Serdica
Bulgariacae mathematicae publicationes

Сердика
Българско математическо списание

COMMON INFORMATION MODELS IN DATA BASE MANAGEMENT SYSTEMS*

G. BRACCHI, G. PELAGATTI

A multilevel data base architecture supports different classes of users with different data models. In this environment the features of the Conceptual Schema (the common logical data description) play a crucial role.

Two of the main requirements for the Conceptual Schema are ability to support mappings to the different external views and aptitude to constitute a documentation for interpretation of data semantics. Emphasizing the one or the other of these conflicting requirements leeds to splitting the Conceptual Schema into two new Schemata: the Enterprise Schema and the Central Schema.

In this paper the functions of these schemata are discussed, and the requirements for suitable data models are derived. The evaluation of the existing data models with respect to the two schemata is conducted through analytic discussion of relevant problem areas.

1. Introduction. One of the goals of a Data Base Management System (or briefly DBMS) architecture is to provide different users with different views of data (the External Schemata) at the application program interface [2; 4; 14; 15]. The resulting DBMS architecture is shown in Fig. 1. The system's common view of data is contained in the Conceptual Schema that constitutes a description of the logical structure of all data collected in the database.

The Conceptual Schema representation is intended to be relatively independent of any particular application (External) use of the data, and also independent of strategies for physically storing of the data. The description of how data are actually stored is contained in the Internal Schema.

Operations that are issued by the users with reference to their External Schemata are translated into operations on the Conceptual and the Internal Schema level by the DBMS. Such an architecture requires then a Conceptual Schema data model that makes easy and flexible this translation of operations (mapping). On the other hand, if the Conceptual Schema is the only common description of the logical data structure, it is required also that it constitutes a reference for the users, the Data Base Administrator and all personnel that needs interpretation of the meaning of the data.

This multilevel architecture makes irrelevant the debate that has been going on in the recent years on which is the most suitable model at the application program interface [12]. On the other hand, the problem arises of determining viable data models for the Conceptual Schema. Although a number of researches have been undertaken in this subject no agreement has been found yet [4; 5; 8; 10; 11; 12; 18].

This lack of agreement is partially due to the fact that the two criteria of supporting the mapping between different representation levels and of being a

*Delivered at the Conference on Systems for Information Servicing of Professionally Linked Computer Users, May 23-29 1977, Varna.

SERDICA Bulgaricae mathematicae publicationes. Vol. 4, 1978, p. 184–196.

reference for the semantic interpretation of data can lead to different choices for the data model in the present state of the art.

Therefore, a modification of the architecture of Fig. 1 seems to be necessary and two different Schemata have to be considered, at least in principle,

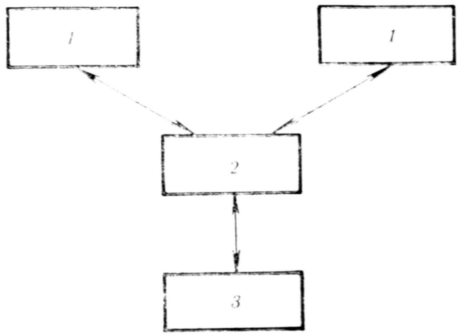

Fig. 1 Multilevel DBMS architecture
1 — External Schema; *2* — Conceptual Schema; *3* — Internal Schema

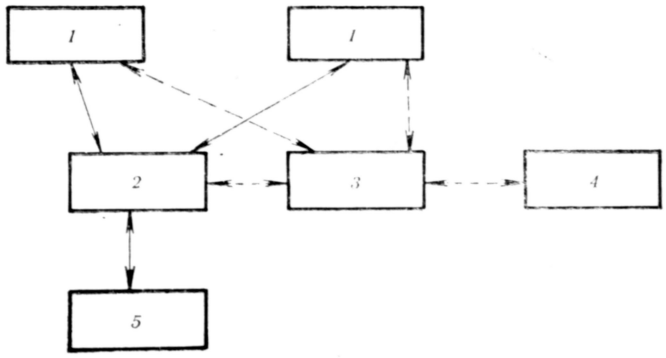

Fig. 2. The role of the Enterprise Schema and of the Central Schema.
1 — External Schema; *2* — Central Schema; *3* — Enterprise Schema; *4* — Real World; *5* — Internal Schema

for the level of common logical data description: the Central and the Enterprise Schema. In Fig. 2 the resulting architecture is shown. In this paper the functions of these two schemata are discussed, and consequently the requirements for suitable data models are derived, and the essential problems that must be decided in order to evaluate the candidate data models are discussed.

2. **The enterprise schema.** Let's first consider the role of the Enterprise Schema and determine what makes it different from the Central Schema. While the Central Schema is directed towards DBMS operation, the Enterprise Schema is directed towards the people who use the data base.

The information in the Central Schema is therefore structured in a way that is suitable for the mapping, whereas the Enterprise Schema is structured in a way that can be more easily understood by human beings.

In fact, the Enterprise Schema is the result of the system analysis and design phase; this result is a complete image of the organization phenomena of interest which will be represented in the information system.

Let's consider an example in order to clarify the above statements. Let's suppose that we have in the Enterprise Schema data about the following facts:

f1) *is-male*
f2) *is-female*
f3) is-*married-to*
f4) *is-husband-of*

Fact f4 can be derived from the other three, because the following semantic property holds:

p1) $\forall$ *x, is-husband-of (x, y)* $\langle\ \rangle$ *is-married-to (x, y)* $\wedge$ *is-male (x).*

We can say, at a very general and intuitive level, that it would be correct to include the four facts and the property p1 in the Enterprise Schema, whereas in the Central Shema it would be better to have only a non-redundant subset of facts (for instance f1, f2 and f3). This concept, that has now been stated intuitively, will be defined more precisely in the rest of this paper.

2.1 Functions of the Enterprise Schema. The Enterprise Schema is used for two major functions: in the data base analysis and design phases it constitutes a description of all information that should be contained in the data base: during operation of the data base it acts as a documentation of the meaning of all data descriptions (all other Schemata) that are used in the data base. Now, before considering the requirements that a data model must satisfy in order to be used for both above mentioned functions, two preliminary questions must be answered.

Question 1. Why should the Enterprise Schema be a real Schema (in computer readable form) and not a simple paper documentation?

One answer to this question is that the size of the documentation that is required in the data analysis process is such that automatic documentation is required; this automatic documentation should not be completely disjoint from the DBMS (as it is the case today if for instance an independent "Data Dictionary" system is used in the design of a data base) in order to allow a certain degree of automatic control of correctness of the mappings between the diferent Schemata. In fact, there is a strong interrelationship between the documentation of the meaning of data and the automatic checking of consistency constraints, as it will be shown at the end of this section.

Question 2. Why cannot the Central Schema contain all the information that is contained in the Enterprise Schema?

In principle, using the Enterprise Schema directly as the Central Schema this would be possible, but in fact today's state of the art forces the DBMS designer to choose for the Central Schema a simpler data model in the sense that this model should be more structured and contain less redundancy than it is necessary for performing the functions of the Enterprise Schema. (See Section 2.5).

2.2 Requirements for the Enterprise Schema as a Design Tool. The analysis and design of a data base in a real life environment is characterised by the large volume of data that must be handled, such that it

is unusual that a single person has a complete control of the data, and therefore a large number of persons are involved in the project. As a consequence, the specifications that are collected are redundant, incomplete and ambiguous. For these reasons, the data model that is used should satisfy the following requirements:

1) it must be formal, in order to permit automatic documentation;

2) it must permit incremental specification, and therefore allow redundancy, besause the new requirements that are collected should not modify the existing ones even if they partially overlap;

3) it must allow different views to coexist, and the correspondence between them (cross view semantics) to be defined;

4) it must allow a description of the structure of the problem without unnecssary details.

There is another requiremet, which should hold for the data models at every level, but is particularly important for the External Schema:

5) it must be easy to formulate and to understand. The problem with this repuirement is that it is difficult that different people agree on which data model satisfies this requirement best: in fact only experimental evidence could provide an answer.

2. 3 Requirements for the Enterprise Schema as a documentation of the semantic meaning of other schemata. It is necessary to have, while the system is working, a tool to interpret the meaning of the various Schemata. The translation between the External Schemata and the Conceptual Schema must rely on some definition of the data equivalence between two different Schemata. Most work on data translation relies on an intuitive definition of data equivalence; however, this is inadequate to avoid ambiguities and misunderstandings. The Enterprise Schema should provide tools for assuring that the translations correspond to the intended meaning of the data in the various Schemata and to the meaning of data in the intuition of the data base designers. For this reason we can add the following requirement to the previous ones:

6) the data model and language for the Enterprise Schema should be suited to express the meaning of data structures.

2. 4 Data models for the Enterprise Schema. One of the most elementary definitions that could be given for an Enterprise Schema is that it is the union of a Central Schema and a Dictionary.

However, this definition, although it stresses the important aspect that the problems that are connected with naming conventions should be put in the Enterprise Schema, seems to be too restrictive, because it does not satisfy requirement 6 and it fulfills only partially (depending on the quality of the Central Schema) requirements 3 and 4.

A model that satisfies the requirements for the Enterprise Schema should be based on the semantic structure of sentences instead of on the usual concepts of data definition. Some researches have taken this approach [3; 11; 19].

Although these studies have drawn different conclusions with respect to the proposed models, a few concepts have emerged that are underlying all the approaches.

In the following the most important concepts that should be applied to the model for the Enterprise Schema will be briefly explained. The data collected in a data base describe in some way a state of the external reality (real

world state). A state description can be given by a set of predicative sentences. A predicative sentence is built by a noun phrase and a verb phrase [3].

A way to build a model of the reality is to map:

1) everything which is described by a noun phrase into an entity, and

2) verb phrases into predicates.

Different solutions can be applied with regard to this second point. In the following we will give a few examples of the complexities that arise in mapping verb phrases into predicates. A similar discussion can be found in [3].

Verb phrases can be mapped into two different constructs: types (unary predicates) and relations (*n*-ary predicates). In a given state, the set of entities that belong to the same type is called its population [9].

In order to be meaningful a state must satisfy many constraints, especially with respect to the types in which a given entity participates (for instance, an entity can be a member of both types *is person* and *is male*, but not of *is white* and *is black*).In fact, these constraints are very complex and difficult to express, and many data models (as we will see in Section 3) don't consider them at all, for instance by requiring that only disjoint types be contained in a data base).

Let's consider a few examples of these problems:

1) Relationships between types: for instance the type *is-Person* and the type *is-Employee* are related in such a way that insertion of an entity into the population of *is Employee* requires also its insertion into the population of *is-Person.* This relationship is simple because it means simply inclusion of *is-Employee* into *is-Person*, but any kind of set relationship can exist between different types.

2) Completeness of type definition: a type is completely defined if its elements in all possible states (all possible populations) are defined. Types can be completely defined only by axioms or by enumeration.

For instance *is-month* is defined by enumeration {January, February...}, *is-salary* can be defined by axioms, but *is-person* cannot. Ultimately the definition of all types is reconducted to a few basic types: *is-integer*, *is-string*, etc..., that are already defined.

3) Identifying properties: for instance let's consider a type *is cat.* An entity of this type can be *the white cat of Bob.* If a person can be owner of only one cat of a given colour, we can say that the two relations *has-colour* and *is-owned* are an identifying set for the type *is cat.* A type can have many identifying sets. If a type does not have any identifying set and is not a complete type, an artificial identifying relation is needed.

4) Domains of relations: the domain of a relation is usually required to belong to a type, i.e. for every allowed state the projection of a relation on a domain has to be a subset of the population of a type.

The above examples don't cover all the possible constraints on a data base state; a few other can be found in data models. In fact in order to express all possible semantic properties of data the whole power of predicate calculus is needed.

2.5 A Schema: a tool for documentation or for enforcement of constraints? At this point we can resume the first question that was posed in Section 2.1 and reconsider the relatiorship that exists between two aspects of a Schema: the documentation and the constraint aspect. A Schema can be defined as a set of axioms that restrict the set of allowed states of

the data base; on the other hand, it is also used as a documentation tool. We have seen that the opportunity for considering the Enterprise and the Central Schema as two different Schemata derives from the need of emphasising differently the two aspects in the two Schemata; on the other hand, one would wonder why the separation should not be complete so that the Enterprise Schema becomes a pure documentation tool. The answer has emerged from the previous discussion of the data model for the Enterprise Schema: the axioms that define the allowed states of a data base are determined from the meaning of the information; they are a precise way of expressing this meaning. Therefore, a Schema is also naturally a documentation tool.

With respect to any Data Definition Language we can say the following: every clause that does not carry information that is used by the DBMS in order to determine if an operation is allowed belongs to the pure documentational aspect.

At today's state of the art, many facts that must be collected in the Enterprise Schema cannot be used by the DBMS to control semantic consistency of the data base state: therefore many features of the Enterprise Schema remain still pure documentation.

A last remark on the Enterprise Schema: in order to be a Common Schema for all users of the data base, the Enterprise Schema should contain *basic facts*. The definition of what basic facts are will be given in the next section (on the Central Schema) because this feature is more relevant to the Central Schema than to the Enterprise Schema.

3. The central schema

3.1 *Requirements for the central schema*. From the architecture shown in Fig. 2 and the above discussion of the Enterprise Schema it should be clear that the main requirement for the Central Schema is that it has to be suitable for the definition and implementation of the mappings between the different representation levels of the data base.

Let's now consider how the mapping between different levels can be implemented. A mapping from an External to a Central Schema consists of establishing a correspondence between an operation at the external level and a sequence of operations at the central level.

The simplest way of achieving this goal is explicitly writing, for each External Schema, the sequence of *central operations* corresponding to each operation on it. This method has the obvious advantages of flexibility and of minimal requirements on the DBMS. The disadvantage is that each new External Schema definition requires the writing of a program (possibly a not simple one), and no advantage can be taken from the fact that mappings between similar External Schemata and the same Central Schema require very similar programs.

A more sophisticated solution is to *factorize* the common characteristics of the External Schemata, i. e. to classify them in categories like *relational*, *network*, *hierarchical*, with common sets of operations having a common meaning. Let's now suppose that it is possible to define a generalized way of translating any operation on a category of External Schemata into a sequence of operations on the Central Schema. In this case the definition of an External Schema could be performed simply by establishing a *structural* correspondence between its components and the components of the Central Schema. This correspondence, together with the generalized mapping procedure, would be sufficient for translating any external

operation. The advantage of this method is the easiness of the External Schema definitions and the possibility of using optimal generalized mappings. The disadvantages are the lack of flexibility (an External Schema of a new type could not be defined without constructing the generalized mapping procedure) and the difficulty of creating the generalized mappings themselves.

In [16] the above two mapping methods have been called Operational Mapping Definition and Structural Mapping Definition.

Now, the main requirement for the Central Schema data model can be stated as follows: this model must allow the definition of a powerful set of operators that can be easily used as primitives for the Mapping Definitions. Other requirements for this model (like easiness of formulation and understanding) seem to play a secondary role with respect to the previous one.

A very useful feature that can be added to the above requirement and in some way completes it, is that the interpretation of the Central Schema using the Enterprise Schema (a manual mapping, probably) should be possibly easy and straightforward. This means that it is useful to have a clear correspondence between the concepts ot the Central and those of the Enterprise Schema. A review of the data models that have been recently proposed for the Central Schema indicates that the following problem areas are relevant with respect to the choice of the data model for the Central Schema:

1) Distinction between Entities and Values (or Entity Names): how is it possible to refer to Entities that do not have corresponding values (see Section 3.2);

2) Approach to the problem of types (see Section 3.3);

3) Use of Basic Constructs or Complex Constructs at the Central Schema level (see Section 3.4);

4) Choice of a Basic Construct: Binary or Irreducible Relations (see Section 3.5);

5) Opportunity of introducing redundancy in the Central Schema (see Section 3.6).

The above problems are to some extent orthogonal so that different models have chosen different combinations of solutions; this fact partially explains the existence of a variety of models.

3.2 Entities and Values. We have seen in Section 2.4 that an Entity is everything to which one can refer with a noun phrase. For instance the number 10 is an entity, the name *Mr. Smith* is an entity and *the* 10 *years old son of Mr. Smith* is an entity. Complex relationships exist between the above entities (in particular the identifying sets *of age* and *son of*) and we have seen that it is meaningful to represent them in the Enterprise Schema. If we now consider the Central Schema an easier representation is needed. Very simple data models require that every Set of Entities has a corresponding Set of Values. As a result of this restriction the modelling power is lower but many difficulties are avoided.

A different solution to the above problem is given by considering some special values which are suited for the representation of objects that don't have any corresponding usual values. These are called Internal Domains in [1], Surrogates in [13] and Internal Sets of Concepts in [5]. The advantage of this approach is for instance to allow considerations of relations over domains not having usual values such as Persons, whereas otherwise only Person-Numbers or Person-Names could be considered. This feature is

very useful at the Central Schema level, because different Users (and therefore different External Schemata) could refer to Persons either by Person-Number or by Person-Name. Moreover, the possibility of changing the Person-Number or the Person-Name of a same Person is provided in this way [16].

3.3 Approach to types. A first distinction can be operated between those models that contain types (and their populations) and associations between types, and those models that only contain relations (and the type population is therefore restricted to be in any state the projection on the same domain of the relations). This distinction has been widely discussed in [17]. The most important aspect of this distinction is that in the first class of models (let's call them Type/Association models) there are type operations (e. g. Insert and Remove an Object from a Type) and Association Operations (e. g. combine two Associations and build a new Association), while in the second class (let's call them Relational Models) there are only Relation Operations.

It can be immediately seen that the first class is nearer to the view of the Enterprise Schema and is more powerful in modelling, but the implementation is more complex due to the consistency constraints that must be enforced; the second class is nearer to the views that are common in the traditional data models and therefore to the views of the External Schemata.

A second distinction can be operated with respect to the degree of sophistication that is allowed in modelling the properties of types. Very few attempts have been done to provide a real possibility in this sense. The DDL ENALIM [15], for instance, which belongs to the Type/Association class, considers the possibility of overlapping types, as for instance *is Person*, and *is Man, is Woman.*

In relational models, as for example in the well known *n*-ary relational model of Codd [6; 7], we can identify the concert of type as being equivalent to that of fundamental or underlying domain. Most relational models imply that fundamental domains are disjoint, thus giving a restrictive solution to the problem of types. A confusion seems to exist on this point, due to the fact that fundamental domains play a secondary role in relational models: in fact the projections of relations on domains are not necessarily disjoint, while the fundamental domains are disjoint. In the relational literature it is very often obscure whether by the word domain the underlying domain or the role domain are intended.

A join between relations is allowed only if they have a Common underlying domain; it is implied by this fact that the two involved role domains in the relations to be joined are a subset of the same underlying domains. However, this is the simplest way in which types can be related to each other: one type (the common underlying domain) is the inclusion of all the role domains that belong to it.

3.4. Basic constructs or complex constructs. A basic construct is a construct that cannot be decomposed in more elementary ones. It is recognised that basic constructs are, for example, Irreducible Relations [13] and Binary Relations [1; 4; 5; 18]. This aspect will be considered in the next section.

The use of Bisic Constructs in the data model for the Central Schema has been considered because it is apparent that if two different users have different external complex constructs, the only way of mapping them to a set of central complex constructs is to decompose and then reassemble them [5]. The

basic constructs are therefore considered as building blocks for composing external complex constructs.

In [17] the necessity of having complex constructs (called Complex Objects Types) in the Central Schema is proposed for integrity reasons. It is in fact true that allowing really different users views could introduce problems of integrity; however, the Complex Object Type idea cannot constitute a general solution: in this way the problem is avoided by forcing the users to be restricted to the complex object types that are in the Central Schema.

The problem can be restated as follows: the need of providing different users with really different external views has been widely recognised, but the integrity problems that arise if this aim is pursued have not yet been solved. Therefore, it is reasonable to apply at today's state of the art some structuring restrictions, as with Complex Object Types; however the claim that in principle providing really different views implies introduction of basic facts in the Central Schema seems to be correct.

An important problem that has not received up to now an adequate consideration is the determination of how much different users' views can be independent from each other; in fact a user will have to take care to some extent of the interference of other users upon the data base.

Cases of interference in a multiple external models environment arise with the operations that change the state of the Central Data base, like the update, insert, delete, and lock operation. For example, if an external user utilizing a hierarchical model deletes a node, it must be decided if that imply that no other external user, even utilizing a different model (e. g. a relational model), and sharing the same data, can view any part of the subtree that originated at that node. In the real situations, depending on the semantics of data, sometimes this will be true, sometimes the deletion of the subtree will affect only users of the hierarchical model.

In general, knowlege has to be gained on what an operation on one external data model means in terms of another data model. For instance, today's external data models have different behaviour with respect to locking procedures (i. e. procedures for preserving data base integrity under concurrent usage): therefore mappings between External and Central data bases must be able to support multiple locking policies on a data model in order to handle interference of different locking procedures on different models.

In general, these mappings should be able to handle cross data model operations: in order to ensure correctness of these operations, tools for defining cross data model semantics have to be developed.

3.5 Basic facts: irreducible or binary relations. Having shown that basic constructs play an important role in modelling the Central Schema, we will comparatively discuss the two basic constructs that have recently received a wide attention: the binary model [1; 4; 5; 18] and the irreducible relations model [13].

In [13] the existence of irreducible relations is claimed, i. e. of relations that cannot be further decomposed to become binary without some artificial trick. In the following we will discuss an example and try to show that this artificial trick (called in [13] excess entity set) can lead to better modelling capabilities than the use of irreducible relations.

An example of the existence of irreducible relations is the ternary relation shown in Fig. 3 (the arrows represent functional dependencies).

In fact, in order to reduce this relation to a set of binary relations, a new Set (Internal Set) is needed, for instance E (exams), as it is shown in Fig. 4.

In order to represent completely the reality of Fig. 3, some constraints representing the functional characteristics of the relations must be added, as:

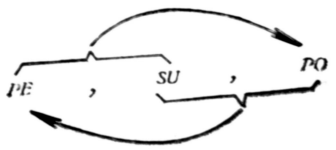

Fig. 3. An irreducible ternary relation. PE=Person, SU=Subject, PO=Position

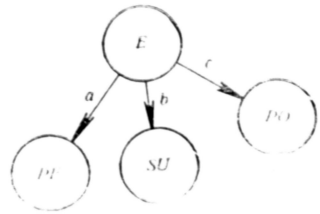

Fig. 4. The three relations corresponding to the ternary relation of Fig. 3

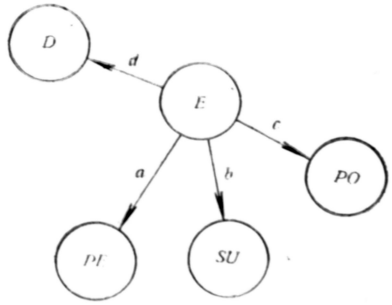

Fig. 5. Addition of a new domain in the binary model

1) relations a, b, c are functional dependencies, and 2) $(PE, SU) \rightarrow E$ and $(SU, PO) \rightarrow E$ are functional dependencies.

It seems that the binary solution of Fig. 4 introduces 2 disadvantages, namely:

1) an internal set is needed;

2) the definition of constraints is more complex.

Let us now extend the example to consider the fact that we want to know the D (date) of every examination.

In the binary model there is a straightforward solution, that is shown in Fig 5.

Only one constraint has to be added, if we suppose that D is functionally dependent on E (other dependencies, for instance transitive dependencies, can be derived, but the problem of transitive dependencies is not discussed in this example).

In order to extend the irreducible model, the relation

can be added. But in this case it is impossible to derive the other existing functional dependency $(SU, PO) \rightarrow D$. A solution could be to add a third relation

but now the problem of equivalence arises, because the date retrieved by finding PE given a certain pair (SU, PO) and then taking the functional dependency $(PE, SU) \rightarrow D$ must be the same as the date retrieved by $(SU, PO) \rightarrow D$.

Summarizing the discussion, we can say that analysis of the difficulties found in extending the irreducible schema shows that they origin from the following facts:

1) The concept D is associated with E in reality (in the world of the users);

2) Different users use different Primary Keys.

By separating the definition of the functionality of the relations (and of groups and chains of relations) from the definition of the structure and by using Internal Sets to represent concepts that the users consider independent of particular value sets, the resulting model allows more flexible changes and more easy mapping to different External Schemata.

3.6. The problem of redundancy. It is assumed that redundancy should be avoided in order to avoid consistency problems. One type of redundancy is transitivity, because a fact is expressed directly and through a transitive equivalence to other facts. In this section an example will be presented that shows the need of introducing a certain degree of transitivity in the Central Schema in order to satisfy user's requirements.

Let us suppose we have three domains: Employee (E), Department (D), and Manager (M), and three one to many binary relations (functional dependencies):

Department of Employee (x)
Manager of Department (y)
Manager of Employee (z)

If we exclude transitivity, the Schema will contain:

$$x : E \rightarrow D \text{ and } y : D \rightarrow M$$

(z cannot be explicitely defined as an independent relation).

In the reality the following situation could arise: a Manager decides how to assign his dependents to his Departments, while the attribution of Employees to Managers is decided elsewhere, so that in reality relation z has an existence of its own and must be independently used. In terms of External Schemata, we could have that the company administration views the External Schema Relation ⟨Employee, Manager⟩ (z) and the office of the manager views the External Schema Relation ⟨Employee, Department⟩ (x) to assign his Employees to his Departments.

In this case the "real" situation is represented by three binary relations as it is shown in Fig. 6, and a "Transitive Equivalence Constraint" must be declared, stating that relation z is transitively equivalent to the concatenation of x and y (for existing values of x!).

The process of normalization utilized in the n-ary relational model tries to eliminate the need of declaring (and checking) these types of constraints; however, if the real situation is the one of the above example, normalization (through elimination of z) would lead to the impossibility of inserting instances of z independently of the existence of the corresponding instances of x, y.

We can conclude that it is sometimes impossible representing an information structure with a complete elimination of consistency conditions, because there are consistency conditions that are contained in the very logical nature of the information to be represented. An explicit statement of these consistency conditions seems to be more appropriate than the apparent elimination of such conditions through a modification of the data structure, as it happens with a normalization process.

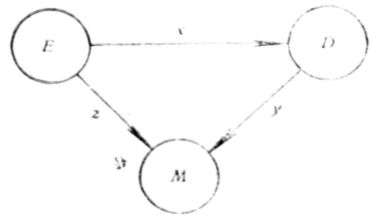

Fig. 6. Transitive equivalence

4. Conclusions. A large number of data models have been recently proposed for use in data base management systems.

There is presently wide interest and agreement on a multilevel DBMS architecture that supports different classes of user's data models.

The discussion on optimal user oriented data models has thus been replaced by the debate on which data model is best suited for the Conceptual Schema (the common logical data description).

Two main requirements for the Conceptual Schema are:

a) being able to support mappings to the external representation levels at the application program interface, and

b) being a reference for the application programmers, the Data Base Administrator and all personnel needing interpretation of the meaning of data.

In this paper we have shown that in the present state of the art these two requirements can lead to different choices for the data model.

Therefore, two different schemata, the Enterprise Schema and the Central Schema, have been discussed for the level of common logical data description.

The functions and the requirements of both the Enterprise and the Central Schema have been discussed.

The evaluation ot the different data models with respect to the two Schemata has been conducted through the analytic discussion of the relevant problem areas.

Emphasis has been given not only to the static (descriptive) properties of the models, but also to their dynamic (operational) characteristics.

REFERENCES

1. J. R. Abrial. Data Semantics. *Data Base Management*, North Holland, 1974.
2. Interim Report ANSI/X3/SPARC Study Group on Data Base Management Systems. ANSI, Feb. 1975.
3. H. Biller, E. J. Neuhold. On the semantics of data bases: the semantics of data models. *Internal Rept.*, University of Stuttgart, 1976.
4. G. Bracchi, A. Fedeli, P. Paolini. A multilevel relational model for data base management systems. *Data Base Management*, North-Holland, 1974.

5. G. Bracchi, P. Paolini, G. Pelagatti. Binary logical associations in data modelling. *Modelling in Data Base Management Systems*. North Holland, 1976.
6. E. F. Codd. A relational model of data for large shared data banks, *Comuns. ACM*, 13, 1970, 377—387.
7. E. F. Codd. Further normalization of the data base relational model. Courant Computer Science Symposia 6, *Data Base Systems*, New York, 1971.
8. M. Adiba, C. Delobel, M. Leonard. A unified approach for modelling data in logical data base design. *Modelling in Data Base Management Systems*. North-Holland, 1976.
9. R. Durcholz, G. Richter. Concepts for data base management systems. *Data Base Management*, North-Holland, 1974.
10. R. Durcholz, G. Richter. Information management concepts for use with DBMS interfaces. *Modelling in Data Base Management Systems*. North-Holland, 1976.
11. E. Falkenberg. Concepts for modelling information. *Modelling in Data Base Management Systems*. North-Holland, 1976.
12. L. Kerschberg, A. Klug, D. Tsichritzis. A taxonomy of date models. *Systems for Large Data Bases*. North-Holland, 1976.
13. P. Hall, J. Owlett, S. Todd. Relations and entities. *Modelling in Data Base Management Systems*. North-Holland, 1976.
14. G. M. Nijssen. A gross architecture for the next generation DBMS. *Modelling in Data Base Management Systems*, North-Holland, 1976.
15. G. M. Nijssen. An exercise in conceptual schema design. *Working paper IFIP WG. 2. 6*, October 1976.
16. P. Pelagatti, P. Paolini G. Bracchi. Mappings in database systems. Istituto di Elettrotecnica ed Elettronica, Politecnico di Milano, Int. Rept. 76—11. November 1976.
17. H. A. Schmid. An analysis of some constructs for conceptual model. Preprints IFIP Working Conference on Modelling in Data Base Management Systems, Nice, January 1977.
18. M. E. Senko. DIAM as an example of the ANSI/SPARC architecture. *Modelling in Data Base Management Systems*. North Holland, 1976.
19. B. Sundgren. Conceptual foundation of the infologicai approach to data bases. *Data Base Management*. North-Holland, 1974.

Instituto di Elettrotenica ed Elettronica
Politecnico di Milano
20133 Milano Piazza L. da Vinci 32

Received 11. 9. 1977