VU Research Portal

Extending the relational model version 2 to support generalization hierarchies

Veldwijk, R.J.; Boogaard, M.; Spoor, E.

1991

Extending the Relational Model Version 2 to Support Generalization Hierarchies

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

Research Memorandum 1991-78
December 1991

vrije Universiteit *amsterdam*

Extending the Relational Model Version 2 to Support Generalization Hierarchies

by

R J VELDWIJK, M BOOGAARD and E R K SPOOR

Support for generalization hierarchies is widely recognized as a desirable feature of data models. While such hierarchies are indeed supported by data models such as NIAM and object oriented models, the relational model is still deficient in this respect. Proposed enhancements suggested by Smith and Smith in their SHM model and by Codd in his RM/T model have not been included in Codd's new version of the relational model (RM/V2). The present paper proposes an extension to RM/V2 needed to support generalization hierarchies and thus solve database design problems induced by generalization hierarchies. The proposed extension to RM/V2 is first introduced as an extension to a formal model of RM/V2 expressed in RM/V2 itself, i.e. an RM/V2-catalog schema. The paper shows that a small extension to relational language, that uses the extended catalog schema, suffices to solve the acknowledged problems with respect to inquiries on a generalization hierarchy. Next, the paper shows that even in an environment as powerful as SHM or extended RM/V2 there exist inquiries that cannot be implemented easily or at all. It identifies the distinction between data and meta data (or catalog instance data) as the root of this problem. As a powerful solution it then proposes the imploded relational model, in which the distinction between data and meta data has been completely removed, as a conceptual vehicle with which a fundamental solution can be offered.

Key words and phrases:
generalization, relational model, catalog, flexibility, data independence, structural redundancy, imploded relational model.

1. INTRODUCTION

An important aim of data models is to provide application designers with constructs that enable them to represent relevant aspects of reality in a natural manner. Inadequate support for a construct to be modelled reduces the understandability of application designs and increases realization and maintenance effort, because the modelling must then be done in an indirect manner.

The construct of the generalization hierarchy is often encountered when modelling reality. Indeed, generalization is considered a basic abstraction mechanism, together with classification, aggregation, and association [BROD84]. Any data model should thus be able to represent at least these abstractions. Unfortunately, the relational model does not, in contrast with other data models such as NIAM [NIJS89] and object oriented data models [BHAL91]. Extending the relational model

with generalization support has been advocated by Smith and Smith, and by Codd. Smith and Smith have proposed an extension to the relational model[1], known as the semantic hierarchy model (SHM) in [SMIT77]. Codd [CODD79] has introduced generalization support as part of his RM/T-version of the relational model, but has now come to the conclusion that the world of DBMS vendors and consumers is not yet ready for RM/T, which deviates considerably from the existing version of the relational model. Consequently, the second version of the relational model (RM/V2), as introduced in [CODD90], offers no solution to modelling problems caused by generalization hierarchies. Support for generalization is deferred to a future version of the relational model [CODD90, p.480]. The authors feel that implementing the frequently occurring generalization hierarchies in today's RDBMS environments is so cumbersome that an extension to the relational model and to RDBMS-products would be welcomed, especially if it would prove to be a simple one.

In a previous paper the authors introduced a modelling approach that enables the designer to deal with generalization hierarchies in a very flexible manner [BOOG91a]. However, this approach was presented as a modelling solution *within* a relational framework, not as an extension to the relational model itself. In a more recent paper [VELD91c] the authors proposed a catalog standard for DBMSs that conform to RM/V2.

The present paper integrates the catalog standard with aspects of the data structuring approach advocated in [BOOG91a], thereby making it possible to represent generalization hierarchies of the SHM type. It appears that a simple extension to the relational language is sufficient to solve the problems identified in [SMIT77]. Next, the shows that there still remain modelling and programming problems frequently encountered with but not limited to generalization hierarchies. It argues that a more radical approach, based on the elimination of the distinction between data and meta data, is necessary to solve these. Examples are given of what this approach might look like and what it could accomplish.

The paper is divided into six sections. Section 2 discusses the most abundant class of generalization structures and its problematic implementation in current relational environments. It also briefly revisits SHM as introduced by Smith and Smith. Section 3 introduces the relevant aspects of the proposed RM/V2 catalog standard extended with support for generalization hierarchies. Section 4 proposes an extension to relational language that exploits the enriched RM/V2 catalog and discusses the effects on database design, data manipulation, and data independence. Section 5 identifies classes of queries that remain hard to implement, suggests the distinction between instance data and meta data (or catalog instance data) as the origin of these problems, and proposes a powerful solution to implement them. Finally, section 6 contains a number of remarks with respect to the application and the limitations of the concepts and

---

[1] Although Smith and Smith only speak of their work as a "*structuring discipline*", they propose new relational integrity rules and suggest extensions to relational operators.

constructs introduced in this paper.

2. GENERALIZATION HIERARCHIES AND THEIR IMPLEMENTATION

Generalization can be defined as an abstraction in which a set of similar objects is regarded as a generic object, ignoring differences between individual objects [SMIT77]. Although different types of generalization can be distinguished [BRAC83] [CODD79], the most commonly occurring type is the generalization hierarchy, in which an object simultaneously belongs to a generic object class and to one or more subclasses of that class. Subclasses can be divided into several sub-subclasses and so on. A classic example can be found in [SMIT77], a simplified version of which is displayed in figure 1.

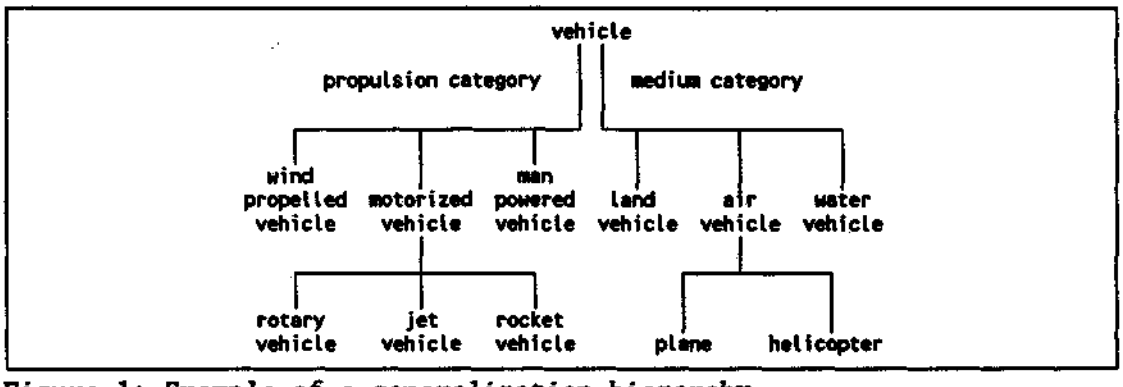


Figure 1: Example of a generalization hierarchy

Every object that is a vehicle does also belong to one or more subclasses of the generic object 'vehicle'. Furthermore, the generalization hierarchy is subject to *clustering*, which in this case means that an individual vehicle must belong to two subclasses, one for the propulsion category and one for the medium category. An individual bicycle thus belongs to the subclasses 'man powered vehicle' (propulsion) and 'land vehicle' (medium). An object that belongs to a subclass also belongs to every superior class of that subclass. Thus, an individual helicopter is also a 'vehicle'.

The rationale for distinguishing between classes and subclasses can be explained by the fact that certain information requirements are relevant for vehicles in general, while others are only relevant for specific subclasses. Each instance of a subclass inherits the properties of its superior classes but also has attributes of its own.

If all vehicle objects were represented in one relation VEHICLE containing all attributes pertinent to any vehicle, the result would be a large number of 'not-applicable' nulls. In RM/V2's four-valued logic terminology these nulls would be denoted by so called I-marks. Whether implemented by unspecified RM/V1 null values or by RM/V2 I-marks, only a subset of the tuples in the relation (or R-table in RM/V2 terminology) can accept such a null/I-mark. Every null/I-mark thus has to be accompanied by an appropriate user-defined database constraint. Moreover, inquiries like "Retrieve all attributes of

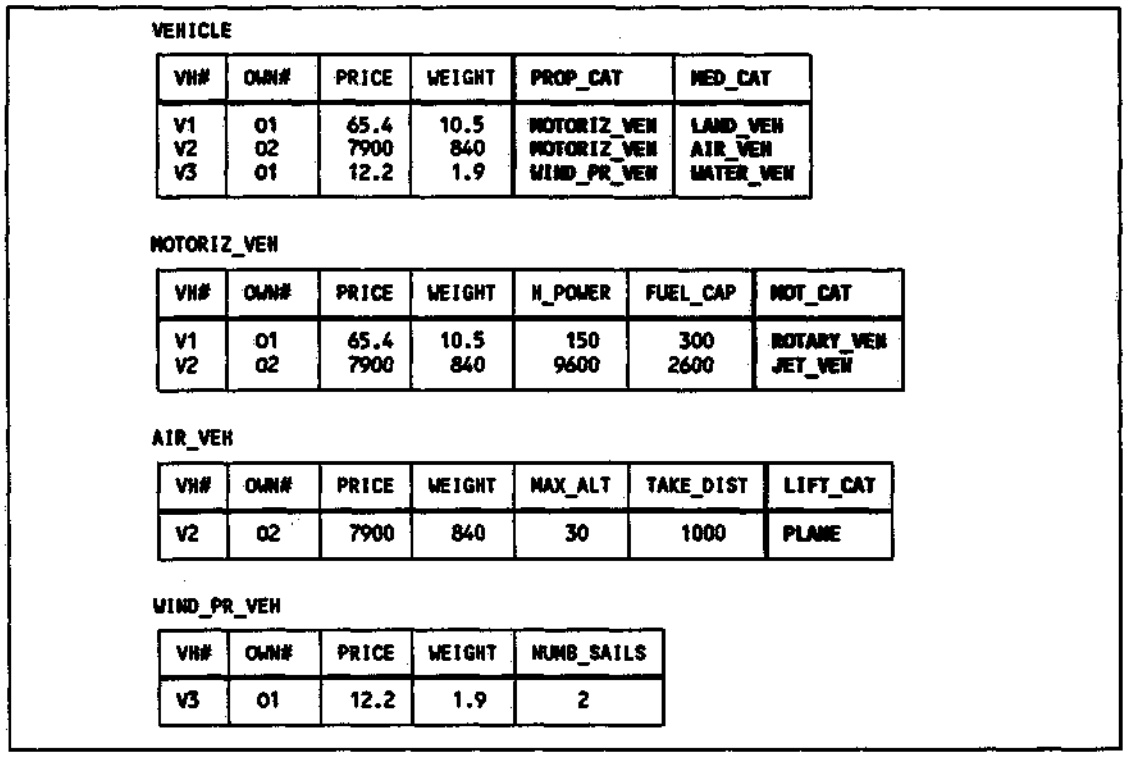
vehicle V1" cannot easily be interpreted, unlike similar queries on relations where generalization problems do not apply.

Another extreme approach would be to regard every terminal subclass as a class in itself and disregard the generalization structure. This approach of disregarding the generalization hierarchy leads to semantically poor and complex database structures and gives rise to complex queries. Suppose for example that every vehicle has an owner. If the owners were represented by a separate relation OWNER, the number of foreign key constraints would be large and the inquiry "Retrieve all vehicle numbers of vehicles owned by owner O1" would result in a lengthy query statement expressing what relations contain vehicle data. For this inquiry the generalized (single-relation) modelling solution would be far superior.

The problem with generalization hierarchies manifests itself by the existence of several modelling alternatives, each of which captures the semantics of generalization hierarchy in an indirect manner by a number of database constraints. McGee [MCGE76] regards the occurrence of indirect modelling solutions and multiple equivalent modelling solutions for a real-world construct as indications of a data model's shortcomings. Partly based on this observation, [VELD91b] argues that minimization of the number of database constraints is a good yardstick against which the quality of a database design can be measured. The number of database constraints relevant to a particular database design depends on both the competence of the database designer and the expressive power of the data model used. If the relational model can be extended with a feature that permits direct modelling of generalization hierarchies, its usefulness can be assessed by considering the effect on the number of ad hoc database constraints in cases where generalization hierarchies occur.

Smith and Smith [SMIT77] have suggested SHM as an extension to the relational model which indeed reduces the number of database constraints induced by generalization hierarchies. Their approach requires the introduction of so called *image domains*. An attribute that belongs to an image domain designates a subclass to which an instance of a superior class belongs. Figure 2 (overleaf) displays relational data structures that represent part of the vehicle generalization hierarchy. Image domain data are displayed in bold face.

All relations in the vehicle hierarchy have the same primary key (VH#). The subclasses to which a tuple with the same primary key value must belong are designated by the values of the image domain attributes. The set of permitted values of these attributes consists of the set of names of the immediate subrelations. If image domains are supported, the DBMS can enforce most relevant constraints. Neither the problem of large numbers of I-marks nor the problem of complex database structures occurs. Moreover, the DBMS can control the redundancy that occurs in subrelations caused by the fact that these relations also contain the attributes of their superior relations. The inquiries discussed above would pose no problems. Both could be expressed by simple queries on the VEHICLE relation, provided the specific (sub)classes, in which the vehicle data can be found, are known.



VEHICLE

VH#	OWN#	PRICE	WEIGHT	PROP_CAT	MED_CAT
V1	01	65.4	10.5	MOTORIZ_VEH	LAND_VEH
V2	02	7900	840	MOTORIZ_VEH	AIR_VEH
V3	01	12.2	1.9	WIND_PR_VEH	WATER_VEH

MOTORIZ_VEH

VH#	OWN#	PRICE	WEIGHT	H_POWER	FUEL_CAP	MOT_CAT
V1	01	65.4	10.5	150	300	ROTARY_VEH
V2	02	7900	840	9600	2600	JET_VEH

AIR_VEH

VH#	OWN#	PRICE	WEIGHT	MAX_ALT	TAKE_DIST	LIFT_CAT
V2	02	7900	840	30	1000	PLANE

WIND_PR_VEH

VH#	OWN#	PRICE	WEIGHT	NUMB_SAILS
V3	01	12.2	1.9	2

Figure 2: Some relations in the vehicle generalization hierarchy

The introduction of a new construct, the image domain, is characteristic for the SHM approach. Image domains reflect the structure of the generalization hierarchy on the instance level, but contain in fact meta data, namely relation names. Image domains are primarily used because it is often known that a generic object (say vehicle V1) exists, but its *precise* classification is not known. If one is interested in the values of all relevant attributes of vehicle V1, a second order query language is needed because the value of an image domain (i.e. a relation name) must be used in the structural part of a query (i.e. the FROM clause in SQL-terms). As we shall see in section 5, image domains are a manifestation of a more general problem that occurs whenever queries request both instance data and meta data.

The problem is basically caused by the assumption in the relational model that the relation (class) to which a tuple (object) belongs is always known beforehand. Wherever generalization hierarchies occur, this assumption is often not valid. A point can also be made for the inverse proposition: whenever the class to which an object belongs is not known, the object participates in some sort of generalization structure. Because generalization hierarchies are commonly encountered, it is worthwhile to extend the relational model. The real issue is to minimize the extensions needed in data structures, integrity rules and operators to achieve this objective. If we compare the SHM solution to the solution offered by Codd in his RM/T paper [CODD79] it appears that the RM/T solution for generalization hierarchies is found at the meta data or catalog level, instead of the instance level. The present paper demonstrates that this latter

approach is superior. To attain this objective, a catalog standard for RM/V2 is needed. The next section introduces such a catalog schema together with an appropriate extension for generalization hierarchies.

3. EXTENDING THE RM/V2 CATALOG SCHEMA

Like the original relational model, its RM/V2 successor is a fairly simple formal model. The fact that Codd needs a rather voluminous book [CODD90] to describe RM/V2 is induced by the fact that (1) RM/V2 is as much a DBMS description as a data model and (2) the specifications of RM/V2 are cast in a verbal form, rather than in a formal one. Formal specifications only exist for relational languages like relational algebra, relational calculus, and of course SQL. The authors have therefore developed a formal specification of the static or non-manipulative aspects of RM/V2 [VELD91c]. The choice for RM/V2 itself as the modelling tool is obvious if one regards the relational model as an expressive data model [VELD91b]. A fortunate side effect is that the resulting description can be seen as a candidate catalog standard for future RM/V2 DBMS products.

Basic to both RM/V2 and the original relational model (RM/V1) are concepts like 'relation', 'domain', 'attribute', 'key', 'entity integrity' and 'referential integrity'. It appears that these concepts and the way in which they are related to each other can be captured elegantly in a relational database schema. It also appears that an extension to support generalization hierarchies requires only a small change to this schema, depicted in figure 3a. The boxes represent relations (R-tables in RM/V2 terms), the arrows represent foreign key to primary key references between these R_tables. In the R_table descriptions the primary key attributes (columns in RM/V2) have been underlined. The additional R_Table and columns that have been added to the schema introduced in [VELD91c] are displayed in bold face.

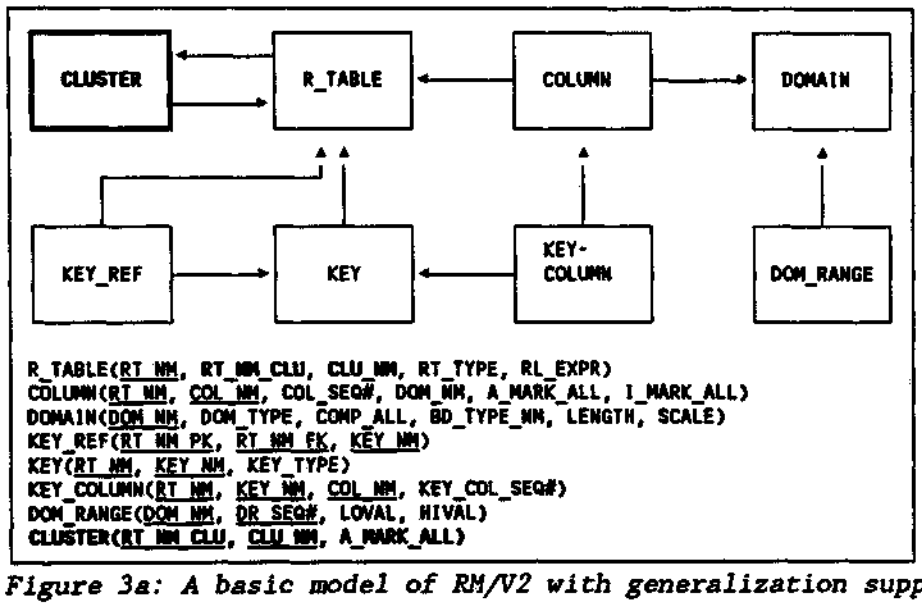


Figure 3a: A basic model of RM/V2 with generalization support added

Disregarding the extensions to support generalization hierarchies for the moment, the schema can be interpreted as follows.

An R_TABLE in the relational model must have a unique name (RT_NM). An R_Table is either a base relation or a view (RT_TYPE). Views are defined in terms of other views and/or base relations by means of a statement in a relational language (RL_EXPR).

A COLUMN (or attribute) is identified by its name together with the name of the relation to which it belongs (RT_NM, COL_NM). Columns are assigned a sequence number for default presentation purposes. Every column belongs to one domain (DOM_NM). Non-primary key columns can be allowed to be 'missing-but-applicable' (A_MARK_ALL), 'missing-and-inapplicable' (I_MARK_ALL) or both.

A DOMAIN must have a unique name (DOM_NM). Depending on the question whether a primary key column draws its values from the domain, it is a primary or a secondary domain respectively (DOM_TYPE). An indicator (COMP_ALL) designates whether comparison of domain values is meaningful. Domains are extended data types based on DBMS dependent basic data types (BD_TYPE_NM) like 'character', 'number' and 'date'. Depending on the datatype, the columns LENGTH and SCALE can be applicable.

The relation DOM_RANGE permits the specification of multiple ranges of values per domain by means of the columns LOVAL and HIVAL. If no range of values is specified, the permitted values are limited only by the domains basic data type and format limitations.

A KEY is formed by one or more columns of one relation that identify a tuple in some relation. Primary and alternate keys identify tuples in the R-table to which these keys belong themselves. Foreign keys identify tuples in one or more R-tables designated by KEY_REF.

KEY_COLUMN contains the columns constituting a key. A key column is identified by the combination of the identifiers of the R-tables KEY and COLUMN. The column KEY_COL_SEQ# provides the mechanism to couple a foreign key column to a primary key column. Composite primary/foreign key combinations must have matching sequence numbers for their columns. A foreign key references at least one R-table. Every such reference is expressed by a KEY_REF tuple.

It appears that the core aspects of RM/V2 can be expressed quite concisely and, in the our opinion, fairly elegantly by means of RM/V2 itself. Nevertheless, there remain a number of constraints that are not enforced by the data structure of figure 3a. These constraints have to be expressed in an ad hoc manner by means of a relational language and stored in the catalog (see [VELD91c] section 7). Figure 3b (overleaf) lists the most important constraints pertinent to the basic model of figure 3a. A full discussion can be found in [VELD91c]. The constraint identification (BM##) has been copied from that article.

BM03- Every tuple in R_TABLE for which RT_TYPE = 'BASE' or 'CHECKOUT' is referenced by a KEY tuple having KEY_TYPE = 'PRIMARY'.

BM04- The subset of KEY tuples for which KEY_TYPE = 'PRIMARY' does not contain duplicate values for RT_NM.

BM05- Every tuple in KEY having KEY_TYPE = 'FOREIGN' is referenced by at least one KEY_REF tuple, while other tuples in KEY are not referenced by KEY_REF tuples.

BM08- KEY tuples with KEY_TYPE = 'PRIMARY', that are referenced via KEY_REF by a KEY tuple with KEY_TYPE = 'FOREIGN', are referenced by pairs of KEY_COLUMN tuples with corresponding values for KEY_COL_SEQ#, referencing COLUMN tuples with the same value for DOM_NM.

BM10- The column RL_EXPR in R_TABLE contains an I-mark if and only if RT_TYPE = 'BASE'.

BM13- No tuple in COLUMN that is referenced by a tuple in KEY_COLUMN that references a tuple in KEY having KEY_TYPE = 'PRIMARY' has a 'YES' value for either A_MARK_ALL or I_MARK_ALL.

Figure 3b: User-defined constraints on the basic model of RM/V2 without generalization support

Constraints BM03 and BM04 assert that every base R-table must have exactly one primary key. Constraint BM05 asserts that a reference from one R-table to another applies if and only if the key is a foreign key. Constraint BM08 asserts that foreign keys and primary keys are realized by matching pairs of columns belonging to similar domains. Constraint BM10 asserts that base tables have no view definition. It is an example of simple generalization hierarchy at the definitional level of RM/V2 itself.

The extensions needed to support generalization hierarchies appear to be very limited. The only significant addition is the R-table CLUSTER containing data about clusters like 'propulsion category' and 'medium category'. Every R_TABLE tuple that contains information about a subcategory must reference a CLUSTER tuple that in turn references an R_TABLE tuple designating the direct superior category. Generic objects (like 'vehicle') or objects that do not take part in generalization hierarchies (like 'owner') are represented by R_TABLE tuples that do not reference a CLUSTER tuple. A cluster is identified by the superior R_table to which it belongs (RT_NM_CLU) and by its name (CLU_NM). The column CLU_NM also designates the name of the image domain that is added to the superior table as a pseudo column[2], while A_MARK_ALL indicates whether an image domain value must be supplied for every tuple in the superior R_Table. The values the image domain column can accept are the names of the R_TABLE tuples referencing the CLUSTER tuple.

[2] Because CLUSTER tuples display a close correspondence to COLUMN tuples, image *column* would be a more appropriate term.

The introduction of support for generalization hierarchies leads to some new constraints that are listed in figure 3c below (GH stands for generalization hierarchy).

GH01- The columns RT_NM_CLU and CLU_NM in R_TABLE are both either unmarked or A-marked.

GH02- R_TABLE tuples for which RT_NM_CLU is not A-marked have the value 'BASE' for their column RT_TYPE.

GH03- No R_TABLE tuple may reference itself directly or indirectly via the R-table CLUSTER.

GH04- R_TABLE tuples referencing a CLUSTER tuple are not referenced via COLUMN by KEY_COLUMN tuples that reference a KEY tuple for which KEY_TYPE = 'PRIMARY'.

GH05- The set of CLU_NM values for CLUSTER tuples referencing an R_TABLE tuple must be disjunct from the set of COL_NM values for the COLUMN tuples referencing it.

GH06- No COLUMN tuple can have a value for COL_NM that also occurs in another COLUMN tuple that references a superior (via CLUSTER) R_TABLE tuple.

GH07- No COLUMN tuple can have a value for COL_NM that also occurs in another COLUMN tuple that belongs to a different cluster in the same generalization hierarchy.

GH08- Any two COLUMN tuples with equal values for COL_NM that reference R_TABLE tuples that are part of a generalization hierarchy have also equal values for their DOM_NM columns.

Figure 3c: User-defined constraints on the basic model of RM/V2 to enforce generalization support.

Because Codd's definition of referential integrity allows foreign keys to be partially marked, constraint GH01 forbids meaningless references to the CLUSTER R-table. Constraint GH02 limits generalization hierarchies to base R-tables. Constraint GH03 asserts that the generalization hierarchy contains no cycles. Constraint GH04 enforces the rule that primary keys are inherited from the generic object by all lower class objects. Constraint GH05 expresses the pseudo column nature of clusters. Constraint GH06 enforces the rule that columns are inherited from superior R-tables. Finally, constraints GH07 and GH08 specify that columns in different subclasses having the same names are in fact the same attributes, which implies that no individual object designated by a primary key value can have two or more columns with the same name. This restriction may seem no more than a reasonable modelling advice at this point. However, in the next section it will prove to be quite important.

Due to the introduction of catalog support for generalization hierarchies some existing catalog constraints have to be modified. Figure 3d lists the new versions with the modifications printed in bold face.

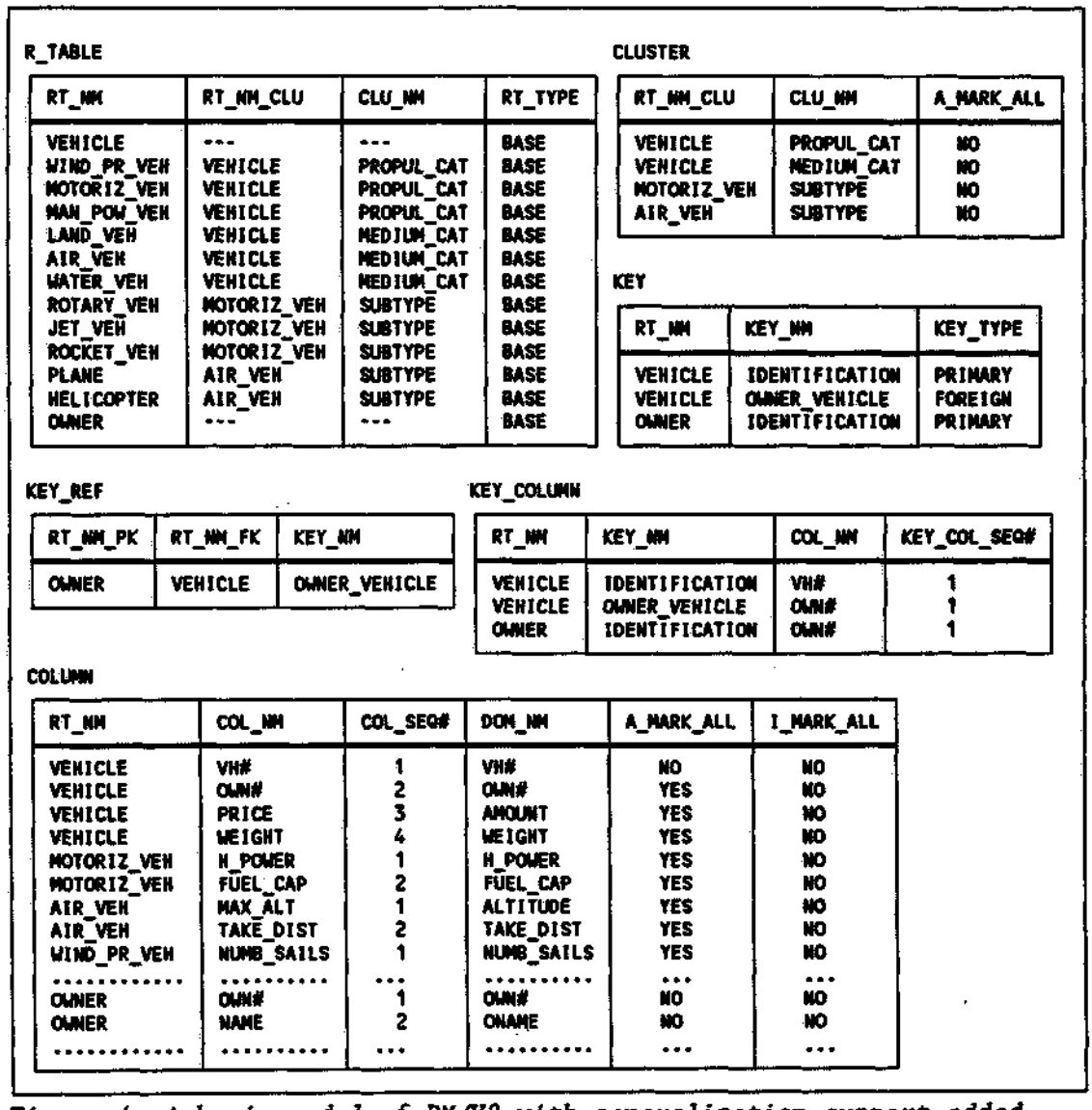
BM03- Every tuple in R_TABLE for which RT_TYPE = 'BASE' or 'CHECKOUT' **and for which RT_NM_CLU is not A-marked** is referenced by a KEY tuple having KEY_TYPE = 'PRIMARY'.

BM08- KEY tuples with KEY_TYPE = 'PRIMARY', that are referenced via KEY_REF by a KEY tuple with KEY_TYPE = 'FOREIGN', are referenced by pairs of KEY_COLUMN tuples with corresponding values for KEY_COL_SEQ#, referencing COLUMN tuples with the same value for DOM_NM. **For R_TABLE tuples for which RT_NM_CLU is not A-marked, this rule is applied the primary KEY_TYPE tuple that references the R_TABLE tuple recursively referenced via CLUSTER and R_TABLE.**

Figure 3d: Altered user-defined constraints on the basic model of RM/V2 due to the introduction of generalization support.

Constraints GH03, GH06, GH07, GH08, and BM08 cannot be expressed in a relational language. This situation will cease to exist if these languages are extended with a *recursive join* operator as suggested by Codd in [CODD90].

At this point we can express the vehicle example by means of tuples in the extended RM/V2 catalog, as is done in figure 4 (overleaf) for the catalog relations discussed above. It has been assumed that there exists an OWNER R-table containing data about the vehicles' owners. Because they add little insight the R-tables DOMAIN and DOM_RANGE have been omitted. With respect to the R-table COLUMN only those columns depicted in figure 2 have been displayed. Due to the introduction of generalization support, no column in the vehicle example is allowed to contain I-marks. Note that the image domain columns of figure 2 do not occur as COLUMN tuples and that the generalization hierarchy is fully determined at the catalog level. Instance data and meta data are not mixed and there is no need for an image domain construct anymore.



R_TABLE

RT_NM	RT_NM_CLU	CLU_NM	RT_TYPE
VEHICLE	---	---	BASE
WIND_PR_VEH	VEHICLE	PROPUL_CAT	BASE
MOTORIZ_VEH	VEHICLE	PROPUL_CAT	BASE
MAN_POW_VEH	VEHICLE	PROPUL_CAT	BASE
LAND_VEH	VEHICLE	MEDIUM_CAT	BASE
AIR_VEH	VEHICLE	MEDIUM_CAT	BASE
WATER_VEH	VEHICLE	MEDIUM_CAT	BASE
ROTARY_VEH	MOTORIZ_VEH	SUBTYPE	BASE
JET_VEH	MOTORIZ_VEH	SUBTYPE	BASE
ROCKET_VEH	MOTORIZ_VEH	SUBTYPE	BASE
PLANE	AIR_VEH	SUBTYPE	BASE
HELICOPTER	AIR_VEH	SUBTYPE	BASE
OWNER	---	---	BASE

CLUSTER

RT_NM_CLU	CLU_NM	A_MARK_ALL
VEHICLE	PROPUL_CAT	NO
VEHICLE	MEDIUM_CAT	NO
MOTORIZ_VEH	SUBTYPE	NO
AIR_VEH	SUBTYPE	NO

KEY

RT_NM	KEY_NM	KEY_TYPE
VEHICLE	IDENTIFICATION	PRIMARY
VEHICLE	OWNER_VEHICLE	FOREIGN
OWNER	IDENTIFICATION	PRIMARY

KEY_REF

RT_NM_PK	RT_NM_FK	KEY_NM
OWNER	VEHICLE	OWNER_VEHICLE

KEY_COLUMN

RT_NM	KEY_NM	COL_NM	KEY_COL_SEQ#
VEHICLE	IDENTIFICATION	VH#	1
VEHICLE	OWNER_VEHICLE	OWN#	1
OWNER	IDENTIFICATION	OWN#	1

COLUMN

RT_NM	COL_NM	COL_SEQ#	DOM_NM	A_MARK_ALL	I_MARK_ALL
VEHICLE	VH#	1	VH#	NO	NO
VEHICLE	OWN#	2	OWN#	YES	NO
VEHICLE	PRICE	3	AMOUNT	YES	NO
VEHICLE	WEIGHT	4	WEIGHT	YES	NO
MOTORIZ_VEH	H_POWER	1	H_POWER	YES	NO
MOTORIZ_VEH	FUEL_CAP	2	FUEL_CAP	YES	NO
AIR_VEH	MAX_ALT	1	ALTITUDE	YES	NO
AIR_VEH	TAKE_DIST	2	TAKE_DIST	YES	NO
WIND_PR_VEH	NUMB_SAILS	1	NUMB_SAILS	YES	NO
.....	.....	...	.....	...	...
OWNER	OWN#	1	OWN#	NO	NO
OWNER	NAME	2	ONAME	NO	NO
.....	.....	...	.....	...	...

Figure 4: A basic model of RM/V2 with generalization support added.

Finally, the reader may have noticed that support for generalization hierarchies requires not only an extension of the catalog schema, but also a change in the catalog's structure wherever a generalization hierarchy occurs in the catalog schema itself. Indeed, [VELD91c] claims that the RM/V2 universe of discourse is littered with generalization hierarchies. With respect to the partial catalog structure presented, we can identify generalization structures in R_TABLE in which base tables, views etcetera are not identified as such and in the R-tables CLUSTER and COLUMN in which a generalization hierarchy is ignored (image domain as a pseudo column). Thus, the unsatisfactory implementations of generalization hierarchies in current relational databases can be found in the catalog schema itself too. The proposed extension makes it possible to eliminate these shortcomings. However, the present paper concentrates on adding a viable extension to the catalog schema instead of improving the schema

itself. Therefore, the fact that the RM/V2 self-descriptive catalog schema has become suboptimal is ignored for now.

4. ADAPTING RELATIONAL LANGUAGE

In [BOOG91a] the authors identified several problems concerning the use of SHM in current RDBMS environments. One problem has to do with problematic integrity monitoring due to the introduction of a large number of database constraints wherever generalization hierarchies occur, as discussed in section 2. However, this problem only occurs in DBMS environments that do not support generalization. Any DBMS with a catalog structure at least as powerful as the one discussed in the preceding section can enforce generalization hierarchies with a minimal number of *generalized* constraints, represented by appropriate R_TABLE en CLUSTER tuples. I-mark constraints due to over-generalization or large numbers of other constraints due to over-specialization do not occur anymore.

Another problem discussed in [BOOG91a] occurs when inquiries are made with respect to columns that are relevant to many but not all subclasses. Consider the inquiry "Retrieve all vehicle numbers and their weights". This inquiry can be translated into an very simple query: a projection on the VEHICLE relation over its columns VH# and WEIGHT (left column of figure 5 overleaf). Now consider what would be the case if the WEIGHT column were not to apply to one vehicle subclass, say the subclass of rocket vehicles. As a consequence WEIGHT ceases to be a VEHICLE column and must be transferred to the relations WIND_PR_VEH, ROTARY_VEH, JET_VEH and MAN_POW_VEH. Our simple projection query now becomes the union of three projections (middle column of figure 5 overleaf). It appears that a trivial and information preserving[3] database restructuring operation leads to significant changes in DML coding.

The need for such changes would be eliminated if the relational language would exploit the information in the extended catalog schema. We therefore propose to extend relational languages with a *GENERIC* qualifier which designates that the DML operation is to be executed on the union of all terminal relations participating in a cluster of generalization hierarchy for which at least one of the requested columns is relevant. An I-mark is displayed for every column irrelevant to a subclass. The right column of figure 5 shows what such a query would look like. It is equivalent to the other two queries depending on the specific instance of the generalization hierarchy.

The GENERIC qualifier can be used on any level in a generalization hierarchy. It has no effect if it is applied to a terminal relation in a generalization hierarchy or to a relation that does not participate in such a hierarchy. Furthermore, it is important to note that the

---

[3] Information preserving means that the information content of the database is not altered by the restructuring operation. Only the rules to which the data in the database must conform are changed. See [VELD91a] for a further discussion of this topic.

qualifier does not violate the operational closure principle of relational languages. The virtual R-table constructed by the GENERIC qualifier is the union of a variable number of relations (extended with I-marks) and is thus a relation itself. A final important point is that the addition of constraints GH07 and GH08 in figure 3c is essential if the GENERIC qualifier is to be applied properly.

The GENERIC qualifier can also be applied to INSERT, UPDATE and DELETE operations. In case of an INSERT operation though, the DBMS must ensure that the terminal R-table or R-tables to which a tuple is added can be unambiguously determined.

WEIGHT AS A GENERAL COLUMN	WEIGHT DOES NOT APPLY TO ALL VEHICLE SUBCLASSES	WEIGHT IS APPLICABLE TO ONE OR MORE VEHICLE SUBCLASSES
SELECT VH#, WEIGHT FROM VEHICLE ORDER BY VH#	SELECT VH#, WEIGHT FROM WIND_PR_VEH UNION SELECT VH#, WEIGHT FROM ROTARY_VEH UNION SELECT VH#, WEIGHT FROM JET_VEH UNION SELECT VH#, WEIGHT FROM MAN_POW_VEH ORDER BY 1	SELECT VH#, WEIGHT FROM GENERIC.VEHICLE WHERE WEIGHT IS NOT I_MARKED ORDER BY VH#

Figure 5: Three implementations of an inquiry on the vehicle hierarchy

We conclude that the extension to the catalog schema described in the preceding section and the introduction of the GENERIC qualifier in this section raise the level of logical data independence significantly for any query for which the *precise* classification in a generalization hierarchy is unimportant. Because the current generalization structure has become irrelevant for such queries, programmers' productivity is increased and recoding due to changing data structures within a generalization hierarchy is eliminated. Problems remain however, if one is interested in relevant information about objects whose specific classification within the generalization hierarchy is unknown but important. The following section gives an example of such an inquiry and offers a conceptual solution to these problems.

5. COPING WITH THE CLASSIFICATION PROBLEM

Up to now we have dealt with the generalization problem very much in the spirit of SHM[4]. However, if we take a closer look at the nature of the proposed extensions to the relational model, we find that we have only provided a solution for an aspect of a broader problem, namely that *the specific (and current) data structure relevant to a query*

[4] And also in the spirit of RM/T. The paper does not make a comparison with RM/T because only RM/V2 can be easily extended in the spirit of SHM, rather than in the spirit of RM/T.

must always be known beforehand and must be 'hard-coded' into the query structure. The solution suggested by Smith and Smith is to merge instance data with meta data and to extend relational languages with second order facilities, in other words, to permit the specification of meta data like R-tables and columns as variables in DML-statements[5]. The solution advocated in the preceding section aims at raising the abstraction level of relational languages by providing higher level data structures. Although a higher level of logical data independence is attained, the approach is essentially similar to the treatment of relational languages of subrelational constructs like indexes, or physical file structures, which are completely transparent at the relational language level.

We conclude that although the preceding section presents a practical and easy to implement solution for a serious design and programming problem, we still face the same fundamental problem that caused the introduction of the GENERIC qualifier. It is not hard to come up with realistic inquiries that are hard to implement or maintenance-prone. In some cases query formulation is merely impossible. As an example consider the inquiry "Retrieve all relevant data about the vehicles owned by owner 01". This query cannot realistically be translated into a query, not only because it concerns both instance data and structural data or meta data, but because the query structure depends on the specific time varying set of vehicles owned by owner 01. The GENERIC qualifier does not help us here because we cannot know which columns are relevant to the query and which are not. Of course we can attempt to extend the relational language further, but we can also address the problem in a fundamental way by recognizing the fact that the inquiry problem is caused by the distinction between data instances and data structures (or meta data). As long as this distinction exists it seems possible to conceive meaningful queries that straddle data and meta data and thus cannot be formulated in a relational language. The need for such queries can be expected in particular in circumstances where generalization hierarchies occur, as we shall show at the end of this section.

Basic to any solution of this problem is the viewpoint that meta data are just data. As the two preceding sections show, the boundary between data and meta data is quite fluent (compare for instance figures 2 and 4). Still, the prevalent information systems paradigm presents it more or less as fixed and axiomatic for information systems and data model design. If we drop the conceptual distinction between data and meta data we realize that the data structures and meta data instances in the catalog represent a form of redundancy. As is the case with other known forms of redundancy this phenomenon which we shall label *structural redundancy* can be removed if desired. The way in which this can be achieved is shown in figure 6 by means of the

---

[5] The SQL language already contains a primitive second order facility, namely the wildcard '*' in the SELECT clause. However, this is nothing compared to the extensions that indicated by the suggestions of Smith and Smith.

so called *implosion* mechanism[6]. The R-table VALUE, that can be considered to reference the catalog R-table COLUMN, contains the value of every data item that occurs in the database. It contains data about both data structures (columns RT_NM and COL_NM) and about instances (TUPLE_ID, VAL). Any conventional R-table can be represented in an imploded form and it can be restored by joining the VALUE R-table with itself for every column in the R-table. All information necessary to perform these operations is available in the catalog of section 3.

VALUE

RT_NM	COL_NM	TUPLE_ID	VAL
VEHICLE	VH#	1	V1
VEHICLE	OWN#	1	01
VEHICLE	PRICE	1	65.4
VEHICLE	WEIGHT	1	10.5
VEHICLE	VH#	2	V2
VEHICLE	OWN#	2	02
VEHICLE	PRICE	2	7900
VEHICLE	WEIGHT	2	840
VEHICLE	VH#	3	V3
VEHICLE	OWN#	3	01
VEHICLE	PRICE	3	12.2
VEHICLE	WEIGHT	3	1.9

RT_NM	COL_NM	TUPLE_ID	VAL
MOTORIZ_VEH	VH#	1	V1
MOTORIZ_VEH	H_POWER	1	150
MOTORIZ_VEH	FUEL_CAP	1	300
MOTORIZ_VEH	VH#	2	V2
MOTORIZ_VEH	H_POWER	2	9600
MOTORIZ_VEH	FUEL_CAP	2	2600
AIR_VEH	VH#	1	V2
AIR_VEH	MAX_ALT	1	30
AIR_VEH	TAKE_DIST	1	1000
WIND_PR_VEH	VH#	1	V3
WIND_PR_VEH	NUMB_SAILS	1	2

Figure 6: IMPLOSION: the vehicle database with structural redundancy removed

With respect to figure 6 it is important to recognize that the imploded version of the vehicle database together with the catalog content of figure 4 contains the same information as the original database of figure 2[7]. The values of TUPLE_ID have no specific meaning. In combination with the column RT_NM they just denote what VALUE tuples constitute an original (non imploded) tuple. Another important observation is that, thanks to operational closure, any query on the original database can be reformulated on the imploded database because any original R-table can always be restored using relational language.

The benefit of implosion lies in the fact that the kinds of queries we have identified as straddling data and meta data can now be formulated *without any extension* to the relational language. The need for extensions to relational languages like the GENERIC qualifier is eliminated. Note however that the need for an extended catalog schema remains.

---

[6] The implosion concept was introduced in [VELD91a] and first applied to generalization hierarchies in [BOOG91a].

[7] Obviously, there still exists structural redundancy with respect to the catalog's self-descriptive content (not shown in figure 4) and the catalog R-table structures. This remaining redundancy can be removed by imploding the catalog R-tables too.

Figure 7 displays the "impossible" query that induced the introduction of the imploded version of the vehicle database.

```
SELECT DISTINCT B.VAL, D.COL_NM, D.VAL
FROM VALUE A, VALUE B, VALUE C, VALUE D
WHERE A.RT_NM     = 'VEHICLE'
  AND A.COL_NM    = 'OWN#'
  AND A.VAL       = '01'
  AND B.RT_NM     = 'VEHICLE'
  AND B.COL_NM    = 'VH#'
  AND B.TUPLE_ID  = A.TUPLE_ID
  AND C.COL_NM    = 'VH#'
  AND C.VAL       = B.VAL
  AND D.RT_NM     = C.RT_NM
  AND D.TUPLE_ID  = C.TUPLE_ID
ORDER BY B.VAL, D.COL_NM, D.VAL
```

B.VAL	D.COL_NM	D.VAL
V1	FUEL_CAP	300
V1	H_POWER	150
V1	OWN#	01
V1	PRICE	65.4
V1	VH#	V1
V1	WEIGHT	10.5
V3	NUMB_SAILS	2
V3	OWN#	01
V3	PRICE	12.2
V3	VH#	V3
V3	WEIGHT	1.9

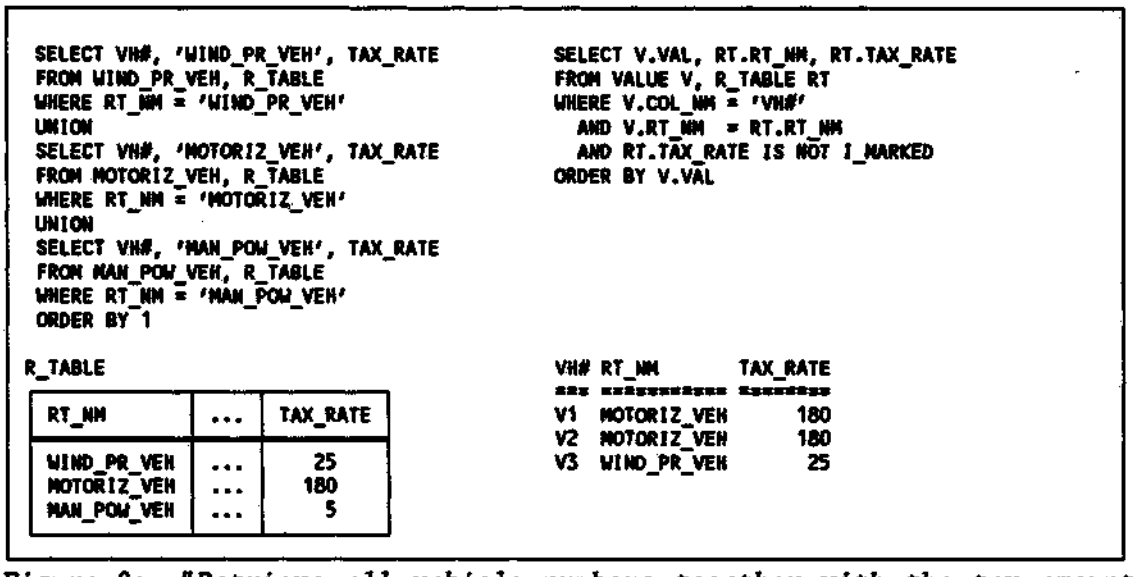
Figure 7: "Retrieve all relevant data about the vehicles owned by owner 01"

The query in figure 7 is independent of both the current generalization hierarchy and the current set of vehicles owned by 01. Equally important, extensions to the relational language are not needed anymore. The query of figure 5 can be reformulated on the imploded database without need of a GENERIC qualifier. The extension of the catalog schema with the R-table CLUSTER remains necessary.

Unfortunately, there are some problems involved with the implosion approach. First, queries on an imploded database are difficult to formulate, not only because of the large number of joins that has to be made but also because programming on multiple levels of abstraction is at least as hard as thinking on multiple levels of abstraction. Second, the presentation of the query output is not very attractive. This is no problem if the DBMS is supplied with information concerning which data are meta data with respect to the query and which are not. In the case of figure 7 the output can be converted to a conventional format (with I-marks) because COL_NM is known to the DBMS as a meta data item. Third, confronted with the imploded model, many people feel that implosion is a purely academic exercise because the performance problems attached to imploded databases are prohibitive to any implementation. Performance however is not the issue here. The imploded database is a purely *conceptual* construct. With a sufficiently powerful catalog, it is always possible to translate any DML-statement on an imploded database to an equivalent statement on a conventional database.

The impact of the implosion concept goes beyond mere generalization hierarchies. As noted above, implosion is useful whenever a query crosses the boundary between data and meta data. Consider as a final extension to the vehicle example that every vehicle is taxed according to its specific class. This example shows how fuzzy the boundary between instance data and meta data can be, because we find ourselves required to extend the catalog structure. We have to add a column

TAX_RATE to the relation R_TABLE[8]. What we have here is an example of *cover generalization*. The fact that even plain inquiries offer significant problems for query formulation is evident; a query that accesses the catalog column TAX_RATE will almost always access instance data too. Consider the inquiry: "Retrieve all vehicle numbers together with the tax amount due on them". Figure 8 depicts the new (sub-)table TAX_TABLE, a conventional SQL query, the imploded version of the query and the query results. Again the imploded version is independent of current database structure.



```
SELECT VH#, 'WIND_PR_VEH', TAX_RATE
FROM WIND_PR_VEH, R_TABLE
WHERE RT_NM = 'WIND_PR_VEH'
UNION
SELECT VH#, 'MOTORIZ_VEH', TAX_RATE
FROM MOTORIZ_VEH, R_TABLE
WHERE RT_NM = 'MOTORIZ_VEH'
UNION
SELECT VH#, 'MAN_POW_VEH', TAX_RATE
FROM MAN_POW_VEH, R_TABLE
WHERE RT_NM = 'MAN_POW_VEH'
ORDER BY 1
```

```
SELECT V.VAL, RT.RT_NM, RT.TAX_RATE
FROM VALUE V, R_TABLE RT
WHERE V.COL_NM = 'VH#'
  AND V.RT_NM  = RT.RT_NM
  AND RT.TAX_RATE IS NOT I_MARKED
ORDER BY V.VAL
```

R_TABLE

RT_NM	...	TAX_RATE
WIND_PR_VEH	...	25
MOTORIZ_VEH	...	180
MAN_POW_VEH	...	5

VH#	RT_NM	TAX_RATE
V1	MOTORIZ_VEH	180
V2	MOTORIZ_VEH	180
V3	WIND_PR_VEH	25

Figure 8: "Retrieve all vehicle numbers together with the tax amount due on them"

We conclude that, together with a catalog that supports generalization hierarchies, the implosion concept is a promising extension to the relational model.

6. CONCLUDING REMARKS

The aim of the paper has been to introduce an extension to the latest version of the relational model. This extension enables database designers and programmers to deal with generalization hierarchies in an elegant and flexible manner. It appears that a simple extension to the data structures and operators of RM/V2 is sufficient to solve many (but not all) problems induced by RM/V2's disability to support generalization hierarchies. The extensions are in fact so simple that pending support by RDBMS-products information systems designers can themselves reap the benefits of increased programming productivity and decreased maintenance effort by building their own DML-generator. Wherever complicated or instable generalization hierarchies occur they

[8] Again we ignore the fact that this change introduces another generalization hierarchy.

are advised to consider the possibility.

In the discussion of the extensions needed to support generalization hierarchies, extending the RM/V2 catalog schema has been the first step. The introduction of the implosion concept in section 5 provides justification for this (meta)data driven approach because it shows that only the addition of new data structures and constraints to the catalog schema is significant. Extensions to relational languages in order to support new abstractions are practical, but not necessary and not very powerful. The implosion approach, on the other hand, may be less practical but is certainly much more powerful, as the examples in section 5 show.

Another relevant question concerns the assessment of the implosion concept. On the one hand it must be stressed again that for the purpose of this paper the implosion concept is a *conceptual* construct. It should be seen as a kind of canonical form in which queries can be casted by the RDBMS or by application programmers. It seems that any query on an imploded database can be translated to a conventional query, based on the contents of the (possibly imploded) catalog (see also [VELD91a]). The example of the query in figure 8 indicates that the reverse translation may not be so simple because the conversion algorithm must then be able to recognize literals as representing catalog data.

On the other hand it is also possible to view implosion as an alternative way of looking data. In this view it is appropriate to speak about an imploded version of the relational model. Such a model should not be regarded as an alternative to database design nor as a model that cannot realistically be implemented due to performance limitations. It should rather be seen as a means to express queries in a way that captures more *intent*, thus offering higher levels of data independence or flexibility. The price we pay for this gain is loss of simplicity from a human point of view. It is an intriguing question, and an interesting research subject (see [BOOG91b]), whether an equally powerful but more user-friendly and comprehensible way to express queries exists. The question what other applications the imploded relational model might have in areas in which the relational model is weak, like database restructuring (see [VELD91a] and [BOOG92]) and temporal databases is hardly less intriguing.

References:

[BHAL91] **Bhalla N**, "Object-oriented data models: a perspective and comparative review," *Journal of Information Science*, 17 (1991) pp. 145-160.

[BOOG91a] **Boogaard M, Veldwijk R J, Spoor E R K**, "An Alternative Approach to Generalization in the Relational Model," To appear in the *Proceedings of the IFIP WG 8.1 Working Conference*, Alexandria/Egypt, April 1992.

[BOOG91b] **Boogaard M, Dijk M V van, Spoor E R K and Veldwijk R J**, "Inherently Flexible Information Systems," To appear in the *Proceedings of the STINFON conference*, Nijmegen/The Netherlands, December 1991.

[BOOG92] **Boogaard M, Veldwijk R J and Spoor E R K**, "Decomposition of Structural Database Alterations," *In Preparation*.

[BRAC83] **Brachman R J**, "What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks," *IEEE Computer*, 10, pp. 30-36.

[BROD84] **Brodie M L**, "On the Development of Data Models," In M.L. Brodie: *On Conceptual Modelling*, New York: Springer-Verlag, 1984, pp. 19-47.

[CODD79] **Codd E F**, "Extending the Database Relational Model to Capture More Meaning," *ACM Transactions on Database Systems* 4, no. 4, december 1979, pp. 397-434.

[CODD90] **Codd E F**, *The Relational Model for Database Management, Version 2*, Reading, Massachusetts: Addison-Wesley Publishing Company (1990).

[MCGE76] **McGee W C**, "On User Criteria for Data Model Evaluation," *ACM Transactions on Database Systems*, Vol 1 No 4 (1976)

[NIJS89] **Nijssen G M and Halpin T A**, *Conceptual Schema and Relational Database Design, A Fact Oriented Approach*, Sydney, Australia: Prentice Hall (1989)

[SMIT77] **Smith J M, and Smith, D C P**, "Database Abstractions: Aggregation and Generalization," *ACM Transactions on Database Systems*, Volume 2, No. 2, June 1977, pp. 105-133.

[VELD91a] **Veldwijk R J, Boogaard M, Dijk M V van and Spoor E R K**, "EDSOs, Implosion and Explosion: Concepts to Automate a Part of Application Maintenance in Relational Databases," *Information and Software Technology*, Vol 33 No 5 (june 1991), pp. 343-250.

[VELD91b] **Veldwijk R J, Spoor E R K, Dijk M V van and Boogaard M,** "On The Expressive Power of the Relational Model, a Database Designers Point of View," To appear in the *Proceedings of the 12th International Conference on Information systems*, New York, December 1991

[VELD91c] **Veldwijk R J, Buitendijk R B, Spoor E R K and Boogaard M,** "Towards a Catalog Standard for the Relational Model Version 2: A Manifesto," *Research Memorandum 1991-50*, Vrije Universiteit, Amsterdam (1991).

The MESDAG Research Group

INTRODUCTION

The MESDAG project is a joint project endorsed by three organizations in the Netherlands: the N.V. Nederlandse Spoorwegen (The Netherlands Railways Company), RAET N.V. and the Vrije Universiteit of Amsterdam. The MESDAG project originated at RAET N.V. during the second half of 1989 as an outgrowth of research done in the field of active data dictionary models. This research and a prototype of an active data dictionary form the basis for the mission of the MESDAG project that officially started its activities in September 1990.

MESDAG is an abbreviation of:

MEta Systems Design And Generation

MISSION AND OBJECTIVES

The mission of the MESDAG project is to prove the feasibility of developing inherently flexible information systems by introducing higher levels of logical data independence.

Derived from this mission following are the two main objectives:

1. Examine the feasibility and initiate the development of an active, self-referential data dictionary model in which both a description of the database data and a description of all specifiable application design data can be stored. This data dictionary model should contain sufficient semantic aspects (like domains, constraints and time aspects) to assure the integrity, consistency and validity of the stored (meta) data, to avoid maintenance and to support query-formulation independent of current database structure.

2. Examine the feasibility and initiate the development of the possibilities of data dictionaries in general and the described data dictionary in specific. This analysis of possibilities is directed at the embedding in and developing methods, techniques, methodologic guidelines and automated tools for the design, implementation and maintenance of flexible information systems.

MEMBERS OF THE MESDAG RESEARCH GROUP

1. Dr. E.R.K. Spoor
Dr. E.R.K. Spoor is associate professor at the Vrije Universiteit Amsterdam. He teaches and consults in the area of database systems and database development with a focus on the use of these technologies in organizations. His eighteen years of experience with computer technology includes eight years with NCR and six years with the Vrije Universiteit, first as a systems engineer and later as a computer scientist. He is one of the founders and board members of two automation oriented organizations: PSB (Amsterdam) and VDA (Hilversum).

2. Drs. R.J. Veldwijk
Drs. R.J. Veldwijk graduated from the Vrije Universiteit Amsterdam in 1986. In his quality as consultant at RAET N.V. Utrecht, he is among others responsible for the design and implementation of data models. His main interest lies in developing and implementing self-knowledgeable database models, aimed at reducing maintenance costs and at improving the accessibility of databases by end-users. Furthermore he teaches courses in data modelling.

3. Drs. M. Boogaard
Drs. M. Boogaard is assistant researcher at the Vrije Universiteit Amsterdam. Furthermore, he is part-time involved in projects by the Netherlands Railways Company. He graduated from the Vrije Universiteit Amsterdam, in August 1990. The objective of his research is to develop an approach to achieve higher levels of logical data independence for both end-users and application programs and to analyze the consequences of the level of logical data independence accomplished on the system development life cycle in general and on software maintenance and database inquiry in particular.

ACCOMMODATION ADDRESS

Vrije Universiteit
Faculteit Economie & Econometrie
Vakgroep BIK
De Boelelaan 1105
1081 HV Amsterdam Phone: +31-20-548708
The Netherlands Fax : +31-20-6462645