A New Normal Form for Nested Relations*

Z. Meral Ozsoyoglu and Li-Yan Yuan

Computer Engineering and Science Department
and
Center for Automation and Intelligent Systems
Case Western Reserve University
Cleveland, Ohio 44106

ABSTRACT

We consider nested relations whose schemes are structured as trees, called scheme trees, and introduce a normal form for such relations, called the nested normal form. Given a set of attributes U, and a set of multivalued dependencies (MVD´s) M over these attributes, we present an algorithm to obtain a nested normal form decomposition of U with respect to M. Such a decomposition has several desirable properties, such as explicitly representing a set of full and embedded MVD´s implied by M, and being a faithful and nonredundant representation of U. Moreover, if the given set of MVD´s is conflict free, then the nested normal form decomposition is also dependency preserving. Finally, we show that if M is conflict free, then the set of root-to-leaf paths of scheme trees in nested normal form decomposition is precisely the unique 4NF decomposition [Fa, L2] of U with respect to M.

1. Introduction

A relational database [Co] is a collection of relations where each relation is at least in First Normal Form (1NF), i.e., tuple components of a relation are atomic (nondecomposable) values. However, for non-business database applications such as text processing, computer aided design, form management, and statistical databases, 1NF relations are not appropriate. It is generally agreed that utilizing relational databases for such applications requires non-first normal form, (i.e., nested) relations, whose tuple components can be sets or even relations themselves.

Makinouchi [Mak] recognizes the need for using relational databases for non-business database applications. He introduces nested relations, extends functional dependencies (FD´s) and multivalued dependencies (MVD´s) to nested relations, and defines a normal form by incorporating extended dependencies into 4NF definition. In [Mak], however, a method to obtain a database scheme in this normal form is not given. Utilizing non-first normal form relations for structuring database outputs is discussed by Kambayashi et. al. [KTT]. Fischer and Gucht study one level nested relations and their characterization by a new family of dependencies [FG]. Roth, Korth and Silberschatz define a normal form called Partitioned Normal Form for nested relations [RKS].

Recently, several researchers have reported query languages for non-first normal form relational databases. Jaeschke and Schek [JS] extended relational algebra by nest and unnest operators over single attributes to obtain nested relations from normalized relations and vice versa. Fischer and Thomas [FT] study operators for non-first normal form relations in a more general setting. Abiteboul and Bideout [AB] describe the use of non-first normal form relations in the VERSO machine [Ba], and give algebraic operators and their properties. Ozsoyoglu and Ozsoyoglu [OO] consider operators similar to that of [JS], for set valued relations (i.e., relations with at most one level of

* This research is supported in part by the NSF under Grant No-8306616, and an IBM Faculty Development Award. The paper appears in CM Transactions on Database Systems, 12(1):111-136, 1987.

nesting), and extend the basic algebra for such relations by aggregation operators. A relational calculus for set valued relations with aggregate functions is given, and its equivalence to the corresponding relational algebra is shown in [OzOM]. A relational calculus for non-first normal form relations is given by Roth, Korth and Silberschatz [RKS], who also define a basic relational algebra for non-first normal form relations, and show its equivalence to the relational calculus.

In this paper, we introduce a normal form for nested relations, called the *nested normal form* (NNF). One of the objectives of NNF; as in previous normal forms, is to reduce redundancy [BBG]. Unlike flat normalized relations (such as 3NF, BCNF, 4NF, etc.), a nested relation has a nesting structure of its attributes which imply some semantic connections among attributes. Thus, a normal form for nested relations aims not only to group attributes into related sets of attributes, but also to choose a nesting structure which is a good representation of the set of semantic connections that already exist in the real world among attributes. We consider the real world to be modeled by the database as a collection of attributes, U, and a set of multivalued dependencies over U. We first define the NNF to meet the above objectives, and then give a decomposition algorithm to obtain a NNF database scheme for U with respect to the given set of dependencies. For convenience, we consider FD´s as MVD´s, similar to Lien [L1], i.e., we consider MVD counterparts of FD´s. Informally, a relation is in NNF if it is structured in the form of a tree, called the *normal scheme tree*. Vertices of a normal scheme tree are pairwise disjoint sets of attributes, and edges represent MVD´s implied by the given set M of MVD´s. Precise definitions for the NNF and the normal scheme tree will be given later (Section 4). We first give an example.

Example 1.1. Consider a database about course offerings, which contains information about COURSEs offered for study in a university [Ha]. Each COURSE is offered in any number of SECTIONs, and the same TEXT is used for the same COURSE in all SECTIONs. Each SECTION consists of a set of class meetings on a number of DAYs, and has a set of teaching assistants (GRADERs) assigned to the section. (To keep the example simple, we omit the instructors, classrooms, prerequisites etc. which are also related to course offerings.) That is,

U = {COURSE, TEXT, SECTION, DAY, GRADER},

and the set of MVD´s, is

M = { COURSE $\twoheadrightarrow$ TEXT; {COURSE, SECTION} $\twoheadrightarrow$ DAY}.

Then R = {(COURSE, TEXT), (COURSE, SECTION, DAY), (COURSE, SECTION, GRADER)} is a 4NF decomposition for U. Consider the following tree T_R to represent R. We can see that each edge of T_R represents an MVD implied by M.

T_R COURSE .

e1 e2

TEXT . . SECTION

e3 e4

DAY . . GRADER

e1 = COURSE $\twoheadrightarrow$ TEXT
e2 = COURSE $\twoheadrightarrow$ {SECTION,DAY,GRADER}
e3 = {COURSE,SECTION} $\twoheadrightarrow$ DAY
e4 = {COURSE,SECTION} $\twoheadrightarrow$ GRADER

In this example T_R is the normal scheme tree; and the scheme for the corresponding NNF relation is R = COURSE (TEXT)*(SECTION (DAY)*(GRADER)*)*, where * represents a repeating group. A possible instance for R with two tuples using the notation in [AB] is shown below:

R :

COURSE	$(TEXT)^*$	(SECTION	$(DAY)^*$	$(GRADER)^*)^*$
c1	Design	s1	Mon	John
	Analysis		Wed	Mary
		s2	Tue	Joe
			Thur	Su
c2	Data Structure	s1	Mon	Sally
	Database		Wed	
			Fri	

Note that, a flat relation over U storing the above instance has 22 tuples, which is shown below. The significance of the tree structure representation for R is that MVD´s implied by M are explicitly represented in the scheme itself. Furthermore, it suggests a storage structure with minimal redundancy.

U :

COURSE	TEXT	SECTION	DAY	GRADER
c1	Design	s1	Mon	John
c1	Design	s1	Mon	Mary
c1	Design	s1	Wed	John
c1	Design	s1	Wed	Mary
c1	Analysis	s1	Mon	John
c1	Analysis	s1	Mon	Mary
c1	Analysis	s1	Wed	John
c1	Analysis	s1	Wed	Mary
c1	Design	s2	Tue	Joe
c1	Design	s2	Tue	Su
c1	Design	s2	Thur	Joe
c1	Design	s2	Thur	Su
c1	Analysis	s2	Tue	Joe
c1	Analysis	s2	Tue	Su
c1	Analysis	s2	Thur	Joe
c1	Analysis	s2	Thur	Su
c2	Data Structure	s1	Mon	Sally
c2	Data Structure	s1	Wed	Sally
c2	Data Structure	s1	Fri	Sally
c2	Database	s1	Mon	Sally
c2	Database	s1	Wed	Sally
c2	Database	s1	Fri	Sally

This example motivates us to define a scheme tree to represent hierarchically organized data. The relation R in the above example is called a *nested relation.* The scheme of a nested relation (which is called format in [AB]) is a parenthesized representation of the scheme tree. As it will be explained later in the paper, some scheme trees are better than others in reducing redundancy, and in representing the MVD´s implied by M. This is the motivation for the normal scheme tree and the NNF. In the above example, T_R is a normal scheme tree of U with respect to M, and the set of MVD´s implied by T_R is equivalent to M. Moreover, the *path set* of T_R, i.e. {(COURSE, TEXT), (COURSE, SECTION, DAY), (COURSE, SECTION, GRADER)} is the unique 4NF decomposition [Fa] of U with respect to M, which is also in Split-Free Normal Form (SFNF) [BK2]. (Note that M in Example 1.1 is a conflict free set of MVD´s [L2, BFMY].)

Given U and a set M of MVD´s, we present a decomposition of U into a set of NNF relations (Section 4). Since the scheme of an NNF relation is a normal scheme tree, such a decomposition F={T_1, ..., T_n} is called a *normal scheme forest* (NSF) decomposition of U with respect to M. In addition to being a database scheme for nested relations, an NSF decomposition F has several desirable properties. First, F represents a set of full and embedded MVD´s implied by M. Second, F is a faithful [Ma], and a nonredundant (to be explained later) representation of U. Finally, if M is conflict free then F is a dependency preserving decomposition, i.e. MVD´s represented by F is equivalent to M. Moreover, we also show that if M is conflict free, then the path set of F is the unique SFNF decomposition of U with respect to M.

The differences between the Nested Normal Form and the other normal forms for nested relations, namely the Partitioned Normal Form of [RKS], and the Normal Form of [Mak] are as follows.

(1) The definition of the Normal Form (NF) [Mak] is based on extended (functional and multivalued) dependencies, and is an extension of 4NF of normalized relations to nested relations. Our approach in defining nested normal form is based on dependencies existing among attributes, and does not explicitly use extended dependencies. The nested normal form aims at finding a structure for a nested relation (i.e., normal scheme tree) which is a good representation of the given set of MVD´s among attributes. Although both normal forms have the same motivation of reducing redundancies, and are similar, the definition of redundancies (see section 4.1) in our context differs from the one given in [Mak]. That is, in [Mak] redundancies are defined in terms of extended dependencies while in this paper they are defined in terms of regular dependencies. Moreover, in this paper, we present a nested normal form decomposition which is dependency preserving if M is conflict free, while in [Mak] only a recursive definition of the NF is given.

(2) A normal form for nested relations, called Partitioned Normal Form (PNF), is defined in [RKS]. It is also shown that the class of PNF relations are closed under the set of extended algebra operations for nested relations [RKS]. The PNF is precisely the same as structuring nested relations in the form of scheme trees. That is, a nested relation R is in PNF iff the scheme of R is a scheme tree with respect to the given set of MVD s. However, not all scheme trees are good structures for nested relations. The anomalies which may exist in nested relations structured as scheme trees are discussed in section 4.1, and the normal scheme tree is defined to avoid such anomalies. A relation is in Nested Normal Form (NNF) if its scheme is a normal scheme tree with respect to M. Since a normal scheme tree is a scheme tree, any relation that is in NNF is also in PNF, but the reverse is not true. For example,

$$R = \text{COURSE (TEXT)}^* \text{ (SECTION (DAY)}^* \text{ (GRADER)}^*)^*$$

of Example 1.1 is in both PNF and NNF. However, any flat relation is also in PNF trivially, since in this case the scheme tree consists of a single node whose label is the set of all attributes of the relation.

Recently, we have presented [YO] a unifying approach to incorporate FD’s and MVD’s into database design. Utilizing these results will improve the design of nested normal form database schemes, but is beyond the scope of this paper.

The rest of the paper is organized as follows. In section 2, we discuss the basic concepts, terminology and definitions. In section 3, some properties of reduced MVD´s are discussed. Section 4 presents normal scheme tree, normal scheme forest, and their properties. Section 5 contains the NSF decomposition algorithm. The properties of the NSF decomposition are given in section 6.

2. Preliminaries

We assume that the reader is familiar with the theory of functional dependencies, multivalued dependencies, and join dependencies (JD´s). We also assume familiarity with the notions of fourth normal form (4NF) [Fa, Ma, Ul], and the acyclic database schemes [BFMY].

In this section, we briefly review some fundamental concepts, and present the notations used throughout the paper. U denotes the universal relation scheme, i.e. a finite set of attributes. X, Y, Z denote subsets of U. An FD is denoted as X→Y, an MVD is denoted as X→→Y, and X→→Y | Z is a shorthand for X→→Y and X→→Z. We use $*(R_1, ..., R_n)$ or *(R), where R=$\{R_1, ..., R_n\}$, to denote a join dependency. To simplify the notation we may use XYA to denote the set $X \cup Y \cup \{A\}$, where X, Y are sets of attributes, and A is an attribute.

In this paper, the following set of inference rules for MVD´s is used.

FM0:	If X→Y then X→→Y.
M1:(Complementation)	If X→→Y then X→→(U − XY).
M2:(reflexivity)	If $Y \subseteq X$ then X→→Y.
M3:(augmentation)	If X→→Y and $Z \subseteq W$ then XW→→YZ.
M4:(transitivity)	If X→→Y and Y→→Z then X→→Z − Y.

The inference rules M1 − M4 are shown [BFH] to be sound and complete for MVD´s. Since we consider only MVD´s, we use the inference rule FM0 to obtain MVD counterparts of FD´s as in [L1]. For a set M of MVD´s, M^+ denotes the *closure* of M, i.e. the set of all MVD´s, that are implied by M. Given two sets M and N of MVD´s, M is a *cover* of N if $M^+ = N^+$.

2.1. Reduced MVD´s, Keys

Definition 2.1 Given a set M of MVD´s over U, an MVD X→→W in M^+ is said to be

(a) *trivial* if XW = U, W = ∅ or $W \subseteq X$,

(b) *left-reducible* if ∃X´, X´⊂X, such that X´→→W is in M^+,

(c) *right-reducible* if ∃W´, W´⊂W, such that X→→W´ is a nontrivial MVD in M^+.

(d) *transferable* if ∃X´, X´⊂X, such that X´→→W(X − X´) is in M^+,

An MVD X→→W is said to be *reduced* if it is nontrivial, left-reduced (i.e. non left-reducible), right-reduced (i.e. non right-reducible), and nontransferable.

Let M^- be the set of all reduced MVD´s implied by M, i.e., $M^- = \{$X→→W | X→→W is a reduced MVD in $M^+\}$. Then, from the following lemma and the definition of reduced MVD´s, M^- is a cover of M^+.

Lemma 2.1 [OY]. Let X→→W be a nonreduced MVD in M^+. Then there exists a reduced MVD X´→→W´ in M^+ such that X´ ⊂ X and X´W´ ⊆ XW, and if W ∈ DEP(X) then W´ = HW, where H ⊆ (X − X´). □

Definition 2.2. A set of MVD´s M is said to be a *minimal cover* if

(1) every MVD in M is reduced, and

(2) no proper subset of M is a cover of M.

There are several definitions of the minimal cover for MVD´s [BFMY, L1, ZM] which are different from the one given above. The difference is that we require each MVD in the minimal cover to be a reduced MVD. That is, if M is a minimal cover then $M \subseteq M^-$. An algorithm (polynomial in the number of keys) to find such a minimal cover is given in [OY]. The motivation for considering the minimal cover of reduced MVD´s is to eliminate the redundancies of nonreduced MVD´s. It should be noted that, minimal covers of left- and right-reduced (but not necessarily reduced) MVD´s are considered in [ZM], and covers of right-reduced MVD´s are considered in [BFMY]. As usual, we use LHS(M) to denote the set of left hand sides of the MVD´s in a set M of MVD´s.

Definition 2.3. Let N be a set of MVD´s, and M be a minimal cover of N. Then elements in LHS(M^-) are called *keys* of N, and the elements in LHS(M) are called *essential keys* of N. Elements in LHS(M^-) − LHS(M) are called *nonessential keys* of N.

The above definition of keys and essential keys is also different from those in [BFMY, L1]. Note that, a set N of MVD´s may have different sets of essential keys since the minimal cover of M is not unique. However, the set of keys for N is unique since M^- is the set of all reduced MVD´s in $M^+ = N^+$.

Given a set M of MVD´s, and a set X of attributes, DEP(X) denotes the *dependency basis* of X with respect to M [Ul]. Elements of DEP(X) are called *dependents* of X. Let RDEP(X) = {W ∣ X→→W is reduced}. Obviously, RDEP(X)⊆DEP(X). Elements of RDEP(X) are called *reduced dependents* of X. By definition, a set of attributes X is a key of M iff RDEP(X)≠∅.

Example 2.1. We use U = ABCDGH, and M = {A→→G, B→→H, GH→→C ∣ D} from [L1] to illustrate keys and reduced MVD´s. M is a minimal cover for itself. Thus, the set of essential keys is LHS(M) = {A, B, GH}. Since AB→→C, AH→→C and BG→→C are also reduced MVD´s in M^+, AB, AH, BG are nonessential keys of M. It is easy to verify that there are no other nonessential keys. Thus, the set of all keys for M is, LHS(M^-) = LHS(M) ∪ {AB, AH, BG}.

Proposition 2.2 [OY]. Let M be a set of MVD's, Z ∉LHS(M) be a key of M. Then, there exist a key X ⊂ Z, V ∈ RDEP(X), and Y ∈ LHS(M) such that Y splits V and Z = X (Y ∩ V).

The above proposition gives us a method to find all keys of a given set of MVD's. The algorithm to do so is presented in [OY].

Let M be a set of MVD´s on U, and V ⊆ U and X ⊆ V. We use the notation MD(X, W, V) to denote MVD X→→W embedded in U. If X→→Y is an MVD on U and V ⊂ U then MD(X, Y∩V, V) holds for V [Fa]. We say M ⊨ MD(X, Y∩V, V), if M ⊨ X→→Y, where V ⊂ U and X ⊆ V.

Definition 2.4. A dependent W of X is *essential* if

(1) X→→W is reduced, i.e. W is a reduced dependent of X, and

(2) If there are essential keys Y_1, ..., Y_n, n ≥ 1, of M such that Y_i≠X and W ∈ DEP(Y_i) for i = 1,...,n, then X→→W can be derived from M less those MVD´s having Y_i as their left sides.

Otherwise, W is a *nonessential dependent* of X. Essential dependents of X is denoted as EDEP(X).

Example 2.2. Let U = ABCDE and M = {A→→BC ∣ D, B→→A}. DEP(A) = {BC, D, E}, DEP(B) = {A, C, D, E}. Then EDEP(A) = DEP(A), and EDEP(B) = {A, C}. □

In this example, D is a dependent of both A and B. However, B→→D is derived from B→→A and A→→D by transitivity. M − {A→→BC ∣ D} ⊭ B→→D. Thus, D is an essential dependent of A but not B. In [Sc], a different definition for essential dependents is given. With respect to that definition, D in the above example is an essential dependent of both A and B. Thus, the definition in [Sc] does not distinguish the relationship of A to D from the relationship of B to D.

2.2 Conflict Free MVD´s and Normal Forms

Let M be a set of MVD´s and X, V be sets of attributes. X *splits* V if there are two dependents W_1, W_2 of X such that $W_1 \cap V \neq \emptyset$ and $W_2 \cap V \neq \emptyset$. A set M of MVD´s splits V if there is a Y ∈ LHS(M) such that Y splits V. It is shown in [BFMY] that a set M of MVD´s split V iff M^+ splits V. A set M of MVD´s is *conflict-free* [BFMY, L2, Sc], if

(1) M does not split elements in LHS(M), and

(2) DEP(X) ∩ DEP(Y) ⊆ DEP(X ∩ Y), where X, Y ∈ LHS(M).

Conflict-free MVD´s have several desirable properties as discussed in [BFMY, L2, Sc].

One of the normal forms which is considered to have good properties is *fourth normal form* (4NF) [Fa]. Let G be a set of MVD´s and FD´s and R = {R_1, ..., R_n}. Then R is in 4NF [Fa] if

(1) G implies *(R_1, ..., R_n), and

(2) for all R_i ∈ R, if whenever a nontrivial MVD X→→Y holds in R_i then so does the FD X→A for each A in R_i.

Let m be X→→W. We say U_1⊆U is *decomposable* with respect to m, or m-decomposable, if X⊂U_1, and U_1∩(W − X) and U_1 − XW are both nonempty. U_1⊆U is said to be decomposable with respect to a set of MVD´s M or M-decomposable, if there is at least one MVD m ∈ M such that U_1 is m-decomposable; Otherwise U_1 is *non-decomposable* with respect to M [L1]. By definition, if there is no FD´s (or FD´s are considered as MVD´s), a relation R is in 4NF iff R is nondecomposable with respect to M^+. However, there is another definition for 4NF, which we call *weak 4NF* (W4NF). Let G be a set of FD´s and MVD´s and R = {R_1, ..., R_n}. Then R is in W4NF if

(1) G implies *(R_1, ..., R_n), and

(2) for all R_i ∈ R, if G ⊨ (X →→ Y) and XY ⊂ R_i then G ⊨ (X → Y).

Obviously, 4NF implies W4NF, but W4NF does not necessarily imply 4NF.

Recently, [BK] introduced a new normal form, called *Split-Free Normal Form* (SFNF). A set of relations R = {R_1, ...R_n} is in SFNF with respect to M, if

(1) M ⊨ *(R), and

(2) no R_i ∈ R is split by M.

By definition, SFNF implies 4NF, but the reverse is not true, as illustrated by the following example.

Example 2.3. Let U_1 = ABCDE, M_1 = {A →→ B, AC →→ D, BD →→ E}, and R_1 = {AB, ACD, ACE}. Then R_1 is in 4NF and hence in W4NF. R_1 is not in SFNF since BD splits ACE. Let U_2 = ABCDE, M_2 = {A →→ B, C →→ D, E →→ AB}, and R_2 = {AB, CD, ACE}. Then R_2 is not in 4NF since E splits ACE. However, R_2 is in W4NF since there is no nontrivial MVD on ACE which can be derived from M_2 using the inference rules M1-M4.

It was shown [L2] that, if M is conflict free, then there is a unique 4NF decomposition. It was also shown [BK] that if M is conflict free then there is a SFNF decomposition R of U. Since SFNF implies 4NF, R must be unique. This proves the following proposition.

Proposition 2.3. Let M be a conflict free set of MVD´s on U, and R = {R_1, ..., R_n} be a 4NF decomposition of U. Then R is unique and in SFNF. □

A set M of MVD´s is said to satisfy the *intersection property* [BFMY] if whenever the MVD´s X→→V and Y→→V are implied by M (with V disjoint from both X and Y), then X ∩ Y→→V is also implied by M. The following is a useful lemma, which will be used to show the properties of keys and conflict free MVD´s later in the paper.

Lemma 2.4 [BFMY]. If M is a conflict free set of MVD´s, then M satisfies intersection property.

2.3 Nested Relations, Scheme Trees

Nested relations (or nonfirst normal form relations) are discussed by several researchers [AB, FG, JS, KTT, Mak, OO]. Informally, a nested relation is a relation whose tuple components are atomic values or repeating group instances. In this paper, we consider a special type of nested relations that are structured as a tree. Let U be a universal scheme, and T be a tree whose vertices are labeled by pairwise disjoint sets of attributes over U. Such a tree is called a *scheme tree.* If T is a scheme tree then *nested relation scheme* R(T) represented by T is defined as follows.

(1) If T is a single node labeled X then R(T) = X.

(2) Let A be the root of T and T_1,...,T_n denote its subtrees. Then R(T) = A(R(T_1))*...(R(T_2))*.

Note that, nested relation scheme is the same as the *format* defined in [AB]. An instance of a nested relation R(T) is a subset of DOM(R(T)), which is defined recursively as follows.

(1) If T is a single node labeled A_1...A_k then DOM(R(T))= DOM(A_1)x...xDOM(A_k).

(2) Let A be the root of T and T_1,...,T_n denote its subtrees. Then
DOM(R(T)) = DOM(A)xP(DOM(R(T_1)))x...xP(DOM(R(T_n))).

Where P(D) denotes the powerset of the set D, x denotes the cartesian product.

A scheme tree, a nested relation scheme and its instance are given in Example 1.1. Since there is one-to-one correspondence between a scheme tree and the corresponding nested relation scheme, in the rest of the paper, we use scheme trees to denote nested relation schemes.

3. Properties of Keys and Conflict Free MVD´s

Conflict free MVD´s and their properties have been discussed in [BFMY, L2, Sc]. In this section, we present some properties of conflict free MVD´s which are also reduced MVD´s. The results presented in this section will be used in the rest of the paper. Let M be a set of reduced MVD´s which is also a minimal cover. (Hereafter, M is used to denote such a set of MVD´s unless stated explicitly.) The following two lemmas state some useful properties of M.

Lemma 3.1 [OY]. Let M be a set of MVD´s, X be a key of M, and X´ ⊂ X. Then there exists a V ∈ DEP(X´) such that

(1) X ⊂ X´V,

(2) for each W ∈ RDEP(X), W ⊆ V,

(3) X´→→ V is left-reduced, and if X´ is a key, then V ∈ RDEP(X´).

The proof of the following lemma is straightforward, and is omitted.

Lemma 3.2. Let X, Y be sets of attributes, $V_X \in$ DEP(X), and $V_Y \in$ DEP(Y).
If $V_X \cap$ Y = ∅, $V_Y \cap$ X = ∅, and $V_X \cap V_Y \neq \emptyset$, then $V_X = V_Y$.

If M is conflict free then it satisfies several other properties, as stated by the following lemmas.

Lemma 3.3. Let M be a conflict free set of MVD´s, X, Y, and V are sets of attributes. Then

(1) If X →→ V and Y →→ V, which are left- and right-reduced, then X = Y.

(2) If X is a key, V ∈ RDEP(X), and Y is an essential key such that Y ∩ V ≠ ∅, then Y splits XV.

(3) If X is a nonessential key, then X has only one reduced dependent, and for any nonreduced dependent V of X,

there is an essential key X´ ⊂ X such that X´ →→ V.

(4) If X is a nonessential key then X is the union of some essential keys.

Proof: (1) Directly follows Lemma 2.4

(2) Assume not, i.e. Y does not split XV. It implies that there exists a $V_0 \in$ DEP(Y) such that XV ⊆ YV_0, and M is conflict free, so Y ⊆ XV. Let DEP(Y) = {V_0, V_1, ..., V_n}. The fact that X →→ V, XV ⊇ Y, and $V_i \in$ DEP(Y), for i ≥ 1, implies that X→→V_i − V, but YV_0 ⊇ XV => V ∩ V_i = ∅, therefore, $V_i \in$ DEP(X), for i ≥ 1. By Lemma 2.4, $V_i \in$ DEP(X ∩ Y), for i ≥ 1. But Y ∩ V ≠ ∅ implies (X ∩ Y)) ⊂ Y, hence, for i ≥ 1, Y →→ V_i is left-reducible. Moreover, X ∩ Y →→ V_1...V_n, i.e. X ∩ Y →→ V_0(Y − X), by M1. Thus Y →→ V_0 is transferable. For each dependent V_i, Y →→ V_i is nonreduced, which contradicts that Y is a key.

(3) It was shown in [OY] that if X is a key with more than one reduced dependents then X is an essential key. Since any key must have at least one reduced dependent, if X is a nonessential key then it has only one reduced dependent W_1. Let W_2 be a nonreduced dependent of X. By Lemma 2.1, there exists a reduced MVD X´ →→ HW_2, where X´ ⊂ X and H ⊆ (X − X´). By Lemma 3.1, there is a V ∈ RDEP(X´) such that (X − X´) ⊆ V and W_1 ⊆ V. It follows that V ≠ W_2 and H = ∅, i.e. $W_2 \in$ REDP(X´) Since X´ has two reduced dependents V and W_2, so X´ is an essential key.

(4) It directly follows Proposition 2.2.

Lemma 3.4. Let M be a conflict free set of MVD´s, X, Y, V be sets of attributes. Then

(1) If X splits V then there is an essential key X´ ⊆ X such that X´ splits V.

(2) If M does not split X, and X →→V is a left- and right-reduced MVD in M^+, then, for any essential key Y that splits XV, Y ∩ V ≠ ∅.

Proof: (1) X splits V, then there exist two dependents W_1 and W_2 of X such that W_1 ∩ V ≠ ∅ and W_2 ∩ V ≠ ∅. Assume X is not an essential key, otherwise, X´ = X . By Lemma 3.3 (3), at least, one of W_1 and W_2 is nonreduced dependent of X, say W_1, i.e. X →→ W_1 is nonreduced. By Lemma 2.1, there is a reduced MVD X´ →→ HW_1 such that X´ ⊂ X and H ⊆ (X − X´). Since W_2 ∩ V ≠ ∅ and W_2 ⊆ (U − XW_1), there is W_2´ ∈ DEP(X´) such that W_2´ ∩ V ≠ ∅ and V − X´W_2´ ≠ ∅. Similarly, assume X´ is a nonessential key, by Lemma 3.3 (3), there is an essential key X" ⊂ X´ such that X" →→ W_2´, i.e. X" splits V.

(2) Assume there is an essential key Y such that Y splits XV and Y ∩ V = ∅. Y ∩ V = ∅ implies that Y does not split V, otherwise, X splits V, which is a contradiction. Since M does not split X, Y does not split X also. Thus, Y splits XV implies that there are W_1, W_2 in DEP(Y) such that X ⊆ YW_1 and V ⊆ YW_2, and X ⊄ Y, i.e. X ∩ Y ⊂ X. Since Y→→YW_1, YW_1 ⊇ X, and X→→V, Y→→(V − YW_1), i.e. Y→→V. It follows that V = W_2, and by Lemma 2.4, X ∩ Y→→V, which contradicts that X→→V is left-reduced. □

4. Normal Scheme Tree and Forest

In this section, we first present some properties on a scheme tree, and then by utilizing these properties, we define a normal scheme tree, (i.e. the scheme of a nested normal form relation). Let U be a universal scheme, T be a scheme tree over U, and e = (u, v) be an edge of T. We use the following notations throughout the paper.

F(v): the father of v, i.e. u.
A(v): the union of labels of all ancestors of v, including v.
D(v): the union of labels of all descendants of v, including v.
S(T): the union of labels of all nodes of T.
M(e): the MVD represented by the edge e, i.e. MD(A(u), D(v), S(T)),
which is the MVD A(u)→→D(v) on S(T).

MVD(T): the set of MVD´s represented by edges of T.

By definition, any instance of a nested relation R satisfies the MVD(T), where T is the scheme tree of R. Given a set M of MVD´s over U, and a scheme tree T with attributes S(T) $\subseteq$ U, T is called a *scheme tree with respect to M* if M $\models$ MVD(T).

Definition 4.1. Let T be a scheme tree, and u_1, ..., u_n be all the leaf nodes of T. Then the *path set* of T, P(T) = {A(u_1), ..., A(u_n)}. Note that, for a leaf node u, A(u) is the union of labels of all the nodes in the path from the root of T to u in T.

In Example 1.1, MVD(T_R) = {COURSE $\to\to$ TEXT; COURSE $\to\to$ {SECTION, DAY, GRADER}; {COURSE, SECTION} $\to\to$ DAY $|$ GRADER}, and P(T_R) = {(COURSE, TEXT), (COURSE, SECTION, DAY), (COURSE, SECTION, GRADER)}. The following proposition gives some properties of a scheme tree, the proof of which is straightforward using the results in [BFMY].

Proposition 4.1 If T is a scheme tree, then

(1) P(T) is an acyclic database scheme,

(2) MVD(T)$\Leftrightarrow$*(P(T)), and

(3) MVD(T) is conflict free.

Let M be a set of MVD´s over U, and T be a scheme tree over U with respect to M, i.e. M implies MVD(T). If we consider P(T) as a decomposition of U with respect to M, then we must have M implies *(P(T)). The above proposition shows that, P(T) of a scheme tree T with respect to M is a decomposition of U with respect to M.

4.1 Normal Scheme Tree

In this section, we give properties that characterize a normal scheme tree. Let T be a scheme tree with respect to M, where S(T) $\subseteq$ U, and (u, v) be an edge in T. Assume there is a key X of M such that there exists Z $\in$ DEP(X) and D(v) = Z $\cap$ S(T). Then, v is said to be a *partial redundant* in T with respect to X if X $\subset$ A(u). Note that, in this case, there is some redundancy in the scheme tree since the MD(A(u), D(v), S(T)) represented by the edge (u, v) can be derived by augmentation from MD(X, D(v), S(T)), i.e. MD(X, D(v), S(T)) is a *partial dependency* in S(T). Similarly, if there exist some sibling nodes v_1,...,v_n of v in T such that W = $\bigcup_{i=1}^{n}$ D(v_i), X $\subset$ A(u)W, and M $\not\models$ MD(XW, A(u), S(T)), then v is said to be *transitive redundant* with respect to X in T. In this case, MD(X, D(v), S(T)) is said to be a *transitive dependency* in S(T). Note that, if v is partial or transitive redundant in T with respect to X, then MD(X, D(v), S(T)) is a projection [BK1] of a right-reduced MVD over U, which is implied by M.

Example 4.1 Let U = ABCDEFGH and M = {A $\to\to$ B, AC $\to\to$ EFG, BE $\to\to$ AF $|$ D}. Keys of M are A, AC, BE, AE, AEC and BEC. Consider the following scheme tree T.

T: A .

B . . C

E . .D . H

F.   .G

F is partial redundant with respect to AE, since for the edge (E, F) in T, M $\models$ (AE $\rightarrow\!\!\!\rightarrow$ F) and AE $\subset$ A(E) = ACE. D is transitive redundant with respect to AE, since for the edge (C, D), M $\models$ (AE $\rightarrow\!\!\!\rightarrow$ D) and AE $\subset$ ACEFG but M $\not\models$ (AEFG $\rightarrow\!\!\!\rightarrow$AC).

A normal scheme tree does not have partial or transitive redundant nodes. The following example shows two scheme trees without any partial or transitive nodes but one is preferred to the other.

Example 4.2. Let U = ABCEF M = {AB $\rightarrow\!\!\!\rightarrow$ CE, BC $\rightarrow\!\!\!\rightarrow$ F}. AB and BC are both essential keys. Consider the following scheme trees.

T_1:   AB . T_2:   AB

F.   .C   F.   .E

.E   .C

T_1 and T_2 have no redundant nodes and M $\models$ MVD(T_1), M $\models$ MVD(T_2). However, the essential key BC is separated by E in T_2 while no essential key is separated in T_1. In this example, T_1 is a normal scheme tree while T_2 is not. The reason why T_1 is better than T_2 will be clear when we consider scheme trees which represent MVD´s embedded in U.   □

In order to avoid anomalies in scheme trees (such as in T_2 of above example), we define a set of attributes called a fundamental key. Let M be a set of MVD´*s** on U and V $\subseteq$ U. The set of *fundamental keys* on V, denoted FK(V), is defined as follows.

$$FK(V) = \{V \cap X \mid X \in LHS(M) \text{ and } V \cap X \neq \emptyset, \text{ and no } Y \in LHS(M) \text{ s.t. } X \cap V \supset Y \cap V \neq \emptyset\}.$$

Obviously, FK(V) = $\emptyset$ iff for each X $\in$ LHS(M), X $\cap$ V = $\emptyset$. Now we can define a normal scheme tree.

Definition 4.2. A scheme tree T is said to be *normal* with respect to a set of MVD´s M, if

(C1) M $\models$ MVD(T),

(C2) for each edge (u, v) of T, MD(A(u), D(v), S(T)) is left and right-reduced,

(C3) for each node u in T, there is no key X of M such that u is transitive redundant with respect to X, and

(C4) the root of T is a key, and for each other node u in T, if FK(D(u))$\neq\emptyset$ then u $\in$ FK(D(u)).

In the above conditions, C1 is implied since T is a scheme tree with respect to M, C2 excludes partial dependencies, and C3 excludes transitive dependencies in T. The condition C4, refers to the situation in Example 4.2. Note that, T_1 in this example satisfies C4 but T_2 does not since for the node E, D(E) = CE, FK(CE) = {C} but E $\notin$ FK(CE). The proposition 4.3, below, shows that the path set P(T) of a normal scheme tree T with respect to a conflict free set of MVD´s M is in W4NF (i.e. in 4NF if we consider only those MVD´s that are not embedded). First, we show the following lemma.

* Note that, we use M to denote minimal cover of a set of MVD´s.

Lemma 4.2. Let T be a normal scheme tree with respect to a set M of MVD´s, P be a path in T, and L be the leaf of P. If there is a key $X \subset P$ such that X splits P, then $L \subseteq X$.

Proof: Let $X \subset P$ be a key of M such that $L \not\subseteq X$. If $L \cap X \neq \emptyset$, then $FK(D(L)) = FK(L) \neq \emptyset$ and $L \notin FK(L)$, which contradicts to C4 in the definition 4.2. Therefore, we have $L \cap X = \emptyset$. Let $A = A(F(L))$, then $X \subseteq A$, and since T is a normal scheme tree, MD(A, L, S(T)) is left- and right-reduced, so there is a $V \in DEP(X)$ such that $V \supseteq L$. Let $V_A = V \cap A$, then $XV_A \twoheadrightarrow XV_A\overline{V}$, where $\overline{V} = U - XV$, i.e. $MD(XV_A, XV_A(\overline{V} \cap S(T)), S(T))$. Since MD(A, L, S(T)) and $XV_A(\overline{V} \cap S(T)) \supseteq A$, we have $MD(XV_A(\overline{V} \cap S(T)), L, S(T))$. Therefore, by the transitivity rule, we get $MD(XV_A, L - XV_A(\overline{V} \cap S(T)), S(T))$, i.e. $MD(XV_A, L, S(T))$. If X splits P, then $XV_A \subset A$, i.e. MD(A, L, S(T)) is left-reducible. A contradiction. □

Proposition 4.3. If M is conflict free then the path set P(T) of a normal scheme tree with respect to M is in W4NF.

Proof: By Proposition 4.1, $M \models *(P(T))$, therefore, we need only show that for each P in P(T), P is in W4NF. Assume not. Then there is a key X and a $W \in RDEP(X)$ such that $XW \subset P$. Let u be the lowest node in P such that $u \cap W \neq \emptyset$, and L be the leaf of P, by Lemma 4.2, $L \subseteq X$, so $u \neq L$. By Lemma 3.3 (4), there is an essential key $X' \subseteq X$ such that $L \cap X' \neq \emptyset$, i.e. $D(u) \cap X' \neq \emptyset$. Therefore, $FK(D(u)) \neq \emptyset$, that means there is an essential key Y such that $u = D(u) \cap Y \in FK(D(u))$. $W \cap u \neq \emptyset$ and $u \subseteq Y$ imply that $Y \cap W \neq \emptyset$. Hence, by Lemma 3.3 (2), Y splits $XW \subset P$, and, by Lemma 4.2, $L \subseteq Y$, which contradicts that $u = D(u) \cap Y$. □

4.2 Normal Scheme Forest

The above discussions show that, if T is a normal scheme tree over U with respect to M then T is a good representation for instances of U satisfying M. However, it is not always possible to obtain such a normal scheme tree, even when M is conflict free. Instead, given U and M, we can obtain a set of normal scheme trees.

Example 4.3 For U = ABCDEFG and $M = \{A \twoheadrightarrow B, AC \twoheadrightarrow EFG, BE \twoheadrightarrow AF \mid D\}$, given in Example 4.1 there is no single normal scheme tree. However, $F = \{T_1, T_2\}$ is a set of normal scheme trees over U with respect to M, where

T_1: A.

B. .C

E. .H

G.

T_2: AE.

F. .D

Definition 4.3. Let M be a set of MVD´s over U, and $F = \{T_1, ..., T_n\}$ be a set of normal scheme trees. F is a *normal scheme forest* (NSF) with respect to M if

(1) $S(F) = \bigcup_{i=1}^{n} S(T_i) = U$, and

(2) $M \models *(S(T_1), ..., S(T_n))$.

In Example 4.3, $F = \{T_1, T_2\}$ is an NSF. Note that, any 4NF relation scheme can be structured in the form of a normal scheme tree, which consists of a single path. Thus, any 4NF decomposition of U with respect to M is a trivial NSF decomposition. However, in our context, we are interested in finding an NSF of scheme trees which are not necessarily single paths. In the next section, we present a decomposition algorithm to obtain a normal scheme forest with respect to a given set M of MVD´s over U. Properties of a normal scheme forest are given in section 6.

5. Decomposition Algorithm

The decomposition starts by decomposing U into a scheme tree T and then decomposes T into a set of normal scheme trees $T_1, ..., T_n$. Informally, an intermediate scheme tree manipulated by the algorithm, is called a *semi-normal tree.*

Definition 5.1 A scheme tree T is a *semi-normal tree* with respect to M, if

(1) M $\models$ MVD(T)

(2) For each edge (u, v) in T, MD(A(u), D(v), S(T)) is right-reduced.

(3) For each inner node* u in T, u $\in$ FK(D(u)).

By comparing the above definition with the definition 4.2, it follows that a normal scheme tree is a semi-normal tree, but the reverse is not true, as illustrated by the following example.

Example 5.1 Let U = ABCDE, M = {A $\rightarrow\!\!\rightarrow$ B, C $\rightarrow\!\!\rightarrow$ D, C $\rightarrow\!\!\rightarrow$ E}.

T_1: A.

B. .CDE

T_2: A.

B. .C

D. . E

Scheme trees T_1 and T_2 are both semi-normal, but not normal scheme trees. T_1 is not a normal scheme tree since for the node CDE, D(CDE) = CDE, FK(D(CDE)) $\neq \emptyset$, but CDE is not in FK(D(CDE)). T_2 is not a normal scheme tree since the MVD AC$\rightarrow\!\!\rightarrow$D represented by the edge (C, D) is not left reduced.

Let T be a semi-normal scheme tree with respect to M, and V be a leaf of T. Then V is said to be *decomposable* if there is a V_0 in FK(V) such that $V_0 \subset$ V. (In fact, if V is decomposable then FK(V)$\neq\emptyset$ and for any V_0 in FK(V), $V_0 \subset$ V). A semi-normal tree is *nondecomposable* if no leaf in T is decomposable. A node in semi-normal tree T is *redundant* if it is partial redundant or transitive redundant with respect to a key X of M. A semi-normal tree is *nonredundant* if it has no redundant nodes. In Example 5.1, T_1 is decomposable since FK(CDE) = {C}$\subset$CDE, but nonredundant. T_2 is nondecomposable but redundant since nodes D and E are both partial redundant with respect to the key C. The following proposition directly follows from definitions.

Proposition 5.1. If a semi-normal tree T with respect to M is nonredundant, nondecomposable, and the root of T is a key, then T is a normal scheme tree.

5.1. Basic Procedures

Given a set M of MVD´s over U, a single node, labeled U, is a semi-normal tree. In this section, we give two procedures; the first one decomposes a semi-normal tree into a nondecomposable semi-normal tree; and the second

* An inner node is a node which is neither the root nor a leaf in T.

one constructs two semi-normal trees from the one given by eliminating some redundancies. Let T be a semi-normal tree with respect to M, and V be a leaf of T such that V is decomposable. The procedure DECOMP1(V) manipulates T by decomposing V into a set of nodes such that T remains to be a semi-normal tree.

Procedure DECOMP1(V);
begin
  Select V_0 from FK(V);
  Let Z = A(F(V))V_0 and W = $\{w_i \mid \exists V_i \in$ DEP(Z) and $w_i = V_i \cap V \neq \emptyset\}$
  Change V into V_0 in T, and
  for each $w_i \varepsilon W$, attach w_i as a son of V_0
end;

It is easy to see that if V is a decomposable leaf of a semi-normal tree T, then DECOMP1(V) returns a semi-normal tree in which V has one or more descendants. The following is a recursive version of DECOMP1(V), with an enhancement. That is, in step (2) of the procedure DECOMP(V), if there is a V_i satisfying the condition, then V_i is selected as V_0. The significance of this selection will be explained in Section 6. Since a tree with a single node U is a semi-normal tree, and in a semi-normal tree, only the leaves are decomposable, DECOMP(U) returns a semi-normal tree which is nondecomposable.

Procedure DECOMP(V);
begin
  If V is decomposable *then*
  begin {*steps (1) and (2) select V_0 from FK(V)*}
    (1) V_0 is any element in FK(V);
    (2) *If* V is not the root of the scheme tree T
        then let A´ ⊆ A(F(V)) such that MD(A´, V, S(T)) is left-reduced
        else A´ = ∅;
      If $\exists V_i \in$ FK(V) such that for any essential key X of M,
      where X∩(V − V_i)≠∅, X does *not* split A´V_i *then* $V_0 = V_i$;
    (3) *If* V is not the root of the scheme tree T
        then Z = A(F(V))V_0
        else Z = V_0;
      W = $\{W_i \mid \exists V_i \in$ DEP(Z) and $W_i = V_i \cap V \neq \emptyset\}$.
    (4) Change V into V_0 in T, and
      for each $W_i \in$ W *do*
      begin attach W_i as a son of V_0 and DECOMP(W_i) *end*
  end
end;

Let T be a semi-normal tree with respect to M, and X be a key of M such that X ⊂ S(T). Thus, we can find the set V = $\{V_1, ..., V_n\}$ of all nodes which are partial or transitive redundant with respect to X in T. The following procedure modifies T, and constructs a new tree T_{new}, for any given X and V, such that T_{new} and the modified version of T are both nondecomposable semi-normal trees which do not have any redundant node with respect to X.

Procedure NEWTREE(X, V, T, T_{new})
begin
Let T_{new} be a tree with only one node labeled X;
for each V_i in V *do*
begin delete D(V_i) from T, and
attach D(V_i) as a subtree of X in T_{new} *end*
end;

To show the correctness of NEWTREE(X, V, T, T_{new}), we need the following lemmas.

Lemma 5.2. Let W = $W_0 W_1 ... W_k$ be a dependent of both X and Y with respect to M. Then, if $W_i \in$ DEP(XW_0) then $W_i \in$ DEP(YW_0) for $1 \le i \le n$, where W_i, X, and Y are subsets of U.

Proof: Let $\overline{W}$ = U – W, then X$W_0 \subseteq \overline{W}W_0$. Y $\to\!\!\to$ W implies Y $\to\!\!\to \overline{W}$, therefore, Y$W_0 \to\!\!\to \overline{W}W_0$. Since X$W_0 \to\!\!\to W_i$, $\overline{W}W_0 \to\!\!\to W_i$, thus, Y$W_0 \to\!\!\to W_i$, for i=1,...k. If Y$W_0 \to\!\!\to W_i$ is right-reducible, for some $1 \le i \le k$, then, similarly, X$W_0 \to\!\!\to W_i$ is right-reducible, which is a contradiction. □

Lemma 5.3. If T is a nondecomposable semi normal tree, then both T and T_{new} returned by NEWTREE(X, V, T, T_{new}) are both nondecomposable semi normal trees.

Proof: Let T´ be T before it was modified. Obviously, T is a nondecomposable semi normal tree, since T´ is a nondecomposable semi normal tree. Lemma 5.2 guarantees that T_{new} satisfies the first two conditions of a semi normal tree. The root of T_{new} is X, which is a key of M, and each inner node and leaf in T_{new} appears the same as it does in T´. Thus, it follows that T_{new} is a nondecomposable semi normal tree.

5.2. Algorithm

Given a set M of MVD´s on U, a normal scheme forest can be constructed as follows: Let F = {T_1} where T_1 is the nondecomposable semi-normal tree constructed by DECOMP(U). For each key X of M, find the set V of all nodes in some $T_i \in$ F which are redundant with respect to X, and call NEWTREE (V, X, T_i, T_j) to construct a new tree T_j such that both T_i and T_j are nondecomposable semi-normal trees without any redundant nodes with respect to X. While considering the keys of M we use a *p-ordering* [L1] (i.e. an ordering X_1, ..., X_n where $X_i \subseteq X_j$ only if $i \le j$ for $1 \le i, j \le n$. Obviously, p-ordering of a set of keys is not unique). A p-ordering of keys, together with Lemma 3.1 (1), guarantees that before calling NEWTREE (X, V, T_i, T_j), for any key X_i of M, if there is a T_i such that X$\subset$S(T_i), then T_i is the only tree containing X among the trees constructed so far. Let F = {T_1, ..., T_k}, after all keys of M are processed as described above. Then each $T_i \in$ F is a nondecomposable semi-normal tree without any redundant nodes. Thus, by Proposition 5.1, F is a set of normal scheme trees with respect to M. Since each attribute in U is in at least one tree of F, S(F) = U. In order to show, M $\models$ *(S(T_1), ..., S(T_k)), consider the case with k = 2, where T_2 is constructed from T_1 by NEWTREE (X, V, T_1, T_2). Let T_1´ denote the only scheme tree, i.e. T_1, before T_2 is constructed. Then, by construction, S(T_1)$\cap$S(T_2) = X. Since S(T_2) – S(T_1) = W = $\bigcup_{i=1}^{n}$D(W_i), where V = {W_1, ..., W_n}, and X $\to\!\!\to$ W in S(T_1´) = U, M $\models$ *(S(T_1), S(T_2)). Since {T_1, ..., T_n} is obtained by repeated applications of this procedure, it follows that M $\models$ *(S(T_1), ..., S(T_k)). Thus, Algorithm D, correctly produces a normal scheme forest F with respect to M.

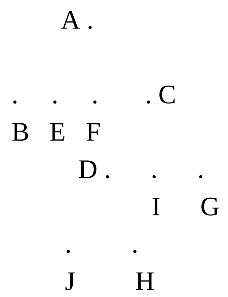
Algorithm D

Input: A set of attributes U and a set of MVD´s M on U.
Output: A normal scheme forest F = {T_1, ..., T_j}.
(1) Calculate all keys of M [OY].
(2) T_1 := DECOMP(U); j := 1;
(3) Let X_1, ..., X_n be a p-ordering of keys of M.
    for i := 1 to n *do*
        begin if $X_i \subset$ S(T_k) for some $1 \le k \le j$ *then*
            begin V := $\emptyset$
                for each node W in T_k such that $\exists$Z $\in$ DEP(X_i) *and* D(W) = Z $\cap$ S(T_k) *do*
                    if W is redundant with respect to X_i *or* (Z $\in$ EDEP(X_i) *and* Z $\notin$EDEP(A(F(W)))
                        then V:=V $\cup$ {W};
                    if V $\neq \emptyset$ *then begin* j := j + 1; *and* NEWTREE(X_i, V, T_k, T_j) *end*
            end
        end;
4. Output F = {T_1, ..., T_j};

Note that, in step (3), the set V constructed for the key X_i may contain a node W $\in$ T_k where W is not redundant in T_k with respect to X_i. In this case, W is the part of an essential dependent of X_i in S(T_k), but not of A(F(W)), and it seems better to attach W as a descendant of X_i in T_j even though W is not redundant in T_k with respect to X_i. The correctness of Algorithm D, with this enhancement, follows from the discussions above. The following example, illustrates Algorithm D.

Example 5.2. Let U = ABCDEFGHIJ, and M = {A $\twoheadrightarrow$ B, B $\twoheadrightarrow$ E $|$ F, AC $\twoheadrightarrow$ DJH, AJ $\twoheadrightarrow$ H, AD $\twoheadrightarrow$ J $|$ G}. M is a minimal cover. The set of essential keys of M is LHS(M) = {A, B, AC, AJ, AD}, and the only nonessential key is ACJ. Suppose we choose a p-ordering A, B, AC, AD, AJ, ACJ of keys. The scheme tree T_1 obtained in step (1) is

T_1



Since E and F are redundant in T_1 with respect to the key B, NEWTREE(B, {E, F}, T_1, T_2) modifies T_1 and constructs T_2 as follows:

T_1: A. $\quad$ T_2: B.

B. .C $\quad$ E. . F

D . . .
I G

J . .H

Similarly, after all the keys are considered in step (3), the resulting normal scheme forest is F = $\{T_1, T_2, T_3, T_4\}$, where

T_1: A. $\quad$ T_2: B. $\quad$ T_3: $\quad$ T_4:

B. .C $\quad$ E. .F $\quad$ AD. $\quad$ AJ.

D . . I $\quad$ J. . G $\quad$ H.

□

In this example, the normal scheme forest produced by the algorithm is the same if keys AD and AJ are interchanged in the p-ordering used in step (3). However, T_1 produced in step (1) depends on the ordering of essential keys in FK(V) used in decomposing a node V of an intermediate semi-normal tree. In the next section, we present properties of a normal scheme forest produced by Algorithm D.

6. Properties of Decomposition

Given a set of MVD´s M over U, let F = $\{T_1, ..., T_n\}$ be an NSF decomposition obtained by Algorithm D. By definition, for each $T_i \in$ F, M $\models$ MVD(T_i). However, there may be some other MVD´s implied by M but not represented by a T_i in F. To capture some of these dependencies we define a *construction graph* J(F) for F as a connected graph (F, E), where the normal scheme forest F is the set of vertices and E is an ordered set of directed edges. An edge e = $(T_i, T_k) \in$ E iff T_k is constructed from T_i by the procedure NEWTREE (X, V, T_i, T_k) (in step 3), and the label of e is the root of T_k, i.e. L(e) = X. The order of edges is determined by the order T_i´s are constructed, i.e. for $e_i = (T_s, T_k)$ and $e_j = (T_x, T_m)$, i<j iff T_k is constructed before T_m.

Example 6.1. For F = $\{T_1, T_2, T_3, T_4\}$ in Example 5.2, the construction graph J(F) is as follows.

B

T_1. $\quad$ .T_2

e_1

AD e_2

AJ

T_3. $\quad$ .T_4

e_3

Obviously, for any F constructed by Algorithm D, J(F) is a tree. □

For the MVD´s represented by edges of J(F), we need some notations. Let DS(T_i), where $T_i \in$ F, denote the set of all attributes in trees that are descendants of T_i in F, including T_i. For each $e_i \in$ F, let AT(e_i) denote the set of all attributes in the connected component containing e_i after all edges e_j where j<i are deleted from J(F). The MVD represented by an edge e = (T_i, T_k) is MVD(e) = MD(L(e), DS(T_k) − L(e), AT(e)). The set of MVD´s represented by F = {T_1, ..., T_n} is MVD(F) = { MVD(e_i) | $e_i \in$ J(F)} $\cup$ ($\bigcup_{i=1}^{n}$ MVD(T_i)).

Example 6.2. For the F and J(F) given in Examples 5.2 and 4.4,

DS(T_1) = AT(T_1) = U,
DS(T_2) = S(T_2) = BEF, and
DS(T_3) = S(T_3) $\cup$ S(T_4) = ADJGH.

The MVD´s represented by edges e_1 = (T_1, T_2), e_2 = (T_1, T_3), and e_3 = (T_3, T_4) are as follows:

MVD(e_1) = MD(B, EF, U) i.e. B $\twoheadrightarrow$ EF
MVD(e_2) = MD(AD, JGH, ABCDGHIJ), and
MVD(e_3) = MD(AJ, H, ADJGH).

Note that, MVD(e_1) is a full MVD´ which is nonreduced. However, B $\twoheadrightarrow$ EF together with the embedded MVD MD(B, E, EF) in T_1, implies that B $\twoheadrightarrow$ E also holds in U. Similarly, MVD(e_2) and MVD(e_3) are embedded MVD´s. Obviously, in this example M $\models$ MVD(e_i) for i = 1, 2, 3. It is shown below that M $\models$ MVD(F) for any F obtained by Algorithm D.

The path set P(F) of an NSF decomposition F is defined as the union of path sets P(T_i) for T_i´s in F. That is, for the F in Example 5.2, P(F) = {AB, ACD, ACI, BE, BF, ADJ, ADG, AJH}. Using Proposition 4.1, the proof of the following proposition is straightforward.

Proposition 6.1. If F is an NSF decomposition of U with respect to M obtained by Algorithm D, then M $\models$ MVD(F) $\models$ *(P(F)).

Thus, for any set of MVD´s M over U, P(F) is a lossless join [Ul] decomposition of U with respect to M. A significant property of an NSF decomposition F with respect to M is that it explicitly represents a set of MVD´s implied by M, which in turn implies a JD over U. (Note that, *(P(F)) is not necessarily an acyclic JD.) However, F does not always preserve M, i.e. we do not have MVD(F) $\models$ M.

Example 6.3. Let U = ABCDGH, and M = {A $\twoheadrightarrow$ G, B $\twoheadrightarrow$ H, GH $\twoheadrightarrow$ C | D}, as in Example 2.1. M is not conflict free since both A and B split GH. An NSF decomposition of U with respect to M with the p-ordering A, B, GH, AB, AH, BG of keys is F = {T_1, T_2} , where

T_1 A. T_2
.B J(F):
G. . B . .
.H T_1 T_2
C. . D

P(F) = {AG, ABC, ABD, BH}. By proposition 6.1, MVD(F) $\models$ *P(F)). In this example, *(P(F)) is an acyclic JD, and MVD(F) $\Leftrightarrow$ *(P(F)). However, since M is not conflict free, MVD(F) $\not\models$ M, e.g. GH $\twoheadrightarrow$ C | D is not represented by F, although AB $\rightarrow\!\!\!\rightarrow$ C ∣ D ∣ G ∣ H implied by M, but not by M − {GH $\rightarrow\!\!\!\rightarrow$ C ∣ D}, is also implied by MVD(F). □

Another significance of this example is that P(F) is a 4NF decomposition of U with respect to M, and it is easy to verify that for this example, any F´ produced by Algorithm D satisfies P(F) = P(F´) This is because of the enhancement in Step (2) of the procedure DECOMP(V). That is, since GH is split by A and B, the essential key GH is never used to decompose a node V which contains A or B. Lien [L1] also observed that his decomposition algorithm produces a 4NF decomposition for this example if a p-ordering in which GH precedes both A and B is used; and fails to produce a 4NF decomposition otherwise. This is because, in [L1] only the keys in LHS(M), but not all the keys [OY], are used for the decomposition.

In the above example, P(F) = {AG, ABC, ABD, BH} is a lossless 4NF decomposition of U with respect to M, but it is not dependency preserving. It is easy to see that P(F) is not an SFNF decomposition of U with respect to M [BK2], since ABC (and also ABD) are split by GH $\rightarrow\!\!\!\rightarrow$ C ∣ D. The next proposition shows that if M is conflict free, then for any F produced by Algorithm D, P(F) is a unique dependency preserving SFNF decomposition of U.

Proposition 6.2 Let F be an NSF decomposition of U with respect to M obtained by Algorithm D, and M be a conflict free set of MVD´s over U. Then

(1) P(F) is a unique SFNF decomposition of U,

(2) M ⇔ MVD(F) ⇔ *(P(F)), and

(3) P(F) is acyclic.

In order to show this proposition we need some lemmas. Algorithm D constructs F = {T_1, T_2, ...,T_J} one after another by calling NEWTREE. Let T_i´ be the T_i in F when it was just constructed in the algorithm. Obviously, T_i´ may not be the same as T_i, since some nodes may be deleted later. By Lemma 5.3, T_i´, 1 ≤ i ≤ J, is a nondecomposable semi normal tree. Thus, T_1´ is the nondecomposable semi normal tree obtained from DECOMP(U) in step 2 of the algorithm, and for any inner or leaf node v in T_i, 1 ≤ i ≤ J, there is a corresponding node v_1 in T_1´ such that v_1 = v.

Lemma 6.3. Let F = {T_1, T_2,...T_J} be an NSF decomposition obtained by Algorithm D, e = <u, v> be an edge in T_i, 1 ≤ i ≤ J, and v_1 be the node corresponding to v in T_1´, A(u) and D´(v_1) be defined as A(u) and D(v_1) in T_i and T_1´ respectively. Then A(u) $\rightarrow\!\!\!\rightarrow$ D´(v_1), and it is a left- and right-reduced MVD.

Proof: There is a sequence of trees T_{k1}, T_{k2}, ..., T_{kn} in F such that T_{k1} = T_1, T_{kn} = T_i, and T_{kj} is constructed from $T_{k(j-1)}$, for 2 ≤ j ≤ n. For 1 ≤ j ≤ n, let T_j´ be the T_{kj} when it was just constructed, and e_j = <u_j, v_j> be an edge in T_j´ such that v_j is the node corresponding to v, i.e. v_j = v. A(u_j), D(v_j), A´(u_j), and D´(v_j) are defined in T_{kj} and T_j´ respectively. Obviously, A(u_j) = A´(u_j), D(v_j) ⊆ D(v_j), and D(v_j) = D´(v_j) ∩ S(T_{kj}). First we show that, for any 1 ≤ j ≤ n, A(u_j) $\rightarrow\!\!\!\rightarrow$ D´(v_1) is right-reduced by induction on j.

Basis: j = 1, it is trivial.

Induction step: Assume A(u_{n-1}) $\rightarrow\!\!\!\rightarrow$ D´(v_1) is right-reduced. Let $\tilde{T}$ be the $T_{k(n-1)}$ just before T_n´ was constructed by calling NEWTREE(X, V, $\tilde{T}$, T_n´). Thus, e_{n-1} = <u_{n-1}, v_{n-1}> is also in $\tilde{T}$. Let $\tilde{D}$(v_{n-1}) be D(v_{n-1}) defined in $\tilde{T}$, then $\tilde{D}$(v_{n-1}) = D(v_n) ⊆ D´(v_{n-1}). By Lemma 5.3, $\tilde{T}$ is a semi normal tree, so MD(A(u_{n-1}), $\tilde{D}$(v_{n-1}), S($\tilde{T}$)) is right-reduced. Since T_n´ is constructed from $\tilde{T}$, by Lemma 5.2, MD(A(u_n), $\tilde{D}$(v_{n-1}), S($\tilde{T}$)) is also right-reduced. These two MVD´s are projections of some full MVD´s implied by M on S($\tilde{T}$), i.e. there exist D_{n-1} ∈ DEP(A(u_{n-1})) and D_n ∈ DEP(A(u_n)) such that $\tilde{D}$(v_{n-1}) = D_{n-1} ∩ S($\tilde{T}$) and $\tilde{D}$(v_{n-1}) = D_n ∩ S($\tilde{T}$). By Lemma

3.2, $D_{n-1} = D_n$. By the induction assumption, $D'(v_1) \in$ DEP(A(u_{n-1})) and $D_{n-1} \cap D'(v_{n-1}) \neq \emptyset$, thus $D'(v_1) = D_{n-1} = D_n$. That is $D'(v_1) \in$ DEP(A(u_n)).

T_{kn} is a normal scheme tree, so MD(A(u_n), D(v_n), S(T_{kn})) is left- and right-reduced, and obviously, this MVD is the projection of the MVD A(u_n) $\rightarrow\!\!\rightarrow$ $D'(v_1)$ on S(T_{kn}). It follows that A(u_n) $\rightarrow\!\!\rightarrow$ $D'(v_1)$ is left- and right-reduced. □

Let M be a conflict free set of MVD´s, Z, V be sets of attributes such that Z$\rightarrow\!\!\rightarrow$V is left- and right-reduced, M does not split Z and FK(V) $\neq \emptyset$, and K = $\{Y_i \mid Y_i$ is an essential key and $Y_i \cap V \neq \emptyset\}$. A binary relation "<" is defined on K as follows.

$Y_i < Y_j$ iff ZY_i is split by Y_j but not by any $Y_j' \subset Y_j$ in K.

Then we have the following lemma.

Lemma 6.4. "<" is irreflexive, antisymmetric, and transitive.

Proof: (1) "<" is irreflexive, i.e. $Y_i \not< Y_i$. Y_i does not split Z, so Y_i does not split Y_iZ.

(2) "<" is antisymmetric. Showing that $Y_i < Y_j$ => Y_i does not split ZY_j is sufficient. Assume not, i.e. $Y_i<Y_j$ and Y_i splits ZY_j. $Y_i<Y_j$ implies that there exists a $V_j \in$ DEP(Y_j) such that $Z_j = V_j \cap Z \neq \emptyset$ and $V_j \cap Y_i = \emptyset$. Y_i splits ZY_j implies that there exists a $V_i \in$ DEP(Y_i) such that $Z_i = V_i \cap Z \neq \emptyset$ and $V_i \cap Y_j = \emptyset$, and $Y_j - Y_i \neq \emptyset$, i.e. $Y_i \cap Y_j \subset Y_j$. We claim that $Z_i \cap Z_j \neq \emptyset$, i.e. $V_i \cap V_J \neq \emptyset$, otherwise, since Y_i does not split Z, $Z_j \subseteq Y_i$. This implies Y_j splits Y_i, which is a contradiction. Therefore, by Lemma 3.2 and 2.4, $V_i = V_j \in$ DEP($Y_i \cap Y_j$). That means $Y_i \cap Y_j$ splits ZY_i, and, by Lemma 3.4 (1), there is an essential key $Y' \subseteq (Y_i \cap Y_j)$ such that Y´ splits ZY_i, and, by Lemma 3.4 (2), $Y' \in$ K, which contradicts that $Y_i<Y_j$.

(3) "<" is transitive, i.e. $Y_i < Y_j$ and $Y_j < Y_k$ => $Y_i < Y_k$. $Y_i<Y_j$ implies that DEP(Y_j) = $\{V_{j1}, V_{j2}, ...\}$, where $Z \subseteq Y_jV_{j1}$, $Z \cap V_{j1} \neq \emptyset$, $Y_i \subseteq Y_jV_{j2}$, and $Y_i \cap V_{j2} \neq \emptyset$. $Y_j < Y_k$ implies that DEP(Y_k) = $\{V_{k1}, V_{k2}, ...\}$, where $Z \subseteq Y_kV_{k1}$, $Z \cap V_{k1} \neq \emptyset$, $Y_j \subseteq Y_kV_{k2}$, and $Y_j \cap V_{k2} \neq \emptyset$.

First, we show that Y_k splits ZY_i. If $V_{k1} \cap Y_i = \emptyset$ and $Y_i - Y_k \neq \emptyset$, then Y_k splits ZY_i. Suppose $Y_i - Y_k = \emptyset$, i.e. $Y_i \subseteq Y_k$. Then Y_j splits ZY_k since Y_j splits ZY_i, which contradicts to $Y_j < Y_k$. Now, we need to show $V_{k1} \cap Y_i = \emptyset$. Let $Y_{i1} = Y_i \cap Y_j$ and $Y_{i2} = Y_i \cap V_{j2}$, since $Y_i \subseteq Y_jV_{j2}$, $Y_i = Y_{i1}Y_{i2}$. $Y_j \cap V_{k1} = \emptyset$ implies $Y_{i1} \cap V_{k1} = \emptyset$. The fact that $Y_k \rightarrow\!\!\rightarrow Y_kV_{k2}$ and $Y_kV_{k2} \rightarrow\!\!\rightarrow V_{j2}$ implies $Y_k \rightarrow\!\!\rightarrow (V_{j2} - Y_kV_{k2})$, i.e. $Y_k \rightarrow\!\!\rightarrow (V_{j2} - V_{k2})$. $Y_k \rightarrow\!\!\rightarrow (V_{j2} - V_{k2})$ and $Y_k \rightarrow\!\!\rightarrow V_{k1}$ implies $Y_k \rightarrow\!\!\rightarrow (V_{k1} - (V_{j2} - V_{k2}))$, i.e. $Y_k \rightarrow\!\!\rightarrow (V_{k1} - V_{j2})$. Since $V_{k1} \in$ DEP(Y_k), $(V_{k1} - V_{j2}) = V_{k1}$, i.e. $V_{j2} \cap V_{k1} = \emptyset$, so $Y_{i2} \cap V_{k1} = \emptyset$. It follows that $Y_i \cap V_{k1} = \emptyset$. Thus, Y_k splits ZY_i.

Then, we show that no $Y' \subset Y_k$ in K such that $Y_i < Y'$. Assume not, i.e. there is a $Y' \subset Y_k$ in K such that $Y_i < Y'$. $Y_i < Y'$ implies that DEP(Y´) = $\{V_1', V_2', ...\}$, where, $Z \subseteq Y'V_1'$, $Z \cap V_1' \neq \emptyset$, $Y_i \subseteq Y'V_2'$, and $Y_i \cap V_2' \neq \emptyset$. $Y_j < Y_k$ implies that Y´ does not split ZY_j, so $Y_jZ \subseteq Y'V_1'$. Similarly, we have DEP(Y_j) = $\{V_{j1}, V_{j2}, ...\}$, where, $Y_kZ \subseteq Y_jV_{j1}$, $Z \cap V_{j1} \neq \emptyset$, $Y_i \subseteq Y_jV_{j2}$, and $Y_i \cap V_{j2} \neq \emptyset$. Thus, $V_2' \cap Y_j = \emptyset$, $V_{j2} \cap Y' = \emptyset$. If we can show that $V_2' \cap V_{j2} \neq \emptyset$, then, by Lemma 3.2 and 2.4, $V_2' = V_{j2} \in$ DEP($Y_j \cap Y'$). So, $Y_j \cap Y'$ splits ZY_i, bu Lemma 3.4 (1), there is an essential key $X \subseteq (Y_j \cap Y')$ such that X splits ZY_i. $Y_i < Y_j$ and $Y_i < Y'$ implies that $Y' \not\subseteq Y_j$, i.e. $Y_j \cap Y' \subset Y'$. Therefore, $X \subset Y'$, and, by Lemma 3.4 (2), X is in K. Contradiction to $Y_i < Y'$.

Now, we show that $V_2' \cap V_{j2} \neq \emptyset$. Let $A = V_2' \cap Y_i$, $B = V_{j2} \cap Y_i$. We have shown that A and B are nonempty. Suppose $A \cap B = \emptyset$, the fact that $Y_j \rightarrow\!\!\rightarrow V_{j2}$ and Y_j does not split Y_i implies that $A \subseteq Y_j$, i.e. $Y_j \cap V_2' \neq \emptyset$, which contradicts that $Y_jZ \subseteq Y'V_1'$.

Lemma 6.5. Let M be a conflict free set of MVD´s, Z, V be sets of attributes such that Z $\rightarrow\!\!\rightarrow$ V is left- and right-reduced. If M does not split Z and FK(V) $\neq \emptyset$, then there exists an X_0 in FK(V) such that M does not split ZX_0.

Proof: Let K = $\{Y_i \mid Y_i$ is an essential key and $Y_i \cap V \neq \emptyset\}$. A binary relation "<" is defined on K as follows.

$Y_i < Y_j$ iff ZY_i is split by Y_j but not by any $Y_j' \subset Y_j$ in K.

By Lemma 6.4, "$<$" is irreflexive, antisymmetric, and transitive on K, thus, there exists at least one Y in K such that for any Y´ in K, Y $\not<$ Y´, i.e., no Y´ in K splits ZY. By Lemma 3.4, M does not split ZY $\supseteq ZX_0$ for some X_0 in FK(V).

Lemma 6.6. Let F be an NSF decomposition of U with respect to M obtained by Algorithm D. If M is conflict free then any path p in P(F) is nondecomposable with respect to M.

Proof: Refer to Algorithm D and the procedure DECOMP. Let T_1' be the nondecomposable semi normal tree returned by DECOMP(U) in step 2 of the algorithm. Lemma 6.5 guarantees that the selecting condition in step 2 of DECOMP is always satisfied. Thus, for any edge $e = \langle u_1, v_1 \rangle$ in T_1', there exists an X $\subseteq$ A(u_1) in T_1' such that X $\twoheadrightarrow$ D(v_1) is left- and right-reduced and, X is not split by any essential key Y, where Y $\cap$ D(v_1) $\neq \emptyset$. Let $e = \langle u, v \rangle$ be any edge in p, then there exists an edge $e = \langle v_1, v_1 \rangle$ in T_1' such that v_1 is the corresponding node of v, i.e. v_1 = v. By Lemma 6.3, A(u) $\twoheadrightarrow$ D(v_1) is left- and right-reduced, and by Lemma 3.3 (1), A(u) = X, and D(v) $\subseteq$ D(v_1). Therefore, A(u) is not split by any essential key Y, where Y $\cap$ D(v) $\neq \emptyset$. Let L be the leaf of p, and A be A(F(L)) in p. Assume p is decomposable with respect to M, then by Lemma 4.2 and 3.4 (1), there exists an essential key Z $\supseteq$ L and Z splits p, i.e. Z splits A, which is a contradiction.

Proof of Proposition 6.2:

(1) From Proposition 6.1, and Lemma 6.6, it follows that P(F) is a 4NF decomposition of U with respect to M. Since M is conflict free, by Proposition 2.3, P(F) is unique and also in SFNF.

(2) >From (1) and Lemma 8.7 in [BFMY], M $\Leftrightarrow$ *(P(F)). Thus from Proposition 6.1, it follows that
M $\Leftrightarrow$ MVD(F) $\Leftrightarrow$ *(P(F)).

(3) Follows from (2) and Theorem 3.4 in [BFMY].

Let M be a set of MVD´s over U, and F = $\{T_1,...,T_n\}$ be a normal scheme forest decomposition of U with respect to M. If M is not conflict free, then the join dependencies implied by the decomposition, i.e., *(S(T_1),...,S(T_n)) and *(P(F)) are not necessarily acyclic join dependencies. We conclude this section by an example illustrating this situation.

Example 6.4. Let U = ABCDE, and M = {A $\twoheadrightarrow$ B | C, BC $\twoheadrightarrow$ E, D $\twoheadrightarrow$ B}. Then, {A, BC, D} is the set of all keys of M. Let A, BC, D be a p-ordering used in the decomposition, then F = $\{T_1, T_2, T_3\}$ is a normal scheme forest.

T_1: A . T_2 : BC . T_3 : . D

C . . D E . . B

In this example, the join dependencies *(S(T_1), S(T_2), S(T_3)) = *(ACD, BCE, DB), and *(P(F)) = *(AC, AD, BCE, BD) both are cyclic.

7. Conclusions

We have introduced a normal form for nested relations, and have presented an algorithm to obtain a nested normal form (NNF) decomposition of a given set of attributes with respect to a given set of MVD’s among these attributes. An NNF decomposition is a collection of nested relations where each relation has a tree structured scheme, called normal scheme tree. That is an NNF database scheme is a forest of normal scheme trees. One of the objectives of NNF is to reduce redundancy. Moreover, since the normal scheme forest is structured with respect to dependencies existing among attributes, the database scheme explicitly represents these dependencies. If the given set M of MVD´s is conflict free, then any NNF decomposition is dependency preserving, i.e., the MVD´s represented by the database scheme is equivalent to the existing MVD´s among attributes. Nevertheless, if M is not conflict free, then the NNF database scheme represents a set of nontrivial MVD´s, that are implied by M. The NNF decomposition of a set of attributes U with respect to any set M of MVD´s is a nonredundant and faithful (lossless) decomposition of U. In a sense, the nested normal form decomposition can be considered as a formal way of designing a hierarchical database by utilizing the MVD´s existing among attributes.

It has been argued [Sc] that the real world MVD´s are conflict free, and the desirable properties of conflict free MVD´s have been discussed by several researchers [BFMY, L2]. The NNF decomposition also has more desirable properties when the given set M of MVD´s is conflict free. We have shown that if M is conflict free, then the set of root-to-leaf paths in a normal scheme forest is precisely the unique 4NF decomposition of the given set of attributes with respect to M. This unique 4NF decomposition is also the Split Free Normal Form decomposition [BK2], which is a new normal form with desirable properties.

For convenience we consider FD´s, in this paper, as their MVD counterparts as in [L1] in obtaining NNF decomposition. Since an FD is more than an MVD, some of the semantic connections among attributes are not represented in the set M of MVD´s with this formulation. Hence, not all of the semantic connections existing among attributes in the form of FD´s and MVD´s are utilized in obtaining NNF. We are now investigating a normal form for nested relations in the context of FD´s and MVD´s, by incorporating the results of [YO] into nested normal form decomposition.

References

[AB] Abiteboul, S. and Bidoit, N., Non First Normal for Relations to Represent Hierarchically Organized Data, Proc. ACM PODS, 1984 pp. 191-200.

[Ba] Bancilhon, F., et al.., Verso: A Relational Back End Data Base Machine, Proc. International Workshop on Database Machine, San Diego, 1982.

[BBG] Beeri, C., Bernstein, P.A., and Goodman, N., A Sophisticate´s Introduction to Database Normalization Theory, Proc. VLDB, 1978, pp. 113-123.

[BFH] Beeri, C., Fagin, R., and Howard, J., A Complete Axiomatization for FD´s and MVD´s, Proc. ACM SIGMOD, 1977, pp. 47-61.

[BFMY] Beeri, C., Fagin, R., Maier, D. and Yannakakis, M., "On the Desirability of Acyclic Database Schemes", JACM, July 1983, pp. 479-513.

[BK1] Beeri, C. and Kifer, M., Elimination of Intersection Anomalies from Database Schemes, Proc. ACM PODS, 1983, pp. 340-351.

[BK2] Beeri, C. and Kifer, M., Comprehensive Approach to the Design of Relational Database Schemes, Proc. VLDB, 1984, pp. 196-207.

[Co] Codd, E., A Relational Model for Large Shared Data Bank, Comm. ACM, June 1970, pp. 377-387.

[Fa] Fagin, R., Multivalued Dependencies and a New Normal Form for Relational Databases, ACM TODS, Sept. 1977, pp. 262-278.

[FG] Fischer, P.C. and Gucht, D.V., Weak Multivalued Dependencies, Proc. ACM PODS, 1984, pp. 266-274.

[FT] Fischer, P.C. and Thomas, S.J., Operations for Non-First-Normal Form Relations, IEEE Computer Software and Applications Conference, Oct. 1983, pp. 464-475.

[Ha] Hawryszkiewycz, I.T., *Database Analysis and Design*, SRA, 1984.

[JS] Jaeschke, G. and Scheck, H.J., Remarks on the Algebra of Nonfirst Normal Form Relations, Proc. ACM PODS, 1982, pp. 124-138.

[KTT] Kambayashi, Y., Tanaka, K. and Takeda,K., Synthesis of Unnormalized Relations Incorporating more meaning, *Information Sciences*, 1983.

[L1] Lien, Y.E., Hierarchical Schemata for Relational Databases, ACM TODS, March 1981, pp. 48-69.

[L2] Lien, Y.E., On the Equivalence of Database Models, JACM, April 1982, pp. 333-362.

[Ma] Maier, D., *The Theory of Relational Databases*, Computer Science Press, 1983.

[Sc] Sciore, E., Real World MVD´s, Proc. SIGMOD, 1981, pp. 121-132.

[OzOM] Ozsoyoglu, G., Matos, V., and Ozsoyoglu, Z.M., Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions, Technical Report, Department of Computer Science, Case Western Reserve University, 1985.

[OO] Ozsoyoglu, Z.M. and Ozsoyoglu, G., An Extension of Relational Algebra for Summary Tables, Proc. Statistical Database Workshop, 1983, pp. 202-211.

[OY] Ozsoyoglu, Z.M. and Yuan, L.Y., Reduced MVD’s and Minimal Covers, Technical Report, Department of Computer Science, CES-84-06, Case Western Reserve University, 1984.

[RKS] Roth, M.A., Korth, H.F., and Silberschatz, A., Theory of Non-First-Normal Form Relational Databases. TR-84-36, Department of Computer Science, University of Texas at Austin (December 1984).

[Ul] Ullman, J.D., *Principles of Database Systems*, Computer Science Press, Potomac, Maryland, 1983.

[YO] Yuan, L.Y. and Ozsoyoglu, Z.M., Unifying Functional and Multivalued Dependencies for Relational Database Design, to appear in Proc. of the 5th ACM PODS, 1986.

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