On The Representation of Temporal Object Roles in Object Oriented Databases

Niki Pissinou
The Center For Advanced Computer Studies
The University of Southwestern Louisiana
Lafayette, Louisiana 70504

Kia Makki
Department of Computer Science
University of Nevada
Las Vegas, Nevada 89154

Abstract

This paper outlines a generic, core temporal object model that provides support for the modeling of *temporal object roles*. This model draws from notions introduced in some of our previous works on temporal object modeling by providing a specific database model that provides a conceptual modeling context for temporal object roles [3, 4, 5, 6, 9, 7, 2, 8, 10, 11]. The model achieves an integration of the abstract concepts that characterize temporal objects. The main purpose of the model is to provide a basic framework for *temporal object* information models and information systems. The model is based on a small number of simple "temporal" constructs and primitive "temporal" operations. The simple set of modeling constructs of this model are *temporal objects* and *temporal mappings*. A set of primitive operations is defined on temporal objects that allows uniform viewing, definition, insertion and manipulation of objects in temporal object databases. A simple set of temporal constraints is also provided.

1 Introduction

One of the problems faced by existing data models is their weak expressiveness of the representation of temporal information and in particular the representation of the temporal roles of objects. In consequence, typical database systems that embody these models, support the storage and retrieval of "facts" about the world and certain aggregation and operations on them, but do not perform operations on either the perception to map situations in the world into the database contents, temporal object roles or temporal operations. In view of this, a strong interest of ours is in developing a database model that represents "realistic worlds." To achieve this, in our previous works we have studied the semantics of time in the context of object database systems, identified changes to existing notions of temporal data that are necessary because of the transition from the relational to the object model and specified an approach to extending an archetyped, extensible object model, to incorporate a number of generic modeling concepts, with concrete definitions, underlying temporal objects [3, 4, 5, 6, 9, 7, 2, 8, 10, 11]. The work is in line with the infrastructure presented in [11].

A major aim of this research, is to achieve an integration of the abstract concepts that were assumed to characterize temporal data, presented in our earlier works. In order to achieve this goal, in this paper we map these abstract concepts to a simple object data model, thus articulating them as concrete concepts [9, 10, 7]. This provides a specific context for our approach to temporal object database modeling, and allows us to define a conceptual framework for a simple and generic object model. The model addresses temporal issues at the finest level of data granularity, viz. the object level.

Our work presents the foundations towards the synthesis of a powerful integrated object data model that supports the temporal/dynamic aspects of data modeling in addition to the structural and behavioral ones. Our model is intended to be both a valuable temporal model in its own right, as well as a mechanism for describing and analyzing temporal extensions to other database models as illustrated in [3, 5, 4]. This can be mainly achieved by using our model's small number of simple constructs and primitive operations, that can be used as the basis for the specification and stepwise development of temporal information models and systems of increasing complexity and levels of abstraction.

The remainder of this paper is organized as follows: In the next section we outline some preliminary concepts and definitions. In section 3, we describe the modeling elements that encompass our temporal object model. Section 4 briefly describes the temporal object definition and manipulation language while section 5 defines a simple set of temporal constraints. In the final section we provide some final remarks.

2 Preliminary Definitions

To describe the structure of an object as it evolves over time and across its multiple representations (i.e., versions and history) we introduce the notions of *universal object identity* and *possible object world*. Together these two notions allow us to describe the meaning, characteristics, properties, behavior and role of each individual object at different times and at a particular point in time.

Definition. The *Universal Object Identity* of an object, denoted as UOI, refers to an object's interpretation or sense and remains time invariant.

Definition. The *Possible Object World* of an object, denoted as POW, refers to an object's denotation and is usually time variant.

An object cannot have a possible object world unless its universal object identity is defined. An object can have different possible object worlds at different times, and also several possible object worlds at a particular time, but its universal object identity remains unique over a specified lifespan. For example, the object "fight" could be a physical struggle in 1989 followed by an emotional struggle in 1990. Also the object "fight" can concurrently be a physical and an emotional struggle. In general, a universal object identity encapsulates the time invariant semantic meaning of an object, while a possible object world encapsulates the static and dynamic structure, behavior and role of each individual object at different temporal intervals.

Further, the UOI of an object can be derived from all its possible POWs at all different times with respect to our model and the world we are modeling by taking the union of possible POWs over time. Thus, we can formally define the UOI of an object from all its possible POWs at all different times with respect to POWs, our database model and the world we are modeling, as follows:

$$UOI_\alpha(M) =_{def} \{t \rightarrow POW_\alpha(M,t) | t \in T\}$$

where

- UOI is the universal object identity of an object α
- POW_α is the possible object world of an object α
- M is our database model
- $t \in T$ is the time a possible object world is active (current)

The above definition assumes that users interact with only a single world during a time period. When users interact with different worlds then the UOI of an object is defined as follows:

$$UOI_\alpha(M) =_{def} \{< w,t > \rightarrow POW_\alpha(M,w,t) \mid w \in W \text{ and } t \in T\}$$

where

- UOI_α, POW_α and M are as defined previously
- $w \in W$ is the "mini-world" we are modeling (may be assumed to be a version)

Given these two new notions of an object, we are now able to give the definitions of a universal object identity and possible object world class and time slice. These are as follows:

Definition. A *Universal Object Identity Class*, related to a universal object identity UOI, is defined as all objects that can potentially, independent of time, have the same universal object identity.

Definition. A *Possible Object World Class*, related to a possible object world POW, is defined as a collection of (POW,t_i) s.t. $t = 1, 2, ...n$ tuples, of all objects that can potentially have the same possible object world. Since an object can have several POWs at a given time, an object can belong to several possible object world classes.

From our formal definition of a UOI of an object we can conclude that an object can be a member of only a single Universal Object Identity Class but belong to several Possible Object World Classes during a particular lifespan. Consequently, an object is not active if it does not belong to some particular Universal Object Identity Class at any given time t. If an object is not active it can not belong to any Possible Object World Classes. In other words, an object can not be a member of a Possible

Object World Class if it were not a member of a Universal Object Identity Class.

Definition. A *Time Slice* of a possible object world class is all objects with the same possible object world at time t_i. It can be viewed as a set of instances of a possible object world class at time t_i. Thus the union of all time slices of a possible object world class is the possible object world class.

Definition. A *Universal Object Existence* is a function

$$\mathcal{F} : (UOI, t) \rightarrow \{0, 1\}.$$

A value of zero at time t implies that the universal object identity of an object does not exist at that time or is undefined.

Definition. A *Possible Object World Existence* is a function

$$\mathcal{F}\text{: (POW,t)} \rightarrow \{0,1\}.$$

A value of zero at time t implies that the POW of an object does not exist at that time or is undefined.

One major requirement for the kind of temporal object model described here, is to have a general and convenient means for extracting the lifespan of an object and the history of that object. For this reason, we introduce the notions of *time-priority sequence* and *object history*. In the following definitions, we specifically adopt modified notions of a time sequence introduced in [12]. In these definitions we also incorporate the notion of priorities for POWs of an object. The idea of assigning priorities to possible object worlds is particularly useful in resolving possible conflicts among co-existing possible object worlds for a particular object and in the metadata. For example, given that an object has two possible object worlds *teaching assistant* and *research assistant* then by assigning priorities to these two POWs we can determine the object's primary role at any given time t_i.

Definition. The *Time Sequence* of a POW of an object, denoted as *TSEQ(POW)* is defined as

$$(POW_{t_1}, \ \ldots\ldots\ , POW_{t_n}) = (\text{POW}, (t_1, t_2, \ \ldots\ldots\ , t_n)).$$

Thus the lifespan of a POW of an object can be easily determined since t_1 denotes the *valid from* time and t_n denotes the *valid to* time.

Definition. The *Time-Priority Sequence* of a POW of an object, denoted as *TPSEQ(POW)* is defined as

$$(POW_{(t_1,p_i)}, \ \ldots\ldots\ , POW_{(t_n,p_k)}) = (\text{POW}, (t_1, p_i), (t_2, p_j), \ \ldots\ldots\ , (t_n, p_k)).$$

Definition. An *Object History* refers to the collection of time sequences *TSEQ(POW)* of all the POWs of an object.

Thus the lifespan of an object can be derived from the lifespans of all the POWs of that object. There is a partial function F_{lpow} that determines the lifespan of a POW of an object. The domain of F_{lpow} is $POW = \{POW_1, POW_2, ..., POW_n\}$, the set of all possible object worlds of an object, and its range consists of finite subsets of $T = \{t_1, ..., t_n\}$.

3 Modeling Constructs

Given the temporal semantics introduced in our previous works on temporal object modeling, we can now illustrate our proposed approach to the modeling constructs appropriate for the design and implementation of a generic and core temporal object database model. In this section we show how objects with several possible object worlds are defined, and develop the necessary mappings that allow a transition from one possible object world to another to occur over time.

Much in the spirit of several other data models, the temporal object model is based entirely on the notions of objects and inter-object relationships. A temporal relationship among object cannot exist unless all the components are existing database objects, or relationship is proactive such as "James is going to be promoted to full professor as of next month" (given that in the second case proactive changes are not allowed.) In addition since temporal databases need to retrieve and manipulate historical data, an object is not removed from a database unless there was an error. Instead the object and all the relationships in the database in which it participates are "virtually deleted", which is a form of archiving the object.

Formally, a temporal object database system can be thought of as consisting of temporal objects and temporal relations. Each temporal object in the database consists of a single universal object identity and a set of possible object worlds, as defined shortly, and corresponds to a temporal relation on the set of all database temporal objects. Let $(\alpha_{t_i}, \delta_{t_j}, \beta_{t_k})$ be a relation. For each temporal relationship,

$(\alpha_{t_i}, \delta_{t_j}, \beta_{t_k})$, δ_{t_j} is a temporal relation. In particular, $\delta_{t_j} = \{ (\alpha'_{t_i}, \beta'_{t_k}) \mid$ the temporal relationship $(\alpha'_{t_i}, \delta_{t_j}, \beta'_{t_k}) exists\}$. If proactive relationships are disallowed then the constraint, $t_i < t_j$ and $t_k < t_j$ is explicitly imposed on the temporal relationship.

In the following section we introduce our definition of an object. This definition draws heavily from the notions introduced in [3, 4, 5, 6, 9, 7, 2, 8, 10, 11].

3.1 Temporal Objects

The restrictions that some object models, impose on an object are too strong in the face of the desire to model the role of temporal objects. For example, in may such models an object is restricted to having only a globally time invariant object definition generated by the system. In an effort to model the "real-world" more realistically, we relax these restrictions by defining two new concepts: the *universal object identity* of an object that refers to an object's interpretation or sense and remains time invariant, and the *possible object world* of an object that refers to an object's denotation or reference. An object only has a single universal object identity but can have several possible object worlds at a given time or at different times during its lifespan.

Each temporal object is a tuple consisting of a universal object identity and a sequence of possible POWs i.e.

$$\text{object} = (\text{UOI}, (POW_{\alpha_{(t_1,p_i)}}, \ \ldots\ , POW_{\omega_{(t_n,p_k)}}))$$

where UOI is the universal object identity of the object, which is time invariant; and $POW_{\alpha_{(t_1,p_i)}}, \ \ldots\ , POW_{\omega_{(t_n,p_k)}}$ is a sequence consisting of the possible object worlds of the object and their corresponding priorities at some specific time.

In our model an object id (oid) has no intrinsic meaning. Instead it is an entry point (reference or handle) for accessing information about a temporal object such as an object's UOI and POWs. It "derives" its meaning from its associated universal object identity and from its possible POWs. With each universal object identity we associate a universal object identity id, and with each possible object world POW_α, we associate a possible object world id, a time priority sequence $TPSEQ$, a set of temporal constraints, and a set of operations.

Our model also allows the creation and manipulation of **Temporal Composite Objects** (TCO.) A temporal composite object consists of a group of temporal objects. It exists at time T if each of its components exist at time t_i i.e.

$$TCO_T \circ (O_{1_{t_1}} + O_{2_{t_2}} + \ldots + O_{n_{t_n}})$$

where $T > t_i$ s.t $i = 1, 2, ...,$ $\circ$ denotes the temporal compose operator, O denotes the temporal object or an atomic temporal object, and + denotes the temporal aggregation operator.

A temporal composite object has a possible object world if the possible object world or reference of the whole object can be described as a function of the possible object worlds or the references of its parts. Thus, for a composite object to exist at t_j, the universal object identity and possible object world existence of each of its components should be 1 at t_i such that $t_j > t_i$.

To more accurately represent temporal data and the evolution of temporal objects, it is desirable to model different time dimensions such as valid, transaction and user-defined times. For simplicity in representing this information and in developing an initial experimental prototype we consider only valid time, but the other two kinds of time can easily be incorporated into the model. In our model we treat time as defined in [11].

4 Temporal Object Definition and Manipulation

The data definition and manipulation language is described as a set of primitive operations that are embedded within a host programming language. The host language supports the data types of temporal objects and set of objects, the usual set operations, a looping construct to iterate on elements of sets, a counter for elements of sets, and a data type for time.

Programs written in the host language can reference temporal objects. The purpose here is not to propose a specific approach to host language embedding of data manipulation operations, but rather to define a set of primitive building blocks for high-level database systems. In this section we briefly present our temporal operations [1]. and address some constraint issues.

[1] The operations presented here are by no means exhaustive; we are only presenting a flavor of the temporal operations supported by our approach.

In our model all operations can be specified as either messages or processes. A process provides inferencing capabilities such as creating messages from other given messages. It consists of a set of preconditions and a set of actions. In addition to these processes there are other implied ones that are initiated automatically as a result of some specified relationship. We have used these concepts to develop a set of specialized temporal operations on objects and mappings. In the following we describe a sample subset of such operations.

4.1 Create Operations

The set of Create Operations supports the creation, activation, deactivation, and context release of objects. In particular, there are three primitive operations associated with the POW of an object. These are *create-pow*, *deactivate*, and *activate*. The possible object world of an object may be created, activated or deactivated using the following messages:

1. **createpow(oid,(t_i, p_k),powid)**. This operation creates a new POW for an object. t_i is the object's time of creation and p_k is the initial priority assigned to that object. Here powid is a possible object world identifier. This new POW may not be created at t_j unless its associated UOI exists at some time t_i such that $t_i < t_j$. (An UOI for an object may be created only once during its lifetime.) Initially, no object is bound to a POW.
2. **deactivate(oid,(t_i, p_k),t_j,powid)**. The DEACTIVATE operation suspends the POW of an object at time t_i until t_j; If t_j is set to ∞ then the object is suspended indefinitely. In our temporal object model an object is never removed from the database. It is either suspended indefinitely or it is archived through the ARCHIVE operation, which is similar to a "virtual" delete.
3. **activate(oid,(t_i, p_k),powid)**. The ACTIVATE operation resumes the POW of an object at time t_i where t_i is greater than the time of its suspension. Before we activate (and even create) a new POW for an object we need to verify that it is not currently active. This is done through the EXIST-POW operation which is described later.
4. **release(oid,(t_i, p_k),powid)**. The RELEASE operation releases a specified POW for a specified time from being bound to a specified object. If the POW is not bound to the object the operation has no effect.
5. **error_delete(oid,(t_i, p_k),powid)**. The ERROR_DELETE operation deletes the POW of an object. Since in our model we only allow virtual deletes, this operation is meant to be used where there is a transaction error which is of no use and has no historical value e.g., a user inputs the wrong time. Thus, the sole purpose of this function is to delete erroneous information.
6. **create_fun(oid,(t_i, p_k),powid,flag)**. The CREATE_FUN operation creates one of the four mapping functions discussed in [3]. These mapping functions are: POW-to-POW, POW-to-UOI, UOI-to-POW and UOI-to-UOI. The type of transition is denoted through the flag. Also, the function *create_map* creates the actual relation.

4.2 Attach and Exist Operations

The set of Attach and Exist Operations is used for providing context to universal object identities of objects and temporal contexts to possible object worlds of objects. Before we can create the POW of an object, we need to verify that it has an associated universal object identity. If it does not have a UOI then the appropriate universal object identity is attached to it. This sequence of events is captured through the following operations.

1. **exist-uoi(oid,t_i,uoid)**. The EXIST-UOI operation verifies that an object's universal object identity is active at t_i. If so, it returns true otherwise it returns false.
2. **attach-uoi(oid,meaning)**. If an object's universal object identity does not exist at t_i i.e. if the specified UOI is already bound to the specified object, it is created through the ATTACH-UOI operation.
3. **exist-pow(oid,t_i,pow_α)**. The EXIST-POW operation verifies that an object's POW is active at t_i. If so, it returns true otherwise it returns false.

4. **attach-pow(oid,**(pow_α, t_i)**,powmeaning)**. The ATTACH-POW operation attaches temporal context to a possible object world. If the given POW of an object does not exist at t_i, it is created through the CREATE-POW operation and bound through the attach-pow operation. If the specified pow is already bound to the specified object, the operation has no effect.

4.3 Temporal Migration Operations

To facilitate the temporal evolution of objects and to accommodate the four types of mappings described in section 3.1, we have designed the following set of corresponding operations. These operations facilitate an object's migration from one UOI to a different UOI (or similarly from one possible object world (or a set of POWs) to another possible object world (or a set of POWSs)) at any given time.

1. **transfer-pow(oid,uoid,** **POW(**α, t_i**),POW(**β, t_j**))**. The TRANSFER-POW operation allows the migration of an object from one possible object world at time t_i, to a different possible object world at a different time t_j, where $t_i < t_j$. If the object's possible object worlds at t_i already exists then an appropriate warning is broadcasted.
2. **multitransfer-pow(oid,uoid,POW(**α, t_i**),** **{(POW(**β, t_j**),...(POW(**γ, t_j**)))})**. The MULTITRANSFER-POW operation allows the migration of an object from one possible object world to a new set of possible object worlds. The transitional path of this migration is also recorded. As with the previous operation, if the new set of possible object worlds already exists then an appropriate warning is broadcasted.
3. **evolve(oid,**$uoid_i$**,**$uoid_j$**)**. The EVOLVE operation creates a new universal object identity for a given object. As such, this operation facilitates the transformation of an object to an entirely new object over time.
4. **grow(oid,**$uoid_i$**,**$POW(\alpha, t_j)$**)**. The GROW operation allows an object that has only a universal object identity to obtain its first temporal context. As such, this operation facilitates the first temporal representation of objects.

4.4 Temporal Evolution Operations

In addition to the above operations we have developed a set of operations that allows us to manipulate the time priority sequences of an object.

1. **TPSEQ-evolve(oid,**$TPSEQ_1$**,**$TPSEQ_2$**)**, The TPSEQ-EVOLVE operation creates a new time priority sequence for an object. This operation allows us to extract the temporal behavior of objects over a period of time.

A different operation is used for changing the priorities of a given possible object world at any given time t_i,

4.5 Historical Operations

We have also developed a specialized set of operations which allows us to manipulate and extract the lifespan of an object and its history, compare inter-temporal relationships among objects, as well as to accommodate the relations introduced in [3]. In general, this set of operations supports the historical definition and manipulation of objects.

5 Temporal Object Constraints

The interaction of objects with other objects has to be subjected to some strict temporal constraints. This tends to be a rather complicated task since there is a significant difference between conventional "non-temporal" data models and temporal object models. While in the first case only true messages are allowed, and any checks on such messages are made at the time the message is input, in temporal data models any message could potentially be true. Therefore, our model's constraints should categorize information into true, false and meaningful the later making the specification of such constraints complicated.

In our previous works [3, 11] we have defined various temporal operators such as starts, meet, finish. Based on these operators, we have developed a set of primitive explicit constraints which can be applied on the mappings or the objects themselves, based on [1], some of which are

stated below. These constraints are enforced when we process such queries as "who were UK's prime ministers while Mr. Bush was in office?". In defining these constraints, and for simplicity we assume the existence of only two POWs and explicitly indicate the *valid to* and *valid from* times of a TSEQ(POW). However, the generalization of these constraints is straightforward. These constraints are defined as follows:

Constraints for Lifestarts:

The create_time of POW_α equals the create_time of POW_β; i.e., given $(POW_\alpha,(t_i,t_j))$ $starts$ $(POW_\beta,(t_k,t_l))$ then t_i $equals$ t_k.

Constraints for Lifefinishes:

The virtual-delete_time of POW_α equals the virtual-delete_time of POW_β; i.e., given $(POW_\alpha,(t_i,t_j))$ $finishes$ $(POW_\beta,(t_k,t_l))$ then t_j $equals$ t_l.

Constraints for Lifequals:

The create_time of POW_α equals the create_time of POW_β and The virtual-delete_time of POW_α equals the virtual-delete_time of POW_β. i.e, given $(POW_\alpha,(t_i,t_j))$ equals $(POW_\beta,(t_k,t_l))$ then t_i $equals$ t_k and t_j $equals$ t_l.

Constraints for lifeoverlaps

The create_time of POW_α is less than the create_time of POW_β and the virtual-delete_time of POW_α is less than the virtual-delete_time of POW_β and the create_time of POW_β is less than the virtual-delete_time of POW_α; i.e., given $(POW_\alpha,(t_i,t_j))$ lifeoverlaps $(POW_\beta,(t_k,t_l))$ then $t_i < t_k \wedge t_j < t_l \wedge t_k < t_j$.

Constraints for lifeduring

The create_time of POW_α is greater than the create_time of POW_β and the virtual-delete_time of POW_α is less than the virtual-delete_time of POW_β. i.e., given, $(POW_\alpha,(t_i,t_j))$ during $(POW_\beta,(t_k,t_l))$ then $t_i > t_k \wedge t_j < t_l$

Constraints for lifecontains

Given, $(POW_\alpha,(t_i,t_j))$ contains $(POW_\beta,(t_k,t_l))$ then $(POW_\beta,(t_k,t_l))$ $during$ $(POW_\alpha,(t_i,t_j))$.

Constraints for lifemeets

The create_time of POW_α equals the create_time of POW_β; i.e., given $(POW_\alpha,(t_i,t_j))$ $meets$ $(POW_\beta,(t_k,t_l))$ then t_j $equals$ t_k.

Constraints for lifebefore

The virtual-delete_time of POW_α is less than the create_time of POW_β; i.e, given $(POW_\alpha,(t_i,t_j))$ $lifebefore$ $(POW_\beta,(t_k,t_l))$ then t_j $less\ than$ t_k.

Constraints for lifeafter

$(POW_\alpha,(t_i,t_j))$ $lifeafter$ $(POW_\beta,(t_k,t_l))$ then $(POW_\beta,(t_k,t_l))$ $lifebefore$ $(POW_\alpha,(t_i,t_j))$

Constraints for Temporal relationships

For each relationship, $(\alpha_{t_i},\delta_{t_j},\beta_{t_k})$, and assuming no future relationships can be established t_i $before$ t_j and t_k $before$ t_j.

Finally, we can have a set of general state constraints viz., semantic constraints such as:

- an object cannot have a POW unless its UOI is defined.
- a temporal composite object cannot exist at t_j if the UOI and POW of each of its components is not 1 at t_i such that $t_i < t_j$.

In addition to these constraints in our previous works [3, 11] we have suggested a set of temporal principles that our model should adhere to. We call this set "meta-temporal semantic constraints." Although the constraints studied in this section are very restricted, it is believed that our temporal object database system provides a good basis for studying other, more general constraints.

6 Concluding Remarks

In this paper we have outlined an approach to the design and development of a model that integrates time with object databases. Our work applies temporal notions to the problem of object and meta-data evolution. It is a step towards the synthesis of a powerful integrated object data model that supports the temporal aspects of data modeling in addition to the structural and dynamic ones. In our results we developed a set of temporal constructs, operations and constraints for supporting and manipulating temporal objects and their roles.

Our framework is a very simple temporal object database model for modeling temporal objects and temporal relationships. All data in the database are treated uniformly as temporal objects; relationships among these objects allow the temporal evolution of objects to be modeled. Operations on the data allow temporal behavioral properties to be modeled. Mechanisms are provided to allow the modeling of temporal relationships of objects, the temporal evolution of objects and object

migration. In addition, the model provides a high degree of temporal and semantic expressiveness.

Our temporal object database system is not a high-level model appropriate for unsophisticated database users; for example, it is easy to create meaningless relationships and operations. The model lacks mechanisms for data protection and integrity control, and high-level constraints. However, the main purpose of our system is not to define a high-level temporal object database model, but on the contrary, to define a small set of fundamental concepts to be used as a vehicle in the design and implementation of temporal object models and for providing more expressive models.

References

[1] J. F. Allen. An interval-based representation of temporal knowledge. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 191–204, Vacouver, B.C., 1981.

[2] K. Makki and N. Pissinou. A new storage organization for temporal databases. *International Journal of Systems and Sciences (to appear)*, 1994.

[3] N. Pissinou. *Time in Object Databases*. PhD thesis, Department of Computer Science, University of Southern California, Los Angeles, California, December 1991.

[4] N. Pissinou and K. Makki. T-3dis: An approach to temporal object databases. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 185–192, 1992.

[5] N. Pissinou and K. Makki. A Framework for Temporal Object Database Models. In T.W. et. al. Finin, editor, *Information and Knowledge Management:Expanding the Definition of Database*, volume 752. Springer-Verlag, 1993.

[6] N. Pissinou and K. Makki. Separating semantics from representation in a temporal object databases. In *ACM Proceedings of the International Conference on Information and Knowledge Management*, pages 295–304, Washington, DC, November 1993.

[7] N. Pissinou and K. Makki. A unified model and methodoly for temporal object databases. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):201–223, 1993.

[8] N. Pissinou and K. Makki. Separating semantics from representation in a temporal object databases. *International Journal of Computer Information Systems*, Spring, 1994.

[9] N. Pissinou, K. Makki, and Y. Yesha. On temporal modeling in the context of object databases. *ACM SIGMOD RECORD*, 22(3), September 1993.

[10] N. Pissinou, K. Makki, and Y. Yesha. Research perspective on time in object databases. In R. Snodgrass, editor, *Proceedings of the International Workshop on Infrastructure for Temporal Databases Databases*, pages 295–304, Arlington, Texas, June 1993.

[11] N. Pissinou, R. Snodgrass, R. Elmasri, I. Mumick, M.T. Ozsu, B. Pernici, A. Segev, and B. Theodoulidis. Towards an infrastructure for temporal databases. Technical Report TR93, Departmet of Computer Science, University of Arizona, 1993.

[12] A. Segev and A. Shoshani. Logical modeling of temporal data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 454–466, 1987.