First publ. in: 6. Internat. Workshop on Foundations of Models and Languages for Data and Objects : Integrity in Databases. 1996

On Consistency in Temporal Object Bases

Holger Riedel
Universität Konstanz
Fakultät für Mathematik und Informatik
Postfach 5560/D188
D-78434 Konstanz, Germany
E-mail: Holger.Riedel@uni-konstanz.de

Abstract

In a temporal object-oriented database (TOODB) not only the actual state but also some parts of the history are stored. We identify some problems for the consistency of the database and the update operations in a bitemporal environment. We propose a strategy how a database which is only partially temporal can be interpreted intensionally as a temporal database. Additionally we analyze some related work which was done for relational and OO frameworks.

1 The surroundings

We focus on temporal extensions of an object-oriented data model [RS96] where temporal information is stored for some instances of the database. The data model supports type constructors like structs, sets, bags, and lists which can be combined orthogonally. Object types are modelled as a set of (partial) functions. A class is a collection of objects of a certain type. This data model is enhanced by temporal concepts in the following way. The data model is bitemporal, i.e., it supports both, valid time and transaction time. Thus, it can be recorded in the database at which instant an object should exist in the past or in the future and also the changes of its attribute values (valid time). Additionally, the transaction time of committed changes of the database instance can be recorded. In our approach, it is possible that only parts of the database schema are temporal. Another approach would be to record all data of the past in the database. Then, consistency checking would be rather simple, but this approach seems impractical or undesired in many real applications:

- The amount of storage space needed to record everything could be too large in many cases, even under the most positive assumptions for the evolution of storage technology.
- Temporal information that is not be needed by the application is recorded in vain, with massive waste of resources (space and — more importantly — also time for query processing).
- Often the necessary information of the past is just not available to the application (due to lack of recording, insufficient privileges, or legal restrictions).

In the $\mathcal{TCROQUE}$ model, each part of the database schema can be temporal or non-temporal. For example, the following cases are possible:

- A class is bitemporal, but the corresponding type is non-temporal. This means that it will be recorded at which instant an object is an element of this class (valid time) and also when this information is processed in the database (transaction time). But former values of the attributes of objects of this class are not recorded in the database.

```
define type Person
   name: STRING;

define class Persons: Person;

define class Employees@[MONTH] isa
                              Persons
   ssno: integer;
   salary: MONEY;

define class Cooks isa Employees
   specialities: SET OF STRING;

define class Businessmen isa Persons;
```

```
define type time-interval:
       (start: date, end: date);

define class Restaurants: isa Objects
   r_name: STRING;
   owner: Person;
   the_cooks: SET OF Cooks;
   open: time-interval;
   menu: LIST OF dish;
     dish: STRUCT OF d_name,price,ingredients;
       d_name: STRING;
       price@[DAY]: MONEY;
       ingredients@[HOUR:DAY]: SET OF ingredient;
         ingredient: STRING;
```

Figure 1: A temporal database schema

- Also the opposite is possible: Only the changes of attribute values of a specific class are recorded but not the explicit membership of the object in that class.
- Only some attributes of a class are bitemporal or valid time or transaction time. The others are non-temporal.
- Only some parts of a complex attribute are temporal. This is an important difference compared with relational approaches where temporal attributes with the key attributes and the related timestamps can be factored out into additional relations using temporal normalization [NA93]. In a TOODB there can be a lot of information within a single structured attribute which cannot be "normalized" on the conceptual level.

Example 1 The $\mathcal{TCROQUE}$ database schema of Figure 1 and the database of Figure 2 show some information about restaurants. Time support is marked by the use of the notation "`@[`*valid-time-unit*`:`*transaction-time-unit*`]`". There are different kinds of time supported in the running example:

- The class *Employees* is valid time recording the months of employment. The attributes *ssno* and *salary* of *Employees* are non-temporal.
- In the class *Restaurants*, the attribute *open* is a user-defined time interval. The *menu* as a whole is non-temporal but for the component *price* the valid-time changes are stored day-by-day. The set-valued component *ingredients* is bi-temporal:
  - the valid-time information is recorded up to an hour,
  - the granularity of transaction-time is a day. □

Notice that in the $\mathcal{TCROQUE}$ model we use time points instead of time intervals. The interval notion in Figure 2 is just an abbreviaton. It is further important to stress the different semantics of the notions of time: transaction-time is always the commit point of an update transaction. Further discussion of the differences of valid time versus transaction time can be found in [CI94].

2 The problems

In the TOODB, a lot of information on the history of the database objects and the database itself are directly accessible, such that some temporal constraints can now directly be checked using the recorded data. Nonetheless, there are new problems, which can be classified as follows. The OO data model captures much more application semantics than the relational data model which has to be monitored accurately. An update in a temporal database may have much

Persons

oid	*Name*
p_1	John Smith
p_2	Peter Green
p_3	Mary Green
p_4	Theo Turner

Employees

oid	*ssno*	*salary*
p_1 [11/75 - now]	55446677	67000
p_2 [3/92 - 5/95, 12/96-4/2004]	55448867	34000
p_3 [3/90 - 5/95]	55338967	45000

Cooks

oid	*specialities*
p_2	croque, salad
p_3	wild, dessert

Businessmen

oid
p_4

Restaurants

oid	*r_name*	*owner*	*the_cooks*	*open*
r_1	Theo's	p_4	$\{p2, p3\}$	(5pm,3am)
r_2	Villey's	p_4	{}	(0am,12pm)

Restaurants (continued)

	menu		
	d_name	*price*	*ingredients:{ingredient}*
	Theo's Best	{45@[1/12/90 - now]}	{{wild boar,spices, salad} @[1/12/90/9am − 31/11/94/12pm : 1/6/91], {wild boar, spices} @[1/12/94/12am - now]:1/10/94}}
	Standard	{25@[1/12/90 - 31/11/94, 30@[1/12/94 - 23/4/2003}}	{{beef,spices} @[1/4/88/9am - now:1/6/91]}
	Villey's Breakfast	{2.50@[1/3/93] - 31/11/94], 3.00@[1/12/94 - 23/4/2003]}	{{ham, eggs, soft cake} @[1/3/93/8am - now:1/1/93]}

Figure 2: A temporal database

more side effects than in a non-temporal database. The flexible use of temporal data is in some way contradicting the assumptions of the OO data model. For example, according to the history of some objects it can be deduced that other objects have to exist in certain classes at certain instants. But this information is not recorded in the database, because these classes are non-temporal.

2.1 Modelling aspects

When time is part of the modelling process, it is more difficult to interpret the database correctly. In a non-temporal database the creation/deletion of an object or an update of an attribute value reflects the change of knowledge within the application area. Usually the temporal aspect is disregarded, i.e., the values seem to be valid in the interval `[now,now]`.
In a database where all temporal data is recorded the situation is different. Each data item has an exact timestamp of its valid time and/or transaction time. Thus, there is no need for further semantic interpretation of time. Also there is no need to relate the valid-time information to the `now` instant. Maybe the database solely consists of historical information with no relation to the actual instant. Also it may occur that some future information is already present in the database, e.g., fixed raises of the salary.
In the relational context there exists the alternative to distinguish between tuple timestamping and attribute timestamping. In the object-oriented context it is necessary to adapt a modified version of attribute-based timestamping. But now there rises the problem to identify the "key" of the stored information. In a temporal relational database (TRDB) the key is given by (a subset of) the non-temporal attributes. This may not be applicable in a TOODB, where the object is usually identified by the object identifier which is unique over time. Nevertheless, different problems can occur. For example, it may happen that the occurrence of an object in a class is temporal while the attributes are non-temporal.

Example 2 John Smith is an employee in the database of Figure 2 since 11/75, but we have only one salary which may be interpreted as the current one. Even this interpretation of the database gets critical when we examine the database on 18/9/96 (`now=18/9/96`) for either p_2 or p_3:

- There is only historical information for p_3. So the *salary* reflects only one known state (hopefully the last one).
- The information for p_2 is more difficult to interpret. Because he is not an employee on 18/9/96, the salary may be related to his last job or to his future employment. □

Another problem is the occurence of a non-temporal object with temporal attributes.

Example 3 The restaurant `Theo's` is currently in the database but there is additional information available about prices and ingredients. Also we know from the point of view of 18/9/96 that the restaurant should exist in some future time because of future information about prices. □

It may seem that the above problems can be solved demanding further constraints for the database schema or for the instance, e.g., either the class *Employee* and its attributes are temporal or both are non-temporal. But further problems arise due to inheritance.

- An object is simultaneously in several classes which may be temporal or not.
- An attribute (e.g. *the_cooks*) of one class refers as a component object to another class and these classes use different notions of time.

Example 4 In our database Peter Green (p_2) is a cook. We have no further information since when or in what places he worked. Because the class *Cooks* is a specialization of *Employees*, we know that Peter Green can only be employed as a cook within the temporal element [3/92 - 5/95, 12/96 - 4/2004]. Therefore, we get new information about the employment of Peter Green at Theo's, although there is no "direct" temporal information for *the_cooks*. □

Finally, it is necessary to clarify the possibilities when structured values are not temporal as a whole but only have some (complex) temporal components. Because values differ from objects that they are not identified by separate object identifiers but by their values, it is necessary that (parts of) the temporal components are the "keys" of a complex value (like in the relational case).

Example 5 Take a look on the *menu* attribute. The tuple *dish* consists of a non-temporal component *d_name* which serves as a key for the temporal attributes *price* and *ingredients*. □

2.2 The update problem

The situation is as follows: (a) there is a database which is partially temporal, (b) there is an update for a certain object of a specific class. The changes directly described in the update statement are either temporal or non-temporal.
Two problems must be solved. At first, it is necessary to cascade further updates in order to get a new consistent database state. Secondly, the semantics of non-temporal parts of the database depends on an accurate interpretation. So how does the intensional semantics change when the database is updated?

Example 6 We add the fact to the database that Peter Green is additionally employed in the time interval [4/95 - 12/97]. The new instance is $(p_2[3/92 - 4/2004], 55448867, 34000)$. This new information can be used to imply that he may be a cook in `Theo's` for the whole time. □

Example 7 The object p_2 will be deleted from the *Persons* class. This implies that it must also be deleted from its non-temporal subclasses (in this case from *Cooks*). But what should happen to the valid-time class *Employees*? □

3 A basic strategy

In this section we present a method to solve the discussed problems within our framework. It is based on an intensional interpretation of the database extending the work which has been done for TRDB with one time dimension. Here we do the following:

- the database schema is defined using complex types, subclassing, and the flexible notion of valid time and transaction time as given in Section 1.
- the *extensional database state* is defined according to the database schema and can look like in Figure 2.
- there is an *intensional interpretation of the database*, where each class and each attribute is interpreted as bitemporal. Clearly this interpretation depends on the specific transaction-time instant.

Here we give only a short outlook of the theoretical framework which is based on the formalization of the non-temporal data model CROQUE [RS96], and an intensional interpretation of time as given below. In the future, the presented solution has to be extended in order to capture more "logical" parts adapted from temporal logic and an adequate object logic. In our model we use instants isomorphic to integers for valid time and transaction time. In the formal model we do not treat different granularities of times specifically.

Definition 1 An extensional database instance is given by:

1. Let τ be a type, then $v \in dom(\tau)$ is a possible value for τ. The domain $dom(\tau)$ of complex types is constructed as usual [LS93, RS96, BFG96], e.g., $dom(\texttt{set\_of}(\tau)) = \wp(dom(\tau))$. The temporal domain is given by $dom(\mathcal{TIME}) = \mathcal{INSTANTS} \cup \{\texttt{now}\}$. The domain of temporal types is given by partial functions:
   - $dom(\texttt{valid\_time\_of}(\tau)) = \{\mathcal{TIME} \mapsto dom(\tau)\}$,
   - $dom(\texttt{transaction\_time\_of}(\tau)) = \{\mathcal{TIME} \mapsto dom(\tau)\}$.
2. $\mathcal{D}_{Object}$ is a countable set of possible objects.
3. A class C is given as a set of objects with a fixed type τ_C. The definition of the domain is given by:
   - C is non-temporal: $dom(C) = \wp(\mathcal{D}_{Object})$,
   - C is valid-time: $dom(C) = \{\mathcal{TIME} \mapsto \wp(\mathcal{D}_{Object})\}$,
   - C is transaction-time: $dom(C) = \{\mathcal{TIME} \mapsto (\wp(\mathcal{D}_{Object})\}$,
   - C is bitemporal: $dom(C) = \{\mathcal{TIME} \mapsto \{\mathcal{TIME} \mapsto \wp(\mathcal{D}_{Object})\}\}$.
4. Subclassing is simplified for temporal classes, i.e., it is sufficient that the object is in the superclass at some instant: C `subclass_of` $C_1 \wedge o \in C \Rightarrow o \in C_1$. [1] □

In the following we define the intensional database describing which data is valid at a certain instant vt when we use the given extensional database at the instant tt. In the intensional database interpretation, each class and type is used as bitemporal. Thus it is possible to look at the database as a full bitemporal database. Because only the available data of the extensional database is used, the intensional database differs for different transaction instants.

Definition 2 Let $d(tt)$ be the database at instant tt. The intensional database instance $d^*(tt)$ for a valid-time instant vt at a certain instant tt is the minimal database instance given by the following rules:

[1] Because we do not know how time is supported for C and C_1, we do not make further assumptions.

1. Let $f(vt)$ be a valid-time fact of $d(tt)$, $vt \neq \texttt{now}$, then $f(vt)$ is also in the intensional database ($f(vt) \in d(tt) \Rightarrow f(vt) \in d^*(tt)$).
2. Let $f(\texttt{now})$ be a valid-time fact of $d(tt)$, then `now` is interpreted as the actual transaction instant ($f(\texttt{now}) \in d(tt) \Rightarrow f(tt) \in d^*(tt)$).
3. Let o be an object, C a class and $(o \in C(vt)) \in d^*(tt) \wedge C$ `subclass_of` $C_1 \Rightarrow (o \in C_1(vt)) \in d^*(tt)$.
4. Let o be an object, F an attribute of o, $o.F(vt) \in d^*(tt)$, A an object-valued component of F, A refers to the class C, and $o.F(vt).A = o'$. Then $(o' \in C(vt)) \in d^*(tt)$. □

The intensional database is a specific interpretation of the extensional database. It respects in each transaction instant tt the usual class-based consistency, meaning that each object of class C at instant tt is element of all super classes at the same instant, and that each object used as a component object at instant tt is also an object of the related class at this instant. It should also be noted that additionally several other implication rules could have been used which lead to different implications for the intensional database.

Example 8 We refer to the database of Section 1. According to the definitions above, we look at some of the examples of Section 2:

- *Example 2:* Viewing the intensional database $d^*(18/9/96)$, we get that the salary is only interpreted as the salary of 18/9/96, because there are no implication rules which infer from temporal classes to the attributes of the same class.
- *Example 4:* The class *Person* inherits the time values of the class *Employees*. Thus, John Smith is a member of the class *Persons* in the interval `[11/75 - 18/9/96]` in the intensional interpretation $d^*(18/9/96)$. Nonetheless there are no implications for subclasses, thus all cooks seem to be present only at 18/9/96 in $d^*(18/9/96)$.
- *Example 7:* Here a more exact specification of the update is necessary. In order to get a clean extensional database, it is necessary to delete all the information of p_2 from the class *Employees*. □

4 Related Work

In this paper we added bitemporal possibilities to an object-oriented data model with complex values. In the literature there are several other approaches which solve parts of the mentioned problems. These can be classified as follows:

- *bitemporal relational:* The main analysis is done for a single bitemporal relation.
- *deductive:* A relational valid-time database is interpreted using extensions of Datalog rules or similar frameworks [BCW93].
- *temporal logic:* In the proposed frameworks so far, such logics are used for relational databases with one dimension of time.
- *OO frameworks:* The work done up to now only deals with some restricted cases.

  For instance, in [BFG96] an object-oriented model is introduced, where complex values can be partially temporal, but only for one dimension of time (see below).

  OODAPLEX [WD93] uses an approach based on abstract data types where valid time and transaction time can be modelled using appropiate functions. The consistency of the database is enforced through additional constraints. A query language based on nested loops is proposed to query the database.

  Jeusfeld and Jarke [JJ91] analyze possibilities to check and simplify static constraints using update operations within an object-model without complex objects.

We examine some approaches in more detail.

Won&Elmasri. In [WE96] the interpretation of bitemporal relations is addressed. In order to clarify which temporal information is valid at a certain instant, each bitemporal relation is augmented with an additional attribute *Action* which signals for each tuple how it should be interpreted w.r.t. the other tuples of the relation. This can be used to query the database at each instant about historic and future data. Another advantage is that, in case of an update, older tuples of the relations need not be changed, even if constants like `now` are used. In the paper the following actions are discussed: `cancel` and `augment` can be used to delete or correct older information. Therefore, tuples for certain valid-time instants are inserted into the relation which either indicate that older information is invalid (`cancel`) or has been extended (`augment`). Furthermore, the `augment` can be divided in `regular` or `weak` to indicate that a regular version cannot be overwritten by weak versions.

Temporal logic. Another framework to solve the consistency problem of temporal databases is an extension of logic where several constraints are added using special operators like `always` or `since` to express dynamic constraints. In order to check the constraints there are several techniques which can be classified as follows: (i) *history-less checking:* only the actual database and some auxiliary structures are used. [LS87] use transition graphs, while Chomicki proposes the use of an additional relation of fixed size [Cho93] or the use of condition-action rules [Cho92]. (ii) *history-based checking*: Here the older database states must be used to verify the new database state.
The application of these techniques within the scope of this paper has to be evolved in the future in greater depth. One problem is the presence of some parts of the historic data as part of the current database. This can be used to verify a bigger set of constraints without data from former transaction instants. A more severe problem is the complexity of the object model which has to be incorporated more accurately in the former approaches. A first step is the use of the ER model as presented in [GL95].

A deductive approach. Wüthrich [Wüt90] analyzes the possibilities how a relational deductive DB can be extended to capture transaction time. Therefore, he uses the sequence of database states for each relation. Such a particular state can be manipulated by the use of update predicates. The notion of consistency is given by formulas of temporal logic which have to be fulfilled for each update. A constraint is classified as *present dependent* if it is related to the initial database in a certain way. Furthermore, a temporal deductive database is consistent if all constraints are *present dependent* and the initial database is empty.

T_Chimera. In [BFG96] a temporal OODB model is defined which supports one time dimension (`temporal`) as a type constructor for the support of valid-time information. Each attribute with temporal components is viewed as "historic" in constrast to non-temporal (so-called "static") attributes. The database is always viewed from the `now` instant. The consistency of attribute values is enforced by additional invariants which differ for historical and static attributes. The class hierarchy can be a DAG. For each class the temporal information about an object is stored explicitly. There are invariants stating that objects of subclasses have to be valid at that instant in the superclasses. The same holds for component classes. It is required that for each object and each instant there is a unique most specific class it belongs to. Subtyping rules for temporal types can be used to substitute values of subtypes. There are different notions of value-based equality, especially *instantaneous equal*, meaning there is at least one instant where the values are equal, and *weak-value equal* where there values can be equal at different instants. A snapshot function can be used to get a view of historic data of a certain instant in relation to the current time.

5 What comes next?

In this paper we presented some problems for bitemporal databases and discussed some solutions. Specifically, we have shown that the interpretation of an object-oriented database is much more difficult than in the relational case due to additional constraints of the database model and the flexible use of the temporal constructors. We used an intensional interpretation of the database and analyzed some consequences.
Some interesting topics have to be done in the future. It would be desirable to incorporate temporal logic in this framework to formulate temporal constraints. Also the maintenance of additional constraints using update rules have to be explored.
Finally, we are interested in interpreting queries in this framework and in object-oriented implementation techniques.

Acknowledgement. The author is grateful to Marc Scholl and the anonymous referees for valuable comments.

References

[BCW93] M. Baudinet, J. Chomicki, and P. Wolper. Temporal Deductive Databases. In *[TCG+ 93]*, pages 294–320, 1993.

[BFG96] E. Bertino, E. Ferrari, and G. Guerrini. A Formal Temporal Object-Oriented Model. In *EDBT*, 1996.

[Cho92] J. Chomicki. Real-Time Integrity Constraints. In *PODS*, pages 274–282, 1992.

[Cho93] J. Chomicki. On the Feasibility of Checking Temporal Integrity Constraints. In *PODS*, pages 202–213, 1993.

[CI94] J. Clifford and T. Isakowitz. On the Semantics of (Bi)Temporal Variable Databases. In *EDBT*, pages 215–230, 1994.

[DKM91] C. Delobel, M. Kifer, and Y. Masunaga, editors. *Proc. 2nd International Conference on Deductive and Object-Oriented Databases, München*. Springer, December 1991. LNCS 566.

[GH90] J. Göers and A. Heuer, editors. *Proceedings of the 2nd Workshop on Models and Languages for Data and Objects*. Informatik-Bericht 90/3, TU Clausthal, 1990.

[GL95] M. Gertz and U.W. Lipeck. Temporal Integrity Constraints in Temporal Databases. In *Recent Advances in Temporal Databases*, Workshops in Computing, pages 77–92. Springer, 1995.

[JJ91] M. Jeusfeld and M. Jarke. From Relational to Object-Oriented Integrity Simplification. In *[DKM91]*, pages 460–477, 1991.

[LS87] U. Lipeck and G. Saake. Monitoring Dynamic Integrity Constraints based on Temporal Logic. *Information Systems*, 12(3):255–269, 1987.

[LS93] C. Laasch and M.H. Scholl. A Functional Object Database Language. In *4th Intl. Workshop on Database Programming Languages*, 1993.

[NA93] S.B. Navathe and R. Ahmed. Temporal Extensions to the Relational Model and SQL. In *[TCG+ 93]*, pages 92–109, 1993.

[RS96] H. Riedel and M.H. Scholl. *The CROQUE-Model: Formalization of the Data Model and Queries*. in preparation, 1996.

[TCG+93] A.U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings, 1993.

[WD93] G.T.W Wuu and U. Dayal. A Uniform Model for Temporal and Versioned Object-Oriented Databases. In *[TCG+ 93]*, pages 230–247, 1993.

[WE96] J. Won and R. Elmasri. Representing Retroactive and Proactive Versions in Bi-Temporal DataBases (2TDB). In *Proceedings of the IEEE International Conference on Data Engineering*, pages 85–94, 1996.

[Wüt90] B. Wüthrich. Temporal Deductive Databases: Data Manipulation and Consistency. In *[GH90]*, pages 181–190, 1990.