情報管理への演繹データベースの適用

浜口幸弘

1　はじめに

コンピュータ，通信網などの情報システムを基盤にした情報化社会の目覚ましい発展は，企業の経営活動にも大きな影響を与え，今や企業内の情報処理業務の自動化は，避けられない重要課題となっている。そして現在の企業の関心事は，一昔前の課題であった部門別のルーチン・ワークのコンピュータ化から，全社的な情報システムによる，管理活動，意思決定などの知的業務の支援に移ってきた。ここに近年，情報システムのインテリジェント化が要請される理由がある。本稿ではこうした情勢を踏まえ，従来のデータ管理方式であるデータベース・システムに，推論機能を有する知識ベース・システムを組合せた演繹データベース・システムのデザインについて論じる。より具体的に言えば，演繹データベース・システムの推論機能を利用して，検索機能をどこまで高められるか，その可能性を考察する。本稿の構成は以下のようになっている。

第2章では，データ資源管理と情報資源管理の相違を明らかにし，情報資源管理の性格を論じる。さらにデータベースを含めた情報システムの特性についても触れていく。第3章では，データベースについて現在の問題点を挙げ，これに対処していくための新しいデータ管理形式，演繹データベースの概要を説明する。第4章では，ユーザからの問い合わせを意味的側面から処理できる演繹データベースについて理論的考察を行う。第5章では，演繹データベースにおける今後の課題を述べ，本稿の締め括りとする。

2　情報資源管理

情報資源管理が近年になって注目されるようになってきた背景には，コンピュータやデータベースに代表される情報技術がOA，FAという具体化された情報

システムとして企業内に浸透してきたことと，それと同時に企業を取り巻く環境の急速な変化によって，企業が情報システムに求める情報自身も絶えず変化しているということが，大きく影響している。ここではまず最初に情報資源に対する管理の認識の変化を論じ，続いて情報システムの特性について概括してみる。

2．1　データ管理に対する認識の変化

コンピュータ・システムが企業に利用され始めた頃（1950～1960年代），その業務対象は給与計算，人事管理などの単純な計算業務および記録保管業務を中心としたもので，データ処理とよばれるものであった。データ処理を目的とする当時のシステム設計思想は，トータル・システム実現に対する技術的な未熟さもあって，個々の独立したコンピュータ・システムの開発，あるいは特定の目的に添った個々のプログラムの開発に重点を置いており，そのため各システムの相互連携性は低く，業務別の独立したデータ管理が行なわれていた。つまりデータベースという概念が生まれる以前では，データファイルとアプリケーション・プログラムは一対になっており，あるファイルが他のファイルとデータを共有することはなく，各組織ごとに自らの業務処理に都合のよいシステム，データ・フォーマットを採用していた。またデータ処理部門はいかに効率よくコンピュータを稼動するかという技術的な側面を強調するあまり，エンドユーザに対するサービス，使い易さへの配慮を欠いていた。このようにデータ処理中心の時代にあっては，企業に点在するさまざまなデータを，それを必要とするエンドユーザに迅速かつ要求通りに提供するという論理は確立しておらず，専らデータ処理部門側の論理が罷り通っていた。したがってコンピュータ・システムを戦略的な業務に利用するには，およそ不都合であったと言える。

その後，コンピュータの処理速度，記憶容量の飛躍的向上，オンライン・リアルタイム処理の実現など情報技術の進歩に伴い，コンピュータ・システムの対象業務が企業全体に広がり，データの規模が増大しユーザの数が急速に増えるにつれて，従来のデータ管理方法が再考されるようになった。つまり部門間で重複するデータ，伝達されるデータを効率良く管理するためにデータベース・システムによるデータの統合，共有という考え方が導入されたのである。これは予めデータのプール，すなわちデータベースを準備しておいて，ユーザからの要求に応じてデータを処理し提供するもので，従来にない大規模なデータ量

を管理対象とすることから，もっぱらデータベース検索の容易性，保守の容易性等の技術的な側面が重視されていた。したがって，技術力を備えたデータ処理部門が依然としてコンピュータ・システム開発の主導権を握っていたわけだが，この段階で注目すべきことは，データベースの導入を契機に，企業がデータを重要な資源の一つとして認識しはじめたことにある。これまでデータは個々のアプリケーション・プログラムと一体になっていて，共有資源として認識されることはなかったが，データベースの導入によってデータがアプリケーション，プログラムと独立して関連業務間で一元的に管理され，組織内外の統一された最新のデータがユーザに提供されるようになった。この結果，環境の変化に即応したリアルタイムなデータ管理が実現し，データ管理を企業の戦略的業務に利用していくことが可能になったわけで，このようにデータベースを通じてデータを管理することを，データ資源管理と呼ぶことにする。データ資源管理は技術的データ処理という点で企業に大きな影響を与えたが，まだまだエンドユーザやマネジメントの側に立った使い易さの論理が尊重されているとは言い難かった。

そして漸く最近になって（1970年代後半），データ処理部門主体のデータ資源管理が見直されるようになり，ユーザのニーズに適した内容をよりわかりやすい形で提供するユーザ本位のデータ管理へと移行する兆しが見えてきた。ここに情報資源管理という新たな概念が登場するのである。情報資源管理では図2.1に示すように，データベース，データを処理する情報システム，情報システム部門，意思決定者たるユーザが管理対象となる。つまり従来の物理的なデータ資源管理にユーザという有機的要素が加わったハイブリッド・システムを対象としており，トータルシステムとしての境界（boundary）が広がったことを意味する。この図では，データは情報システムを通じて情報に変換されているが，これはデータを単に物理的に管理されているにすぎない対象として，情報を特定の意思決定に際し評価を受けた対象として考えているからである。そして企業の経営活動に直接関わりをもってくるのは，情報を提供する情報システムとユーザのインタフェース部分である。

情報資源管理に関し，G. M. Scott は「情報資源管理とはシステム自身あるいは構成要素であるハードウェア，ソフトウェアよりもシステムによって生成される情報に注目することである。情報資源管理では情報およびその利用可能

性（availability），使用方法（usage）が優先し，コンピュータ・システムは情報を生成し，管理するのに必要だという点で，重視されるにすぎない」と述べている。またI. F. Jacksonは「情報資源管理の機能は，企業の諸活動を前進させるためのさまざまな技術を統合化させることに関連している」と述べている。以上のことを踏まえ，ここではScottの強調している「情報の利用可能性，使い方」，Jacksonの強調している「情報技術の統合化」という2点に注目し，情報資源管理を「企業に存在する管理可能な全社的データを，コンピュータ・システムを構成する種々の要素を統合化することによって集中管理し，ユーザからの要求に応じてそのデータをユーザ・ニーズに適合した情報，すなわちユーザならびに組織の意思決定に有用な情報へ変換して提供すること」と定義する。

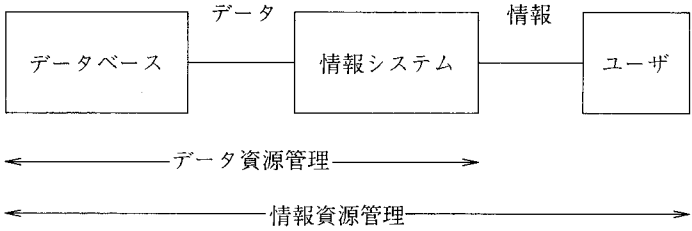


図 2.1　情報資源管理の対象範囲

2．2　情報資源と他の資源との比較

ここでは情報資源の特性について理解を深めるために，情報資源と他の資源（原材料，設備など）との類似点，相違点に関して，Scottの記述をもとに各々列挙してみる。

類似点は次のようになる。

(1) 情報はコスト（収集，処理，維持）を伴う。その総合価値は総合コストを上回らなければならない。

(2) 情報は投資収益を持つ。もっとも他の資源に比べて評価するのは難しいわけだが。

(3) 情報を持たないことによる機会コストが生じる。他の機会コスト同様，その機会コストは見逃しやすい。

(4) 情報要素の組み合わせは，情報要素の部分の総和より大きな価値を提供す

る。つまり要素間に相乗効果が存在する。

(5) 情報を効果的に使うためには，使用に際して十分構造化しておかなければならない。

一方，相違点は次のようになる。

(1) 他の多くの資源は生産時に（物理的に）消費されたり，使い古されたりするが，情報は追加使用当たり低い限界コスト（理想的には）無限に再使用される。このことは情報を特に有用な資源として特徴づけている。

(2) 情報は無形である。このため情報がどのように使われるべきか，またどのように効果的に使われてきたか評価するのは難しい。このことは情報管理の概念がごく緩やかにしか現われてこなかった事実をある程度説明している。

上の記述で重要な点は，常識的なことだが情報が再使用可能なユーザニーズに依存した無形物だということにある。この特性ゆえ情報を管理するには企業独自の方法が要求され，他の資源に比べて扱い方は難しくなってくる。もう一つ注目すべき点は，類似点(4)で挙げている情報の組み合わせ，すなわち情報理論的にいえば情報の結合性（connectivity）である。これは，ある情報と別の情報が結びついて，新たな価値ある情報が生まれる可能性を意味している。例えば，「製品Aは国内で供給過剰にある」という情報と「米国では製品Aに対する需要が高まっている」という情報が結合して，「製品Aを米国に輸出する方策を講じろ」という新たな情報が生まれる。Scott は情報資源と他の物理的資源との類似点として結合性を挙げているが，情報資源の結合性は他の物理的資源の結合性に比べて柔軟性，つまり資源の結合するパターンの豊富さという点で優れているので，注意しておく必要があろう。そして情報資源をデータベース・システムで管理していくには，この情報の結合性に如何に対処するかが問題となってくる。従来のデータベース・システムでは，これは非常に困難であるが，知識ベースを取り入れたデータベース・システム，すなわち演繹データベース・システムは情報を内部で自動的に結合する可能性を与えてくれる。この情報の結合の可能性については第4章で論じることにする。

2．3　情報システムの構造と特性

本節では，情報システム全体の立場からシステムの構造およびその特性について考察してみる。

システムの特性を記述する糸口としては，それを構成するサブシステム間の

相互作用に注目するのがよいであろう。相互作用には結合と資源共有の二つのタイプがあり，結合とはあるサブシステムのアウトプットが他のサブシステムのインプットとして働く，あるいは逆方向，相互に働くことで，資源共有とは同一の資源をいくつかのサブシステムが共有することを意味する。これを情報システムの場合に当てはめてみると，結合はネットワークを通じてシステムが接続されたコンピュータ・プロセッシング・コントロールに相当し，資源共有はデータファイル，データベース等に蓄積されるデータ資源の共有に相当する。そしてサブシステム間の相互作用が弱い，即ち調整をそれほど必要としない情報システムを独立系情報システム，相互作用が強い，即ち緊密な調整を必要とする情報システムを統合化情報システムと呼ぶことにする。もちろん情報システムは明確に二つのどちらかに分類されるわけではなく，中間の形態も存在する。

サブシステム間の相互作用の弱い独立系情報システムをデザインする場合，サブシステムのつながりを切り離して（decouple）やらなければならない。デカップリングを実現するには，各サブシステムが一つのまとまった機能を果たすようにしてやるか，あるいは各サブシステムの相互作用の程度を極力小さくしてやればよい。その具体的手段としてバッファ・スラック資源を保有したり，サブシステム間のインタフェースにおけるインプットとアウトプットを標準化することが考えられる。バッファとは，あるサブシステムのアウトプットが別のサブシステムのインプットとして供給されるとき，両者の処理能力に差異があると一旦アウトプットをバッファに入れておいてその差を補償するデバイスである。スラック資源とは，例えば過剰な負荷がある情報システムにかかったとき，他のサブシステムに影響を及ぼさないように予め演算能力，記憶容量等が通常の必要量を越えて与えられたりする余裕のことである。これらはシステムへのランダムな擾乱が起こったとき，素早い応答性，大きなロバストネスを与えている。標準化とは，インプット，アウトプットをある規格内に納まるようにして，各サブシステムが他のサブシステムに問い合せることなくインプットを機械的に処理できることを意味している。以上は独立系情報システムのデザインに伴う利点であるが，同時にバッファ，スラック資源を持つことで余分な資源を抱えたり，機能の重複を招いてコストがかさむという欠点も持っている。

これに対して統合化情報システムをデザインする場合，サブシステム間の強い相互作用を調整するために，各サブシステムの現況に関する情報を即座に収集，伝達，検索，変換できるような高度な情報処理能力が要求される。この高度な情報処理能力をもつ調整（調整役として例えばメインフレーム，データベース管理システム）を通じて，システム全体が同期化して運営され，資源共有が効率的に行われるのである。したがって各コンピュータシステムが共有データベースを核にネットワークによって統合化された情報システムでは，ある種のデータがある一つのサブシステムに入力されても，それを要求する他のサブシステムはリアルタイムで入手することが可能になるわけで，ユーザは自分の端末から好きな時間に最新のデータにアクセスすることが可能となる。またデータの共有，高速伝送は業務間の連携を強化し，製品開発の期間短縮化に重要な役割を果たすであろう。しかし当然のことであるが，統合化に伴う欠点もいくつか存在する。例えばサブシステム間のコミュニケーションを同期化する技術は非常に高度なため，システムの開発費用，管理費用は大きなものとなる。またバッファ，スラック資源の保有を極力抑えているため環境の不確実性に対するロバトネスは小さくなってしまう。

以上，独立系情報システムと統合化情報システムのデザインについて見てきたが，今日どのような情報システムが求められているのだろうか。前にも述べたように，企業にコンピュータによるデータ処理が導入された当初はアプリケーションとデータファイルが一体になった独立系システムが採用されていたが，その後は共通業務，関連業務を処理するに当たってデータベースを中心にした統合化システムが導入されるようになってきた。ここに独立から統合への流れを見てとれるが，すべてのシステムが統合化されるわけではないだろう。つまり今後の方向として，独立性と統合性の両方を兼ね備えたシステム，すなわち分散情報システムが主流になると考えられる。

分散情報システムにおいては，他部門との関連性が薄く処理量がそれほど大きくない業務に対しては，通信網で接続されるメインフレームとか他部門の強力なコンピュータが利用されるのではなく，ユーザ部門のミニコンピュータ，あるいはマイクロコンピュータが独立して利用される。というのは，他部門のシステム利用に伴う面倒な手続き，調整を回避できるし，自部門独自のやり方で業務を処理できるからである。このことは最近注目されるようになってきた

エンドユーザ・コンピューティングの考え方の根拠を与えている。

一方，他部門との関連性が強い業務，処理量の大きい業務に対しては，ユーザ部門のコンピュータ・システムが共有データベース，メインフレームを核とする統合化情報システムの構成要素の一つとして統合化され利用される。このときメインフレームはローカルコンピュータで対応しきれない大規模な計算処理を実行し，またローカルコンピュータ間の調整を行なう。システムの統合化は各部門間にまたがる問題であり，技術的にも高度なことから情報システム部門が調整役を演じることになる。

このように分散情報システムは独立性と統合性の二つの面をもっており，そのトレードオフが簡潔さ，専門性の経済に有利な場合には独立性が，またトレードオフがデータの共有，規模の経済に有利な場合には統合性が採用されることになる。またその二面性ゆえ，上に述べた情報システム部門とユーザ部門の分業体制が要求されよう。

3 データベース

データベースには二つの利用形式がある。一つは応用プログラムが必要とするデータを供給することで，各応用プログラムはデータベース管理システムが提供する一定のアクセス手順をプログラムに埋め込んでデータベースにアクセスすることになる。もう一つはユーザが直接，データベース内のデータにアクセスし，データを意思決定に利用する形式である。本稿のテーマは，ユーザ・インタフェースを対象としているので，後者に関する議論を進めるものとする。そこで，本章では関係データベースの基礎概念，関係データベースと知識ベースの関係を論じ，演繹データベースの概念を明らかにしておく。

3．1 記憶機能としてのデータベース

本来データベース・システムとはコンピュータの記憶能力を利用するシステムである。数値演算のように比較的小規模のデータに複雑な演算を施して所要の結果を得ることを目的にしたシステムと異なり，大規模なデータを体系的に管理し，要求に応じて迅速に所要データをその中から取り出してくる操作は，一見単純に見えるが，そこには記憶情報に固有のさまざまな問題が内包されている。注意すべき問題として，情報の供給者と利用者の間で生じる解釈のギャップを挙げることができる。記憶機能を利用する方式のもとでは，一般に情報を

与える操作とそれを利用する操作はそれぞれ別々のものとして切り離されており，したがって情報の供給者は与えた情報が後にどのように用いられるかについて事前に知ることができない点が異質である。この性質ゆえ情報の供給者と利用者の間で情報に対する解釈が異なると，思いがけない誤解や混乱が生じる可能性がある。これを防ぎ記憶情報を有効に利用するためにとられる一般的方法は，情報処理機能としての枠組，すなわち処理目的とその範囲を適切に定め，そこで必要な情報表現の形式を定めることである。情報の供給者も利用者もそこで定められている形式や手順に従って意味やその利用法を表現することにより，情報の意味の理解に関して解釈の違いが生じる可能性を小さくしている。この情報処理機能の枠組が適切に設定され，定められている形式や手順に従って，情報の表現形式およびそれに基づいて定義される処理手順が適切に設計されていれば，記憶機能を利用するためのシステムを開発することができよう（図 3.1)。

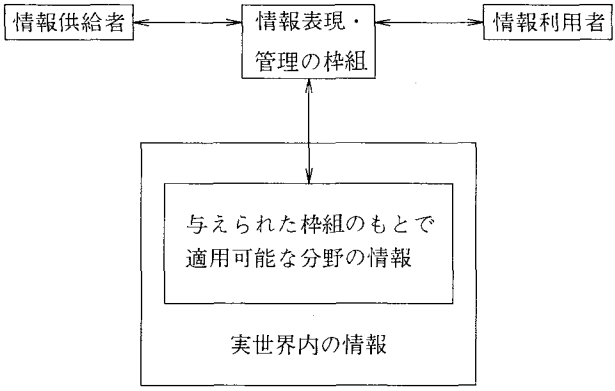


図 3.1　記憶利用の方式

この記憶機能を利用する難しさは，本来記憶というものが知的機能，例えば推論機能と密接に結びついているからであり，そこに内包されている問題を解決するためには，人間の知的活動を解明しなければならない。しかし，これは現在の科学のレベルでは不可能であって，実用的なシステムをデザインするには，このように高度なレベルまで連続している問題の鎖を適当なところで断ち

切って，閉じた技術領域を設定しなければならない。本論文の狙いはこの閉じた技術領域に基づくデータベース・システムを何処まで拡張できるか，ということにある。

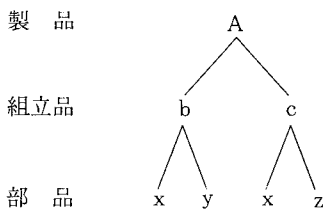
3．2　現行データベースの限界

データベース・システムの基本的な動作は，ユーザからの問い合わせに対して，記憶データ（事実）からその条件を満たすデータを選択してユーザに提供するという，上述の記憶機能に基づくものである。データベース・システムはデータの選択に当たって，データベース上で定義されているエンティティ（一種のカテゴリィ）内のデータの直接の関係しか扱えないわけで，ここに問題が生じてくることになる。ここでは生産管理における情報管理を例に問題点を提示してみよう。

最近，多品種少量生産化の要請が高まっている生産管理では，木構造をなす部品構成表の中から自由に製品と部品の関係，あるいは部品と部品の関係を検索できるデータベースの構築が要求されているが，現行のデータベースでは以下のような問題が起こってくる。

例えば図 3.2に示すような部品構成表があるとする。このときデータベースとして親子関係を示すデータを表形式で記憶したとする。データベース・システムの機能を利用して，ユーザは親の名前（例えば製品Ａ）を与えて，その子の名前を求めたり，逆に子の名前から親の名前を知ることができる。ところが我々の世界では親子関係が既知であれば，その系譜も当然求めることができるが，データベース・システムは親という属性と子という属性しか扱っていないので，例えば製品Ａの孫に相当する部品を尋ねても答えてくれない。系譜をすべて知るためには，製品と組立品，製品と部品，組立品と部品という三組の表形式のデータベースを用意しておく必要があり，親子関係のデータベースに比べて大変冗長なものとなってしまう。このように既存のデータから本来（常識的に考えれば）導けるはずのことが，簡単には得られないという問題点が生じてくる。これは従来のデータベース技術が記憶機能を基本としたもので，推論機能が欠如していることに起因する。もし親子関係のデータベースに加え，「製品Ｘが部品Ｙを持つのは，ＸがＺの親であり，ＺがＹの親であるようなＺが存在するときである」という知識が与えられると，推論によって製品Ａの部品を求めることは可能となる。この結果，親子関係のデータベースと製品と組立品，

製品と部品，組立品と部品という三組のデータベースがデータ操作上同一のものになるのである。



親	子
A	b
A	c
b	x
b	y
c	x
c	z

製品	組立品
A	b
A	c

製品	部品
A	x
A	y
A	z

組立品	部品
b	x
b	y
c	x
c	z

図 3.2　部品構成表のデータベース

一般に，エンティティ同士の関係がｎ段まで続く木構造（図左）をしている場合について，各エンティティ内のデータがすべて異なるものとして組データの総数を比較すると，親子関係のデータベースでは総数（2^n-2）個の組データが，任意の二つのエンティティ同士の関係を表したデータベースでは総数（$(n-2)\cdot 2^n+2$）個の組データが得られる。明らかに親子関係のデータベースの方が表現するデータ数は少なくてすむ。さらに，一般的な場合として，エンティティの関係がネットワーク構造（図右）をしているときにも，枝で直接連結されているエンティティ同士間の関係を求め，間接的に連結されている

エンティティ同士の関係は推論機能を使って求める方が有効である。

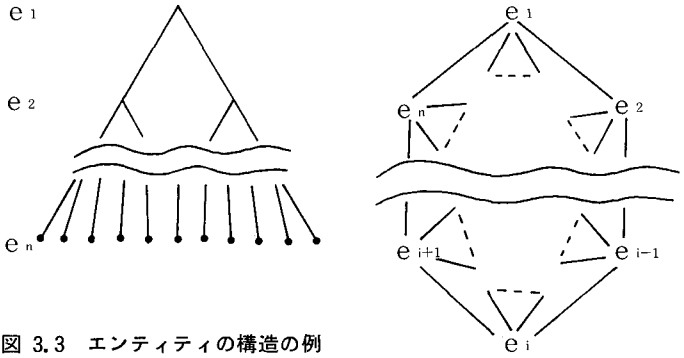


図 3.3　エンティティの構造の例

したがって従来のデータベース技術で扱われなかった表現形式のデータを形式化し，データベースに与えれば（知識ベースと呼ばれる），推論機能が付与されデータベースの問題解決能力は増大するはずである。以下の節では，今日の代表的なデータベースである関係データベースの基礎概念を説明し，続いて知識ベースを取り入れた関係データベースによる問題解決について考察する。

3．3　関係データベース

従来，企業などで使用されていたデータベースには階層型とネットワーク型があったが，1970年にE. F. Coddが大型共有データベースの論理的構造として関係モデルを提案して以来，その使いやすさ，理論的発展可能性から最近では関係データベースが主流となってきた（以下，本稿で対象とするデータベースは関係データベースとする）。関係データベースはデータを表の形式で一元管理し，一般ユーザ向けの問い合わせ言語によってデータを有効活用できる環境を提供する。関係モデルの基本的要件は，次に示す3つである。

(1)　アプリケーションはデータ独立である。すなわちアプリケーションの変更と記憶領域の構造およびアクセス戦略の変更は独立して行なえる

(2)　表中のデータは互いに矛盾することはない。それには次の一貫性を満足しなければならない。

実体の一貫性：表の各行を識別するデータ項目が存在する（基本キー）。

参照の一貫性：基本キーを他の表の一部として用いるとき，互いに一致

していなければならない。

(3)　データ操作言語（問い合わせ言語）を備えている。この言語は対話式に端末から指示することも，プログラムの中に組み込むこともできる。

上述の(1)は，データベースシステムという複雑なシステムが独立した構成要素に分解できるという点から特に重要な要件である。また(3)の問い合わせ言語の基本的機能を代数的表現で表すと，集合演算（和集合，積集合，差集合），直積，射影，選択，制約，商，結合という演算になる。これらのうち，和集合，差集合，直積，射影，選択が基本演算であり，他はこれらの組み合わせによって表すこともできるという意味で補助的な演算である。基本演算の機能について以下列挙する。

R_1，R_2を関係表，tを組データ（n組）を表す変数とする。

- 和集合；$R_1 \cup R_2 = \{t \mid t \in R_1 \vee t \in R_2\}$
- 差集合；$R_1 - R_2 = \{t \mid t \in R_1 \wedge t \in R_2\}$
- 直積　；R_1，R_2を各々，項数としてn_1，n_2をもつ関係とする。直積$R_1 \times R_2$は，R_1の組データとR_2の組データをすべて組み合わせた（$n_1 + n_2$）項の組データから成る集合である。
- 射影；関係Rのなかから，属性a_1，…，a_kの部分を取り出す演算を次のように定める。

  $\pi_{a1, \cdots, ak}(R) = \{t[a_1, \cdots, a_k] \mid t \in R\}$

  なお，$t[a_1, \cdots, a_k]$は，組データtのなかの属性a_1，…a_kのみからなるものを表す。
- 選択；関係表中の必要な組データを取り出す。関係Rにおいて所要の組データはFという条件を充たすとすると，選択$\sigma_F(R)$は次のように定められる。

  $\sigma_F(R) = \{t \mid t \in R, F(t)\}$

さて関係データベース・システムとユーザのインタフェースを考えたとき，使用の柔軟性と操作の複雑さ，という概念を導入できる。使用の柔軟性はシステムへタスクを依頼する際にユーザに与えられた柔軟性を意味し，システムのサポートによってユーザが実行できる処理の種類により決まる。一方，操作の複雑さは，仕事をするためにユーザが克服しなければならない複雑さを意味する。インタフェースの設計は，使い方における自由度の最大値と操作性におけ

る複雑さの最小値との間の最適化問題と考えられよう。

3．4 関係データベースと知識ベース

データベース発展の歴史を眺めたとき，米国でデータベース管理システムが普及し始めたのは1960年代の中頃からで，その性格上，実用的色彩の濃い発展経緯であった。一方，知識ベース・システムは人工知能の一領域に属し，当初は理論が先行していたが，最近になって医療支援システムであるMYCINを始めとする数々のエキスパート・システムが世に登場してきた。データベース管理システムと知識ベース・システムはその目的，経緯からみても全く別々のものであったが，関係データベースのモデルがCoddによって提案されてから，その両者の歩み寄りが次のように注目されてきている。

関係データベース・システム側から

・関係データベース・システムに知識ベースを組み込むことによって，現行の関係データベースでは，ユーザからの問い合わせへの対応が複雑になったり（検索回数が多くなる），あるいはその対応が不可能に（関係データベース上で定義されてない概念が含まれる）なったりする問い合わせに対しても，可能な限り応答できるデータベースが望まれている。

知識ベース・システム側から

・知識ベース・システムの規模が大きくなると，作業記憶の管理，知識ベース自体の管理が問題となるので，これを解決するためにデータベース管理システムの技術の応用が検討されている。

また理論的側面からみても，関係データベースと知識ベースの間には，ある種の親近関係が見いだされる。つまり，関係代数における定理は等価な一階述語論理に置き換えられるし，関係データベースで用いられる関係表は一階述語論理である論理プログラムのホーン節の集合として完全に記述できる。このことから，関係データベースにおいては，演繹推論機能を導入することが可能であり，事実のみでなく変数を含む論理式，すなわちルールの集合を扱うことができる。

さて論理プログラムによる問題解決の手続きは，概念的に次のように記述できる。

(1) 最終目標を定め，それを“goal (Materials)”のような形式で表現する。

(2) 最終目標をいくつかの下位目標に分解する。

goal (Materials)→sub1 (M1) & sub2 (M2)... & subn (Mn)

(3)　変数がどのようなものか，論理式で表現する。

(4)　(2)で示された下位目標については，必要に応じてさらに下位目標に分解して，(1)～(3)の方法で記述する。

(5)　下位目標を解決していくことにより，最終目標に到達する。

上述の手続きは，下位目標を解決しながら最終目標と現在の状況の乖離を次第に解消していく点で，H. A. Simon, A. Newell が提示した GPS と同じ考え方をしており，思考を記号化したものと言えよう。

4　演繹データベース

データベース内のデータを操作するためには言語が定義される。これにはデータベースに対して代数演算を定義する関係代数と，論理的に操作を記述する関係論理がある。これらの言語は，当初はデータベースに関する最も基本的な操作を保証するものであったが，次第にユーザ向けに，より使いやすい言語，より記述性の高い言語への指向が強まり，同時に，質問言語を解釈して最適な評価を行うことが問題とされるようになってきた。

この発展としてデータベースからの直接的検索のみでなく，演繹的検索を可能にすること，例えば前述のように親子関係を表す関係表を用い，「親の親は祖親である」という推移則を用いて祖親―孫の関係を導きだすこと，が考察されるようになった。これを演繹的検索という。

こうした演繹的検索への傾向は，データベースが単純なデータ貯蔵庫的な機能から，記述対象である世界のモデルへと進化している過程である。この傾向はファイルの概念から一歩進んでデータベースという概念が導入されたときに生じていた。データ貯蔵庫とモデルの類似点は，最も低いレベルでは記号化された情報の記憶と，最小限その検索ならびに更新・付加・削除が必要とされる点である。データ貯蔵庫であればこの程度で十分であるのに対し，モデルはその上の論理レベル，すなわち意味レベルで，多様な概念を扱うとともに，これらの概念を実現するための手段や機能も要求される。

本研究の対象とするモデルは，図 4.1に示すような関係データベースと知識ベースを通信機能によって接続した結合方式の演繹データベース・モデルであって，推論システムが知識ベースを用いて演繹を行い，必要に応じてデータベー

ス・システムを呼び出しデータを取り出すことを前提とする。ここで述語論理式で表された知識は，知識ベースが管理し，事実を表すデータは，関係データベースの組データとしてデータベース中に格納されるものとする。

本研究では演繹的質問応答を実現する上で有用な

(1) データベースの構造化（structuring database）

(2) 問い合わせの評価プロセスを内包的関係の評価と外延的関係の評価の二つのプロセスに分解（decomposition of query evaluation process）

(3) 意味的な問い合わせの改善（semantic query improvement）

の三つのプロセスに照準を当てて，定理の自動証明理論を根拠に理論的考察を試み，演繹データベースシステムの機能の拡張，強化について議論する。同時に，この理論を導入した演繹データベース・システムとユーザとの間の演繹的質問応答について，通常のデータベース・システムに対する検索との比較を行いながら，ユーザインタフェイスの側面からその優れている点を明らかにしていく。

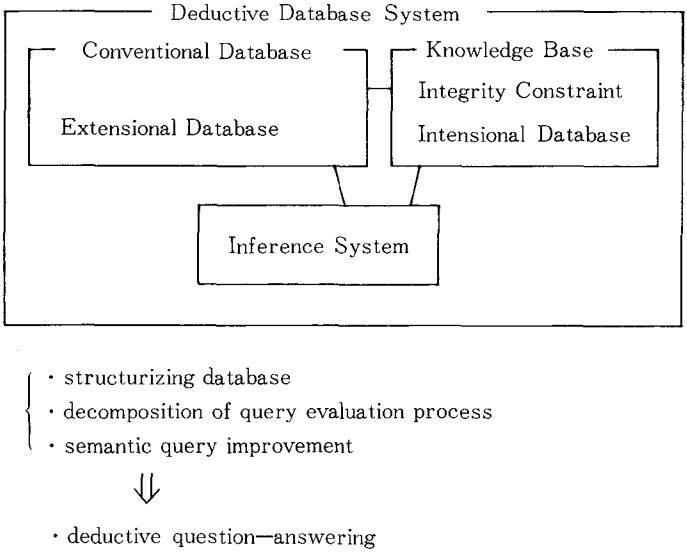


図 4.1 本稿で対象とする演繹データベース・モデル

4．1　データベースの構造化

ここではデータベースの構造化について論じるが，まずその前に，本章全体にわたって使用する用語の定義，概念の仮定について記しておく。

〖定義および仮定〗

まず最初に本稿で使用する用語の定義を述べておく。

・ホーン節（Horn clause）

$A \leftarrow B_1, B_2, \cdots, B_n$ のように左辺（頭部）に高々一つ，右辺（胴部）に任意個の素論理式を並べた節。右辺は素論理式の論理積を表す。

・単節（unit clause）

胴部を持たない頭部だけのホーン節。

・空節（null clause）

胴部と頭部を持たない節で，矛盾を表す。

・限定含意節（definite implicational clause）

胴部に少なくとも一つの素論理式，頭部にちょうど一つの素論理式を持つ節である。

・領域制限節（range-restricted clause）

頭部に現われるすべての変数が，非数値述語（nonevaluable predicate）の引数として胴部にも現われる節である。

・インスタンス（instance）

$\theta = \{v_1/t_1, \cdots, v_n/t_n\}$ を代入，Aを論理式としたとき，$A\theta$ をインスタンスという。

・包含（subsumption）

論理式C，Dに対し，$C\sigma$ がDの部分節（subclause）になるような代入 σ が存在するなら，CはDを包含するという。

上の定義に基づいて演繹データベース DBは，以下の構成要素 EDB, IDB, ICによって　DB=EDB ∪ IDB ∪ IC と表せる。

・EDB (extensional database)

関係表中の組データを表す単節の集合で，通常のデータベース内のデータがこれに相当する。また，この単節は外延的関係（extensional relation）または外延的述語と呼ばれ（"＊" 印で表記），頭部に現われることはない。

・IDB (intensional database)

非再帰的で領域制限のある，関係数を持たない限定含意節の集合で，次のICとともに，演繹データベース独特のデータベースである。ホーン節の頭部に現われる述語は内包的関係（intensional relation）または内包的述語と呼ばれる。

・IC (integrity constraints)

一貫性制約を表す非再帰的で領域制限のあるホーン節の集合。一貫性制約とは，演繹データベースの記述世界が満足すべき条件である。

さらに本論文では以下の仮定を採用する。

・閉世界仮説（Closed World Assumption; CWA）をメタルールとして使用する。すなわち，$\vdash \neg P \longleftrightarrow$ not $\neg P$である。

・すべての問い合わせは関数形を持たないゴール節とし，出力変数を〝*〟印を使って表すものとする。たとえば←Q (y1*, …, yn*, z1, …, zm, c1, …, ck)。

〖データベースの構造化〗

さて，上に述べた一貫性制約が限定含意節の形をなしているなら，新たなデータを生成する可能性を持っている。しかし，一貫性制約とはデータに対する制約であるから，それから新たなデータが生成されることはデータ管理上好ましくない。そこで一貫性制約をデータベースのコンシステンシィを確認するだけの節にするために，

・すべての関係は純粋に外延的か，純粋に内包的であるようにする。つまり関係はどちらか一方に分類されるものとする。

・一貫性制約は外延的関係と数値述語（evaluable predicate）だけで表す。のような条件を与え，最初に与えられたEDB, IDB, ICをこの条件を満足するように同値変換すればよい。これをデータベースの構造化と呼ぶことにする。このアルゴリズムは次のように記述できよう。

〔アルゴリズム1〕

構造化されたデータベース DB*={EDB*, IDB*, IC*}は，次のステップを通じて DB={EDB, IDB, IC}から得られる。

1．ハイブリッドな関係H（EDBにもIDBの頭部にも使われている関係）を内包的関係Hと外延的関係H*に分割せよ。そしてH←H*をIDBに加えよ。

2．一貫性制約において内包的関係が胴部に存在するならば，適当な単一化を

施した後，置換することによって，すべての一貫性制約の胴部から内包的関係を削除せよ。

3．EDB と IC を使って内包的関係の組データを外延的関係に生成することによって，すべての一貫性制約の頭部から内包的関係を削除せよ。

上のアルゴリズムを通じて構造化されたデータベースを導くことができる。簡単な例を以下に記しておく。

(例1)

EDB:　$R1^*$, $R2^*$, H2

IDB:　$H1(x,y,z) \leftarrow R1^*(x,y), H2(y,z)$

$H2(x,y) \leftarrow R1^*(x,y), R2^*(y,c)$

IC:　$H1(x,y,d) \leftarrow R1^*(x,d), R2^*(x,y)$

$\leftarrow H2(a,b)$

step1: H2 を $H2^*$ に変更することによって EDB を修正し，更に $H2(x,y) \leftarrow H2^*(x,y)$ を IDB に加えよ。

step2:　$\leftarrow H2(a,b)$ を IC から除去し，次の節を ICに加えよ。

$\leftarrow R1^*(a,b), R2^*(b,c)$

$\leftarrow H2^*(a,b)$

step3:　$H1(x,y,d) \leftarrow R1^*(x,d), R2^*(x,y)$ を IC から除去して，$H1^*$ を EDB, $HI(x,y,z) \leftarrow H1^*(x,y,z)$ を IDB に加えよ。

以上の手続きから得られる DB^* は次の通りである。

EDB^*:　$R1^*$, $R2^*$, $H1^*$, $H2^*$

IDB^*:　$H1(x,y,z) \leftarrow R1^*(x,y), H2(y.z)$

$H2(x,y) \leftarrow R1^*(x,y), R2^*(y,c)$

$H1(x,y,z) \leftarrow H1^*(x,y,z)$

$H2(x,y) \leftarrow H2^*(x,y)$

IC^*:　$\leftarrow R1^*(a,b), R2^*(b,c)$

$\leftarrow H2^*(a,b)$

4．2　問い合わせ評価プロセス（query evaluation process）の分解

本節では先のデータベースの構造化に続く次のステップ，問い合わせ評価プロセスの分解について考察する。本論文で提示する問い合わせ評価プロセスの分解とは，質問応答のタスクを内包的データベースに関する定理の証明プロセ

ス（内包的処理）と外延的データベースに関する検索（外延的処理）に分解することである。このことについて先程の例(1)を参考にしながら説明していこう。

まずデータベース・システムは，IDB*に含まれるすべての内包的述語を取り出し，定理証明器を用いてそれぞれの証明探索木（proof search tree）を構成する。このとき定理証明器は EDB*にアクセスすることはない。この手続きは内包的述語を外延的述語に変換するためのプロセスであり，IDB*全体を編集（compile）することになる。証明探索木を構成するアルゴリズムは次のように記述できる。

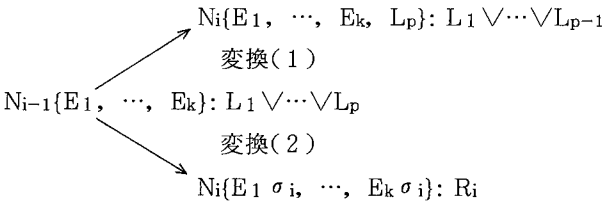
〔アルゴリズム 2〕

ノード$N_{i-1}=\{E_1, \cdots, E_k\}: L_1 \vee \cdots \vee L_p$（節）の拡張を1および2の方法に従って，最終的に NIL が得られるまで，つまり拡張不可能になるまで続行する。

1．ノードN_{i-1}をノードN_iにおいて$N_i=\{E_1, \cdots, E_k, L_p\}: L_1 \vee \cdots \vee L_{p-1}$と変換せよ。

2．ノードN_{i-1}の節$L_1 \vee \cdots \vee L_p$が IDB*内の節と導出可能ならば，ノードN_{i-1}を$N_i=\{E_1\sigma_i, \cdots, E_k\sigma_i\}: R_i$とせよ（$R_i$は導出節，$\sigma_i$は mgu)。

すなわち，



である。

上のアルゴリズムによれば，上述の内包的述語 H1(x,y,z) についての証明探索木は，図4．2のように表される。

こうして編集された IDB*は予めデータベースに保存される。そこでユーザからの問い合わせがあった時，定理証明器が与えられた問い合わせに関連するすべての情報を抽出しながら，IDB*全体にわたって証明探索木を掃引する。そして証明探索木に基づいて元の問い合わせに対応する新たな問い合わせの集合が導かれ，関係データベース・システムによって外延的に評価される。

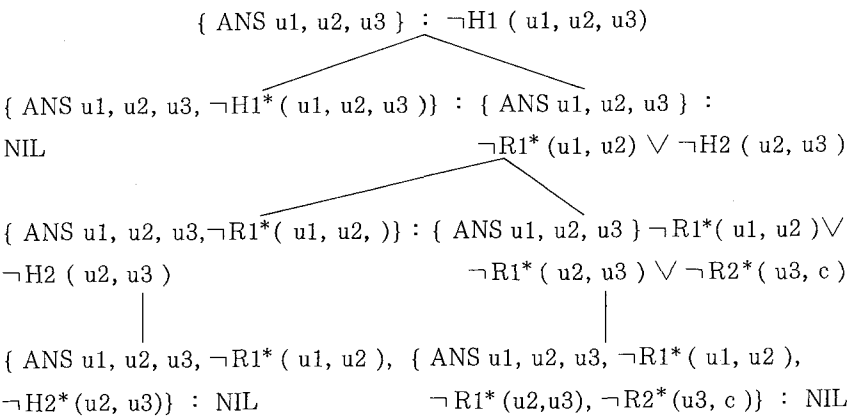


図 4.2　証明探索木の例

例えば，Q: ← H1(u1,a,b) なる問い合わせが与えられたなら，

Q1: ← $H1^*$ (u1,a,b)

Q2: ← $R1^*$ (u1,a)，$H2^*$ (a,b)

Q3: ← $R1^*$ (u1,a)，$R1^*$ (a,b)，$R2^*$ (b,c)

なる新しい問い合わせに変換され，これらが EDB^* 上で評価される。

したがって，新たに導かれた問い合わせに対する解の和集合が，元の問い合わせに対する解集合と一致することになる。

この質問応答タスクの分解はどのような意義をもつのか。次のようにまとめられよう。

(1) IDB^* の証明探索木を，EDB^* にアクセスすることなしに定理証明器によって生成し，次に証明探索木から得られる新たな問い合わせの検索を EDB^* のみにて行う分業体制は，通常の定理証明で行われている定理証明器が IDB^* と EDB^* の間を行ったり来たりするような繁雑なアクセスを避けることができる。

(2) 一旦，IDB^* の内包的述語の証明探索木を作って保存しておけば，問い合わせがある度に再利用することができる。また問い合わせの評価時間の短縮にも貢献する。

4．3　意味的な問い合わせの改善

従来から研究されていた問い合わせの改善が，アクセス・パスやストレージ

構造の最適化を通じて検索効率を高めることに重点を置いていたのに対し，ここで述べる意味的な問い合わせの改善は，データベース内の事実，ルールの論理的意味を抽出して検索効率の改善に役立てようとするものである。

意味的な問い合わせ改善のプロセスを二つのフェーズ，編集フェーズ（compilation phase），問い合わせ変換フェーズ（query transformation phase）に分けて考えることにする。前者はデータベース内のデータに関する一貫性制約とルール集合を使って，問い合わせに対する制約条件を導きだすフェーズであり，後者はその制約条件を問い合わせに付与するフェーズである。以下この順序に沿って，定理の自動証明理論に基づいて J. Minker ら（1988）が提案した residue の考え方を基礎に議論を進めていく。

《編集フェーズ》

意味的編集の基本的考え方は，拡張した一貫性制約（補遺参照）の部分節によって IDB の論理式の胴部を包含することにある。この部分的包含の結果として生ずる一貫性制約の残りのインスタンスを，residue と呼ぶことにする。すなわち，residue は内包的論理式と一貫性制約との相互作用を表すものであり，その厳密な定義は補遺で示している（$((IC+-IC+m)\ \sigma)-)\ \theta^{-}$である。residue を使った問い合わせ改善の例を参照しながら，residue の有効さを評価してみよう。

（例2）

A: $H(x,z) \leftarrow R1^{*}(x,y),\ R2^{*}(y,z)$

IC: $\leftarrow R1^{*}(u,u)$

一般に IC 内の変数をそのままにしておくと，IC の部分節が内包的論理式Aを包含しない場合が出てくる。そこで IC を拡張形 IC+ に変換して（拡張変換のアルゴリズムは補遺参照），包含可能性を調べるようにする。

IC+: $\leftarrow R1^{*}(u,x1),\ (x1=u)$

上の IC+ は最初のリテラルが論理式Aを包含するので，このとき residue は $\leftarrow (x=y)$ である。したがって，ある問い合わせ $\leftarrow H(x,z)$ が与えられたときに，x=y が成り立つならば，この問い合わせは答えを持たないことが解探索する以前にわかる。

（例3）

A: $H(x,y) \leftarrow R1^{*}(x,y)$

IC: ← R1* (u,v), GT (v,100)

ここでICを拡張形IC+に変換すると，次のようになる。

IC+: ← R1* (u,v), GT (v,x1), GE (x1,100)

IC+は最初のリテラルが論理式Aを包含し，residueは← GT (y,x1), GE (x1,100) となる。この論理式はより縮小した論理式← GT (y,100) に変換することができる。(縮小変換のアルゴリズムは補遺参照)。したがって，ある問い合わせ← H (x,y) が与えられたとき，residueの存在によってyが100より小さい範囲内で解探索できる。residueがなければEDB全体にわたって解探索しなければならない。

以上のようにresidueは，ユーザからの問い合わせに何らかの論理的意味を与え，解探索の労力を軽減してくれる（解探索空間を狭める）可能性を持っている。さらに別の利点として，residueは事前にデータベースシステム内で求めておくことができるので，問い合わせ時にresidueを求める操作は不要となる。しかし，すべてのresidueが有用な情報を提供してくれるとは限らない。residueが特殊な形をなす場合について，その性質を述べてみる。

(1) null residue

residueが空節になるとき，null residueと言い，これはICが論理的Aの胴部を包含するときに起こる。この場合，論理式Aは成り立たないので，問い合わせに対する解は存在しない。すなわち，論理式が一貫性制約に矛盾することを意味する。

(2) maximal residue

residueがIC+自身になるときで，論理式Aとの相互作用がないことを意味する。

(3) redundant residue

residueが恒真になるときで，節の頭部が真または胴部が偽の場合である。

(1)は問い合わせに対して直接貢献することはないが，論理式のデータベース内での整合性を確認する上で役に立つ。一方，(2)，(3)は問い合わせの改善に対して何ら有用な情報を提供してくれないことがわかる。

〖問い合わせ変換フェーズ〗

まず，問い合わせ変換フェーズを議論するうえで根拠をなす重要な定理を挙げておく。なお証明は省略する。

〔定理4．1〕

ICを一貫性制約，Aを述語Hを定義する（H ← P1*，…，Pn*，項は省略）論理式とする。ICとAから生成されるすべてのresidueは，ICU{P1*，…，Pn*}の論理的帰結である。■

前節で述べたように演繹データベースでは，問い合わせは外延的述語に関する問い合わせの集合に同値変換される。そこでresidueがこのプロセスにどのように導入され，いかなる種類のresidueが問い合わせに組み込まれる得るのか，論じていく。

問い合わせQ，述語Qiに対し編集された論理式Ai（胴部から内包的述語が除去された論理式）が次のような形をしているとする。ここにQiは述語，項tijは変数または定数で，Pij*の項は省略する。

Q: ← Q1（t11，…，t1k1），…，Qn（tn1，…，tnkn）

Ai: Qi（t1，…，tki）← Pi1*（ ），…，Pimi*（ ）

さてここで，定理4．1から論理式Aiにresidue Rijを付与しても論理式Aiの真偽には影響しないことがわかる。よって次式を得る。

Qi（t1，…，tki）← Pi1*（ ），…，Pimi*（ ） {Ri1，…，Riri}

問い合わせの編集とは，編集された論理式Aiを使って内包的述語Qiを外延的述語に変換することであった。そこで上のresidueを付与された論理式を使って，問い合わせQを編集すると，次のようなresidueによって意味的に制約された問い合わせを得る。

SCQ: ← P11*（ ）τ 1，…，P1m1*（ ）τ 1，P21*（ ）τ 2，…，Pnmn*
（ ）τ n {R11 τ 1−，…，R1r1 τ 1−，R21 τ 2−，…，Rnrn τ n−}

ここでτiは，代入{tij/tj; j＝1，…，ki}を意味し，Pi1*，…，Pim*およびRi1，…，Ririに既存の変数を追加する代わりに，これまで使われていない新しい変数を代入することを意味する。

次に，意味的に制約された問い合わせの中でresidueが如何なる役割を果たすかを，residueのタイプ別に一個のresidueの適用を例に考察していく。

(1) 空節

先にも述べたように，residueを生成したIDBの論理式が一貫性制約に矛盾する。

(2)　ゴール節

(イ)residue が問い合わせを包含するなら，問い合わせは解を持たない。

例　Q: ← $Q1^*$ (x^*,a,b)，　R: ← $Q1^*$ (u,a,v)

(ロ)residue がある形式で解を制限する。

例　Q: ← $Q1^*$ (x^*,a,b)，　R: ← GE (x^*,c)

解探索は c より小さい値に制限される。

(3)　単節

問い合わせと residue を組合せて，次の形式を得る。

Q: ← $Q1^*$( )，…，Qn^*( )，R: U( )←

→　← $Q1^*$( )，…，Qn^*( )，U( )

(イ)U が問い合わせの変数を条件付けたり，制限したりする数値述語ならば，それは関係代数における選択条件に相当する。

(ロ)U が問い合わせの中に存在しない非数値述語なら，それは関係代数における結合に相当する。

(ハ)U が問い合わせの中に存在する非数値述語，すなわち U=Qi^* なら，Qi^* と U は問い合わせから削除することができる。

(4)　限定的含意節

residue の胴部が問い合わせに対して真である必要があるなら，解探索は制限される。

例　Q: ← $Q1^*$ (x^*,u,b)，$Q2^*$ (u,v)

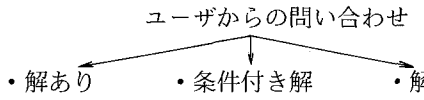
R: LT (u,100) ← $Q1^*$ (x^*,u,b)

→　← $Q1^*$ (x^*,u,b)，$Q2^*$ (u,v)，LT (u,100)

4．4　演繹的質問応答

さて，これまで述べたことを根拠に，実際ユーザとデータベース・システムの間で交わされる質問応答について考察してみる。

データベース・システムにユーザからの問い合わせがあった場合，次のような状況を想定できよう。



通常の関係データベースにユーザからの問い合わせがあった時，解が存在す

れば検索は終了するが，解が存在しない場合，ユーザは検索を中止するか，条件を適当に修正して代替案を調査する。ところが条件の修正は手続き化されたものでなく，経験などに依存することが多いので，検索に時間を要したり，最悪の場合，代替案に到達できない危険性を胎んでいる。

一方，本稿で提示する演繹データベースにユーザからの問い合わせがあったとき，まず意味的な問い合わせの改善が施され，前者より効率的な検索が行われる。解が存在すればそれで質問応答は終了するが，そうでない場合，すなわち条件付き解なら存在する場合，あるいは解がない場合の代替案の検討については，次のような可能性を与えてくれる。

・条件付き解について

これは解の存在がユーザ側の状況に依存することを意味している。すなわちある問い合わせを証明探索木を用いて編集していった結果，¬Q（t1，…，tn）なる外延的でない述語を得たなら，これはIDBには存在するがEDBには存在しない関係（述語）を示しており，EDB内では処理することはできない。そこで，データベース・システムはユーザに条件Q（t1，…，tn）を提示して（もちろん自然言語に変換して），ユーザ側でその条件が満足されるか否かを検討し，満足されるならデータベースにその応答Q（a1，…，an）を提供すれば，当初の問い合わせに対する解を得ることができる。満足されないなら，問い合わせに対する解は存在しないことになる。

・代替案の検討について

ユーザからの問い合わせを編集していった結果，意味的に制約された問い合わせ，例えば$Q1^*$（a,x），$Q2^*$（x,y），GT（y,10），R（R: residue）を得たとする。このとき検索が失敗に終わるということは，residueが満足されないか，述語$Q1^*$（a,x），$Q2^*$（x,y），GT（y,10）が同時に満足されないことを意味している。したがって，ユーザが代替案を要求するなら，データベース・システムを，問い合わせがresidueで失敗したのか，あるいは述語で失敗したのかをユーザに通知できるように設定しておく。ユーザはこの結果に基づいて，residueで失敗した場合には，residueの条件を満たすように問い合わせを変更する。述語で失敗した場合には，問い合わせが述語のどの条件に反しているのか見つけ出す必要がある。この作業は試行錯誤にやると面倒なものになるので（述語がn個あれば，最大2^n-2通り考える必要がある），以下のような方策

を提案する。

まずデータベース・システムは各述語を拡張した形式をユーザに提示し，ユーザはその条件を検討して重要度の順（変更したくない条件の順）に並べる。上の例の場合，たとえば

Q1* (s1,s2)，Q2* (s3,s4)，(s2=s3)，(s1=a)，GT (s4,10)

として，条件を重要度の順に (s2=s3)，(s1=a)，GT (s4,10) と並べるとする。重要度の順に並べた理由は，この順に解探索が行われるので，最初の条件によって解候補集合が決定され，その後は，この解候補集合が条件によって絞られていくからである。要はユーザにとって最も重要な条件が，解候補集合を決定することにある（最初の条件で解が求められなければ条件を根本的に変更する必要があろう)。そして，この手続きを行っていけば，解候補集合を満足しない条件に突き当たるはずである。ここでデータベース・システムはユーザにその条件を提示し，ユーザはこの条件に関する問い合わせの条件を，residue を満たす範囲内で変更する。こうすればユーザ内の最低限望む要求が満足された解が得られることになる。

5　むすび—今後の課題

本稿では，情報資源管理の重要な一側面であるマン・マシン・インタフェースに焦点を当て，関係データベースと知識データベースを組合せた演繹データベース・システムのモデルを提示し，ユーザの要求する情報を効果的に提供するための方法として，演繹的質問応答に関する一つの可能性を考察してみた。これは従来のデータベース・システムに演繹的推論機能を付加することにより，柔軟な検索機能を実現することを目指している。演繹質問応答は，システムとユーザが対話形式で問題を解決していく協同的問題解決プロセスであり，システムはユーザの知識，推論を補う役割を果たしている。それは理想的にはユーザの知らない，あるいは解決できない領域すべてをシステムが担当して，ユーザを支援するものである。しかし，これは実際には不可能なことであって，ここにシステム・デザインの明確な枠組み（担当できる領域）が要求される。関係データベース・システムのデザインに関しては，既にある程度構造化された方法論が確立しているので，比較的障害は小さいであろう。厄介なのは知識ベース・システムのデザイン，特に知識獲得の問題である。知識獲得の問題は二つ

の段階に分割して考えることができる。第一段階は，システムに如何にして最初の知識を与えるかの問題であり，その分野の専門家の協力が必要とされる。第二段階は，既に一定の知識を持つシステムに対して，具体的なデータを与えてシステムの挙動を調べながら知識ベースの内容を修正する問題である。これらの問題は各々で一つの大きなテーマになるものであるから，ここではこの程度の記述に留めておく。

結局，知識ベース・システムはそれ自体，人間的要素まで含めたより大きな全体システムの一部であって，そこでどのような機能が要求されるかは，全体システムとの兼ね合いを考えることなしには決定し難いものである。知識ベース・システムをデータベース・システムに取り入れるに当たっては，全体システムの目的と性格を十分に把握しておかなければならない。知識ベース・システムの開発にはまだまだ克服すべき問題が残されているのが現状である。

〔補遺〕 第4章に関して

(1) 拡張節 C+ は，Cの胴部を次のように変更することによって得られる。

1．定数及び変数を含む数値述語は，それと同値の二つの述語に変更される。たとえば，GT（u,c），は GT（u,x），GE（x,c）で置き換えられる。

2．定数あるいは以前に出現した変数を含む外延的述語は，その定数，変数を新しい変数に変換し，適切な等号を加えることによって変更される。

(2) 縮小節 C− は，Cを次のように変更することによって得られる。

1．頭部が偽の素論理式なら，頭部を削除せよ。

2．Cの胴部において

a．各素論理式の出現は一回にせよ。

b．真なる素論理式は削除せよ。

c．(1)のステップ1．，2．の逆を実行せよ。

(3) 上記(1)，(2)において C+，C−，C は，論理的に同値である（証明は簡単なので略す。）

(4) 定義1

一貫性制約 IC が論理式Aの胴部を包含しないで，IC+ の部分節がAの胴部を包含するならば，IC は部分的にAを包含するという。

(5) 定義2

節Cの部分節Dが性質Pを持ち，かつD⊆D'⊆Cなるすべての節D'に対してD'が性質Pを持つとき，DがD'のすべての非数値述語を含むならば，Dは性質Pに関して極大であるという。

(6) 定義3

上で述べた定義1，定義2によれば，一貫性制約ICと論理式Aから導かれるresidueは，(((IC+−IC+m) σ) −) θ⁻である。ここにIC+mはAの胴部を包含するIC+の極大部分節で，θはAを基礎節にする代入で，σは(IC+m) σをAのインスタンスにする代入である。なお最初の〝−〟は集合の差を表している。

参考文献

(1) 長谷川幸男『多品種少量生産システム』，日刊工業新聞社，1984年。

(2) 島田達巳・海老澤栄一編『戦略的情報システム』日科技連，1988年。

(3) 海老澤栄一編『情報資源管理』日刊工業新聞社，1989年。

(4) 有川節夫『データベースと知識ベース』オーム社，1989年。

(6) Scott, G. M., *Principles of Management Informations Systems,* McGraw-Hill Book Company, 1986.

(7) Jackson, I. F. *Corporate Information Management,* Prentice-Hall, 1986.

(8) Simon, H. A., *The Science of the Artifical, 2nd edition,* MIT press, 1981.(邦訳あり)

(9) Newell, A. and Simon, H. A., *Human Problem Solving,* Prentice-Hall, Englewood Cliffs, NJ, 1972.

(10) Davis, G. B. and Olson, M. H.,*MANAGEMENT INFORMATION SYSTEMS,* McGraw-Hill Book Company, 1984.

(11) Chang, C. L. and Lee, R. C. T., *Symbolic Logic and Mechanical Theorem Proving,* Academic Press, 1973.(邦訳あり)

(12) Lloyd, J. W., *Foundations of Logic Programming,* Springer-Verlag, 1984.

(13) Gallaire,H., Minker,J. and Nicolas J.M., "Logic and Databases", *ARTIFICIAL INTELLIGENCE and DATABASES,* Morgan Kaufmann, 1989.

(14) Chakravarthy, U. S., Grant, J. and Minker, J. "Foundations of Semantic Query Optimization for Deductive Databases", *Deductive Databases and Logic Programming,* 1988.

(15) Reiter, R., "Deductive Qusuion-Answering on Relational Databases", *ARTIFICIAL INTELLIGENCE and DATABASES,* Morgan

Kaufmann, 1989.
(16) Taylor, R. S., *Value-Added Processes in Information Systems*, Ablex Publishing Corpolation, 1986.
(17) Ullman, J. D., *Principles of Database and Knwledge-Base Systems voll*, Computer Science Press, 1988.
(18) CLocksin, W. F. and Mellish, C. S., *Programming in Prolog*, Spriger Verlag, 1987. (邦訳あり)