THE INDIVIDUAL MODEL

by C. DEHENEFFE
J.-L. HAINAUT
Institut d'Informatique de Namur

and H. TARDIEU
Centre d'étude technique de l'équipement d'Aix-en-Provence

1. — Introduction

1.1. — *Individual Model : a tool for analysis*

The individual model is conceived as a tool for designing and structuring data bases : it is intended for data analysts and data administrators.

In this way, this model is conceived first as a structuring tool for a set of data. It should allow the analyst the possibility to build, simply and efficiently, data structures. We consider that a necessary prerequisite for simplicity and efficiency of the process is the ability the data analyst to use a model representing structures that will be manipulated by future users of the data base.

We know by our personal experience that a user thinks in terms of units of information and associations between units. We define a unit of information as a set of elementary data — properties which characterize an *individual.* This individual may be either a subject (employee, customer, supplier) or an object (end products, parts, etc.) or a concept (income, profit center, etc.). An individual may have properties (ex. : name, age, sex, address) which are called the individual's attributes. The definition of attributes and corresponding values depends on the specific context of the class of application involved; for instance, we shall distinguish for the same individual between identifiers (customer, supplier, products), events (orders, inventories) and results. This model is qualified as *individual* in opposition to models (1) which are only concerned with elementary data. Besides information units, the user "sees" and "manipulates" associations between individuals.

These are defined as couples of individuals (e.g. association "employee" which ties up a person and a company). These associations which we call "relations", may themselves have properties. For instance insurance card, is an attribute of the relation "owner" between a person and a car. Units of information and association are in fact organized as structures : vectors, blocks, hierarchies, networks, etc. We have chosen in this model a set of structures which covers most of the real situation.

The types of structures are dedicated to "applications" problems and are completely independent of the implementation. Our purpose is to complete those structures by an access model which will be processed by the data administrator.

It is not enough to propose a tool for building structures : the model should allow the description of these structures and it will be used as the basis support of a data definition language (DDL).

As an analysis tool, the proposed model has another capability : it will be used as a basis for the definition of a data manipulation language (designation, deletion, insertion, modification). This model which defines a graph, allows description of structure "navigation".

Next, we shall associate to this model an optimization mechanism in order to improve access time and facility of processing.

The proposed model associates with relations, existence properties (cf. infra 2.2.1.3) which in turn can generate procedure simplifying the updating and questioning of the data base.

Last, the proposed formalism should allow implementation of algorithms in order to optimize the chosen structure (e.g. redefining a structure which can minimize "navigation" time in a

(1) See Codd's "Relational Data Model", Bibliography [4].

subset of the data). The model should also allow optimization of programs written in DML.

1.2. — *Functional model and access model*

In accordance with our method of analysis which goes from the "perceived" problem to implementation, the individual model has been divided into two embedded models : the functional model and the access model.

The first model consists of a structuring tool that the analyst can use to represent "the real world" (1). It will allow the building and description of data structures, the choice between different structures and the definition of a language manipulating data described by structures.

The second one is a set of tools used by the analyst and the data administrator to implement structures built up with the help of the conceptual model.

In the first model, an analysis of applications should lead the analyst to the list of individuals, attributes and semantic relations between the individuals. In the second model, applications considered as access algorithms depend on the particular accesses to data : we have to combine semantic relations and access relations. The access model will allow description and choice of the best structure for access and definition of a data manipulation language explicitly taking into account access structures.

In embedding these two models, we wish to ensure that the second model does not reduce the semantic content of the first; it should just complete the first one, through dealing with the access relations contained in the semantic relations.

2. — Individual model of a data base : a conceptual model

2.1. — *Introduction*

In order to find essential notions which we will define formally, it seems useful to refer first to an example of a way of building a data base.

(1) Cf. Introductory report.

Suppose, for instance, a personnel data base. The user lists all elementary information of any value to him; say for example : name, first-name, sex, employee-number, social security number, etc. This information is structured as entities of *individuals* : the user builds in this way the following individuals :

. PERSON (SOCIAL-SECURITY-NUMBER, NAME, SEX, AGE)

. DEPARTMENT (N° DEPT, MANAGER, ACTIVITY)

. CHILD (NAME, AGE)

. etc.

Next the user points out the following relations :

. PERSONS BELONG TO ONE OR MANY DEPARTMENTS

. CHILDREN BELONG TO ONE PERSON

. etc.

Considering such a process, we believe, that a conceptual model for a data base should allow :

— definition of individuals and their associations with attributes

— definition of relations

This model will be designated to represent as well as possible the steps of the process.

2.2. — *Objects of the model*

2.2.1. — Properties.

A property is an elementary information that the user wishes to have in his data base. This implies the definition of a mechanism which will assign value to any property.

Example :

NAME is a property which can take values such as "SMITH", "TARDIEU" or "DEHENEFFE"

2.2.2. — Individual.

An individual is an object to which we associate certain properties. An individual may be given a value by giving a value each property, EXAMPLE :

To the individual PERSON we may associate the properties NAME, CHRISTIAN NAME, SEX and AGE. If we give to name the value "SMITH", to christian name the value "JOHN", to sex the value "MALE" and to age the value "27", the 4-tuples (SMITH, JOHN, MALE, 27) is considered as an *occurrence* of the individual PERSON.

2.2.3. — RELATION.

If we consider two individuals, we can associate to the so-defined couple, 0 or many relations : a relation is an association between two individuals which allows us to associate semantically two occurrences of those individuals.

We may give properties to a relation. Those properties are called *intersection properties.*

Let us consider for instance the following individuals MANUFACTURING-UNIT(UNIT-NAME, DESCRIPTION), PRODUCT(PRODUCT-NO, PRODUCT-NAME, DESCRIPTION), let us create the relation MAKING between MANUFACTURING-UNIT and PRODUCT. This relation is a way to relate PRODUCTS to MANUFACTURING-UNITS that make them; we can also give some properties to this relation, MAXIMAL-CAPACITY, MINIMAL-CAPACITY which are properties of the couple MANUFACTURING-UNIT and PRODUCT tied up by the relation MAKING.

2.2.4. — TYPOLOGY OF RELATION.

We can describe a typology of relations with two points of view :

(1) *Depending on the number of individuals tied up by a relation.*

This typology can be explained using the following example : the relation SPOUSE which relates two occurrences of the individual PERSON could only be a one to one relation.

Let us consider two individuals I and J; then we shall define a relation between I and J as :

1 — 1 if an occurrence of I is related to only one occurrence of J and if to an occurrence of J is related only one occurrence of I.

1 — n if only one occurrence of I can be related to an occurrence of J.

n — 1 if only one occurrence of J can be related to an occurrence of I.

n — m in other cases.

(2) *Depending on an existence property of the two members of the relation.*

(*a*) strong member of an individual by a relation.

J is a strong member of I by the relation R if :

— deleting an occurrence *i* of the individual I implies deleting all the occurrences *j* of J related to I.

— inserting an occurrence *j* of J supposes that *j* is related to at least one occurrence *i* of I.

(*b*) weak member of an individual by a relation.

J is a weak member of the individual I by the relation R if it is not a strong member by this relation.

2.3. — *Data manipulation and designation language*

This model introduces some new objects and mechanism that we should be able to describe and manipulate. Consequently, the model is the basis of two languages :

(*a*) data description language which allows the description of individuals, attributes and relations,

(*b*) data manipulation language which has two purposes

— specify an instruction

— retrieve objects (individuals or relations) in the data base to operate on them.

2.4. — *Implementing the model*

There are two ways of implementing the model :

(*a*) to write a system which allows, in fact, all the possibilities offered in the model

(*b*) to use some existing DBMS such as IMS, MIISFIIT, SOCRATE, SESAM and implement the model with the tools proposed by the DBMS.

We think that despite some particular constraints due to specific DBMS, the best way to use this model is for analysis at the conceptual and logical level before creating and manipulating data with an existing DBMS.

2.5. — *Formal presentation of the individual model*

Before presenting the individual model, we think it is useful to recall some "basic concepts" used in the theory of predicates. This old theory appears to be a very interesting theoretical approach to data bases.

2.5.1. — Essential concepts of the predicate theory.

A data base may be considered as a set of logical propositions. All these propositions, coming from the "real world" may be analysed using models and it is precisely the purpose of a conceptual model to propose a logical approach.

When we have to conceive a conceptual model, we should especially take care of the logical process which transforms a set of logical statements unknown by the user into a host structure for all inquiries by the user. This process which we call "*returning process*" consists of the "designation" and this "designation" is especially well formalized in the theory of predicates.

(i) *Definition.*

We call *elementary logical proposition* a logical statement which can only take two values

True or false noted T, F

(*a*) 1 place — predicate or attribute predicate.

Any elementary logical proposition may be analysed in

subject + predicate

EXAMPLE :

SOCRATE	*is a man*
subject	predicate

we shall distinguish :

variables

x, y, z	subject variables
a, b, c	attribute predicate variables

constants

χ, ξ	subject variables
α, β, γ	attribute — predicate constants

An elementary logical proposition could be written :

$$\underset{\text{predicate}}{\alpha}\ \underset{\text{subject}}{\chi}\quad T$$

We read this proposition : $\alpha\chi$ is true; this means that for all χ the proposition $\alpha\chi$ is true.

N. B. — We note that a logical proposition is only realized by association of a subject constant and a predicate constant.

(*b*) 2 place — predicate or relational predicate.

Our former analysis may be insufficient because certain logical proposition are best analysed by 2 place-predicates :

predicate — subject 1, subject 2

EXAMPLE :

Louis XIII is the son of Henri IV

is the son of —	*Louis XIII —*	*Henri IV*
predicate	subject 1	subject 2

N. B. — A one place predicate analysis would have shown the predicate "is the son of Henri IV". We see that some information would have been lost, because in this way we should have eliminated the transformation which indicated that "Henri IV is the father of Louis XIII", we can introduce an additional notation :

r, s	relational predicate variable
ρ, σ	relational predicate constant

EXAMPLE : the previous example could be written.

$$\rho\, x\, z$$

(ii) *Designation.*

Designation simply consists in setting up a logical proposition which we want to be true and whose subject is not determined. Formally the subject constant χ is replaced by a subject variable x. In order to precisely show the subject, that we want to designate, we write at the very beginning of the designation the variable of subject x followed by the sign | which means "such that".
EXAMPLE : $x \mid \alpha x$ is read : x such that αx and means "all x such that αx is true".

We see that this operation is very simple if and only if an analysis in terms of subject and predicate has been previously made and if this analysis is known to the user.

We can also obtain a designation using a relational predicate

$$x \mid \mathscr{L} xz$$

EXAMPLE :

(1) if α is the attribute predicate "is a man"
$x \mid \alpha x$ designate all men

(2) if $\mathscr{L}$ is the relational predicate "is the son of" and z is the the subject constant Henri IV
$x \mid \mathscr{L} xz$ designate all sons of Henri IV

The individual model using predicate theory will allow a logical formalization involving data definition and data manipulation. The process of designation essentially uses the notion of individual.

2.5.2. — DESCRIPTION OF INDIVIDUAL MODEL.

(i) *Let us introduce the following spaces :*

P : property space (or elementary data space)
(Elements are predicate variables a, b, c).

J : individual space. We call individual a being whose only property is to exist or not exist.

V : value space
(Elements are attribute predicate constants α, β, δ).

R : relation space which ties up couples of individuals
(Elements are relational predicate constants).

These four spaces have been shown (cf. Introduction) to form a rather natural representation of the "real world".

Any user, who has defined his objectives, most frequently uses the following way :

(*a*) he quite clearly sees elementary data which he needs. These data correspond to elements of P,

(*b*) he secondly determines a set of entities in building associations between elementary data.

(ii) *Definition.*

An *application* of the set A to the set B is a correspondence which associates to one element of A, 0 or many elements of B.

We shall note :

$D : A \rightarrow B$

$a \varepsilon A \quad (a) = b_1, \ldots b_m \quad b_i \varepsilon B, i = 1, \ldots m$

An application which associates to an element of A one and only one element of B is a *function.*

(iii) *Study of the individual space* J.

An individual is a being whose only property is to exist or not exist.

We can figure out this individual space as a space of names or numbers which designate unambiguously an individual, thus recording the facts chosen to be interesting for our objectives.

(*a*) application ENTITY of an individual.

The application ENTITY is defined by :

$$\text{ENTITY} : J \rightarrow P$$

if $I \varepsilon J$, ENTITY (I) $= p_1, p_2, \ldots, p_i \ldots p_m \varepsilon P$

The properties p_i are called attributes of the individual.

We believe that the application ENTITY is a quite natural way of thinking for the user.

When he has chosen a set of "elementary data" the user next executes the following operations.

1° choice of a name for the individuals : this is the building of the space J.

2° unique choice (depending on application or semantic of the problem) of a set of attributes which he canonically associates to an individual.

EXAMPLE :

If we consider the individual "person", we associate attributes such as name, christian name, sex if these data are interesting for defined applications.

N. B. — It is very important to notice well the difference between entity and individual. Actually the difference made in this model is the same difference as the one made by linguists between significant (entities) and signified (individual). The classic example is the word h-o-r-s-e which is the significant as opposed to the 4 legged-animal which is the signified.

(*b*) Application VALUE of a property.

The application VALUE can be defined by :

$$\text{VALUE } \mathsf{P} \rightarrow \mathsf{V}$$

if $p \in \mathsf{P}$

$$\text{VALUE } (p) = V_1, V_2 \ldots V_m \quad \text{with } V_1, V_2 \ldots V_m \in \mathsf{V}$$

This application is used to associate to a property the set of values that this property can take.

(*c*) Application REALIZATION of an individual.

We define the application REALIZATION of an individual I as an application

$$\mathsf{J} \rightarrow \mathsf{V}$$

if we have $\mathrm{I} \in \mathsf{J}$ and i_n an element of I, we shall write :

$$\text{REALIZATION } (i_n) = v_{1,n'}, v_{2,n'}, \ldots v_{n,n}$$

with $v_{1,n'}, v_{2,n'}, \ldots v_{n,n} \in \mathsf{V}$.

We shall verify that if

$$\text{ENTITY (I)} = p_1, p_2, \dots p_n$$

then

$$\text{VALUE } (p_1) = v_{1,1}, \dots v_{1,n}, \dots v_{1,m}$$
$$\text{VALUE } (p_2) = v_{2,1}, \dots v_{2,n}, \dots v_{2,m}$$
$$\vdots$$

N. B. — We can define the function : projection on p_m of REALIZATION of I.

$$\mathsf{J} \times \mathsf{P} \rightarrow \mathsf{V}$$

If i_n is an element of I and $\text{I} \varepsilon \mathsf{J}$ and p_m is a property such that $p_m \varepsilon \mathsf{P}$, we shall have :

$$\text{PROJECTION}_{p_m} \text{ REALIZATION } (i_n) = v_{m,n}$$

with $v_{m,n} \varepsilon \mathsf{V}$.

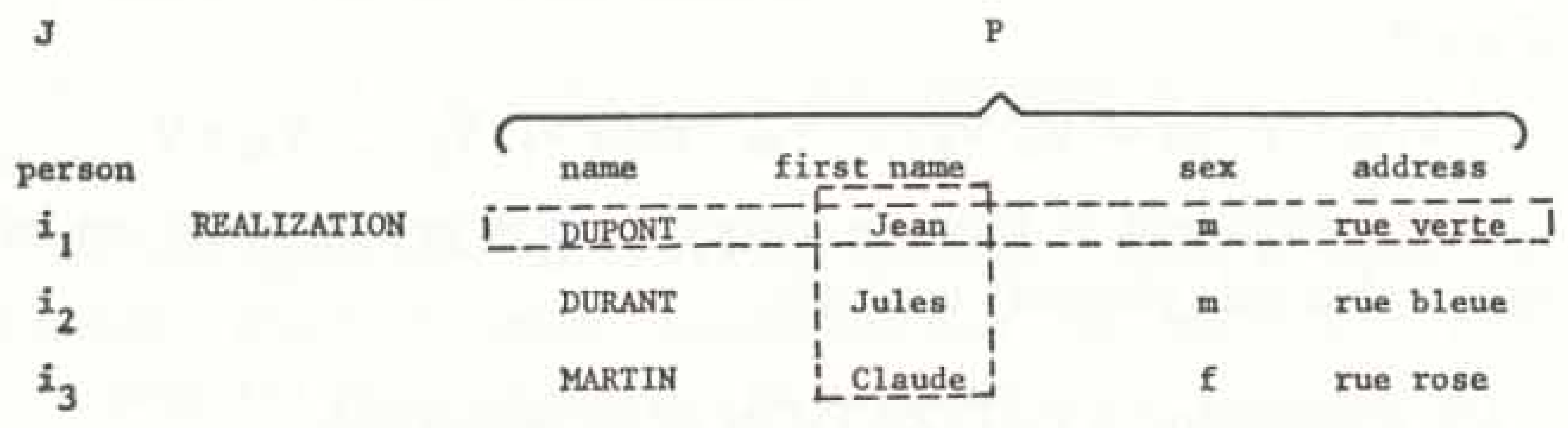
EXAMPLE :

J = {person, child, car}

P = {name, first name, sex, address, age, type, color, power}

the application ENTITY gives

person (name, first name, sex, address)
child (name, first name, sex, age)
car (type, color, power).



J		P			
person		name	first name	sex	address
i_1	REALIZATION	DUPONT	Jean	m	rue verte
i_2		DURANT	Jules	m	rue bleue
i_3		MARTIN	Claude	f	rue rose

REALIZATION (i_1) = DUPONT, Jean, m, rue verte
VALUE (christian name) = Jean, Jules, Claude.
$\text{PROJECTION}_{\text{name}}$ REALIZATION (i_1) = DUPONT.

Remark :

In an application VALUE, each property can take a special value "U" = "unknown", which is given to an attribute whose value is not known.

(iv) *Study of the space of relations* R.

(*a*) Association between two individuals by relation.

This association can be formally introduced by the application :

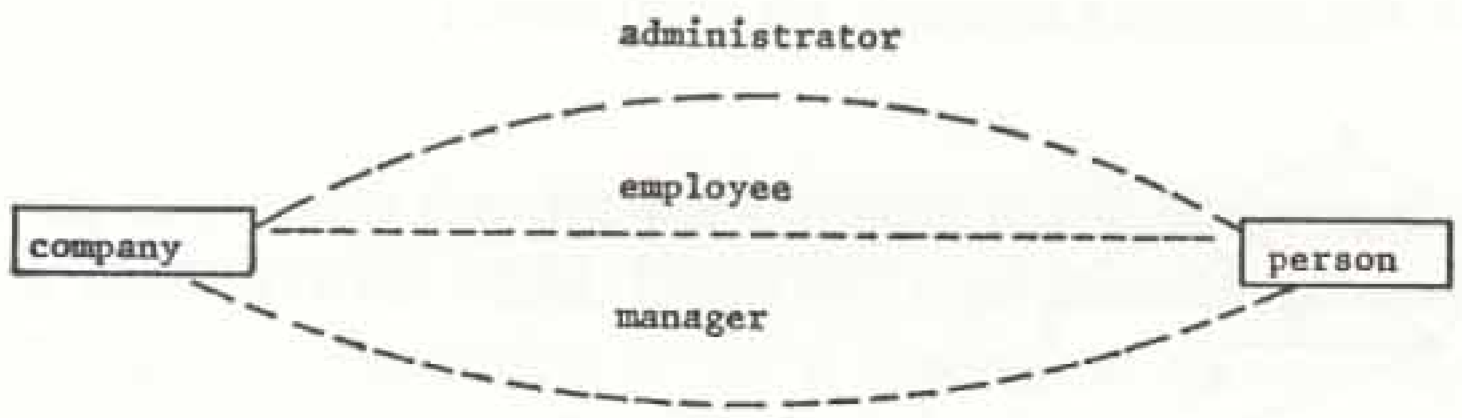
$$\text{RELATION} : \mathsf{J} \times \mathsf{J} \rightarrow \mathsf{P}$$

$\forall$I, J ε J

$$\text{RELATION (I, J)} = r_1, r_2, \ldots, r_n \qquad \text{with } r_i \,\varepsilon\, \mathsf{R}$$

this application allows us to obtain all the semantic relations existing between two individuals.

EXAMPLE :



RELATION (company, person) = Administrator, employee, ınanager.

(*b*) Application ENTITY of a relation.

We call entity of a relation, the application

$$\mathsf{R} \rightarrow \mathsf{P}$$

ENTITY $(r_n) = p_1, p_2 \ldots p_m$ with $p_1, p_2 \ldots p_m \,\varepsilon\, \mathsf{P}$.

This application allows us to associate to a relation, the set of attributes corresponding to this relation (this is especially interesting for $n - n$ relations).

EXAMPLE :

In the previous example we could have :

ENTITY (employee) = (starting date, end date).

(*c*) Application REALIZATION of a relation r between I and αJ.

We call REALIZATION of the relation r between I and J the application :

REALIZATION $(r_n) = (i_1, j_1), (i_2, j_2) \ldots (i_n, j_n)$

with $r_n \varepsilon$ R

$(i_1, i_r \ldots i_n) \varepsilon$ I I, J ε J

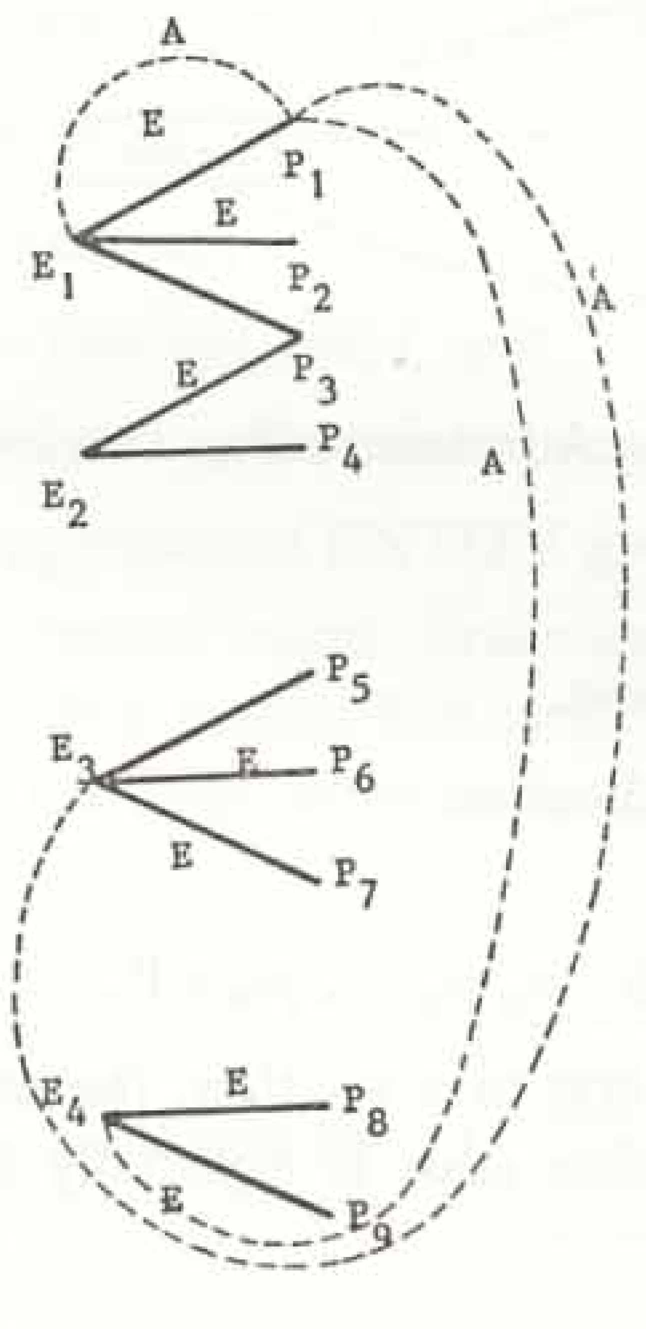
$(j_1, j_r \ldots j_n) \varepsilon$ J

We see that an application REALIZATION gives the correspondence b(tween a given relation r_n and all couples (i_n, j_n) which satisfy it.

N. B. — We can use the notation "couple for r_n" to designate all couples resulting from the application REALIZATION (r_n).

EXAMPLE :

With the previous example we shall have :



E : employee
A : administrator

REALIZATION (A) =
$(E_1, P_1), (E_3, P_1), (E_4, P_1)$

REALIZATION (E) =
$(E_1, P_1), (E_1, P_2), (E_1, P_3), (E_2, P_3)$...

(*d*) Association between an individual and a relation.

RELATION : J $\times$ R $\rightarrow$ R

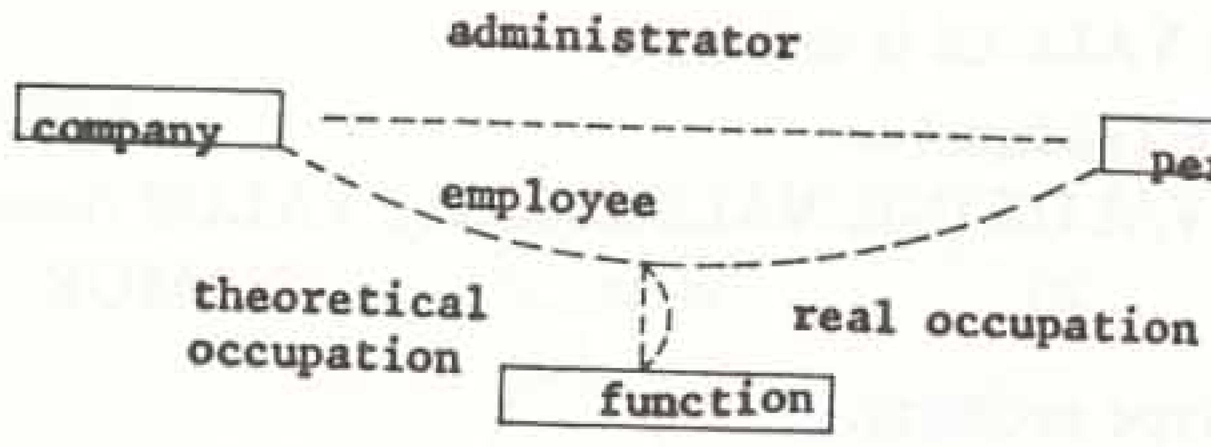
K ε J

$r_n \varepsilon$ R

RELATION (K, r_n) $= r_1, r_2 - - - r_m$.

EXAMPLE :

Using the previous example, we associate the individual function to the relation employee.



RELATION (function, EMPLOYEE) = theoretical occupation
real occupation.

In fact, this means that we can associate to the occurrence of the relation EMPLOYEE (IBM, DUPONT) the function programmer either for a real occupation or for a theoretical occupation. This is a well known problem for personal managers.

N. B. — The application RELATION can be repeated iteratively and allows us to obtain *n*-tuples.

2.5.3. — TYPOLOGY OF SPACES P AND R.

(i) *Typology of the space of property.*

We shall establish a typology of properties; we can admit that there are several kinds of properties :

(*a*) Elementary property.

These properties take their values by the application

$$\text{VALUE}\ (p_n) = \text{V}_1, \text{V}_2 \ldots \text{V}_n$$

We can define the list of authorized values : discrete values slices, etc. One can prohibit value "U".

EXAMPLE :

AGE 0 through 110

(*b*) Block type property.

A block type property is an object structured as follows:

$\alpha = (\alpha_1, \alpha_2 \ldots \alpha_n)$ where α is the object obtained by concatenating elementary properties $\alpha_1, \alpha_2 \ldots \alpha_n$.

Application VALUES is defined :

VALUE (address) =
(VALUE (N)), VALUE (street), VALUE (town))
20 verte NAMUR

(*c*) Vector type property.

A vector type property is a couple (α, J) where α is the name of a property and J an index. The application VALUE is so defined :

$$\text{VALUE}(\alpha, \text{J}) = (\text{VALUE}(\alpha), \text{VALUE}(\text{J}))$$

EXAMPLE :

VALUE (first name, I) = VALUE (first name), VALUE (I)
(JEAN, 1), (GUSTAVE, 2) . . .

which means Jean is the first christian-name and Gustave the second one.

(ii) *Typology of the relation space.*

The application of RELATION statically describe the possible relations between individuals. But a date base is dynamic since it is possible to add or to delete occurrences of individuals.

It seems useful to study properties of relations that will describe for a date base the some properties that are invariant.

We can introduce two typologies :

(*a*) typology using cardinality
(*b*) typology using the existence property.

A) Typology using cardinality.

Let us define a new application which we call PROJECTION

$$\text{R} \times \text{J} \rightarrow \text{J}$$

If we have a relation : $r_n \in \text{R}$ between I and J, we write :

$$\text{PROJECTION}(r_n, i_k) = j_1, j_2 \ldots j_e$$

we define all j_n associated with an i_k by the relation r_n. We define the relation r_n between i and J as :

$1 - 1$ if $\forall i_n \varepsilon \mathrm{I}\ \exists_1 j_k$ PROJECTION $(r_n, i_n) = j_k$

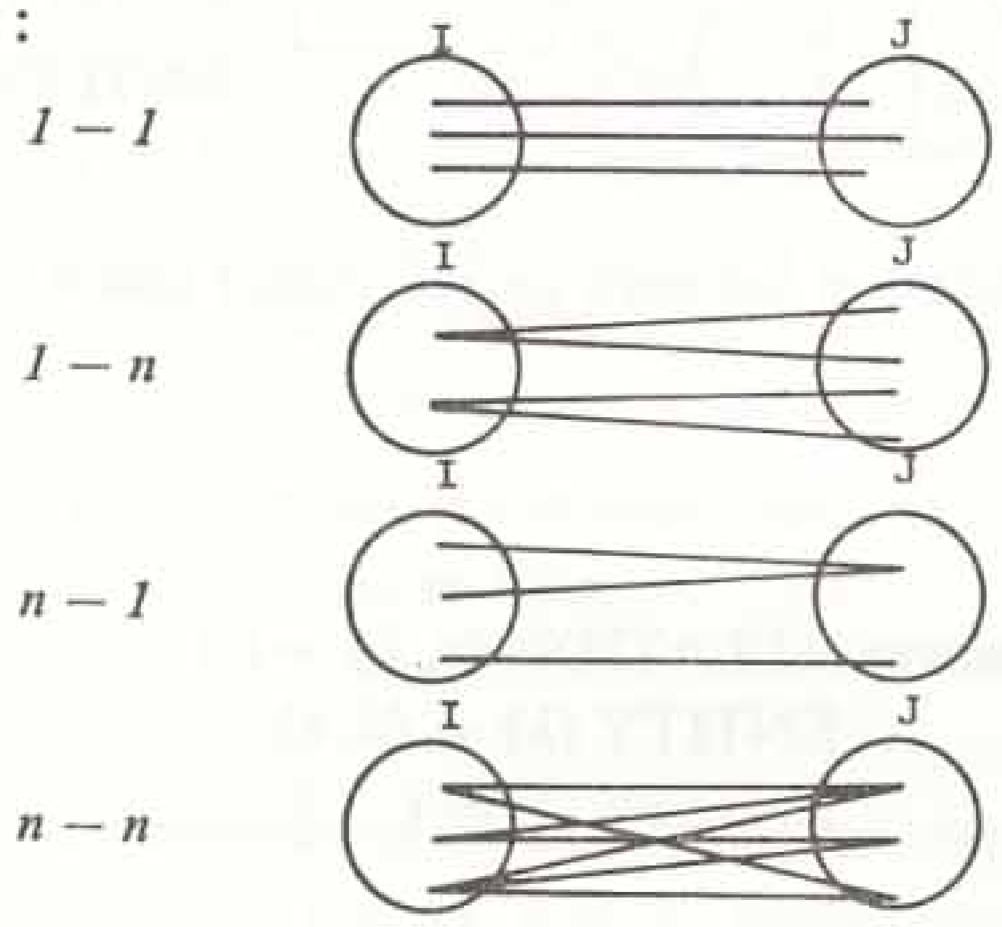
and $\forall j_n \varepsilon \mathrm{J}\ \exists_1 i_k$ PROJECTION $(r_n, j_n) = i_k$

$1 - n$ if $\forall j_n \varepsilon \mathrm{J}\ \exists_1 i_k$ PROJECTION $(r_n, j_k) = i_k$

$n - 1$ if $\forall i_n \varepsilon \mathrm{I}\ \exists_1 j_k$ PROJECTION $(r_n, i_n) = j_k$

$n - n$ in other cases.

EXAMPLE :



(B) Typology using property of existence

If r_n is a relation between I and J, we define

. J is a strong member of I through R if the following properties are true

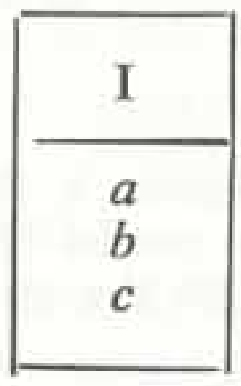
— deleting an occurrence $i_n \varepsilon \mathrm{I}$ implies deleting all $j_n \varepsilon \mathrm{J}$ defined by PROJECTION $(r_n, i_n) = j_1, j_2 \ldots j_p$

— adding an occurrence $j_n \varepsilon \mathrm{J}$ implies the existence of an occurrence i_k such that PROJECTION $(r_n, j_n) = i_1, i_2 \ldots i_k, i_n$

. J is a weak member in all other cases.

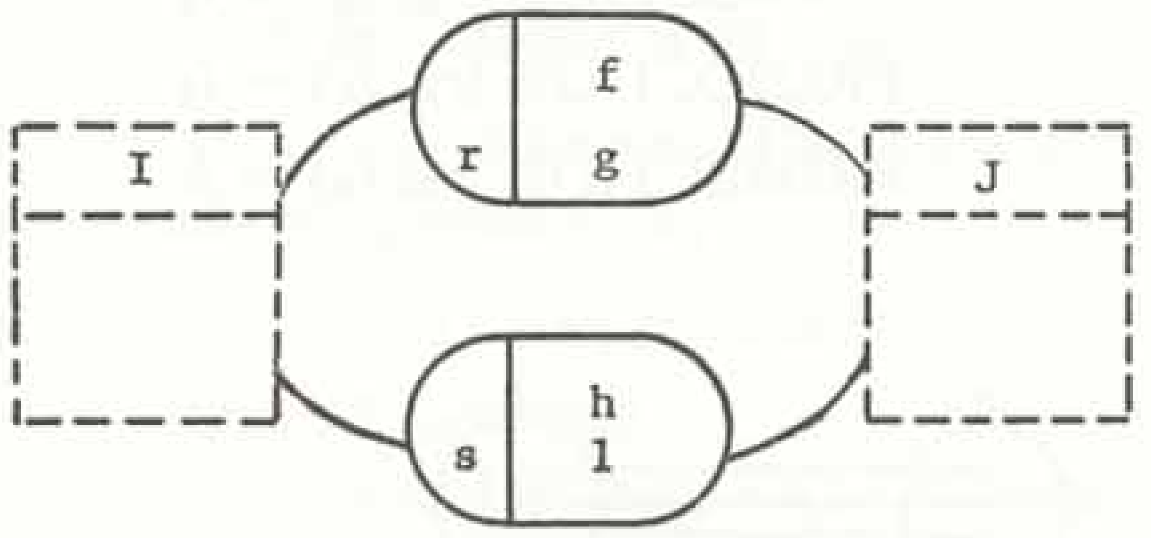
2.5.4. — GRAPHICAL REPRESENTATION.

We represent an individual as



where ENTITY (I) = (a, b, c)

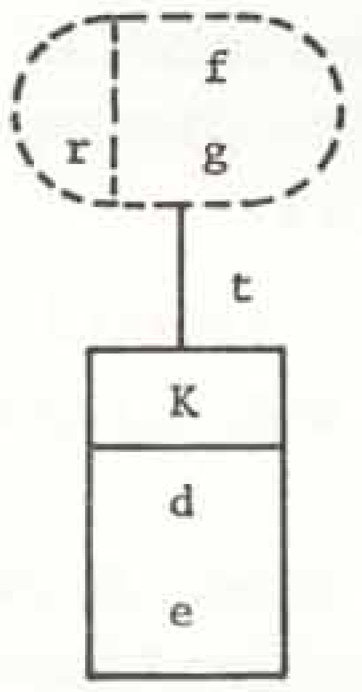
we represent relations between individuals as



where

RELATION
(I, J) = (*r*, *s*)
ENTITY (*r*) = (*f*, *g*)
ENTITY (*s*) = (*h*, *l*)

we represent a relation between an individual and a couple as :



where RELATION (*r*, *k*) = (*s*)
ENTITY (*k*) = (*d*, *e*)
ENTITY (*t*) = ( ).

3. — Access model of a data base

The access model aims at describing in greater detail the individual model of an information system by emphasizing the logical accesses to data that are required by applications.

A general description of the whole model is to be found in ([11], [12], [14]).

This section confines itself to that part of the model which is actually implemented (1). This subset and the programming system associated with it are described in greater detail in [15].

A model of this type is a trade-off between the more general relational models and the more "rigid" conventional DBMS.

(1) The target system chosen for the first implementation is the Siemen's SESAM System. It should be noted, however, that the model has not been designed for a particular target system, and that it allows the use of DBMS such as CODASYL, IMS, SOCRATE and others.

Regarded as a complement of the conceptual model, it may be considered as a compromise between certain semantic models (such as in [16]) and logical implementation models (such as in [17]).

3.1. — *Elements of the access model*

The model provides the user with a description of the units of information and accesses between them. The accesses assume the form of access relations defined on two objects, each of which corresponds to one unit of information. At a given time, the data base consists of a set of objects and a set of relations.

N. B. — The examples mentioned here refer to the graph described at the end of the section.

The objects.

An object corresponds to an homogeneous class of information that is significant for one or several applications. Thus, within the framework of a firm, the following objects will be defined : "product", "order", "customer", "quantity", "product number", "address", ... Each element of that class is a *realization* of the object. At any time, to an object are associated its name and the set of its realizations; from now on we will refer to that set by means of the name of the object. The set of the objects of the D. B. is partitioned into three subsets : the elementary objects, the complex objects and the root object. Before discussing the objects, we will explain further what we mean by "access relation".

The access relations.

Let R be an access relation defined on objects A and B; A is called the origin object of R and B its target object.

At any time it is possible to associate with R a set of pairs (*a*, *b*) such that *a* belongs to A and *b* to B. These pairs are the *occurrences* of the relation. If (*a*, *b*) is an occurrence of R, then a privileged access path exists to the realization *b* from the realization *a*. R is said to allow *b* to be accessed from *a*.

EXAMPLE :

employee (FIRM, PERSON) means that for one realization of FIRM it is possible to access a set of realizations of PERSON in the framework of ("by means of") the access relation employee. This structure may be vizualized by means of the oriented graph (actually a "multigraph")

$$\begin{tikzcd}\boxed{\text{FIRM}} \arrow[r, dash] & (\text{employee}) \arrow[r] & \boxed{\text{PERSON}}\end{tikzcd}$$

The name of a relation may be omitted, providing that no ambiguity arises.

Examples :

— (FIRM, PRODUCT) expresses the access from a firm to the products it manufactures.
— (CUSTOMER, NAME) expresses the access from a customer to his name.
— (NUMBER, PRODUCT) expresses the access from a number to the product corresponding to it (if the latter exists).

In terms of the graph, we will write :

$$\begin{tikzcd}\boxed{\text{CUSTOMER}} \arrow[r, dashed] & \boxed{\text{NAME}}\end{tikzcd}$$

Inverse access relation.

Let R(A, B) and S(B, A) be two access relations. We will denote
by R(*a*) the image of $a \in$ A in B by R
and by S(*b*) the image of $b \in$ B in A by S.

If R and S are said to be the inverse of each other, then, at any time

$$b \in \mathrm{R}(a) \Leftrightarrow a \in \mathrm{S}(b)$$ (1)

Examples :

— *employee* (FIRM, PERSON) and *employer* (PERSON, FIRM) are the inverse of each other.
— as are (CUSTOMER, NAME) and (NAME, CUSTOMER).

N. B. — Any relation that is the inverse of itself is said to be symetrical; e. g. *spouse* (PERSON, PERSON).

Relation with an ordered target.

The set of the occurrences of a relation R(A, B) is always partly ordered. Indeed, the image of any realization of A is ordered, the order being the order of access to the various elements of the image.

(1) Though an access relation may not possess an inverse relation, it can be compared with the "access function" in [16]. Nevertheless, the consistency rules associated with it are different.

However, while certain orders are meaningless, depending merely on the implementation techniques, other orders are significant; for instance, a person's children will be accessed in order of increasing age.

The relation R will be said to be a relation with an ordered target if the order of access is explicitly specified in the description of the relation R.

If an order is specified, it will be possible to refer to it by means of the ordinal in the D. M. L.

Characteristics of a relation.

Any access relation R(A, B) is characterized by 4 integers *i-j*, *k-l* indicating that at any time :

— *n* realizations of B may be accessed from any $a \in A$ by means of R, with $i \leqslant n \leqslant j$

— any $b \in B$ may be accessed from *m* realizations of A by means of R, with $k \leqslant m \leqslant l$.

So, *spouse* is characterized by	0-1, 0-1
employee	0-∞, 0-10
(ORDER, CUSTOMER)	1-1, 0-∞
(CUSTOMER, NUMBER)	1-1, 0-1

If R is characterized by *m-n*, *o-p* and if S is the inverse of R, then S is characterized by *o-p*, *m-n*.

N. B. (1) — a relation with a compulsory member is characterized by 0-*j*, 1-*l*

— a relation with a weak member is characterized by 0-*j*, 0-*l*.

(2) We must always be very cautious when representing a real system by means of an information system described by its access model; for instance, since we know that a person has from 0 to 30 children and that a person is always the child of two other persons, persons, we could feel tempted to define :

child (PERSON, PERSON) : 0-30, 2-2

but such a relation would make it impossible to create realizations of PERSON corresponding to the ancestors of the population described. Hence the definition :

child (PERSON, PERSON) : 0-30, 0-2

Dynamics of a relation.

In process of time, the set of the occurrence of a relation will be altered by the addition of new elements, or the removal of existing elements. These alterations must be performed in accordance with the characteristics *i-j, k-l.*

The elementary objects (EO).

An elementary object is defined by its name, the set of its realizations and the set of those relations in which it takes part. A realization of an elementary object is a value that may be manipulated (read, written, used in a calculation...).

Example :
— "SMITH" is a realization of NAME.
— 26 is a realization of AGE.
— "10 Downing Street, London" is a realization of RESIDENCE.

The realizations of certain elementary objects cannot be broken down (AGE, PRICE, NAME...) whereas those of other elementary objects result from the concatenation of several significant values. The last example mentioned above shows that the decomposition :

10, Downing Street London

remains significant.

The elementary object RESIDENCE is said to be *made up* of the elementary objects : NUMBER, STREET, TOWN. Thus a distinction should be made among elementary objects between simple and compound objects. These relations of composition assume the form of the access relations characterized by *j-j, l-l*; hence the following graph :

$$\begin{tikzcd}
 & \boxed{\text{RESIDENCE}} \arrow[dl] \arrow[d] \arrow[dr] & \\
\boxed{\text{NUMBER}} & \boxed{\text{STREET}} & \boxed{\text{TOWN}}
\end{tikzcd}$$

As to the set of the realizations of an elementary object, it will not be altered during the life of the D. B.

Thus,

— *AGE.* described as an *integer between 0 and 150,* possesses a set of 151 realizations :

{0, 1, 2, ..., 149, 150}

— NAME, described as *a string of at most 10 alphabetical characters*, possesses a set of $26 + 26^2 + 26^3 + ... + 26^{10}$ realizations :

{A, B, C, ..., AA, AB, AC, ..., ZZZZZZZZZZ}.

— SEX possesses two realizations {M, F}.

"Unknown" is also a value of an EO.

The set of realizations of a compound EO is the cartesian product of the sets of realizations of the EO composing it.

The complex objects (CO).

A complex object is defined by its name, by the set of those relations in which it takes part and by the operations that may be performed on its realizations.

For instance, there exists an object LINE, taking part in the relations (ORDER, LINE), (LINE, NUMBER), (LINE, Q), (LINE, PRODUCT) and for which it is possible to create or remove realizations. The set of realizations associated with a complex object will usually be altered in process of time, by means of the operations of creating or removing realizations.

Whereas a realization of an EO may be "perceived" merely by knowing the value corresponding to it, a realization of a CO may be perceived only by knowing realizations of EO directly — or not — related to that CO. E.g. each realization of the CO LINE will be known by the number of the order possessing that line, the number of the line, the quantity and number of the product referred to by that line, etc. However, a subset of this information is usually sufficient to identity a realization of a CO. Two cases may be distinguished :

— the CO takes part in a relation (or composition of relations) characterized by *i-j, n-1*; in this case the identification of a realization of the other domain of the relation may identify a realization of that CO.

Examples :

— $R \equiv$ (PRODUCT, NUMBER) is characterized by 1-1, 0-1; a value n of NUMBER is sufficient for identifying the realization p of PRODUCT such that $(p, n) \in R$.

— the relation *spouse* (PERSON, PERSON) is also an identifying relation.

— if such is not the case, it will be possible to specify a list of relations in which the CO takes part and such that the identification of a realization of the other domain of each one of them makes it possible to identify a realization of that CO.

E.g., if we state that R ≡ (ORDER, LINE) and S ≡ (LINE, ORDER) is the identifying list of the CO LINE, then, to one realization of ORDER and one realization of NUMBER corresponds at most one realization of LINE.

i.e. in formal terms :

$$\forall e \in \text{ORDER}, \forall n \in \text{NUMBER}, \ | R(e) \cap S^{-1}(n) | \leqslant 1.$$

It should be noted, on the other hand, that a realization of a CO may be identified by its rank in the image by any relation (with an ordered target) of a realization of origin object of that relation.

Dynamics of the complex objects.

The creation and removal of realizations of CO must be performed in accordance with the characteristics of the relations. E. g., removing a realization of ORDER will automatically result in the removal of the realizations of LINE associated with it. On the other hand, creating a realization of LINE will necessitate first identifying :

— a realization of ORDER
— a realization of Q
— a realization of PRODUCT
— a realization of NUMBER

The root object.

A simple realization, namely the D. B. itself, is associated with the root object. From that object certain CO are accessed by relations corresponding to a sequential access.

The graph.

The structure of data, considered in terms of objects and relations, may be represented by an oriented multigraph whose vertices are the objects and whose arcs are the access relations.

Building a graph of structure obeys a set of rules designed to obtain the description of a feasible and coherent D. B. (see [15]).

3.2. — *The data manipulation language (see* [11], [13], [14]*)*

The language makes it possible to describe collections of realizations of objects and to command the performance of operations on these collections. The collections are described by specifying an access path in the graph of structure. It is worth noticing that this language has been designed in such a way as to be used both as a sub-language and as a self-containing language. Its definition will not be detailed further here for we think the following examples highlight its main features.

The conditions.

The D. M. L. makes it possible to choose those realizations of an object that satisfy a condition which is a simple criterion or a boolean expression of simple criteria.

There exist two types of criteria :

(1) "*belonging criteria*" : the realizations selected belong (do not belong) to a described set.

— Q(=10-1000)

This expression denotes those realizations of Q whose values are between 10 and 1000.

— PRODUCT (≠ FINISHED-PRODUCT).

If FINISHED-PRODUCT has first been defined as denoting the finished products, the expression denotes all the products which go into another product.

(2) "*relation criteria*" : the realizations selected are linked to a certain number of (or to certain) realizations of an object possibly satisfying a condition.

— PRODUCT (R : NUMBER (=2724))

denotes the product whose number is 2724.

R is the name of the relation from PRODUCT to NUMBER; in fact, that relation has not been given a particuler name and R corresponds to an empty string; it will then be written : PRODUCT (: NUMBER(=2724)).

— FIRM (employee : 20-50 PERSON (spouse : 0 PERSON))

denotes the firms employing from 20 to 50 unmarried persons.

— PERSON (child : 1st-3rd PERSON (: AGE(=21-150)))

denotes the persons whose first three children have attained their majority.

— LINE (* : ORDER (: DATE(=000000-010168))) (1)

denotes the lines of orders passed before January 2nd 1968.

N. B. — The sign * following the name of a relation (here an empty string) denotes the inverse of that relation even though it may not exist in the graph.

The accesses.

The D. M. L. makes it possible to command the access from each element of a described set (described, for instance, by means of a condition) to other realizations linked to it by an access relation and possibly satisfying a condition. The access allows the description of new collections.

— CUSTOMER(:NAME(=JONES))[:ALL ORDER(:DATE(=150973))...]

That expression means : for every customer whose name is JONES, access all his orders passed on 15/9/73 (by the relation without name).

— ORDER (:DATE (=00000-010168)) [:ALL LINE...]

for every order passed before 2/1/68, access all the corresponding lines.

It will be noticed that this expression is equivalent, from a "semantic" point of view, to that of example (1). It is indeed allowed to ask the same question in several equivalent ways. The optimization of algorithms must take place at this level, since the questions are not equivalent from the point of view of their execution time.

The actions.

The D. M. L. makes it possible to specify the operations to be performed on each element of a described collection : suppress (S), print (P), ...; it also makes it possible to add realizations, to create and remove occurrences of relations.

Given certain rules to be obeyed, sequences in the host-language may be considered as actions.

— ORDER (:NO (=*x*)) [:ALL LINE [:ALL PRODUCT [:NAME P]]]

means : print the name of all products related to order number *x*.

— ORDER (:NO (=*x*)) [:ALL LINE [:NUMBER P]; S]

means : for order number *x*, access its lines, print their number, then remove those lines.

— PERSON (:NO (=*x*)) [C spouse : PERSON (:NO (=*y*))]

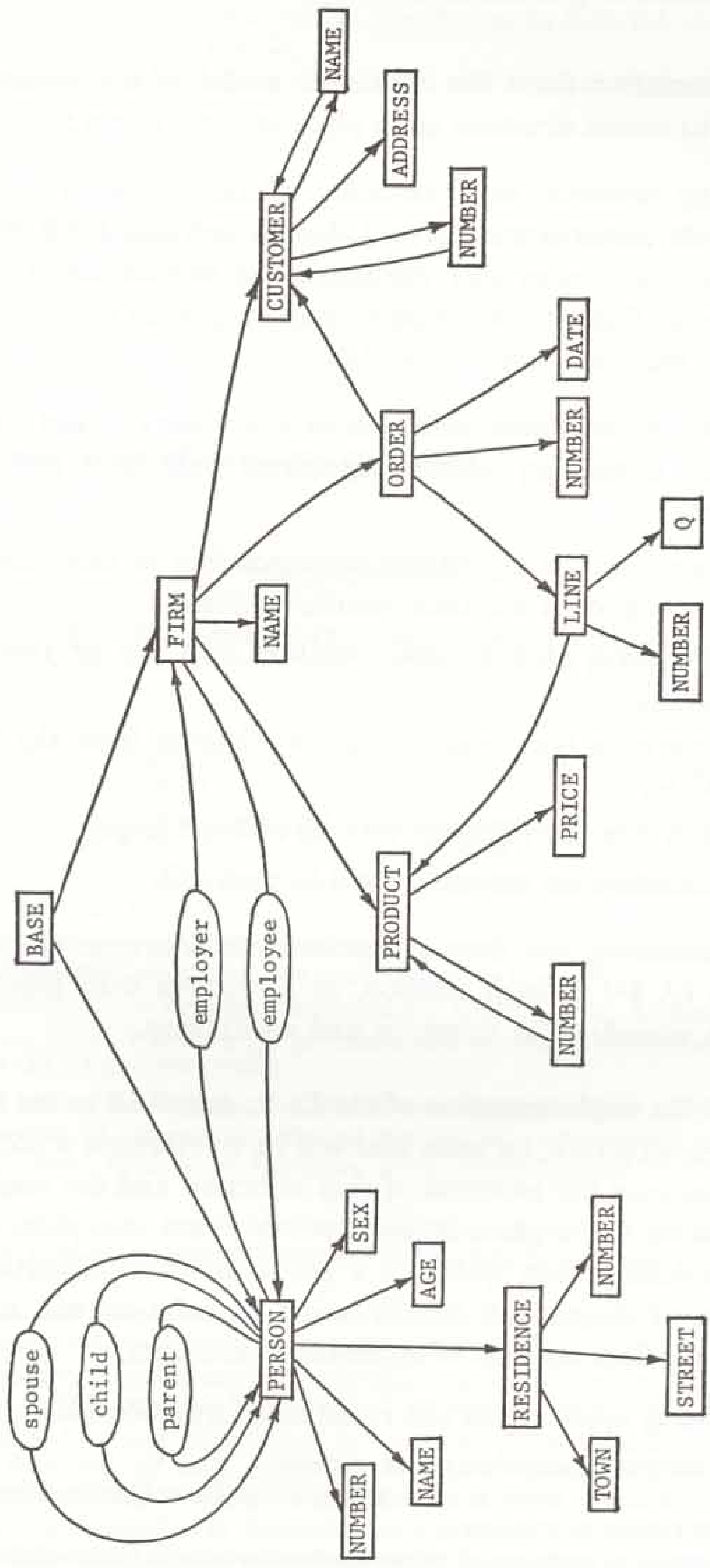
A relation "spouse" is created between person no. *x* and person no. *y*.

Other concepts.

The D. M. L. also allows the constitution and manipulation of temporary collections, the passing of values of EOs from and to zones reserved in the host-language as well as simultaneous work on several D. B. (e. g. creation of a D. B. from two others).

At the present time, a compiler for a subset of this language (to which certain algorithmic mechanisms have been added) has been completed (see [14]).

3.3. — *Example of a graph*



4. — Transition from the conceptual model to the access model and implementation of the access model

The transition from the individual model of the collection of data to its access structure takes place in three stages :

(*a*) Any semantic many-to-many relation between A and B owning attributes is transformed into an individual AB endowed with those attributes and related to the individuals A and B. We note, in fact, that all implementation systems create a special "record" with such *intersection data.*

(*b*) To an individual corresponds a complex object allowing access to elementary objects associated with that individual's attributes.

By examining the algorithms corresponding to each current or future application it will then become possible:

— to transform each semantic relation into one or two access relations;

— to determine the necessary access relations from the EOs to the COs;

— to determine the relations with an ordered target;

— to determine the actions related to each CO.

(*c*) Examining the data properties will determine the characteristics *i-j*, *k-l* of each relation, in agreement with typology of relation according to existence and cardinality.

As to the implementation of the D. B. described in the form of an access structure, its main aim will be to establish a correspondence between the elements of that structure and the tools made available by the implementation system. From that point of view each implementation system is a particular case. Nevertheless, a problem of choice will usually arise; its solution will be made easier by a finer analysis of applications and data.

Analysing applications will result in information such as

— the frequency of use of an access relation;

— the realization of removal and addition of complex objects realizations and of occurrences of relations;

— the method of exploitation of the application using a given relation.

Analyzing data will determine information such as :

— the probability that a realization of A will be linked by R to *n* realizations of B (e.g. though *child* is a relation characterized by 0-30, 0-2, the average value of | child (*a*) | is only 2);
— the probability that a realization of a CO will be linked to the realization *V* of an EO;
— the average length of the values of an EO;
— etc.

This information will also be used for optimizing the access paths corresponding to one algorithm.

Obviously, if a fairly performing D. B. is to be built, it will be impossible to proceed through the two steps outlined above independently, and studying the implementation will bring about the reorganization of the access structure.

BIBLIOGRAPHY

1. ABRIAL, J. R. — Projet SOCRATE : données, mémoires et utilisateurs d'une banque de données. Congrès AFCET/IRIA Banques de données (1971).
2. BACHMAN, C. — The programmer as a navigator (ACM annual conference, Atlanta (1973).
3 CARNAP. — Introduction to symbolic logic and its application. Dover publication Inc. (1958).
4. CODD, E. F. — A relational model of data for large shared data banks. Communication of ACM, Vol. 13, No. 6 (June 1970).
5. CODD, E. F. — Further normalization of the data base relational model. IBM Research Laboratory, San José, California (August 1971).
6. CODD, E. F. — A data base sublanguage founded on the relational calculus. IBM Research Laboratory, San José, California (July 1971).
7. KLEENE, S. C. — Logique mathématique. Collection U, Armand Colin.
8. PICARD, C. — Eléments de la théorie des graphes. Publication CNRS No. AMT/25.11.8.IBI (1969).
9. TARSKI, A. — Introduction à la logique. Gauthier-Villars (1969).
10. DEHENEFFE, Cl., HENNEBERT, H. and PAULUS, W. — Relational model for a data base. Proc. IFIP Congress, 1974, North-Holland Publish. Co., 1022-1025 (1974).
11. HAINAUT, J.-L. — A Semantic formal model of an implemented data base and its data manipulation language. Publication de l'Institut d'Informatique (July 1973).
12. Présentation et spécifications du modèle d'accès. Documents internes de l'Institut d'Informatique (15 mai 1973, 22 juin 1973, 22 janvier 1974).
13. Présentation et spécifications du langage rigide. Documents internes de l'Institut d'Informatique (15 mai 1973, 10 septembre 1973, février 1974).
14. HAINAUT, J.-L. and LECHARLIER, B. — An extensible semantic model of Data Base and its Data Language. Proc. IFIP Congress 1974, North-Holland Publish, Co., 1026-1030 (1974).
15. Système de conception et d'exploitation d'une base de données. Partie I : Le modèle d'accès. Introduction au système de programmation; Partie II :

Manuel de référence; Partie III : Description d'une base de données d'essai. Publication de l'Institut d'Informatique (novembre 1974).

16. Abrial, J. R. — Data Semantics, in Data Base Management. Proc. IFIP WC 1974, North-Holland Publish. Co., 1-60 (1974).

17. Senko, M. E., Altman, E. B., Astrahan, M. M. and Fehder, P. L. — Data structures and accessing in data base systems. *IBM Systems Journal*, Vol. 12, No. 1 (1973).