ELSEVIER

Data & Knowledge Engineering 17 (1995) 59–86

DATA & KNOWLEDGE ENGINEERING

Functional requirements for historical and interval extensions to the relational model

Nikos A. Lorentzos[a,*], Yannis Manolopoulos[b]

[a]*Informatics Laboratory, Agricultural University of Athens, Iera Odos 75, 118 55 Athens, Greece*
[b]*Department of Informatics, Aristotle University, 54006 Thessaloniki, Greece*

Received 27 February 1995; accepted 4 August 1995

Abstract

In recent years a lot of divergent approaches have been proposed for the management of Valid Time (Historical) data. However, no systematic effort has been reported, concerning the identification of the properties of such a model. In the present work, it is initially shown that all proposed Valid Time (VT) models can be applied to areas of practical interest, not related to VT data management. All VT models are also classified with respect to two orthogonal parameters, the way time is represented, and the level at which it is incorporated (tuple or attribute). This enables to identify that two *reference* VT models can be specified, VT-1NF, a simple extension to the conventional relational model and VT-NESTED, a more general one, which supports, in addition, relation-valued attributes. The properties of these models are identified. Two more reference *interval* relational models are proposed, I-1NF and I-NESTED, which support any type of *interval* data. I-NESTED is the most general, in that it can be applied to all the areas in which all others are applicable. Results are also reported, concerning the evaluation of all VT models.

Keywords: Data type; Data model; Time point; Time interval; Valid time data; Historical data; Interval data; Historical database; Interval database

1. Introduction

Commercial Database Management Systems (DBMS) have been based on models which are suitable for the representation and manipulation of the most recent snapshot of a subset of the real world, we are interested in. Using such a model, we can either record new pieces of data or delete/update any piece which does not reflect this most recent snapshot. Such a model is called *snapshot*. However, in recent years, a continuously growing interest can be witnessed, in the modelling of *Historical Data* or *Valid Time* (VT) data, according to a new terminology proposed in [13], i.e. data which was valid at some previous state or will be valid at some state in the future. Indeed, although Codd's relational model [6] can handle certain

* Corresponding author. Email: lorentzos@auadec.aua.ariadne-t.gr

SSDI: 0169-023X(95)00026-7

types of such data, Codd himself admits [7] that the management of VT data is a distinguishing problem.

On the other hand, research in VT databases has mainly focused in the formalisation of VT extensions to the relational model. However, pioneers in the area admit [1, 3, 4, 19] that the properties which a VT DBMS should satisfy are far from obvious, as is also verified by the fact that over a dozen of diverge proposals have appeared in the literature [1–3, 5, 9, 12, 14–16, 19, 21, 23–29, 32]. Today, the properties, which a VT data model should satisfy, is still considered to be an open problem, in spite of TSQL2, a *Temporal extension to TSQL2*, which was recently proposed [27].

One of the purposes of this present paper is the identification of the properties of a *reference VT model*. Based mainly on the representational capabilities of the most diverge VT extensions to the relational model, which have been proposed, we identify, in particular, the properties of two *reference VT models*, VT-1NF and VT-NESTED. They are, respectively, extensions to the S-1NF, the *Snapshot* model which satisfies First Normal Form (1NF) [6], and S-NESTED, the *Snapshot* model which neglects 1NF [11]. Although it could be argued that the identification of the properties of a reference model cannot be based on objective criteria, the following should be noted:

(i) Our work has been based on the necessity to support certain requirements, which are of practical interest, according to the opinion of many researchers.
(ii) We have taken into consideration the properties of the VT models which have been proposed. Indeed, it is shown that the models which represent simple extensions to Codd's, have many properties in common, in spite of their seemingly many differences.
(iii) We also report on the evaluation of VT-1NF with respect to actual user requirements, as undertaken within ORES[1], a project aiming at the management of VT data.

It should also be noted that there is no objective way, either, to show that the set of properties of each reference VT model is exhaustive. Again, however, we notice that although Codd's model [6] has been adopted, its properties are not exhaustive if one requirement is either the satisfaction of the transitive closure property [33] or recursiveness. Although a DBMS, which is based on [6], relieves application programmers from a substantial amount of coding, the use of programming languages is still necessary. As a general principle, if the properties of a model are too few, it is most likely that a substantial number of user requirements will not be satisfied. At the other end, a model may have arbitrarily many properties but, in this case, it may turn out that it is too complex to use. Only one relatively objective measurement of its functionality can be made, how effective it is, in satisfying quite common user requirements within the area it is dedicated to serve.

One characteristic of VT models is that they implicitly incorporate *time points* and/or *time intervals* as primitive data types. However, the paper also serves a second purpose, it shows that all the models have the potential to be applied to a wide range of areas of practical interest, not related, in any way, to the management of VT data, by incorporating a generic type of *point* and/or *interval* data. Thus, we also identify the properties of two more reference models, I-1NF and I-NESTED. We show that they are proper supersets of VT-1NF and VT-NESTED, respectively. I-NESTED is the most general of all, in that it inherits the

[1] ORES comes from the Greek word ΩΡΕΣ, which stands for *HOURS*.

properties of all other models. Although the management of interval data has many application areas, we are not aware of any piece of work, to identify the properties which a relevant model should satisfy.

The remainder of this paper is outlined as follows: In Section 2 we show that all proposed VT models can be applied to areas of practical interest, not related to VT data management. All VT models are then classified with respect to two orthogonal parameters, the way time is represented, and the level at which it is incorporated (tuple or attribute). This enables to identify that two reference VT data models can be proposed, VT-1NF and VT-NESTED, which are consistent extensions to S-1NF and S-NESTED, respectively. Their properties are presented in Sections 3 and 4. In Section 5 we evaluate all VT models which are simple extensions to Codd's, with respect to VT-1NF. We thus show that they have many similarities, in spite of their major differences. We also report on the evaluation of VT-1NF with respect to all others and against actual user requirements. In Section 6 we show that *time* is characterised by a high degree of abstraction. Based on this observation, we identify the necessity to define two more radical extensions to S-1NF and S-NESTED.

2. Valid time models

In this section we provide a brief overview of the most characteristic VT models, in terms of data representation and operations. In all models, time is considered as discrete and totally ordered. Hence, for each model, we choose another discrete, totally ordered set. Thus, we show that the particular model can be applied to areas other than VT databases. Note that, for easiness of reading, we adopt a uniform syntax for the operations defined in each model, equivalent to the original.

2.1. Navathe and Ahmed's model

In relation SALARY (Fig. 1) we see how VT data, concerning the evolution of employee salaries with respect to time, can be represented in this approach [21]. In attributes From and To we record the boundaries of time intervals, closed on both sides. Thus, the first tuple shows that John's salary during [10, 20], was 10k. The second tuple shows that this salary was again 10k, during [30, 40].

SALARY

Name	Amount	From	To
John	10k	10	20
John	10k	30	40
Alex	14k	5	30
Alex	18k	31	60

SERVICE

Desk	Name	From	To
1	John	AAA	CCC
1	John	EEE	FFF
2	Alex	GGG	HHH
2	Alex	KKK	LLL

Fig. 1. Relations in Navathe and Ahmed's approach.

However, if time is replaced by the set A = {'AAA', 'BBB', . . . , 'ZZZ'}, which is lexicographically ordered, then SERVICE (Fig. 1) shows a *non*-VT relation, which concerns employees who serve citizens in a public organization: Now, the first tuple shows that citizens whose name starts from AAA, BBB, or CCC, are served by John, at desk 1. The second one shows that John also serves those whose name starts from EEE-FFF. We now demonstrate how the operations defined in [21] can manipulate relation SERVICE.

The *Union* of SERVICE with a *union-compatible* relation S of a unique tuple, (1, John, DDD, DDD), yields R1 (Fig. 2). That is, the tuples with the *overlapping* or *adjacent* intervals, [AAA, CCC], [DDD, DDD] and [EEE, FFF], *merge* into a single tuple, whose value, for attributes From and To, represent the interval [AAA, FFF]. Inversely, the *difference* R1-S yields SERVICE, again. *Selection* has been extended by predicates like *overlaps*, *follows* etc., which can be applied to pairs of intervals, thus enabling to retrieve a subset of the tuples of a relation. *Projection* has not been changed. *Cartesian Product* is not formalised but four distinct types of a *Join* are defined: Two of them yield a relation with one pair of From, To attributes, whereas the other two yield a relation with two such pairs. The remainder operations function as follows:

TIME-SLICE[['BBB', 'GGG']](SERVICE), retrieves the first three tuples of SERVICE, because the intervals in them intersect with ['BBB', 'GGG'].

INNER-TIME-VIEW[['BBB', 'CCC']](SERVICE) retrieves the first tuple of SERVICE because ['BBB', 'CCC'] is a sub-interval of ['AAA', 'CCC'].

OUTER-TIME-VIEW[['DDD', 'FFF']](SERVICE) retrieves the second tuple of SERVICE because ['DDD', 'FFF'] is a super-interval of ['EEE', 'FFF']. A VT extension to SQL is also defined.

The representation of VT data, as described in the above model, is also adopted in the models proposed by Jones and Mason [15], Ben-Zvi [2], Sadeghi [23], Snodgrass [26], Sarda [24, 25] and Lorentzos and Johnson [16], except that the time intervals may be open to the left and closed to the right. The operations are not defined formally in all these models. Individual remarks on the operations, are as follows:

In principle, *Union* and *Difference* function as in Navathe and Ahmed's. Similarly, some new predicates [15, 25, 26] and functions [25, 26] are defined.

In [2] one operation retrieves the tuples which are valid at a user-supplied time.

In [23] one operation retrieves the From, To values of selected tuples. *Cartesian Product*

R1

Desk	Name	From	To
1	John	AAA	FFF
2	Alex	GGG	HHH
2	Alex	KKK	LLL

R2

Desk	Name	From	To
1	John	AAA	BBB
1	John	BBB	CCC
1	John	EEE	FFF
2	Alex	GGG	HHH
2	Alex	KKK	LLL

Fig. 2. Relations in Navathe-Ahmed's and Sarda's approach.

has been extended in a way which incorporates intersection semantics. For example, assume that a relation S(A, From, To) has a single tuple, (1, HHH, KKK). We then notice that the interval [HHH, KKK] intersects with the intervals of the last two tuples of SERVICE. Therefore, the application of *Cartesian Product* to SERVICE and S yields a relation R(Desk, Name, A, From, To) with two tuples,

(2, Alex, 1, HHH, HHH),
(2, Alex, 1, KKK, KKK),

whose intervals represent the intersection of the intervals of SERVICE with the interval in S.

The approaches in [25, 26] also support *event* relations, relations with only one attribute, Time, in place of attributes From and To. Time can also be represented as a point in [16].

In Sarda's approach [24, 25] intervals are closed to the left and open to the right. The traditional *Projection* and *Cartesian Product* operations do not change. The former may result in a relation without From To attributes and the latter may yield a relation with two pairs of such attributes. Two more operations in this approach are the following:

EXPAND(SERVICE) yields R2 in Fig. 2, i.e. each interval is *split* into *elementary* intervals, i.e. their value for attribute To equals that for attribute From, increased by one. (Recall that in this approach, the values x, y, for attributes From, To, respectively, are interpreted as an interval [x, y).)

Inversely, COALESCE(R2) returns SERVICE, again. More generally, this operation *merges* into one, all the tuples whose values for From and To form adjacent or overlapping intervals. Thus, if the tuples of a relation S(A, B) are

(1, AAA, EEE),
(1, CCC, KKK),
(1, KKK, MMM),
(1, PPP, TTT)

then COALESCE(S) returns a relation with tuples

(1, AAA, MMM),
(1, PPP, TTT).

The approach by Lorentzos and Johnson [16] appeared shortly before Sarda's [24] and it is more general: Firstly, two operations, UNFOLD and FOLD, are functionally equivalent to EXPAND and COALESCE, respectively. Secondly, a relation may have more than one pair of From, To attributes. Finally, the *Union* and *Difference* operations can be applied to relations with more than one pair of such attributes.

2.2. Tansel's model

Fig. 3 shows an instance with scheme DEPARTMENT, in Tansel's approach [28]. The instance consists of two tuples. In this relation we record, for each department, its number, current manager, and current employees. In addition, we record, in attribute Effectiveness, DEPARTMENT

No	Manager	Effectiveness	Employees
1	Alex	{[100, 300) 1, [300, 500) 2, [500, 600) 3}	{Alex, Carol, Mark}
2	John	{[100, 250) 1, [250, 400) 2, [550, 600) 2}	{Jim, John, Roger}

COMPETITION

No	Chief	Record	Members

Fig. 3. One instance in Tansel's approach, under two distinct schemes.

history records of the effectiveness of the department. Specifically, a value of a tuple for attribute Effectiveness is a set, whose elements are pairs of the form (*time-interval*, *value*). Thus, for department 1 we can see that during the time interval [100, 300), the department was marked with 1. Then, during [300, 500) and [500, 600), the mark was 2 and 3, respectively. Clearly, therefore, DEPARTMENT contains VT data.

Now, consider another scheme, COMPETITION(No, Chief, Record, Members), which is depicted at the bottom of Fig. 3. It is used to store the daily records of a climbing competition: For each team, we record its number, chief and members. As opposed to the previous scheme, in attribute Record, we now record a set whose elements are pairs of the form (*integer-interval*, *time-point*). In this attribute we now keep records of the daily effectiveness of each team. Assuming, therefore, that this relation has exactly the same data with DEPARTMENT, we can see that, on the first day of the competition, team 1 started at height 100 metres and reached 300 metres. The next two rows of this tuple show the record of the team on the second and third day. If we compare the instance, under the two distinct schemes, we notice that, a pair in Effectiveness is interpreted as (*time-interval*, *value*) and, in the Record, as (*integer-interval*, *time-point*). However, we argue that COMPETITION again contains VT data, except that the roles of the *time* and *value* components, in attribute Record, have been reversed: Now, intervals represent *heights* and values represent *time*. Hence, according to Tansel's terminology [28], we now have values for time, *stamped* by height intervals. Obviously, the operations defined in this model can directly be applied to COMPETITION. We outline them briefly.

The *Union* of COMPETITION with a union-compatible relation S of a single tuple,

$$(2, \text{John}, \{[400, 550), 2\}, \{\text{Jim}, \text{John}, \text{Roger}\}),$$

yields a relation R, whose first tuple matches the first tuple of COMPETITION, and its second tuple is

2	John	{[100, 250) 1, [250, 600) 2}	{Jim, John, Roger}

Thus, adjacent and/or overlapping intervals, in attribute Record merge again. Operation R-S yields COMPETITION again. *Selection* incorporates predicates that can be applied to set-valued attributes and enables the retrieval of a subset of tuples. *Projection* and *Cartesian Product* are defined as in the traditional way.

To save space in the sequel, let us assume that COMPETITION contains only the first tuple. We then demonstrate the functionality of two operations, UNPACK and PACK, originally defined in [22]:

UNPACK[Members](COMPETITION) yields R1, with three tuples (Fig. 4). Similarly, UNPACK[Record](COMPETITION) yields R2 (Fig. 4), with three tuples. PACK performs

R1

No	Chief	Record	Members
1	Alex	{[100, 300) 1, [300, 500) 2, [500, 600) 3}	Alex
1	Alex	{[100, 300) 1, [300, 500) 2, [500, 600) 3}	Carol
1	Alex	{[100, 300) 1, [300, 500) 2, [500, 600) 3}	Mark

R2

No	Chief	Record	Members
1	Alex	[100, 300) 1	{Alex, Carol, Mark}
1	Alex	[300, 500) 2	{Alex, Carol, Mark}
1	Alex	[500, 600) 3	{Alex, Carol, Mark}

R3

No	Chief	RV	RF	RT	Members
1	Alex	1	100	300	{Alex, Carol, Mark}

Fig. 4. Result of operations in Tansel's approach.

the inverse, i.e. when applied to R1 on Members (to R2 on Record) it returns the first tuple of COMPETITION.

To save space again, we assume that R2 contains only the first tuple. Then another operation, TRIPLET-DEC[Record](R2), decomposes attribute Record into three distinct attributes (see R3, Fig. 4). Inversely, TRIPLET-FORM[RV, RF, RT](R3) returns again the first tuple of R1.

Some more operations, in this approach, are proved to be expressible in terms of the above. Finally, we do not demonstrate a final operation, AGGREGATE-FORMATION; it incorporates aggregate functions and can be used to non-VT relations, too.

A variation of this model is proposed by Tansel and Garnet in [29]. The major difference, from the approach described above, is that PACK and UNPACK are replaced by operations NEST and UNNEST, respectively, originally defined in [11].

2.3. *Clifford's model*

The most recent version of this model has been formalised in [5]. VT time can be represented either as a point or as an interval closed on both sides. If we replace Time by the set of integers, we can apply this approach (Fig. 5), in order to represent the average pH values and average CaCO3 composition, for various profiles (holes vertical to the surface) and horizons (intersection of holes with layers of land). We have chosen to represent time as point, in order to elaborate the difference of this approach from Tansel's.

LAND has two tuples and each of them is associated with a *depthspan* (*lifespan* in Clifford's approach) shown in the last column of LAND. Also, each attribute is associated with an interpolation function. This enables to deduce the value of a tuple for an attribute, at a depth not explicitly recorded in the relation. The data recorded in each attribute is valid only at the intersection of the depth, which is deduced by the interpolation function, with the depthspan of the tuple: Assume, for example, that the interpolation function associated with the pH attribute is a step function. It can then be deduced that, for profile 1 and horizon 1, the average pH value is 8.0 for the depth [0, 30]. This value is 8.2 for the depths [30, 50] and [80, 200].

Two variations for *Union* are provided: The first functions the ordinary way: The result relation is obtained by appending in it the tuples of the two relations which participate in *Union*. The second merges tuples: Assuming that LAND contains only the second tuple, its union with S (Fig. 5) yields R1 in the same figure. We notice that the depthspan of R1 is the union of the depthspans of the two relations. Inversely, R1-S yields again the second tuple of LAND. *Projection* and *Cartesian Product* are defined the ordinary way (except that *Product* may result in a relation containing some null values).

SELECT-IF retrieves a subset of the tuples of a relation.

SELECT-WHEN retrieves the *portions of those tuples* which satisfy the selection formula. The depthspan of each tuple, in the result relation is a subset of the depthspan of the tuple in the original relation. For example, SELECT-WHEN[pH = 8.2](LAND) retrieves R2 (Fig. 5).

TIME-SLICE is similar to SELECT-WHEN except that it gets, as an argument, a depth value: Thus, assuming that LAND contains only the first tuple, then TIME-SLICE[[10, 20]](LAND) returns R3.

LAND

Pno	Hno	pH	CaCO3	depthspan
1	1	0 8.0 30 8.2	0 22.8 60 22.4	[[0, 50], [80, 200]]
1	2	0 8.4 50 8.3	0 32.5 80 38.6	[[0 , 150]]

S

Pno	Hno	pH	CaCO3	depthspan
1	2	50 8.3	150 37.0	[[150, 200]]

R1

Pno	Hno	pH	CaCO3	depthspan
1	2	0 8.4 50 8.3	0 32.5 80 38.6 150 37.0	[[0, 200]]

R2

Pno	Hno	pH	CaCO3	depthspan
1	1	30 8.2	0 22.8 60 22.4	[[30, 50], [80, 200]]

R3

Pno	Hno	pH	CaCO3	depthspan
1	1	10 8.0	10 22.8	[[10, 20]]

Fig. 5. Relations in Clifford's approach.

WHEN returns the depth values at which the data which satisfy the selection formula are valid, i.e. WHEN[pH = 8.4](LAND) returns [0, 50]. Finally, some variations of a *Join* have also formalised, but they are not discussed here, for brevity reasons.

2.4. Gadia's model

We choose the set

$$\text{PLATE} = \{\text{XY} \mid \text{'A'} \leq \text{X} \leq \text{'Z'}, 0 < \text{Y} < 10000, \text{Y integer}\},$$

which can substitute the role of time in the VT model defined in Gadia's approach [9]. This set is totally ordered, in the ordinary lexicographic way.

CARPLATES (Fig. 6) shows the departments of a public organization which are authorised to issue car plates. For each department, the relation also shows the employees who have been assigned this task. Thus, D1 is responsible for the car plates in [AAA0001, CCC0001). Those in [AAA0001, BBB0001) are processed by Tom whereas those in [BBB0001, CCC0001) are processed by Mary. Each attribute value in this approach must be accompanied by a *plate domain* (*temporal domain* in Gadia's approach), which is defined as the union of car plate intervals. For example, the plate domain of department D2 equals [EEE0001, GGG0001) ∪ [KKK0001, MMM0001). In CARPLATES, we notice that the plate domain of D1 (Fig. 6) equals the union of the plate domains of the values recorded in attribute Manager, of this

CARPLATES

Department	Manager
[AAA0001, CCC0001) D1	[AAA0001, BBB0001) Tom [BBB0001, CCC0001) Mary
[EEE0001, GGG0001)U [KKK0001, MMM0001) D2	[EEE0001, FFF0001) Alex [FFF0001, GGG0001) Mark [KKK0001, MMM0001) John

R1

Department	Manager
[AAA0001, DDD0001) D1	[AAA0001, BBB0001) Tom [BBB0001, DDD0001) Mary
[EEE0001, GGG0001)U [KKK0001, MMM0001) D2	[EEE0001, FFF0001) Alex [FFF0001, GGG0001) Mark [KKK0001, MMM0001) John

Fig. 6. Relations in Gadia's approach.

same tuple. Gadia asserts that this property (termed *homogeneity*) must be satisfied by all the tuples in every relation.

The *Union* of CARPLATES with a relation S, of a single attribute,

(([CCC0001, DDD0001) D1), ([CCC0001, DDD0001) Mary)),

yields R1 in Fig. 6. Inversely, R1-S yields CARPLATES again. *Projection* is defined the usual way. The same also applies to *Cartesian Product* except that, because of the homogeneity property, the plate domain of each tuple in the result relation equals the intersection of the plate domains of the tuples from which it is deduced. *Select* retrieves *portions of tuples*: Thus, SELECT[Manager = 'Mark'](CARPLATES) yields a single tuple,

(([FFF0001, GGG0001) D2), ([FFF0001, GGG0001) Mark)).

Finally, one *operator*, TDOM, returns the plate domain of a relation.

2.5. McKenzie and Snodgrass's model

To show how the model defined in [19] can be applied to an area, another than VT databases, let the role of time be played by the set T = {0.0, 0.1, . . . , 99999.9}. T is a totally ordered subset of the set of reals. Let us now consider relation RAIL (Fig. 7): For each portion of a railway system, we record, in this relation, the employee who inspects various parts of the railway track. Thus, the first tuple shows that Alex supervises tracks 0.0, 0.1, . . . , 4999.9 Km of the route Athens–Lamia. In each attribute of a relation we must record a *value* component and, optionally, a *valid* component. The latter represents a subset of T. For example, in the first two attributes of RAIL we record only value components, whereas in the third one we record both value and valid components. Each relation must have at least one attribute with both *value* and *valid* components. Also, a relation may not have two tuples which satisfy the property that *all their value components are pairwise identical.*

The *Union* of RAIL with a relation S of two tuples,

(Athens, Patra, {6000.0, . . . , 9999.9} Paul),

(Athens, Halkis, {0000.0, . . . , 3999.9} Paul)

yields R1 (Fig. 7), i.e. tuples with pairwise identical values merge into one. R1-S gives RAIL again. The ordinary *Projection* has been revised in [19]: For example, PROJECT [Employee](R1) would normally give a relation of four tuples. In this approach, however, this operation gives R2 (Fig. 7), where we observe a merging of tuples. Cartesian *Product* functions the ordinary way. The selection formula of a *retrieval* may not incorporate *valid* components. A final operation, HISTORICAL-DERIVATION, not described here, enables either the modification of a valid component or the assignment of a valid component to a value component.

2.6. TSQL2

TSQL2 [27], an extension to SQL2 [10], is the result of a joint effort. A relevant conceptual algebra has been formalised in [31]. The approach adopts Gadia's homogeneity property and

RAIL

Departure	Destination	Employee
Athens	Lamia	{0.0, 0.1, ..., 4999.9} Alex
Athens	Lamia	{5000.0, ..., 9999.0} John
Athens	Patra	{0.0, 0.1, ..., 4999.9} Paul

R1

Departure	Destination	Employee
Athens	Lamia	{0.0, 0.1, ..., 4999.9} Alex
Athens	Lamia	{5000.0, ..., 9999.0} John
Athens	Patra	{0.0, 0.1, ..., 4999.9, 6000.0, ..., 9999.9} Paul
Athens	Halkis	{0.0, 0.1, ..., 3999.0} Paul

R2

Employee
{0.0, 0.1, ..., 4999.9} Alex
{5000.0, ..., 9999.0} John
{0.0, 0.1, ..., 4999.9, 6000.0, ..., 9999.9} Paul

Fig. 7. Relations in McKenzie and Snodgrass's model.

temporal domains. To demonstrate the functionality of the algebra in areas other than VT databases, let the set of time points be replaced by the set A = {'AAA', 'BBB', . . . , 'ZZZ'}. Then relation SERVICE, in Fig. 1, can be represented as is shown in Fig. 8. A relation may have at most one attribute, where valid time is recorded.

Union and *Difference* apply as in Gadia's [9]. *Cartesian Product* incorporates intersection semantics, as in Sadeghi's [23]. A *Selection* may contain predicates which incorporate time.

SERVICE

Desk	Name	Range
1	John	[AAA, CCC] U [EEE, FFF]
2	Alex	[GGG, HHH] U [KKK, LLL]

Fig. 8. A relation in TSQL2.

A *projection* consists not only of an attribute list, it also includes a function which can modify the valid time of a tuple. Assume, for example, that *td1, td2* are plate domains and that *intersection(td1, td2)* returns their intersection. Then, an example of the extended *Projection* is PROJECT[Name, *intersect*(Range, [EEE, KKK])(SERVICE), and yields two tuples,

(John, [EEE, FFF])

(Alex, [GGG, HHH] ∪ [KKK, KKK]).

Operation SLICE splits a relation with temporal domains to another, where tuples are stamped with maximal time intervals. For example, SLICE(SERVICE) yields a relation with tuples

(1, John, [AAA, CCC])

(1, John, [EEE, FFF])

(2, Alex, [GGG, HHH])

(2, Alex, [KKK, LLL]).

This is the only operation by which the result relation may contain tuples whose values on all attributes, except Range, are identical. Another operation projects out valid time and transforms a relation to the respective snapshot. Some more operations have also been defined, including variations of a *Join*, which are not described here, for brevity reasons.

Alternatively, the value of all tuples for Range may be represented as a set of time points (*event* relation), as in [19]. For example, if SERVICE (Fig. 8) were an *event* relation, the value of its first tuple for attribute Range would be {AAA, BBB, CCC, EEE, FFF}. In such relations, the previously presented operations have been adapted, accordingly.

2.7. Other approaches

Some more approaches have also been proposed, not described here: Tuzhillin and Clifford [32] define an algebra of six operations and an equivalent calculus, independent of some particular VT model. Jensen, Soo and Snodgrass [14] define an algebra of seven algebraic operations, of a conceptual model. The algebra by Jang and Johnson [12] incorporates transaction time, consists of eight operations and has similarities with Tansel's [29]. A completely different approach is undertaken by Ariav [1], who defines an algebra and SQL extension: Time is incorporated at the relation level. The set of algebraic operations is incomplete (only three operations are defined) and, as Ariav admits, further work is needed in this approach. Again, for all these models, similar, non-VT database applications, of practical interest, can be identified.

2.8. Classification of models

In the table of Fig. 9, all proposed VT models have been classified with respect to two orthogonal parameters, the way valid time is represented (point, interval, etc.) and the level at which time is incorporated (tuple, attribute, relation). The numbers in brackets denote the

Level of Time Recording / Time Representation	Tuple Time-Stamping	Attribute Time-Stamping	Relation Time-Stamping
Time point	Snodgrass Sarda (7) Lorentzos-Johnson (7)	Clifford-Croker (8)	Ariav (3)
Time interval	Jones et al Ben-Zvi (6) Snodgrass Navathe-Ahmed (11) Sadeghi (6) Sarda (7) Lorentzos-Johnson (7)	Clifford-Croker (8) Tansel (9)	—
Set of time points	TSQL2	McKenzie-Snodgrass (6) Jang-Johnson (8)	—
Temporal Domain		Gadia (5) Gadia-Yeung (5)	—

Fig. 9. Classification of VT data models.

number of fundamental algebraic operations defined in each model. The approaches in [14, 32], have not been included, since they are argued to be independent of any particular time and relation representation.

We notice that the models in the second and third column, termed in the literature [20] as *tuple time-stamping* and *attribute time-stamping*, respectively, represent two distinguishing modelling approaches:

(i) Models whose basic characteristic is that the value of a tuple for an attribute is either atomic or time (second column). If we call *Snapshot First Normal Form Relational Model* (S-1NF) the one defined in [6] and consider it as a *reference* model, then these approaches can be seen as simple extensions to S-1NF.

(ii) Models whose basic characteristic is that both values (either atomic or non-atomic) and time are recorded in an attribute (third column). Such models clearly violate First Normal

Form. Hence, if we call *Snapshot Nested* (S-NESTED) the model defined in [11] and consider it as a *reference* model, too, all these models can be seen as extensions to S-NESTED. Note that TSQL2, though a tuple time-stamping approach, also belongs to this class, since it violates First Normal Form.

The above imply that the properties of two *reference* VT models can be identified, VT-1NF and VT-NESTED, which can be seen as extensions to S-1NF and S-NESTED, respectively: Their properties are identified in Sections 3 and 4, respectively.

3. Properties of VT-1NF

In this section we identify the properties of VT-1NF. We initially notice that a general property of every data model is that it must be characterised by a satisfactory degree of abstraction and simplicity [30]. In [30], the generic properties of a data model are also identified, i.e., the data types and structures which are supported, as well as its operations. Thus, in the following, we investigate, separately, the general properties of a model and the individual properties of a VT-1NF model, in terms of data types, data structures and operations. Since time is the key parameter in a VT model, the properties of time are also included. Each proposed property is justified by some explanation.

3.1. General properties

One fundamental property is adopted by all researchers, hence it is listed without comments:

G1. *VT-1NF has to be abstract, simple and its operations must be closed.*

A second reasonable property is:

G2. *In a VT-1NF relation, it has to be possible to record data valid in both the past, present and future.*

3.2. Properties for time

In principle, Time is isomorphic to the set of reals. The latter is a continuous, uncountably infinite, totally ordered set. However, in a database, only a proper, discrete subset of reals can be recorded. Continuity is, in practice, relaxed, by adopting an appropriate time precision. By analogy therefore,

T1. *It suffices to consider VT time as discrete and totally ordered.*

The next property is accepted by all authors:

T2. *Various granularities for VT time must be supported.*

As is shown in Fig. 9, four representations for time have been proposed. The two first, *time points* (also supported in S-1NF) and *time intervals* are widely used in every day life. The other two, sets of points and temporal domains are not such commonly used. In addition, sets of points imply, in principle, a nested model, whereas temporal domains are complicated structures, not complying with the simplicity of a simple model like VT-1NF. Thus, we conclude that

T3. *VT time should be supported exclusively either as time point or as time interval.*

A simple and easy way to support time intervals is to maintain two distinct attributes in a relation, for the recording of the interval boundaries. However, this solution has certain drawbacks:

(i) The user has to declare to the DBMS pairs of attributes which form intervals. If this is not done, the DBMS will not be in a position to automatically disallow the recording of *invalid* time intervals, like those whose starting point is greater than their end-point.
(ii) Certain simple queries are difficult to formulate in terms of the boundaries of the time intervals (see the discussion for property O2, in Section 3.5).
(iii) As is also shown below, in discussion for property S2 (Section 3.4), it is meaningful to consider relations with more than one time interval attribute. In cases like this, however, the user is likely to lose track of the correct pairing of these attributes, especially after a combination of projections (in which only one of the attributes which form a pair is rejected) and joins between relations (in which time point attributes, derived from distinct relations, have to be considered as forming a pair of time interval attributes).

To eliminate the above shortcomings, we assert a new property,

T4. *A VT time interval should be supported as a primitive data type.*

As was shown is Section 2, various VT data models support only one type of intervals. In practice, however, there are four distinct types, [i, j], [i, j), (i, j] and (i, j). Thus, for reasons of user-friendliness, we conclude that:

T5. *All four types of VT time intervals should directly be supported.*

This property, in conjunction with T2, implies that intervals over various time granularities should also be supported.

3.3. Data types

The satisfaction of the next property is obvious.

D1. *Every data type, valid in S-1NF, must also be valid in VT-1NF.*

3.4. Data structures

By the definition of VT-1NF, time and data must be recorded under distinct attributes. From this observation, in conjunction with property T4, it can directly be deduced that the relations in Fig. 10 should be valid structures in VT-1NF. We describe, briefly, the semantics associated with them. SALARY has the same semantics with the relation in Fig. 1. In INFLATION we record the rate at which the inflation of various countries was running during various intervals of time. For example, the first tuple shows that, during the first three months of the year, the inflation of country A was running at 3%. In the next two relations we record the president of an enterprise, at various times. The former uses time intervals and the latter time points.

We now notice that relation S (Fig. 10) is also valid in S-1NF. This gives an indication that, more generally, a valid S-1NF relation should also be valid in VT-1NF. Another justification, for the satisfaction of this property, is given in Section 3.5 below, in the discussion for *Project*. Thus,

S1. *Every S-1NF relation must also be valid in VT-1NF.*

From property T2, more than one time granularities should be supported in VT-1NF. In S-1NF, however, a relation may have more than one attribute of a time point type, therefore,

SALARY

Name	Amount	Time
John	10k	[10, 20)
John	10k	[30, 40)
Alex	14k	[5, 30)
Alex	18k	[31, 60)

INFLATION

Country	Time	Rate
A	[1, 3]	3
A	[1, 6]	4
A	[7, 12]	5
A	[1, 12]	5

R

Year	President
[1984, 1987]	john
[1988, 1990]	george
[1992, 1992]	alex

S

Year	President
1984	john
1985	john
1986	john
1987	john
1988	george
1989	george
1990	george
1992	alex

Fig. 10. Relations in VT-1NF.

from S1, the same also applies to a VT-1NF relation. Clearly, the same also should be possible if a relation has more than one time interval attribute or both time point and time interval attributes. This is also justified by the following: Firstly, as is shown later in property O1, it is possible for one such relation to be derived, as a result of a *Cartesian Product* operation. Secondly, Fig. 11 shows that such relations are useful and meaningful in VT-1NF: In particular, in SHIFT we record employee shifts during various data intervals. For example, the first tuple shows the time interval [1, 6), during which John will be working and, for each of the dates in this interval, it also shows the shift [8, 16), he has been assigned. From this, it can be deduced that:

S2. *Relations with more than one VT time attribute (of the same or of a different time granularity) of either a point or interval type, should be possible to be declared to the DBMS.*

3.5. Operations

The operations defined in a data model may form either a relational algebra or a relational calculus (tuple or domain). In principle, however, for each relational algebra an equivalent relational calculus is formalised and, next, a calculus-oriented query language is defined and implemented, usually based on the calculus. In terms of operations, therefore, and without loss of generality, our discussion here will restrict to algebraic operations.

From S1, every S-1NF relation, R, must also be valid in VT-1NF. As a valid S-1NF relation, R may have time point attributes. In addition, R is handled by the S-1NF operations *Union*, *Difference*, *Projection* and *Cartesian Product*. Hence, one first conclusion is that these operations should also be valid, when applied to VT-1NF relations which have time point attributes. We shall now demonstrate, by examples, that these operations must also remain exactly the same for VT-1NF relations with time interval attributes.

Union: Consider INFLATION (Fig. 10) and assume that it consisted of only the first three tuples. In order to insert the fourth tuple (a, [1, 12], 5), the S-1NF *Union* operation has to be issued, thus yielding a relation with these four tuples. This is exactly what we would like to obtain. Incidentally, we notice that it is valid for INFLATION to contain tuples whose Country and Rate components are identical, but their Time components form overlapping intervals (last two tuples).

SHIFT

Name	Day	Hour
john	[1, 6)	[8, 16)
john	[8, 13)	[8, 16)
alex	[1, 6)	[16, 24)
alex	[8, 13)	[16, 24)

Fig. 11. A VT-1NF relation with two VT time attributes.

Difference: By a similar argument, it can be deduced that the S-1NF *Difference* also has to remain exactly the same in VT-1NF.

Projection: Consider SALARY in Fig. 10 and the query *"list the names of all the employees who were ever paid"*. Then the S-1NF *projection* is necessary to be issued, in order to get a relation without time attributes. Thus, the result relation, which is valid in S-1NF, must also be valid in VT-1NF. It is also reasonable to issue a *projection* on attributes of a time type. Considering for example INFLATION (Fig. 10), a query could be *"list the distinct time intervals for which the inflation rate for country A has been recorded"*. The result of its execution should be exactly the time intervals which have been recorded for country A, that is [1, 3], [1, 6], [7, 12] and [1, 12].

Cartesian product: This operation is usually involved indirectly, in S-1NF, as part of a *Join*. The same also applies to VT-1NF. For example, SHIFT (Fig. 11) can be obtained by an appropriate join of two relations, DSHIFT(Name, Day) and HSHIFT(Name, Hour), in which the S-1NF *Cartesian Product* is involved indirectly.

The above observations lead to the property that:

O1. *The S-1NF operations Union, Difference, Projection and Cartesian Product must remain the same in VT-1NF.*

From this property, in conjunction with G2, it should be obvious that the insertion, deletion and update of pieces of data valid in the past, present or future must directly be supported.

By an argument identical with that of the previous operations, it is clear that the S-1NF *Selection* must also be valid for relations not having time interval attributes. In addition, however, we notice that in S-1NF six relational operators, "$<$", "$\leq$", "$=$", "$<>$", "$\geq$", "$>$" are used in comparisons between two *comparison-compatible* pieces of data. However, there are thirteen relative positions of a time interval $x = [x1, x2)$ with respect to another $a = [a1, a2)$. It is then easy to realise that the direct support of time interval predicates can greatly simplify the formulation of certain queries. For example, a predicate *cp* (*common points*) is defined in a way that "x *cp* a" evaluates to *true*, if the intervals x and a have *points in common*. In the absence of this predicate, the user has to type the equivalent expression, ($x1 \leq a2$ *and* $a1 \leq x2$), assuming that he is provided with capabilities to access the boundaries of intervals. However it is not easy for *every* user to formulate this query so easily: Our experimentation has shown that it is usually misformulated. Hence,

O2. *The Selection operation of S-1NF has to be generalized in VT-1NF, by the inclusion of interval predicates, so as to also be applied to both VT time points and VT time intervals.*

We now show that additional operations, variations of the above, have also to be defined in VT-1NF. We do so, by providing examples for which the respective operation can hardly, if at all, be formulated by using solely the S-1NF operations.

In Section 2, we saw that, in almost all tuple time-stamping models, the union of SALARY with another relation, requires the merging of the intervals of the tuples whose values on all other attributes are pairwise identical. We call the respective operation *Points-Union*(*P-Union*). It only has to be noted that *P-Union* has to be general enough, so as to apply to relations like SHIFT, with more than one time interval attribute. Similar observations can be made, in order to obtain the difference of two relations. The respective operation is termed *P-Difference* (*P-Difference*).

Consider relation SALARY (Fig. 10) and assume that a person is employed if and only if he is paid. Then the query "*retrieve the time interval(s) during which at least one person was employed*", should yield a tuple of a single time interval value, [5, 60). Various queries can be found which require such a *projection* functionality. We call this operation *P-Project*. Considering the above, the next property can be concluded:

O3. *Three more operations have to be defined in VT-1NF, P-Union, P-Difference and P-Project.*

In fig. 10 we demonstrated two distinct ways of representing VT data, concerning the presidents of an enterprise. However, people have a dual perception for time: Occasionally they interpret time as point and, in other cases, as interval. It is reasonable, therefore, for both *views* to be supported, simultaneously. This leads to another property:

O4. *Operations have to be defined, to enable transformations between the two complement views of VT time, time points and time intervals.*

3.6. Semantics, constraints and functions

From property S1, it is obvious that the semantics and, more generally, the constraints (functional dependencies, key of a relation etc.) captured by S-1NF, must also be captured by VT-1NF. Similarly, the functions defined in S-1NF must also be supported by VT-1NF. Since, however, VT-1NF also incorporates time intervals, all these characteristics must be enhanced in VT-1NF. Thus:

V1. *The semantics, constraints and functions of S-1NF must be enhanced in VT-1NF.*

This completes the properties of VT-1NF. Because of properties D1, S1, O1, O2 and V1, it can then be deduced that VT-1NF will be, in all respects, a consistent generalisation to S-1NF.

4. Properties of VT-NESTED

Since VT-1NF is a consistent generalisation to S-1NF, it is reasonable to assert a similar requirement, that VT-NESTED should also be a consistent generalisation to S-NESTED (Fig. 12). This implies that the reference model, S-NESTED, has first to be identified. In fact, at least two such candidate models have been identified [11, 22]. From these, we consider, as *reference*, the former, for the following reasons: It is more general, because it has specially

$$\begin{tikzcd}
\text{S-1NF} \arrow[r] \arrow[d] & \text{S-NESTED} \arrow[d] \\
\text{VT-1NF} \arrow[r] & \text{VT-NESTED}
\end{tikzcd}$$

Fig. 12. Generalisation of relational models.

been defined for nested relations and supports an arbitrary, application dependent, nesting. Its properties have been investigated [11] and an implementation of a prototype SQL extension has been reported [8]. In contrast, the latter has particularly been defined for the management of statistical data, and supports only a one-level nesting.

The properties of S-NESTED [11] can be summarised as follows: It supports all S-1NF data types and structures. In addition, it supports structures with relation valued attributes. All the S-1NF operations have been generalised and can be applied to all its valid structures. Finally, two more operations, *Nest* and *Unnest* have been defined [11], with functionality similar to PACK and UNPACK, described in Section 2.2: The former can transform S-1NF relations to nested. The latter can perform the inverse. Thus, S-NESTED is also, in all respects, a consistent extension to S-1NF. In order therefore to identify the properties of VT-NESTED, we notice the following.

Similarly to the fact, that VT-1NF is a generalisation to S-1NF, VT-NESTED should, by analogy, also be a generalisation of S-NESTED. Since S-NESTED is a generalisation to S-1NF, VT-NESTED should be a generalisation to VT-1NF, too (Fig. 12). This leads to the following properties for VT-NESTED:

G1 remains the same. However, it should be noted that VT-NESTED may not maintain the simplicity of VT-1NF, exactly as it is the case for S-NESTED, as compared with S-1NF. However, this is balanced by the fact that VT-NESTED can handle more complicated objects than both VT-1NF and VT-NESTED.

Properties G2, T1-T5, D1, and S2 are inherited from VT-1NF to VT-NESTED.

Property S1 is adapted in that every relation valid in either VT-1NF or S-NESTED (and subsequently every valid S-1NF) relation must also be valid in VT-NESTED. This implies that the relation in Fig. 13 is a valid structure in VT-NESTED. (Note that, in the more general case, VT data may be nested in a much lower level, in a VT-NESTED relation.)

From O1, operations *Union*, *Difference*, *Project* and *Cartesian Product* are the same both in S-1NF and VT-1NF. However, they have been extended from S-1NF to S-NESTED. By a similar argument, therefore, these operations are defined, in VT-NESTED, exactly the same way as in S-NESTED.

From O2, *Selection* has been extended from S-1NF to VT-1NF, by the incorporation of predicates which apply to time intervals. This operation then further extends from VT-1NF to VT-NESTED, in a way similar to that *Selection* was extended from S-1NF to S-NESTED.

By analogy to the fact that the five primitive operations of S-1NF have been generalised in

EMPLOYEE

Name	H-SALARY		H-ASSIGNMENT	
	Amount	PERIOD	Department	PERIOD
		Time		Time
john	10k	[10, 18) [20, 28) [20, 28)	hardware	[10, 18)
			shoe	[10, 18)
alex	11k	[20, 30)	shoe	[25, 35)
	18k	[40, 60)		

Fig. 13. A relation in VT-NESTED.

S-NESTED, the operations of VT-1NF whose definition is proposed in properties O3 and O4 have to be generalised in VT-NESTED.

V1 is adapted in that a VT-NESTED based language must support the semantics, constraints and functions of both a VT-1NF and an S-NESTED based language. In addition, new ones must also be defined in VT-NESTED.

Four more properties inherited from S-NESTED, are the following:

U1: Support of relation-valued attributes whose domain is the powerset of time points.

U2: Support of relation-valued attributes whose domain is the powerset of time intervals.

N1: Support of operation *Unnest.*

N2: Support of operation *Nest.*

It should be noted that, as a consequence of all the above properties, VT-NESTED transitively inherits all the properties of S-1NF, hence, it is the most general of all four models, S-1NF, S-NESTED, VT-1NF, VT-NESTED.

5. Evaluation of VT models

In this section we evaluate the tuple time-stamping approaches with respect to VT-1NF. To this end, particular effort was made, to be as precise as possible. Note that another evaluation, between attribute time-stamping approaches and VT-NESTED was avoided. The reason is that S-NESTED is not widely accepted, therefore there are substantial differences between this and the relevant VT extensions. To provide a measurement of the functionality of the reference models, we also report on the results of another theoretical evaluation, between the reference models and all VT models. Finally, we report on the evaluation of VT-1NF against actual user requirements.

The table in Fig. 14 is a summary of the properties satisfied by tuple time-stamping VT models. Answers are denoted by Y (Yes), N (no) and P (Partly), whereas "?" stands for *not specified in the literature*. Explanations are as follows:

G1: This property is too subjective, therefore we have assumed that it is satisfied by all models.

T2: The necessity for the satisfaction of this property is explicitly reported in most models, therefore, we have assumed that it is satisfied by all.

S1: From [21], it is implied to be satisfied by Navathe's and Ahmed's. It is also assumed to be satisfied by Jones's, Snodgrass's, Ben-Zvi's, Sarda's and Sadeghi's.

O1: It is partly satisfied by Snodgrass's, Navathe's and Sadeghi's because the Cartesian Product between two VT relations applies only under an interval intersection condition. Sarda's satisfies it, although it is argued that the result is *not* a VT relation [24].

O2: In Ben-Zvi's it is partly satisfied because the result relation of a selection is valid only for one time point. In some models (e.g. Sadeghi's) a special operation has been defined for retrievals over VT time. We have assumed that such models satisfy this property by making the

Model	G1	G2	T1	T2	T3	T4	T5	D1	S1	S2	O1	O2	O3	O4	V1
Jones	Y	Y	Y	Y	N	N	N	Y	Y	N	?	Y	?	N	P
Snodgrass	Y	Y	Y	Y	Y	N	N	Y	Y	N	P	Y	?	N	P
Ben-Zvi	Y	Y	Y	Y	N	N	N	Y	Y	N	?	P	?	N	P
Sarda	Y	Y	Y	Y	Y	N	N	Y	Y	N	Y	Y	Y	P	P
Navathe	Y	Y	Y	Y	N	N	N	Y	Y	N	P	Y	Y	P	P
Sadeghi	Y	?	Y	Y	N	N	N	Y	Y	N	P	Y	?	N	P
Lorentzos	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	P

Fig. 14. Evaluation of tuple time-stamping models with respect to VT-1NF.

assumption that the desired result can be obtained by the composition of two distinct retrieval operations.

O3: In principle, it has been assumed that *P-Union* and *P-Difference* is satisfied by all models. The "?" has been put because the satisfaction of *P-Project* is not clear in all models.

O4: Navathe's approach makes use of *window*, which splits time intervals into sub-intervals. It is assumed, therefore, that *window* can transform time intervals to *elementary* time intervals; the latter consist of only one time point, each, hence, up to an isomorphism, each time interval is equivalent with a time point. Sarda's satisfies O4 partly, too. The inverse operation is also assumed to be supported indirectly, in both models, by the incorporation of *P-Union*.

V1: It is assumed that this property is supported partly by all models, because, in our opinion, further work is needed on this topic.

We notice that, in spite of their seemingly major differences, tuple time-stamping models have many similarities: Properties G1, G2, T1, T2, D1, S1 and O2 are satisfied by almost all of them, whereas T4, T5 and S2 are satisfied by almost none. The remainder properties are satisfied by almost half of the models. The reason for this similarity is due to the fact that all these VT extensions have been based on Codd's model [6], which is widely accepted.

Another theoretical evaluation [17] concerned a comparison of VT-1NF with almost all VT models. The evaluation showed the following:

(i) Every piece of VT data which can be represented in some VT model can also be represented in VT-1NF.
(ii) For any result relation, derived by the application of the operations defined in any of these models, an equivalent result can be obtained, by applying the operations of VT-1NF to VT-1NF relations. One exception is the functionality of the *window* clause, defined in Navathe and Ahmed's approach [21].
(iii) Using the operations of VT-1NF, it is possible to reduce valid VT-1NF relations from any proposed VT model, and inversely. (Note that, to transform between VT-1NF and attribute time-stamping models, VT-1NF has to be enhanced by the operations *Nest* and *Unnest*.)
(iv) VT-1NF enables, in addition, the representation and manipulation of VT relations with more than one valid time attribute (property S2).

Since VT-NESTED is a generalisation of both S-NESTED and VT-1NF, it is obvious that the above results also apply to a comparison between VT-NESTED and all proposed VT models.

Another evaluation of VT-1NF, of practical interest, was also undertaken: Within the ORES project, an algebra, VT-AL, was defined, which satisfies all the properties of VT-1NF, except T5. VT-AL was measured against actual user requirements [17]. It was thus shown that it oversatisfied the requirements of the users of the particular application. Although it could be argued that VT requirements may vary from one application to another, to our knowledge no similar evaluation has been reported for some other approach. The same conclusions were also drawn for VT-SQL [18], an SQL extension, based on VT-AL. Both VT-AL and VT-SQL have been implemented.

6. Interval models

In this section we show that *time* is characterised by a high degree of abstraction. We thus elaborate on the necessity to define two more radical extensions to S-1NF and S-NESTED, denoted by I-1NF and I-NESTED, respectively, and specify their properties.

Firstly, we notice that whether some value should be treated as time or not, is only a matter of interpretation. This is demonstrated by the following examples:

Example 1. Consider the value *12-05-95.* We then notice that this can be interpreted in many distinct ways: (i) May 12, 1995 (European interpretation of date). (ii) December 5, 1995 (American interpretation of date). (iii) A value of a special data type, interpreted as the 12th article of the 5th law, voted in 1995. (iv) A value of another special type, interpreted as an individual car plate in some particular country. (v) An arbitrary value in CHAR(8).

Example 2. In the previous example we showed that a value, optically reminding of a date may/may not represent time. We now show that, inversely, an integer may represent time. Consider, that, for the purposes of a scientific experiment, it is necessary to record how the values of certain parameters change with respect to the 1st, 2nd, . . . , nth day. Then the set of integers has to be treated exactly the same way as valid time. (Note incidentally, that, for simplicity reasons, in almost all VT data modelling approaches, time is represented by the set of integers.)

Example 3. The values of other conventional data types may also be interpreted as valid time: For example, the elements of $\{A, \ldots, Z\}$ may denote the successive phases for the completion of a project. It may then be necessary to schedule various activities of the project and record them in a relation. For example one tuple of SCHEDULE(Activity, Phase) could be $(1, [A, C]))$, indicating that activity 1 will last from phase A up to phase C. (Incidentally, we notice that it may not be necessary for all phases to have the same time duration.)

Secondly, it was shown, by specific examples, that every VT model can be applied to many other areas of practical interest. To achieve this, we only had to identify another set with properties identical with those of time. In fact, many such sets can be identified. We can thus conclude that, although all VT models have specially been defined for VT data management, in fact, both their representational capabilities and operations are *independent* of the necessity for VT data handling.

From the above, it can be concluded that, rather than defining VT reference models, which are extensions to S-1NF or S-NESTED, one should rather define two more general extensions, (i) and (ii), described below:

(i) We notice that S-1NF does not support time intervals, whereas VT-1NF does. Also, the representation and operational capabilities of VT-1NF are actually independent of time.

Hence, a more general reference model, I-1NF, can be defined, in place of VT-1NF: It is a simple extension to S-1NF, enhanced by appropriate capabilities for *interval* data management. Its properties are those specified for VT-1NF, except that the concept of a *time interval* is now replaced by the more general concept, *interval*.

(ii) By a similar argument, I-NESTED can also be specified, with properties those of VT-NESTED. This is the most general, in that it can handle every piece of data handled by either S-1NF, S-NESTED or I-1NF.

It is then obvious that the management of VT data is only one special application of the above interval-extended models.

7. Conclusions

Starting from the necessity to support valid time data, we identified the properties of two reference relational models, VT-1NF, VT-NESTED, which are consistent extensions to the snapshot models proposed in [6] and [11]. We also showed that more radical extensions to [6] and [11] should rather be attempted, I-1NF and I-NESTED, which support data of an *interval* type.

In [20] an evaluation of various VT models was undertaken, different than that, in the present work. It was also shown [20] that certain desirable properties of a VT model are *conflicting*, i.e. there is no VT model which can satisfy them all. However, I-1NF and I-NESTED support valid time data and, at the same time, they satisfy all the properties in [20].

Further work concerns the formal definition of IN-SQL, an *Interval Nested* extension to SQL.

Acknowledgement

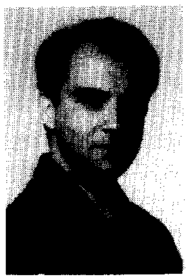
This work has partly been supported by the European Commission from within the ORES project, contract number P7224.

References

[1] G. Ariav, A temporally oriented data model, *ACM Trans. Database Systems* 11(4) (1986) 499–527.
[2] J. Ben-Zvi, The time relational model, Ph.D. Dissertation, Department of Computer Science, University of California, Los Angeles, 1982.
[3] J. Clifford and A.U. Tansel, On an algebra for historical relational databases: Two views, *Proc. ACM SIGMOD*, Austin, TX (1985) 247–265.
[4] J. Clifford and G. Ariav, Temporal data management: Models and systems, in: G. Ariav and J. Clifford, eds., *New Directions for Database Systems* (Ablex Publishing, Norwood, NJ, 1986) 168–185.
[5] J. Clifford and A. Croker, The Historical Relational Data Model (HRDM) revisited, in: A. Tansel, J. Clifford, S. Gadia, A. Segev and R. Snodgrass, eds., *Temporal Databases: Theory, Design and Implementation* (Benjamin/Cummings, 1993) 6–27.

[6] E.F. Codd, Relational completeness of data base sublanguages, in: R. Rustin, ed., *Data Base Systems* 6, Courant Computer Symposia Series (Prentice-Hall, Englewood Cliffs, 1972) 65–98.
[7] E.F. Codd, Extending the database relational model to capture more meaning, *ACM Trans. Database Systems* 4(4) (1979) 397–434.
[8] P. Dadam, K. Kuespert, F. Anderson, H. Blanken, R. Erbe, J. Guenauer, V. Lum, P. Pistor and G. Walch, A DBMS prototype to support extended NF2 relations: An integrated view on flat tables and hierarchies, *Proc. ACM SIGMOD*, Washington (1986) 356–367.
[9] S.K. Gadia, A homogeneous relational model and query languages for temporal databases, *ACM Trans. Database Systems* 13(4) (1988) 418–448.
[10] The International Organization for Standardization, *Database Language SQL* (ISO 9075:1992(E) 1992).
[11] G. Jaeschke and H.J. Schek, Remarks on the algebra of non first normal form relations, *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems* (1982) 124–138.
[12] Y.P. Jang and R.G. Johnson, Nested relation based temporal data representation, *Proc. 4th Int. Hong Kong Computer Society Database Workshop*, Hong Kong (1992) 94–111.
[13] C.S. Jensen, J. Clifford, S.K. Gadia, A. Segev and R.T. Snodgrass, A glossary of temporal database concepts, *SIGMOD Record* 21(3) (1992) 35–43.
[14] C.S. Jensen, M.D. Soo and R.T. Snodgrass, Unification of temporal data models, Tech. Rep. TR92-15, Department of Computer Science, University of Arizona, 1992.
[15] S. Jones and P.S. Mason, Handling the time dimension in a database, in: S.M. Deen and P. Hammersley, eds., *Proc. Int. Conf. on Data Bases* (British Computer Society, 1980) 65–83.
[16] N.A. Lorentzos and R.G. Johnson, TRA: A model for a temporal relational algebra, in: C. Rolland, F. Bodart and M. Leonard, eds., *Temporal Aspects in Information Systems* (North-Holland 1988) 203–215.
[17] N.A. Lorentzos and Y.G. Mitsopoulos, Specification of Valid Time Formalism, ORES Deliverable C3, Agricultural University of Athens, 1993.
[18] N.A. Lorentzos and Y.G. Mitsopoulos, Specification of valid time SQL, ORES Deliverable D2, Agricultural University of Athens, 1993.
[19] E. McKenzie and R. Snodgrass, Supporting valid time: An historical algebra, Tech. Rep. TR87-008, Department of Computer Science, University of North Carolina, Chapel Hill, 1987.
[20] L.E. McKenzie and R.T. Snodgrass, Evaluation of Relational Algebras Incorporating the Time Dimension in Databases, *ACM Computing Surveys* 23(4) (1991) 501–543.
[21] S.B. Navathe and R. Ahmed, Temporal extensions to the Relational Model and SQL, in: A. Tansel, J. Clifford, S. Gadia, A. Segev and R. Snodgrass, eds., *Temporal Databases: Theory, Design and Implementation* (Benjamin/Cummings, 1993) 92–109.
[22] G. Ozsoyoglou, Z.M. Ozsoyoglou and V. Matos, Extending relational algebra and relational calculus with set-valued attributes and aggregate functions, *ACM Trans. Database Systems* 12(4) (1987) 566–592.
[23] R. Sadeghi, A database query language for operations on historical data, Ph.D. Dissertation, Dundee College of Technology, 1987.
[24] N.L. Sarda, Algebra and query language for a historical data model, *Computer J.* 33(1) (1990) 11–18.
[25] N.L. Sarda, Extensions to SQL for historical databases, *IEEE Trans. Knowledge and Data Engineering* 2(2) (1990) 220–230.
[26] S. Snodgrass, The temporal query language TQUEL, *ACM Trans. Database Systems* 12(2) (1987) 247–298.
[27] R.T. Snodgrass, I. Ahn, G. Ariav, D. Batory, J. Clifford, C.E. Dyreson, R. Elmasri, F. Grandi, C.S. Jensen, W. Kafer, N. Kline, K. Kulkarni, T.Y.C. Leung, N. Lorentzos, J.F. Roddick, A. Segev, M.D. Soo and S.M. Spirada, TSQL2 language specification, *SIGMOD Record* 23(1) (1994) 65–86.
[28] A.U. Tansel, Adding time dimension to relational model and extending relational algebra, *Information Systems* 11(4) (1986) 343–355.
[29] A.U. Tansel and L. Garnett, Nested Historical Relations, *Proc. ACM SIGMOD* (1989) 284–293.
[30] D.C. Tsichritzis and F.H. Lochovsky, *Data Models* (Prentice Hall, NJ, 1982).
[31] TSQL2 Language Committee, An algebra for TSQL2, 1994.
[32] A. Tuzhilin and J. Clifford, A temporal relational algebra as a basis for temporal relational completeness, *Proc. 16th Very Large Data Bases Conf.*, Brisbane, Australia (1990) 13–23.
[33] J.D. Ullman, *Principles of Database Systems*, 2n ed. (Computer Press, 1982).

Nikos A. Lorentzos got his first degree in Mathematics from the University of Athens (1975), his Master's degree (Computer Science) from Queens College, City University of New York (1981), and his PhD from Birkbeck College, London University (1988). In 1981 he was hired as a Systems Analyst at the Greek Ministry of Mercantile Marine and in 1988 he became the Head of the Department of Informatics. In 1989 he moved to the Agricultural University of Athens, where he is currently an Assistant Professor, in the Science Department. He has worked as a research assistant at the Research Foundation of City University of New York, and as a temporary lecturer at Birkbeck College, UK. He has been involved in a number of projects, some of them funded by the European Union, such as ORES(ESPRIT III), MACQU (AIR II), BARBARA (RACE II), DISNET (IMPACT), and TEMPUS. As the Technical Manager of ORES, he contributed to its successful completion, by the development of many temporal data management software tools, including an algebra and an SQL extension. As a member of the TSQL2 Committee, he contributed to the specification of TSQL2, aiming at the management of bitemporal data. He has published in a number of journals, including Information Systems, IEEE/KDE and Computer Journal. He has also contributed to the writing of books. He has acted as a reviewer for many recognised journals, and he was on the program committee of the International Workshop on Temporal Databases, in Zurich, 1995. He is a member of ACM, BCS, Greek Computer Society and Greek Mathematics Society.



Yannis Manolopoulos got his first degree (5 years studies) in Electrical Engineering from the Aristotle University, Thessaloniki, Greece in 1981, and his PhD degree in Computer Engineering from the same institution in 1986. He was with the Department of Electrical Engineering during the period 1982–1993. Since 1993 he is an Assistant Professor in the Department of Informatics of the Aristotle University. He spent two sabbatical years with the Computer Science Department of the University of Toronto and the Computer Science Department of the University of Maryland, at College Park. He has published in ACM TODS, IEEE TSE, Information Systems, Information Sciences, IPL, BIT, Nordic Journal of Computing, Image and Vision Computing, Information and Software Technology, etc. He received a best paper award from ACM SIGMOD 94 Conference. His research interests include temporal and spatial databases, data/file structures and algorithms. He is a member of IEEE Computer Society, ACM, Greek Computer Society and Technical Chamber of Greece.