IFIP Working Conference on
Modelling in Data Base Management Systems
Nice, France, 3-7 January 1977

organized by
IFIP Technical Committee 2, Programming
International Federation for Information Processing

NORTH-HOLLAND PUBLISHING COMPANY
AMSTERDAM • NEW YORK • OXFORD

ARCHITECTURE AND MODELS IN DATA BASE MANAGEMENT SYSTEMS

Proceedings of the IFIP Working Conference on
Modelling in Data Base Management Systems

edited by

G. M. NIJSSEN
Control Data Europe
Brussels, Belgium

1977

NORTH-HOLLAND PUBLISHING COMPANY
AMSTERDAM • NEW YORK • OXFORD

Architecture and Models in Data Base Management Systems, G.M. Nijssen,(ed.), North Holland Publishing Company, 1977

SOME TOOLS FOR DATA INDEPENDENCE IN MULTILEVEL DATA BASE SYSTEMS

J-L HAINAUT
Institut d'Informatique
Facultés Universitaires Notre-Dame de la Paix
rue Grandgagnage, 21
B - 5000 Namur
(Belgium)

INTRODUCTION

Data independence is the capability of a Data Base System to insulate the application programs from such physical characteristics of the data base as the access paths or record structures.
A multilevel architecture is generally recognized to be suitable for archieving this goal. The proposals of the ANSI/SPARC report [14], especially, define three levels : the semantic level (the conceptual schema), as stable as the "perceived real world", the access paths implementation level (the internal schema) and the level of the (semantic) views peculiar to the various users (the external schemata).

Such an architecture, however, gives rise to a difficult problem : since an application program is "working on" the conceptual schema (or an external schema, in which case the problem is the same) that does not describe the access paths, how to generate, not only an equivalent program working on the internal schema, but, if possible, a program with the minimum running time (see COLLMEYER [21]).

The objective of data independence can be achieved by methods which transform a program working on a schema into a program working on another schema (given the mapping between them) and which optimize the latter program by choosing the optimal access paths.
A second problem is tightly linked to the former : given a conceptual schema, and the description of a set of application programs, how to choose an optimal internal schema ?

Whereas the first problem has not been devoted many studies, the second one has been paid more attention.

Models making it possible to select the best file organization of a data base for a given set of queries are currently to be found in the work of numerous authors. Among the simulation models mention should be made of the work by SENKO et al. about FOREM [1], by CARDENAS [2] and by NAKAMURA et al. [3].
On the other hand, LEFKOVITZ [4], KING [5] and SHI BIN YAO [6] are to be cited in the field of analytical models providing cost equations.
Those models usually describe data bases with a very simple organization : independent files with one or several access keys [2, 5, 6] , hierarchical structure within the records [1] or between the various types of records [3].

Yet, more sophisticated structures, such as those of IDS, IMS, CODASYL (and all the systems derived from it) are difficult to describe through such models.
Moreover, performance is assessed for very simple queries (except for [3]) such as : retrieve all the records of a file the data field values of which verify a Boolean expression of criteria - which also seems rather difficult to use for calculating the cost of a real application program.

This paper outlines some results of a research on Data Independence in Data Base Systems, especially the optimization of complex application programs working on complex data bases.
The first part of this paper puts forward the main elements of the data base logical structure description model and of the queries and application programs description model.
The second part outlines a statistical model of the data base and methods for calculating the execution cost of a program.
The third part examines how to use the results achieved in two important fields :
- the determination of the optimal access path pertaining to a given program for an existing data base,
- the development of a tool for designing a data base optimal access structure for a set of application programs.

The problem of the transformation of algorithms has not been dealt with in the following pages ; some characteristics of this process can be found in appendix B.

PART I

LOGICAL DESCRIPTION OF DATA BASE STRUCTURES AND PROGRAMS

1. THE INFORMATION STRUCTURE MODEL (ISM)

One of the purposes of any data base is to give the data describing a real system (e.g. an organization) a suitable structure. The data are usually required to describe the structure of the real system. The most straightforward approach consists in giving the data a structure analogous to that of the system to be described.
Let us assume :
- that the real system is viewed as being made up of elements related with each other by means of binary relations,
- that one information unit may be associated with any element of the system and that associations between the elements may be represented by associations between the corresponding information units,
- that some elements of the system are so simple that they may be described by an elementary information (such as a price, a name or a quantity) whereas others may be described by a more complex information unit only (such as a customer, a sale, a contract),
- that elements of the same nature form classes (customers class, product numbers class), which allows us to consider various types of information units as well as various types of associations between them.

If the real system and the data describing it may be structured that way, then the following model may be used to describe those data.

Let us denote by OBJECT any type of information unit and by RELATION any type of association. Any data base is then viewed as a set of objects and a set of relations.

Let us give every object and every relation a name. It is convenient to accept the empty string as a possible name for certain relations - these will then be referred to as "nameless".

Let us finally call *realization of object* A, any A-type information unit.

1.1. Relations

Let R(A,B) be the relation whose name is R and which has been defined on objects A and B ; A is called *origin object* and B *target object*. At any moment a set of couples (a,b) such that $a \in A$ and $b \in B$ corresponds to R. Let us call those couples *occurrences of* R.
If (a,b) is an occurrence of R, then b may be accessed from a through R. That access path is logical in that an implemented access path (hashing, index, inverted file, list, ...) does not necessarily correspond to R.

Examples :

```
          employee(DEPART,PERSON)
          spouse(PERSON,PERSON)
                (DEPART,DEP#)                                (nameless)
                (NAME,PERSON)                                (nameless)
```

If R(A,B) and S(A,B) are stated to be *inverse* of each other then to each couple (a,b) of R corresponds the couple (b,a) of S and conversely.

Examples :

```
          employee(DEPART,PERSON) and employer(PERSON,DEPART)
                  (PERSON,NAME)   and         (NAME,PERSON)
```

In order to specify the properties of a relation R(A,B) the *characteristics* I_R-J_R,K_R-L_R are associated with it : these are four integers stating that at any moment
- n realizations of B may be accessed from any realization of A through R with $I_R \leqslant n \leqslant J_R$.
- any realization of B may be accessed from m realizations of A through R with $K_R \leqslant m \leqslant L_R$.

Examples :

```
          spouse(PERSON,PERSON) ≡ 0-1,0-1
        employee(DEPART,PERSON) ≡ 0-1000,1-1
           child(PERSON,PERSON) ≡ 0-20,0-2
                (PERSON,NAME)   ≡ 1-1,0-∞
                (DEPART,DEP#)   ≡ 1-1,0-1
```

A relation may be declared with an *ordered target* if the target realizations are ordered during an access from one origin realization. The "sort keys" are specified by means of the relations relating them to the target.

1.2. Objects

The set of the objects of a data base is partitioned into three subsets :

- *elementary objects* correspond to elementary information units represented by values (NAME, DEP#, ...)

- *complex objects* correspond to more complex information units that have to be represented by the associations relating them to other information units (DEPART, PERSON, ...) rather than by a simple value.

- *the root object* corresponds to a particular information unit, namely the data base itself. Each data base contains a root object that may be the origin of relations of "0-J,1-1" type leading to complex objects and corresponding to "sequential" access paths to the realizations of those objects.

In this elementary description we did not mention the "creation" and "suppression" operations. A more detailed description of ISM is to be found in [7] and [8].

1.3. Graphic representation

It is often useful to concretize the data structure by means of a diagram. Let us represent :
- an object by a box in which the name of the object is written,
- a relation by an oriented and labeled arc between two boxes.

In order to make that diagram easier to read, however, let us agree to represent the root by a double box, a complex object by a name in a box and an elementary object merely by its name. Finally, if two relations are inverse of each other and are given the same name they will be represented by a single, two-ways, arc. (see next pages).

1.4. Similar models

This model belongs to the binary relational model family ; similar formalisms have been studied by ABRIAL [9], BRACCHI and al. ([22] and previous papers), SENKO [27] and CABANES [57].

ISM, like those models, provides a description of the semantic structure of the data ; however, it is also fit, at a lower level, for actual access paths modelling.

1.5. Levels of description of a data base

The model as it has just been put forward may constitute a semantic description of the data pertaining to an organization. Given the suitable technology, that description should be stable with respect to implementation modifications of the data base (in the ANSI SPARC report [14] that level of description is referred to as the *conceptual schema*).
However, the model may as well describe the information units and access paths actually implemented in the data base. It may be decided, for instance, to represent
- a record type by a complex object,
- a data field by an elementary object,
- an access path between record types by a relation between complex objects,
- the fact that a data field belongs to a given record type by a relation from the complex object to the elementary object,
- an access by key by a relation from an elementary object to a complex object.
- a sequential access by a relation from the root to a complex object

(that level corresponds to *the internal schema* in [14]).

This may be seen to allow a uniform and concise representation of the data structure of the main existing systems : IMS, CODASYL (with the restrictions suggested in [15]), IDS (chain with only one detail record type) and other hierarchical or network systems, conventional files.

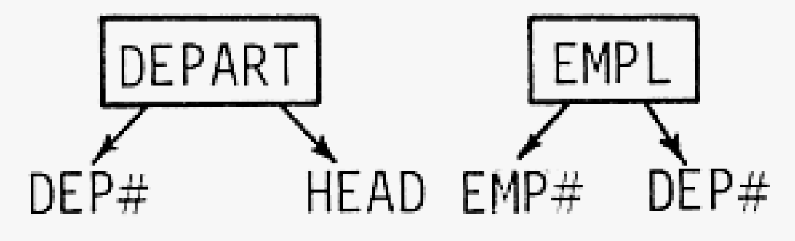
Thus the model we have put forward affords a unique mode for representing semantic and access structures.
As an example, let us examine a real sub-system comprising departments and their employees, the head of each department being an employee.

A possible relational structure (as described in [16]) of the data describing this sub-system is as follows :

```
        DEPART(DEP#,HEAD,...)
          EMPL(EMP#,DEP#,...)
```

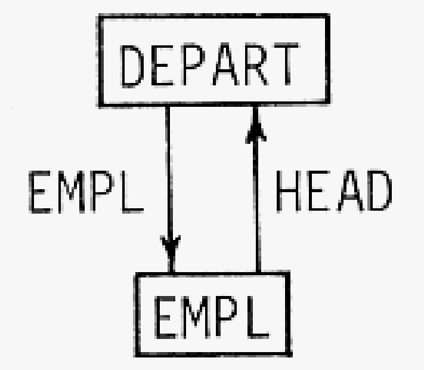
with : Proj(DEPART/HEAD) $\subset$ Proj(EMPL/EMP#)

The ISM describing that structure may be :

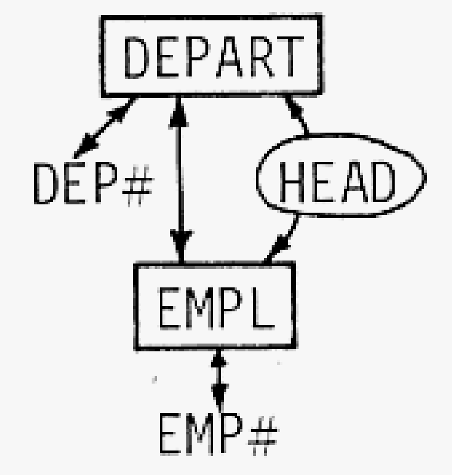


```
(DEPART,DEP#)  ≡ 1-1,0-1
(DEPART,HEAD)  ≡ 1-1,0-v
(EMPL,EMP#)    ≡ 1-1,0-1
(EMPL,DEP#)    ≡ 1-1,0-n
```

In the CODASYL system [17], these data could be implemented according to the following diagram :



for which the ISM is :



```
    (DEPART,EMPL) ≡ 0-n,1-1
head(DEPART,EMPL) ≡ 1-1,0-v
    (DEPART,DEP#) ≡ 1-1,0-1
    (EMPL,EMP#)   ≡ 1-1,0-1
```

In the statistical description of the data base and the algorithms, ISM is assumed to describe the data base implementation as in the CODASYL instance above.
The ISM as a semantic model is more completely described in [7] and [13] whereas [8], [12] (reporting on an implementation by the data base staff of the Institut d'Informatique de Namur) [25] and [58] provide a description of the ISM as an access model.

2. ACCESS ALGORITHMS DESCRIPTION LANGUAGE (ADL)

ADL makes it possible to describe the architecture of an application program or a query in a simple and structured way as far as the access to, and manipulation of, the data are concerned. ADL will be described as a programming language by means of which collections of realizations of objects will be described and operations on them will be commanded.

The collections are described by specifying selection conditions and access paths. ADL will not be defined in much detail here as we think the following examples outline its main features.

2.1. Conditions

The language makes it possible to select those realizations of an object that satisfy a condition which is a simple criterion or a Boolean expression of simple criteria.

We consider two types of criteria :

(i) "*Belonging criteria*" : the realizations selected belong (do not belong) to a described set.
- PRICE (= 50 - 150)
  describes those realizations of PRICE whose value is between 50 and 150.

- NAME (= "SMITH", "CARTER")
  describes the realizations "SMITH" and "CARTER" of NAME
- PRICE (< 100) stands for PRICE (= 0 - 99)
- PRICE (≠ 0 - 80) stands for PRICE (> 80)

Such criteria exist for complex objects as well :
- EMPL (= TC1) where TC1 is the name of a temporary collection of realizations of EMPL

but for simplicity they will not be described in more detail.

(ii) "*Relation criteria*" : the realizations selected are linked to a certain number of (or to certain) realizations of an object possibly satisfying a condition.
- DEPART (head : EMPL (= TC2))
  describes the departments whose head belongs to TC2
- DEPART (: 10 - 20 EMPL)
  describes the departments with 10 to 20 employees (a nameless relation is used here).
- DEPART ((: DEP#(<42)) & (: 1 - EMPL(: NAME = "SMITH")))
  describes the departments whose number is under 42 and with at least one employee called "SMITH" (1 - stands for 1 - ∞).
- DEPART (: ALL EMPL(head : 1 + DEPART))
  describes the departments where no employee is head of more than one department (1 + stands for 0 - 1).

Those criteria are called *cardinal relation criteria* since the *number* of target is concerned. In the *ordinal relation criteria* the test concerns the *rank* of the target (see [58]).

2.2. The actions

After such a description of a collection a list of commands to be executed for each element of the collection (⟨ A - LIST ⟩) may be specified. Among these : PRINT, S (suppress a complex object realization), A (add a complex object realization), and other actions pertaining to relations. But the most important command, aimed at describing an access algorithm, is the access command, the form of which is used in the following expression :

DEPART (: DEPT#(= 25)) [head : ALL EMPL ⟨ A - LIST ⟩] ; ...
which means : from department n° 25, access through *head* the employee who is head of it;execute for him the commands of ⟨ A - LIST ⟩. (ALL may be omitted as each department has only one head).

The following example is more general :

DEPART (C1) [: ALL EMPL ⟨ A - LIST ⟩] ;
which means : from each department satisfying C1 access all its employees and execute the commands of ⟨ A - LIST ⟩ on each of them.

The square brackets structure thus defines a loop structure ; as the commands in ⟨ A - LIST⟩ may be access commands, complex access algorithms may be described.

The following example describes the printing of the listing :

"for each machine of type 10 :
- its number
- for each of its sales (if the amount is greater than 100) :
  - its amount
  - the name of the customer."

(see appendix A)

```
ROOT [ : ALL MACH( : TYP(= 10)) [ : M# PRINT ] ;
                                [ : ALL SALES( : AM(> 100)) [ : AM PRINT ] ;
                                [ : CUST [ : NAME PRINT ]]
                                ]
     ]
```

In that program, the relation (ROOT,MACH) allows sequential access to all realizations of MACH ; the different commands of a same ⟨ A - LIST ⟩ are separated by " ; ". Other classical programming structures are allowed thanks to the IF THEN ELSE, NEXT and EXIT statements [58].

N.B. : A complete description of the language is to be found in [7] and [8]. A dialect of the language (to which certain algorithmic mechanism have been added) has been defined as a Data Manipulation Language for COBOL [12]. A second version of the compiler is now completed. It is interesting to note that Data Base languages affording the loop structure exist on the market, especially SOCRATE [11] on CII, IBM, SIEMENS and DATABASIC with IDS Data Bases on Honeywell Bull [10].

2.3. Level of description of an algorithm

Like ISM, an ADL program may be interpreted at two levels :

(i) As a logical description of data collections and of actions to be executed ; that description has been written without taking efficiency criteria into account and is thus a description at the semantic level.

(ii) As an access algorithm ; the description is at the level of the implemented access paths.

The distinction is exemplified as follows :

"print the date of the sales of the machines n° 10"

At the semantic level, one may write for instance :

ROOT [: ALL SALES(: MACH(: M#(= 10))) [: DATE PRINT]]

but if interpreted as an access algorithm, the program is inefficient ; an equivalent but less expensive program may then be written :

ROOT [: ALL MACH(: M#(= 10)) [: ALL SALES [: DATE PRINT]]]

When studying how to measure the cost of an algorithm, ADL programs will be interpreted as simple access algorithms.

Application 2 in part III will show why a distinction should be made between the two levels of description.

2.4. Conclusions

The language makes it possible to describe simple queries as well as complex application programs in a simple and natural way. By the way, it could be interesting to note that the dialect mentioned above has afforded a clear redaction of programs such as its own compiler (see [26]) and a program for interactive retrieval and updating of data.
It is obvious, however, that such a high level language can hardly describe, without programming tricks, some particular complex programs, and that some programs could have been written in a very different (and more efficient) manner in an actual data base programming language ; it is also obvious that an ALGOL program may be trickier and less efficient than the equivalent ASSEMBLER program.

The language provides access algorithms whose logical ("semantic") structure is explicit ; that property is lacking in most Data Base languages, the commands of which may be scattered "at random" in the host program. The knowledge of this structure makes it possible to calculate the expected cost of the algorithm.

PART II

STATISTICAL DESCRIPTION OF A DATA BASE AND MEASUREMENT OF THE ACCESS ALGORITHMS

A data base can be described by very simple statistics such as : for each relation R, average number of target realizations linked by R to an origin realization (e.g. : KING [5] and YAO [6]). The description can also be very comprehensive by describing, among others, correlations.
It can be shown that a precise evaluation of the expected cost of ADL algorithms is impossible with the first description, whereas the second one leads to intricate calculations.
That is the reason why an intermediate description made up of frequency functions has been chosen, in which, however, random variables are independent.

1. STATISTICAL MODEL OF AN ISM DATA BASE

As the various elements of the model are linked to those of ISM, we will examine the statistical description of a complex object, an access relation and an elementary object seriatim.

(i) A complex object (A) is described by :

- N_A : the number of its realizations,
- C_{AC}: the average cost of creating one realization,
- C_{AS}: the average cost of suppressing one realization.

(ii) A relation R(A,B) is described by :

- $F_{RO}(n)$: the frequency function of the realizations of A taking part in <u>n</u> occurrences of R exactly (if A is an elementary object, F_{RO} is constructed from $P_A(a)$),
- $F_{RT}(n)$: idem for B,
- $C_{RA}(n)$: the average cost function of accessing n target realizations of B from a realization of A through R. A linear expression $C_{R1} + C_{R2}.n$ seems to constitute a satisfactory approximation,
- C_{RC} : the average cost of creating one occurrence of R,
- C_{RS} : the average cost of suppressing one occurrence of R.

(iii) For each relation R in which an elementary object B takes part, one will specify :

- n_{BR} : the number of realizations of B that are actually present in one occurrence of R at least,
- $P_{BR}(b)$: the frequency function of the occurrences of R where $b \in B$ is present. If B takes part in one relation only, then one may write $P_B(b)$; this hypothesis will be adopted from now on.

That model, though limited, appears to be more complete than those commonly used ([1], [2], [5], [6]).
The way it has just been put forward makes it appear as a description of an existing data base (see Application 1). It may be considered another way, however :
- as the frequency functions represent properties of the data reflecting the intrinsic properties of the organization to be described,

- and as the cost functions depend on the features of the implementation technique (DBMS, access methods, hardware, ...) and on the volume of the data,

the statistical model of a future data base whose ISM has already been defined may be constructed with a relatively satisfactory approximation (see Application 2).

<u>N.B.</u> : For making the calculation more efficient, redundant data will be added to the model, such as

$$I_{ROm} = \min\ (n/F_{RO}(n) \neq 0)$$
$$J_{ROM} = \max\ (n/F_{RO}(n) \neq 0)$$
$$m_R = \sum_{k=I_R}^{J_R} k.F_{RO}(k)$$

2. CALCULATING THE COST OF AN ALGORITHM

The reader is reminded of the fact that this section considers an ISM structure as a description of the implemented access paths ; moreover, an ADL program is considered as an access algorithm (that may be an optimized form of an original ADL program - see § B.3).

The cost expressions we are going to construct refer to the total running time (CPU time + IO time) of a program only, without taking into account the space occupied in the core memory, for instance, or the interferences with other programs that would happen to run at the same time with the same data base.
Besides, in order to make this paper more concise, we will not examine the updating operations, nor certain calculations that are note involved in our example.
An important restriction should now be imposed on the algorithms. A real program comprises both commands for accessing a data base and instructions written in a conventional programming language (COBOL or PL/1) referred to as host-language.
Now an ADL program does not describe the statements in the host-language ; at the very most is it possible to take this into account by means of the ADL special OPER(k) command where k is the execution cost of that sequence.

Hence the restriction : "the actual execution of a program is controlled by the ADL access commands structure and not by the control instructions of the host-language (IF...ELSE, PERFORM...VARYING, DO...WHILE, ...)". Though the restriction may often be overcome by generalizing the ADL description to other parts of a program or by the use of the dummy condition COND(p,k) [58], we will not deal with that case in this paper.

If ADL is assumed to be a satisfactory description of the commonly used access algorithms then it is enough that for each elementary form of the language we should define its cost expression.

It should be kept in mind, however, that an ADL program is a description which has to be translated into a conventional data programming language, and to each of the elementary forms of ADL will correspond a particular execution procedure that depends on the DBMS and/or on the programmer.

E.g. : The procedure for the conditional form (k : i - $B(C_B)$) will be different depending on whether or not a targets counter exists for R in the representation of the origin.

We will choose for each form the procedure that appears to be the most efficient one in most cases.

2.1. Execution cost of the access action

We will consider the commonest form only, namely :

$$[R : \text{ALL } B(C_B) \langle \text{A-LIST} \rangle]$$

If K(1-LIST) (i.e. the execution cost of the action of ⟨A-LIST⟩ (1) for a realiza-of B) and p_{CB} (i.e. the probability that a realization of B will satisfy C_B) are known, then the cost of this access action for a realization of A, (the origin of R) is :

$$K = C^{\star}_R(C_B) + m_R . p_{CB} . K(\text{A-LIST})$$

where $C^{\star}_R(C_B)$ is the total cost of access through R and of selection by C_B.

It is interesting, here, to consider access and selection as a single action, which makes it possible to calculate the cost of complex access paths such as that by several access keys or that via indexed SETS (or CHAINS) in CODASYL (or IDS).

In the cases where the condition C_B is evaluated *after* and *independent of* the access, and if C_{RA} is linear, the cost function is :

$$C^{\star}_R(C_B) \cong C_{RA}(m_R) + m_R . K(C_B)$$

where $K(C_B)$ is the cost of evaluating C_B for a realization of B.

Hence :

$$K = C_{RA}(m_R) + m_R . (K(C_B) + p_{CB} . K(\text{A-LIST}))$$

We must then work out the expression of p_C and K(C).

2.2. Probability and cost of a condition

(i) Probability and cost of a Boolean expression of criteria $\mathfrak{B}$

The following restriction will be agreed upon : *all criteria are independent* (this restriction is still stronger than the independence hypothesis of the statistical model).
Calculating p_B is then easy.
On the contrary, calculating $K(\mathfrak{B})$ is more intricate as this cost depends on the order to be followed for the criteria evaluation. Actually we will choose the order that minimizes $K(\mathfrak{B})$ according to an optimizing algorithm that will be discussed later on (Application 1).
When that order has been chosen, let C_i be the first criterion to be evaluated. The following recursive expression is obtained :

$$K(\mathfrak{B}) = K(C_i) + p_{Ci} . K(\mathfrak{B}_{Ci=T}) + (1-p_{Ci}) . K(\mathfrak{B}_{Ci=F})$$

$$K(x) = 0$$

where x = TRUE(T) or FALSE(F)

$\mathfrak{B}_{Ci=x}$ denotes the reduced Boolean expression for $C_i = x$.

---

(1) If A_i i=1,...n are the actions specified in ⟨A-LIST⟩ then the execution cost of ⟨A-LIST⟩ is assumed to be

$$K(\text{A-LIST}) = \sum_{k=1}^{n} K(A_i) \text{ where } K(A_i) \text{ is the cost of } A_i \text{ ;}$$

however, this assumption is not always true during the optimizing process.

(ii) Probability and cost of a "belonging criterion"

If we confine ourselves to the case B(=i-j) pertaining to the elementary object B (as the other cases are more complex) we obtain :

$$p_{CB} = \sum_{b=i}^{j} P_B(b)$$

$$K(C_B) \cong 0 \qquad \text{(comparison in core memory)}$$

(iii) Probability and cost of a "relation criterion"

The results we are going to discuss involve the expression $\Pi_{CB}(k,\ell)$ representing the probability that exactly ℓ among k realizations of B will satisfy the condition C_B.
Let us define

$$n = P_{CB} . N_B$$
$$N = N_B$$
$$p = p_{CB}$$

The exact expression is yielded by the hypergeometric probability function :

$$\Pi_{CB}(k,\ell) = \frac{C^{\ell}_n \, C^{k-\ell}_{N-n}}{C^k_N}$$

Yet the binomial approximation, which is easier to evaluate, provides excellent results in most cases :

$$\Pi_{CB}(k,\ell) \cong C^{\ell}_k . p^{\ell} . (1-p)^{k-\ell}$$

We will omit the demonstration of the formulae we mention for the main criterion and for the most interesting particular instances.
These criteria are assumed to be evaluated "efficiently and without any trick" (no counter or pointer manipulation), which is in agreement with the possibilities of most DBMS's from the programmer point of view.

Let us call m the average number of realizations of B that have to be accessed in order to evaluate the condition. If the access path is independent of C_B, the cost is easily calculated :

$$K = C_{RA}(m) + m . K(C_B)$$

In the following we will use the notation :

$$\mathfrak{F}_R(k) \equiv \sum_{\ell=k}^{J_R} F_R(\ell)$$

The results for the general expression are :

(R : i-j $B(C_B)$)

$$p = \sum_{k=i}^{J_R} F_R(k) . \sum_{\ell=i}^{a} \Pi_{CB}(k,\ell) \qquad m = \sum_{k=1}^{J_R} \mathfrak{F}_R(k) . \sum_{\ell=0}^{J_R} \Pi_{CB}(k-1,\ell)$$

$$a = \inf (k,j)$$

and for the most interesting cases :

(R : TS B(C_B))

$$p = \sum_{k=0}^{J_R} F_R(k)\Pi_{CB}(k,k) \qquad m = \sum_{k=1}^{J_R} \mathfrak{J}_R(k).\Pi_{CB}(k-1,k-1)$$

(R : 1-B(C_B))

$$p = 1 - \sum_{k=0}^{J_R} F_R(k)\Pi_{CB}(k,0) \qquad m = \sum_{k=1}^{J_R} \mathfrak{J}_R(k).\Pi_{CB}(k-1,0)$$

PART III

APPLICATIONS

1. OPTIMIZATION OF THE ACCESS ALGORITHMS

The problem of the optimized access algorithm (i.e. in this paper, the access algorithm with the shortest expected running time) has already been dealt with by several authors, particularly SENKO et al. who have defined a very sophisticated ISM (the String Model) (see [18], [19], [20]). COLLMEYER describes another approach using a network ISM but devoid of any statistical model (see [21]). Neither author considers the general problem of application programs such as they may be described in ADL.

It should be noted that although a better algorithm than that we refer to as "optimal" may sometimes be found, this would require a more complete statistical model or would lead to generating sophisticated algorithms.

A complete description of the optimizing algorithm would fall beyond the scope of this paper ; it is more interesting to describe its various steps without formalizing them by applying them to a simple example.

This example is the following : (see ISM in Appendix A.)

"Select all the customers whose class is greater than 12, and for each of them :
- suppress the sales corresponding to a type B-machine, and relative to
  - either one engineer of the "s" category at least
  - or the department n° 3."

```
CUST( : CLASS(>12))
    [ : ALL SALES(( : MACH( : TYPE(=B)))
                & (( : 1 - ENG( : CAT(=S)))
                  V( : DEPT( : DEP#(=3))) ) )
                  S]
```

First step : reducing the program to the conditional form

So as to make operations uniform the algorithm operates on conditional forms (in the general version the algorithm examines more complex forms). The goal of this step is to transform any access command into a conditional form so that the new algorithm will provide the same results as the original one.

In § 2.3., Part I, the transformation of the second expression into the first one is a valid reduction.

In our example the access path through (MACH,SALES) is transformed :

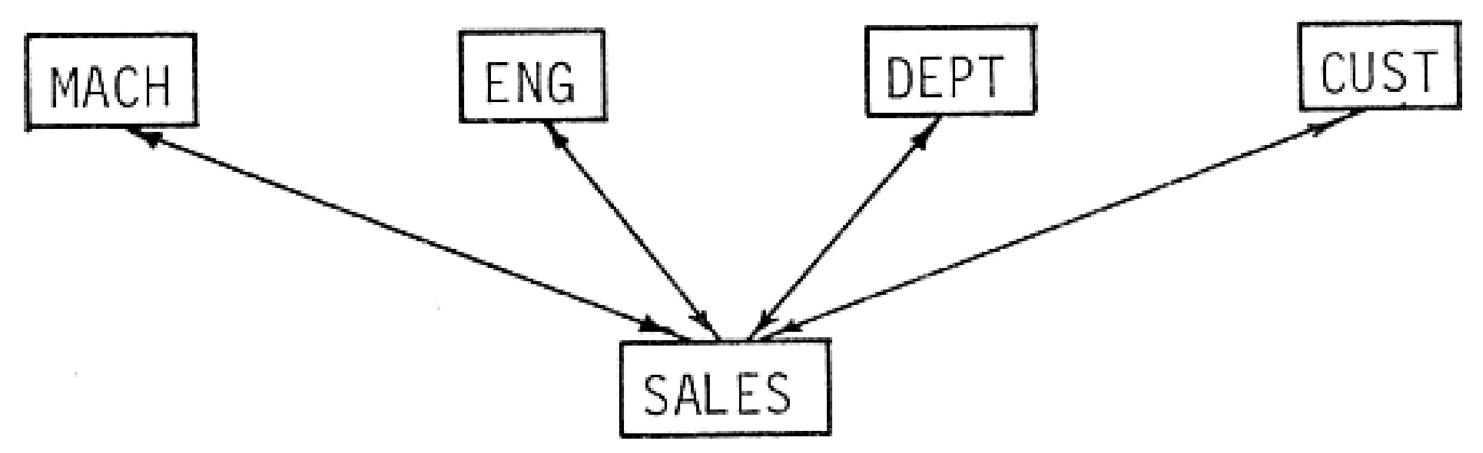
```
SALES( ( : CUST( : CLASS(>12)))
     & ( : MACH( :TYPE(=B)))
     & ( ( : 1 - ENG( : CAT(=S)))
        V( : DEPT( : DEPT#(=3))) ) ) S
```

This reduction is verified as being valid since L_1=1 (see Appendix A).

Second step : constructing the access paths tree

This operation consists in constructing the tree of those access relations between complex objects that are used in the conditional form. If similar branches arise, they will be merged.

For our example the access paths tree is :



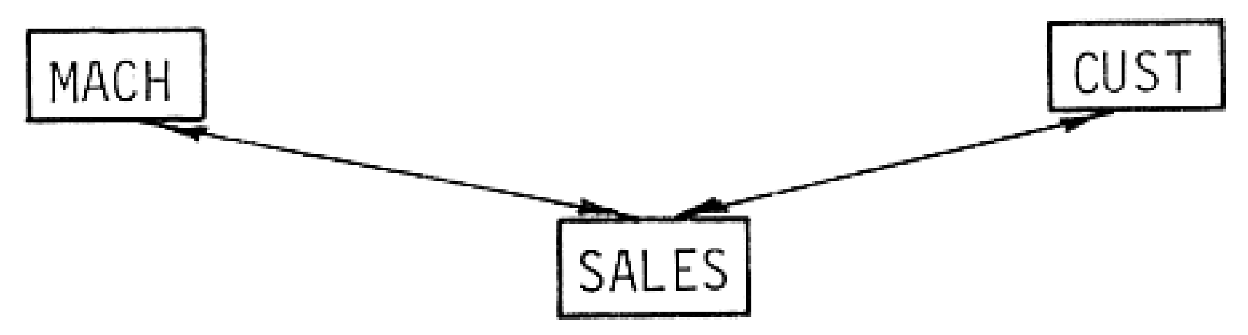
Third step : seeking the starting nodes

The point is to seek in the access paths tree the nodes that may serve as "entry-points"(1), which means that the initial access to the realizations of one of those nodes affords to get the results of the original algorithm.

In the simplified optimizing algorithm, the root of the access paths tree will be assumed to be a starting point.

In our example, ENG and DEPT may not be entry-points because the relations (ENG, SALES) and (DEPT,SALES) are characterized by $K_3 = K_5 = 0$ and because the conditions corresponding to DEPT and ENG are linked by an OR operator. An access path from these two nodes could result in valid realizations of SALES being lost.

The starting nodes are as follows



Fourth step : seeking the optimal access path

Let us examine a starting node N, n sons of which (S_1, S_2, ..., S_n) are starting nodes, and let us seek the best access path to N. Let us suppose that to each relation (N,S_i) corresponds only one relation criterion (the other cases are more difficult to deal with).
There are n+1 possible paths to N :
- access the realizations of S_1 via the best path,
  access by the relation (S_1,N) the realizations of N, then
  verify the conditions possibly imposed on N that relate to S_2,...S_n
- idem for S_2,...S_n

(1) See in[21] the concept of "port".

- access all the realizations of N in order to verify the criteria on $S_1, S_2, \ldots S_n$.

When the cost of each of these paths has been calculated, the path with the minimum cost is chosen.
That procedure should be noted to be recursive as it has to be applied to $S_1, S_2, \ldots S_n$.

Let us call that procedure P(N).

The main output is $\mathcal{A}$(N), the optimal access algorithm to the realizations of the node N. Yet it has to provide the following outputs as well :
- KA(N) : the total cost of the optimal access,
- p(N) : the probability of the condition linked to the node N,
- KC(N) : the cost of evaluating that condition

thanks to which it is possible to calculate the probability and costs pertaining to N's father.

If R is the root of the access paths tree, seeking the optimal access path corresponding to the reduced conditional form is performed by :

P(R)

Before dealing with the processing of the example we have to solve the following problem : in what order have the criteria of a condition to be evaluated if the average evaluating time is to be minimum ?

It is possible to demonstrate ([24]) that if the condition is a conjunction of disjunctions, or a disjunction of conjunctions, of criteria, if the criteria are independent and if their costs are constant, then the optimal order is the one in which:
- the factors C_i of a conjunction are arranged in ascending order of the values of

$$\frac{K(C_i)}{1-p_{Ci}}$$

- the terms C_j of a disjunction are arranged in ascending order of the values of

$$\frac{K(C_j)}{p_{Cj}}$$

Actually that algorithm has to be modified when the evaluation of a criterion may modify the cost of another criterion (non constant costs).

Now let us analyse the operations of P(SALES)(1)

SALES has two sons which are starting points : MACH and CUST.
To SALES correspond three possible paths : from MACH, from CUST and from SALES itself.

P(MACH) will yield $\mathcal{A}$(MACH), KA(MACH), p(MACH) and KC(MACH).
P(CUST) will yield $\mathcal{A}$(CUST), KA(CUST), p(CUST) and KC(CUST).

These data will enable us to determine the cost of each of the three access paths and thus to choose the best one.

(1) As each node corresponds to a different object, we will give the node the name of the object.

P1. P(MACH)

As there is no access key for MACH, the only possible access is the one through (ROOT, MACH) :

$$p(\text{MACH}) = \sum_{k=1}^{1} F_{13}(k).\Pi_{(=B)}(k,1) = p(=B) = P_T(B) = 0.05$$

$$KC(\text{MACH}) = 0$$

$$KA(\text{MACH}) = C_{20}(N_M) + N_M.KC(\text{MACH}) = 21$$

$\mathcal{A}$(MACH) = sequential access to the realizations of MACH.

P2. P(CUST)

CLASS is an access key to CUST ; a choice will have to be made between the access through (CLASS,CUST) and the one through (ROOT,CUST)(sequential access).

$$p(\text{CUST}) = \sum_{k=1}^{1} F_{10}(k).\Pi_{(>12)}(k,1) = p(>12) = \sum_{k=13}^{20} P_{CL}(k) = 0.4$$

$$KC(\text{CUST}) = 0$$

Regardless of the access to the values of CLASS (the index), the cost of the access by key is at least :

$$\sum_{k=13}^{20} (1 \times 25) = 200$$

The sequential access through (ROOT,CUST) costs : $C_{60}(N_C) = 51$

KA(CUST) = 51
$\mathcal{A}$(CUST) = sequential access to the realizations of CUST.

P3. Calculating the probability and cost of the criteria corresponding to SALES

P31. $C_M \equiv (: \text{MACH}(: \text{TYP}(=B)))$

$$P_M = \sum_{k=1}^{1} F_2(k)\ \Pi_{(:TYP(=B))}(k,1) = 0.05 \qquad m = \sum_{k=1}^{1} \mathfrak{F}_2(k)\ \Pi_{(:TYP(=B))}(k-1,0) = 1$$

$$K_M = C_2(1) + KC(\text{MACH}) = 1$$

P32. $C_C \equiv (: \text{CUST}(: \text{CLASS}(>12)))$

$$p_C = 0.4$$

$$K_C = 1 \qquad \text{(as P31)}$$

P33. $C_D \equiv (: \text{DEPT}(: \text{DEP\#}(=3)))$

$$p_D = p(: \text{DEP\#}(=3)) = p(=3) = 0.04$$

$$K_D = 1$$

P34. $C_E \equiv (\ :\ 1 - ENG(\ :\ CAT(=3)))$

$$p(\ :\ CAT(=3)) = p(=3) = 0.1$$

$$p_E = 1 - \sum_{k=0}^{4} F_4(k)\ \underset{(:CAT(=3))}{\Pi}(k,0) = 0.22$$

$$m = \sum_{k=1}^{4} \mathfrak{F}_4(k)\ \underset{(:CAT(=3))}{\Pi}(k-1,0) = 2.2$$

$$K_E = 2.2$$

P4. Calculating the cost of the access from SALES

Before calculating KC(SALES) we must choose an evaluation order for $(C_M \ \&\ C_C \ \&\ (C_D \vee C_E))$

- $C_{ED} \equiv C_D \vee C_E$: $\frac{K_E}{p_E} < \frac{K_D}{p_D}$ hence the order : $C_E \rightarrow C_D$

$$p_{ED} = 0.25$$

$$K_{ED} = 2.98$$

- $C_M \ \&\ C_C \ \&\ C_{ED}$: $\frac{K_M}{1-p_M} < \frac{K_C}{1-p_C} < \frac{K_{ED}}{1-p_{ED}}$ hence the order : $C_M \rightarrow C_C \rightarrow C_{ED}$

$$p(SALES) = 0.005$$

$$KC(SALES) = 1.11$$

$$KA_S(SALES) = C_{40}(N_S) + N_S.KC(SALES) = 1211$$

P5. Calculating the cost of the access from MACH

The access through (MACH,SALES) means C_M = 'TRUE'

$$K_{CDE} = K(C_C \ \&\ (C_D \vee C_E)) = 2.19$$

$$KA_M(SALES) = KA(MACH) + p(MACH).N_M.(C_1(m_1) + m_1.K_{CDE}) = 180.$$

P6. Calculating the cost of the access from CUST

The access through (CUST,SALES) means C_C = 'TRUE'.
Calculating as in P5, we obtain :

$$KA_C(SALES) = 910.$$

P7. Optimal access path

$KA_M(SALES)$ is the minimum cost and corresponds to the optimal access path.

Thus the algorithm that corresponds to the initial program is :

```
𝒜(SALES) ≡ - access    all the realizations of MACH ;
           - select those that verify (:TYPE(=B))
           - for each of them, access through (MACH,SALES)    the realizations of
             SALES
           - verify (C_C & (C_E V C_D)) in the order C_C → C_E → C_D ;
           - suppress the realizations of SALES selected that way.
```

The total expected cost of this algorithm is

$$K = 180 + 5.C_{SS} = 230$$

2. DESIGN OF AN OPTIMAL ACCESS STRUCTURE FOR A DATA BASE

This application is related to another part of this study, devoted to Data Independence in data bases. We will restrict ourselves, however, to defining the concepts absolutely necessary for trying to solve the problem of designing an access structure.

One of the first steps in the definition of a data base consists in describing :
- the applications and their frequency,
- the data of the data base - regardless of efficiency (conceptual or semantic structure).

We have seen that a conceptual data structure may be described by an ISM.

For the first step, we suggest
- to construct the ISM of the conceptual structure (CS) regardless of the access paths to implement, and to establish the statistical description of the data ;
- to describe each application by means of an ADL program operating on the CS.

Obviously it would not make sense to adopt that description as the description of the implementation since the latter will usually be very inefficient.

A second ISM (AS) will thus have to be constructed for describing the access path to implement so as to make this structure "optimal" for the applications foreseen.

However, many possible access structures AS_i may exist, for which the applications have to be translated in order to determine their cost.
Is it possible to translate an ADL program operating on CS into an equivalent program (providing the same results) operating on AS_i automatically ? It is, if CS and AS_i are "*semantically equivalent*". That concept is defined as follows :

"Two structures S_1 and S_2 are semantically equivalent if, for any element of S_1 (S_2) there exists an ADL procedure operating on S_2 (S_1) and making it possible to retrieve or construct that element".

For instance, the CODASYL and relational structures in § 1.4, Part I, are semantically equivalent.

The set of those procedures on AS_i for all the elements of CS constitutes the mapping of CS on AS_i. Thanks to that mapping a program on CS may be automatically translated into a program on AS_i (see the example in Appendix B).

Let $P_1, P_2, \ldots P_m$ be the ADL description of the applications foreseen and $f_1, f_2, \ldots f_m$ their frequency.

Let us denote by $K_i(P_j)$ the cost of P_j (translated for AS_i) operating on AS_i. If AS_i represents the implementation of CS, a measure of the total expected cost of the applications is thus :

$$K_i = \sum_{k=1}^{m} f_k.K_i(P_k) + KS_i$$

where KS_i represents the storage cost of the data described by AS_i (this problem is not dealt with here ; see, for instance, [6], and [34] for the CODASYL structures).

AS_i is an "optimal" - or at least a good - access structure from the total cost point of view if :

$$K_i \leqslant K_j \text{ for } j = 1,2,\dots n$$

Although an entirely automatic design of an optimal access structure might be imagined, it seems more interesting to define an interactive tool for constructing that structure.

Four principles underlie that tool :

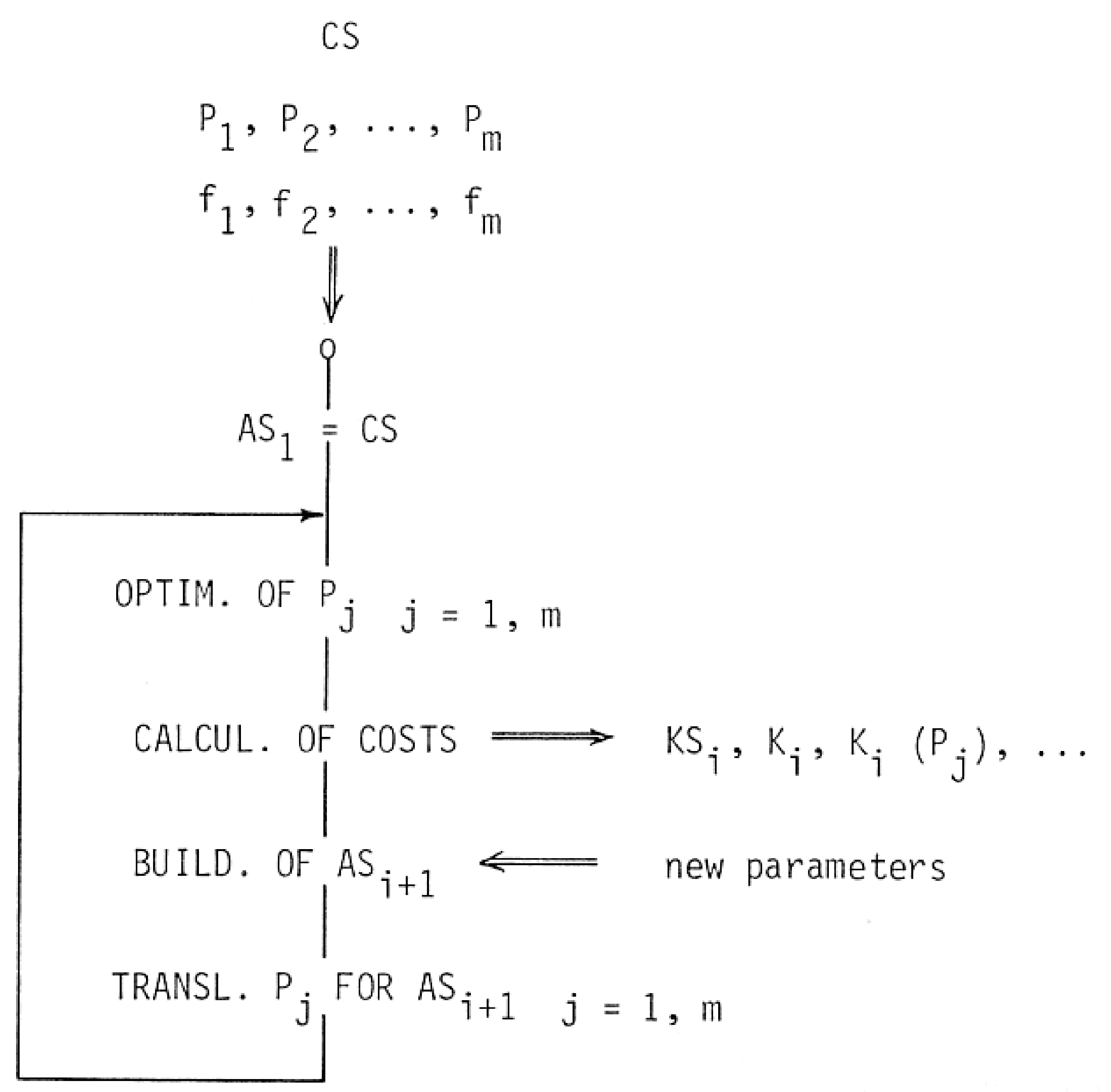
(i) In the commonest cases, the statistical model of AS_i may be deduced automatically from that of CS (except for the cost function, of course). Those cases correspond to distinct forms of mapping.

(ii) There exists an algorithm transforming any program on CS into a program on AS_i (see Appendix B).

(iii) There exists an algorithm optimizing any program on AS_i (see Application 1).

(iv) The control means at the designer's disposal are :
- adding (or suppressing) an element to a structure by merely specyfying the mapping procedure,
- choosing the implementation parameters of the access paths (i.e. choosing the cost functions of the statistical model).

The work of the designer could be imagined to proceed as follows :



Such a procedure should make it possible to examine interesting solutions that are not optimal from the point of view of the total cost.

It is worth to notice that the problem of the functional description of the data structures and the applications have been studied by authors such that WANG [23].

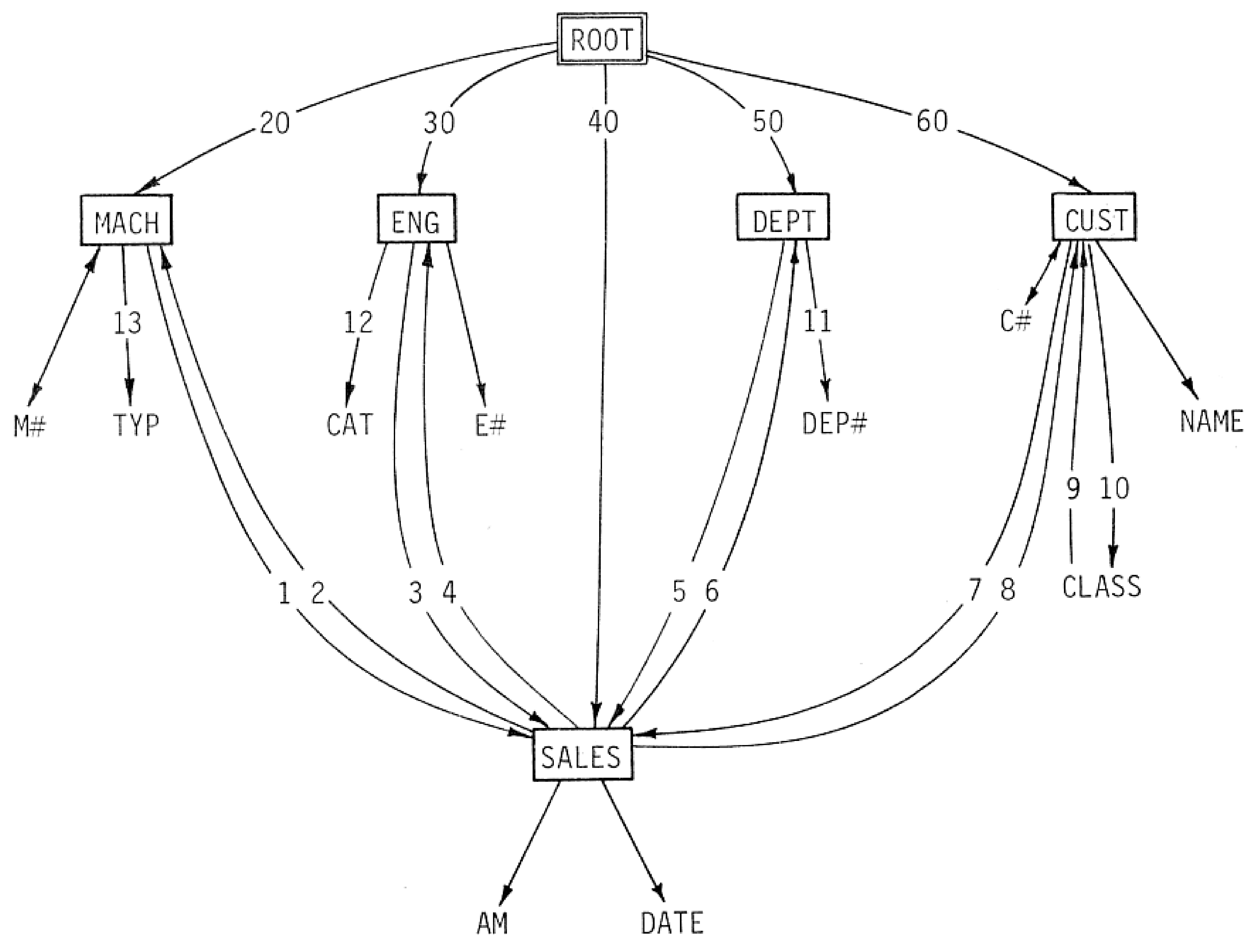
APPENDIX A

INFORMATION STRUCTURE OF A DATA BASE

1. THE REAL SYSTEM TO BE DESCRIBED

A manufacturer sells machines to customers.
Each machine, together with the necessary equipment, is usually (but not always) sold by a department and installed by a team of engineers.
The machines are partitioned into several types, the customers into classes and the engineers into categories.
The machines, engineers, departments and customers are identified by a number.
Each sales is characterized by its date and the cost ("amount") of the installation.

2. GRAPHICAL REPRESENTATION OF THE ISM



3. THE INFORMATION STRUCTURE MODEL AND THE STATISTICAL MODEL

In order to simplify the description, we will not describe certain elements which are not relevant to application 1.
All relations are nameless ; they will be referred to by means of a number to shorten the (A,B) designation.

The complex objects

Name	*Number of realizations*	*Costs*
MACH	$N_M = 200$	
ENG	$N_E = 60$	
DEPT	$N_D = 25$	
CUST	$N_C = 500$	
SALES	$N_S = 1000$	$C_{SS} = 10$

The elementary objects

Name	*Number of active realizations*	*Frequency functions*
TYP	$n_T = 20$	$P_T = 0.05$
CAT	$n_{CT} = 5$	P_{CT}=A(0.1),B(0.2),C(0.3), G(0.3),S(0.1) (*)
DEP#	$n_D = 25$	$P_D = 0.04$
CLASS	$n_{CL} = 20$	$P_{CL} = 0.05$

The relations

R	*I-J,K-L*	m_R	$F_{RO}(n)$
(MACH,SALES)	0-∞,1-1	$m_1 = 5$	$F_1(n) \cong e^{-5} \cdot \frac{5^n}{n!}$ $n \in [0-20]$
(ENG,SALES)	0-∞,0-∞	$m_3 = 40$	
(SALES,ENG)	0-∞,0-∞	$m_4 = 2.4$	$F_4(n)$=1(0.2),2(0.3),3(0.4), 4(0.1)
(DEPT,SALES)	0-∞,0-1	$m_5 = 40$	$F_5(n)=0.2$ $n\in\{30,35,40,45,50\}$
(CUST,SALES)	0-∞,1-1	$m_7 = 2$	
(CLASS,CUST)	0-∞,1-1	$m_9 = 25$	$F_9(25) = 1$

For other relations :

$$F_2(1) = F_6(1) = F_8(1) = F_{10}(1) = F_{11}(1) = F_{12}(1) = F_{13}(1) = 1.$$

Access costs

$$C_{20}(n) = C_{40}(n) = C_{60}(n) = 1 + 0.1 \times n$$

$$C_1(n) = C_2(n) = C_4(n) = C_6(n) = C_7(n) = C_8(n) = C_9(n) = n$$

APPENDIX B

MAPPING OF SCHEMATA AND ALGORITHM TRANSFORMATION

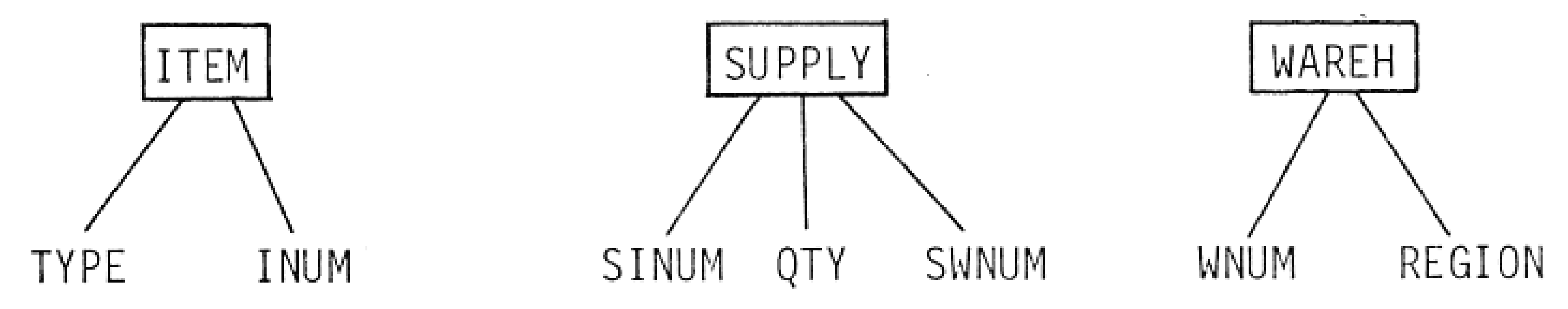
In a retail firm, a central warehouse supplies local warehouses with various items. One wishes to know at any moment the quantities of each item that have been delivered to each local warehouse since a given date.

(*) The point $P_{CT}(x) = y$ is noted : x(y)

The following set of 3NF relations might represent the data describing that sub-system :

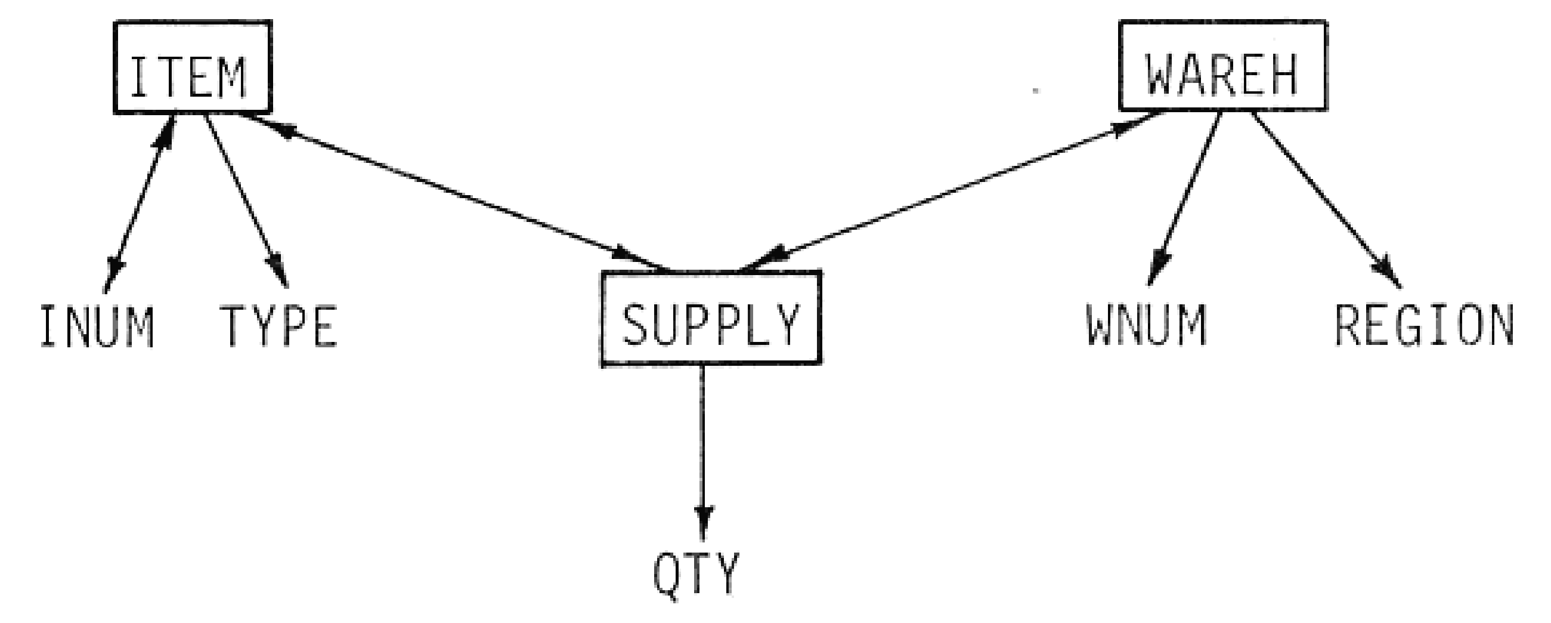
ITEM (INUM,TYPE)	key (ITEM) = INUM
WAREH (WNUM,REGION)	key (WAREH) = WNUM
SUPPLY (SINUM,SWNUM,QTY)	key (SUPPLY) = SINUM, SWNUM

A simplified ISM of that conceptual structure would then be :



(ITEM,INUM)	: 1-1,0-1	(SUPPLY,SINUM)	: 1-1,0-∞
(ITEM,TYPE)	: 1-1,0-∞	(SUPPLY,SWNUM)	: 1-1,0-∞
(WAREH,WNUM)	: 1-1,0-1	(SUPPLY,QTY)	: 1-1,0-∞
(WAREH,REGION)	: 1-1,0-∞	id (SUPPLY) = ((SUPPLY,SINUM),(SUPPLY,SWNUM))	

The analysis of the foreseen applications leads to an access schema described by the following ISM :



That schema differs from the previous one as follows :

(SUPPLY,ITEM) : 1-1,0-∞
(SUPPLY,WAREH) : 1-1,0-∞
id (SUPPLY) = ((SUPPLY,ITEM),(SUPPLY,WAREH))

The following mapping of the conceptual schema in the access schema makes them *semantically equivalent* :
- the mapping is one to one except for :

(SUPPLY,SINUM) ≡ SUPPLY [: ITEM [: INUM]]
(SUPPLY,SWNUM) ≡ SUPPLY [: WAREH [: WNUM]]

These relations are thus described as simple compositions of access relations.
So as to show how an algorithm on a schema can be transformed into an algorithm on another schema, let us consider the following applications :

"list of the supplies of the type X items to a warehouse in region Y"

If the data are viewed as in the conceptual schema, that application may be written in ADL as follows. Let

```
        INUM (: ITEM (: TYPE = X) )
```

be the realizations of INUM linked to an ITEM linked to a TYPE = X :

```
        SUPPLY ((: SINUM = INUM (: ITEM (: TYPE = X)))
             & (: SWNUM = WNUM (: WAREH (: REGION = Y))) )
```

It is interesting to notice that this expression is isomorphic to the correspon-ding SEQUEL expression :

```
        SUPPLY WHERE (SINUM = SELECT INUM FROM ITEM WHERE TYPE = X)
                AND SWNUM = SELECT WNUM FROM WAREH WHERE REGION = Y.
```

A functionally equivalent algorithm working on the access schema should be built up. Let us first replace the composed relations by their definition :

```
        SUPPLY ((: ITEM (: INUM = INUM (: ITEM (: TYPE = X))))
             & (: WAREH (: WNUM = WNUM (: WAREH (: REGION = Y))) ) )
```

Now, the following reductions may easily be proved to be valid :

1) $A(r : B = B(C_B)) \rightarrow A(r : B(C_B))$

2) $A(r : B(r^{-1} : A(C_A))) \rightarrow A(C_A)$ if $I_r = L_r = 1$

Applying reduction 1 yields :

```
        SUPPLY ((: ITEM (: INUM (: ITEM (: TYPE = X))))
             & (: WAREH (: WNUM (: WAREH (: REGION = Y)))) )
```

And by reduction 2 we obtain the minimal expression :

```
        SUPPLY ((: ITEM (: TYPE = X))
             & (: WAREH (: REGION = Y)) )
```

REFERENCES AND BIBLIOGRAPHY

[1] SENKO, LUM, OWENS : A *File Organization Evaluation Model*, in Proc. IFIP Congress 1968.

[2] CARDENAS, A., F. : *Evaluation and Selection of File Organization - A Model and System*, Com. ACM, Sept. 73, Vol. 16, N° 9.

[3] FUMIO NAKAMURA, IKUZO YOSHIDA, HIDEFUMI, KONDO : A *Simulation Model for Data Base System Performance Evaluation*, in Proc. National Computer Conference 1975.

[4] LEFKOVITZ : *File Structures for On-Line Systems*, Spartan Book, 1969, New York.

[5] KING, W.F. : *On the Selection of Indices for a File*, IBM Research RJ 1341, Jan. 1974.

[6] SHI BING YAO : *Evaluation and Optimization of File Organizations through Analytic Modelling*, Ph. D. Dissertation, Univ. of Michigan, 1974.

[7] HAINAUT, J-L. : A *Semantic Formal Model of an implemented Data Base and its Data Manipulation Language*, Institut d'Informatique de Namur, July 1973.

[8] HAINAUT, J-L and LECHARLIER, B. : *Présentation et spécification du modèle d'accès et du langage rigide*, Internal Papers, Institut d'Informatique de Namur, Feb. 1974.

[9] ABRIAL, J.R. : *Data Semantics*, *in Data Base Management*, IFIP WC 1974, North-Holland Publish. Co., 1974.

[10] BACHMAN, Ch., BREWER, S. : A *Direct Access System with Procedurally Generated Virtual Data Structuring Capability*, Honeywell Comp. Journal, 1974, Vol. 8, n° 1.

[11] ABRIAL, BAS, BEAUME, HENNERON, MORIN, VIGLIANO : *Projet Socrate*, Institut de Mathématique Appliquée de Grenoble, 1970.

[12] PAULUS, LECHARLIER, HENNEBERT, HAINAUT, DEHENEFFE : *Système de conception et d'exploitation d'une base de données*, Publication de l'Institut d'Informatique de Namur, 1974.

[13] HAINAUT, J-L., LECHARLIER, B. : *An Extensible Semantic Model of Data Base and its Data Language*, Proc. IFIP Congress 1974, North-Holland Publish. CO, 1974.

[14] *Interim Report ANSI/X3/SPARC Study Group on Data Base Management Systems*, ANSI, Feb. 1975.

[15] *Data Base Description*, Proc. of IFIP TC-2 WC, 1975, North-Holland Publish.Co.

[16] CODD, E., F. : A *Relational Model of Data for Large, Shared Data Banks*, CACM, Vol. 13, n° 6, June 1970.

[17] *CODASYL*, Data Base Task Group 1971 Report, ACM, 1971.

[18] SENKO, ALTMAN, ASTRAHAN, FEHDER : *Data Structure and Accessing in Data Base System*, IBM System Journal, Vol. 12, n° 1, 1973.

[19] GOSH, SENKO : *On the Analysis of search Procedure in an Information Retrieval System based on String Paths*, IBM Research RJ 1140, 21 Dec. 1973.

[20] GOSH, ASTRAHAN : A *Translator Optimizer for Obtaining Answers to Entity Set Queries from an Arbitrary Access Path Network*, IBM Research RJ 1282, 27 Sept. 1973.

[21] COLLMEYER, A., J. : *Implications of Data Independence on the Architecture of Database Management Systems*, 1972 ACM SIGFIDET Workshop Report.

[22] BRACCHI, PAOLINI, PELAGATTI : *Binary Logical Associations in Data Modelling*, Internal Report n° 75-12, Sept. 75, Ist. di Elettrotecnica ed Elettronica del Politecnico di Milano.

[23] WANG, C.P. : *Parametrization of Information System Application*, IBM Research, RJ 1199, April 1973.

[24] SLAGE, J.R. : *An Efficient Algorithm for Finding Certain Minimum Cost Procedures of Making Binary Decisions*, JACM, Vol. 11, n° 3, July 1964.

[25] DEHENEFFE, HAINAUT, HENNEBERT, LECHARLIER, PAULUS : *Modèle d'accès pour une base de données*, Journées d'Etudes sur les Systèmes généraux de bases de données. Montpellier, 15-16 May 1975.

[26] LALOUX, J.P. : *Implémentation du modèle d'accès par lui-même*, Mémoire de fin d'études, Institut d'Informatique de Namur, Juin 1976.

[27] SENKO, M.E. : *The DDL in the context of a Multilevel Structured Description : DIAM II with FORAL*, in Data Base Description, North-Holland Publish. Co. 1975, pp. 239-257.

[28] ABRIAL, J.R. : *Notes du cours sur l'architecture des systèmes*, Alpe d'Huez, Dec. 1972.

[29] ASTRAHAN, GOSH : *Search Path Selection for the D.I.A.M.*, IBM Research, RJ 1327, Dec. 1973.

[30] BAYER, McCREIGHT : *Organization and Maintenance of Large Ordered Indexes*, Acta Informatica 1, 1972, pp. 173-189.

[31] CARDENAS : *Analysis and Performance of inverted Data Base Structures*, IBM Research. RJ 1375. 1974.

[32] CASEY : *Design of Tree Structures for Efficient Querying*,CACM, Sept. 1973.

[33] COLLMEYER, SHEMER : *Analysis of Retrieval Performance for Selected File Organization Techniques*, F J C Conference 1970.

[34] de BEER, SMIT : *The Performance of Data Base Systems in Relation to Data and Storage Structure*, Proc. of EUROCOMP Conference on Software Systems Engineering, Sept. 1976.

[35] FLORY, GUNTHER, KOULOUMDJIAN : *Data Base Reorganization by Clustering Methods*, Université de Lyon, France, 1975.

[36] SCHKOLNICK : *The Optimal Selection of Secondary Indices for Files*, Proc. Internat. Computing Symposium 1975, North-Holland Publish. Co. 1975.

[37] STONEBRAKER : *Retrieval Efficiency Using combined Indices*, ACM SIGFIDET, 1972.

[38] LUM, LING : *An Optimization Problem on the Selection of Secondary Keys*, Proc. ACM, Nat. Conf. 1971.

[39] SILER : *A Stochastic Evaluation Model* : CACM, 19, 2, Feb. 1976.

[40] LOWE : *The Influence of Data Base Characteristics and Usage*, JACM 15, 4, Oct. 1968.

[41] WATERS : *Analysis of Self-indexing, Disc Files*, The Computer Journal 18, 3

[42] WEDEKIND : *On the Selection of Access Path in Data Base System*, Data Base Management, North-Holland Publish. Co., 1974.

[43] SEVERANCE : *Identifier Search Mechanisms: A Survey and Generalized Model*, Computing Surveys, Vol. 6,3, Sept. 1974.

[44] SEVERANCE, DUHNE : *A Practitioner's Guide to Addressing Algorithms*, CACM, June, 1976.

[45] HUANG, GOEL : *An Analytical Model for Information Processing Systems*, Proc. NCC 1974.

[46] HAMMER, CHAN : *Index Selection in a Self-adaptative Data Base Management System*, ACM SIGMOD 1976.

[47] GOSH, SENKO : *File Organization : On the Selection of Random Index Points for Sequential Files*, JACM, October 1969.

[48] MELBOURNE : *Direct Access Storage Device Timing Evaluations*, IBM, October 1974.

[49] THEOREY, SUNDAR : *Application of an Analytical Model to Evaluate Storage Structures*, ACM SIGMOD 1976.

[50] MARTIN : *Design of Real-time Computer Systems*, Prentice Hall, 1967.

[51] OMAHEN : *Estimating Response Time for Auxiliary Memory Configurations with Multiple Movable-head Disk Modules*, ACM Conference on Very Large Data Base, 1975.

[52] BENCI, BODART, BOGAERT, CABANES : *Concepts for the Design of Conceptual Schema*, Proc. WG-TC2, IFIP, North-Holland Publish. Co., 1976.

[53] CHAMBERLIN, BOYCE : *SEQUEL : A Structured English Query Language*, ACM SIGMOD 1974.

[54] DEHENEFFE, HAINAUT, TARDIEU : *The Individual Model*, Proc. International Workshop on Data Structure Models, Namur 1974, Presses Universitaires de Namur, 1975.

[55] LUM, LING, SENKO : *Analysis of a Complex Data Management Access Method by Simulation Modelling*, FJC Conference, 1970.

[56] ISAO MIYAMOTO : *Hierarchical Performance Analysis Models for Data Base Systems*, ACM Conference on Very Large Data Bases, 1975.

[57] CABANES, A. : *Data Independence and Physical Implementation*, Proc. International Workshop on Data Structure Models, Namur, 1974, Presses Universitaires de Namur, 1975.

[58] HAINAUT, J.L. : *Evaluation des performances d'une base de données par modèle probabiliste*, Proc. Séminaire INFORSID, Grenoble, oct. 1976, IRIA.

[59] CABANES, A. : *Rapport Indépendance dans les SGBD*, Mars 1977, Institut d'Informatique d'Entreprise, C.N.A.M., Paris.