ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

Rainer Manthey and Viacheslav Wolfengagen (Eds)

Advances in Databases and Information Systems 1997

Proceedings of the First East-European Symposium on Advances in Databases and Information Systems, (ADBIS'97), St Petersburg, 2-5 September 1997

Temporal Data Model

Manuk G. Manukyan

Published in collaboration with the
British Computer Society

©Copyright in this paper belongs to the author(s)

ISBN: 3-540-76227-2

Time-Oriented Data Model

Manuk G. Manukyan

Department of Computer Science, Yerevan State University
Yerevan, Republic of Armenia

Abstract

A minimal extension of relational data model for logical modeling of the category of time in databases is offered. The time is considered as a totally ordered set. In the frame of suggested data model the homogeneous model of representation of time in relations is adopted. Allowing the homogeneity, the modeling of the temporal aspect of databases is achieved by means of two temporal attributes. The temporal attribute is considered as abstract data type. A formal definition of temporal relation which is based on abstraction of generalization is given. The semantics of operators **update** and **delete** is changed. A temporal algebra and calculus are developed. Also new aggregate functions are suggested considering the time aspect of the object behavior in the given enterprise. In order to give temporal integrity constraints closed formulas are used. The problems of partial support of evolution of the database schemes are considered.

1 Introduction

A great deal of research both in the area of DM and, in particular, in relational models has been done after the publications of early works of E. F. Codd. Results of this research are conceptual DM in which advanced declarative and/or procedural apparatus for data modeling is maintained. But the problems of effective organization and manipulation of data do *not* exhaust all the specter of problems in the theory of DB. Problems of logical modeling of category of time in databases (CTDB) accept a special importance. The roots of such research are in that within the frame of traditional DM it is possible to present *one* state of the modeling object only. This brings to the loss of information which is necessary to take certain decision. Thus the aim of CTDB is to present information on evolution of real world objects in DB. It should be mentioned that problems of modeling of time are *not* a prerogative of information processing systems only. In scientific publications known are works in the field of artificial intelligence, logic, linguistics, etc.

The explicit presentation of the temporal context and its representation in constructions of information systems assumes development of special language tools because the temporal dimension, temporal attributes of relations of DB have a special semantics. In particular, the creation of temporal DM assumes the necessity of preservation in DB of those values of properties and relations of objects which are no longer true. Besides, the traditional interpretation of operations of DB updating changes: each actualization of DB is considered as a insertion of new data in it (the information of its previous state preserved).

In general, DM is based upon a conception of consistency of DB. This means that it is impossible to present in DB a fact and its contradiction simultaneously. This conception (with CTDB) should be extended by possibility to represent the state of DB in different discrete moments of time (DMT). The traditional invariants of DB take place in each of these states. Additional means are necessary for presentation of DB invariants, that take into account the category of time. An effective realization of CTDB requires to store information on objects in those DMT only, that correspond to an certain event of the life of the object. The missing explicit information on the object in the interval between events should be inferable.

Nowadays, CTDB supporting research in DM develops intensively. Such research is necessary for further development of conceptual DM for solution practical problems. On the other hand the appearance of cheaper environments of storage (optical disks, for instance) makes it possible to realize the temporal DB practically.

Offered DM differ, first of all, in homogeneity of represented temporal information regarding properties of the object. According to the homogeneous model, temporal periods, during which the values of properties of the object maintain true, are the same for all values of attributes of a tuple of relations. Such a model is offered in [7] and is assumed in many papers based on relational DM. The opposite of that model is heterogeneous model, in which every property of the object in tuple has a proper temporal attribute, as in [14]. There are also versions of representation in which homogeneity is required only for subsets of properties in tuple of relation. With the assumption of homogeneity, modeling of temporal aspect in DB is reached by means of two temporal attributes. The first attribute defines the temporal interval of existence, when, according to the user, the object of enterprise (or fact) corresponding to tuple, exists (takes place). The values of attribute of existence are given by the user during insertion of tuple in DB or during its change. The second attribute - attribute of belief - has temporal periods as values, when the system "believes" in existence of the object (in truth of the fact). The system sets values of attribute of belief in a manner that the beginning moment of temporal interval coincides with the moment of transaction (the moment on insertion of information into DB); this provides insertion of a new, or creation of a changed tuple. The last moment coincides with the moment of transaction, that provides deletion of tuple in DB (or its modification). When a new tuple is formed, the value of the last moment of time is equal to an infinite moment of time in the future. The values of properties of the object, which appear in some DMT, are also assumed to have the property of persistence. This means that their interval of existence also goes maximally far to the right by time axis. In general, binary temporal predicates are used in order to form timeslices of relations in given temporal intervals. Such predicates are considered, particularly, in [1]. In papers [11, 13, 22] the semantics of operations of relational algebra is changed for the representation of semantics of time. It should be mentioned, that in [11] are considered infinite temporal date.

The interpretation of aggregate functions strongly depends on temporal aspect. In general, the functions *max* and *min* become multivalued. I. e. in different periods of time the maximal and minimal values of the attribute - argument of function can be different. As a value of such functions may be the distribution of maximal (minimal) values of argument of the function in time. Therefore one should have two different versions of such functions. One - for the definition of the mentioned distribution; and the other for the definition of the only greatest (smallest) value of argument of function in all times. Analogously, the functions *total*, *average* and *count* produce not only single values, but also distribution of values in time. Apart from listed aggregate functions in [13, 21] some new aggregate functions are considered.

During the life of DB their schemes can change. DM with temporal orientation ought to maintain evolution of the scheme of DB. This assumes the preservation of those schemes of DB which have been changed in time (because the queries of temporal DB can be oriented both to current scheme of DB and to those schemes which have taken place in past). Particularly, in [16] studied are the questions of maintenance of temporal DB with changing schemes.

A characteristic property of advanced DM with time orientation is the existence of procedural tools for CTDB modeling on conceptual level. We can use the procedural tools in cases when it is necessary to establish such temporal dependences (on conceptual level), that cannot be defined by declarative means.

In consideration of temporal DB one of the central problems is integrity constraints. In the frame of relational DM closed formulas of predicate calculus are universal means of support of DB integrity. In [6] in order to give DB temporal integrity constraints offered are binary relations. The sequence of events in DB is defined by means of this relations. Such binary relations are convenient in cases when the temporal attribute in DB is supported *not* explicitly. In [4] closed formulas of past temporal logic language are used for definition of temporal integrity constraints.

Nowadays, several languages of description of temporal DB and corresponding query languages are offered [2, 7, 8, 10, 13, 14, 20, 21, 22]. The detail analyses of language means for CTDB support can be found in [3, 17, 19].

In this paper an approach to the support of temporal DB in the frame of unified conceptual date model (UCDM) of system of integration of heterogeneous DB [12] is offered. In the future the BNF will be used as a metalanguage for syntax constructions .

2 The Language Tools of Logical Modeling of CTDB

2.1 The Representation of Data with Temporal Orientation

In the frame of UCDM it is adopted the homogeneous model of representation of time in relations. According to the homogeneous model, the enterprise ought to be presented by set of relations with schemes that the values of all attributes of each tuple of relations have the same period of existence. I. e. the properties of the object of enterprise which corresponds to the tuple, become true or false simultaneously. With the assumption of homogeneity, modeling of temporal aspect in DB is reached by means of two temporal attributes. The first attribute defines the temporal interval of existence when, according the user, the object of enterprise (or fact) corresponding to tuple, exists (takes place). The second attribute - attribute of belief - has temporal periods as values, when the system "believes" in existence of the object (in truth of the fact). In UCDM the names of this attributes are reserved and are the same in all schemes of relations. The name of the attribute of existence is **valid** and the name of the attribute of belief - **belief**. The values of attribute of **valid** are given by the user on insertion of tuple in DB or on its change. The system sets values of attribute of **belief** in a manner that the beginning moment of temporal interval coincides with the moment of transaction (the moment on insertion of information into DB); this provides insertion of a new, or creation of a changed tuple. The last moment coincides with the moment of transaction, that provides deletion of tuple in DB (or its modification). When a new tuple is formed, the value of the last moment of time **belief** is equal to **inf** (an infinite moment of time in the future). The values of properties of the object, which appear in some DMT, are also assumed to have the property of persistence. This means that their interval of existence also goes maximally far to the right by time axis. The user gives this fact by setting the corresponding moment in the future or by means of the constant **inf**.

Definition 1 Let us T is a totally ordered set of DMT. Let $t_1 \in T$ and $t_2 \in T$ then the object

$$[t_1, t_2] = \{t \mid (t \in T) \& (t_1 < t < t_2)\} \cup \{t_1\} \cup \{t_2\}$$

is the **time period (TP)**.

Definition 2 Let us $T_R = \langle A, K \rangle$ is a scheme of the temporal relation t_r , where $A = A_1, A_2, \dots, A_n$, **valid**, **belief** is the set of attributes of temporal relation and $K = \{A_{i_1}, A_{i_2}, \dots, A_{i_m}, \text{belief}\}$ is the primary key, $m \leq n$. Then $R = \langle A, K_1 \rangle$, $S_R = \langle A, K \rangle$, $G_R = \langle A, K \rangle$ are schemes of relations r , s_r , g_r interpreting t_r . Here $K_1 = K \setminus \{\text{belief}\}$, and **valid**, **belief** are temporal attributes interpreted as ADT, representing TP [15].

In this definition r , s_r and g_r have the following meaning:

- r - represents the current state of DB (i. e. the set of tuples in which the value of the final moment of time of the attribute of **belief** is equal to **inf**);
- s_r is to keep the values of those tuples r which have been changed;
- g_r is the abstraction of generalization of relations r and s_r [18].

2.2 Time as ADT

The temporal attribute is represented by means of ADT. ADT **datetime** (DTP) and ADT **interval** are the basic constructions for the time modeling. These provide the representation corresponding to identical values of the language SQL2 [9].

An item of ADT **datetime** is made up of a contiguous subset of the fields :

yy : year
 mo : month within year
 dd : day within month
 hh : hour within day

mi : minute within hour
 ss : second within minute
 ff(n) : fraction within second

An item of ADT **interval** is made up of a contiguous subset of the fields :

yy : year
 mo : months

or

dd : days
 hh : hours
 mi : minute
 ss : seconds
 ff(n) : fraction of a second

2.3 Operations over ADT Datetime, Time Period and Interval

ADT **datetime**, **time period** and **interval** are parametrized generic types. The temporal qualifier is accepted as a parameter of that type. ADT is represented by a set of polymorph functions. The considered set of polymorph functions can be divided into groups in each of which included are functionally homogeneous functions:

- arithmetical functions;
- predicates;
- transformation functions;
- input - output functions.

Bellow the full list of polymorph functions is presented:

<identifier of arithmetical function>::=*and* | *minus* | *mult* | *div*
 <identifier of predicate>::=*before* | *equals* | *after* | *starts* | *during* | *finishes* | *meets* | *overlaps* | *before_starts*
 | *before_ends* | *le* | *lt* | *ne* | *ge* | *gt* | *eq*
 <identifier of transformations function>::=*year* | *month* | *day* | *day_of_week* | *hour* | *minute* | *second*
 | *fraction* | *length* | *init* | *fin* | *intersect* | *union* | *current* | *extend* | *i_convert* | *si_convert* | *sp_convert* |
n_interval
 <identifier of input-output function>::=*input* | *output*

The function *input* is designed for the transformation of temporal literal to internal representation. *Output* does the inverse transformation.

The function *current* provides the access to the clock of computer. The value of function *current* is the current DTP to an approximation of $[yy, ff(3)]$. If the argument of the function *current* is the temporal qualifier then the current DTP is the value of that function. The set of fields of DTP gives the temporal qualifier - the function argument.

If the argument of the function *extend* is the DTP only, then that very DTP is the value of the function to an approximation of $[yy, ff(3)]$. And if the argument of the function *extend* is the DTP *as well as* temporal qualifier, then its value is the same DTP to an approximation of the argument of the function defined by means of temporal qualifier.

Let us T be DTP set with ATD **datetime** structure, I be a set of intervals with ATD **interval** structure, S be a set of time periods with ATD **time period** structure, $N = \{0, 1, 2, \dots\}$, and E be the set of integers.

Let f is one of the functions $year, month, day, hour, minute, second, fraction$. Then f defines the following mappings:

$$\begin{aligned} f &: T \rightarrow N, \text{ where } f \text{ is a total function,} \\ f &: I \rightarrow E, \text{ where } f \text{ is a partial function.} \end{aligned}$$

The function day_of_week is a total function and defines the following mapping:

$$day_of_week : T \rightarrow dow,$$

where $dow = \{'Mon', 'Tue', ..., 'Sun'\}$.

The function $length$ is a total function and defines the length of the TP:

$$length : S \rightarrow I.$$

The function $i_convert$ is a total function and converts the argument of the function to an integer (in units of the finite field of the argument's temporal qualifier):

$$i_convert : I \rightarrow E.$$

Let $Q = \{[yy, yy], [mo, mo], [dd, dd], [hh, hh], [mi, mi], [ss, ss], [ff, ff]\}$ - be the set of temporal qualifiers. Functions $sp_convert$ and $si_convert$ define the following mappings:

$$\begin{aligned} sp_convert &: N \times Q \rightarrow T \\ si_convert &: E \times Q \rightarrow I. \end{aligned}$$

The value of the function $n_interval$ is the value of the argument of the function (a value of type ADT **interval**) with a minus sign.

The functions $union, intersect, init, fin$ are total functions and define the following mappings:

$$\begin{aligned} intersect &: S \times S \rightarrow S \\ union &: S \times S \rightarrow S \\ union &: T \times T \rightarrow S \\ init &: S \rightarrow T \\ fin &: S \rightarrow T \end{aligned}$$

Arithmetical functions are obvious: add is the temporal addition, $minus$ is the temporal subtraction, $mult$ is the temporal multiplication, div is the temporal division:

$$\begin{aligned} add &: T \times I \rightarrow T \\ add &: I \times I \rightarrow I \\ minus &: T \times I \rightarrow T \\ minus &: T \times T \rightarrow I \\ mult &: I \times N \rightarrow I \\ div &: I \times N \rightarrow I. \end{aligned}$$

Predicates :

Let us f is one of functions ge, gt, le, lt, eq, ne . Then f is a partial function and gives the following mapping:

$$f : I \times I \rightarrow \{true, false, \perp\}.$$

Temporal predicates :

Assume x and y define DTP or TP. Bellow listed are the temporal predicates that enable to determine the relationship between DTP and TP. The predicates are *true* with following conditions:

- $before(x, y)$: DTP x is before DTP y or DTP x is before TP y or these two TP do not overlap and x is before y ;

- *equals*(x, y): x and y are the same DTP or the same TP;
 - *after*(x, y): DTP x is after DTP y or DTP x comes later than TP y or these two TP do not overlap and x is comes later than y ;
 - *during*(x, y): TP y contains DTP x or TP y contains TP x (*contains* means that the beginning and end of TP x , TP y do not coincide);
 - *starts*(x, y): DTP x coincides with the beginning of TP y or TP x , TP y start simultaneously and the end of x is in y ;
 - *finishes*(x, y): the end of TP y coincides with DTP x or the ends of TP x , TP y coincide and the beginning of x is in y ;
 - *meets*(x, y): TP x starts before TP y and there is no DTP between these two TP and no DTP is shared by them;
 - *overlaps*(x, y): TP x starts before the end of TP y and TP y starts before the end of TP x ;
 - *before_starts*(x, y): TP x comes before TP y (TP x and TP y can overlaps);
 - *before_ends*(x, y): the end of TP x is before the end of TP y (TP x and TP y can overlaps).
- Say x and y are arguments in temporal predicates to an approximation of $[a, b]$ and $[c, d]$ respectively; then:
- if $a < c$ then x is replaced by *extend*($x, [c, b]$);
 - if $a > c$ then y is replaced by *extend*($y, [a, d]$);
 - if $b < d$ then x is replaced by *extend*($x, [a, d]$);
 - if $b > d$ then y is replaced by *extend*($y, [c, b]$).

2.4 Temporal Algebra

The semantics of operations of relational algebra has been changed. Let us T_1 and T_2 are the temporal attributes of operands of the operation, and T is temporal attribute of the resulting relation. The following functions (for the computation of temporal attributes of tuples of resulting relations) are offered:

1. *For the operation of natural join:*

put $T = \text{intersect}(T_1, T_2)$ for each pair of joined tuples; if $T = \emptyset$ then the joined tuple is not from resulting relation.

2. *For the operation of intersection of relations:* we put $T = \text{intersect}(T_1, T_2)$ for each pair of equal tuples to an approximation of temporal attributes. This differs from the traditional semantics in that the resulting relations contain those tuples only, for which $T \neq \emptyset$.

3. *For the operation of Cartesian product:* we define $T = \text{intersect}(T_1, T_2)$ for each pair of tuples. The resulting relations contain those pairs of tuples, for which $T \neq \emptyset$.

4. *For the operation of difference of relations:* for each pair of equal (without temporal attributes) tuples defined is : $T = \text{intersect}(T_1, T_2)$. If $T = \emptyset$, the first tuple gets into resulting relations. In contrary two ends are possible. If the predicate *during*(t, T_1) is *true*, then two equal tuples (which only differ in values of temporal attributes) get into the resulting relation. In the first tuple the value of the temporal attribute is equal to TP for which the beginning moment is equal to *init*(T_1) and the end is equal to such a DTP t_1 , that the predicate *meets*($[\text{init}(T_1), t_1], t$) is *true*. In the second tuple the final moment TP is equal to *fin*(T_1) and the beginning moment TP is equal to such a DTP t_1 , that the predicate *meets*($t, [t_1, \text{fin}(T_1)]$) is *true*. If the *during*(t, T_1) is *false*, then one tuple only gets into resulting relation. The value of the temporal attribute of that tuple is TP, the beginning moment is equal to *init*(T_1) and the end is equal to such a DTP t_1 , that the predicate *meets*($[\text{init}(T_1), t_1], t$) is *true* or the last moment is equal to *fin*(T_1) and the beginning moment is equal to such a DTP t_1 , that the predicate is *meets*($t, [t_1, \text{fin}(T_1)]$). Note, that the resulting relations contain those tuples of the first operand, that do not coincide with tuples of the second operand (to an approximation of temporal attributes).

5. *For the operations of selection, projection and union:* the offered semantics differs from the traditional semantics in that for each pair of equal tuples (to an approximation of temporal attributes) the following condition is checked: *meets*(T_1, T_2) $\vee$ *overlaps*(T_1, T_2) $\vee$ *meets*(T_2, T_1), where T_1 and T_2 are the temporal attributes of this tuples. If the condition is *true*, then these tuples are replaced by an only tuple with *union*(T_1, T_2) as value of the temporal attribute.

6. *For the operations of division:* Assume R and S are the schemes of temporal relations r and s respectively. Let $L = R \setminus S$ and $t = \bigcap_i T_i$ where T_i is the temporal attribute of i -th tuple of s . Let us divide r into groups according to the values of L . Assume $p_i = \bigcap_j T_j$ where T_j is the temporal attribute of j -th tuple of i -th group of the temporal relation r . Say $t \neq \emptyset$. We build a relation r' from r in the following manner: r' contains those tuples of i -th group of the r for which $p_i \neq \emptyset$ and $t \cap p_i \neq \emptyset$. Simultaneously, we replace the value of the temporal attribute in r' by the following: $t \cap p_i$ (we do it for each tuple). The following semantics is suggested for the operation $r \div s$ (the semantics is defined by means of operations of classical algebra):

$$\Pi_{L,valid}(r') \setminus \Pi_{L,valid}((\Pi_{L,valid}(r') \times \Pi_Q(s)) \setminus r'),$$

where Q is the set of attributes of the temporal relation S without temporal attributes.

2.5 Query Language

The query language of UCDM (the language of *filters*) is based upon domain calculus. Giving a query, it is important to have possibility to give temporal restrictions which define the timeslice DB (the latter should be considered in the given query). As such a timeslice one can take the current state of DB. Such timeslices can be given by temporal predicates according to the attributes **valid** and **belief** in every formula of the *filter*: in each formula it is necessary to use the condition defining the timeslice we need. The situation when the timeslice should be the same for all formulas of the *filter* is essential. In these circumstances it is not convenient to repeat the same predicate in all formulas.

2.5.1 Temporal Calculus

There are following types of atoms:

- a1. $r(x_1, x_2, \dots, x_n)$, where r is a k -ary relation and any x_i is a constant or variable in domain.
- a2. $x\Theta y$, where x and y are constants or variables in domains and $\Theta \in \{=, \neq, \geq, \leq, <, >\}$.
- a3. Predicates, temporal predicates are atoms as well.

Formulas are defined recursively as follows:

- f1. Any atom is a formula.
- f2. If f - formula, then $\neg f$ is a formula.
- f3. If f and g - formulas, then $f \& g$ and $f \vee g$ are formulas too.
- f4. If f is a formula, $(\exists x)(f)$ is a formula.
- f5. If f is a formula, $(\forall x)(f)$ is a formula.
- f6. If f is a formula, then f **on** <selector> **as of** <selector>, f **on** <selector>, f **as of** <selector> are, as well formulas, where:

< selector > ::= < variable > | < temporal constant > | < temporal function >

< temporal function > ::= *year* | *month* | *hour* | *minute* | *second* | *fraction* | *init* | *fin* | *intersect* | *union* | *sp_convert*

The constructions **on** gives a timeslice according to the attribute of existence, the construction **as of** gives a timeslice according to the attribute of belief. The selector gives the value of temporal constant TP or DTP. The constructions **on** and **as of** act in such a manner that the following predicate is assumed in formulas of the *filter* : *in*(**valid**, < selector >) or *in*(**belief**, < selector >) (here the temporal predicates are not given according to attribute of existence or attribute of belief).

Note that in this case selector should be given as a TP. If the selector is given as a DTP the predicate $in(< selector >, \mathbf{valid})$ or $in(< selector >, \mathbf{belief})$ is assumed in formula.

f7. If f - formula and the temporal restrictions according to the attribute of existence and/or according to the attribute of belief are not given, then f is interpreted by the following formulas:

- f **on current as of current**;
- f **on current**;
- f **as of current**.

f8. If f is a formula, (f) is a formula too.

In these formulas the predicate in is defined as follows:

$$in(x, y) \equiv starts(x, y) \vee during(x, y) \vee finishes(x, y).$$

2.5.2 Formulas with Aggregate Functions

The conception of temporal DB enables to build special types of aggregate functions. Some examples of such functions are considered below:

$\langle \text{formula with function} \rangle ::= \langle \text{identifier function} \rangle (t, f) \mid \langle \text{identifier function} \rangle (t, f) \Theta \langle \text{expression} \rangle$
 $\langle \text{expression} \rangle ::= \langle \text{temporal literal} \rangle \mid \langle \text{arithmetic expression} \rangle$
 $\langle \text{identifier function} \rangle ::= asc \mid const \mid desc \mid t_asc \mid t_const \mid t_desc \mid max \mid min \mid count \mid total \mid average \mid$
 $t_max \mid t_min \mid t_count \mid t_total \mid t_average$

Here t is a certain attribute, according which the additional criterion of the selection is given. f is a formula in *filter* language, $\Theta \in \{=, \neq, \geq, \leq, >, <\}$. The aggregate functions in formula are used in order to give additional criterions of the selection according to the truth set of f which is one of the arguments of the function. The other argument of the function is the attribute. The attribute should get into free variables of f . If f has a free attribute only (the argument of the function), the formula with temporal function becomes closed. If f has more than one free variable, then its truth set is a subset of projections of the truth set of f onto the area of values of free variables (in the given formula with the function). Assume, for the future, that $f(x, y)$ is a certain formula with two free variables x and y (x is an argument of the function) in *filter* language.

Function asc. Let us the formula is of the form $asc(x, f(x, y))$. The meaning of this formula in that from the truth set of formula $f(x, y)$ chosen are those values y , such that all values of x (ordered chronologically and satisfying with y the formula $f(x, y)$) form a monotone ascending sequence.

Function desc. Let us the formula is of the form $desc(x, f(x, y))$. The meaning of this formula in that from the truth set of formula $f(x, y)$ chosen are those values y , such that all values of x (ordered chronologically and satisfying with y the formula $f(x, y)$) form a monotone descending sequence.

Function const. Let us the formula is of the form $const(x, f(x, y))$. The meaning of this formula in that from the truth set of formula $f(x, y)$ chosen are those values y , such that all values of x (ordered chronologically and satisfying with y the formula $f(x, y)$) do not change in time.

Function t_asc. Let us the formula is of the form $t_asc(x, f(x, y))$. The truth set of this formula is the normalized relation, the areas of definition of which are t and y . The value of t in each tuple of such a relation is the value of maximal TP during which all values of x (satisfying with y the formula $f(x, y)$) form a monotone ascending sequence.

Function max. Let us the formula is of the form $max(x, f(x, y))$ *gt* 20. The meaning of this formula in that from the truth set of formula $f(x, y)$ chosen are those values y , such that the maximal of the values of x (in all times) satisfying with y the formula $f(x, y)$ is larger than 20.

Function t_max. Let us the formula is of the form $t_max(x, f(x, y))$ *gt* 20. The truth set of this formula is the normalized relation, the areas of definition of which are t and y . The value of t in each tuple of such a relation is the value of maximal TP during which the maximal value of x (satisfying with y the formula $f(x, y)$) is larger than 20.

The semantics of other formulas with functions is defined analogously.

2.6 Procedural Tools

At the disposal of UCDM there is an advanced procedural apparatus for data modeling in conceptual level. The use of UCDM procedural tools for CTDB modeling is inevitable in case, when there is a need of setting temporal dependencies in conceptual level between relations of more general type, than ones, which are given by declarative tools. Unlike many researches in area of temporal DB, in which the direct change of **update** and **delete** operator semantics is provided, in considered extensional relational DM the change of the semantics of these operators is attained by the use of procedural tools. The procedures that provide the change of **update** and **delete** operator semantics are connected in conceptual scheme with relations, which need to be changed.

For example:

For r :
before **update** updproc;
before **delete** delproc.

Proceed from definition of temporal relation, in considered extension the previous states of tuples of relation r are inserted into relation s_r (the tuples which are changed or deleted from r). Besides:

- if $fin(valid)$ is equal to **inf** in the tuple that has been changed, then the value of $fin(valid)$ changes and becomes equal to such a DTP t_1 , that the predicate $meets([init(valid), t_1], t_2)$ is true. Here $t_2 = valid$ of source tuple;
- $fin(belief)$ of changed or deleted tuple as well varies and becomes equal to the current DTP.

2.7 Integrity Constraints

The suggested CTDB conception assumes the separate organization of current state of DB. Only the tuples of current state of DB are changed. To give the integrity constraints for current state of DB, one can use the traditional tools of relational DM: functional dependencies, closed formulas, etc.

By means of closed formulas in *filters* language the examples of temporal integrity constraints are considered below:

1. Let the total functional dependence [12] exists between relations r_1 and r_2 :

$$r_1(K) \rightarrow r_2(K), K - \text{primary key of relation } r_2.$$

Then the following closed formula

$$\text{Ex } K (\text{All } t_1, t_2 (r_1(K, t_1) \ \& \ r_2(K, t_2) \rightarrow \text{before}(t_2, t_1)))$$

excludes the existence of tuples in relation r_1 , that do not satisfy the predicate $\text{before}(t_2, t_1)$, i. e. the formula defines the temporal sequence of events.

Assume r is a relation for keeping such objects, occurrence of which must form a monotone ascending sequence according to the value of the attribute x in time. Then the following closed formula

$$\text{asc}(x, r(x))$$

excludes the existence of such tuples in relation r , that break the given order.

2.8 The Partial Support of Evolution of Database Schemes

During the life of DB their schemes can be changed. The queries on DB can be oriented to the current state of DB scheme as well as to its previous states. Therefore the necessity of keeping of the changed DB schemes follows. The following cases are considered:

- a) Some attributes are added in scheme of relation;
- b) Some attribute are deleted from scheme of relation.

Case a). Let $R_1 = \langle A, K \rangle$ is a scheme of temporal relation r_1 , where $A = \{A_1, A_2, \dots, A_n, \text{valid}, \text{belief}\}$ and $K = \{A_{i_1}, A_{i_2}, \dots, A_{i_l}, \text{belief}\}$ is a primary key ($l \leq n$). Let us new attributes $A_{n+1}, A_{n+2}, \dots, A_{n+m}$ are added in scheme of relation r_1 at a DTP t . In considered variant of evolution of DB scheme, this brings to the creation of new temporal relation r_2 with scheme $R_2 = \langle A', K \rangle$, where $A' = \{A_{i_1}, A_{i_2}, \dots, A_{i_l}, A_{n+1}, A_{n+2}, \dots, A_{n+m}, \text{valid}, \text{belief}\}$. To obtain a single composite representation of an object; the abstraction join [12] is applied. In other words, with the help of abstraction join the relation r with the scheme $R = \langle A \cup A', K \rangle$ is declared as the join of relations r_1 and r_2 . In virtual relation r the attributes $A_{n+1}, A_{n+2}, \dots, A_{n+m}$ set to *null*, in those tuples, in which the predicate *before* ($r_1.\text{belief}, t$) satisfied. Note that the virtual relation r consists of the tuples of the type:

$$r = (r_1 \bowtie r_2) \cup \{r' (A_1, A_2, \dots, A_n, A_{n+1}, A_{n+2}, \dots, A_{n+m}, \text{valid}, \text{belief}) \mid (r' [A] \in r_1 \overset{semi}{\bowtie} (\Pi_K(r_1) \setminus \Pi_K(r_2))) \& (r' [A_{n+1}] = \perp) \& \dots \& (r' [A_{n+m}] = \perp)\}.$$

Case b). Let $R_1 = \langle A, K \rangle$ is a scheme of temporal relation r_1 , where $A = \{A_1, A_2, \dots, A_n, \text{valid}, \text{belief}\}$ and $K = \{A_{i_1}, A_{i_2}, \dots, A_{i_l}, \text{belief}\}$ is a primary key ($l \leq n$). Let $A' \subset A$ is a set of attributes that are *deleted* from scheme r_1 at a DTP. This brings to the creation of new temporal relation r_2 with scheme $R_2 = \langle A \setminus A', K \rangle$. In this case the support of evolution of DB scheme is obtained by applying abstraction generalization. In other words, the virtual relation r with scheme $R = \langle A \setminus A', K \rangle$ is declared a generalization of r_1 and r_2 . Let us $L = A \setminus A'$. Then the virtual relation r consists of tuples of type:

$$r = \Pi_L(r_1) \cup r_2.$$

In description of semantics of virtual relations, the operations of traditional relational algebra has been used.

2.9 Examples

Assume in a certain DB -"Faculty employee" a type of object **employee** is defined

Queries:

1. Find the names of the employees of faculty who a) are Professors; b) have worked as researchers before entering the post-graduate studies; c) have been researchers and graduate students for no more than five years:

```
Ex T1, T2, number((post = 'Prof.' & employee(number, name)) &
le(length(T3), 'interval (5)yy (1)to yy') & (((post = 'researcher' & ¬ equals(fin(belief), inf) &
employee(number, T1) on union(¬inf, inf)) & (post = 'post-graduate' & ¬ equals(fin(belief), inf) &
employee(number, T2) on union(¬inf, inf))) & before(T1, T2) form T3 = union(T1, T2)))
with (T1 = valid, T2 = valid),
```

in the subformula $\text{post} = \text{'Prof.'} \& \text{employee}(\text{number}, \text{name})$ of formula the condition $\text{in}(\text{current}, \text{valid})$ and $\text{in}(\text{current}, \text{belief})$ is assumed default.

2. Find names of employees of the faculty such that: a) they are Professors; b) their salary hasn't decreased throughout the TP (T_1):

```
asc(salary, Ex number((post = 'Prof.' & employee(number, name)) &
(overlaps(T1, valid) & employee(salary, number) as of union(¬inf, inf)))).
```

3 Conclusion

The extension of relational DM for CTDB logical modeling is suggested. The time is considered as a totally ordered set. This allows to provide the high - level interface for the user. Allowing the homogeneity, the modeling of the temporal aspect of DB is achieved by means of two temporal attributes. The temporal attribute is considered as ADT. This enables to realize CTDB without the change of DML on a conceptual level. The semantics of operations on temporal attributes is expressed by predicates connected with ADT. During the formation of logical conditions of data selection in temporal DB, the temporal predicates as well as other types of atoms are used as atoms of condition. A formal definition of temporal relation which is based on abstraction of generalization is given. The use of abstraction of generalization allows to combine the use of the current state of DB with the possibility of access to objects characteristics, connected with past time moments. So, the working effectivity with temporal DB is provided. The existence of special constructions **on** and **as of** in language allows to consider temporal DB in the context of traditional DB, and provides high - level interface for the user. The semantics of **update** and **delete** operators is changed. The use of procedural tools allows to avoid the direct change of semantics of these operators. So, a flexible mechanism for work with DB during the actualization of temporal DB is suggested. Also suggested are new aggregate functions considering the time aspect of the object behavior in given enterprise. The implementation of these functions results from the nature of temporal DB and is a natural extension of the relational DM. An advanced apparatus of time attribute transformation is suggested. The semantics of relational algebra operations is changed in order to reflect the time aspect. The suggested concept of temporal DB assumes the separate storage of DB current state. This allows to use the traditional tools of relational DM in order to provide the integrity constraints of DB. The usage of suggested language tools as a part of closed formulas in *filter* language allows to effectively give very complex temporal integrity constraints. The questions of DB scheme evolution are considered in context of adding or deleting of some attributes in the temporal relation scheme. In order to support DB scheme evolution the traditional declarative tools of data modeling are used. The use of abstraction generalization and abstraction join enables to support CTDB with changing schemes, without physical reorganization of temporal DB.

The analysis and comparison with known approaches of support of CTDB shows, that the suggested approach takes in the best characters of these systems and has in its arsenal enough language tools for CTDB logical modeling.

ACKNOWLEDGMENTS

The author would like to express his gratitude to Professor of the Moscow State University L. A. Kalinichenko for scientific support during many years of cooperation in the area of databases. The author is very much thankful to Professor R. T. Snodgrass (University of Arizona) and to Professor S. K. Shoukouryan (Yerevan State University) for help which stimulated the author to write this paper. This research is partially supported by the Economics ministry of Republic of Armenia (Grant No 94 -7).

References

- [1] J. F. Allen, P. J. Hayes, A Common Sense Theory of Time, in Proc. of the ninth International Joint Conference on Artificial Intelligence, Los Angeles, pp. 528-531, 1985.
- [2] G. Ariav, A Temporally Oriented Data Model, ACM TODS, vol. 11, no. 4, pp.499-527, 1986.
- [3] A. Bolour, T. L. Anderson, L. J. Dekeyser, H. K. Wong, The Role of Time in Information Processing: A Survey, ACM SIGMOD Review, vol. 12, no. 3, pp. 27 - 50, 1982.
- [4] J. Chomicki, D. Toman, Implementing Temporal Integrity Constraints Using an Active DBMS, IEEE Transactions on Knowledge and Data Engineering, vol. 7, no. 4, pp. 566 - 582, 1995.
- [5] J. Clifford, D. S. Warren, Formal Semantics for Time in Databases, ACM TODS, vol. 8, no. 2, pp. 214 - 254, 1983.

- [6] E. F. Codd, Extending the Database Relational Model to Capture More Meaning, *ACM TODS*, vol. 4, no. 4, pp. 397 - 434, 1979.
- [7] S. K. Gadia, J. Vaishnav, A Query Language for a Homogeneous Temporal Database, in *Proc. of 4th Annual ACM SIGAT - SIGMOD Symposium on Principles of Database Systems*, New York:, ACM, pp. 51 - 56, 1985.
- [8] S. K. Gadia, A Homogeneous Model and Query Language for Temporal Databases, *ACM TODS*, vol. 13, no. 4, pp. 418 - 448, 1988.
- [9] ISO - ANSI (working draft) SQL2, ISO IEC JTC1 SC2 WG3 N440, ANSI X3H2 88 - 1, 1988.
- [10] S. Jones, P. Mason, R. Stamper, LEGOL 2.0: A Relational Specification Language for Complex Rules, *Information Systems*, vol. 4, no. 4, pp. 293 - 305, 1979.
- [11] F. Kabanza, J-M. Stevenne, P. Wolper, Handling Infinite Temporal Data, *Journal of Computer and System Sciences*, no. 51, pp. 3 - 17, 1995.
- [12] L. A. Kalinichenko, *Methods and Tools of Integration of Heterogeneous Databases (Russian)*, Moscow, pp. 423, 1983.
- [13] L. A. Kalinichenko, M. G. Manukyan, *The Language Tools of the Temporally-Oriented Data Model (Russian)*, *Programmirovaniye*, Moscow, pp. 72 - 85, 1990.
- [14] M. Koubarakis, J. Mylopoulos, M. Stanley, A. Borgida, Telos: Features and Formalization, *Technical Report KRR - TR - 89 - 4*, University of Toronto, Department of Computer Science, pp. 1 - 84, 1989.
- [15] M. G. Manukyan, *Temporal Data Model (Russian)*, Ph.D. Dissertation, Faculty of Numerical Mathematics and Cybernetics, Moscow State University, 1990.
- [16] N. G. Martin, S. B. Navathe, R. Ahmed, Dealing with Temporal Schema Anomalies in History Databases, in *Proc. of VLDB - 87*, pp. 177 - 184, 1987.
- [17] G. Özsoyoğlu, R. T. Snodgrass, Temporal and Real - Time Databases: A Survey, *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 4, pp. 513 - 532, 1995.
- [18] J. M. Smith, D. C. P. Smith, Database Abstractions: Aggregation and Generalization, *ACM TODS*, vol. 2, no. 2, pp. 105 - 133, 1977.
- [19] R. T. Snodgrass, Research Concerning Time in Databases, *SIGMOD Record*, vol. 15, no. 4, pp.19 - 39, 1986.
- [20] R. T. Snodgrass, The Temporal Query Language TQuel, *ACM TODS*, vol. 12, no. 2, pp. 247 - 298., 1987.
- [21] R. T. Snodgrass, I. Ahn, G. Ariav, D. S. Batory, J. Clifford, C. E. Dyreson, R. Elmasri, F. Grandi, C. S. Jensen, W. Käfer, N. Kline, K. Kulkanri, C. Y. T. Leung, N. Lorentzos, J. F. Roddick, A. Segev, M. D. Soo, and S. M. Sripada, *TSQL2 Language Specification*, *ACM SIGMOD Record*, vol. 23, no. 1, pp. 65 - 86, 1994.
- [22] U. A. Tansel, Adding Time Dimension to Relational Model and Extending Relational Algebra, *Information Systems*, vol. 11, no. 4, pp. 343 - 355, 1986.