A Study of the Dependency between Domains on the Relational Data Base

Hiroshi ARISAWA*

A B S T R A C T

This paper proposes a new view of the domain and the file in the Relational Data base. As the TNF (Third Normal Form) does not keep complete information of original data structure, it is not always convenient for the casual user to retrieve data. This defect is removed by the use of the domain-oriented data base model. In the model, the dependencies between domains are represented by the lattice and its tabular form, Global Table.

1. Introduction

Recently, the computer system managed large scale and various information. The concept "data base" is now stressed in many fields such as File Management, Information Retrieval, Question Answering and Artificial Intelligence. This term has, however, different sense in each field [1)-8)]. In File Management, the data base is usually divided into several files. A file is a set of records which have a certain structure. The main interests here are to reduce redundancy, to keep high efficiency in the retrieval and to update files or records easily. On the other hand, in QA systems and AI languages, data base usually consists of one file, which contains various types of records in the jumble. Some systems save the simple predicates as the records [5)], and some can save inference rules or semantics [6)]. The most important, in the fields, is to improve the inferential ability and efficiency in the retrieval.

The data base is to be shared by a variety of software like above in the near future. The basic theory of general purpose data base is, however, not completely established now. Most of the studies have been done only in the viewpoint of practical business. The CODASYL report [1),2)], for instance, concentrates

---

This paper first appeared in Japanese in Joho－Shori (Journal of the Information Processing Society of Japan), Vol. 17, No. 11 (1976), pp. 1026～1032.

* Faculty of Engineering, Yokohama National University

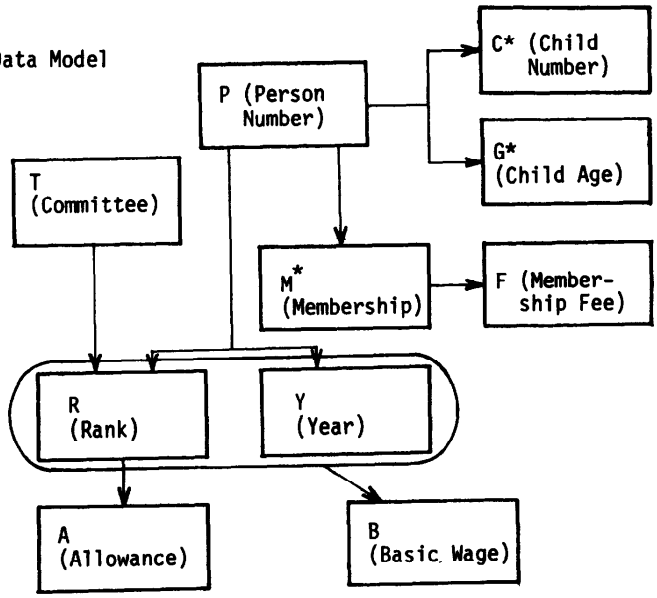
on the classification and arrangement of functions in the actual data base system.

The Relational Data Model by E. F. Codd has the remarkable feature to consider the general purpose data base on the theoretical side. In this paper, several problems of the Relational Model about semantic description are pointed out. Then the lattice is introduced to settle these problems mathematically.

2. Problems of the Relational Data Model

Codd proposed three levels of normalization and insisted the ideal data base is a set of TNF (Third Normal Form) files in his work [8),9)]. The user cannot define the data structure explicitly, but the TNF file schema reflects the semantics of the object world to some extent. A simple example is shown in Fig. 1. Here the attributes are represented as a capital letter. The "*" means repeating group.

In fig. 1(a), the arrow shows the semantic determination. The arrow from (Y, R) to B means, for example, "the basic wage is determined by his rank and the year of his addmission". The root of the arrow is corresponding to the primary key. However, this correspondence is not realized when the key contains repeating group. In the MEMBER relation in fig. 1(b), for example, the



(a) Semantic structure

PERSON [P, Y, R]

P	Y	R
1	55	President
2	60	Director
3	65	Manager
4	65	Salesman
5	70	Salesman
6	70	Salesman

MEMBER [P, M]

P	M
1	Golf
1	Tour
1	Music
2	Golf
2	Swimming
4	Tour

CHILD [P, C, G]

P	C	G
2	1	12
2	2	8
3	1	8
4	1	9
4	2	9

MEMBER FEE [M, F]

M	F
Golf	3.0
Tour	1.0
Music	0.5
Swimming	0.5

BASIC WAGE [Y, R, B]

Y	R	B
55	President	20
60	Director	16
65	Manager	12
65	Salesman	10
70	Salesman	8

ALLOWANCE [R, A]

R	A
President	10
Director	7
Manager	5
Salesman	2

(b) TNF File Schema and Occurrence

Fig. 1 Sample Data Base

semantics of "P determines M" cannot be read. The interpretation of the CHILD relation will be more complicated. In addition to the above, the TNF schema cannot be used in the key-breaking [9] structure and repeating key occurrence, such as the T attribute in fig. 1(a).

3. The Lattice Representation of the Dependencies

There are two ways to introduce semantics to the relational file schema. One is to divide files into smaller subfiles by the semantics [13]. The other is to use a certain diagram which represents the total system of dependencies. The lattice description method is a latter case.

3.1 The Lattice Description

The dependency is generally represented as follows:

$$D_{p1}D_{p2}\cdots D_{pn} \rightarrow D_{qi} \quad (1 \leq i \leq m).$$

It is satisfied automatically if D_{qi} is contained in $D_{p1}\cdots D_{pm}$. We call it the *trivial dependency*. All of the trivial dependencies in the data base constitute a lattice adding zero element. The lattice of P, R, Y, T and B in fig. 1 is shown in Fig. 2 using fine lines. On the lattice, each element is located to a certain

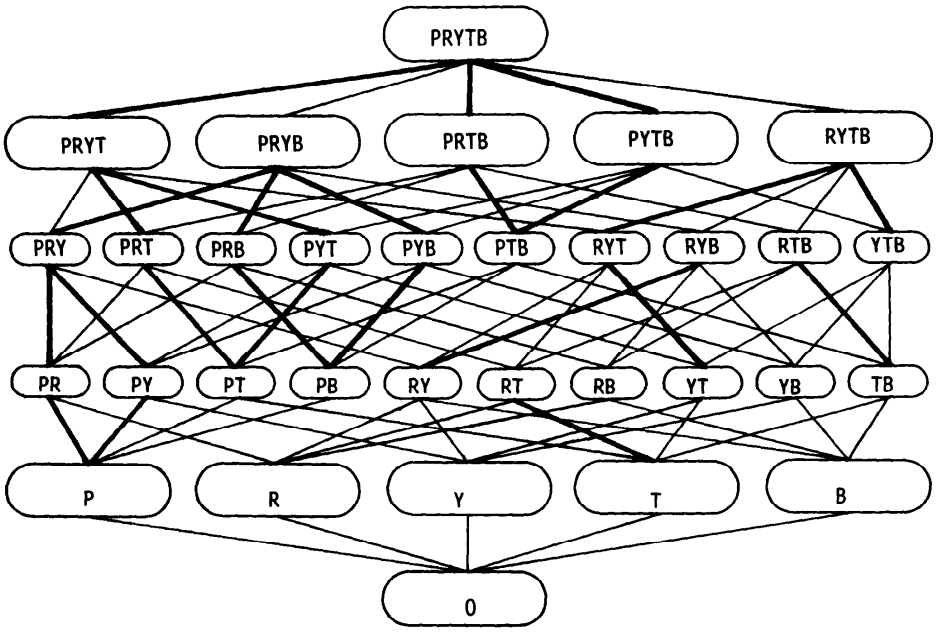


Fig. 2 An Example of Lattice Representation

level, and the lower level elements *depend on* the higher elements which are combined with lines. If we add non-trivial dependencies to the lattice, the levels of some elements are shifted. If (R, Y) → B is added, for example, RY is shifted to RYB: if an element (such as B) *depends on* RYB, then the element *depends on* RY. In this example, the shift occurs on RYB, RYT and PRYT. All of the shifts according to P → R, P → Y, T → R and (Y, R) → B are shown with heavy lines in fig. 2.

The lattice is useful to test the existence of a dependency. For instance, the existence of the dependency (Y, T) → B is directly displayed in fig. 2 because YT is shifted to RYTB.

3.2 The Global Table of the Lattice

In the computer software, the lattice of dependencies can be maintained as a bit table, namely the *Global Table*. The Global Table of the lattice in fig. 2 is shown in Fig. 3. The element of the lattice is expressed as a binary number of 5 figures in the left side column. The dependencies are displayed by the bit-on on each column P, R, Y, T and B. The row of an element identifies the number of the highest element which is equivalent to the original element with respect to dependency. We call it the *lattice value* of the element.

The generation algorithm of the Global Table is as follows:

Step 1: Make bit-on on all points corresponding to the trivial dependencies.

Step 2: Add dependencies: if you add

$A_1A_2 \cdots A_k \rightarrow B$,

make bit-on on the B-column of the element which contains $A_1A_2 \cdots A_k$.

Step 3: For all element z in the Global Table, update the lattice value of z as follows.

S 3.1: X ← z (X is an element variable).

S 3.2: If the lattice value of X is equal to X, then update the lattice value of z to X and end; else continue.

S 3.3: X ← the lattice value of X, jump to S 3.2.

		P	R	Y	T	B
10000	P	1	(1)	(1)		(1)
01000	R		1			
00100	Y			1		
00010	T		(1)		1	
00001	B					1
11000	PR	1	1	(1)		(1)
10100	PY	1	(1)	1		(1)
10010	PT	1	(1)	(1)	1	(1)
10001	PB	1	(1)	(1)		1
01100	RY		1	1		(1)
01010	RT		1		1	
01001	RB		1			1
00110	YT		(1)	1	1	(1)
00101	YB			1		1
00011	TB		(1)		1	1
11100	PRY	1	1	1		(1)
11010	PRT	1	1	(1)	1	
11001	PRB	1	1	(1)		1
10110	PYT	1	(1)	1	1	
10101	PYB	1	(1)	1		1
10011	PTB	1	(1)	(1)	1	1
01110	RYT		1	1	1	(1)
01101	RYB		1	1		1
01011	RTB		1		1	1
00111	YTB		(1)	1	1	1
11110	PRYT	1	1	1	1	(1)
11101	PRYB	1	1	1		1
11011	PRTB	1	1	(1)	1	1
10111	PYTB	1	(1)	1	1	1
01111	RYTB		1	1	1	1
11111	PRYTB	1	1	1	1	1

Fig.3 An Example of Global Table

In fig. 3, ① displays the bit-on in the Step 2, ⚠ in the Step 3. The Global Table keeps the same information as the lattice representation. In the above example, (Y, T) → B is directly displayed by the bit-on on the B-column of YT. The above algorithm is effective and faster than the method using Boolean functions [12].

4. Conclusion

By means of the Global Table, the user or the system can find out subordinate attributes for any combinations of the attributes easily. As the lattice representation is independent to the actual file construction or the restriction of TNF, the user can define the dependency network freely according to his semantics. It will be convenient for the casual users because they usually access only a part of the large data base.

Reference

1) Data Base Task Group Report to CODASYL Programming Language Committee, ACM, (April, 1971).
2) Feature Analysis of Generalized Data Base Management Systems, ACM, (May, 1971).
3) I. Kobayashi: A Formalizm of Information and Information Processing Structure, the SOKEN KIYO, Vol. 5, No. 1, (1975).
4) CODASYL Development Comittee: An Information Algebra: Phase I Report, Comm. of the ACM Vol. 5, No. 4, (April, 1962).
5) J. L. Kuhns: Answering Questions by Computer: A Logical Study, RAND Corp. RM-5428-PR, (1967).
6) T. Winograd: Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, MAC TR-84, M.I.T. (Jan. 1971).
7) E. F. Codd: A relational Model of Data for Large Shared Data Banks, Comm. of the ACM, (June, 1970).
8) E. F. Codd: Normalized Data Base Structure, A Brief Tutorial, ACM-SIGFIDET, (1971).
9) E. F. Codd: Further Normalization of the Data Base Relational Model, Courant Computer Science Symposia 6, (1971).
10) E. F. Codd: Relational Completeness of Data Base Sublanguages, Courant Computer Science Symposia, 6, (1971).
11) E. F. Codd: A Data Base Sublanguage Founded on the Relational Calculus, ACM-SIGFIDET, (1971).
12) C. Delobel and R. G. Casey: Decomposition of Data Base and the Theory of Boolean Switching Functions, IBM Journal Research Develop. (Sept, 1973).
13) H. A. Schmid and J. R. Swenson: On the Semantics of the Relational Data Model, Proc. ACM-SIGMOD, (1974).