Faculteit der Economische Wetenschappen

The Transformation of Extended Entity Relationship Generalization Hiërarchies into Tables and Meta Tables

E.R.K. Spoor
R.J. Veldwijk
M. Boogaard

Research Memorandum 1991-47

vrije Universiteit *amsterdam*

THE TRANSFORMATION OF EXTENDED ENTITY RELATIONSHIP GENERALIZATION HIERARCHIES INTO TABLES AND META TABLES

by

E R K SPOOR, R J VELDWIJK, and M BOOGAARD

The transformation of generalization hierarchies into relational format cannot be accomplished in a natural fashion. The most popular way to implement a generalization structure is either to compress the hierarchy into one table or to represent it by means of a separate table for each class and subclass in the hierarchy. However, both techniques are accompanied with specific problems that become unacceptable in case of hierarchies that are structurally unstable, have many subclasses and/or use specializing attributes that apply to several of the subclasses.
This article proposes a different technique in which generalization structures are mapped into collections of tables and meta tables. Meta tables contain data about the hierarchy.
The Entity Relationship approach is chosen as a frame of reference.

1. INTRODUCTION

A survey of the literature that proceeded from the Entity Relationship Model (ERM) [Chen76] reveals several proposals and many discussions about the ways in which the ERM abstraction principles could be extended to include the concept of generalization/specialization (see e.g. [Sche80], [Lenz83], [Ceri83], [Teor86], and [Tuch90]). Currently, two constructs, commonly known as *generalization* and *subset hierarchy*, are widely used vehicles for the representation of this desired concept. Both constructs are based on the principle that subclasses of a certain class E are to be represented as subsets of the entity set that represents E. The difference between the constructs originates from the existence of a selection mechanism in case of generalization. This mechanism causes the partition of the class E into subclasses. The subsets that represent these classes are always disjoint.
Subset hierarchy, on the contrary, does not include a selection facility, so partitioning is therefore not enforced and the subsets in the hierarchy may overlap.

Being conceptual tools, interpretations of the ERM including the extension described above (usually known as Extended Entity Relation-

ship (EER) approaches, a convention which we will adapt) have been a focal point in the development of design methodologies. Especially the phase of transformation of EER structures into implementable hierarchical, network or relational structures was, and as far as the latter is concerned still is, a favourite topic on ER meetings (see e.g. [Dave83], [Marc88], and [Loch90]).
A review of the suggestions concerning transformation of EER structures over the years shows that all but the generalization constructs can be transformed into relational format in a rather straightforward manner. It appears that the relational counterparts that have been suggested for the generalization and hierarchy constructs usually imply a total loss of hierarchy and/or usage problems (see e.g. [Ceri83]).

This article introduces an alternative technique to transform generalization and subset hierarchy structures into collections of relational tables. The core idea is that many problems can be solved if we use a combination of tables and meta-tables (i.e. tables that contain data about the hierarchy) to map onto.

The profile of the article is as follows. In Section 2 a frame of reference is introduced, that is, the formal constructs of generalization and subset hierarchy from an Entity Relationship point of view are given. Next, in Section 3 details are presented about the consequences of the usual way to accomplish the transformations concerned. In Section 4 the alternative technique is introduced by means of an example. Finally, in Section 5, some considerations concerning the general applicability of the proposed transformation technique are made.

2. A FRAME OF REFERENCE

As stated in the previous section, the difference between generalization and subset hierarchy is the existence in case of generalization of a selection mechanism to partition a class into subclasses. Conform to the principles of ERM, this mechanism is usually defined to be a function from entities to values, called the *underlying attribute* [Schi79]. For each entity of the superset the value of the underlying attribute indicates to which subset the entity belongs. This value is the name of the concerned subclass. So, in order to define generalization properly, a distinction between names of classes and entity sets has to be made.

The origin of the distinction is the axiomatic difference between the *existence* of an entity and its *essence*. An entity is not only supposed to exist, but apart from this aspect, it is distinct from other entities in some (organizational) context by means of its properties.
This abstraction principle can be given a formal basis by means of the mathematical relation TYPE. TYPE is a subset of the Cartesian product {e|e is an entity} x $\tilde{N}$, in which $\tilde{N}$ is a set of type names. Two arbitrary entities e_1 and e_2 are of the same type if and only if there is a type name N such that (e_1,N) and (e_2,N) are elements of TYPE. In other

words, the relation TYPE adds to each entity at least one type name of a class to which it belongs. Entities that represent a particular class at a certain instance of time have the same type name. More formally, let N be a type name, then with δN the set of entities representing the class with type name N is given.
In the following discussion, the terms class and type will be treated as synonyms.
Attributes typical for ERM, are functions from entity sets onto value sets. An attribute A defined on the entity set δN is a mapping onto a set of (possibly compound) values V, denoted as: A: δN -> V.
The collection of attributes defined on an entity set characterizes this set. To be more specific, the *characterization* of an entity set of type N, upon which the attributes A and B are defined, is given by N(A,B).[1] The characterization of δN gives information about the essence of its entities.
The attribute or collection of attributes that maps an entity set in a one-to-one fashion to the corresponding value set or collection of value sets is called key. Precisely one key is arbitrarily chosen to be the primary key, also called *identifier*. In the characterization of the entity set concerned, this key is indicated by an underscore of all the attributes involved.

The ERM has also facilities to represent relationships between entities. A relationship is considered to be a mathematical relation between entities. Relationship sets, which represent relationship classes, can be symbolized in quite the same manner as entity sets. The precise adjustments to the set of typenames $\tilde{N}$ and the way a relationship set δR can be characterized however, goes beyond the scope of this article (a thorough description can be found in [Spoo89]).

Now, having introduced notions such as type, typename, entity set and characterization, the definition of the generalization construct can be given. Let δN_1 and δN_2 be disjoint entity sets having the attributes A and B in common, i.e. these attributes are defined on an entity set δN_g which contains the sets δN_1 and δN_2.
The characterization of the set δN_g is given by $N_g(A,B,C)$, in which C is the underlying attribute defined by C: δN_g -> $\{N_1,N_2\}$. In other words, the attribute C maps from the root entity set onto a subset of the set of type names $\tilde{N}$. These type names correspond to the names of the subsets of δN_g.
The definition of subset hierarchy is the same, except for the underlying attribute C, which must be omitted in that case.

3. ORDINARY TRANSFORMATIONS

With the relevant EER representation principles as a frame of reference, the rules that are usually applied to transform EER structures into relational structures can be evaluated.

---

[1] Actually, the characterization of an entity set should also contain the value sets to which the attributes map. However, this does not contribute to the discussion (see [Spoo89]).

The transformation of non-generalization constructs may be accomplished quite straightforward. Moreover, if entity existence and entity essence are strictly separated, as in this article, the transformation appears to be quite natural. The most central rule concerns the transformation of a characterization into a relational table. To be more specific, if $N(\underline{A},B)$ is a characterization of some entity set, then it may be interpreted as a table named N with relational attributes A and B, of which A is the primary key. The domains of these attributes are their respective value sets.

Relationship sets can be converted in at least two different ways. One way is transform a relationship set into an additional foreign key to be included in one of the related tables. This can only be accomplished with 1:1 or 1:n relationship sets since they have a functional character. They other way is create a separate table for the relationship set (in case of 1:n or n:m relationship sets). In order to exist as a table of its own right, however, the relationship set must first be 'aggregated' into an entity set (i.e. each relationship is considered to be an entity). The construct that creates the characterization of this new entity set adds the keys of the involved entity sets. Subsequently, this characterization can be transformed into a table (see [Spoo89] for a precise description of the aggregation construction and for a complete set of transformation rules).

Given the separation of entity sets and typenames, the usual transformation of subset hierarchy and generalization, can be recalled as follows. In the case of subset hierarchy, let $N_1(\underline{K})$ and $N_2(\underline{K},A)$ be characterizations of σN_1 and σN_2 respectively (σN_2 is a subset of σN_1). This hierarchy is to be represented by means of the tables $N_1(\underline{K})$ and $N_2(\underline{K},A)$ or by means of one table $N_1'(\underline{K},A)$.
In the case of generalization, let $N_G(\underline{K},C)$, $N_1(\underline{K},A)$, and $N_2(\underline{K},B)$ be characterizations of σN_G, σN_1, and σN_2, such that σN_1 and σN_2 are disjoint subsets of σN_G and C is the underlying attribute. The transformation of this type of generalization reveals either the tables $N_G(\underline{K},C)$, $N_1(\underline{K},A)$, and $N_2(\underline{K},B)$ or just one single table $N_G'(\underline{K},A,B,C)$.
Thus, the possible choices for the relational representation of both types of generalization are to introduce separate tables or to ignore the hierarchy and put all the information into one table.
These solutions, however, cause severe problems in practice, especially if one has to face hierarchies with many subsets in dynamic environments.
In the remainder of this section these problems will be amplified by means of a real world example.

The example concerns a consolidation system for an international corporation. This system contains book values of all the subsidiary companies divided over about 150 different balance sheet items. Furthermore, to enable financial analysis on corporate level, additional attributes register values depending on the kind of balance sheet item. For the example, only five of the 150 items will be taken into consideration, i.e. (1) property, with additional attributes purchase, sales, and depreciation, (2) land, with purchase and sales; (3) remaining assets, without additional attributes; (4) equity capital,

with the attribute interest, and (5) loan capital, also with the attribute interest. Furthermore, historical, legal, and intercompany relations, as well as memo entries, profit and loss accounts etc. will not be considered. The balance sheet items are clustered into about 42 groups depending on the specializing attributes they share. In the example, there are three subsets recognized, i.e. property, land, and capital. Figure 1 shows the example hierarchy.[2]

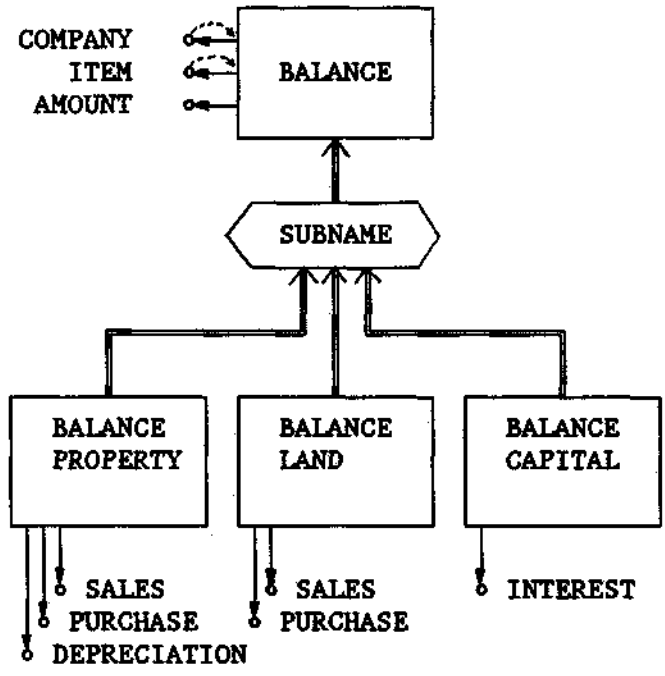


Figure 1

Having described the problem environment, the question is what kind of transformation rule should be applied to convert the EER consolidation scheme into a relational scheme. As already noticed by several authors, both the described rules for the transformation of generalization have certain drawbacks (see e.g. [Smit77]).
According to the first rule the hierarchy should be compressed into one single table (see Figure 2). This solution causes several problems. The most important is, how to interpret the NULL values that appear in the table? A NULL value does not indicate whether it stands for 'not applicable' or 'not known'. This ambiguity also causes uninterpretable query results. Maintaining the meaning of the NULL values is the programmer's responsibility.

---

[2] Notice that the hierarchy is not complete, i.e. the superset also contains the remaining assets. The applied symbols and their meaning are from [Ceri83].

BALANCE (COMPANY,ITEM,AMOUNT,PURCHASE,SALES,DEPRECIATION,INTEREST)

COMPANY	ITEM	AMOUNT	PURCHASE	SALES	DEPRECIATION	INTEREST
THAILND	PROPERTY	200	50	0	10	---
THAILND	LAND	50	---	20	---	---
THAILND	ASSETS	150	---	---	---	---
THAILND	EQUITY	300	---	---	---	10
THAILND	LOAN	100	---	---	---	20
...	...	...	...	...	...	...

Figure 2

The second transformation rule implies the organization of a structure of separate tables, one for each entity type in the hierarchy. Figure 3 shows this transformation for the example case.

```
BALANCE           (COMPANY,ITEM,AMOUNT,SUBNAME)
BALANCE_PROPERTY  (COMPANY,ITEM,PURCHASE,SALES,DEPRECIATION)
BALANCE_LAND      (COMPANY,ITEM,PURCHASE,SALES)
BALANCE_CAPITAL   (COMPANY,ITEM,INTEREST)
```

Figure 3

Although the mentioned NULL value interpretation problem now has been solved, other imperfections remain. First, the database structure does not implicitly enforce a one-to-one correspondence between the names of the subordinate tables and the values of the underlying attribute SUBNAME. This correspondence is necessary to guarantee the distinctness of the subordinate tables and to prevent incorrect registrations like the presence of a tuple in the BALANCE_PROPERTY table having the ITEM-value 'PROPERTY'. Maintaining the one-to-one mapping again is a programmer's responsibility.
Second, it might become difficult to query the tables (remember that the real environment involves as much as 42 subordinate tables). As an example, suppose one wishes to retrieve the following information: 'all data about the loan sheet item of the Thailand subsidiary'. Since the items are divided into groups, depending on the specializing attributes they share, the first necessary step is to obtain the relevant group from the BALANCE table and then to query both the BALANCE table and the selected subordinate table. Thus, the formulation of the second query depends on the outcome of the first. This might be acceptable in an interactive session, but it is of course

impossible to embed it into a program.[3] Figure 4 shows the queries and their results.

```
1: SELECT SUBNAME FROM BALANCE WHERE ITEM = 'LOAN'

Result:
SUBNAME
---------------
BALANCE_CAPITAL

2: SELECT BALANCE.COMPANY, BALANCE.ITEM, AMOUNT,INTEREST
   FROM   BALANCE, BALANCE_CAPITAL
   WHERE  BALANCE.ITEM = 'LOAN'
     AND  BALANCE.ITEM = BALANCE_CAPITAL.ITEM
     AND  BALANCE.COMPANY = BALANCE_CAPITAL.COMPANY

Result:
COMPANY  ITEM  AMOUNT  INTEREST
-------  ----  ------  --------
THAILND  LOAN   100       20
```

Figure 4

A third imperfection of the transformation rule in question has to do with the instability of generalization structures in practice. Data-oriented design methodologies usually support the widespread misunderstanding that data may vary over time while data structures should be stable. In case of evolving organizational structures, however, data structures quite often have to be changed too. The discussed consolidation system, for instance, has to be revised each year in order to include new groups, delete certain groups, or change groups.
Whereas the single table transformation is concerned, these mutations imply modifications in the attribute structure of the target table, which is not always easy to accomplish. In case of the second transformation rule, one has to face the problem of restructuring the collection of target tables, i.e. adding new tables, deleting tables, etc.

Summarizing the previous discussion, it can be argued that the two transformation rules considered are not particularly useful whenever the amount of subordinate entity types is large, the hierarchy is unstable, and/or many of the specializing attributes apply to more than one subordinate entity type.

---

[3] Actually a higher order query language is required in this case, as already has been posed by [Smit77].

4. THE META TABLE TECHNIQUE

The core idea behind the meta table technique to be presented can be expressed as follows. Since the power of relational database management systems is to facilitate manipulation of tuples in flat structures of tables that are meant to be stable, the necessary transformation rule should organize all information concerning generalizations and all unstable structural information into tuples of separate tables. Such tables will be called *meta tables* hereafter.
Using the conceptual scheme facility as a tool again, the intended transformation may also be considered as a two-step process. First, the subject EER structure is to be transformed into an ER structure without generalization constructs. This can be accomplished by means of so called *abstract entity types* [Wagn89]. Abstract entity types are the conceptual counterparts of meta tables.
In the second step the intermediate ER structure can be converted into a single or multiple table structure as described in Section 3.
Figure 5 illustrates the result of the first step applied to the consolidation hierarchy.

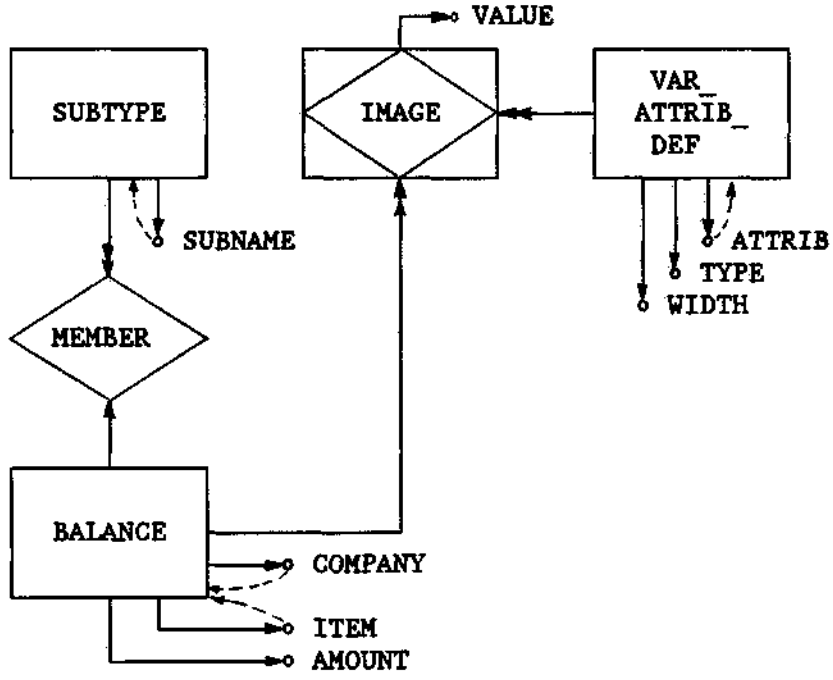


Figure 5

The diagram reveals three abstract entity types, i.e. two entity types (SUBTYPE and VAR_ATTRIB_DEF) and one aggregated relationship type (IMAGE). Each of these abstract entity types is defined in a similar manner as described in Section 2. For instance, SUBTYPE is a typename, σSUBTYPE is a set of abstract entities (i.e. subclasses) and SUBNAME

is a function from σSUBTYPE onto the set of names (PROPERTY,LAND,ASSETS,CAPITAL).
The diagram also shows that the most general attributes (and also the most stable ones in the case situation) have been brought together into the concrete entity type BALANCE, while all specializing attributes are considered to be variable and as such defined by means of the entity type VAR_ATTRIB_DEF and the aggregated relationship type IMAGE. The transformation of this intermediate structure into a collection of relational tables is given in Figure 6.

BALANCE (COMPANY,ITEM,SUBNAME,AMOUNT)
SUBTYPE (SUBNAME)
VAR_ATTRIB_DEF (ATTRIB,TYPE,WIDTH)
IMAGE (COMPANY,ITEM,ATTRIB,VALUE)

BALANCE

COMPANY	ITEM	SUBNAME	AMOUNT
THAILND	PROPERTY	PROPERTY	200
THAILND	LAND	LAND	50
THAILND	ASSETS	ASSETS	150
THAILND	EQUITY	CAPITAL	300
THAILND	LOAN	CAPITAL	100

IMAGE

COMPANY	ITEM	ATTRIB	VALUE
THAILND	PROPERTY	PURCHASE	50
THAILND	PROPERTY	SALES	0
THAILND	PROPERTY	DEPRECIATION	10
THAILND	LAND	PURCHASE	--
THAILND	LAND	SALES	20
THAILND	EQUITY	INTEREST	10
THAILND	LOAN	INTEREST	20

VAR_ATTRIB_DEF

ATTRIB	TYPE	WIDTH
PURCHASE	NUM	3
SALES	NUM	3
DEPRECIATION	NUM	3
INTEREST	NUM	3

SUBTYPE

SUBNAME
PROPERTY
LAND
ASSETS
CAPITAL

Figure 6

Observe that the 1:n relationship set MEMBER, which has no attributes of its own, is represented by the foreign key SUBNAME in the BALANCE table. Furthermore, regard the VALUE attribute of IMAGE. Although this attribute contains only numbers due to the fact that all specializing attributes concern amounts, generally this is not the case. Therefore the underlying domain of VALUE has to be primitive (i.e. character strings).

The question to be answered is whether the ER structure given in Figure 5 and its relational implementation really offer a better solution for the consolidation problem. The three sticking points discussed in the previous section were: (1) retaining the integrity of the structure, (2) querying the structure, and (3) the ability to change the structure in time.

As far as integrity is concerned, consider figure 6 again. The values of the SUBNAME-attribute in the table SUBTYPE, which represent the values of the underlying attribute in the original hierarchy, do not refer any longer to subordinate tables. Instead, BALANCE is the only 'real' table in the structure. All data concerning the hierarchy and the specializing attributes of the subordinate classes in the hierarchy are represented by means of meta tables. The partitioning of the class of balance entities is automatically ensured by the referential integrity constraint BALANCE -> SUBTYPE.
The only additional integrity rule that is necessary, as far as the consolidation example is concerned, must guarantee that each value of the attribute VALUE meets the requirements of its corresponding format given in VAR_ATTRIB_DEF (the correspondence itself is assured by the referential integrity constraint IMAGE -> VAR_ATTRIB_DEF). However, this rule is of a lower order than the underlying attribute constraint.

Now, how about querying? In contrast with the ordinary transformations described in Section 3, the request for 'all data about the loan sheet item of Thailand' (see Section 3) can be reformulated into a simple query, the results of which do not suffer from the mentioned interpretation problems. Figure 7 shows the query and its results.

```
SELECT B.COMPANY, B.ITEM, B.AMOUNT, I.ATTRIB, I.VALUE
FROM   BALANCE B, IMAGE I
WHERE  B.COMPANY = I.COMPANY
  AND  B.ITEM = I.ITEM


Result:
COMPANY  ITEM  AMOUNT  ATTRIB    VALUE
-------  ----  ------  --------  -----
THAILND  LOAN   100    INTEREST    20
```

Figure 7

Of course, since the query uses both table and meta table data, the results of the query have a mixed nature. They have to be read as follows: COMPANY is THAILAND, ITEM is LOAN, AMOUNT is 100, and (from left to right) INTEREST is 20. Whenever the group of the loan item contains more than one attribute, the AMOUNT, COMPANY, and ITEM values are repeated for each additional attribute.

The final aspect to be touched upon in this section is the flexibility of the presented relational implementation, in other words, is the solution able to adapt to changes in the hierarchical structure? The consolidation system in particular appears to be subject to frequent changes as far as its balance hierarchy is concerned. Each year new groups are added, other groups are changed or deleted.
Considering Figure 6 again, it is obvious that new groups can easily be added just by inserting the groupname into SUBTYPE. Furthermore, as long as the two referential integrity constraints are not violated, groups may also be deleted. Finally, groups can be changed, i.e. the attributes of each group may be modified as long as the referential integrity constraint between IMAGE and VAR_ATTRIB_DEF is not violated and all VALUE values meet the requirements given in VAR_ATTRIB_DEF.

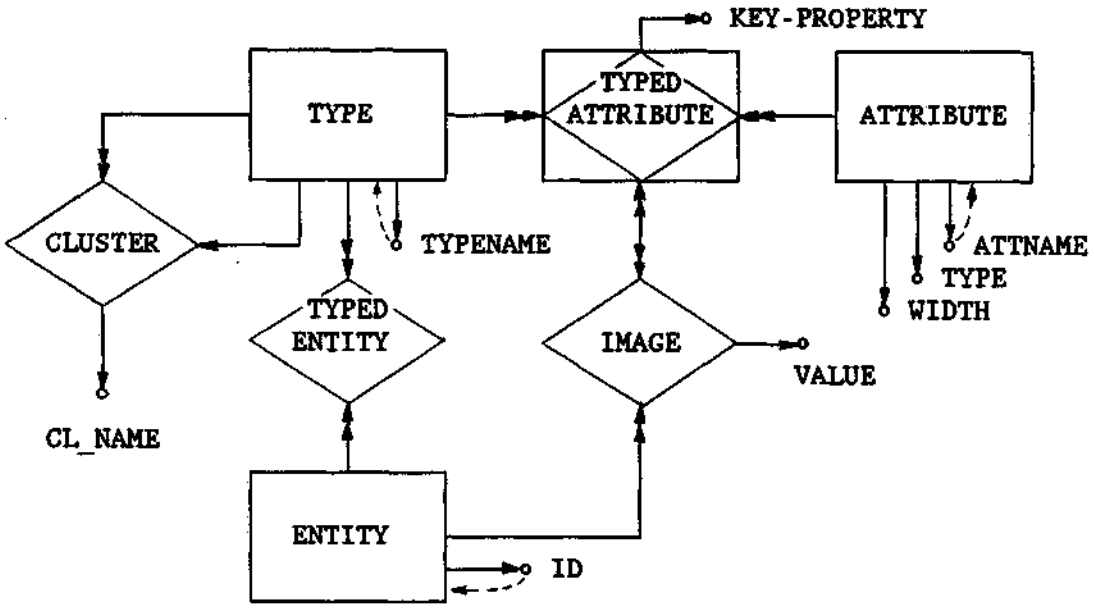
Summarizing the preceding discussion, the meta table technique applied to the consolidation domain has certain advantages over the conversion types described in Section 3. However, questions arise whether this technique does not introduce other difficulties. Does the magnitude of the database play a role? Can multilevel hierarchies also be transformed? These points will be touched upon in the next section.

5. SOME CONSIDERATIONS

The collection of tables presented in Figure 6 constitutes a partial dedicated solution to a specific problem. Although we urge the fact that the problem is quite common in daily practice, it is also specific because it concerns only one particular hierarchy. This hierarchy contains two levels of which the number of subclasses and their attributes evolve in time, while the attributes of the superclass

remain stable. The presented solution is partial dedicated because the stable superclass is represented by a dedicated table and all variable subclass information is put into the general tables IMAGE and VAR_ATTRIB_DEF.

In order to generalize from the specific character of the consolidation domain, the utility of tables versus meta tables as well as the implementation of multi-level hierarchies have to be discussed.
First, tables versus meta tables. Suppose the AMOUNT attribute of the BALANCE type as shown in Figure 5 is to be considered unstable too, e.g. it may be deleted or its name may be changed in time. It is clear that the attribute should then be represented by means of tuples in the IMAGE and VAR_ATTRIB_DEF tables. This can be accomplished quite easily. In fact, all classes and attributes in the system may be represented by means of tuples in meta tables. Figure 8 depicts a more generalized intermediate ER structure together with its relational implementation that does not contain problem specific entity types any more.
Observe the fact that the BALANCE entity type is replaced by the entity type ENTITY that has one single attribute. This attribute is defined to be the identity function ID: δENTITY -> δENTITY, which is necessary because the key attributes COMPANY and ITEM can no longer serve as identification vehicles. The transformation of the abstract entity type ENTITY to a relational format results in a table containing existency symbols with the same function as surrogate values in [Codd79]. Note further, that in figure 8 all the type names in a hierarchy are entities of TYPE and that the hierarchical relationship between a class and its subclasses now is represented by means of the relationship type CLUSTER. Finally, regard the new entity type TYPED_ATTRIBUTE. This entity type is required to register the difference between a key and a non-key attribute.
Thus, we may conclude that the designer performing the transformation from EER to ER scheme has a certain freedom to choose whether to represent a certain class by means of entities of abstract entity types or by means of a separate entity type. His choice therefore will not only be determined by the fact that the class is a subclass in a hierarchy or by the instability of the class. He also may take performance considerations into account. Performance plays a role whenever the magnitude of the class as far as its attributes and tuples are concerned, is large. The cardinality of, for instance, the IMAGE table may grow very fast. Such grow eventually leads to performance problems that might take the edge of the solution.
Another way to avoid an unacceptable performance is to combine the CLUSTER, TYPE, TYPED_ATTRIBUTE, and ATTRIBUTE meta tables with one of the usual transformation results given in Section 3, for instance with the BALANCE table in Figure 2. Because the meta tables contain all the hierarchy information necessary, integrity can be maintained and queries like the one in the previous section may also be suitably formulated. As far as structural changes are concerned, all necessary DML statements on the meta tables and corresponding DDL statements on the BALANCE table can be performed by means of automated procedures. This reduces the effort necessary for the maintenance of the system (see [Veld91] for further details of this technique).



Relational implementation:

ENTITY	(ID)
ATTRIBUTE	(ATTNAME,TYPE,WIDTH)
TYPE	(TYPENAME,CL_NAME)
CLUSTER	(CL_NAME,TYPENAME)
TYPED-ENTITY	(ID,TYPENAME)
TYPED-ATTRIBUTE	(TYPENAME,ATTNAME,KEY-PROPERTY)
IMAGE	(ID,TYPENAME,ATTNAME,VALUE)

Figure 8

The second aspect that needs to be elaborated is the representation of multi-level hierarchies. As Figure 8 shows, the representation of such hierarchies, which may include a mixture of generalization and subset hierarchy constructs, is accomplished by means of the relationship type CLUSTER. The relational counterpart of this relationship type and entity type involved has been defined in such a way that the 1:n constraint is enforced by means of the referential integrity constraints TYPE -> CLUSTER and CLUSTER -> TYPE.
The hierarchy can be made explicit by means of an outer-join between TYPE and CLUSTER (see figure 9)[4].

[4] The symbol (+) represents the outer-join specific for the database management system ORACLE in which the example has been implemented.

```
SELECT TYPE.TYPENAME, CLUSTER.TYPENAME
FROM TYPE, CLUSTER
WHERE TYPE.CL_NAME = CLUSTER.CL_NAME (+)
```

Figure 9

6 CONCLUSIONS

We conclude that the transformation of a hierarchical structure into a collection of tables and meta tables has advantages over the usual way it is accomplished. Instead of simply ignoring the hierarchy by placing all data into one single table, or dedicating a table to each class and subclass, a representation structure is proposed that includes problem domain independent tables (called meta tables) and may (but not necessarily must) include domain dependent tables. The meta tables contain information about the hierarchy and the involved classes that is otherwise lost if the transformation is done in the traditional way.

The combination of meta tables and tables also provides the database designer with a certain amount of freedom to choose whether to represent a certain class (not only information about its internal structure but also about its members) as tuples in the meta tables or as a dedicated table. This choice depends on requirements of flexibility and performance. Representation of classes by means of tuples in the meta tables implies a high degree of flexibility as far as structural changes are concerned. However, this is only sensible to do with classes of relatively low magnitude, i.e. having a limited amount of members and attributes; otherwise performance problems will be inevitable. If flexibility is not a hard criterion, but performance is, then it is preferable to represent the class by a dedicated table.
If both flexibility and performance play a role, the designer might choose a mean between the two extremes. This compromise consists of two parts, i.e. (1) the result of one of the traditional transformations, and (2) the part of the proposed scheme that only contains information about the hierarchy and the internal structure of the involved classes, but not about the individual members of these classes. Such a combination of techniques offers an interesting increase of flexibility because all structural changes, which imply rewriting DDL and DML statements and recompiling of the involved programs, can be done by automated procedures. In this case performance is no problem.

REFERENCES

[Ceri83] **Ceri S ed.** *Methodology and Tools for Data Base Design* North Holland (1983)

[Chen76] **Chen P P** 'The Entity-Relationship Model - Toward a Unified View of Data' *ACM Transactions On Database Systems* 3 (1976)

[Codd79] **Codd E F** 'Extending the Database Relational Model to Capture More Meaning' *ACM Transactions on Database Systems* Vol 4 No 3 (1979)

[Davi83] **Davis C G, Jajoda S, Ng P A, and Yeh R T eds.** *Entity Relationship Approach to Software Engineering* Procs. Third Int. Conf. E-R, North Holland (1983)

[Lenz83] **Lenzerini M and Santucci G** 'Cardinality Constraints in the Entity-Relationship Model' in: [Davi83]

[Loch90] **Lochovsky F H ed.** *Entity-Relationship Approach to Database Design and Querying* Procs. Eighth Int. Conf. On E-R, North Holland (1990)

[Marc88] **March S T ed.** *Entity Relationship Approach* Procs. Sixth Int. Conf. On E-R, North Holland (1988)

[Sche80] **Scheuermann P, Schiffner G, and Weber H** 'Abstraction Capabilities and Invariant Properties Modelling within the Entity-Relationship Approach' in: [Chen80a]

[Schi79] **Schiffner G and Scheuermann P** 'Multiple Views and Abstractions with an Extended-Entity-Relationship Model' *Computer Languages* Vol 4 (1979)

[Smit77] **Smith J M and Smith D C P** 'Database Abstractions: Aggregation and Generalization' *ACM Transactions on Database Systems* Vol 2 No 2 (1977)

[Spoo89] **Spoor E R K** *Expert System Design in an Entity Relationship Environment* Ph.D. dissertation, Free University Press, Amsterdam (1989)

[Teor86] **Teorey T J, Yang D and Fry J P** 'A Logical Design Methodology for Realtional Databases Using the Extended Entity-Relationship Model' *ACM Computing Survey* Vol 18 No 2 (1986)

[Tuch90] **Tucherman L, Casanova M A, Gualandi P M, and Braga A P** ' A Proposal for Formalizing and Extending the Generalization and Subset Abstractions in the Entity-Relationship Model' in: [Loch90]

[Veld91] **Veldwijk R J, Boogaard M, Dijk M V van, and Spoor E R K** 'EDSO's, Implosion and Explosion: Concepts to Automate a Part of Application Maintenance' *Information and Software Technology* (june 1991)

[Wagn89] **Wagner C F** 'Implementing Abstraction Hierarchies' *Entity-Relationship Approach: A Bridge to the User* **Batin C ed.** North Holland (1989)