Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases

C. J. DATE
Independent Consultant
and
RONALD FAGIN
IBM Almaden Research Center

A key is *simple* if it consists of a single attribute. It is shown that if a relation schema is in third normal form and every key is simple, then it is in projection-join normal form (sometimes called fifth normal form), the ultimate normal form with respect to projections and joins. Furthermore, it is shown that if a relation schema is in Boyce-Codd normal form and *some* key is simple, then it is in fourth normal form (but not necessarily projection-join normal form). These results give the database designer simple sufficient conditions, defined in terms of functional dependencies alone, that guarantee that the schema being designed is automatically in higher normal forms.

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

General Terms: Design, Theory

Additional Key Words and Phrases: Boyce-Codd normal form, BCNF, database design, fifth normal form, 5NF, fourth normal form, 4NF, functional dependency, join dependency, projection-join normal form, PJ/NF, multivalued dependency, normalization, relational database, simple key

1. INTRODUCTION

In his first papers on the relational model, Codd [4, 5] observed that relation schemas in which certain patterns of functional dependencies occur exhibit undesirable behavior. He defined various *normal forms* where this undesirable behavior does not occur. The strongest (most restrictive) such normal form that he defined then is *third normal form* (*3NF*). Later Codd [6] defined an "improved third normal form", usually called *Boyce-Codd normal form* (*BCNF*), which is stronger still. He discussed ways to *normalize*, that is, to convert relation schemas not in a given normal form into ones that are, by

Authors' addresses: C. J. Date, P.O.B. 1000, Healdsburg, CA 95448; R. Fagin, IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120.

making use of the projection and join operators. Fagin [8] observed that even BCNF relation schemas may have some of the anomalous behavior that Codd's normal forms were intended to prevent. By making use of *multivalued dependencies*, Fagin [8] defined a new *fourth normal form* (*4NF*), which is even stronger than BCNF. Later, by making use of *join dependencies* [10], Fagin [9] defined *projection-join normal form* (*PJ/NF*), sometimes called *fifth normal form* (*5NF*), which is stronger than 4NF, and which is the "ultimate" normal form, when only the projection and join operators are allowed.

Nearly every textbook on databases discusses 3NF, many consider BCNF and 4NF, and a number of them also discuss PJ/NF. There tends to be a problem with presentations of higher normal forms, namely 4NF and PJ/NF, both in print and in live seminars:[1] Either they are excessively concerned with the formalism, giving accurate and precise definitions of the higher normal forms, but very little in the way of practical insight, or else they fall into the opposite trap and make statements that are so imprecise and inaccurate as to be virtually useless.

The problem is that 4NF and PJ/NF are defined in terms of multivalued dependencies and join dependencies, which are harder to understand than functional dependencies. The purpose of this paper is to give some easy-to-understand conditions, which are defined in terms of functional dependencies alone, and that hold in a wide class of situations, that are sufficient to guarantee higher normal forms. In particular, we show that if a relation schema is in third normal form and every key is simple (that is, consists of a single attribute), then it is in projection-join normal form. Furthermore, we show that if a relation schema is in Boyce-Codd normal form and *some* key is simple, then it is in fourth normal form (but not necessarily projection-join normal form). These results give conditions that are easy for the practitioner to understand and that are sufficient to guarantee the higher normal forms. Thereby, they provide a practical database design guideline, which may make the database designer's job a little easier. These results are also useful for the database instructor, who can give the class practical situations in which projection-join normal form can be achieved, without requiring knowledge of multivalued dependencies or join dependencies.

In Section 2, we give an informal description of our results. In Sections 3, 4, and 5, we proceed formally. In Section 3, we give definitions. In Section 4, we prove that if a relation schema is 3NF and every key is simple, then the relation schema is PJ/NF. In Section 5, we consider the effect of assuming only that *some* key is simple, rather than *every* key is simple. We show that if a relation schema is BCNF and some key is simple, then the relation schema is 4NF. However, we show by a counterexample that if a relation schema is BCNF, and some key is simple, then it is not necessarily PJ/NF. Sections 3, 4, and 5 can be skipped by those interested only in the results but not the proofs.

[1] The authors of this paper include their own presentations in this regard!

2. INFORMAL DISCUSSION OF RESULTS

Normal forms can be thought of as "good" ways to structure relations in a relational database. As Codd explained in his early work, "good" means that certain anomalous behavior does not occur. There are various levels of normalization. The higher the normal form, the more desirable it is. *Normalization* is a process to convert the relation schema (the structure of the relations, along with the integrity constraints) into a higher normal form.

Nearly every textbook on databases discusses normalization through third normal form (3NF). There are a number of essentially equivalent definitions of 3NF. For the purposes of our discussion, a convenient definition of 3NF says that if A is an attribute that functionally depends on some set X of attributes (written $X \rightarrow A$) then either (1) A is in X (so the dependence is trivial), (2) X contains a key (so X functionally determines *every* attribute), or (3) A is part of a key. If we eliminate the third clause, then we obtain the stronger (more desirable) *Boyce-Codd normal form* (*BCNF*). Thus, BCNF demands that the only nontrivial functional dependencies are the result of keys. 3NF is a little less demanding: It allows a nontrivial functional dependency $X \rightarrow A$ if A is part of a key. The reason this might be reasonable is that in passing from 3NF to BCNF, the normalization process would split up a key, which may not be desirable.

Normal forms are intended to correspond to "goodness". However, even though a relation schema is BCNF, it may still have some of the same problems that normalization was intended to prevent. Let us consider an example from Date [7] of a schema that is BCNF but not 4NF. There are attributes COURSE, TEACHER, and TEXT. A sample instance is in Figure 1. The semantics are as follows. A tuple (c, t, x) appears in the relation if and only if course c can be taught by teacher t and uses text x as a reference. For a given course, there can exist any number of corresponding teachers and any number of corresponding texts. It is assumed that teachers and texts are independent of each other; that is, no matter who actually teaches any particular offering of the given course, the same texts are used. This example is BCNF, since it is "all key" (no proper subset of the three attributes is a key). However, it is still not structured in a good way. This is because, for example, the information that Professor Green teaches physics appears twice, once for each physics textbook. This redundancy may lead to inconsistencies. *Fourth normal form* (*4NF*) is intended to remedy the problem, as we now discuss.

The functional dependency of TEACHER on COURSE, i.e., COURSE $\rightarrow$ TEACHER, does not hold, since a course may be taught by multiple teachers. However, there is a "multivalued dependency" of TEACHER on COURSE (written COURSE $\rightarrow\rightarrow$ TEACHER), and similarly a multivalued dependency COURSE $\rightarrow\rightarrow$ TEXT. For the schema to be 4NF, every multivalued dependency must be a consequence of keys. This is not the case here, so the relation schema is not 4NF. However, by decomposing it into two relation schemas, one with attributes COURSE and TEACHER, and another with attributes COURSE and TEXT, as in Figure 2, we obtain 4NF.

COURSE	*TEACHER*	*TEXT*
Physics	Prof. Green	Basic Mechanics
Physics	Prof. Green	Principles of Optics
Physics	Prof. Brown	Basic Mechanics
Physics	Prof. Brown	Principles of Optics
Math	Prof. Green	Basic Mechanics
Math	Prof. Green	Vector Analysis
Math	Prof. Green	Trigonometry

Fig. 1. BCNF but not 4NF.

COURSE	*TEACHER*
Physics	Prof. Green
Physics	Prof. Brown
Math	Prof. Green

COURSE	*TEXT*
Physics	Basic Mechanics
Physics	Principles of Optics
Math	Basic Mechanics
Math	Vector Analysis
Math	Trigonometry

Fig. 2. Decomposing to obtain 4NF.

Although it is certainly possible to give a formal, precise definition of multivalued dependencies (and of 4NF), the problem is that multivalued dependencies are not as intuitive as functional dependencies. Indeed, if the concept is stated too informally (such as, "there is a multivalued dependence of Y on X if for each X, there is a set of Y's"), then one could incorrectly conclude that for *every* pair X, Y, there is a multivalued dependency $X \rightarrow\rightarrow Y$. This make it difficult for practitioners to understand multivalued dependencies and 4NF (which is defined in terms of multivalued dependencies).

We are now ready to discuss the contributions of this paper. We refer to a key consisting of a single attribute as *simple*. One of our results is that BCNF, along with some key being simple, implies 4NF. Thus, the only possible "counterexamples" of schemas that are BCNF but not 4NF have the property that every key is a compound key, consisting of more than one attribute. This gives us a condition involving only functional, and not multivalued dependencies, that guarantees 4NF.

As we noted earlier, the COURSE-TEACHER-TEXT example we gave, that is BCNF but not 4NF, is all key. In fact, an examination of handbooks on database design and similar documents reveals that examples of schemas that are BCNF but not 4NF always seem to be of this same general form, and in particular all key. In some sense, the examples are all really the same example. This might lead us to believe that every example of a relation schema that is BCNF but not 4NF is necessarily all key. However, this is not the case. We can simply modify our COURSE-TEACHER-TEXT example by

adding a constraint that a given teacher cannot use the same text in more than one course (this corresponds to adding the functional dependency {TEACHER, TEXT} → COURSE). Then {TEACHER, TEXT} is a key, and in particular this example is not all key. Since the multivalued dependence COURSE →→ TEACHER is not a consequence of keys, the schema is still not 4NF, although it is BCNF.

We just saw by an example that the conjecture is false that every schema that is BCNF but not 4NF is all key. What our two examples of schemas that are BCNF but not 4NF do have in common is that in each case, there is only one key ({COURSE, TEACHER, TEXT} in the first example, and {TEACHER, TEXT} in the second example), and these keys are compound. More generally, we show that in every schema that is BCNF but not 4NF, every key is compound. Or, putting it another way, we show that if a schema is BCNF and some key is simple, then it is 4NF.

So far in this section, we have discussed normal forms through 4NF. There is a still higher normal form, called *projection-join normal form* (*PJ/NF*) or sometimes *fifth normal form* (*5NF*), which is the "ultimate" normal form as far as decomposing with respect to projections and joins is concerned. Just as 4NF is defined by using multivalued dependencies, PJ/NF is defined by using *join dependencies*, which are also somewhat hard for the practitioner to understand. Once again, we can give simple conditions, which we now describe, that are defined using functional dependencies alone, that are sufficient to guarantee PJ/NF.

As we have discussed, the assumptions that

(1) the relation schema is BCNF, and
(2) some key is simple

are enough to guarantee 4NF. However, as we show later, these assumptions are not sufficient to guarantee PJ/NF. But if we strengthen the second assumption by assuming that *every* (not just *some*) key is simple, then we can show that this is enough to guarantee PJ/NF. In fact, we can then even weaken the first assumption by assuming only that the relation schema is 3NF, rather than assuming BCNF. Thus, we show that the assumptions that

(1) the relation schema is 3NF, and
(2) every key is simple

are enough to guarantee PJ/NF. For those database designers who are most comfortable with 3NF, this gives them an additional condition (that every key is simple) which, when combined with 3NF, automatically guarantees PJ/NF.

This concludes the informal part of this paper. Those who are interested in formal definitions and proofs can read on.

3. DEFINITIONS

We are given a fixed finite set U of distinct symbols, called *attributes*, which represent column names of a relation. If X and Y are sets of attributes, then we may write XY for $X \cup Y$. If $X = \{A_1, \ldots, A_n\}$, then we may write

$A_1 \cdots A_n$ for X. In particular, we may write simply A to represent the singleton set $\{A\}$.

Let T be a set of attributes (that is, a subset of U). A *T-tuple* (or simply *tuple*, if T is understood) is a function with domain T. Thus, a T-tuple is a mapping that associates a value with each attribute in T. A *relation (over T)* is a set of T-tuples. If s is a U-tuple, then $s[T]$ denotes the T-tuple obtained by restricting the mapping s to T.

Assume that the relations $R_1, \ldots, R_n$ are over attribute sets $T_1, \ldots, T_n$ respectively. The *join* of the relations $R_1, \ldots, R_n$, which is written $\bowtie \{R_1, \ldots, R_n\}$, is the set of all tuples s over the attribute set $T_1 \cup \cdots \cup T_n$ such that $s[T_i]$ is in R_i for each i. (Our notation exploits the fact that the join is associative and commutative.)

A *functional dependency (FD)* [5] is a statement of the form $X \rightarrow Y$, where X and Y are sets of attributes. The FD $X \rightarrow Y$ is said to hold for a relation R if every pair of tuples of R that agrees on each of the attributes in X also agrees on the attributes in Y. That is, the FD $X \rightarrow Y$ holds for relation R if whenever s and t are tuples of R where $s[X] = t[X]$, then $s[Y] = t[Y]$.

A *multivalued dependency (MVD)* [8] is a statement of the form $X \rightarrow\rightarrow Y$, where X and Y are sets of attributes. Let Z be the set of attributes not in X or Y, that is, $Z = U - XY$. The MVD $X \rightarrow\rightarrow Y$ is said to hold for a relation R if whenever there are tuples s and t of R where $s[X] = t[X]$, then there is a tuple u of R where $u[XY] = s[XY]$ and $u[Z] = t[Z]$. Later, we shall make use of the simple fact [2] that the MVD $X \rightarrow\rightarrow Y$ is equivalent to the MVD $X \rightarrow\rightarrow (Y - X)$. In this way, we can replace an MVD by an equivalent MVD where the left-hand side and right-hand side are disjoint.

A *join dependency (JD)* [10] is a statement of the form $\bowtie \{X_1, \ldots, X_n\}$, where $X_1, \ldots, X_n$ are sets of attributes. The JD $\bowtie \{X_1, \ldots, X_n\}$ is said to hold for a relation R if $R = \bowtie\{R[X_1], \ldots, R[X_n]\}$. We shall make use later of the simple fact [8] that the JD $\bowtie \{X_1, X_2\}$ is equivalent to the MVD $X_1 \cap X_2 \rightarrow\rightarrow X_2$.

If Σ is a set of dependencies and σ is a single dependency, then Σ *logically implies* σ (or σ *is a logical consequence of* Σ) if every relation that satisfies Σ also satisfies σ. Thus, Σ logically implies σ if there is no "counterexample relation" that satisfies Σ but not σ. As an example, the set $\{A \rightarrow B, B \rightarrow C\}$ of FD's logically implies the FD $A \rightarrow C$ (this is called *transitivity*). As another example, the FD $X \rightarrow Y$ logically implies the MVD $X \rightarrow\rightarrow Y$. A dependency is *trivial* if it is valid, that is, a logical consequence of the empty set. For example, the FD $A \rightarrow A$ is trivial, since every relation where one of the attributes is A satisfies this dependency. It is straightforward to verify that an FD $X \rightarrow Y$ is trivial if and only if $Y \subseteq X$; an MVD $X \rightarrow\rightarrow Y$ is trivial if and only if either $Y = \emptyset$ or $X \cup Y = U$; and a JD $\bowtie \{X_1, \ldots, X_n\}$ is trivial if and only if $X_i = U$ for some i.

A *relation schema* is a pair $\langle U, \Sigma \rangle$, where U is a set of attributes and Σ is a set of dependencies involving only these attributes. Thus, a relation schema describes the attributes U, along with the dependencies, where they are

thought of as constraints. A dependency σ is a *dependency of the schema* $\langle U, \Sigma \rangle$ if σ involves only attributes in U, and if σ is a logical consequence of Σ. A *key* (sometimes called *candidate key*) of a schema is a set K of attributes such that (a) the FD $K \rightarrow U$ is an FD of the schema, and (b) if K' is a proper subset of K, then $K' \rightarrow U$ is not an FD of the schema. A *superkey* is a superset of a key. Thus, K is a superkey precisely if the FD $K \rightarrow U$ is an FD of the schema. An FD $K \rightarrow U$, where K is a key of the schema, is called a *key dependency* (*KD*).

We now define the various normal forms that we shall consider in this paper, in increasing order of strength.

Third normal form was originally defined by Codd [5]. In this paper, we shall make use of Zaniolo's [11] definition of third normal form, which is equivalent to Codd's, but is easier to use for our purposes. To define third normal form, we need another definition. In a relation schema, an attribute is a *key* attribute (sometimes called a *prime* attribute) if it is contained in some key. Otherwise, it is a *nonkey* attribute. A relation schema is in *third normal form* (*3NF*) if whenever $X \rightarrow A$ is a nontrivial FD of the schema, where A is a single attribute, then either X is a superkey or A is a key attribute.

A relation schema is in *Boyce-Codd normal form* (*BCNF*) [6] if whenever $X \rightarrow Y$ is a nontrivial FD of the schema, necessarily X is a superkey. It is shown by Fagin [9] that a relation schema is BCNF if and only if every FD of the schema is a logical consequence of the set of key dependencies of the schema.

A relation schema is in *fourth normal form* (*4NF*) [8] if whenever $X \rightarrow\rightarrow Y$ is a nontrivial MVD of the schema, necessarily X is a superkey. It is shown by Fagin [9] that a relation schema is 4NF if and only if every MVD of the schema is a logical consequence of the set of key dependencies of the schema.

A relation schema is in *projection-join normal form* (*PJ/NF*) [9], sometimes called *fifth normal form* (*5NF*), if every JD of the schema is a logical consequence of the set of key dependencies of the schema.

4. ASSUMING EVERY KEY IS SIMPLE

In this section, we prove that a simple condition, involving only functional dependencies, and one that holds in a wide class of situations, guarantees PJ/NF. Specifically, we show:

THEOREM 4.1. *Assume that a relation schema is 3NF, and that every key is simple. Then the relation schema is PJ/NF.*

To prove this theorem, there are two steps. We first show that under the hypotheses of the theorem, the relation schema is BCNF (this is very easy to show). We then prove the harder result that if a relation schema is BCNF, and every key is simple, then it is PJ/NF.

The next lemma is the first step in the proof of Theorem 4.1.

LEMMA 4.2. *Assume that a relation schema is 3NF, and that every key is simple. Then the relation schema is BCNF.*

PROOF. Let $X \rightarrow A$ be a nontrivial FD of the schema, where A is a single attribute. Assume that the schema is 3NF. To show that it is BCNF, it is sufficient to show that X is a superkey. Since the schema is 3NF, either X is a superkey, or A is a key attribute. If A is a key attribute, then A is a key itself, since by assumption, every key is simple. Since $X \rightarrow A$ is an FD of the schema, it follows that X is a superkey. So in any case, X is a superkey. This was to be shown. □

If $\bowtie \{X_1, \ldots, X_m\}$ is a JD, then we refer to the X_i's as *components* of the JD. We shall make use of the following simple lemma, whose proof is straightforward.

LEMMA 4.3. [3] *Let J_1 be a JD, and let J_2 be a JD obtained from J_1 by replacing two components of J_1 by their union. Then J_1 logically implies J_2.*

As an example of Lemma 4.3, the JD $\bowtie \{ABC, AD, BC, CE\}$ logically implies the JD $\bowtie \{ABC, AD, BCE\}$, where we replace the components BC and CE by their union BCE.

We shall also make use of the following Membership Algorithm for determining whether a given JD is a logical consequence of a set of KD's. The correctness of the algorithm is stated in Fagin [9]; it also follows from the results of Aho et al. [1].

The input is a JD $\bowtie \{X_1, \ldots, X_m\}$ and a set $\{K_1 \rightarrow U, \ldots, K_s \rightarrow U\}$ of KD's. Initialize set $\mathscr{S}$ as $\{X_1, \ldots, X_m\}$. Apply the following rule until it can be no longer applied: if $K_i \subseteq Y \cap Z$ for some key K_i $(1 \leq i \leq s)$ and for some members Y and Z of $\mathscr{S}$, then replace Y and Z in $\mathscr{S}$ by their union, that is, remove the sets Y and Z from $\mathscr{S}$ and add to $\mathscr{S}$ the single member $Y \cup Z$. (In particular, the number of members of $\mathscr{S}$ then decreases by one.) Let $\{Y_1, \ldots, Y_r\}$ be the final result. Then $\bowtie \{X_1, \ldots, X_m\}$ is a logical consequence of the set $\{K_1 \rightarrow U, \ldots, K_s \rightarrow U\}$ of KD's precisely if some Y_i equals the set U of all attributes.

Example. Assume that the attributes are $U = \{A, B, C, D\}$, and that the input consists of the JD $\bowtie \{AB, AD, BC\}$ and the KD's $A \rightarrow U$ and $B \rightarrow U$. We now show, by using the Membership Algorithm, that the JD is a logical consequence of the KD's. Initialize $\mathscr{S}$ as $\{AB, AD, BC\}$. Since $A \rightarrow U$ is a KD and since AB and AD are in $\mathscr{S}$, we replace AB and AD in $\mathscr{S}$ by ABD. At this stage, $\mathscr{S}$ is $\{ABD, BC\}$. Since $B \rightarrow U$ is a KD and since ABD and BC are in $\mathscr{S}$, we replace ABD and BC by $ABCD$. We are left with $\mathscr{S} = \{ABCD\} = \{U\}$. So the Membership Algorithm tells us that, indeed, the JD $\bowtie \{AB, AD, BC\}$ is a logical consequence of the set $\{A \rightarrow U, B \rightarrow U\}$ of KD's. □

The next lemma is the second (and final) step in the proof of Theorem 4.1.

LEMMA 4.4. *Assume that a relation schema is BCNF, and every key is simple. Then the relation schema is PJ/NF.*

PROOF. Assume that the relation schema is BCNF, and every key is simple. We wish to show that the schema is PJ/NF. Assume not. Then there

is a JD $\bowtie \{X_1, \ldots, X_m\}$ of the schema that is not a logical consequence of the KD's.

Let $\{Y_1, \ldots, Y_r\}$ be the final result after applying the Membership Algorithm with the JD $\bowtie \{X_1, \ldots, X_m\}$ and the KD's of the schema as the input. Since, by assumption, $\bowtie \{X_1, \ldots, X_m\}$ is not a logical consequence of the KD's, it follows from the Membership Algorithm that none of the Y_i's equals the set U of all attributes. Note also that it follows from repeated applications of Lemma 4.3 that the JD $\bowtie \{X_1, \ldots, X_m\}$ logically implies the JD $\bowtie \{Y_1, \ldots, Y_r\}$.

By assumption, every key is simple. Since every relation schema has at least one key, it follows in particular that there is some simple key. Let A be an attribute that is a simple key. Then A is in exactly one Y_i. This is because A is certainly in at least one Y_i, since the union of the Y_i's is the set U of all attributes. Further, A cannot be a member of both Y_i and Y_j, where $i \neq j$, because the procedure would have replaced Y_i and Y_j by $Y_i \cup Y_j$, since A is a key.

Without loss of generality, let Y_1 be the Y_i that contains A. Define $W = Y_2 \cup \cdots \cup Y_r$ to be the union of all of the Y_i's except for Y_1; there is at least one such Y_i, since as we noted above, none of the Y_i's equals the set U of all attributes, and in particular $Y_1 \neq U$. By Lemma 4.3 applied repeatedly, it follows that the JD $\bowtie \{Y_1, W\}$ is a logical consequence of $\bowtie \{X_1, \ldots, X_m\}$. As we noted earlier, the JD $\bowtie \{Y_1, W\}$ is equivalent to the MVD $Y_1 \cap W \rightarrow\rightarrow W$. Therefore, the MVD $Y_1 \cap W \rightarrow\rightarrow W$ is an MVD of the schema.

We now make use of the following inference rule for functional and multivalued dependencies, which is a special case of the inference rule FD-MVD2 in the complete axiomatization of Beeri et al. [2] for FD's and MVD's.

(*) *If* $X \rightarrow\rightarrow Z$ *and* $Y \rightarrow Z$, *where* Y *and* Z *are disjoint, then* $X \rightarrow Z$.

Let X be $Y_1 \cap W$, let Y be A, and let Z be W. We now show that we can apply the inference rule (*) to derive an FD of the schema.

(1) $X \rightarrow\rightarrow Z$ *is an MVD of the schema*: Here $X \rightarrow\rightarrow Z$ is $Y_1 \cap W \rightarrow\rightarrow W$, which we showed above is an MVD of the schema.

(2) $Y \rightarrow Z$ *is an FD of the schema*: Here $Y \rightarrow Z$ is $A \rightarrow W$, which is an FD of the schema, since A is a key.

(3) Y *and* Z *are disjoint*: Here Y and Z are A and W, respectively. We saw above that A is in precisely one of the Y_i's, namely Y_1. Since $W = Y_2 \cup \cdots \cup Y_r$ (the union of all of the Y_i's except for Y_1), it follows that $A \notin W$. Thus A and W are indeed disjoint.

It follows from inference rule (*) that $X \rightarrow Z$, that is, $Y_1 \cap W \rightarrow W$, is an FD of the schema. We now show that the FD $Y_1 \cap W \rightarrow W$ is nontrivial, that is, W is not a subset of $Y_1 \cap W$. If W were a subset of $Y_1 \cap W$, then W would be a subset of Y_1. Since $Y_1 \cup W = Y_1 \cup Y_2 \cup \cdots \cup Y_r = U$, it would follow that $Y_1 = U$, which we have shown is not the case. So indeed, the FD $Y_1 \cap W \rightarrow W$ is nontrivial. Since it is an FD of the schema, and since the schema is BCNF,

it follows that $Y_1 \cap W$ is a superkey. Since by assumption, every key is simple, it follows that $Y_1 \cap W$ contains some simple key B. Since $B \in Y_1 \cap W \subseteq W = Y_2 \cup \cdots \cup Y_r$, we know that $B \in Y_j$ for some j with $2 \leq j \leq r$. But also, $B \in Y_1 \cap W \subseteq Y_1$. Thus, B is a member of both Y_1 and Y_j. But this is impossible, because the procedure would have replaced Y_1 and Y_j by $Y_1 \cup Y_j$, since B is a key. □

Theorem 4.1 now follows from Lemmas 4.2 and 4.4.

5. ASSUMING SOME KEY IS SIMPLE

The assumption that every key is simple is fairly strong. In this section, we show that we can sometimes get by with the weaker assumption that *some* key is simple. In particular, we show that every BCNF relation schema where some key is simple is 4NF. However, we show by a counterexample that there is a BCNF relation schema where some key is simple that is not PJ/NF.

In order to prove that every BCNF relation schema where some key is simple is 4NF, we make use of the following lemma, which gives us information about the structure of a relation schema that is BCNF but not 4NF. Recall that if a relation schema is not 4NF, then there is a nontrivial MVD $V \rightarrow\rightarrow W$ of the schema where V is not a superkey.

LEMMA 5.1. *Assume that a relation schema is BCNF but not 4NF. Let $V \rightarrow\rightarrow W$ be a nontrivial MVD of the schema where V is not a superkey. Let W' be the set of attributes not in V or W. Then every key of the schema contains a member of W and a member of W'.*

PROOF. As we noted earlier, we can assume without loss of generality that V and W are disjoint (by replacing W by $W - V$ if necessary), so that each attribute is in exactly one of V, W, or W'. If the assumptions of the lemma hold but not the conclusion, then let K be a key of the schema that does not contain both a member of W and a member of W'. The roles of W and W' are symmetric, since $V \rightarrow\rightarrow W'$ is also a nontrivial MVD of the schema. Therefore, without loss of generality, we can assume that K does not contain a member of W, that is, K and W are disjoint. Now $K \rightarrow W$ is an FD of the schema, since K is a key. Let X be V, let Y be K, and let Z be W. We now show that we can apply the inference rule (*) to derive and FD of the schema.

(1) *$X \rightarrow\rightarrow Z$ is an MVD of the schema*: Here $X \rightarrow\rightarrow Z$ is $V \rightarrow\rightarrow W$, which by assumption is an MVD of the schema.
(2) *$Y \rightarrow Z$ is an FD of the schema*: Here $Y \rightarrow Z$ is $K \rightarrow W$, which is an FD of the schema, since K is a key.
(3) *Y and Z are disjoint*: Here Y and Z are K and W, respectively, which by assumption are disjoint.

If follows from inference rule (*) that $X \rightarrow Z$, that is, $V \rightarrow W$, is an FD of the schema. Since the schema is BCNF and W is not a subset of V, it follows that V is a superkey. This is a contradiction. □

As an example of Lemma 5.1, let us consider our two COURSE-TEACHER-TEXT examples from Section 2. In both of these, the MVD COURSE $\rightarrow\rightarrow$ TEACHER holds. In Lemma 5.1, we can take V, W, and W' to be singleton sets consisting of COURSE, TEACHER, and TEXT respectively. Therefore, Lemma 5.1 tells us that each key must contain {TEACHER, TEXT}. Indeed, in our first COURSE-TEACHER-TEXT example, there is only one key, namely {COURSE, TEACHER, TEXT}, and in our second COURSE-TEACHER-TEXT example, there is only one key, namely {TEACHER, TEXT}.

We can now prove:

THEOREM 5.2. *Assume that a relation schema is BCNF, and some key is simple. Then the relation schema is 4NF.*

PROOF. Assume that a relation schema is BCNF but not 4NF; we shall show that no key is simple. Since the schema is not 4NF, it has a nontrivial MVD $V \rightarrow\rightarrow W$ where V is not a superkey. Let W' be the set of attributes not in V or W. Since the MVD $V \rightarrow\rightarrow W$ is nontrivial, it follows that W and W' are nonempty. By Lemma 5.1, every key of the schema contains a member of W and a member of W'. Since W and W' are nonempty and disjoint, it follows that no key is simple. This was to be shown. □

Counterexample. We cannot replace "4NF" by "PJ/NF" in Theorem 5.2, as the following counterexample shows. Let the schema have attributes $U = \{A, B, C, D\}$. Let the dependencies of the schema be the logical consequences of the KD's $A \rightarrow U$ and $BC \rightarrow U$ and the join dependency $\bowtie \{ABC, BD, CD\}$. Although one of the keys is simple, one of the keys is not.

We now show that the only nontrivial FD's $X \rightarrow Y$ of the schema are those where X contains either A or both of B and C (that is, we show that any FD not of this form is not a logical consequence of $\{A \rightarrow U, BC \rightarrow U, \bowtie \{ABC, BC, CD\}\}$). It is easy to see that the only subsets X of $\{A, B, C, D\}$ where X contains neither A nor both of B and C are subsets of $\{B, D\}$ and subsets of $\{C, D\}$. So we must show that if $X \subseteq \{B, D\}$ or $X \subseteq \{C, D\}$, and Y is not a subset of X, then $X \rightarrow Y$ is not an FD of the schema. By the symmetric role of B and C, we can restrict attention to the case where $X \subseteq \{B, D\}$. If $X \rightarrow Y$ were a nontrivial FD of the schema, then it is straightforward to verify that one of four FD's $B \rightarrow D$, $D \rightarrow B$, $BD \rightarrow A$, or $BD \rightarrow C$ would be an FD of the schema. However, consider the 2-tuple relation where one tuple has 0's in every entry, and the other tuple has 0's as the B and D entries, but 1's as the A and C entries. It is straightforward to verify that this relation satisfies all of the dependencies of the schema, but neither of the FD's $BD \rightarrow A$ nor $BD \rightarrow C$. Similarly a 2-tuple relation where one tuple has 0's in every entry, and the other tuple has 0's as the B (respectively, D) entry, but 1's as all the other entries, satisfies all of the dependencies of the schema, but not the FD $B \rightarrow D$ (respectively, $D \rightarrow B$).

From what we just showed, it follows that the relation schema is BCNF. By applying the Membership Algorithm given above, we see that the JD $\bowtie \{ABC, BD, CD\}$ is not a logical consequence of the KD's $A \rightarrow U$ and $BC \rightarrow U$. So the relation schema is *not* in PJ/NF. □

ACKNOWLEDGMENTS

The authors are grateful to Shuky Sagiv and Moshe Vardi for helpful comments.

REFERENCES

1. AHO, A. V., BEERI, C., AND ULLMAN, J. D. The theory of joins in relational data bases. *ACM Trans. Database Syst. 4*, 3 (Sept. 1979), 297–314.
2. BEERI, C., FAGIN, R., AND HOWARD, J. H. A complete axiomatization for functional and multivalued dependencies in database relations. In *Proceedings of the 1977 ACM SIGMOD Conference*, D. C. P. Smith, Ed., pp. 47–61.
3. BEERI, C., AND VARDI, M. Y. On the properties of join dependencies. In *Advances In Databases—Vol. 1*, H. Gallaire, J. Minker, and J. M. Nicolas, Eds., Plenum Press, 1981, pp. 25–72.
4. CODD, E. F. A relational model of data for large shared data banks. *Commun. ACM 13*, 6 (June 1970), 377–387.
5. CODD, E. F. Further normalization of the data base relational model. In *Courant Computer Science Symposium 6: Data Base Systems*, R. Rustin, Ed., Prentice-Hall, 1972, pp. 33–64.
6. CODD, E. F. Recent investigations in relational data base systems. In *Information Processing 74*, North-Holland, 1974, pp. 1017–1021.
7. DATE, C. J. Further normalization. In *An Introduction to Database Systems: Volume 1*, Ch. 21, 5th edition, Addison-Wesley, 1990.
8. FAGIN, R. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst. 2*, 3 (Sept. 1977), 262–278.
9. FAGIN, R. Normal forms and relational database operators. In *Proceedings of the 1979 ACM SIGMOD Conference*, P. A. Bernstein, Ed., pp. 153–160.
10. RISSANEN, J. Theory of relations for databases—a tutorial survey. In *Proceedings of the 7th Symposium on Mathematical Foundation of Computer Science*, Lecture Notes in Computer Science 64, Springer-Verlag, 1978, pp. 537–551.
11. ZANIOLO, C. A new normal form for the design of relational database schemata, *ACM Trans. Database Syst. 7*, 3 (Sept. 1982), 489–499.

Received April 1991; revised August 1991; accepted August 1991