An Integrated Approach to Logical Design of Relational Database Schemes

CATRIEL BEERI
The Hebrew University of Jerusalem
and
MICHAEL KIFER
SUNY at Stony Brook

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

(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, that is, constraints that do not bear any structural information about the data and which should therefore be discarded at the design stage. This separation is supported by a simple syntactic test filtering out nonstructural dependencies.
(c) Automatic correction of schemes which lack certain desirable properties.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design—*normal forms, schema and subschema*

General Terms: Design, Theory

Additional Key Words and Phrases: Acyclic schemes, conflict-free sets of dependencies, decomposition, functional dependencies, logical design, multivalued dependencies, schema extension, synthesis

1. INTRODUCTION

We use the general framework and the concepts as developed by the classic design theory during the last decade [4, 11, 15, 16, 19, 22, 25, 27]. 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 contains all the attributes. Among the properties considered in the literature are those of *preservation of dependencies*; *normal forms* such as 3NF, BCNF, and 4NF [26];

A preliminary report on this research appears in the *Proceedings of the 10th Conference on Very Large Data Bases* (Singapore, Aug. 1984), 196–207.
Most of this work was done while M. Kifer was at the Hebrew University of Jerusalem.
Authors' addresses: C. Beeri, Institute of Mathematics and Computer Science, the Hebrew University of Jerusalem, Israel; M. Kifer, Department of Computer Science, SUNY at Stony Brook, Stony Brook, LI, NY 11794.

schema independence [22]; *acyclicity* [16], and so on. Unfortunately, of this variety of goals 3NF is the only goal always achievable within the classic framework [11, 15, 23, 24].

The common characteristic of all the classical approaches to database design is that the initially given attributes and dependencies are left intact during the design process. Further, all the specified dependencies are assumed to have equal rights to guide the design procedure. Such an approach suffers from several drawbacks. The first concerns the method for deriving the database scheme. For that purpose two different methods have been proposed. One of them is scheme *synthesis* [11]; the other, known as scheme *decomposition*, was proposed by Codd and generalized by Fagin [15]. Scheme synthesis works well in the context of functional dependencies (FDs), while scheme decomposition is suited for multivalued dependencies (MVDs). So far, the two approaches are considered to be incompatible.

The success of a design theory depends on its ability to supply satisfactory answers to the situations that inherently need different types of dependencies. Different researchers tried to cope with the case where MVDs and FDs are brought together. Lien [19] adheres to the decomposition approach, considering the FDs as MVDs, and neglects the different semantics of FDs and MVDs. Melkanoff and Zaniolo [27] also tried to cope with the problem, giving first priority to the notion of faithful representation of dependencies. However, both methods are unsatisfactory. First, they fail when the MVD-set is not *conflict-free* [19]. Second, 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.

In this paper we restrict the class of data dependencies to FDs and MVDs. It is widely understood that this is far from being enough to specify a reasonably sophisticated database. Nevertheless, the design problem turned out to be difficult even for such a restricted class of dependencies. In the past ten years this problem has been attacked many times [4, 15, 19, 27], yet no solution agreeable to a majority of researchers has been found. We believe that the method described in this paper provides such a solution. Our approach takes into consideration the different roles of FDs and MVDs, and works well if both types are given. It incorporates both the synthesis and decomposition approaches and employs ideas about scheme corrections [7, 17].

Although we consider only a restricted class of dependencies, our belief is that the general ideas of the approach are pertinent to the design problem as a whole, and can serve as a framework for future research in this area.

The paper is organized as follows. In Section 2 we introduce some basic notions and notation used in the paper. In Section 3 we discuss the semantic role of dependencies in the design process. Section 4 presents the ideas behind our approach and the design algorithm. In Section 5 we compare our method with other methods and give the examples. Section 6 concludes the paper.

2. PRELIMINARIES

We assume that the reader is familiar with the theory of FDs, MVDs, and join dependencies (JDs), and with the notions of third (3NF) and fourth (4NF) normal forms on the level of [26]. We also assume that the reader is familiar with acyclic database schemes [6, 16].

For the MVD-set $\{X \twoheadrightarrow V_i \mid i = 1, \ldots, n\}$ we shall often write $X \twoheadrightarrow S$, where $S = \{V_i \mid i = 1, \ldots, n\}$. Alternatively, we write $X \twoheadrightarrow V_i \mid \cdots \mid V_n$ instead of $X \twoheadrightarrow S$. As usual, $DEP_D(X)$ denotes the dependency basis of X with respect to (abbr. w.r.t.) the dependency set D, and X_D^* denotes the closure of X w.r.t. the FDs implied by D. We omit the dependency set if it is obvious or immaterial. For future reference we bring Beeri's algorithm [2] for computing $DEP_D(X)$ and X_D^* when $D = M \cup F$ consists of MVDs of M and of FDs of F.

$DEP_{M \cup F}(X)$ is computed by refining P, a *candidate partition for* $DEP(X)$, as follows: Start with $P = \{U - X\}$. Then apply the following rule until no more changes to P are possible: If Y is in the partition, $S \twoheadrightarrow R$ is in $M \cup F$ (any FD $S \rightarrow A \in F$ is treated as an MVD $S \twoheadrightarrow A$), and $S \cap Y = \emptyset$, then replace Y by $Y \cap R$ and $Y - R$. If P is known to be a candidate partition and $S \twoheadrightarrow R$ changes P in this way, then we say that $S \twoheadrightarrow R$ *refines* P. The algorithm terminates when P cannot be refined any more. This final partition is $DEP_{M \cup F}(X)$, [2]. To get $X_{M \cup F}^*$, first compute $DEP_{M \cup F}(X)$ and then pick up those singleton elements $\{A\} \in DEP_{M \cup F}(X)$ such that (s.t.) $S \rightarrow A$ is in F. $X_{M \cup F}^*$ consists of X and all such singletons [2].

An MVD of the form $X \twoheadrightarrow DEP(X)$ is called *full* [6], and a set of MVDs containing only full MVDs is called a *set of full MVDs*. If D is a set of FDs or MVDs, we denote by LHS(D) the set of left-hand sides (LHS) of the members of D. Without loss of generality (w.l.o.g.) we assume throughout the paper that all the FDs are of the form $X \rightarrow A$, where A is an attribute.

If I (resp. t) is a relation (resp. tuple) over a set of attributes U and $X \subseteq U$, then $I[X]$ (resp. $t[X]$) will denote the projection (resp. restriction) of I (resp. t) on X. Finally, we note that it is the usual practice in database theory to write XY for $X \cup Y$, where X and Y are sets of attributes.

3. THE ROLE OF DEPENDENCIES IN DATABASE DESIGN

Consider the MVD $X \twoheadrightarrow V \mid W$. Intuitively, it denotes the fact that there is *no close 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 that persons and the projects they work on are related. It also states, though, that there is no relationship between a child and a project except for the *indirect connection* via a person that may be the father of the child and is currently working for the project.

This explains the drawback to the approaches that consider FDs as just MVDs.

In fact, an FD is *not* a "pure" MVD, for it is not intended to represent the second 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(hild)*, *(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 zip-codes are only indirectly related. However, it is obviously not the case here. On the other hand, we can still be justified in inferring that children are not directly related to the projects.

Another problem in scheme design is that the dependencies may represent not the presence or absence of relationships between attributes, but rather constraints which have little influence on the way the data should be structured. This distinction is due to [16], from which 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 FDs $C \rightarrow T$, $TH \rightarrow R$, $HS \rightarrow R$, $HR \rightarrow C$, and $CS \rightarrow G$ and MVD $CT \twoheadrightarrow HR \mid GS$. This MVD expresses the fact that a course may meet more than once a week and that any student attends all the meetings of the course he takes. Here $HS \rightarrow R$ expresses the physical fact that students cannot be in two places at once. From this and $HR \rightarrow C$ one can derive $HS \rightarrow C$, which means that students' time tables must be feasible, that is, no student is permitted to take courses whose meeting times clash. However, these restrictions on what students are 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 the fact 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.

Clearly, only (ii) and (iii) are significant in scheme design. We have seen that FDs always express (i) and sometimes (ii). Hence, an FD that does not express (ii) must be disregarded in the design process. Similarly, MVDs always express (i) and sometimes (iii) and (ii). Our contention is that the major semantic intention of MVDs is to express (iii). That is, in our view, the intention of an MVD $X \twoheadrightarrow V \mid W$ (used as a means of data structuring) should be a representation of the fact that there is no close relationship between attributes of V and W. However, there still may be some connection between X and V and between X and W. In this case, V and W would be indirectly related through X.

The discussion above shows that syntactic specifications such as data dependencies are not quite suitable as a description of structural information about 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 relationships among the attributes, separation of attributes (i.e., indirect relationship). 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 is implied by their role as integrity constraints. Apparently, this is not so.

The following observation supports this claim. Data dependencies, in their role as integrity constraints, are monotonic in the sense that the 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 longer monotonic. For instance, the MVD $C \twoheadrightarrow A \mid B$ specifies a separation of A from B. However, when the FD $A \rightarrow B$ is added, it specifies the fact that A and B are closely related, which changes the structure suggested by the MVD.

In summary, some of the 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 during the database lifetime, but are irrelevant at the design stage.

To avoid possible confusion we emphasize that in this paper our concern is the *logical database design*. There is a related activity, called *physical database design*, which deals with the indexing structure, clustering of records and parts thereof on disk pages, and so on. Dependencies ignored at the logical design level appear as input specifications to the physical design stage. Thus it is the latter stage where integrity constraints should be taken care of in order to facilitate their checking.

In talking about the structural role of data dependencies we should keep in mind that some of the specified dependencies may be implied by others. As soon as one learns that a dependency d is an implied one, the feeling is that it is not as relevant to the design as the dependencies from which d was derived. There is a wide consensus in the database community that, for the logical design, one only needs dependencies from some minimal cover. Unfortunately, different covers may exist for the same dependency set. All classical methods are dependent on a choice of a particular minimal cover, which is another drawback, since different covers give rise to different designs. In the next section we provide a partial solution to this problem.

4. HOW TO DESIGN

4.1 The Decompositional Approach and the Splitting Graph

We start with analysis of the decomposition approach, assuming that only MVDs are given. The basis of this approach is the *separation principle* of [4], and its declared goal is 4NF. We shall see that attaining 4NF does not solve the problems it is supposed to solve. After showing the drawbacks of 4NF we argue in favor of a better design goal based on the concept of *splitting*, introduced in [6].

Let (U, M) be a universal scheme with MVD-set M. From now on assume that M is a set of *full* MVDs. An MVD $X \twoheadrightarrow DEP(X)$ *splits* a set of attributes

$Y \subseteq U$ if for some distinct $V, W \in DEP(X)$ both $V \cap Y \neq \emptyset$ and $W \cap Y \neq \emptyset$ hold. Y 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 the set XVW should not be in one relational scheme, because this introduces redundancies and potential consistency problems [15]. 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, MVDs 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 not be used in the decomposition process [19]. In such a case $X \twoheadrightarrow V \mid W$ imposes an interrelational constraint which may cause consistency problems.

Another problem with the decomposition approach is that it may yield schemes whose attribute sets are split. Assuming M consists only of structural MVDs (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 [4]. 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. ([23]) 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 MVDs: $m_1: A \twoheadrightarrow T \mid L$ and $m_2: T \twoheadrightarrow A \mid L$. The first MVD states that authors do not discriminate among locations, and stick to all of them (i.e. any author publishes all his papers in all the locations he uses to publish papers). The second MVD simply says that if a paper is written by more than one author, then the authors publish that paper in the same locations.

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. If we had more than one A-value in the original relation, then, for each A-value in the ATL-relation, the corresponding fragment of the TL-relation would be a Cartesian product. Therefore, we have the same sort of redundancy that was supposed to be prevented by 4NF. Of course, the decomposition $\{AT, AL\}$ has the same drawback.

The assumption that 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* (SFNF) if no R_i is split by M.

It is shown in [6] that this property does not depend on the choice of cover for M, that is, if $M_0^+ = M^+$, then $\mathbf{R}$ is a SFNF scheme over (U, M) iff it is a SFNF scheme over (U, M_0). From that observation it now easily follows that SFNF implies 4NF. Indeed, if $\mathbf{R}$ is in the SFNF then no MVD implied by M can decompose any of R_i, or else R_i is split by M.

To obtain a SFNF database scheme we propose to use the splitting graph of [6] 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 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 it 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* [6]. If this is the case, then this JD is *acyclic* [6, 16], 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 shown that it can be achieved when the given universal scheme has a conflict-free set of MVDs. 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-lhs* anomaly [19] if at least one $X \in \mathbf{LHS}(M)$ is split by M. Otherwise, M is called a *split-free* set of (full) MVDs. M has an *intersection*

anomaly [7] if there are X, $Y \in \mathbf{LHS}(M)$ s.t. $(X \cap Y) \twoheadrightarrow (DEP(X) \cap DEP(Y)) \notin M^+$ (DEP (. . .)) is a set of subsets of U, so the intersection on the right-hand side is the set of common elements of $DEP(X)$ and $DEP(Y)$.

It turns out [19] that M is equivalent to a single JD iff it has neither split-LHS nor intersection anomalies. After Lien [19], we call such schemes *conflict-free.*

In Section 3 we argued that only structural dependencies should be used in database design. The question is, however, how structural dependencies can be recognized syntactically. In the rest of this section we give a partial answer to this question (which is sufficient for our further considerations).

We conjecture that in real-world situations LHSs of structural MVDs are not split (i.e., the split-LHS anomaly does not occur). A similar assumption was advanced in [23]. Of course, we refer only to the MVDs 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 conflict-free scheme, which supports our claim 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 MVDs:

$$BV \twoheadrightarrow P \mid C$$
$$PC \twoheadrightarrow B \mid V$$

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 vendor in any currency agreeable to themselves. The second MVD says that no one 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 with the latter vendor also (and vice versa). We see that the LHSs of both MVDs are split.

One 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. On the other hand, the first dependency seems more basic, as it represents such fundamental things as contracts between buyers and vendors on products and currencies. Thus, only the first MVD should be considered during database design. To a certain extent the choice to treat the second dependency as a constraint is mostly a matter of taste. However, our point is not the particular choice we made, but the fact that splitting of LHSs indicates that the given schema specifications are semantically problematic.

Another way to look at the situation of this example was proposed by Sciore [23], who argues that very likely the designer's specifications did not match his probable intention. A designer's plausible intention might be that buyers are

interested in certain 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 that is the structural dependency that needs to be used for the design. Both of the above MVDs are implied by this JD, hence they are not basic facts and need not be used in the design process. Thus, assuming the second explanation, FDs and MVDs do not constitute an adequate set of schema specifications for that particular database, which puts this scheme beyond the scope of our method.

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 MVDs.

This assumption is not a syntactic definition of structuralness. A rough approximation of that concept should involve the notion of objects [16, 25], and is beyond the scope of this paper. One cannot even use this assumption to find out whether a dependency is structural or not, since LHSs of such dependencies may be split by nonstructural ones. Nevertheless, this assumption can be used to detect a semantic problem with the dependency set as a whole. The correct set of structural dependencies can then be extracted using heuristics and a designer's assistance. Detailed treatment of these questions is beyond the scope of this paper.

4.2 Incorporating Functional Dependencies

In this section we extend our model to support FDs as well. Let $(U, M \cup F)$ be a scheme where F is an FD-set and M is a set of full MVDs. The previous discussion of the role of FDs and MVDs in the structuring of data shows that it is natural to use MVDs to separate clusters of attributes from each other. Unlike MVDs, the role of FDs is to express close relationships between the attributes. How does the incorporation of FDs affect the clusters? Suppose that $A, V \in DEP(X)$ and $X \rightarrow A$ hold, but $X \rightarrow V$ does not. In this case we cannot rule out a 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 3). 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 3, where *Zip* transitively depends on *Person*). From this discussion, the need for a separation of the use of FDs and MVDs in the design process becomes evident. Indeed, as we pointed out in the introduction, FDs are customarily associated with synthesis while MVDs 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 FDs are given, there is, in a sense, a unique miminal cover [11]. This is intuitively pleasing, since it implies the existence of a unique minimal set of basic facts. Unfortunately, MVDs do not enjoy this uniqueness property. It is interesting to note that if the set of MVDs satisfies Assumption 1, then uniqueness of a minimal cover is guaranteed [6].

This is yet another hint of the significance of this assumption. The lack of uniqueness of minimal covers for sets of MVDs is aggravated by the presence of FDs. We show below that a proper separation of FDs and MVDs eliminates this problem, again, when the MVDs satisfy Assumption 1.

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

To achieve the goal described above, we propose the following procedure [7]: For each $X \twoheadrightarrow DEP(X) \in M$, replace X by $X^*_{M \cup F}$ and add the FD $X \rightarrow X^*_{M \cup F}$ to F. The resulting MVD-set will thus consist of all the MVDs $\bar{X} \twoheadrightarrow DEP(\bar{X})$, where $\bar{X} = X^*_{M \cup F}$, $X \in \mathbf{LHS}(M)$. This transformation yields an equivalent scheme whose MVD-set has closed LHSs. 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 and MVD sets 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 nonsplitting is preserved by the above transformation:

PROPOSITION 1. ([7]) *If the LHSs 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 $A \in V$, $X \rightarrow A$ 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 $X \rightarrow A_0$ nor $X \rightarrow B_0$ holds. The situation here is different from that of the MVD $P \twoheadrightarrow A \mid C$ of the example in Section 3. We could not rule out the relationship between A-values and C-values there because of the FD $P \rightarrow A$. On the other hand, the situation here is similar to that of the MVD $P \twoheadrightarrow C \mid J$ (in the same example), where C and J are not directly related. Thus, for the MVD $X \twoheadrightarrow V \mid W$ above, we can conjecture that no direct relationship exists between attributes of V and W.

THEOREM 1. ([7]) *Let $(U, M \cup F)$ be LHS-closed and $X \subseteq U$ be $(M \cup F)$-closed. Then, $DEP_{M \cup F}(X) = DEP_M(X)$, that is, the dependency basis of X w.r.t. $M \cup F$ can be computed using M alone.*

THEOREM 2. *Under the assumptions of Theorem 1, the maximal cliques of $SG(M, U)$ are $(M \cup F)$-closed.*

PROOF. Let K be a maximal clique and $f: K \rightarrow A$ 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 $V \cap K \neq \emptyset$, $A \in W$. Since K is not split, $K \subseteq VX$. But then $K \rightarrow A$ implies $X \rightarrow A$ by an FD–MVD derivation rule.[1] Since $A \notin X$, this contradicts the closedness of X. □

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

Assumption 2. If $X \rightarrow A$ is structural, then X is not split.

Our next theorem shows that if the LHSs of the FDs of F are not split by M (which, by the assumption, holds if they are structural), then we achieve an even stronger separation of FDs and MVDs.

THEOREM 3. *Let* $(U, M \cup F)$ *be LHS-closed and suppose the elements of* $\mathbf{LHS}(F)$ *are not split by* M. *Then,* $X \rightarrow A \in (M \cup F)^+$ *iff* $X \rightarrow A \in F^+$. *Thus MVDs are not needed to establish FDs.*

PROOF. The "if" direction is trivial. For the "only if" part assume w.l.o.g. that $X \rightarrow A \in (M \cup F)^+$ and $A \notin X$. The proof is by induction on the number of attributes in $U - X$. Let $U - X = \{A\}$. By Beeri's algorithm, since $X \twoheadrightarrow A \in (M \cup F)^+$, there should be an FD $Z \rightarrow A \in F$ s.t. $A \notin Z$. Since $X = U - \{A\}$ by assumption, it follows that $Z \subseteq X$. Hence $X \rightarrow A$ is implied by F alone. For the inductive step, suppose that for any X s.t. $\#(U - X) \leq n$, $X \rightarrow A \in (M \cup F)^+$ implies $X \rightarrow A \in F^+$.

Suppose that $\#(U - X) = n + 1$. To derive $X \rightarrow A$, we must first compute $DEP_{M \cup F}(X)$ by refining the candidate partition for $DEP_{M \cup F}(X)$ by FDs and MVDs of $M \cup F$. Since the order of applications of dependencies is immaterial, we apply M first, after which the candidate partition will become $DEP_M(X)$. At that time only the FDs of F can possibly be applied to refine the partition.[2] Suppose $Y \rightarrow B \in F$ refines some $V \in DEP_M(X)$. Then, by Beeri's algorithm, $B \in V$ and $Y \cap V = \emptyset$. Since Y is not split by M, it is inside some maximal clique K. By Theorem 2, $B \in K$ also. By [6], $SG(M, U) = SG(M^+, U)$, hence YB must not be split by M^+, particularly by $X \twoheadrightarrow DEP_M(X)$. Thus $YB \subseteq XV$. Since we have assumed that $Y \cap V = \emptyset$, it follows that $Y \subseteq X$. If $B = A$, then we are done. Otherwise, consider $X' = XB$. Clearly, $X' \rightarrow A \in (M \cup F)^+$ and $\#(U - X') = n$. By the inductive assumption, $X' \rightarrow A \in F^+$. It remains to note that, since $Y \subseteq X \subseteq XB = X'$, the pair of FDs $Y \rightarrow B$ and $X' \rightarrow A$ (both of which are in F^+) imply $X \rightarrow A \in F^+$. □

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

[1] This rule says (cf. [5, rule FD–MVD2; 21, rule C2]): from $X \twoheadrightarrow W$, $K \rightarrow A$, $A \in W$, and $K \cap W = \emptyset$ infer $X \rightarrow A$.

[2] When the first FD refines the candidate partition, the MVDs may again be activated to refine the partition even further. So we do not claim that $DEP_{M \cup F}(X)$ can be computed by first applying MVDs alone and then FDs alone.

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 $X \rightarrow A \in F_0$. 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 LHSs of all given dependencies are not split. We have shown that the MVDs and FDs can be separated so that the FDs are essentially useless for deriving new MVDs, and vice versa. This was done by closing the LHSs of the initially given MVDs. The LHSs of the new dependency set also are not split. Hence there exists a *unique* minimal cover, M_0, consisting of full MVDs [6]. 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 MVDs and FDs is reduced to the similar problem for the FDs only.

After this preliminary step we still have 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.

4.3 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 LHSs are split, on the grounds that they 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 exend 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 specifications of the new scheme on the attributes of the old scheme is precisely the set of old scheme specifications. Thus, intuitively, we can interpret this as meaning that some of the relevant specifications may be missing from the old scheme. Fortunately, however, the scheme contains implicit information about what is missing, so we can make it explicit automatically.

Let $(U, M \cup F)$ be LHS-closed and let M contain only structural MVDs. By Assumption 1, the LHSs of M are not split. Thus, by Lien's result [19], M fails to be equivalent to a single JD only if it has intersection anomalies. The problem of removing intersection anomalies was studied in [7, 8] and [17]. In the rest of this section we present an informal description of the method and its properties. The basic idea and the motivation for the method first appeared in [23].

Consider an intersection anomaly between a pair of MVDs $m_X: X \twoheadrightarrow V_1 | \cdots | V_n | X_1 | \cdots | X_m$ and $m_Y: Y \twoheadrightarrow V_1 | \cdots | V_n | Y_1 | \cdots | Y_k$. To remove it, we would like to add something like $X \cap Y \twoheadrightarrow V_1 | \cdots | V_n$ to our MVD-set. Unfortunately, a straightforward addition of this MVD will change the initial semantics of the scheme too drastically. A safer way is to assume that all the dependencies for the *given set of attributes* were specified correctly, but that some

other specifications were missing. The formal notion corresponding to this approach is called schema extension. First, we need the following definition from [1].

The projection of an FD $X \rightarrow V$ or an MVD $X \twoheadrightarrow V$ on a set of attributes S is defined to be $X \rightarrow (V \cap S)$ or $X \twoheadrightarrow (V \cap S)$ if $X \subseteq S$; it is undefined if $X \not\subseteq S$ (w.l.o.g., we can take projection in the latter case to be any trivial dependency, e.g., $S \rightarrow S$ or $S \twoheadrightarrow S$). For a dependency d, its projection on S is denoted by $d[S]$. For a set D of FDs and MVDs, the projection of D on S, denoted $D[S]$, is $\{d[S] \mid d \in D\}$.

It was proved in [1] that projection of FDs and MVDs according to the definition above is a sound and complete inference rule. That is, if I is a relation over U, then $I[S]$ satisfies all and only the FDs and MVDs that are projections of dependencies that hold in I. It follows that for a scheme $(U, M \cup F)$, an FD or MVD d is satisfied in the projection on $S \subseteq U$ of all relations over U that satisfy $M \cup F$ *iff* d is obtained by projection from a dependency in $(M \cup F)^+$.

A scheme $(\bar{U}, \bar{D})$ is called an *extension* of another scheme (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 and $SAT((\bar{U}, \bar{D}))[U]$ stands for $\{\bar{I}[U] \mid \bar{I} \in SAT((\bar{U}, \bar{D}))\}$. Thus, by extending a scheme, we do not change the original semantics since the new scheme being restricted to the old set of attributes represents exactly the same information as that represented by the old scheme. However, $(\bar{U}, \bar{D})$ extends the scope of (U, D), and may introduce new relationships between the old attributes and the added ones.

With this intention in mind and following Sciore's original proposal, we enrich the scheme with a new attribute $P_{X,Y}$ and replace the problematic MVDs above by the following three dependencies:

$$
\begin{aligned}
m'_X: &\quad XP_{X,Y} \twoheadrightarrow V_1 \mid \cdots \mid V_n \mid X_1 \mid \cdots \mid X_n,\\
m'_Y: &\quad YP_{X,Y} \twoheadrightarrow V_1 \mid \cdots \mid V_n \mid Y_1 \mid \cdots \mid Y_n,\\
m_{P_{X,Y}}: &\ X \cap YP_{X,Y} \twoheadrightarrow V_1 \mid \cdots \mid V_n.
\end{aligned}
$$

In this way the anomaly caused by m_X and m_Y will be removed. To ensure that the resulting scheme extends the original one, we add a pair of FDs, $X \rightarrow P_{X,Y}$ and $Y \rightarrow P_{X,Y}$. It is now obvious that, within the old set of attributes, all and only the old dependencies will hold, namely, m_X and m_Y. Nevertheless, several problems still exist. First, it has to be shown how $P_{X,Y}$ can be incorporated into the rest of the dependencies when m_X and m_Y are not the only dependencies specified in the scheme. Second, if there are intersection anomalies caused by other MVDs, it has to be shown that iterative use of the above basic step will eventually terminate and remove all the anomalies. The third problem concerns the semantic meaning of the new added attributes. The last issue is the uniqueness of the scheme resulting from the above process. All of these problems are resolved in [7, 8, 17].

Before we describe the basic step of the transformation, we propose one more preliminary step—the removal of redundant MVDs from M ($m \in M$ is *redundant* if $m \in (M - \{m\})^+$; M is *nonredundant* if it has no redundant MVDs). It is known that if a set of full MVDs M is split-free, then it has a *unique minimal cover* (see [7]). A scheme whose MVD-set is LHS-closed and nonredundant is

called **P**-*reduced*. We can now explain how an anomaly formed by a pair of MVDs can be removed.

Transformation *P*

Input: A ***P***-reduced scheme $(U, M \cup F)$ and a pair of LHSs $X, Y \in \mathbf{LHS}(M)$.
Output: A scheme $(U', M' \cup F')$ extending $(U, M \cup F)$ in which $X^*_{M' \cup F'}$ and $Y^*_{M' \cup F'}$ do not form intersection anomalies.

1. Construct the new universe: $U' \leftarrow U \cup \{P_{X,Y}\}$, where $P_{X,Y}$ is a new attribute.
2. Start with empty MVD-set M' for the new scheme. Add an MVD which is supposed to remove the anomaly between $XP_{X,Y}$ and $YP_{X,Y}$ to the new scheme (as suggested in the above discussion):
$$M' \leftarrow \{m_{P_{X,Y}}: P_{X,Y}(X \cap Y) \twoheadrightarrow DEP(X) \cap DEP(Y)\}.$$
3. Introduce another FD expressing the fact that $P_{X,Y}$ "represents" $X \cap Y$:
$$F' \leftarrow F \cup \{P_{X,Y} \rightarrow X \cap Y\}.$$
4. Incorporate $P_{X,Y}$ into the rest of the MVDs.
   for each full MVD $Z \twoheadrightarrow V_1 \mid \cdots \mid V_n \in M$ **do**[3]
   (i) If either $Z \supseteq X$ or $Z \supseteq Y$ or there are distinct V_i and V_j in $DEP(Z)$ s.t. $X \cap V_i \neq \emptyset$ and $Y \cap V_j \neq \emptyset$, then put
   $M' \leftarrow M' \cup \{ZP_{X,Y} \twoheadrightarrow V_1 \mid \cdots \mid V_n\}$
   $F' \leftarrow F' \cup \{Z \rightarrow P_{X,Y}\}$.
   (ii) Otherwise, there exists $V_i \in DEP(Z)$ s.t. $V_i \cap X \neq \emptyset$ and $V_i \cap Y \neq \emptyset$. Then,
   $M' \leftarrow M' \cup \{Z \twoheadrightarrow V_1 \mid \cdots \mid V_iP_{X,Y} \mid \cdots \mid V_n\}$.
   end
5. *make the MVDs in M' full,*
6. *make M' nonredundant,*
7. *output*$((U', M' \cup F'))$.

Let $(U, M \cup F)$ be a ***P***-reduced scheme and X, Y be a pair of LHSs that form an intersection anomaly. We denote the result of applying ***P*** to this pair by $\boldsymbol{P}((U, M \cup F); X, Y)$.

We see that each application of ***P*** adds a new attribute $P_{X,Y}$, a new MVD $m_{P_{X,Y}}$, and some FDs. In addition, the old MVDs are transformed into the MVDs of the resulting scheme as described in step 4 of the algorithm. We say that these transformed MVDs are *inherited* from the original scheme $(U, M \cup F)$. The new MVD, $m_{P_{X,Y}}$, may make some of the inherited MVDs redundant. Redundant MVDs are removed in step 6. The importance of ***P*** stems from the following facts proved in [7]:

(1) $\boldsymbol{P}((U, M \cup F); X, Y)$ is an extension of $(U, M \cup F)$.
(2) If $Z \in \mathbf{LHS}(M)$, then if Z is as in (i) of step 4 of the algorithm above, then $Z^*_{M' \cup F'} = ZP_{X,Y}$ and $DEP_{M' \cup F'}(ZP_{X,Y}) = DEP_{M \cup F}(Z)$; if Z is as in (ii) of step 4, then $Z^*_{M' \cup F'} = Z$ and $DEP_{M' \cup F'}(Z)$ is as described in (ii) of step 4.

From the second fact it follows that $DEP_{M' \cup F'}(XP_{X,Y}) \cap DEP_{M' \cup F'}(YP_{X,Y})$ is a subset of $DEP_{M' \cup F'}(m_{P_{X,Y}})$, hence $XP_{X,Y}$ and $YP_{X,Y}$ do not form an intersection anomaly.

[3] Note that the MVDs m'_X and m'_Y, mentioned earlier, will be added at that step (in (i)) when $Z = X$ or Y.

Example 4. Let $U = ABDFG$ and let the MVDs be

$$m_1: A \twoheadrightarrow F \mid G \mid BD$$
$$m_2: B \twoheadrightarrow F \mid G \mid AD$$
$$m_3: D \twoheadrightarrow F \mid ABG$$

For future references, denote this scheme by $\mathbf{R}_0$. Application of $\boldsymbol{P}$ to the first two MVDs introduces a new attribute P, adds the FDs $A \rightarrow P$ and $B \rightarrow P$, and transforms the original MVD-set to the following:

$$m_1': AP \twoheadrightarrow F \mid G \mid BD$$
$$m_2': BP \twoheadrightarrow F \mid G \mid AD$$
$$m_3': D \twoheadrightarrow F \mid ABGP$$
$$m_P: P \twoheadrightarrow F \mid G \mid ABD$$

Now the first pair of MVDs does not form an intersection anomaly (moreover, m_1' and m_2' are redundant, and hence will be removed in step 6). However, the last pair of MVDs still forms an anomaly.

This example shows that a single application of $\boldsymbol{P}$ may not suffice. In order to remove all the anomalies, we have to apply $\boldsymbol{P}$ as long as there remain pairs of (nonredundant) LHSs which form intersection anomalies. We denote such an iterative application of $\boldsymbol{P}$ by $\boldsymbol{P}^*$. Of course, $\boldsymbol{P}^*$ depends on the choice of particular pairs of LHSs used in each application of $\boldsymbol{P}$. We omit these particular pairs of LHSs from our denotations because they will always be either clear from the context or unimportant.

Example 4. (Continued) Further applying $\boldsymbol{P}$ to m_3' and m_P, we end up with the MVD-set:

$$m_Q: Q \twoheadrightarrow F \mid ABDGP$$
$$m_P': PQ \twoheadrightarrow F \mid G \mid ABD$$

and the FD-set $A \rightarrow P$, $B \rightarrow P$, $D \rightarrow Q$, $P \rightarrow Q$. The MVD inherited from m_3' is redundant and therefore removed. Let us denote this scheme by $\mathbf{R}_1$.

It can easily be verified that $\mathbf{R}_1$ is conflict-free and that it extends $\mathbf{R}_0$. An astute reader may have already noticed that $\boldsymbol{P}$ can be applied to the scheme of Example 4 in different ways. We show this in the next example.

Example 4. (Continued) If instead of applying $\boldsymbol{P}$ to m_1 and m_2 in $\mathbf{R}_0$, we used m_2 and m_3, the resulting scheme would have the following MVDs:

$$m_1': A \twoheadrightarrow F \mid G \mid BDQ_1$$
$$m_2': BQ_1 \twoheadrightarrow F \mid G \mid AD$$
$$m_{Q_1}: Q_1 \twoheadrightarrow F \mid ABDG$$

and the FD-set would be $B \rightarrow Q_1$, $D \rightarrow Q_1$.

The last application of $\boldsymbol{P}$ was not as efficient as before when we applied $\boldsymbol{P}$ to m_1 and m_2. We are still left with two anomalies (between m_1' and m_2' and between m_1' and m_{Q_1}), compared to only one anomaly in Example 4. Thus, not all possible ways of applying $\boldsymbol{P}$ to a scheme advance us equally well towards the goal of

removing all intersection anomalies. In fact, applying $\boldsymbol{P}$ to certain schemes may cause even more anomalies than it had originally [7]. All this gives rise to a question about the termination of the process affecting $\boldsymbol{P}^*$. It is proved in [7] that

> $\boldsymbol{P}^*$ terminates regardless of the choice of pairs of LHSs used for the intermediate $\boldsymbol{P}$-transformations. Furthermore, for any scheme R (with nonsplit LHSs), $\boldsymbol{P}^*(R)$ is a conflict-free extension of R.

Example 4. (Continued) Now let $\boldsymbol{P}$ be applied to m_1' and m_2' in the last example. This results in introduction of another new attribute P' and of the FDs $A \rightarrow P'$, $BQ_1 \rightarrow P'$. The MVD-set now looks as follows:

$$m_{P'}\colon P' \twoheadrightarrow F \mid G \mid ABDQ_1$$
$$m_{Q_1}\colon Q_1 \twoheadrightarrow F \mid ABDGP'$$

The remaining intersection anomaly can be removed by successive applications of $\boldsymbol{P}$. We then end up with a pair of MVDs:

$$m_{P'}'\colon P'Q' \twoheadrightarrow F \mid G \mid ABDQ_1$$
$$m_{Q'}\colon Q' \twoheadrightarrow F \mid ABDGP'Q_1$$

and with FDs $A \rightarrow P'$, $BQ_1 \rightarrow P'$, $B \rightarrow Q_1$, $D \rightarrow Q_1$, $P' \rightarrow Q'$, and $Q_1 \rightarrow Q'$. Note that m_{Q_1} is now implied by $m_{Q'}$, and therefore removed from the MVD-set. For future reference, we denote the resulting scheme by $\mathbf{R}_2$.

We demonstrated that different orders of applications of $\boldsymbol{P}$ may result in different schemes. Thus the scheme $\mathbf{R}_2$ of the last example has one more attribute then the scheme $\mathbf{R}_1$ obtained earlier in the example. The FDs are also slightly different.

Though such a nonuniqueness is an unpleasant property, we can show that schemes obtained from the same original scheme by different series of $\boldsymbol{P}$-transformations can be naturally reduced to a single *canonical* scheme. This canonical scheme is also a conflict-free extension of the initial scheme. To see how it can be done, let us have another look at schemes $\mathbf{R}_1$ and $\mathbf{R}_2$ obtained above.

Observe that both schemes have the same number of (nonredundant) MVDs. If we rename Q' to Q and P' to P in $\mathbf{R}_2$, then the MVDs of $\mathbf{R}_2$ will differ from those of $\mathbf{R}_1$ in one new attribute (namely, Q_1) that was introduced into $\mathbf{R}_2$ by an intermediate $\boldsymbol{P}$-transformation. This attribute is somewhat different than Q' and P', since its associated MVD m_{Q_1} is redundant in $\mathbf{R}_2$. If we drop Q_1 from the MVDs of $\mathbf{R}_2$, then they will look exactly like the MVDs of $\mathbf{R}_1$. Furthermore, let F_2 and U_2 denote, respectively, the FD-set and the set of attributes of $\mathbf{R}_2$. It is straightforward to verify that $F_2^+[U_2 - \{Q_1\}]$ is equivalent to the FD-set of $\mathbf{R}_1$ (provided that Q' and P' are renamed to Q and P as assumed above). It is now obvious that $\mathbf{R}_2$ is a conflict-free extension of $\mathbf{R}_1$ which in its turn is a conflict-free extension of the initial scheme of Example 4. In summary, we reduced $\mathbf{R}_2$ to a canonical scheme $\mathbf{R}_1$ essentially by dropping attribute Q_1 and renaming the rest of the new attributes of $\mathbf{R}_2$. This situation is an instance of the general paradigm described next.

Recall that each application of $\boldsymbol{P}$ introduces a new attribute $P_{X,Y}$ along with an associated MVD $m_{P_{X,Y}}$. At the moment it is introduced, $m_{P_{X,Y}}$ is nonredundant [7]. However, it may become redundant at later stages, after additional new MVDs (and attributes) are added. Thus, in Example 4, m_{Q_1} becomes redundant in $\mathbf{R}_2$.

Let $R = (U, M \cup F)$ and denote $\boldsymbol{P}^*(R)$ by $R' = (U', M' \cup F')$. We say that a newly added attribute Q (i.e., s.t. $Q \in U' - U$) is *weak* if its associated MVD m_Q becomes redundant at some stage. Weak attributes can be removed from R' as described above. More formally, the result of removing all weak attributes is the scheme $\bar{R} = (\bar{U}, \bar{M} \cup \bar{F})$, where $\bar{U}$ is U' with all the weak attributes removed; $\bar{M}$ is obtained from M' by dropping all the weak attributes from wherever they appear in the MVDs of M'. Finally, $\bar{F}$ is taken to be a set of FDs equivalent to $(F')^+[\bar{U}]$.

Let Φ be the transformation that results from applying $\boldsymbol{P}^*$ first (using some series of pairs of LHSs) and then deleting all the weak attributes. It is shown in [8] that Φ is independent of the choice of a particular series of LHSs used in $\boldsymbol{P}^*$. Of course, this independence should be understood modulo renaming of the (nonweak) new attributes. For instance, in Example 4 we had to rename P' and Q' to P and Q, respectively, in order to make schemes equivalent.

An efficient algorithm for Φ can be found in [8]. This alogrithm requires a deep knowledge of the structure of intersection anomalies and will not be presented here. It is also shown in [8] that, for any R, $\Phi(R)$ is a conflict-free extension of R. Furthermore, it is, in a well-defined sense, the most natural solution to the problem of elimination of intersection anomalies from R [17].

In short, we can define a structure of *algebraic category* [20] on the collection of all schemes with nonsplit LHSs in which Φ is an *algebraic functor* to a subcategory of conflict-free schemes. It is argued that any natural solution to our problem should have certain functorial properties (which Φ has too). It turns out then that any functor with these properties is equivalent to Φ. The interested reader is referred to [17] for details.

While the above argues for the naturalness of the proposed transformation, the next remarkable property from [7] shows that Φ fits well into our design philosophy:

> for any $X \subseteq U$, X is split by M iff it is split by $\bar{M}$. Thus $SG(M, U) = SG(\bar{M}\ \bar{U}) \cap U$ and Φ preserves both close and indirect relationships among the attributes.

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

The only problem may be with the FDs. If no $\bar{M}$-cover for $\bar{F}$ is embedded into the maximal cliques of $SG(\bar{M}, \bar{U})$, then the FDs impose interclique constraints. However, since in scheme design we are dealing with structural FDs only, it follows from Assumption 2 and Theorem 4 that there is an embedded M-cover for F. In such case Φ assures that $\bar{F}$ also has an embedded $\bar{M}$-cover [7]. We summarize the discussion in the following theorem:

THEOREM 5. ([7]) *Suppose that the elements of* $\boldsymbol{LHS}(M \cup F)$ *are not split, and let* $(\bar{U}, \bar{M} \cup \bar{F}) = \Phi((U, M \cup F))$. *Then the elements of* $\boldsymbol{LHS}(\bar{M} \cup \bar{F})$ *are not split. Particularly,* $\bar{F}$ *is embedded into the maximal cliques of* $SG(\bar{M}, \bar{U})$.

4.4 The Design Algorithm

Theorems 4 and 5 ensure that the following design procedure works well: First, find an FD-cover embedded in the maximal cliques of $SG(M, U)$. Then extend the scheme to an acyclic one. Decompose the scheme using only MVDs (or, equivalently, using the single JD). At the last stage, locally refine the resulting schemes using, say, the synthesis algorithm of [12].

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

Example 5. Consider the following dependencies:

$$
\begin{aligned}
m&: ABCDEF \twoheadrightarrow K \mid L \\
f_1&: K \rightarrow A \\
f_2&: L \rightarrow A \\
f_3&: CA \leftrightarrow AF \\
f_4&: BA \leftrightarrow AE \\
f_5&: AEF \rightarrow D \\
f_6&: ABC \rightarrow D
\end{aligned}
$$

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: $ABCDEFK$ and $ABCDEFL$. 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 $ABCDEF$ will be represented by different sets of schemes.

This problem is readily rectified by first finding some global embedded M-cover F and then synthesizing inside each cluster using only the FDs of F embedded in 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-LHS anomaly is detected). In such case, we, at least, are able to point to the semantic problems with scheme specifications supplied by the designer. Depending upon the policy pursued, the design process either stops signaling the detected problem or proceeds trying to eliminate the dependencies with split LHSs. Regardless of the chosen policy, the designer must be informed by the algorithm about the semantic inconsistency in his specifications.

The Design Algorithm

Input: $(U, M \cup F)$

Preliminary Stage

1. Make the MVDs full and close their LHSs.

The Nonconventional Stage

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

The Conventional Stage

1. Find an $\bar{M}$-cover embedded in $\bar{J}$. Note that by Theorems 4 and 5, and since the LHSs in $\mathbf{LHS}(M' \cup F')$ are not split, $\bar{F}$ *is* such a cover.
2. Decompose according to $\bar{M}$ (or, equivalently, according to $\bar{J}$).
3. **For each** constituent W of $\bar{J}$ **do**
   Apply the synthesis algorithm of [12] to W, using only those FDs of $\bar{F}$ that are embedded in W.
   end
   By [12], this operation losslessly decomposes each component of $\bar{J}$. Let $\mathbf{R}$ denote the resulting set of schemes and let $\tilde{F}$ stand for the FD-set produced by the synthesis algorithm [12] ($\tilde{F}$ is embedded in $\mathbf{R}$).

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 $\boldsymbol{R}$. Then $\tilde{D}$ is equivalent to $\bar{M} \cup \bar{F}$.*

PROOF. As noted earlier, $\bar{M} \cup \bar{F}$ is equivalent to $\bar{J} \cup \bar{F}$. Since the synthesis algorithm of [12] preserves FDs, $\tilde{F}^+ = \bar{F}^+$. Clearly, $\tilde{J}$ implies $\bar{J}$ [10]. 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, in turn, follows from the fact that the synthesis algorithm losslessly decomposes each constituent of $\bar{J}$ [12]. □

The nonconventional design stage of the algorithm can be accomplished in polynomial (in size of the schema) time, as shown in [8]. The complexity of the conventional stage is also polynomial. An efficient decomposition algorithm for conflict-free sets of MVDs is given in [19]. The complexity of the synthesis algorithm is discussed in [3].

The synthesis algorithm of [12] yields 3NF database schemes, which is nice. It is known, however, that locally consistent databases over 3NF schemes may be globally inconsistent [22]. Sagiv shows that *scheme independence* [22] is a much better goal (than 3NF) and is free from the above deficiency. To make $\mathbf{R}$ independent w.r.t. $\tilde{J} \cup \tilde{F}$, it suffices to ensure scheme independence within every constituent of $\bar{J}$ w.r.t. the embedded FDs [18]. Unfortunately, independence w.r.t. FDs is unachievable within the classical setting. We believe that the solution can be found within the framework delineated in this paper and in [17].[4]

5. COMPARISON TO OTHER METHODS

The only algorithm we know about that reasonably tries to benefit from both FDs and MVDs is due to Zaniolo and Melkanoff [27]. 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 approach is conceptually complicated. We demonstrate our method on a scheme of Example 7 (below), and achieve the same result by a far simpler (both conceptually and computationally) method than in [27].

[4] In this connection we would like to cite a recent improvement of the Bernstein's synthesis algorithm due to Biskup and Meyer [13]. This algorithm is quite different from that of [11]. Its most important characteristic is that synthesis is guided by the semantics pertaining the FDs. However, it is still a traditional design algorithm which does not change original schema specifications. Particularly, schema independence cannot be achieved in this way.

Example 6. ([27]) Consider the following dictionary database. The attributes are *F(rench)*, *G(erman)* *and* *E(nglish)*. The MVDs $E \twoheadrightarrow F \mid G$, $F \twoheadrightarrow E \mid G$, and $G \twoheadrightarrow E \mid F$ denote the fact that any term in any language has (possibly more than one) translation into any other language.

The execution of our algorithm skips over the preliminary stage and the first part of the nonconventional stage. It is easy to see that a single application of P suffices to extend the original scheme to a conflict-free one. The resulting scheme has the universe $U' = FGEC$, where C is the new attribute, and the MVD $C \twoheadrightarrow F \mid G \mid E$. The FDs are $F \rightarrow C$, $G \rightarrow C$, and $E \rightarrow C$. At the conventional stage we obtain the following database scheme: ($\underline{FC}$), ($\underline{GC}$), and ($\underline{EC}$)(keys are underlined).

By instant inspection of the dependencies we conclude that semantically the new attribute, C, stands for Concept(s) represented by terms in different languages. Interestingly, Zaniolo and Melkanoff arrived at the same results as we did, by using intuitive arguments. Their algorithm, however, is unable to design such a scheme.

Example 7. Consider the following enterprise borrowed from [27]:

LIC	licence number of motor vehicles,
MAKE	manufacturer of motor vehicles,
MODEL	model of vehicle,
YEAR	year in which the vehicle was manufactured,
VALUE	current value of vehicle,
OWNER	unique identifier of a person,
DRVL	driver's licence number,
VIOL	code number for traffic violation
DATE	date of violation.

The dependencies (as specified in [27]) are as follows:

$LIC \rightarrow MAKE\ YEAR\ MODEL\ VALUE\ OWNER\ DRVL$
$LIC \twoheadrightarrow \{VIOL\ DATE\}$
$\{MAKE\ YEAR\ MODEL\} \rightarrow VALUE$
$\{MAKE\ YEAR\ MODEL\} \twoheadrightarrow \{LIC\ OWNER\ DRVL\ VIOL\ DATE\}$
$OWNER \rightarrow DRVL$
$DRVL \rightarrow OWNER$
$OWNER \twoheadrightarrow \{LIC\ MAKE\ YEAR\ MODEL\ VALUE\} \mid \{VIOL\ DATE\}$
$DRVL \twoheadrightarrow \{LIC\ MAKE\ YEAR\ MODEL\ VALUE\} \mid \{VIOL\ DATE\}$

Our algorithm closes LHSs and finds a minimal cover at the preliminary stage yielding the following dependencies:

$\{DRVL\ OWNER\} \twoheadrightarrow \{LIC\ MAKE\ YEAR\ MODEL\ VALUE\} \mid \{VIOL\ DATE\}$
$OWNER \rightarrow DRVL$
$DRVL \rightarrow OWNER$
$LIC \rightarrow MAKE\ YEAR\ MODEL\ VALUE\ OWNER$
$\{MAKE\ YEAR\ MODEL\} \rightarrow VALUE$

We arrive at a scheme with a single MVD and a collection of FDs with nonsplit LHSs. Hence there is no need for the nonconventional stage. Decomposition now yields two big clusters: $\{VIOL\ DATE\ DRVL\ OWNER\}$ and $\{DRVL\ OWNER\ LIC\ MAKE\ YEAR\ MODEL\ VALUE\}$. Synthesizing separately inside each

component, we obtain the following final scheme, where different keys are underlined differently:

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

This result is identical to that of [27], but is obtained in a far simpler way.

Observe that in the example the second and the fourth MVDs are implied by the first and the third FDs, respectively, and thus are not used in our design process. However, the method of [27] requires these implied MVDs to be explicitly specified (as they are in Example 7), 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 FDs.

Note also that the nonconventional stage in the example is empty. This is the reason why the conventional approach of [27] succeeds here.

We conclude with a more sophisticated example in which more than one application of the $\boldsymbol{P}$-transformation is required at the nonconventional stage.

Example 8. Let the attributes be $E(mployee)$, $L(aboratory)$, $C(omputer)$, and $(pro)J(ect)$. The meaning of a tuple $\langle e, l, c, j \rangle$ is that employee e has an access to laboratory l, has an account on computer c, and is working for project j.

Suppose that from the employee's point of view all the laboratories and computers he is permitted to use are the same, that is, he can log in on any computer (to which he has an access) using terminals of any laboratory he is permitted to work in. An employee can work for any of his projects in any laboratory and on any computer he has an access to. These conditions are expressed by the following MVD:

$$E \twoheadrightarrow L \mid C \mid J$$

Suppose furthermore that employees working for the same project have similar working conditions, that is, they can access the same laboratories and computers. This is expressed by the following MVD:

$$J \twoheadrightarrow L \mid C \mid E$$

Finally, assume that for any computer all the users have the same rights in using the laboratories, that is, for any particular computer all its users are permitted to work in any laboratory from which they can access that computer. This is expressed as follows:

$$C \twoheadrightarrow L \mid EJ$$

Summarizing, we have the following dependencies:

$$E \twoheadrightarrow L \mid C \mid J$$
$$J \twoheadrightarrow L \mid C \mid E$$
$$C \twoheadrightarrow L \mid EJ$$

Application of $\boldsymbol{P}$ to the first pair of MVDs results in the following scheme, where W is a new attribute:

$$W \twoheadrightarrow L \mid C \mid JE$$
$$C \twoheadrightarrow L \mid EJW$$
$$E \rightarrow W, \quad J \rightarrow W$$

The last scheme still has an intersection anomaly which can be removed by a subsequent application of $\boldsymbol{P}$. This application introduces another new attribute A, and results in the following scheme:

$$A \twoheadrightarrow L \mid CEJW$$
$$WA \twoheadrightarrow L \mid C \mid JE$$
$$C \rightarrow A, \quad W \rightarrow A, \quad E \rightarrow W, \quad J \rightarrow W$$

Without going deeply into the formal semantics that can be ascribed to the new attributes (which can be found in [7] and [8]), we describe this semantics with informal arguments only.

Let r be a relation satisfying the dependencies of the final scheme. Any value w of W defines the set $V_w \equiv (\sigma_{W=w} r)[LC]$, where $\sigma_{W=w}$ denotes selection on attribute W and the square brackets stand for projection on LC.

From the FD $E \rightarrow W$, it follows that every employee has only one associated W-value. Let w be associated with an employee e. Then, V_w represents all pairs $\langle l, c \rangle$ s.t. e can log in on c from a terminal in laboratory l. The set of these pairs corresponds to what we called earlier *employee's working conditions*. Thus, W-values represent working conditions of different employees. The FD $J \rightarrow W$ states that employees working for the same project have the same working conditions, which complies with our original motivation for introducing the MVD $J \twoheadrightarrow L \mid C \mid E$.

Similarly, any A-value a defines the set $V_a \equiv (\sigma_{A=a} r)[L]$. It is easily verified that the FDs $J \rightarrow A$, $E \rightarrow A$, and $C \rightarrow A$ hold in our scheme. Let e be an employee associated with a and let j and c be a project and a computer associated with a. Then, V_a represents the set of laboratories where e is permitted to work. Alternatively, we can look at V_a as the set of laboratories in which project j can be carried out or from which computer c can be accessed. Thus, A describes laboratory assignments to employees, projects, and computers.

At that point the nonconventional stage of design has been completed. The decompositional phase of the conventional stage produces the following clusters of attributes:

$$AL \quad AWC \quad AWJE$$

The first cluster is in the 3NF. The second cluster will be decomposed by the synthesis algorithm of [12] into (A $\underline{W}$), ($\underline{C}$ A), and ($\underline{WC}$). The third cluster has the following embedded FDs: $J \rightarrow W$, $E \rightarrow W$, and $W \rightarrow A$. Thus, after the synthesis, we have ($\underline{J}$ W), ($\underline{E}$ W), ($\underline{W}$ A), and ($\underline{JE}$). In summary, we arrive at the following collection of schemes: ($\underline{AL}$), ($\underline{C}$ A), ($\underline{W}$ A), ($\underline{WC}$), ($\underline{J}$ W), ($\underline{E}$ W), and ($\underline{JE}$).

The price we pay for the luxury of nonconventional design is that we now have to provide values for the entities represented by the new attributes. Fortunately, their values can be provided automatically [7, 8]. Since the new attributes can be ascribed clear semantic meaning, it will also be easy for a user to provide appropriate values when adding new tuples.

On the positive side, we achieve significant simplification in schema structure and in the exposition of connections between schema attributes. This pays off at database maintenance and querying time. For instance, any attempt to design the scheme of Example 8 within the old scope of attributes will necesarily obscure some of the connections represented by the schema MVDs (see the discussion in Section 4.1). This in its turn will cause computational problems in preserving database consistency. More seriously, shading in some of the schema interconnections may expose database users to errors in formulating their queries, that is, an obscure schema semantics may cause users to join data along lossy join paths (i.e., along the so-called *connection traps* [14]).

6. SUMMARY AND CONCLUSIONS

Our approach reveals some interesting properties of the structure of real-world databases. It shows that there is a macrostructure delineated by the global JD (whose components are the SFNF-clusters obtained at the decomposition stage). The infrastructure is defined at the synthesis stage by refining the macrostructure. This sheds a new light upon the celebrated question whether the real world is cyclic or acyclic: in the case where specifications are given by MVDs and FDs, the world has an acyclic macrostructure (a JD equivalent to a set of MVDs is acyclic [16]), but the infrastructure is most probably cyclic. In the situations where FDs and MVDs are not the only data dependencies, even the macrostructure is likely to 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 is discussed elsewhere [18].

In summary, we present a consistent approach to the problem of scheme design, based on new principles. We argue that database design methods should have a preliminary stage (which we call "nonconventional" in order to contrast it to the approaches which have prevailed so far in the literature). This new stage is intended to deal with various "bad things" that plagued most of the previous approaches. Here we have spotted two kinds of such "bad things." Nonstructural additives to schema specifications, which do not really belong to the design process, are one kind of such "bad things." The second one occurs when the database designer overlooks some relevant specifications. To deal with these issues a comprehensive design system should combine syntactic tests to spot the problems (such as Assumptions 1 and 2 in Sections 4.1 and 4.2) with theoretically well-founded schema improvement methods (such as Φ in Section 4.3), along with various heuristics and the designer's assistance. We then applied these ideas

to the classic problem of schema design for FDs and MVDs and came up with a new method that extends all the known approaches and leads to a successful design for a wide variety of situations for which FDs and MVDs constitute an adequate class of schema specifications.

However, in many real cases it is not enough to specify FDs and MVDs alone. Unfortunately, we have no 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 only provides a partial solution to the design problem for FDs. Its drawback is that the resulting schemes may not be independent [22]. Therefore, there are interrelational constraints which are hard to enforce. It is known that no synthesis algorithm can design independent databases. A plausible solution is to find a suitable scheme correction [17, 24]. We believe that the general ideas described in this paper can serve as a framework for the further study of these issues.

ACKNOWLEDGMENTS

The authors are grateful to Yehoshua Sagiv for his helpful comments. We also appreciate the contributions made by Joachim Biskup and two other anonymous referees, whose remarks helped to improve the exposition. Many thanks to Philip Bernstein who kindly agreed to read this manuscript and pointed out some errors.

REFERENCES

1. AHO, A. V., BEERI, C., AND ULLMAN, J. D. The theory of joins in relational databases. *ACM Trans. Database Syst. 4*, 3 (Sept. 1979), 297–314.
2. BEERI, C. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Trans. Database Syst. 5*, 3 (Sept. 1980), 241–259.
3. BEERI, C., AND BERNSTEIN, P. A. Computational problems related to the design of normal form relational schemes. *ACM Trans. Database Syst. 4*, 1 (Mar. 1979), 30–59.
4. BEERI, C., BERNSTEIN, P. A., AND GOODMAN, N. A sophisticate's introduction to database normalization theory. In *Proceedings of the International Conference on Very Large Data Bases* (West Berlin, 1978), 113–124.
5. BEERI, C., FAGIN, R., AND HOWARD, J. H. A complete axiomatization for functional and multivalued dependencies in database relations. In *Proceedings of ACM-SIGMOD International Conference on Management of Data* (1977), ACM, New York, 47–61.
6. BEERI, C., FAGIN, R., MAIER, D., AND YANNAKAKIS, M. On the desirability of acyclic database schemes. *J. ACM 30* (July 1983), 479–513.
7. BEERI, C., AND KIFER, M. Elimination of intersection anomalies from databases schemes. *J. ACM* (July 1986), to appear.
8. BEERI, C., AND KIFER, M. A theory of intersection anomalies in relational database schemes. *J. ACM*, to appear.
9. BEERI, C., AND KIFER, M. A comprehensive approach to the design of relational database schemes. In *Proceedings of the 10th International Conference on Very Large Data Bases* (Singapore, Aug. 1984), 196–207.
10. BEERI, C., MENDELSON, A. O., SAGIV, Y., AND ULLMAN, J. D. Equivalence of relational database schemes. *SIAM J. Comput. 10*, 2 (May 1981), 352–370.
11. BERNSTEIN, P. A. Synthesizing third normal form relations from functional dependencies. *ACM Trans. Database Syst. 1*, 4 (Dec. 1976), 247–298.
12. BISKUP, J., DAYAL, U., AND BERNSTEIN, P. A. Synthesizing independent database schemes. In *Proceedings of ACM-SIGMOD International Conference on Management of Data* (1979), ACM, New York, 143–151.

13. BISKUP, J., AND MEYER, R. Design of relational database schemes by deleting attributes in the canonical decomposition. Unpublished manuscript, Dortmund Univ.
14. CODD, E. F. A relational model of data for large shared data banks. *Commun. ACM 13* (1970), 377–387.
15. FAGIN, R. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst. 2*, 3 (1977), 262–278.
16. FAGIN, R., MENDELSON, A. O., AND ULLMAN, J. D. A simplified universal assumption and its properties. *ACM Trans. Database Syst. 7*, 3 (Sept. 1982), 343–360.
17. KIFER, M. Nonconventional design theory for relational database schemes. TR 85-07, Dept. of Computer Science, SUNY at Stony Brook, Feb. 1985.
18. KIFER M., AND SAGIV, Y. Computing windows on real-world databases. In preparation.
19. LIEN, Y. E. On the equivalence of database models. *J. ACM 29*, 2 (Apr. 1982), 333–362.
20. MACLANE, S. *Categories for Working Mathematicians*. Springer Verlag, New York, 1971.
21. MAIER, D. *The Theory of Relational Databases*. Computer Science Press, Potomac, Md., 1983.
22. SAGIV, Y. A characterization of globally consistent databases and their correct access paths. *ACM Trans. Database Syst. 8*, 2 (June 1983), 266–286.
23. SCIORE, E. Real-world MVDs. Tech. Rep. 80/014, Dept. of Computer Science, SUNY at Stony Brook, Nov. 1980.
24. SCIORE, E. Improving database schemes by adding attributes. *ACM PODS* (1983), 379–383.
25. SCIORE, E. The universal instance and database design. Ph.D. dissertation, TR 271, Princeton Univ., June 1980.
26. ULLMAN, J. D. *Principles of Database Systems*. Computer Science Press, Potomac Md., 1980.
27. ZANIOLO, C., AND MELKANOFF, M. A. On the design of relational database schemata. *ACM Trans. Database Syst. 6*, 1 (Mar. 1981), 1–47.

Received March 1985; revised September 1985; accepted November 1985