論　文

リレーショナル・データベース上の領域間の決定関係についての考察*

有　澤　　博**

Abstract

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 use of the domain-oriented data base model. In the model, the determinisity between domains is stressed. It is represented by the lattice and its tabular form, Global Table. The Global Table is useful to examine the existence of a certain determinisity or to find out essential determinisity among domains.

1. はじめに

近年，計算機システムが大型化し，その取り扱う情報が多量で多様なものになるにつれて，データベースの重要性が強調されるようになった．現在データベースという用語は，ファイル・マネジメント・システムはもちろん，各種の情報検索システム，QA (Question-Answering) システム及び人工知能向き言語にまで広く用いられており，これらの間では少しずつ違った意味を持たされている[1)~9)]．たとえばファイル・マネジメント・システムでは，データベースとは一定の構造をもったデータ集合(レコード)の集合であり，いくつかのファイルに分割されている．利用者の要求は，あらかじめ定義されているデータ構造に従ってファイルへアクセスする操作に翻訳される．ここでは冗長性が少なく，検索効率の高いファイル作成，ファイル更新の容易さ，機密保護や安全性等が問題にされる．一方QA システムや人工知能向き言語では，データベースは1個のファイルから成り，この中にいろいろな形式のレコードを雑居させている．この場合のレコードとは，一般的な事実 (fact) を述語 (predicate) の形で記述したもので，簡単な述語のみを許すもの[6)]の他に，手続的な処理規則や，いくつかの事実から別の事実への推論規則を含み得るもの[7)]もある．利用者の質問や指令は，ある論理式の真偽を判定したり，論理式を満足させる事実の集合を抜き出したりする操作に変換される．従ってこの演算能力の程度と効率を高めることが課題になる．

今後，データベースは上記の諸分野をはじめ，複雑，大規模な情報処理には必要不可欠なものになるといわれている．その場合のデータベースは，システムの違いや利用目的によらず共通にアクセスされ，いろいろなレベルの処理要求に対処できなければならない．しかしそのようなデータベースの形態が一般的にどうあるべきかについて，基礎理論的な研究は非常に少ない，今までデータベースの研究は，きわめて実務的，現実的な面からのみ行われてきた．たとえば CODASYL の報告書では，現存するファイル・マネジメント・システムの機能の分類整理と，既存システムとの互換性を持たせた拡張に注意が向けられている[1),2)]．

ところが最近，QA や情報検索向きの演算が可能で，しかも既存のファイル・マネジメント・システムにも応用できるモデルとして，E. F. Codd のリレーショナル・データベースが注目され，それぞれの立場から批判や議論が活発に行われるようになった[3)]．Codd モデルはデータの表現や蓄積に n 項関係 (n-ary relation) を用いており，汎用性という点からも，基礎理論を考える上で非常に有益な性質を備えている．本稿

* On the determinisity between domains in the relational data base by Hiroshi ARISAWA (Yokohama National University)

** 横浜国立大学工学部情報工学科

では，はじめに Codd モデルの性質を分析し，次に実際のデータベースにこのモデルを採用する場合，データ蓄積に関していくつか問題があることを指摘する．最後に問題点の解決策として，領域間に存在する決定性を数学的に束によって取り扱う方法を述べる．

2. Codd モデルの問題点

2.1 データベース・モデル

はじめに本稿で用いる用語の意味について述べる．データベースに蓄積されるデータの最小単位は一つの数値や単語（文字列）である．これをアトム（atom）とよぶ．アトムには，物の名前であるとか，長さであるというように一定の意味付けがされており，これをアトムの属性（attribute）とよぶ．ある属性に関してのアトムの値の集合を，その属性の領域（domain）という．逆に領域の側からみて各アトムを具体値（occurrence）という．アトムの組み合わせで事象を表すことができる．たとえば「人名」属性のアトム1個と，「趣味」属性のアトム数個の組で「個人の趣味」が表される．事象に対応するアトムの組をレコードとよぶ．レコードは，ひとまとまりのデータとして利用者がアクセスできる単位にもなる．レコードに用いられている属性の組み合わせをレコード型という．

ところで，データベース・システムが取り扱うデータは複雑な構造をもっていることが多く，レコードとしてどういう型を用意すればよいか一意に決めることはできない．またレコード間にも構造をもたせる必要がある．既存のデータベース・システムでは，そのためにレコード集合を，レコード型によってファイルに分割している．CODASYL の報告[2]では，レコードを集団（group）と項目（item）の集りとし，集団の入れ子状の定義と，繰返し集団（repeating group）によって，任意の深さと広さを持ったデータ構造を1レコード内に定義できる．またセット（set）によってレコード間構造を作ることができる．このような方法は，データの意味や性質に基づいて，データへのアクセス・パスを詳細に定義し，それに従ってデータ蓄積を行うことを意味している．検索要求が一定のメニューに従ったもののみの場合には，この方法は強力であり，能率もよい．しかし利用法を限定せず，いろいろなアクセス方法が考えられる場合や，Codd の ALPHA 言語[12]のような高度の検索オペレーションを実行したい場合には，データの意味論に強く依存した構造は，かえって不便である．

これに対して，データベースのモデル化の1方法として，レコードを各属性によって張られる情報空間内の点としてとらえ，ファイルを点集合，データ操作を集合間の演算と考える方法がある[4),5)]．しかしこの場合も，何を1区切りの事象と考え，事象間の連想（association）をどう定義するかについて設計者の主観が入り，完全に一意にファイルの型式を決めることはできない．

Codd のリレーショナル・データベースは，これまでの考え方とは異なり，領域の n 項関係によってファイルを構成する．n 項関係は次のように定義される．

定義 $D_1, \cdots, D_n$ を領域とする．

$$R(D_1, \cdots, D_n) = \{(d_1, \cdots, d_n) : d_1 \in D_1 \wedge \cdots \wedge d_n \in D_n \wedge r(d_1, \cdots, d_n)\}$$

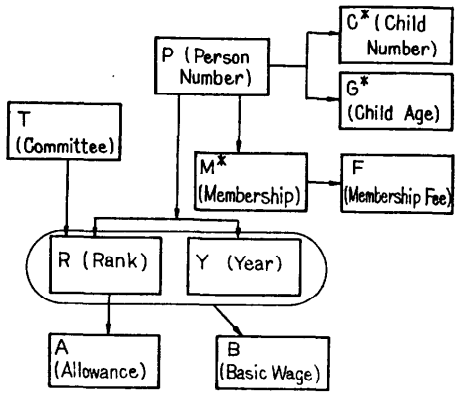
で定義される集合を $D_1, \cdots, D_n$ 上の n 項関係とよぶ．ただし r は，$D_1 \times D_2 \times \cdots \times D_n$ 上のブール関数である．

Codd モデルでは R をファイルに対応させ，R に含まれる属性の種類と順序の記述をファイル・スキーマ（file schema）とよぶ．また各ファイル中の一つの $d_1, \cdots, d_n$ の組（tuple）をレコードに対応させている．そのため各レコードは固定した形式となり，複雑な内部構造を持たせたり，アクセス・パスを陽に定義したりすることができない．このことは必要なデータの能率的なアクセスという面では多少不利である．しかし論理的に簡明なファイルとレコードの構成は，述語論理に基づく検索論理を組み立てること，それを手順的な集合演算に翻訳することが容易であり，QA システムや人工知能向き言語との融和がしやすいという特徴がある．ここでも n 項関係を基本にしたファイルが，もとの意味的な構造とどのように対応するかが重要な問題になるが，この点については次節で具体的に述べる．

2.2 Codd モデルの問題点

本稿では属性名を大文字の英字1字で表し，属性 A, B, C がファイルを構成している場合，$[A, B, C]$ のように表すことにする．ファイルに名前をつけたときは [ ] の前にその綴りを書く．

Codd はファイル・スキーマの記述に3段階の正規化（normalization）の手続きを提唱した[9),10)]．Codd のデータベース・モデルは第3正規形（Third Normal Form, TNF）のファイル集合で構成される．このモデルではデータ構造を陽に定義できないが，TNF のファイル・スキーマは自然に，ある程度データの意味的



(a) Data Structure

- Unnormalized
  PERSON [*P*, SALARY [*Y*, *R*, *B*, *A*], CHILDREN [*C*, *G*], MEMBERSHIP [*M*, *F*]]
- First Nomal Form
  SALARY [*P*, *Y*, *R*, *B*, *A*], CHILD [*P*, *C*, *G*], MEMBERSHIP [*P*, *M*, *F*]

(b) Normalization of File Schema

PERSON	[*P*,	*Y*,	*R*]
	1	55	President
	2	60	Director
	3	65	Manager
	4	65	Salesman
	5	70	Salesman
	6	70	Salesman

MEMBER	[*P*,	*M*]
	1	Golf
	1	Tour
	1	Music
	2	Golf
	2	Swimming
	4	Tour

CHILD	[*P*,	*C*,	*G*]
FEE	2	1	12
	2	2	8
	3	1	8
	4	1	9
	4	2	5
	5	1	3

MEMBER	[*M*,	*F*]
	Golf	3.0
	Tour	1.0
	Music	0.5
	Swimming	0.5

BASIC	[*Y*,	*R*,	*B*]
WAGE	55	President	20
	60	Director	16
	65	Manager	12
	65	Salesman	10
	70	Salesman	8

ALLOWANCE	[*R*,	*A*]
	President	10
	Director	7
	Manager	5
	Salesman	2

(C) Third Normal Form File Schema and Occurrence Values

Fig. 1 Sample Data Base

構造を反映したものになる．**Fig. 1** に例を示した．Fig. 1 (a) はデータの意味的な決定関係を示したもので，意味的構造に類似のものである．たとえば (Y, R) から B への矢は入社年とランクによって基本給を決定する，という意味付けに対応している．ここで＊のついている属性は繰り返し集団の性格をもつことを示している．Fig. 1 (b) は (a) の構造のうち T を除く部分を正規化前，第1正規形でそれぞれ表したものである．Fig. 1 (c) は同じものを TNF のファイル・スキーマで表し，具体値の例を付したものである．この図で下線がひいてある属性の組は，その具体値を指定すると，レコード内の他の属性の具体値が決定できることを示している．たとえば $[\underline{Y, R}, B]$ では，Y と R の具体値を指定すれば B の具体値は一意に決まる．このとき，$Y, R \rightarrow B$ と書き，Y, R は $[Y, R, B]$ のキー (key) 領域であるという．また Y, R は B に対して決定性をもつという．キー領域はレコードの識別子である．TNF のファイルでは，キー領域が他の領域を決定する以外には決定性が存在しない*．

Fig. 1 の例からも明らかなように，TNF ファイルはレコードの集りが簡明で冗長性の少ない表形式になるうえ，決定性が，もとのデータの意味的な決定関係とほぼ対応している．またデータベースに対する好ましくないオペレーションを事前に避けられるとされている．しかし Codd モデルで実際のデータベースを構成すると，次のような問題点がある．

第1に決定性が多数存在するファイル・スキーマの場合，これを TNF に分解すると，ファイル・スキーマの数が非常に多くなる．TNF への分解方法は1種ではなく，また分解の結果，同じ属性がいろいろなスキーマ中に現れたり，意味的に関係の深い属性が別々のファイルに分散したりする．そのためユーザは，あらかじめファイル・スキーマの全体を調べて検索の方針を立てなければならず，1回の簡単な検索を行うにも多くのファイルにアクセスしなくてはならない．

第2に属性間の決定性は，一般的にはデータの意味的構造をかなり忠実に反映しており，TNF のファイル・スキーマから，もとの構造を読みとることも困難でないが，繰り返し性をもつデータに対しては必ずしもそうではない．これを Fig. 1 の例で示そう．$[\underline{P}, \underline{M}]$, $[\underline{Y, R}, B]$, $[\underline{P, C}, G]$ の3つのファイルを比較する．これらはキーとして2つの属性の組をもつが，その意味が異なっている．まず $[\underline{Y, R}, B]$ では，Y と R の間には何の関係もなく，2つが組になって B を決定するに過ぎない．$[\underline{P, M}]$ では，M は P に従属する繰返し集団である．$[\underline{P, C}, G]$ では，C が P に従属する繰り返し集団であり，かつ P と C を指定することによって G の具体値が決定される．このようにキーの意味が異なっていると，データ構造が見にく

* Codd は「A が B に対して決定性をもつ」というかわりに，逆に「B が A に対して依存性 (dependency) をもつ」と定義している．TNF は依存性を使って，次のように一般的に定義される[10]．

関係 R が第1正規形であって，かつ R の属性の集まり C をどのようにとっても，「C に含まれていない属性が C に依存するならば，R 中のすべての属性が C に依存する」が満たされるとき，R は TNF であるという．

COMMITTE	[T,	R]
	C_1, C_2, C_3	President
	C_2, C_3	Director
	C_3, C_4	Manager
	C_3	Manager
	C_5	Salesman

Fig. 2 Example of Repeating key

くなる．

第3に TNF で直接表せないデータが存在する．一つは Codd 自身が指摘した[10]キー破壊的 (key-breaking) な依存性の場合である．たとえば [$\underline{Y, R}, B$] で，$B \to R$ かつ $B \nrightarrow Y$ と仮定すると，もはや TNF ではないが，どのように分解しても TNF のファイル・スキーマにはならない．いま一つはキー自身が繰り返し性をもつ場合である．たとえば Fig. 1 で，T はランクの決定機関であるとする．ここで，数種の機関の組が一つのランクを決めることもあるし，1機関が2種以上のランク決定に携わることもあるとすると，TNF 形式では表せない．強いて表形式にした例を **Fig. 2** に示した．このような例は繰り返し集団がキーになっている場合に，いつでも起り得るものである．

3. 領域間の決定性の束による取り扱い

今までに述べたように，TNF でファイルを構成する場合，データの意味的構造によって，一意的にファイルの形を決めることはできない．またいったん定義したファイルが，いろいろな検索，操作上の要求に対して，必ずしもいつも妥当な形式であるとは限らない．Codd モデルの改善策として，TNF ファイルを構成する各属性間の意味的相異に着目し，TNF ファイルをさらに5種のタイプの細分化したファイルに再構成する方法が Schmid らによって提案されている[15]．しかし，このように細分化しても，データの意味論のとらえ方自体が，データベース設計者の主観や，アプリケーションの環境によって異なってくる場合もあり，ファイルの形はやはり一意的には決まらない．このような一意性がないと，データベースに対して，ファイル間にまたがる検索や更新等の操作を行いたい場合，まず各ファイルと，ファイル群全体の構造を調べてから論理を組み立てなければならない．**1.** にも述べたように，今後データベースが多くの分野に進出し，共有化が進むことが予想されるが，その場合特に，上記の問題を解決する必要がある．本稿では，以上述べてきた立場に立って，むしろファイルという概念を表面に出さずにデータベース操作が行えることが，汎用的な検索，更新のために有効であると考える．**3.1** 以降では，リレーショナル・データベースで，このような考え方を実現させる手段として，領域間決定性の，論理的な取り扱いについて考察している．

3.1 束による決定性の記述

いま領域の集合 $\boldsymbol{D}=\{D_1, D_2, \cdots, D_n\}$ によってデータベースが構成されているとする．リレーショナル・モデルにおける決定性とは，一般にある領域の組に対して具体値を与えることによって，他の領域の具体値が決定できることであるから，決定性は

$$D_{p1}D_{p2}\cdots D_{pn} \to D_{q1}D_{q2}\cdots D_{qm} \qquad (1)$$

の形に表すことができる．ここで左辺は $D_{p1}, D_{p2}, \cdots, D_{pn}$ の組としての意味があるが，右辺の $D_{q1}, D_{q2}, \cdots, D_{qm}$ は独立な領域の集りにすぎないから，(1) は

$$D_{p1}D_{p2}\cdots D_{pn} \to D_{qi} \qquad (2)$$

の形の式の集りとして表すことができる．決定性の定義から明らかなように，D_{qi} が，集合 $\{D_{p1}, D_{p2}, \cdots, D_{pn}\}$ に含まれるときは，(2) は無条件に成立する．このことを「自明な決定性」と呼ぶことにする．自明な決定性の全体は，零元 "**O**" を加えることによって束 (lattice) を形成する．この束はブール代数の元に大小関係を導入してできる半順序集合と同等のものである*．例として，Fig. 1 (a) の，P, R, Y, T, B に対する自明な決定性は，**Fig. 3** で実線（太線と細線の区別をしない）で表される．この図で長円の中に書かれているのは領域名の組であり，自分と実線で結ばれている，下方の領域の組を決定している．ここに現れるような領域の組を束の「元」とよぶことにする．また零元に近い元を遠い元より「下にある」として，元の間に上位，下位の概念を定義する．

いま自明な決定性のみで作られた束に，新たに決定

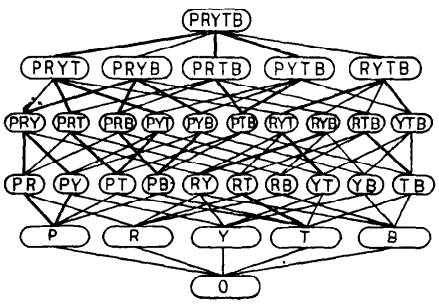


Fig. 3 An Example of Lattice Representation

* ブール代数 K の元 A, B について，$A \cdot B = A$ が成り立つとき $A \leqq B$ であると定義して K に大小関係を導入できる．

性がつけ加えられたとすると，束の構造が変化すると考えられる．たとえば Fig. 3 において，$RY \rightarrow B$ という決定性を付け加えた場合，元 RY と RYB，元 PRY と $PRYB$，元 RYT と $RYTB$，元 $PRYT$ と $PRYTB$ がそれぞれ同等の決定性をもつようになる．これはまた，決定性の上では，RY を最小元，$PRYT$ を最大元とする最大の部分束が，PRY を最小元，$PRYTB$ を最大元とする最大の部分束に重ね合わされた，ということもできる*．このように，領域の集合に決定性を与えていくと，その都度，いくつかの元が上位に移動することになる．Fig. 3 では決定性 $P \rightarrow R$，$P \rightarrow Y$，$T \rightarrow R$，$YR \rightarrow B$ を与えたとき，束内でおこる各元の決定性の変化が図示されている．ここで太線で結ばれている元は，上述の移動の結果，同等の決定性をもっている．ここで注目すべきことは，いくつかの決定性が組み合わさると新しい決定性が派生することである．たとえば，$Y, R \rightarrow B$ と $T \rightarrow R$ から，$Y, T \rightarrow B$ が派生する．Fig. 3 では元 YT が元 $RYTB$ と同等の決定性をもち，B を決定できるという形でこのことが示されている．またこれは，領域 B の具体値を検索する場合，Y，R の組をキーに指定してもよいし，Y，T の組をキーに指定してもよいことを意味している．従来，Codd のデータベース・モデルでは，ある領域の具体値にアクセスする場合，まずファイル名を指定し，次にそのファイル上のキー領域を利用してレコードを指定し，求める領域の具体値であるアトムを引き出す．という方法がとられた[11]．しかし今まで述べてきたように，ファイルという概念を表に出さず，データベースが領域の集合と，それらの間の決定性だけによって定義されているという見方に立つと，束の元（領域の組）は，本来どれでもキーのように用いてよいことになる．いまこれを「決定子」とよぶ．また決定子によって決定される領域をその「被決定子」とよぶことにする．すなわち，リレーショナル・データベースで，ファイルという固定した領域の組み合わせを介在させなくてよいとすれば，ある領域（または領域の組）を決定子として指定しさえすれば，その被決定子になっているどの領域の具体値も，ただちに引き出せるようにできる．そのためには，決定子からはじまって被決定子に至るまでの決定関係の道筋を順々にたぐって，領域間に Codd の結合 (join) 演算にあたるものを施していけばよい．

以上の考え方をデータベースのユーザから見た場合，データベースにアクセスするには，「何の領域を取り出すには，どの領域が決定子として使用できる」という事実を，各自の関心のあるものについてのみ経験的に知っていれば十分になり，特に大規模なデータベースを非専門的ユーザが共有するような状況下では利点が多いと思われる．

3.2 束の表現と操作

3.1 で述べた決定子と被決定子の結合を行うには，束の元の間で，派生した決定性を含むすべての決定性の有無が直接調べられなければならない．そのために最も簡明な方法は，すべての元についての決定性のテーブルを作ることである．これを決定性のグローバル・テーブルとよぶ．グローバル・テーブルは領域名の総数 n とした場合，$n \times 2^n$ の大きさのビット・テーブルで表される[16]．Fig. 3 の場合の例を **Fig. 4** に示した．ここで，束の元がいくつの領域を含んでいるかをその元のレベルとよぶ．Fig. 4 では，元はレベルの低いものから順に並べられており，レベルの区切りで横線が引いてある．

		P	R	Y	T	B
10000	P	1	①	①		△
01000	R		1			
00100	Y			1		
00010	T		①		1	
00001	B					1
11000	PR	1	1	①		△
10100	PY	1	①	1		△
10010	PT	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	△
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. 4 An Example of Global Table

* 束 L を構成する元の部分集合 L' がまた束を構成するとき，L' を L の部分束という．最大元 l と最小元 s を固定してできる部分束のうち，最多数の元を含むものを「l を最大元，s を最小元とする最大の部分束」という．上例で RY を最小元，$PRYT$ を最大元とする最大の部分束は，RY, PRY, RYT, $PRYT$, より成る．

グローバル・テーブルに現れる 2^n 個の束の元は，n 個の領域のどれを含むかによって，n ビットの2進数に対応させて識別することができる．Fig. 4 ではそれが左側の2列に示されている．第3列以降は各列が各領域名（属性）に対応しており，ビット・オンによってその領域を決定していることを示している．このテーブルの各行はそれぞれ1つの n ビットの2進数とみることもでき，これを元の値とよぶことにする．完成したグローバル・テーブルの各元の値は，束で表した場合に自分と同等の決定性をもつ最上位の元を表す2進数となっている．

グローバル・テーブルは次の手順によって簡単に作ることができる．まず，自明な決定性のみから成っているとき，各元の値としては自分自身を示している．Fig. 4 の例では，○と△のついていない1のみが立っている状態である．ここに（2）式より，$A_1\cdots A_k \rightarrow B$ の形をした決定性任意個を与えるアルゴリズムは次のようになる．

アルゴリズム

A1. $A_1,\cdots,A_k$ を含んでいる元について，その元の行で B に対応している列に1をたてる．これを，新しくつけ加える決定性すべてについて行う．

A2. 束のすべての元について，次の手順でその値を更新する．いま，値を更新しようとしている元を α とする．X は束の元を動く変数である．

A2.1 $X\leftarrow\alpha$（X の初期値として α を与える）

A2.2 元 X の値が自分自身を示していれば，その値を α の値として終り．そうでなければ A2.3 へ．

A2.3 $X\leftarrow$元 X の値によって示される元として A2.2 へ．

このアルゴリズムで $A1$ の部分は **3.1** で述べた部分束を重ね合わす操作に対応している．$A2$ の部分では，各元について $A1$ で行われた何回かの元の移動を合成して，各元の値を，自分と同等の決定性をもつ最上位元に引き上げる操作を行っている．

例として Fig. 3 で，いま自明な決定性に加えて，$T\rightarrow R$, $YR\rightarrow B$ を与えた場合を次に示す．ただし一般にグローバル・テーブル上で $D_1D_2\cdots D_m$ という元に対応する行で，D_k という領域に対するビット位置を $(D_1D_2\cdots D_m, D_k)$ で表すものとする．また，元 $D_1D_2\cdots D_m$ の値を $|D_1D_2\cdots D_m|$ で表す．

$(A1)$ $T\rightarrow R$ より
$(T,R)\leftarrow 1$, $(PT,R)\leftarrow 1$, $(YT,R)\leftarrow 1$, $(TB,R)\leftarrow 1$, $(PYT,R)\leftarrow 1$, $(PTB,R)\leftarrow 1$, $(YTB,R)\leftarrow 1$, $(PYTB,R)\leftarrow 1$

$YR\rightarrow B$ より
$(RY,B)\leftarrow 1$, $(PRY,B)\leftarrow 1$, $(RYT,B)\leftarrow 1$, $(PRYT,B)\leftarrow 1$

$(A2)$（元の値が更新されるもののみ）

$(A2.1)$ $X\leftarrow YT$
$(A2.2)$ $|X|=|YT|=RYT.$ $A2.3$ へ
$(A2.3)$ $X\leftarrow RYT.$ $A2.2$ へ
$(A2.2)$ $|X|=|RYT|=RYTB$ $A2.3$ へ
$(A2.3)$ $X\leftarrow RYTB$ $A2.2$ へ
$(A2.2)$ $|X|=|RYTB|=RYTB$
$|YT|=RYTB.$ として終り．

この例で，$|YT|$ が B を含むことから，$YT\rightarrow B$ が派生していることが示されている．Fig. 4 では，$P\rightarrow R$, $P\rightarrow Y$, $T\rightarrow R$, $YR\rightarrow B$ を与えたとき，$A1$ で立てられる1には○，$A2$ で立てられる1には△をつけて示してある．この結果，Fig. 4 は Fig. 3 と同等のものをあらわしている．以上述べてきたように，グローバル・テーブルを使えば，ある領域の組がどの領域を決定しているかが直接分かり，決定子と被決定子の対応を簡単に調べることができる．なお，領域数 n の場合のグローバル・テーブルの大きさ $n\times 2^n$ は，n が大きくなるにつれて非常に大きくなるが，Fig. 4 からも明らかなように，レベルの大きいところでは本質的でない情報（○や△のついていない“1”.）がふえ，またグローバル・テーブルの性質上，あるレベルから先の元の値は下のレベルの元の値から簡単に求められるので，必ずしもテーブル上の全情報を常時保持している必要はない．

最後に本稿で述べてきた決定性を扱う別の方法として．ブール代数を使う方法が知られているが[13]，この方法では，決定性の全体を積和標準形であらわしているため，項の増加に伴って式の変形に対する困難がふえる．本稿の方法は直観的で手順的操作が少なく，利用上有効である．

4. むすび

データベースを真に汎用性の高いものにするためには，a）データベースの中立性，独立性が高いこと，b）高いレベルの検索論理が効率よく実行できることが重要である．本稿では Codd モデルの考え方を基礎におき，a）の点をより充実させるために，領域の数学的取り扱いを中心に議論した．b）の点を含め，データベースのより一般性のあるモデル化と，そのソフ

ト的，ハード的な実現方法について現在研究中である．

謝 辞

著者に Codd モデルを紹介し，本研究の端緒を開いて下さった東京大学理学部国井利泰助教授，日本ユニバック総合研究所小林功武取締役，また日頃ご指導いただいている横浜国立大学工学部国枝寿博教授，土肥康孝助教授に感謝の意を表します．本稿の草稿に目を通し，貴重なご意見ご指摘を下さった電子技術総合研究所植村俊亮博士，古川康一氏にお礼申し上げます．

参 考 文 献

1) Data Base Task Group Report to the CODASYL Programming Language Committee, ACM (April 1971).
2) Feature Analysis of Generalized Data Base Management Systems, ACM (May 1971). (日本語訳：西村・中川監修「データベース・システム」, bit 臨時増刊, 共立出版, 1973).
3) たとえば小林：CODASYL DBTG 提案, Relational モデルと情報空間モデル，情報処理学会データベース研究会資料 74-17, 1974.
4) I. Kobayashi: A Formalizm of Information and Information Processing Structure, the SOKEN KIYO Vol. 5, No. 1, 1975.
5) CODASYL Development Committee: An Information Algebra: Phase 1 Report, Comm. of the ACM, Vol. 5, No. 4 (Aplil 1962).
6) J. L. Kuhns: Answering Questions by Computer: A Logical Study, RAND Corp. RM-5428-PR, 1967.
7) T. Winograd: Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, MAC, TR-84 M. I. T. (Jan. 1971).
8) E. F. Codd: A Relational Model of Data for Large Shared Data Banks, Comm. of the ACM (June 1970).
9) E. F. Codd: Normalized Date Base Structure, A Brief Tutorial, ACM-SIGFIDET, 1971.
10) E. F. Codd: Further Normalization of the Data Base Relational Model, Courant Computer Science Symposia 6, 1971.
11) E. F. Codd: Relational Completeness of Data Base Sublanguages, Courant Computer Science Symposia 6, 1971.
12) E. F. Codd: A Data Base Sublanguage Founded on the Relational Calculus, ACM-SIGFIDET, 1971.
13) C. Delobel and R. G. Casey: Decomposition of Data Base and the Theory of Boolean Switching Functions, IBM Journal Research Develop. (Sept. 1973).
14) E. F. Codd: Recent Investigations in Relational Data Base Systems, Proc. IFIP, 1974.
15) H. A. Schmid and J. R. Swenson: On the Semantics of the Relational Data Model, Proc. ACM-SIGMOD, 1974.
16) 有澤・國井：リレーショナル・データベースにおけるデータ構造の記述について，電子通信学会大会，1974.

(昭和 50 年 5 月 7 日受付)
(昭和 51 年 4 月 13 日再受付)