Chapter IV
Managing Temporal Data

Abdullah Uz Tansel
Baruch College, CUNY, USA

INTRODUCTION

In general, databases store current data. However, the capability to maintain temporal data is a crucial requirement for many organizations and provides the base for organizational intelligence. A **temporal database** maintains time-varying data, that is, past, present, and future data. In this chapter, we focus on the relational data model and address the subtle issues in modeling and designing temporal databases.

A common approach to handle temporal data within the traditional relational databases is the addition of time columns to a relation. Though this appears to be a simple and intuitive solution, it does not address many subtle issues peculiar to temporal data, that is, comparing database states at two different time points, capturing the periods for concurrent events and accessing times beyond these periods, handling multi-valued attributes, coalescing and restructuring temporal data, and so forth, [Gadia 1988, Tansel and Tin 1997].

There is a growing interest in temporal databases. A first book dedicated to temporal databases [Tansel at al 1993] followed by others addressing issues in handling time-varying data [Betini, Jajodia and Wang 1988, Date, Darwen and Lorentzos 2002, Snodgrass 1999].

TIME IN DATABASES

The set *T* denotes time values and it is a total order under '≤' relationship. Because of its simplicity, we will use natural numbers to represent time {*0, 1 ... now*}. The symbol *0* is the relative origin of time and *now* is a special symbol that represents the current time. *Now* advances according to the time granularity used. There are different time granularities, such as seconds, minutes, hours, days, month, year, etc. (for a formal definition see [Betini, Jajodia and Wang 1988]).

A subset of *T* is called a *temporal set*. A temporal set that contains consecutive time points {t_1, t_2... t_n} is represented either as a closed interval [t_1, t_n] or as a half open interval [t_1, t_{n+1}). A *temporal element* [Gadia 1988] is a temporal set that is represented by the disjoint maximal intervals corresponding to its subsets having consecutive time points. Temporal sets, intervals, and temporal elements can be used as time stamps for modeling temporal data and are essential constructs in temporal query languages. Temporal sets and temporal elements are closed under set theoretic operations whereas intervals are not. However, intervals are easier to implement. Time intervals, hence temporal elements and temporal sets, can be compared. The possible predicates are *before*, *after*, *meet*, *during*, etc. [Allen 1983]. An interval or a temporal set (element) that includes *now* expends in its duration. Other symbols such as *forever* or *until changed* are also proposed as alternatives to the symbol *now* for easier handling of future data.

There are various aspects of time in databases [Snodgrass 1987]. *Valid time* indicates when a data value becomes effective. It is also known as *logical* or *intrinsic time*. On the other hand, the *transaction time* (or *physical time*) indicates when a value is recorded in the database. *User defined time* is application specific and is an attribute whose domain is time. Temporal databases are in general append-only that is, new data values are added to the database instead of replacing the old values. A database that supports valid time keeps historical values and is called a *valid time* (*historical*) database. A *rollback* database supports transaction time and can roll the database back to any time. Valid time and transaction time are orthogonal. Furthermore, a bitemporal database that supports both valid time and transaction time is capable of handling retroactive and post-active changes on temporal data. In the literature, the term temporal database is generically used to mean a database with some kind of time support.

In this chapter we focus our discussion on the valid time aspect of temporal data in relational databases. However, our discussion can easily be extended to databases that support transaction time or both as well.

REPRESENTING TEMPORAL DATA

A *temporal atom* is a time stamped value, <*t*, *v*> and represents a temporal value. It asserts that the value *v* is valid over the period of time stamp *t* that can be a time point, an interval, temporal set, or a temporal element. Time points are only suitable for values that are valid at a time point not over a period. Time can be added to tuples or attributes and hence, temporal atoms can be incorporated differently into the relational data model. To represent temporal atoms in tuple time stamping, a relation is augmented with two attributes that represents the end points of an interval or a time column whose domain is intervals, temporal sets, or temporal elements (temporally ungrouped). Figure 1 depicts salary (SAL) history of an employee, E1 where intervals or temporal elements are used as time stamps with a time granularity of month/year. Salary is 20K from 1/01 to 5/02 and 8/02 to 6/03. The discontinuity is because the employee quitted at 6/02 and came back at 8/02. The salary is 30K since 6/03. Figure 2 gives the same salary history in attribute time stamping (temporally grouped). An attribute value is a set of temporal atoms. Each relation has only one tuple that carries the entire history. It is also possible to create a separate tuple for each time stamped value (temporal atom) in the history, i.e. three tuples for Figure 2.b (two tuples for Figure 2.c). Temporally grouped data models are more expressive than temporally ungrouped data models but their data structures are also more complex [Clifford, Croker, and Tuzhilin 1993].