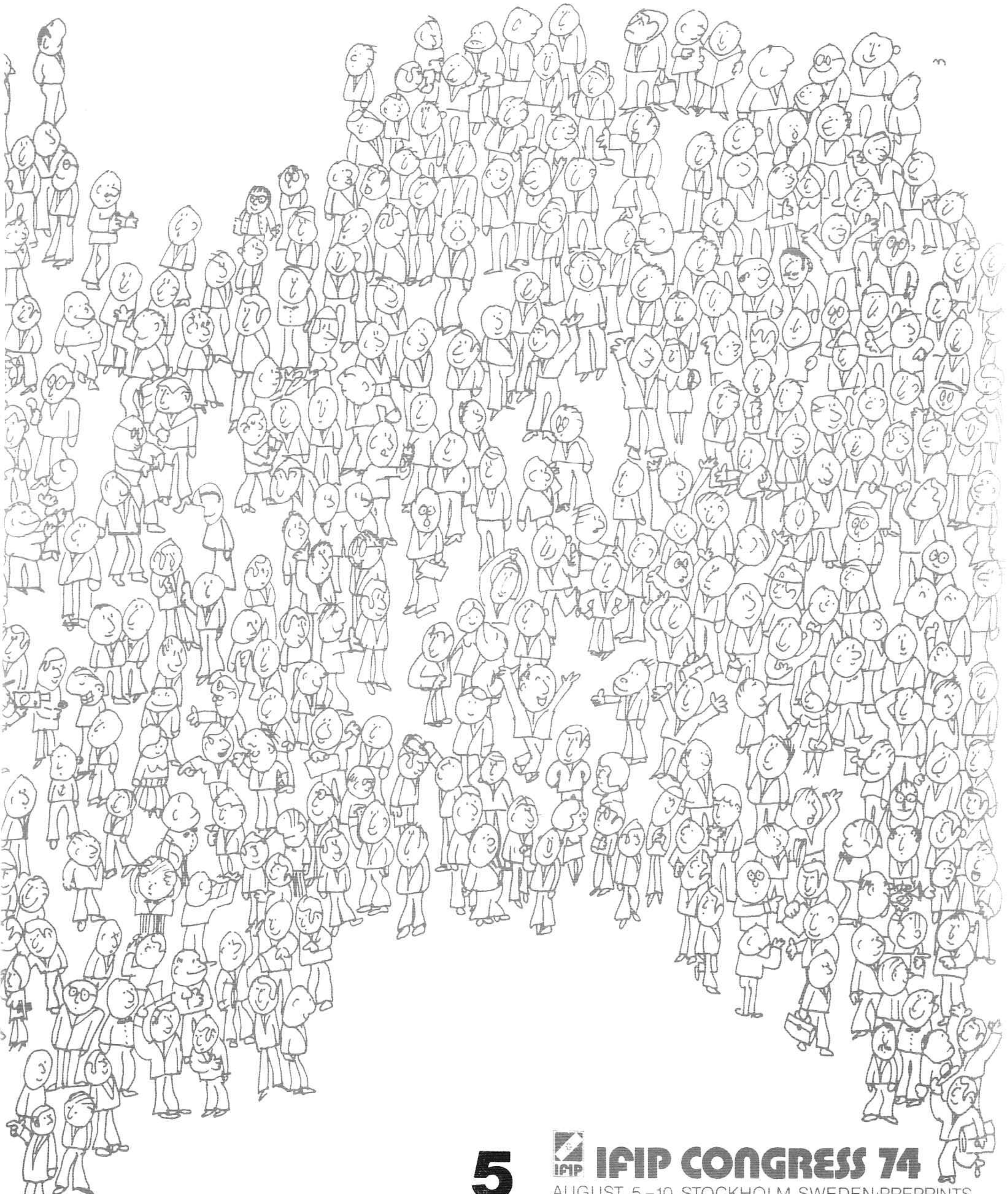
INFORMATION PROCESSING 74

SYSTEMS FOR MANAGEMENT AND ADMINISTRATION



5

IFIP CONGRESS 74

AUGUST 5 – 10 STOCKHOLM SWEDEN·PREPRINTS

INFORMATION PROCESSING 74 – NORTH-HOLLAND PUBLISHING COMPANY (1974)

AN EXTENSIBLE SEMANTIC MODEL OF DATA BASE AND ITS DATA LANGUAGE*

Jean-Luc HAINAUT and Baudouin LECHARLIER

Institut d'Informatique des Facultés Universitaires Notre-Dame de la Paix
Namur, Belgium

The utilization of commercial Information Management Systems for the implementation of Data Bases is usually impaired by such problems as a poor adaption to the complex semantic structures and the fact that programs are not independent from the implementation system and the implemented data structures. This paper briefly outlines, on the one hand, a semantic model of Data Bases and the data manipulation language associated with it, and, on the other hand, a data management system based on the model and the language, which makes it possible to utilize the commercial Information Management Systems while limiting as much as possible the above mentioned drawbacks.

1. INTRODUCTION

A data base is to be regarded as a computer representation, from the point of view of a class of applications, of a subset of the information describing a system of the real world ("real system").

A real system is organized in such a complex way that it can only be perceived roughly.

The subset of information that will be chosen to describe a system will offer a partial and simplified view of it, depending on the point of view from which that system is considered, and principally on the goals to be achieved ; this subset of information will be structured in order to be as well adapted as possible to the organization of a part of the real system ; at this level we will speak of the semantic structure of this information.

On the other hand, a structured set of information, if it is to be manipulated, must be represented by means of a data-processing system, namely, a Data Base System (e.g. : IMS, IDS, CODASYL, GIM, or even COBOL and PL/1).

However this DB System has only a limited power of representation compared with the semantic structure of the information describing a real system, and its ways of representation are different from those of the other systems ; these ways of representation are "data structures".

On the other hand, the data manipulation languages associated with these DB Systems, besides the fact that they work on different types of data structure, also have different syntaxes and rules of utilization.

As to the application programs related to a DB, they are usually bound to work on a given structure implemented in a given DB System.

As a result of the above analysis, the implementation of a DB in one of the commercial systems dictates a certain number of constraints of which we will emphasize the following :
- the semantic structure of a set of information must often be limited or simulated, by means of guiles, before being represented by the DB System.
- a single semantic structure will be translated into different data structures, according to the DB System chosen for the implementation.
- since a DB structure is often a compromise between the requirements of different applications, this structure is often poorly adapted to some of these applications or to new ones ; consequently, there is a risk that the programs executing these applications become artificially complex. (IMS and CODASYL, to mention only a few, solve this problem very partially)
- access to the data is commanded in DML by the description of the search algorithm and not by a "natural" description of these data, independent from the implemented data structures.
- the choice of a new implementation system, and even the slightest modification in the implemented structure, usually involves rewriting all the programs working on the modified DB, even though the semantic structure of the information has not been modified.
- the protection of data is not always sufficient.

Our purpose is to devise a system allowing easy utilization by several users of one or several DB, implemented in one or several conventional DB Systems, while eliminating, as much as possible, the above mentioned drawbacks.

It may be useful to compare our work with that done by some of those who have tackled similar problems :
- whereas we try to solve the problem of data independence by correspondences between graphs, Codd looked for a solution in a particular structure of the representation of information - n-ary relations defined on simple domains [1] - together with the definition of a predicative language [2]. We think, however, that our approach, based on the analysis of the semantic structure of the information, leads to a more natural model, especially for complex situations ; on the other hand, our system defines ways of allowing the expression and extension of an existing DB structure.
- working on bases similar to those of Codd, Strnad [3], among others, took also into account the problems of implementation but without allowing the use of existing DB, which was one of our aims.
- although we have not tackled the formalization and implementation of data structures, our work may be related to that of Senko, Altman, Astrahan and Fehder [4] : it should be noticed, however, that the adoption, in our system, of a single formalism for the different levels corresponding to the levels "Entity Set Model" and "String Model" in [4], imparts a great simplicity to the expression of correspondences between graphs and to the

*Within the framework of the national program for the advancement of research in Computer Science, sponsored by the Commission Interministérielle de la Politique Scientifique.

extension of these graphs.
- in comparison to the various studies on data graph models (e.g. Earley in [5]), we will point out that the limited field on which we have focused has led us to consider very particular structure properties, which are missing, as far as we know, in the models mentioned above.

2. WHICH SEMANTIC STRUCTURES SHALL WE REPRESENT ?

Let us consider the real system constituted by a firm. It is described by a set of information concerning "objects" such as : PERSON, PRODUCT, CUSTOMER, NAME, PRICE, NUMBER. We would like to represent the semantic structure of this information in a way that would correspond, among others, to the following properties and relations between objects :
- a person may owns residences, but we are interested in these only if they belong to a person.
- a person may be the spouse of only one other person.
- a person is necessarily either a clerk or a worker (but not both) in a department.
- a product is necessarily manufactured by a machine or held in stock, or both ; thus it disappears when it is neither manufactured nor held in stock any longer.
- if a person is a clerk or a worker in a department, the department is then the employer of that person.
- a person always has a name and a number, and possibly a maiden name or military status (but not both), an age ...
- to a number corresponds at most one person, whereas several persons may correspond to a name.
- a product may be added or suppressed ; a person may marry, divorce or work in another department.

3. A MODEL OF THE SEMANTIC STRUCTURE OF A SET OF INFORMATION

We have chosen to represent this structure by a directed multigraph. The nodes, which we call OBJECTS, correspond to semantic "beings" meaningful for the foreseen applications ; to one object corresponds a set of realizations. The arcs, that we call RELATIONS, denote that a certain number of realizations of the target object may be associated with one realization of the origin object. For instance* the relation "worker", from DEPARTMENT to PERSON, indicates that with any department may be associated the persons that are workers in it. If a person P1 is a worker in the department D1, then (D1, P1) constitutes an occurrence of the relation "worker".

We will now very briefly describe the main features and properties of the elements of the graph.

3.1 The relations

(i) Any relation bears a name which is not necessarily unique in the graph. From one object to another, there cannot exist two relations bearing the same name. For simplicity, the name of relations "owns" and "belongs to" may be implicit in the graph. (e.g. : the relation "owns" from PERSON to NAME bears no name in the graph.)

(ii) We will also characterize a relation R from A to B by the minimum (I) and maximum (J) numbers of realizations of B that may be associated with a realization of A, as well as by the minimum (K) and maximum (L) numbers of realizations of A with which a realization of B may be associated. For instance, a department employs less than 1000 clerks, but a person in necessarily a clerk or a worker in a single department ; the relation "clerk" is thus written as follows : 0-1000, 0-1 ; similarly, "spouse" : 0-1, 0-1 and "owns" (from PERSON to NAME) : 1-1, 0-

(iii) The presence of inverse groups of relations will be indicated too. For instance, the group (" worker", "clerk") is inverse of "employer" ; "owns" from PERSON to NAME of "belongs to" from NAME to PERSON.
Some relations (groups of a single element) are symmetrical. (e.g. : "spouse")

(iv) Moreover, it is possible to suppress an occurrence of a relation R from A to B, (this means that the realization of B may not be associated any longer by R with the realization of A.) as well as to create an occurrence of R from a realization of A and a realization of B. (e.g. : P1 and P2 marry or divorce.)

3.2. The objects

Any object bears a name chich is not necessarily unique in the graph. We will make a distinction between elementary and complex objects.

(i) Elementary object

Each of its realizations is constituted by an elementary value identifying it. An elementary object is defined by its name, the relations in which it takes part, and the description of its domain of values. (e.g. : AGE, belongs to PERSON, integer between 0 and 150 ; SEX, belongs to PERSON, value : male or female.) At this level, it is not possible to modify the set of the realizations of an elementary object, since it is considered that all the possible realizations exist at any time.

(ii) Complex object

A complex object is defined by its name, the operations that may be performed on its realizations, and the relations in which it takes part. All the relations concerning a complex object have, of course, different functions in connection with that object : a realization of the object must be part of one or several occurrences of certain relations, whereas its taking part in some others is optional ; on the other hand, it may be bound to be part of at least one occurrence of one relation of a given group, or else it may be allowed to take part in only one occurrence of a relation of a given group. (e.g. :
- a PERSON necessarily owns a NAME and a NUMBER, is necessarily either "clerk" or "worker", but not both, in a DEPARTMENT ; may own several RESIDENCES, an AGE, a SEX and a "spouse", together with a MAIDEN NAME or MILITARY STATUS, but not both.
- a PRODUCT must own a NUMBER, and belong to a MACHINE or a STOCK, or both ; it may have a NAME.)

If we adopt the signs V (inclusive OR), $\wedge$ (AND) and + (exclusive OR) and if we decide to denote by "R" the presence of an occurrence of R concerning a realization of the complex object, we will write (see fig. 1) :
PERSON : compulsory relations : $R10 \wedge R12 \wedge$(worker + clerk) ; facultative relations : R19, R17, R18, spouse, (R15 + R16).
PRODUCT : compulsory relations : $R25 \wedge (R14 \vee R21)$; facultative relations : R23, R26.

These expressions constitute the conditions of existence of the complex object realizations.

N.B. : When a relation has an inverse, the latter is not mentioned in the expression since their occurrences are linked together.

*Examples in this paper refer to the graph of fig. 1.

It must be possible to identify a realization of a complex object. For certain objects the means of identification may be deduced from a simple examination of the features of the relations the target or origin of which is that object. If a realization of an object A is the target of an occurrence of a relation such that J=1, then the identification of one realization of the origin object identifies at most one realization of the object A ; such is the case with a relation whose A is the origin and such that L=1. (e.g. : if R10 ≡ 1-1, 0-1, then one value of NUMBER allows the identification of, at most, one PERSON ; the same being applicable to "spouse"

Other objects, however, cannot be identified by means of only one relation ; it will be specified, then, which relations are necessary (e.g. : R29 and R27 for LINE).

It is now possible to define the notion of realization of a complex object : it is made up of the identification of all the complex or elementary objects forming an occurrence of relation with that object.

It should be noted that there may be a variety of operations which could be performed on the realizations of an object, e.g. : hiring a person, suppressing a product, editing an order line, recording a new customer, etc.

Two operations, however, are common to all complex objects :
- adding a realization, which is accepted by the system only if the occurrences of relation created for that realization satisfy the conditions of existence associated with the object.
- suppressing a realization, which may lead to other successive suppressions (e.g. : if a machine is suppressed, and if the manufactured products are not held in stock, these disappear automatically).

4. THE DATA LANGUAGE (DL)

In this section we will merely describe the main features of the DL by means of examples, without explaining its precise formalism. We add that the DL has been designed to be used as a "self-contained" language or "sublanguage", and that we have chosen its present syntax, which is likely to undergo further changes, only for simplicity and conciseness.

4.1 The conditions

The DL makes it possible to choose those realizations of an object that satisfy a condition that is a simple citerion or a boolean expression of simple criteria. Two types of simple criteria are accepted :

(i) The realizations belong (do not belong) to a described set :
- ALL AGE (= 21 - 45) (1)
  (that expression denotes the realizations of AGE whose values are comprised between 21 and 45)
- ALL PERSON (≠ WOMAN) (2)
  (if WOMAN has been defined as the set of persons whose SEX is female, that expression denotes men)

(ii) The realizations are linked to a certain number of realizations of an object possibly satisfying a condition.
- ALL PERSON (↓ : NUMBER (= 2600)) (3)
  (denotes the persons whose number is 2600)
  N.B. : the sign ↓ stands for "owns" when the corresponding arc is not given a name in the graph.
- ALL DEPARTMENT (worker : 20-30 PERSON (spouse : 0 PERSON)) (4)
  (denotes the departments employing from 20 to 30 workers having no spouse)
- ALL RESIDENCE (↓☆ : PERSON (↓ : NAME (= Smith))) (5)
  (denotes all the residences of all the persons whose name is "Smith")
  N.B. : the sign ☆ following a relation name means that the inverse of the relation is denoted, even though it does not exist in the graph.

As to the boolean expressions, they are under the form :
ALL PERSON (↓ : AGE (= 21)) AND (↓ : NAME (= D)) OR (= WOMEN)) (6)

4.2 The accesses

The DL makes it possible, for each realization of an object, possibly satisfying a condition, to choose those realizations of an object that are linked to it by a relation in the graph and that possibly satisfy a condition.
- ALL DEPARTMENT (↓ : NAME (= SALE)) [worker : ALL PERSON (= WOMAN)] (7)
  (for each department whose name is "SALE", choose among the workers those who are women)
- ALL PERSON (↓ : NAME (= Smith)) [↓ : ALL RESIDENCE] (8)
  (for all the persons whose name is Smith, choose all the residences)

It should be noticed that expression (8) is equivalent to expression (5), which constitutes a very important feature of the language : the sets of realizations are described in the way that suits the user best. Obviously, for a complex description a great variety of equivalent expressions may exist.

4.3 The actions

(i) When a set is described, it is possible to command operations on that set (e.g. : PRINT, SUPPRESS, HIRE, DISMISS...)
- ALL MACHINE (= AUTOMATIC) [↓ :ALL PRODUCT [↓ : NAME PRINT]] (9)
  (for all automatic machines, find the product and write the name of each of them)
- ALL ORDER (↓ : NUMBER (= X)) [↓ : ALL LINE [↓ : NUMBER PRINT] SUPPRESS] (10)
  (for order number X, find the lines, print their number, then suppress those lines)

(ii) The DL also makes it possible to add a realization of an object, to create, suppress, and modify a relation.
- ADD PERSON ((↓ : NUMBER (= X)) AND (↓ : NAME (= Y)) AND (worker☆ = DEPARTM. (...))) (11)
  (one verifies that the given information satisfies the conditions of existence of the object PERSON)
- PERSON (↓ : NUMBER (= X)) [C spouse : PERSON (↓ : NUMBER (= Y))] (12)
  (describes the marriage of person number X with person number Y)

5. GENERALIZATION OF THE MODEL AND DATA LANGUAGE

An important feature of the model is that any graph describing a semantic structure may in turn be completely described by means of a graph of structure that we will call a "fundamental graph" (see fig.2). Consequently, it is possible to consult, modify and extend any graph of structure by means of the DL, given certain conditions that will be examined in the following section.

6. ORGANIZATION OF THE SYSTEM OF GRAPHS OF A DATA BASE

Let there be a DB implemented in a commercial system. It is possible to describe the semantic structure expressed in that DB by means of a graph (called "descriptive basic graph") which will be a faithful image of the DB and its behavior ; the actions that may be performed correspond to those of the implementation system.

However, either because of the compromises between the applications or because of the limited power of the implementation system, that semantic structure happens to be unsatisfactory compared with the real system. Therefore we will draw a second graph (called "semantic basic graph") that better corresponds to the semantic structures to be represented ; each new element will be expressed, in terms of the descriptive basic graph, by means of DL programs which will constitute the interface between those two graphs. By this means it is possible, for instance, to simulate IDS in a simple way on the basis of COBOL files, or to make the master and details of an IDS chain independent from each other. In other words, the guiles and difficulties will be dealt with by the interface and no longer by the programmer.

Sub-graphs, called "secondary graphs", will be extracted from that semantic graph, and one or several users will be entrusted with them. These users will thus only have a partial view of the DB.

A user is allowed to modify and extend his secondary graph by expressing the new elements in terms of the old ones by means of the DL. In this way it is possible to introduce into a graph new objects, relations and actions. (e.g. : to create an object HOUSEHOLD and all its relations with PERSONS in fig. 1, and to define the action : SUPPRESS a HOUSEHOLD.)

Any program written in DL on the basis of a secondary graph will be transformed by a macrogenerator into another DL program related to the descriptive basic graph. The last program will then be optimized, thanks to the knowledge of the features of the elements of the graph on the one hand, and of statistical information on the other hand ; it will at last be compiled according to the primitives of the Data Manipulation Language associated with the implementation system.

7. CONCLUSION

The model and language we have designed, together with the system of graphs, meet, as far as possible, the requirements that have been outlined in the introduction, since they allow :
- an easy description of any implemented DB (descriptive basic graph)
- an increase in the power of a given DB System as in the semantic structure of an implemented DB (semantic basic graph)
- the protection of data (by the various secondary graphs)
- the modification and extension of a given structure while respecting data protection (modification of a secondary graph)
- the possibility of writing programs describing in a natural way the data to manipulate
- some independence of the programs from a change of implementation system or from a modification of the implemented structure (secondary graphs and DL)

At present, the proposed system is only at the implementation stage ; it is already possible, however, to notice certain drawbacks, such as the restriction of relations to the single binary type, along with execution overhead, which is so much the more important as the implemented structure is simple and as the expressed semantic extensions are reduced.

We hope, however, that our system will partially solve the problems arising from the variety of the DB Systems and from their limited power. As to further developments of the system, they will concern automatic generation of DB in a commercial system from a semantic graph.

REFERENCES

[1] E.F.Codd, A Relational Model of Data for Large Shared Data Banks, Comm.ACM, 13,6,June 1970, 377-387

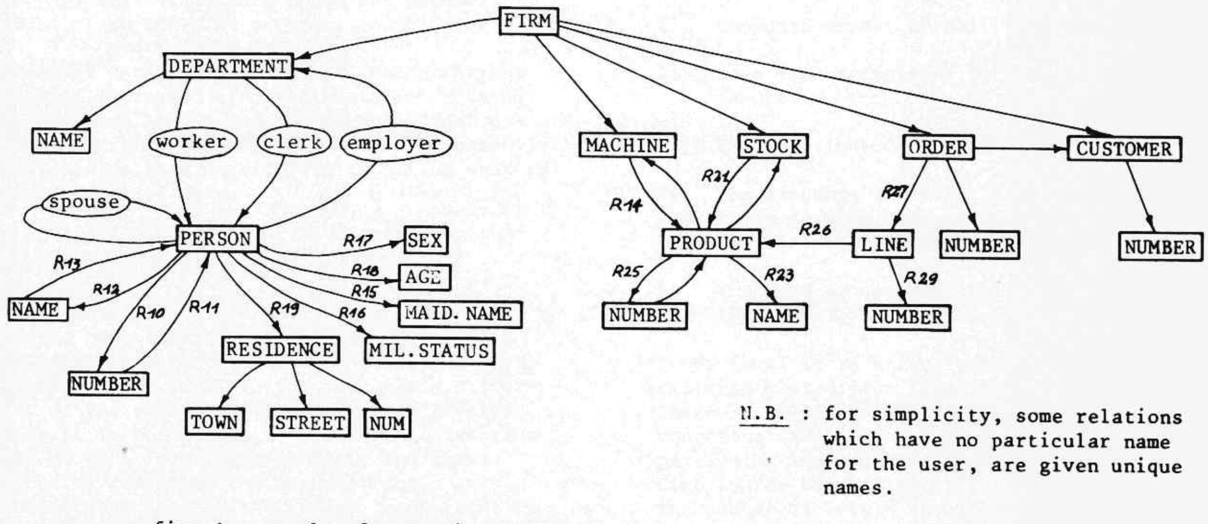
[2] E.F.Codd, A Data Base Sublanguage founded on the Relational Calculus, IBM Research Report RJ893, San Jose, California, July, 1971

[3] A.J.Strnad, The relational approach to the management of Data Bases, MAC Technical Memorandum 23, MIT, April 1971

[4] Senko, Altman, Astrahan & Fehder, Data structures and accessing in data-base systems, IBM System Journal, n° 1, 1973, 30-93

[5] J. Earley, Toward an Understanding of Data Structures, ACM, vol. 14, n° 14, October 1971, 617-627

FIGURES



N.B. : for simplicity, some relations which have no particular name for the user, are given unique names.

fig. 1 example of semantic structure.

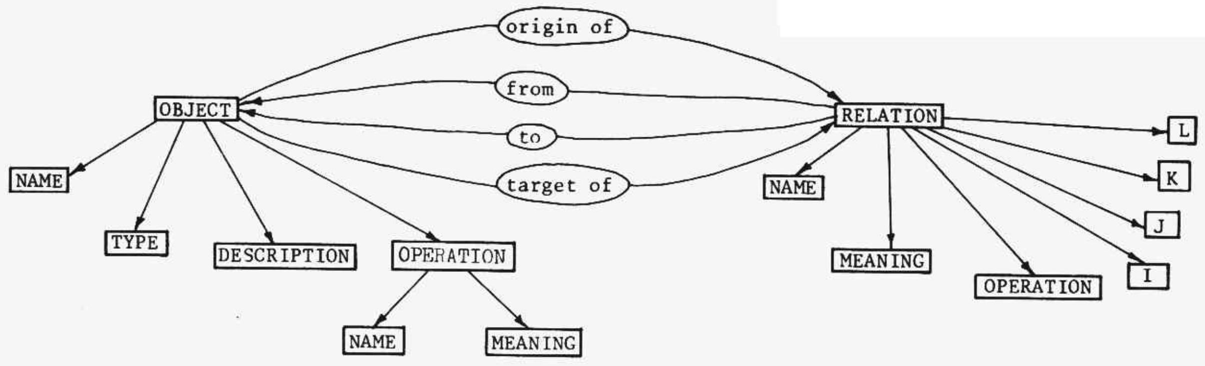


fig. 2 Partial fundamental graph.