LBL-26080
Preprint

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

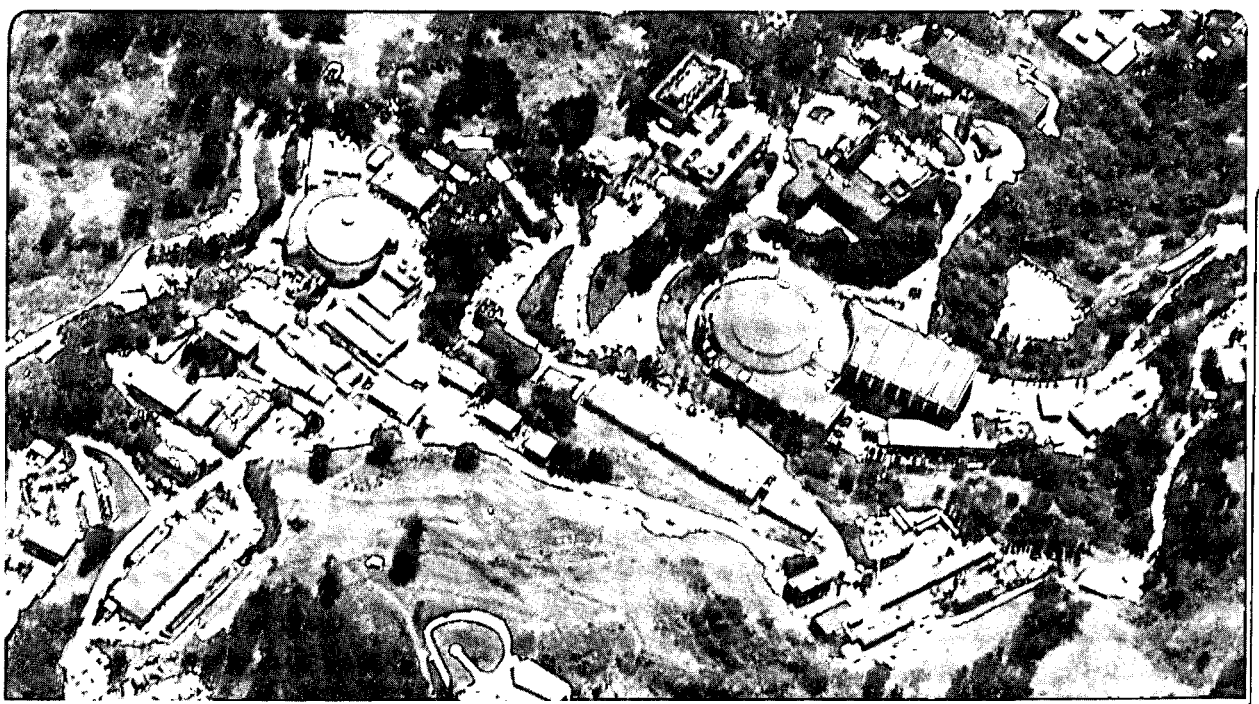
Information and Computing Sciences Division

Published in Database Engineering, IEEE Computer Society Quarterly Bulletin, Decemeber 1988, Vol. 11, No. 4

Functionality of Temporal Data Models and Physical Design Implications

A. Segev and A. Shoshani

September 1989



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

DISCLAIMER

LBL-26080

FUNCTIONALITY OF TEMPORAL DATA MODELS AND PHYSICAL DESIGN IMPLICATIONS

Arie Segev* and Arie Shoshani

Computing Science Research & Development
Information & Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, California 94720

and

*School of Business Administration
University of California, Berkeley

September 1988

Published in IEEE Computer Society special issue on Temporal Databases,
December 1988, Volume 11, No. 4

This work was supported by the Director, Office of Energy Research, Applied Mathematical Sciences Research Program, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

FUNCTIONALITY OF TEMPORAL DATA MODELS AND PHYSICAL DESIGN IMPLICATIONS

Arie Segev† and Arie Shoshani‡

† School of Business Administration and
Information Computing Sciences Division
Lawrence Berkeley Laboratory
The University of California
Berkeley, California 94720

‡ Information Computing Sciences Division
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

Abstract. In previous work, we introduced a data model and a query language for temporal data. The model was designed independently of any existing data model rather than an extension of one. This approach provided an insight into the special requirements for handling temporal data. In this paper, we discuss the main functionality of the model and the implications on physical design. We feel that the functionality and implementation issues discussed in this paper are applicable to all temporal models.

1. INTRODUCTION

In previous papers [Shoshani & Kawagoe 86, Segev & Shoshani 87] we have developed a temporal data model that is independent of existing data models, such as the relational or network models. Our approach differs from many other works [Ariav 86, Clifford & Croker 87, Gadia & Yeung 88, Klopproge 81, Navathe & Ahmed 86, Snodgrass 87, Tansel 86] that extend existing models to support temporal data. Our goal was to design a model (called a *TSC* model for reason that will become clear below) which reflects the semantics and operators of temporal data without being influenced by existing models. Once this model was developed, then it is possible to investigate the incorporation of its structures and operations into existing models. We have performed a requirements analysis for representing the *TSC* model in the context of the relational model [Segev & Shoshani 88]. We showed that the relational model is not sufficient to represent all temporal data and has

This research was supported by the U.S. Department of Energy Applied Mathematics Sciences Research Program of the Office of Energy Research under contract DE-AC03-76SF00098.

to be extended with the construct of a temporal relation having new semantic properties. We have also discussed the options for the representation of a temporal relation and the reasons for our preferred representation. In this paper we highlight the functionality of the *TSC* model, and discuss its physical design implications. Note that our reference list is by no means complete; for a complete list, see the bibliography by Stam & Snodgrass in this issue.

2. THE TEMPORAL DATA MODEL

The *TSC* model is based on the simple observation that temporal values are associated with a specific object, and are totally ordered in time; i.e. they form an ordered sequence. For example, the salary history of some individual forms an ordered sequence in the time domain. Accordingly, we introduce the concept of a *time sequence* (*TS*) for a single object (or entity). Further, the set of *TSs* for an object set forms the basic construct of our model which we call a *time sequence collection* (*TSC*). For example, the set of all salary time sequences for a set of employees forms the salary *TSC*. Simple *TSCs* are constructs that have a surrogate, time, and attribute associated with them, denoted as a triplet (S, T, A). For the salary history of employees, this triplet may be (employee_id, month_year, salary). Complex *TSCs* allows multiple attributes to be associated with the same surrogate and time pairs, and are denoted as $(S, T, \overline{A})$. Complex *TSCs* are usually useful for representing synchronous temporal attributes, such as multiple measurements taken at the same time (e.g., air quality measurements: nitrogen, carbon dioxide, etc.)

Naturally, a *TSC* can be accessed on any combination of its surrogate, time, and attribute values. For example, "find the employees that had a salary more than 30k in July 1988" is a query that accesses both the time and attribute components of the salary *TSC*. As will be discussed below, there are some specific assumptions that can be made regarding the access patterns of temporal data which greatly influence the choice of physical structures.

One of the essential conclusions that was reached in our previous work is that *TSCs* should have certain meta-data properties. These properties characterize the semantics and the interpretation of the *TSC* values, and have a profound effect on physical data structures chosen to implement a *TSC*. Below is a brief description of these properties.

2.1. *TSC* properties

There are four properties of interest [Segev & Shoshani 87]:

(1) *Time granularity*: this property specifies the points in time that can potentially have data values. For example, if salaries are assigned at increments of months, then the

salary *TSC* granularity is months. Note that time points do not necessarily have values (e.g., salaries do not usually change every month for an individual.)

(2) *Life span*: the life span specifies the range of valid time points of a *TSC*. The life span implies that some physical mechanism is necessary in order to distinguish between existence and non-existence of time points in the *TSC*. The life span definition can be fixed or vary over time. In the latter case, the life span is either extended continuously to "current-time" or it forms a "moving-window".

(3) *Type*: the type of a *TSC* can be thought of as specifying the behavior of time sequences in the *TSC*. For example, a type "step-wise constant" (SWC) can be associated with the salary *TSC*, to mean that the values for time points that do not have explicit data values are determined from the last data value. Other useful types are "discrete" and "continuous", which carry the obvious meaning. A user defined type is also allowed.

(4) *Interpolation rule*: this property is associated with the type. For the SWC type, the rule is as was described above. For the discrete type, the rule is simply that missing values cannot be interpolated. For the continuous type, there is a default continuous function to interpolate values. For the user defined type, an interpolation rule must be supplied by the user.

(5) *Regularity*: A *TSC* is regular if for each time point of the given granularity, a data value has been provided, that is, there is no need to interpolate. This property also provide semantic information to the user; for example, one may want to apply certain statistical analysis only to collected data rather than to interpolated data.

The physical implication of the above properties is that we need to support the interpolation rule according to the *TSC*'s type and granularity. In non-temporal systems the lack of an entry (e.g., there is no entry for some project and some part) implies that the information does not exist. In contrast, in order to support a *TSC* structure, the lack of an entry requires the interpolation of the value. For example, if there is no entry for the salary of John in June 1988, it has to be inferred. As will be discussed later, this requirement affects directly the indexing structures.

2.2. Time points, event points, and change points

Recall that we described time points as points that can potentially have a data value. Time points are defined by the granularity of a *TSC*. As mentioned previously, a time point may have an *explicit* value associated with it, may have an *implicit* value determined by the

interpolation rule, or may have no value (*null*). We refer to time points that have explicit data values as *event points*. For example, the time points where a new salary is assigned are considered event points. Clearly, the event points form a subset of the time points. Usually, one would want to store only the event points, and interpolate the values of the other time points when necessary. However, we may wish to interpolate values ahead of time depending on the density of event points, where *event density* is defined as the ratio of event points to time points. In the case of the regular *TSCs* (mentioned above) the event density is 1. As the event density approaches 1, we may prefer to fully populate the *TSC* in order to use simpler data structures and indexing methods. These issues are discussed further in Section 3.

Now, suppose that consecutive event points have the same values. As can often occur with measurement data or statistical data (e.g., consecutive temperature values may be the same). In principle, we could apply a compression operator, so that all repeating "duplicate" values are removed, thus achieving better storage utilization. We call the subset of event points that is left *change points*. One should be careful with compressing out "duplicate values" because it can result in loss of information. For example, two consecutive salary event points may signify "no raise", and the removal of the second event point (which is a duplicate value) would result in the loss of this information. Thus, a compression of duplicates should be a user selected parameter, as it depends on the semantics of the application.

Change points are also important as part of a predicate over time sequences. For example, one may be interested only in the times that employees change departments. In general, all three predicate conditions for finding time, event, and change points should be supported by physical structures.

2.3. Temporal normal forms

In [Segev & Shoshani 88] we discuss the issue of temporal normal forms in the context of a relational representation of the *TSC* model. We argue there that it is appropriate to impose the restriction on *TSCs* that any time slice (at some time point t) would result in a standard First Normal Form relation. We call this condition a First Temporal Normal Form (or 1TNF). The practical implication is that for each instance of a surrogate and time point, each attribute has a single value (possibly null). While this requirement seems obvious, its enforcement may not be trivial depending on the physical structure chosen for the implementation of a *TSC*. We elaborate on this point in Section 3. A definition of another temporal normal form was proposed by [Navathe & Ahmed 86]. The logical implications of that

definition are discussed in [Segev & Shoshani 88].

2.4. Operators

In [Segev & Shoshani 87] we discussed at some length the operators that should be applied to *TSC*s. They include various temporal predicates, aggregate functions in several dimensions, accumulation operators (such as a running average), and operators between multiple *TSCs*. The physical support of such operators is quite complex and is still under study. However, to illustrate the additional complexity that a temporal model introduces we discuss briefly below the implication of the aggregate functions.

Operations over *TSCs* produce a new *TSC*. In particular, operators involving aggregate functions require the determination of a new granularity for the resulting *TSC*, and often a new type as well. For example, we may wish to aggregate over a *TSC* which represents deposits and withdrawals in some bank account, in order to generate a *TSC* for the running balance. The resulting *TSC* will have a type SWC while the original *TSC* is of type discrete. Alternately, getting a monthly sales figure from a daily sales *TSC*, changes from one granularity to another. A more difficult problem arises when we need to decrease the level of time granularity (e.g., estimate daily figures from monthly figures, assuming we know daily patterns). This problem is referred to as the "disaggregation problem" by statisticians. Our work on these problems is in progress, and its description is beyond the scope of this paper.

3. PHYSICAL DESIGN IMPLICATIONS

The problem of physical design of temporal databases still deserves a significant attention. In this section we discuss physical design issues, some of which has been addressed by works such as [Ahn & Snodgrass 86, Gunadhi & Segev 88a, Lum et al 84, Rotem & Segev 87]. At an abstract level, physical design of temporal databases is similar to conventional database design in the sense that the desirable structure is a function of application requirements. There are, however, several important differences. For example, in temporal databases, one expects that many queries will need the data ordered by the time attribute; also, the data itself is likely to be either static or append-only. We will first classify elements of the environment that affect the design, and list the major physical design choices. We will then discuss some of the combinations of environmental parameters and physical design choices.

3.1. Environmental parameters

The following is a list of the parameters that affect the physical design.

(1) *Static* vs. *dynamic* data. Static data represents non-updatable historical data with a fixed life span, while dynamic data arise in cases where temporal data have a variable life span or a fixed life span with updatable data (i.e., missing or incorrect data). In the case of dynamic historical data, an important aspect is its append-only nature.

(2) *Sparse* vs. *dense* data. We define two measures of density that have physical design implications. The first measure, *event density* (introduced in Section 2), is the ratio of the number of event points to the number of time points. The second measure is *existence density*, defined as the ratio of the number of existence points to the number of time points, where an existence point is a time point that has a non-null attribute value (either explicit or implicit). For a discrete sequence, all existence points are event points, and thus the two density measures are the same. However, for a *SWC* or a general continuous sequence, the two measures will differ if the two sequences contain null values.

(3) *Integration of temporal and non-temporal data.* In the case where such an integration is required, a priority may be given to the processing of non-temporal data and the current portion of the temporal data.

(4) *Query types.* This is the most influential parameter as far as the physical design is concerned. Queries can be classified as follows: i) Single-*TSC* queries; either point or range queries. The qualification part can refer to S, T, or A, or any combination. Note the T qualification can be either a value (e.g, June 3, 1988) or a relative position (e.g., third from the beginning). ii) Multi-*TSC* queries. These queries involve some kind of a join plus any of the qualifications of single-*TSC* queries.

3.2. Physical design options

The physical design options can be classified as follows.

(1) *Record structure.* We consider here the possible options for extending existing relational systems. In order to minimize changes to current systems, we have chosen to have a normalized tuple as the physical data unit.

(2) *Sorting.* A major issue is whether to have the data sorted by time (either physically or logically), and if so, whether the time will be the major sort key or a minor (where the primary order is by surrogate).

(3) *Indexing structures.*

i) *Single attribute indexing.* The two main choices here are hashing (which also determines

the placement of data records) or trees (e.g., B-trees).

ii) *Multi-attribute indexing.* There are several options for providing indexing structures:

a) *Separate structures.* In this case each attribute is indexed separately. A multi-attribute access can be done by intersecting pointers from multiple indices.

b) *Joint structures.* The indexing is done directly on the combination of several attributes.

c) *Multi-level structures.* Separate structures are combined in a hierarchical fashion. For example, the leaves of a B+ tree index of one attribute point to a B+ tree index of another attribute.

The above structures have one or more component. In general, each component can be one of two indexing types: either a tree (e.g., R-trees) or a non-tree (e.g., grid files).

The foregoing taxonomy of environmental parameters and physical structures, though general, serves as a helpful framework for investigating the physical design problem. In this paper, we discuss only some the issues; the complete discussion is given in [Gunadhi & Segev 88a and 88b]. It should be noted that we are interested in databases containing a large number of history records. Below is a summary of our conclusions.

3.3. Query types

We anticipate that most of the temporal queries will qualify on S and T (i.e., data records for surrogates at specific time points); if T is absent from the qualification, it means that the query needs the whole history. Moreover, we don't anticipate frequent range qualifications on S. A qualification on A is likely to be combined with either a T or ST qualification (e.g., the data of employees who earned more than 30K at a certain time point). Consequently, range queries are more likely to qualify on T and/or A. In databases which are event-oriented, queries that qualify only on T (i.e., what happened at certain time points?) are more likely; in our analysis, however, we assume that these queries are secondary in importance.

The above discussion implies that we are interested in supporting primarily queries that qualify on ST (note that this is the primary key of the data tuples), or TA, or STA.

3.4. Ordering of the data records

Given the above query types we exclude a primary order of data records by time, but rather have the data sequenced primarily by surrogate and secondarily by time. If the data is static, physical clustering is the best choice; if it is dynamic, an indexing structure may be required to enable access to data in the desirable order.

Note that if the query type do not qualify on the surrogate (i.e., *T* or *TA* qualifications), then a primary order by time is desirable. Furthermore, in this case, since the data is append-only we can keep it in contiguous blocks (a clustered index) with a fill factor of (or approximately of) 100%.

3.5. A static environment

If the data is static, dense, and without many contiguous duplicate values, then the optimal solution is to have it organized as a two dimensional array linearized row-wise (rows and columns represent *S* and *T* respectively). An index on *A* may still be required. If the data is sparse, a direct access by *ST* cannot be calculated, and an index is required (we exclude hashing on *ST* because of the high likelihood of range qualifications on *T*). In the case where there are many contiguous duplicate values, compression is useful. It should be noted, however, that the compression scheme should preserve the distinction between event points and interpolated points. The only exception to this is a regular *TSC* where we know that all points are event points.

3.6. A dynamic environment

Given that we are not interested in *T* as the primary ordering attribute, physical clustering in order of *ST* may cause a problem. If the database is append-only and the rate of growth is low, an indexed-sequential organization is appropriate. If the rate of growth is high, long overflow chains my result, and an unclustered index should be used.

3.7. Indexing

The most important environmental parameter that affects indexing is the query types. Our current conclusions (the details are given in [Gunadhi & Segev 88a and 88b]) are the following:

(1) For a large number of surrogates and a significant number of queries that qualify on a single *S*, we excluded a grid type file for *ST* or *STA* dimensions. The reason is that there will be a partitioning line for each surrogate value, and if the partitioning information fits in main memory, it implies that we can also store the multi-level

S/T index trees in main memory. A grid type file can be constructed on the *ST* or *STA* dimensions if there are range queries on *S* and queries that qualify on a single *S* can be compromised. Grid file types are a viable option on the *TA* dimensions (e.g., the brick file in [Rotem & Segev 87]).

(2) For *ST* qualifications we prefer to have a multi-level B+ tree index, where the leaves of the *S* tree point to *T* trees. This choice enables integration of historical data with current and non-temporal data such that updates to the latter are separable from the temporal structures. In addition, the total storage cost is frequently lower for the multi-level structure.

(3) It is inefficient to have time index on points other than event points (in the case of a *SWC* sequence, indexing event points is equivalent to indexing time intervals, where the terminal points of an interval are event points). The reason is that we can interpolate in order to find values at points other than event points.

(4) For a *TSC* with a high existence density, there is no point in having a separate *T* index (in the case of a multi-level index a *T* index under an *A* or *S* index can still be useful). The reason for the above observation is that in a *T* index with high existence density, a *T* value of the index will point to data records in a non-selective way. The extreme case is when the the density is 1, and thus for each time point in the lifespan all the surrogates have a value.

3.8. 1TNF Enforcement

As was discussed in Section 2, enforcing 1TNF of a *TSC* implies that only one attribute value is allowed for a given combination of *S* and *T* values.† The mechanism for enforcing 1TNF is dependent on the record structure. For example, there are proposals that claim that it is more desirable for access efficiency to store time sequences as tuples containing intervals. If such a structure is chosen, then a mechanism has to be added to ensure that intervals of the same surrogate do not overlap, so that the 1TNF condition is not violated. On the other hand, if one choses to store tuples with time points rather than intervals, then this condition can be simply enforced by existing mechanisms of the relational model, i.e. by defining the surrogate and the time as a composite key.

† In the case of $\overline{A}$, only a single combination of $\overline{A}$ values is allowed for a given combination of S and T values.

REFERENCES

[Ahn & Snodgrass 86]

Ahn I., Snodgrass R., Performance Evaluation of a Temporal Database Management System, *Proceedings of the ACM SIGMOD International Conference on Management of Data,* 1986, pp. 96-107.

[Ariav 86]

Ariav, G., A Temporally Oriented Data Model, *ACM Transactions on Database Systems,* 11, 4(Dec. 1986), pp. 499-527.

[Clifford & Croker 87]

Clifford, J., Croker, A., The Historical Relational Data Model (HRDM) and Algebra Based on Life Spans, *Proceedings of the Third International Conference on Data Engineering,* pp. 528-537. February, 1987.

[Gadia & Yeung 88]

Gadia, S.K., Yeung C-S., A Generalized Model for A Relational Temporal Database, *Proceedings of the ACM SIGMOD International Conference on Management of Data,* June 1988, pp. 251-259. 1986, pp. 390-397.

[Gunadhi & Segev 88a]

Gunadhi H., Segev A., Physical Design of Temporal Databases, *Lawrence Berkeley Lab* Technical Report LBL-24578, 1988.

[Gunadhi & Segev 88b]

Gunadhi H., Segev A., Indexing Structures for Temporal Databases, *Lawrence Berkeley Lab* Technical Report. In progress.

[Lum et al 84]

Lum, V., Dadam, P., Erbe, R., Guenauer, J., Pistor, P., Walch, G., Werner, H., Woodfill, J., Designing DBMS Support for the Temporal Dimension, *Proceedings of the ACM SIGMOD International Conference on Management of Data,* June 1984, pp. 115-130. March, 1986.

[Navathe & Ahmed 86]

Navathe, S.B., Ahmed, R., A Temporal Relational Model and a Query Language, *UF-CIS Tech. Report TR-85-16, Univ of Florida,* April 1986.

[Rotem & Segev 87]

Rotem, D., Segev, A., Physical Organization of Temporal Data, *Proceedings of the Third International Conference on Data Engineering,* pp. 547-553. February,

1987.

[Segev & Shoshani 87]

Segev, A., Shoshani, A., Logical Modeling of Temporal Databases, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, May 1987, pp. 454-466.

[Segev & Shoshani 88]

Segev, A., Shoshani, A., The Representation of a Temporal Model in the Relational Environment, an invited paper to the *4th International Conference on Statistical and Scientific Database Management*, Rome, June 1988.

[Shoshani & Kawagoe 86]

Shoshani, A., Kawagoe, K., Temporal Data Management, *Proceedings of the International Conference on Very Large Databases*, August 1986, pp. 79-88.

[Snodgrass 87]

Snodgrass, R., The Temporal Query Language TQuel *ACM Transactions on Database Systems*, 12, 2, June 1987, pp. 247-298.

[Tansel 86]

Tansel, A.U., Adding Time Dimension to Relational Model and Extending Relational Algebra, *Information Systems*, 11, 4 (1986), pp. 343-355. *ACM Transactions on Database Systems*, 12, 2, June 1987, pp. 247-298.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
1 CYCLOTRON ROAD
BERKELEY, CALIFORNIA 94720