

解説

データベースの知的アクセス*

古川 康一**

1. はじめに

データベースの利用が一般に普及するに従って、エンド・ユーザは non-programmer へと拡大されることが予想されるが、そのような状況では、ユーザと計算機システム間の円滑な情報交換が特に重要となる。この問題は、従来「人間-機械インタフェースの改善」と言われてきた問題そのものであり、以前からその重要性および解決の困難性が指摘されてきた問題である。

人間-機械インタフェースはこれまでに、入出力情報の形態の多様化、会話型利用などに見られる計算機の利用形態の改善、情報のやりとり使用するプログラミング言語やコマンド言語などの高水準化等、利用技術の進歩によって著しく改善されてきた。しかしながら、そのインタフェースは、人間同士のコミュニケーションの仕方に比べると、未だ原始的な段階に止まっているのが現状である。人間同士のコミュニケーションでは、人に仕事を頼む場面を考えれば分かるように、常に相手の理解度を確かめながら自分の意図を相手に伝達しようと努める。同じ問題について、各人は多少とも異なったイメージを持っているかも知れないが、意図が思うように通じないときには、言葉の定義に始まって、言い換えや、図示などのあらゆる手段を用いてイメージの統一を図る。この際、相手も同等の知的能力を持っていて、手持ちの知識をよりどころにイメージの統一のために同じ努力をしている。

人間-機械インタフェースを本質的に改善するためには、上に述べたような人間同士のコミュニケーションに見られる知的活動を、人間-機械間に持ち込むことが不可欠である。Frijda, N. H.⁷⁾ は、人間のように振る舞うシステムの記憶（すなわちデータベース）が備えるべき機能を4つ挙げている。第1は、連想的構

造である。人間の脳における情報検索は、手掛りとなる情報を基にして、それに類似のものや、それから連想されるものを思い起こすことによってなされる。第2は、データベースを教育できることである。すなわち、システムは、情報を受け取ったときに、その真偽を判定できること、および真の場合には、それをデータベースに追加できることが必要である。第3は、推論機能である。システムはデータベースに直接置かれていないことでも、推論によって答を得られなければならない。第4は、情報が入力された時に作られた構造と異なる構造を持った検索要求を処理できることである。

これらの諸機能を実現するための、さらに基本的な機構は、データおよびその表現に関する知識に基づく意味処理である。データの意味の理解は、上の第2および第3の機能の実現に役立つ。データ表現に関する知識の利用は、データの意味処理と合わせて、上の第1および第4の機能の実現に役立つ。

以下では、2でデータベースの知的アクセス処理の概観を示し、3でデータモデルと意味表現の関連について述べる。4では、ユーザの視点である外部スキーマと、システムの立場を表わす概念スキーマのすり合わせについて、いくつかの場合に分けて論じる。5では、データ表現の知識に基づく検索手続きの自動作成と、最適化について述べる。最後に、今後の研究動向などについてふれる。

2. 知的アクセス処理の概観

データベースに対する知的アクセス実現にとって、人間-機械インタフェースの改善がいかに大切であるかは前節で述べたが、それはデータベースの分野で強調されているデータ独立の考えと密接に結びついている。データベース・システムにおいて、効率に対する要求や新デバイスの出現等によって記憶構造 (Internal Schema, 内部スキーマ) を変更したい場合が

* Intelligent Access to Database by Koichi FURUKAWA (Computer Science Division, Electrotechnical Laboratory).

** 電子技術総合研究所ソフトウェア部

あるが、もしそれに伴って応用プログラムの変更も必要であるとしたら、その手間は膨大なものと予想され、内部スキーマの変更は困難となる。これを回避するために導入されたのが、応用プログラムをデータの内部スキーマから独立させる、データ独立の考え方である。そのためには、データの内部スキーマとは独立に、より抽象化した概念に基づくデータの構造 (**Conceptual Schema**, 概念スキーマ) を設定し、応用プログラムはその概念スキーマを操作する命令によって記述する方式とすればよい。

アメリカ規格協会の ANSI/X3/SPARC グループによって書かれた報告書¹³⁾では、「各ユーザが、システムの持つ概念スキーマとは別に、独自のデータの把握の仕方 (**External Schema**, 外部スキーマ) を設定できる」機能を提案しているが、図-1 に示すように、これは上に述べたデータ独立の考えを一歩進め、「ユーザの質問言語を、システムの概念スキーマから独立させる」ことを目指しているものと言える。Abrial, J.R.¹⁴⁾ は、これを意味上のデータ独立 (**semantic data independence**) と呼んでいる。これら2種類のデータ独立化の機構により、人間-機械インタフェースは著しく改善される。

データベースに対する知的アクセスは、この2つのデータ独立化を中心にして実現される。それは、

- (1) ユーザによる質問文の入力、
 - (2) ユーザ質問文と概念スキーマとのすり合わせ、
 - (3) 検索手続きの自動作成と最適化、
 - (4) 検索手続きの実行と結果の表示
- の各部分から成る。ここで、大雑把に言えば(2)がユーザの質問言語のシステムの概念スキーマからの独立化に対応し、(3)が、応用プログラムの内部スキーマからの独立化に対応している。(1)および(4)では、入出力の形態の改善が主要な問題となる。入力、究極

的には自然言語や図面等の人間同士のコミュニケーションでの情報変換手段を目指すべきであるが、QBE²⁵⁾などのグラフィック・ディスプレイをうまく利用した質問言語に見られるように、人間工学的な工夫が役立つ場面も多く、短期的に見ればそれらのアプローチの方が実用的であろう。

以下では、処理の(2)および(3)に焦点をあてて述べることにする。

3. データモデルと意味表現

従来、データベース研究者は、概念スキーマの記述言語としていくつかのデータモデルを提案してきた。その代表的なものは、階層モデル、ネットワーク・モデルおよび関係モデルである。

階層モデルは、歴史的には最も古いもので、IBM社のIMS (Information Management System)^{4), 24)}がこの考えに基づく代表的なシステムである。このモデルの基本的な考えは、会社の組織などに見られる階層構造をそのままモデル化し、計算機に情報を蓄えることである。そして、その階層構造に沿った情報のアクセスを可能にすることである。

ネットワーク・モデルは、CODASYLのDBTG (Data Base Task Group) によって提案されたもので、階層構造で表現し切れない網構造の表現を可能にするものである^{4), 22)}。

これら2つのデータモデルは、効率化のためにデータの取り出し経路 (**Access Path**, アクセス・パス) を指定する機能を重視していたが、関係モデル^{4), 12)}はアクセス・パスの考えを一切含んでいない。Earley, J.⁵⁾ は、概念スキーマのレベルを、関係レベルとアクセス・パス・レベルに2分することを提案している。アクセス・パス・レベルは、データの取り出し方まで指定する点で、関係レベルに比べてより内部スキーマに近いと言えるであろう。

概念スキーマの役割りを

- (1) 外部スキーマに基づく質問言語のターゲット言語、

- (2) 応用プログラムの記述言語

の2つに分けて考えると、階層モデルおよびネットワーク・モデルは、もともと(2)を目的として考え出されたもので、データベース管理者 (DBA) にとっての強力なプログラミング言語を提供している。一方、関係モデルは、データ間の論理的な関係を基にした抽象度の高いモデルで、むしろ(1)の役割りを果すものと

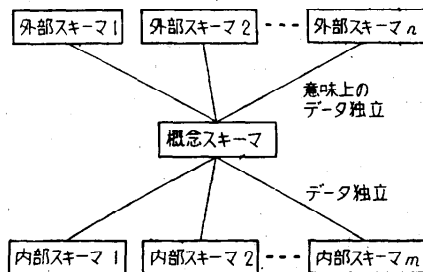


図-1 ANSI/X3/SPARCによる3層のスキーマ

して適当であると思われる。

ここで、概念スキーマの(1)の役割について、もう少し詳しく述べよう。外部スキーマから概念スキーマへの変換は、質問文の担っている意味を保存する形で行われる。このため、概念スキーマは、データベースを構成する個々のデータおよびそれらの関係の意味を記述しているものでなければならない。

データの意味の記述およびその処理は、人工知能の分野で知識表現のために開発されたいくつかのシステムにより可能である。良く知られているものは、一階述語論理システム、セマンティック・ネットワーク、およびフレーム表現である。述語論理は、もともと事実に基づく論証の数学的な基礎理論として発達したもので、定理の証明システムやプログラムの検証システム等に応用されている。また、述語論理と関係モデルを結びつけた理論を土台にしたデータベースの質問応答システムも開発されている^{2), 6), 16)}。セマンティック・ネットワークは、述語論理に比べて、図式的で理解し易く、しかもアクセス・パスを指定できるので、各種の意味処理システムに使われてきた。それによる概念スキーマの記述の例もある¹⁸⁾。フレーム表現は、対象世界に存在する各対象物の詳細な記述を、標準モデルと、それからの差異の形で表現し、常識に基づく推論 (Common Sense Reasoning) のための機構を提供している。

これらは、いずれも概念スキーマの表現に止まらず、外部スキーマの表現および、そこから概念スキーマへの変換過程をも含んだ意味情報処理システムとなっている。

4. 外部スキーマと概念スキーマとのすり合わせ

外部スキーマと概念スキーマの相違には、データモデルが異なる場合と、同じデータモデルでも構造自身異なる場合の2通りがある。

はじめに、外部スキーマと概念スキーマのデータモデルが異なる場合について考えよう。外部スキーマの方が概念スキーマよりも抽象度が低い場合には、その間の変換ができない部分が生じる。たとえば、外部スキーマがネットワーク・モデルで概念スキーマが関係モデルの場合、利用者の質問文中に含まれるレコードの順序の指定 (例えば、ある集合の“一番目の”レコードを取り出せ、など) は、関係モデルには対応する概念がないので、処理が出来ない。また、単純な階層

```
GET 'relation1 WHERE (rell-qual)
AND (join-qual)
AND (rel2-qual)'
```

(a) 外部スキーマの DML のパターン

```
DEFINE REL1
FROM t (relation2) WHERE t (rel2-qual)
LINK WITH lq
TO t (relation1) WHERE t (rell-qual)
KEEP ALL
```

CREATE REL 1

(b) 対応する概念スキーマの DML のパターン

図-2 関係モデルによる外部スキーマを概念スキーマに変換する変換規則の例。t(x) は、x に対応する概念スキーマ上の名前。lq は、join-qual に対応するリンク名

モデルのように、多対多の関係を表現できないものによって概念スキーマを記述すると、多対多の関係を含んだ外部スキーマは、概念スキーマに変換できない。Klug, A. 等¹⁵⁾は、ネットワーク・モデルを拡張したデータモデルにより概念スキーマを記述し、関係モデル、ネットワーク・モデル及び階層モデルによる外部スキーマを、この概念スキーマに変換する方法を提案している。それは、外部スキーマのデータ操作言語 (Data Manipulation Language, DML) の各文を、図-2に示すような、あらかじめ用意された変換規則によって、対応する概念スキーマの DML の文系列に変換する方法を採っている。しかしながら、この方法が適用できるための条件はかなり厳しく、スキーマ間の強い類似性が要求されている。さらに、この変換過程は、パラメータを単純に鋳型へはめ込むだけではすまされず、パラメータ間の関係を見たり、階層構造を辿ったりする処理を含んだ、かなり複雑なものとなっている。このようにこの方法があまりうまく行かないのは、検索要求の意図を考慮に入れていないからである。Falkenberg, E.⁶⁾は、異なるデータモデル間のデータベースの変換を行うのに、そのデータベースを、アクセス・パスを含まないより抽象的なレベルに一度変換して、それから相手のデータモデルへ変換する2段階の方法を提案しているが、同じ方法を上の問題に対して適用することが可能であろう。

つぎに、外部スキーマ、概念スキーマ共、同一のデータモデルで記述されていて、それらの構造が異なる場合について述べよう。ここでは、特にそのデータモデルが関係モデルの場合を考える。

例として、ホテルの予約業務について考えてみよう。図-3(次頁参照)に概念スキーマの例を示す。これは、3つの関係 \$VACANCY\$, \$ROOM\$ 及び \$TYPE\$ から成る。これに対して、図-4 (次頁参照) に示すよ

\$ VACANCY (空き日, 部屋番号)
 \$ ROOM (部屋番号, タイプ)
 \$ TYPE (タイプ, 定員, 一般料金, 会員料金)

図-3 予約管理データベース概念スキーマの例

STATUS (日, 部屋番号, 予約状況)
 ROOM (部屋番号, 定員, 料金)

図-4 予約管理データベースの外部スキーマの例

うな外部スキーマを考えよう。関係 STATUS は、関係 \$ VACANCY に対応するが、情報の表現形式が異なり、ある日にある部屋が予約されている (占) か空いている (空) かの情報が与えられていると考ええる。関係 ROOM は、関係 \$ ROOM と \$ TYPE を一緒にしたものと考えられる。

外部スキーマと概念スキーマとの対応づけは、意味上の等価性によってなされることはすでに述べたが、そのためには、各関係の意味を明確に表わす必要がある。いま、関係 ROOM のある組 (部屋番号: x , 定員: y , 料金: z) について考える。この組は、特定の ROOM を表わしている。このような特定の対象物をエンティティと呼ぶ。そして、そのエンティティは、関係 ROOM の要素と考えられる。いま、このエンティティを i で表わすと、この組は「ROOM i は、部屋番号が x で、定員が y で、料金が z である」ことを意味している。これは、つぎの3つの事実

ROOM i は、部屋番号が x である。

ROOM i は、定員が y である、

ROOM i は、料金が z である

をまとめて表わしたものである。これらの各事実を、簡単に、ROOM/ i /部屋番号/ x /, ROOM/ i /定員/ y /, ROOM/ i /料金/ z / と表わそう。これらは、述語論理での2項述語となっている。

つぎに、関係 \$ ROOM の組 $j = \langle$ 部屋番号: s , タイプ: $t \rangle$ および \$ TYPE の組 $k = \langle$ タイプ: u , 定員: v , 一般料金: w_1 , 会員料金: $w_2 \rangle$ について考えよう。これらの組の意味は、同様に2項述語によって表現できるのは明らかである。

ここで、 $i = j$ とすると、ROOM の要素 i と \$ ROOM の要素 j は全く同じ部屋を表わしていることになるので、その部屋番号は等しく、 $s = x$ が成り立つ。この関係は、次のような論理式によって表わされる。

$$(\forall i)(\forall x)(\text{ROOM}/i/\text{部屋番号}/x/ \\ \equiv \$ \text{ROOM}/i/\text{部屋番号}/x/) \quad (1)$$

このように、外部スキーマの述語に対して概念スキーマ内に1対1に対応する述語が存在する場合を、単純な場合と呼ぼう。

つぎに、部屋の定員について考えると、それは、その部屋のタイプを表わす関係 \$ TYPE の組の定員と対応していることが分かる。すなわち、

$$(\forall i)(\forall x)(\exists j)(\exists y) \\ (\text{ROOM}/i/\text{定員}/x/ \\ \equiv \$ \text{ROOM}/i/\text{タイプ}/y/ \\ \wedge \$ \text{TYPE}/j/\text{タイプ}/y/ \\ \wedge \$ \text{TYPE}/j/\text{定員}/x/) \quad (2)$$

が成り立つ。このような場合を推移的な場合と呼ぼう。

部屋の料金については、予約者が会員かどうかによって料金が異なることを表現しなければならない。いま、会員 (i, k) を、部屋 i の予約者 k が会員であるときに真となり、そうでないときに偽となる述語とすると、述語 ROOM/ i /料金/ x / は、つぎのように定義される。

$$(\forall i)(\forall x)(\exists j)(\exists y) \\ (\text{ROOM}/i/\text{料金}/x/ \\ \equiv \$ \text{ROOM}/i/\text{タイプ}/y/ \\ \wedge \$ \text{TYPE}/j/\text{タイプ}/y/ \\ \wedge (\text{IF 会員}(i, k) \\ \text{THEN } \$ \text{TYPE}/j/\text{会員料金}/x/ \\ \text{ELSE } \$ \text{TYPE}/j/\text{一般料金}/x/)) \quad (3)$$

このような場合を条件的な場合と呼ぼう。

同様の方法によって、関係 STATUS の各述語を、関係 \$ VACANCY の述語により定義できるが、関係 STATUS の '日' あるいは '部屋番号' は、その予約状況が '空' の場合にのみ、\$ VACANCY に対応するエンティティが存在するので、そのような制限つきの定義となる。すなわち、

$$(\forall i)(\forall x)(\text{STATUS}/i/\text{日}/x/ \\ \wedge \text{STATUS}/i/\text{予約状況}/\text{'空'}/ \\ \equiv \$ \text{VACANCY}/i/\text{空き日}/x/) \quad (4)$$

$$(\forall i)(\forall y)(\text{STATUS}/i/\text{部屋番号}/y/ \\ \wedge \text{STATUS}/i/\text{予約状況}/\text{'空'}/ \\ \equiv \$ \text{VACANCY}/i/\text{部屋番号}/y/) \quad (5)$$

もし予約状況が '占' ならば、対応する (部屋番号, 空き日) の組は存在しない。このことは次式により与えられる。

$$(\forall i)(\forall x)(\forall y) \\ (\text{STATUS}/i/\text{日}/x/ \\ \wedge \text{STATUS}/i/\text{部屋番号}/y/ \\ \wedge \text{STATUS}/i/\text{予約状況}/\text{'占'}/ \\ \rightarrow \sim (\exists j)(\$ \text{VACANCY}/j/\text{空き日}/x/$$

Λ\$ VACANCY/j/部屋番号/y/) (6)

このような場合を、制限的な場合と呼ぼう。

以上のように、外部スキーマの各述語と概念スキーマの述語との関連の仕方がいくつかの場合に分類できることを見てきたが、ここでそれぞれの場合がどのような時に起こるかをもう少し考えてみよう。

推移的な場合は、概念の階層構造において、上位概念の属性が下位概念のエンティティに伝播する場合に相当する⁹⁾。‘タイプ’は‘部屋’の上位概念であり、その属性である定員は‘部屋’に伝播する。しかしながら、必ずしも上位概念のすべての属性が下位概念に伝播する訳ではない。例えば、あるタイプに属する部屋の総数はその‘タイプ’の属性と考えられるが、それは下位概念である‘部屋’の属性ではない。このため、概念の階層化を行っても、どの属性がどこまで伝播するかを、はっきりと指定しておく必要がある。

条件的な場合には、例外処理のように記述方法が一樣でない場合と、内容が条件的な場合とがある。一樣でない記述の仕方を許すと、たとえば、データとデータへの参照（ポインタ）を混在させることが可能となり、アクセス・パスの取り扱いが可能となる。

制限的な場合は、外部スキーマの方が概念スキーマよりも細分化されていて、情報量が多いときに起こる。概念スキーマが冗長な場合にも、その冗長性を利用して最適化を図る場合には、制限的な述語の定義が必要となる。それは、(2)式の右辺と左辺を入れ換えたようなものである。

質問文の変換は、述語間の関連が単純あるいは推移的な場合のみならば、Stonebraker, M. の方法²¹⁾が使えるが、条件的、あるいは制限的な場合には、冗長性の除去の手続きや場合分けの処理などが必要となり、述語論理の演繹システムによって統一的に処理することが必要となる⁸⁾。

5. 検索手続きの自動作成と最適化

概念スキーマ上の非手続き的質問文が得られたなら、つぎに、それを満足する答を引き出すことになるが、ここでその質問文を解釈実行するインタプリタ方式²⁰⁾と検索手続きに変換する自動プログラム作成方式^{8), 9), 10), 17)}とが考えられる。一般に対話的システムでは、質問文はその都度作るので、インタプリタ方式で十分のように思えるが、コンパイル処理および最適化によって得られる実行時間の短縮を考慮すると、むしろ後者の方が全体としての効率が良い場合が多いと思

われる。また、他の多くの利点も指適されている^{9), 10)}。以下では、自動プログラム作成方式について述べよう。

Gerritsen, R.⁹⁾ は、HI-IQ (Hierarchical Interactive Query) 言語で書かれた非手続き的質問文を、CODASYL の DBTG (Data Base Task Group) が提案したネットワーク・モデルに基づく DML による検索手続きに変換するシステムを作製した。彼のシステムは、Buchanan, J. R. および Luckham, D. C. によって開発された自動プログラム作成システム APG の上に作られている。APG は一階述語論理に基づいて、論理的にプログラムを作成するもので、μPLANNER の演繹機能を用いて作られている。Gerritsen のシステムはプログラミングに関する一般的な推論規則の他に、個々の命令の定義を与える規則や、その他多くのプログラミング技法に関する知識を持っている。命令の定義は Hoare, C. A. R.¹¹⁾ によって導入された記法 $P\{A\}Q$ を用いて書かれている。ここで、 P, Q は論理式で、 A はプログラムあるいはその一部分である。 $P\{A\}Q$ は、もし A の実行前に P が真で、 A の実行が終了すると、その時点で Q が真であることを意味している。 P は、 A を実行するために必要な前提条件で、 Q は A を実行して得られる結果である。たとえば、与えられたキーを持つレコードを見出す命令 FIND に対する規則は、

$$\begin{aligned} &(\text{HASHKEY } r1 \ k1)(\text{DESIRE } k1 \ v1) \\ &(\text{CONTAINS } k1 \ v1) \\ &\{\text{FIND } r1\}(\text{CURRENT } r1) \end{aligned} \quad (7)$$

となる。この規則は、 $k1$ がレコード $r1$ のハッシュ・キーで、 $k1$ の値が $v1$ であるレコードを見出したいなら、 $k1$ の値を $v1$ として FIND $r1$ を実行すればよいことを示している。ここで CURRENT は DBTG の概念で、対象となるレコードがその時点でデータベース管理システムによってアクセス可能な状態にあることを意味する。

同様に、COBOL の MOVE 命令に対する規則は、
(ISVAR $v1$)(ISITEM $d1$)

$$\{\text{MOVE } v1 \text{ TO } d1\}(\text{CONTAINS } d1 \ v1) \quad (8)$$

と定義される。ここで (ISVAR $v1$) は、 $v1$ が変数かどうかを判定する論理式で、(ISITEM $d1$) は、 $d1$ がレコードの属性であるかどうかを判定する論理式である。

つぎの規則は、命令の定義を与えるものでなく、作

成するプログラム全体の構成を与えるもので、実際にはプログラム作りの最上位における制御に使われる。

```
(OPENED a1)(CURRENT r1)
(LINKED r1 r2)(CLOSED a1)
(STOP x1)→(PROGRAM x1)    (9)
```

つぎにこのシステムがどのようにして、プログラムを作り上げていくかをみてみよう。例として、レコード STUDENT のハッシュ・キーが SNUM であるときに、学生番号 (SNUM) が 126 に等しい学生に関する情報を求められたとする。すると、初期状態で、つぎの2つの事実が成り立つ。

```
(HASHKEY STUDENT SNUM)
(DESIRED SNUM 126)
```

μ PLANNER に対して、(PROGRAM EXAMPLE) が、ゴールとして与えられる。すると、規則(9)が、起動されて、(OPENED a1) が最初のサブゴールとして設定される。このサブゴールが満足されたら、次に (CURRENT r1) がサブゴールとなるが、このサブゴールは、FIND に対する規則(7)の結果と一致するので、その規則が呼ばれる。FIND の規則が設定するサブゴールのうち最初の2つは、初期状態の2つの事実により真となり、変数 r1, k1, v1 には、それぞれ STUDENT, SNUM および 126 が結合 (bind) される。第3のサブゴールは、MOVE に対する規則(8)の結果によって満足されその前提条件が真であることがわかるので、MOVE 126 TO SNUM がプログラムの中に置かれる。また、この結果、FIND の前提条件がすべて満たされるので、FIND STUDENT もプログラム中に置かれる。この時点でのプログラムは、

```
(Block to open files)
MOVE 126 TO SNUM
FIND STUDENT
```

となる。

このシステムは、人工知能の手法によって実際的なプログラムを作り出すことの出来る数少ないシステムの1つであり、その成果が注目される。

この他 Haseman, W. D. 等¹¹⁾は、同様の考え方に基づいて、階層モデルの質問言語を DBTG プログラムへ変換するシステム GPLAN を作成した。そこでは、ユーザの質問文に答えるために、データベースを必要に応じてその都度再構成する方法を用いている。

関係モデルの非手続き言語である関係論理による質問文を、関係代数による手続き表現に変換する方式も報告されている^{3), 6)}。

最後に、最適化の問題について簡単にふれよう。最適化は、質問文を外部スキーマから内部スキーマへ変換する一連の過程の各段階ごとになされる。外部スキーマから概念スキーマへの変換の際には、冗長性が除去されるが、これも一種の最適化である。概念スキーマレベルで非手続き的質問文から検索手続きに変換する際には、逆引きインデックスの存在等を考慮してデータのアクセス順序を決める必要がある。また、アクセス・パスが2通り以上ある場合には、その中から最も効率の良いパスを選ぶのも最適化の仕事である。

6. おわりに

non-programmer にとって真に使い易くしかも実用に耐えるデータベースシステムを実現するためには、データベースの知的アクセスをさらに追求することが必要である。人間と計算機間の概念のすり合わせをより一層なめらかにするためには、対話処理によって質問者の概念構造を把握し、システムの持つ概念構造に結びつけることが必要である。Chang, S. K. 等³⁾の研究は、ごく初歩的なものであるが、データの意味の詳細な記述¹⁾を基にして、ユーザの与える不完全な質問文から完全な質問文を得るシステムを実現している。

これまでの話は、入力質問文の処理に集中していたが、質問文に対する答え方も、人間-機械インタフェースを改善する上で、重要な役割を担っている。人間工学的なレベルでは、答えの表による表示、階層表示、グラフ表示などが有用である。より本質的なレベルでは、人間同士の会話に見られる近似的応答 (approximate responses) を計算機の応答に採り入れることが考えられている¹⁴⁾。その目的は、次の通りである。(1)ユーザの初めの質問文に対する厳密な答えが希望していた答でなかったり、有用な情報でなかった場合に、望むべき答を与えてくれるような質問文を作る手助けを行う。(2)ユーザがデータベースに関する認識が浅い場合には、その構造や中身について知らせる。(3)不必要な細部の情報を無視して、適当なレベルで、要約して答える。

そのために、ユーザの質問文の中に隠されている暗黙の仮定 (presupposition) に注目する。文 S に対する暗黙の仮定とは、S が意味ある文章となるために必要な、あらゆる事柄であると考えてよい。質問文における暗黙の仮定は、厳密な答えが意味あるためには、真でなければならない。もし、検索要求の処理過程のある段階で結果として空集合を生じたとき、それは対

応する暗黙の仮定が真でなかったと考える。その場合、ただ単に“該当する項目なし”と答える代りに、その原因に応じて、適切な理由を付して答えることができるであろう。

答の要約の例は、「利潤分配を担当している従業員は誰か?」という質問文に、具体的な人名（それは、10,000 名いるかも知れない!）を表示する代りに、「すべての副社長」と答えた方が気が利いているであろう。この要約は得られたデータから計算して求めたり、あるいは、データを全く参照せずに、推論によって求める方法も考えられる。

人工知能の分野では、データベースの知的アクセスの問題は、大規模かつ複雑な知識のデータベース化¹⁷⁾、その上での効率良い推論システムの実現^{8), 16), 17)}、および自然言語による質問応答の実現の問題として捉えられており、それを統合したプロジェクトも推進されている^{19), 23)}。

データベース・マシンに見られるハードウェア技術の進歩と、人工知能研究とデータベース研究の協調を一層強めることによって、真に知的なデータベース・システムの近い将来における実現が大いに期待できるであろう。

参 考 文 献

- 1) Abrial, J.R.: Data Semantics, in Database Management (J.W. Klimbie 他 Ed.), North-Holland (1974).
- 2) Chang, C.L.: Deduce—A Deductive Query Language for Relational Databases, in Artificial Intelligence and Pattern Recognition (C.G. Chen Ed.), Academic Press (1976).
- 3) Chang, S.K. & Ke, J.S.: Database Skeleton and its Application to Fuzzy Query Translation, Technical Report, Medical Information Systems Laboratory, Department of Information Engineering, University of Illinois at Chicago Circle (1976).
- 4) Date, C.J.: An Introduction to Data Base Systems, The Systems Programming Series, Addison-Wesley (1975).
- 5) Earley, J.: On the Semantics of Data Structures, in Data Base Systems (R. Rustin Ed.), Prentice-Hall (1972).
- 6) Falkenberg, E.: Coexistence and Transformation of Data, Proc. Third VLDB (1977).
- 7) Frijda, N.H.: Simulation of Human Long-term Memory, Psychol. Bull., Vol. 77, pp. 1~31 (1972).
- 8) Furukawa, K.: A Deductive Question Answering System on Relational Data Bases, Proc. Fifth IJCAI (1977).
- 9) Gerritsen, R.: The Application of Artificial Intelligence To Data Base Management, Proc. Fourth IJCAI (1975).
- 10) Haseman, W.D. and Winston, A.B.: Problem Solving Approach in Data Management, Proc. Fourth IJCAI (1975).
- 11) Hoare, C.A.R.: An Axiomatic Basis for Computer Programming, Comm. ACM, Vol. 12, No. 10, pp. 576~580 (1969).
- 12) 穂鷹良介, 渋谷政昭: データベースの関係形式, 情報処理, Vol. 17, No. 10, pp. 904~910 (1976).
- 13) Interim Report ANSI/X3/SPARC Study Group on Data Base Management Systems, "FDT" 7, No. 2, ACM New York (1975).
- 14) Joshi, A.K. 他: Approximate Responses from a Data Base Query System: An Application of Inferencing in Natural Language, Proc. Fifth IJCAI (1977).
- 15) Klug, A. & Tsichritzis, D.: Multiple View Support within the ANSI/SPARC Framework, Proc. Third VLDB (1977).
- 16) Minker, J.: Search Strategy and Selection Function for an Inferential Relational System, ACM Trans. on Database Systems, Vol. 3, No. 1, pp. 1~31 (1978).
- 17) Ohsuga, S.: Semantic Information Processing in Man-Machine Systems, Proc. 1977 IEEE Conf. on Decision & Control (1977).
- 18) Roussopoulos, N. & Mylopoulos, J.: Using Semantic Networks for Data Base Management, Proc. First VLDB (1975).
- 19) Sacerdoti, E.D.: Language Access to Distributed Data with Error Recovery, Proc. Fifth IJCAI, pp. 196~202 (1977).
- 20) Sagalowicz, D.: IDA: An Intelligent Data Access Program, Proc. Third VLDB, pp. 293~302 (1977).
- 21) Stonebraker, M.: Implementation of Integrity Constraint and Views by Query Modification, Proc. 1975 SIGMOD Workshop on Management of Data (1975).
- 22) 植村俊亮: CODASYL 方式のデータベース・システム, 情報処理, Vol. 17, No. 10 (1976).
- 23) Waltz, D.L.: Writing a Natural Language Data Base System, Proc. Fifth IJCAI (1977).
- 24) 渡辺純一: 階層構造のデータベース, 情報処理, Vol. 17, No. 10 (1976).
- 25) Zloof, M.M.: Query by Example, Proc. AFIPS 1975 NCC, Vol. 44, pp. 431~437 (1975).

(昭和 53 年 6 月 2 日受付)