International Journal of Computer and Information Sciences, Vol. 10, No. 4, 1981

Graphical Normal Forms Based on Root Dependencies in Relational Data Base Systems

Sudhir K. Arora[1] and K. C. Smith[2]

Received April 1981; revised September 1981

Normal forms and dependencies are an area of great current interest in the design of relational data bases. Only a subclass, namely, root dependencies and the normal forms based on them, are of direct interest to the data base designer. Dependencies outside this subclass do not have clear cut semantics and may in the long run prove to be of theoretical interest only. We have proposed the fifth normal form (5NF) to control the pattern of codependancy, the highest known root dependency. We have also shown a strong parallel between root dependencies and their normal forms and a family of hypergraphs called S-diagrams. Graphical normal forms, based on S-diagrams have been proposed and their equivalence to conventional normal forms proved.

KEY WORDS: Relational data bases; semantics of relations; normal forms; dependencies; data base design.

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

This work was supported in part by the Science and Engineering Research Board Grant Number 214-7248.

[1] Department of Electrical and Computer Engineering, Wayne State University, Detroit, Michigan 48202.

[2] Department of Electrical Engineering, University of Toronto, Toronto, Ontario, Canada.

0091-7036/81/0800-0235$03.00/0 © 1981 Plenum Publishing Corporation

storage of data became a desirable goal. This very logically led to data base management systems.

In a data base management system, the data is 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 is 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 study the properties of data. An important outcome of this 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. These structures can be used to model semantic situation that arise in data. Further it is necessary to control the pattern of dependencies in data to avoid semantically undesirable effects when operations are carried out on the data base. Normalization is a step by step process designed to achieve this control of dependencies. Every step of the normalization process leads to a clear cut pattern of dependencies which is called a normal form of the data base.

So far we have not specified any particular type of data base relational, network, hierarchial etc. However all the work in the areas of dependencies and normal forms to be found in the literature is devoted to a relational data base. Extending this work to the other types of data bases is an open problem for research. Also, even though the historical reasons for studying dependencies and normal forms were to control the semantic behavior of a data base, in recent years (particularly last two years), many researchers have studied this area purely from a theoretical stand point. Hence, there are several dependencies and normal forms which have no semantic interpretation, and may prove to be of little interest to a data base designer.

We have isolated a class of dependencies, namely, root dependencies which have very clear semantics associated with them. It is our belief that a data base designer need consider only those normal forms which control the root dependencies. We have interpreted root dependencies and their normal forms in terms of a family of hypergraphs called S-diagrams. This study should prove useful in the design process of a data base; in applying graph theory to the study of data bases and also in enforcing integrity of a data base during operations. The utility of S-diagrams has been presented in detail in [ARSM 80b].

In Section 2 of this paper, we deal with dependencies in general and root dependencies in particular; in section 3 we present the normalization process and propose the highest normal form for root dependencies, and in section 4 we prove the equivalence of these normal forms to graphical normal forms based on S-diagrams. Finally in section 5 we offer some concluding remarks.

2. DEPENDENCIES

In this section, we first introduce the reader to dependencies in general and root dependencies[2] in particular. In section 2.1, we take an example given in[31] and extend it to show the semantic differences between various root dependencies in the literature. We also include formal definitions for every root dependency. It is our intention to convince the reader that only root dependencies have nice semantic interpretations. In section 2.2, we present some definitions and results (without proof) from the theory of well connected relations (WCR's).[5] We also indicate how the formalism of WCR's can be applied to obtain the graphical interpretation of all root dependencies.[2]

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 task of modeling this semantic connection between an employee and his employee number. He must futher 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.[1,8,10] 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,[31] 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[31] 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),[23,24] or a contextual dependency (CD),[6] 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),[16,34] i.e., DEPT $\to\to$ EMPLOYEE | CTYPE. We extend the example of[31] 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),[12] i.e. DEPT:EMPLOYEE | CTYPE | PFOLIO. Here we are looking at <u>each</u> 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)[25] are semantically useful in modeling situations that arise in data. However there are many other dependencies in the literature which do not have a root attribute and for which it is difficult to identify scemantic situations in data. We do not deal with them in this paper but mention them in passing–Join Dependency (JD),[28] Algebraic Dependency,[33] Transitive Dependency,[26] Subset Dependency,[30] Template Dependency,[29] General Dependency,[19] Generalized Mutual Dependency,[21] Boolean Dependency,[14] and Implication Dependency.[13] The dependencies, FD, MVD, MUD, CD, and HD are all special cases of JD. In all 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 i data which could usefully be modeled by a JD without a root attribute. Hence it is useful to introduce others 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[7] we introduce the mixed dependency (MD) and the codependency (COD) and show their usefulness.

We again extend the example of[31] 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 <u>each</u> 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 a 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. And 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. MUD 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 further information on linear and cyclic joins the reader is referred to.(9)

A functional dependency, $X \to Y_1$, exists in a relation, $R[X, Y_1]$, if in every instance of the relation some function f: $X \to Y_1$ exists. In a relation, $R[X, Y_1, Y_2]$, the MVD, $X \to \to Y_1 \mid Y_2$ holds if and only if, $R[X, Y_1, Y_2]$ is the natural joint 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, $X \leftrightsquigarrow Y_1 \mid Y_2$, 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 $R[X, Y_1, Y_2]$. Here, X, Y_1, and Y_2 are mutually disjoint sets of attributes. Both these dependencies are equivalent.

In 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 desjoint 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 $R[X, Y_1, Y_2,..., Y_n]$

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

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

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 $R[X, Y_1, Y_2,..., Y_n]$

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

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

Finally we mention in passing that FD's, MVD's, and MUD's have been defined for the case where the set of attributes are not mutually disjoint. Also in,(20) MVD's have been defined for relations with null values and in(3) they have been defined for joins other than natural, i.e. NMVD's and Operator Multivalued Dependencies (OMVD). We do not address these cases in this paper.

2.2. Graphical Interpretation of Root Dependencies

A theory of well connected relations (WCR's) has been presented in.(5) In this section we present a graphical interpretation of all root dependencies based on WCR's.(2) For this purpose, it is sufficient for the reader to know only a few definitions and results from.(5) These are presented in this section. We state the relevant theorems without proof. Well connected relations are essentially a generalization of quotient relations(17) in the context of relational data bases.

Definition 2.2.1

A binary relation $R[A, B]$ on the sets A and B are a mapping from set A to set B such that every element of A is mapped to at least one element of B. Such a binary relation is total.

The binary relation is partial if some elements of A are not mapped to any element of B.

Definition 2.2.2

A well connected relation (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 well connected relation (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 well connected relation (TWCR) is a WCR in which both the constituents have a single element.

Note 2.2.1

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 $S_R[B]$ (or $S_R[A]$) and $S_R[AB]$ which mean the following,

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

Definition 2.2.3

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

Definition 2.2.4

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 $1 \leqslant i, j \leqslant n$.

Definition 2.2.5

A partition of a binary relation, $R[A, B]$ is a prime partition if,

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

$$= \pi_p(R)$$

$$A = \sum_{i=1}^{n} A_i, \qquad B = \sum_{i=1}^{n} B_i$$

and

$$A_i \cap A_j = \varnothing, \qquad B_i \cap B_j = \varnothing$$

for

$$i \neq j, \quad 1 \leqslant i, \quad j \leqslant n.$$

Here each block of $\pi_p(R)$ is a WCR and $R[A, B]$ is called a Prime Relation. If $n = 1$ then $R[A, B]$ is a WCR and is trivially prime.

Definition 2.2.6

A functional relation, $R[A, B]$, is a prime relation in which the prime partition has only elementary WCR's. In such a case the prime partition is called a functional partition.

Before we can interpret root dependencies graphically, we need a graphical representation of a relation. We present a few definitions (2.2.7, 2.2.8, and 2.2.9) which lead to the S-diagram of an instance of a relation (Definition 2.2.10). We identify loops in the S-diagram which in general correspond to the tuples of the particular instance of the relation. We define four distinct properties of an S-diagrams, 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) above involve the concept of a WCR. The various root dependencies can be interpreted in terms of these properties of the S-diagram and its loops.

Definition 2.2.7

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

Definition 2.2.8

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

Definition 2.2.9

Any binary projection of $R[X, Y_1, Y_2, Y_3, \ldots, Y_n]$ on any two adjacent branch segments is called a branch projection. For example, $R[Y_1, Y_2]$, $R[Y_2, Y_3], \ldots$ are branch projections (see Note 2.2.1 below).

Definition 2.2.10

Any instance of an n-ary relation $R[X, Y_1, Y_2, \ldots, Y_n]$, 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 *S*-diagram of order 4.

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

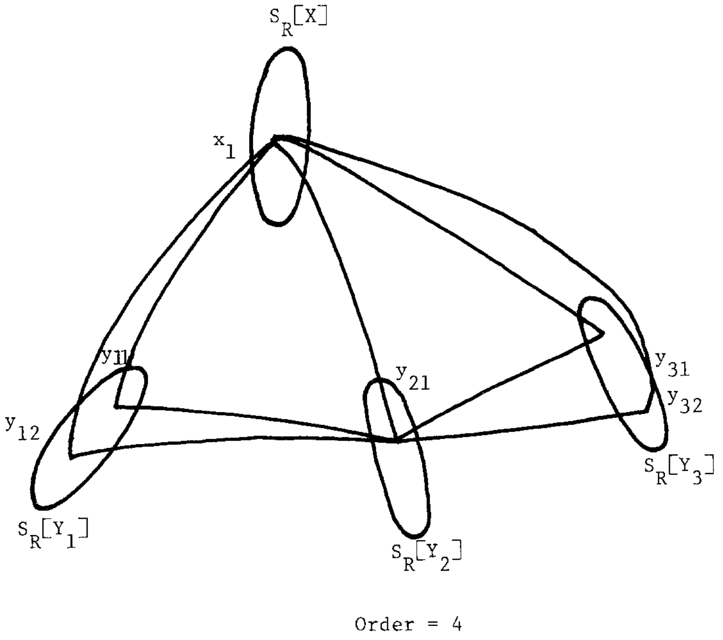

Fig. 1. *S*-Diagram (order 4).

Definition 2.2.11

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.

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 three the converse of this is also true, i.e., every loop is a tuple. However, in S-diagrams of order three 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 2.2.12

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 three 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 2.2.13

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

Definition 2.2.14

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

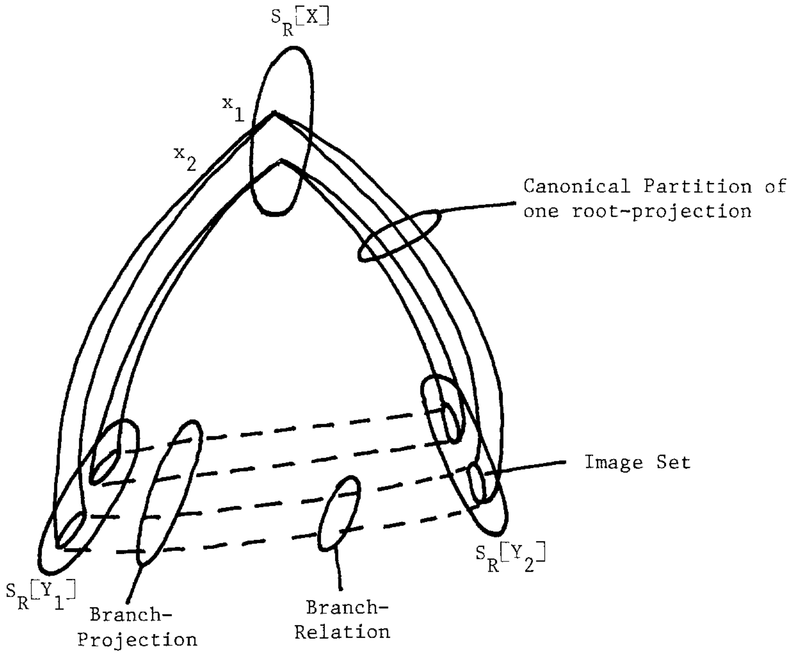

Fig. 2. Components of $\mathcal{S}$-Diagram (Order 3).

We have so far defined four properties of an $\mathcal{S}$-diagram. In general, the $\mathcal{S}$-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 are presented in the following theorem from.(2)

Theorem 2.2.1

In a relation R the following statement are ture. 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 $\mathcal{S}$-diagrams under consideration.

$$\text{FD} \overset{2}{\Rightarrow} (L\text{-}T \text{ condition})$$

$$\text{MVD} \overset{3}{\Leftrightarrow} (L\text{-}T \text{ condition}) \text{ and } (C\text{-condition})$$

$$\text{MUD} \overset{3}{\Leftrightarrow} (L\text{-}T \text{ condition})$$

$$\text{HD} \overset{n}{\Leftarrow} (L\text{-}T \text{ condition}) \text{ and } (\text{Total } C\text{-condition})$$

$$\text{MD} \overset{n}{\Leftarrow} (L\text{-}T \text{ condition}) \text{ and } (\text{Partial } C\text{-condition})$$

$$\text{COD} \overset{n}{\Leftarrow} (L\text{-}T \text{ condition}).$$

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.

Definition 2.2.15

In 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 t x_i.

Definition 2.2.16

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.

Theorem 2.2.2

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.

$$\text{FD} \overset{2}{\Rightarrow} (ML\text{-}T \text{ condition})$$

$$\text{MVD} \overset{3}{\Leftrightarrow} (ML\text{-}T \text{ condition}) \text{ and } (C\text{-condition})$$

$$\text{MUD} \overset{3}{\Leftrightarrow} (ML\text{-}T \text{ condition})$$

$$\text{HD} \overset{n}{\Leftrightarrow} (ML\text{-}T \text{ condition}) \text{ and } (\text{Total } C\text{-condition})$$

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

$$\text{COD} \overset{n}{\Leftrightarrow} (ML\text{-}T \text{ condition})$$

3. NORMALIZATION

Codd[10,11] proposed normalization for a relational data base for two main reasons: first it allowed the data base to be viewed as a collection of tables and second, it permitted the definition of a small class of primitive operators that were capable of manipulating relations to obtain all necessary logical connections among attributes. Further, normalization eliminated some undersirable effects,[10,18] in the relational schema when operations were carried out on them. These were insertion anomaly, deletion anomaly, and consistency on update. Another way to view normalization is as follows. We have pointed out in section 2 of this paper that dependencies are a systematic way to model knowledge and ensure integrity in a data base. Unless we control the pattern of dependencies in the relations of a data base semantically undersirable effects, anomalous behaviour, occur in the data base. Normalization process can be thought of as a way of controlling this pattern of dependencies. Here we do not distinguish between a relational, network or hierarchical data base, although the normal forms presented in the literature and in this paper are all concerned with the relational model. Extending the normalization process to other models is an open problem.

Another concept that is implicit in all the normal forms (so far there are 10 of them in the literature) is that of operations on the data base. Most normal forms (6 of them) are based on the operations, projection and natural join. One normal form is based on projection, natural join, and union; another is based on projection, natural join, union, and splitting; and a third is based on projection and theta-join. The 10 normal forms in the literature are, 1NF, 2NF, 2NF, BCNF, 4NF, PJ/NF, (3, 3)NF, PSJU/NF, DK/NF, and TP/BF.[9,10,11,16,14,32,15,3] Of these, INF controls only the format of the relations; 2NF, 3NF, and BCNF are based on FD and the operations projection, and natural join; 4NF is based on MVD and the operations projection, and natural join; PJ/NF is based on JD and the operations projection, and natural join; (3, 3)NF is based on FD and the operations projection, natural join and union; PSJU/NF is based on JD and the operations projection, splitting, natural join, and union; TP/NF is based on OMVD and the operations projection, and theta-join. PSJU/NF does not yet have a final definition. The author[14] has proposed a tentative definition and pointed out some problems with it. Finally DK/NF is the most general statement of a normal form.

3. A Normal Form Based on Codependency

It is interesting to study the various normal forms from a theoretical point of view. However in practice, a data base designer is faced with the task of madeling semantic situations and avoiding semantically undersirable side effects of operations. As we have pointed out in section 2 of this paper, the only dependencies worth considering from a semantic view point are the root dependencies. Hence it is our belief that the data base designer will be interested in only those normal forms that are based on root dependencies, namely, 1NF, 1NF, 3NF, BCNF, 4NF, (3, 3)NF, and TP/NF. In this paper we consider a subset of these normal forms, namely, those based on root dependencies and the operations projections and natural join. The normal forms in this category are 1NF, 2NF, 3NF, BCNF, and 4NF. The definitions of these normal forms are given below.

1. First Normal Form (1 NF):
   A relation is in 1 NF if every attribute in the relation is based on simple domains.
2. Second Normal Form (2 NF):
   A relation is in 2 NF, if it is in 1 NF and each nonprime attribute is fully dependent upon every key.
3. Third Normal Form (3 NF):
   A relation is in 3 NF if it is in 2 NF and nononprime attribute is transitively dependent on any key.
4. Boyce–Codd Normal Form (BCNF):
   A relation is in BCNF if it is in 1 NF and, for every set of attributes C, if any attribute, not in C, is functionally dependent on C then each and every attribute is functionally dependent on C.
5. Fourth Normal Form (4 NF):
   A relation is in 4 NF if, whenever a nontrivial MVD, $X \rightarrow \rightarrow Y$, holds then so does the FD, $X \rightarrow A$, for every attribute A.

The highest normal form, based on a root dependency and the operations projection and natural join, is the 4NF. It is based on MVD. The question that arises is, How do we control the pattern of other root dependencies, namely, MUD, HD, MD and COD? The dependencies, MUD, HD, MD, and COD could each have a distinct normal form associated with it. In[23,6] it has been pointed out that a new 5NF can be proposed for MUD or CD. The operations here would be projection and natural join. In this section we propose such a normal form: 5NF, which is based on COD. Since MUD, HD, and MD are special cases of COD, this normal form covers them as well. It should be pointed out that 5NF bridges the large gap between 4NF and PJ/NF. It is useful because it is the highest normal form for the root dependencies which are all semantically useful.

Definition 3.1.1

A relation, $R[Y_1, Y_2,\ldots, Y_n]$, is in 5NF when every nontrivial COD, i.e., a COD with at least two branch segments, in R contains a key of R in its root segment.

Theorem 3.1.1

In a relation, $R[Y_1, Y_2,\ldots, Y_n]$

$$5\text{NF} \Rightarrow 4\text{NF} \qquad \text{and} \qquad 4\text{NF} \not\Rightarrow 5\text{NF}.$$

Proof

$\underline{\Rightarrow}$

This follows easily when we consider that an MVD is a special case of a COD.

$\underline{\not\Rightarrow}$

If $4\text{NF} \Rightarrow 5\text{NF}$, then the instances of a relation in 4NF must all be in 5NF also.

We give an example o a relation in 4NF for which there exists an instance which is not in 5NF.

$$R[Y_1, Y_2, Y_3 Y_4] = \begin{array}{cccc} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_1 & c_1 & d_2 \\ a_2 & b_2 & c_2 & d_2 \end{array}$$

Here $Y_1 Y_2$ is a key. The MVD, $Y_1 Y_2 \rightarrow\rightarrow Y_3 \mid Y_4$ holds because $Y_1 Y_2$ is a key and R is in 4NF. But the COD, $Y_1 \equiv Y_2 \mid Y_3 \mid Y_4$ also holds and Y_1 is not a key. So R is not in 5NF.

4. GRAPHICAL NORMAL FORMS

So far all the normal forms we have considered have been defined on the intension of a relational data base. However in the theoretical study of data bases there is a trend to consider the extension of a relational data base.(22,27) The data base is thought of as a collection of all possible instances of its relations. The set of possible instances of a relation is a subset of the cartesian product of the domains of its attributes. The constraints on a relation (dependencies) reduce the membership of instances of a relation in the set of valid instances. Normalization is an important part of the theory of data bases. In this section we have taken normal forms, based on root dependencies and operations projection, and natural join, to the extension of a data base. We have proposed 2GNF (Second Graphical Normal Form), 3GNF, GBCNF, 4GNF, and 5GNF on the sets of instances of a relation. We have also proved the equivalence of these normal forms on the extension to the conventional normal forms on the intension of a relational data base. We use the word graphical to emphasize the fact that these normal forms are based on the S-diagrams of a relation. This effort should prove useful in furthering the theoretical study of data bases.

Consider a relation, $R[Y_1, Y_2, \ldots, Y_n]$ and the set U of all possible instances of this relation. If there were no constraints, (for example, FD's, MVD's, and so on) then this set U would be all subsets of the cartesian product of the domains of values associated with the attributes, $Y_1, Y_2, \ldots, Y_n$. However, in general we have some constraints and the set of valid instances of $R[Y_1, Y_2, \ldots, Y_n]$, we can derive instances of any projection of $R[Y_1, Y_2, \ldots, Y_n]$. Keeping this in mind, we can state the following definitions. These definitions are used for proving subsequent theorems. They are not necessary to the understanding of the statements of the theorems.

Definition 4.1

We define a set, $\mathbf{S}^*$, as the set of all possible S-diagrams of valid instances of a relation, $R[Y_1, Y_2, \ldots, Y_n]$, and its projections.

We define a set, $\mathbf{s}^*$, as the set of all possible S-diagrams of valid instances of a relation, $R[Y_1, Y_2, \ldots, Y_n]$.

Definition 4.2

We define a set, $\mathbf{S}_i^*$, of all possible S-diagrams of order i of valid instances of a relation, $R[Y_1, Y_2, \ldots, Y_n]$, and its projections. $\mathbf{S}_i^*$ includes the following:

1. S-diagrams obtained by taking all possible i mutually disjoint attribute sets from the attributes of R and its projections.
2. S-diagrams obtained by taking all permutations of the i mutually disjoint attribute sets.

Definition 4.3

We define a set, $\mathbf{s}_i^*$, of all possible S-diagrams of order i of valid instances of relation, $R[Y_1, Y_2, \ldots, Y_n]$. The set, $\mathbf{s}_i^*$, includes the following:

1. S-diagrams obtained by taking all possible i mutually disjoint attribute sets from the attributes of R.

2. S-diagrams obtained by taking all permutations of the i mutually disjoint attribute sets.

From the above definitions, the following are obvious:

1. $\mathbf{S}^* = \bigcup_{i=1}^{n} \mathbf{S}_i^*$ for a relation, $R[Y_1, Y_2, \ldots, Y_n]$.
2. $\mathbf{s}^* = \bigcup_{i=1}^{n} \mathbf{s}_i^*$ for relation, $R[Y_1, Y_2, \ldots, Y_n]$.

Definition 4.4

We define a set, $\mathbf{S}_2^*[P, N]$, for a relation, $R[Y_1, Y_2, \ldots, Y_n]$ as a subset of $\mathbf{S}_2^*$, which has only those S-diagrams of order two in which the root segment has only prime attributes and the branch segment has only nonprime attributes of R. Similarly, $\mathbf{S}_2^*[N, P]$ and $\mathbf{S}_2^*[N, N]$ have the obvious meanings.

Definition 4.5

We define a set $K = \{K_1, K_2, \ldots, K_l\}$ over a relation $R[Y_1, Y_2, Y_3, \ldots, Y_n]$ as the set of all keys.

Definition 4.6

A relation, $R[Y_1, Y_2, Y_3, \ldots, Y_n]$ is in 2GNF (Second Graphical Normal Form) if it is in 1NF and an element of $\mathbf{S}_2^*[P, N]$ without a functional partition exists if and only if the root segment does not contain a key of R.

Definition 4.7

A relation, $R[Y_1, Y_2, \ldots, Y_n]$, is in 3GNF if it is in 2GNF and elements of $\mathbf{S}_2^*[N, N]$ without a functional partition always exist for every possible root segment, branch segment permutation.

Definition 4.8

A relation, $R[Y_1, Y_2, \ldots, Y_n]$, is in GBCNF if it is in 1NF and an element of $\mathbf{S}_2^*$ without a functional partition exists if and only if the root segment does not contain a key of R.

Definition 4.9

A relation, $R[Y_1, Y_2, \ldots, Y_n]$, is in 4GNF when an element of $\mathbf{s}_3^*$, not satisfying the L-T condition and C-condition, exists if and only if the root segment does not contain a key of R.

Definition 4.10

A relation, $R[Y_1, Y_2,..., Y_n]$, is in 5GNF when an element of s_p^*, $3 \leqslant p \leqslant n$, not satisfying the *ML-T* condition exists if and only if the root segment does not contain a key of R.

We now propose to prove the equivalence of graphical normal forms on the extension of a relation to the conventional normal forms on the intension of a relation.

Theorem 4.1

In a relation, $R[Y_1, Y_2,..., Y_n]$

$$2\text{GNF} \Leftrightarrow 2\text{NF}$$

Proof

$\Rightarrow$

Let, $R[Y_1, Y_2,..., Y_n]$ be in 2GNF.

Assume it is not in 2NF.

i.e., there is a nonprime attribute of R which is partially dependent on some key.

Let this attribute be, Y_i and let the particular key be, k_l.

$$\therefore \qquad k_l \rightarrow y_i \qquad \text{for} \qquad k_l \subset K_l$$

and k_l is not a key.

Now consider the projection, $R[k_l, Y_i]$. All S-diagrams of its instances belong to $S_2^*[P, N]$ and have functional partitions because of Theorem 2.2.3. Here the root segment is k_l which is not a key. Hence $R[Y_1, Y_2,..., Y_n]$ is not in 2GNF. This is a contradiction.

$\Leftarrow$

Let, $R[Y_1, Y_2,..., Y_n]$ be in 2NF.

Assume it is not in 2GNF.

i.e., there is a projection, $R[A, B]$ whose instances have S-diagrams with functional partitions while the root segment, A contains only prime attributes but not a key.

Hence the FD,

$$A \rightarrow B \text{ holds in } R[A, B].$$

i.e., B is partially dependent on some key which contains A. Hence, $R[Y_1, Y_2, \ldots, Y_n]$ is not in 2NF. This is a contradiction.

Theorem 4.2

In a relation, $R[Y_1, Y_2, \ldots, Y_n]$,

$$3\text{GNF} \Leftrightarrow 3\text{NF}.$$

Proof

$\underline{\Rightarrow}$ Let $R[Y_1, Y_2, \ldots, Y_n]$ be in 3GNF.

Assume it is not in 3NF.

i.e., there is a nonprime attribute, Y_i, of R which is transitively dependent on some key, K_l through another attribute, Y_j.

$$\therefore \quad K_l \to Y_j, \quad Y_j \to Y_i, \quad K_l \to Y_i, \quad Y_j \nrightarrow K_l, \quad Y_i \nrightarrow K_l.$$

Now, Y_j is not a key ($\because$ $Y_j \nrightarrow K_l$).

Consider the projection $R[Y_j, Y_i]$. All S-diagrams of its instances have functional partitions ($\because$ Y_j, $\to Y_i$). Now, Y_j cannot be prime because then $R[Y_1, Y_2, \ldots, Y_n]$ is not in 2NF. ($\because$ Y_i would be partially dependent on a key). Hence, the S-diagrams of $R[Y_j, Y_i]$ belong to $\mathbf{S}_2^*[N, N]$. Hence, $R[Y_1, Y_2, \ldots, T_n]$ is not in 3GNF. This is a contradiction.

$\underline{\Leftarrow}$ Let $R[Y_1, Y_2, \ldots, Y_n]$ be in 3NF.

Assume it is not in 3GNF.

i.e., there is a projection, $R[A, B]$ whose instances have S-diagrams with functional partitions and A and B are nonprime. Hence, $A \to B$. Now, B is not a key because A is not a key and $A \to B$. Let, K_l be a key of R. So in $R[Y_1, Y_2, \ldots, Y_n]$.

$$K_l \to A, \quad A \to B, \quad K_l \to B, \quad A \nrightarrow K_l, \quad B \nrightarrow K_l.$$

Thus B is transitively dependent on K_l. Hence, $R[Y_1, Y_2, \ldots, Y_n]$ is not in 3NF. This is a contradiction.

Theorem 4.3

In a relation, $R[Y_1, Y_2,\ldots, Y_n]$.

$$\text{GBCNF} \Leftrightarrow \text{BCNF}.$$

Proof

$\underline{\Rightarrow}$

Let $R[Y_1, Y_2,\ldots, Y_n]$ be in GBCNF.

Assume it is not in BCNF.

i.e., there exists an FD, $A \rightarrow B$ in R where A is not a key of R. Consider the projection, $R[A, B]$. All S-diagrams of its instances will have functional partitions, and will belong to $\mathbf{S}_2^*$. Hence R is not in GBCNF. This is a contradiction.

$\underline{\Leftarrow}$

Let $R[Y_1, Y_2,\ldots, Y_n]$ be in BCNF.

Assume it is not in GBCNF.

i.e., there is a projection, $R[A, B]$ whose instances have S-diagrams with functional partitions and its root segment does not contain a key of R. Hence, $A \rightarrow B$ and A is not a key. Hence R is not in BCNF. This is a contradiction.

Theorem 4.4

In a relation, $R[Y_1, Y_2,\ldots, Y_n]$,

$$\text{4GNF} \Leftrightarrow \text{4NF}.$$

Proof

$\underline{\Rightarrow}$

Let $R[Y_1, Y_2,\ldots, Y_n]$ be in 4GNF.

Assume it is not in 4NF.

i.e., there exists an MVD, $A \rightarrow\rightarrow B \mid C$ in R such that A does not contain a key of R and both B and C are nonnull (i.e., the MVD is not trivial). Consider the S-diagrams of $R[A, B, C]$. From Theorem 2.2.1, these satisfy the L-T condition and the C-condition and, by definition of s_3^*, belong to s_3^*. Hence, R is not in 4GNF. This is a contradiction.

$\Leftarrow$

Let $R[Y_1, Y_2, \ldots, Y_n]$ be in 4NF.

Assume, it is not in 4GNF.

i.e., there is a partitioning of attributes of R, $R[A, B, C]$, whose instances have S-diagrams which satisfy the L-T condition and the C-condition, and its root segment does not contain a key. Let the root segment be A and the branch segments B and C. From Theorem 2.2.1, $A \to \to B \mid C$ holds in R. Hence R is not in 4NF. This is a contradiction.

Theorem 4.5

In a relation, $R[Y_1, Y_2, \ldots, Y_n]$,

$$5\text{GNF} \Leftrightarrow 5\text{NF}.$$

Proof

$\Rightarrow$

Let $R[Y_1, Y_2, \ldots, Y_n]$ be in 5GNF.

Assume it is not in 5NF.

i.e., a nontrivial COD, $A \equiv B_1 \mid B_2 \rfloor\!\mid B_p$, exists in R and the root segment, A, does not contain a key of R. From Theorem 2.2.2, the S-diagrams of $R[A, B_1, \ldots, B_p]$ satisfy the ML-T condition and belong to s_p^*. Hence R is not in 5GNF. This is a contradiction.

$\Leftarrow$

Let $R[Y_1, Y_2, \ldots, Y_n]$ be in 5NF.

Assume it is not in 5GNF.

i.e., there is a partitioning of attributes of R, $R[A, B_1, B_2, \ldots, B_p]$, whose instances have S-diagrams which satisfy the ML-T condition and the root segment, A, does not contain a key of R. From Theorem 2.2.2, the nontrivial COD, $A \equiv B_1 \mid B_2 \mid \cdots \mid B_p$ holds in R. Hence R is not in 5NF. This is a contradiction.

5. CONCLUDING REMARKS

Dependencies model semantic situations in data. The various dependencies in the literature are, Functional Dependency, Multivalued Dependency, Mutual Dependency, Hierarchical Dependency, Mixed Dependency, Codependency, Join Dependency, Algebraic Dependency, Transitive Dependency, Subset Dependency, Template Dependency, General Dependency, Generalized Mutual Dependency Boolean Dependency, and Implication Dependency. Of these, the first six form the class of root dependencies. It is our belief that a data base designer need consider only the root dependencies as they have clean cut semantic interpretations. We have shown that root dependencies have a very strong parallel in a family of hypergraphs, called, S-diagrams.

Normal forms are proposed to control the pattern of dependencies during operations on a data base. The various normal forms in the literature are 1NF, 2NF, 3NF, BCNF, 4NF, (3, 3)NF, TP/NF, PJ/NF, PSJU/NF, and DK/NF. Of these only the first seven are based on root dependencies and should prove useful to a data base designer. The 1NF controls only the format of the relations; the next four normal forms all involve the operations projection and natural join, and form a hierarchy. The highest normal form in this hierarchy is the 4NF which controls the pattern of multivalued dependencies. We have defined the 5NF which controls the pattern of codependencies, the highest known root dependency. The hierarchy of 2NF, 3NF, BCNF, 4NF, and 5NF has been proved to be equivalent to the graphical normal forms 2GNF, 3GNF, GBCNF, 4GNF and 5GNF, all based on S-diagrams.

REFERENCES

1. W. W. Armstrong, "Dependency Structures of Data Base Relationships," *Proc. of IFIP 74*, 580–583 (North Holland, New York, 1974).
2. S. K. Arora and K. C. Smith, "A Graphical Interpretation of Dependency Structures in Relational Data Bases," (Submitted for publication in the *International Journal of Computer and Information Sciences.*)
3. S. K. Arora and K. C. Smith, "A Normal Form Based on Theta-Join and Projection for Relational Data Bases," *IEEE COMPCON FALL*, **XI**:295–300 (Washington, D.C., 1980).
4. S. K. Arora and K. C. Smith, "Applications of S-diagrams to Relational Data Bases," in preparation.
5. S. K. Arora and K. C. Smith, "A Theory of Well Connected Relations," *J. of Information Sciences* **19**:97–134 (1979).
6. S. K. Arora and K. C. Smith, "A Dependency Theory and a 'New' Dependency for Relational Data Bases," *ACM Computer Science Conference* (Dated, 1979).

7. S. K. Arora and K. C. Smith, "Well Connected Relations in Dependency Structures of Data Bases," Conference of the Canadian Information Processing Society 95–101 (Quebec City, 1979).
8. P. A. Bernstein, "Normalization and Functional Dependencies in the Relational Data Base Model," (Ph.D. Thesis, University of Toronto, 1975).
9. E. F. Codd, "A Rational Model of Data for Large Shared Data Banks," *Comm. of the ACM.* **13** (6):377–387 (1970).
10. E. F. Codd, "Further Normalization of the Data Base Relational Model," in *Courant Computer Science Symposium: Data Base Systems,* Ed. R. Rustin, **6**:33–64 (Prentice Hall, New York 1971).
11. E. F. Codd, "Recent Investigations in Relational Data Base Systems," IFIP 74, (North-Holland Pub. Co., New York 1974).
12. C. Delobel, "Normalization and Hierarchical Dependencies in the Relational Data Model," *ACM TODS.* **3** (3):201–222 (1978).
13. R. Fagin, "Horn Clauses and Database Dependencies," *IBM Res. Report,* RJ2741 (1980).
14. R. Fagin, "Normal Forms and Relational Operators," *ACM* SIGMOD International Conference on Management of Data, 153–160 (Boston, 1979).
15. R. Fagin, "A Normal Form for Relational Data Bases that is Based on Domains and Keys," *IBM Res. Rep.,* RJ 2520 (32950) (1979).
16. R. Fagin, "Multivalued Dependencies and a New Normal Form for Relational Databases," *ACM TODS* **2** (3):262–278 (1977).
17. A. L. Furtado and L. Kerschberg, "An Algebra of Quotient Relations," *ACM* SIGMOD International Conference on Management of Data, 1–8 (Toronto, 1977).
18. I. J. Heath, "Unacceptable File Operations in a Relational Data Base," Proc. *ACM* SIGFIDET Workshop on Data Description, Access and Control, 19–33 (1971).
19. D. Janssens and J. Paredaens, "General Dependencies," University of Antwerp, Res. Rep. 79-35, (Dec. 1979).
20. Y. Lien, "Multivalued Dependencies with Null Values in Relational Databases," VLDB, 61–66 (Rio De Janeiro, Brazil, 1979).
21. A. O. Mendelzon and D. Mairer, "Generalized Mutual Dependencies and the Decomposition of Database Relations," VLDB, 75–82 (Brazil, 1979).
22. D. Mairer, A. O. Mendelzon, and Y. Sagiv, "Testing Implications of Data Dependencies," *Supplement ACM* SIGMOD International Conference on Management of Data, 20–28 (Boston, 1979).
23. J. M. Nicolas, "Mutual Dependencies and Some Results on Undecomposable Relations," VLDB 78, (Berlin, 1978).
24. J. M. Nicolas, "Addendum and Erratum to 'Mutual Dependencies and Some Results on Undecomposable Relations," Unpublished Report, ONERA, CERT, France (August 1978).
25. J. M. Nicolas, Private Communication (Dec. 1978).
26. J. Paredaens and H. Gallaire, "Transitive Dependencies in a Data base Scheme," *R.A.I.R.O. Informatique/Computer Science,* **14** (2):149–163 (1980).
27. J. Rissanen, "Independent Components of Relations," *ACM Trans. on Database Systems,* **2**:4, 317–325 (1977).
28. J. Rissanen, "Theory of Relations for Data Bases—A Tutorial Survey," Proc. 7th Symp. on Math. Foundations of Computer Science, *Lecture Notes in Computer Science,* **64**:537–551(Springer-Verlag, 1978).
29. F. Sadri and J. D. Ullman, "A Complete Axiomatization for a Large Class of Depen-

dencies in Relational Databases," Proc. 12th Annual ACM Symp. on Theory of Computing (1980).

30. Y. Sagiv and S. Walecka, "Subset Dependencies as an Alternative to Embedded Multivalued Dependencies," Dept. of Computer Sciences, University of Illinois at Urbana-Champaign, (UIUCDS-R-79-980), (UILU-ENG 79 1732), (July 1979).
31. M. A. Schmid and J. R. Swenson, "On the Semantics of the Relational Data Model," *ACM SIGMOD* International Conference on Management of Data, 211–223 (1875).
31. J. M. Smith, "A Normal Form for Abstract Syntax," *Proc. VLDB*, 156–162 (West Berlin, 1978).
32. M. Yannakakis and C. M. Papadimitriou, "Algebraic Dependecies," *M.I.T. Research Report*, (1980).
33. C. Zaniolo, "Analysis and Design of Relational Schemata for Database Systems," (Ph.D. Thesis, UCLA-ENG-7669, 1976).