The Theory of Joins in Relational Data Bases

(Extended Abstract)

A. V. Aho
Bell Laboratories
Murray Hill, New Jersey

C. Beeri and J. D. Ullman***
Princeton University
Princeton, New Jersey

SUMMARY

Answering queries in a relational database often requires that the natural join of two or more relations be computed. However, not all joins are semantically meaningful. This paper gives an efficient algorithm to determine whether the join of several relations is semantically meaningful *(lossless)* and an efficient algorithm to determine whether a set of relations has a subset with a lossy join. These algorithms assume that all data dependencies are functional. Similar techniques also apply to the case where data dependencies are multivalued.

1. Introduction

The relational model of data, originating with the work of Codd [5, 6], has recently attracted a considerable amount of interest. In this model, data entries are arranged into one or more multi-column tables called *relations.* The columns of the tables are labeled by *attributes.* The values of the entries in a column are chosen from a set called the *domain* for the corresponding attribute. As an example, consider a database with five tables having the following formats:

NA = (NAME, ADDRESS)
NP = (NAME, PHONE)
PD = (PHONE, DISTRICT)
AZ = (ADDRESS, ZIP)
ZD = (ZIP, DISTRICT)

All attributes above should have a meaning that is self-explanatory, except possibly for "DISTRICT," which is intended to be the territory of the phone company serving the subscriber.

The nature of the data itself may impose restrictions on the values of certain entries. For example, in our database we assume that at any one time each person has only one address, and, for convenience, only one phone. This type of "data integrity" can be captured by *functional dependencies* between sets of attributes.

Informally we say that a set of attributes X *functionally determines* a set of attributes Y, written $X \rightarrow Y$, if for any assignment of values to the attributes of X, there can be at any one time at most one value for each of the attributes of Y. For example, in our database NAME → ADDRESS, PHONE. The complete list of functional dependencies we shall assume for our running example is:

NAME → ADDRESS
NAME → PHONE
ADDRESS → ZIP
PHONE → DISTRICT
ZIP → DISTRICT

Of these, only the last is controversial. We assume, for the sake of an interesting example later on, that each zip code is wholly contained within one district.

To answer queries put to the database, we may have to consult and put together pieces of information from several tables. For example, suppose that in our running example we wish to answer the query:

> "What are the zip codes of the people with phone number 609-452-4646?"

Since we do not have a single table that has both ZIP and PHONE as attributes, to answer the query we must select and piece together information from several tables. Different answers may result depending on how we do this. For example, if we choose to combine the information from the tables PD and ZD, then we incorrectly associate every zip code in New Jersey (the district including

* Work partially supported by NSF grant DCR-74-21939.
** Work partially supported by NSF grant MCS-76-15255.

area code 609) with the phone number 609-452-4646. On the other hand, if we combine the information from the tables NA, NP and AZ, we obtain the correct answer; the phone number is associated only with the correct zip codes, even though there may be different people at different addresses associated with one phone number.

Putting together pieces of information from several tables can be viewed as combining (or joining) the tables to form a bigger table. As we have seen, to answer a query involving several attributes, it is not enough to compose tables that, between them, have all the attributes in the query. The goal of this paper is to characterize when such compositions of tables are correct.

The issue of correctness of joins is not only of intellectual interest, but also of practical significance in the design and manipulation of relational databases. Designing a relational database requires knowing what attributes to group together into single tables. Various such groupings are often possible in a given situation. In existing systems (e.g., INGRES [14]), the user selects and defines the table formats. There are also proposals for constructing table formats automatically, given only the dependencies that pertain to the situation [3,9,12]. In either case, it is important to select a "good" set of table formats, and certainly one property to look for in such a set is the "correctness" of its joins. Once a format has been selected, users formulate queries against the database. Here again, either the user selects the joins to be made, or these are selected by the system; in any case, it is important to make only "correct" joins.

Whether a set of table formats admits only "correct" joins depends on the dependencies that are given. We study here two types of dependencies. Functional dependencies have been known since the emergence of the relational model [5,6] and their properties have been studied exhaustively [1-4]. Recently, another type of dependency called a *multivalued dependency,* has been introduced [4,8,10,15]. For both types of dependencies our results indicate a nontrivial connection between the given dependency structure and correctness of joins.

2. The Formal Model

In order to talk about correctness of operations on a database, we must have a model of the "universe" that the database is supposed to represent. The model we shall use is that of [1,3,4]. The user view of the world is a finite set of tuples, with one component for each attribute of the database. Formally, a *universe* is a finite set of *attributes,* each with an associated *domain* of values, together with a set of *constraints.* For the time being we assume the constraints are functional dependencies. In Section 5 we shall deal with multivalued dependencies.

An *element* of the universe is a map μ from the set of attributes to the corresponding domains. We shall often assume an ordering $A_1, A_2, \ldots, A_k$ for the attributes and shall represent an element μ in the traditional way, as a k-tuple $\mu(A_1), \mu(A_2), \ldots, \mu(A_k)$. An *instance* of the universe is a finite set of elements that satisfies all the given constraints.

A *functional dependency* is a statement $X \rightarrow Y$, where X and Y are sets of attributes. An instance I satisfies this functional dependency if and only if for all μ and ν in I, if $\mu(A) = \nu(A)$ for all A in X, then $\mu(B) = \nu(B)$ for all B in Y. That is, if two rows of I agree in the columns for X, then they agree in the columns for Y. Note that if I satisfies a given set of dependencies, then it also satisfies additional dependencies, e.g., if I satisfies $A \rightarrow B$ and $B \rightarrow C$, it also satisfies $A \rightarrow C$. Inference rules for functional dependencies have been studied, and several complete sets of rules have been presented [1,4,10]. We use here a formalism that is equivalent to these complete sets of rules.

Assume that a set of functional dependencies is given. For a set of attributes X, we define $CL(X)$, the *closure* of X, as follows:

(1) $X \subseteq CL(X)$

(2) If $Y \subseteq CL(X)$, and $Y \rightarrow Z$ is a given functional dependency, then $Z \subseteq CL(X)$.

(3) No attribute is in $CL(X)$ unless it so follows from (1) and (2).

We write $X \stackrel{*}{\rightarrow} Y$ if $Y \subseteq CL(X)$. Essentially, $X \stackrel{*}{\rightarrow} Y$ means that the functional dependency $X \rightarrow Y$ is in or can be derived from the given set of dependencies. Two sets of dependencies are *equivalent* if, for all X, $CL(X)$ is the same under either set of dependencies. It is known that we preserve equivalence if we replace $X \rightarrow Y$ by $X \rightarrow A_1, \ldots, X \rightarrow A_k$, where Y is the set $\{A_1, \ldots, A_k\}$. We shall therefore assume henceforth that all functional dependencies have a single attribute on the right.

A *database schema* is a finite set of table formats, formally called *relation schemes,* where each relation scheme is a set of attributes. A *database* is a collection of relations, one for each relation scheme, where a *relation* is a finite set of *tuples,* that is, maps from the attributes in the relation scheme to their domains. (Each relation can be viewed as a set of rows in a table whose format is specified by the corresponding relation scheme). A database is a representation of an instance of the universe. To define this representation, we need two operations, projection and (natural) join.

Given an instance I on the set of attributes X, and a subset $Y \subseteq X$, we define $\pi_Y(I)$, the *projection* of I onto Y, to be the set of maps μ from Y to the corresponding domains, such that for some ν in I, ν agrees with μ on Y. We can think of the projection of I onto Y as the operation that takes the table represented by I, deletes all columns except those labeled by attributes in Y, and then identifies common rows.

The (natural) *join* is an "inverse" to the projection operator defined as follows. Let $R_1, R_2, \ldots, R_m$ be relation schemes, and let their current values be the relations $r_1, r_2, \ldots, r_m$. Then the join

$$\bowtie_{i=1}^{m} r_i = \{\, \mu \mid \mu \text{ is a map on the set of attributes } \bigcup_{i=1}^{m} R_i \text{ such that for all } 1 \leqslant i \leqslant m \text{ there is a map } \nu_i \text{ in } R_i \text{ that agrees with } \mu \text{ on } R_i \}.$$

For example, if $R_1 = AB$ and $R_2 = BC$, where $r_1 = \{ab,\ a'b'\}$ and $r_2 = \{bc,\ bc',\ b'c''\}$, then

$$r_1 \bowtie r_2 = \{abc,\ abc',\ a'b'c''\}.$$

As another example, if all R_i's are disjoint, then the join

is just the Cartesian product of the r's. Note that the join operator is associative and commutative, and that for each i, $\pi_{R_i}(\bowtie_{j=1}^{m} r_j) = r$.

A database that represents a given instance I of the universe is simply the set of projections of I onto the relation schemes. Rissanen [13] shows that if we want any one-to-one correspondence to hold between instances and projections, then the join is the only operation that can be used to recover an instance from the projections.

We say that a set $\{R_1, R_2, \ldots, R_m\}$ of relation schemes has the *lossless join* property (or, briefly, that $R_1, R_2, \ldots, R_m$ have a lossless join) if, for all instances I that satisfy the given dependencies:

$$\pi_R(I) = \bowtie_{i=1}^{m} \pi_{R_i}(I) \qquad (1)$$

where $R = \bigcup_{i=1}^{m} R_i$. That is to say, whatever instance of the universe pertains at the moment, provided only that it satisfies the constraints associated with the universe, we can recover its projection onto the union of the attributes of the R_i's by using the join operator. If $\{R_1, R_2, \ldots, R_m\}$ does not have this property, we say that it has a *lossy join*.

The term "lossless" refers to the fact that the information contained in I is not lost if we replace I by its projections onto the R_i's, or if I has already been replaced by some of its projections, and one of those is $\pi_R(I)$, then no information is lost if this projection is replaced by the projections on the R_i's.

Note that if (1) does not hold for all I, it is at least true that the left side is contained in the right side. When the containment is strict, however, some information about I is lost. For example, if we had in our running database a relation scheme PDZ = (PHONE, DISTRICT, ZIP), then we would have in the corresponding relation for each phone number the associated zip code. Suppose we have replaced the relation scheme PDZ by PD and DZ. Then when we attempt to recreate PDZ by joining PD and DZ, we can associate with a given phone number, via the join, erroneous zip codes, so we have a lossy join. For instance, suppose PDZ has the value

P	D	Z
201-582-4862	NJ	07974
609-452-4646	NJ	08540

Projecting PDZ onto PD and PZ, we obtain

P	D
201-582-4862	NJ
609-452-4646	NJ

D	Z
NJ	07974
NJ	08540

Taking the join of PD and DZ, we obtain

P	D	Z
201-582-4862	NJ	07974
201-582-4862	NJ	08540
609-452-4646	NJ	07974
609-452-4646	NJ	08540

As we see, this larger relation obscures the information that phone number 201-582-4862 is only associated with zip code 07974 and that 609-452-4646 is only associated with 08540.

3. An Algorithm to Test for Losslessness

This section shows that there is a quartic time algorithm to test whether a set of relation schemes with a given set of functional dependencies has a lossless join. Let us fix our attention on a particular set of relation schemes $R_1, R_2, \ldots, R_m$, whose union R is the set of attributes $A_1, A_2, \ldots, A_n$. Let $A_{n+1}, \ldots, A_p$ be the attributes not in R. By fixing on this attribute ordering, we can represent elements (mappings) as p-tuples. From the definition of join it follows that for any instance I

$$\bowtie_{i=1}^{m} \pi_{R_i}(I) = \{a_1 a_2 \cdots a_n \mid \text{there exist } p\text{-tuples } w_1, w_2, \ldots, w_m \text{ in } \pi_R(I) \text{ such that } w_i \text{ has } a_j \text{ in position } j \text{ if the attribute } A_j \text{ is in } R_i \text{ and an arbitrary value in position } j \text{ otherwise}\}.$$

Our algorithm to test for losslessness begins by writing down the table of w_i's that produce an n-tuple of the form $a_1 a_2 \cdots a_n$ in the join. We use a new symbol b_{ij} if the jth position of w_i is arbitrary. We then use the given functional dependencies to infer equalities among the a_i's and the b_{ij}'s. That is, if two w_i's have the same symbols in the set of columns X, and $X \rightarrow A$ is a dependency, then the symbols in the column for A in these rows may be equated.

If we infer that some w_i must begin with $a_1 a_2 \cdots a_n$, then we have shown that an arbitrary element $a_1 a_2 \cdots a_n$ of $\bowtie_{i=1}^{m} \pi_{R_i}(I)$ is in $\pi_R(I)$, given that I satisfies the dependencies, so $\bowtie_{i=1}^{m} \pi_{R_i}(I) \subseteq \pi_R(I)$. Since inclusion in the other direction is obvious, we have proved (1) for an arbitrary I satisfying the dependencies, and we conclude that the set $\{R_1, R_2, \ldots, R_m\}$ has the lossless join property. In the opposite case, where we cannot infer that some w_i is $a_1 a_2 \cdots a_n$, we have a counterexample to the lossless join property if we let I be the set of w_i's that result after all equality inferences are made, treating the a_i's and b_{ij}'s as distinct constants. That is, $a_1 a_2 \cdots a_n$ is not in $\pi_R(I)$ but is in $\bowtie_{i=1}^{m} \pi_{R_i}(I)$. Thus we have:

Theorem 1. There is an algorithm of time complexity $O(n^4)$ to test whether a set of relation schemes has the lossless join property, where n is the space needed to write down the relation schemes, attributes, and functional dependencies. □

Example 1. Let the set of attributes be $\{A, B, C, D, E, F, G\}$ and let the set of relation schemes be

$$R_1 = (ABDE)$$
$$R_2 = (ACDF)$$
$$R_3 = (BCEF)$$

with the functional dependencies $A \rightarrow B$ and $F \rightarrow E$. We form the following table.

	A	B	C	D	E	F	G
$w_1 =$	a_1	a_2	b_{13}	a_4	a_5	b_{16}	b_{17}
$w_2 =$	a_1	b_{22}	a_3	a_4	b_{25}	a_6	b_{27}
$w_3 =$	b_{31}	a_2	a_3	b_{34}	a_5	a_6	b_{37}

Applying $A \rightarrow B$ we infer that $b_{22} = a_2$. That is, since w_1 and w_2 agree on A, they must agree on B. Similarly, $F \rightarrow E$ allows us to infer that $b_{25} = a_5$, so $w_2 = a_1 a_2 \cdots a_6 b_{27}$, which has only a's in the attributes of $R_1 \cup R_2 \cup R_3$. Hence the set $\{R_1, R_2, R_3\}$ has a lossless join. □

Example 2. To see how inferences of equalities among b's can be important, consider

$R_1 = (BEF)$
$R_2 = (ABE)$
$R_3 = (AD)$
$R_4 = (CEF)$

with dependencies

$A \rightarrow C$ $\quad$ $E \rightarrow D$
$B \rightarrow C$ $\quad$ $D \rightarrow A$
$C \rightarrow D$ $\quad$ $D \rightarrow B$

We begin with the following table.

	A	B	C	D	E	F
$w_1 =$	b_{11}	a_2	b_{13}	b_{14}	a_5	a_6
$w_2 =$	a_1	a_2	b_{23}	b_{24}	a_5	b_{26}
$w_3 =$	a_1	b_{32}	b_{33}	a_4	b_{35}	b_{36}
$w_4 =$	b_{41}	b_{42}	a_3	b_{44}	a_5	a_6

Since $A \rightarrow C$ we may replace b_{33} by b_{13}. Since $B \rightarrow C$ we may replace b_{23} by b_{13}. Since $C \rightarrow D$ we may replace a_4 and b_{24} by b_{14}. Since $E \rightarrow D$ we may replace b_{44} by a_4, giving:

	A	B	C	D	E	F
$w_1 =$	b_{11}	a_2	b_{13}	a_4	a_5	a_6
$w_2 =$	a_1	a_2	b_{13}	a_4	a_5	b_{26}
$w_3 =$	a_1	b_{32}	b_{13}	a_4	b_{35}	b_{36}
$w_4 =$	b_{41}	b_{42}	a_3	a_4	a_5	a_6

Since $D \rightarrow A$ and $D \rightarrow B$, b_{41} becomes a_1 and b_{42} becomes a_2, so $w_4 = a_1 a_2 \cdots a_6$, and the join of R_1, R_2, R_3, and R_4 is lossless. □

Lemma 1. Let **R** be a set of relation schemes. Suppose that for some R and S in **R**, there is an attribute A in $S - R$ such that $R \cap S \overset{*}{\rightarrow} A$. Then **R** has a lossless join if and only if the set $\mathbf{R} - \{R\} \cup \{R \cup \{A\}\}$ has a lossless join.

Proof. For any instance I, the two sets of projections have the same ultimate table after all inferences are made according to the algorithm just described. □

For example, we may adjoin D to R_3 in Example 2.

Lemma 2. Let **R** be a set of relation schemes, let R and S be in **R**, with $R \subseteq S$. Then the **R** has a lossless join if and only if $\mathbf{R} - \{R\}$ has a lossless join.

Proof. It is easy to show that if in the above algorithm the row for R becomes $a_1 a_2 \cdots a_n$, so does the row for S. □

Theorem 2. Let **R** be a set of relation schemes. Then
(a) If **R** has a lossless join, then **R** can be transformed into a single relation scheme by the operations

(1) If $R \cap S \neq \varnothing$, adjoin $CL(R \cap S)$ to R and S.

(2) If $R \subseteq S$, delete R.

(b) If **R** can be transformed to a single relation scheme by the operations

(1) If $R \cap S \neq \varnothing$, adjoin $CL(R \cap S) \cap (R \cup S)$ to R and S.

(2) If $R \subseteq S$, delete R. then R has a lossless join.

Proof. (a) follows by examination of the algorithm for Theorem 1. (b) is from Lemmas 1 and 2. Details are omitted. □

Example 2 shows that (b) is not sufficient for losslessness.

Corollary 1. [13] $R \bowtie S$ is a lossless if and only if $R \cap S \overset{*}{\rightarrow} R$ or $R \cap S \overset{*}{\rightarrow} S$. □

Example 3. In our running example, the relation schemes PD = (PHONE, DISTRICT) and ZD = (ZIP, DISTRICT) have a lossy join, since PD ∩ ZD = {DISTRICT} and neither PHONE nor ZIP functionally depend on DISTRICT.

4. Databases Without Lossy Joins

We now turn to the problem of whether a collection **R** of relation schemes has a subset with a lossy join. We note that if two relation schemes are disjoint, then their join is surely lossy (and usually there is no reason to join disjoint relations). Therefore we define a subset $\{R_1, R_2, \ldots, R_m\}$ of **R** to be *nontrivial* if it has no proper partition into two sets such that the unions over the two sets are disjoint. We also say informally that $R_1, R_2, \ldots, R_m$ have a nontrivial join.

Surely, one condition for R to have no nontrivial lossy joins is that all nondisjoint pairs of relation schemes satisfy the condition of Corollary 1. Let us therefore form a directed graph $G(\mathbf{R})$, whose nodes are the relation schemes of **R**, having an arc from R to S if and only if $R \cap S \overset{*}{\rightarrow} S$. We use the notation $R \Longrightarrow S$ to indicate an arc from R to S. Note that there is a distinction between $\Longrightarrow$ used for arcs and $\rightarrow$ or $\overset{*}{\rightarrow}$ used for functional dependencies. However, if $R \Longrightarrow S$, then $R \overset{*}{\rightarrow} S$.

Example 4. $G(\mathbf{R})$ for our example database of Section 1 is shown in Fig. 1.

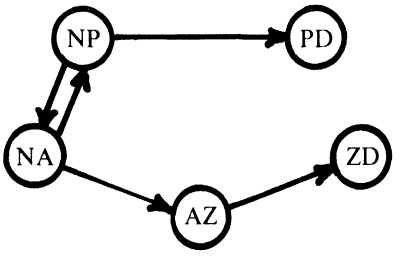

Fig. 1. $G(\mathbf{R})$.

Since whenever a pair of nondisjoint relation schemes does not satisfy the condition of Corollary 1 it has a lossy join, we assume in the rest of this section that each pair

of nondisjoint relation schemes in **R** does satisfy the condition. In the following lemmas we aim to prove a necessary and sufficient condition for **R** satisfying this condition to have no nontrivial lossy join.

Lemma 3. Suppose $X \overset{*}{\not\rightarrow} Y_1$, $X \overset{*}{\not\rightarrow} Y_2, \ldots, X \overset{*}{\not\rightarrow} Y_n$. Then there is an instance $I = \{w, x\}$, where w and x agree on the attributes of X and for all $1 \leqslant i \leqslant n$, disagree on some attribute in Y_i.

Proof. Let w and x agree on all attributes in $CL(X)$ and disagree on all other attributes. Then as $Y_i - CL(X) \neq \varnothing$, we see that I satisfies the conditions of the lemma. To see that I does not violate any functional dependency $Z \rightarrow A$, note that if w and x agree on all attributes of Z, then $Z \subseteq CL(X)$. Therefore, $A \subseteq CL(X)$, so w and x agree on A. □

A *strongly connected region* (SCR) of a graph is a subgraph having a directed path from any node to any other node. The following lemma gives an elementary property of strongly connected regions in $G(\mathbf{R})$.

Lemma 4. In any SCR in $G(\mathbf{R})$, say $\{R_1, R_2, \ldots, R_k\}$, $R_i \overset{*}{\rightarrow} R_j$ for all i and j.

Proof. There is a directed path from R_i to R_j, say $S_1, S_2, \ldots, S_m$, where $R_i = S_1$ and $R_j = S_m$. By definition of $G(\mathbf{R})$, $S_i \cap S_{i+1} \overset{*}{\rightarrow} S_{i+1}$ for all i, $1 \leqslant i < m$. Therefore:

$$S_1 \overset{*}{\rightarrow} S_1 \cap S_2 \overset{*}{\rightarrow} S_2 \overset{*}{\rightarrow} S_2 \cap S_3 \overset{*}{\rightarrow} \cdots \overset{*}{\rightarrow} S_m$$

We have seen in Corollary 1 the condition that makes the join of a pair of relation schemes be lossy. We now give another condition which makes the join of more than two relation schemes be lossy. These two conditions will be seen to characterize those sets of relations that contain a nontrivial lossy join. The forbidden subgraph is shown in Fig. 2. It consists of an SCR T with two other nodes R and S that are not connected by an arc, but that each have arcs into the SCR and no arcs out of the SCR.

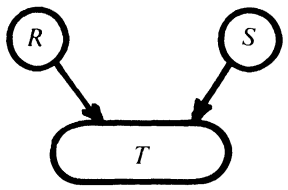

Fig. 2. Forbidden subgraph for lossless joins.

Lemma 5. Let **R** satisfy the condition of Corollary 1 for all pairs of relations with nonempty intersections. Suppose that there is in $G(\mathbf{R})$ an SCR $\{T_1, T_2, \ldots, T_n\}$ and nodes R and S such that $R \Longrightarrow T_1$, $S \Longrightarrow T_n$ and $G(\mathbf{R})$ has no arc $S \Longrightarrow R$ or $R \Longrightarrow S$ and no arc from any T_i to R or S. Then $R \bowtie S \bowtie T_1 \bowtie \cdots \bowtie T_n$ is lossy.

Proof. Let $X = CL(T_1 \cup \cdots \cup T_n)$. If $R \subseteq X$, then by Lemma 4,

$$R \cap T_1 \overset{*}{\rightarrow} T_1 \overset{*}{\rightarrow} T_1 \cup \cdots \cup T_n \overset{*}{\rightarrow} X \overset{*}{\rightarrow} R,$$

so there would be an arc $T_1 \Longrightarrow R$. A similar observation applies to S. Thus, neither R nor S is a subset of X. By our assumption that the condition of Corollary 1 is satisfied for **R**, the fact that there is no arc between R and S implies $R \cap S = \varnothing$. Then by Lemma 3 applied to R, S, and $T_1 \cup \cdots \cup T_n$, there is an instance $I = \{w, x\}$, where w and x agree on $T_1 \cup \cdots \cup T_n$, but disagree on some attribute of R and also on some distinct attribute of S. Thus $\pi_{T_1 \bowtie \cdots \bowtie T_n}(I)$ is a singleton, and

$$\pi_R(I) \bowtie \pi_S(I) \bowtie \pi_{T_1 \cup \cdots \cup T_n}(I)$$

is seen to have four members, while $\pi_{R \cup S \cup T_1 \cup \cdots \cup T_n}(I)$ has only two. Thus this join is lossy. Note the fact that R and S are disjoint, which follows from the condition of Corollary 1, is essential here.

Theorem 3. Let **R** be a set of relations. Then **R** has a lossy join if and only if either

(1) for some R and S in **R**, $R\ R \cap S \overset{*}{\not\rightarrow} R$, *and* $R \cap S \overset{*}{\not\rightarrow} S$, *or*

(2) there exist R, S, and $T_1, \ldots, T_n$ in **R** forming the forbidden subgraph of Fig. 2. That is, there are no arcs between R and S, there are no arcs from any T_i to R or S, and $\{T_1, \ldots, T_n\}$ is strongly connected.

Proof. (If). In case (1), $R \bowtie S$ is lossy by Corollary 1. In case (2) $R \bowtie S \bowtie T_1 \bowtie \cdots \bowtie T_n$ is lossy by Lemma 5.

(Only if). Suppose (1) and (2) are both false, yet **R** has a nontrivial lossy join. We may take this lossy join to be all of **R**, since if the join that is lossy happens to be a proper subset of **R**, we may replace **R** by this subset in the statement of the theorem. The fact that no examples of (1) or (2) exist in **R** implies that no example exists in the subset.

We claim that the relations in any nontrivial join have a graph that is *pseudo-connected,* meaning that the graph, with all arcs replaced by undirected edges, form a connected undirected graph. The proof is by induction on the number of relations joined.

Basis. If two relations are joined, the result follows from condition (1).

Induction. Suppose $R_1 \bowtie \cdots \bowtie R_n$ is joined with $S_1 \bowtie \cdots \bowtie S_m$. Then

$$(R_1 \cup \cdots \cup R_n) \cap (S_1 \cup \cdots \cup S_m) \neq \varnothing$$

so there is an arc between some R_i and some S_j.

It suffices therefore to show that, given (1) and (2), any set of relations with a pseudo-connected graph has a lossless join. We first prove:

Lemma 6. In any pseudo-connected graph for which no instance of (2) can be found, for any nodes R and S there exists a node T with a directed path from T to R and a directed path from T to S. (T may be R or S.)

Proof. The proof is by induction on the length of the shortest pseudo-path between R and S.

Basis. If the pseudo-path has length one, it is an arc and thus the result is immediate.

Induction. Suppose that the result is true for pseudo-paths shorter than i, and let $R, T_1, \ldots, T_{i-2}, S$ be a pseudo-path of length i. By the inductive hypothesis there is some node T' with directed paths to R and T_{i-2}. If there is an arc $T_{i-2} \Longrightarrow S$, then we are done. Therefore assume $S \Longrightarrow T_{i-2}$.

Let the directed path from T' to T_{i-2} be

$U_1 \Longrightarrow \cdots \Longrightarrow U_r$, where $T' = U_1$ and $T_{i-2} = U_r$. If for some i there is an arc from U_i to S, we may let $T = T'$, and we are done. Otherwise, we may prove by backwards induction on j that for some $k_j \geqslant j$, there is an edge $S \Longrightarrow U_{k_j}$ and $\{U_{k_j}, \ldots, U_j\}$ is an SCR.

Basis. If $j = r$, the result becomes immediate when we let $k_r = r$.

Induction. Let there be an edge $S \Longrightarrow U_{k_{j+1}}$ and let $\{U_{k_{j+1}}, \ldots, U_{j+1}\}$ be an SCR. The existence of $k_{j+1} \geqslant j+1$ follows from the inductive hypothesis. There is an edge $U_j \Longrightarrow U_{j+1}$, so we may apply Lemma 5 to SCR $\{U_{k_{j+1}}, \ldots, U_{j+1}\}$ and nodes U_j and S. As **R** is assumed to have no examples of (2), there must be an edge either from

i) $U_m \Longrightarrow S$ for some m, $j < m \leqslant k_{j+1}$,

ii) $U_m \Longrightarrow U_j$ for some m, $j < m \leqslant k_{j+1}$,

iii) $S \Longrightarrow U_j$, or

iv) $U_j \Longrightarrow S$.

Cases (i) and (iv) cannot occur, because we have assumed no arcs from any U_i to S. In case (ii) we have the inductive hypothesis, with $k_j = k_{j+1}$. In case (iii) we have the inductive hypothesis with $k_j = j$.

To complete the proof of Lemma 6, we have only to observe that in the case where there are no arcs $U_i \Longrightarrow S$, letting $j = 1$ shows that there is a directed path from S to T' to R. Hence T may be taken to be S.

Lemma 7. In any pseudo-connected graph for which no instance of (2) can be found, there is a node from which there is a directed path to each node.

Proof. If not, let R be a node that reaches a maximum number of nodes, and let S be a node not reached by a directed path from R. Then by Lemma 6 there is a node T with directed paths to R and S, contradicting the maximality of R.

Returning now to the proof of the theorem, we know by Lemma 7 that the relations of **R** may be ordered $S_1, S_2, \ldots, S_n$, where for each $i \geqslant 2$ there is some $j < i$ with an arc $S_j \Longrightarrow S_i$. Then by induction on i, the join $S_1 \bowtie \cdots \bowtie S_i$ is lossless.

Basis. The case $i = 2$ follows from Corollary 1, since there is an arc $S_1 \Longrightarrow S_2$.

Induction. Assume the inductive hypothesis for $i-1$. Then there is some $j < i$ for which an arc $S_j \Longrightarrow S_i$ exists. That is, $S_j \cap S_i \overset{*}{\rightarrow} S_i$, so surely $S_i \cap (S_1 \cup \cdots \cup S_{i-1}) \overset{*}{\rightarrow} S_i$. Hence by Corollary 1 the join $(S_1 \bowtie \cdots \bowtie S_{i-1}) \bowtie S_i$ is lossless. Letting $i = n$ proves the theorem.

Corollary 2. There is an $O(n^4)$ time algorithm to determine whether a set of relation schemes has a nontrivial lossy join, where n is the space needed to write the relation schemes, attributes and dependencies.

Proof. By Theorem 3 and the fact [3] that $X \overset{*}{\rightarrow} Y$ can be decided in linear time. □

Corollary 3. If $G(\mathbf{R})$ is a forest and the condition of Corollary 1 is satisfied for pairs of relations with a nonempty intersection, then all nontrivial joins are lossless. □

The database designs of [11] are an example where Corollary 3 applies.

5. Extensions to Multivalued Dependencies

Following [10,15], we say that there is a *multivalued dependency* of the set of attributes Y on the disjoint set of attributes X, written $X \twoheadrightarrow Y$, if for all instances I the following condition holds. Let Z be the set of all attributes not in X or Y. A tuple w in I can be viewed as the concatenation of its projections on X, Y and Z, which we denote $w = w[X]w[Y]w[Z]$. Let w_1 and w_2 be two tuples with the same X-component, $w_1 = w_1[X]w_1[Y]w_1[Z]$ and $w_2 = w_1[X]w_2[Y]w_2[Z]$. Then the "interchanged" tuples $w_1[X]w_1[Y]w_2[Z]$ and $w_1[X]w_2[Y]w_1[Z]$ are also in I. In simple words, when $w[X]$ is given, the Y-values that appear within w are independent of the value of any other attribute.

The algorithm of Section 3 generalizes to multivalued dependencies. However, when applied to functional dependencies, the algorithm identifies a_i's, and b_{ij}'s and rows in the table are eliminated. When applied to multivalued dependencies, rows are added to the table. The join is lossless if and only if a word beginning $a_1 a_2 \cdots a_n$ is added to the table. Unfortunately, this algorithm requires exponential time and space. The actual complexity of the lossless join problem in the presence of multivalued dependencies is substantially unknown.

We note that Corollary 1 holds for multivalued dependencies in a formulation stronger than that for functional dependencies [10,15]. The join $R \bowtie S$ is lossless if and only if $R \cap S \twoheadrightarrow R$ and $R \cap S \twoheadrightarrow S$. Further, a multivalued dependency can have the empty set $\varnothing$ as its left side. Thus the above statement holds even if $R \cap S = \varnothing$ (in which case $R \bowtie S$ is simply the Cartesian product). Turning to the problem of lossless joins of subsets, we see that we do not have to restrict our attention to nontrivial subsets of **R**. In the graph $G(\mathbf{R})$, if R and S are connected then they are connected by two arcs, one from R to S and one from S to R. Thus, $G(\mathbf{R})$ is essentially an undirected graph. Clearly, if **R** has no subset with a lossy join then $G(\mathbf{R})$ must be complete. It turns out that this is also a sufficient condition.

Theorem 4. Let **R** be a set of relation schemes. Then **R** has a subset with a lossy join if and only if for some R, S in **R**, $R \cap S \not\twoheadrightarrow R$, and $R \cap S \not\twoheadrightarrow S$. In other words, **R** has a subset with a lossy join if and only if **R** contains a pair with a lossy join.

Proof. Omitted in draft. □

6. On Decompositions

The decomposition theorems for relations with functional dependencies discussed in [6,7] all partition a relation R into a pair R_1 and R_2 such that $R_1 \cap R_2 \overset{*}{\rightarrow} R_2$. By Corollary 1, R can be reconstructed from R_1 and R_2. Our developments suggest that there may be useful decompositions into more than two relations that are not expressible as a cascade of decompositions into two. In particular, we offer the following:

Theorem 5. For all $n \geqslant 3$ there are sets of n relations whose join is lossless but such that no join of a proper

subset is lossless.

Proof. Omitted in draft. Example 1 with attribute G omitted shows the construction for $n = 3$.

References

[1] W. W. Armstrong, "Dependency Structures of Data Base Relationships," *Proc. IFIP 74,* pp. 580-583, North Holland.

[2] P. A. Bernstein, "Synthesizing Third Normal Form Relations from Functional Dependencies," *TODS,* 1:4 (Dec., 1976), pp. 277-298.

[3] P. A. Bernstein and C. Beeri, "An Algorithmic Approach to Normalization of Relational Database Schemas," Technical Report CSRG-73, Computer Science Research Group, University of Toronto, Sept., 1976.

[4] C. Beeri, R. Fagin and J. H. Howard, "A Complete Axiomatization for Functional and Multivalued Dependencies," RJ1977, IBM, San Jose, California, 1977.

[5] E. F. Codd, "A Relational Model for Large Shared Data Bases," *CACM,* 13:6 (June, 1970) pp. 377-387.

[6] E. F. Codd, "Further Normalization of the Data Base Relational Model," in *Data Base Systems* (R. Rustin, Ed.), pp. 33-64, Prentice Hall, Englewood Cliffs, N. J.

[7] E. F. Codd, "Recent Investigations in Relational Data Base Systems," *Proc. IFIP 74,* pp. 1017-1021, North Holland.

[8] C. Delobel, "Contributions Theoretiques a la Conception d'un Systeme d'informations," Ph. D. thesis, Univ. of Grenoble, Oct., 1973.

[9] C. Delobel and R. G. Casey, "Decomposition of a Data Base and the Theory of Boolean Switching Functions," *IBM J. of Res. and Dev.,* 17:5 (Sept., 1972), pp. 370-386.

[10] R. Fagin, "Multivalued Dependencies and a New Normal Form for Relational Databases," to appear in *TODS.*

[11] G. K. Manacher, "On the Feasibility of Implementing a Large Relational Data Base with Optimal Performance on a Minicomputer," *Proc. Intl. Conf. on Very Large Data Bases,* Framingham, Massachusetts, Sept., 1975.

[12] S. B. Guthery and D. M. O'Neill, "The Syntax and Semantics of Functional Dependency," unpublished memorandum, Bell Laboratories, Holmdel, New Jersey.

[13] J. Rissanen, "Independent Components of Relations," RJ1899, IBM, San Jose, California, 1977.

[14] M. Stonebraker, E. Wong, P. Kereps and G. Held, "The Design and Implementation of INGRES," *TODS,* 1:3 (Sept., 1976), pp. 189-222.

[15] C. Zaniolo, "Analysis and Design of Relational Schemata for Database Systems," Tech. Rept. UCLA-ENG-7769, Department of Computer Science, UCLA, July, 1976.