NESTED HISTORICAL RELATIONS

A. U. Tansel and L. Garnett

Bernard M. Baruch College
17 Lexington Avenue, Box 513
New York, N.Y. 10010

ABSTRACT

The paper extends nested relations for managing temporal variation of complex objects. It combines the research in temporal databases and nested relations for nontraditional database applications. The basic modelling construct is a temporal atom as an attribute value. A temporal atom consists of two components, a value and temporal set which is a set of times denoting the validity period of the value. We define algebra operations for nested historical relations. Data redundancy in nested historical relations is also discussed and criteria for well-structured nested relations are established.

1. Introduction

Databases in general carry the most recent data. As the new data becomes available through updates, the existing values are discarded from the database. Such databases are called snapshot databases, since they only contain current information which is a snapshot of the reality. However, in many real life applications, there is a need for both current and historical data. Any organization is an on-going process and its information needs and processing capabilities should be considered in a time perspective. That is, to support managerial informational needs, the database should possess a temporal dimension to store and manipulate time varying data.

Most data models do not address issues on maintenance and processing of temporal data. In these models it is possible to carry the time reference of an attribute as another special attribute. However, this approach is a rather ad hoc and limited solution. It either creates undue data redundancy and/or provides limited time processing capacity. There are two possible directions which can be followed for handling temporal data. One alternative is development of a new model to support time dimension and the other involves augmenting existing data models to support time dimension in a coherent way. In this paper, we adopt the second approach and propose an extension to nested (Non First Normal Form - N1NF) relations for handling time variation of complex objects.

There has been a great deal of recent research addressing the temporal aspects of information systems. Many proposals to extend relational model [Codd 70] as well as other data models for time support have been made. One group of researchers adds special time attributes to flat (1NF) relations [Ariav 86, Ben-Zvi 82, Clifford & Warren 83, Jones 84, Lum et al 84, Navathe & Ahmed 87, Snodgrass 87]. Another group proposes using N1NF relations and time-stamping attributes in contrast to the time-stamping of tuples found in the former approach [Gadia 86, Tansel & Clifford 85, Tansel 86]. Gadia extracts snapshots from historical relations whereas Tansel either applies algebra operations directly or normalizes (flattens) the relations before extracting data. In both approaches, there is only one level of nesting in the model. Ginsberg develops a record based systems to model histories of financial transactions [Ginsberg 84]. An extension to entity relationship model for handling time has been proposed by [Klopprage & Lockemann 83]. Anderson adds time to a binary-based model at the conceptual level [Anderson 81]. A new model for temporal data management has been developed where time varying data is visualized as a time sequence collection represented as a set of triples (surrogate, time, value) [Shoshani & Kawagoe 86]. Surrogate and time represent coordinates of the value. They also define new operations like restriction, selection, composition, etc. Further work on this model has been reported in [Segev & Shoshani 87a, Segev & Shoshani 87b].

Nested (N1NF) relations organize data hierarchically. Each tuple component can be an atomic (simple) value or another relation. Two new operations **unnest** and **nest** have been introduced to relational algebra for restructuring nested relations [Jaeschke & Schek 82]. Later, multi-attribute versions of these operations are also formulated [Fisher & Thomas 83, Jaeschke 84, Schek 86]. Recursive and non-recursive algebras for nested relations are defined [Jaeschke 84, Schek 86]. Pack and unpack are similar to one attribute nest and unnest operations used on set valued relations [Ozsoyoglu, Ozsoyoglu, and Matos 87]. Extended relational algebra and calculus languages for nested relations are formulated and their equivalence is proved [Roth, Korth, and Silberschatz 85, Garnett & Tansel 89].

Temporal dimension for complex objects have been discussed in several papers. Khoshafian and Copeland explored object identity for complex object, and proposed the object model which includes atomic, set and tuple objects [Khoshafian & Copeland 86]. Historical data is kept as <time interval, value> pairs where time interval denotes validity of the value. Copeland and Maier modified smalltalk 80 [Goldberg and Robson 83] for handling complex object and used a semantic data model [Copeland & Maier 84]. The semantic data model is based on labelled sets which consist of heterogeneous values which may be atomic or other sets. In case of historical data, a set element is <a time point, value> pair where time point is the transtation time when the value is recorded in the database.

In this paper, we combine the research in temporal databases and nested (N1NF) relations for nontraditional database applications like CAD/CAM, office automation, etc. We model a simple attribute value as a temporal atom which consists of two components, a temporal set and a value. The temporal set denotes a time period and the temporal atom asserts the value was valid over this period. We adopt and redefine the algebra operations of nested relations for nested historical relations. We also discuss redundancy in nested historical relations and develop criteria for well-structured nested historical relations. Our previous work modelled entity histories [Tansel 86]. This extension to relational model allows to model histories of relationships as well.

Section 2 describes the model and the scheme trees associated with nested historical relations. Relational algebra operations are discussed in section 3. Illustrative query examples are also provided. Section 4 covers redundancy in nested historical relations and establishes criteria for structuring such relations. Section 5 is the conclusion.

2. The model

Let T be set of time points mapped to natural numbers, i.e. {0,1,2,...,n} which is well ordered under less-than-or-equal-to relationship. 0 is the relative beginning point. The symbol n denotes the preset time instant and its value increments as the clock ticks (time advances). We do not consider a time unit. It is user defined and can be any of seconds, minutes, hours, days, etc. A **time interval** is a set of consecutive time points. For instance, [l,u) is a time interval including the time points between l and u, and l but not u. A **temporal set** is a set of disjoint time intervals. In other words, a temporal set is a set of time points which can be grouped into disjoint time intervals. Some example of temporal sets are {[5,10)}, {[5,10), [20,30)}, etc.

The fundamental construct of our model is a temporal atom, <t,v> where t is a temporal set and v is a value. The temporal atom denotes that the value v is valid over the time duration represented by the temporal set t. It represents an attribute's value. The history of an attribute is represented as a set of temporal atoms. Furthermore, attribute values can even be relations whose tuple components are made up of temporal atoms or previously defined relations. As is seen, we do not require that the relations be in first normal form. In other words, N1NF (nested) relations are used to model histories of complex objects. Atomic, time invariant, attributes are also allowed. They can even be modelled as temporal atoms where validity interval is the same as the time over which the related object exists in the database. For the sake of uniformity we assume that all attributes are modelled as temporal atoms in the remainder of the paper.

The concept of a temporal set is first introduced by Gadia as the temporal domain [Gadia 87]. It is later used by other researchers [McKenzie & Snodgrass 87]. In our previous work, we used triplets for modelling historical data [Tansel 86]. A triplet consists of a time interval and a value valid over this interval. A temporal atom is equivalent to one or more triplets, that is, one triplet for each time interval in the temporal set and the value part is the same as the value component of temporal atom. For instance, the temporal atom <{[5,10),[12,15)},a> is equivalent to two triplets which are <[5,10),a> and <[12,15),a>.

Figure 1 depicts a nested historical relation, DEPT (departments) which we adopted from [Jaeschke 84]. Attributes of DEPT are department number(Dno),

DEPT

Dno	DmgrX	Project					Equip		DdescX
	Dmgr	Pname	PleaderX	Emp		PdescX	Ino	Idesc	Ddesc
			Pleader	Eno	Ename	Pdesc			
<0,n,1>	<0,15,Loves> <15,n,Lord>	<5,30,x0>	<5,30,y0>	<5,10,23> <8,25,25> <8,30,27> <5,15,30>	<5,10,Hover> <8,25,Smith> <8,30,Tom> <5,15,Ann>	<0,n,Pms>	<0,n,10> <5,n,17>	<0,n,typew> <5,n,PC>	<0,20,Service> <20,n,Info>
		<0,n,x1>	<0,n,y1>	<3,n,88> <5,15,97> <10,n,23>	<3,n,Lewis> <5,15,Cole> <10,n,Hoover>	<0,n,Sql>	<20,40,20>	<20,40,PBX>	
		<5,n,x2>	<5,25,y2> <25,n,y5>	<5,n,99> <15,25,97> <5,15,60> <12,n,55> <40,n,97>	<5,n,Martin> <5,15,Cole> <5,15,Ann> <12,n,Tom> <40,n,Cole>	<0,n,Qmf>			
		<0,n,x3>	<0,n,y3>	<5,25,11> <5,n,15>	<5,25,Bill> <5,n,Sam>	<0,n,Cms>			
<0,n,2>	<0,10,G> <10,15,Bill> <15,25,Loves> <25,n,Sadri>	<40,n,x0>	<40,n,y0>	<40,n,41> <45,50,5>	<40,n,Liz> <45,50,Ann>	<40,n,Pms>	<8,12,12> <0,n,10>	<8,12,PS/2> <0,n,type>	<0,n,Syst>
		<0,n,x4>	<0,n,y4>	<0,n,60>	<0,n,Croft>	<0,n,Iso>			

DEPT(Dno,DmgrX(Dmgr),Project(Pname,PleaderX(Pleader,Emp(Eno,
Ename),Equip(Ino,Idesc),DdescX(Desc))

Figure 1. Example Nested Historical Relation

department manager (DmgrX), Project, Equipments (Equip) used in the department, and department description (DdescX). All attributes, except Dno, are relations. DmgrX is a unary relation whose only attribute is Dmgr. On the other hand, Project attribute of DEPT is another nested historical relation whose attributes are project name (Pname), project leader (PleaderX), employees working for the project (Emp) and project description (PdescX). All attributes of Project are relations but Pname. Emp is another relation with two attributes employee number (Eno) and employee name (Ename). Equip attribute of DEPT is also a relation with two attributes equipment number (Ino) and equipments description (Idesc). There are two tuples in DEPT relation, one for department number 1 and another one for department number 2. In the tuple for department number 1, there are two tuples in DmgrX subrelation and 4 tuples in Project subrelation. The first tuple of Project subrelation has 4 Emp tuples and the second tuple of Project has 3 tuples and so on. Note that temporal sets in the figure are abbreviated, i.e., $<\{0,1,...,n\},1>$ is written as $<0,n,1>$.

It is possible to refer to the components of a temporal atom. For the temporal atom a, a_V and a_T refer to its value and temporal set components, respectively. Let A be the name of an attribute which can take temporal atoms for values. Then A_V, and A_T represent names for the value and temporal set components of the attribute A.

2.1 Scheme Trees

Let U be a set of attributes, $\{A_1,...,A_n\}$. Each attribute A_i has an associated domain of values denoted as $DOM(A_i)$. $DOM(A_i)$ contains either atomic values or temporal atoms. We will use A, B,..., to denote a single attribute and X, Y,..., to denote sets of attributes i.e., they are the names assigned to the sets of attributes. X is also considered as a high order name (high order attribute) because it is the name assigned to a set of attributes, even though this set may contain only one name. This definition is recursive and a high order name may contain other higher order names. However, the structure of the high order names is hierarchical and can not contain cycles (loops).

The attributes of a nested historical relation are hierarchically organized as a tree which is called a scheme tree. The root of the scheme tree corresponding to a nested historical relation is the relation name. Let $T_1,...,T_n$ be subtrees of the root. Each T_i is either a single attribute name or another tree. Thus, the leaves of the scheme tree are attribute
names and the non-terminal nodes are the higher order names. Hence, attributes (simple or higher order) are the non-root nodes of the scheme tree. Associations among the attributes are represented by the branches of the scheme tree. Figure 2 shows the scheme tree for the example nested historical relation given in Figure 1.

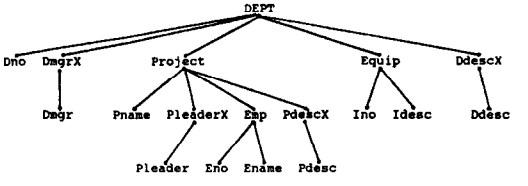

Figure 2. Scheme Tree for DEPT Relation

The leaves of the scheme tree carry the time-stamps for the time-varying attributes. The non-terminal nodes do not carry any time reference. Their time reference can be induced recursively from the time reference of their descendants. Let X be a non-terminal node and $Y_1,...,Y_k$ be its children. Then, the time reference of X, $X_T = Y_{1T} \cup ... \cup Y_{kT}$ The time reference of each Y_i is similarly defined until leaf nodes are reached.

As is clear from the scheme tree, DEPT relation has five subtrees. The first one, Dno, is a leaf node, i.e., a simple (atomic) attribute. On the other hand, DmgrX, Project, Equip and DdescX are higher order attributes, and they are the roots of subtrees. Thus, each department tuple has five components. DmgrX is the root of a subtree whose only descendant is Dmgr. Dmgr is a leaf node. In other words, DmgrX is a unary relation. Project is the root for another nested relation scheme tree which has a subtree, with one simple attribute (Pname) and three higher order attributes (PleaderX, PdescX, Emp). Emp is a subtree of Project and it has two descendants which are leaf nodes, i.e., Eno and Ename. Similarly PleaderX is another subtree with only one attribute, Pleader. Equip is the root of the subtree with two descendants, Ino, and Idesc. DdescX is the subtree corresponding to a unary relation whose only attribute is Ddesc. In DEPT relation, time stamps are attached to only the leaf nodes. For instance, Eno and Ename carry time stamps whereas Emp is a higher order name and does not carry any time reference. Its time reference is defined in terms of the times of Eno and Ename. In other words, $Emp_T = Eno_T \cup Ename_T$. Similarly time reference of Project is $Project_T = Pname_T \cup PleaderX_T \cup Emp_T \cup PdescX_T$. A similar expression can be constructed for DEPT as well. Thus, this condition is a temporal integrity constraint for the nested historical relations.

An instance of a scheme tree R(T) is an element of DOM(R(T)) which is recursively defined as :
$DOM(R(T)) = P(DOM(T_1)X ... X DOM(T_n))$
where $T_1,...,T_n$ are the subtrees of R(T). If T_i is a leaf node whose attribute name is A then $DOM(T_i) = DOM(A)$. Otherwise T_i has children $Y_1,...,Y_m$ and $DOM(T_i) = P(DOM(Y_1)X ... XDOM(Y_m))$.

A historical database consists of nested historical relation schemes and their instances which are sets of historical tuples. Two historical tuples are equivalent if they have the same temporal atom or the nested historical relation in their corresponding attributes. Each historical tuple has an underlying **static tuple** which is obtained by discarding temporal set in temporal atoms and retaining the value part. This definition is applied recursively on the nested historical relation components of tuples. We use historical tuple and tuple interchangeably in the remainder of the paper. If the nested historical relations are not properly structured, they may include great deal of data redundancy. We will discuss this issue in Section 4.

3. Algebraic Operations

Standard relational algebra operations can be applied to nested historical relations with some modifications to accommodate the nested structure and deal with issues created by the time component. We also add some new operations to manipulate historical data. We will briefly discuss them here. The formal definition is beyond the scope of this paper and is the subject of another study [Tansel & Garnett 89].

Unnest (μ), this operation replaces a higher order name by its descendants. In other words, it changes the nesting level of the specified attribute by moving its descendants one level up in the scheme tree. For instance, $\mu_X(R)$ replaces X by its descendants, say $X_1,...,X_k$. For each tuple in R, a family of tuples is created, i.e. one tuple for each value of X. In these tuples, the remaining attributes have the same value. Repeated application of unnest operation normalizes (flattens) a nested historical relation. This is the standard version of the unnest operation and we apply it at the immediate descendants of the root. Similarly, a local unnest operation which can be applied at any level can be defined following the approach of [Jaeschke 84]. A local operation is performed on a subtree of a nested historical relation. Consider the following nested historical relation R.

R

A	B		C
	D	E	F
<0,n,a_1>	<0,5,d_1> <5,n,d_2>	<0,15,e_1> <15,n,e_2>	<0,10,f_1> <10,20,f_2> <20,n,f_3>
<0,40,a_3>	<0,40,d_3>	<0,40,e_3>	<0,40,f_4>

$\mu_B(R)$ gives the following relation:

R'

A	D	E	C
			F
<0,n,a_1>	<0,5,d_1>	<0,15,e_1>	<0,10,f_1> <10,20,f_2> <20,n,f_3>
<0,n,a_1>	<5,n,d_2>	<15,n,e_2>	<0,10,f_1> <10,20,f_2> <20,n,f_3>
<0,40,a_3>	<0,40,d_3>	<0,40,e_3>	<0,40,f_4>

Nest (v) is a kind of aggregate operation which groups the values of specified attributes into a set for the tuples

which agree in the remaining attributes. It is the reverse of the unnest operation. For instance, $v_{X:X1,...,Xk}(R)$ groups the tuples of R with respect to the remaining attributes of R, except $X_1,...,X_k$. Then, $X_1,...,X_k$ components of each group is made into a new set (relation) and the higher order name X is assigned to this relation. In other words, a new nesting level is created. This is the standard version of nest operation applied at the immediate descendants of the root. Local version of nest can similarly be defined. In the above example, nesting R' on the attributes D and E, recreates the original relation R. That is, $R = v_{B:D,E}(R')$.

Projection (π) is the same as the standard projection operation. When a higher order name is projected out all of its descendants are also discarded. Duplicate tuples are eliminated. Tuples, are coalesced (union of temporal sets are taken) if their value parts agree on all their components.

Union (U) is the same as the union operation of relational algebra. It takes tuples from the operand relations. Tuples agreeing on the value parts (i.e., having the same static tuples) are coalesced by taking the union of the temporal sets from corresponding component of each tuple. **Difference** (-) and **Intersection** ($\cap$) operations are also similarly defined. Let r and s be tuples of the relations R and S, respectively. R-S is formed by taking r if r and s does not have the same static tuple. For each attribute, difference of temporal sets from r and s are assigned to their result when r and s have the same static tuple provided that all the attributes in the result have nonempty temporal set. In the case of $R \cap S$, a tuple of the result is formed by taking the intersection of temporal sets of corresponding attributes when r and s have the same static tuple. Again, none of the temporal sets in the result can be empty. Extended set operations for nested relations are defined in [Roth, Korth, and Silberschatz 89]. These operations form the set theoretic operations with respect to a key. Similar keyed versions of set theoretic operations can also be defined for nested historical relations. Such operations are defined for the snapshots extracted from a temporal relation with one level of nesting [Gadia & Yeung 88].

Cartesian Product (X). The attributes of the operand relations are juxtaposed and each tuple of the first relation is concatenated with each and every tuple of the second relation. The scheme tree of the result is formed by replacing the two roots of the operand trees with one root containing all their children.

Selection (σ). The formula in the Selection operation consists of simple conditions, XθY, and complex conditions built out of the simple conditions using the logical connectives $\wedge$, V and $\neg$. X and Y are tuples or tuple components which are children of root or constants. Components of temporal atoms can also be referenced. The symbol θ represents an operator consistent with X and Y, which can be one of simple comparison operators, {=,>,>=,etc}; set comparison operators, {=,⊃,⊇,etc} or the set membership operator ϵ. The time stamps of tuple components (temporal atoms) can be specified in comparison expressions. Set operations {U,∩,-} are used to form new temporal sets in conditions. Local version of the selection operation is similarly defined.

Join (⋈) can be expressed as a combination of cartesian product and selection operations. So, we do not give a special explanation.

Slice is a new operation to manipulate temporal sets of attributes. Given two attributes, it aligns the first attribute's time with respect to the time of second attribute. In other words, the intersection of temporal sets of these attributes is assigned to the first attribute as its time reference. The slice operation is applied on the simple attributes which are children (leaf) of the root. If the operand attributes are not simple or immediate descendants of the root, unnest operations are applied to make the relevant attribute children of the root before employing the slice operation. Similarly, a local version of slice can be defined for the leaf attributes of the same parent. Slice is a kind of intersection operation on the temporal sets. Similarly, other versions of slice operation for set union and set difference of temporal sets can also be defined. Applying $SLICE_{E,D}(R')$, we obtain R" :

R"

A	D	E	C
			F
$<0,n,a_1>$	$<0,5,d_1>$	$<0,5,e_1>$	$<0,10,f_1>$ $<10,20,f_2>$ $<20,n,f_3>$
$<0,n,a_1>$	$<5,n,d_2>$	$<15,n,e_2>$	$<0,10,f_1>$ $<10,20,f_2>$ $<20,n,f_3>$
$<0,40,a_3>$	$<0,40,d_3>$	$<0,40,e_3>$	$<0,40,f_4>$

Slice is a redundant operation which can be expressed by other algebra operations. We define another operation which serves the same purpose as the slice operation but in a different way.

Tranfer-time, replaces the temporal set of an attribute by the temporal set of another attribute. This is one of the basic operations of temporal relational algebra. As an example, $TRANFER\text{-}TIME_{E,D}(R)$ replaces the temporal set of the attribute A with that of the attribute B. Afterwards, set theoretic operations can be applied

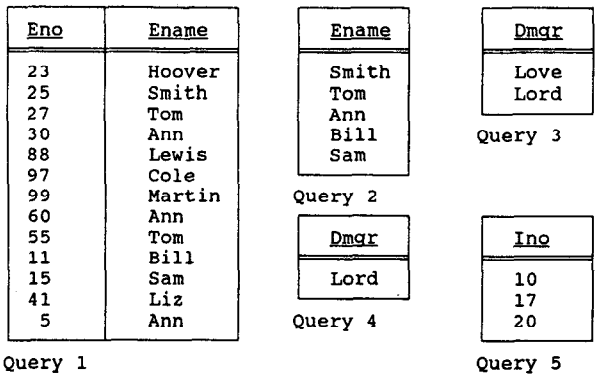

Figure 3. Answers to the Example Queries.

to form union, intersection and difference of time intervals. For instance, $SLICE_{E,D}(R')$ is the same as $R^{*}=TRANFER\text{-}TIME_{E,D}(R')$ followed by $R' \cap R^{*}$. We also add operations to form or to decompose temporal atoms and to discard the time components of temporal atoms. These operations are not included here because of space limitations. Formal definitions of these algebra operations are given in [Tansel & Garnett 89].

3.1 Example Queries

We provide several examples to illustrate the model and the historical relational algebra operations. Local operations are indicated by a bar over the operation symbol. The effect of local operations can be obtained by first unnesting and then applying the general version of the same operation. All the queries refer to the DEPT relation in Figure 1. Note that not all the queries require unnesting. If the query interrogates the children of the root or can be answered by a local operation, it does not require unnesting. Following are complex queries which include conditions involving tuples deep in the DEPT relation. Figure 3 gives the results.

Q1: What are numbers and names of employees in department number 1?

$$\pi_{Eno_V,Ename_V}(\mu_{Emp}(\pi_{Project}(\sigma_{Dno_V=1}(DEPT))))$$

Q2: What are the names of employees who worked in a project when y0 was the project leader?

$$\pi_{Ename_V}(\sigma_{Pleader_V='y0'}(SLICE_{Ename,Pleader}(\mu_{PleaderX}(\mu_{Emp}(\mu_{Project}(DEPT))))))$$

or

$$\pi_{Ename_V}(\overline{SLICE}_{Ename,Pleader}(\bar{\sigma}_{Pleader_V='y0'}(DEPT)))$$

Q3: Who were Tom's department managers?

$$\pi_{Dmgr_V}(SLICE_{Dmgr,Ename}(\mu_{DmgrX}(\mu_{Emp}(\mu_{Project}(\bar{\sigma}_{Ename_V='Tom'}(DEPT))))))$$

Q4: Who are the current managers of all the departments for which Tom has ever worked?

$$\pi_{Dmgr_V}(\sigma_{now \in Dmgr_T}(\mu_{DmgrX}(\bar{\sigma}_{Ename_V='Tom'}(DEPT))))$$

Q5: What equipment was used by any department when y0 was a project leader?

$$\pi_{Ino_V}(\sigma_{Pleader_V='y0' \wedge Pleader_T \cap Ino_T \neq \phi}(\mu_{Pleader}(\mu_{Project}(\mu_{Equip}(DEPT)))))$$

4. Structuring Nested Historical Relations

Nested historical relations can be structured in various ways, each having a different level of data redundancy. Normalization theory in relational database design is based on functional and multivalued dependencies. It attempts to avoid anomalies in update, insertion and deletion operations by reducing data redundancy and arranging relations with respect to inherent relationships among the entities and their attributes. A similar normalization theory has been developed for nested relations in [Ozsoyoglu & Yuan 87, Roth & Korth 87]. In what follows, we adopt this approach for structuring nested historical relations.

From a temporal perspective, attributes of a relation can be classified into different categories with respect to the type and number of values they assume [Shoshani & Kawagoe 86, Shoshani & Segev 87a]. A stepwise constant attribute takes a single value at any time. An example is the salary of an employee. There is only one salary value at any time, which is also valid for a period of time. When a new salary value is assumed as the current value, the previous value becomes no longer valid. Such an attribute in traditional relational theory represents a functional dependency on the relation's key. However, when temporal dimension of the database is considered it becomes a multivalued dependency on the relation's key. There is a set of attribute values (one is valid for a period of time, i.e., a temporal atom) for each key value (multivalued dependency) whereas there is a single attribute value at each time instant (functional dependency) in the case of snapshot relations. Discrete attributes assume a single value at a time point which is valid only at this point but not any other time. In this case, the temporal set consists of a single time point. Values of such attributes also represent functional dependencies at one time point but multivalued dependencies over a time period. Another group of attributes, stepwise constant or discrete, can take a set of values at any time instant. Skills of an employee, is an example for this attribute type. An employee has zero one or more skills at any time. This represents a multivalued dependency. It is still a multivalued dependency when viewed in a time perspective.

A different definition for the scheme trees is given in [Ozsoyoglu & Yuan 87]. This definition constructs the scheme tree for a nested relation with respect to functional and multivalued dependencies among the attributes. Attributes are nodes of the scheme tree, dependencies among the attributes are edges. The ancestors of a node multidetermine (functionally determine) its descendants. Let's consider the functional and multivalued dependencies for the DEPT relation. These dependencies are given below in Figure 4. Note that the first column lists the dependencies without any time dimension. The second column gives the same dependencies in a temporal perspective. That is, the attribute values are temporal atoms. Dno is time invariant; there is only one Dno value throughout the database history for a department. Although we represent it as a temporal atom, it can be considered as a simple (single) value for all purposes. On the other hand, there is a well-defined set of temporal atoms for Dmgr attribute, So, the functional dependency, Dno --> Dmgr of the static case turns into a multivalued dependency, Dno -->> Dmgr in the temporal database. Other dependencies are similarly explained.

Dno --> Dmgr
Dno --> Ddesc
Dno -->> Pname,Pleader,Pdesc
Dno -->> Ino,Idesc
Dno,Pname -->> Eno,Ename
Pname --> Pleader
Pname --> Pdesc
Ino --> Idesc
Eno --> Ename

a. without time

Dno -->> Dmgr
Dno -->> Ddesc
Dno -->> Pname,Pleader,Pdesc
Dno -->> Ino,Idesc
Dno,Pname -->> Eno,Ename
Pname -->> Pleader
Pname -->> Pdesc
Ino --> Idesc
Eno --> Ename

b. with time (attribute values are temporal atoms)

Figure 4. Functional and Multivalued Dependencies for DEPT Relation.

The scheme tree for the DEPT relation is given in Figure 5. Note that the above scheme tree in Figure 5 does not contain high order names, but is equivalent to the scheme tree given in Figure 2 in terms of their information content. They represent the same nested historical relation structure.

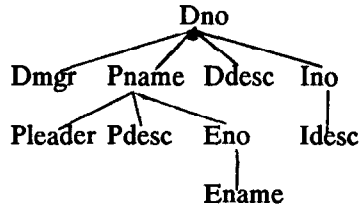

Figure 5. Scheme Tree for DEPT Relation

There is an algorithm giving a correspondence between scheme trees of the type in figure 5 together with dependency information and scheme trees of the type in figure 2.

DEPT relation has redundancies. For instance, the project description (Pdesc) is repeated if a project is transferred from one department to another. Two tuples are created for this project and entire history of Pdesc is included in these tuples. Storing the relevant part of Pdesc history in each tuple, respectively would split the information. Another source of data redundancy is the repetition of Ename. If an employee worked at different times at different projects, employee's name would be repeated in different tuples. The first data redundancy is due to the structure of the scheme tree which does not represent the dependencies among the attributes properly and is caused by the partial dependency of Pdesc on Pname. The scheme tree implies that (Dno,Pname) multidetermines Pdesc which can be obtained from Pname -->> Pdesc by augmentation. The other data redundancy is also caused by a partial functional dependency. That is, (Dno,Pname,Eno) --> Ename can be obtained from Eno --> Ename by augmentation.

A scheme tree is called a normal scheme tree if it does not contain any partial and transitive dependencies [Ozsoyoglu & Yuan 87]. These anomalies are eliminated by decomposing DEPT relation to remove the partial dependencies and creating a scheme forest. Figure 6 depicts the resulting scheme forest for DEPT relation. The nested historical relation schemes in the scheme forest are given in Figure 7.

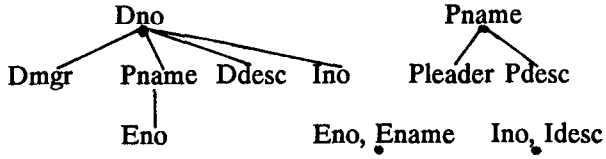

Figure 6. Scheme Forest for DEPT Relation

D1

Dno	DmgrX	Project		Equip	DdescX
	Dmgr	Pname	Emp	Ino	Ddesc
			Eno		

D2

Pname	PleaderX	PdescX
	Pleader	Pdesc

D3

Eno	Ename

D4

Ino	name

D1 (Dno,DmgrX(Dmgr),Project(Pname,Emp(Eno)), Equip(Ino),DdescX(Desc))
D2 (Pname,PleaderX(Pleader),PdescX(Pdesc))
D3 (Eno,Ename)
D4 (Ino,Idesc)

Figure 7. Nested Relation Schemes in the Scheme Forest

A root to leaf path in a scheme tree represents a 4NF decomposition. The scheme forest of Figure 6 implies 8 relations which are all in 4NF, i.e., (Dno,Dmgr), (Dno,Pname,Eno), (Dno,Ddesc),..., (Ino,Idesc). Note that attributes are time stamped. Time can even be factored out and added as another attribute in these 4NF relations. This leads to tuple time-stamping which is followed by many other researchers.

5. Conclusion

In this paper we propose an extension to the relational model for keeping temporal information. The model involves nested relations which organize data in a hierarchical manner. Each attribute value is either an atomic value or a temporal atom which consists of two components, a temporal set and a value valid over the time period(s) represented by the temporal set. We call these relation as nested historical relations. They can be used in modelling temporal variation of complex objects in CAD/CAM, office automation, etc.

There are a few points we want to touch briefly. Nested historical relations may look complicated at first glance. However, when the nested historical relations are properly structured in the form of normal scheme trees to avoid data redundancy, the relations exhibit a natural and intuitive conceptualization of the reality. Nested historical relations can be structured in different ways if the associations among the entities are many to many. In the case of one to many associations, the entity at the one side of relationship is a natural candidate for the parent of the other entity. Different structures would be more convenient for different groups of users. When flat relations are used to represent association, the entities are equally represented. On the other hand, a nested historical relation arranges the entities in a hierarchy and groups the data according to this hierarchical structure. However, the hierarchical structure can be manipulated into different structures by a sequence of nest/unnest operations. A nested historical relation represent the associations among entities, perhaps at several levels, depending on the type of associations. Attributes of an entity is represented at only one level of nesting.

A normal scheme tree is also equivalent to a 4NF decomposition, as it is shown in [Ozsoyoglu & Yuan 86]. The same is also true for the nested historical relations. This also puts the dichotomy of attribute versus tuple time-stamping into proper perspective. Nested historical relation scheme corresponding to a normal scheme tree is equivalent to a set of 4NF relation schemes where each attribute has a single temporal atom as the value. When temporal sets are factored out from a 4NF relation they can be assigned as the tuple time-stamps. This correspondence establishes the equivalence of attribute and tuple time-stamping. As is clear, a nested historical relation naturally groups related data with attribute time-stamping, whereas the equivalent 4NF relations with tuple time-stamping split the information. This also forms the rational for our prior intuitive approach which represents entity histories in nested relations with one level of nesting. Instead of creating many small 4NF relation, the data is represented in a concise manner as a scheme normal tree.

One criticism is that a series of unnest operations is needed to extract information from nested historical relations. This is not always the case, because local versions of the operations are defined. They allow extraction of data without any unnest operation for some queries. Furthermore, it is possible to optimize the query processing methodology and to process nest/unnest operations together with other algebra operations, i.e., selection, join operations, etc. [Selinger et al., 79]. Using algebra operations may also seem complicated at first glance. But, the user will interact with the database by a user friendly query language. For instance, versions of SQL for nested relations have already been proposed [Dadan et al., 85]. Similar enhancements can be added to SQL for nested historical relations as well. Moreover, a graphical query language can also be modified for this

purpose i.e., Time-by-Example [Tansel, Arkun & Ozsoyoglu 89]. All these points are currently under investigation and we expect to report the results in future papers.

Acknowledgement

The research of the first author is supported partially by the research grant PSC-CUNY #6-67299.

References

[Anderson 81]
Anderson, T.L., 'The Database Semantics of Time', Ph.D. Diss. Univ. of Washington, 1981.

[Ariav 86]
Ariav, G., 'A Temporally Oriented Data Model', ACM TODS, Vol 11, No. 4, 1986.

[Ben-Zvi 82]
Ben-Zvi, J., 'The Time Relational Model', Ph.D. Diss., UCLA, 1982.

[Clifford & Warren 83]
Clifford, J., Warren, D.S., 'Formal Semantics for Time in Databases', ACM TODS, Vol. 6, No. 2, June 1983.

[Codd 70]
Codd, E.F. 'A Relational Model of Data for Large Shared Databanks', Comm. of the ACM, Vol. 13, No.6, June 1970.

[Copeland & Maier 84]
Copeland, G., Maier, D., 'Making Smalltalk a Database System', Proc. of ACM SIGMOD Conf., 1984.

[Dadan, et al., 85]
Dadan, P.M., at al., 'A DBMS Prototype to Support NF2 Relations: An Integrated View on Flat Tables and Hierarchies', ACM SIGMOD Conf., 1986.

[Fisher & Thomas 83]
Fisher, P., Thomas, S., 'Operators for Non-First Normal Formal Relations', Proc. of 7th Intl. Computer Software applications Conf., 1983.

[Gadia 86]
Gadia, S. 'A Multi Homogeneous Model for Temporal Databases', 2nd Int. Conf. on Data Engineering, 1986.

[Gadia 88]
Gadia, S.K., 'A Homogeneous model and Query Languages for Temporal Databases'. ACM TODS, Vol. 13, No. 4, 1988.

[Gadia & Yeung 88]
Gadia, S. K., Yeung C. S., 'A Generalized Model for a Relational Temporal Database', Proc. of the ACM SIGMOD Conf. 1988.

[Garnett & Tansel 89]
Garnett, L., Tansel, A.U., 'Equivalence of the Relational Algebra and Calculus Languages for Nested Relations', Submitted for publication, 1989.

[Goldberg & Robinson 83]
Goldberg, A., Robinson, D., D., Smalltalk 80: The Language and its implementation, Addison-Wesley, 1983.

[Jaeschke & Schek 82]
Jaeschke, G., H. Schek, 'Remarks on the Algebra of non First Normal Form Relations', Proc. of ACM PODS Conf., 1982.

[Jaeschke 84]
Jaeschke, G., 'Nonrecursive Algebra for Relations with Relation Valued Attributes', Technical Report, IBM Heilderbeg Scientific Center, 1984.

[Khoshofian & Copeland 86]
Khoshofian, S. N., Copeland, G., 'Object Identity', Proc. of OOPSLA 86, ACM, New York, 1986.

[Lum, et al., 84]
Lum, V., et al., 'Designing DBMS Support for the Temporal Dimension', Proc. of ACM SIGMOD Conf., 1984.

[McKenzie & Snodgrass 87]
McKenzie, E., Snodgrass, R., 'Supporting Valid Time: An Historical Algebra', Technical Report, TR87-008, Computer Science Department, UNC at Chapel Hill, 1987.

[Navathe & Ahmed 87]
Navathe, S.B., Ahmed, R., 'TSQL - A Language Interface for History Databases', Conference on Temporal Aspects of Information Systems', 1987.

[Ozsoyoglu, Ozsoyoglu, and Matos 87]
Ozsoyoglu, G., Ozsoyoglu, M.Z., Matos, V., 'Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions', ACM TODS, Vol. 12, No. 4, 1987.

[Ozsoyoglu & Yuan 87]
Ozsoyoglu, M.Z., Yuan, Li-Yan, 'A New Normal Form for Nested Relations', ACM TODS, Vol. 2, No. 1, 1987.

[Roth, Korth, and Silberschatz 88]
Roth, M.A., Korth, H.T., Silberschatz, A., 'Extended Algebra and Calculus for Nested Relational Databases', ACM TODS, Vol 13, No. 4, 1988.

[Roth & Korth 87]
Roth, M.A., Korth, H.F., 'The Design of ¬ 1NF Relational Databases into Nested Normal Form', ACM SIGMOD Conf., 1987.

[Schek & Scholl]
Schek, H.J., Scholl, M.H., 'The Relational Model with Relation-valued Attributes', Information Systems, Vol. 11, No. 2, 1986.

[Segev & Shoshani 87a]
Segev, A., Shoshani, A., 'Modelling Temporal Semantics', Temporal Aspects of Information Systems Conf., 1987.

[Segev & Shoshani 87b]
Segev, A., Shoshani, A., 'Logical Modelling of Temporal Data', ACM SIGMOD Conf., 1987.

[Selinger, et al., 79]
Selinger, P.G., et al., 'Access Path Selection in Relational Database Management Systems', ACM SIGMOD Conf., 1979.

[Shoshani & Kawagoe 86]
Shoshani,A., Kawagoe, K., 'Temporal Data management', VLDB 12, Kyoto, Japan, 1986.

[Snodgrass 87]
Snodgrass, R., 'A Temporal Query Language', ACM TODS, Vol. 12, No. 2, 1987.

[Tansel & Clifford 85]
Tansel, A.U., Clifford, J., 'On a Historical Relational Algebra: Two Views', Proc. of the ACM SIGMOD Conf., 1985.

[Tansel 86]
Tansel, A.U., 'Adding Time Dimension to Relational Model and Extending Relational Algebra', Information systems, Vol 13, No 4, 1986.

[Tansel, Arkun, and Ozsoyoglu 89]
Tansel, A.U., Arkun, M.E., Ozsoyoglu, G., 'Time-by-Example Query Language for Historical Databases', IEEE Transactions on Software Engineering, April 1989.

[Tansel 87a]
Tansel, A.U., 'A Statistical Inference to Historical Relational Databases', Proc. of 3rd International Conference on Data Engineering, 1987.

[Tansel & Garnett 89]
Tansel, A.U., Garnett, L., 'Relational Algebra for Nested Historical Relations', manuscript in preparation, 1989.