THEORETICAL AND PRACTICAL TOOLS FOR DATA BASE DESIGN

J.L. HAINAUT

INSTITUT D'INFORMATIQUE - UNIVERSITE DE NAMUR
rue GRANDGAGNAGE, 21 - B-5000 NAMUR - BELGIUM

ABSTRACT

The paper presents two families of reversible transformations for a binary information model. The binary model is considered as a special case of a generalized n-ary relational model as are CODD'S and E/R models. This parenthood increases the power of the binary model and makes the transformation valid for other models. Two applications of the transformations are presented within the framework of an Information System Design Approach; they are namely a conceptual model conversion and a valid TOTAL schema production.

1. INTRODUCTION

Data base design, and more generally Information System design, is known to be a complex process. A design method is usually decomposed into successive design steps, which are made more or less explicit, starting with the gathering of user's requirements and ending up with programs and operational files or data bases. Such a method relies on an information model which in some cases may be very elementary, and even implicit.

The Information System Design Approach under development in the Institut d'Informatique of Namur proposes a four steps design process which may be assisted by a sophisticated data dictionary system supported by an extended ISDOS System [4].
These steps are derived from the multilevel design process proposed in the introductory report in [20]; they are namely :
(1) analysis of the real world and gathering of user's requirements; (2) design of the information conceptual schema and design of the applications, viz. the functional specification and the dynamics of the procedures; (3) design of a system architecture into modules and design of the algorithms and of the data access structures needed; (4) mapping of this output into the operational computer environment : hardware, OS, programming language, DBMS, ...

As far as the data base is concerned, the three formalized data descriptions are the conceptual schema from step 2, the basic logical access schema from step 3 and the DBMS/DMS schema from step 4.

The conceptual schema is a description of the data and of their semantic properties; whether the applications will be computerized or not is irrelevant at this stage. The schema must be agreed upon by both the system developers and the users, who are sometimes managers and therefore not computer-minded at all. Such a schema relies on a rigorous, generally mathematical, model such as n-ary, binary or E/R models.

The basic logical access schema is a first proposal for a file or db structure which is as simple as possible (for example, which is graphically close to the conceptual schema); it emphasizes the access structure strictly needed for the algorithms of the applications. Neither criteria of performance nor DBMS restrictions have been taken into account so far. This kind of schema requires an expression formalism adapted to access concepts : the Logical Access Model. Despite its low level of sophistication, a lot of interesting data can be deduced from this schema; these are mainly logical access counts and data transfer volumes for each algorithm.

The next schema is written in the DDL of an actual DBMS, and includes results from performance oriented decisions. This schema is generally twofold; the general access schema used by the application programmers (sometimes via subschemas) and the physical schema which specifies the physical parameters of the data base.

These three levels can be recognized in most current data base design methods although the concepts and formalisms are not always the same; in some methods, the conceptual and logical access levels are not clearly separated. For example a similar approach can be found in [18].

One of the most difficult design steps is the production of the DBMS schema from the basic logical access schema. Much work has been devoted to this step; more or less comprehensive analytic and simulation tools are proposed in [15], [16] and

[19] to mention only somes of them.

This paper is not concerned with such a design tool; rather, it presents a very restricted, but powerful set of theoretical technique which may be useful for a large range of schema transformation methods, either at the conceptual, logical access or DBMS levels. The tools presented here mainly consist of two families of reversible mappings on binary structures. The reversibility ensures loss-free, noise-free transformation not only of the information structure, but of the integrity constraints and of the semantics as well, although they are not dealt with in this paper.

The paper is organized as follows : a generalized relational model (section 2) is very briefly outlined, from which a binary model will be derived (section 3), leading to the presentation of the transformations (section 4). Generally intended for the conceptual level, the binary model will be extended to access concepts for defining the logical access model (section 5). Two applications of the transformations are then presented; (section 6) the translation of a binary (or E/R) schema into CODD'S model and conversely and (section 7) the translation of a conceptual schema into a particular DBMS model.

2. A GENERALIZED RELATIONAL MODEL

Let us consider a n-ary relational model in which relations are defined not only on simple domains but on entity-domains as well. A value of an entity-domain denotes the existence of an object, individual or concept of the real world and constitutes a designation means for this object. Although such a value is often called "surrogate" we will use the term data base entity, or more concisely entity. We will not discuss the notion of entity in more detail; such a discussion can be found in Hall [14], Pirotte [17], Codd [10], Benci and al [3] and Hainaut [11].

With a slight adaptation, relational algebra and functional dependency concepts and properties are still valid for this extended model. An interesting use of this model consists in restricting the possible constructs with a set of constraints. Such sets can generate CODD'S model [9], E/R models [7], and binary models [1],[5],[11];these models therefore inherit most of the powerful concepts and tools of the generalized model.

The reader is supposed to have some knowledge about the fundamentals of relational models. Throughout this paper, we shall make use of the following concepts and notation;

- R(A,B,C) : relation (schema) defined on domains A,B,C
- $A,B \xrightarrow{R} C$: C is functionally dependent on A,B in R (R is any relational expression)
- R(A,B,C) : A,B is a key (or identifier) of R
- R[A,B] : projection of R on A,B
- R[a,B] where a A : projection of R where A=a on domain B
- R*S : natural join on the common domains
- $\overset{1}{\underset{3}{*}}Ri$ = R1*R2*R3 : (R1*R2)*R3
- RoS : composition on the common domains (= projection of the join on the non common domains)
- RoSoT =(RoS)oT

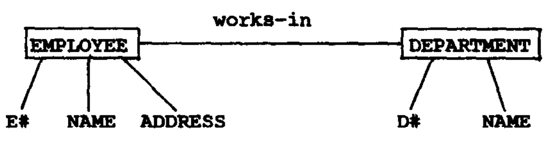
3. A BINARY RELATIONAL MODEL

Binary models enjoy remarkable properties such as the atomiciy of their concepts and graphical representation capabilities. However they are hampered by serious drawbacks; the main of them is that they are unable to express easily some usual constraints such as multi-domain keys (from now on, the term key will be replaced by identifier, the former assuming too many meanings : relational key, sort key, access key, search key, selection key, privacy key, ...).

The current binary model is generated from the generalized model by the following rules :

(1) binary relations only (2) each connected component of the graph so defined contains at least one entity domain (there is always a path from each simple domain to at least one entity domain). This rule is not necessary from the theoretical point of view; it is only intended to define schemas which are processable in the lower levels (access and DBMS levels).

In the graphical representation of a schema, a simple domain will be represented by (the string of) its name, an entity domain by its name enclosed in a box, and a relation by a labelled arc between the representations of its domains; the empty string is a valid label (relation without name).



The representation of a simple domain participating in more than one relation may (but need not) appear several times; this is tolerated to make the drawing clearer. Such is the case of NAME in the example above.

The triplet <relation name, first domain name, second domain name> identifies a relation in a schema. Its name (label) will generally be sufficient to designate a relation; however, in case of ambiguity, or if the relation has no name, a more explicit designation will be needed

To capture more semantics the model will be provided with three kinds of properties or integrity constraints; connectivity, existence constraints, and multi-domain identifiers.

Connectivity of a relation

Connectivity is an expression of the functional dependencies that hold in the relation. In the following table, A and B are simple or entity domains.

relation	connectivity	graphic.
R(A,B)	many to many (N-N)	A—⧓—B

R(A,B)	from A to B: one to many (1-N)
	from B to A: many to one (N-1)
		A—◀—B

R(A,B)	one to one (1-1)	A——B

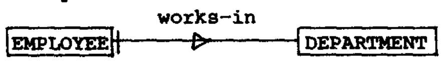
It is worth noticing that 1-1 is a particular case of 1-N and N-1, and that 1-N and N-1 are particular cases of N-N; hence, for example, any property of N-N relations is also a property of 1-N, N-1 and 1-1 relations.

Existence constraint of an entity domain

This constraint consists in forcing, at any time, every entity of a domain to participate in a tuple of a given relation. For instance if an employee always works in a department, the previous example can be completed by the existence constraint :

works-in [EMPLOYEE] = EMPLOYEE

or, graphically :



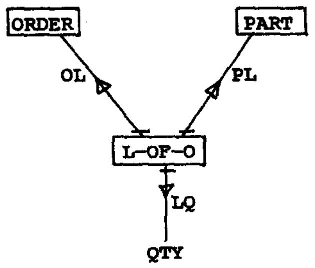
Multi-domain identifier

The connectivity of a relation is an expression of the ability of a domain to identify the other one; in the last example, an EMPLOYEE entity identifies one DEPARTMENT entity via works-in. The multi-domain identifiers of n-ary models, however, seem to give rise to a problem.

The n-ary origin of the binary model allows us to use some of the powerful tools of the n-ary models and so to retrieve some power of the lost paradise. Let's assume the n-ary schema :

LINE-OF-ORDER (ORDER, PART, QTY)

which can be binarized into (see section 6):



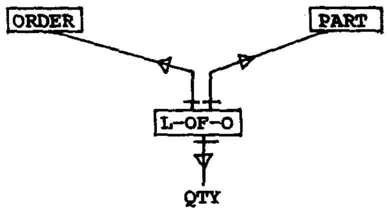
such that:

LINE-OF-ORDER = OL*PL*LQ [ORDER,PART,QTY]

The identifier of LINE-OF-ORDER can be expressed in the binary schema by :

$$\text{ORDER,PART} \xrightarrow{\text{OL * PL}} \text{L-OF-O}$$

or, graphically, by :



4. TWO FAMILIES OF BINARY TRANSFORMATIONS

Now we are going to define two sets of mappings that will allow us to transform a binary schema into an other one in such a way that the former can be deduced back from the latter without information loss or noise (reversibility). These mappings are simple and intuitive. They allow the transformation of the information structure (domains and relations) and of various integrity constraints and of the semantics (meaning) as well; the discussion will deal with information structure only. Despite their simplicity, these mappings are sufficient to generate a large range of the most useful transformations [12].

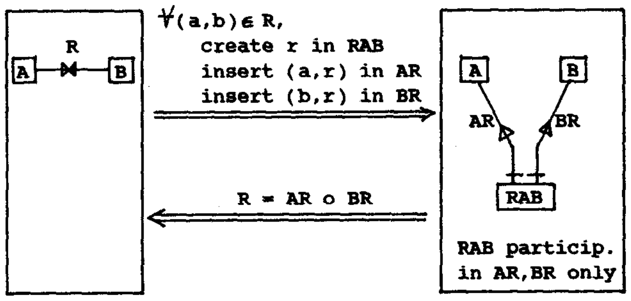
The mappings could have been defined in the generalized model. Yet their expression would have been more complex and far from intuitive, essentially due to the absence of graphical representation ; moreover, it can be proved that they are also valid for other models (CODD', E/R).

The mappings will be presented (without proof) under the form of implication symbols between two schemas; the symbols are topped by the expression of the mapping. A concise notation will also be provided to designate the objects concerned by the mappings.

In the following, the names A, Ai, B, C designate either simple or entity domains.

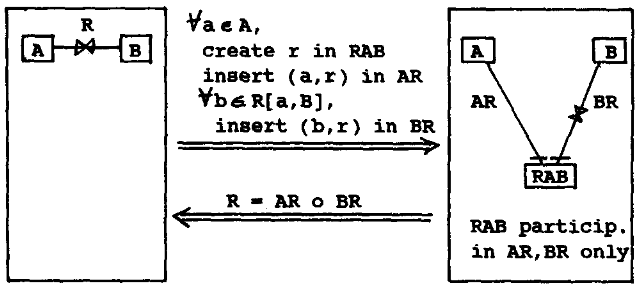
Introducing/removing an entity-domain

Under what circumstances can a new entity-domain be introduced, and an existing one be removed ? Most problems can be solved on the basis of the following mappings.



In concise notation : M11 : R ⟹ RAB, AR, BR
M12 : RAB, AR, BR ⟹ R

This transformation is mainly intended for true N-N relations; for the other connectivities the M2 mappings include M1.

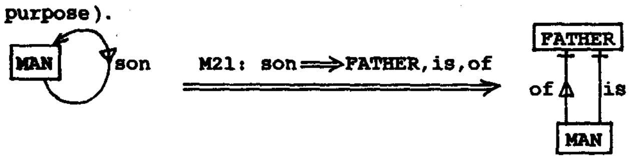


In concise notation : M21 : R $\Longrightarrow$ RAB, AR, BR
M22 : RAB, AR, BR $\Longrightarrow$ R

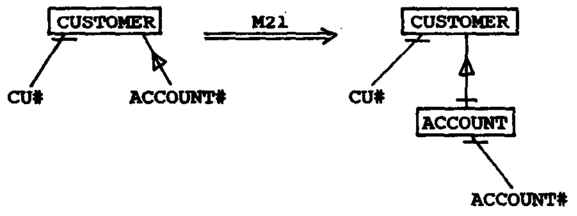
Mappings M11 and M21 are useful to transform constructs that are not accepted by a particular model or DBMS. Such will be the case for N-N relations, repeating and optional attributes and recursive relations in many DBMS. Mappings M12 and M22 make it possible to clean and simplify a schema.

Elementary examples.

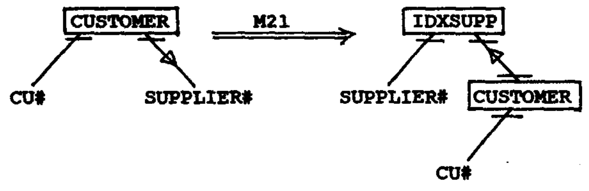
a)Removing a recursive relation (a non-symetrical recursive relation is oriented for designational purpose).



b) Eliminating a repeating, optional attribute.

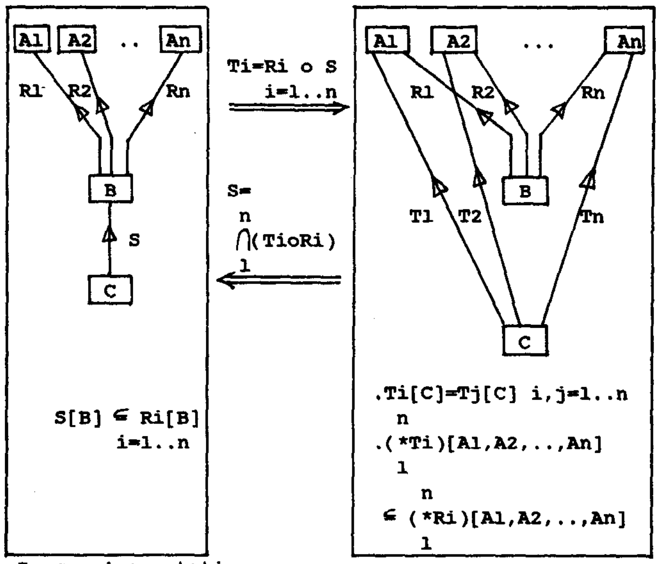


c) Index generation (at the access level).



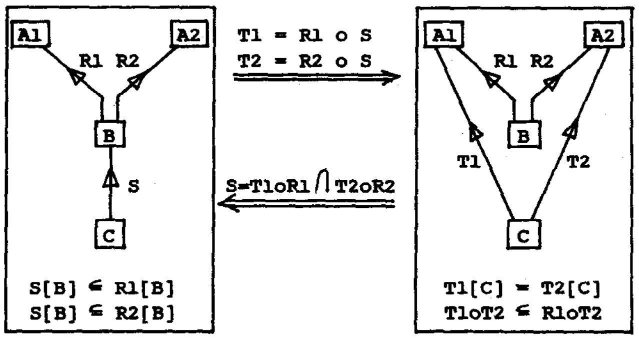
Rotating a relation

These mappings transform a schema by "rotating" a relation from a domain to one or several other ones. The idea is quite simple; if A is to be connected to B, it can be connected to any identifier of B and conversely. This trivial idea gives rise to one of the most powerful mapping family. Its general form is as follows :



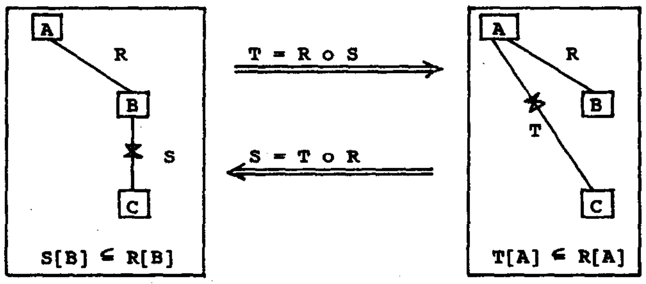
In concise notation:
M31 : S, R1, R2, ... Rn $\Longrightarrow$ T1, T2, ... Tn
M32 : T1, T2, ... Tn, R1, R2, ... Rn $\Longrightarrow$ S

For n = 2, the mapping is a bit simpler :



In concise notation : M41 : S, R1, R2 $\Longrightarrow$ T1, T2
M42 : T1, T2, R1, R2 $\Longrightarrow$ S

For n = 1, R (i.e. R1) must be 1-1. However the mappings are still valid if S is a N-N relation, which is unfortunately false for M3 and M4.

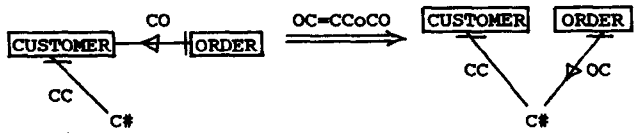


In concise notation : M51 : S, R $\Longrightarrow$ T (symetrical)

Note. Existence constraints on objects in the source schema generally lead to simpler properties in the mappings. For example, if B is imposed existence constraints in Ri i=1..n, the constraints S[B] ⊆ Ri[B] need not be mentioned.

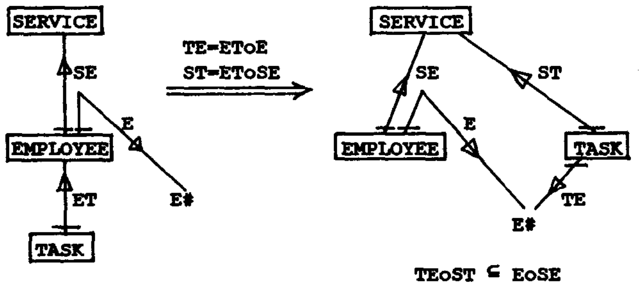
Elementary examples.

a) Removing inter-entity relations.



OC[C#] ⊆ CC[C#]

b) Breaking a hierarchy.



TEoST ⊆ EoSE

5. A LOGICAL ACCESS MODEL AND ITS BINARY EXPRESSION.

Although the DBTG/DDLC schema proposals are a fairly comprehensive set of concepts, a more general and simple model has been preferred : the Logical Access Model [13]. The data structures of this model are quite familiar to application programmers; the data are described in terms of data base, files, records, item values, identifier of a record-type, access-keys and (inter-record) access-paths, etc. (similar concepts can be found in [6])

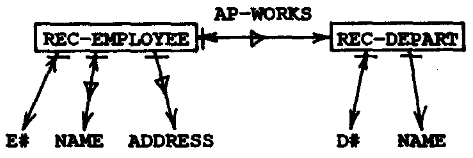
The basic objects to be accessed are the records and the item values. The access mechanisms are from record to item values, from item values to records (access-key) and from record to records (access-path). Given a special, single occurrence, record type called SYSTEM, access-path types from it easily model any kind of sequential access. It should be noted that an access-path is basically directed; if a record a is the origin of an access-path to target records b1, b2, b3, the programmer can sequentially access b1, b2, b3 from a, but cannot access a from, say, b2 via this access-path.

Such structures, however, can be easily formalized by an extended binary model. Its domains are Items (instead of simple domains) and Record-Types (instead of entity domains). Relations on these domains clearly express the access mechanisms mentioned above. Thanks to this formalization, the logical access level can take advantage of the graphical binary representation, the relational expression of integrity constraints and especially the mappings defined above.

The main differences between the pure binary model and the binary logical access model are the following :

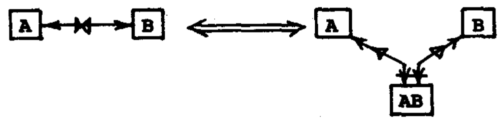
(1) an access relation is directed from the origin domain to the target domain; in the graphical representation, the arcs will also be directed.
(2) the targets of an access relation are ordered.
(3) an access relation can be the inverse of another one; this corresponds to the existence of an inverse access. For example, a CODASYL set type is described by (a) a 1-N access-path type and (b) its N-1 inverse, since the FIND OWNER primitive is always available. In the graphical representation, two inverse access relations are signaled by a special symbol, or, more concisely, by a unique bi-directed arc ; in the latter case, both relations are given the same name.

An example of access structure :



This schema contains the description of : (1) items associated to record-type REC-EMPLOYEE, two of them being access-keys, (2) items associated to record-type REC-DEPART, one of them being an access-key, (3)access-path type AP-WORKS from REC-EMPLOYEE to REC-DEPART and its inverse AP-WORKS from REC-DEPART to REC-EMPLOYEE;

When applying the binary mappings to an access relation which is given an inverse relation, it should be emphasized that each of them can be transformed independently of the other. However the source relations can be transformed as a whole by associating an inverse relation to each target relation. For example, the M1 mappings can be adapted as follows:



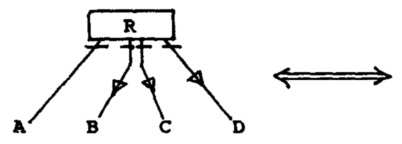
6. FIRST APPLICATION : CODDification of a binary schema.

The problem to be solved is that of the translation of a schema from the binary model into the CODD'S model. We first need a preliminary mapping, given without proof:

any normalized relation R, defined on simple domains, can be represented by (1) an entity domain ER in bijective correspondence with R, (2) binary relations between ER and each simple domains. Connectivity and multi-domain identifiers are derived from the identifiers of R.

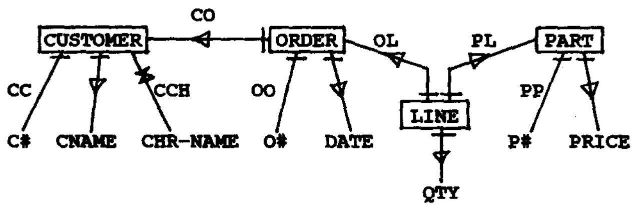
Conversely, a normalized CODD'schema can be generated directly from a binary schema where: (1) each relation is defined on an entity domain and a simple domain, (2) each entity domain participates in at least one relation, (3) entity domains are submitted to existence constraints in each relation in which it participates ("the attributes are mandatory") , (4) no N-N or 1-N relations from entity domains. R is the join of the binary relations.

Example:

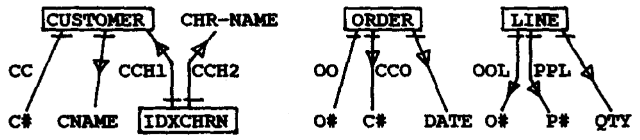


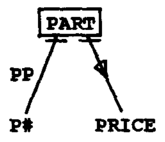
R(A, B, C, D)

Let us solve the problem for the following binary schema :



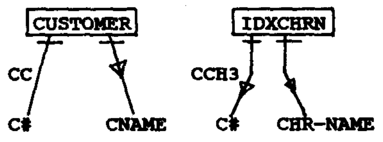
Let us eliminate the inter-entity domain relations :
M51 : CO, CC $\Longrightarrow$ CCO (removes CO)
M51 : OL, OO $\Longrightarrow$ OOL (removes OL)
M51 : PL, PP $\Longrightarrow$ PPL (removes PL)
and the optional and repeating attribute CHR-NAME :
M11 : CCH $\Longrightarrow$ IDXCHRN, CCH1, CCH2





CCO[C#] ⊆ CC[C#]
OOL[O#] ⊆ OO[O#]
PPL[P#] ⊆ PP[P#]

The rotation of CCH1 from CUSTOMER to C# by
M51 : CCH1, CC $\Longrightarrow$ CCH3 leads us to:



CCH3[C#] ⊆ CC[C#]

and gives a schema ready for the final production :

CUSTOMER(C#, CNAME)	
IDX-CHRN(C#, CHR-NAME)	IDX-CHRN[C#] ⊆ CUSTOMER[C#]
ORDER(O#, C#, DATE)	ORDER[O#] ⊆ CUSTOMER[C#]
LINE(O#, P#, QTY)	LINE[O#] ⊆ ORDER[O#]
PART(P#, PRICE]	LINE[P#] ⊆ PART[P#]

Comments

1. The process is systematic and therefore easily performed by an algorithm. Since it is reversible, a similar algorithm can be designed to generate a true binary schema from a CODD'S schema.

2. Translating an E/R schema into a CODD'S schema is an easy task as well; the reverse translation, however, is more difficult since in some cases, distinguishing entities from relationships has to be performed at the semantic level.

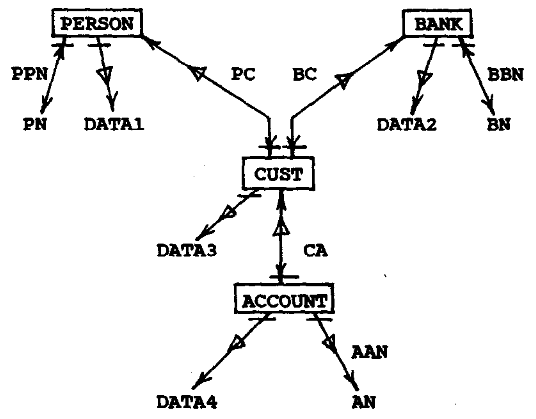
3. A binary (or E/R) schema can be translated if every entity-domain (or Entity-type) can be identified, directly or not, by one or several simple domains. Translation rules for the three models can be found in [12].

4. Such problems have also been studied in [2].

7. SECOND APPLICATION : FROM A LOGICAL ACCESS SCHEMA TO A DBMS SCHEMA

The following example is only intended to demonstrate the use of the mappings in a complex design process. The main objective will be the translation of a general access schema into data structures valid for a given DBMS. The TOTAL system has been chosen in this example because its restricted data structures (compared with CODASYL) raise interesting problems.

Let us suppose that the result of the first algorithm design (3rd step of the Namur Approach) has led to the following access schema.



A record PERSON describes a person and a record BANK describes a banking company; each of them is identified by a number, PN for PERSON and BN for BANK. A person who has at least one account in a given bank is known to be a customer of this company and this status is described by a record CUST to which the records ACCOUNT describing the accounts of the person are linked. An account is identified in a bank by an account number (AN); this can be declared by: "AAN and BCoCA (or BBNoBCoCA) are an identifier of ACCOUNT". On the other hand, given a bank identifier BN and a value of AN, it would be possible to access the unique corresponding record.

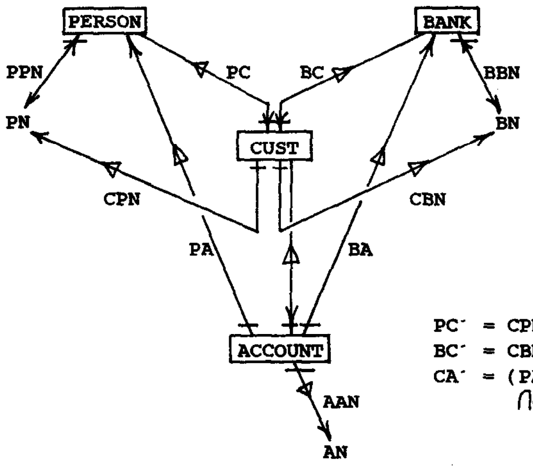
The DATA of each record type will be ignored except for the final schema; for reasons of simplicity, the inverse N-1 access-path types will be given specific names: PC', BC' and CA'.

It will be useful to briefly specify the TOTAL data structure restrictions in terms of the logical access model concepts. TOTAL knows two classes of record types; the MASTER record type, to which an item, which is an access-key and an identifier, must be associated, and the VARIABLE-ENTRY record-type that has no such item. One-to-many access-path types without explicit inverse can be declared from a Master record type to a Variable-entry record type; the inverse access is automatically made available by associating the access-key of the origin with the target record-type.

Solving the problem consists in transforming the structures of the source schema which cannot be directly translated in the TOTAL data structures.

1. TOTAL doesn't tolerate explicit N-1 access-path types. This constraint will be satisfied by applying M51 mapping to the N-1 access-path types PC', BC'. The problem of CA' is more complex; let us try the M41 mapping first.

M51 : PC', PPN $\Longrightarrow$ CPN (removes PC')
M51 : BC', BBN $\Longrightarrow$ CBN (removes BC')
M41 : CA', PC, BC $\Longrightarrow$ PA, BA (removes CA')



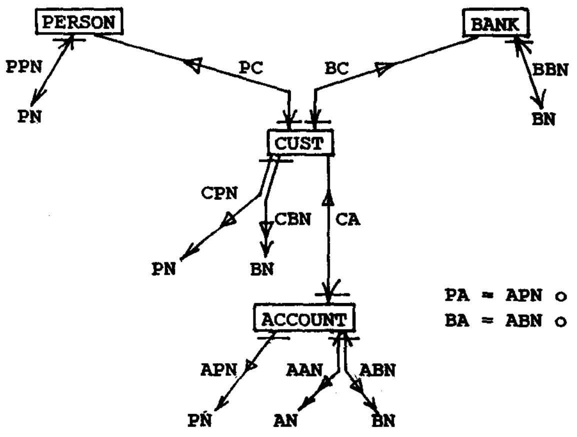
PC' = CPN o PPN
BC' = CBN o BBN
CA' = (PAoPC) $\cap$ (BAoBC)

Remark.
Not only does the transformation ensure semantic equivalence, as in a conceptual schema but it also ensures access equivalence since PN and BN are access-keys.

The problem of PA and BA which are explicit N-1 access-path types can be solved easily by applying

M51 : PA, PPN $\Longrightarrow$ APN (removes PA)
M51 : BA, BBN $\Longrightarrow$ ABN (removes BA)

A cleaned and completed schema can then be proposed.

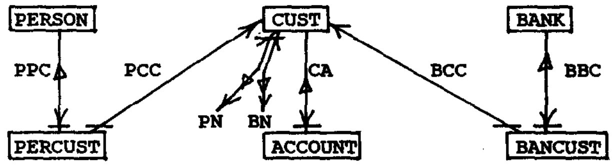


PA = APN o PPN
BA = ABN o BBN

Since (AAN,BBNoBCoCA) is an identifier of account, and ABN= BBNoBA=BBNoBCoCA', (AAN,ABN) is also an identifier of ACCOUNT. Moreover, the access to ACCOUNT from AN and BN can now be stated explicitly by an access key.

2. TOTAL doesn't tolerate the Target of an access-path type (CUST in this case) being itself the origin of an access-path type. Inserting new record types into PC and BC allows CUST to be put at the top level:

M21 : PC $\Longrightarrow$ PERCUST, PCC, PPC
M21 : BC $\Longrightarrow$ BANCUST, BCC, BBC

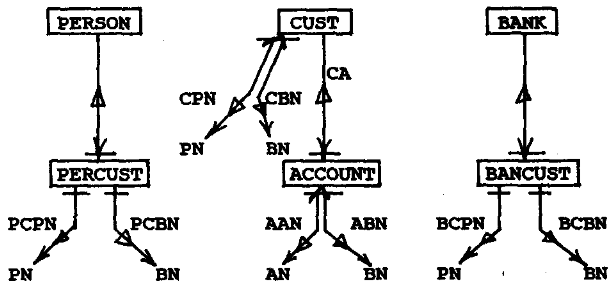


PC = PPC o PCC
BC = BBC o BCC

Being now a Master record-type, CUST must be associated with an identifier/access-key; (PN, BN) has of course been chosen.

3. For the same reason as in 1. (explicit N-1 access-path type) , PCC and BCC must be transformed, e. g. by rotation:

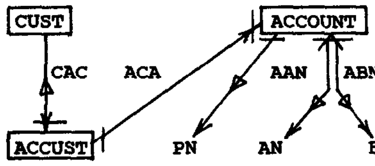
M41 : PCC, CPN, CBN $\Longrightarrow$ PCPN, PCBN
M41 : BCC, CPN, CBN $\Longrightarrow$ BCPN, BCCN



PCC = (PCPNoCPN)∩(PCBNoCBN)
BCC = (BCPNoCPN)∩(BCBNoCBN)

4. The target of an access-path type is a Variable-entity record-type and hence cannot be associated with an access-key. Due to its access-key, ACCOUNT must then be made a Master record-type. As in 2, a M21 record-type insertion is a solution:

M21 : CA ⟹ ACCUST, ACA, CAC

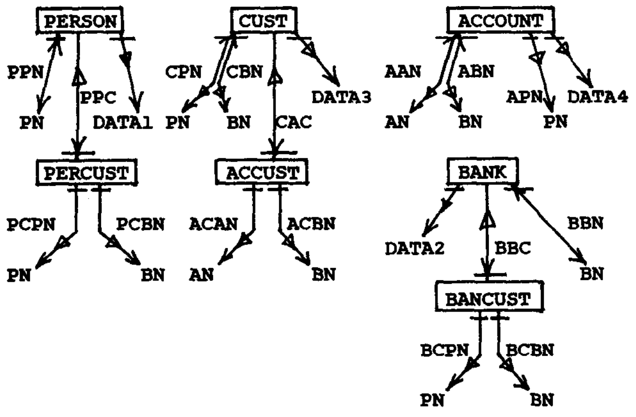


CA = CAC o ACA

N-1 access-path type ACA will be transformed by :

M41 : ACA, AAN, ABN ⟹ ACAN, ACBN
so that ACA = (ACAN o AAN)∩(ACBN o ABN)

Hence the final schema :



The translation rules between the source schema and the final schema are the following:

- PERSON,BANK,CUST,ACCOUNT,PPN,BBN,AAN : no change
- access key to ACCOUNT : (AN via AAN, BN via ABN)
- PC = PPC o ((PCPNoCPN) ∩ (PCBNoCBN))
  PC′= CPN o PPN
  BC = BBC o ((BCPNoCPN) ∩ (BCBNoCBN))
  BC′= CBN o BBN
  CA = CAC o ((ACANoAAn) ∩ (ACBNoABN))
  CA′= (CPNoPPNoPC) ∩ (CBNoBBNoBC)

Some integrity constraints are not made explicit in the schema above; they are derived from the successive mappings and can be most simply expressed by "PC′ is inverse of PC", "BC′ is inverse of BC" and "CA′ is inverse of CA".

Conclusions

This schema can be immediately translated into a TOTAL DBDL text and in programming rules that will ensure valid data base state with respect to the integrity constraints.

Furthermore, the algorithms of the application programs can be expressed as working on the first access schema which is both simpler and clearer than the final one. The actual programs can be obtained essentially by two methods. The first one consist in hand-translating each statement making use of a transformed substructure (PC,PC′,...) according to the translation rules given above; each of these algebraic rules corresponds to a very simple algorithmic expression. The second method is more complex but provides the application programs with a high level of logical data independence. It consists in translating each access statement into a call to an access module dedicated to this database and that carries out the mapping dynamically. Using access modules is common practice in file and d.b. programming; the above process, however, gives acurate and strict rules for the design of such modules.

Although this schema has probably good performance, our only goal was to obtain a TOTAL schema regardless of any performance criteria. Other TOTAL schemas can be derived by transforming the invalid substructures in another ordering; on the other hand a TOTAL schema can be transformed into another TOTAL schema, for instance by splitting or merging record-types or by migrating items. Such a process can be considered as a "valid TOTAL schemas generation"; it can be easily automated (this concept is also described in [15]).
It is worth noticing that the final schemas are complete in that they contains the data structure description, the translation of the initial integrity constraints and the derived integrity constraints as well.

8. CONCLUSION

The paper is mainly concerned with the proposal of a set of reversible transformations that are both powerful and intuitive. They have been defined for a binary model. This model, however, is considered as a particular case of a generalized relational model. Consequently, the transformations are also valid for other models such as CODD′S and E/R models, considered as other particular cases of the generalized model. Two applications of the transformations have been developed within the framework of the Information System Design Approach of Namur. The first application is concerned with the translation of a conceptual schema from a model into another and the second application is concerned with the adaptation of a logical access schema to a given DBMS. An interesting feature of the transformation is its ability to automatically deduce integrity contraints that are frequently forgotten in manual design procedures. The generality of the model and of the transformations makes them well suited to the expression of such classical practices as record-type splitting, record-type merging, conversion of flat files from network structure and conversely, item migration, removing of N-N or recursive relationship, etc ...

Besides its educational advantages [12], the transformation can be starting point of general or DBMS oriented functions of a CAD system for Data Bases development.

9. REFERENCES

[1] ABRIAL J.R., "Data Semantics", in Data Base Management, North-Holland, 1974.

[2] ADIBA, DELOBEL, LEONARD, "An unified approach for modelling data in logical data base design" in Modelling in data base management systems, North-Holland, 1976.

[3] BENCI, BODART, BOGAERT, CABANES, "Concepts for the design of a conceptual schema", in Modelling in data base management systems, North-Holland, 1976.

[4] BODART, PIGNEUR, "A model and a Language for functional specifications and Evaluation of Information System Dynamics" in Formal Models and Practical tools for Information Systems Design, North-Holland, 1979.

[5] BRACCHI, PAOLINI, PELAGATTI, "Binary Logical Associations in Dataü Modelling" in Modelling in data base management systems, North-Holland, 1976.

[6] CABANES A., "Rapport Indèpendance dans les SGBD", Institut d' Informatique d'Entreprise, CNAM, Paris (Mars 1977).

[7] CHEN P.P., "The Entity-Relationship Model : toward a unified view of Data, ACM TODS 1,1, 1976.

[8] CODASYL DDLC report. Information Systems 3, 4, 1978.

[9] CODD E.F., "A Relational Model of data for large, shared Data Banks", CACM, 13, 6, 1970.

[10] CODD E.F., "Extending the data base relational model to capture more meaning", IBM RJ2472, 1979.

[11] HAINAUT, LECHARLIER, "An extensible semantic model of data bases", Proc. IFIP Congress 1974, North-Holland.

[12] HAINAUT, "Modèles conceptuels", Lecture notes, Public. Institut d'Informatique, Namur, 1980 (french).

[13] HAINAUT, "Un modèle de description de fichiers : le modèle d'accès", Lecture notes, Public. Institut d'Informatique, Namur, 1980 (french).

[14] HALL, OWLETT, TODD, "Relations and Entities", in Modelling in Data Base Management Systems, North-Holland, 1976.

[15] IRANI, PURKAYASTHA, TEOREY, "A designer for DBMS-processable logical database structures", Proceedings VLDB, 1979.

[16] MITOMA, IRANI, "Automatic Data Bases Schema Design" Proc VLDB 1975, ACM (1975). [17] PIROTTE, "Explicit description of entities and their manipulation in languages for the relational data base model", Thèse de doctorat - Universitè libre de Bruxelles, 1976.

[18] TARDIEU, NANCI, PASCOT, "Conception d'un système d'information", Les Editions d'Organisation/Gaëtan Morin, 1979.

[19] TEOREY, FRY, "The logical access approach to database design", Computing Surveys, 12, 2, 1980.

[20] "Data Structure Models for Information Systems", "Proc. Intern. Workshop of Namur, 1974, Presses Universitaires de Namur (1975).