

データ従属性理論と関係データベース論理設計

勝野 裕文

(日本電信電話公社 武蔵野電気通信研究所)

1. はじめに

1970年にCoddが関係データベースの概念を提唱して以来、種々の関係データベース論理設計法が議論され始めた。なかでも、関数従属(Functional Dependency, FDと略す)等々データ従属性にもとづいた設計法は正規化理論と呼ばれ、多くの理論的研究がなされてきた。また、データ従属性に関する理論は単にデータベース論理設計に適用されるだけでなく、質問処理等の分野に適用されるとともに、応用を離れてそれ自体が理論的な興味の対象となっている。しかし、データ従属性に関する理論的な研究では着実に新しい結果が導き出されるのに^[1]、その出発点となつた正規化理論では新たな進展は見えず、正規化理論の問題設定に対する批判、問題点の指摘も多い。

本稿では、正規化理論の問題点をもとにして、正規化理論の目的を再検討し、従来の正規化理論とは異なる問題設定のもとで、データ従属性をデータベース設計へ応用する方法を考察する。

以下では、まず正規化理論の概要、その問題点を述べる。次に、同じ属性名が異なる実体(entity)又は役割(role)を表わしたり、属性名が表わす"その"の間に上下関係がある場合のデータ従属性を考察する。最後に、いくつかの関係スキーマを統合する時に、データ従属性を応用する方法について議論する。

2. 諸定義と記号の説明

関係、属性、関数従属、射影、結合の定義は既知であると仮定する([5]参照)。属性集合 U 上の関係 r を $r(U)$ と書く。 r の属性集合 $X (C U)$ 上への射影を

$r[X]$ と書く。関係 r_1 と r_2 の結合を $r_1 \bowtie r_2$ と書く。

$\langle R, U, F \rangle$ を関係スキーマと言ひ、 R は関係スキーマ名、 U は属性集合、 F は U 上の関係に関する制約を表わす。関係スキーマを表わすのに、しばしば $\langle R, U, F \rangle$ のかわりに R を使う。関係スキーマ R の具体例とは、 U 上の関係で、制約 F を満たすものをいう。

属性集合 X が次の条件を満たす時、 X は R の鍵であるという。

- (i) R 上のすべての具体例において、 $FD: X \rightarrow U$ が成り立つ。
- (ii) X のどの真部分集合 X' も(i)の性質を満たさない。

[注意] (i)の条件は、 F から $X \rightarrow U$ が導けるという条件と同値である。

$X = \bigcup_{i=1}^n X_i$ とする時、 $r(U)$ において、 $r[X] = r[X_1] \bowtie \dots \bowtie r[X_n]$ が成り立てば、 r において潜在結合従属性(Embedded Join Dependency, EJD) $\bowtie [X_1, \dots, X_n]$ が成り立つという。特に $X = U$ なるEJDを結合従属性(Join Dependency, JD)という。また、 $n=2$ の時のEJD(JD)を特に潜在多値従属性, EMVD, (多値従属性, Multivalued Dependency, MVD)という。

データ従属性は近年広いつクラスの制約を表わす意味で使われ始めたが、本稿では、FD, JD, EJDをデータ従属性と呼ぶ。

R が関係スキーマの集合を表わし、 J が R の元である関係スキーマ間の制約を表わす時、 $\langle R, J \rangle$ をデータベース・スキーマという。

3. 正規化理論

3.1 正規化理論の概要^{[2],[6]}

関係スキーマ $\langle R, U, F \rangle$ が "ある条件" を満たしている時に、 R は正規形であるという。ここでいう "条件" の違いによって種々の正規形が定義されている。

正規形という概念を考える意義の一つは、データベースに更新が行われた時に、種々の問題が発生しないような関係スキーマの形を求めることである。

正規化理論では、正規形の関係スキーマからなるデータベース・スキーマを求める、次のようなアルゴリズムを考える。

[正規化アルゴリズム]

入力：関係スキーマ $\langle R, U, F \rangle$ 。ここで、 R は "何らかの方法" で得られた、現実世界の一つのモデルであり、 F はデータ従属性の集合である。
出力：データベース・スキーマ $\langle R, U \rangle$ 。但し、 R の各関係スキーマは正規形で、 $U = \phi$ 、かつ R と $\langle R, U \rangle$ はある意味で等価である。

このアルゴリズムの本体を構成する戦略は次の二つに分類できる。

[合成法] 与えられた F (普通は FD のみからなる集合を考える) から冗長な FD を取り除いてできた FD の集合を F' とする。 F' の各 $FD: X \rightarrow Y$ に対して、関係スキーマ $\langle R_i, XY, X \rightarrow Y \rangle$ を構成することによって、データベース・スキーマを得る。

[分解法] F のデータ従属性をもとにして、 R を分解していく。この時、分解して新しくできた関係スキーマが正規形に近づくように、分解の仕方を選ぶ。

今、もはやこれ以上本質的な分解ができないという意味で、正規形が情報

*本来ならば、 $\langle R_i, XY, \{X \rightarrow Y\} \rangle$ と書くべきだが、煩雑なので、このように書く。

の基本単位を表わしているという見方をすると、合成法、分解法はそれぞれ次のように解釈できる。

(i) 合成法では、与えられた F のなかの非冗長な FD が情報の基本単位であるという認識に立って、データベース・スキーマを構成する。

(ii) 分解法では、 F をもとにして R を分解することによって、情報の基本単位を求めていく。

3.2 正規化理論の問題点^{[1],[3],[9],[2]}

正規化理論の問題点としてこれまで指摘されてきた点のなかで、ここでは次の点に焦点をあてて考える。

(i) アルゴリズムの入力として使われる関係スキーマ R を現実世界から抽出することが難しい。

① 一般に属性名はある実体又は実体の役割りを示しているが、現実世界では、異なる実体が同じ名前を持ったり、同じ実体が異なる名前を持つことがあるので、現実世界が正しく反映されるように、属性集合 U を選ぶことは難しい。

[例] 1^[1]

図.1 に示すように、属性集合と制約を抽出したとすると、従業員が決まると、その電話番号も一意に決まる ($EMP\# \rightarrow PHONE$)。しかし、居室 ($OFC\#$) と実験室 ($LAB\#$) にはそれぞれ別の電話番号があるので、矛盾が生じる。この矛盾の原因は、同じ電話番号でも場所によって区別しなければならぬのに、同じ $PHONE$ という名前を持つので、同一視している点にある。

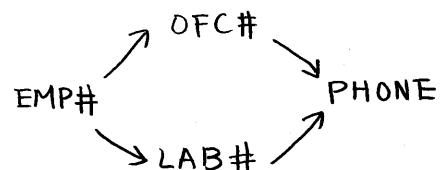


図.1

② データ従属性をアルゴリズムへの入力として用いる為には、データ従属性を現実世界から抽出することが簡単でなければならない。その為には、データ従属性が現実世界のどのような意味に対応しているかが明確となる必要がある。しかし、FD以外のデータ従属性の現実世界における意味は直観的にわかりにくい。

(ii) 入力として、属性名、データ従属性に関する情報しか用いてなく、他の情報、例えばどのようなデータベース操作が行なわれるかといった情報が設計の過程で使われていない。

(iii) 現実世界をモデル化する時に、我々は普通なるべく基本となる単位(例えばFDによって示される関数関係、多対多の関連等)に注目して、それを抽出する。従って、与えられたFDがそのような基本単位の情報を含んでいると仮定すると、分解法では、せっかく抽出した基本単位をわざわざまとめて、再度基本単位を求めるといった無駄を行っている。

3.3 データ従属性理論の最近の結果

上のような問題点に対して、従属性理論の側からいくつかの結果が得られている。

(i) 一つのFDから導けるMVDの集合が明らかとなり、それをもとにして実体・関連モデル、バックマン図式等でMVDがどのような概念に対応するかが明らかになり、^{[7],[8]}

(ii) (i)の結果の裏付けのもとに、MVDとFDからなる集合をもとにしたデータベース設計だけでなく、FDの集合と一つのFDをもとにしたデータベース設計が考えられている。^[7]

(iii) データの変更の単位として、objectという概念が提案され、objectとデータ従属性との間の関係が議論されている。^[14]

4. 属性名の取り扱い

本稿では、データベース設計者がまず情報の基本単位となるような関係スキーマを現実世界から抽出して、以後それらのスキーマの問題点を検討して、より良く現実世界を表わすように修正を繰り返すという立場をとる。その為、関係スキーマ R_1, R_2 が同じ属性名 A を含んでも、 A が同じ実体又は関連を表わすとは限らないという考え方をする。即ち、ID#が人の識別番号と本の識別番号の両方の意味で用いられているかもしれない。

また例1において、さらに電話番号→料金という関数関係を一緒に考えると、電話番号という属性名を、居室、実験室ごとに変更するのは不自然である。

本節では、上のような状況のもとでの属性名の取り扱い方を考察する。

属性 A が関係スキーマ R の属性であることを明示する時には、 A, R と書くことにする。

データベース・スキーマ $\langle R, J \rangle$ に対する属性森、Forest(R)とは、次の条件を満たす木の集合 $\{T_1, \dots, T_n\}$ をいう。

(i) 各 T_i は有向木であり、弧の向きは木の根から木の葉の方へ向かう。

(ii) T_i の各節点 N に対して、ラベル $l(N)$ がついている。 $l(N)$ は属性の集合であり、 N が T_i の根でなければ、 $l(N) \neq \emptyset$ である。

(iii) R の任意の属性 A, R に対して、 $A, R \in l(N)$ なる性質を持つ、Forest(R)の節点 N が一意的に決まる。

[例2]

例1の状況において、次の関係スキーマからなる $\langle R, J \rangle$ を考える。

$\langle R_1, \{EMP\#, OFC\# \}, EMP\# \rightarrow OFC\# \rangle$

$\langle R_2, \{EMP\#, LAB\# \}, EMP\# \rightarrow LAB\# \rangle$

$\langle R_3, \{OFC\#, PHONE\}, OFC\# \rightarrow PHONE \rangle$

$\langle R_4, \{LAB\#, PHONE\}, LAB\# \rightarrow PHONE \rangle$

$\langle R_5, \{PHONE, FEE\}, PHONE \rightarrow FEE \rangle$
この時, $Forest(R)$ の一部分を図.2 に示す.

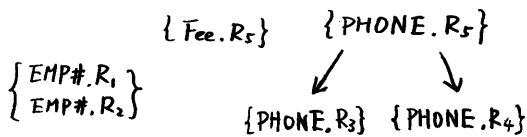


図.2 $Forest(R)$ の一部分

$Forest(R)$ において、節点 N から節点 M への有向道がある時に、 $N \rightarrow M$ と書くことにする。(但し、 $N \rightarrow N$ とは)

$Forest(R)$ は次のような意味を持つ。

(i) $A.R_i, B.R_j \in \mathcal{L}(N)$ の時。

(i) の条件が成り立つ時には、 $A.R_i$ と $B.R_j$ が全く同じ実体又は役割りを表わすと、データベース設計者が考えている事を意味している。例えば、例2における $EMP\#, R_1$ と $EMP\#, R_2$ である。但し、この場合でも、 R_i と R_j の任意の時点での具体例 r_i, r_j に対して、 $r_i[A] = r_j[B]$ が成り立つとは限らない。

(ii) $N \rightarrow M$ から $A.R_i \in \mathcal{L}(N)$ から $B.R_j \in \mathcal{L}(M)$ が成り立つ時。

この時、 $A.R_i$ は $B.R_j$ を含むより広い実体、役割りを表わしている。例えば、例2の $PHONE, R_5$ は特定の場所の電話ではなく、設置場所にとらわれない電話一般を指している。

[注意]

① $A.R_i$ と $B.R_j$ が表わす実体又は役割りの間に共通部分があり、それを $C.R_k$ が表わすような場合は、各 T_i は木ではなく、rooted acyclic directed graph と考えなければならぬが、以後の議論は T_i が木であるとしても本質的な違いはないので、簡単な方を選んだ。

② $A.R_i$ と $B.R_j$ が同じ実体、役割りを表わすが、異なるものを表わすか、設計者が無関心な場合が考えられる。この場合、他との違いが明確にされている属性は木の根のラベルに含めて、根がからんだ場合の上の (i), (ii) の解釈を変

更することによって扱えるが、議論が煩雑になるので省略する。

属性 A の意味が現われた関係スキーマによって異なる場合、記号を次のように定義する。

$A.R_i = B.R_j \Leftrightarrow A.R_i, B.R_j \in \mathcal{L}(N)$ なる N が存在する。

$A.R_i \rightarrow B.R_j$

$\Leftrightarrow N \rightarrow M$ から $A.R_i \in \mathcal{L}(N)$ から $B.R_j \in \mathcal{L}(M)$ なる N, M が存在する

$A.R_i \approx B.R_j \Leftrightarrow A.R_i \rightarrow B.R_j$ 又は $B.R_j \rightarrow A.R_i$

$X \sqsubset Y \Leftrightarrow$ 任意の $A.R_i \in X$ に対し、ある $B.R_j \in Y$ が存在して $A.R_i \approx B.R_j$ である

$X \approx Y \Leftrightarrow X \sqsubset Y$ から $Y \sqsubset X$

[注意]

① 属性の上位下位概念を表わす $\approx$ に關しては推移律が成り立たない。

② 集合和 $\cup$, 集合積 $\cap$, についても $\approx$ に基づいた新しい記号が必要だが、一般に前後関係が $\approx$ に基づいているが、 $\approx$ に基づいているかの区別は容易なので、新たな記号は設けない。

データベースの論理設計を進めて行く過程で、矛盾が生じたりして、 $A.R_i = B.R_j$ が間違いであることが発見されたり、属性名を変更したりして、より良く現実を反映するように、 $Forest(R)$ の構造を変更することがある。この操作を以後、属性識別操作と呼ぶことにする。

5. データベース論理設計

ここではデータベースの論理設計を次の二つの段階にわけて考える。

[第一段階]

現実世界から情報を基本単位を抽出して、現実世界を正しく記述するデータベース・スキーマの一つを構成する段階。

[第二段階]

第一段階で得られたデータベーススキーマをより使い易いデータベーススキーマへ変換する段階。

5.1 第一段階の設計

第一段階の設計の結果得られるデータベーススキーマは次の条件(★)を満たさなければならぬ。

(★) 設計時点で考えられる、現実世界のデータ検索要求、更新要求がデータベーススキーマにおいて、正しく実現できる。

この条件を満たすデータベーススキーマを設計する手法が従来から多く提案されてきた。正規化理論はその一つにあげられるべきである。

本稿では以下の立場をとる。

- (i) データ従属性は制約の一部を表わしているにすぎないので、その言葉だけで、データベース設計を議論できない。
- (ii) データ従属性はおもにデータベーススキーマの意味的な矛盾、冗長性の検出に用いる。

5.1.1 基本関係スキーマ

最初に設計者は情報の基本単位(こゝでは基本関係スキーマと呼ぶ)と基本関係スキーマ間にまたがる制約を現実世界から抽出する。次に、得られたデータベーススキーマが正しく現実世界をモデル化しているか検討し、問題があれば修正を繰り返す。

本稿では、現実世界からどのように基本関係スキーマを選び出すかについては議論しない。しかし、多くのデータベース設計手法では、基本関係スキーマに類した、情報の基本単位をまず選び出していると考ええる。

[定義]

$\langle R, U, F \rangle$ が次の条件のいずれかを満たす時、 R は基本関係スキーマである

という。

- (i) R は U の間の多対多の関連を表わし、 F は FD, JD を含まない。
- (ii) R は $X \rightarrow A$ という FD に対応する関数関係を表わし、 $U = X \cup \{A\}$ から R の鍵は X である。

[注意]

- ① R が基本関係スキーマであったとしても、 F が $EJD, EMVD$ を含むことがありえるし、従属性以外制約を含むこともありえる。
- ② 基本関係スキーマの集合に含まれる R_1, R_2 において、 $U_1 \subset U_2$ から $F_1 = F_2|_{U_1}$ となることもある。但し、 $F_2|_{U_1}$ は F_2 の U_1 上への制限を示す。

[例3]

学生が履習した科目の成績を判定者が判定する時に、各科目ごとに複数の判定者がいて、そのすべての判定者がそれぞれの主観にもとづいて、学生の成績を出すとする。この時、基本関係スキーマは以下の通りである。

$\langle R_1, \{ \text{学生, 科目} \}, \phi \rangle$

$\langle R_2, \{ \text{科目, 判定者} \}, \phi \rangle$

$\langle R_3, \{ \text{学生, 科目, 判定者, 成績} \}, F \rangle$

$F = \{ \text{学生, 科目, 判定者} \rightarrow \text{成績}, \{ \{ \text{学生, 科目} \}, \{ \text{科目, 判定者} \} \}$

5.1.2 ハイパーグラフ

データベーススキーマ $\langle R, J \rangle$ の冗長性、意味的な矛盾を検出する道具として、ハイパーグラフを定義する。

[定義]

ハイパーグラフ $H = \langle V, E \rangle$ が次の条件を満たす時、データベーススキーマ $D = \langle R, J \rangle$ に対応するハイパーグラフという。

- (i) $V = \{ N \mid N \text{ は } Forest(R) \text{ の節点で} \\ \text{かつ } l(N) \neq \phi \}$

- (ii) $E = \{ e_{R_i} \mid R_i \in R \} \cup \{ e_{NM} \mid Forest(R) \\ \text{において、} N \text{ と } M \text{ の間に弧が存在する} \}$

$e_{R_i} = \{ N \mid A \in U_i \text{ から } A, R_i \in l(N) \}$

$e_{NM} = \{ N, M \}$

e_{R_i} をスキーマ辺、 e_{NM} を属性辺と呼ぶ。

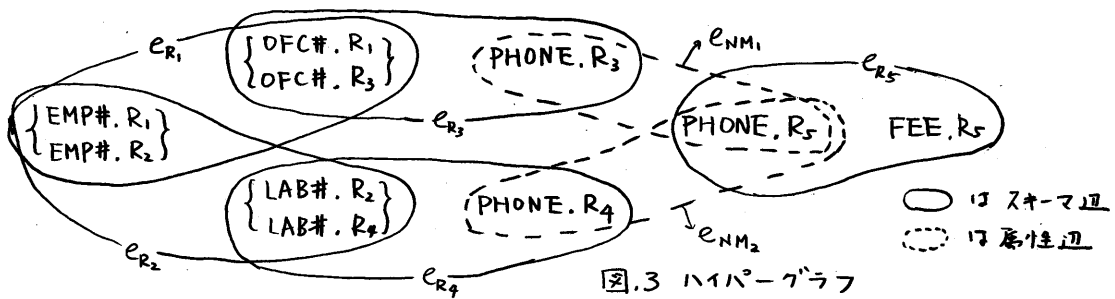


図.3 ハイパーグラフ

[例4]

図.3に例2のデータベーススキーマに対応するハイパーグラフを示す。

[定義]

$D = \langle V, E \rangle$ において、 $\langle e_1, \dots, e_k \rangle$ が道であるとは、 $e_i \cap e_{i+1} \neq \emptyset$ ($1 \leq i \leq k-1$) なることをいう。

道 $\langle e_1, \dots, e_k \rangle$ が正規道であるとは、 e_i, e_j が属性辺 e_{NM1}, e_{NM2} であり、 N_1, N_2 がともに T_x の節点であれば、 $N_1 \neq N_2$ かつ ($N_1 \rightarrow N_2$ 又は $N_2 \rightarrow N_1$) が成り立つことをいう。

正規道ではない道を非正規道という。

[例5]

図.3において、 $\langle e_{R1}, e_{R3}, e_{NM1}, e_{R5} \rangle$ は正規道、 $\langle e_{R3}, e_{NM1}, e_{NM2} \rangle$ は非正規道

[定義]

正規道 $\langle e_0, e_1, \dots, e_k \rangle$ が D の γ -サイクル $\Leftrightarrow e_0, e_1, e_k$ はそれぞれスキーマ辺、 e_{R1}, e_{R4}, e_{R3} であり、 $A, B \in U_i$ なる属性で次の条件を満たすものが存在する。

(i) $\{A, R_1\} \not\subset U_p, R_p$ かつ $\{B, R_1\} \subset U_q, R_q$ 但し、 $U_p, R_p = \{C, R_p \mid C \in U_p\}$

(ii) e_j ($j=0, 1, k$) がスキーマ辺 e_{R1} なるば、 $\{A, R_1\} \not\subset U_r, R_r$ かつ $\{B, R_1\} \subset U_s, R_s$ であり、また $\{A, R_1\} \not\subset U_q, R_q$ かつ $\{B, R_1\} \subset U_p, R_p$ が成り立つ

[例6]

図.3において、 γ -サイクルは存在しない。もし、 e_{NM2} がスキーマ辺であったならば $\langle e_{R1}, e_{R3}, e_{NM1}, e_{NM2}, e_{R4}, e_{R2} \rangle$ は γ -サイクルになる。

[注意]

上で定義した γ -サイクルは Fagin の定義した γ -サイクルに対応している。

5.1.3 冗長性、意味的矛盾の検出 [定義]

$D = \langle R, J \rangle$ において R_i が冗長である $\Leftrightarrow R_i$ と異なる $R_{i1}, \dots, R_{ik}$ が存在して、いつの時点においても、

$$r_i = r_{i1} \boxtimes \dots \boxtimes r_{ik} [U_i, R_i]$$

が成り立つ。但し、 $\boxtimes$ は結合対象となる属性が全て一致することを示す。

上で定義した意味での冗長性や、意味的矛盾を検出するのに、次の二つの方法が使える。

[方法1]

$U_i, R_i \subset U_j, R_j$ となるような関係スキーマ R_i, R_j を探す。

この時、 R_i が冗長かどうかは、データの意味にさかのぼらなければわからない。その時の一つの目安は、 r_i における更新操作が、現実世界の実際の更新の基本単位を反映しているかという点である。もし、反映していれば、 R_i は一般に冗長ではない。

また、 $R_i \neq R_j \mid U_i$ である時には、Forest(R)の構成に問題があるか、 R_j の制約として何らかの見逃しがあるかどうかである。

[例7]

$\langle R_1, ABC, AB \rightarrow C \rangle, \langle R_2, BC, C \rightarrow B \rangle$ において、 $BC, R_2 \subset ABC, R_1$ とすると、 R_1 の制約として、 $C \rightarrow B$ を見逃している可能性がある。

[方法2]

D の γ -サイクルをさがりとする。

①が γ -サイクルを持つ理由として次の点が考えられる。

- (i) 冗長な関係スキーマがある。
- (ii) 同じ属性名が異なる型体又は役割りを表わしているのに、設計者がそれを区別しようとしていない。この場合、属性名を変更する等、適当な属性識別操作を行うと、①から γ -サイクルを除去できる。

個々の γ -サイクルが (i), (ii) のどちらに対応するかは、一般に個々の関係スキーマの意味を検討し直さないとわからない。しかし、その時に役に立つ道具として次を定義する。

[定義]

$R_i \rightarrow R_j \Leftrightarrow \mathcal{D}$ の正理道 $\langle e_{R_i}, e_1, \dots, e_k \rangle$ が存在して次の条件を満たす。

- (i) $e_k = e_{R_j}$
- (ii) $e_p (1 \leq p \leq k)$ がスキーマ e_{R_i} であり、 $e_1, \dots, e_{p-1}$ のうちのすべてがスキーマ e_{R_i} の鍵 X で、 $X, R_i \subset U_i, R_i \cup U_i, R_i \cup \dots \cup U_k, R_i$

を満たすものが存在する。

$R_i \twoheadrightarrow R_j \Leftrightarrow \mathcal{D}$ の正理道 $\langle e_1, \dots, e_k \rangle$ で次の条件を満たすものが存在する。

- (i) $e_1 = e_{R_i}$ か、 $e_k = e_{R_j}$
- (ii) $e_l (1 \leq l \leq k)$ がスキーマ e_{R_i} ならば、 R_i の鍵は U_i である。

[例8]

例2のデータベーススキーマにおいて、 $R_1 \rightarrow R_5, R_1 \rightarrow R_4$ であるが、 $R_1 \twoheadrightarrow R_5$ ではない。

(i) 冗長性の検出

$\langle R, J \rangle$ において、次の条件(★★)が成り立つ時、 R が冗長な関係スキーマを含む可能性がある。

(★★) ある $R_i, R_j, R_k \in R$ が存在して

- ① R_i の鍵は U_i であり、即ち R_i は関数関係を示す。
- ② $R - \{R_i\}$ において、 $R_j \rightarrow R_k$
- ③ R_i の鍵 X で、 $X, R_i \subset U_i, R_j$

なるものが存在する。

④ $(U_i - X), R_i \subset U_k, R_k$

条件(★★)は $\langle R_i, AB, A \rightarrow B \rangle, \langle R_j, BC, B \rightarrow C \rangle, \langle R_k, AC, A \rightarrow C \rangle$ において、 R_k が冗長だとする考え方に対応している。

しかし、(★★)が成り立つからといって、 R_i が冗長だと即断することはできない。冗長性の定義に基いて、確認しなければならぬ。

(★★)は $\mathcal{D}$ において $e_{R_i}, e_{R_j}, e_{R_k}$ を含む γ -サイクルが存在することを示している。しかし、冗長な関係スキーマを含む γ -サイクルがすべて(★★)の条件に対応している訳ではない。例えば、 $\langle R_i, AB, \emptyset \rangle, \langle R_j, BC, \emptyset \rangle, \langle R_k, AC, \emptyset \rangle$ において、 R_k が冗長なことはありえる。

(ii) 意味的な矛盾の検出

次の条件(★★★)を考える。

(★★★) R_i, U_i の部分集合 X, R_j, R_k, R_l が存在して次の条件を満たす。

① R_j の鍵 Y で $X, R_i \approx Y, R_j$ となるものが存在する。

② $R - \{R_i\}$ において、 $R_j \rightarrow R_k$

③ R において、 $R_i \twoheadrightarrow R_l$

④ U_l に含まれる属性 A で
 $\{A, R_l\} \subset U_k, R_k \cap U_l, R_l$
 $\{A, R_l\} \not\subset X, R_i$

を満たすものが存在する。

(★★★)は、 $R_i \rightarrow R_j \rightarrow R_k$ によつて、 X を決めると A が決まるという関数関係が存在することと示し、 $R_i \twoheadrightarrow R_l$ によつて X と A の間に多対多の関連があることを示している。従つて、(★★★)が成り立つ時には、意味的な矛盾があるので、その原因を調べる必要がある。

$i=j=k$ の場合を除いて、(★★★)は $\mathcal{D}$ に γ -サイクルが存在することと示している。

5.2 第二段階設計

第二段階では、第一段階で得られたデータベーススキーマをより使い易い

形に変換する。

最も単純な場合として次の場合が考えられる。第一段階では基本関係スキームを出発点としていたので、最終的に得られたデータベーススキームのうち、 $\langle R_1, XA, X \rightarrow A \rangle, \langle R_2, XB, X \rightarrow B \rangle$ があり、 $X.R_1 = X.R_2$ なる場合がある。この時には、一般に $\langle R_3, XAB, X \rightarrow AB \rangle$ を考えた方がわかり易い。但し、任意の時点で $r_1[X] = r_2[X]$ が成立つてなければ、 R_1, R_2 を R_3 で置きかえる為には、Null値の導入が必要となる。

データ従属性理論ではすでに、

- ・データベーススキームの等価性^[10]
- ・View上の更新問題^[4]
- ・基底関係上のデータ従属性がどのようにViewに反映されるか^[11]
- ・Null値の取り扱い^[5]

等について、多くの結果が得られている。しかしながら、

- ・更新操作を考慮したデータベーススキームの等価性
- ・Null値に関しては種々の意味付けが可能で、その個々の意味ごとの取り扱いが必要

等に関して、残されている問題が多い。

上記の既に得られている結果、残された問題の解決を通じて、使い易いデータベーススキームの候補を生成するのにデータ従属性理論が使えることが期待できる。

6. おわりに

本稿では、正規化理論の見直しを行ない、正規化理論とは異なる形で、データ従属性をデータベース設計に使う方法を示した。具体的には、データベーススキーム内の冗長性、意味の矛盾を検出する一つの道具として、データ従属性を用いた。

[謝辞] 日頃御指導頂くデータベース理論研究会の皆様、第一、第八研究室の皆様に深謝致します。

参考文献

- [1] Atzeni, P. and D.S. Parker: Assumptions in Relational Database Theory, Proc. PODS, pp.1-9, 1982
- [2] Beer, C., P.A. Bernstein and N. Goodman: A Sophisticated's Introduction to Database Normalization Theory, Proc. 4th Inter. Conf. on VLDB, pp.113-124, 1978
- [3] Bernstein, P.A. and N. Goodman: What Does Boyce-Codd Normal Form Do?, Proc. 6th Inter. Conf. on VLDB, pp.245-259, 1980
- [4] Dayal, U. and P.A. Bernstein: On the Correct Translation of Update Operations on Relational Views, ACM TODS, Vol.7, No.3, pp.381-416, 1982
- [5] Fagin, R.: Normal Forms and Relational Database Operators, Proc. 1979 ACM-SIGMOD, pp.153-160
- [6] Fagin, R.: Types of Acyclicity for Hypergraphs and Relational Database Schemes, IBM Research Report, RJ3330, 1981
- [7] Fagin, R., A.O. Mendelzon, and J.D. Ullman: A Simplified Universal Relation Assumption and its Properties, ACM TODS, Vol.7, No.3, pp.343-360, 1982
- [8] Goldstein, B.S.: formal Properties of Constraints on Null Values in Relational Databases, Technical Report 80-013 of SUNY at Stony Brook, 1980, revised in 1981
- [9] Imielinski, T. and W. Lipski: A Systematic Approach to Relational Database Theory, Proc. SIGMOD, pp.8-14, 1982
- [10] Imielinski, T. and W. Lipski: A Technique for Translating States Between Database Schemata, Proc. SIGMOD, pp.61-68
- [11] Klug, A.: Calculating Constraints on Relational Expressions, ACM TODS, Vol.5, No.3, pp.260-290, 1980
- [12] LeDoux, C.H. and D.S. Parker: Reflections on Boyce-Codd Normal Form, Proc. 8th Inter. Conf. on VLDB, pp.131-141, 1982
- [13] Lien, Y.E.: On the Semantics of the Entity-Relationship Data Model, Entity-Relationship Approach to System and Design, P.P. Chen (ed.), pp.155-167
- [14] Sciore, E.: The Universal Instance and Database Design, Princeton University Technical Report, TR-271, 1980
- [15] Ullman, J.D.: Principles of Database Systems, Computer Science Press, 1980
- [16] 上林: 関係データベースの論理設計, 情報処理, Vol.23, No.7, pp.659~675, 1982
- [17] 上林: 従属性理論, 情報処理, Vol.23, No.9, pp.835~847, 1982