Multivalued Dependencies and a New Normal Form for Relational Databases

RONALD FAGIN
IBM Research Laboratory

A new type of dependency, which includes the well-known functional dependencies as a special case, is defined for relational databases. By using this concept, a new ("fourth") normal form for relation schemata is defined. This fourth normal form is strictly stronger than Codd's "improved third normal form" (or "Boyce-Codd normal form"). It is shown that every relation schema can be decomposed into a family of relation schemata in fourth normal form without loss of information (that is, the original relation can be obtained from the new relations by taking joins).

Key words and phrases: database design, multivalued dependency, functional dependency, fourth normal form, 4NF, third normal form, 3NF, Boyce-Codd normal form, normalization, decomposition, relational database
CR Categories: 3.59, 4.33, 6.1

1. INTRODUCTION

The concept of "functional dependencies" [1, 3–6, 8, 10, 11] has proved to be useful in the design and analysis of relational databases. In fact in one approach [3, 4] to the logical design of relational databases, functional dependencies are essentially the only input. We introduce "multivalued dependencies," which are a generalization of functional dependencies. We believe that multivalued dependencies significantly extend the understanding of logical database design. Multivalued dependencies lead to a new ("fourth") normal form for relational databases. Roughly speaking, we say that a relation schema is in fourth normal form if all dependencies (functional and multivalued) are the result of keys.

We now introduce multivalued dependencies and compare them with functional dependencies, by way of example. A precise definition appears in Section 2. Let S(EMPLOYEE,SALARY,CHILD) be the relation that appears in Table I. Following Codd [6], we say that the relation S obeys the *functional dependency* EMPLOYEE→SALARY. Intuitively this means that each employee has exactly one salary. The precise meaning is that if two tuples (that is, rows) of S agree in the EMPLOYEE column, then they agree in the SALARY column. When we say that a functional dependency (or a multivalued dependency) holds for a relation

Authors' address: IBM Research Laboratory K53/282, 5600 Cottle Road, San Jose, CA 95193.

Table I

EMPLOYEE	SALARY	CHILD
Hilbert	\$40K	Hubert
Gauss	\$50K	Gwendolyn
Gauss	\$50K	Greta
Pythagoras	\$20K	Peter

Table II

EMPLOYEE	CHILD	SALARY	YEAR
Hilbert	Hubert	\$35K	1976
Hilbert	Hubert	\$40K	1976
Gauss	Gwendolyn	\$40K	1975
Gauss	Gwendolyn	\$50K	1976
Gauss	Greta	\$40K	1975
Gauss	Greta	\$50K	1976
Pythagoras	Peter	\$15K	1975
Pythagoras	Peter	\$20K	1976

schema S*, we mean that every relation S that is an instance ("snapshot") of the schema S* is constrained to obey the dependency. (For our purposes, a *relation schema* is simply a set of column names, along with a set of dependencies, which can be thought of as "integrity constraints." See Cadiou [5] for a careful, thorough discussion of relation schemata.) The relation schema S*(EMPLOYEE,SALARY, CHILD), of which the relation S of Table I is an instance, "obeys" (that is, has as a part of its definition) the functional dependency EMPLOYEE→SALARY. So every instance must obey this functional dependency.

What are the multivalued dependencies? First, the multivalued dependency EMPLOYEE→→SALARY (which can be read "EMPLOYEE multidetermines SALARY") holds for S*, since functional dependencies (in which the left- and right-hand sides are disjoint) turn out to be special cases of multivalued dependencies. Furthermore the multivalued dependency EMPLOYEE→→CHILD holds for S* because intuitively, an employee's set of children is completely determined by the employee and is "orthogonal" to the salary. In this case multivalued dependencies remedy one of Schmid and Swenson's [12] objections to functional dependencies, that an employee "has" a set of children just as he "has" a salary and there should be no arbitrary distinction. Thus both of the multivalued dependencies EMPLOYEE→→SALARY and EMPLOYEE→→CHILD hold for this schema.

As another example, let T*(EMPLOYEE,CHILD,SALARY,YEAR) be a relation schema that specifies the children and salary history of each employee. An instance T of T* appears in Table II. A tuple, such as (Pythagoras,Peter,\$20K, 1976), appears in relation T *iff* (1) Pythagoras is an employee, (2) one of his children is named Peter, and (3) during at least part of 1976, his salary was \$20K. Although T* has no functional dependencies, it does have the multivalued dependencies EMPLOYEE→→CHILD (as in the previous example), and also EMPLOYEE→→{SALARY,YEAR}, because intuitively, an employee's salary history is completely determined by the employee and is orthogonal to his set of children. *Caution*: As we shall see, it does *not* follow from EMPLOYEE→→{SALARY, YEAR} that either EMPLOYEE→→SALARY or EMPLOYEE→→YEAR. The pair {SALARY,YEAR} is in some sense a "cluster."

Multivalued dependencies provide a necessary and sufficient condition for a relation to be decomposable into two of its projections without loss of information (in the usual sense that the original relation is guaranteed to be the natural join of the two projections; note that in all cases the projections taken together never contain *more* information than the original relation). An exact statement of the equivalence between multivalued dependencies and decomposability without loss of information appears in Theorem 1 of Section 2. As an example of the theorem, the fact that the multivalued dependency EMPLOYEE$\rightarrow\rightarrow$CHILD holds for T^*(EMPLOYEE,CHILD,SALARY,YEAR) implies that T^* can be decomposed without loss of information into T_1^*(EMPLOYEE,CHILD) and T_2^*(EMPLOYEE, SALARY,YEAR). Under this decomposition, the relation T in Table II would decompose into the relations T_1(EMPLOYEE,CHILD) and T_2(EMPLOYEE, SALARY,YEAR) as given in Table III. This decomposition is desirable because, for example, in the (undecomposed) schema T^*(EMPLOYEE,CHILD,SALARY, YEAR) an employee's entire salary history is repeated once for every one of his children. The original schema T^* is in third normal form and even in the stronger "Boyce-Codd normal form" [7] since it is "all key" (that is, no proper subset of the four column names form a key for T^*). However, as we shall see, it is not in fourth normal form. To obtain fourth normal form, it is necessary to decompose T^* as above into T_1^* and T_2^*. The example is due to Schmid and Swenson [12], who recommend decomposition on semantic grounds.

In Section 2 we precisely define the concept "multivalued dependency." We also give a few simple properties.

In Section 3 we define a new (fourth) normal form by using multivalued dependencies. We show that each relation schema in fourth normal form is automatically in Boyce-Codd normal form. We also show that every relation schema can be decomposed into a family of relation schemata in fourth normal form without loss of information.

In Section 4 we present an example of the fourth normal form normalization process.

The last part of the paper (Sections 5–11) provides a more in-depth exploration of multivalued dependencies. The topics covered are: various ways to view multivalued dependencies (Section 5); multivalued dependencies with the empty set as the left-hand side (Section 6); transitivity (Section 7); the disjointness of the left- and right-hand sides (Section 8); a partitioning property that leads to a generalized notation for multivalued dependencies (Section 9); interactions between functional and multivalued dependencies (Section 10); and multivalued dependencies that hold for the projection of a relation (Section 11).

We note that Zaniolo independently discovered multivalued dependencies. In

Table III

EMPLOYEE	CHILD
Hilbert	Hubert
Gauss	Gwendolyn
Gauss	Greta
Pythagoras	Peter

EMPLOYEE	SALARY	YEAR
Hilbert	$35K	1976
Hilbert	$40K	1976
Gauss	$40K	1975
Gauss	$50K	1976
Pythagoras	$15K	1975
Pythagoras	$20K	1976

[13] he presents illuminating examples to illustrate their use. Also, Delobel and Léonard [9] define a concept, called "first-order hierarchical decomposition," that is related to the concept of multivalued dependencies.

2. DEFINITION OF MULTIVALUED DEPENDENCIES

In this section we define multivalued dependencies and give a few simple properties.

Let $R(X_1, \ldots, X_m, Y_1, \ldots, Y_n, Z_1, \ldots, Z_r)$ be a relation (i.e. a set of tuples) with $m + n + r$ column names (thus no column name appears twice). For notational convenience, we write boldface $\mathbf{X}$ for $\{X_1, \ldots, X_m\}$; $\mathbf{Y}$ and $\mathbf{Z}$ are defined analogously. Whenever we write, say, $R(\mathbf{X},\mathbf{Y},\mathbf{Z})$, we assume automatically that $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ are pairwise disjoint as above. If $x_1, \ldots, x_m$ are entries that appear under columns $X_1, \ldots, X_m$, then we write $\mathbf{x}$ for $(x_1, \ldots, x_m)$; $\mathbf{y}$ and $\mathbf{z}$ are defined analogously. Define $\mathbf{Y}_{\mathbf{xz}}$ to be $\{\mathbf{y} : (\mathbf{x},\mathbf{y},\mathbf{z}) \in R\}$. Of course $\mathbf{Y}_{\mathbf{xz}}$ is nonempty iff $\mathbf{x}$ and $\mathbf{z}$ appear together in a tuple of R (with x_1 in column X_1, etc.). The multivalued dependency $\mathbf{X} \to\to \mathbf{Y}$ is said to hold for $R(\mathbf{X},\mathbf{Y},\mathbf{Z})$ if $\mathbf{Y}_{\mathbf{xz}}$ depends only on $\mathbf{x}$; that is, if $\mathbf{Y}_{\mathbf{xz}} = \mathbf{Y}_{\mathbf{xz}'}$ for each $\mathbf{x}$, $\mathbf{z}$, $\mathbf{z}'$ such that $\mathbf{Y}_{\mathbf{xz}}$ and $\mathbf{Y}_{\mathbf{xz}'}$ are nonempty.

As an example, recall the relation T(EMPLOYEE,CHILD,SALARY,YEAR) in Table II. The multivalued dependency EMPLOYEE$\to\to$CHILD holds for T since, for example, $\text{CHILD}_{\text{Gauss,\$40K,1975}}$ and $\text{CHILD}_{\text{Gauss,\$50K,1976}}$ both equal {Gwendolyn,Greta}, and hence $\text{CHILD}_{\text{Gauss,\$40K,1975}} = \text{CHILD}_{\text{Gauss,\$50K,1976}}$.

We now give an example of the *failure* of the multivalued dependency EMPLOYEE$\to\to$CHILD. Define a new relation T′ which is the result of deleting the tuple (Gauss,Greta,\$50K,1976) from T. Then EMPLOYEE$\to\to$CHILD does *not* hold for T′. The reason is that $\text{CHILD}_{\text{Gauss,\$40K,1975}}$ and $\text{CHILD}_{\text{Gauss,\$50K,1976}}$ are unequal in T′; the former is {Gwendolyn,Greta}, while the latter is {Gwendolyn}.

As an aside, note that T′ is not an instance of the schema T*(EMPLOYEE, CHILD,SALARY,YEAR) since T′ does not obey the multivalued dependency EMPLOYEE$\to\to$CHILD, which is a part of the definition of the schema T*. As we see from the previous two paragraphs, multivalued dependencies may be helpful in dealing with the problem of the updating of views [7]. In this paper we do not explicitly pursue the "view updating" problem.

Note that the multivalued dependency $\mathbf{X} \to\to \mathbf{Y}$ has been defined only when $\mathbf{X}$ and $\mathbf{Y}$ are disjoint. In a later paper [2] we find it convenient to generalize slightly by allowing multivalued dependencies in which the left- and right-hand sides are not necessarily disjoint. See Section 8 of this paper for a discussion. Except in Section 8, we assume for the sake of simplicity that a multivalued dependency $\mathbf{X} \to\to \mathbf{Y}$ is defined only when $\mathbf{X}$ and $\mathbf{Y}$ are disjoint.

We can modify the definition of multivalued dependency to obtain a definition of *functional* dependency by not only requiring that $\mathbf{Y}_{\mathbf{xz}}$ depend only on $\mathbf{x}$, but by also requiring that $\mathbf{Y}_{\mathbf{xz}}$ be a set that contains at most one member. Therefore functional dependencies are special cases of multivalued dependencies (as long as the left- and right-hand sides of the functional dependency are disjoint). We record this observation in Proposition 1.

PROPOSITION 1. *If* $\mathbf{X}$ *and* $\mathbf{Y}$ *are disjoint, and if the functional dependency* $\mathbf{X} \to \mathbf{Y}$ *holds for a relation R, then the multivalued dependency* $\mathbf{X} \to\to \mathbf{Y}$ *also holds for R.*

Of course Proposition 1 (and later results) can be converted from statements about relations to statements about relation schemata. For example, Proposition 1 implies that if the functional dependency $X \rightarrow Y$ holds for relation *schema* R^* (that is, if this functional dependency holds for every instance R of R^*), then the multivalued dependency $X \rightarrow\rightarrow Y$ necessarily holds for R^* (that is, this multivalued dependency then necessarily holds for every instance R). We sometimes express Proposition 1 loosely by saying that a functional dependency "is" a multivalued dependency.

We now prove our claim in the Introduction that multivalued dependencies provide a necessary and sufficient condition for a relation to be decomposable into two of its projections without loss of information (in the sense that the original relation is guaranteed to be the join of the two projections). Recall that the ("full" or "natural") join R(X,Y,Z) of R_1(X,Y) and R_2(X,Z) is the set of tuples (x,y,z) where (x,y) is a tuple of R_1 and where (x,z) is a tuple of R_2.

THEOREM 1. $X \rightarrow\rightarrow Y$ *holds for the relation* R(X,Y,Z) *iff* R *is the join of its projections* R_1(X,Y) *and* R_2(X,Z).

PROOF. It is simple to verify that R(X,Y,Z) is the join of its projections R_1(X,Y) and R_2(X,Z) iff the following condition holds: Whenever (x,y,z) and (x,y′,z′) are tuples of R, then so are (x,y′,z) and (x,y,z′). This latter condition holds iff $Y_{xz} = Y_{xz'}$. The theorem now follows from the definition of multivalued dependencies. □

Since the right-hand side of the "iff" in Theorem 1 is symmetric in the role of Y and Z, the next proposition is immediate.

PROPOSITION 2 (Complementation). $X \rightarrow\rightarrow Y$ *holds for* R(X,Y,Z) *iff* $X \rightarrow\rightarrow Z$ *holds.*

For example, since EMPLOYEE→→CHILD holds for the relation T(EMPLOYEE,CHILD,SALARY,YEAR) in Table II, it follows from Proposition 2 that so does EMPLOYEE→→{SALARY,YEAR}. This complementation property is certainly not true in general of *functional* dependencies!

We now present a convenient generalization of our notation for multivalued dependencies. Let $X,Y_1,Y_2, \ldots, Y_k$ be sets which partition the column names of $R(X,Y_1,Y_2, \ldots, Y_k)$; that is, each column name of R is contained in exactly one of the sets $X,Y_1,Y_2, \ldots, Y_k$. By $X \rightarrow\rightarrow Y_1 \mid Y_2 \mid \cdots \mid Y_k$, we mean that $X \rightarrow\rightarrow Y_i$ holds for each set Y_i. As an example, we could write EMPLOYEE→→CHILD | {SALARY,YEAR} for the schema T*(EMPLOYEE,CHILD,SALARY,YEAR). If we consider the schema P*(EMPLOYEE,CHILD,SALARY,YEAR,SPOUSE), which is like T* except that there is an additional column SPOUSE for the name of the spouse, then P* would obey (have as part of its definition) EMPLOYEE→→CHILD | {SALARY,YEAR} | SPOUSE. This is shorthand for the three multivalued dependencies

EMPLOYEE→→CHILD,
EMPLOYEE→→{SALARY,YEAR},
EMPLOYEE→→SPOUSE.

In this case, not only would the multivalued dependency EMPLOYEE→→SPOUSE hold as above, but so would the *functional* dependency EMPLOYEE→SPOUSE (assuming no bigamy!).

It follows from Theorem 1 (which relates multivalued dependencies and joins) that $X \to\to Y_1 \mid \cdots \mid Y_k$ holds for $R(X,Y_1,\ldots,Y_k)$ iff R is the join of its projections $R_1(X,Y_1)$, $R_2(X,Y_2),\ldots,R_k(X,Y_k)$. If P is an instance of the schema P*(EMPLOYEE,CHILD,SALARY,YEAR,SPOUSE), then it follows that P is the join of its projections P_1(EMPLOYEE,CHILD), P_2(EMPLOYEE,SALARY, YEAR), and P_3(EMPLOYEE,SPOUSE). So, P can be decomposed into P_1, P_2, and P_3 without loss of information.

Note that by Proposition 2 (Complementation), $X \to\to Y$ holds for $R(X,Y,Z)$ iff $X \to\to Y \mid Z$ holds. We discuss further properties of this generalized notation in Section 9.

3. A NEW NORMAL FORM

In this section we introduce fourth normal form. We show that fourth normal form is strictly stronger than Boyce-Codd normal form [7] (an improved, stronger version of Codd's third normal form [6]). Furthermore, we show that every relation schema is decomposable into a fourth normal form family of relation schemata.

If V is a subset of U, then the functional dependency $U \to V$ necessarily holds (for every relation which contains U in its set of column names). We call such functional dependencies "trivial." A relation schema R* is in *Boyce-Codd normal form* if, whenever a nontrivial functional dependency $X \to Y$ holds for R*, then so does the functional dependency $X \to A$ for every column name A of R*. Intuitively all functional dependencies are the result of keys.

To define fourth normal form, we need the concept of "trivial multivalued dependencies" (which should not be confused with trivial *functional* dependencies, defined above). It is easy to verify that the multivalued dependencies $X \to\to \emptyset$ and $X \to\to Y$ necessarily hold for $R(X,Y)$. For example, the multivalued dependency $\{A,B\} \to\to C$ holds for every relation $R(A,B,C)$ with exactly three columns A,B,C. We call these "trivial multivalued dependencies."

A relation schema R* is in *fourth normal form* (4NF) if, whenever a nontrivial multivalued dependency $X \to\to Y$ holds for R*, then so does the *functional* dependency $X \to A$ for every column name A of R*. Intuitively all dependencies are the result of keys. In particular a 4NF relation schema can have no nontrivial multivalued dependencies that are not functional dependencies. Note the parallel between our definitions of Boyce-Codd normal form and 4NF.

THEOREM 2. *If a relation schema R* is in 4NF, then it is in Boyce-Codd normal form.*

PROOF. Assume that there is a relation schema R* that is in 4NF but not in Boyce-Codd normal form; we derive a contradiction. Since R* is not in Boyce-Codd normal form, there is a nontrivial functional dependency $X \to Y$ which holds for R*, and there is a column name A such that the functional dependency $X \to A$ does not hold for R*. Let Y_1 be the set difference $Y - X$, that is, the set of members of Y that are not in X. Since the functional dependency $X \to Y$ holds for R*, so does $X \to Y_1$. Since the functional dependency $X \to Y_1$ holds for R*, and since X and Y_1 are disjoint (by construction of Y_1), it follows from Proposition 1 that the multivalued dependency $\mathbf{X} \rightarrow\rightarrow \mathbf{Y}_1$ holds for R*. This multivalued dependency is nontrivial since (1) $\mathbf{Y}_1 \neq \emptyset$ (or else the original functional dependency $\mathbf{X} \rightarrow \mathbf{Y}$ would have been trivial) and (2) **X** and $\mathbf{Y}_1$ do not partition the column names of R* (because A is not in either **X** or $\mathbf{Y}_1$). By the definition of 4NF, since the nontrivial multivalued dependency $\mathbf{X} \rightarrow\rightarrow \mathbf{Y}_1$ holds for R*, so does the functional dependency $\mathbf{X} \rightarrow A$. This is a contradiction. □

We now show that every relation schema is decomposable into a 4NF family of relation schemata. We say that a decomposition of a relation R into a family of some of its projections is *nonloss* (or *without loss of information*) if R can be obtained from these projections by taking joins. We say that a decomposition of a relation *schema* R* is nonloss if for each instance R of R* the decomposition is nonloss.

THEOREM 3. *If a relation schema R* is not in 4NF, then there is a nonloss decomposition of R* into a 4NF family of relation schemata.*

PROOF. Assume that our original relation schema R* is not in 4NF. Then, in particular, it follows by definition of 4NF that there is a nontrivial multivalued dependency $\mathbf{X} \rightarrow\rightarrow \mathbf{Y}$ which holds for R*. Let **Z** be the set of column names of R* that are not in $\mathbf{X} \cup \mathbf{Y}$. By Theorem 2, the decomposition of R*(**X**,**Y**,**Z**) into R_1*(**X**,**Y**) and R_2*(**X**,**Z**) is nonloss (note that **Y** and **Z** are each nonempty since the multivalued dependency $\mathbf{X} \rightarrow\rightarrow \mathbf{Y}$ is nontrivial). If the relation schemata R_1* and R_2* are each in 4NF, then we are through. If not, and, say, R_1* is not in 4NF, then in particular there is a nontrivial multivalued dependency which holds for R_1* (in Section 11, we discuss issues associated with multivalued dependencies holding for projections). We decompose R_1* just as we decomposed R* above. The process must terminate since each time we decompose a relation schema we obtain two new relation schemata, each of which has a smaller number of column names than the number of column names in the relation schema we just decomposed. □

Note that putting a relation schema into 4NF does *not* necessarily decompose it "as far as possible." For example, assume that a relation schema R*(A,B,C,D) has no dependencies other than the functional dependencies $A \rightarrow B$, $A \rightarrow C$, and $A \rightarrow D$ (and consequences of these). That is, the only dependencies in R* are those that are the result of A being the key. Then R* is in 4NF, although it is possible to decompose R* without loss of information into its projections R_1*(A,B), R_2*(A,C), and R_3*(A,D). The point is that this decomposition is not necessary since it does not seem to "buy" anything.

4. AN EXAMPLE

In this section we present an example of the 4NF normalization process. We have a university database, with attributes CLASS, SECTION, STUDENT, MAJOR, EXAM, YEAR, INSTRUCTOR, RANK, SALARY, TEXT, DAY, and ROOM. Intuitively a given CLASS (such as German 101) is divided into SECTIONs (such as Section 2), each of which has one INSTRUCTOR (such as Schwarzkopf) and various STUDENTs (such as Kelly). Each CLASS has a set of TEXTs (such as Feuer's Introductory German), which are used by all SECTIONs of the CLASS. Each SECTION of a CLASS has various meeting DAYs (such as Wednesday), and on a given DAY, it meets in one ROOM (such as 353 Evans). Each STUDENT has one MAJOR (such as Math), and one YEAR (such as Sophomore). A STUDENT in a given CLASS and SECTION has several EXAM scores (such as 93 percent). Each INSTRUCTOR has one RANK (such as Associate Professor) and one SALARY (such as $20K).

We begin the normalization process by forming a single relation schema

$$\begin{aligned}&R^*(\text{CLASS,SECTION,STUDENT,MAJOR,EXAM,YEAR,}\\&\text{INSTRUCTOR,RANK,SALARY,TEXT,DAY,ROOM})\end{aligned} \tag{4.1}$$

with all of these attributes as column names. What are the dependencies which are part of the definition of this schema?

We have the following functional dependencies (note that for simplicity we do not distinguish between a singleton set {A} and its only member A; e.g. we write INSTRUCTOR for {INSTRUCTOR}):

$$\{\text{CLASS,SECTION}\}\rightarrow\text{INSTRUCTOR}, \tag{4.2}$$
$$\{\text{CLASS,SECTION,DAY}\}\rightarrow\text{ROOM}, \tag{4.3}$$
$$\text{STUDENT}\rightarrow\{\text{MAJOR,YEAR}\}, \tag{4.4}$$
$$\text{INSTRUCTOR}\rightarrow\{\text{RANK,SALARY}\}. \tag{4.5}$$

What are the multivalued dependencies? First of all, we consider those multivalued dependencies with {CLASS,SECTION} as the left-hand side. Using the generalized notation at the end of Section 2, we have

$$\begin{aligned}&\{\text{CLASS,SECTION}\}\rightarrow\rightarrow\{\text{STUDENT,MAJOR,EXAM,YEAR}\}\mid\\&\{\text{INSTRUCTOR,RANK,SALARY}\}\mid\text{TEXT}\mid\{\text{DAY,ROOM}\}.\end{aligned} \tag{4.6}$$

That is, we have

$$\{\text{CLASS,SECTION}\}\rightarrow\rightarrow\{\text{STUDENT,MAJOR,EXAM,YEAR}\}, \tag{4.7}$$
$$\{\text{CLASS,SECTION}\}\rightarrow\rightarrow\{\text{INSTRUCTOR,RANK,SALARY}\}, \tag{4.8}$$
$$\{\text{CLASS,SECTION}\}\rightarrow\rightarrow\text{TEXT}, \tag{4.9}$$
$$\{\text{CLASS,SECTION}\}\rightarrow\rightarrow\{\text{DAY,ROOM}\}. \tag{4.10}$$

Since we assume in this example that all sections of a given class use the same textbooks, our relation schema R^* obeys the multivalued dependency

$$\text{CLASS}\rightarrow\rightarrow\text{TEXT}, \tag{4.11}$$

which is even stronger than (4.9). (It is simple to verify that if $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}$, then $\mathbf{X} \cup \mathbf{V}\rightarrow\rightarrow\mathbf{Y}$, whenever $\mathbf{V}$ is disjoint from $\mathbf{Y}$. It follows that (4.11) implies (4.9)).

Finally, we have

$$\{\text{CLASS, SECTION,STUDENT}\}\rightarrow\rightarrow\text{EXAM}. \tag{4.12}$$

We now begin the 4NF normalization process. On the basis of the multivalued dependency (4.7), we decompose R^* (as given by (4.1)) into

R_1^*(CLASS,SECTION,STUDENT,MAJOR,EXAM,YEAR) and
R_2^*(CLASS,SECTION,INSTRUCTOR,RANK,SALARY, (4.13)
TEXT,DAY,ROOM).

In Section 11 we prove that if a multivalued dependency $\mathbf{X} \rightarrow\rightarrow \mathbf{Y}$ holds for a relation, then it also holds for every projection that contains at least all of the column names $\mathbf{X} \cup \mathbf{Y}$ (in fact, we prove a slightly stronger result). Hence, since the multivalued dependency (4.12) holds for R^*, it also holds for its projection R_1^*. Therefore R_1^* can be decomposed into

R_{11}^*(CLASS,SECTION,STUDENT,EXAM) and
R_{12}^*(CLASS,SECTION,STUDENT,MAJOR,YEAR).

Although R_{11}^* is now in 4NF, R_{12}^* is not since STUDENT→{MAJOR,YEAR} (functional dependency (4.4)), whereas it is not the case that STUDENT→CLASS or STUDENT→SECTION. Using (4.4), we decompose R_{12}^* into two 4NF schemata

R_{121}^*(STUDENT,MAJOR,YEAR) and
R_{122}^*(CLASS,SECTION,STUDENT).

We now decompose R_2^* in (4.13). By (4.8), it is possible to decompose R_2^* into

R_{21}^*(CLASS,SECTION,INSTRUCTOR,RANK,SALARY) and
R_{22}^*(CLASS,SECTION,TEXT,DAY,ROOM).

Using functional dependency (4.5), we decompose R_{21}^* into

R_{211}^*(INSTRUCTOR,RANK,SALARY) and
R_{212}^*(CLASS,SECTION,INSTRUCTOR).

Using multivalued dependency (4.11), we decompose R_{22}^* into

R_{221}^*(CLASS,TEXT) and
R_{222}^*(CLASS,SECTION,DAY,ROOM).

We are now left with a 4NF family. It is possible to remove the relation schema R_{122}^*(CLASS,SECTION,STUDENT) from the family since it is a projection of R_{11}^*(CLASS,SECTION,STUDENT,EXAM), which is already in the family (hence R_{122}^* is redundant.) As our resulting 4NF family, we are left with

R_{11}^*(CLASS,SECTION,STUDENT,EXAM),
R_{121}^*(STUDENT,MAJOR,YEAR),
R_{211}^*(INSTRUCTOR,RANK,SALARY),
R_{212}^*(CLASS,SECTION,INSTRUCTOR),
R_{221}^*(CLASS,TEXT),
R_{222}^*(CLASS,SECTION,DAY,ROOM).

We note that our final result could have been different if we had decomposed differently. An interesting (and potentially important) research problem is to determine, given a set of functional dependencies and multivalued dependencies, how to form an "optimal" 4NF family (part of the problem is to define "optimal"). There are several heuristics that might be employed. For example, Zaniolo [13] suggests that if on a given step there is a choice of applying two multivalued dependencies $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ and $\mathbf{W} \twoheadrightarrow \mathbf{Y}$, where $\mathbf{X}$ is a subset of $\mathbf{W}$, then $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ should be applied rather than $\mathbf{W} \twoheadrightarrow \mathbf{Y}$. The reason is that the former multivalued dependency is stronger than (i.e. implies) the latter, as noted earlier in this section.

As another example of a heuristic; assume that $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ and $\mathbf{Y} \twoheadrightarrow \mathbf{Z}$ each hold for $R^*(\mathbf{X},\mathbf{Y},\mathbf{Z})$. It might be preferable to decompose R^* on the basis of the multivalued dependency $\mathbf{Y} \twoheadrightarrow \mathbf{Z}$ to obtain $R_1^*(\mathbf{X},\mathbf{Y})$ and $R_2^*(\mathbf{Y},\mathbf{Z})$ rather than to decompose on the basis of the multivalued dependency $\mathbf{X} \twoheadrightarrow \mathbf{Y}$, which gives $R_1'^*(\mathbf{X},\mathbf{Y})$ and $R_2'^*(\mathbf{X},\mathbf{Z})$. Rissanen [11] gives theoretical reasons that justify this choice. (Rissanen works in the context of functional, rather than multivalued, dependencies; however, his arguments carry over to the multivalued case). Intuitively the dependency of $\mathbf{Z}$ on $\mathbf{X}$ is "deduced" (by transitivity), and so the schema $R_2'^*(\mathbf{X},\mathbf{Z})$ is "derived," not "primitive."

By initially forming a single large relation schema (as in (4.1)), a 4NF normalization algorithm has at least as many options as if it begins with many small relation schemata which are then decomposed further. That is, there are at least as many possible final results in the former case. Hence an algorithm has more of a chance to optimize.

If there is a human in the normalization loop, then the problem of determining all functional and multivalued dependencies that hold in a given situation seems less formidable because the human can "notice" a previously neglected dependency at a late stage of the 4NF normalization process, and then can either apply it at that stage or incorporate it in the list of "known" dependencies and start over.

5. OTHER WAYS TO VIEW MULTIVALUED DEPENDENCIES

Multivalued dependencies are a more complex concept than are functional dependencies. Therefore it is helpful to look at them in more than one way in order to understand them better.

The proof of Theorem 1 gives us a "constructive" characterization of multivalued dependencies. This characterization (Proposition 3 below) says that if certain tuples are present in a relation that satisfies a certain multivalued dependency, then certain other tuples must also appear.

PROPOSITION 3. $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ *holds for* $R(\mathbf{X},\mathbf{Y},\mathbf{Z})$ *iff whenever* $(\mathbf{x},\mathbf{y},\mathbf{z})$ *and* $(\mathbf{x},\mathbf{y}',\mathbf{z}')$ *are tuples of* R, *then so are* $(\mathbf{x},\mathbf{y}',\mathbf{z})$ *and* $(\mathbf{x},\mathbf{y},\mathbf{z}')$.

As a demonstration of Proposition 3, consider again our example T(EMPLOYEE,CHILD,SALARY,YEAR) in Table II. Since EMPLOYEE$\twoheadrightarrow$CHILD holds for T, and since (Gauss, Gwendolyn, $40K, 1975) and (Gauss, Greta, $50K, 1976) are tuples of T, it follows that (Gauss, Greta, $40K, 1975) and (Gauss, Gwendolyn,$50K,1976) are also tuples of T.

Just as functional dependencies are constraints as to which tuples may appear in a relation, so too are multivalued dependencies by Proposition 3. Thus multivalued dependencies, like functional dependencies, are "integrity constraints" on the "form" of the relation.

Another helpful viewpoint is that the multivalued dependency $\mathbf{X} \rightarrow\rightarrow \mathbf{Y}$ holds for $R(\mathbf{X},\mathbf{Y},\mathbf{Z})$ iff $\mathbf{Y}$ and $\mathbf{Z}$ are "orthogonal" or "independent" sets of column names. In T(EMPLOYEE,CHILD,SAL'RY,YEAR), the fact that EMPLOYEE$\rightarrow\rightarrow$ CHILD holds would be interpreted as saying that CHILD and {SALARY,YEAR} are orthogonal. This orthogonality holds in the sense that the set of tuples of T with, say, Gauss as the EMPLOYEE entry is the set of four tuples

$$\{\text{Gauss}\} \times \{\text{Gwendolyn,Greta}\} \times \{(\$40\text{K},1975),(\$50\text{K},1976)\}.$$

Yet another viewpoint is that a multivalued dependency $\mathbf{X} \rightarrow\rightarrow \mathbf{Y}$ (and its "complement" $\mathbf{X} \rightarrow\rightarrow \mathbf{Z}$) imply "independent relationships" in $R^*(\mathbf{X},\mathbf{Y},\mathbf{Z})$. That is, the intuitive meaning of a multivalued dependency $\mathbf{X} \rightarrow\rightarrow \mathbf{Y}$ holding for a relation schema $R^*(\mathbf{X},\mathbf{Y},\mathbf{Z})$ is that we really have two "independent relation schemata" $R_1^*(\mathbf{X},\mathbf{Y})$ and $R_2^*(\mathbf{X},\mathbf{Z})$.

6. A SPECIAL MULTIVALUED DEPENDENCY

We briefly consider the interesting special case $\emptyset \rightarrow\rightarrow \mathbf{Y}$ of multivalued dependencies (where the left-hand side is the empty set).

It follows from our definition that the multivalued dependency $\emptyset \rightarrow\rightarrow \mathbf{Y}$ holds for the relation schema $R^*(\mathbf{Y},\mathbf{Z})$ precisely if R^* is the Cartesian product of its projections $R_1^*(\mathbf{Y})$ and $R_2^*(\mathbf{Z})$. (In the definition of multivalued dependencies in Section 2, we make the natural convention that if $\mathbf{X} = \emptyset$ then $\mathbf{Y}_{\mathbf{xz}} = \{\mathbf{y} : (\mathbf{y},\mathbf{z}) \in R\}$.) Further, the natural definition of the (degenerate) functional dependency $\emptyset \rightarrow A$ (where A is a column name) is that there is exactly one value that appears in the A column. Under these conventions, one special case of our 4NF requirements is that if a relation schema $R^*(\mathbf{Y},\mathbf{Z})$ is the Cartesian product of its projections $R_1^*(\mathbf{Y})$ and $R_2^*(\mathbf{Z})$, then R^* is not in 4NF and must be decomposed to obtain 4NF. Actually there is one (strange) case in which $R^*(\mathbf{Y},\mathbf{Z})$ is in 4NF in spite of $R^*(\mathbf{Y},\mathbf{Z})$ being the Cartesian product of $R_1^*(\mathbf{Y})$ and $R_2^*(\mathbf{Z})$: namely if the functional dependencies $\emptyset \rightarrow A$ hold for every column name A (in other words, each instance R of the relation schema R^* contains no more than one tuple!).

Although every binary relation schema (that is, every relation schema with exactly two column names) is in Boyce-Codd normal form (as the reader can easily verify), it is not quite true that every binary relation schema is in 4NF. The only exception that can occur is if a binary relation schema $R^*(A,B)$ is the Cartesian product of $R_1^*(A)$ and $R_2^*(B)$. The reason that this is the only possible exception is that the only possible nontrivial multivalued dependency for $R^*(A,B)$ is $\emptyset \rightarrow\rightarrow A$ (or equivalently $\emptyset \rightarrow\rightarrow B$).

7. TRANSITIVITY

Assume that $\mathbf{X} \rightarrow\rightarrow \mathbf{Y}$ and $\mathbf{Y} \rightarrow\rightarrow \mathbf{Z}$. Is it necessarily true that $\mathbf{X} \rightarrow\rightarrow \mathbf{Z}$? The answer is no since $\mathbf{X}$ and $\mathbf{Z}$ need not be disjoint, and so the multivalued dependency $\mathbf{X} \rightarrow\rightarrow \mathbf{Z}$ might not even make sense. However, if $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ are pairwise disjoint (so that all of the multivalued dependencies $\mathbf{X} \rightarrow\rightarrow \mathbf{Y}$, $\mathbf{Y} \rightarrow\rightarrow \mathbf{Z}$, and $\mathbf{X} \rightarrow\rightarrow \mathbf{Z}$ make sense), then transitivity holds; that is, if $X \rightarrow\rightarrow Y$ and $Y \rightarrow\rightarrow Z$ hold for R(X,Y,Z,W), then so does $X \rightarrow\rightarrow Z$. Note that we need not assume that X, Y, and Z by themselves partition the column names.

THEOREM 4 (Transitivity). *If* X, Y, *and* Z *are pairwise disjoint* (*but do not necessarily partition the column names of relation R*), *and if both* $X \rightarrow\rightarrow Y$ *and* $Y \rightarrow\rightarrow Z$ *hold for R, then* $X \rightarrow\rightarrow Z$ *holds for R.*

PROOF. We make use of the characterization of multivalued dependencies in Proposition 3 (Section 5). Assume that X, Y, Z, W partition the column names. We assume that the multivalued dependencies $X \rightarrow\rightarrow Y$ and $Y \rightarrow\rightarrow Z$ hold for relation R and show that the multivalued dependency $X \rightarrow\rightarrow Z$ holds for R. To show that $X \rightarrow\rightarrow Z$ holds for R, we must show (by Proposition 3) that if

(1) (x,y,z,w) and
(2) (x,y',z',w')

are tuples of R, then also

(3) (x,y,z',w) and
(4) (x,y',z,w')

are tuples of R.

Assume that R contains tuples (1) and (2) above (that is, the tuples (x,y,z,w) and (x,y',z',w')). Since $X \rightarrow\rightarrow Y$ holds for R, it follows from Proposition 3 that in addition to tuples (1) and (2), relation R contains the tuples

(5) (x,y',z,w) and
(6) (x,y,z',w').

Since R contains tuples (1) and (6), and since $Y \rightarrow\rightarrow Z$ holds for R, it follows from Proposition 3 that R contains (x,y,z',w), that is, tuple (3) above, and also (x,y,z,w').

Since R contains tuples (2) and (5) and since $Y \rightarrow\rightarrow Z$ holds for R, once again we use Proposition 3 to find that R contains (x,y',z,w'), that is, tuple (4) above, and also (x,y',z',w). We have shown that R contains (among others) tuples (3) and (4) above, as desired. □

The fact that multivalued dependencies are transitive is one argument in favor of using an arrow notation (as we have done), rather than talking only in terms of nonloss decompositions.

It would be nice to remove the requirement that the left- and right-hand sides of a multivalued dependency be disjoint. Unfortunately, under the natural modification of our definition, transitivity fails. We discuss this phenomenon in Section 8.

8. THE DISJOINTNESS OF THE LEFT- AND RIGHT-HAND SIDES

Throughout this paper we have required that the left- and right-hand sides of a multivalued dependency be disjoint. How should we modify our definition to eliminate this restriction?

Let R be a relation, and let X and Y be (not necessarily disjoint) subsets of the column names of R. Let Z be the set of column names of R that are not in $X \cup Y$.

Based on Theorem 1, it seems that the most natural modification of our definition is that $X \rightarrow\rightarrow Y$ holds for R iff R is the join of its projections $R_1(X,Y)$ and $R_2(X,Z)$. Equivalently, the "modified" multivalued dependency $X \rightarrow\rightarrow Y$ (where X and Y need not be disjoint) holds for R iff the (usual) multivalued dependency $XV \rightarrow\rightarrow$ holds for R, where V is the set difference $Y - X$. Note that the modified and the *corresponding usual multivalued dependencies say the same thing if the left- and* right-hand sides of the modified multivalued dependency are disjoint.

By using this modified definition (and thus allowing a more general "multivalued dependency"), it is possible to exhibit [2] a complete axiomatization of multivalued dependencies in which none of the axioms are very complex.

One might hope that under this modified definition, transitivity would always hold: that is, that if the (modified) multivalued dependencies $X \rightarrow\rightarrow Y$ and $Y \rightarrow\rightarrow Z$ hold for a relation R (which has as its column names the union of X, Y, and Z, and perhaps more), then $X \rightarrow\rightarrow Z$ would hold for R. Unfortunately, this is not always the case. For example, consider the 5-column relation Q(A,B,C,D,E) in Table IV (where $b \neq b'$, etc.) The multivalued dependency $A \rightarrow\rightarrow \{B,C\}$ (or as we write it, $A \rightarrow\rightarrow BC$) holds for Q, as does the modified multivalued dependency $BC \rightarrow\rightarrow CD$ (for the modified multivalued dependency $BC \rightarrow\rightarrow CD$ to hold, we mean by definition that the usual multivalued dependency $BC \rightarrow\rightarrow D$ holds). Since the modified multivalued dependencies $A \rightarrow\rightarrow BC$ and $BC \rightarrow\rightarrow CD$ hold for Q, we would expect by transitivity that the modified multivalued dependency $A \rightarrow\rightarrow CD$ would hold for Q; however, it does not. Thus, under this modified definition, general transitivity fails.

In [2], it is shown that if the modified multivalued dependencies $X \rightarrow\rightarrow Y$ and $Y \rightarrow\rightarrow Z$ hold for a relation R, then although the modified multivalued dependency $X \rightarrow\rightarrow Z$ need not hold, it is always the case that the modified multivalued dependency $X \rightarrow\rightarrow Z - Y$ does hold.

Table IV

A	B	C	D	E
a	b	c	d	e
a	b′	c′	d′	e′
a	b	c	d′	e′
a	b′	c′	d	e
a	b	c	d	e′
a	b	c	d′	e
a	b′	c′	d′	e
a	b′	c′	d	e′

9. A PARTITIONING RESULT

In this section we show that it is possible to summarize a set of multivalued dependencies that all have the same left-hand side X by a single generalized multivalued dependency $X \rightarrow\rightarrow Y_1 \mid \cdots \mid Y_k$ (as defined in Section 2).

By using Proposition 3, it is straightforward to show the following two properties of multivalued dependencies (Propositions 4 and 5). In Proposition 5, by the set difference $\mathbf{Y}_1 - \mathbf{Y}_2$, we mean the set of members of $\mathbf{Y}_1$ that are not in $\mathbf{Y}_2$.

PROPOSITION 4. *If* $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_1$ *and* $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_2$, *then* $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_1 \cup \mathbf{Y}_2$.

We note that the converse to Proposition 4 fails (although the analogous result holds for *functional* dependencies). For example, although EMPLOYEE$\rightarrow\rightarrow$ {SALARY,YEAR} holds for T(EMPLOYEE,CHILD,SALARY,YEAR) in Table II, neither EMPLOYEE$\rightarrow\rightarrow$SALARY nor EMPLOYEE$\rightarrow\rightarrow$YEAR holds. We now demonstrate that EMPLOYEE$\rightarrow\rightarrow$YEAR fails for T. If it held, then $\mathrm{YEAR}_{e,c,s}$ (where e is an employee, c is a child, and s is a salary) would depend only on e. However, $\mathrm{YEAR}_{\mathrm{Pythagoras,Peter,\$15K}} \neq \mathrm{YEAR}_{\mathrm{Pythagoras,Peter,\$20K}}$.

PROPOSITION 5. *If* $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_1$ *and* $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_2$, *then* $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_1 \cap \mathbf{Y}_2$, $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_1 - \mathbf{Y}_2$, *and* $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_2 - \mathbf{Y}_1$. *That is, if* $\mathbf{X}\rightarrow\rightarrow\mathbf{U} \cup \mathbf{V}_1$ *and* $\mathbf{X}\rightarrow\rightarrow\mathbf{U} \cup \mathbf{V}_2$, *where* $\mathbf{U}$, $\mathbf{V}_1$, $\mathbf{V}_2$ *are disjoint, then* $\mathbf{X}\rightarrow\rightarrow\mathbf{U}$, $\mathbf{X}\rightarrow\rightarrow\mathbf{V}_1$, *and* $\mathbf{X}\rightarrow\mathbf{V}_2$.

We now discuss the claim in the first paragraph of this section. Recall from Section 2 that when we say that $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_1 \mid \cdots \mid \mathbf{Y}_k$ holds for $\mathrm{R}(\mathbf{X},\mathbf{Y}_1,\ldots,\mathbf{Y}_k)$, we mean that $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_i$ holds for each of the sets $\mathbf{Y}_i$. As before, the sets $\mathbf{X},\mathbf{Y}_1,\ldots,\mathbf{Y}_k$ partition the column names of R. By Proposition 4, it follows from $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_1 \mid \cdots \mid \mathbf{Y}_k$ that $\mathbf{X}\rightarrow\rightarrow\mathbf{V}$ holds for each union $\mathbf{V}$ of a subset of $\{\mathbf{Y}_1,\ldots,\mathbf{Y}_k\}$. For example, A$\rightarrow\rightarrow$BC | D | E implies that each of the following eight multivalued dependencies holds (as before, we write BC for {B,C}, etc.): A$\rightarrow\rightarrow\emptyset$, A$\rightarrow\rightarrow$BC, A$\rightarrow\rightarrow$D, A$\rightarrow\rightarrow$E, A$\rightarrow\rightarrow$BCD, A$\rightarrow\rightarrow$BCE, A$\rightarrow\rightarrow$DE, A$\rightarrow\rightarrow$BCDE. From Propositions 4 and 5, it follows that a collection $\mathbf{X}\rightarrow\rightarrow\mathbf{W}_1,\ldots,\mathbf{X}\rightarrow\rightarrow\mathbf{W}_r$ of multivalued dependencies (all with the same left-hand side $\mathbf{X}$) hold for a relation R iff the single generalized multivalued dependency $\mathbf{X}\rightarrow\rightarrow\mathbf{Y}_1 \mid \cdots \mid \mathbf{Y}_k$ holds for R, where $\{\mathbf{Y}_1,\ldots,\mathbf{Y}_k\}$ is the smallest set (k minimal) such that

(a) $\mathbf{X},\mathbf{Y}_1,\ldots,\mathbf{Y}_k$ partition the set of column names of R.
(b) Each $\mathbf{W}_j$ is a union of a subset of $\{\mathbf{Y}_1,\ldots,\mathbf{Y}_k\}$.

For example, if the column names are A,B, . . . ,F, then the three multivalued dependencies A$\rightarrow\rightarrow$BC, A$\rightarrow\rightarrow$CD, A$\rightarrow\rightarrow$EF are equivalent to the single generalized multivalued dependency A$\rightarrow\rightarrow$B | C | D | EF. That is, the three multivalued dependencies A$\rightarrow\rightarrow$BC, A$\rightarrow\rightarrow$CD, and A$\rightarrow\rightarrow$EF hold for R(A,B,C,D,E,F) iff the single generalized multivalued dependency A$\rightarrow\rightarrow$B | C | D | EF does.

10. INTERACTIONS BETWEEN FUNCTIONAL AND MULTIVALUED DEPENDENCIES

Functional dependencies and multivalued dependencies interact in complicated ways. For example, if a relation R has column names A, B, C, and perhaps others, and if the multivalued dependency A$\rightarrow\rightarrow$B and the functional dependency C$\rightarrow$B hold for R, then the functional dependency A$\rightarrow$B necessarily holds for R. A complete characterization of the interplay between functional and multivalued dependencies appears in [2].

11. EMBEDDED MULTIVALUED DEPENDENCIES

Multivalued dependencies "project down" in the following manner.

THEOREM 5. *Assume that the multivalued dependency* $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ *holds for relation* $R(\mathbf{X},\mathbf{Y},\mathbf{Z})$. *Let* $R'(\mathbf{X},\mathbf{Y}',\mathbf{Z}')$ *be a projection of R, where* $\mathbf{Y}' \subseteq \mathbf{Y}$ *and* $\mathbf{Z}' \subseteq \mathbf{Z}$. *Thus* R' *has (among others) all of the columns* **X**. *Then the multivalued dependency* $\mathbf{X} \twoheadrightarrow \mathbf{Y}'$ *holds for* R'.

PROOF. By Theorem 1, we know that R(X,Y,Z) is the join of its projections $R_1(X,Y)$ and $R_2(X,Z)$. It is easy to see that the projection $R'(X,Y',Z')$ is then the join of its projections $R_1'(X,Y')$ and $R_2'(X,Z')$. By Theorem 1 again, the multivalued dependency $X \twoheadrightarrow Y'$ holds for R'. □

For example, since EMPLOYEE$\twoheadrightarrow${SALARY,YEAR} holds for T(EMPLOYEE,CHILD,SALARY,YEAR) in Table II, it follows from Theorem 5 that EMPLOYEE$\twoheadrightarrow$YEAR holds for the projection T_1(EMPLOYEE,CHILD, YEAR).

We have the following important special case of Theorem 5.

COROLLARY 1. *Assume that the multivalued dependency* $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ *holds for relation* $R(\mathbf{X},\mathbf{Y},\mathbf{Z})$. *Let* $R'(\mathbf{X},\mathbf{Y},\mathbf{Z}')$ *be a projection of R which has (among others) all of the columns* **X** *and* **Y**. *Then the multivalued dependency* $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ *holds for relation* R' *also.*

For example, since EMPLOYEE$\twoheadrightarrow$CHILD holds for T(EMPLOYEE, CHILD,SALARY,YEAR), it also holds for the projection T_1(EMPLOYEE, CHILD,YEAR).

We see from the example that follows Theorem 5 that multivalued dependencies are sensitive to their context in a sense in which functional dependencies are not. Thus EMPLOYEE$\twoheadrightarrow$YEAR holds for T_1(EMPLOYEE,CHILD,YEAR), but as we saw in Section 9, it does not hold for T(EMPLOYEE,CHILD,SALARY, YEAR). Note that this problem of context never arises for functional dependencies since, for example, if $R'(A,B,C)$ is the projection of a relation R(A,B,C,D), then the functional dependency $A \rightarrow B$ holds for R if and only if it holds for R'.

This context problem makes it technically convenient to define "embedded multivalued dependencies." If $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ (or equivalently, $\mathbf{X} \twoheadrightarrow \mathbf{Y} \mid \mathbf{Z}$) holds for the projection $R_1(\mathbf{X},\mathbf{Y},\mathbf{Z})$ of $R(\mathbf{X},\mathbf{Y},\mathbf{Z},\mathbf{W})$, then we say that the *embedded multivalued dependency* $\mathbf{X} \twoheadrightarrow \mathbf{Y} \mid \mathbf{Z}$ holds for the bigger relation $R(\mathbf{X},\mathbf{Y},\mathbf{Z},\mathbf{W})$. In the case we discussed, the embedded multivalued dependency EMPLOYEE$\twoheadrightarrow$ YEAR | CHILD holds for T(EMPLOYEE,CHILD,SALARY,YEAR). In this case the embedded multivalued dependency is simply a "projection" of an ordinary multivalued dependency (via Theorem 5). Unfortunately Theorem 5 does not have the natural converse; that is, there can be embedded multivalued dependencies that are not the "projections" of ordinary multivalued dependencies. Such examples seem to occur infrequently in practice. When they do occur, one can simply include embedded multivalued dependencies along with ordinary multivalued dependencies and functional dependencies as integrity constraints for a relation schema.

As an example of a relation with an embedded multivalued dependency that is not the projection of an ordinary multivalued dependency, consider the relation N(A,B,C,D) in Table V (where $a \neq a'$, etc.). The embedded multivalued dependency $A \twoheadrightarrow B \mid C$ holds for N(A,B,C,D) (in other words, the multivalued de-

Table V

A	B	C	D
a	b	c	d
a	b′	c′	d
a	b′	c	d′
a	b	c′	d′
a′	b′	c	d

pendency $A \to\to B$ holds for the projection $N_1(A,B,C)$). However, neither $A \to\to BD$ nor any other nontrivial multivalued dependency holds for $N(A,B,C,D)$.

12. SUMMARY

We have introduced "multivalued dependencies," which we believe significantly extend the understanding of the logical design of relational databases. Multivalued dependencies, which are a generalization of the well-known functional dependencies, provide a necessary and sufficient condition for a relation schema to be decomposable into a family of relation schemata without loss of information. Multivalued dependencies lead to a new 4NF, which is strictly stronger than Boyce-Codd normal form.

ACKNOWLEDGMENTS

The author had many enjoyable and helpful conversations with John Howard, Jean-Marie Cadiou, and Steve Zilles about this research. The author is grateful for a careful reading of various versions of this paper by them and by Bill Kent, Jorma Rissanen, and John Smith. The readability of this paper was enhanced by many comments from Phil Bernstein, who courageously read every draft. Howard suggested the generalized notation $X \to\to Y_1 \mid \cdots \mid Y_k$ and Bernstein, Cadiou, and Howard suggested alternate ways to view multivalued dependencies (Section 5). Special thanks go to Ted Codd for his warm enthusiasm in encouraging the author to pursue these ideas.

REFERENCES

1. Armstrong, W.W. Dependency structures of database relationships. Information Processing 74, North-Holland Pub. Co., Amsterdam, 1974, pp. 580–583.
2. Beeri, C., Fagin, R., and Howard, J.H. A complete axiomatization for functional and multivalued dependencies in database relations. Proc. ACM SIGMOD Conf., D.C.P. Smith, Ed., Toronto, Canada, August 1977, pp. 47–61.
3. Bernstein, P.A. Synthesizing third normal form relations from functional dependencies. *ACM Trans. Database Syst. 1*, 4 (Dec. 1976), 277–298.
4. Bernstein, P.A., Swenson, J.R., and Tsichritzis, D.C. A unified approach to functional dependencies and relations. Proc. ACM SIGMOD Conf., W.F. King, Ed., San Jose, Calif., May 1975, pp. 237–245.
5. Cadiou, J.-M. On semantic issues in the relational model of data. Proc. Int. Symp. on

Math. Foundations of Comptr. Sci., Gdańsk, Poland, Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, Sept. 1975.

6. CODD, E.F. Further normalization of the data base relational model. In *Courant Computer Science Symposium 6: Data Base Systems*, Prentice-Hall, Englewood Cliffs, N.J., May 1971, pp. 65–98.
7. CODD, E.F. Recent investigations in relational data base systems. Information Processing 74, North-Holland Pub. Co., Amsterdam, 1974, pp. 1017–1021.
8. DELOBEL, C., AND CASEY, R.G. Decomposition of a data base and the theory of Boolean switching functions. *IBM J. Res. and Develop. 17*, 5 (Sept. 1973), 374–386.
9. DELOBEL, C., AND LÉONARD, M. The decomposition process in a relational model. Proc. Int. Workshop on Data Structure Models for Information Systems, Presses U. de Namur, Namur, Belgium, May 1974, pp. 57–80.
10. FAGIN, R. Functional dependencies in a relational database and propositional logic. *IBM J. Res. and Develop. 21*, 6 (Nov. 1977).
11. RISSANEN, J.J. Independent components of relations. Res. Rep. RJ1899, IBM Res. Lab., San Jose, Calif., Jan. 1977.
12. SCHMID, H.A., AND SWENSON, J.R. On the semantics of the relational data model. Proc. ACM SIGMOD Conf., W.F. King, Ed., San Jose, Calif., May 1975, pp. 211–223.
13. ZANIOLO, C. Analysis and design of relational schemata for database systems. Ph.D. Diss., Tech. Rep. UCLA-ENG-7669, U. of California, Los Angeles, Calif., July 1976.

Received July 1976; revised December 1976