AN ABSTRACT INTRODUCTION TO THE TEMPORAL-HIERARCHIC DATA MODEL (THM) +

Ulrich Schiel ++

Dep. de Sistemas e Computação, Univ. Fed. da Paraíba
Av. Apr. Veloso, 882 - 58100 Campina Grande/Pb - Brazil

Abstract

The static concepts of a semantic database model are formalized by axioms of first-order predicate calculus and set theory. Then, the basic operations are defined and, in order to maintain a database consistent, a set of dynamic axioms and side-effect axioms is stated using dynamic and temporal logic.
The necessity and sufficiency of the dynamic rules is stated and an example shows how the side-effects works.

I. PRELIMINARIES

The informal philosophical background for the concepts of the Temporal-Hierarchic Data Model is an idea of the existence of three worlds: a concrete world of physical things, an abstract world of 'metaphysical' things and the model world in which we model or represent concrete and abstract things /BN, Sc2/. From the first two, called real world, the part of interest for a specific application is the universe of discourse. In the model world we distinguish two levels /Su, BN, ANSI/, the conceptual and the internal level. There is a mapping from the universe of discourse to the conceptual level called representation. It maps objects to entities, object types to classes, properties and associations to relationships, processes to operations and ocurrences to events. The image of the complete universe of discourse gives the conceptual schema and the information base /ISO2/. The inverse of the representation is the interpretation of concepts from the model world.

+ financial support from the Stifterverband der Deutschen Wissenschaften
++ actually at Institut für Informatik, Universität Stuttgart, Germany.

The theoretical basis is set theory and first order predicate calculus with extensions to dynamic and temporal logic. In order to facilitate the lecture and not exagerate formalization some trivial details are avoided and all not quantified variables are considered universally quantified.

The static concepts of the data model are specified by a series of axioms which must hold in all states of the database. To guarantee this, restrictions on the basic operations are stated by dynamic axioms and single operations are extended to valid database transformations by so-called side-effect axioms.

There is a general attempt to formalize the concepts of semantic data models /BM/, as was done for TAXIS /BW, MW/ and SHM+ /Br/. The difference to the other approaches is that THM includes three abstractions with several special cases, time concepts and dynamic aspects (operations) with corresponding semantic side effects. Since all semantic models have several similarities the results of this paper can easily be used for other models.

An informal introduction to THM, with illustrative examples can be seen elsewhere /Sc1, Sc2, SFCN/.

II. STATIC CONCEPTS

The only basic primitive is the entity. It may be interpreted as the representation of an object from the real world into the (abstract) information base of the conceptual level. Entities at this level are abstract ideas, they cannot be 'touched' or 'seen' and only be identified by their properties or relationships to other entities /Fa/.

A <u>class</u> is a pair

$$C = < I_C , M_C >$$

where I_C is the identification of C composed of its name, N_C, and eventually other information about the class such as an informal description, statistical values such as number of acesses, number of members, etc., and other values related to the class as a whole (so-called class-relationships). M_C is a set of entities called the members of C. Here we make the observation that in this chapter we consider only the static aspect of the concepts. The content of classes and relationships changes over time and the correct writing of a class C is C_t, meaning 'class C at time instant t' and C is the family of all C_t. The first axiom is:

A1: $N_C=N_d \Rightarrow C=D$ (distinct classes must have distict names)

Depending on the context we speak of a class C and mean, alternatively, C, M_C or N_C. In this sense we always write $e \in C$ istead of $e \in M_C$.

Given two classes C and D, a <u>relation</u> r from C to D is a system

$$r = < N_r , R_r , min_r , max_r >$$

where N_r is the name of r, $R \subset M_C x M_d$, min_r is a positive integer and max_r is a positive integer or a special symbol, denoted as *. Min and max are called the minimal and maximal cardinalities of the relation r.

As we do for classes, in the text we do not always distinguish between r, N_r and R_r. We also refer to a relation as r(min,max) or, if we want identify the related classes, CrD. If CrD is a relation and c a member of C we define r(c) as the set of members of D related to c, i.e.

$$r(c) = \{ d \in D / <c,d> \in R_r \}$$

and r(c,d) is interpreted as a predicate which is true iff $<c,d> \in R_r$. The following axioms must hold

A2: $(CrD \wedge CsE \wedge N_r=N_s) \Rightarrow r=s$
(a class cannot have two relations with the same name)

A3: $max_r \neq * \Rightarrow min_r \leq max_r$

A4: $c \in M_C \Rightarrow$
$(\# \{<c,d> / <c,d> \in R_r\} \geq min_r \wedge$
$(max_r \neq * \Rightarrow \# \{<c,d>/<c,d> \in R_r\} \leq max_r))$

A5: $r(c,d) \Rightarrow \exists C,D(c \in C \wedge d \in D \wedge CrD)$
(entities can be related only if the corresponding classes are related)

If we admit the existence of a special member of each class, called 'nothing' /BW/, an inverse of axiom A5 also holds

A5': $CrD \wedge c \in C \Rightarrow \exists d \in D (r(c,d))$

Each relation has an inverse

A6: $CrD \Rightarrow \exists s(DsC \wedge (r(c,d) \Leftrightarrow s(d,c))$

s is also denoted as r^{-1}. Depending on the maximal cardinalities of a relation and its inverse, the following characterizations are obtained:

if $max_r = max_{r^{-1}} = 1$
then r is one-to-one,
if $max_r > 1$ and $max_{r^{-1}} = 1$
then r is one-to-many,
if $max_r = 1$ and $max_{r^{-1}} > 1$
then r is many-to-one, and
if $max_r > 1$ and $max_{r^{-1}} > 1$
then r is many-to-many.

Now we introduce the hierarchical structures which can occur in a database schema. The structures and some special cases are characterized by a predicate which is true iff the corresponding structure occurs. Thus the first equivalence $\Leftrightarrow$ of the axioms may be interpreted as a definition $\Leftrightarrow_{def}$.

To give a precise and meaningful characterization of the generalization/specialization hierarchy we define first a <u>role</u> as a disjunctive predicate

$$p(e) = p_1(e) \vee \ldots \vee p_n(e)$$

which can be applied to entities e of a generic class G and such that $p_i(e)$ is true iff e is a member of subclass C_i. Then, a <u>generalization</u> G of classes C_1, ... , C_n by role p is given by

A7: for i=1,2,...n is-a(C_i,G,p)
$\Leftrightarrow ((e \in G \wedge p_i(e)) \Leftrightarrow e \in C_i)$

This characterizes the classes C_i as subclasses of G with $M_{C_i} \subset M_G$. A generalization is denoted as $p(G)=\{C_1,\ldots,C_n\}$ and $p_i(G)=C_i$. If $C=C_i$ we also write $p_C(e)$ instead of $p_i(e)$. In practice the predicate can be determined by the values of a relation of the generic class. A role applied to a class D is <u>disjunctive</u> if the subclasses are disjoint:

A8: for $1 \le i,j \le n \wedge i \neq j$
$$\text{disjunctive}(D, C_1, .., C_n, p) \Leftrightarrow (p_i(e) \Rightarrow \sim p_j(e)))$$

If each entity of the generic class is in at least one subclasss the role is <u>covering</u>:

A9: $$\text{covering}(G, C_1, .., C_n, p) \Leftrightarrow (e \in G \Rightarrow \exists i \; p_i(e))$$

From the last definitions it is immediate that

i) if p is disjunctive then $C_i \cap C_j = \emptyset$ for $i \neq j$

ii) if p is covering then $\bigcup_{i=1}^{n} C_i = G$

The second hierarchy is obtained when entities are joined to form new compound entities. A class A is an <u>aggregation</u> of classes $C_1, \ldots, C_n$ if

A10: $$\text{aggregated-by}(A, C_1, .., C_n) \Leftrightarrow M_a \subset M_{C_1} x \; \ldots \; x M_{C_n}$$

and A is an <u>aggregation of</u> $C_1, \ldots, C_n$ <u>by relations</u> r_{ij} iff exactly the entities related by r_{ij} are in the aggregated class:

A11: $\text{aggregated-by}(A, C_1, .., C_n, \{r_{ij}\}) \Leftrightarrow$

i) $\langle c_1, .., c_n \rangle \in A \Leftrightarrow$
$(\langle c_1, .., c_n \rangle \in C_1 x..x C_n \wedge$
$(r \in \{r_{ij}\} \Rightarrow \exists \; 1 \le k, l \le n \; r(c_k, c_l))$
(the components of members of A must be in the component classes and related together)

ii) $CrD \Rightarrow (r \in \{r_{ij}\} \Rightarrow C, D \in \{C_1, .., C_n\})$

iii) all component classes are transitively related:
for $i,j=1,\ldots,n \; \exists \; k_1, .., k_m$
$(C_i r_{ik_1} C_{k_1} r_{k_1 k_2} \ldots r_{k_m j} C_j) \wedge$
$C_{k_1}, \ldots, C_{k_m} \in \{C_1, \ldots, C_n\}$)

The last hierarchy is given by the definition of group entities as sets of single entities. A class G is a <u>grouping</u> of another class C by predicate p, if its elements are sets of elements of C and p holds between elements and groups:

A12: $\text{is-elem}(C, G, p) \Leftrightarrow G \subset \mathcal{P}(C) \wedge$
$((g \in G \wedge x \in C \wedge x \in g) \Rightarrow p(x,g))$
$\wedge \; (p(x,g) \Rightarrow (x \in C \Leftrightarrow x \in g))$

A grouping is <u>disjunctive</u> if each entity occurs in at most one group:

A13: $\text{disjunctive}(C, G, p) \Leftrightarrow$
$(\text{is-elem}(C, G, p) \wedge$
$(g_1, g_2 \in G \Rightarrow g_1 \cap g_2 = \emptyset))$

A grouping G of C is <u>covering</u> if all entities of the element class occur in at least one group:

A14: $$\text{covering}(C, G, p) \Leftrightarrow (\text{is-elem}(c, G, p) \wedge \cup G = C)$$

II. <u>TIME CONCEPTS</u>

Time considerations were also made for the Infological Model /Su/ and for CSL of the Object-Role Model /BFM/. In our context time is considered as a class T of tuples $t=(t_1:u_1, \; \ldots \; ,t_n:u_n)$ such that

i) there is a set of constants $p_2, \ldots, p_n$ called periods,

ii) $t_1, .., t_n$ are positive integers such that $t_i < p_i$

iii) u_i are strings, called units (such as years, days, seconds),

iv) for $i=2, .., n$ $\; p_i:u_i = 1:u_{i+1}$ (for ex. 60:seconds=1:minute)

The lowest unit u_n is called the <u>granularity</u> of the time points. This means that a time point is not an infinitely small point but a 'little interval'. There are two types of 'subtuples' of a time point $t = \langle t_1, .., t_n \rangle$. For $1 < k < n$, points of type $\langle t_1, .., t_k \rangle$ are time points with a higher granularity (e.g. days instead of seconds); and points of type $\langle t_k, .., t_n \rangle$ are <u>periodical</u> (e.g. each minute).

There is a special time point (or function or variable) which reflects the 'present moment' /Al/ or 'now' /An/. We call it <u>clock</u>, as the representation of the time instant from the real world reported by a clock (with calendar). If only a special unit is needed we write 'clock.day', 'clock.second' and so on.

T has a natural ordering relation < of time points given by the concepts of 'before' and 'after'. A class I is a <u>time interval class</u> if it is a grouping of a time class T such that each time point between two points in a time group (or interval) is in this group:

A15: $\text{interval}(I, T) \Leftrightarrow$
$(p, r \in T \wedge p, r \in i \in I) \Rightarrow (p < q < r \Rightarrow q \in i)$

The lower limit of a time interval is called the <u>from</u> point and the upper limit is the <u>to</u> point, and if t=from and s=to we write i=(t,s). The interval (t,*) means 'from t to now'.

A class with time is an aggregated class $C'=C\times I$ of a class C and a time interval class I, such that

A16: timed(C',C,I) <=>
interval(I,T) $\wedge$ aggregated-by(C',C,I) $\wedge$
$(<c,i_1>, <c,i_2> \in C' \Rightarrow (i_1=i_2 \vee i_1 \cap i_2=\emptyset))$

A17: timed(C',C,I) =>
$(x \in C \Leftrightarrow \exists t(<x,(t,*)> \in C'))$

If we update one relationship (e.g. 'has-salary' from EMPLOYEE to SALARY class) it may be of interest to preserve the old salary of the employee /WFW/. A relation with old values from C to D is a system

$$r' = < N, R', min, max >$$

such that

i) $<N,R'>$ is a class with $R' \subset M_c x M_d x I$ where I is a time interval class,

ii) $<c,d,i_1>, <c,d,i_2> \in R' \Rightarrow (i_1=i_2 \vee i_1 \cap i_2 = \emptyset)$

iii) for each time point $t \in i \in I$, if R_t is the t-projection of R' on $M_c x M_d$, i.e. $R_t = \{ s \in M_c x M_d \,/\, \exists i(t \in i \vee <s,i> \in R' \}$ then $r_t = < N , R_t , min , max >$ is a relation from C to D,

iv) $\forall c \in C \exists t_0 \forall t(t \geq t_0 \wedge t \leq clock \Rightarrow \exists i,d\ (t \in i \wedge <c,d,i> \in R')$

This concept determines the axiom

A18: old(r') <=> conditions i) to iv) above holds.

In order to avoid an indefinite increasing of the content of a database using classes with time a concept of lifetime was also considered for THM /Sc2/. The formalization of this concept is left to the reader.

Another useful concept of THM is the possibility to make some statements about the past and future of the entities from a class. We can state a so-called pre-post relation between classes (denoted as C >--> D), which means that entities deleted from C may be inserted into D and entities inserted into D may be originated from C. C is a pre-class of D and D a post-class of C. If we replace the 'may' by a 'must' we have an exclusive pre- and/or post class (denoted as C>>-->>D). For example

CANDIDATE >-->> EMPLOYEE >>--> CANDIDATE

means that some candidates can be employed in future but all employees must be candidates before beeing employed. All dismissed return to be candidates but some candidates never were employed. Now the axioms

A19: excl-pre(C,D) <=>
$(\sim x \in D \wedge o(x \in D) \Rightarrow x \in C)$

A20: excl-post(C,D) <=>
$(x \in C \wedge o(\sim x \in C) \Rightarrow o(x \in D)$

(the temporal operator o(p) means 'in the next state p is true')

III. DYNAMIC ASPECTS

For the description of database operations it is usual to use the notions of database state and state transformation /BS, SNF, SFNC/. We concentrate only on the operations themselves and use an approach related to the specification of abstract data types /GH, Sc1/. Consider the representation of all states of the universe of discourse in the past, present and future /LMP/ and call this UDD (universe of discourse description /ISO1/). UDD is composed of:

UE = { e / e is an entity of UDD }
UC = { C / C is a class in UDD }
UR = { r / r is a relation in UDD }

The universe of entities is a union of disjoint sets, called entity types. For an entity e the type it belongs to is denoted as Te, and we say that e is of type Te.

UC has also a decomposition into disjoint subsets, called metaclasses /MW/, such that there is a bijection between entity types and metaclasses. If MC $\subset$ $\mathcal{P}$(UC) is the set of all metaclasses and ET the set of all entity types, we have the mappings

rep : ET -> MC and int : MC -> ET

The basic operations of THM are insert and delete of entities and establish and remove of relationships. If T and S are entity types and M=rep(T) a corresponding metaclass, then we define:

1) insert ins : TxM -> M
$(e,C) \rightarrow C \cup \{e\}$

2) delete del : TxM -> M
$(e,C) \rightarrow C - \{e\}$

3) establish est : TxSxUR -> UR
$(e,g,r) \rightarrow r \cup \{<e,g>\}$

4) remove rem : TxSxUR -> UR
$(e,g,r) \rightarrow r - \{<e,g>\}$

Additional operations update and move can be defined as combinations of basic operations or directly as

5) move
   mov : TxMxM -> M x M
   (e,C,D) → (C',D') where C'=C-{e} and D'=D ∪ {e}

6) update
   upd : TxSxSxUR -> UR
   (e,g,h,r) → (r-{<e,g>}) ∪ {<e,h>}

Two special operations for groups and its elements are needed: g-insert(e,g) assigns e as a new element of g and g-delete(e,g) deletes e from g. The effect of these functions is the same as for insert and delete, only the domains are different.

The definitions above determine only the functional effect of the operations without considerations about a possible conceptual schema with its own semantics. As the axioms in the first part, we need similar statements which must hold for operations acting on a schema designed with THM. These statements are called schema side effects, because the execution of one primitive operation has as consequence other primitive operations that guarantees the semantic integrity of the database. For a concrete application additional side-effects can be explicitly defined by events and triggers, these are the user side effects. The side effects, in conjunction with the operation concept of THM /Sc2/ guarantees completely the integrity of a database, avoiding the necessity of stating an explicit set of integrity constraints.

The axioms for the dynamic rules are written in dynamic logic /Ha/ with two types of formulas:

1) p ⊢ [op] => q for the dynamic axioms means 'in a state with p true, op is allowed only if q is true';

2) p ⊢ [op1] => [op2] for the side effects means 'in a state with p true execution of op1 implies execution of op2'.

The temporal operator o(p) ('in the next state p is true') /MP/ was also needed for some side effects. A similar approach for conceptual schema specification can be seen in /SFCN/. First of all, some general rules, called dynamic axioms, are:

DA1: ⊢ [establish(x,y,r)] =>
   ∃ C,D(x∈C ∧ y∈D ∧ CrD ∧
   (max_{r}=* ∨ #{<x,z> / r(x,z)}<max_r)
   (establish must keep the maximal cardinality)
DA2: ⊢ [remove(x,y,r)] =>
   #{ <x,z> / r(x,z)} > min_r
   (remove must keep the minimal cardinality)
DA3: is-a(C,D,p) ⊢ [insert(x,C)] => p_C(x)
   (insert must keep the role)
DA4: covering(D,C_1,..,C_n) ∧
   (is-a(C_i,D,p) => ~ x∈C_i)
   ⊢ [insert(x,D)] => ∃ i(p_{C_i}(x))
   (in a covering generalization we can not allow an entity only in the generic class)
DA5: timed(C',C,I)
   ⊢ [insert(x,C)] => ~ <x,(t,*)>∈C'
   (in a class with time we cannot insert an entity actually present)
DA6: is-elem(C,G,p)
   ⊢ [g-insert(x,g)] => p(x,g)
   (group elements must keep the grouping predicate)

These axioms establish that the operations are allowed to be executed only if some conditions hold. Before presenting the side effects we define two predicates about entities

part(x_i,y) : entity x_i is the i-th component of the aggregated entity y
aggregated(y,x_1,..,x_n,r_1,..,r_m) : entity y is is composed of x_1,..,x_n related by r_1,..,r_m; in this case we write also y = <x_1,..,x_n>

The applicability of a side effect depends on the hierarchical position of the affected class. We present the possible side effects for each relative position of the class in the three hierarchical structures generalization, aggregation and grouping with its inverses, called specialization, decomposition and dissolution respectively.

GENERALIZATION

SE1: is-a(C,D) ∧ ~ in(x,D)
   ⊢ [insert(x,C)] => [insert(x,D)]
   (an entity of the subclass must be in the superclass)
SE2: is-a(C,D) ∧ disjunctive(D,C_1,..,C_n)
   ∧ ∃ i(1≤i≤n ∧ x∈C_i ∧ C≠C_i)
   ⊢ [insert(c,C)] => [delete(x,C_i)]
   (an insert may not violate a disjoint generalization)
SE3: is-a(C,D) ∧ covering(D,C_1,..,C_n) ∧
   (C_i≠C=> ~ x∈C_i)
   ⊢ [delete(x,C)] => [delete(x,D)]
   (a delete may not violate a covering generalization)

SE4: $x \in C \wedge$ is-a(C,D,p)
$\vdash$ ([establish(x,y,r)] $\vee$
[remove(x,y,r)]) =>
$(p_C(x) \wedge o(\sim p_C(x)) \Rightarrow$ [delete(x,C)]
$\wedge(\sim p_C(x) \wedge o(p_C(x)) \Rightarrow$ [insert(x,C)]
(if as a consequence of an alteration of a relationship an entity is no longer allowed to remain in a subclass it must be moved to another compatible subclass).

SPECIALIZATION

SE5: is-a(C,D,p) $\wedge\ p_C(x)$
$\vdash$ [insert(x,D)] => [insert(x,C)]
(a new entity of a generic class must be inserted in all compatible subclasses)

SE6: is-a(C,D) $\wedge\ x \in C$
$\vdash$ [delete(x,D)]=> [delete(x,C)]

AGGREGATION

SE7: aggregated$(y,x_1,..,x_n,r_1,...,r_m) \wedge$
$y \in D \wedge (1 \le i \le n \Rightarrow x_i \in C_i)$
$\vdash$ [delete(x_i,C_i)] => [delete(y,D)]
(without one component an aggregated entity must be deleted)

SE8: aggregated$(y,x_1,..,x_n,r_1,..,r_m) \wedge$
$y \in D \wedge (1 \le i \le n \Rightarrow x_i \in C_i) \wedge r_k \in \{r_1,..,r_m\}$
$\vdash$ [remove(x_i,x_j,r_k)]=>[delete(y,D)]
(for an aggregation by relationships these relationships must hold for the aggregated entities)

SE9: aggregated-by$(D,C_1,..,C_n,r_1,..,r_m)$
$\wedge\ x_1 \in C_1 \wedge .. \wedge x_n \in C_n \wedge$
$\exists !\ r_k \in \{r_1,..,r_m\}$
$(C_i r_k C_j \wedge \sim$ related$(x_i,x_j,r_k))$
$\vdash$ [establish(c_i,c_j,r_k)] =>
[insert$(\langle x_1,..,x_n\rangle,D)$]
(as inverse of SE8, if for a set of entities for which all relations of an aggregation hold, the corresponding aggregated entity must be in the aggregated class).

DECOMPOSITION

SE10: aggregated-by$(D,C_1,..,C_n)$
$\vdash$ [insert(y,D)] => $(1 \le i \le n \wedge$
part$(x_i,y) \Rightarrow$ [insert(x_i,C_i)])
(the parts of an aggregated entity must be in the component classes)

SE11: aggregated-by$(D,C_1,..,C_n,r_1,..,r_m)$
$\vdash$ [insert(y,D)] => $(1 \le i \le n \wedge$
part$(x_i,y) \Rightarrow$ [insert(x_i,C_i)] $\wedge$
(is-related$(C_i,C_j,r_k) \wedge$
$r_k \in \{r_1,..,r_m\} \wedge$
=> [establish(c_i,c_j,r_k)]))
(same as for SE10 with the addition that the corresponding relationships are established)

GROUPING

SE12: is-elem(C,G,p) $\wedge\ g \in G \wedge p(x,g)$
$\vdash$ [insert(x,C)] <=> [g-insert(x,g)]
(if p(x,g) holds then x is of the element class iff it is in g)

SE13: covering(C,G,p) $\wedge \sim \exists\, g(p(x,g))$
$\vdash$ [insert(x,C)] => [insert({x},G)]
$\wedge$ p(x, {x})
(by a covering grouping each entity of the element class must be in at least one group)

SE14: is-elem(C,G,p) $\wedge\ x \in g$
$\vdash$ [delete(x,C)]=> [g-delete(x,g)]
(see comment on SE12)

DISSOLUTION

SE15: is-elem(C,G)
$\vdash$ [insert(g,G)]=> $(x \in g \wedge \sim x \in C$
=> [insert(x,C)])
(the elements of a group must be in the element class)

SE16: covering(C,G)
$\vdash$ [delete(g,G)] => $(x \in g \wedge$
$\sim \exists\, h(h \in G \wedge h \ne g \wedge x \in h)$
=> [delete(x,C)]

SE17: disjunctive(C,G)
$\vdash$ [g-insert(x,g)] => $\exists\, h((h \ne g \wedge x \in h$
=> [g-delete(x,h)])
(if a group-insert violates the disjoint property the entity is deleted from the other groups)

Among these, we have the side effects

SE18: related(x,y,r) $\wedge\ x \in C$
$\vdash$ [delete(x,C)] => [remove(x,y,r)]
(for a delete all existing relationships must be removed)

SE19: $\sim$ related(y,x,r^{-1})
$\vdash$ [establish(x,y,r)]
=> [establish(y,x,r^{-1})]
(each relation must have an inverse)

and some side effects envolving time aspects

SE20: timed(C',C,I)
$\vdash$ [insert(x,C)]
<=> [insert(<x,(clock,*)>,C')]

SE21: timed(C',C,I)
$\vdash$ [delete(x,C)] <=> $(\langle x,(t,*)\rangle \in C'$
=> [delete(<x,(t,*)>,C')] $\wedge$
[insert(<x,(t,clock)>,C')]
(in a class with time a deleted entity is moved to the past)

SE22: old(r') $\wedge$ t=clock
$\vdash$ [establish(x,y,r_t)]
<=> [insert(<x,y,(t,*)>,R']
(this and the next side effect maintains relations with old values)

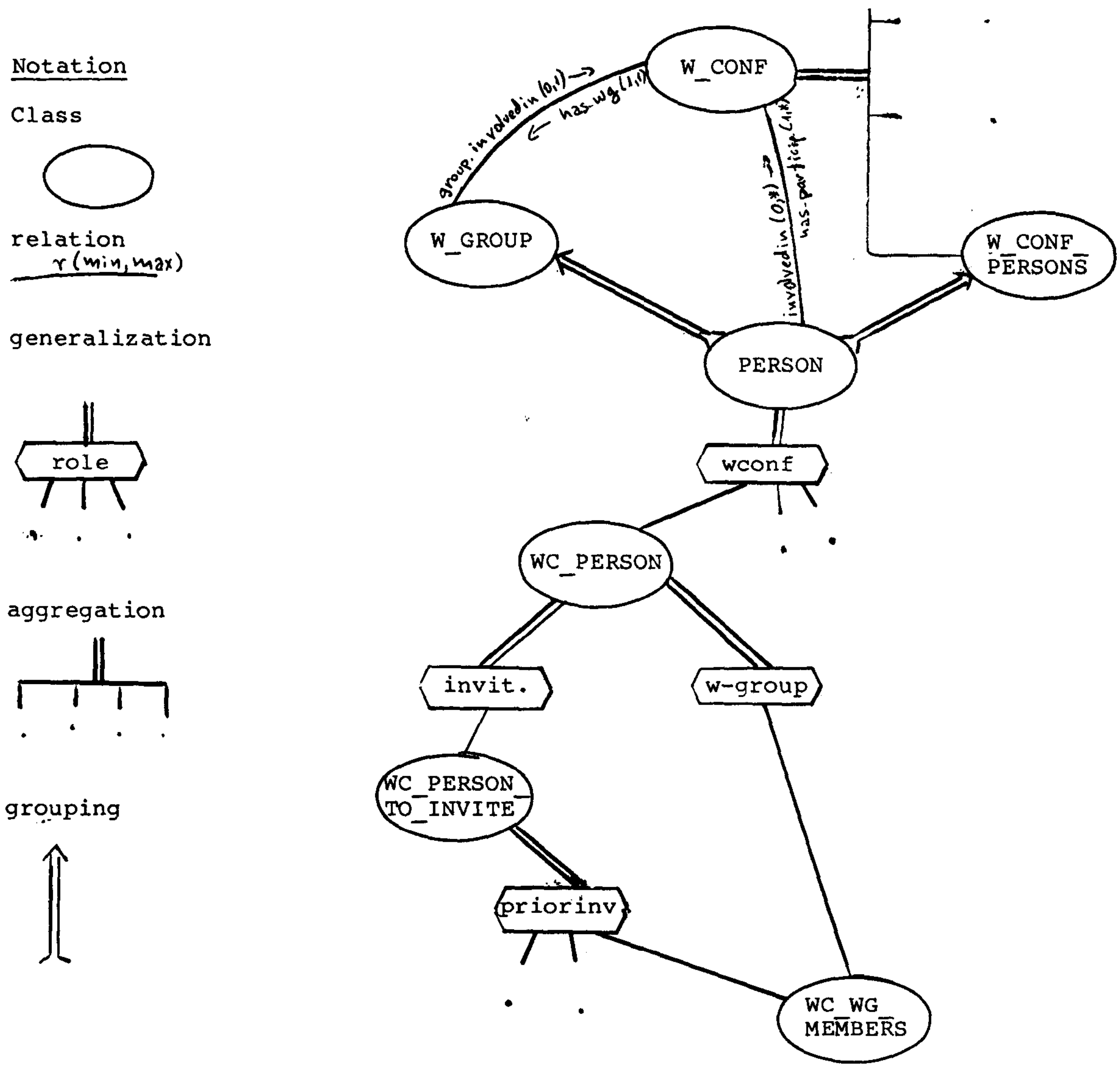
SE23: old(r') ∧ t=clock
⊢ [remove(x,y,r_t)]
<=> (<x,y,(t_o,*)>∈R' =>
[delete(<x,y,(t_o,*)>,R')] ∧
[insert(<x,y,(t_o,t)>,R')]
SE24: excl-pre(C,D)
⊢ [insert(x,D)] => [delete(x,C)]
SE25: excl-post(C,D)
⊢ [delete(x,C)] => [insert(x,D)]

To finish this section we present the main theorem who connects the dynamic and static formulas and shows the completness of dynamic rules.

THEOREM: The dynamic axioms DA1-DA6 and the side effects SE1-SE25 are necessary and sufficient to maintain a database in a consistent state, according to axioms A1-A20.

We let the proof for another publication /Sc4/ and present here an example to illustrate how the side-effects work.

There is the well known example of an information system about the organization if an IFIP Working Conference /OSV/. This Example was described with THM in unpublished notes by A. Horndasch and we take a little slice out of it. The corresponding (partial) data schema is in the figure below. We show the consequences of a single statement establishing a new relationship between two entities. First we define an additional user side-effect which, in fact can also be generalized to a schema side-effect:



USE1:
⊢ [establish(x,y,group.involvedin] => ($p \in x$ => [establish(p,y,involvedin)])
(all members of a Working-Group who organizes a Working Conference are involved in this conference)

The statements are written in THM/DML /Sc2/ but we hope that they are self explanatory enought. Given a new Working Conference 'wc' organized by the Working Group 'wg':

1. establish wc has-wg wg
   (original statement)
2. establish wg group.involvedin wc
   (by SE19 applied to 1.)
3. for each p elem wg
   establish p involvedin wc
   (by USE1 applied to 2.)
4. establish wc has-participant p
   (by SE19 applied to 3.)
5. insert p into WC_PERSON
   (by SE4 applied to 3.)
6. let wc be involvedin(p)
   let wps be part.W_CONF_PERSONS(wc)
   establish p elem wps
   (by SE12 applied to 5.)
7. insert p into WC_WG_MEMBER
   (by SE5 applied to 5.)
8. insert p into WC_PERSON_TO_INVITE
   (by SE5 applied to 5. or by SE1 applied 7.)

IV. CONCLUSION

According to a three level architecture we intend to define a mapping of a THM conceptual schema to an internal relational schema /Sc3, Sc4/. To analyse the correspondence of the two schemata a formalization in mandatory. If classes and relationships determine relations in the internal schema, operations gives transactions and the first idea was to generate triggers from the side-effects. But, since it is not an easy task to implement triggers, assertions and dependencies for relational databases and there are crucial design problems, we have chosen another way. The side-effects of THM/DML operations at the conceptual level are added to the operations as additional statements or suboperations, such that for the transformation only consistent operations are mapped to transactions. Only if we want to allow a direct access for an user to the internal schema the consistence conditions of the conceptual schema must be expressed in relational semantics. Actually we are analysing correspondeces between grouping and multivalued dependencies /Fag/ and between generalization and inclusion dependencies /CFP/.

THM is part of a project called PROSEM, intended to define the complete process of database design and use within the three-level architecture. Thanks to Prof. E.J. Neuhold and the members of the PROSEM group, Angelika Horndasch, Inge Walter and Ramin Yasdi for valuable discussions about the data model. Special thanks also to Udo Pletat and Rudi Studer for a critical reading of a draft of this paper.

R E F E R E N C E S

/Al/ J.F. Allen. "An interval-based representation of temporal knowledge". Proc. Int. Joint Conference on Artificial Intelligence, Vancouver, 1981.

/An/ T.L. Anderson, "Modeling tima at the conceptual level" in Improving Database Usability and Responsivness, P. Scheuermann (ed.), North Holland, 1982.

/AS/ "The ANSI/X3/SPARK DBMS framework. Report of the Study group on Database Management Systems", D. Tsichritzis and A. Klug (eds.) Inf. Systems, Vol.3 pp.173-191, 1978.

/BS/ F. Bancilhon, N. Spyratos. "Data Base Mappings, Part I: Theory", in notes of the Advanced Seminar on Theoretical Issues in Data Bases (TIDB), Cetraro, Italy, 1981.

/BN/ H. Biller, E.J. Neuhold, "Semantics of Data Bases: The Semantics of Data Models", Inf. Systems Vol.3 , 1978

/BM/ A. Borgida and J. Mylopoulos, "Semantic Models in Databases: some formal aspects". In notes of the Advanced Seminar on Theoretical Issues in Databases (TIDB), Cetraro, Italy, 1981.

/BW/ A. Borgida and H. K. T. Wong, "Data models and data manipulation langages: complementary semantics and proof theory", Proc. Very Large Data Bases, 260-271, 1981.

/BFM/ B. Breutmann, E. Falkenberg, R. Maurer, "CSL: A Language for Defining Conceptual Schemas", in Data Base Architecture, Bracchi and Nijssen (eds.), North Holland, 1979

/Br/ M. Brodie, "Axiomatic definitions for data model semantics". Inform. Systems, Vol.7 No.2, pp. 183-197, 1982.

/CFP/ M.A. Casanova, R. Fagin and C.H. Papadimitriou, "Inclusion Dependencies and their Interaction with Functional Dependencies", IBM Res. Report RJ3380, 1982.

/Fag/ R. Fagin. "Multivalued Dependencies and a New Normal Form for Relational Databases". ACM TODS 2,3 1977, 262-278.

/Fa/ E. Falkenberg. "Concepts for Modelling Information". In Modelling in Data Base Management Systems. G. M. Nijssen (ed.) North Holland, 1976

/GH/ J.V. Guttag and J.J. Horning, "The Algebraic Specification of Abstract Data Types", Acta Informatica, Vol. 10, 27-52, 1978.

/Ha/ D. Harel. "First-order dynamic logic". In Lecture Notes in Computer Science, vol. 68, Springer-Verlag, 1979.

/ISO1/ ISO TC97/SC5/WG3 "Concepts and terminology for the Conceptual Schema, 1981.

/ISO2/ ISO TC97/SC5/WG3 "Concepts and terminology for the Conceptual Schema and the Information Base", J.J. Griethuysen (ed.), 1982.

/LMP/ H. Laine, O. Maanavilja and E. Peltola, "Grammatical data base model", Inform. Systems Vol.4 pp. 257-267, 1979.

/MP/ Z. Manna and A. Pnueli, "Verification of concurrent programs: the temporal framework", in The Correctness Problem in Computer Science, S. Boyer and J. S. Moore (eds.). Academic Press, 1981.

/MW/ J. Mylopoulos and H.K.T. Wong, "Some features of the TAXIS data model", Proc. 6th. VLDB, Montreal, 1980.

/OSV/ T.W. Olle, H.G. Sol and A.A. Verrijn-Stuart (eds.), "Information Systems Design methodologies: a comparative review", North-Holland, 1982.

/RU/ Rescher and Urquhart. "Temporal Logic". Library of Exact Philosophy, Springer Verlag (1971).

/SNF/ C.S. dos Santos, E.J. Neuhold and A.L. Furtado, "A Data Type Approach to the Entity-Relationship Model", Proc. Entity-Relationship Approach to Systems Analysis and Design, 1980.

/Sc1/ U. Schiel, "Data Structures as Categories", working paper, Inst. für Informatik, U. Stuttgart, 1981.

/Sc2/ U. Schiel, "The Temporal-Hierarchic Data Model (THM)" Bericht 10/82 Institut für Informatik, Univ. Stuttgart, 1982.

/Sc3/ U. Schiel, "A semantic database model and its mapping to an internal relational model" in Databases: Role and Structure" P. Stocker, M. Atkinson and P. Gray (eds.), Cambridge University Press, 1983.

/Sc4/ U. Schiel, "Ein semantisches Datenmodell und seine Abbildung auf die interne Ebene", Dissertation, Institut für Informatik, Univ. Stuttgart (to appear).

/SFNC/ U. Schiel, A.L. Furtado, E.J. Neuhold and M.A. Casanova, "Towards multilevel and modular conceptual schema specifications", Inform. Systems, Vol.9, No.1, 1984.

/Su/ B. Sundgreen, "Conceptual foundation of the Infological Approach to data bases", in Data Base Management, J.W. Klimbie and K.L. Koffeman (eds), North Holland, 1974

/WFW/ G. Wiederhold, J.F. Fries and S. Woyl, "Structured organization of clinical data bases", Proceedings AFIPS, 1975.