解　説

データベースへの知的アクセス†

古　川　康　一††*

1. はじめに

データベースは，情報の入れ物として，机の引き出しの中に置かれたファイルに取って代わり，銀行での預金管理，企業での人事資料，製造業での部品管理などに広範に使われるようになってきた．しかしながら，それらのデータベースの使われ方は，きまりきった定形業務が主であり，システム作成時に仕様として与えられた利用目的にないような情報が欲しいときには，データベースに精通したシステム・プログラマの助けを借りて，その都度個別のプログラムを作ることが必要となる．

データベースの利用を，一定の訓練を受けた専門集団から一般家庭へと普及させるためには，ユーザの広範な要求に答えて，多種多様な使われ方ができるようになることが必要である．このギャップは大変大きく，システムの単純な改良によってそれが埋められるものではないことは広く知られている．

その原因は，2つあるように思われる．その1つは，一般家庭では，計算機で簡単に答の用意できるような単純な情報はそれほど必要ではなく，むしろ料理のコツであるとか，木の上手な育て方，テニスの上達法，急病人の看護，交通事故の処理などの各分野での専門的な知識を得たい場合が多いと考えられる．

もう1つは，今の計算機に一般的にいえることであるが，素人にとって計算機システムを使いこなすことが容易なことではない点である．データベースのアクセスのように，計算機と会話しながら処理を進めるとき，会話の相手が理解できない限り，その会話の進行には常に困難がつきまとう．相手が，自分のいったことを理解できるか，もしできなければ，どういえばよいか，それが大きな問題となる．

† On Intelligent Access to Data Bases by Koichi FURUKAWA (Information Systems Section, Computer Science Division, Electrotechnical Laboratory)
†† 電子技術総合研究所ソフトウェア部情報システム研究室
* 現在(財)新世代コンピュータ技術開発機構
Institute for New Generation Computer Technology

前者の問題を解決する努力は，知識工学と呼ばれる人工知能の一分野で，専門知識を扱う技術の開発を中心に積極的になされている．本特集号では，「知識ベース技術の展望」で，その話題が取り上げられているので，ここでは，主として，後者の問題に焦点を当てて論じたい．

後者の問題は，データベースのアクセスに限っても，そのアクセスの全経路に沿って，それぞれ解決すべき課題が山積みしている．最も人間に近い部分では，言語の問題がある．相手である計算機に，どんな言葉を使ってこちらの意志を伝達するかである．計算機になじみのない人でも使えるような言語はどんなものか，自然言語が使えるようになるのか，そしてそれが本当に使いやすいものなのか，これが，第1の問題である．

質問文は，つぎに意味解析を経て検索手続きに翻訳される．意味解析の段階では，あいまい性の除去や，質問の妥当性が調べられる．この検査は，引き続く処理の中にも現れるが，質問者の意図を正確に理解し，さらに，もし質問者が思い違いをしていることが分かればそれを正さなければならない．そのときに必要となる仕組を明らかにすることが，第2の問題である．

関連した問題に，答え方の問題がある．人間の対話の原則の1つは，通信量をできるだけ減らすことである．そのためには物事を要約して回答する工夫が必要となる．

第3の問題は，他の問題と多少性質が異なるものであるが，翻訳されて得られた質問手続きを，効率のよいプログラムへ変換する問題である．これは，質問言語が手続き言語より論理的に高度なものに設定したことによって生じた問題である．

データベースへの知的アクセスを達成するためには，この他にも解決しなくてはならない問題がまだたくさんあると思われるが，本稿ではこれらの3点にとくに注目して，その技術がどの程度進歩してきているのか，今後の見通しはどうか，などの点について，以下に述べていきたい．

2. 質問言語の問題

質問言語には自然言語がよいか，あるいは使いやすい人工言語で十分か，の問題は長い間論争が続けられてきたが，まだ決着はみていないようである．

その原因は，自然言語処理技術がこれまで未発達であったこともあるが，たとえそれが可能になったとしても，はたして自然言語が本当に使いやすいかが十分明らかでない点であろう．自然言語はたしかに最も柔軟な通信媒体であるが，自然言語のみによって伝達できる情報量は，非常に限られていることも認識すべきである．とくに，相手に対する理解が不足しているときは，意味の伝達が困難となる．このことをいい換えると，自然言語がその威力を発揮するのは，別に敷かれたチャネルによって情報の伝達がみかけ以上に円滑に行われているときである．たとえば，相手が，質問者と共通の専門知識を持ち，あるいは常識を持ち合わせていると，表面上はごく少ない情報量を伝達することによって，十分その意図を伝えることが可能となる．すなわち，自然言語による質問応答は，専門知識や常識の理解といった，人工知能，とくに知識表現の研究成果に深く依存しているといえよう．

他方，十分に使いやすい人工言語が存在するのか，あるいは今後そのようなものが出現するのかという問題も，答えるのが困難である．現在，用途によっては，十分に使いやすい言語がいくつか出現している．その代表的なものは QBE[1], SQL[2]などである．QBE(Query By Example) は，グラフィック端末を用いた2次元的な図形言語であり，システムがテーブルの形式を端末に表示することによって，システムのもっているデータベースに関する知識を，質問者に伝達している．この機能は，マン－マシン間のインタフェースを良くするのに大変役立っている．しかしこれで十分というわけにはいかない．第1に，端末に示された関係の枠組から，そこに置かれた情報の意味を誤解なく読み取れるかどうかが問題である．第2に，質問の表現方法が限定されているので，その方法に合った単純な質問であればよいが，多くの関係にまたがる複雑な問題の表現は決して容易でないことが知られている．

再び自然言語に話を戻すと，自然言語の最も得意とするところは，実は，ものごとを最大限簡略化して表現する能力である．今，QBE で図-1のように2つの関係にまたがって表される質問は，自然言語では，「太郎の祖父の誕生日は？」のように短く表される．

父子関係	父	子
	秀　忠	太　郎
	家　康	秀　忠
履　歴	**人**	**誕生日**
	家　康	P. 10月9日

下線のある語は変数を表す．「太郎の父をたとえば秀忠とすると，…」のように読めるので，「例示による質問」(Query By Example) と名付けられている．

図-1　太郎の祖父の誕生日を求める質問の QBE による表現例

質問言語に，自然言語のもつこの種の短縮表現能力を持たせる研究が，いくつかなされている．その1つは，単純な演繹能力を持たせる試みである．論理型言語として知られている Prolog[3]は，演繹能力をもった関係データベース・システムにもなっている．祖父の誕生日という概念は，Prolog では，基本関係，「Xの父はY」と「Uの誕生日はV」を使って，つぎのように定義する．

$$\left.\begin{array}{l}\text{「Xの祖父の誕生日はV」}\\ \quad\leftarrow\text{「Xの父はY」かつ}\\ \quad\quad\text{「Yの父はZ」かつ}\\ \quad\quad\text{「Zの誕生日はV」.}\end{array}\right\}\qquad(1)$$

Prolog では，関係データベース中の個別データも，上に示した言葉の定義も統一的に扱われる．これらはすべて手続き定義であり，論理学における言明 (assertion) である．

質問文は，つぎのようなゴール文で表される．

←「太郎の祖父の誕生日は W」.　　(2)

このゴール文を実行すると，手続き（1）が呼び出され，ついで，その手続きの実行中に，関係「～の父は～」の要素である「太郎の父は一郎」，「一郎の父は権兵衛」などが呼び出される．

（1）式のような手続きは，言葉の定義だけでなく，データの冗長表現を避けるルールの表現にも用いられる．たとえば，大学のカリキュラムで，工学部の学生は，1年次に線形代数が必須であるという事実は，ルール

$$\left.\begin{array}{l}\text{takes (X, 線形代数)}\\ \quad\leftarrow\text{student (X, 工学部, 1年次).}\end{array}\right\}\qquad(3)$$

によって表される．また，このルールがあれば，工学部に属するすべての1年次の学生について，各人が線形代数を受講していることを個々に述べる必要はない．

このような仕組で，Prolog では演繹データベースを実現している．

Tanaka[4]は，さらにカテゴリの理論を用いて，「父の父の誕生日」という表現を解析する方法を示している．そこでは「父の〜」という表現を，自分の世界から父の世界へ行くための射像と考え，その射像自身を数学的に扱って，解析を行っている．言葉の定義をそのような細かいレベルで与えることによって，質問に使える語彙を飛躍的に増加させるのが，そこでの目的である．

自然言語の解析技術自身は，工学的にみれば，この数年間で大きな発展を遂げた．構文と意味の解析を並行して行うのに適した高性能の構文解析機LINGOL[5]がLispの上で作られ，わが国でも田中らによって拡張された[6]．その後，Prolog自身が，LINGOLに匹敵する高性能な構文解析機であることが明らかにされた[7]．このように，Prologの出現によって，自然言語によるデータベース・アクセスは，技術的な障壁がとりはらわれたとみることができるであろう．しかもあとで述べるが，検索手続きの効率化も同じ枠組の中で見事に解決されている．

3. 知的な応答機能

データベースに対する知的アクセスを実現する上で，質問者の言語と並んで重要なのは，システムが質問者に対してどのような情報を与えるかである．もしシステムが人間並みの知的水準に達していれば，質問者の意図を積極的にくみ取ることができるはずである．たとえば，もし質問者がデータベースに対して意味のない質問を発した場合，質問者は，データベースの内容について何か誤解をしていたことになる．そのような誤解を指摘し，もっと有効な質問にするための材料を提供してやることが，そのような機能の1つである．

たとえば，**図-2**のような関係から成るデータベースに，つぎのような質問を発したとする．

「言語学を専攻している学生で，計算機科学第Iの成績が良以上の者は誰か」

この質問は，つぎの5段階の処理に分けられる．

1) 学生の集合から言語学を専攻している者の集合を取り出す．

2) カリキュラム関係で教科＝計算機科学第Iとして，それに付随する教科番号を取り出す．

3) 2)の結果を用いて，学生の受講関係から，その教科を取っている学生の集合を求める．

4) 3)の結果から，その成績が良以上の者を取り出す．

学生（氏名，学生番号，専攻）
カリキュラム（教科，教科番号）
受講（教科番号，学生番号，成績）

図-2 受講データベースを構成する関係の例

5) 4)の結果と1)の結果の共通集合を求めて解を出す．

以上の1から5までの処理中に，その結果が空集合になると，答はやはり空集合になるが，どの段階で空集合が得られたかによって，その意味は全く異なってくる．いま，1)の結果が空集合になったとすると，それは，言語学を専攻している学生が1人もいないことを意味する（あるいは，そもそも言語学のコースがないのかもしれない）．2)の結果が空となった場合には，計算機科学第Iという教科がないことになる．3)の結果が空となった場合は，計算機科学第Iを受講している学生が1人もいないことになる．4)の結果が空となった場合は，計算機科学第Iで良以上の成績を取った者が1人もいないことを表している．すなわち，これらの場合では，単に「そのような人は1人もいない」と答えただけでは，不十分なわけである．そのような答が妥当なのは，5)で始めて空集合になった場合だけである．

1）から4）の間で結果が空集合となる場合，質問者は，そもそも思い違いをしていたことになる．この種の誤った思い込み(Presupposition)を指摘することは，円滑な会話の進行にとって欠くことができない．

Janas[9]は，関係データベースの上で質問文を解析し，上に述べた1）〜5）までの途中の答を出すような質問文を自動的に作り出して，場合に応じて適切な応答をする方式を提案している．

知的な応答機能のもう1つの例は，要約して答える機能である[8]．人間は，相手がどんな情報が欲しいかを推察して，応答しているが，データベースへの問い合わせでは，通常は機械的なアルゴリズムで得られたデータの集合を，そのまま出力するだけである．一般に，出力情報が膨大になれば，その情報としての価値は乏しくなる．すなわち，質問者がその答の中から本当に欲しい情報を探すことになる．あるいは，個々のデータが必要なのではなく，場合によっては，その集合を特徴づける共通の属性を示すことがより適切かもしれない．たとえば，前節のカリキュラム・データベースの例で，「線形代数を受講しているのはだれか？」という質問を発したとき，工学部の1年次のすべての

学生を列挙する代わりに,「工学部の1年次の学生」と答えた方が,要を得ているかもしれない.この種の要約応答ができることと,状況に応じてどのレベルの答を与えたらよいかを判断できることが大切である.要約応答は,質問文の実行を記号レベルの処理にとどめることによって実現できる.上の例では,質問文

←takes (X, 線形代数).

を実行するときに,ルール (3) を呼び出すと,新しいゴール手続きが,

←student (X, 工学部, 1年次).

となるが,プログラムの実行をここで止めると,それは上の要約応答になっている.

4. 検索手続きの効率化

データベースに対する質問言語が計算機に対する指令を表す手続き言語から離れて,人間にとってより自然な言語に近づくと,その質問文を手続きに翻訳する作業と同時に,得られた検索手続きの効率を向上する作業が必要となってきたことはすでに述べた.関係データベースへの検索手続きを改善する研究は,これまでも数多くなされてきた.そのような改善は,最適化と呼ばれている.

最適化の手法の中でも,検索手続きを表す関係代数式を操作して行う方法が,これまで良く知られてきた.Smith ら[10]は,結果を変えない範囲で関係演算の順序を変更し,それによって,演算の中間結果を格納するのに要する記憶領域をできるだけ小さくする方式を与えた.それは,たとえば,ある条件を満足する組の集合を取り出す restriction や,ある関係のいくつかの欄のみから成るような関係を作る projection などの,演算結果が元の関係よりも小さくなるような演算を優先させ,2つの関係をある条件で結合して新たな関係を作る join のような,演算結果が大きくなる可能性のある演算をできるだけ後回しにする方法である.また,いくつかの restriction の中では,情報量の大きい,すなわち,取り得る値の集合が大きい変数の値が定まっているものを優先して実行させると,その結果のサイズはより小さくなることが期待できる.結果のサイズの予測値は関係の中に含まれる組の数と,各欄のとり得る領域の大きさとから計算される.古川[11]は,その考えをインデックスや冗長表現などのデータ表現を利用した最適化にまで拡張した.すなわち,データ表現自身を関係代数式によって表わし,その式を用いて,データ表現に合った検索手続きの導出を,関係代数式の操作により行っている.

Prolog がそれ自身で演繹機能をもった関係データベースであることはすでに述べたが,その実行手続きを最適化する手法が Warren によって与えられた[12].Warren の方法の特徴は,集合演算の実現の仕方にある.それは Prolog の実行制御方式に強く依存したものであり,ある性質を満たすすべてのレコードを取り出すのに,「失敗」による後戻り制御の仕組みを利用している.Prolog での手続き呼出しは,それが「成功」した場合には,きちんと値が求まっているが,ときには「失敗」することがある.「失敗」したときは,そこで呼出し得る他の手続きを呼び出す.これは,人間の試行錯誤の過程に似たものである.関係データベースでは,ある関係の各組は同じ手続き名をもった手続き定義であるとみなされる.ただし,各手続きには手続き本体はなく,呼出し時のパターン照合が成功したら「成功」で,失敗したら「失敗」である.

ある解が求まったとき,そこで強制的に「失敗」を起こさせると,Prolog の処理系は,自動的につぎの解を探し始める.

この制御方式では,集合演算を何段か行う場合に必要となる中間結果を蓄積するための記憶領域をほとんど使わずにすむ.後戻りによる処理は一見効率が悪そうに思えるが,Prolog 処理系では,それが十分に工夫されており,余分な時間はほとんど要しない.

Warren は,この方式を基にして,これまでに関係代数式上の操作でなされてきた最適化と同様の最適化を,その上で実現している.それは,Prolog のプログラマが効率のよいプログラムを作るときの know how を取り込んだ形になっている.

Warren のアプローチは,主記憶内に収まる小規模なデータベースについては,非常に有効である.より大規模なデータベースについては,関係代数に基づく手法の方が,現在のところ有望である.それは,データベース・マシンの機能とも関連している.Warren 流のアプローチを,2次記憶,あるいはデータベース・マシンを含むシステムに適用するためには,Prolog を並列実行する効率のよいアルゴリズムとハードウェアの出現を待つことが必要となろう.

5. おわりに

本稿ではデータベースへの知的アクセスを実現するための主要な問題について述べてきたが,それらに共通の事柄で,見逃すことのできない大切な点をいくつ

か指摘したい．その第1は，システムを開発するための道具であるプログラミング言語の重要性である．これまでにも，何度か触れたが，論理型言語として注目を浴びている Prolog は，知的な機能をもったデータベースを作るのに大変適した言語であるといえよう．Prolog 自身は自然言語の研究から生れたものであるが，現在は，より広範な応用分野への適用性が調べられている．その中でもデータベースへの応用は，特に適合性がよいことが認められつつある．

第2の点は，やや抽象的であるが，データベース自身についての知識を利用することの大切さである．データベースの高度な利用の仕方を考えると，その対象であるデータベースについての深い理解が必要となる[13),14)]．自然言語で与えられた質問文を検索手続きに変換する規則や，データ表現を利用した最適化のための変換規則などは，データベースに関する知識の一種といえよう．利用の仕方を自分で把握していて，大きな図書館の図書館員のように，適当なキーワードから必要な情報を提供してくれるようなデータベースが，本当の意味での知的なデータベースであろう．そのような究極的な姿を目指して，今後の一層の研究の発展が望まれる．本稿がそのための一助となれば幸いである．

参 考 文 献

1) Zloof, M.: QUERY BY EXAMPLE: A Data Base Language, IBM Syst. J. 4, pp. 324-343 (1977).
2) Astrahan, M.M. and Chamberlin, D.D.: Implementation of a Structured English Query Language, CACM, **18**, 10, pp. 580-587(1975).
3) Clocksin, W. F. and Mellish, C. S.: Programming in Prolog, Springer-Verlag (1981).
4) Tanaka, Y.: Information Space Model, in Proc. Formal Bases for Data Bases, Toulouse, (Dec. 12-14, 1979).
5) Pratt, V. R.: LINGOL-A Progress Report, Proc. Fourth International Joint Conf. on Artificial Intelligence, pp. 422-428 (1975).
6) 田中，佐藤，元吉：自然言語処理のためのプログラミング・システム——拡張 LINGOL について——，電子通信学会論文誌，J 60-D, 12, pp. 1061-1068 (1977).
7) 田中：自然言語処理技術研究の新たな展開に向けて，情報処理学会，人工知能と対話技法研究会資料，25-5, pp. 27-34 (1982).
8) Joshi, A. K., Kaplan, S. J. and Lee, R. M.: Approximate Responses from a Data Base Query System: An Application of Inferencing in Natural Language, Proc. Fifth International Joint Conf. on Artificial Intelligence, pp. 211-212 (1977).
9) Janas, J. M.: How to Not to Say "Nil"-Improving Answers to Failing Queries in Data Base Systems, Proc. Sixth International Joint Conf. on Artificial Intelligence, pp. 429-434 (1979).
10) Smith, J. M. and Chang, P.T.Y.: Optimizing the Performance of a Relational Algebra Database Interface, CACM, **18**, 10, pp. 568-579 (1975).
11) 古川：関係データベースに対するデータアクセスの数式処理による最適化について，情報処理学会論文誌，**22**, 1, pp. 68-75 (1981).
12) Warren, D. H. D.: Efficient Processing of Interactive Relational Database Queries Expressed in Logic, Proc. Seventh International Conf. on Very Large Data Bases, Cannes, pp. 272-281 (1981).
13) Konolige, K.: A Metalanguage Representation of Relational Databases for Deductive Question-Answering Systems, Proc. Eighth International Conf. on Artificial Intelligence, pp. 496-503 (1981).
14) Kowalski, R.A.: Logic as a Database Language, Department of Computing, Imperial College, London (1981).

(昭和57年8月17日受付)