g., the Time dimension is organized as a hierarchy involving days at the lower level and months and years at higher levels). A cube represents factual data on which the analysis is focused and associates measures (e.g., in a store chain, a measure is the quantity of products sold) with coordinates defined over a set of dimension levels (e.g., product, store, and day of sale). Interrogation is then aimed at aggregating measures at various levels. DWs are often implemented using multidimensional or relational DBMSs. Multidimensional systems directly support the multidimensional data model, while a relational implementation typically employs star schemas

(or variations thereof), where a fact table containing the measures references a set of dimension tables.

One important problem that remains open in the Data Warehouse context is schema evolution, the dynamic modification of the schema of a database (Banerjee, Kim, Kim & Korth, 1987, Roddick, 1995). Early works assumed that there was no reason to evolve the schema of a DW. However, practitioners recognized several reasons later (Zurek & Sinnwell, 1999): (i) it is important to support incremental approaches to schema design; (ii) it is necessary to improve the design or to fix errors; and (iii) new user requirements arise. If these changes are not integrated, the DW becomes incomplete and users' information needs are not satisfied anymore.

Schema evolution is a problem that has been mainly studied for relational and object-oriented (OO) databases. However, existing solutions (e.g., Banerjee et al., 1987; McKenzie, & Snodgrass, 1990; Roddick, 1995) are only partially suited for the DW context because they do not consider the multidimensional nature of data. In response to these deficiencies, several works have proposed conceptual evolution models for DWs that consider those features.

The objective of this entry is to introduce the problem of DW schema evolution, explaining current solutions and identifying open problems. It is organized as follows. First, background information covering traditional solutions to the schema evolution problem will be introduced. Then, research on conceptual evolution models and languages will be presented, comparing reference works to others. Open issues will be introduced. Finally, this entry will conclude with a summary.

BACKGROUND

Schema evolution is a subject that has been studied for years. This section briefly introduces traditional works and introduces the problem in the DW context.

Schema Evolution on Relational and Object-Oriented Databases

The problem of schema evolution has been mainly studied for relational and OO databases. In the relational case, few concepts (relations and attributes) are used to describe a database schema. Thus, possible changes are limited to (McKenzie & Snodgrass, 1990) addition/suppression of an attribute or modification of its type, addition/suppression of a relation, and so forth. In the OO context, the model (classes, attributes, methods, is-a relationships) is richer, and then schemas are more complex. Possible changes are (Banerjee et al., 1987) addition, suppression or modification of attributes and methods in a class definition, or changes in superclass/subclass relationships.

Schema Evolution Management

Evolving a schema raises problems at schema, instance, and application levels. At *schema* level, a modification in a part of a schema can cause inconsistencies with other parts. At *instance* level, data can become inconsistent with respect to their schema. At *application* level, applications can become incompatible with the schema after a change. Existing approaches to manage schema evolution try to minimize those problems. These approaches are schema modification, usage of views, and versioning.

Schema Modification

In the schema modification approach, the database schema has only one definition, shared by all applications. So each time the schema needs to be changed, it is updated, and its associated instances are modified accordingly (Penney & Stein, 1987). Therefore, information can be lost if changes imply the removal or modification of