解　説

1. 古典データベースから演繹データベースへ†

小　林　功　武††

1. はじめに

データベースの理論はよりよいスキーマを求めるための設計論とすぐれた操作系を得るための処理論で構成されている．Codd の関係モデル[1]の提起により，これらがインプリメンテーションと切り離され，論理的な議論の基礎が築かれたといえる．1970年代に，前者について正規形および従属性の理論，後者に対しては関係論理と関係代数を土台としたさまざまな最適化技法が開発され，その成果は商用の関係型データベース・マネジメント・システムに取り入れられてきた．しかしながら，これらの理論展開には Codd の枠組がそのまま用いられており[2),3)]，スキーマ変換問題や仮想関係の処理，整合性保証，機密保護などになお多くの問題を積み残していた．

1980年代に入り，論理型プログラミング言語[4),5)]の発達がデータベース分野にも新しい波をもたらした．論理型プログラミング言語は検索言語としても利用できるが，そのときの効率は多量のデータを蓄えたデータベース上の検索に耐え得るものではなく，この意味で古典的なデータベース技術をなんらかの形で取り入れる必要があった．一方，データベース側で未解決な問題の多くは，一般規則の扱い，とくに演繹機構そのものにかかわるものであった．ここで，両者の接点としての演繹データベースへの動機が与えられたことになる[6)]．

本稿では，古典データベースで積み残されていた種種の問題のうち，とくに一般規則の扱いに密接にかかわるものを取りあげた．これらは演繹データベース開発の背景となるものであり，新しい枠組の中で解決され，あるいは今後の発展がまたれるものであるといえよう．解説に当たっては，論理型プログラミングとデータベースの共通の基盤として，述語論理による形式化を用いる．

† Classical to Deductive Databases by Isamu KOBAYASHI (Sanno Institute, College of Management and Informatics).
†† 産能大学経営情報学部情報学科

2. 述語論理とデータベース

まず，述語論理の立場からデータベースのいくつかの基本概念を見直してみよう．ここでは関数をもつ一階述語論理を仮定する．一階述語論理の式の文法は既知のものとする．一階述語論理の解釈は，与えられた一つの対象領域 D の上で定義された．

［I1］ 式中に現れたそれぞれの定数に D 中の一つの要素を割り当てる．

［I2］ 式中に現れたそれぞれの n 変数関数記号に D 上の1個の n 変数関数

$$f: D^n \rightarrow D$$

を割り当てる．

［I3］ 式中に現れたそれぞれの n 変数述語記号に D 上の1個のブール値関数

$$P: D^n \rightarrow \{true, false\}$$

を割り当てる．

の三つの写像で構成されている．一つの解釈の下で，n 個の自由変数をもつ開いた式に対しては n 変数ブール値関数が割り当てられ，閉じた式には真偽値が割り当てられる．

証明論の立場では対象領域を固定せず，任意の解釈を考える．閉じた式の集合の充足可能性は Herbrand 領域の上で判定できる[7)]．しかし，式の真偽値を問題とするデータベースの環境では，対象領域 D としてアプリオリに与えられたものを用い，写像［I1］も［I2］も固定する．さらに，一部の述語については写像［I3］も固定される．つまり，これらに対しては特定の解釈しか考えない．

［I3］が固定されたものを評価可能述語ということにする．評価可能述語には，対象領域の構造に依存しないものとその構造に由来するものがある．記号列としての等しさを判定する等号は前者の例であり，たとえば整数1と実数1.0のように数値としての等しさを判定する等号は後者の例である．対象領域に順序が導

入されていれば不等号が定義できる．対象領域が複合値を表す要素を含む場合には，さらにさまざまな評価可能述語が導入されよう．一つの評価可能述語について無限個の基礎原子式が真値をとりうる．評価可能述語をもつ基礎原子式の真偽値はなんらかの（有限の）手段により常に評価しうるという前提をおく．

評価可能でない，つまり［I3］が固定されていないものを関係述語ということにしよう．実世界における個別の事実は関係述語に対する解釈を与えることによって表される．データベース関係は特定の関係について一つの解釈の下で真値を与えられた基礎原子式の集合である．関係述語をさらに基底関係に対応するものと仮想関係に対応するものに区分する．前者に対する基礎原子式の真偽値はファイルとして表現された関係の参照によって直接決定される．これらの関係中の，つまり真値が与えられる基礎原子式はたかだか有限個であると仮定される．後者についての基礎原子式の真偽値は間接的に，つまりなんらかの演繹処理によって決定されなければならない．

さて，（実世界を表現するために）与えられた関係述語の集合をデータベース・スキーマと呼ぶことにしよう．各述語が何変数のものであるかは示されているものとする．通常は，データベース・スキーマに各関係に与えられた属性の値の範囲が示され，したがって関係述語がタイプ付けされる．各関係の識別項目となる属性を与えることもある．しかし，ここではそのような立場をとらず，領域規則，たとえば

$$\forall x \forall y \forall z(R(x,y,z) \rightarrow D_1(x) \wedge D_2(y) \wedge D_3(z)) \tag{2.1}$$

や，基本関数従属，たとえば

$$\forall x \forall y \forall y' \forall z \forall z'(R(x,y,z) \wedge R(x,y',z') \rightarrow y=y' \wedge z=z') \tag{2.2}$$

（＝は領域に依存しない等号）

も一般規則として扱うことにする．データベース・スキーマ S 中の関係述語に対する可能な解釈全体の集合を $[S]$ で表すことにしよう．

$[S]$ の要素がデータベースである．一つのデータベースは関係述語をもつ一つの基礎原子式集合であるが，これをさらに基底関係で構成された基底データベースと，仮想関係で構成された仮想データベースに分割する．前者を外延データベースということもある．実際には，ある規則群によって制約された基底データベースと，その基底データベースと別の規則群によって生成された仮想データベースを考える．

与えられた定数，変数，関数記号，評価可能述語および S 中の関係述語を使って構成された閉じた式の集合を規則集合といい，規則集合の要素を一般規則と呼ぶ．規則集合が基礎原子式を含んでも差し支えない．規則集合を内包データベースということもある．与えられた規則集合を $C(S)$ で表すことにしよう．また $[S]$ の中の解釈で $C(S)$ 中の規則をすべて満足するものの集合を $[S, C(S)]$ で表すものとする．その中のデータベースは整合しているといわれる．

$C(S)$ の論理的帰結であって仮想関係に対応する関係述語を含まない式の集合が（古典データベースにおける）整合性制約である．（整合した）基底データベースはこの意味の整合性制約のモデルの一つである．仮想データベースは，規則集合 $C(S)$ に一つの（整合した）基底データベース中の基礎原子式を加えたものの最小モデルと定義される．（ここでは最小モデルが存在する場合のみを考える．）

上述のデータベース・スキーマ，データベース，および規則集合の定義は以下の議論の展開に十分なものである．複合値の扱いや集約関数の導入に当たって一階述語論理の文法そのものを拡張することもあり，その場合は式の解釈に手を加える必要があるが，この問題の詳細は本特集の他項[8]にゆずった．

3. 述語系の選択と変換

表現すべき世界がまったく同じであったとしても，その記述は各人各様である．述語論理の立場からみれば，各人が異なった対象領域と異なった述語系を用いているものと言えよう．異なった対象領域間の対応問題や，対象領域と述語系間のインタラクションはそれ自身興味ある問題ではあるが，ここでは対象領域は一つのものを固定し，その上の述語系の選択と変換問題を考えてみよう．

データベースの設計論は，なんらかの意味ですぐれたデータベース・スキーマ，つまり関係述語系を求める問題であると言えよう．同じ世界を同じ詳しさで表現しうる二つの述語系が与えられたとき，そのいずれを選ぶべきかはデータベースの物理表現の容易さ，記憶容量，検索効率，更新効率，整合性保証の容易さ，その他の処理の記述の容易さおよび効率などさまざまな要因に関係し，そのどれを重視すべきかはシステム環境に依存する．正規形の理論は一つの指針を与えたが，十分な一般性を有するものとは言い難い．ここでは前章で述べた枠組のもとで，二つのスキーマ，つま

り述語系の等価性を論じよう.

二つのデータベース・スキーマ S と T が与えられ, $[S]$ 中の任意のデータベースを $[T]$ 中のデータベースに書き換える規則が与えられたとき, これを S から T へのスキーマ変換と呼ぶ. たとえば, 3変数述語 P で構成されたデータベース・スキーマ S と, 2個の2変数述語 Q_1, Q_2 で構成されたデータベース・スキーマ T の間に,

$$\forall x \forall y (\exists z P(x,y,z) \rightleftarrows Q_1(x,y)) \quad (3.1)$$

$$\forall x \forall z (\exists y P(x,y,z) \rightleftarrows Q_2(x,z)) \quad (3.2)$$

で一つのスキーマ変換が定義できる. これはよく知られた射影変換である.

スキーマ変換は $[S]$ から $[T]$ の中への関数として一般に単射にはならない. しかし, S 上にいくつかの一般規則を仮定することによってこれを単射にし得る. 上述の例では, 多値従属

$$\forall x \forall y \forall y' \forall z \forall z' (P(x,y,z) \wedge P(x,y',z') \rightarrow P(x,y,z')) \quad (3.3)$$

を与えれば射影変換は単射となる. (多値従属は射影変換が単射となるための十分条件ではあるが必要条件ではない.) これらの一般規則で構成された規則集合を $C_0(S)$ とすれば, スキーマ変換は $[S, C_0(S)]$ から $[T]$ への単射となる.

$[S, C_0(S)]$ のスキーマ変換による像は, T 上のいくつかの一般規則で特徴づけられる. 上例では, 二つの包含従属

$$\forall x \forall y \exists z (Q_1(x,y) \rightarrow Q_2(x,z)) \quad (3.4)$$

$$\forall x \forall z \exists y (Q_2(x,z) \rightarrow Q_1(x,y)) \quad (3.5)$$

で特徴づけられることになる. これらの一般規則で構成された T 上の規則集合を $C_0(T)$ とすれば, このスキーマ変換は $[S, C_0(S)]$ から $[T, C_0(T)]$ への双射となり, したがって逆変換が定義できる. 上例では,

$$\forall x \forall y \forall z (Q_1(x,y) \wedge Q_2(x,z) \rightleftarrows P(x,y,z)) \quad (3.6)$$

で定義された自然結合変換が逆変換となる. (3.1), (3.2), (3.6)のような変換規則は $S \cup T$ 上の一般規則と考えてよい.

$[S, C_0(S)]$ と $[T, C_0(T)]$ 間の双射を無損失変換と呼ぶ. 無損失変換があれば両者の表現力は等価である. 等価性は二つのスキーマ (述語系) それぞれの上の規則集合を考慮してはじめて成立することに注意せよ. 実際, 上例では包含従属(3.4)と(3.5)は変換規則(3.1), (3.2)の論理的帰結であり, 多値従属(3.3)は変換規則(3.6)の論理的帰結となっている. もし $C_0(T)$ 中の二つの包含従属の一方あるいは双方を欠く場合には, そのままの形では等価性は成立しない. この場合は, 対象領域 D に (定義されていないことを示す) 空値を導入し, 空値に関するいくつかの規則を加え, さらに書き換え規則を修正することによって同値性が保たれる.

通常, S の上には $C_0(S)$ 中の一般規則以外にさまざまな一般規則が課せられている. スキーマ変換にともない, これらの一般規則も変換されなければならない. 規則集合 $C(S)$ が与えられたとき, $C(S) - C_0(S)$ 中の一般規則はスキーマ変換規則 ($S \cup T$ 上の一般規則) を用いて T 上の一般規則に変換できる. たとえば, 前述の射影/自然結合変換の例で, $C(S)$ 中に二つの関数従属

$$\forall x \forall y \forall y' \forall z \forall z' (P(x,y,z) \wedge P(x,y',z') \rightarrow y = y') \quad (3.7)$$

$$\forall x \forall x' \forall y \forall z \forall z' (P(x,y,z) \wedge P(x',y,z') \rightarrow z = z') \quad (3.8)$$

があったとすれば, 射影変換にともなって

$$\forall x \forall y \forall y' (Q_1(x,y) \wedge Q_1(x,y') \rightarrow y = y') \quad (3.9)$$

$$\forall x \forall x' \forall y \forall z \forall z' (Q_1(x,y) \wedge Q_2(x,z) \wedge Q_1(x',y) \wedge Q_2(x',z') \rightarrow z = z') \quad (3.10)$$

に変換される. これを $C_0(T)$ に加えて $C(T)$ を構成すれば射影変換は $[S, C(S)]$ から $[T, C(T)]$ への双射となる.

データベース設計に関連してさまざまな無損失変換の例が知られ[9)~11)], そのいくつかは正規形を作るさいに使われる.

とくに射影によって得られる正規形に関連して, 1970年代後半に従属性の理論が展開された. 一つの関係述語の上で定義された関数従属の集合からその論理的帰結となるすべての関数従属の集合 (閉包) を得るための健全かつ完全な推論規則の提示に続き, 関数従属と多値従属の集合についての同様な規則が得られ, さらにより一般的な従属性集合についての推論規則とその閉包を求める計算量が調べられた[12)]. しかし, これはデータベース設計の道具としての実用的意味より, むしろ特別な (負リテラルはすべて関係述語をもち, 正リテラルは変数についての等式あるいは関係述語をもつ原子式である) 確定 Horn 規則のある部分クラスに属する式の集合の最小被覆を求める問題と捉えるべきであろう[13)].

スキーマ変換問題は, 古典的にはスキーマ設計のほ

か視野データベースの更新問題，データ・モデル間の変換問題などに関係するが，演繹データベースの環境でも基本的な問題の一つと言えよう．

4. 式の評価と質問

ここでは，データベース・スキーマ S 中の関係述語がすべて基底関係に対応する場合を考え，$[S]$ の要素（基底データベース）が一つ与えられたとする．定数，変数，関数記号，評価可能述語および S 中の関係述語を使って構成された式を与えられたデータベースの上で評価することを考えよう．ただし，閉じた式の評価はその式の真偽値を計算することであり，開いた式の評価はその式に真値を与える式中の自由変数への代入すべてを求めることであるとする．

ところで，任意の述語に対する基礎原子式集合は与えられているが，一般に対象領域 D は陽に与えられていない．したがって，与えられた式の評価はデータベースのみに依存し，対象領域にはかかわらないことが望ましい．この性質をもつとき式は領域独立[14]であるという．ただし，開いた式中のある自由変数に対してどんな代入を施しても真値をとる $P(x)\vee Q(x,y)$ のような式も Nicolas らに倣って領域独立であるということにする．領域独立性は意味論的な性質で，一般には決定不能であることが知られている[15]．領域独立性の十分条件を与える文法的な性質で，しかも十分大きな式のクラスを与えるものとして変域制限がある[16]．変域制限は母式に含意記号を含まない冠頭標準形で与えられた式に対して，

（1） 式中に自由あるいは存在限定された変数 x を含むリテラル L があれば，L 自身正リテラルであるか，あるいは x を含む正リテラル L' が L に連言で結ばれている．このような正リテラルを x の変域リテラルと呼ぶ．

（2） 式中に全称限定された変数 x を含むリテラル L があれば，L 自身負リテラルであるか，あるいは x を含む負リテラル L' が L に選言で結ばれている．このような負リテラルも x の変域リテラルと呼ばれる．

の性質がある．変域制限をみたす式の評価は，各変数の変域リテラルに対する原子式に真値を与える代入について式の残りの部分の真偽値を計算すればよい．前半は基礎原子式集合をしらべればすみ，したがって対象領域には依存しない．

変域制限は式の有限評価可能性を保証するものではない．実際，変域リテラルが評価可能述語をもてば，無限個の基礎原子式が真値をとり得る．変域リテラルを関係述語をもったものに限定すれば，真値をとる基礎原子式の集合の有限性は保証される．関数がある場合には，さらに変数 x の変域リテラル中には x が（少なくとも一つは）裸の引数として（関数の引数としてではなく）現れる必要がある．変域制限にこの二つの条件を追加して関係変域制限と呼ぶことにする．関係変域制限は式の有限評価可能性の十分条件である．

関係変域制限をみたす式のクラスは順序組変数を用いる順序組論理[17]で表されるクラスと一致することが知られている．順序組（レコード）単位の処理が基本となる古典データベース上での検索および更新には，一階述語論理よりも順序組論理ないしはそれに制限を加えた形の関係論理が扱いやすい．

さて，データベースへの質問 Q は一つの開いた式 P と，P 中の自由変数で構成された項のリスト

$$G=(t_1, t_2, \cdots, t_l)$$

の対 $G:P$ として記述される．P を検索式，G を生成リストということにしよう．生成リストに現れた自由変数への代入は有限個でなければならないから，これらはいずれも検索式のすべての連言成分中に自由変数として現れ，したがってその中に関係述語をもった変域リテラルが存在するものでなければならない．たとえば，$Emp\ (u, v, w)$ が“従業員 u の基本給が v，時間外手当単価が w”であることを，$Ot\ (x, y)$ が“従業員 x の当月残業時間が y”であることを表すとすれば，給与計算は質問

$$(u, v, w\times y,\ v+w\times y) : Emp(u, v, w)\wedge Ot(u, y)$$

と書ける．

P 中の自由変数を $x_1, x_2, \cdots, x_n$，そのうち G に現れるものを $x_1, x_2, \cdots, x_m$ $(m\leqq n)$ とするならば，質問 $G:P$ は規則

$$\forall x_1 \forall x_2 \cdots \forall x_m (\exists x_{m+1} \exists x_{m+2} \cdots \exists x_n P \rightleftarrows Q(t_1, t_2, \cdots, t_l)) \quad (4.1)$$

を与えること，つまり述語 Q を定義しているものとみなせる．前例は，

$$\forall u \forall v \forall w \forall y (Emp(u, v, w)\wedge Ot(u, y) \rightleftarrows Pay(u, v, w\times y,\ v+w\times y)) \quad (4.2)$$

で述語 Pay が定義されているといえる．

古典データベース上では与えられた検索式を効率よく評価するアルゴリズムを求めるデータベース探索問題が追求されてきた．検索式を，それと同値であって，しかも効率よい単位操作実行手順を反映する式に文法的に変形し，ここで得られた手順に従って与えら

れた物理表現を利用してそれぞれの単位操作を実行する．データベースの物理表現，つまりファイル構成や索引づけに当たって，いくつかの単位操作を最適化する考慮が払われる．通常，関係述語 R に対して

$$R(x_1, x_2, \cdots, x_{i-1}, c, x_{i+1}, \cdots, x_n)$$

や

$$R(x_1, x_2, \cdots, x_n) \wedge P(x_i, c)$$

($i(1 \leqq i \leqq n)$ はある添字，P はある2変数評価可能述語，c は定数) を効率よく評価する手段が講じられる．これは選択演算に相当する．二つの関係述語 R_1 と R_2 に対して

$$R_1(x_1, x_2, \cdots, x_m) \wedge R_2(y_1, y_2, \cdots, y_{j-1}, x_i, y_{j+1}, \cdots, y_n)$$

($i(1 \leqq i \leqq m)$ と $j(1 \leqq j \leqq n)$ はそれぞれある添字) を効率よく評価する手段が組み入れられる場合もある．これが自然結合演算に対応することはいうまでもない．

さて，

$$\forall x_1 \forall x_2 \cdots \forall x_n (P \rightarrow Q(t_1, t_2, \cdots, t_l)) \qquad (4.3)$$

と，述語 Q をもつ正基礎原子式が存在しないこと（Q は空関係に対応すること）を表す

$$\forall x_1 \forall x_2 \cdots \forall x_m \sim Q(t_1, t_2, \cdots, t_l) \qquad (4.4)$$

をデータベースに加えて得られた規則の集合が充足不能であることを（導出法を用いて）証明し，その副産物として矛盾（空節）を導いた $t_1, t_2, \cdots, t_l$ への代入を求めることによっても質問に答えうる．これは，論理型プログラミングをデータベース検索に用いうることを示している．しかし，規則の集合が充足不能であることを示すこの機構は，データベース（特定のモデル）上の式の真偽値計算を基礎としたデータベース検索とはいくつかの点で異なっている．

論理型プログラミングでも，処理効率向上のためにさまざまな最適化手段が講じられているが，それは導出手順の最適化であって，古典データベースにおける最適化の立場とは異なったものであり，データベース検索言語としては後者に匹敵する効率を期待しえない．また，導出法では(4.1)式の逆方向の含意

$$\forall x_1 \forall x_2 \cdots \forall x_m \exists x_{m+1} \exists x_{m+2} \cdots \exists x_n (Q(t_1, t_2, \cdots, t_l) \rightarrow P) \qquad (4.5)$$

を直接には用いていないため，答として得られた基礎原子式以外の基礎原子式には必ずしも偽値が与えられるとは言えないことに注意を要する．

ところで，基底データベースが整合性制約のモデルであるところから，ある種の質問には整合性制約から（データベースを参照することなく）直接答えうる．たとえば，述語 R, P_1, P_2 (R は関係述語) が与えられ，整合性制約

$$\forall x(\sim P_1(x) \vee \sim P_2(x)) \qquad (4.6)$$

$$\forall x \forall y(R(x, y) \rightarrow P_1(y)) \qquad (4.7)$$

であるならば，質問

$$(x) : R(x, y) \wedge P_2(y)$$

の答は空となる．これは整合性制約に

$$\forall x \exists y(Q(x) \rightarrow R(x, y) \wedge P_2(y)) \qquad (4.8)$$

$$\exists x Q(x) \qquad (4.9)$$

を加え，矛盾を導くことによって証明できる．整合性制約を検索式の変換に利用できる場合もある．たとえば整合性制約

$$\forall x \forall y \forall z(R(x, y) \wedge P_2(y, z) \rightarrow P_1(x, f(z))) \qquad (4.10)$$

が与えられたならば，質問

$$(x, y) : R(x, y) \wedge P_2(y, c)$$

は，

$$(x, y) : R(x, y) \wedge P_1(x, f(c)) \wedge P_2(y, c)$$

と等価である．もし物理表現に当たって

$$R(x, y) \wedge P_1(x, c)$$

の形の式を効率よく評価する手段が与えられているならば，この変換は質問評価の最適化に利用できる．これらの場合をどれだけ一般化できるかは今後の研究に残されている[18]．

5. 仮想関係と演繹

前章に述べたように，(4.1)で与えられる質問は一つの述語 Q を定義しているが，この Q は仮想関係に対応するものであると考えてよい．ところで，ある質問の中の検索式はこうして他の質問で定義された仮想関係を含んでいてもよい．たとえば，仮想関係 Pay が(4.2)で定義されており，質問

$$(q, s) : Pay\ (q, r, s, t) \wedge r \leqq c$$

が与えられたとしよう．これは基本給 c 以上の従業員 q とその当月時間外手当 s を問う質問である．

この質問処理に当たり，まず仮想関係 Pay を定義する質問を評価してその関係を実際に構成し，その上で上述の質問を評価することもできるが，関係 Pay には第2の質問に直接かかわらない基礎原子式（順序組）がふくまれているから，明らかにこの処理は冗長である．(4.2)を用いて原子式 $Pay\ (q, r, s, t)$ をこれと同値な式で置き換えれば，第2の質問を

$$(u, w \times y) : Emp(u, v, w) \wedge Ot(u, y) \wedge v \leqq c$$

と書き換えることができるが，これは通常冗長な基礎

原子式を生成することなく効率よく処理しうる.

上述の書き換えは式の同値変換を利用したものであるが, これを演繹操作に置き換えることができる. 上例では

$$\forall u \forall v \forall w \forall y(Emp(u, v, w) \wedge Ot(u, y) \rightarrow Pay(u, v, w \times y,\ v + w \times y)) \quad (5.1)$$

を一般規則に加え, これとデータベースに

$$\forall q \forall r \forall s \forall t \sim(Pay(q, r, s, t) \wedge r \leqq c) \quad (5.2)$$

を加えて得られた集合が充足不能であることを証明し, その副産物として矛盾をおこした q および s への代入を求めればよい. 演繹機構は仮想関係を含む質問に対してもそれを含まない質問とまったく同じ形で利用できる点で, 古典データベース上の検索操作にない利点をもつものと言える.

古典データベースの検索操作と演繹機構のこの相違は再帰質問の処理に関してさらに明確なものとなる. 再帰質問はある仮想関係の定義に当たってその関係そのものが使われている質問である. たとえば,

$$\forall x \forall y(Parent(x, y) \vee \exists z(Ancestor(x, z) \wedge Parent(z, y)) \rightleftarrows Ancestor(x, y)) \quad (5.3)$$

の中で, 仮想関係に対応する関係述語 *Ancestor* は含意記号の両辺に現れている. 再帰質問で定義された仮想関係は一般に有限であるとは言えない. その有限性を保証するためには, 検索式 (含意記号の左辺) に対して前章で述べた関係変域制限を課すとともに, 生成リスト (含意記号の右辺) を構成する項に対しても制約が必要となる. 生成リストが関数を含まなければ仮想関係が有限であることは明らかであるが, これはより弱い条件で置き換えることができる.

古典データベースの検索言語は再帰質問を定義するステートメントをもたない. 再帰質問は単位操作の手順(プログラム)によって記述され, 実行される. 処理手順には, 再帰質問で定義された仮想関係(探索空間)を横型探索するものと縦型探索するものがある. 前者は, 検索式に含まれた基底関係 (上例では *Parent*) を種として第一次の中間的関係を求め, これと基底関係から第二次の中間的関係を計算する. こうして次々と中間結果を拡大して, それ以上拡大しなくなったらば (最小不動点が得られたならば) これが求める仮想関係である. 後者はバックトラッキングをともなう木の探索で, 順序組 (レコード) 単位の演算を用いて実現できる[19]. データベース上では集合演算を基礎とする横型探索が有利とされているが, 部品展開のように探索空間の構造が問題となるアプリケーションでは縦型探索が望ましい場合もある.

演繹機構は基本的にそのままの形で再帰質問に利用できる. 上例では,

$$\forall x \forall y(Parent(x, y) \rightarrow Ancestor(x, y)) \quad (5.4)$$

$$\forall x \forall y \forall z(Ancestor(x, z) \wedge Parent(z, y) \rightarrow Ancestor(x, y)) \quad (5.5)$$

および

$$\forall x \forall y \sim Ancestor(x, y) \quad (5.6)$$

をデータベースに加えて得られた集合が充足不能であることを示せばよい. もっとも, 論理型プログラミングをそのまま適用すれば, データベース中の同じ基礎原子式を何回も参照したり, 同じ中間結果を何回も生成することになり, よい処理効率は望むべくもない.

演繹データベースの環境では, より一般的な形で規則集合を与える. ここでは, 各規則が質問に対応しているかどうかなどは意識されない. 演繹データベースの (宣言的) 意味は, 規則集合 (内包データベース) に基底データベース (外延データベース) を加えて得られた集合の最小モデルと定義される. 規則が両方向含意によって仮想関係を定義しているものとは限らないから, 得られた質問の答の意味は必ずしも自明でない. この場合特に否定質問にどう答えるかの問題が生じる. また, 選言や不明値 (存在限定された変数と考えることができる.) を含むデータベースや規則集合, つまり非 Horn 規則を含む式の集合をどう扱うかの問題もある. この場合には最小モデルが存在しない. これらについては本特集の他項[20]に詳述される.

最小モデルを求めるにあたって導出法あるいはそれに改良を加えた方法が用いられることが多いが, 検索言語としての効率を上げるために古典データベース上の演算, たとえば集合演算を組み込むさまざまな提案がなされている. 特に, 再帰質問処理の最適化には多くの試みがみられるが, これらについても他項[21]に述べられることになろう.

6. 整合性保証

データベース・スキーマ S と規則集合 $C(S)$ が与えられたとき, データベースが整合していることは, それが $[S, C(S)]$ の要素であることを意味する. そのためには, 基底データベースが整合性制約のモデルであればよい. これは整合性制約が基底データベース更新の正当性検証条件として使われることを示す.

整合している, つまり整合性制約のモデルとなっている基底データベースになんらかの更新が施されたと

き，更新後の基底データベースもまた整合していなければならない．基底データベースの更新ののちすべての整合性制約を調べて，これが偽値をとらないことを確認すれば更新は正当であったことになる．しかし，これは一般に高価な手続きといえる．更新前のデータベースは整合しているから，更新によって影響をうける可能性のある整合性制約のみを調べればよいことは明らかである．さらに，これらの整合性制約を直接調べなくても，整合性制約にある変換を施して得られるより簡単な（変数の少ない）式の価を評価することによって更新の正当性を判定しうる場合がある．

基底関係に1個の基礎原子式を追加したり，基底関係から1個の基礎原子式を削除したりする単位更新について考えてみよう．たとえば領域規則

$$\forall x \forall y \forall z(R(x,y,z) \rightarrow D_1(x) \wedge D_2(y) \wedge D_3(z)) \tag{6.1}$$

と関数従属

$$\forall x \forall y \forall y' \forall z \forall z'(R(x,y,z) \wedge R(x,y',z') \rightarrow y = y') \tag{6.2}$$

が整合性制約として与えられていたとしよう．基礎原子式 $R(a,b,c)$ を追加する単位更新は，

$$D_1(a) \wedge D_2(b) \wedge D_3(c) \tag{6.3}$$

が真値をとり，また

$$\forall y \forall z \sim R(a,y,z) \vee \exists z R(a,b,z) \tag{6.4}$$

が更新前のデータベースで真値をとれば正当となる．これらの式の評価は整合性制約そのものの評価よりはるかに容易である．与えられた整合性制約からこのような式を導くアルゴリズム[22),23)]が知られているが，整合性制約中の全称限定，存在限定の並び方によって，このような式が得られる場合と得られない場合がある．

ある種の整合性制約は単一の単位更新によって必ず破壊される．つまり，基底データベースの一時的不整合をさけえない．単位更新によって生じたこのような一時的不整合は，これにつづくいくつかの単位更新によって修復されなければならない．このため，データベースの環境では，オペレーティング・システムの単純な施錠/解錠機能に比し，はるかに複雑なトランザクション制御機能を必要とする．

演繹データベースの環境では，通常基底データベースの更新のほか一般規則の更新も許す．後者に対しては，整合性制約の定義を多少拡張しなければならない．整合性制約の定義には二つの異なった立場がある．Kowalski ら[24)]は，ある式を規則集合に加えて全体が充足可能でなければならないという条件のあるとき，その式を整合性制約と呼んでいる．一方，Minker ら[25)]は，ある式が規則集合の論理的帰結でなければならないという条件のあるとき，その式を整合性制約と呼んだ．いずれの立場もそれぞれある特別の場合には困難の生ずることが知られており，一種の様相演算子を利用した定義の拡張の試み[26)]などもある．

基底データベースの整合性保証は，更新後の基底データベース上で整合性制約がすべて真値をとることを確認すればよかった．つまり，特定のモデルの上で整合性制約の真偽値を評価すればよい．一方，規則集合に対する整合性制約では，規則集合の充足可能性や，整合性制約が規則集合の論理的帰結であることを証明する手続きが必要である．また，規則集合の更新に当たり，現在の基底データベースが更新後の規則集合と矛盾してはならないという条件を課す場合には，上述の証明過程で規則集合に基底データベース中の基礎原子式を加え，さらに基底関係に対応する述語をもつ基礎原子式で基底データベース中にないものはその否定を加えたものを用いなければならない．いずれにせよ，一般規則の更新の正当性検証は基底データベースのそれに比し，はるかにむずかしい問題を含んでいる．

7. 機密保護

各ユーザにデータベースのどの部分を検索あるいは更新する権利を与えるか，また許されないアクセスに対していかにしてデータを保護するかの問題は重要である．データベース更新に対する保護は比較的容易であるから，以下検索に対する機密保護について考えよう．ユーザの検索権限の与え方には，機密でない領域を指定する方法と秘密そのものを指定する方法がある．古典データベースでは，通常各ユーザに検索を許された基底関係あるいはその一部を指定する．この検索権限は，ユーザにいくつかの（選択）質問

$$(x_{i_1}, x_{i_2}, \cdots, x_{i_m}) : R_k(x_1, x_2, \cdots, x_n) \wedge \Phi$$

を割り当てることである．ただし，R_k は関係述語，Φ は関係述語を含まず $x_1, x_2, \cdots, x_n$ 以外の自由変数を含まない式

$$\{i1, i2, \cdots, im\} \subset \{1, 2, \cdots, n\}$$

である．与えられた質問 $G: P$ の処理に当たり，検索式 P の変形過程で

$$R_k(y_1, y_2, \cdots, y_n) \wedge \Psi$$

（Ψ は関係述語を含まず $y_1, y_2, \cdots, y_n$ 以外の自由変数

を含まない式）がその成分として得られたとしよう．この式は R_k の部分関係を規定しているが，その上で Φ の自由変数 x_i $(i=1,2,\cdots,n)$ を y_i で置き換えて得られた式 Φ' が真値をとる必要がある．部分関係を実際に構成しなくても，たとえば

$$\forall y_1 \forall y_2 \cdots \forall y_n (R_k(y_1, y_2, \cdots, y_n) \wedge \Psi \rightarrow \Phi') \tag{7.1}$$

が証明されればよい．ただし，生成リストG中には

$$\{y_1, y_2, \cdots, y_n\} - \{y_{i1}, y_{i2}, \cdots, y_{im}\}$$

中の変数を含んではならない．

もし整合性制約を知っているならば，ユーザはこれを用いてある種の推論を行うことができる．つまり，上述の機密保護機構は不十分なものとなる．たとえば，検索権限

(x, y, z): $R(x, y, z) \wedge \Phi_1(x)$

(x, y): $R(x, y, z) \wedge \Phi_2(x)$

が与えられたユーザが整合性制約

$$\forall x \forall x' \forall y \forall z \forall z' (R(x, y, z) \wedge R(x', y, z') \rightarrow z = z') \tag{7.2}$$

を知っていたとしよう．このユーザは $\Phi_2(a)$ をみたす基礎原子式 $R(a, b, c)$ について c の価を知る権利を与えられていないが，もし $\Phi_1(a')$ をみたす基礎原子式 $R(a', b, c')$ が存在するならば，c' と整合性制約から $c(=c')$ を知りうる．秘密にされた価を正確に知りえなくてもその価の範囲をある程度しぼれる場合もある[27]．

個々のデータを集約した価を含む統計データベース上の機密保護は，集約関数を含む整合性制約をユーザが知っている場合とみなしてよい．（集約関数を組み込むためには一階述語論理の文法と解釈を少し拡張する必要がある．）個別データは物理表現されていないが，これらは決して検索を許されない基底関係を構成するものと考えられる．集約関数の性質を用いて個別データをある程度の確実さで推定できる場合が多い．この確実さを低く押さえるためにさまざまな手法が考案されてはいるが[28]，有用な結果はほとんど得られていないといってよい．

演繹データベースの環境では，一般規則を使って得られる演繹結果自身機密保護の対象となりうる．これらについてはむしろ秘密そのものを指定すべきであろう．ところで，ユーザの質問に対する答とユーザがすでにもっていると想定される知識から秘密が推論できる場合に質問への解答を拒否するとしても，これには秘密をユーザの知識から導出しうるかどうかの判定を要し，これはもちろん一般に可解とは言えない．さらに，一連の質問への解答とそのうちいくつかについては解答そのものが拒否されたという事実から秘密を推論しうる場合があることも知られている．きわめて強い仮定のもとではこの種の秘密暴露を防ぐ方法が提案[29]されているが，実用上有用な手法とは言い難い．

ユーザが一般規則を知らなければ演繹（統計的推論を含む）を行いえないが，多くの場合ユーザはシステムの外側で一般規則を知り得る立場にある．さらに，ユーザは個別データから一般規則を帰納的に導くかもしれない．インテリジェントなユーザを前提とした機密保護は結局データベース・システムとユーザの智恵くらべの様相を呈し，あまりにも困難が予想されるためかほとんど未開拓の分野といってよい．

8. おわりに

古典データベースの環境では整合性制約や演繹の基礎となる一般規則を直接意識することなく，更新時の正当性検証や仮想関係構成のプログラムとして間接的に扱っていた．一般規則をより直接に意識し形式的議論を展開することにより，多くの問題が厳密に定式化される．これが今日の演繹データベース開発の背景であり，またその枠組の中で解決されるべき問題であろう．

形式化に当たって一階述語論理の枠組を用いた．これは，現在のところもっとも実り多いアプローチと思えるが，やや直観性に欠けるきらいもある．また対象領域の構造や述語系とのインタラクションの問題は扱われていない．

対象指向データベースに関連して述語論理とは異なった枠組を考える動きもある．導出法の適用を許しながら，ある程度高階論理に踏み込む可能性が指摘されている[30),31]が，その評価がすでに定着したものとは言い難い．

草稿を通読し貴重な助言をいただいた NTT 武蔵野通研勝野裕文氏，産能大三浦孝夫氏に謝意を表したい．

参考文献

1) Codd, E. F.: A Relational Model of Data for Large Shared Data Banks, *Commun. ACM*, 13 (6), pp. 377-387 (1970).
2) Date, C. J.: An Introduction to Database Systems, 383 p, Addison-Wesley (1983).
3) Ullman, J. D.: Database and Knowledge-base Systems Vol. 1, 631 p., Computer Science Press

(1989).
4) Kowalski, R.: Predicate Logic as Programming Language, *Proc. IFIP 74*, pp. 569-574 (1974).
5) Kowalski, R.: Logic for Problem Solving, 287 p., North-Holland (1979).
6) Gallaire, H., Minker, J. and Nicolas, J.-M.: Logic and Databases: A Deductive Approach, *ACM Comp. Surveys*, 16 (2), pp. 153-185 (1984).
7) Chang, C. L. and Lee, R. C. T.: Symbolic Logic and Mechanical Theorem Proving, 331 p., Academic Press (1973).
8) 三浦孝夫, 塩谷 勇: 複合オブジェクトに基づく演繹データベース, 情報処理, Vol. 31, No. 2, pp. 206～215 (1990).
9) Kobayashi, I.: Losslessness and Semantic Correctness of Database Schema Transformation: Another Look of Schema Equivalence, *Information Systems*, 11 (1), pp. 41-59 (1986).
10) Kobayashi, I.: Classification and Transformations of Binary Relationship Relation Schemata, *Informations Systems*, 11 (2), pp. 109-122 (1986).
11) Kobayashi, I.: Temporal Aspect of Databases: Introduction between State and Event Relations, *Proc. IFIPTc 2.6 Workshop '86*, pp. 223-232 (1986).
12) 上林弥彦: データベースの基礎理論 (3): 従属性理論, 情報処理, Vol. 23, No. 9, pp. 835-847 (1982).
13) Fagin, R.: Horn Clause and Database Dependencies, *J. ACM*, 29 (4), pp. 952-985 (1982).
14) Kuhns, J. L.: Answering Questions by Computers—A Logical Study, *Rand Memo RM 5428 PR*, Rand Corp. (1967).
15) DiPaora, R. A.: The Recursive Unsolvability of the Decision Problem for the Class of Definite Formulas, *J. ACM*, 16 (2), pp. 324-327 (1969).
16) Nicolas, J.-M.: A Property of Logical Formulas Corresponding to Integrity Constraints on Database Relations, *Proc. of the Workshop on Formal Bases for Data Bases* (1979).
17) Kobayashi, I.: Tuple Calculus: Syntax and Semantics, to appear in *Proc. 2nd Scandinavian-Japanese Seminar on Information Modeling and Knowledgebases* (1989).
18) Sundin, U.: Some Aspects of Using Integrity Constraints in Knowledge-Based Systems, to appear in *Proc. 2nd Scandinavian-Japanese Seminar on Information Modeling and Knowledgebases* (1989).
19) Vielle, I.: Recursive Axioms in Deductive Databases: The Query-Subquery Approach, *ECRC Internal Rep.*, KB-10 (1986).
20) 勝野裕文: 演繹データベースの形式的意味論, 情報処理, Vol. 31, No. 2, pp. 198～205 (1990).
21) 宮崎収兄, 世木博久: 演繹データベースの問合せ処理, 情報処理, Vol. 31, No. 2, pp. 216～224 (1990).
22) Nicolas, J.-M.: Logic for Improving Integrity Checking in Relational Databases, *Acta Informatica*, 1 (18), pp. 227-253 (1982).
23) Kobayashi, I.: Validating Database Updates, *Informaton Systems*, 9 (1), pp. 1-17 (1984).
24) Kowalski, R.: Logic for Data Description, H. Gallaire, and J. Minker, eds., Logic and Data Bases, pp. 77-103, Plenum Press (1978).
25) Nicolas, J.-M. and Gallaire, H.: Database: Theory vs. Interpretation, H. Gallaire and J. Minker, eds., Logic and Data Bases, pp. 33-54, Plenum Press (1978).
26) Reiter, R.: On Integrity Constraints, *Theoretical Aspects of Reasoning about Knowledge*, pp. 97-111 (1988).
27) Bancilhon, F. M. and Spyratos, N.: Protection of Information in Relational Data Bases, *Proc. VLDB '77*, pp. 494-500 (1977).
28) Denning, D. E. and Denning, P. J.: Data Security, *ACM Comp. Surveys*, 11 (3), pp. 227-249 (1979).
29) Sicherman, G. L., De Jonge, W. and Van de Riet, R. P.: Answering Queries without Revealing Secrets, *ACM Trans. Database Systems*, 8 (1), pp. 41-59 (1683).
30) Maier, D.: A Logic for Objects, *Proc. of the Workshop on Foundation of Deductive Databases and Logic Programing*, pp. 6-26 (1986).
31) Kifer, M. and Warren, D. S.: A Logic for Object Oriented Logic Programs, *PODS* (1989).

(平成元年10月12日受付)