Modeling and Querying Temporal Data

Abdullah Uz Tansel[1]
Bilkent University, Turkey

INTRODUCTION

Databases in general 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* has a time dimension and maintains time-varying data (i.e., past, present, and future data). In this article, we focus on the relational data model and address the subtle issues in modeling temporal data, such as comparing database states at two different time points, capturing the periods for concurrent events, and accessing to times beyond these periods, handling multivalued attributes, coalescing, and restructuring temporal data (Gadia 1988, Tansel & Tin, 1997). Many extensions to the relational data model have been proposed for handling temporal data.

There is a growing interest in temporal databases in many application domains. The first book dedicated to temporal databases, by Tansel et al. (1993), was followed by others addressing issues in handling time-varying data (Betini, Jajodia, & Wang, 1988; Date, Darwen, & Lorentzos, 2002; Snodgrass 1999).

BACKGROUND

The set *T* denotes time values, and it is a total order under the "≤"relationship, hence allowing comparisons and calculations. The set *T* can be represented by integers (*I*) or real numbers (*R*). Time is continuous, and real numbers are a better approximation because they can easily accommodate time granularity. Because of its simplicity, time values in a calendar system are commonly implemented by the integers 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, months, years, and so forth (for a formal definition, see Betini, Jajodia, & 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 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*, and so forth. 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 intuitive 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 in the past. Valid time and transaction time are orthogonal. However, a temporal 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.

This chapter will focus on the valid time aspect of temporal data in relational databases. However, the discussion herein can easily also be extended to databases that support transaction time.

MODELING 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, interval, temporal set, or temporal element. Time points are suitable only for values that are valid at a time point, not over a period. Time can be added to tuples or

Figure 1. Salary in tuple time stamping

E#	SAL	FROM	TO
E1	20K	1/01	5/02
E1	20K	8/02	6/03
E1	30K	6/03	*now*

a. Intervals

E#	SAL	TIME
E1	20K	{[1/01, 5/02) ∪ [8/02, 6/03)}
E1	30K	{[6/03, *now*]}

b. Temporal elements

Figure 2. Salary in attribute time stamping

E#	SAL
E1	{<[1/01, 5/02), 20K> <[8/02, 6/03), 20K> <[6/03, *now*], 30K>}

a. Intervals

E#	SAL
E1	{<{[1/01, 5/02) ∪ [8/02, 6/03)}, 20K> <{[6/03, *now*]}, 30K>}

b. Temporal elements

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 represent the end points of an interval or a time column whose domain is intervals, temporal sets, or temporal elements. Figure 1 depicts salary (SAL) history of an employee (E1), in which 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 from 8/02 to 6/03. The discontinuity is the result of the employee quitting on 6/02 and returning on 8/02. The salary is 30K since 6/03. Figure 2 gives the same salary history in attribute time stamping. 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.a, two tuples for Figure 2.b).

One noteworthy aspect of data presented in Figure 2 is that the time stamps are glued to attribute values. In other words, attribute values are temporal atoms. In forming new relations as a result of query expressions, these time stamps stay with the attribute values. On the other hand, in tuple time stamping, a time stamp may be implicit (glued) or explicit (unglued) to tuples. This is a design choice, and the relations in Figure 1 can be interpreted as having implicit or explicit time stamps. An implicit time stamp is not available to the user as a column of a relation, though the user can refer to it. On the other hand, an explicit time stamp is like any other attribute of a relation and it is defined on a time domain. Implicit time stamps restricts the time of a new tuple created from two constituent tuples, since each tuple may not keep its own time stamp and a new time stamp needs to be assigned to the resulting tuple. Explicit time stamps allow multiple time stamps in a tuple. In this case, two tuples may be combined to form a new tuple, each carrying its own time reference. However, the user needs to keep track of these separate time references.

TEMPORAL RELATIONS

Figure 3 shows some sample employee data for the EMP relation over the scheme E# (Employee number), ENAME (Employee name), DNAME (Department name) and SALARY. E# and ENAME are (possibly) constant attributes, whereas DNAME and SALARY change over time. In EMP relation, temporal elements are used in temporal atoms for representing temporal data. Time stamp of E# represents the life span of an employee that is stored in the database. Note that EMP is a nested (NINF—Non-First Noral Form) relation. It is one of the many possible relational representations of the employee data (Clifford & Tansel, 1985; Gadia, 1988; Tansel, 2004). Figure 4 gives, in tuple time stamping, three 1NF relations, EMP_N, EMP_D, and EMP_S for the EMP relation of Figure 3 (Lorentzos & Johnson, 1987; Navathe & Ahmed 1987; Sarda, 1987; Snodgrass 1987). In Figure 3, temporal sets (elements) can also be used as the time reference. Similarly, in the relations of Figure 4, intervals, or temporal sets (elements), can also be used as the time reference in a time attribute that replaces the Start and End columns.

Note that in tuple time stamping, a relation may contain only attributes whose values change at the same time; attributes changing at different times require separate relations. Each particular time stamping method imposes restrictions on the type of base relations allowed as well as the new relations that can be generated from the base relations. The EMP relation in Figure 3 is a unique representation of the employee data, in which each tuple contains the entire history of an employee (Clifford & Tansel, 1985; Gadia, 1988; Tansel, 1997). The E# is a temporal grouping identifier, regardless of the time stamp used (Clifford, Croker, & Tuzhilin, 1993). In the case of tuple time stamping an employee's data is dispersed into several tuples (i.e., there are three salary tuples for employee 121 in Figure 4.c). These tuples belong to the same employee because their E# values are equal.

For the relations in Figures 3 and 4 there are many other possible representations that can be obtained by taking subsets of temporal elements (intervals) and creating several tuples for the same employee. These relations are called *weak relations* (Gadia, 1988). Though they contain the same data as the original relation in unique representation, query specification becomes very complex. Weak relations naturally occur in querying a temporal database. Weak relations can be converted to an equivalent unique relation by coalescing tuples that belong to the same object (employee) into one single tuple (Bohlen, Snodgrass & Soo 1996; Sarda, 1987).

Design of relational databases is based on functional and multivalued dependencies. Roughly, a relation is created to represent the data for similar objects (entities),

Figure 3. The EMP relation in attribute time stamping

E#	ENAME	DNAME	SALARY
<[1/01,*now*], 121>	Tom	<[1/01, 2/02), Sales> <[4/02, 8/02), Mktg>	<[1/01, 5/02), 20K> <[5/02, 7/02), 25K> <[7/02, *now*], 30K>
<[3/03,8/03), 133>	Ann	<[3/03, 8/03), Sales>	<[3/03, 8/03), 35K>
<[8/02, *now*], 147>	John	<[8/02, *now*], Toys>	<[8/02, *now*], 42K>

Figure 4. The EMP relation in tuple time stamping

E#	NAME
121	Tom
133	Ann
147	John

(a) EMP_N Relation

E#	DNAME	START	END
121	Sales	1/01	2/02
121	Marketing	4/02	8/02
133	Sales	3/03	8/03
147	Toys	8/03	*Now*

(b) EMP_D Relation

E#	SALARY	START	END
121	20K	1/01	5/02
121	25K	5/02	7/02
121	30K	7/02	*Now*
133	35K	3/03	8/03
147	42K	1/01	*Now*

(c) EMP_S Relation

such as employees. Moreover, a separate relation is defined for the many-to-many relationships between entities. Any attribute involved in a multivalued dependency is also placed into a separate relation. In temporal databases, functional dependencies are transformed into multivalued dependencies. Therefore, each time a dependent attribute is placed into a separate relation in the case of tuple time stamping as is seen in Figure 4. If attribute time stamping and nested relations are used allof the time dependent attributes that belong to similar entities, such as employees, can be placed in one relation, as seen in Figure 3. Details of designing nested relations based on functional and multivalued dependencies are discussed in Ozsoyoglu and Yuan (1987) and Tansel and Garnett (1989).

CRITICAL ISSUES IN MODELING TEMPORAL DATA

Following are the critical issues in modeling temporal data (Tansel & Tin, 1997). Let D_t denote the database state at time *t*:

1. The data model should be capable of modeling and querying the database at any instance of time (i.e., D_t). Note that when *t* is *now*, D_t corresponds to a traditional database.
2. The data model should be capable of modeling and querying the database at two different time points, intervals, and temporal set (elements) (i.e., D_t and $D_{t'}$ where $t \neq t'$).
3. The data model should allow different periods of existence in attributes within a tuple (i.e., nonhomogenous [heterogeneous] tuples should be allowed). In a homogenous temporal relation, all the attribute values in the tuple should be defined using the same period of time (Gadia, 1988).
4. The data model should allow multivalued attributes at any time point (i.e., in D_t).
5. A temporal query language should have the capability to return the same type of objects on which it operates. This may require coalescing several tuples to obtain the desired result.
6. A temporal query language should have the capability to regroup the temporal data according to a different grouping identifier that could be one of the attributes in a temporal relation.
7. The model should be capable of expressing set-theoretic operations, as well as set comparison tests, on the time stamps, be they time points, intervals, or temporal sets (elements).

Issues 3, 5, 6, and 7 need more elaboration, whereas the rest do not require any further justification. Homogeneity (Gadia, 1988) requires that the attribute values of a tuple should be defined over the same period of time. Implicit time attributes, by definition, require homogenous tuples. Let *r* be a temporal relation and τ and τ' be two of its tuples. Let τ_T and τ'_T be the temporal elements (sets) over which τ and τ' are defined, respectively. Cartesian product of these two tuples can only be defined over $\tau_T \cap \tau'_T$. In other words, from these two tuples, data can be extracted only in the common period defined by $\tau_T \cap \tau'_T$. Parts of τ_T and τ'_T outside of their intersection are not accessible (i.e., $\tau_T - \tau'_T$, or $\tau'_T - \tau_T$). One can set the semantics of a query language to allow a virtual Cartesian product of tuples with different times. Though this allows the interpretation of one single expression, it is not possible to carry the intermediate results from one expression to another.

Ideally, a temporal query language should retrieve relations in unique representation. In case the query language retrieves weak relations, it should have the capability to transform them into equivalent unique relations in unique representations. Consider the scheme of the EMP relation given in Figure 3. Let r_1 be the instance of EMP relation containing Tom's data in the interval [3/01, 10/01), and let r_2 be the relation instance having Tom's data in the interval [6/01, 1/02). Hence, $r_1 \cup r_2$ contains two tuples that can be coalesced into a single tuple for the time period [3/01, 1/102). The former would be a weak relation, whereas the latter would be a unique relation.

The capability to restructure temporal relations should be provided in a temporal query language. In Figure 3, the employee data is grouped with respect to E#. The employee data of Figure 3 can also be grouped with respect to the department values. This will facilitate answering queries that involve the department attribute.

A sample query might be, "give the E#, name, and salary of the employee for each department" or "does the validity period of any department include [3/01, *now*]"?

For issue 7, any data model using temporal sets (elements) as time stamps naturally supports set-theoretic operations and set-comparison tests on time stamps. However, the case of time points and intervals is not straightforward; any data model using them should be able to simulate these operations.

FUTURE TRENDS

Increased hardware and software capacity is paving the way for implementing temporal databases in many application domains. Commercial database vendors will continue adding functionality for temporal support to their software products. Data warehousing capabilities such as maintaining aggregate data are also expected to be added to temporal databases because they contain the source data for these aggregates. However, efficient solution methodologies are needed for the maintenance of the aggregated data.

There are many open issues, especially in the implementation of temporal databases with functionality addressing the issues discussed earlier, such as storage structures, indexing temporal data, and efficient processing of temporal queries.

CONCLUSION

Modeling of temporal data presented in the previous sections also has implications for the temporal query languages that can be used. There are many language proposals for temporal databases (Lorentzos & Mitsopoulos, 1997; Snodgrass, 1987 1995; Tansel, Arkun, & Ozsoyoglu, 1989). In addition to traditional algebraic operations and calculus constructs, a temporal query language should have projection and selection operations on the time dimension. Moreover, in the case of attribute time stamping, restructuring operations, such as *nest* and *unnest*, are also needed (Tansel, 1997). Operations to form new temporal atoms, depending on how they are incorporated into attribute and tuple time stamping, are also needed. SQL2 (Structured Query Language) has a time domain capability for implementing tuple time stamping with intervals. SQL3 has the capabilities for implementing temporal elements as well as for attributing time stamping. However, what temporal constructs should be added to SQL3 is an open issue.

ACKNOWLEDGMENT

Research is supported by the grant# 65406-00-34 from the PSC-CUNY Research Award Program.

REFERENCES

Betini, C., Jajodia, S., & Wang, S. (1988). *Time granularities in databases, data mining and temporal reasoning*. New York: Springer-Verlag.

Bohlen, M. H., Snodgrass, R. T., & Soo, M. D. (1996). Coalescing in temporal databases. *Proceedings of the International Conference on Very Large Databases*, Bombay, India (pp. 180-191).

Clifford, J., Croker, A., & Tuzhilin A. (1993). On completeness of historical data models. *ACM Transactions on Database Systems*, *19*(1), 64-116.

Clifford, J., & Tansel, A. U. (1985). On an algebra for historical relational databases: Two views. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 247-265).

Date, C. D., Darwen, H., & Lorentzos, N. (2003). *Temporal data and the relational data model*. San Francisco: Morgan Kaufmann.

Etzion, O., Jajodia, S., & Sripada, S. (Eds.). (1998). *Temporal databases: Research and practice*. New York: Springer-Verlag.

Gadia, S. K. (1988). A homogeneous relational model and query languages for temporal databases. *ACM Transactions on Database Systems, 13*(4), 418-448.

Lorentzos, N. A., & Johnson, R. G. (1988). Extending relational algebra to manipulate temporal data. *Information Systems, 13*(3), 289-296.

Lorentzos, N. A., & Mitsopoulos, Y. G. (1997). SQL extension for interval data. *IEEE Transactions on Knowledge and Data Engineering, 9*(3), 480-499.

Navathe, S. B., & Ahmed, R. (1987). TSQL-A language interface for history databases. *Proceedings of the Conference on Temporal Aspects in Information Systems* (pp. 113-128).

Ozsoyoglu, M. Z., & Yuan, L. -Y. (1987). A new normal form for nested relations. *ACM Transactions on Database Systems*, 12(1), 111-136.

Sarda, N. L. (1987). Extensions to SQL for historical databases. *IEEE Transactions on Software Engineering, 12*(2), 247-298.

Snodgrass, R. T. (1987). The temporal query language Tquel. *ACM Transactions on Database Systems, 12*(2), 247-298.

Snodgrass, R. T. (1995). *The TSQL2 temporal query language*. Norwell, MA: Kluwer Academic.

Snodgrass, R. T. (1999). *Developing time oriented applications in SQL*. San Francisco: Morgan Kaufmann.

Tansel, A. U. (1986). Adding time dimension to relational model and extending relational algebra. *Information Systems, 11*(4), 343-355.

Tansel, A. U. (1997). Temporal relational data model. *IEEE Transactions on Knowledge and Database Engineering, 9*(3), 464-479.

Tansel, A. U. (2004). On handling time-varying data in the relational databases. *Journal of Information and Software Technology, 46*(2), 119-126.

Tansel, A. U., Arkun, M. E., & Ozsoyoglu, G. (1989). Time-by-example query language for historical databases. *IEEE Transactions on Software Engineering, 15*(4), 464-478.

Tansel, A. U., Clifford, J., Gadia, S. H., Jajodia, S., Segev, A., & Snodgrass, R. T. (Eds.). (1993). *Temporal databases: Theory, design and implementation.* Redwood City, CA: Benjamin Cummings.

Tansel, A. U., & Garnett, L. (1989). Nested temporal relations. *Proceedings of ACM SIGMOD International Conference on Management of Data* (pp. 284-293).

Tansel, A. U., & Tin, E. (1997). Expressive power of temporal relational query languages. *IEEE Transactions on Knowledge and Data Engineering, 9*(1), 120-134.

KEY TERMS

For a detailed coverage of the terminology see (Tansel et al., 1993, appendix A; and Etzion, Jajodia & Sripada, 1998, pp. 367-413).

Coalescing: Combining tuples whose times are contiguous or overlapping into one tuple whose time reference includes the time of constituent tuples.

Homogenous Temporal Relation: Attribute values in any tuple of a relation are all defined with the same period of time. In a heterogeneous relation, attribute values in a tuple may have different time periods of existence.

Rollback: Restoring the database (a temporal relation) to a state that is recorded as of a given time point, interval, or temporal element in a database that supports transaction time.

Temporal Data Model: A data model with constructs and operations to capture and manipulate temporal data.

Temporal Database: A database that has transaction time support, valid time support, or both. In the literature, it is loosely used to mean a database that has some kind of time support.

Temporal Element: Union of maximal time intervals in which no two time intervals overlap or meet.

Time Granularity: Unit of time, such as seconds, minutes, hours, days, months, years. Time advances by each clock tick according to the granularity used.

Time Interval (Period): The consecutive set of time points between a lower bound (l) and an upper bound (u) where $l < u$. The closed interval [l, u] includes l and u whereas the open interval (l, u) does not include l and u. Half-open intervals, [l, u) or (l, u] are analogously defined.

Transaction Time: Designates the time when data values are recorded in the database.

Valid Time: Designates when data values become valid.

ENDNOTE

[1] On leave from Baruch College, the City University of New York, USA.