MODELLING APPROACH FOR DISTRIBUTED DATA BASES

Michel ADIBA
Laboratoire d'Informatique
Université de Grenoble
B.P. 53 / 38041 GRENOBLE Cedex (France)

ABSTRACT

This paper describes a general model for distributed data (MOGADOR) together with a language for describing and manipulating dispersed data.

In providing an homogeneous level for the description and the behaviour of distributed data bases, MOGADOR can be viewed also as a logical tool for designing heterogeneous distributed data bases management systems.

1. INTRODUCTION : DIFFERENT APPROACHES TO DISTRIBUTED DATA BASES MANAGEMENT SYSTEMS

The advent of computer networks and the increasing development of data base technology brought a great potential for sharing data among heterogeneous computing facilities.

This area of research is currently refered to as distributed data bases, one of the main problems being the design and the implementation of a distributed data base management system (DDBMS) [12][23].

In France, a national project ("SIRIUS") projet [24]) sponsored by IRIA, coordinates several research projets on this area such as the one described here which is in process at the Grenoble University.

There is a common agreement to recognize two kinds of DDBMS [11] :
1) Homogeneous or standardized DDBMS where the description and the manipulation of the distributed data base components are made by the same kind of DBMS which is implemented on each sites [26][12].

2) Heterogeneous or integrated DDBMS where these description and manipulation functions are assumed by heterogeneous DBMS such as I.M.S, I.D.S, SOCRATE, etc ... [25][14].

The second approach seems to be more realistic in the way that a great variety of DBMS are to day commercially available. Often in some big enterprises or administrations data processing has been made by sectorization then creating several data bases

This research is supported by IRIA SIRIUS Project (contract 77 076).

with heterogeneous implementations but having however semantic links. It is this common semantic which allows the gathering of different data bases in order to implement new applications.

It is not conceivable to come back to a centralized approach which goes against the natural entreprise structure, but rather to have a distributed data base approach in which each component data base keeps a part of autonomy, while being able to share data with other data bases.

Making the assumption that several data bases are currently in existence and that we want to use them without modification, leads to a cooperation approach where the distributed data base corresponds to the gathering of data stored in these existing data bases [25].

On the other hand, the implementation of a new data base with a distributed DBMS is rather a distributed approach which is easier because of the freedom we have to define the component data bases [14][12].

These two approaches are possible either with homogeneous or heterogeneous DDBMS.

The goals of the POLYPHEME project developed at the Grenoble University are the study and the design of an heterogeneous DDBMS in a cooperation approach [25].

The system architecture, a prototype of which is currently being implemented, stands upon a relational data model (MOGADOR General Model for distributed data [5]) which provides an homogeneous level for :

1) the description and the manipulation of the cooperating data bases and of the distributed data base.
2) the behaviour of the cooperating data bases in order to be able to share data. Each data base is considered as a standard abstract machine.

The goal of this paper is to present this particular data model MOGADOR.

At section 2, we define basic concepts of MOGADOR, i.e level and spaces, object, category, functions.

In section 3, we describe LDDM, i.e a language for distributed data description and manipulation based upon MOGADOR concepts.

With this language, it is possible to ensure, at the local level the homogeneisation of the cooperating data bases by describing them with local views. Through the global view concept this language is used also to describe and manipulate the distributed data base.

2. MOGADOR : BASIC CONCEPTS

A distributed data base is first a data base and we make the assumption that it is described by a kind of conceptual schema [7] and that the users access it through external schemas (Figure 2.1).

Let us ignore for the moment this distinction between conceptual and external schemas in order to define the nature of the model and of the corresponding tools (languages) we need, to implement new applications involving the cooperation of different and heterogeneous, data bases.

It is a well-known fact that relational models can be used to describe data which are structured in a hierarchical or network way [18][19].

In a previous paper [4], using Abrial's Data Semantics [1] formalism we give a first methodological approach for distributed data bases.

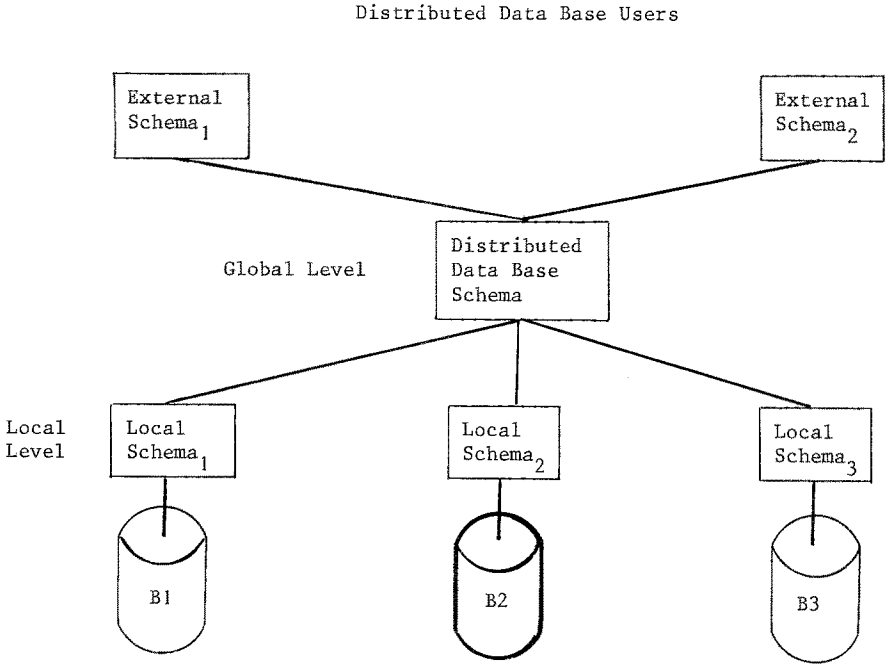


Fig. 2.1 - Local Data Bases and Distributed Data Base

In going further in this direction, we define a general model for distributed data : (MOGADOR) which, in the framework of POLYPHEME project, allows us to implement tools for describing, retrieving, updating distributed data, tools which are available at different system levels by interfaces and languages.

Besides set theory, MOGADOR is based on three fundamental concepts :

1) element concept : objects and names

2) category concept : set of elements

3) function concept to express relationships between categories.

These concepts are used at three levels :

1) At the level of the definition of MOGADOR itself with pre-defined categories and functions.

2) At the local level of the cooperating data bases to describe the behaviour of these data bases by making homogeneous the data description (local names and objects) and the operations they can execute.

3) At the global level which concerns the local data bases cooperation, in order to describe the distributed data base schema and its manipulation (global names). Particularly we have to define the mapping between this level and the local ones for the following two types of operations (global rules) :

- access to the distributed data base : how to process local objects to transform them into global ones ?
- creation and updating at the global level : what are the repercussions of these operations at the local data base levels ?

2.1. Elements : Object Space, Name Space

Elements in MOGADOR are divided into two spaces, namely object spaces and name spaces.

2.1.1. Objects Space

We define four types of objects.

2.1.1.1. Simple object

They correspond to an elementary value belonging to one of the following sets (predefined categories) :
INTEGER (set of integers), REAL (set of real numbers), LOGICAL (true, false), STRING (set of character strings).

2.1.1.2. Compound object

It is a tuple of simple objects.
For example <F56, NEW-YORK, 525, 10000>.

2.1.1.3. Program

A "program object" corresponds to the set of instructions executable by a given machine.

2.1.1.4. Process

Execution of a given program by a given machine.

2.1.2. Names Space

It concerns the description of an object space as we are going to see in section 3.

A name is a character string. It is used to give names to categories and functions.

By convention, we use upper case letters for categories and lower case letters for functions and we make the distinction between functions which send back one element (monovalued functions) and those which send back a set of elements (multivalued functions [2]).

We consider also that there exists a special name space constituted by predefined categories and functions (3.1).

2.2. Categories

2.2.1. Definition of Categories

In MOGADOR we suppose the existence of predefined categories like INTEGER, REAL, STRING, LOGICAL but also those which correspond to the description of data bases. We shall find in section 3.4.1. a table giving these main predefined categories.
It is possible to define a category using already defined ones.
For this, we use the following operators

1) Assignment ":="

A := B define category A as the set B.

2) Cartesian Product "×"

$A \times B = \{(a,b) \mid a \in A,\ b \in B\}$.

3) Restriction "(predicate)" after a category is used to define a subset of this category :

A (predicate) = $\{a \in A \mid$ predicate (a) = true$\}$.

Examples

i) AGE := INTEGER (18..65) define AGE as a set of integers which are between 18 and 65.

ii) DAY := INTEGER (1..31)
MONTH := INTEGER (1..12)
YEAR := INTEGER (0..99)
DATE := DAY × MONTH × YEAR.

iii) LCC := STRING (length ≤ 8) defines LCC (Local Concrete Category) as a set of strings (names set).

2.2.2. Operations on categories

Four basic operations are defined on categories :

1) Creation of an element of a category.
2) Deletion of an element.
3) Test of the existence of an element in a category.
4) Enumeration of all the elements of a category.

For each space we define operators to realize these operations :

- for object space we have manipulation operators and
- for name space description operators.

These operators will be used in the language for describing and manipulating distributed data (LDDM), see section 3.2.

2.2.3. Abstract Categories

They correspond to sets of simple or compound objects upon which we cannot apply creation and deletion operators. This means that abstract objects already exist in our universe and that we can use them directly.

For example AGE is an abstract category. The character string "AGE" is the name of the category and AGE is the name of a set of integers.

If AC is the name of the abstract categories set we have

$$AGE \in AC \quad \text{(Name)}$$

and for instance $26 \in AGE$ (Object).

This notion of abstract category can be viewed as the domain notion in the relational data model [18][19].

At the local level we consider local abstract categories (LAC) and at the global level, global abstract categories (GAC).

2.2.4. Concrete Categories

They correspond to sets of objects upon which the category operations are defined (section 2.2.2.).

This notion is analogous to the relation concept in Codd's relational model but as it was pointed out by J.M. Smith and D.C.M. Smith in [10], this notion supports two distinct forms of abstraction : aggregation, i.e materialization of a relationship into a set of objects, and generalization, where similar objects are regarded as a generic object.

In MOGADOR, to make explicit the difference between these two forms, we consider the function concept as it is described in section 2.3.

To define a concrete category we need at least two elements :

1) the name of the category, for example PERSON, RESERVATION

2) the cartesian product of abstract categories which can be used to identify the concrete object (key). We call it the identifier name of the concrete category.

For instance, if a set of persons are being identified by a social security number, we have :

- SSN := STRING (length = 13)
- Concrete Category PERSON identified by SSN.

If RESERVATION is a set of couples SSN and H# (Hotel number) we have Concrete Category RESERVATION identified by SSN × H#.
We consider local concrete categories (LCC) and global concrete categories (GCC) together with local identifier name (LIN) and global identifier name (GIN).

2.3. Functions

The function concept is a well known mathematical notion [13] which has been applied by Abrial in [1] to data models.

This concept presents a double aspect :

- static aspect, namely the existence of a named relationship f, for example, between two sets A and B
- a dynamic aspect, namely given one object $a \in A$ and a function f how the related object f(a) can be obtained. If function f is completely determined by the existence of its graph (i.e by the set of couples (a, f(a)), then from a given a, we can obtain f(a) by accessing objects in the graph.

The second possibility is to have the set of operations (the equation) to apply on a to obtain f(a).

Applying these mathematical notions to distributed data bases, provides a very flexible way for :

1) expressing the existing relationships between local objects
2) taking into account logical access paths between categories of objects
3) making a given data base execute some data access programs
4) expressing new relationships between distributed objects.

2.3.1. Names and objects

A function in MOGADOR is defined by the following elements :

- the name of the functions (written in lower case letters)
- the type of the function, namely if it is mono or multi-valued
- the source and target, i.e if f goes from A to B, A is the source of f and B its target. Note that A and B can be cartesian product, for instance : birthdate is a monovalued function from PERSON to DATE (DAY × MONTH × YEAR)
- if a relationship between A and B is completely determined by the graph of a function f, this mean that there exists, for example in a local data base, a

set of objects belonging to a concrete category C :

$$C = \{(a, f(a)) \mid \forall a \in A\}.$$

Note however that we are not concerned by the physical representation of concrete categories.

- To denote the inverse function of a function f we use the notation inv f
- Functions can be composed to form new functions
- There is an identify function, named "id".

2.3.2. Operations on functions

We define four basic operations on functions :

1) Access i.e given a to obtain f(a) which can be an element if f is monovalued or a set if f is multivalued.

   By extension if f applies to a set this means that it has to be applied successively to each element of the set :

$$X = \{x_1, x_2, \ldots, x_n\} \subseteq A$$
$$f(X) \equiv \{f(x_1), f(x_2), f(x_3), \ldots, f(x_n)\}.$$

2) Link a set of objects to a given object, for example :

$$f(a) := \{b\}$$
$$\text{or } g(x) := \{y_1, y_2, \ldots, y_n\}.$$

3) Erase the link between an object and its related objects

$$f(a) :\neq \{b\}.$$

4) Graph : to obtain the graph of a function $\{(a, f(a))\}$.

3. LDDM : A LANGUAGE FOR DISTRIBUTED DATA DESCRIPTION AND MANIPULATION

3.1. Predefined Categories and Functions

As we have said at the beginning of section 2 the basic concepts of MOGADOR are used at three levels, the first one concerning MOGADOR definition, the second and third ones concerning respectively local and global levels.

Predefined categories, functions and corresponding operations are basic elements of the LDDM language. This language is intended to provide an homogeneous way to describe and use both the components of the distributed data base and the distributed data base itself. Our purpose is not to provide a complete and new data base language like SEQUEL [21] or an equivalent language, but rather to define a minimum set of primitives for describing and manipulating dispersed data, primitives available in a high level host language like PL/1.

It is beyond the scope of this paper to give a complete list of all the predefined categories and functions [5].

The predefined functions are divided into descriptive functions to express relationships between names in a given space or between two different spaces (local and global), and manipulating functions which are in fact operators.

The following table T shows some of the main predefined categories and functions and explanations on its content are given in the following sections.

3.2. Operators

Operators in the LDDM are divided into description and manipulation operators.

3.2.1. Description Operators

They are used to describe both local and global views. These descriptions are given to the cooperation system which stores them in an internal format into local and global machines (Figure 3.1).

These operators apply to predefined categories and functions in order to define name spaces. To simplify the description these operators are combinations of elementary operations seen at section 2.2.2. and 2.3.2.

For example to create a name of a local concrete category (LCC) and to link it with its identifier name (LIN), we use two operators lcc and lin in the following manner :

lcc PERSON lin NAME × FIRST-NAME.

To define a global monovalued function (GOF), together with its source and target we write :

gof age from PERSON to AGE.

3.2.2. Manipulation operators (see table TOP)

They are used to manipulate local data bases through the local views and the distributed data base through the global view. Software systems which manipulate the distributed data are viewed as standard automata or abstract data base machines.

We assimilate the name of each local data base with the name of the machine which permits its utilization (see figure 3.1) and we say that the global machine (named "g") is the one accessed by distributed data base users.

Each machine is able to execute two kinds of operations :

- primitive operations on categories and functions according to the corresponding local or global view ;
- operations on objects or set of objects : these operations can be applied to the result of "enumerate", "access" and "graph" primitives. They are used to derive new sets of objects upon which other operations can be applied and so on.

Sets of compound objects are in facts n-ary relations so we find here a complete set of relational operations [18].

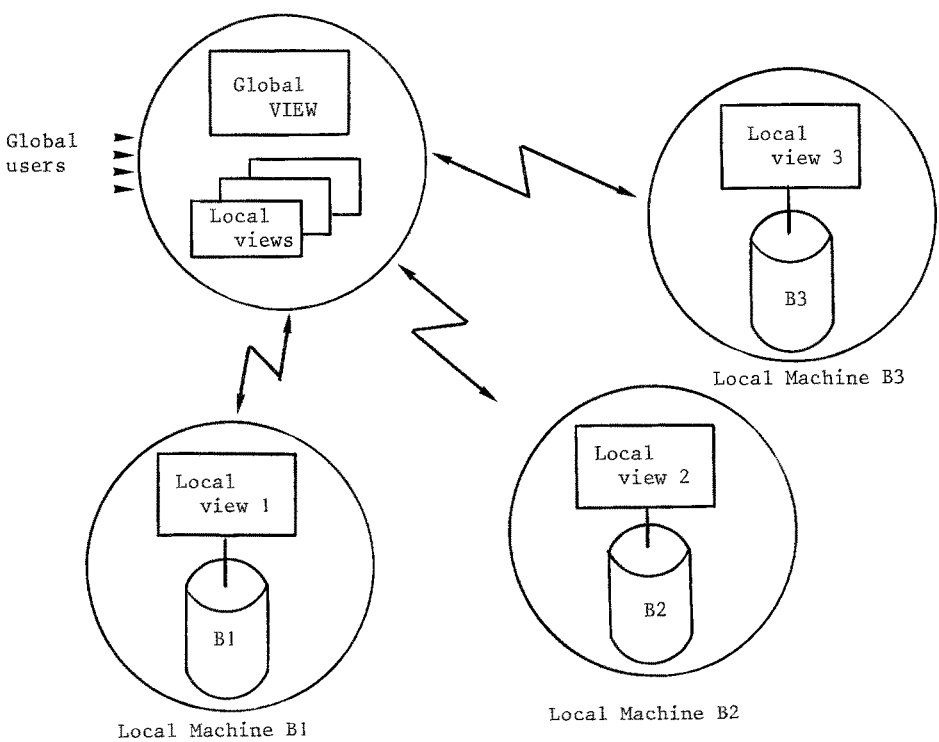


Fig. 3.1 - Logical Structure of the Distributed Data Base System

	SPACES / LEVELS	1 OBJECTS	2 NAMES-OBJECTS MAPPING	3 NAMES
1	LOCAL	. BASE local data bases . LSO local simple objects . LCO local compound object . LID local identifiers (keys) . LPC local programs codes . LPS local processes	Manipulation operators on local categories and functions (table TOP for m ∈ BASE) (local machines)	. LAC abstract categories . LCC concrete categories . LIN local id. name . LAF (LOF, LMF) mono and multi-valued functions . LPD local programs description
2	LOCAL/GLOBAL MAPPING	GR : global rules on categories and functions (objects localization) GEN, GCR, GDEL, GEX, GFA, GFL, GFE	Decomposition Process (section 3.5.1)	Name's localization GAC → Gacloc → BASE×LAC GCC → Gccloc → BASE×LCC GAF → Gafloc → BASE×LAF BASE GAC → locnamegac → LAC BASE GCC → locnamegcc → LCC BASE GAF → locnamegaf → LAF
3	GLOBAL	GSO global simple object GCO global compound object GID global identifier GPC global program codes GPS global processes	Manipulation operators on global categories and functions (table TOP for m = g) (global machine)	. GAC global abstract categories . GCC global concrete categories . GAF (GOF, GMF) global functions . GIN global identifier name . GPD global program description

Table T - Predefined Categories in MOGADOR

Table TOP : operators

Primitive Operations

Operation	Syntax	Output
creation	create (m, c, i)	ℓ ∈ LOGICAL
deletion	delete (m, c, i)	ℓ
existence test	test (m, c, i)	ℓ
enumeration	enumerate (m, c)	X(Set of i)
access	access (m, f, X) or f(X)	Z
link	link (m, f, x, Y)	ℓ
erase	erase (m, f, x, Y)	ℓ
graph	graph (m, f)	Z

Operations on Simple objects

Operation	Syntax	Output
addition	add (s1, s2)	s3 = s1 + s2
subtraction	sub (s1, s2)	s3 = s1 - s2
multiplication	mult (s1, s2)	s3 = s1 * s2
division	div (s1, s2)	s3 = s1 / s2

Operations on Compound objects

Operation	Syntax	Output
union (∪)	union (X, Y)	Z
intersection (∩)	inter (X, Y)	Z
difference (-)	diff (X, Y)	Z
cartesian product (×)	cart prod (X, Y)	Z
projection	project (X, filter)	Z
restriction	select (X, filter)	Z
join	join (X, Y, condition)	Z
division	divide (X, Y, condition)	Z
cardinality	card (X)	s ∈ INTEGER
sum, product	sum (X), prod (X)	s ∈ INTEGER ∪ REAL
eliminate redundancy	unique (X)	Y
concatenation	concat (X, Y)	Z

Notations

- machine name m m ∈ BASE ∪ {g} for categories and functions operations (g stands for "global")
- category c ∈ LCC ∪ GCC
- function f ∈ GAF ∪ LAF (N.B. Composition of function is a function)

- objects $\ell \in$ LOGICAL

  x, y, z objects, X, Y, Z set of objects

  i identifier i $\in$ LID $\cup$ GID

  s simple object.

3.3. Local Level (T11, T12, T13 of table T)

3.3.1. Local view description

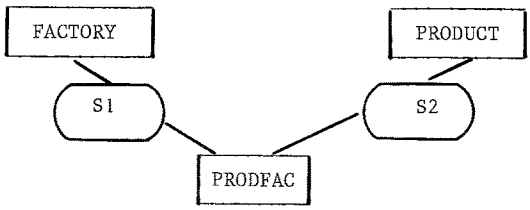
Given one local data base in use under a given DBMS, we consider that this data base corresponds to the following elements :

- a local object space constituted by the data stored in the data base : we assume that it is always possible to see this data as simple and compound objects (T11) [3][5]
- a local name space constituted by the data base schema which is re-interpreted in MOGADOR terms to form what we call the local view of the data base (T13).

This means that the data is seen as a collection of n-ary relations but explained in terms of local abstract categories (LAC), local concrete categories (LCC) with their identifier names (LIN) and local access functions (LAF).

Furthermore, associated to this view, we consider a serie of local programs which are pre-compiled in the data base and which realize at least the elementary operations of creation, updating and accessing.

Let us consider a simple example of distributed data base. We have a big enterprise managing several factories making several products. These factories are distributed over the country but data processing is done in 3 computing centers C1, C2 and C3. In C1 we consider data base B1 implemented under a codasyl-like system with the following schema :



S1 and S2 are codasyl-sets.

Each factory is described by a number F#, a town, the number of employees (NBEMP), the total of all salaries (TOTSAL) and the functioning budget (FBUD).

Each product has a number (P#), a name (PNAME).

Set S1 links a factory to all the products made in this factory and set S2 links a product to all the factories which made it.

In one record PRODFAC we find a factory number (F#), a product number (P#) and the

number of products made per day (NBPD).

In relational terms, we have :

B1 {
FACTORY (F#, TOWN, EMPNB, TOTSAL, FBUD)
PRODUCT (P#, PNAME)
PRODFAC (P#, F#, NBPD)

With MOGADOR concepts, the local view of this data base B1 is :

LAC = {F#, TOWN, EMPNB, TOTSAL, FBUD, P#, PNAME, NBPD}
LIN = {F#, F# × P#, P#}
LCC = {FACTORY, PRODUCT, PRODFAC}.

Local Function (LAF) :

Function's name	Type	Source	Target	Graph
town	mono	FACTORY	TOWN	in FACTORY
empnb	mono	FACTORY	EMPNB	in FACTORY
totsal	mono	FACTORY	TOTSAL	in FACTORY
fbud	mono	FACTORY	FBUD	in FACTORY
pname	mono	PRODUCT	PNAME	in PRODUCT
nbpd	mono	PRODFAC	NBPD	in PRODFAC
factory	multi	PRODUCT	FACTORY	in PRODFAC
production	multi	FACTORY	PRODFAC	in PRODFAC
fabrication	multi	PRODUCT	PRODFAC	in PRODFAC

N.B. To be complete, this local view example must contain local program descriptions involving :

- the program name
- the type of operation (access, update)
- the nature of inputs and outputs.

The program code is supposed to be stored into the local data base.

N.B. Programs are linked to concrete categories rather than to functions. In fact an access program linked to category C realizes all the monovalued functions having C as source.

In data base B2, we consider also works and products :

B2
MANUFACTORY (M#, TOWN, EMPNB, MBUDGET)
PRODUCT (P#, PNAME)
PROMAN (P#, M#, NBPD).

Note that in B2 the budget is not split into two components as in B1 (TOTSAL and FBUD).

In MOGADOR, we have :

LAC = {M#, TOWN, EMPNB, MBUDGET, P#, PNAME, NBPD}

LCC = {MANUFACTORY, PRODUCT, PROMAN}

LIN = {P#, F#, P# × F#}

Local functions of B2

Name	Type	Source	Target	Graph
town	mono	MANUFACTORY	TOWN	in MANUFACTORY
empnb	mono	MANUFACTORY	EMPNB	in MANUFACTORY
nbudget	mono	MANUFACTORY	MBUDGET	in MANUFACTORY
pname	mono	PRODUCT	PNAME	in PRODUCT
nbpd	mono	PROMAN	NBPD	in PROMAN
manufactory	multi	PRODUCT	MANUFACTORY	in PROMAN
production	multi	FACTORY	PROMAN	in PROMAN
fabrication	multi	PRODUCT	PROMAN	in PROMAN

Finally, in B3 we have :

B3 PRODUCT (P#, PNAME, PDESCR, SELLPRICE, COSTPRICE)

For each product made by the enterprise we have here a complete description with cost and selling prices.

In MOGADOR :

LAC = {P#, PNAME, PDESCR, SELLPRICE, COSTPRICE}

LCC = {PRODUCT}

LIN = {P#}

Local functions of B3 :

Name	Type	Source	Target	Graph
pname	mono	PRODUCT	PNAME	in PRODUCT
pdescr	mono	PRODUCT	PDESCR	in PRODUCT
sellprice	mono	PRODUCT	SELLPRICE	in PRODUCT
costprice	mono	PRODUCT	COSTPRICE	in PRODUCT

3.3.2. Local data base manipulation

The local view is stored by a MOGADOR local machine (see figure 3.1) which is an abstract machine whose physical components can be distributed. This machine can at least execute basic operations on local categories and functions (see T12). These executions involve in fact calls to local programs which are executed by the local DBMS. This MOGADOR local machine provides a standardized behaviour for heterogeneous data bases. It is used by a MOGADOR global machine form which global users manipulate the distributed data base (Figure 3.1).

3.4. Global level and global view

The description of the distributed data base can be logically divided into three parts :

- definition of global names
- localization or mapping between global and local names
- global rules on categories and functions.

3.4.1. Global names space (T31)

It is composed with names of global abstract categories (GAC), global concrete categories (GCC) with their global identifier name (GIN) and global functions GAF (GOF, GMF). Global categories and functions are of two kinds :

- distributed where the global name has some synonym into several local views. This mean that the global objects are in fact local objects dispersed over several object spaces.
- calculated where the global name has no equivalent in the local views. This means that the corresponding global objects are going to be elaborated at the global level by mean of a calculation expressed by a global rule (see section 3.4.3).

N.B. From this global name space, we consider that it is possible to derive external schemas given to users of the distributed data base and for whom the distribution of objects will be transparent.

Example of global view for B1, B2 and B3 :

At the global level we want to see data bases B1, B2 and B3 in the following manner : we consider two global concrete categories (GCC) namely :

- FACTORY which corresponds to distributed but not duplicated objects on B1 and B2. The global identifier name (GIN) is F#. We consider that the criterion for distributing factories depends on the value of the town attribute. For example factories located in New York, Boston or Washington are managed by data base B1 and factories located in Denver, Los Angelès, San Francisco are managed by data base B2. We shall come back to this point in section 3.4.3.1.

- PRODUCT which corresponds to distributed and duplicated objects over B1, B2 and B3 (GIN is P#). We make the assumption that each product is described at least in B3 i.e B3 contains the general catalogue of all the products.

We consider the following global abstract categories (GAC) :

- distributed : F#, TOWN, NBEMP, P#, PDESCR, PNAME, SELLPRICE, COSTPRICE and BUDGET.

The last one is not a strictly distributed category because it exists in B2 (MBUDGET) and not directly in B1 (TOTSAL+FBUD) ; this will be expressed together with the global function "budget" (see 3.4.3).

- calculated : let TOTALP be, for a given product the total number of this product made per day, over all the factories (B1 and B2).

We consider also the following global functions (GAF) :

GAF name	Type	Source	Target	Graph
town	mono	FACTORY	TOWN	in B1 or B2
nbofemp	mono	FACTORY	NBEMP	in B1 or B2
budget	mono	FACTORY	BUDGET	calculated in B1, exists in B2
pname	mono	PRODUCT	PNAME	in B1, B2, B3
pdescr	mono	PRODUCT	PDESCR	in B3
costprice	mono	PRODUCT	COSTPRICE	in B3
sellprice	mono	PRODUCT	SELLPRICE	in B3
production	multi	FACTORY	PRODUCT	in B1 or B2
inv production	multi	PRODUCT	FACTORY	in B1 and B2
totalp	mono	PRODUCT	TOTALP	calculated

3.4.2. Localization on names (T23)

For each distributed GAC, GCC and GAF we have to give the corresponding LAC, LCC and LAF using predefined functions gacloc, gccloc and gafloc (see table T : T23).

For instance

- gacloc (F╪) := ((B1,F╪), (B2,M╪))
- gccloc (PRODUCT) := ((B1,PRODUCT), (B2,PRODUCT), (B3,PRODUCT))
- gccloc (FACTORY) := ((B1,FACTORY), (B2,MANUFACTORY))
- gafloc (nbofemp) := ((B1,empnb), (B2,empnb))
- gafloc (production) := ((B1, inv factory), (B2, inv manufactory))
- gafloc (inv production) := ((B1,factory), (B2,manufactory))
- gafloc (budget) := ((B1, calc), (B2, budget))
- gafloc (totalp) := ∅.

3.4.3. Global rules (T21)

Global rules are a very important notion relating to distributed data bases. We have defined two kinds of global rules, i.e on global concrete categories (GCC) and on global functions (GAF). These global rules express what are the repercussions of a global operation concerning GCC or GAF at the local levels.

For example :

- how to execute the creation of a factory ?
- how to enumerate all the products ?
- how to calculate the TOTALP of a given product ?

Obviously to express all these semantics we need manipulation operators. In the following two sections, we give some examples of global rules. We want to stress, that global rules can express semantic properties of the distributed data base and in this

way, they have to be written by a global administrator. However in some simple cases these rules can be deduced automatically by the cooperation system given, for example, only global names and name localizations.

3.4.3.1. Global rules on concrete categories (GCC)

They concern the four operations creation, deletion, enumeration, existence test. For example to express the following semantic :

→ GR1 global enumeration of factories is realized by enumeration of local components, we write :

gen (FACTORY) := enumeration (gccloc (FACTORY)).

Note that gccloc (FACTORY) gives the set :

((B1, FACTORY), (B2, MANUFACTORY))

so the global rule will be interpreted as (see section 2.3.2)

(enumeration (B1,FACTORY), enumeration (B2,MANUFACTORY)).

At name's level this corresponds to the creation of two independant processes which can be executed in parallel one on the local machine B1, the other on B2.

→ GR2. To express that the creation of a factory depends on the value of attribute TOWN, we suppose the existence of a special global function named "locfactory" from TOWN to BASE and whose graph is :

TOWN	NEW-YORK	BOSTON	WASHINGTON	DENVER	LOS ANGELES	SAN FRAN-CISCO
BASE	B1	B1	B1	B2	B2	B2

Then the global rule for the creation of a factory is :

gcr (FACTORY, F#, TOWN) := create (locfactory (TOWN), locaname
locnamegcc (locfactory (TOWN), FACTORY), F#)

Creation of factory F15 located in DENVER will be :

create (B2, MANUFACTORY, F15).

(locnamegcc is defined in T23).

N.B. Note that if the graph of the localization function (here "locfactory") is not available, this function will be calculated. This allows more complex localization criteria.

→ GR3. The enumeration of all the products is to be made only on B3

gen (PRODUCT) := enumeration (B3, PRODUCT)

→ GR4. The deletion of a product is not allowed :

gdel (PRODUCT) := not allowed.

3.4.3.2. Global rules on functions (GAF)

They concern the three basic operations : access, link, erase.

For example :

→ GR5 : access to the number of employees of a given factory (GAF is nbofemp)

gfa (F#) := (access (gafloc (nbofemp), F#))

Two accesses will be generated, one on B1, one on B2 since we dont know where the factory is located (unless we define a localization function on factories).

GR6 : access to a factory budget (GAF is budget)

Let us give all the description of this function :

guf budget from FACTORY to BUDGET

locgaf (budget) := ((B1, calc), (B2, mbudget))

gfa (F#) := (add (access (B1, totsal, F#), access (B1, fbnd, F#)),
access (B2, mbudget, F#))

gfl ( ) := impossible the link operation is impossible at the global level

gfe ( ) := impossible idem.

GR7 : Access to all the products made by a given factory :

global multivalued function "product"

gfa (F#) := (access (gafloc (production), F#))

N.B. The result set will come only from B1 or B2.

GR8 : Access to all factories which made a given product : global and multivalued function "inv production".

gfa (P#) := concat (access (B1, factory, P#), access (B2, manufactory, P#)).

The result set is the concatenation of the two sets coming respectively from B1 and B2 because of duplication of product.

GR9 : Obtain the total number of product per day : global monovalued function totalp

gfa (P#) := add (sum (access (B1, nbpd o fabrication, P#)),
sum (access (B2, nbpd o fabrication, P#)))

gfl ( ) := impossible

gfe ( ) := impossible.

The operator "o" denotes the composition of functions.

3.5. Manipulation of the distributed data base

We have seen the main elements of LDDM language but these elements cannot constitute the external form of this language given to an end-user. It is beyond the scope of this paper to give the syntactic form of this external LDDM but we shall discuss briefly two points, i.e decomposition of a global transaction into local operations and execution of these local transactions.

3.5.1. Decomposition Process

Let us consider a sample of global transaction :

Q1 give number and town of all the factories which made products whose costprice is > p.

Q1 can be expressed in LDDM as :

X ← F# : id × TOWN : town [(inv product (inv.costprice (> p)))]

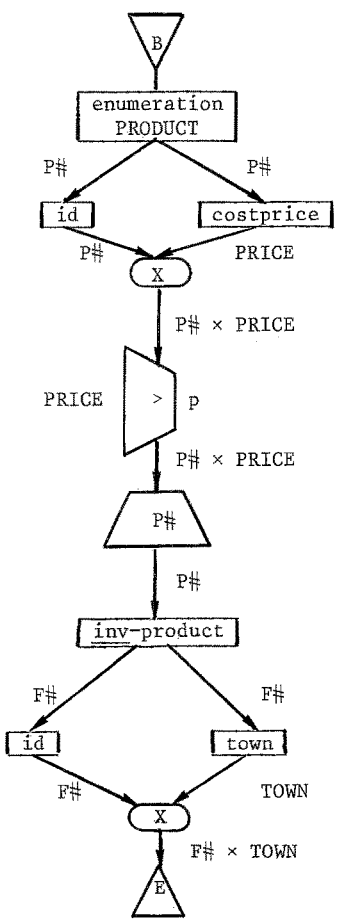
This means that from "p" we apply the inv costprice function to find all products whose costprice is greater than p. On the result (set of P#) we apply inv-product which gives all the factories (set of F#) making those products. On this set of F# we apply two functions, "town" and "id" (the identity function) to form a set of tuple :

(F#, TOWN).

Since inv.costprice is not defined in the global view, this expression is in fact :

X ← F# : id × TOWN : [town (inv.production (project (select
(P# : id × PRICE : costprice (enumeration (PRODUCT)), PRICE > p), P#)))].

Which can be transformed into the following graph, showing the macrosynchronization of operations :



All operations in square boxes are going to be decomposed into local operations using global rules. This will give another graph where some parts are to be executed by local machines. From this graph we have to generate a distributed program and to execute it [29].

3.5.2. Execution process

The distributed program is composed of several procedures which are distributed over several sites. On each site mechanisms are provided to execute these procedures some of them involving calls to procedures which are located in another site [15][16].

Therefore a global transaction is transformed into several global procedures which call local procedures in order to initialize local program execution and which are called themselves by local procedures when local objects are available.

CONCLUSIONS

We have presented here the basic concepts of our distributed data model MOGADOR together with the elements of a language for describing and manipulating distributed data.

In providing an homogeneous level for the description and the behabiour of distributed data bases MOGADOR is not only a data model but also a logical tool for the design of heterogeneous distributed data base management systems.

ACKNOWLEDGMENTS

We are grateful for the comments of J.R. Abrial, C. Delobel and M. Léonard and of all SIRIUS people. We also acknowledge the contribution of all the POLYPHEME team : J.M. Andrade, E. André, J.Y. Caleca, P. Decitre, C. Euzet, Nguyen Gia Toan and A. Stiers. We also acknowledge Professor M. Shave for correcting our english.

REFERENCES

[1] J.R. ABRIAL, Data Semantics, IFIP-TC2 Working Conference, Cargèse, Avril 1974.

[2] J.R. ABRIAL, Langage de spécification Z. Paris, Mai 1977.

[3] M. ADIBA, C. DELOBEL, M. LEONARD, A unified approach for modelling data in logical data base design. IFIP-TC2, Freudenstadt, January 1976.

[4] M. ADIBA, C. DELOBEL, The cooperation problem between different Data Base Management Systems. IFIP-TC2 Working Conference, Nice, January 1977.

[5] M. ADIBA, Projet POLYPHEME : MOGADOR : Un MOdèle GénérAl de DOnnées Réparties. Laboratoire d'Informatique. Research Report 81, July 1977, Grenoble.

[6] ASTRAMAN et al., System R. Relational Approach to Data Base Management. ACM-TODS, Vol.1 n°2, June 1976.

[7] ANSI/SPARC, Interim Report ACM Sigmod FDT.7, December 1975.

[8] G. BRACCHI, G. PELAGATTI, P. PAOLINI, Models views and Mappings in multilevel Data Base representation. Politechnico di Milano. 1976.

[9] BROOKS, CARDENAS, NAHOURAII, An approach to data communication between different GDBMS. Very Large Data Base Conference. Brussels, September 1976.

[10] J.M. SMITH and D.C.P. SMITH, Data base abstractions. Aggregations and Generalization ACM-TODS. Vol.2 Nb.2, June 1977.

[11] M.E. DEPPE, J.P. FRY, Distributed Data Bases : a summary of research. Computer Networks 1.1976.

[12] J.B. ROTHNIE, N. GOODMAN, An overview of the preliminary design of SDD-1 : a system for distributed data bases C.C.A. Cambridge. 1977.

[13] C.C. PINTER, Set Theory. Addison Wesley Publishing Company. 1971.

[14] G. GARDARIN, M. JOUVE, C. PARENT, S. SPACCAPIETRA, Designing a distributed data base management system. AICA. October 1977.

[15] E. ANDRE, P. DECITRE, On providing Distributed Applications Programmers with control over synchronizations. Accepted for publication in computer network protocols symposium, Liège, February 1978.

[16] DANG, G. SERGEANT, System and Portable Language intended for distributed and heterogeneous network applications. ENSIMAG, December 1976.

[17] R. DEMOLOMBE, M. LEMAITRE, Rôles d'un modèle commun dans la conception d'un SGBD réparti : analyse des principaux modèles CERT. Research Report. March 1977.

[18] E.F. CODD, A relational model of data for large shared data banks. CACM 13, 6 (June 1970).

[19] P. PIN-SHAN CHEN, The Entity-Relationship model. Toward a unified view of data ACM TODS, March 1976.

[20] J.Y. CALECA, J.P. FORESTIER, L'interrogation simultanée de plusieurs bases de données. Rapport de D.E.A., Université de Grenoble. Juin 1976.

[21] D. CHAMBERLIN et al., Sequel 2. A unified approach to Data Definition, Manipulation and Control. IBM Journal of Research and Development, Vol.20 n°6, November 1976.

[22] D. CHAMBERLIN et al., Views Authorization and locking in a relational Data Base System. Proc. 1975 National Computer Conference Anaheim Ca., May 1975.

[23] Canning Publications, Distributed data Systems. EDP Analyser, June 1976, Vol.14 n°6.

[24] J. LE BIHAN, SIRIUS Project, IRIA, Domaine de Voluceau, 78150 LE CHESNAY, France.

[25] POLYPHEME, Propositions pour un modèle de répartition et de coopération de Bases de Données dans un réseau d'ordinateurs. Laboratoire Informatiques CII/ENSIMAG/USMG, Université de Grenoble. Rapport de Recherche n° 29, Décembre 1975.

[26] E. NEUHOLD, M. STONEBRAKER, A distributed data base version of INGRES. Memorandum n° ERL-M612, Septembre 1976, Université de Californie, Berkeley.

[27] M.E. SENKO, DIAM as a detailled example of the ANSI/SPARC architecture. IFIP-TC2 Working Conference, Freudenstadt, Germany, January 1976.

[28] M. STONEBRAKER et al., The Design and Implementation of INGRES. ACM TODS Vol.1 n°3, September 1976.

[29] M. ADIBA, J.Y. CALECA, Modèle relationnel de données réparties, problème de décomposition. Journées sur le modèle relationnel. Paris. Avril 1978.