Temporal Databases

Fabio Grandi
University of Bologna, Italy

INTRODUCTION

Time is a ubiquitous aspect of real world phenomena and most computer applications require the management of time-varying information (e.g., processing of scientific and census data, banking and financial transactions, record-keeping and scheduling applications). Hence, the management of the temporal dimension has become a recognized important requirement of advanced database applications, in which the evolution of dynamic objects has to be represented in full relief and non-destructive changes must be applied to data. The advent of increasingly large and inexpensive storage devices has been the technological spring for the introduction of systems maintaining historical data and keeping track of past activities.

In this article we will briefly resume and discuss the main scientific results in the field of temporal databases. In particular, we will survey the features of proposed temporal data models based on extensions of the relational model and of temporal query languages based on extensions of the SQL standard, which are the most relevant for mainstream application development. Finally, we will survey the currently available implementations of temporal facilities in DBMS platforms.

BACKGROUND

Temporal databases have been an active research area for several decades, primarily focusing on temporal extensions of data models and query languages but also considering several other aspects of database technology. Extensions have been proposed mainly for the relational data model but also for object-oriented, XML, RDF and conceptual models like the Entity-Relationship model. A quite large literature, with pioneering works published in the early 1980s, is the outcome of such an effort as witnessed by several surveys and bibliographies (the latest thereof is Grandi (2012)), which also includes references to previous ones).

All extensions are based on the adoption of one or more time domains of interest for applications, whose values are used to assign a temporal pertinence (e.g., as *timestamps*) to data. The most popular and relevant time dimensions are *valid time* and *transaction time* (Jensen et al., 1998):

- The valid time of a fact is the time when the fact is true in the modeled reality.
- The transaction time of a fact is the time when the fact is stored in the database.

A database equipped with both valid and transaction time is said to be a *bitemporal* database. In a temporal database, a snapshot relation is a traditional relation, without time support. Further time dimensions (e.g., event/decision time, efficacy time or generic "user-defined" time) have also been considered in some application fields. For the modeling of a time domain, several aspects have been taken into account and studied, concerning its structure and features (e.g., discrete versus dense, linear versus branching, finite versus unbounded, besides granularity, periodicity, indeterminacy or probability, calendar support), and special values have been defined and characterized (e.g., "beginning," "now," "∞," "until changed"). In order to exploit the potentialities of a temporal database in applications, several temporal query languages (e.g., SQL extensions) have been proposed.

A milestone for the foundation and development of the discipline was the International Workshop on an Infrastructure for Temporal Databases, which was organized in 1993 under the auspices of the U.S. ARPA/NSF. As a side initiative, a panel of experts gathered to discuss and compile a consensus glossary of widely used technical terms specific to the temporal databases, which was published in 1994. A consolidated, revised and extended version of the glossary (Jensen et al.,

DOI: 10.4018/978-1-4666-5888-2.ch184

1998) came to light after the Dagstuhl Seminar on Temporal Databases held in 1997. Another follow-up of the ARPA/NSF workshop was the setting up of a committee, chaired by Richard Snodgrass, in charge of designing a temporal extension of the standard query language SQL-92: the TSQL2 Language Design Committee produced a first draft in 1994 and the final TSQL2 specification was published in a book (Snodgrass et al., 1995). Parts of TSQL2 were accepted by ANSI and included in a substandard of SQL3 named SQL/Temporal. Due to disagreements within the ISO committee, the project responsible for temporal support was canceled in 2001. However, concepts and constructs from SQL/Temporal were subsequently included in the latest SQL standard published in 2011 and have been implemented in several database platforms.

Temporal database studies have little by little entered into practice, as mainstream commercial DBMSs currently include some support for time-referenced data and temporal query facilities.

TEMPORAL EXTENSIONS OF THE RELATIONAL MODEL AND OF SQL

Temporal Data Models

Time can be associated with data in several different ways. In an extension of the relational model, time *points*, *intervals* or temporal *elements* (i.e., disjoint unions of intervals) can be used as timestamps. Temporal elements have been defined by Gadia (1988) in order to have a closed algebra of timestamp operators (differently from intervals, union and difference of elements is always an element). Moreover, *tuple-timestamping* or *attribute-timestamping* can be used (giving rise to also called *homogenous* and *inhomogeneous* models, respectively). In the former case, timestamps can be stored in implicit or explicit columns added to the table schema. In the latter case, a nested relation structure is needed to encode timestamping.

Let us consider, as simple example, the career of an employee which follows:

1. John was hired as a programmer (PRG) with initial salary 2K at time 1;
2. John's salary was raised to 3K at time 3 (but recorded in the DB at time 4);
3. John became a database administrator (DBA) at time 6.

Notice that (b) involves a retroactive update. This information can be stored in a valid-time, transaction-time or bitemporal table as shown in Figure 1 (tuple-timestamping with intervals is adopted; "–" means "until changed" or "forever" in valid time and "now" or "until changed" in transaction time). As it can also be verified from the figure, a valid-time relation allows users to effect retro- or pro-active changes, that is changes non necessarily effective when they are executed (but for which there is no way to know, after they were effected, whether they were on-time, retro- or pro-active). A transaction-time relation only allows to effect on-time changes, or it would be better to say that changes can only be interpreted as they were effective when applied (in our example, from Figure 1(2), it seems that John started earning 3K from time 4 and there is no way to see that arrears from time 3 were due). A bitemporal relation allows users to effect retro- and pro-active changes and to keep track of them (e.g., a tuple with valid start lesser than transaction start marks the result of a retroactive change). In the presence of retro- or pro-active changes, a bitemporal database only provides for full auditing and accountability.

The same data as in the valid-time relation of Figure 1(1) can also be represented as shown in Figure 2 by adopting other temporal modeling solutions. In particular, the representation in Figure 2(2), which is also representative of interval-timestamping employed at attribute level, corresponds to the case of a temporally *grouped* or history-oriented model (Clifford et al., 2005). In a grouped model, the temporal dimension is implicit in the structure of data representation and data objects are substituted by their histories: attributes can be regarded as functions that map time into domains (Gadia, 1988). Grouped models and query languages have been shown to be more expressive and friendly for human users. They do not lend themselves to implementation in a 1NF relational DBMS, but they have been proposed for implementation in nested relational or XML DBMSs (Wang, Zaniolo, & Zhou, 2005). Temporal models based on addition of timestamping columns are *ungrouped* indeed.

Another distinction involves *point-based* versus *interval-based* data models. In a point-based model, truth values of facts are associated to time points, whereas time intervals can be used merely as a compact