ABSTRACT

In spite of the obvious importance of time in data warehousing and OLAP, current commercial systems do not support tracking the history of a data warehouse, either at the schema or instance level. In this chapter we address this issue, introducing the Temporal Multidimensional Model and a query language, denoted TOLAP, allowing expressing temporal OLAP queries at a high level of abstraction. Further, we show that previous work in temporal databases needs to be extended in order to handle evolution and versioning in OLAP. Finally, we present an implementation, along with preliminary experimental results.

INTRODUCTION

In previous chapters it was shown that in a relational implementation of OLAP (ROLAP), facts are usually stored in fact tables, while each dimension is described in a dimension table. Although time is obviously central to most data warehousing and OLAP applications, currently available commercial systems do not account for the data warehouse’s history. In this chapter we concentrate on data warehouse evolution and versioning, introducing a temporal multidimensional data model and a temporal query language supporting it, which we called TOLAP (Temporal
OLAP), combining some of the temporal features of query languages like TSQL2 (Snodgrass, 1995) or SQL/TP (Toman, 1997) with some of the high-order features of languages like HiLog (Chen, Kifer, & Warren, 1989) or SchemaLog (Lakshmanan, Sadri, & Subramanian, 1993).

A data warehouse could be regarded as a materialized view of data located in multiple sources (Widom, 1995). Thus, it is not difficult to imagine a scenario in which the structure of these sources changes, a new source is added, or an old one dropped. Any of these changes may require updates to the structure of some dimensions. Moreover, as multidimensional views are designed according to requirements from end users, a redefinition of the initial requirements may cause a dimension update. For instance, let us consider Figure 1, which depicts a geography dimension where regions are defined within the same country. A business decision may relax this constraint, allowing regions to be spread across different countries. This may be represented in the lattice of Figure 1 by deleting the edge joining the region and country levels, and adding a new edge from region to the distinguished level All. Figure 2 shows the resulting dimension. Additionally, new salespersons may be hired or fired; new kinds of coverage introduced or discontinued; regions may be reorganized, merged, or split; etc. However, a user may be interested in querying the multidimensional database as of the instant depicted in Figure 1. Moreover, since the schemas of the fact tables are composed of attributes from associated dimensions, certain updates may trigger schema evolution over such fact tables. Suppose we wish to collect data at a granularity level finer than city, for instance neighborhood. Any fact table associated with the geography dimension would require its schema to be updated. We argue in this chapter that in an evolving scenario like this, OLAP systems need temporal features that allow tracking of the different states of a data warehouse throughout its lifespan.

Figure 1: A Geography Dimension

![Figure 1: A Geography Dimension](image-url)
XTEngine: A Twin Search Engine for XML
www.igi-global.com/chapter/xtengine-twin-search-engine-xml/52957?camid=4v1a

Quadtree-based Image Representation and Retrieval
www.igi-global.com/chapter/quadtree-based-image-representation-retrieval/29660?camid=4v1a