The Application of Functional Dependency Theory to Relational Databases

C. Robert Carlson and Adarsh K. Arora
Bell Laboratories Naperville-Wheaton Road, Naperville, Illinois 60566, USA
Miroslava Milosavljevic Carlson
Northeastern Illinois University, Bryn Mawr at St Louis Avenue, Chicago, Illinois 60625, USA

This paper examines three areas where the application of functional dependency theory to relational databases has had an impact. These areas are relational views, database translation and logical database design. The paper refines our earlier work on relational views in which the concepts of consistent and updatable views were proposed. In the area of database translation, it identifies six levels of information preserving relational transformations. Finally, a relational database design algorithm is proposed which combines the best features of the classical synthesis and decomposition approaches while avoiding their identified shortcomings.

INTRODUCTION

This paper shows that functional dependency theory provides a unified framework for the analysis of several database problems. The paper represents both a synthesis and extension of our earlier work on each of these problems. The first section contains background information to familiarize the reader with the terminology and concepts of functional dependency theory.

In the second section, functional dependencies are used to define the concepts of consistent and updatable views. Basically, a 'consistent view' is one whose interpretation is consistent with that of the underlying database. An 'updatable view' is one whose updates can be translated into appropriate updates of the underlying database. Together, these definitions describe a class of useable views.

In section three, six levels of information preserving relational transformations are identified. These levels reflect the different degrees with which transformations preserve the update and retrieval properties of databases. These levels are important since previously many applications have mistakenly assumed that all the information embodied in their database is preserved by the restructuring operations they have employed.

Finally, some shortcomings of the classical synthesis and decomposition approaches to relationship database design are identified in the fourth section. A new algorithm is then proposed which combines the best features of these approaches while avoiding their identified shortcomings.

BACKGROUND

Attributes are identifiers taken from a finite set $\{A_1, A_2, \ldots, A_m\}$. Each A_i has associated with it a domain, denoted DOM(A_i), which is the set of possible values for that attribute. We shall use the letters $A, B, \ldots$, for single attributes and the letters X, Y, . . ., for sets of attributes. For simplicity, we write XY for the union of X and Y.

A *relation* on an attribute set $T = \{A_1, A_2, \ldots, A_n\}$ is a finite subset of the Cartesian product $DOM(A_1) \times DOM(A_2) \times \cdots \times DOM(A_n)$ and is denoted by either R(T) or $R(A_1, A_2, \ldots, A_n)$. The elements of a relation are called *tuples*.

A set of complete primitive operators capable of manipulating relations has been defined by Codd[1]. Two operators will be of particular interest to us: projection and equi-join. The *projection* of R(X, Y, Z) over a set of attributes X is denoted R[X], and is defined to be the set $\{x \mid \langle x, y, z\rangle \in R\}$. The *equi-join* of relations R1(X, Y) and R2(Y, Z) over the attribute set Y is denoted R1*R2 and is defined to be the set $\{\langle x, y, z\rangle \mid \langle x, y\rangle \in R1$ and $\langle y, z\rangle \in R2\}$.

The projection operator can also be defined for individual tuples of a relation. If u is a tuple in R, then u[X] is the X-component of u. If the X-component of each tuple always uniquely identifies that tuple and no proper subset of X has this property, then X is a *key* of R. It is possible for a relation to have several keys. It is usual to impose the constraint that no component of a key value may be null.

Let X and Y be arbitrary attribute sets in some relation R(T). A *functional dependency* (FD) from X to Y, denoted $X \rightarrow Y$ exists in R(T) if, independent of time, for every X-value that appears in R, the corresponding Y-value is unique. For a particular R(T), $X \rightarrow Y$ is a set $S = \{\langle x, y\rangle \mid u \in R$ and $u[X] = x$ and $u[Y] = y\}$ with the property that $\langle x1, y1\rangle \in S$ and $\langle x1, y2\rangle \in S$ implies that $y1 = y2$. $X \rightarrow Y$ is a *nontrivial FD* if $Y \not\subseteq X$ and a *full FD* if for any $V \subset X$, $V \not\rightarrow Y$.

We assume that a relational database consists of a set of relations $\{Ri\langle Ti, Di\rangle \mid 1 \leq i \leq n\}$ such that each relation R*i* is defined over some set of attributes T*i* and D*i* denotes the set of functional dependencies in R*i*. This description of the database is called its *intension*. R*i*(T*i*) denotes the *extension* (set of tuples) of relation R*i*. Obviously, all instances of R*i*(T*i*) must satisfy the functional dependencies of D*i*.

Given a set of FDs in a relation, it is often possible to deduce other FDs that also hold in that relation. Given an intension $R\langle T, D\rangle$ and an FD g, it is said that D implies g if g holds in every instance of R(T). It is possible to decide whether g is implied by D using systems of inference rules.[2,3] A system of inference rules is (i) *sound* if every g generated by it is in fact an FD, (ii) *complete* if

CCC-0010-4620/82/0025-0068 $03.00

every FD implied by D can be generated by it, and (iii) *nonredundant* if no inference rule in this system is superfluous. Fig. 1 shows a system of inference rules which is sound, complete and nonredundant.

(1) If $X \subseteq Y \subseteq T$, then $Y \rightarrow X$.
(2) If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$.
(3) If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.

Figure 1. FD Inference rules for a relation.

The set of all FDs derivable from D using a system of inference rules C is called the *closure* of D under C and is denoted by $D^+(C)$, or simply D^+ when there is no ambiguity. A covering of D under C is a set E such that $E^+(C) = D^+(C)$. E is a *minimal cover* if no proper subset of it is a covering.

RELATIONAL VIEWS

The main purpose of database management systems is to provide convenient access to shared data for a community of users having assorted requirements and database experience. This is accomplished in part by providing each user with a 'view' of only the relevant portions of the database. This section refines our work on relational views which is reported elsewhere in detail[4,5].

A relational view $Vj\langle Sj, Ej\rangle$ defined over a relational database $R = \{Ri\langle Ti, Di\rangle \mid 1 \leq i \leq n\}$ is a 'virtual' relation such that $Vj(Sj)$ denotes the extension of Vj and Ej denotes the set of FDs in Vj. The definition of Vj constitutes a mapping from the extension of R to the extension of Vj. We consider only those views which are derived from R using projection and equi-join operators.

Note that the inference rules described in Fig. 1 are applicable to a single relation. Specifically, inference rule (1) describes only those projections realizable in a single relation. Since we are now dealing with multiple relations, we need a rule which describes those projections which are realizable through the lossless join of multiple relations.[6] Rule (4) shown in Fig. 2 has been added for this purpose.

(1) If $X \subseteq Y \subseteq Ti$ for some $Ri(Ti)$, then $Y \rightarrow X$.
(2) If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$.
(3) If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.
(4) If $Z \rightarrow X$ and $Z \rightarrow Y$ and $XYZ \not\subseteq Ti$ for any $Ri(Ti)$, then $XY \rightarrow A$ for $A \in XY$.

Figure 2. FD Inference rules for single and multiple relations.

Let SOURCE(Vj) denote the set of relations from which a view Vj is derived, and let $N = \{i \mid Ri \in \text{SOURCE}(Vj)\}$. We say a view is 'good' if the FDs embodied by this view can be obtained from its corresponding source relations by using the inference rules shown in Fig. 2. Formally,

> A view $Vj\langle Sj, Ej\rangle$ is *FD-consistent* (or consistent, for short) with SOURCE(Vj) if $Ej^+ \subseteq (\bigcup_{i \in N} Di)^+$.

To understand the type of ambiguity that arises when a view is not FD-consistent with its source relations, consider view V1 obtained from relations R1 and R2 (Fig. 3). V1 is not FD-consistent since EMP,PROJ $\rightarrow$ ACCT NO. $\in E1^+$ but EMP,PROJ $\rightarrow$ ACCT NO. $\notin (D1 \cup D2)^+$. Note that the derivation from R1 and R2 of V1 insists that every EMP in a DEPT must work on every

V1(EMP	DEPT	PROJ	ACCT NO.)
e1	*d1*	*p1*	*a1*
e1	*d1*	*p4*	*a2*
e2	*d2*	*p2*	*a1*

E1 = {EMP $\rightarrow$ DEPT; DEPT, PROJ $\rightarrow$ ACCT NO.}

V1 = (R1*R2)[EMP, DEPT, PROJ, ACCT NO.]

R1(EMP	DEPT)
e1	*d1*
e2	*d2*

R2(DEPT	PROJ	ACCT NO.)
d1	*p1*	*a1*
d1	*p4*	*a2*
d2	*p2*	*a1*

D1 = {EMP $\rightarrow$ DEPT} D2 = {DEPT, PROJ $\rightarrow$ ACCT NO.}

Figure 3. Non-FD-consistent view.

PROJ being worked on by that DEPT and furthermore each EMP uses the ACCT NO. for each of the PROJs of the DEPT. However, the FDs associated with V1 are not sufficient to impose the constraint that each EMP must work on all PROJs being worked on by the EMP's DEPT. This constraint can only be achieved by including multivalued dependency (MVD) information in the descriptions of both the source relations and the view.[7,8] However, we are not convinced that the semantics associated with an MVD can be easily understood by the users of a view. Thus, we do not allow views of this type and do not include MVD information in our analysis.

FD-consistency is a strong enough condition to allow retrievals from a view because the user's interpretation of a view is 'consistent' with the interpretation of the underlying source relations. However, it is not a sufficient condition for update purposes. Note that the view V3 derived from relation R3 (Fig. 4) is an FD-consistent

V3(AGE	NO. KIDS)
a1	*n1*
a2	*n2*

E3 = ϕ

V3 = R3[AGE, NO. KIDS]

R3(EMP	AGE	NO. KIDS	SPOUSE)
e1	*a1*	*n1*	*s1*
e2	*a2*	*n2*	*s2*

D3 = {EMP $\rightarrow$ AGE | NO. KIDS | SPOUSE}

Figure 4. Nondeletable FD-consistent view.

view. However, what are the semantics of a deletion from V3? One way to ensure that the deletion is well-defined is to insist that a key of the view is a key in at least one of its source relations. Formally,

> A view $Vj\langle Sj, Ej\rangle$ is *deletion supportable* if it is FD-consistent and a key of Vj is a key of some relation $Ri \in \text{SOURCE}(Vj)$.

We claim that an insertion through a view should be

allowed only if it does not force us to assign a null value to an attribute which appears on the left-hand side of some nontrivial FD. As a result, view V4 of relations R4 and R5 (Fig. 5) is not allowed since an insertion through

V4(EMP MGR)
*e*1 *m*1
*e*2 *m*2

E4 = {EMP → MGR}

V4 = (R4*R5)[EMP, MGR]

R4(EMP DEPT)
*e*1 *d*1
*e*2 *d*2

R5 (DEPT MGR)
*d*1 *m*1
*d*2 *m*2

D4 = {EMP → DEPT} D5 = {DEPT → MGR}

Figure 5. A non-insertion supportable view.

it will result in the insertion of a tuple in relation R5 containing a (unique) *NULL* value for the key attribute DEPT. This condition also ensures that view V5 of relation R6 (Fig. 6) cannot be used for insertion purposes.

V5(EMP MGR)
*e*1 *m*1
*e*2 *m*2

E5 = {EMP → MGR}

V5 = R6[EMP, MGR]

R6(EMP DEPT MGR)
*e*1 *d*1 *m*1
*e*2 *d*2 *m*2

D6 = {EMP → DEPT|MGR; DEPT → MGR}

Figure 6. A non-insertion supportable view.

Note that although DEPT is not a key attribute now, at some point we might wish to decompose R6 into R4 and R5 to accommodate additional views (e.g. view (DEPT, MGR)) and then DEPT will be a key attribute.

Formally,

> A view V*j*⟨S*j*, E*j*⟩ is *insertion supportable* if it is FD-consistent and for each full FD X → Y ∈ $(\bigcup_{i\in N} D_i)^+$, Y ⊆ S*j* implies that X ⊆ S*j*.

It is not a difficult exercise to show that an insertion supportable view is also deletion supportable. Thus,

> The class of *updatable views* is precisely the class of insertion supportable views.

RELATIONAL DATABASE TRANSLATIONS

Relational database translation deals with the reorganization of a relational database for use on other systems, to improve system performance and to achieve a more flexible organization of data. A fundamental problem in database translation is to determine what information from the original database is preserved by the reorganized version of the database.

An extensively studied class of transformations involves decomposing a relation into a set of relations using only the projection operator.[3,9,10,11] In this section, we will develop meaningful criteria for describing the properties of a wider class of transformations.

Let T be a transformation which when applied to a set of relations R = {R*i*⟨T*i*, D*i*⟩ | 1 ≤ *i* ≤ *n*} yields a set of relations Q = {Q*j*⟨W*j*, F*j*⟩ | 1 ≤ *i* ≤ *m*}. *We assume that* (*i*) ⋃T*i* = ⋃W*j*, (ii) each D*i* and F*j* contains information about FDs, and (iii) each relation Q*j* is derived from R by using projection and/or equi-join operators.

The loss (if any) of information from R to Q can be of two types. First, it is possible that Q*j* = R*i**R*k* but that the equi-join operator failed for some specific tuple of R*i* and as a result some specific query resolvable by R*i* cannot be resolved by Q*j*. This is illustrated in Fig. 7. If Q = Q3, then after the transformation the query 'retrieve DEPT for EMP = *e*1' cannot be resolved. We call this phenomenon to be the loss of information in the extensional sense. One way to avoid it is to make a strong assumption that if an attribute set X appears in relations R*i* and R*k* and the derivation of Q*j* necessitates computation of R*i**R*k*, then it must be the case that R*i*[X] = R*k*[X]. We are not particularly fond of this assumption since it prohibits some very simple but appealing transformations (Fig. 7 where Q = {Q1, Q2, Q3}). Instead, we make the assumption that if an attribute A appears in R*i*, then there exists a $Q_j \in Q$ such that R*i*[A] ⊆ Q*j*[A]. This weaker assumption is sufficient to guarantee that there will not be a loss of information in the extensional sense. From this point on, it is assumed that a transformation does not lose information in the extensional sense.

To study the loss of information in the intensional sense, we define six types of transformations. A transformation applied to a database R yielding a database Q is (i) 'update equivalent' if the class of updates which can be performed in R is precisely the class of updates which can be performed in Q, (ii) 'update preserving' if all updates of R can be performed in Q, (iii) 'weakly update preserving' if all allowable updates of R can be performed in Q but some nonallowable updates of R cannot be recognized as such in Q and thus are allowable in Q, (iv) 'retrieval equivalent' if the class of queries supported by R is precisely the class of queries supported by Q (v)

R1(EMP DEPT)
*e*1 *d*1
*e*2 *d*2

R2(DEPT MGR)
*d*2 *m*2

Q1(EMP DEPT)
*e*1 *d*1
*e*2 *d*2

Q2(DEPT MGR)
*d*2 *m*2

Q3(EMP DEPT MGR)
*e*2 *d*2 *m*2

Figure 7. Loss (Q = {Q3}) and no loss (Q = {Q1,Q2,Q3}) of information in the extensional sense.

'retrieval preserving' if the queries of R can be resolved by Q, and (vi) 'partially retrieval preserving' if some queries of R can be resolved by Q.

Formally, let T be a transformation which when applied to a relational database $R = \{Ri\langle Ti, Di\rangle \mid 1 \leq i \leq n\}$ yields a relational database $Q = \{Qj\langle Wj, Fj\rangle \mid 1 \leq j \leq m\}$. Let $SOURCE(Qj) = \{Ri \mid Ri$ is used in the derivation of $Qj\}$ and let $SOURCE^*(Ri) = \{Qj \mid Ri \in SOURCE(Qj)\}$.

Definition. Transformation T is

(1) Update equivalent (UE) if each Ri is an updatable view of $SOURCE^*(Ri)$ and each Qj is an updatable view of $SOURCE(Qj)$.
(2) Update preserving (UP) if each Ri is an updatable view of $SOURCE^*(Ri)$.
(3) Weakly Update preserving (WUP) if for each $Ri\langle Ti, Di\rangle$, an updatable view $Vj\langle Sj, Ej\rangle$ is supported by $SOURCE^*(Ri)$, where $Ti = Sj$ and $Ej^+ \subseteq Di^+$.
(4) Retrieval equivalent (RE) if each Qj is a consistent view of $SOURCE(Qj)$ and for each $Ri\langle Ti, Di\rangle$, a consistent view $Vj\langle Sj, Ej\rangle$ is supported by $SOURCE^*(Ri)$, where $Ti = Sj$ and $Ej^+ \subseteq Di^+$.
(5) Retrieval preserving (RP) if for each $Ri\langle Ti, Di\rangle$, a consistent view $Vj\langle Sj, Ej\rangle$ is supported by $SOURCE^*(Ri)$, where $Ti = Sj$ and $Ej^+ \subseteq Di^+$.
(6) Partially retrieval preserving (PRP) if for some attribute set X, at least one $Qj[X]$ is a consistent view of $SOURCE(Qj)$.

In the context of Fig. 8, let T1 be a transformation which yields $Q = \{2, 3\}$ when applied to $R = \{1\}$. Note that T is UE since the class of updates which can be performed in relation (1) is exactly the class of updates allowable by relations (2) and (3).

If we take $R = \{4\}$ and $Q = \{5, 6\}$, then the transformation T2 is not UE since Q allows an update of relation (6) of the form 'insert $\langle d1, m1\rangle$' but this update cannot be performed in relation (4). Note that T2 is UP since all updates of relation (4) can be simulated in Q.

Now, let $R = \{7\}$ and $Q = \{8, 9\}$. This transformation T3 is not UP since a nonallowable update which violates CITY, ADDR → POSTAL of relation (7) will be allowable in Q. This is due to the fact that it is impossible to derive this FD in Q. Transformation T3, however, is WUP since Q does support an updatable view (CITY, ADDR, POSTAL, NAME) with the set of FDs {ADDR, POSTAL → NAME; POSTAL → CITY}.

If $R = \{5,6\}$ and $Q = \{4\}$, then T4 is not WUP since, corresponding to relation (6), no updatable view is supported by Q. Since relation (4) is a consistent view of relations (5) and (6) and they in turn are consistent views of relation (4), transformation T4 is RE, and, therefore, the classes of queries resolvable by R and Q are identical.

Consider transformation T5 applied to $R = \{5,10\}$ yielding $Q = \{11\}$. If one looks at the FDs associated with relation (11), the semantics of this relation are: an employee works in a single department, a department working on a project has a unique account number and each employee working on a project has a unique account number. Thus, relation (11) indicates that a query of the type 'retrieve EMPs working on PROJ p1' is resolvable by it. However, no query of this type is resolvable by R, and, therefore, T5 is not RE. Since every query of R is resolvable by Q, T5 is RP.

Finally, let us consider relation (11) again with the above semantics and a transformation T6 which yields $Q = \{5, 10\}$ from $R = \{11\}$. Obviously, T6 is not RP since, as before, 'retrieve EMPs working on PROJ p1' cannot be resolved by Q although it is resolvable by R. Clearly, T6 is PRP since some queries of R can be resolved in Q. It should be pointed out that the property of PRP is very weak since every transformation satisfies it.

From these definitions, it is clear that UE implies UP which in turn implies WUP. Also, RE implies RP which in turn implies PRP. To show that the implications depicted in Fig. 9 hold, we need to prove the following.

Lemma. If a transformation T is WUP, then it is also RE.

Proof. Since each updatable view is also consistent, the second condition of RE is satisfied. It remains to be shown that if T is WUP, then each Qj is a consistent view of $SOURCE(Qj)$.

Consider a $Qj \in Q$ and let $SOURCE(Qj) = \{R1, R2, \ldots, Rp\}$. Then, for $1 \leq i \leq p$, $SOURCE^*(Ri)$ contains Qj. Consider a $Rk \in SOURCE(Qj)$ and let $SOURCE^*(Rk) = \{Q1, Q2, \ldots, Qj, \ldots, Qr\}$. Since T is WUP, for Rk a relation $Vk\langle Sk, Ek\rangle$ is updatable view of $SOURCE^*(Rk)$, and, therefore, all key attributes of

1(EMP AGE PHONE)
FD1 = {EMP → AGE|PHONE}

2(EMP AGE)
FD2 = {EMP → AGE}

3(EMP PHONE)
FD3 = {EMP → PHONE}

4(EMP DEPT MGR)
FD4 = {EMP → DEPT|MGR; DEPT → MGR}

5(EMP DEPT)
FD5 = {EMP → DEPT}

6(DEPT MGR)
FD6 = {DEPT → MGR}

7(CITY ADDR POSTAL NAME)
FD7 = {CITY, ADDR → POSTAL|NAME; POSTAL → CITY}

8(POSTAL CITY)
FD8 = {POSTAL → CITY}

9(POSTAL ADDR NAME)
FD9 = {POSTAL, ADDR → NAME}

10(DEPT PROJ ACCT)
FD10 = {DEPT, PROJ → ACCT}

11(EMP PROJ DEPT ACCT)
FD11 = {EMP → DEPT; DEPT, PROJ → ACCT}

Figure 8. Example relations.

$$UE \precneqq UP \precneqq WUP \precneqq RE \precneqq RP \precneqq PRP$$

Figure 9. Relationships among properties of transformations.

Qj are in Vk. Since Rk and Vk are defined over the same set of attributes, all key attributes of Qj are in Rk. Thus, for $1 \leq i \leq p$, Ri contains key attributes of Qj, making Qj consistent with SOURCE(Qj).

A well studied class of transformations involves normalization of a relation[12]. We observe that our axioms used in conjunction with Bernstein's algorithm will always produce an UP Third Normal Form (3NF) representation.[13] However, it is known that a transformation to Boyce-Codd Normal Form is,[14] in general, WUP.[9]

RELATIONAL DATABASE DESIGN

Synthesis and decomposition are two major approaches for relational database design.[12,13] This section briefly discusses these approaches and then proposes an algorithm for database design which combines the best features of both of them while staying away from their shortcomings.

The synthesis approach assumes that the designer is provided with a set G of FDs. The algorithm described by Bernstein[13] essentially finds a nonredundant cover H of G using Armstrong's axioms[2] and then constructs one relation for each subset of H which contains all the FDs with identical left hand sides. With Kambayashi's modification,[15] Bernstein's algorithm yields a minimum number of relations which must be used to 'represent' G.

Let us look at this approach from a slightly different point of view. Assume that G consists of FDs from a set R of relations and that a minimum set Q of relations, embodying the information contained in R, needs to be generated. Given this scenario, there are two shortcomings of the way the synthesis approach has been used. First, it is assumed that Armstrong's axioms[2] yield semantically equivalent derivations of an FD. Given that only FD information is known, this assumption is rarely true. The net result is to yield a set Q which is only partially retrieval preserving of the information in R.[9] For example, if R consists of the single relation 11 described in Fig. 8, the synthesis approach would produce relations (5) and (10), but these relations provide only partial retrieval support of relation (11). Admittedly, determination of semantic equivalence of various derivations of an FD is a difficult problem. However, if our axioms (Fig. 2) are used, it is possible to get some improvement. This is due to the fact that: (1) at least in the context of a single relation, it is probably true that these axioms yield semantically equivalent derivations of FDs, and (2) no derivation of an FD involves a connection trap or lossy join and, hence, there is a better likelihood that the derivations will be semantically equivalent. Even with our axioms, schema Q is, in general, only retrieval equivalent to schema R. This is the other problem with the way the synthesis approach is generally used—no attempt is made to preserve the updates of R in Q.

The decomposition approach, on the other hand, attempts to decompose a relation into several relations to gain more flexibility in terms of updates. Unless one is careful with this approach, it is possible to generate a schema which, while allowing new updates, cannot determine the validity of some old updates. For example, consider relation R(A B C D) with the set of FDs{AB→ C; C→ D; A→ D}. The decomposition approach could produce relations R1(A B C) and R2(A D) which are only weakly update preserving. Also, no consideration is given to the issue of redundancy by this approach.

The database design algorithm we are proposing is as follows:

Step [0] The design algorithm is provided with a collection of relational views $V = \{Vi\langle Ti, Ei\rangle | 1 \leq i \leq m\}$.

Step [1] The consistency of interpretation across all views is checked. View Vi is consistent with view Vj if the following condition holds:
$X,Y \subseteq Ti \,\&\, X \rightarrow Y \in (\bigcup Ej)^+ \Rightarrow X \rightarrow Y \in Ei^+$
If the property $X \rightarrow Y \in Ei^+$ is not desired, then these attributes must be renamed in view Vi so that consistency of interpretation across all views holds.

Step [2] The conflict free property of multiple FD derivations is checked. Two derivations of the FD X→ Y are said to be conflict free if whenever these derivations yield tuples of the form $\langle x, y_1\rangle$ and $\langle x, y_2\rangle$ where x ≠ *NULL*, then either $y_1 = y_2$ or $y_1 = NULL$ or $y_2 = NULL$. Note that this property is slightly weaker than saying that the two derivations are semantically equivalent. For each occurrence of a multiple FD derivation where this property is not desired, some of these attributes must be renamed. If the conflict free property is desired, then a trigger mechanism must be associated with the source relations embodying these derivations which checks that updates do not violate the conflict free constraint. For example, if the following views were provided it might be desirable to maintain the conflict free property for both the COURSE→ PROF→ DEPT and COURSE→ TA→ DEPT derivations of the FD COURSE→ DEPT:

V1(COURSE PROF TA)
V2(PROF DEPT)
V3(TA DEPT)

Step [3] Maximize data sharing

(i) Through view redefinition
Consider the following views:

V1(EMP DEPT)
V2(DEPT MGR)
V3(EMP MGR)

At this point in the algorithm, it can be assumed that the derivations of the FD EMP→ MGR embodied by views V1, V2 and V3 are to be maintained in a conflict free manner. If these derivations are represented separately, then the appropriate trigger mechanism (described in Step [2]) must be utilized to maintain

this property. On the other hand, if the user is willing to redefine view V3 as (EMP, DEPT, MGR), where DEPT→MGR, then the following source relations will be obtained in part (ii) of this step:

R1(EMP, DEPT)
R2(DEPT, MGR)

Each view is now an updatable view, and redundant representation of FDs has been minimized.

(ii) Generate a minimal 3NF schema.

Using the synthesis approach and our axioms, an update preserving 3NF representation of the renamed views is constructed subject to any redundancy imposed in part (i).

The relational schema generated by the above algorithm has several desirable properties. They include:

* Each view is retrieval and update supportable.
* A consistent interpretation across views is supportable.
* Only well-defined tuples (non-*Null* values in key attributes) in all views are allowed.
* Incorrect view insertions are reversible.
* A conflict free representation of FDs is supportable.
* Redundant representation of FDs is minimized.

CONCLUSION

Functional dependencies first emerged in the context of normalization theory. Since then they have been applied to many areas of database study. What we have done in this paper is to show that functional dependencies provide a unified framework for the analysis of various database problems.

Several concepts have been formalized in this paper. For view design, 'consistency' and 'updatability' allow us to identify retrieval and update supportable views, respectively. In the area of data translation, six levels of information preserving transformations have been identified. A database design algorithm has been proposed which produces a relational schema based on the concepts of 'consistent views', 'updatable views', 'conflict free FD representation', and 'nonredundant FD representation'.

Acknowledgment

The authors are grateful to Richard T. Bagley who implemented the relational database design algorithm and with whom we have had many useful discussions. We would also like to thank the users of the algorithm for forcing us to clarify our statement of the algorithm.

REFERENCES

1. E. F. Codd, *Relational Completeness of Data Base Sublanguages,* Data Base Systems, Courant Computer Symposia Series, Vol. 6, pp. 65–98. Prentice-Hall, Englewood Cliffs, New Jersey (1972).
2. W. W. Armstrong, Dependency structures of data base relationships. *Proceedings of the International Federation for Information Processing* (1974) 580–583.
3. A. K. Arora and C. R. Carlson, The information preserving properties of relational database transformations. *Proceeding of 1978 VLDB, Berlin, W. Germany,* 352–359 (1978).
4. A. K. Arora and C. R. Carlson, *On the Updatability of Relational Views,* Technical Memorandum 80–5424–2, Bell Laboratories (1980).
5. C. R. Carlson and A. K. Arora, Updatability of relational views based on functional dependencies, *Proc. COMPSAC 1979,* Chicago, Illinois, 415–420 (1979).
6. A. V. Aho, C. Beeri and J. D. Ullman The theory of joins in relational data bases. *Proceedings of the 18th Annual ACM Symposium on Foundations of Computer Science,* 107–113, November 1977.
7. C. Beeri, R. Fagin and J. H. Howard, A complete axiomatization for functional and multivalued dependencies in database relations. *Proceedings of the ACM-Special Interest Group on Management of Data SIGMOD,* Toronto, Canada, 47–61 (1977).
8. C. A. Zaniolo, Analysis and design of relational schemata for database systems, PhD Dissertation, Computer Science Department, UCLA (1976).
9. A. K. Arora and C. R. Carlson, *A Formal Characterization of the Information Preserving Properties of Relational Database Transformation,* Technical Memorandum 79–5424–3, Bell Laboratories (1979).
10. C. Beeri, P. Bernstein and N. Goodman, A sophisticates introduction to database normalization theory, *Proceedings of 1978 VLDB, Berlin, W. Germany,* 113–123 (1978).
11. C. Beeri and P. A. Bernstein, Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems* **4** (No. 1), 30–59 (1979).
12. E. F. Codd, *Further Normalization of the Data Base Relational Model,* Data Base Systems, Courant Computer Symposia Series, Vol. 6, pp. 33–64. Prentice-Hall, Englewood Cliffs, New Jersey (1972).
13. P. A. Bernstein, Synthesizing Third Normal Form Relations from Functional Dependencies, *ACM Transactions on Database Systems* **1** (No. 4), 277–298 (1976).
14. E. F. Codd, Recent investigations into relational data base systems. *Proceedings of the International Federation for Information Processing 1974,* 1017–1021 (1974).
15. Y. Kambayashi, K. Tanaka and S. Yajima, Relational database design, *Proceedings of the 13th IBM Computer Science Symposium—Software Engineering Series No. 1,* Working Conference on Database Engineering (November 1979).

Received November 1980