Proc. 1979 ACM-SIGMOD (ed. P. A. Bernstein), 153-160.

Normal forms and relational database operators

by

Ronald Fagin
IBM Research Laboratory
San Jose, California 95193

ABSTRACT: We discuss the relationship between normal forms in a relational database and an allowed set of relational operators. We define "projection-join normal form" (PJ/NF), which is the ultimate normal form when only projection and join are allowed. Aho, Beeri and Ullman made the counterintuitive discovery that there is a relation schema with a valid decomposition into three of its projections without the decomposition being equivalent to a cascade of decompositions, each into two projections. Because of this possibility, there exist bizarre relation schemata that are in fourth normal form but not in PJ/NF. We also discuss issues associated with allowing the union operator.

1. Introduction

The process of normalization consists of converting a relation schema (or set of relation schemata) into another form that stores the same data but in a different format (see Beeri, Bernstein, and Goodman [BBG] for a detailed discussion). The question of which operators are allowed in the conversion process is, of course, critical. Surely, the two most important relational operators are projection and (natural) join (see Codd [Co2]). In this paper, we introduce "projection-join normal form" (PJ/NF), in which the only legal operators are projection and join. PJ/NF is almost by definition the ultimate normal form when only projection and join are allowed. The reason why PJ/NF is stronger than fourth normal form [Fa1] is because of a surprising phenomenon that was first discovered by Aho, Beeri, and Ullman [ABU], and that was also investigated by Nicolas [Ni]. We also discuss in this paper the issue of allowing the union operator.

We note that PJ/NF could logically be called "fifth normal form", since it is stronger than fourth normal form. However, we instead choose to call it projection-join normal form for several reasons. First, we wish to emphasize its finality with respect to the projection and join operators. Second, we feel that from now on, it will be desirable to explicitly point out the relationship between normal forms and the allowed relational operators. For example, as we show in Section 4, by allowing a new relational operator (the union operator), one might normalize further. The first 3 sections of this paper provide a clean completion to the normalization question, with respect to the usual operators (projection and join). On the other hand, the final section merely "opens the door a crack" with respect to new operators; a great deal of further research remains to be done in this latter area.

If S[UV] and T[UW] are relations (where the attributes, or column names, in common are the attributes in U), then the join S*T of S and T is the relation consisting of the tuples (u,v,w), where (u,v) is in S and (u,w) is in T. The concept of multivalued dependency (see Fagin [Fa1]) is intimately related to that of join. Specifically, if U and V are subsets of attributes of a relation R, and if W is the set of attributes of R not in U or V, then the *multivalued dependency* $U \twoheadrightarrow V$ holds in R if and only if R is the join of its projections R[UV] and R[UW]. (This is Theorem 1 of [Fa1].)

NOTE: When we write UV, where U and V are sets of attributes, we mean the set UUV. When we write AB, where A and B are attributes, we mean the set {A,B}; similarly for ABCD and so on.

It is convenient for us to use a generalized definition of join, due to Aho, Beeri and Ullman [ABU], that gives the join of an arbitrary collection of relations. If $R_1,...,R_m$ are relations, then the *join* of the set $\{R_1,...,R_m\}$ is the set of all tuples $(a_1,...,a_n)$, where $A_1,...,A_n$ are the attributes appearing in at least one of $R_1,...,R_m$, and where for each i, the projection of the tuple $(a_1,...,a_n)$ onto the attributes of R_i is a tuple in the relation R_i. It is straightforward to verify that the old and new definitions of join agree when there are only two relations to be joined. Furthermore, this generalized join can be "built up" out of binary joins. For example, the join $*\{R_1,R_2,R_3,R_4\}$ is equal to $(((R_1*R_2)*R_3)*R_4)$, that is, the result of joining R_1 and R_2, joining the result with R_3, and then joining this result with R_4. By the associativity and commutativity of the binary join, we can take the binary joins in any order we wish, and parenthesize however we wish, and still get the same answer. For example, $*\{R_1,R_2,R_3,R_4\}$ also equals $(R_2*R_3)*(R_4*R_1)$.

Let $X_1,...,X_m$ each be subsets (not necessarily disjoint) of the attributes of relation R, where each attribute of R is contained in at least one of $X_1,...,X_m$. Following Rissanen [Ri2], we say that R obeys the *join dependency* $*\{X_1,...,X_m\}$ if R is the join of its projections $R[X_1],...,R[X_m]$.

As noted in [ABU], the join of the projections $R[X_1],...,R[X_m]$ of R equals

{t: there are tuples $w_1,...,w_m$ of R such that $w_i[X_i] = t[X_i]$ for each i $(1 \le i \le m)$}

Thus, the join dependency $*\{X_1,...,X_m\}$ holds for the relation R if and only if R contains each tuple t for which there are tuples $w_1,...,w_m$ of R (not necessarily distinct) such that $w_i[X_i] = t[X_i]$ for each i $(1 \le i \le m)$.

If R and U,V,W are as in the definition above of multivalued dependency, then the multivalued dependency $U \twoheadrightarrow V$ holds in R if and only if the join dependency $*\{UV,UW\}$ holds in R. Thus, it is possible to represent each multivalued dependency by a join dependency. We note that Nicolas [Ni] defined the *mutual dependency*, which is the special case of the join dependency where there are exactly three relations to be joined. Also, Dayal and Bernstein [DB] explore properties of join dependencies, which they call *interdependencies*.

Aho, Beeri, and Ullman [ABU] present a surprising example to show that a relation can be the join of three of its projections, without this join being the result of cascading 2-way projections. Thus, let R, with attributes ABCDEF, obey the functional dependencies $A \rightarrow B$ and $F \rightarrow E$. They show that the join dependency $*\{ABDE,ACDF,BCEF\}$ necessarily holds. That is, if we abbreviate the projections R[ABDE], R[ACDF], and R[BCEF] by R_1, R_2, and R_3 in any order, then the relation R equals the join of its three projections R_1, R_2, and R_3. However, it is not hard to show that this decomposition is not necessarily the result of cascading 2-way decompositions. That is, there are not necessarily projections S_1 and S_2 of R such that

(1) R is the join of S_1 and S_2.

(2) S_2 is the join of two of its projections S_{21} and S_{22}.

(3) R_1, R_2, and R_3 equal S_1, S_{21}, and S_{22} respectively.

Thus, this 3-way decomposition is not built out of 2-way decompositions. Let us look a little more carefully about what goes wrong in this example. Since $R = *\{R_1,R_2,R_3\}$, we know that R is the join of R_1 with R_2*R_3. In this case, R_2*R_3 has all of the attributes of R, and contains all of the tuples of R, *and possibly more*. The effect of joining R_1 with R_2*R_3 is only to *remove* enough tuples from R_2*R_3 to make it exactly equal to R. As a concrete example of the join removing tuples, assume that the relation R contains, possibly among others, the tuples (a,b,c,d,e,f) and (a′,b′,c,d′,e,f), where $a \neq a'$, $b \neq b'$, and $d \neq d'$, and that the relation R obeys the functional dependencies $A \rightarrow B$ and $F \rightarrow E$. In particular, the tuple (a,b′,c,d,e,f) is not in R, or else the functional dependency $A \rightarrow B$ would be violated. Let R_1, R_2, and R_3 be the projections R[ABDE], R[ACDF], and R[BCEF] respectively. As noted earlier, the relation R equals the join of R_1 with R_2*R_3. It is easy to verify that R_2*R_3 contains, among others, the tuple (a,b′,c,d,e,f), which, as we saw, is not in R. When we join the relation R_2*R_3 with R_1, this tuple disappears.

Fourth normal form is defined in terms of functional and multivalued dependencies alone. Since multivalued dependencies correspond to 2-way decompositions, and since, as we saw, there are decompositions that are not the result of cascading 2-way decompositions alone, an "ultimate" PJ/NF must also consider the more general join dependencies.

2. Projection-join normal form

In this section, the only relational operators that we consider are projection and join. Thus, in this section, the ground rules for normalization are:

(1) Each relation that we consider is already in Codd's first normal form (1NF); that is, each entry is atomic.

(2) When a relation is replaced by a set of relations, each of the new relations is a projection of the original relation.

(3) The original relation must be the join of the set of new relations.

Second normal form, third normal form, Boyce-Codd normal form (BCNF), and fourth normal form (4NF) all fulfill these ground rules, as will the new projection-join normal form (PJ/NF).

From now on, we will abbreviate functional dependency, multivalued dependency, and join dependency by FD, MVD, and JD respectively. Let X be a set of attributes, let σ be a dependency (FD, MVD, or JD), and let Σ be either a dependency or a set of dependencies. When we say that Σ *logically implies* σ *(in the context of X)*, or that σ *is a logical consequence of* Σ *(in the context of X)*, we mean that whenever Σ holds for a relation with attributes X, then so does σ. That is, there is no "counterexample relation" R with attributes X such that every dependency in Σ holds in R, but such that σ does not hold in R. We write $\Sigma \models_X \sigma$, or, if the context X is understood, simply $\Sigma \models \sigma$. As a simple example,

$\{A \rightarrow B, B \rightarrow C\} \models A \rightarrow C$.

We remark that the context X is not important for FDs alone. That is, it follows from the completeness of Armstrong's axioms [Ar, Fa2] that if Σ is an FD or a set of FDs, and if σ is a single FD, then $\Sigma \models_X \sigma$ if and only if $\Sigma \models_Y \sigma$, as long as X and Y each contain all of the attributes appearing in Σ and/or σ. However, as noted in [Fa1], the context *is* important when MVDs (or JDs) are involved. For example, if X = ABC, then $A \twoheadrightarrow B \models_X A \twoheadrightarrow C$ holds, whereas it does not hold if X = ABCD. When Ω is a set of dependencies, then by $\Sigma \models \Omega$, we mean that $\Sigma \models \sigma$ for each σ in Ω. Later on, we will make use of the easily verified fact that if ρ, σ, and τ are dependencies (or sets of dependencies), if $\rho \models \sigma$ and if $\sigma \models \tau$, then $\rho \models \tau$ (transitivity).

We say that a dependency is *trivial* (in the context of the set X of attributes) if it holds in every relation that has the set X as its attributes. That is, a dependency σ is trivial (in the context of X) if $\varnothing \models_X \sigma$, where $\varnothing$ is the empty set. It is easy to verify that the only trivial FDs $U \rightarrow V$ are those where V is a subset of U. Further, the only trivial MVDs $U \twoheadrightarrow V$ are those where either V is a subset of U, or where the union of U and V equals X. Finally, it follows easily by using the techniques in [ABU] that the only trivial JDs $*\{X_1,...,X_m\}$ are those where one of the X_i's equals X.

Following Cadiou [Ca], we define a *relation schema* to be a set of attributes, along with a set of dependencies (in our case, FDs, MVDs, and JDs). Cadiou actually includes more, that does not concern us here, in his definition of relation schema. So that we can always speak of dependencies *in* the schema, rather than dependencies that are *logical consequences* of those in the schema, it is convenient to assume that the set of

dependencies in the schema is closed under logical consequence. That is, if Σ is the set of dependencies in the schema, and if σ is a dependency such that $\Sigma \models \sigma$, then σ is also a dependency in the schema.

Let R^* be a relation schema with attributes X. If K is a subset of X, then we say that K is a *key* (of the relation schema) if the FD K→X is in the schema, and if there is no proper subset L of K such that the FD L→X is also in the schema. We call such a functional dependency K→X a *key dependency* of R^*. We note that each theorem in this paper that mentions a key dependency would also be true if we were to modify our definition of key dependency to allow an arbitrary functional dependency L→X, where the left-hand side L is either a key or a superset of a key (a "superkey" [Be]).

To emphasize the analogy between BCNF, 4NF, and PJ/NF, it will be helpful to define each of them in several distinct (but equivalent) ways.

Definition 1(a) [Co3]. A 1NF relation schema R^* with attributes X is in BCNF if, for each nontrivial FD U→V in R^*, the FD U→X is in R^*.

Definition 1(b). A 1NF relation schema R^* with attributes X is in BCNF if $\Delta \models \sigma$ for each FD σ in R^*, where Δ is the set of key dependencies of R^*. Thus, every FD is the result of keys.

Definition 1(c). A 1NF relation schema R^* with attributes X is in BCNF if, for each FD σ in R^*, there is a key dependency K→X of R^* such that K→X $\models \sigma$. Thus, every FD is the result of a key.

Theorem 1. Definitions 1(a), 1(b), and 1(c) are equivalent.

Proof: Assume first that a relation schema R^* is in BCNF according to Definition 1(a). We will show that it is in BCNF according to Definition 1(c). Let U→V be an FD that holds in R^*. By Definition 1(a), we know that either (i) U→V is a trivial FD, or else (ii) the FD U→X holds. In case (i), let K be an arbitrary key of the schema. Then K→X $\models$ U→V, since the FD U→V is trivial. This completes case (i). In case (ii), we know that U contains a key K. Since K→X $\models$ U→X and U→X $\models$ U→V, it follows by transitivity that K→X $\models$ U→V, as desired. Thus, in either case, R^* is in BCNF according to Definition 1(c).

The fact that a relation schema in BCNF according to Definition 1(c) is also in BCNF according to Definition 1(b) is trivial.

Now assume that R^* is in BCNF according to Definition 1(b). The proof is complete if we show that R^* is in BCNF according to Definition 1(a). Let U→V be a nontrivial FD in R^*. We need to show that the FD U→X is in R^*. From Definition 1(b), we know that $\Delta \models$ U→V. Write Δ as $\{K_1 \rightarrow X, \ldots, K_r \rightarrow X\}$. We now show that K_i is a subset of U for some i ($1 \le i \le r$). Assume not; we will derive a contradiction. Define a two-tuple relation Q with attributes X, such that one tuple has only zeros as entries, and the other tuple has zeros in the U columns and ones everywhere else. (We do *not* make any assumption that Q is a valid instance of the relation schema R^*.) For each i ($1 \le i \le r$), the two tuples of Q disagree in at least one column of K_i, since K_i is not a subset of U. Hence, the relation Q obeys each FD in Δ. However, relation Q does not obey the FD U→V, since the two tuples agree in the U columns but disagree in at least one column of V, since V is not a subset of U (or else the FD U→V would be trivial). So, relation Q is a counterexample relation that obeys Δ but disobeys U→V. This contradicts the fact that $\Delta \models$ U→V. We have shown by contradiction that K_i is a subset of U for some i. Hence, $K_i \rightarrow X \models$ U→X. Since $K_i \rightarrow X$ holds in R^*, and since $K_i \rightarrow X \models$ U→X, it follows that the FD U→X holds in the schema R^*, which was to be shown. □

We now present three definitions of 4NF.

Definition 2(a) [Fa1]. A 1NF relation schema R^* with attributes X is in 4NF if, for each nontrivial MVD U→→V in R^*, the FD U→X is in R^*.

Definition 2(b). A 1NF relation schema R^* with attributes X is in 4NF if $\Delta \models \sigma$ for each MVD σ in R^*, where Δ is the set of key dependencies of R^*. Thus, every MVD is the result of keys.

Definition 2(c). A 1NF relation schema R^* with attributes X is in 4NF if, for each MVD σ in R^*, there is a key dependency K→X of R^* such that K→X $\models \sigma$. Thus, every MVD is the result of a key.

Theorem 2. Definitions 2(a), 2(b), and 2(c) are equivalent.

Proof: The proof that a relation schema in 4NF by Definition 2(a) is also in 4NF by Definition 2(c) is almost identical to the proof that a relation schema in BCNF by Definition 1(a) is also in BCNF by Definition 1(c). We leave the trivial modifications to the reader. As before, the implication from 2(c) to 2(b) is trivial. To conclude the proof, we show that a relation schema in 4NF according to Definition 2(b) is also in 4NF according to Definition 2(a). Let U→→V be a nontrivial MVD in R^*. We must show that the FD U→X is in R^*. We can assume (by replacing V with the set difference V−U if necessary) that U and V are disjoint. Let W be the complement (in X) of the union of U and V. Since the MVD U→→V is nontrivial, we know that V and W are both nonempty. From Definition 2(b), we know that $\Delta \models$ U→→V. Write Δ as $\{K_1 \rightarrow X, \ldots, K_r \rightarrow X\}$. We now show that K_i is a subset of U for some i ($1 \le i \le r$). Assume not. As in the proof of Theorem 1, define a two-tuple relation Q with attributes X, such that one tuple has only zeros as entries, and the other tuple has zeros in the U columns and ones everywhere else. As before, relation Q obeys every FD in Δ. We can write the two tuples of Q as (u,v,w) and (u,v′,w′), where u,v, and w each contain only zeros, and v′ and w′ each contain only ones. Since the tuple (u,v′,w) does not occur in Q, it follows by Proposition 3 of [Fa1] that Q does not obey the MVD U→→V. So, relation Q is a counterexample relation that obeys Δ but disobeys U→→V. This contradicts the fact that $\Delta \models$ U→→V. We have shown by contradiction that K_i is a subset of U for some i. Hence, $K_i \rightarrow X \models$ U→X. Since $K_i \rightarrow X$ holds in R^*, and since $K_i \rightarrow X \models$ U→X, it follows that the FD U→X holds in the schema R^*, which was to be shown. □

Let U→V be an FD in a 4NF relation schema R^*. Then, of course, the MVD U→→V holds in R^* (since U→V $\models$ U→→V), and so, by Definition 2(c), there is a key K of R^* such that K→X $\models$ U→→V. Is the stronger fact true that K→X $\models$ U→V? As we now show, the answer is yes.

Theorem 3. Let U→→V be a nontrivial MVD. Denote the set of attributes by X. If K→X $\models$ U→→V, then K is a subset of U. In particular, K→X $\models$ U→V.

Proof: This follows from the proof of Theorem 2. □

Corollary 4. Assume that the FD σ holds in the 4NF relation schema R^* with attributes X. Then there is a key dependency K→X of R^* such that K→X $\models \sigma$.

Proof: Assume that the FD σ is U→V. Then the MVD U↠V is in R^*. By Definition 2(c), there is a key dependency K→X of R^* such that K→X $\models$ U↠V. By Theorem 3, K→X $\models$ U→V, which was to be shown. □

Before we define PJ/NF, we present the following membership algorithm, which tells whether or not a join dependency σ is a member of the set of logical consequences of the set Δ of key dependencies, that is, whether or not $\Delta \models \sigma$. It follows in a fairly straightforward manner from the more general membership algorithm of [ABU] that our membership algorithm succeeds (with inputs σ and Δ) if and only if $\Delta \models \sigma$.

Membership algorithm. The inputs are a join dependency $\sigma = *\{X_1,...,X_m\}$, and a set $\Delta = \{K_1 \rightarrow X, ..., K_r \rightarrow X\}$ of key dependencies. Initialize set $\mathcal{S}$ as $\{X_1,...,X_m\}$. Apply the following rule until it can be no longer applied: if $K_i \subseteq Y \cap Z$ for some i ($1 \leq i \leq r$) and for some members Y and Z of $\mathcal{S}$, then replace Y and Z in $\mathcal{S}$ by their union, that is, remove the sets Y and Z from $\mathcal{S}$ and add to $\mathcal{S}$ the single member $Y \cup Z$. (In particular, the number of members of $\mathcal{S}$ then decreases by one). If the algorithm terminates with X, the set of all attributes, as a member of $\mathcal{S}$, then we say that the algorithm *succeeds*, otherwise, we say that it *fails*.

Example: Let the set X of attributes be ABCD. Let σ be the JD *{AB,AD,BC}, and let Δ be {A→ABCD, B→ABCD}. We now show, by using our membership algorithm, that $\Delta \models \sigma$. (We will make use of this example later). Initialize $\mathcal{S}$ as {AB,AD,BC}. Since A→ABCD is in Δ, and since AB and AD are in $\mathcal{S}$, we replace AB and AD in $\mathcal{S}$ by ABD. At this stage, $\mathcal{S}$ is {ABD,BC}. Since B→ABCD is in Δ, and since ABD and BC are in $\mathcal{S}$, we replace ABD and BC by ABCD. We are left with $\mathcal{S} = \{X\}$. Thus, the membership algorithm succeeds, and so $\Delta \models \sigma$.

In this example, we can understand intuitively what is going on in the decomposition associated with the JD σ. There is one relation (with attributes AB), that gives the association between the keys A and B; a second relation (with attributes AD) that relates D to the key A; and a third relation (with attributes BC) that relates C to the key B.

Since, as noted, our membership algorithm succeeds if and only if $\Delta \models \sigma$, the following pair of definitions are equivalent.

Definition 3(a) A 1NF relation schema R^* with attributes X is in PJ/NF if, for each JD σ in R^*, the membership algorithm succeeds with inputs σ and Δ, where Δ is the set of key dependencies of R^*.

Definition 3(b). A 1NF relation schema R^* with attributes X is in PJ/NF if $\Delta \models \sigma$ for each JD σ in R^*, where Δ is the set of key dependencies of R^*. Thus, every JD is the result of keys.

Since Definitions 1(b) and 1(c) are equivalent, as are Definitions 2(b) and 2(c), we might naturally assume that PJ/NF is equivalent to what we call *overstrong PJ/NF* in the following definition.

Definition 4. A 1NF relation schema R^* with attributes X is in overstrong PJ/NF if, for each JD σ in R^*, there is a key dependency K→X of R^* such that K→X $\models \sigma$. Thus, every JD is the result of a key.

However, as we will show in Section 3, overstrong PJ/NF is *not* equivalent to PJ/NF.

By comparing Definitions 1(b), 2(b), and 3(b), we see the striking relationship between BCNF, 4NF, and PJ/NF. In particular, since every MVD can be represented by a JD, the following theorem is immediate.

Theorem 5. PJ/NF implies 4NF.

Of course, we already know (from Theorem 2 of [Fa1]) that 4NF implies BCNF. A new proof of this result is trivially obtained from Definition 1(c) and Corollary 4.

From Definition 2(c), Corollary 4 and Theorem 5, we obtain the following result.

Theorem 6. Let R^* be a relation schema in either 4NF or PJ/NF. Then, for every FD and MVD σ, there is a key dependency K→X of R^* such that K→X $\models \sigma$.

Thus, in a PJ/NF relation schema, every dependency (FD, MVD, and JD) is determined by keys (the FDs and MVDs each by a single key, and the JDs by perhaps several keys). **Hence, in a PJ/NF relation schema, all that needs to be specified is the set of attributes and the set of keys; all FDs, MVDs, and JDs (and in particular, all decompositions) are determined by these.** This is good, since keys are quite fundamental and easily understood. If all relation schemata that correspond to stored relations in a database are in PJ/NF (and if there are no inter-relational dependencies), then the database management system need only have a mechanism for supporting keys, rather than a mechanism for supporting more general dependencies. We remark that in fact, System R [ABC] supports keys (via "unique indices"), but does not support general functional dependencies.

We now show that if a relation schema obeys only those dependencies (FDs, MVDs, and JDs) that are logical consequences of a set of FDs, then BCNF, 4NF, and PJ/NF coincide.

Theorem 7. Assume that the set of dependencies (FDs, MVDs, and JDs) in relation schema R^* equals $\{\sigma: \Sigma \models \sigma\}$, where Σ is a set of FDs. Then all of the following are equivalent:

(i) R^* is in BCNF.

(ii) R^* is in 4NF.

(iii) R^* is in PJ/NF.

Proof: We already know that (iii) implies (ii), and that (ii) implies (i). So we need only show that (i) implies (iii). Thus, assume that R^* is in BCNF; we will show that R^* is in PJ/NF. Let Δ be the set of key dependencies of R^*. Since R^* is in BCNF, we know by Definition 1(c) that for each FD τ in Σ, there is a key dependency τ' in Δ such that $\tau' \models \tau$. Since for each τ in Σ there is τ' in Δ such that $\tau' \models \tau$, it follows that $\Delta \models \Sigma$. Let σ be an arbitrary JD of R^*; thus, $\Sigma \models \sigma$. Since $\Delta \models \Sigma$ and $\Sigma \models \sigma$, it follows by transitivity that $\Delta \models \sigma$. So by Definition 3(b), schema R^* is in PJ/NF. □

Similarly, the following theorem holds.

Theorem 8. Assume that the set of dependencies (FDs, MVDs, and JDs) in relation schema R^* equals $\{\sigma: \Sigma \models \sigma\}$, where Σ is a set of FDs and MVDs. Then the following are equivalent:

(i) R^* is in 4NF.

(ii) R^* is in PJ/NF.

Proof: We already know that (ii) implies (i). We will show that (i) implies (ii). So, assume that R^* is in 4NF; we will show that it is in PJ/NF. Once again, let Δ be the set of key dependencies of R^*. For each FD or MVD τ in Σ, we know by Theorem 6 that there is a key dependency τ' in Δ such that $\tau' \models \tau$. So, $\Delta \models \Sigma$. Let σ be an arbitrary JD in R^*. Since $\Delta \models \Sigma$ and $\Sigma \models \sigma$, it follows by transitivity that $\Delta \models \sigma$. Hence, by Definition 3(b), R^* is in PJ/NF. □

It is easy to see that by repeated decomposition, we can always convert a relation schema that is not in PJ/NF into a family of relation schemata that are (although inter-relational dependencies might be introduced). Compare Theorem 3 of [Fa1], which deals with the 4NF case. We remark that "embedded" JDs must be considered in the PJ/NF decomposition, just as embedded MVDs must be considered in the 4NF case. Investigating the interrelationships between embedded JDs and ordinary JDs is an interesting research problem.

It is important to note that a relation schema in PJ/NF is not necessarily decomposed "as far as it can go". For example, let R^* be a relation schema with attributes ABCD and with its only dependencies being the logical consequences of A→ABCD. Thus, the only dependencies in R^* are those that are the result of A being a key. Then R^* is in PJ/NF, although it is possible to decompose R^* into its projections on AB, AC, and AD, and still to be able to reobtain the original relation from the projections by taking the join. The point is that this decomposition is not necessary, since it does not seem to "buy" anything.

It follows from Definition 3(b) that PJ/NF is the "ultimate" normal form with respect to projection and join, since for a PJ/NF relation schema, the only JDs, and hence the only decompositions, are those that are the result of keys. That is, **a PJ/NF relation schema cannot be decomposed any further, except by decompositions based on keys.** It is an important question to consider criteria by which one decomposition into PJ/NF is "better than" another. A good example of a partial answer to the question is given by Rissanen's concept of "independent components" [Ri1]. When we say that PJ/NF is "ultimate" (with respect to projections and joins), we simply mean that there is no need to decompose further.

We close this section by presenting the following bizarre example (modified slightly from the example given by Nicolas [Ni]), which shows that it is possible for a relation schema to be in 4NF but not in PJ/NF. Assume that there are only three attributes A,P,C, where the tuple (a,p,c) means, intuitively, that agent a represents product p produced by company c. We want this schema to obey the JD $*\{AP,AC,PC\}$, which we will abbreviate by σ. In English, the JD σ says precisely the following:

> "Assume that agent a represents product p produced by company c', that agent a represents product p' produced by company c, and that agent a' represents product p produced by company c. Then agent a represents product p produced by company c."

Nicolas actually assumes the following statement, which we will abbreviate by ν:

> "An agent must not work for two companies whose products overlap."

We now show that $\nu \models \sigma$ (note that we are using "$\models$" in a slightly extended way from before, since ν is not an FD, MVD, or JD.) Assume that ν holds in a relation R. We must show that σ holds in R, that is, that if (a,p,c'), (a,p',c), and (a',p,c) are all tuples of R, then so is the tuple (a,p,c). We will, in fact, show the stronger result that it is impossible for all of (a,p,c'), (a,p',c), and (a',p,c) to appear as tuples in R. Assume that they do. Since (a,p,c') and (a',p,c) both appear, companies c and c' both carry product p. Since agent a works for company c (because of the tuple (a,p',c)), he cannot work for company c', and so the tuple (a,p,c') cannot appear. Since it does, we have reached a contradiction.

Nicolas shows, by giving an example, that ν does not logically imply any nontrivial FD or MVD. Hence, if we let R^* be a relation schema with attributes APC, and with only those dependencies that are logical consequences of σ, then no nontrivial FDs or MVDs hold (because if $\sigma \models \tau$ for a nontrivial FD or MVD τ, then, since $\nu \models \sigma$, it follows by transitivity that $\nu \models \tau$, which contradicts Nicolas' result.) So R^* is a relation schema in 4NF (since it has no nontrivial FDs or MVDs). However, it is not in PJ/NF, since the only key dependency of R^* is the trivial FD APC→APC, and Definition 3(b) is violated (since the JD σ, being nontrivial, is not a logical consequence of the trivial dependency APC→APC.) To obtain PJ/NF, it is necessary to decompose on the basis of σ, into the projection of R^* on each of AP, AC, and PC.

Note that if the only restriction we know about the schema R^* is the JD σ given by $*\{AP,AC,PC\}$ (and if, in particular, we do *not* assume that Nicolas' statement ν holds), then it is certainly desirable to decompose R^* into its projections on AP, AC, and PC. For, it is then possible to independently update each of these 3 relations; a tuple can be inserted into or deleted from any of these relations with no effect on any other tuple. The decomposition automatically enforces the JD σ. By contrast, in the original relation with attributes APC, the addition of a tuple may force the presence of other tuples. Also, in this original relation, it may be illegal to delete certain tuples, since the deletion may cause the violation of the JD σ.

We remark that in this example, there is a simple direct proof [Me] that if τ is an FD or MVD such that $\sigma \models \tau$, then τ is necessarily trivial. For, if τ is nontrivial, then it is easy to see that there is a two-tuple relation R such that τ fails in R. However, we leave to the reader the fairly straightforward verification that, since the union of any two of AP, AC, and PC is the set X of all attributes, necessarily $\sigma = *\{AP,AC,PC\}$ holds in every 2-tuple relation. Thus, it is impossible that $\sigma \models \tau$, since our 2-tuple relation R is a counterexample relation in which σ holds and τ fails.

3. An overstrengthening of PJ/NF

In this section, we discuss "overstrong PJ/NF", defined in Definition 4 of the previous section.

Since Definitions 1(b) and 1(c) are equivalent, as are Definitions 2(b) and 2(c), it is natural to believe by analogy that PJ/NF (Definition 3(b)) is equivalent to overstrong PJ/NF (Definition 4). We now show that this is not the case. Let R^* be a relation schema with attributes X = ABCD, and with only those dependencies that are the logical consequences of the key dependencies A→X and B→X. Thus, the dependen-

cies in R^* are precisely those that are the result of A and B being keys. So, R^* is in PJ/NF. We now show that R^* is not in overstrong PJ/NF. Let σ be the JD $*\{AB,AD,BC\}$. As we showed in the example immediately following our description of the membership algorithm, σ is a logical consequence of the key dependencies A→X and B→X. Hence, σ is a JD of R^*. However, it is false that $A \rightarrow X \models \sigma$, as the reader can see by considering our membership algorithm where we let Δ be $\{A \rightarrow X\}$. Similarly, it is false that $B \rightarrow X \models \sigma$. So, there is no key dependency K→X of R^* such that $K \rightarrow X \models \sigma$. Hence, R^* is not in overstrong PJ/NF.

Note also that R^*, being in PJ/NF, is in BCNF. So, Theorem 7 fails if we replace "PJ/NF" by "overstrong PJ/NF". A similar comment holds for Theorem 8.

We feel that, as the name implies, overstrong PJ/NF is too demanding, since, for example, the schema R^* we just analyzed is not in overstrong PJ/NF. However, we find it interesting from a theoretical point of view to explore overstrong PJ/NF a little more carefully. Hence, we will give two definitions of overstrong PJ/NF, and show that they are equivalent. First, we need to define two more concepts.

We say the JD $*\{X_1,...,X_m\}$ *covers* the JD $*\{Y_1,...,Y_p\}$ if

(i) they contain the same attributes, that is, $\cup\{X_i: 1 \le i \le m\} = \cup\{Y_j: 1 \le j \le p\}$; and

(ii) each Y_j equals an X_i, that is, for each j ($1 \le j \le p$) there exists i ($1 \le i \le m$) such that $Y_j = X_i$.

For example, the JD $*\{Z_1,Z_2,Z_3\}$ covers the JD $*\{Z_1,Z_2\}$, provided they contain the same set of attributes, that is, provided every attribute in Z_3 is also in Z_1 or Z_2. We say that a JD in a relation schema is *minimal* if no JD that it covers is also in the schema.

Lemma 9: If the JD σ covers the JD τ, then $\tau \models \sigma$.

Proof: Assume that the JD τ holds in relation R. We must show that the JD σ also holds in R. Let us write σ as $*\{X_1,...,X_m\}$, and τ as $*\{Y_1,...,Y_p\}$. Let t be an arbitrary tuple for which there are tuples $w_1,...,w_m$ of R such that $w_i[X_i] = t[X_i]$ for each i. As we noted in the introduction, to show that σ holds in R, we need only show that the tuple t is in R. For each j ($1 \le j \le p$), find i_j such that $X_{i_j} = Y_j$. Then $w_{i_j}[Y_j] = w_{i_j}[X_{i_j}] = t[X_{i_j}] = t[Y_j]$. Now, the tuples w_{i_j} are in R ($1 \le j \le p$), and we have shown that $w_{i_j}[Y_j] = t[Y_j]$ ($1 \le j \le p$). So, since τ holds for R, the tuple t is in R. This was to be shown. □

We remark that Lemma 9 is a special case of a stronger result in [ASU]. Further, it is essentially the same as Proposition 5 of [Ni].

We now present two definitions of overstrong PJ/NF. The second definition (Definition 4(b) below) is the same as our earlier Definition 4.

Definition 4(a). A 1NF relation schema R^* with attributes X is in overstrong PJ/NF if, for each minimal JD $*\{X_1,...,X_m\}$ in R^*, there is a key K of R^* such that K is a subset of each X_i.

Definition 4(b). A 1NF relation schema R^* with attributes X is in overstrong PJ/NF if, for each JD σ in R^*, there is a key dependency K→X of R^* such that $K \rightarrow X \models \sigma$. Thus, every JD is determined by a key.

To see why the word "minimal" cannot be eliminated from Definition 4(a), let R^* be a relation schema with attributes ABC, whose only dependencies are those caused by A being a key. That is, the only dependencies in R^* are A→ABC and its logical consequences. Then R^* is in overstrong PJ/NF, according to Definition 4(b). We now show that it does not obey a modified version of Definition 4(a) with the word "minimal" eliminated. By Heath's Theorem [He],

$$A \rightarrow ABC \models *\{AB,AC\}.$$

By Lemma 9,

$$*\{AB,AC\} \models *\{AB,AC,BC\}.$$

By transitivity,

$$A \rightarrow ABC \models *\{AB,AC,BC\}.$$

Thus, $*\{AB,AC,BC\}$ is in R^*. However, the (only) key A is not in BC.

Theorem 10. Definitions 4(a) and 4(b) are equivalent.

Proof: Assume first that a relation schema R^* is in overstrong PJ/NF according to Definition 4(a). We will show that it is in overstrong PJ/NF according to Definition 4(b). Let σ be a JD in R^*. We must show that there is a key dependency K→X such that $K \rightarrow X \models \sigma$. It is easy to see that the schema R^* contains a minimal JD τ that is covered by σ (τ may, of course, be σ itself.) By Lemma 9, $\tau \models \sigma$. Write τ as $*\{Y_1,...,Y_p\}$. By Definition 4(a), there is a key K of R^* such that K is a subset of each Y_j. It follows easily from our membership algorithm that, since K is a subset of each Y_j, necessarily $K \rightarrow X \models \tau$. Since $K \rightarrow X \models \tau$, and since $\tau \models \sigma$, it follows by transitivity that $K \rightarrow X \models \sigma$. This was to be shown.

Now assume that a relation schema R^* is in overstrong PJ/NF according to Definition 4(b). We must show that it is also in overstrong PJ/NF according to Definition 4(a). Let σ be a minimal JD of R^*. Assume that σ is $*\{X_1,...,X_m\}$. By Definition 4(b), there is a key dependency K→X of R^* such that $K \rightarrow X \models \sigma$. Since K is a key, we need only show that K is a subset of each X_i. Assume not. Without loss of generality, assume that K is not a subset of X_1. Denote by A an attribute in K that is not in X_1. We will derive a contradiction.

Since σ is a *minimal* JD of R^*, we know that $*\{X_2,X_3,...,X_m\}$ is not a JD of R^*. There are two possible reasons why.

Case 1: There is an attribute in X_1 that is not in any of the other X_i's. Denote this attribute by B. Note that $A \neq B$, since B is in X_1, and A is not. Define a two-tuple relation Q as follows. One tuple (call it u) has only zeros as entries. The other tuple (call it v) has a one in the A and B columns and zeros everywhere else. Denote by t a tuple that has a one in the B column and zeros everywhere else. In particular, tuple t is not in relation Q. Now, $v[X_1] = t[X_1]$, since both $v[X_1]$ and $t[X_1]$ have a one in the B column and zeros everywhere else (recall that X_1 contains B but not A.) Further, $u[X_i] = t[X_i]$, for $2 \le i \le m$, since they both contain only zeros (recall that X_i does not contain B, if $2 \le i \le m$.) It now follows from the definition of join dependency that since t is not in Q, the JD σ does not hold in Q. However, the FD K→X holds in Q, since the two-tuples of Q differ in the A column (and A is in K). So, relation Q is a counterexample relation that obeys K→X but disobeys σ. This contradicts the fact that $K \rightarrow X \models \sigma$.

Case 2: X_1 is contained in the union of $X_2, X_3, \ldots, X_m$. Denote the JD $*\{X_2, X_3, \ldots, X_m\}$ by τ. Since σ is minimal, we know that τ is not in the relation schema R^*. In particular, since the FD K−X is in R^*, we know that it is false that K−X $\models \tau$. Thus, there is a counterexample relation S such that K−X holds for S, but τ does not. Since τ does not hold in relation S, there are tuples $w_2, w_3, \ldots, w_m$ in S and a tuple t not in S such that $w_i[X_i] = t[X_i]$, for $2 \le i \le m$. Let z be a new tuple that agrees with t everywhere except in the A column (recall that A is an attribute that is in K but not in X_1). The value of z in the A column is defined to be a new value, that does not appear in t or in any tuple of S. Let T be a new relation that consists of all tuples in S, along with the tuple z. We will now show that T is a counterexample relation, in which K−X holds, but in which σ fails. This contradicts our assumption that K−X $\models \sigma$. It is easy to see that the FD K−X holds in T, since it holds in S, and T differs from S only by having a new tuple with a new value in the A column (and A is in K). To complete the proof, we need only show that σ does not hold in T. Now $z[X_1] = t[X_1]$ by construction of z, since attribute A is not in X_1. We already know that $w_i[X_i] = t[X_i]$, for $2 \le i \le m$. However, t is not in T. So the JD σ does not hold in T, as was to be shown. □

4. Allowing other relational operators

Perhaps the next important operator to consider is the union (and a corresponding inverse operator, which we will call the split). Let $\psi(t)$ be a formula of the relational calculus [Co2], with one free tuple variable t. The *split* (with respect to $\psi(t)$) of a relation R is the set of two relations R_1 and R_2, each with the same attributes as R, where the set of tuples in R is the union of the tuples in R_1 and R_2. The relations R_1 and R_2 are disjoint, and a tuple t of R is in R_1 if and only if it obeys $\psi(t)$.

In the previous section, our ground rules allowed us to decompose a relation, that is, to replace it by a set of its projections, provided the original relation was the join of the new relations. In this section, we not only allow a relation to be decomposed (provided, as before, that the operation can be reversed by taking joins); we furthermore allow a relation to be split (possibly repeatedly); the relation that was split is then the union of the new relations. We allow a concatenation of decompositions and splits, possibly interleaved.

Smith [Sm] gives several interesting examples where splitting is desirable. As one of his examples, assume that there is a relation schema P_ROLE with attributes PERSON and ROLE. A PERSON is assume to have two types of ROLEs: sex role (male or female) and profession. It is assumed that a person can have several professions, but only one sex. The schema can be split into two schemata, SEX_ROLE and PROF_ROLE. Here, $\psi(t)$ is

$$(t.ROLE = \text{'male'}) \vee (t.ROLE = \text{'female'}).$$

The schema SEX_ROLE obeys the FD PERSON−ROLE, while the schema PROF_ROLE does not. As Smith notes, an advantage of splitting is that the system's key maintenance mechanism can then prevent the insertion of two sex roles for one person.

As a second of Smith's examples, assume that there is a VEHICLE schema with a number of attributes, such as WINGSPAN and SAIL_AREA. Certain of these attributes do not apply to certain vehicles. For example, WINGSPAN applies to air vehicles, but does not apply to water vehicles. The schema can be split into several schemata (such as AIR_VEHICLE and WATER_VEHICLE), on the basis of VEHICLE_TYPE. It is then desirable to "drop" inapplicable attributes (by taking projections) in the new schemata. The net result is to eliminate what Codd calls "property inapplicable" nulls (as opposed to his "value at present unknown" nulls) [Co4].

A possible definition (which, as we will show later, must be modified) of a "projection-split-join-union normal form" (PSJU/NF) is: a relation schema R^* is in PSJU/NF if (i) it is in PJ/NF, and (ii) there is no way to split R^* (on the basis of an expression $\psi(t)$) into relation schemata R_1^* and R_2^*, where the set of dependencies that hold in R_1^* is not the same as the set of dependencies that hold in R_2^*.

Smith's first example then violates PSJU/NF (and is therefore split), since the schema SEX_ROLE obeys the FD PERSON−ROLE, while the schema PROF_ROLE does not.

Note that if attribute A "does not apply" to tuples in the relation R_1, then R_1 obeys the FD ∅−A (where ∅ is the empty set). Thus, in Smith's second example, an instance of the schema WATER_VEHICLE obeys the FD ∅−WINGSPAN, whereas the set of remaining tuples do not. Thus, a split is called for.

A major defect of our definition of PSJU/NF as it stands is that we have given no restrictions on what $\psi(t)$ may be. For example, by letting $\psi(t)$ completely specify a single tuple, the split operator would simply split off a single tuple, and this process of splitting off single tuples could take place repeatedly (note that a one-tuple relation obeys every FD, MVD, and JD.) As a more subtle example, assume that we have a SUPPLY relation schema with only two attributes, SUPPLIER and PART. Let $\psi(t)$ be the relational calculus equivalent of "t.SUPPLIER supplies every PART". On the basis of $\psi(t)$, the SUPPLY schema is split into two schemata UNIV_SUPPLY and NON_UNIV_SUPPLY, where universal SUPPLIERs (those SUPPLIERs who supply every PART) appear in the UNIV_SUPPLY relation; thus, this schema obeys the MVD ∅→→SUPPLIER. So, the UNIV_SUPPLY schema can then be decomposed into two schemata UNIV_SUPPLIER (with one attribute, SUPPLIER), and PART (with one attribute, PART). We are left with three schemata: one lists the universal suppliers, one lists all parts, and the third lists all (supplier,part) pairs for non-universal suppliers. We note that in Codd's relational algebra [Co2], the "quotient" of the SUPPLY schema by the PART schema is the UNIV_SUPPLY schema, and the "remainder" is the NON_UNIV_SUPPLY schema. Is it worthwhile to split the original SUPPLY schema in the first place? It probably depends on whether or not we expect a large proportion of the suppliers to be universal suppliers.

We believe that it is an important research problem to clarify which expressions $\psi(t)$ we should allow in our definition of PSJU/NF.

5. Acknowledgements

The author is grateful to Y. Sagiv and E. F. Codd for valuable suggestions. He also thanks C.J. Date for helpful comments.

BIBLIOGRAPHY

[ABC] M. M. Astrahan, M.W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson, System R: A relational approach to data base management. TODS 1,2 (June 1976), 97-137.

[ABU] A. V. Aho, C. Beeri, and J. D. Ullman, The theory of joins in relational data bases. Proc. 19th IEEE Symp. on Foundations of Computer Science (Oct. 1977), 107-113.

[ASU] A. V. Aho, Y. Sagiv, and J. D. Ullman, Equivalences among relational expressions. To appear, SIAM J. Computing.

[Ar] W. W. Armstrong, Dependency structures of database relationships. Proc. IFIP 74, North Holland (1974), 580-583.

[BBG] C. Beeri, P. A. Bernstein, and N. Goodman, A sophisticate's introduction to database normalization theory. Proc. 1978 VLDB (Berlin), 113-124.

[Be] P. A. Bernstein, Synthesizing third normal form relations from functional dependencies. TODS 1,4 (Dec. 1976), 277-298.

[Ca] J.-M. Cadiou, On semantic issues in the relational model of data. Proc. Int. Symp. on Math. Foundations of Computer Science, Gdansk, Poland, Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, Sept. 1975.

[Co1] E. F. Codd, A relational model of data for large shared data banks. Comm. ACM 13,6 (June 1970), 377-387.

[Co2] E. F. Codd, Relational completeness of data base sublanguages. Courant Computer Science Symposia 7, Data Base Systems, New York City, May 24-25, 1971, Prentice Hall.

[Co3] E. F. Codd, Recent investigations in relational data base systems. Proc. IFIP Congress 74, North-Holland (1974), 1017-1021.

[Co4] E. F. Codd, Understanding relations. Installment No. 7, FDT (bulletin of ACM SIGMOD) 7, 3-4, Dec. 1975.

[DB] U. Dayal and P. A. Bernstein, The fragmentation problem: lossless decomposition of relations into files. Technical report CCA-78-13, Computer Corporation of America, Cambridge Mass. (Nov. 15, 1978).

[Fa1] R. Fagin, Multivalued dependencies and a new normal form for relational databases. TODS 2,3 (Sept. 1977), 262-278.

[Fa2] R. Fagin, Functional dependencies in a relational database and propositional logic. IBM J. Res. and Devel. 21,6 (Nov. 1977), 534-544.

[He] I. J. Heath, Unacceptable file operations in a relational data base. Proc. 1971 ACM-SIGFIDET workshop on data description, access, and control, San Diego, Calif., Nov. 11-12, 1971.

[Me] A. Mendelzon, Private communication received transitively through Y. Sagiv.

[Ni] J. M. Nicolas, Mutual dependencies and some results on undecomposable relations. Proc. 1978 VLDB (Berlin), 360-367.

[Ri1] J. Rissanen, Independent components of relations. TODS 2,4 (Dec. 1977), 317-325.

[Ri2] J. Rissanen, Theory of relations for databases - a tutorial survey. Proc. 7th Symp. on Math. Found. of Comp. Science, Lecture notes in Comp. Science, 64, Springer-Verlag, 537-551.

[Sm] J. M. Smith, A normal form for abstract syntax. Proc. 1978 VLDB (Berlin), 156-162.