Temporal Properties in Object Modeling and their Implementation in Relational Databases

Jonas Skučas, Algimantas Juozapavičius
Vilnius University, Lithuania
{jonas.skuicas, algimantas.juozapavicius}@maf.vu.lt

Abstract
Modeling is an essential component of information systems design and analysis. Nevertheless, many modeling methods and methodologies evolve from experience, and usually give no support to such essential features as historic information in databases. We propose the theoretical concepts and empirical considerations how to involve temporal properties of data into object modeling and how to implement them in relational databases, that correspond to the SQL-2 standard. Temporal properties of the elements of object model are discussed, following the need to achieve the support of various manipulations on time-varying information within contemporary information systems. The general dynamic model for temporal objects suggested here are augmented by the consideration of the mapping of objects to relational database, and the temporal properties of such mappings are discussed. As a consequence, the denormalization procedures of data schema to support the most common types of temporal queries (queries over current state and time slice queries) are suggested.

Keywords: information systems and databases, object modeling, historic information, temporal data, temporal properties of attributes, dynamic model of temporal objects.

1. Introduction

The subject of the paper is to discuss conceptualization, design and implementation of temporal properties in the modeling methodology of information systems. The framework specific for the modeling investigated is object orientation, and the implementation of temporal properties is considered in relational databases that correspond to the SQL-92 standard.

Databases that store time varying information are called temporal databases. Specifically, the word “temporal” means data varying in time, and it doesn’t mean temporally used data, as do many procedures of data manipulation in databases. As a consequence, the important aspect of databases appear, a “temporal” one:

- how to find out the entire history of data changes of data; in other words – how to get the ability to restore the constitution of the data at any given moment of time.

While tracking (or tracing) the history of data changes in information system based on relational database, a number of problems appear:

- how to model the temporal aspects of data?
- how to implement these temporal aspects in database?
- how to search efficiently temporal data stored in the database?
- how to insure the temporal integrity of data?
- how the operations modifying data could support the integrity of the data stored in database?
- how the interface of temporal applications could be designed in a sound and metaphorical way?

Many real world information systems experience temporal data (also known in the literature as time-referenced one), just to mention financial, medical or travel applications. The temporal aspects of data in databases were explored for the first time more than two decades ago [1, 2]. Many active researchers contributed to this problem, introducing various tools or suggesting different approaches (special types of data structures, semantic models with time component, extended
relational algebra, conceptual models, temporal query languages and their implementations, specifications of interval data, rewrite grammars, other procedures) [2–11]. The theory and practice (especially) of temporal databases however appeared much more complicated, and a lot of questions remains open till now [12].

Among the most essential origins of the problems in temporal databases are:

- the inherent complexity of temporal relations of the data,
- the lack of support for the temporal data and queries in the SQL language (in fact, recent SQL-92 standard or its commercial implementations, gives no support, say for such temporal data type as PERIOD).

The nascent SQL/3 standard includes the component "SQL/Temporal", addressing time varying data problems. Also, built-in temporal support offered by current database products is limited to predefined, time-related data types, e.g. the Informix Time Series DataBlade and the Oracle/8 Time Series Cartridge, and extensibility facilities that enable the user to define new, e.g. temporal, data types. However temporal support is needed that goes beyond data types and extends the query language itself.

The temporal aspects of the information are important for most database applications. For example: the system of personal management should store the information when the employee started to work at the company and when and what incumbents he or she had. Accounting system stores the information about the balance changes in the clients account, and it is obvious that to capture only the last or current state of information is not sufficient.

So, the lack of the standard and convenient methodology, for the design and implementation of temporal databases still exists [12, 13]. Quite common is the case when designers creating a conceptual model of the data, are used to ignore the temporal aspects of data in their solutions, and just supplement the diagrams with comments like this “the temporal support is needed”. The attempt to specify all temporal aspects of the information by the means of usual design notation leads to cluttered diagrams, instead of simple and intuitive ones. Programmers are mapping the temporal properties of the data to relational schemas by hand and somehow. This takes a lot of time and leads to a great number of errors.

Furthermore, if a transition from the conceptual model diagrams to database schemas is not well documented, it is difficult to maintain the database.

The subject of this article is to introduce new methodology and features of temporal aspects, adding them to the existing concepts and methods. The approach covers the conceptualization, design and implementation of temporal properties in the modeling methodology of information systems. The framework of modeling considered in the article, is object orientation (the most common one recently), and the implementation of temporal properties is investigated with respect to relational databases corresponding to the SQL-92 standard. The approach suggested in the article follows the discussion started in [14].

2. Temporal Properties of Elements of the Object Model

The rapid development of the object-oriented software motivates the need for building object-oriented applications in the relational database. The paradigm of object orientation gives a lot of convenient and natural tools to the user to express the real world features. These triggers also the need for mapping object to relational database. The temporal aspects of such mapping are introduced and described in the subject of this article.

Before getting into detailed discussion how to capture temporal aspects of data in the object diagrams and how to map temporal properties of objects into the relational data schemas, there is a need to introduce explicite meaning of main terms of temporal database (tab. 1). Recently, there are
five semantically different aspects of time mostly used in relation to temporal information (more formal definitions may be found in [4]):

Table 1. The definition of main terms of the temporal database.

<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mini-world</strong></td>
<td><em>Mini-world</em> reflects the part of the reality the database stores information about (example: the <em>Mini-world</em> of University Information System covers students, teachers and courses, as well as relationships between them and various constraints).</td>
</tr>
<tr>
<td><strong>Valid time</strong></td>
<td>The time when the fact was true in a mini-world (example: Ann worked at the IBM from 1995 till 1999: the valid time of the fact “Ann works at IBM” is the period of time [1995, 1999]). <em>Valid time</em> is usually supplied by the user.</td>
</tr>
<tr>
<td><strong>Life span time</strong></td>
<td>The time when the entity exists in the mini-world: it’s the time that the fact “Entity exists in mini world” is true. <em>Life span time</em> is usually supplied by the user.</td>
</tr>
<tr>
<td><strong>User-defined time</strong></td>
<td>User interpreted values. No temporal impact for other attribute values (example: birth date).</td>
</tr>
<tr>
<td><strong>Transaction time</strong></td>
<td>The time when fact is current in the database and could be retrieved. Also means the time thought the fact is true (example: Ann loaned videotape to the friend of her on Friday and hoped to get it back on Saturday. On Saturday it becomes known that a tape will be given back on Sunday. The fact that type was loaned for one day was current on Friday. It was changed by the fact that type was loaned for two days on Saturday). <em>Transaction time</em> is usually generated by the system clock.</td>
</tr>
<tr>
<td><strong>Bitemporal time</strong></td>
<td>Bitemporal fact has associated two types of time: valid time and transaction time or life span time and transaction time.</td>
</tr>
</tbody>
</table>

The definition of temporal database could be introduced now as follows:

*A temporal database is a database that supports some aspects of time, others as user-defined time.*

To illustrate the main ideas to be discussed later and to specify modeling features in details, the running example is introduced (fig. 1):

`Figure 1`. Cars’ registration database (current state)
The object diagram shows the conceptual data model of car registration database, and this model is designed to represent only the last (or current) state of the data.

Nevertheless, the database needs history information available while storing data about registered cars and their owners. A car has registration number and an owner. The owner of the car must register it in the police, and the registration number is assigned to the car. If the owner of the car changes, the car must be reregistered. The date of registration, engine number, body number, color, year of manufacture and total weight are recorded for the car. The engine number, body number and color can change because of various reasons, for example, the car can be repainted after an accident. The history of such changes should be stored in the database, and for each given moment of time one should be able to obtain the data of the car for that moment. Time period from the date of the first registration of a car till the date of write off of a car, is treated as the period of existence of the car.

The social security number, name, second name, date of birth and address are recorded for each owner of a car in the database. The social security number and the date of birth can not change in time. The name, second name and address of the owner can change in time and the possibility should be provided to track these changes.

The list of possible models of cars is stored in the database. One model is assigned to each car. The model has the time of existence, and the name of the model can not change in time. The model also identifies the type of the car, which is permanent.

The next diagram (fig. 2) shows, how the attempt to capture all temporal aspects of information, directly in the diagram, may clatter it.

![Object Diagram](image)

**Figure 2.** Cars’ registration database (direct implementation)

Static properties of the database conceptual model are shown in the object model (tab. 2), which is the most important view of the database application.

This table summarizes the temporal support of the elements of object model. The temporal properties could be or could not be assigned to main elements of the object model (in the first column). The assignment of temporal properties to the elements of object model was done following Jensen and Gregersen work on TimeER notation [13].
Table 2. Static attributes of object model

<table>
<thead>
<tr>
<th>Elements</th>
<th>Non-temporal</th>
<th>Life span</th>
<th>Valid time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes (objects)</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Attributes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Superclass/subclass associations</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Associations-attributes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Associations-classes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

In the model, some temporal properties are reasonable for an element and others are not, this follows from the nature of an element.

Classes and objects are treated uniformly because of a class is the identification of the set of objects with the same structure. The presence of the object in a database correspond to the fact that object exists, and the existence time for object is to be registered. Because of the presence of the object implies only one fact “the object exist”, and it's captured by object’s life span time, valid time is meaningless.

Attributes can’t exist separately from the object, so there is no need to support the life span of attribute. Attributes implies some facts that can be true or not at some point in time, so the possibility to register valid time for attribute is needed. Furthermore, the valid time of attribute should be the subset of life span of object, and the attribute should have the value for every moment of object's life span.

The paradigm of object oriented modeling usually considers the superclass/subclass associations as changeless in time, so there is no need to support life span time or valid one. Nevertheless, the inheritance principle states that temporal properties of superclass can not be changed in subclass, they only can be added. If superclass has no life span support it can be added to the subclass. But if superclass registers the existence time then the subclass also must have support for life span. The same holds for the inheritance of temporal properties of attributes.

Associations can have the semantics of the class or attribute, and because of this they may inherit temporal properties of the class or attribute. Associations should satisfy the number of temporal referential integrity constraints (for example, the valid time of the relation-attribute should be the subset of intersection of life span times of classes that participate in relation).

The temporal variability of objects in the mini-world can be captured using times in different granularities (for example, the changes of salary of employee could be captured with precision of one day). So there is a need to provide the possibility to assign different time granularities for the elements of the object model. The model of the running example, cars’ registration database, after the assignment of temporal properties (life span and valid time), may be expressed as in fig. 3:

![Figure 3. Cars’ registration database (temporal notation)](image-url)
3. The Dynamic Model of a Temporal Object (State Diagram)

The dynamic model, which is the development of the ideas exposed in [14], reflects the time-dependent behavior of the object and consists of the specific set of object’s state diagrams. This model explains how the states of the object changes in time, according to the events or messages it receives. Events or messages are to be implemented by specific operations. It means all the modifications of the state of the object are to be done using the operations object provides. Operations in turn, must insure data integrity, including temporal integrity, in the database. Therefore, direct modification of data by the statements of INSERT, DELETE and UPDATE is undesirable, because of it is difficult to insure data integrity by using table constraints and/or triggers. That is why so important to provide simple, but yet flexible and powerful set of operations to manipulate temporal objects.

The complete set of operations applicable to the temporal object, is shown in the diagram (fig. 4). All these operations are needed for the objects with full set of temporal properties:

- life span time is supported by object,
- the object has attributes supporting valid times.

![Figure 4. The dynamic model of the temporal object.](image)

The tab. 3 presents the meaning of a state in the object’s state diagram:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black dot)</td>
<td>Black dot represents the state of the object before its creation, the object is not stored in the database.</td>
</tr>
<tr>
<td>exist</td>
<td>The object is stored in the database, and if it supports life span property, the beginning of the life span period is known.</td>
</tr>
<tr>
<td>not exist</td>
<td>The object is stored in the database, and the end of life span period is known (this state is impossible for the objects without life span property).</td>
</tr>
<tr>
<td>Black dot in the circle)</td>
<td>Black dot in the circle represents the state of the object after it’s physical deletion, the object is no longer stored in the database.</td>
</tr>
</tbody>
</table>

The operations have the following meaning (tab. 4), while changing the states within dynamic model:
Table 4. The description of operations changing the state of an object.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INS</td>
<td>INSert: creates an object in the database.</td>
</tr>
<tr>
<td>UPDT</td>
<td>UPDate technically: it updates object's attribute values without changing valid time period.</td>
</tr>
<tr>
<td>DELT</td>
<td>DELete technically: it physically deletes object from the database.</td>
</tr>
<tr>
<td>DELL</td>
<td>DELete logically: it ends life span period of object.</td>
</tr>
<tr>
<td>UNDL</td>
<td>UNDelete logically: it undoes the logical delete of object (opposite to DELL).</td>
</tr>
<tr>
<td>UPDT_XXX</td>
<td>UPDate technically: technical update for the temporal attribute or the group of temporal attributes.</td>
</tr>
<tr>
<td>INSS_XXX</td>
<td>INSert state: creates new state of temporal attribute.</td>
</tr>
<tr>
<td>DELS_XXX</td>
<td>DELete state: deletes the last valid state of temporal attribute or group of temporal attributes.</td>
</tr>
</tbody>
</table>

(here XXX denotes a short name of attribute or a group of attribute).

Objects can be clustered into four different groups, according to the temporal properties they support (tab. 5). Each group has different set of states and operations. The prototypes (the set of formal parameters and their data types) of the operations also differ for each group. The number of operations, needed to implement, depends on the temporal properties supported by the object.

Table 5. The classification of objects with respect to temporal properties.

<table>
<thead>
<tr>
<th>Operations</th>
<th>Non-temporal attributes</th>
<th>Life span attributes</th>
<th>Valid time attributes</th>
<th>Life span, valid time attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>INS</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UPDT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DELT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DELL</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>UNDL</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>UPDT_XXX</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UPDL_XXX</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DELS_XXX</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

The set of operations determines what and when can be, or not, done with the object by the database application programs. The convenience of user interface also depends on this set. Some special operations are included, for the correction of user errors, they are: UPDT, DELT, UNDL, UPDT_XXX, DELS_XXX.

4. A Note on Life Span and Valid Time Implementation

The life span and valid time implementation has to be discussed within the standard of SQL-92. The values of life span and valid time temporal features are related to the data type of period. Period is an anchored duration of time, and has no support in SQL. Usually periods are modelled by using two timestamps: one for the beginning of the period and another for the end of the period. They are usually interpreted as interval data - [closed, opened), which is the most convenient interpretation (because of it allows us to avoid the problem with overlaped ends of periods). Other possibilities are: [closed, closed], (opened, closed).

To indicate the fact which is valid now, a special value is needed: FOREWER. FOREWER means the instance of time large than any other possible instance of time in the mini-world (for example FOREVER could be implemented as the date of 3000-01-01 (such value of date is more
suitable because of other suggestions like 9999-01-01, are not supported by commercial products). Similarly a special instance of time can be used to indicate the unknown beginning of the period (like BEGINNING: 1000-01-01).

Now the mapping of temporal object diagrams to relational schemas can be produced in the way like this below:

- car(oid, lss, lse, registration_num, owner_id f.k., engine_num, body_num, color, manufacture_year, total_weight, model_id f.k.)
- car_registration(sid, oid f.k., vts, vte, registration_num, owner_id f.k.)
- car_engine(sid, oid f.k., vts, vte, engine_num)
- car_body(sid, oid f.k., vts, vte, engine_body)
- car_color(sid, oid f.k., vts, vte, color)

where: oid – unique object identifier; sid – unique state identifier (surrogate values, usually automatically generated unique integer values); vts – valid time start; vte – valid time end.

Classes/objects are implemented as tables in relational database. One of the tables stores current (last) state of object of class, as well as the period of object's life span, this table is called the class table.

Attributes are implemented in a regular way, as the attributes in relational tables. Current (last) value of attribute of object id are stored in the class table. One separate table for each temporal attribute is added to the database schema, this table stores one row for each valid time period. Current state values are stored in class table and states table both, and they are duplicated.

Composite attributes or the group of attributes that changes in time simultaneously are stored in one table which attributes one row for each valid time period. Key attributes are implemented as additional attributes over the domain of surrogates. This insures primary key integrity. Time instance FOREVER is used to indicate that the period is currently valid.

Associations-attributes are implemented as attributes, and associations-classes are implemented as classes. Referenced table is always a class table, and referencing table can be a class table or a states one. Temporal referential integrity constraints should be implemented in the operations used to manipulate the objects.

The special attention is necessary for normalization (versus denormalization). The level of data schema normalization by means of the 1st, 2nd and 3rd normal forms shows the quality of design of ordinary database used for transaction processing. Highly normalized data schema allows fast data modifications, but is inconvenient for querying. Queries on temporal data usually are complicated. Such query as “history of average annual salaries of employees” is quite easily formulated by human language, but may require contemplate writing to be expressed in SQL language. In order to achieve good performance and simplicity of queries, data schema should be denormalized. In the implementation example above we sow the denormalized data schema being used.

There are three main different groups of temporal queries [15, 16]:

- queries over the current state; they are used to ask “that is true now?”;
- historical or sequenced queries; they used to do some historical analysis of stored data.

The queries of the 1st and 2nd type are used very frequently, so it is important to optimize relational schemas for these types of queries. Queries of the 3rd type are very complicated by themselves. We need more intelligent query engine to reduce their complexity.

The support for queries over the current state is provided by the CLASS table. This table stores the rows with only the last values of attributes of objects. All attributes of object can be found in one table. So this eliminates the need of joins between tables.
The support for the time-slice queries, is provided by storing the rows with the current values of attributes of objects in the STATES table. These values are the copies of values stored in the CLASS table, so they denormalize the data schemas, and allow formulate time-slice queries more easily.

5. Conclusions and future work

Temporal properties of the elements of object model discussed in this article are needed to achieve the support of various manipulations on time-varying information within contemporary information systems. The motivation of the research also is expanded by the wish to have a set of convenient formalism needed for such a support. The general dynamic model for temporal objects suggested here was augmented by the consideration of the mapping of objects to relational database, and the temporal properties of such mappings were discussed. As a consequence, the denormalization procedures of data schema to support the most common types of temporal queries (queries over current state and time slice queries) were suggested.

Spatial and temporal information bear uncertainty, especially as regards the relations between spatial and temporal facts. It is recognized that the existing spatial relations do not cover the areas of metrics and the area of uncertainty. If such concepts would be supported, a new powerful tools for query processing would arise. There is a great amount of knowledge that resides in the spatial and temporal relations among objects in a relevant framework. It is worth to model this knowledge representing the uncertainty of spatial information related to the position, and the shape of an object and also the uncertainty related to spatial relations features (topology, direction, metrics) as well as the uncertainty in the temporal dimension (including the proximity of a fact to a desired time point).

The immediate prospective directions for the research in this area are suggested as:

• to develop the dynamic model for bitemporal time support,
• to explicate the temporal properties implementation in SQL3-standard databases,
• to propose the users’ interface design methodology for temporal applications,
• to investigate and develop the temporal properties implementation in XML language.

6. References


