Chapter 10
From Temporal Databases to Ontology Versioning:
An Approach for Ontology Evolution

Najla Sassi
MIRACL Laboratory, Tunisia

Zouhaier Brahmia
MIRACL Laboratory, Tunisia

Wassim Jaziri
MIRACL Laboratory, Tunisia

Rafik Bouaziz
MIRACL Laboratory, Tunisia

ABSTRACT

The problem of versioning is present in several application areas, such as temporal databases, real-time computing and ontologies. This problem is generally defined as managing changes in a timely manner without loss of existing data. However, ontology versioning is more complicated than versioning in database because of the usage and content of ontology which incorporates semantic aspects. Consequently, ontology data models are much richer than those of database schemas. In this chapter, the authors are interested in developing an ontology versioning system to express, apply and implement changes on the ontology.

INTRODUCTION

In computer science, several application areas are especially concerned with versioning such as software development, CAD/CAM applications, temporal databases and ontologies.

Temporal databases support time-varying information and maintain the history of the modelled data (Jensen et al., 1998) (Özsoyoğlu et al., 1995) (Tansel et al., 1993). Versions of temporal data are kept along one or two time dimensions: valid time and transaction time (Jensen et al., 1998). The valid time concerns the time of the modelled real world and denotes the time a fact was, or will be true, whereas the transaction time is the one of the system and concerns the one during which the fact was or is current in the database as a stored data.

DOI: 10.4018/978-1-61520-859-3.ch010

The interest on schema versioning in temporal databases arose as a logical extension of the work formerly done on data.

Ontologies are defined as an explicit representation based on the identification of conceptual entities (concepts or relationships) and their semantics. Since ontology has to be continually changed for many reasons, it is interesting to manage its evolution and to take into account the different versions of this ontology.

Ontologies are like database schema and schema versioning in temporal databases can be useful in order to propose an approach for ontology versioning. In fact, we can benefit from principles and tools we previously defined for schema versioning in such databases, in order to ensure an efficient management of versions in ontological databases.

We are interested in developing an ontology versioning system to express, apply and implement changes on the ontology. The adopted ontology versioning approach is based on three steps: evolution changes, ontology coherence and versioning management. Our goal is to assist users in expressing evolution requirements, observing their consequences on the ontology and comparing ontology versions.

This paper is structured as follows. Sections 2, 3 and 4 present the versioning in software development, databases and ontology. In section 5, we propose an approach of ontology evolution based on three steps: evolution changes, ontology coherence and versioning management. Section 6 is dedicated to the process of ontology versioning storage. Finally, we conclude in section 7.

VERSIONING IN SOFTWARE DEVELOPMENT AND CAD/CAM APPLICATIONS

Software systems are rarely stable following initial implementations. They have complex structures which are likely to continually undergo changes during their lifetime. Software versioning is the process of assigning unique versions to unique states of computer software. These versions correspond to new developments in the software; sometimes they are also called revisions. A software version can be identified by a sequence of numbers and/or letters (for example, "Oracle 10g"), date (for example, "Wine 20040505"), year (for example, "Windows 2000"), or alphanumeric code (for example, "Adobe Dreamweaver CS4").

CAD/CAM, CIM, CASE tools and other engineering applications first put forward the requirement of managing multiple design versions (Kim et al., 1990). The primitive concepts of versioning were introduced to support different users concurrently working on parallel, or even merging, versions of the same piece of data (objects) (Katz, 1984) (Katz, 1990). These non-conventional applications not only often demand the support of many database states, but also design alternatives. To fulfil such requirement, works like (Kim et al., 1989) (Talens et al., 1993) have focused on the question of version support. A version describes an object in a period of time or from a certain point of view. Although some design alternatives are stored as versions, not all the history about data modification is recorded. The full history is just accessible if a temporal model is used. The majority of the above proposals have extended their data model using either temporal concepts or just version control mechanisms.

Temporal databases support time-varying information and maintain the history of the modelled data. Versions of temporal data are kept along one or two time dimensions: valid time and transaction time (Jensen et al., 1998). Versioning can be temporal or non-temporal. In a temporal versioning context, any version has a temporal connotation (a temporal interval or a temporal point); versions are necessarily successive and there are no overlapping temporal connotations of two versions. Temporal versioning is usually used in temporal databases for data and schema versioning. In a non-temporal versioning environment, versions