International Journal of Computer and Information Sciences, Vol. 10, No. 3, 1981

A Graphical Interpretation of Dependency Structures in Relational Data Bases

Sudhir K. Arora and K. C. Smith

Received July 1980; revised April 1981

The designer of a relational data base must use dependency structures of data to model semantic situations that arise in data. He must further ensure that these dependencies are not violated during operations on the data base. In this paper we study a subclass of dependencies, namely, root-dependencies and introduce a common graphical picture (S-diagram) for all of them. This effort offers a possible application of graph theory to the study of relational data bases. The S-diagram offers a pictorial insight to all the root-dependencies. We also discuss, briefly, other possible uses of our work such as automatic constraint checking and recovery of data in a damaged data base.

KEY WORDS: Data base design; hypergraphs; information systems; relational model.

1. INTRODUCTION

Before the advent of data base management systems, the role of data in a computer system was to serve as an input to the various programs. If data was to have any structure, it was defined in the program. This meant that each program had its own data to work with. However, it became increasingly clear that several programs used the same data and that data played a vital role in the functioning of an enterprise. Efficient handling and storage of data became a desirable goal. This very logically led to data base management systems.

In a data base management system, the data are structured and stored in the computer. The programmers do not impose their own structure on the data. They are constrained to use the data in the computer. Data are a resource of the enterprise which is available to several users and must be managed properly. As data became more important, considerable effort was devoted to studying the properties of data. An important outcome of this

0091-7036/81/0600-0187$03.00/0

study is the various dependency structures that may be found in data and may be usefully exploited to improve the performance of the data base.

Dependency structures have been studied using set theory,[1] propositional logic,[12] and Boolean algebra.[9] In this paper we study a subclass of dependencies, namely, root-dependencies using a theory of well-connected relations.[2] As a result a very neat graphical picture of root-dependencies emerges in the form of an S-diagram. It should be noted that the S-diagram is a hypergraph. Zaniolo[29] has also applied hypergraphs to the study of relational data bases. However, his study is not directed towards dependency structures.

In Section 2 we familiarize the reader with root-dependencies and the necessary portions of the theory of well-connected relations (WCR's). Before interpreting root-dependencies graphically (Sections 4 and 5), we introduce a graphical representation (S-diagram) of an instance of a relation (Section 3). Finally, in Section 4, we briefly discuss possible practical uses of our approach to the study of root-dependencies. In the Appendix, we present three properties of the natural joins which are used throughout this paper.

2. BACKGROUND

In this section, we first introduce the reader to dependencies in general and root-dependencies in particular. This paper is concerned with finding a graphical interpretation for all root-dependencies. In section 2.1, we take an example given in Ref. 26 and extend it to show the semantic differences btween various root-dependencies in the literature. We also include formal definitions for every root-dependency. In Section 2.2, we present some definitions and results (without proof) from the theory of WCR's.[2] We also indicate how the formalism of WCR's can be applied to binary as well as n-ary relations. In this paper we use this formalism to obtain the graphical interpretation of all root dependencies.

2.1. Root-Dependencies

Dependencies are a systematic way of dealing with semantically useful situations that arise in data. People think in terms of semantic connections among data rather than in terms of relational algebra which is highly mathematical. For example, in a company *each* employee may have a distinct employee number. The data base designer is faced with the rank of modeling this semantic connection between an employee and his employee number. He must further ensure that no two employees ever get the same employee number during operations on the data base. For this situation, we have the functional dependency (FD), see for example Refs. 1, 6, and 8. The

data base designer uses the FD between employee number and employee in his design process and also specifies it as an integrity constraint which must not be violated during operations on the data base.

In Ref. 26, the authors point out that FD is not adequate to model some semantic situations that arise in data. They give the following example. Suppose an employee works in several departments, then EMPLOYEE $\not\rightarrow$ DEPT (EMPLOYEE does not functionally determine DEPT), and suppose that *each* department has only one contract type, DEPT $\rightarrow$ CTYPE. Clearly, there is knowledge about employees and contract types on which they are working. But it cannot be expressed as an FD. Since Ref. 26 was published, other dependencies have been identified in the literature which capture or model the knowledge present in the above example. We can look at it in two different ways. If we wish to look at *each* employee and his relation to DEPT and CTYPE then it can be expressed by a mutual dependency (MUD),[19,20] or a contextual dependency (CD),[3] i.e., EMPLOYEE $\leftrightsquigarrow$ DEPT | CTYPE. The data base designer can now specify this MUD as an integrity constraint. However, if we wish to look at *each* department and its relation to EMPLOYEE and CTYPE then it can be expressed by a multivalued dependency (MVD),[11,29] i.e., DEPT $\rightarrow\!\!\!\rightarrow$ EMPLOYEE | CTYPE. We extend the example of Ref. 26 as follows. Suppose each contract type is divided into several portfolios and each employee of the department has access to all the portfolios in that department. We can model the knowledge about departments, employees, contract types, and portfolios by a hierarchical dependency (HD),[10] i.e., DEPT: EMPLOYEE | CTYPE | PFOLIO. Here we are looking at *each* department and its relation to EMPLOYEE, CTYPE, and PFOLIO.

So far we have seen FD, MVD, MUD, CD, and HD. All of them [MUD and CD are equivalent[21]] are semantically useful in modeling situations that arise in data. However, there are many other dependencies in the literature for which it is difficult to identify semantic situations in data. We do not deal with them in this paper, but mention them in passing—join dependency,[23] algebraic dependency,[28] transitive dependency,[22] subset dependency,[25] template dependency,[24] general dependency,[16] generalized mutual dependency,[18] boolean dependency,[11] and implication dependency.[13] The dependencies FD, MVD, MUD, CD, and HD are all special cases of JD. In all of these, as is demonstrated by the earlier discussion and underlining of the word "each," we are looking at every value of an attribute or attribute set and its relation to values of other attributes. We choose to call the former (i.e., left-hand side of these dependencies) the root-attribute. *Summing up, we can easily identify semantic situations in data which can be modeled by a dependency with a root-attribute. And it is difficult to identify a semantic situation in data which could usefully be*

modeled by a JD without a root-attribute. Hence, it is useful to introduce other dependencies which have a root-attribute and model semantic situations in data which cannot be modeled by the existing dependencies with a root-attribute. In Ref. 4 we introduce the mixed dependency (MD) and the co-dependency (COD) and show their usefulness.

We again extend the example of Ref. 26 as follows. Suppose EMPLOYEE, DEPT, and CTYPE have the same constraints as before, but in addition, an employee works on several projects and each project is assigned to one department. Clearly there is knowledge about PROJ, DEPT, CTYPE, and EMPLOYEE which cannot be modeled by FD, MVD, MUD, or HD. This knowledge can be modeled for *each* employee by a COD, i.e., EMPLOYEE $\equiv$ PROJ | DEPT | CTYPE. The MD is a special case of COD. If we modify this example and say that an employee can work only on one contract type at a time then we can model this knowledge for each employee by an MD, i.e., EMPLOYEE $\simeq$ PROJ | DEPT $\int$ CTYPE.

It can now be pointed out that an FD is a dependency with two sets of attributes (one root-attribute set on the left-hand side and another attribute set on the right-hand side which we choose to call a branch-attribute set). An MVD has three attribute sets—one root and two branch. An HD is an extension of MVD to 'n' attribute sets—one root and '$n-1$' branch. An MUD has three attribute sets— one root and two branch. A COD is an extension of MUD to 'n' attribute sets—one root and '$n-1$' branch. An MD is also defined on 'n' attribute sets—one root and '$n-1$' branch. However, it exhibits features of both MVD and MUD. *FD, MVD, MUD, HD, MD, and COD form a class of dependencies which we choose to call root-dependencies.*

We now give definitions for each root-dependency for the sake of completeness of this section. From these definitions, it should be noted that:

1. MVD is based on a linear 2-join;
2. MVD is based on a cyclic 3-join;
3. HD is based on a linear n-join;
4. MD is based on a combination of a linear n-join and 'p' cyclic 3-joins where $p < n$;
5. COD is based on a combination of a linear n-join and 'n' cyclic 3-joins.

For futher information on linear and cyclic joins the reader is referred to Ref. 7.

A functional dependency, $X \rightarrow Y_1$, exists in a relation, $R[X, Y_1]$, if in every instance of the relation some function f: $X \rightarrow Y_1$ exists.

In a relation, $R[X, Y_1, Y_2]$, the MVD, $X \rightarrow\!\!\rightarrow Y_1 \mid Y_2$ holds if and only

if $R[X, Y_1, Y_2]$ is the natural join of its projections, $R[X, Y_1]$ and $R[X, Y_2]$. Here X, Y_1, and Y_2 are disjoint sets of attributes.

In a relation, $R[X, Y_1, Y_2]$, the MUD, $X \rightsquigarrow Y_1 \mid Y_2$, holds if and only if $R[X, Y_1, Y_2]$ is the natural join of its three projections—$R[X, Y_1]$, $R[X, Y_2]$, and $R[Y_1, Y_2]$. In a relation, $R[X, Y_1, Y_2]$, the contextual dependency, $X \rightsquigarrow Y_1 \mid Y_2$ (X "determines" Y_1 in the context of Y_2), holds if and only if when $(x_i y_{11} y_{21})$, $(x_i y_{12} y_{22})$, and $(x_j y_{11} y_{22})$ are tuples in $R[X, Y_1, Y_2]$ then $(x_i y_{11} y_{22})$ is also a tuple, and this is true for all x_i and all instances of $R[X, Y_1, Y_2]$. Hence, X, Y_1, and Y_2 are mutually disjoint sets of attributes. Both of these dependencies are equivalent.

In a relation, $R[X, Y_1, Y_2, \ldots, Y_n]$, the HD, X: $Y_1 \mid Y_2 \mid \cdots \mid Y_n$, holds if $R[X, Y_1, Y_2, \ldots, Y_n] = R[X, Y_1] * R[X, Y_2] * \cdots * R[X, Y_n]$ where $*$ is the natural join. Here $X, Y_1, Y_2, \ldots, Y_n$ are mutually disjoint sets of attributes and $R[X, Y_i]$ are projections of $R[X, Y_1, Y_2, \ldots, Y_n]$.

The COD

$$X \equiv Y_1 \mid Y_2 \mid \cdots \mid Y_n$$

holds in a relation $R[X, Y_1, Y_2, \ldots, Y_n]$ if

$$R = R[X, Y_1] * R[X, Y_2] * \cdots * R[X, Y_n] * R[Y_1, Y_2] * R[Y_2, Y_3] * \cdots * R[Y_{n-1}, Y_n]$$

Here $X, Y_1, Y_2, \ldots, Y_n$ are mutually disjoint sets of attributes and each relation on the right-hand side is a projection of $R[X, Y_1, Y_2, \ldots, Y_n]$.

The MD

$$X \simeq Y_1 \int Y_2 \mid Y_3 \int Y_4 \mid Y_5 \mid \cdots \int Y_n$$

holds in a relation $R[X, Y_1, Y_2, \ldots, Y_n]$ if

$$R = R[X, Y_1] * R[X, Y_2] * \cdots * R[X, Y_n] * R[Y_1, Y_2] * R[Y_3, Y_4] * \cdots * R[Y_{n-1}, Y_n]$$

Here X, $Y_1, Y_2, \ldots, Y_n$ are mutually disjoint sets of attributes and each relation on the right-hand side is a projection of $R[X, Y_1, Y_2, \ldots, Y_n]$.

Finally, we mention in passing that FD's, MVD's, and MUD's have been defined for the case where the sets of attributes are not mutually disjoint. Also in Ref. 17, MVD's have been defined for relations with null values and in Ref. 5 they have been defined for joins other than natural, i.e., NMVD's and operator multivalued dependencies (OMVD's). We do not address the problem of trying to find graphical interpretations for these cases in this paper.

2.2 Well-Connected Relations

A theory of WCR's has been presented in Ref. 2. In this paper we present a graphical interpretation of all root-dependencies based on WCR's. For this purpose, it is sufficient for the reader to know only a few definitions and results from Ref. 2. These are presented in this section. WCR's are essentially a very restricted binary relation. In the context of a relational data base, they are a generalization of quotient relations.[14]

Definition 2.1.2. A *WCR* is a binary relation. W on two sets A and B such that

$$(\forall a)(a \in A)(\forall b)(b \in B)(aWb)$$

The sets A and B are called the first and the second *constituents* of the WCR.

An *elementary WCR (EWCR)* is a WCR in which the first constituent has a single element. The second constituent is then called the "*image set*" of the first constituent.

A *trivial WCR (TWCR)* is a WCR in which both the constituents have a single element.

Note 2.1.1

1. In this paper we do not consider partial binary relations. Also the mapping from set A to set B is always taken to be an onto mapping.

2. A binary relation is $R[A, B]$ while a WCR is $W[A; B]$. We also use $S_R[AB]$ and $S_R[B]$ (or $S_R[A]$) which mean the following:

$$\begin{aligned} S_R[AB] &= \{(a, b): (a \in A), (b \in B), (aRb)\} \\ &= S_R \\ S_R[B] &= \{(b): (b \in B) \text{ and } (\exists a)(a \in A)(aRb)\} \end{aligned}$$

Definition 2.1.3. A relation $R[A, B]$ can be expressed as

$$\begin{aligned} R[A, B] &= \sum_{i=1}^{n} R_i[A_i, B_i] \\ &= R_1[A_1, B_1] \cup R_2[A_2, B_2] \cup \cdots \cup R_n[A_n, B_n] \\ &= \pi(R) = \text{partition of } R \end{aligned}$$

where

$$R_i[A_i, B_i] \cap R_j[A_j, B_j] = \varnothing$$

for

$$i \neq j,\ 1 \leqslant i, j \leqslant n, \text{ and } A = \bigcup_{i=1}^{n} A_i \text{ and } B = \bigcup_{i=1}^{n} B_i$$

Definition 2.1.4. A partition of a binary relation $R[A, B]$ is a *canonical partition* if

$$R[A, B] = \sum_{i=1}^{n} W_i[A_i; B_i]$$

where

1. $W_i[A_i; B_i]$ is a WCR for $1 \leqslant i \leqslant n$.
2. A_i is a set with a single element for $1 \leqslant i \leqslant n$.
3. $A_i \neq A_j$ for $i \neq j$ and $1 \leqslant i, j \leqslant n$.

3. GRAPHICAL REPRESENTATION OF A RELATION

Before we can interpret root-dependencies graphically, we need a graphical representation of a relation. In this section we present a few definitions (3.1, 3.2, and 3.3) which lead to the S-diagram of an instance of a relation (Definition 3.4). We identify loops in the S-diagram which, in general, correspond to the tuples of the particular instance of the relation. This paper is concerned with a study of the properties of S-diagrams and their loops. In this section, we define four distinct properties of an S-diagram, namely

1. L-T condition;
2. C-condition;
3. total C-condition;
4. partial C-condition.

Since an S-diagram is a graphical representation of an instance of a relation, the above four properties are of necessity on the extension of a relation. Also properties (2), (3), and (4) involve the concept of a WCR. We show in Section 4 that the various root-dependencies can be interpreted in terms of these properties of the S-diagram and its loops.

Definition 3.1. In a relation, $R[X, Y_1, Y_2, \ldots, Y_n]$, we call $S_R[X]$ the *root-segment* and $S_R[Y_1]$, $S_R[Y_2]$,... the *branch-segments* (see Note 3.1). Throughout this paper, we assume $X, Y_1, Y_2, \ldots, Y_n$ to be mutually disjoint sets of attributes.

Definition 3.2. Any binary projection of $R[X, Y_1, Y_2,..., Y_n]$ on the root-segment and one of the branch-segments is called a *root-projection.* For example, $R[X, Y_1]$, $R[X, Y_2]$,... are root-projections.

Definition 3.3. Any binary projection of $R[X, Y_1, Y_2, Y_3,..., Y_n]$ on any two adjacent branch-segments is called a *branch-projection.* For example, $R[Y_1, Y_2]$, $R[Y_2, Y_3]$,... are branch-projections (see Note 3.1).

Note 3.1. The above definitions raise two questions. Firstly, are we implying an ordering among the attributes of a relation while in the relational model the ordering of attributes is immaterial? Secondly, how do we decide which attribute is the root-segment? In this section, we are going to introduce a graphical representation for any instance of a relation. This representation is easier to study if we draw it with the attributes in a certain order. For example, $R[X, Y_1, Y_2, Y_3]$ is the same relation as $R[X, Y_2, Y_1, Y_3]$; however, the graphical representation of the former may be more meaningful than that of the latter. In the rest of this paper we will assume that for the relation under consideration, the optimum ordering has been obtained. We do not address the problem of how to obtain this ordering in this paper. It was pointed out in an earlier section of this paper that the root-segment is that attribute or attribute set of a relation for which there is a semantic connection between *each* of its values and values of the other attribute sets. Hence, the choice of the root-segment is solely dependent on the semantic connection we wish to model or examine.

Definition 3.4. Any instance of an n-ary relation $R[X, Y_1, Y_2,..., Y_n]$, can be drawn graphically as follows:

1. Take the root-projections and the branch-projections of the relation.
2. Represent each projection as a mapping between the corresponding segments.
3. Draw all these mappings together in a single diagram.

We call this diagram an *S-diagram* of the relation. The *order* of an S-diagram is the number of segments in it. In Fig. 1, we show an example of an S-diagram of order 4 and the corresponding instance of the relation.

Definition 3.5. A *loop* is any closed path in an S-diagram of $R[X, Y_1, Y_2,..., Y_n]$ which starts at an element in the root-segment and passes once through each of the branch-segments, in the order specified, and then returns to the same element in the root-segment.

A loop in an S-diagram of order 2 is a simple *edge* while a loop in an S-diagram of order 1 is a simple *point.*

$$R[X,Y_1,Y_2,Y_3] = \begin{matrix} x_1 & y_{11} & y_{21} & y_{31} \\ x_1 & y_{12} & y_{21} & y_{32} \\ x_1 & y_{12} & y_{21} & y_{31} \end{matrix}$$

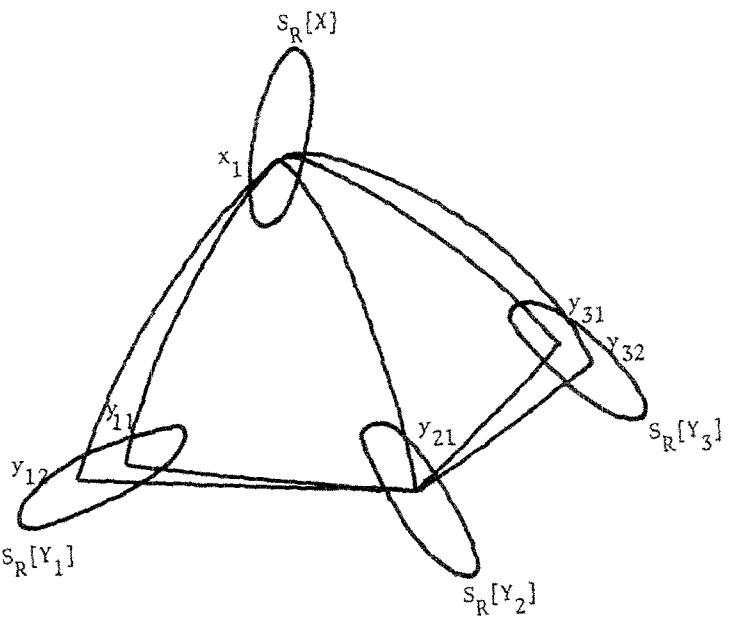


Fig. 1. S-diagram (order 4).

For any instance of $R[X, Y_1, Y_2, ..., Y_n]$, *it is trivially true that every tuple is a loop in the S-diagram. In S-diagrams of order less than* 3 *the converse of this is also true, i.e., every loop is a tuple. However, in S-diagrams of order* 3 *or more, every loop need not be a tuple. For example, in Fig.* 1, $(x_1 y_{11} y_{21})$ *is a loop, but there is no corresponding tuple in* $R[X, Y_1, Y_2]$.

Definition 3.6. The extension of a relation is said to satisfy the *loop–tuple condition* (*L-T condition*) if every loop in its S-diagrams is also a tuple in the corresponding instances.

We can redraw an S-diagram of a relation as follows:

1. Draw the canonical partition of each root-projection. Now each element in the root-segment will have an image set in each of the branch-segments.

2. The image sets, in adjacent branch-segments, of a particular element in the root segment are themselves connected by a binary subrelation called a *branch relation.* Hence, a branch-projection is made up of a set of branch-relations.

3. An S-diagram for order 3 is shown in Fig. 2. In general, the image sets in a branch-segment need not be disjoint. The diagrams show them to be disjoint only for the sake of clarity.

Definition 3.7. In the extension of a relation, $R[X, Y_1, Y_2, \ldots, Y_n]$, a branch-projection satisfies the *completeness condition* (*C-condition*) if every branch-relation in it is always a WCR.

Definition 3.8. The extension of a relation, $R[X, Y_1, Y_2, \ldots, Y_n]$, satisfies the *total C-condition* if every branch-relation in every branch-

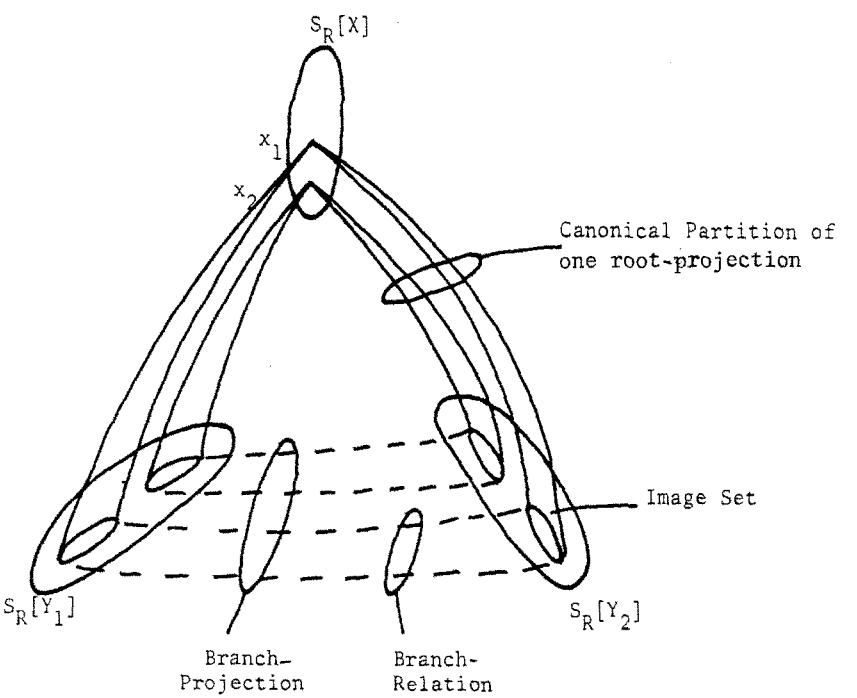


Fig. 2. Components of S-diagram.

projection is always a WCR. It satisfies the *partial C-condition* if every branch-relation in at least one of the branch-projections is always a WCR.

4. GRAPHICAL INTERPRETATION OF DEPENDENCIES

In Section 3, we have defined four properties of an S-diagram. In general, the S-diagrams of a relation need not satisfy these properties. However, if some of the root-dependencies hold in the relation then some of these properties are always satisfied. Again if some of these properties are always satisfied then some of the root-dependencies hold in the relation. These facts have been presented in this section as a series of propositions, leading to Theorem 4.1 which sums up the results. We have left out some proofs which can easily be constructed on lines, similar to other proofs in this section.

Proposition 4.1. In a relation $R[X, Y_1]$, if the $F \cdot DX \rightarrow Y_1$ holds then so does the L-T condition.

Proof. Trivially true.

Proposition 4.2. In a relation $R[X, Y_1, Y_2]$ the MUD $X \twoheadrightarrow Y_1 \mid Y_2$ holds if and only if the L-T condition is satisfied.

Proof. If

Assume $X \twoheadrightarrow Y_1 \mid Y_2$ in $R[X, Y_1, Y_2]$. It is trivially true that every tuple in instances of R is a loop in the S-diagrams. So we assume that every loop is not a tuple. Let (x, y_1, y_2) be a loop in the S-diagram and not a tuple in an instance of R. By definition of an MUD

$$R[X, Y_1, Y_2] = R[X, Y_1] * R[X, Y_2] * R[Y_1, Y_2]$$

Let

$$R_1[X, Y_1, Y_2] = R[X, Y_1] * R[X, Y_2]$$
$$R[X, Y_1, Y_2] = R_1 * R[Y_1, Y_2]$$

Now

$$(xy_1) \in R[X, Y_1] \tag{1}$$
$$(xy_2) \in R[X, Y_2] \tag{2}$$

and

$$(y_1 y_2) \in R[Y_1, Y_2] \tag{3}$$

from the definition of S-diagram.

Also in the join

$$R_1 * R[Y_1, Y_2]$$

no new tuples are added to R_1. Some tuples in R_1 whose projection over $[Y_1, Y_2]$ is not present in $R[Y_1, Y_2]$ are removed from R_1.

Again, from Eqs. (1) and (2), it follows

$$(xy_1y_2) \in R[X, Y_1] * R[X, Y_2] = R_1$$

But

$$(xy_1y_2) \notin R[X, Y_1, Y_2]$$
$$(y_1y_2) \notin R[Y_1, Y_2]$$

This is a contradiction to (3).

Only if

Let every loop in the S-diagram be a tuple. Assume

$$X \nrightarrow Y_1 \mid Y_2$$

i.e.,

$$R[X, Y_1, Y_2] \neq R[X, Y_1] * R[X, Y_2] * R[Y_1, Y_2] \tag{4}$$

Case 1. Let

$$(xy_1y_2) \in R[X, Y_1, Y_2] \notin \text{r.h.s. in Eq. (4)}$$

But from Theorem A.2 in the Appendix

$$(xy_1\, y_2) \in \text{r.h.s. in Eq. (4)—contradiction}$$

Case 2. Let

$$(xy_1y_2) \notin R[X, Y_1, Y_2] \in \text{r.h.s. in Eq. (4)}$$

Therefore

$$(xy_1) \in R[X, Y_1] \Rightarrow A \text{ path exists from } x \text{ to } y_1 \text{ in S-diagram}$$
$$(xy_2) \in R[X, Y_2] \Rightarrow A \text{ path exists from } x \text{ to } y_2 \text{ in S-diagram}$$

and

$$(y_1y_2) \in R[Y_1, Y_2] \Rightarrow A \text{ path exists from } y_1 \text{ to } y_2 \text{ in S-diagram}$$

Therefore

$$(xy_1y_2) \text{ is a loop in the S-diagram}$$

therefore

$$(xy_1y_2) \text{ is a tuple-contradiction}$$

therefore

$$R[X, Y_1, Y_2] = R[X, Y_1] * R[X, Y_2] * R[Y_1, Y_2]$$

Hence, the proposition is true.

Proposition 4.3. In a relation $R[X, Y_1, Y_2]$ the MVD

$$X \rightarrow\rightarrow Y_1 \mid Y_2$$

holds if and only if the L-T condition and the C-condition are satisfied.

Proof. The proof follows along the same lines as Proposition 4.2 and uses the fact that MVD is a special case of MUD.

Proposition 4.4. In a relation $R[X, Y_1, Y_2, \ldots, Y_n]$, if the L-T condition holds then so does the COD:

$$X \equiv Y_1 \mid Y_2 \mid \cdots \mid Y_n$$

Proof. Let all loops in any S-diagrambe tuples.
Assume, COD does not hold.
That is

$$R[X, Y_1, Y_2, \ldots, Y_n] \neq R[X, Y_1] * R[X, Y_2] * \cdots * R[X, Y_n] * R[Y_1, Y_2] * R[Y_2, Y_3] * \cdots * R[Y_{n-1}, Y_n] \quad (5)$$

Case 1. Let

$$(xy_1y_2 \cdots y_n) \in R[X, Y_1, Y_2, \ldots, Y_n] \notin \text{r.h.s. in (5)}$$

But by Theorem A.2 in the Appendix

$$(xy_1y_2 \cdots y_n) \in \text{r.h.s. in (5)—contradiction}$$

Case 2. Let

$$(xy_1y_2 \cdots y_n) \notin R[X, Y_1, Y_2, \ldots, Y_n] \in \text{r.h.s. in (5)}$$

therefore

$$(xy_1) \in R[X, Y_1] \Rightarrow A \text{ path exists from } x \text{ to } y_1 \text{ in S-diagram}$$
$$(y_1 y_2) \in R[Y_1, Y_2] \Rightarrow A \text{ path exists from } y_1 \text{ to } y_2 \text{ in S-diagram}$$
$$\vdots$$
$$(xy_n) \in R[X, Y_n] \Rightarrow A \text{ path exists from } y_n \text{ to } x \text{ in S-diagram}$$

therefore

$(xy_1 y_2 \cdots y_n)$ is a loop in the S-diagram

$(xy_1 y_2 \cdots y_n)$ is a tuple—contradiction

therefore

the COD,

$$X \equiv Y_1 \mid Y_2 \mid \cdots \mid Y_n \text{ holds}$$

Proposition 4.5. If in a relation, $R[X, Y_1, Y_2, \ldots, Y_n]$, the L-T condition and the total C-condition hold then so does the HD:

$$X: Y_1 \mid Y_2 \mid \cdots \mid Y_n$$

Proof. The proof follows along the same lines as Proposition 4.4 when we consider the fact that in (5) the joins with the branch-projections remove no tuples if the total *C*-condition holds.

Proposition 4.6. If in a relation, $R[X, Y_1, Y_2, \ldots, Y_n]$, the L-T condition and a partial C-condition hold then so does the appropriate MD:

$$X \simeq Y_1 \int Y_2 \mid \cdots \int Y_n$$

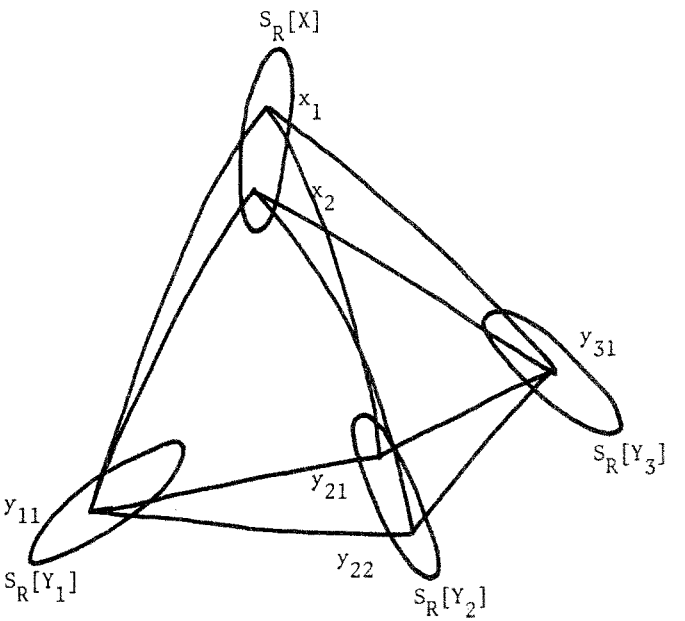
Proof. Follows along the same lines as Proposition 4.4 when we consider the fact that in (5) the joins with those branch-projections in which the C-condition is satisfied remove no tuples.

Proposition 4.4. If in a relation, $R[X, Y_1, Y_2, \ldots, Y_n]$, the HD

$$X: Y_1 \mid Y_2 \mid \cdots \mid Y_n$$

holds then the L-T condition need not be satisfied.

Proof. We give an example of a relation (Fig. 3) in which the HD holds, but the L-T condition does not.



$$R[X,Y_1,Y_2,Y_3] = \begin{matrix} x_1 & y_{11} & y_{21} & y_{31} \\ x_2 & y_{11} & y_{22} & y_{31} \end{matrix}$$

$X \equiv Y_1|Y_2|Y_3 \Rightarrow$ L-T condition.

$(x_1\ y_{11}\ y_{22}\ y_{31})$ is a loop but not a tuple.

Fig. 3. Violation of L-T condition.

Proposition 4.8. If in a relation, $R[X, Y_1, Y_2,..., Y_n]$, an MD or COD hold then the L-T condition need not be satisfied.

Proof. Obvious from Proposition 4.7 and the fact that HD is a special case of MD or COD.

Theorem 4.1. In a relation R the following statements are true. The left-hand side of each statement is a dependency in the intension of R while the right-hand side is a condition on the extension of R. The number in the middle is the order of the S-diagrams under consideration:

FD $\Leftrightarrow^2$ (L-T condition)

MVD $\Leftrightarrow^3$ (L-T condition) and (C-condition)

MUD $\Leftrightarrow^3$ (L-T condition)

HD $\Leftarrow^n$ (L-T condition) and (Total C-condition)

MD $\Leftarrow^n$ (L-T condition) and (Partial C-condition)

COD $\Leftarrow^n$ (L-T condition)

Proof. Follows from earlier propositions.

5. MODIFIED LOOPS

In Section 4, we presented a graphical interpretation of root-dependencies. The results of Section 4 are summed up by Theorem 4.1. In this theorem the implication statements for FD, HD, MD, and COD are unidirectional while for MVD and MUD they are bidirectional. Stronger, bidirectional implication statements can be proved in the case of HD, MD, and COD if we consider a subset of all the loops in the S-diagrams of a relation. We call this subset of loops—modified loops—and present these stronger results in this section. Again we have left out some proofs which can be constructed on the same lines as the proof in the case of COD (Proposition 5.3).

Definition 5.1. In a relation, $R[X, Y_1, Y_2, ..., Y_n]$, we define a *modified loop* as any path in an S-diagram which starts from an element x_i in the root-segment, passes in the order specified through those elements in the branch-segments which are in the image sets of x_i, and then returns to x_i.

Definition 5.2. The extension of a relation, $R[X, Y_1, Y_2, ..., Y_n]$, satisfies the *modified loop-tuple condition (ML-T condition)* if every modified loop in its S-diagrams is also a tuple in the corresponding instances.

Proposition 5.1. There exist relations which do not satisfy the ML-T condition.

Proof. We give an example of a relation in Fig. 4 which does not satisfy the ML-T condition.

Proposition 5.2. In a relation R the following statements are true. The left-hand side of each statement is a dependency in the intension of R while the right-hand side is a condition on the extension of R. The number in the middle is the order of the S-diagrams under consideration:

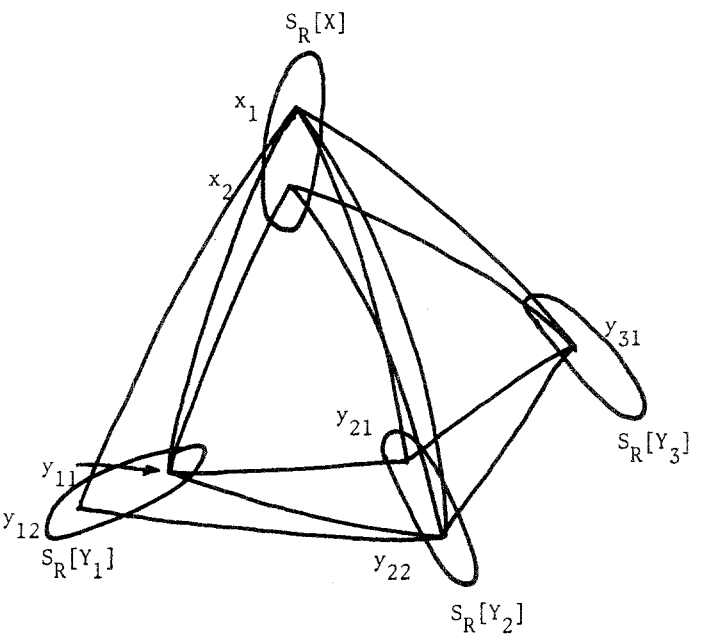
FG $\Rightarrow^2$ (ML-T condition)

MVD $\Leftrightarrow^3$ (ML-T condition) and (C-condition)

MUD $\Leftrightarrow^3$ (ML-T condition)

Proof. The result in the case of FD is trivial. For MVD and MUD it follows easily by showing that every loop is a modified loop also in their case.

Proposition 5.3. In a relation, $R[X, Y_1, Y_2, ..., Y_n]$, the COD $X \equiv Y_1 \mid Y_2 \mid \cdots \mid Y_n$ holds if and only if the ML-T condition is satisfied.



$$R[X,Y_1,Y_2,Y_3] = \begin{matrix} x_1 & y_{11} & y_{21} & y_{31} \\ x_1 & y_{12} & y_{22} & y_{31} \\ x_2 & y_{11} & y_{22} & y_{31} \end{matrix}$$

$(x_1\ y_{11}\ y_{22}\ y_{31})$ is a modified loop but not a tuple.

Fig. 4. Violation of ML-T condition.

Proof. If

Let the COD hold in $R[X, Y_1, Y_2,..., Y_n]$. That is,

$$R[X, Y_1, Y_2,..., Y_n] = R[X, Y_1] * R[X, Y_2] * \cdots * R[X, Y_n] * R[Y_1, Y_2] * R[Y_2, Y_3] * \cdots * R[Y_{n-1}, Y_n] \quad (6)$$

Assume the ML-T condition is not satisfied.

Case 1. Let the modified loop

$$(xy_1 y_2 \cdots y_n) \in R[X, Y_1, Y_2,..., Y_n] \notin \text{r.h.s. in (6)}$$

Because $(xy_1 y_2 \cdots y_n)$ is a modified loop

$$(xy_1) \in R[X, Y_1]$$
$$(xy_2) \in R[X, Y_2]$$
$$\vdots$$
$$(xy_n) \in R[X, Y_n]$$
$$(y_1 y_2) \in R[Y_1, Y_2]$$
$$\vdots$$

and

$$(y_{n-1} y_n) \in R[Y_{n-1}, Y_n]$$

therefore

$$(xy_1 y_2) \in R[X, Y_1] * R[X, Y_2] * R[Y_1, Y_2]$$
$$(xy_1 y_2 y_3) \in R[X, Y_1] * R[X, Y_2] * R[X, Y_3] * R[Y_1, Y_2] * R[Y_2, Y_3]$$
$$\vdots$$
$$(xy_1 y_2 \cdots y_n) \in \text{r.h.s. in (6)—contradiction}$$

Case 2. Let the modified loop

$$(xy_1 y_2 \cdots y_n) \notin R[X, Y_1, Y_2,..., Y_n] \in \text{r.h.s. in (6)}$$

This is impossible by Theorem A.2 in the Appendix.

Only if

Obvious from Theorem 4.1 and the fact that modified loops are a subset of the loops in an S-diagram.

Proposition 5.4. In a relation, $R[X, Y_1, Y_2,..., Y_n]$, the MD, $X \simeq Y_1 \mid Y_2 \wr Y_3 \mid \cdots \wr Y_n$, holds if and only if the ML-T condition and an appropriate partial C-condition are satisfied.

Proof. Follows along the same lines as Proposition 5.3 when we consider the fact that in (6) taking natural joins with the branch-projections in which the C-condition is satisfied removes no tuples.

Proposition 5.5. In a relation, $R[X, Y_1, Y_2,..., Y_n]$, the HD, X: $Y_1 \mid Y_2 \mid \cdots \mid Y_n$, holds if and only if the ML-T condition and the total C-condition are satisfied.

Proof. Follows along the same line as Proposition 5.3.

Theorem 5.1. In a relation R the following statements are true. The left-hand side of each statement is a dependency in the intension of R while the right-hand side is a condition on the extension of R. The number in the middle is the order of the S-diagrams under consideration:

FD $\Leftrightarrow^2$ (ML-T condition)

MVD $\Leftrightarrow^3$ (ML-T condition) and (C-condition)

MUD $\Leftrightarrow^3$ (ML-T condition)

HD $\Leftrightarrow^n$ (ML-T condition) and (total C-condition)

MG $\Leftrightarrow^n$ (ML-T condition) and (partial C-condition)

COD $\Leftrightarrow^n$ (ML-T condition)

Proof. Follows from earlier propositions.

6. UTILITY CONSIDERATIONS

In this paper we have demonstrated how WCR's can be used to give a graphical insight into the various root-dependencies in data. Root-dependencies are useful in modeling knowledge about the real world and also in preserving the integrity of the data base. We now discuss other possible uses of our approach to the study of root-dependencies. The discussion is brief and meant only to demonstrate the utility of our approach. We assume that derived relations are obtained only by projection and natural join operations. We do not address the wider problem of how dependencies, especially, HD, MD, and COD behave under other operations in relational algebra.

Firstly, we show a possible use in constraint checking to preserve integrity during operations on a derived relation. Consider the following set of base relations:

$$\begin{aligned}
\text{AGENT-LOC}[\underline{\text{AGENT \#}}, \text{TOWN}] &= a_1 \quad t_1 \\
&\quad\ a_2 \quad t_1 \\
&\quad\ a_2 \quad t_2 \\
\text{SALESMAN}[\underline{\text{SALESMAN \#}}, \underline{\text{AGENT \#}}] &= s_1 \quad a_1 \\
&\quad\ s_2 \quad a_2 \\
&\quad\ s_2 \quad a_1 \\
&\quad\ s_1 \quad a_2 \\
\text{PRODUCT}[\underline{\text{PRODUCT}}, \underline{\text{AGENT \#}}] &= p_1 \quad a_1 \\
&\quad\ p_1 \quad a_2 \\
&\quad\ p_2 \quad a_2 \\
\text{SALESMAN-AREA}[\underline{\text{SALESMAN \#}}, \text{TOWN}] &= s_1 \quad t_1 \\
&\quad\ s_2 \quad t_1 \\
&\quad\ s_2 \quad t_2
\end{aligned}$$

From these a user obtains his derived relation as follows:

DISTRIBUTION[AGENT #, TOWN, SALESMAN #, PRODUCT]
= AGENT-LOC * SALESMAN * PRODUCT * SALESMAN-AREA

a_1	t_1	s_1	p_1
a_1	t_1	s_2	p_1
a_2	t_1	s_2	p_1
a_2	t_1	s_1	p_1
a_2	t_2	s_2	p_1
a_2	t_1	s_2	p_2
a_2	t_1	s_1	p_2
a_2	t_2	s_3	p_2

In DISTRIBUTION, the following MD holds:

$$\text{AGENT \#} \simeq \text{TOWN} \int \text{SALESMAN \#} \mid \text{PRODUCT}$$

Now consider the following operations on this derived relation.

1. The user deletes the following tuple:

$$(a_2 \quad t_2 \quad s_2 \quad p_2)$$

What will be the effect of carrying out this command on the base relations as well? The new base relations will be

$$\begin{aligned}
\text{AGENT-LOC} = a_1 \quad & t_1 \\
a_2 \quad & t_2 \\
\text{SALESMAN} = s_1 \quad & a_1 \\
s_2 \quad & a_1 \\
s_1 \quad & a_2 \\
\text{PRODUCT} = p_1 \quad & a_1 \\
p_1 \quad & a_2 \\
\text{SALESMAN-AREA} = s_1 \quad & t_1 \\
s_2 \quad & t_1
\end{aligned}$$

And the derived relation should be

$$\begin{aligned}
\text{DISTRIBUTION} = a_1 \quad & t_1 \quad s_1 \quad p_1 \\
a_2 \quad & t_1 \quad s_1 \quad p_1 \\
a_1 \quad & t_1 \quad s_1 \quad p_1
\end{aligned}$$

to truly reflect the information contained in the base relations. Here we have actually carried out the command on the base relations and then recomputed the derived relation to proserve the integrity of the data base. To avoid modifying the base relations and to know the effect of the user command *before* executing it we can use the results in this paper. From Theorem 5.1 we know

$$\text{MD} \Leftrightarrow^{n} \text{(ML-T condition) and (partial C-condition)}$$

Deletion of tuple $(a_2\ t_2\ s_2\ p_2)$ and modification of base relations will mean that $\{t_2\}$, $\{s_2\}$, and $\{p_2\}$ will no longer be in the image sets of $\{a_2\}$. Hence, in the S-diagram of DISTRIBUTION any loop involving $\{t_2\}$ or $\{s_2\}$ or $\{p_2\}$ and $\{a_2\}$ will no longer be a modified loop. Hence, the corresponding tuples must be removed leaving us with a derived relation exactly as the one obtained by actually modifying the base relations and recomputing the derived relation. We check for partial C-condition and find that it holds true in the new relation. In passing we point out that in the S-diagram of the modified derived relation $(a_2\ t_1\ s_2\ p_1)$ is a loop, but not a modified loop and, hence, it is also not a tuple. Of the two approaches presented here which one requires lesser computation is unknown at the moment.

2. The user adds the following tuple

$$(a_2 \quad t_2 \quad s_1 \quad p_1)$$

Modifying the base relations and recomputing the derived relation indicates that another tuple $(a_2\ t_2\ s_1\ p_2)$ must also be added to preserve the integrity of the data base. Alternatively, the image sets of $\{a_2\}$ are $\{t_1, t_2\}$, $\{s_1, s_2\}$, and $\{p_1, p_2\}$. In the branch projection DISTRIBUTION[SALESMAN $\neq$, PRODUCT] every branch relation must be a WCR to preserve the partial C-condition. Hence $(a_2\ t_2\ s_1\ p_2)$ must also be added to the modified DISTRIBUTION. Again we do not know at this point which approach requires lesser computation.

3. If we take a projection of the derived relation, the MD still holds true as long as the root-attribute set is in the new relation (Theorem A.3 in the Appendix) and the new relation can be examined as in the earlier two examples.

For further information on modifying a data through derived relations the reader is referred to Refs. 15 and 27. We now show another possible use for our approach in the partial recovery of information in a damaged data base.

Consider a data base with a set of base relations and a set of derived relations. We know beforehand the type of dependency, HD, MD, or COD, that holds in each derived relation. Now if the data base is damaged (inadvertently or willfully) we are faced with the task of recovering lost data. Using results in this chapter we can examine each derived relation which yields a set of tuples (possibly empty) which must be added to preserve the dependency. Now the base relations must have had at least the tuples necessary to compute the new derived relations. This could be one of the ways to partially recover lost data.

7. CONCLUDING REMARKS

In this paper, we have studied the class of root-dependencies using a theory of WCR's. Both of these—the class of root dependencies and a theory of WCR's—have been presented in the literature. Our effort results in a graphical representation of an instance of a relation. We have called it the S-diagram of the relation. Several properties of the S-diagram have been defined, namely, the L-T condition, C-condition, partial C-condition, total C-condition, and ML-T condition. Based on these properties, we are able to get a graphical interpretation of all root-dependencies in the literature. This

approach, besides giving a pictorial insight to all the root-dependencies, can also prove useful in automatic constraint checking in a relational data base. Another possible use can be in the recovery of data in a damaged data base.

8. APPENDIX. PROPERTIES OF NATURAL JOINS

Theorem A.1. If $R[A]$ is a relation over the set of attributes A, then for every set of attributes, Y_1 and Y_2, contained in A,

$$R[Y_1] * R[Y_2] \supseteq R[Y_1, Y_2]$$

where $R[Y_1]$, $R[Y_2]$, and $R[Y_1, Y_2]$ are projections of $R[A]$ and $*$ is the natural join.

Proof. Trivial

Theorem A.2. If $R[A]$ is a relation over the set of attributes A, then for sets of attributes $Y_1, Y_2, \ldots, Y_n$ contained in A

$$R[Y_1] * R[Y_2] * \cdots * R[Y]_n \supseteq R[Y_1, Y_2, \ldots, Y_n]$$

where $R[Y_i]$ for $1 \leqslant i \leqslant n$ and $R[Y_1, Y_2, \ldots, Y_n]$ are projections of $R[A]$.

Proof. Let $t[A]$ be a tuple in $R[A]$.
Let $t[Y_i]$ be a projection of $t[A]$ on the attribute set Y_i. Therefore

$$t[Y_1] \in R[Y_1]$$
$$t[Y_2] \in R[Y_2]$$

and

$$t[Y_1, Y_2] \in R[Y_1, Y_2]$$

But,

$$R[Y_1, Y_2] \subseteq R[Y_1] * R[Y_2] \quad \text{(Theorem A.1)}$$

therefore

$$t[Y_1, Y_2] \in R[Y_1] * R[Y_2]$$

Again,

$$t[Y_1, Y_2, Y_3] \in R[Y_1, Y_2, Y_3]$$

and

$$R[Y_1, Y_2, Y_3] \subseteq R[Y_1, Y_2] * R[Y_3] \quad \text{(Theorem A.1)}$$

and

$$R[Y_1, Y_2] * R[Y_3] \subseteq R[Y_1] * R[Y_2] * R[Y_3] \quad \text{(Theorem A.1)}$$
$$t[Y_1, Y_2, Y_3] \in R[Y_1] * R[Y_2] * R[Y_3]$$

Similarly,

$$t[Y_1, Y_2, \ldots, Y_n] \in R[Y_1] * R[Y_2] * \cdots * R[Y_n]$$
$$R[Y_1] * R[Y_2] * \cdots * R[Y_n] \supseteq R[Y_1, Y_2, \ldots, Y_n]$$

Theorem A.3. If

$$R[X, Y_1, Y_2, \ldots, Y_n] = R[X, Y_1] * R[X, Y_2] * \cdots * R[X, Y_n] * R[Y_1, Y_2] * R[Y_2, Y_3] * \cdots * R[Y_{n-1}, Y_n]$$

then

$$R[X, Y_1', Y_2, \ldots, Y_n] = R[X, Y_1'] * R[X, Y_2] * \cdots * R[X, Y_n] * R[Y_1', Y_2] * R[Y_2, Y_3] * \cdots * R[Y_{n-1}, Y_n]$$

where

$$Y_1' \subseteq Y_1 \text{ and } X, Y_1, Y_2, \ldots, Y_n \text{ are mutually disjoint.}$$

Proof. Let

$$R[X, Y_1, Y_2, \ldots, Y_n] = R[X, Y_1] * R[X, Y_2] * \cdots * R[X, Y_n] * R[Y_1, Y_2] * R[Y_2, Y_3] * \cdots * R[Y_{n-1}, Y_n]$$

Assume,

$$R[X, Y_1', Y_2, \ldots, Y_n] \neq R[X, Y_1'] * R[X, Y_2] * \cdots * R[X, Y_n] * R[Y_1', Y_2] * R[Y_2, Y_3] * \cdots * R[Y_{n-1}, Y_n]$$

i.e.,

$$(xy_1') \in R[X, Y_1']$$
$$(xy_2) \in R[X, Y_2]$$
$$\vdots$$
$$(xy_n) \in R[X, Y_n]$$
$$(y_1' y_2) \in R[Y_1', Y_2]$$
$$(y_2 Y_3) \in R[Y_2, Y_3]$$
$$\vdots$$
$$(y_{n-1}, y_n) \in R[Y_{n-1}, Y_n]$$

and

$$(x, y_1', y_2, \ldots, y_n) \notin R[X, Y_1', Y_2, \ldots, Y_n]$$

But $R[X, Y_1', Y_2, \ldots, Y_n]$ is a projection of $R[X, Y_1, Y_2, \ldots, Y_n]$.

There is no y_1'' for which

$$(xy_1'' y_1' y_2 \cdots y_n) \in R[X, Y_1, Y_2, \ldots, Y_n]$$

Again

$$R[X, Y_1'] \text{ is a projection of } R[X, Y_1]$$

and

$$R[Y_1', Y_2] \text{ is a projection of } R[Y_1, Y_2]$$

Therefore there exists y_1'', such that

$$(xy_1'' y_1') \in R[X, Y_1]$$

and

$$(y_1'' y_1' y_2) \in R[Y_1, Y_2]$$

i.e.,

$$(xy_1'' y_1' y_2 \cdots y_n) \in R[X, Y_1, Y_2, \ldots, Y_n]\text{—contradiction}$$

Hence, the theorem is true.

REFERENCES

1. W. W. Armstrong, "Dependency Structures of Data Base Relationships," *Proc. of IFIP '74*, pp. 580–583 (1974).
2. S. K. Arora and K. C. Smith, "A Theory of Well Connected Relations," *J. of Information Sciences* **19**(2):97–134 (1979).
3. S. K. Arora and K. C. Smith, "A Dependency Theory and a 'New' Dependency for Relational Data Bases," *ACM Computer Science Conference* (February 1979).
4. S. K. Arora and K. C. Smith, "Well Connected Relations in Dependency Structures of Data Bases," Proc. of Conference 1979 CIPS/DPMA/FIQ, pp. 95–101 (June 1979).
5. S. K. Arora and K. C. Smith, "A Normal Form Based on Theta Join and Projection for Relational Data Bases," *IEEE COMPCON FALL*, pp. 295–300 (1980).
6. P. A. Bernstein, "Normalization and Functional Dependencies in the Relational Data Base Model," Ph.D. Thesis, Dept. of Computer Science, University of Toronto, (1975).
7. E. F. Codd, "A Relational Model of Data of Data for Large Shared Data Banks," *Comm. of the ACM* **13**(6):377–387 (une 1970).
8. E. F. Codd, "Further Normalization of the Data Base Relational Model," in *Courant Computer Science Symposium 6, Data Base Systems*, R. Rustin, Ed. (Prentice Hall, May 1971), pp. 33–64.
9. C. Delobel and R. G. Casey, "Decomposition of a Data Base and the Theory of Boolean Switching Functions," *IBM J. of Res. Develop.* **17**(5):374–386 (1973).
10. C. Delobel, "Normalization and Hierarchical Dependencies in the Relational Data Model," *ACM TODS* **3**(3):201–222 (1978).
11. R. Fagin, "Multivalued Dependencies and a New Normal Form for Relational Databases," *ACM TODS* **2**(3):262–278 (1977).
12. R. Fagin, "Functional Dependencies in a Relational Database and Propositional Logic," *IBM J. Res. Develop.* **21**(6):534–544 (1977).
13. R. Fagin, "Horn Clauses and Database Dependencies," *IBM Res. Rep. RJ2741*, (March 1980).
14. A. L. Furtado and L. Kershcberg, "An Algebra of Quotient Relations," *ACM SIGMOD*, pp. 1–8 (1977).
15. A. L. Furtado and K. C. Sevcik, "Permitting Updates Through User Views of Data Bases," *P.U.C.E. Tech. Rep.* (Dec. 1977).
16. D. Janssens and J. Paredaens, "General Dependencies," University of Antwerp, *Res. Rep.* 79-35, (December 1979).
17. Y. Lien, "Multivalued Dependencies with Null Values in Relational Data Bases," *Proc. of Fifth Inernational Conference on Very Large Data Bases*, pp. 61–66 (Oct. 1979).
18. A. O. Mendelzon and D. Mairer, "Generalized Mutual Dependencies and the Decomposition of Database Relations," *Proc. 5th. Int. Conf. on VLDB*, pp. 75–82 (October 1979).
19. J. M. Nicolas, "Mutual Dependencies and Some Results on Undecomposable Relations," *VLDB 78* (Sept. 13–15, 1978).
20. J. M. Nicolas, "Addendum and Erratum to 'Mutual Dependencies and Some Results on Undecomposable Relations'," Unpublished Report (August 1978).
21. J. M. Nicolas, Private Communication (Dec. 1978).
22. J. Paredaens and H. Gallaire, "Transitive Dependencies in a Database Scheme," *R.A.I.R.O. Informatique/Computer Science* **14**(2):149–163 (1980).
23. J. Rissanen, "Theory of Relations for Data Bases—A Tutotial Survey," *Proc. 7th Symp. on Math. Foundations of Computer Science, Lecture Notes in Computer Science* (Springer-Verlag, 1978), pp. 537–551.

24. F. Sadri and J. D. Ullman, "A Complete Axiomatization for a Large Class of Dependencies in Relational Databases," *Proc. 12th. Annual ACM Symp. on Theory of Computing* (April 1980).
25. Y. Sagiv and S. Walecka, "Subset Dependencies as a Alternative to Embedded Multivalued Dependencies," Dept. of Computer Science, University of Illinois at Urbana-Champaign, (UIUCDCS-R-79-980), (UILU-ENG 79 1732), (July 1979).
26. M. A. Schmid and J. R. Swenson, "On the Semantics of the Relational Data Model," *ACM SIGMOD International Conference on Management of Data* (May 1975), pp. 211–223.
27. K. C. Sevcik and A. L. Furtado, "Complete and Compatible Sets of Update Operations," *Proc. Int. Conf. on Data Base Management Systems* (June 1978), pp. 247–260.
28. M. Yannakakis and C. M. Papdimitriou, "Algebraic Dependencies," *M.I.T. Research Report* (1980).
29. C. Zaniolo, "Analysis and Design of Relational Schemata for Database Systems," Ph.D. Thesis, (UCLA-ENG-7669), (1976).