Comprehensive Approach to the Design of Relational Database Schemes

by

Catriel Beeri and Michael Kifer

Institute of Mathematics and Computer Science
The Hebrew University of Jerusalem

ABSTRACT

We propose a new approach to the design of relational database schemes. The main features of the approach are:

(a) A combination of the traditional decomposition and synthesis approaches, thus allowing the use of both functional and multivalued dependencies.

(b) Separation of structural dependencies relevant for the design process from integrity constraints, i.e., constraints that do not bear any structural information about the data and must therefore be discarded at the design stage. This separation is supported by a simple syntactic test filtering out non-structural dependencies.

(c) Automatic correction of schemes that lack certain desirable properties.

1. INTRODUCTION.

We use the general framework and the concepts as developed by the classic design theory during the last decade [Ber, BBG, Fa, FMU, Li, ZM, Sa, Sc3]. A *universal database scheme* is a collection of attributes (a *universe*) and a collection of *data dependencies*. The problem addressed by the theory is how one can derive a database scheme with certain desirable properties from a given universal scheme. A database scheme is a collection of subschemes of the universe that contain all the attributes. Among the properties considered in the literature are: *preservation of dependencies*; normal forms like *3NF*, *BCNF*, *4NF* [Ul1]; *schema independence* [Sa]; *acyclicity* [FMU] etc. Unfortunately, 3NF is the only goal that is achievable within the classic framework [BB, Fa, Sc1, Sc2].

The classic approach described above suffers from several drawbacks. The first concerns the method for deriving the database scheme. For that purpose two different methods have been proposed. The first is scheme *synthesis* [Ber]; the other, known as scheme *decomposition*, was proposed by Codd and generalized by Fagin [Fa]. Scheme synthesis works well in the context of functional dependencies (abbr. FD's), while scheme decomposition is suited for multivalued dependencies (abbr. MVD's). So far, the two approaches were considered to be incompatible.

The success of a design theory depends on its ability to supply satisfactory answers to the situations which inherently need different types of dependencies. Different researchers tried to cope with the case where MVD's and FD's are brought together. Lien [Li] adheres to the decomposition approach considering the FD's as MVD's and neglects the different semantics of FD's and MVD's. Melkanoff and Zaniolo [ZM] also tried to cope with the problem, giving first priority to the notion of faithful representation of dependencies. However, both

*Second Author's address: Dept. of Computer Science, SUNY at Stony Brook, Stony Brook NY 11794, USA.

Proceedings of the Tenth International Conference on Very Large Data Bases.

Singapore, August, 1984

methods are unsatisfactory. Firstly, they fail when the MVD-set is not *conflict-free* [Li]. Secondly, neither approach explains the roles of different types of dependencies and how they influence the design process.

The second drawback is the assumption of a fixed environment; that is, the assumption that the set of attributes and the set of dependencies are fixed. Typically, in practical database design the initial specifications are not preserved and new attributes and dependencies are added or deleted by the designer in the course of the design process. Such an activity can be viewed as tuning the initially obtained specifications to better reflect the designer's intention. The tuning is necessary because it is common that the initial specifications may miss some important information; also, they may contain some details irrelevant to the structuring of data. The third issue is that the classical theory neglects to distinguish between dependencies that reflect structural properties of the data and those that are merely integrity constraints.

We propose in this paper a general approach to database scheme design that solves these problems. Our approach takes into consideration the different roles of FD's and MVD's, and works well if both types are given. It incorporates both the synthesis and decomposition approaches and employs the ideas about scheme corrections [BeKi1,Ki]. We conjecture that it can also serve as a framework for the cases when other types of dependencies take part.

The paper is organized as follows. In Section 2 we discuss the semantic role of dependencies in the design process. Section 3 presents the ideas behind our approach and the algorithm. Section 4 presents conclusions.

2. THE ROLE OF DEPENDENCIES IN DATABASE DESIGN.

We assume that the reader is familiar with the theory of FD's, MVD's, and join dependencies (abbr. JD's), and with the notions of third (abbr. 3NF) and forth (abbr. 4NF) normal forms on the level of [Ull]. We also assume that the reader is familiar with acyclic [FMU, BFMY] database schemes.

For the MVD-set $\{X \twoheadrightarrow V_i \mid i=1,...,n\}$ we shall often write $X \twoheadrightarrow S$, where $S=\{V_i \mid i=1,...,n\}$. Alternatively we shall write $X \twoheadrightarrow V_1 \mid ... \mid V_n$ instead of $X \twoheadrightarrow S$. As usual $DEP_D(X)$ denotes the dependency basis of X w.r.t. the dependency set D [B]. We shall omit the dependency set if it is obvious or immaterial. An MVD of the form $X \twoheadrightarrow DEP(X)$ is called *full* [BMFY] and a set of MVD's containing only full MVD's is called a *full set of MVD's*. If D is a set of FD's or MVD's we denote by $\mathbf{LHS}(D)$ the set of left-hand sides (abbr. LHS) of the members of D. Finally we note that it is the usual practice in the database theory to write XY for $X \cup Y$, where X and Y are sets of attributes.

Consider the MVD $X \twoheadrightarrow V \mid W$. Intuitively, it denotes the fact that there is *no* direct relationship between the attributes in V and W and, possibly, there is some relationship between X and V and between X and W. For instance, the MVD *Person* $\twoheadrightarrow$ *Child* | *proJect* states that persons and their children are related and persons and the projects they work for are related, while there is no relationship between a child and a project except for the indirect connection via a person which may be the father of the child and is currently working for the project.

This explains the drawback of the approaches that consider FD's as just MVD's. In fact, an FD is *not* a 'pure' MVD for it is not intended to represent the first aspect (the indirect relationship) of the semantics of an MVD. Let us extend the above example as follows. Assume the attributes are *P(erson)*, *C(ild)*, *(home)A(ddress)*, *Z(ip)* and *(pro)J(ect)* with the obvious relationships between them. Let the dependencies be $P \twoheadrightarrow C \mid J$, $P \rightarrow A$ and $A \rightarrow Z$. This implies the MVD $P \twoheadrightarrow A \mid Z \mid C \mid J$. This MVD (by itself) seems to state that children, addresses and zipcodes are only indirectly related. However, it is obviously not the case here. On the other hand, we can be still justified in inferring that children are indirectly related to the projects.

Another problem in scheme design is that the dependencies may represent not a presence or an absence of a close

relationship between the attributes but rather a constraint which has little influence on the way the data should be structured. This distinction is due to [FMU] from where the following example is taken, slightly modified.

Example 1. Suppose our attributes are $C(ourse)$, $T(eacher)$, $R(oom)$, $H(our)$, $S(tudent)$ and $G(rade)$ with FD's $C \to T$, $TH \to R$, $HS \to R$, $HR \to C$ and $CS \to G$ and MVD $CT \twoheadrightarrow HR \mid GS$. This MVD expresses the fact that a course may meet more than once a week and any student attends all the meetings of the course he takes. Here, $HS \to R$ expresses the physical fact that students cannot be in two places at once. From it and $HR \to C$ one can derive $HS \to C$, which means that students' time tables must be feasible, i.e., no student is permitted to take courses whose meetings clash. However, these facts about time and what a student is permitted to do have nothing to do with the way the data is structured; they do not represent any new basic relationship between the attributes, only restrictions on the way the students attend the courses. ■

The phenomenon illustrated by the above example is explained by that an FD (as used in the classical design theory) is intended to express both a basic relationship and an integrity constraint. The 'anomalous' FD of Example 1 expresses only the second aspect. So it is better to say that it is a constraint which syntactically (but not semantically) appears as an FD. In summary, a dependency may express the following facts:

(*i*) an integrity constraint

(*ii*) a basic relationship

(*iii*) an indirect relationship, i.e., separation.

Clearly, only (*ii*) and (*iii*) are significant in scheme design. We have seen that FD's always express (*i*) and sometimes (*ii*). Hence, an FD that does not express (*ii*) must be disregarded in design process. We also conjecture that MVD's which are relevant to the design process always express (*iii*) and sometimes (*ii*).

The discussion above shows that syntactic specifications such as data dependencies are not quite suitable as a description of the structural information about the data. The design process deals with database schemes, i.e., with the intentional aspect of databases. It normally exploits information about the internal structure of schemes such as close relationship among the attributes, separation of attributes, etc. Though data dependencies were extensively used in design theory, their role in structural descriptions was not quite clear. Usually it was (implicitly) assumed that the role of the data dependencies as structural descriptors follows from their role as integrity constraints. Apparently, it is not so.

The following observation supports this claim. Data dependencies, in their role as integrity constraints, are monotonic in the sense that addition of new dependencies does not contradict the old ones (i.e., the extended set has a satisfying database). However, if we look at the structural role of the dependencies, they are no more monotonic. For instance, The MVD $C \to A \mid B$ specifies a separation of A from B. However, when the FD $A \to B$ is added, it specifies the fact that A and B are closely related and thus changes the structure that was suggested by the MVD.

Thus, some of dependencies are both structural descriptors and integrity constraints. We call them *structural dependencies*. Other dependencies are devoid of the structural aspect and we call them *nonstructural dependencies* (or *integrity constraints*). Integrity constraints should be checked routinely at run time but are irrelevant at the design stage.

Certainly, the distinction between structural dependencies and integrity constraints concerns only the basic facts such as dependencies from a minimal cover and not all the implied dependencies in the closure of a dependency set. It is well known that different covers may exist. This gives rise to the problem of determining of the best one. The classical methods are all dependent on the choice of a particular minimal cover which is a very serious problem. We shall see how this problem can be facilitated within the nonconventional framework.

3. HOW TO DESIGN.

3.1. The Decompositional Approach and the Splitting Graph.

We discuss in this section the decomposition approach, assuming that only MVD's are given. The basis of this approach is the *separationprinciple* of [BBG], and its declared goal is 4NF. However we show that attaining 4NF does not solve the problems it is supposed to solve. The concept of *splitting*, introduced in [BFMY, Sc1], is shown to provide a better goal.

Let (U,M) be a universal scheme with MVD-set M. From now on assume that M is a set of *full* MVD's. An MVD $X \twoheadrightarrow DEP(X)$ *splits* a set of attributes $Y \subseteq U$ if for some distinct $V,W \in DEP(X)$ both $V \cap Y \neq \phi$ and $W \cap Y \neq \phi$ holds. X is split by M if it is split by some MVD of M.

Let $X \twoheadrightarrow V \mid W$ be an MVD. The idea underlying 4NF is that XVW should not be in one relational scheme, because this introduces redundancies and potential consistency problems. Thus, in the decomposition approach, this MVD is used to decompose XVW into XV and XW. It can be used to decompose any scheme Y, provided that Y contains X. Hence, MVD's can be used to decompose schemes until 4NF is achieved. However, we note that once another MVD that splits X is used then $X \twoheadrightarrow V \mid W$ may be not used in the decomposition process [Li]. In such a case this MVD imposes an interrelational constraint which may cause consistency problems.

Another problem of the decomposition approach is that it may yield schemes whose attribute sets are split. Assuming M consists only of structural MVD's (in the sense of the previous section) then, if a pair of attributes A,B is split, we can conclude that A and B are not closely related. Putting them into one scheme violates the separation principle of [BBG]. Example 2 below shows that this may cause the same redundancy problems as in the case of the non-4NF schemes. Hence, in this case 4NF does not achieve its goal.

Example 2. ([Sc1]) Let $U = ATL$, where a tuple atl means that A(*uthor*) a publishes T(*itle*) t in L(*ocation*) l. Suppose there are the following MVD's: $m_1 : A \twoheadrightarrow T \mid L$ and $m_2 : T \twoheadrightarrow A \mid L$. Fagin's decomposition algorithm may be applied in two ways resulting in two different decompositions. If m_1 is applied first then we obtain $R_1 = AT$, $R_2 = AL$ and m_2 is now inapplicable. If m_2 is applied first, then the resulting schemes are $R_1' = AT$ and $R_2' = TL$ and m_1 cannot be further applied. In both cases, R_2 and R_2' contain a pair of split attributes. Consider the following relation:

A	T	L
a_1	t_1	l_1
a_1	t_2	l_2
a_1	t_1	l_2
a_1	t_2	l_1

It satisfies both dependencies. If we choose the decomposition $\{AT, TL\}$ then we have

r_1 = A	T
a_1	t_1
a_1	t_2

r_2 = T	L
t_1	l_1
t_2	l_2
t_1	l_2
t_2	l_1

We see that r_2 is a Cartesian product of its projections on T and L. This is exactly the situation that was supposed to be prevented by 4NF. Of course, the decomposition $\{AT, AL\}$ has the same drawback. ■

Assuming that the attribute sets of schemes should not be split, leads directly to the following concept. Let $\mathbf{R} = (R_1, \ldots, R_n)$ be a database scheme over the universal scheme (U,M). We say that $\mathbf{R}$ is in the *split free normal form* (abbr. SFNF) if no R_i is split by M. Obviously, SFNF implies 4NF.

To obtain a SFNF database scheme we propose to use the splitting graph of [BMFY] as the guide of the design process The *splitting graph* $SG(M,U)$ has U as a node set. Two nodes of $SG(M,U)$ are connected by an edge iff the corresponding pair of attributes is not split by M. It is easily seen that the coarsest SFNF database scheme over (U,M) is the scheme consisting of the *maximal cliques* of $SG(M,U)$. (A set of nodes K of graph G is a *clique* if any pair of nodes of K is connected by an edge.)

Thus, one might be tempted to use this scheme as an SFNF database scheme.

Unfortunately, maximal cliques cannot always be used without loss of the representation power of the scheme. In Example 2 the maximal cliques are $\mathbf{R} = \{AT, L\}$. But the relation

A	T	L
a	t	l
a'	t'	l'

cannot be stored into $\mathbf{R}$ without disconnecting the L-values from the values of the other attributes. Stated a little differently, this relation does not satisfy the JD associated with $\mathbf{R}$, hence is not equal to the join of its projections on the schemes of $\mathbf{R}$. Such a loss of information does not occur if M is equivalent to a *single join dependency* [BFMY]. If this is the case, then this JD is *acyclic* [FMU, BFMY] and the maximal cliques of $SG(M,U)$ are precisely the constituents of this JD.

Having shown the disadvantages of 4NF, we have introduced a stronger normal form and we have shown that it can be achieved when the given universal scheme is acyclic. However, schemes are not always acyclic, so we may have once more introduced a nicer normal form that is not always achievable. One of our goals in this paper is to provide evidence to support the claim that this normal form can be achieved in the databases that occur in practice.

Let us then consider the situation when M is not equivalent to an acyclic JD. We say that M has a *split-key* anomaly [Li] if at least one $X \in \mathbf{LHS}(M)$ is split by M. Otherwise, M is called a *nonsplit-key* set of (full) MVD's. M has an *intersection anomaly* [BeKi1] if there are $X, Y \in \mathbf{LHS}(M)$ s.t.

$$(X \cap Y) \twoheadrightarrow (DEP(X) \cap DEP(Y)) \notin M^+.$$

It turns out [Li] that M is equivalent to a single JD iff it has neither split-key nor intersection anomalies. After Lien [Li] we call such schemes *conflict-free*.

We conjecture that in real-world situations LHS's of structural MVD's are not split (i.e., the split-key anomaly does not occur). A similar assumption was advanced in [Sc1]. Of course, we refer only to the MVD's in a minimal cover. As a motivation, consider some structural MVD $X \twoheadrightarrow V \mid W$. It expresses the fact that V and W are indirectly related via X. It is unlikely that the intermediary X itself contains indirectly related attributes, for such a fact does not seem to be basic.

This conjecture deserves further attention and study. As we show later, a scheme that satisfies this assumption can be converted to a split free scheme, which fact supports our claim above that SFNF can be achieved in practice. However, it is not a trivial observation about reality. Let us consider the following example where this does not seem to be the case.

Example 3. Let the attributes be $B(uyer)$, $V(endor)$, $P(roduct)$ and $C(urrency)$. Assume the following MVD's:

$$BV \twoheadrightarrow P \mid C$$

$$PC \twoheadrightarrow B \mid V$$

The first MVD expresses the fact that buyers and vendors have agreements on sets of products and currencies s.t. a buyer can pay to a vendor in any currency agreed by them. The second MVD expresses the fact that nobody boycotts anyone. That is, if a buyer has an agreement about a product and a currency with some vendor and another vendor sells that product in that currency (to anybody else) then the buyer must have an agreement also with this latter vendor (and vice versa). We see that the LHS's of both MVD's are split. ■

A possible explanation is that in the example the second MVD is a restriction on the trading policy of that community of buyers and vendors and is not structural. Thus, only the first should be considered during database design. Another way to look at the situation of this example was proposed by Sciore [Sc1] who argues that very likely the designer's specifications do not match his probable intention. A plausible designer's intention might be that buyers are interested in lists of products and can pay in several currencies. Symmetrically, vendors hold lists of products they sell and accept several currencies. Therefore, the following JD must hold: $BP * PV * VC * CB$. It is this JD which is

the structural dependency that needs to be used for the design. Both of the above MVD's are implied by this JD, hence they are not basic facts and need not be used in the design process.

Summing up we postulate the following assumption:

Assumption 1 If an MVD $X \twoheadrightarrow DEP(X)$ is structural then X is not split by other structural MVD's. ■

3.2. Incorporating Functional Dependencies.

In this section we extend our model to support FD's as well. Let $(U, M \cup F)$ be a scheme where F is an FD-set and M is a set of full MVD's. The previous discussion of the role of FD's and MVD's in structuring of data shows that it is natural to use MVD's to separate clusters of attributes from each other Unlike MVD's, the role of FD's is to express close relationships between the attributes. How does the incorporation of FD's affect the clusters? Suppose that $A, V \in DEP(X)$ and $X \rightarrow A$ holds, but $X \rightarrow V$ does not. In this case we cannot rule out possible relationship between A and V. Indeed, if $X \rightarrow A$ denotes a close relationship then the indirect relationship of A and V via X may be actually rather close (as in the example about persons, children, addresses etc. of Section 2). Moreover, we cannot rule out a relationship between A and V even if, say, A transitively depends on X (again as in the example of Section 2, where *Zip* transitively depends on *Person*). From this discussion the need in a separation of the use of FD's and of MVD's in the design process becomes evident. Indeed, as we pointed out in the introduction, FD's are customarily associated with synthesis while MVD's are associated with decomposition. How to combine the methods is the subject of this section.

Let us consider another problem that has not been treated satisfactorily in the literature. It is widely acknowledged that the design process begins by finding a minimal cover of a dependency set. When only FD's are given, there is, in a sense, uniqueness of the minimal cover [Ber]. This is intuitively pleasing, since it implies the existence of a unique minimal set of basic facts. Unfortunately, MVD's do not enjoy this uniqueness property. It is interesting to note that if the set of MVD's satisfies Assumption 1 then uniqueness of a minimal cover is guaranteed [BFMY]. This is yet another hint about the significance of this assumption. The lack of minimal covers for sets of MVD's is aggravated by the presence of FD's. We show below that a proper separation of FD's and MVD's eliminates this problem, again, when the MVD's satisfy Assumption 1.

The solution we propose can be briefly described as follows. Since FD's represent close relationship, we should try to put attributes that are closely related to each other together. MVD's should be used to separate attributes that are only indirectly related. Thus, the MVD's should be used to create clusters of attributes, such that FD's are embedded in these clusters. This implies that we should use MVD-based decomposition first to create these clusters. At the second stage the FD's will be used to refine the clusters. Of course, to implement this approach, we have to show how to make sure that the FD's are indeed embedded in the clusters. This is dealt with in this section. In the next section we consider the collection of clusters and its properties.

To achieve the goal described above, we propose the following procedure [BeKi1]: For each $X \twoheadrightarrow DEP(X) \in M$ replace X by $X \rightarrow X^*_{M \cup F}$ and add the FD $X \rightarrow X^*_{M \cup F}$ to F (where $X^*_{M \cup F}$ is the closure of X w.r.t. the FD's implied by $M \cup F$). This transformation yields an equivalent scheme whose MVD-set has closed LHS's. We call such schemes *LHS–closed*. In the following we state and prove properties of such schemes. In particular, we show that, under suitable assumptions, the FD set and MVD set in such schemes are separated from each other, so each can be used in a design method appropriate to it, without fear of side effects caused by the existence of the other set.

The next proposition assures that non-splitting is preserved by the above transformation:

Proposition 1. ([BeKi1]) If the LHS's of M

are not split, then the same holds for their closures. ■

Lemma 1. Let X be $(M \cup F)$-closed and $V \in DEP(X)$. Then for no $A \in V$, $X \to A$ is implied by $M \cup F$.

Proof. Trivial. ■

So, if a structural MVD $X \twoheadrightarrow V \mid W$ holds and X is closed then for any $A_0 \in V$ and $B_0 \in W$ neither $X \to A_0$ nor $X \to B_0$ holds. The situation here is different from the situation with the MVD $P \twoheadrightarrow A \mid C$ of the example in Section 2. We could not rule out the relationship between A-values and C-values there because of the FD $P \to A$. On the other hand, the situation here is similar to that with the MVD $P \twoheadrightarrow C \mid J$ (in the same example), where C and J are not directly related.

Theorem 1. ([BeKi1]) Let $(U, M \cup F)$ be LHS-closed and $X \subseteq U$ be $(M \cup F)$-closed. Then $DEP_{M \cup F}(X) = DEP_M(X)$, i.e. to compute the dependency basis of X one uses only the MVD's of M. ■

Theorem 2. If the above assumptions hold, then the maximal cliques of $SG(M,U)$ are $(M \cup F)$-closed.

Proof. Let K be a maximal clique and $f: K \to A$ be implied by $M \cup F$. If $A \notin K$ then some MVD splits KA. Let it be $X \twoheadrightarrow V \mid W \in M$ and $V \cap K \neq \phi$, $A \in W$. Since K is not split, $K \subseteq VX$. But then $K \to A$ implies $X \to A$. Since $A \notin X$, this contradicts the closedness of X. ■

Let $X \to A$ be a structural FD. It expresses a close relationship between the attributes of XA. According to the structural role ascribed to FD's and MVD's, XA should not be split by M. By Theorem 2 this is equivalent to the nonsplitting of X. We thus argue the following assumption:

Assumption 2 If $X \to A$ is structural then X is not split. ■

If the LHS's of the FD's of F are not split by M (which, by the assumption holds if they are structural), then we achieve even stronger separation of FD's and MVD's.

Theorem 3. Let $(U, M \cup F)$ be LHS-closed and the elements of $\mathbf{LHS}(F)$ are not split by M. Then $X \to A \in (M \cup F)^+$ iff $X \to A \in F^+$.

Proof. The 'if' direction is trivial. For the 'only if' part assume w.l.o.g. that $X \to A \in (M \cup F)^+$ and $A \notin X$. The proof is by induction on the number of attributes in $U - X$. If $U - X = \{A\}$ the claim is trivial. So, suppose that for any X s.t. $\#(U - X) \leq n$, $X \to A \in (M \cup F)^+$ implies $X \to A \in F^+$.

Suppose that $\#(U - X) = n + 1$. To derive $X \to A$ we must first compute $DEP_{M \cup F}(X)$ [B]. Since the order of the applications of dependencies is immaterial, we apply M first. Thus $X \twoheadrightarrow DEP_M(X) \in M^+$. By that time only the FD's of F can be, possibly, applied to refine $DEP_M(X)$. Suppose $Y \to B \in F$ refines some $V \in DEP_M(X)$. Then $B \in V$ and $Y \cap V = \phi$ [B]. Since Y is not split by M, it is inside some maximal clique K. By Theorem 2, also $B \in K$. By [BFMY], $SG(M,U) = SG(M^+,U)$, hence YB must not be split by M^+, particularly, by $X \twoheadrightarrow DEP_M(X)$. Thus $YB \subseteq XV$. Since we have assumed that $Y \cap V = \phi$, it follows that $Y \subseteq X$. If $B = A$ then we are done. Otherwise, consider $X' = XB$. Clearly $X' \to A \in (M \cup F)^+$ and $\#(U - X') = n$. By the inductive assumption $X' \to A \in F^+$. It remains to note that, since $Y \subseteq X \subseteq XB = X'$, the pair of FD's $Y \to B$ and $X' \to A$ (that both are in F^+) implies $X \to A$. ■

Let F_0 be such that $(M \cup F_0)^+ = (M \cup F)^+$. We call F_0 an *M-cover* of F.

Theorem 4. Let $(U, M \cup F)$ be LHS-closed. An M-cover F_0 (of F) is embedded into the maximal cliques of $SG(M,U)$ iff the elements of $LHS(F_0)$ are not split (by M).

Proof. The 'only if' part is trivial. For the 'if' part let $X \to A \in F_0$. Since X is not split, it is inside some maximal clique K. By Theorem 2 maximal cliques are closed, hence $A \in K$. ■

In summary, assume that the LHS's of all given dependencies are not split. We have shown that the MVD's and FD's can be separated so that FD's are essentially

useless for deriving new MVD's, and vice versa. The LHS's of the new MVD's also are not split. Hence, there exists a *unique* minimal cover M_0 consisting of full MVD's [BFMY]. By Theorem 4, F is an embedded M_0-cover. We propose to view the dependency set $M_0 \cup F$ as a natural candidate cover to start with. Thus, the problem of the nonuniqueness of minimal covers for MVD's and FD's is reduced to the the similar problem for the FD's only.

After this preliminary step there still remains the problem of possible inadequacy of the decomposition into the maximal cliques of $SG(M_0, U)$. This issue is addressed in the next section.

3.3. How to Cope with Cyclic Schemes.

From now on, we posit assumptions 1 and 2. Thus, we remove from the given scheme the dependencies whose LHS's are split, on the grounds that these are not structural dependencies. This still does not guarantee that decomposition will produce the maximal cliques of the splitting graph. Indeed, we have seen that nonsplitting is but one of two properties needed for that guarantee. We present here results to the effect that it is possible to extend the given scheme, by adding new attributes and dependencies, so that the extended scheme satisfies both properties. Extension is used here in a strong sense - the projection of the new scheme on the attributes of the old scheme is precisely the old scheme. Thus, intuitively, we can interpret this as meaning that some of the relevant specifications may be missing from the old scheme. However, fortunately the scheme contains implicit information about what is missing, so we can make it explicit automatically. We also refer to the extension as *scheme correction*.

Let $(U, M \cup F)$ be LHS-closed and let M contain only structural MVD's. By Assumption 1, M fails to be equivalent to a single JD only if it has intersection anomalies. However, there exists a transformation [BeKi1] which for any such scheme yields an *extension* $(\bar{U}, \bar{M} \cup \bar{F})$ s.t. $\bar{M}$ is conflict-free. (A scheme $(\bar{U}, \bar{D})$ is an *extension* of (U, D) if $U \subseteq \bar{U}$, $\bar{D}^+[U] = D^+$ and $SAT((\bar{U}, \bar{D}))[U] = SAT((U, D))$, where $SAT((U,D))$ is the set of all relation instances over U satisfying D.)

We denote this transformation by Φ: $(\bar{U}, \bar{M} \cup \bar{F}) = \Phi((U, M \cup F))$. In a sense it is the most natural transformation among the others of that type [Ki]. The details of this transformation are rather complicated and we do not present them here. The interested reader is referred to [BeKi1]. Here we shall use only the following remarkable property of Φ from [BeKi1]: for any $X \subseteq U$, X is split by M iff it is split by $\bar{M}$. Thus $SG(M,U) = SG(M,\bar{U}) \cap U$ and Φ preserves both close relationships among the attributes and their absence.

Since $\bar{M}$ is conflict-free, it is equivalent to a single JD $\bar{J}$ whose constituents are the maximal cliques of $SG(M,U)$ (and therefore are in the split-free normal form). Thus, SFNF is achievable.

The only problem may be with FD's. If no $\bar{M}$-cover for $\bar{F}$ is embedded into the maximal cliques of $SG(\bar{M},\bar{U})$ then the FD's impose inter-clique constraints. Luckily, in scheme design we deal only with structural FD's. Hence, by Assumption 2 and Theorem 4, there exists an embedded M-cover for F. In this case also $\bar{F}$ has an embedded $\bar{M}$-cover [BeKi1].

Let $(U, M \cup F)$ be an LHS-closed scheme all whose dependencies are structural. Since Φ preserves both splittings and nonsplittings, the left-hand sides of the old dependencies remain nonsplit [BeKi1]. The dependencies added by Φ also have nonsplit left-hand sides.

Theorem 5. ([BeKi1]) Let the elements of $LHS(M \cup F)$ be not split and $(\bar{U}, \bar{M} \cup \bar{F}) = \Phi((U, M \cup F))$. Then the elements of $LHS(\bar{M} \cup \bar{F})$ are not split. (Particularly, $\bar{F}$ is embedded into the maximal cliques of $SG(\bar{M}, \bar{U})$.) ■

3.4. The Design Algorithm.

Theorems 4 and 5 ensure that the following design procedure works well: First find an embedded cover for FD's. Then extend the scheme to an acyclic one. Decompose the scheme using only MVD's (or, equivalently, using the single JD). At the last stage locally refine the resulting

schemes using, say, the synthesis algorithm of Bernstein.

Note that we still have to ensure that if V and W are two clusters obtained at the decomposition phase then the results of the synthesis inside V and W agree on $V \cap W$. To illustrate this point consider the following example:

Example 4. Consider the following dependencies:

$$m : A\,B\,C\,D\,E\,F \twoheadrightarrow K \mid L$$
$$f_1 : K \rightarrow A$$
$$f_2 : L \rightarrow A$$
$$f_3 : C\,A \longleftrightarrow A\,F$$
$$f_4 : B\,A \longleftrightarrow A\,E$$
$$f_5 : A\,E\,F \rightarrow D$$
$$f_6 : A\,B\,C \rightarrow D$$

Note that the FD-sets $F_1 = \{f_3, f_4, f_5\}$ and $F_2 = \{f_3, f_4, f_6\}$ are equivalent.

Decomposition by m yields the following two clusters: $A\,B\,C\,D\,E\,F\,K$ and $A\,B\,C\,D\,E\,F\,L$. If we use $G_1 = F_1 \cup \{f_1\}$ in Bernstein's synthesis process for the first cluster and $G_2 = F_2 \cup \{f_2\}$ for the second one then their common part $A\,B\,C\,D\,E\,F$ will be represented by different sets of schemes. ■

This problem is readily rectified if we find some global embedded M-cover F first and then synthesize inside each cluster using only the FD's of F embedded into that cluster.

Before presenting our design algorithm we have to decide on a policy in the case when (nonstructural) integrity constraints and structural dependencies are intermingled in the original specifications (i.e., when a split-key anomaly is detected). In this case we at least can detect and point out the problems with the given scheme specifications. Depending upon a pursued policy, the design process either stops or proceeds trying to eliminate the dependencies with split LHS's. Regardless of the policy chosen the designer must be informed by the algorithm about the problems with his specifications.

The Design Algorithm.

Input $(U, M \cup F)$

Preliminary Stage:

1. Make the MVD's full and close their LHS's.

The Nonconventional Stage:

1. Filter out nonstructural dependencies This stage results in a scheme $(U', M' \cup F')$, where the elements of $LHS(M' \cup F')$ are not split by M'.
2. Apply Φ: $(U, \bar{M} \cup \bar{F}) = \Phi((U', M' \cup F'))$. Let $\bar{J}$ be the JD equivalent to $\bar{M}$.

The Conventional Stage:

1. Find an embedded $\bar{M}$-cover.
2. Decompose according to $\bar{M}$ (or $\bar{J}$).
3. For each constituent of $\bar{J}$ synthesize (using FD's only) inside each constituent of $\bar{J}$ apart. (By [BDB], this operation losslessly decomposes each component of $\bar{J}$). Let $\mathbf{R}$ be the resulting set of schemes and $\tilde{F}$ be the (embedded into $\mathbf{R}$) FD-set found at the current stage.

Output $(\mathbf{R}, \tilde{F})$.

Theorem 6. Let $\tilde{D} = \tilde{J} \cup \tilde{F}$, where $\tilde{J}$ is a JD whose components are the schemes of $\mathbf{R}$. Then $\tilde{D}$ is equivalent to $\bar{M} \cup \bar{F}$.

Proof. As we noted earlier $\bar{M} \cup \bar{F}$ is equivalent to $\bar{J} \cup \bar{F}$. Since the synthesis algorithm of [B] preserves FD's, $\tilde{F}^+ = \bar{F}^+$. Clearly $\tilde{J}$ implies $\bar{J}$ [BMSU]. Thus $\tilde{D}$ implies $\bar{M} \cup \bar{F}$. The claim will follow if we prove that $\bar{J} \cup \bar{F}$ implies $\tilde{J}$. This follows from the fact that the synthesis algorithm losslessly decomposes each constituent of $\bar{J}$ [BDB]. ■

Unfortunately, Bernstein's synthesis achieves only 3NF schemes. It is highly desirable to achieve *scheme independence* [Sa] inside each constituent of $\bar{J}$. Then also schemes in $\mathbf{R}$ will be independent w.r.t $\tilde{J} \cup \tilde{F}$ [KiSa].

Unfortunately it is not feasible to achieve independence within the classical setting. We believe that the solution can be obtained by means of scheme correction methodology ([Sc2, BeKi1, Ki]).

3.5. Comparison to Other Methods.

The only algorithm we know about that reasonably tries to benefit from both FD's and MVD's is due to Zaniolo and Melkanoff [ZM]. However they act within the classical setting which explains why their algorithm fails to design even simple schemes like the one presented in Example 6 (below). Besides, their algorithm is conceptually complicated. We demonstrate our method on a scheme of Example 7 (below) and achieve the same result by far a simpler (both conceptually and computationally) method than it is done in [ZM].

Example 6. ([ZM]) Consider the following dictionary database. The attributes are *F(rench), G(erman) and E(nglish).* The MVD's $E \twoheadrightarrow F \mid G$, $F \twoheadrightarrow E \mid G$ and $G \twoheadrightarrow E \mid F$ denote the facts that any term in any language has (possibly more than one) translations into any other language.

The execution of our algorithm skips over the preliminary stage and the first part of the nonconventional stage. It then extends the original scheme by adding a new attribute C (for *C*oncept). The resulting scheme [BeKi1] has the universe $U' = F\,G\,E\,C$, and the MVD $C \twoheadrightarrow F \mid G \mid E$. The FD's are $F \to C$, $G \to C$ and $E \to C$. At the conventional stage we obtain the following scheme: ($\underline{F}\ C$), ($\underline{G}\ C$) and ($\underline{E}\ C$) (the keys are underlined).

Note that Zaniolo and Melkanoff obtain the same result using intuitive arguments. Their algorithm, however, fails to design this scheme. ■

Example 7. ([ZM]) Let we deal with the following enterprise:

LIC - licence number of motor vehicles
MAKE - manufacturers of motor vehicles
MODEL - models of vehicles
YEAR - year in which the vehicle was manufactured
VALUE - current value of vehicle
OWNER - unique identifier of a person
DRVL - driving licence number
VIOL - code numbers for traffic violations
DATE - date of violation

The dependencies are as follows:

LIC → *MAKE YEAR MODEL VALUE OWNER DRVL*
LIC ↠ { *VIOL DATE* }
{ *MAKE YEAR MODEL* } → *VALUE*
{ *MAKE YEAR MODEL* } ↠ { *LIC OWNER DRVL VIOL DATE* }
OWNER → *DRVL*
DRVL → *OWNER*
OWNER ↠ { *LIC MAKE YEAR MODEL VALUE* } | | { *VIOL DATE* }
DRVL ↠ { *LIC MAKE YEAR MODEL VALUE* } | | { *VIOL DATE* }

Our algorithm closes LHS's and finds a minimal cover at the preliminary stage yielding the following dependencies.

{ *DRVL OWNER* } ↠ { *LIC MAKE YEAR MODEL VALUE* } | { *VIOL DATE* }
OWNER → *DRVL*
DRVL → *OWNER*
LIC → *MAKE YEAR MODEL VALUE OWNER*
{ *MAKE YEAR MODEL* } → *VALUE*

We arrive at a scheme with a single MVD and a collection of FD's with nonsplit LHS's. Hence there is no need in the nonconventional stage. Decomposition now yields two big clusters: { *VIOL DATE DRVL OWNER* } and { *DRVL OWNER LIC MAKE YEAR MODEL VALUE* }. Synthesizing inside each component apart we obtain the following final scheme:

(*DRVL OWNER*)
(*OWNER VIOL DATE*)
(*LIC MAKE YEAR MODEL OWNER*)
(*MAKE YEAR MODEL VALUE*)

This result is identical to that of [ZM] but is obtained in far a simpler way. ■

Observe that in the example the second and the forth MVD's are implied by the first and the third FD's respectively. However, the method of [ZM] requires these implied MVD's to be explicitly specified which makes it difficult to catch the essential part of the scheme specifications. In comparison, from the point of view of our method the essential specifications of the above scheme are extremely simple: there are two SFNF-clusters with embedded FD's.

Note also that the nonconventional stage in the example is empty. This is the reason why the conventional approach of [ZM] succeeds here.

4. SUMMARY AND CONCLUSIONS.

Our approach reveals some interesting properties of the structure of real world

databases. It shows that there is a macro-structure delineated by the global JD (whose components are the SFNF-clusters obtained at the decomposition stage), and an infra-structure obtained at the synthesis stage by refining the macro-structure. This throws a new light upon the celebrated question whether the real world is cyclic or acyclic: in case when specifications are given by MVD's and FD's the world has an acyclic macro-structure (a JD equivalent to a set of MVD's is acyclic [FMU]) but the infra-structure is possibly cyclic. In general schemes, however even the macro-structure may be cyclic.

The two-level structure of the real-world databases can be exploited in the query evaluation process. Namely, a query can be split into subqueries posed about local databases obtained by restricting the database to the components of the JD. Each such subquery is evaluated over the corresponding local database independently of other subqueries (and other parts of the database). The local results are then joined to obtain the answer to the global query. This issue will be discussed elsewhere [KiSa].

In summary, we presented a consistent approach to the problem of scheme design, based on new principles. It extends the known approaches and leads to a successful design for a wide variety of situations.

Unfortunately, we have no such solution for other types of dependencies. The most important ones are embedded join dependencies and inclusion dependencies. Another problem, already mentioned, is that Bernstein's synthesis algorithm provides only partial solution to the design problem for FD's. Its drawback is that the resulting schemes are dependent [Sa]. Therefore, there are interrelational constraints which it is hard to enforce. It is known that no synthesis algorithm can design independent databases. A plausible solution is to find a suitable scheme correction [Ki, Sc2]. This issue needs further study.

5. REFERENCES.

[ASU] Aho,A.V., Y.Sagiv and J.D.Ullman, Equivalence of relational expressions *SIAM J. Computing 8:2 (1979)*, pp. 218-246

[B] Beeri,C., ' On the Membership Problem for Functional and Multivalued Dependencies in Relational Databases ', *ACM Trans. on Database Systems. vol 5, no. 3 (Sept 1980), 241-259.*

[Ber] P.A. Bernstein, ' Synthesizing Third Normal Form Relations from Functional Dependencies ', *ACM Trans. on Database Systems. 4 (Dec. 1976), 247-298.*

[BDB] Biskup,J., U.Dayal and P.A.Bernstein, ' Synthesizing Independent Database Schemes ', *Proc. ACM-SIGMOD Int. Conf. on Management of Data*, 1979, pp. 143-151.

[BFMY] Beeri,C., Fagin,R., Maier,D., Yannakakis,M., ' On the Desirability of Acyclic Database Schemes ', *J. of ACM*, July 1983.

[BeKi1] Beeri,C. and Kifer,M., ' Elimination of Intersection Anomalies from Databases ', *ACM PODS, 1983.*

[BeKi2] Beeri,C. and Kifer,M., ' Uniqueness of Elimination of Intersection Anomalies from Database Schemes ', *to be submitted.*

[BMSU] Beeri,C., A.O. Mendelson, Y.Sagiv and J.D.Ullman, ' Equivalence of Relational Database Schemes ', SIAM J. of Computing 10:2, May 1981, pp 352-370.

[Fa] Fagin,R., ' Multivalued Dependencies and a New Normal Form for Relational Databases', *Trans. on Database Syst. 2:3, pp. 262-278, 1977.*

[FMU] Fagin, R., Mendelson, A.O., and Ullman, J.D., 'A Simplified Universal Assumption and its Properties', *ACM TODS, Sept 1982.*

[Ki] Kifer,M., Ph. D. Thesis, The Hebrew University, June 1984.

[KiSa] Kifer M., Sagiv,Y., 'Computing Windows on Real-World Databases', *in preparation.*

[Li] Lien,Y.,E., ' On the Equivalence of Database Models ', *J. of ACM vol.*

29, no. 2, 1982 (April), pp. 333-362.

[Sa] Sagiv,Y., ' A Characterization of Globally Consistent Databases and their Correct Access Paths ', *ACM TODS*, June 1983.

[Sc1] Sciore,E., ' Real-World MVD's ', *Technical Report #80/ 014 , Dept. of Comp. Science, SUNY at Stony Brook, Nov 1980.*

[Sc2] Sciore,E., ' Improving Database Schemes by Adding Attributes', *ACM PODS*, 1983, pp. 379-383.

[Sc3] Sciore,E., ' The Universal Instance and Database Design ' , *Ph.D Thesis*, TR# 271, Princeton University, June 1980.

[ZM] Zaniolo,C., and Melkanoff,M.A., ' On the Design of Relational Database Schemata ', *ACM TODS* 6:1, March 1981, pp. 1-47.

[Ull] Ullman,J.D., ' Principles of Database Systems', *Computer Science Press, Potomac MD, 1980.*