TRANSACTIONS MODELING

C. ROLLAND, C. RICHARD
Centre de Calcul
UNIVERSITE DE PARIS I
Panthéon - Sorbonne

ABSTRACT

We are concerned by data processing in both centralized and distributed environments. We claim that it is profitable and efficient to make the same effort to the design of the transactions system as we made to the design of the data system. To define such a system, we present in this paper two levels of transactions modeling. We discuss the impact of the proposed modeling upon concurrency and parallelism management.

I - INTRODUCTION

We are concerned by data processing in both centralized and distributed environment. The DBMS or DDBMS that work out this function through the control of the transactions execution ignore what are really the transactions (i.e. their contend and their impact upon the data base) and their interrelations. They discover the transactions in proportion as the execution demands submitted by the users. Consequently, the concurrency and consistency problems are difficult to solve.

Conversely, to manage the data, they dispose of both their definition and the knowledge of their structure. Powerful modeling tools allow the DBMS to take into account the structure of the data they have to manage.

We claim that it is profitable and efficient to make the same effort to the design of the transactions system as we made to the design of data system. Consequently, the DBMS must be provided with the description of the transactional system it has to manage.

We present in this paper two levels of transactions modeling to define a transactions system. We show the impact of the modeling upon the transactions management procedures and namely the management of concurrency and parallelism.

II - TRANSACTIONAL MODELS

As for the data modeling (1) we propose two levels of transactions modeling :
- a conceptual level that allows an abstract representation of the semantics of a transactions system.
- a logical level that integrates the parameters related to both the planned use of the transactions system and the technical environment in which the system will operate.

II.1. Conceptual transactions modeling

II.11. The semantic significance of a transaction

The purpose of a transaction is to make (8) the data base evolve from a consistent state to another consistent state. It appears as the behavioural unit of the data base. A transaction generates a new state of the data base which is consonant with the evolution of the real system state. By this way, the data base and the reality progress in phase. The real system reacts to every event in generating some actions issuing in a new state.

Similarly, a transaction makes the data system evolve to a new state representative of the real system state. To avoid any discordance between the data base and the reality, the transactions must be the transposition of the actions executed by the real system.
So, a transaction translates to the data the rules that the real system applies to the real entities :

- the data are the representation of the real entities and

- the transactions are the representation of the real system behaviour.

Then, to build up a conceptual view of a transactions system independently of any technical parameter is to model the real system behaviour.
That is the way we have retained : we define a conceptual transactions schema as the conceptual modeling of the real world behaviour. It is the whole set of operations, events and their interrelations representing all the real world transformations and their application rules.

II.12. A conceptual behaviour modeling

i) Model

We have yet developed a model to represent in a formal way the real systems behaviour (15)(17). It is grounded upon three concepts defined in a relational normalized form that we name c-object, c-operation and c-event.

- The C-OBJECT concept represents a time consistent aspect of a real world objects class or an association of objects class. It is the representation of the largest set of properties of a real world objects class (or association class) having an identical dynamic behaviour (properties created, modified or deleted at the same times). It is a decomposition of a third normal form relation (3) into several relations of which properties identically evolve in the course of time.
By this way, we make the relational formalism able to take time into account. To summarize, we will say that a c-object represents an elementary state of the data base. For example, the 3 NF relation CUSTOMER (NCUST, CUSTNAME, CUSTADDRESS) is decomposed into the two c-object relations

(1) CUSTOMER (NCUST, CUSTNAME)
(2) CUSTOMER-ADDRESS (NCUST, ADRDATE, CUSTADDRESS)

because the name of the customer is permanent and his address can be modified in the course of time.

- The C-OPERATION concept represents a real world operations class. It is defined by reference to the c-object concept : it is the representation of an elementary transformation that can be applied to a unique c-object. The occurrences of a c-operation represent the operations that modify in a unique way the state of objects belonging to a unique c-object. To summarize, we will say that a c-operation represents an elementary action working on the data base. As a c-object, a c-operation represents a temporal aspect of a real system operations class and several c-operations represent the complete real operations class. For example, the real operations class "availability updating" of the example of reservation system considered in the next paragraph is represented by the two c-operation relations :

(1) AVAILABILITY-UPDATING-EXECUTION (NAVDEC, EXECDATE, NUMAVROOM)
(2) AVAILABILITY-UPDATING-PERMANENT (NAVDEC, TYPECRE, OPTEXT).

because there are many executions of "availability updating" operations and one and only one updating rule and modification type of objects belonging to the c-object "availability" (NUMAVROOM) for all the c-operation life (2).

- The C-EVENT concept represents a real world events class. It is defined by reference to the c-object and the c-operation concepts. It is the representation of an elementary noteworthy state change that could affect a c-object and that triggers one or several c-operations. The occurrences of a c-event are events that ascertain a determined state change type of objects corresponding to a unique c-object and that trigger operations corresponding to one or several c-operations. This triggering can be conditional and/or iterative.
To summarize, we will say that a c-event represents an elementary noteworthy state change of the real system that triggers one or several c-operations.

A c-event represents a temporal aspect of a real events class and several c-events represent the complete real events class. For example, the real events class "room modification arrival" of the next example is represented by the three c-event relations :

(1) ROOM-MODIFICATION-ARRIVAL (ROOMAR, ARRDATE, NROOM)

(2) ROOM-MODIFICATION-PERMANENT (ROOMAR, PRED, TYPECRE)

(3) TRIGGER-AVAILABILITY-UPDATING-EVENT (ROOMAR, NAVDEC, DATTRIG)

Relation (1) describes the arrivals of the event type "room modification arrival". Relation (2) describes the state change that defines the event. Relation (3) describes the triggering of the operations belonging to the type "availability updating" associated to the events "room modification arrival".

ii) Conceptual modeling

- Then, the conceptual modeling of a real system issues in a collection of relations belonging to the three previous types. It can be split up in two sub-collections :

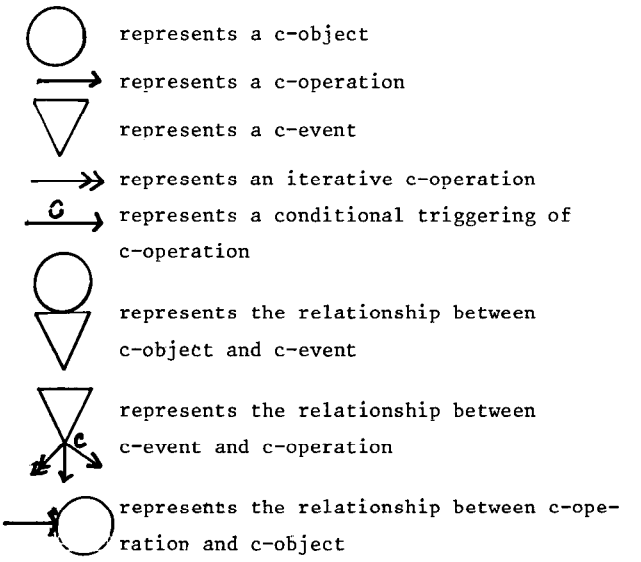
. the static data schema corresponding to the conceptual data structure (the collection of the c-objects relations) ;

. the behavioural schema corresponding to the conceptual modeling of the real system behaviour. It is the collection of the c-operations and c-events relations. It can be represented by a 3-alternate graph of c-objects, c-events and c-operations that expresses the behaviour of the real system in a causal way : the state change of a c-object induced by a c-operation can be a c-event that triggers one or several c-operations issuing in state changes of c-objects that can be events...

Moreover, we have demonstrated (17) that the normalization of the concepts issues in a conceptual modeling which is minimal (in the sense of minimal cover of a set of relations (4)) : it is the smallest set of objects, events and operations necessary and sufficient to describe the real system behaviour.

- Example of conceptual modeling

This example concerns a reservation system of hotel rooms in ski resorts. Figure 1 gives the collection of the DB c-objects and a graphic representation of the corresponding behavioural modeling based on the following conventions :



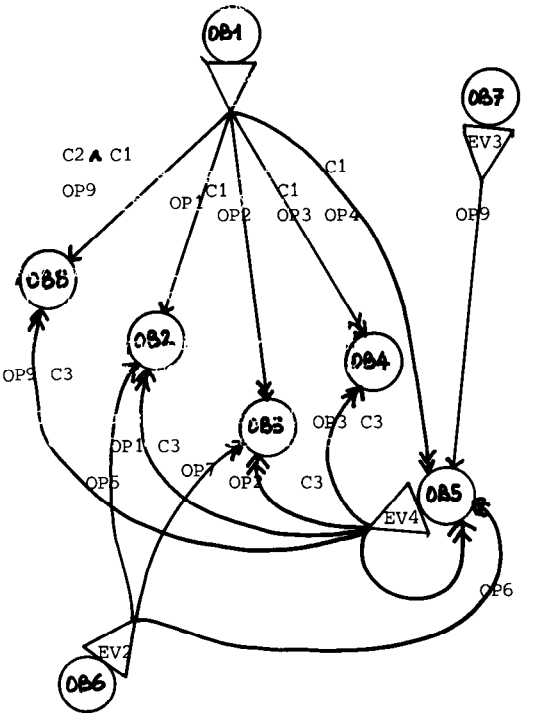


Figure 1 : An example of behavioural modeling

C-OBJECTS

OB1 : Demand
OB2 : Reservation state
OB3 : Reserved room
OB4 : Reservation
OB5 : Availability
OB6 : Cancellation demand
OB7 : Room
OB8 : Customer

C-OPERATIONS

OP1 : Reservation state creation
OP2 : Reserved room creation
OP3 : Reservation creation
OP4 : Availability decreasing
OP5 : Reservation state updating
OP6 : Availability increasing
OP7 : Reserved room suppression
OP8 : Availability updating
OP9 : Customer creation

C-EVENTS

EV1 : Demand arrival
EV2 : Cancellation demand arrival
EV3 : Room modification arrival
EV4 : Availability increasing arrival

CONDITIONS

C1 : The reservation demand can be satisfied
C2 : The customer does not exist in the DB
C3 : The availability increasing allows to satisfy a postponed reservation demand

Comments :

The EV1 type event represents the arrival of a demand. It corresponds to the creation of a new occurrence of OB1 in the DB. EV1 triggers alternative operations. Either the demand refusal (OP1 setting the demand in a "refused "state"") if the condition C1 is false, or the booking corresponding to the creation of one occurrence of OB4 by OP3, the creation of several occurrences of OB3 by OP2 and OB5 by OP4 and the creation of one occurrence of OB8 by OP9 (condition C2 true) if the condition C1 is true.

The EV2 type event represents the arrival of a cancellation demand. It corresponds to the creation of a new occurrence of OB6 in the DB. EV2 triggers operations that lead to the cancellation of a booking (suppression of the corresponding reserved rooms (OB3) by OP7, updating of the availability OB5 by OP6 and setting of the reservation in a "cancelled" state by OP5).

. The EV3 type event represents the arrival of a room modification. It corresponds to a change in the amount of OB7 occurrences. EV3 triggers the operation of availability updating (OB5 modification by OP8).

. Finally, the EV4 type event represents the arrival of an availability increasing. It corresponds to an increasing of the amount of OB8 occurrences. The arrival of an EV4 type event leads to try to process a postponed reservation demand (conditional triggering of OP1, OP2, OP3, OP4 (condition C3) and OP9 (condition C3$\wedge$ C2)).

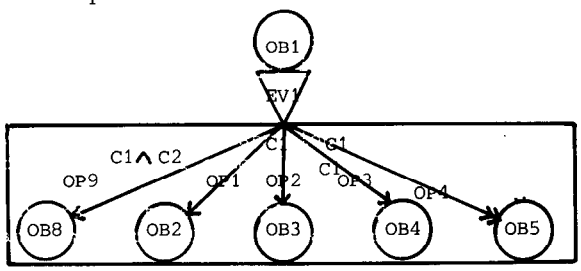
II.13. The conceptual transactions schema

The behavioural modeling is an abstract view of the transactions system. It defines the whole set of both the elementary transactions and their interrelations. To synthetize the transactional structure it is useful to introduce the concept of elementary transaction. The conceptual transactions schema gives an interpretation of the behavioural schema.

i) The elementary transaction notion

Definition : An elementary transaction is the undecomposable set involving all the c-operations triggered by a c-event.

Example :
If we refer to the previous example, the transaction T_1 "demand analysis" is defined as :



Transaction T1 : "demand analysis"

In the proposed modeling, all the c-operations associated to a given c-event are undissociable. They correspond to the set of elementary actions consequential to an event. They define a transition from a state to another one. This transition is consistent if all the c-operations are executed in a consonant way with their conceptual description. The elementarity of the transaction issues from the elementary of the c-object, c-event and

c-operation concepts.

ii) The set of elementary transactions

Then, the conceptual behavioural schema defines the minimal set of elementary transactions necessary to make the data base evolve in the course of time. They are involved in the conceptual behavioural modeling, as the sets of c-operations associated to each c-event of the modeling. There are four elementary transactions in the previous example : T1 (OP1, OP2, OP3, OP4), T2 (OP5, OP6, OP7), T3 (OP8), T4 (OP1, OP2, OP3, OP4, OP9).

Moreover, the conceptual behavioural schema summarizes the sequencing and synchronization of the transactions. This sequencing can be synthetized through a graph named transactions sequencing graph which is equivalent to the chronology graph of the triggering events of the transactions.

To define these two graphs, we introduce the concept of chronological dependency (14).

- The chronological dependency concept

1) Systematic Chronological Dependency (SCD)

A c-event EV_j is in SCD with a c-event EV_j if and only if

i) EV_i triggers unconditionally a c-operation OP_i^k that induces the state change of a c-object OB_j ascertained by EV_j

ii) This state change always defines an event belonging to the EV_j type (EV_j always follows EV_i). In the previous example EV4 is in SCD with EV2 (EV2 $\xrightarrow{s}$ EV4) because EV2 triggers unconditionally the c-operation OP6 that always induces a state change of OB5 corresponding to an availability increasing and this state change always defines an event belonging to EV4 type.

2) Conditional chronological dependency (CCD)

A c-event EV_j is in CCD with a c-event EV_i if and only if

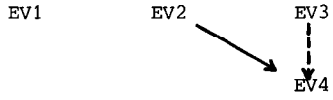
i) EV_i triggers a c-operation OP_i^k that induces the state change of a c-object OB_j ascertained by EV_j.

ii) Either EV_i triggers conditionally OP_i^k, or the state change induced by OP_j^k does not always define an event belonging to the EV_j type (EV_j may follow EV_i).

In the previous example, EV4 is in CCD with EV3 (EV3 $\xrightarrow{c}$ EV4) because EV3 triggers unconditionally the c-operation OP8 (availability updating) but the induced state change does not always define an event belonging to EV4 type (availability increasing).

- The conceptual chronology graph of events (CCGE)

It is the graph that involves all the chronological dependencies (SCD and CCD) of the conceptual behavioural modeling. It synthetizes the possible sequences of events that could happen in the real system. The chronology graph of the events of the reservation system example is represented on figure 2.



——→ refers the SCD relation

- - - - - - - - - - → refers the CCD relation

Figure 2 : Conceptual chronology graph of events.

It points out the independence of EV1 in regard with the other c-events of the system, the systematic dependence of EV4 with EV2 and its conditional dependence with EV3.

- The conceptual transactions sequencing graph (CTSG)

It simply derives from the chronology graph of events in replacing every c-event by the associated elementary transaction. It expresses the dependency links between the transactions that are necessary to maintain a consistent behaviour of the data base (18). Then, the transactions sequencing graph of the example is represented on figure 3.

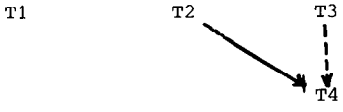


Figure 3 : Conceptual transactions sequencing graph

It points out the independence of the elementary transaction T_1 in regard with the other transactions of the system , the systematic subordination of T_4 to T_2 and its conditional subordination to T_3.

These two graphs are equivalent. They express the allowable sequences of data base updating actions either in regard with the transactions (CTSG) or in regard with the events that trigger the transactions (CCGE).

II.2. - The logical definition of the transactions

The logical view of the transactions aims :

- to meet as well as possible the users requirements
- to define an efficient solution integrating the technical constraints.

The logical expression of the transactions issues from the conceptual expression through derivation rules developed in this chapter.

II.21. The concept of logical transaction

A logical transaction is the gathering of several elementary transactions that were connected by subordination links at the conceptual level. This gathering aims to meet the users requirements and the technological constraints introduced at this level. It is the result of the successive applications of the derivation rules.

II.22. Derivation of the logical transactions

Derivation rule 1 : Transactions gathering

(1) $EV_i \xrightarrow{s} EV_j \Rightarrow T_i . T_j$

(2) $EV_i \xrightarrow{c} EV_j \Rightarrow$ choice $T_i * T_j$ or T_i, T_j separate

(1) If a c-event EV_j is in SCD with a c-event EV_i, EV_j always follows EV_i. Then the transaction T_j associated to EV_j is always triggered after the execution of the one associated to EV_i. In this case, to avoid an unnecessary test of EV_j arrival, we propose to systematically concatene these two transactions (denoted $T_i . T_j$) except if an high parallelism ratio is required (see paragraph III.2).

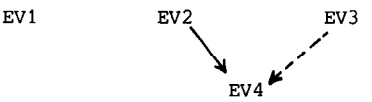
(2) If EV_j is in CCD with EV_i, EV_j may follow EV_i. Then the transaction T_j associated to EV_j can be triggered after the execution of the one associated to EV_i if it generates an occurrence of EV_j. In this case, either the two transactions are concatened through the insertion of EV_j arrival test (denoted $T_i * T_j$) or they are kept separate. This alternative be solved according to the adequacy of the two solutions in respect with the operating environment.

In both cases((1) and (2)) the triggering c-event of a logical transaction is therefore the c-event that triggers the first elementary transaction of the concatenation.

Remark : When a c-event EV_k is dependent of several others, the application of the transactions gathering rule can issue in the duplication of the transaction T_k associated to EV_k. The number of T_k duplications is equal to the number of c-events source of dependency with EV_k.

Conversely, when EV_k is a source of several dependencies, the associated transaction must not be duplicated (in this case, it would be executed several times instead of once). But we have either to gather all the transactions to build one and only one logical transaction or to keep them separate.

For example, as the conceptual chronology graph of events corresponding to the reservation system is :



the application of the derivation rule 1 leads :

(1) To concatene the transactions T2 and T4 (denoted T2.T4) because the triggering c-event EV4 of T4 is in SCD with the triggering c-event EV2 of T2 (part (1) of this rule).

(2) and eventually to concatene the transactions T3 and T4 by inserting the test of EV4 arrival (denoted T3 * T4) because the triggering c-event EV4 of T4 is in CCD with the triggering c-event EV3 of T3 (part (2) of the rule), issuing in the duplication of T4.

Derivation rule 2 : independent transactions definition

In an operating environment that involves very few synchronization mechanisms it could be in the designer's interest to define a transactions system constituted only by independent transactions. So, he can be sure that the triggering of the transactions does not issue in inconsistencies due to a bad respect of synchronization rules.

Then to build up a set of independent transactions is to gather in a same logical transaction all the elementary transactions triggered by the c-events belonging to a same connected

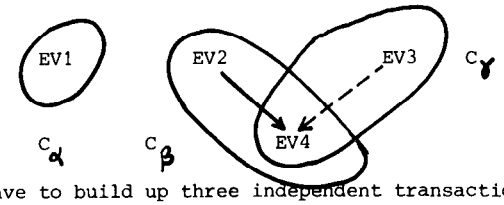
component of the conceptual chrononoly graph of events.

A connected component C_x of the chronology graph is defined as a part of the graph in which all the c-events are connected by a sequence of SCD or CCD relations :

$$(C_x \in \mathcal{P}(\mathcal{E})) \ (\forall (EV_i, EV_j) \in C_x^2) \ (i \neq j) \Rightarrow$$
$$((EV_i \xrightarrow{+} EV_j) \vee (EV_j \xrightarrow{+} EV_i))$$

In which $\mathcal{E}$ refers the c-events set and $\xrightarrow{+}$ refers the strict transitive closure of the direct chronological dependency relation defined as the union of the SCD and the CCD relations.

For example, as the conceptual chronology graph of events involves three connected components,



we have to build up three independent transactions:

- T_α corresponding to the connected component C_α that involves only the elementary transaction T_1 associated to EV1 (single element of C_α).

- T_β corresponding to the connected component C_β . T_β is the concatenation of the two elementary transactions T_2 and T_4 (T_2.T4) associated to the two elements EV2 and EV4 of C_β which are in SCD.

- T_γ corresponding to the connected component C_γ . T_γ is the concatenation of the two elementary transactions T_3 and T_4 (T_3 * T4) associated to the two elements EV3 and EV4 of C_γ which are in CCD.

Remark : Generally, the intersection of two connected components is not an empty set. Consequently, the definition of independent transactions often issues in the duplication of a great number of elementary transactions.

II.3. Conclusions about the proposed transactions modeling

We point out three main conclusions :

1) The transaction concept has been semantically identified and formally defined. It is compatible with the current acceptation of the transaction notion Our definition of a transaction meets the three axioms proposed by J. GRAY (8) :

- a transaction carries out a consistent transition of the data base,
- it is atomic : either its triggering c-event is ascertained and consequently all the involved c-operations are triggered or it is not ascertained and no operation is triggered ;
- it is durable : as the c-operations of a transaction are executed nothing could alterate their effects upon the data, except the execution of a compensating transaction.

But it completes and precises the current definitions. It allows both to identify and to specify at the very time of the design process all the transactions necessary to make the data base evolve in a consistent way.

2) The conceptual transactions schema (CTS) has been identified as the conceptual view of the transactions synchronization structure. The CTS is the necessary complement of the data schema : it defines the whole set of synchronization links of the transactions associated to a data schema. The normalization of this schema ensures both that the conceptual transactional structure is redundancy-free (the transactions are not redundant, i.e there are no repeated actions) and that the number of interrelations among the transactions is minimal (it is the smallest set of links necessary to completely describe the real behaviour of the system). The CTS represents an "invariant" in the real system behaviour. As the conceptual data system describes the stable set of data to manage to meet all the users needs, the CTS describes the stable set of transactions to manage to make the data base evolve in a consonant way with the real system behaviour. It permits to derive a logical transactions schema that defines a transactional system well-adapted to a particular operating environment.

3) The chronology graph of triggering events and the transactions sequencing graph are equivalent. Therefore, it is equivalent to reason on one graph or the other for every problem related to the synchronization or the parallel processing of a set of transactions.

III - MODELING IMPACT ON THE MANAGEMENT TOOLS

The technical means are not independent of the models they manage. In our point of view, the tran-

sactions system modeling affects the choice of the solutions the DBMS needs to manage the transactions. Two main impacts of the modeling are

1) The DBMS architecture,

2) The parallelism management.

III.1 DBMS architecture

According to the proposed reasoning, the DBMS must involve a Transactions Management Monitor (TMM). The TMM has not only to manage and control the transactions executions, but also to trigger them in response to the real system events. Then, the DBMS architecture can be summarized on figure 4 :

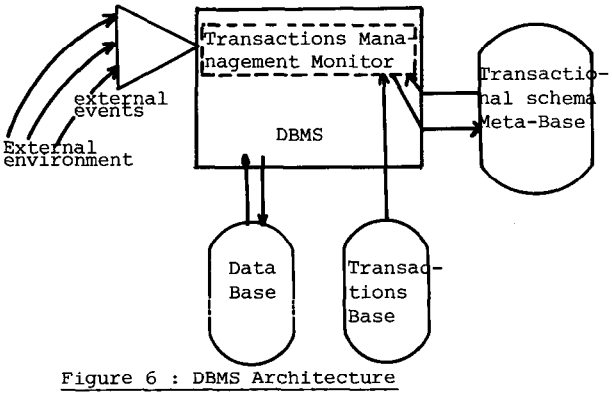


Figure 6 : DBMS Architecture

The TMM uses the transactions base programmed from the transactions schema and the meta-base of transactions schema implementation. It operates on the data base through the DBMS other modules. Finally, it communicates with the external environment that informs it about the external events.

The TMM is listening in the real system. It analyzes the signals it receives and carries out four primitives :

1) to recognize if a real system state change is an event:comparison with the transactional system events description stored in the meta-base ;

2) to determine which is the transaction associated to this event. This information is provided by the meta-base ;

3) to select it in the transactions base and to trigger and control its execution ;

4) to analyze the consequences of the transaction execution using the meta-base defining all the state change types induced by the transaction and to return to the first primitive to determine if these state changes are events.

The TMM recognizes and manages the external events. But it automatically carries out both the recognizing of the internal events and the synchronization of the corresponding transactions. In an environment without parallelism, this synchronization results from the properties of the transactional schema (14) and the two derived isomorphic graphs: the events chronology graph and the transactions sequencing graph.

The TMM triggers the transactions in the order of their triggering events arrivals. It works according to a synchronization schema defined during the design process which is applied by the TMM all over the data base operating. We have built a prototype of such a tool connected with a relational DBMS (11) running on a CII-HB IRIS 80 computer at the university of Nancy.

III.2. About parallelism management

If the technical environment allows parallel triggerings of the transactions, the DBMS (or D-DBMS) has to work out a consistent functioning of the system, despite of the parallel executions of the transactions. We show how the proposed modeling allows to determine the transactions that can be executed in a parallel way without any special precaution and those that requires guarding mechanisms to be parallely triggered.

III.21. Parallelism determination

To determine the logical transactions that can be parallely triggered is to determine the triggering events that can be parallety processed, because of the equivalence between the transactions sequencing graph and the chronology graph of events.

i) Parallel triggering of two transactions

Let $EU(EV_i)$ be the set of the c-objects used by the c-operations triggered by EV_i (mainly to the evaluation of the triggering conditions). Likewise, let $EM(EV_i)$ be the set of the c-objects modified by the execution of those c-operations.

Then, a c-event EV_j is in execution dependency (ED) with a c-event EV_i (denoted $EV_i \xrightarrow{E} EV_j$) if and only if the c-operations triggered by EV_j use c-objects modified by the c-operations triggered by EV_i :

$$(EV_i \xrightarrow{E} EV_j) \Leftrightarrow (EU(EV_j) \cap EM(EV_i) \neg = \emptyset)$$

Therefore, the transaction triggered by EV_j could be triggered after the execution of the transaction triggered by EV_i, but these two transactions must never be executed in a parallel way.
Under these hypothesis, we name context of a c-event EV_j (denoted $\mathcal{C}(EV_j)$) the set of c-events with which EV_j is in execution dependency :

$$\mathcal{C}(EV_j) := \{ EV_i \in \mathcal{E}_T \ / \ EV_i \xrightarrow{E} EV_j \}$$

Then, two c-events EV_i and EV_j (and consequently the associated transactions) can be processed in a parallel way if and only if none is in execution dependency with the other one, i.e. if and only if none belongs to the context of the other :

$$CP(EV_i,EV_j) := \neg((EV_i \xrightarrow{E} EV_j) \ V \ (EV_j \xrightarrow{E} EV_i)$$

$$CP(EV_i,EV_j) := (\{EV_i,EV_j\} \notin \mathcal{C}(EV_i) \ U \ \mathcal{C}(EV_j))$$

ii) Parallelism determination over the whole set of transactions

We have now to apply this condition to every pair of c-events that trigger logical transactions to determine the sets of transactions that can be executed in a parallel way :

Let C_i be the column vector that represents EV_i's context :
$C_i = (C_i^j)_{1 \leq j \leq |\mathcal{E}_T|}$ with $C_i^j = \begin{cases} 1 \text{ iff } EV_j \in \mathcal{C}(EV_i). \\ 0 \text{ else} \end{cases}$
Likewise, let V_i be the column vector equal to zéro, except at range i (it represents "EV_i's position").

Let + be the composition law defining the boolean union of two column vectors and let. be the composition law defining their boolean intersection. Finally, let $tr(X_i)$ be the trace of a column vector, i.e the boolean union of its elements.

Then, we can define the parallelism matrix P over the logical transactions sequencing graph in applying the former parallelism condition as

$$P = (pij)_{\substack{1 \leq i \leq |\mathcal{E}_T| \\ 1 \leq j \leq |\mathcal{E}_T|}} \text{ with } pij = tr\ ({}^tV_i.C_j + {}^tV_j.C_i)$$

in which the column vector P_i represents all the transactions that can be executed in a parallel way with the transaction T_i associated to EV_i.

iii) Properties of the logical transactions parallelism

1) The transactions parallelism is defined a-priori once at a time. Indeed, as soon as the transactions sequencing graph and the chronology graph of events have been built, it is possible to determine the sets EU and EM for every c-event of the graph. Consequently, we can define the context of every c-event and therefore we can build up the parallelism matrix P as presented above.

2) In the case of elementary transactions, the parallelism is maximal. Indeed, as an elementary transaction is the smallest gathering of actions that keeps the system consistent, the sets of objects EU and EM are minimal. Consequently the probability to find empty intersections is the highest and the parallelism is maximal. That can justify to keep only elementary transactions at the logical level. Conversely, if the logical transactions sequencing graph involves only independent transactions, the parallelism ratio will be the lowest, as the transactions size is the greatest and the elementary transactions duplications number is the largest.

iv) Example

If we suppose that a high parallelism ratio is required, the logical chronology graph of the reservation system example must be the same as the conceptual one. Then the execution dependencies of this example are :

EV1 $\xrightarrow{E}$ EV1 EV1 $\xrightarrow{E}$ EV4
EV2 $\xrightarrow{E}$ EV1 EV2 $\xrightarrow{E}$ EV4
EV3 $\xrightarrow{E}$ EV1 EV3 $\xrightarrow{E}$ EV4
EV4 $\xrightarrow{E}$ EV1 EV4 $\xrightarrow{E}$ EV4

Consequently, $\mathcal{C}(EV1) := \mathcal{C}(EV4) :=$ EV1,EV2,EV3,EV4 and $\mathcal{C}(EV2) := \mathcal{C}(EV3) := \emptyset$ and the parallelism matrix is

$$P := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Every column vector P_i precises the transactions that can be executed in a parallel way with the transaction T_i so :

- T2 can be triggered in a parallel way with itself and T3
- T3 can be triggered in a parallel way with itself and T2.

III.22. The parallelism management mechanisms

As they a priori know what are the sets of transactions that can be activated in a parallel way, the parallelism management mechanisms are simple ones : as soon as an event is detected, they have to determine if the associated transaction can be triggered in a parallel way with the running transactions by use of the parallelism matrix. If it is possible, the transaction is triggered. Else, either the transaction waits until it can be triggered, or it must be submitted to guarding mechanisms such as monitors (9), semaphores (5), synchronization primitives (6), (10) or time-stamps (2), general locking algorithms (7) or circulating tickets (12) in a distributed data base (and distributed DBMS) environment.

IV - CONCLUSION

Starting from the purpose of transactional systems design, we have drawn out two main conclusions :

1) The transactions conceptual schema notion can be identified. It is isomorphic to that of real world behaviour conceptual schema. It allows both to define the minimal set of transactions necessary and sufficient to make a data base evolve in a consistent way in the course of time and to isolate the real system behaviour invariant as the complement of the real system entities invariant represented by the conceptual data schema.

2) When a DBMS knows the transactional schema of the data base all over its operating period it can manage the transactions in a different way leading to decide by itself of the triggering of some transactions. Likewise, in a parallel environment, such as in a distributed data base, the DDBMS architecture is modified and the concurrency management mechanisms are simplified. We have built up such a DDBMS architecture at the university of Paris VI-Jussieu.

V - BIBLIOGRAPHY

1 - ANSI/X3/SPARC, Study group on data base management systems, Interim Report, FDT bulletin of ACM SIGMOD 7, nb21 (1975).

2 - BERNSTEIN, P.A. and GOODMAN, N., Timestamp based algorithms for concurrency control in distributed data base systems, Proceedings of the 6th International conference on VLDB, Montreal (Sept 80).

3 - CODD, E.F., A relational model of data for large shared data banks, Communications of the ACM, Vol 13, nb 6 (1970).

4 - DELOBEL, C., Contribution théorique à la conception et à la définition d'un système d'information appliqué à la gestion, Thèse de Doctorat d'Etat, Grenoble (1973).

5 - DIJKSTRA, E.W., Cooperating sequential processes, in Programming Languages, F. Genuys editor, Academic Press (1967)

6 - DIJKSTRA, E.W., Guarded commands, non determinacy and formal derivation of programs, Communications of the ACM, Vol 18, nb 8 (1978).

7 - ELLIS, C.A., A robust algorithm for updating duplicate data bases, Proceedings of the 2nd Berkeley Workshop on distributed data management and computer networks (May 1977)

8 - GRAY, J., The transaction concept : virtues and limitations, Proceedings of the 7th international conference on VLDB, Cannes (Sept 81).

9 - HOARE, C.A.R., Monitors, an operating systems structuring concept, Communications of the ACM, Vol 17, nb 10 (1974).

10 - HOARE, C.A.R., Communicating sequential processes, Communications of the ACM, Vol 21, nb 8 (1978)

11 - LEIFERT, S., Un langage de spécification des systèmes d'information. Un outil pour leur gestion. Thèse de docteur ingénieur, Université de Nancy I (Mai 1980).

12 - LE LANN, G., Algorithm for distributed data sharing which use tickets, Proceedings of the 3rd Berkeley Workshop on distributed data management and computer networks, (Aug. 78).

13 - RICHARD, C., Une approche conceptuelle des problèmes de synchronisation, Thèse de 3ème cycle, Université de Nancy I (Oct. 1979).

14 - RICHARD, C. and ROLLAND, C., A distributed information system concurrency control method, Proceedings of the IFDO/IASSIST Congress, Grenoble (Sept 81).

15 - ROLLAND C. and FOUCAUT, O., Concepts for the design of an information system conceptual schema and its utilization in the Remora Project, Proceedings of the 4th international conference on VLDB, Berlin (1978).

16 - ROLLAND, C., LEIFERT, S and RICHARD, C., Tools for information system dynamics management, Proceedings of the 5th international conference on VLDB, Rio de Janeiro (Oct 1979).

17 - ROLLAND, C, LEIFERT, S. and RICHARD, C., A proposal for information system design and management, ACM SIGDA, Vol 20, nb 2 (1980).

18 - ROLLAND, C. and RICHARD, C., The Remora methodology for information systems design and management, Proceedings of the INFORSID Worshop, La Baule, (Oct 1981).