1992-5

ET

05348

Faculteit der Economische Wetenschappen en Econometrie

Serie Research Memoranda

Why Views Do Not Provide Logical Data Independence

J.M. de Graaff
R.J. Veldwijk and
M. Boogaard

Research Memorandum 1992-5
maart 1992

vrije Universiteit *amsterdam*

Why Views Do Not Provide Logical Data Independence

J M de Graaff, R J Veldwijk and M Boogaard

Most information systems are subject to changing requirements that can lead to alterations of the underlying database schema. Part of these alterations are information-preserving and can be regarded as schema restructuring. One of the objectives of the relational model is to provide logical data independence, i.e. to insulate application programs from the effects of information-preserving changes to the database schema. It is claimed that the relational view mechanism is sufficiently powerful to achieve logical data independence. This paper argues that the relational view mechanism fails to fulfil these claims in both a theoretical and a practical sense.

Key Words: Relational Model, Data Independence, View Mechanism, Maintenance

1. Introduction

In designing and building information systems it is necessary to design database schemata that are adequate representations of some Universe of Discourse (UoD). These schemata should capture the relevant rules, objects and object structure that apply to the environment to be modelled. For most, if not for all information systems, the UoD represented by the information system changes over time. Since the database schema has to remain an adequate representation of the UoD, changes in the UoD have to result in modifications of the database schema. However, schema modifications require time and money consuming changes to application programs that run against the schema. In order to avoid this application programs and terminal users should be independent of changes in the way data is structured and stored in the database. This quality is known as *data independence*.

One of the main objectives claimed by the relational model is to offer this independence (see, e.g. Codd 1990). Within this model physical and logical data independence are distinguished. According to Codd (1990, p. 345) *physical* data independence means that "... *application programs and terminal activities remain logically unimpaired when changes are made in storage representation, access method or both*".

Again according to Codd (1990, p. 346), *logical* data independence means that "... *application programs and terminal activities remain logically unimpaired when information-preserving changes are made to the base tables* ...". This type of independence deals with changes at the relational level instead of changes below this level. We can define information-preserving changes as modifications of the database structure with no information explicitly supplied to or deleted from the database extension. In other words, information-preserving changes do not affect the information content of the database. Making information-preserving changes can be regarded as *restructuring* the database schema.

Returning to our discussion of schema changes induced by changes in the UoD rules, we contend that these never lead to changes at the physical level directly. Modifications at the physical level are typically induced by changes in the hardware configuration, in the cardinality of the base tables, or by changing priorities assigned to application programs. Adjustability below the relational level, i.e. physical data independence, is obviously very important with respect to the efficient allocation of scarce machine resources, but has no direct relationship with the state of the UoD.

A similar look at the adjustability *at* the relational level, i.e. logical data independence, results in a picture that is not so clear. It is possible to conceive alterations of the database structure that do not reflect changes in the rules the database enforces. An example of such an alteration would be the partitioning of a table into two or more tables by rows using row content. Such an alteration would typically be effected for technical reasons. The really interesting aspect of logical data independence concerns support for schema changes that reflect changes in the rules pertinent to the UoD. An example of such a change would be the alteration of a one-to-many relationship between two tables into a many-to-many relationship.

The claim of the relational model and relational DBMS products to provide physical data independence is largely achieved. In relational DBMS environments application programs are independent of many changes at the implementation level. With respect to logical data independence the relational model offers the view mechanism as a vehicle to make application programs immune to schema changes. Because implementation of the view mechanism in relational DBMS products is still very limited (see Date 1990, p. 383) these products do not offer any practical form of logical data independence.

The question addressed by the present paper is whether the relational view mechanism is sufficiently powerful to offer logical data independence, especially with respect to information-preserving schema alterations induced by changes in the UoD (the interesting kind). With respect to this kind of changes we take the position that *the view mechanism provided by the relational model does not provide the user with logical data independence at all*. The paper attempts to show the validity of this position by means of both theoretical and pragmatic arguments. These arguments are applied to the latest version of the relational model as published by Codd (1990), rather than against some implementation of the model.

The structure of this paper is as follows. Section 2 discusses some examples of information-preserving changes as given by Codd (1990) and shows that these examples have little or nothing to do with changes in the state of the UoD. Section 3 examines the view mechanism with respect to a basic UoD-induced information-preserving change to a database schema and shows that it does not work on both practical and theoretical grounds. Section 4 discusses one of the practical problems of the application of the view mechanism in more detail. Section 5 confronts the ambitious claim that the view mechanism provides logical data independence with our equally strong claim that it does not. The paper ends with some suggestions aimed at developing a mechanism that does provide some form of logical data independence.

2. Exemplary Applications of the View Mechanism

Views can be defined as virtual relations represented by their names and definitions only. By using views the users are insulated from the base relations. Some of the advantages of using views are that they allow the same data to be seen by different users in different ways and that they provide a powerful authorization mechanism. However, the most important feature of views is to provide logical data independence. To quote Codd (1990, p.322): "... *application programs and terminal users should always use views as the means of interacting with a relational database - the only way now known for application programs and end users to be able to cope with many kinds of changes in the logical database design without the need for reprogramming and retraining. This is also known as logical data independence*".

In discussing the subject of logical data independence Codd (1990, ch. 20) provides three examples of information-preserving changes in which views should be used to provide logical data independence. These examples are listed below.

1. Partitioning a table into two or more tables by rows using row content.
2. Splitting a table into two or more tables by columns using column names, provided the original primary key is preserved in each result.
3. Combining two tables into one by meaning of a non-loss join.

Although Codd does not give a detailed description of the application of the view mechanism with respect to these examples, it is easy to conceive the view definitions necessary to represent the situation as it was before the information-preserving change. The question is, however, whether these changes are reflections of changes in the UoD.

Looking at the first example in which an table is partitioned horizontally, we find that the obvious reason to make this change is a physical one, e.g. to store the different parts of the table on different devices or different locations. In this example the UNION operator makes it possible to combine the two new tables into one view that represents the situation before the change to the database schema. Of course it is essential that the view is updatable. Because Codd's new specification of the relational model provides a feature that determines in which base relation an insertion of a new tuple should be made (1990, p. 313), the UNION-view is indeed updatable.

Although the view mechanism provides logical data independence according the definition given by Codd, the example does not represent a case in which any change in UoD-rules has occurred. Instead it demonstrates the applicability of the view mechanism to insulate the application programs from changes induced by physical considerations.

To end our discussion of Codd's first example, it should be noted that this information-preserving change (and the one described in Codd's second example) allow the database to contain data that the original database structure prohibited. Specifically, it is possible that tuples with the same values for the primary key columns occur in more than one of the partitioned relations. Therefore, constraints need to be defined to restrict the allowed contents of the database in accordance with the constraints implicitly enforced by the original schema[1] (see Veldwijk et al, 1991b).

Codd's second example concerns the replacement of an table by two or more projections that all inherit the primary key of the original relation but are otherwise disjunct. Provided no changes are made to the primary keys of the new tables, these alterations will be effected for the same kind of technical reasons mentioned in our discussion of Codd's first example. Thus, the same considerations as mentioned for this example are valid.

Codd's third example deals with combining two tables into one by means of a non-loss join. Because several kinds of non-loss joins are feasible it is unfortunate that Codd is not any more specific. Although in certain situations this alteration could represent a changing rule in the UoD, in practice the most obvious reason for this non-loss join would be to reverse the alteration as described in the second example or to denormalize a relation for performance reasons. It is clear that these reasons are also of a technical nature and are not induced by changes in the UoD.

In conclusion, the examples given by Codd do not represent information-preserving changes due to alterations in the UoD. They only demonstrate the fact that views provide a kind of extended physical data independence.

3. The View Mechanism and Changes in the Universe of Discourse

This section deals with an information-preserving schema change that is unmistakably caused by a change in the rules of the UoD the database schema reflects. The example is a very basic one which has been discussed before by Date (1986, ch. 19). It concerns a database containing information on employees and the departments they work for. Every employee works for exactly one department (one-to-many relationship) and each attribute is mandatory (not null). The normalized database schema is displayed in figure 1.

[1] Theoretically, the information-preserving change could be induced by the wish to drop these constraints. If so, the example would indeed reflect a change in the UoD.

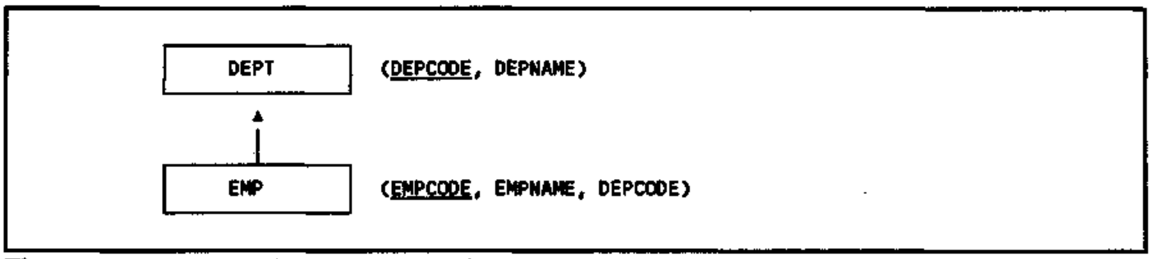
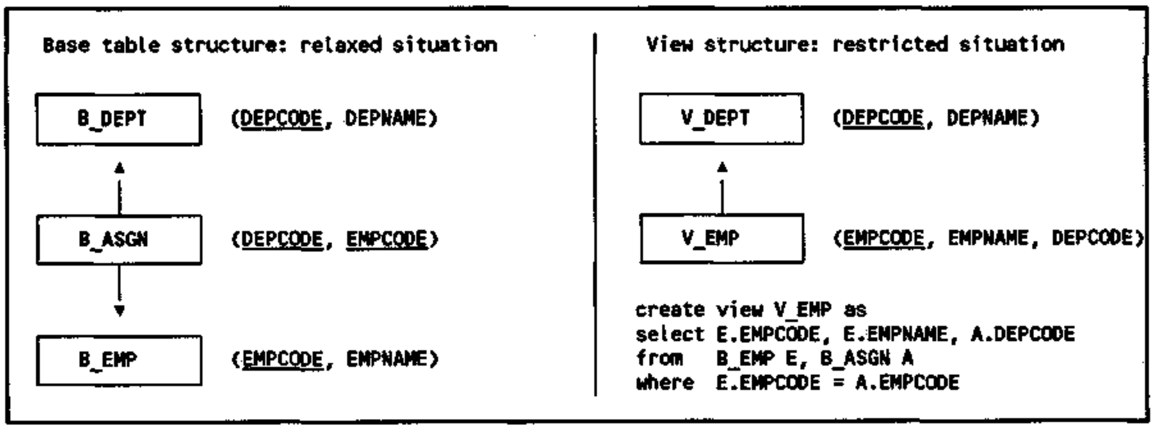


Figure 1. *Normalized database schema for the employee database*

In this figure, the primary key of each relation is underlined. The arrows represent foreign key to primary key references between the relations. During the design process the database designer could pay attention to the eventuality of changing requirements in time. For instance, the constraint that any employee must work for exactly one department might be relaxed. Because this constraint is enforced by the schema of figure 1 such a constraint relaxation leads to a different database schema in which a many-to-many relationship between employees and departments occurs.

In order to prepare for the structural rearrangements, the database designer is supposed to apply the view mechanism. In theory there are two possible approaches to use views in this situation. One is to have the views reflect the situation of figure 1 while the base tables reflect the many-to-many database structure. In the second approach the situation is reversed: the base tables reflect the situation of figure 1 while the views reflect the many-to-many database structure. We shall describe both approaches and we shall demonstrate that both fail to provide immunity from changes to the schema.

Figure 2 depicts the first approach in which the views reflect the restricted (one-to-many) state of the UoD. The view V_DEPT is a one-to-one representation of the base table B_DEPT.


Base table structure: relaxed situation
B_DEPT
(DEPCODE, DEPNAME)
B_ASGN
(DEPCODE, EMPCODE)
B_EMP
(EMPCODE, EMPNAME)
View structure: restricted situation
V_DEPT
(DEPCODE, DEPNAME)
V_EMP
(EMPCODE, EMPNAME, DEPCODE)

```
create view V_EMP as
select E.EMPCODE, E.EMPNAME, A.DEPCODE
from   B_EMP E, B_ASGN A
where  E.EMPCODE = A.EMPCODE
```

Figure 2. *First approach: base tables many-to-many, views one-to-many*

As will be obvious, this application of the view mechanism solves nothing because the application programs that run against the two views must be rewritten if the state of the UoD represented by the base-table structure ever becomes real. Because the application programs are not immune to information-preserving schema changes logical data independence is not achieved. This observation lends itself to generalization too: every application of the view mechanism that is more restrictive than the base table structure on which the views are defined fails to provide immunity from certain information-preserving schema changes.

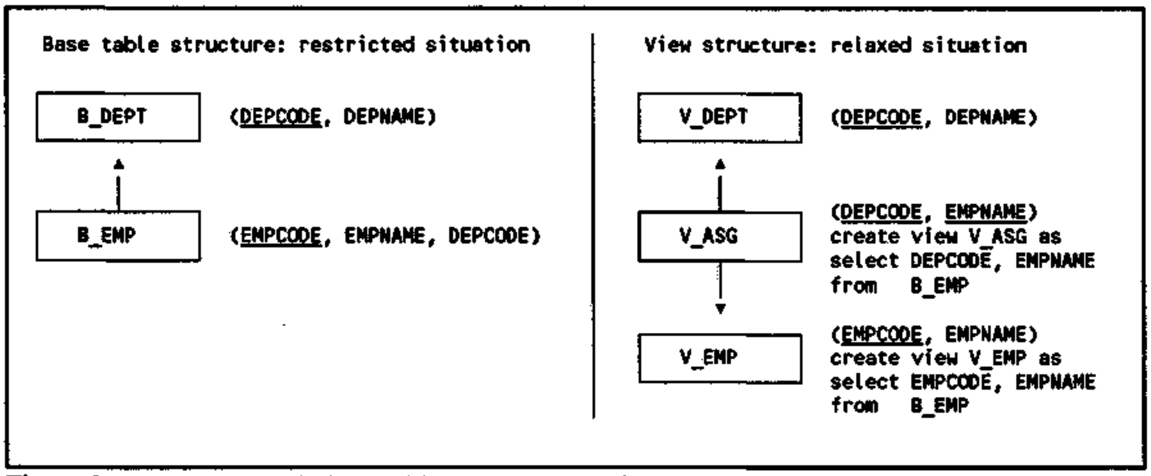


Figure 3. *Second approach: base tables one-to-many, views many-to-many*

The second design option, depicted in figure 3, seems much more promising at first. Here the views reflect the relaxed (many-to-many) situation, while the base tables reflect the restricted (one-to-many) situation. In this approach the schema alterations that are foreseen do not affect the application programs since these were designed to operate on a database structure reflecting a many-to-many relationship. Changes occur only in the structure of the base tables and in the view definitions. To be specific: the base table B_EMP is replaced by two projections that conform fully to the definitions of the projection views V_EMP and V_ASGN. These views V_EMP and V_ASGN become copies of the new base tables B_EMP and B_ASGN, just like V_DEPT is a copy of B_DEPT.

Unfortunately there are certain problems attached to this approach. First, it leads to an increase in programming effort because extra joins are necessary. The programmers find themselves in a situation in which they have to create complex DML-statements that the view translation algorithm has to translate into simpler ones. Second, information-preserving changes like the one in the example are often accompanied by changes in the user-interface. If screen and print lay outs have to be adjusted anyway, the view mechanism alone cannot provide full immunity from application program changes. Third, the database structure quickly becomes unintelligible for all those who have knowledge about the real state of the UoD, because the UoD suggested by the schema does not correspond to the UoD as it is known. Both these observations lead to the conclusion that application of the view mechanism to obtain logical data independence is not without a price.

Third, and all important is the fact that the views V_EMP and V_ASGN are non-updatable according to the powerful view updating algorithm developed by Codd (1990, ch. 17). To demonstrate this we must first examine how the addition of a new employee must be translated from the view level to the base table level.

Suppose employee 'Smith' with code 'E4' working for the existing department 'D2' is inserted into the database. At the view level this requires a transaction consisting of two insert operations:

```
1)   insert into V_EMP  (EMPCODE, EMPNAME) values ('E4', 'Smith')

2)   insert into V_ASGN (DEPCODE, EMPCODE) values ('D2', 'E4')
```

The view updatability mechanism must translate these statements into equivalent statements on the base table B_EMP:

```
1)   insert into B_EMP (EMPCODE, EMPNAME) values ('E4', 'Smith')

2)   update B_EMP set DEPCODE = 'D2' where  EMPCODE = 'E4'
```

Although both single operations temporarily violate the integrity constraint that each property of B_EMP is mandatory, the transaction as a whole does not. However, the second insert must be translated into an update operation on the underlying base table B_EMP (irrespective of sequence).

If we apply Codd's powerful view updatability (VU) algorithm[2] to the example, we instantly run into difficulties because transformations of relational operators (like 'insert') into different relational operators (like 'update') are categorically forbidden by one of the basic assumptions underlying Codd's VU algorithm (1990, p. 298). Provided the VU-algorithm determines the views V_EMP and V_ASGN to be updatable[3], the two insert operations on the views will both be translated into insert operations on the base table B_EMP. Because both inserted tuples have the same value 'E4' for their primary key attribute EMPCODE the transaction will fail. To conclude, the views V_EMP and V_ASGN are not fully updatable and support for logical data independence, if practical, is found to be very problematic if judged by the VU-algorithm.

[2] Although Codd's paper discusses two view updatability algorithms, namely VU-1 and VU-2, the differences between these algorithms are not relevant to the discussion in this paper. Therefore, the term *VU-algorithm* is used without qualification.

[3] In fact, the VU-algorithm will determine that insert operations are not allowed on the views because both lack a column that is mandatory in the underlying base table. The update and delete operators can be applied to the views. If the columns EMPNAME and DEPCODE were not mandatory the views would accept the insert operator.

4. Practical Problems

The previous section has shown that, in case of a common information preserving change, views cannot provide logical data independence because they suffer from updatability problems. But even if we presume that a VU-algorithm can be developed that is powerful enough to handle all possible information preserving changes the application of the view mechanism is still accompanied with major practical problems. One of the most important problems is the extra programming effort required.

Suppose the UoD as described in section 3 is a little bit more complicated. The database representing this UoD not only contains information about the department an employee works for, but also information about his salary and his working hours and information about his job. The normalized structure of this database is depicted in figure 4.

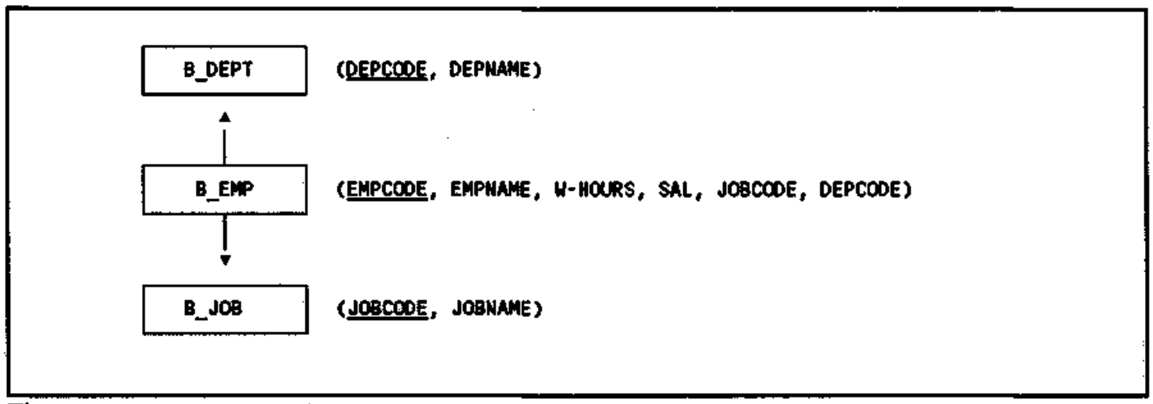
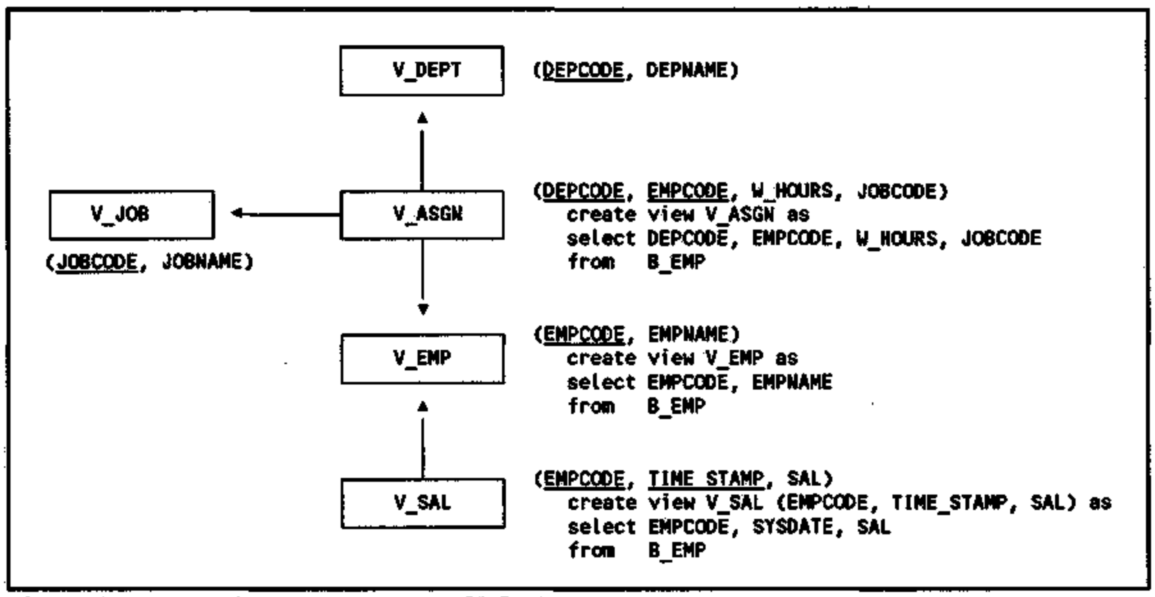


Figure 4. *Normalized database structure*

In order to attain a stable database, one has to consider what changes might appear. Suppose that for the above database one can conceive two possible changes. The first one is that the one-to-many relationship between departments and employees changes into a many-to-many relationship. The second one is that the information about the salary of an employee becomes historical. The view structure necessary to represent this relaxed situation in the UoD is displayed in figure 5[4]. In this figure the columns W_HOURS and JOBCODE are transferred from V_EMP to V_ASGN because this information is necessary for each department an employee works for. Furthermore the column TIME_STAMP is

[4] Note that the views suffer from the same updatability problems as dicussed in section 3.

added to represent the history of the salary of an employee[5]. An implementation dependent expression, namely the ORACLE system date SYSDATE, is used to assign a value to TIME_STAMP. Views V_DEPT and V_JOB are one-to-one representations of the base tables B_DEPT and B_JOB.



```
create view V_ASGN as
select DEPCODE, EMPCODE, W_HOURS, JOBCODE
from   B_EMP

create view V_EMP as
select EMPCODE, EMPNAME
from   B_EMP

create view V_SAL (EMPCODE, TIME_STAMP, SAL) as
select EMPCODE, SYSDATE, SAL
from   B_EMP
```

Figure 5. *Views to simulate the possible UoD situation*

In order to examine the practical consequenses of this approach for the application programs, we will examine a simple information request. Consider for example the question: " *Retrieve the name, the department name and the jobname of all employees with a current salary that exceeds 50.000.*" The query necessary to retrieve this information from the restricted structure is as follows:

```
select EMPNAME, DEPNAME, JOBNAME
from   B_DEPT D, B_EMP E, B_JOB J
where  D.DEOCODE = E.DEPCODE
and    E.JOBCODE = J.JOBCODE
and    SAL       > 50000
```

The query necessary to retrieve the same information from the views representing the relaxed situation is as follows (overleaf):

[5] It can be argued that adding a column TIME_STAMP is no information preserving change. An answer to this objection is that each relation in a database has a virtual attribute TIME-STAMP with a value that represents the present. In otehr words the common change of making information historical can be regarded as information-preserving.

```
select DEPNAME, EMPNAME, JOBNAME
from   B_DEPARTMENT D, ASSIGNMENT A, B_EMPLOYEE E, B_JOB J
where  D.DEPCODE = A.DEPCODE
and    A.EMPCODE = E.EMPCODE
and    A.JOBCODE = J.JOBCODE
and    50000     <
       (select SALARY
        from   SALARY S1
        where  S1.EMPCODE = E.EMPCODE
        and    TIME_STAMP =
               (select MAX(TIME_STAMP)
                from   SALARY S2
                where  S1.EMPCODE = S2.EMPCODE))
```

In the query based on the view structure it is necessary to execute five joins as compared with the two joins necessary for the normalized database structure. This means that programmers have to create complicated statements that are translated into simpler ones by the view translation algorithm. In other words, even for a small UoD with two simple possible changes the application of the view mechanism requires substantially more programming effort as compared with a restricted database structure. If one imagines a much more complicated UoD it is clear that the utilization of views to achieve logical data independence is highly unpractical and requires major investments in the development phase of the information system. The relevance of this matter even increases if one considers the fact that it is not sure whether the anticipated changes will ever occur or whether the changes that will occur are anticipated.

5. Confronting the Claims

It is possible to raise objections against our claim that the view mechanism does not support logical data independence. One objection could be that our criticism is only directed against Codd's present VU-algorithm, which could someday be sufficiently enhanced. Another objection could be that we have given only one example of a situation in which the view mechanism fails to provide logical data independence, which is far from proving that the view mechanism does not provide *any* support for logical data independence.

Our answer to the first objection is that Codd has specified the assumption that relational operations on views may not be translated into different operations on base tables for a good reason, i.e. to make the updating of views comprehensible for the users (1990, p.298). If the assumption were dropped and the VU-algorithm were made sufficiently powerful to overcome its present limitations, the effects of view updating would quickly become unpredictable by database administrators. As a consequence the view mechanism would only be applied reluctantly for fear of unexpected consequences. A second answer to the objection is that it will probably prove to be very hard or even impossible to come up with a sufficiently powerful extended VU-algorithm. For one thing, the algorithm should be able to evaluate any number of view update statements in one transaction and translate these into a number of different update statements on the underlying base tables. As a third and final answer, we feel that whoever makes a claim should either fully prove it or make clear to what extent the claim holds. To be specific, any partial refutation of the claim that views provide logical data independence should either lead to a stronger VU-algorithm or to a clear and reliable redefinition of the concept of logical data independence[6].

The objection that we have only given one example of the failure of views to provide immunity to information-preserving changes can be answered in several ways. First of all, our last answer to the preceding objection is also an answer to this one. A second answer would be that failure to support the common information-preserving schema alteration we have discussed would severely limit the practical applicability of the view mechanism. As a third answer we can observe that limited view updatability reduces the understandability and reliability of the view mechanism. This argument reflects the same philosophy lying behind Codd's assumption that update operations on views may not be translated into different update operations on base tables. As a fourth and final answer, we have found it very hard to find non-trivial information-preserving schema alterations that do not offer the kind of view update problems described in section 3.

6. Alternatives for the View Mechanism

The discussion of the preceding sections shows that the fundamental question is whether or not it is possible to offer logical data independence without limitations, i.e. to find some mechanism that makes it possible to accept any kind of information-preserving change to a database schema without requiring changes to application programs. Such a facility would remain extremely desirable, even considering the observation of section 3 that information-preserving changes are often accompanied by changes in the information system's user interface.

Without pretending that we can offer anything like the ultimate solution we can distinguish two alternative approaches that can lead to a level of logical data independence that is at least practical in the sense that it supports the most common schema changes.

The least ambitious approach would be to enumerate a limited number of commonly occurring information-preserving schema alterations. For each

[6] In fact, Codd provides a more precise definition of the term 'logical data independence' (1990, p. 346) by restricting it to information-preserving changes supported by his algorithm VU-1 *or by a stronger algorithm*. This definition of the term only specifies a lower limit of logical data independence. Moreover, if the added restriction were to rule out the vast majority of information-preserving schema changes, the definition itself would become questionable.

kind of alteration an algorithm would have to be developed that makes it possible to modify both data structures and application programs. This implies that the application programs would have to be changed but these changes would be executed automatically. This would amount to a kind of surrogate logical data independence. View updating problems do not occur because no views are involved. The main limitation is that solutions like this, which have been described by Shneiderman and Thomas (1982) and by Veldwijk et al (1991a), provide at best a pragmatic solution but never one that is applicable to all relevant schema changes.

A very ambitious approach would be to develop a data model on top of the relational model which would deal with information-preserving schema changes in the same way in which the relational model deals with alterations at the physical level (see Boogaard et al, 1991). Such a model should offer even higher level data structures to its users. What such model should look like is hard to tell, although it is obvious that its highest level data structures may not reflect any UoD rule that is liable to change. The development of such a model may very well prove to be a mirage, if only because the number of conceivable information-preserving schema alterations is huge, unlike the number of possible alterations at the physical level. Nevertheless, it may be possible to develop a model in which a limited number of commonly occurring information-preserving schema alterations is made transparent. With respect to logical data independence, however, this would greatly reduce the ambition level of the approach.

In conclusion, we would like to draw attention to the fact that over twenty years after the introduction of the relational model, adequate support for logical data independence is still not in sight. Because of the huge beneficial effects this support would have on the maintenance costs of today's many relational information systems, relational DBMS vendors and users have every incentive to be interested in the outcome of any research in this area.

ACKNOWLEDGEMENT
The authors would like to thank dr. Edu Spoor for his comments on earlier versions of this paper.

REFERENCES

Boogaard M, Dijk, van M V, Spoor E R K and Veldwijk R J (1991), "Inherently Flexible Information Systems", in: *Proceedings of the first STINFON Conference*, Nijmegen, The Netherlands, December 1991.

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

Date C J (1986), "Updating Views" in *Relational Database, Selected Writings* Reading, Massachusetts: Addison-Wesley Publishing Company.

Date C J (1990), *An Introduction to Database Systems: Volume I*, 5th edition, Reading, Massachusetts: Addison-Wesley Publishing Company.

Shneiderman B and Thomas G (1982), "An Architecture for Automatic Relational Database System Conversion", in: *ACM Transactions on Database Systems*, 7, 2, pp. 235-257.

Veldwijk R J, Boogaard M, Dijk M V van and Spoor E R K (1991a), "EDSOs, Implosion and Explosion: Concepts to Automate Part of Application Maintenance of Relational Databases", in: *Information and Software Technology*, 33, 5, pp. 343-350.

Veldwijk R J, Spoor E R K, Boogaard M and Dijk M V van (1991b), "On the Expressive Power of the Relational Model, a Database Designers Point of View", in: *Proceedings of the 12th International Conference on Information Systems*, New York, December 1991.

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 advanced database architectures. 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.

4. J.M. de Graaff
J.M. de Graaff has been attached to the MESDAG research group as a trainee in the period from September 1991 till March 1992. He has planned to graduate from the Vrije Universiteit Amsterdam, in July/August 1992.

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

1991-1	N.M. van Dijk	On the Effect of Small Loss Probabilities in Input/Output Transmission Delay Systems
1991-2	N.M. van Dijk	Letters to the Editor: On a Simple Proof of Uniformization for Continious and Discrete-State Continious-Time Markov Chains
1991-3	N.M. van Dijk P.G. Taylor	An Error Bound for Approximating Discrete Time Servicing by a Processor Sharing Modification
1991-4	W. Henderson C.E.M. Pearce P.G. Taylor N.M. van Dijk	Insensitivity in Discrete Time Generalized Semi-Markov Processes
1991-5	N.M. van Dijk	On Error Bound Analysis for Transient Continuous-Time Markov Reward Structures
1991-6	N.M. van Dijk	On Uniformization for Nonhomogeneous Markov Chains
1991-7	N.M. van Dijk	Product Forms for Metropolitan Area Networks
1991-8	N.M. van Dijk	A Product Form Extension for Discrete-Time Communication Protocols
1991-9	N.M. van Dijk	A Note on Monotonicity in Multicasting
1991-10	N.M. van Dijk	An Exact Solution for a Finite Slotted Server Model
1991-11	N.M. van Dijk	On Product Form Approximations for Communication Networks with Losses: Error Bounds
1991-12	N.M. van Dijk	Simple Performability Bounds for Communication Networks
1991-13	N.M. van Dijk	Product Forms for Queueing Networks with Limited Clusters
1991-14	F.A.G. den Butter	Technische Ontwikkeling, Groei en Arbeidsproduktiviteit
1991-15	J.C.J.M. van den Bergh, P. Nijkamp	Operationalizing Sustainable Development: Dynamic Economic-Ecological Models
1991-16	J.C.J.M. van den Bergh	Sustainable Economic Development: An Overview
1991-17	J. Barendregt	Het mededingingsbeleid in Nederland: Konjunktuurgevoeligheid en effektiviteit
1991-18	B. Hanzon	On the Closure of Several Sets of ARMA and Linear State Space Models with a given Structure
1991-19	S. Eijffinger A. van Rixtel	The Japanese Financial System and Monetary Policy: a Descriptive Review
1991-20	L.J.G. van Wissen F. Bonnerman	A Dynamic Model of Simultaneous Migration and Labour Market Behaviour