解　説

5. 演繹データベース演算処理方式†

清　木　　康††

1. はじめに

一階述語論理 (first order predicate logic) を知識表現の基本的な枠組みとし，関係データベース (relational database) との概念の統合により，知識ベースの基本的な機能を実現する演繹データベース (deductive database)[3),4)] の基本演算の処理方式について概観する．演繹データベースを計算機上に実現するためのデータ構造および基本演算について述べ，さらに，それらの基本演算の処理方式についてまとめる．さらに，演繹データベース処理の実現を試みたいくつかのシステムの概要を示す．

演繹データベースは，高水準な問い合わせを可能とするデータベースであり，知識ベースの実現に向けての有望なアプローチとして注目されている．演繹データベースの研究は，1978年に発表された "Logic and Databases"[3)] を契機に活発になり，モデル論，データ構造，問い合わせの処理方式，問い合わせ処理系，並列処理あるいは分散処理システムが提案されてきた．

演繹データベースにおける問い合わせは，関係データベースにおける問い合わせと同様，あるいはそれ以上に負荷の大きい処理を必要とする．したがって，それらの処理を高速に行うために，ソフトウェア技術およびアーキテクチャ技術による支援が必要である．そのような背景から，演繹データベースの問い合わせ処理の高速化を実現する要素技術として，データベースのデータ構造，問い合わせ最適化，問い合わせ処理アルゴリズム，並列処理アーキテクチャが研究されるようになってきた．

関係データベースでは，固定的な基本演算として，関係データベース演算（関係代数）が設定されていたのに対し，演繹データベースにおいては，問い合わせ処理のための基本演算として明確に設定されたものがない．したがって，演繹データベースの処理系の設計時には，問い合わせ処理を行うための基本演算および基本アルゴリズムの設定における選択肢が，関係データベースの場合に比べて多い[1)]．

演繹データベースの問い合わせ処理と関係データベースのそれとの相違は，演繹データベースにおいて再帰的ルール (recursive rule) を扱う必要がある場合に特に顕著となる[1),12)]．演繹データベースに対する問い合わせが再帰的ルールを扱わない場合には，問い合わせは，コンパイルにより関係データベース演算列に変換可能なので，関係データベースの基本演算だけを用いて問い合わせを処理できる[1),5),18)]．一方，再帰的ルールを扱う場合には，再帰処理のための拡張を行わなければならない[1),5),6)]．

2. 演繹データベースのデータ構造

演繹データベースは，節 (clause) の集合により構成されるデータベースである．演繹データベースでは，節は，ファクト (fact)，ルール (rule)，問い合わせ (query) とよばれる3種類に分類される．ファクトの集合は外延データベース (Extensional Database : EDB) とよばれ，また，ルールの集合は内包データベース (Intensional Database : IDB) とよばれる．ここでは，データベースへの格納の対象となる節をホーン節 (Horn clause) に限って説明する．ホーン節からなる演繹データベース（ホーン・データベース）において，ルール，ファクト，問い合わせは，次の形式により表現される．

ルール：

$$h(X_1, \cdots, X_n) \;:\!-\; b_1(Y_{11}, \cdots, Y_{1n_1}), \cdots, b_m(Y_{m1}, \cdots, Y_{mn_m}).$$

ここで，$h(X_1, \cdots, X_n)$，$b_1(Y_{11}, \cdots, Y_{1n_1})$，$b_m(Y_{m1}, \cdots, Y_{mn_m})$ は，リテラル (literal)．h，$b_1, \cdots, b_m$ は述語名 (predicate)，また，$X_1, \cdots, X_n$，$Y_{11}, \cdots, Y_{mn_m}$ は変数である．$h(X_1, \cdots, X_n)$ は頭部 (head)，$b_1(Y_{11}, \cdots, Y_{1n_1})$，

† Processing Strategies for Operations of Deductive Databases by Yasushi KIYOKI (Institute of Information Sciences and Electronics, University of Tsukuba).

†† 筑波大学電子・情報工学系

$\cdots, b_m(Y_{m1}, \cdots, Y_{mn_m})$ は体部 (body) とよばれる．(頭部にある変数は，体部に現れる．)

再帰的問い合わせとは，問い合わせ処理の過程で，再帰的に定義されたルールを扱うものをいう．再帰的に定義されたルールとは，頭部の述語を体部内に含む構造をもつ節のことである．また，複数の節間において，相互に述語が参照される場合は，相互再帰的とよばれる．再帰的ルールの構造を次に示す．

$$p(X_1, \cdots, X_n) \quad :- \quad b_1(Y_{11}, \cdots, Y_{1n_1}), \cdots, \\ p(Z_{m1}, \cdots, Z_{mn_m}), \cdots.$$

ファクト：

$$f(x_1, \cdots, x_s).$$

ファクトは，頭部のみからなるホーン節であり，引数は，定数だけにより構成される．f は述語名であり，また，$x_1, \cdots, x_s$ は定数である．

問い合わせ：

$$:- \ q_1(X_{11} \text{ or } x_{11}, \cdots, X_{1m} \text{ or } x_{1m}), \cdots, \\ q_m(X_{m1} \text{ or } x_{m1}, \cdots X_{mn_m} \text{ or } x_{mn_m}).$$

問い合わせは，体部のみからなるホーン節であり，引数に変数あるいは定数を含む．$q_1, \cdots, q_m$ は述語名である．

演繹データベースの処理系が扱う節群のデータ表現は，次のように大別できる．

（1） 同じ述語名をもつファクトの集合（外延データベース）を，関係データベースの1リレーションとして表現し，ルールの集合（内包データベース）は，そのままホーン節の形式で表現する．このデータ表現は，もっとも一般的である．

（2） ルールもファクトと同様にリレーションとして表現する[19]．各リレーションはホーン節を細分化したリテラルあるいはリテラル群を値とする属性の並びにより構成される．この場合，1リレーションには，頭部の述語名が異なるルール群およびファクト群が共存することになる．

（3） ファクトおよびルールすべてをホーン節の形式で表現する．これは，論理型プログラムにおける表現形式と同様である．

3. 演繹データベースの演算処理方式

2. で示したようなデータ構造を対象とする問い合わせ処理のために，数多くの方式が提案されている．それらは，インタプリテーション方式とコンパイレーション方式の2種類に大別できる．ここでは，2. において分類した(1)のデータ表現を対象とした問い合わせ処理の基本演算を抽出し，それらの演算の処理方式について述べる．

3.1 インタプリテーション方式

この方式は，バックワード・チェイニングによるトップ・ダウン法で問い合わせを実行することを基本とする．

トップ・ダウン法は，問い合わせが与えられると，IDB のルール群を参照しながらルールの体部を展開していくことにより評価を進め，ルール/ゴール木 (AND/OR木)[6),17)] の葉ノード (leaf-node) の部分で EDB のファクト群を操作する．

図-1 に示すように，問い合わせのインタプリテーションを行うインタプリタは，問い合わせとともに，IDB，EDB を参照して，解を求めていくプログラムである．インタプリタは，任意の問い合わせに対応するために，汎用演算子によって構成される．論理型プログラムのインタプリテーションにおいて行われる演算である AND，OR に対応する演算が設定され，それらの演算が，問い合わせ実行時に動的に適用される．

インタプリテーション方式では，問い合わせの実行時に，ルールに対応してルール・ノードが，また，各ルールの体部の各リテラルに対応してゴール・ノードが生成され，最終的には，IDB を反映したルール/ゴール木が生成される．(AND/OR 木と対応させると，ルール・ノードは AND ノード，ゴール・ノードは OR ノードにそれぞれ対応する．) そして，葉ノード (leaf_node) において，ファクト群が集合（リレー

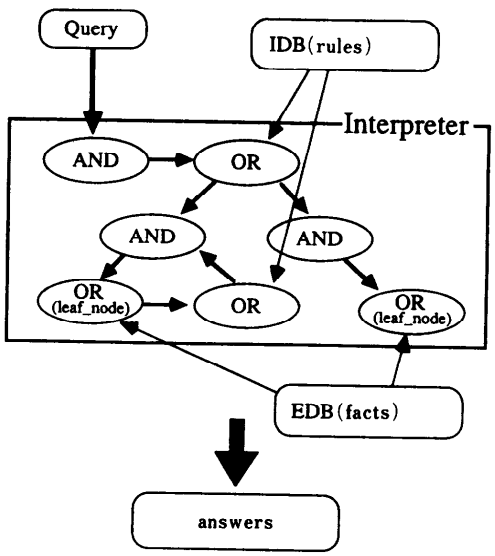

図-1 インタプリテーション方式

ション）として操作される．ファクト群は，木構造中の葉ノードに対応するゴール・ノードにおいてのみ操作される．

問い合わせをインタプリテーション方式により実行する場合，ルール/ゴール木の各ノードに対応して2種類の基本演算（AND およびOR）が用いられる．各基本演算は，論理型プログラムの実行における AND/OR 木の各ノードに対応しているが，論理型プログラムとの相違は，各ノードが，大量のデータからなる集合を扱う点にある．以下に，束縛伝達[1),6),10)]による方法を例として，インタプリテーション方式の基本演算を示す．この方式では，インタプリタへ問い合わせが渡されると，まず，AND 演算が適用される．AND 演算においては，問い合わせを構成する各リテラル，あるいは，ルール体部内の各リテラルの解釈を行うために，各リテラルに対して，OR 演算を適用する．インタプリテーションの過程では，体部全体の解釈のための AND 演算の適用，および，その各リテラルの解釈のための OR 演算の適用を交互に行うことにより，ルール/ゴール木(AND/OR 木）を生成しながら，問い合わせ処理を進めていく．

（1） AND

AND は，ルール，あるいは，問い合わせのインタプリテーションを実行する．AND は，ホーン節体部を受け取り，また，引数の束縛環境（各変数とその変数を束縛 (binding) する値の対応を示す情報）を受け取る．そして AND は，ホーン節体部の各リテラルに対して OR を生成し，それらの OR へ束縛環境を渡す．インタプリテーションは，各 OR 内で束縛環境の書き換えを行うことにより進められる．

（2） OR（中間ノードの場合）

OR は，AND からホーン節体部中の一つのリテラルを受け取り，まず，受け取ったリテラルと同一の述語名をホーン節頭部に有するルールを，IDB の中から検索する．そして，該当する各ルールに対してそれぞれ AND を生成する．そして，各ルールのホーン節体部を各々が対応する AND に渡し，また，各束縛環境と，ルールのホーン節頭部との間で単一化 (unification) を実行することによって新しい束縛環境を生成し，AND へ渡す．

（3） OR（葉ノード (leaf-node) の場合）

葉ノードの位置にある OR は，ホーン節体部中の一つのリテラルを AND から受け取り，また，束縛環境を受け取る．そして，解となる束縛環境を生成する．この OR は，受け取ったリテラルと同一のリレーション名（述語名）をもつタプル（ファクトに対応）と束縛環境の間で単一化を実行する．同一述語名をもつファクト群は，タプル群として EDB の1リレーション中に格納されている．読み込まれたタプル群と束縛環境との間で，単一化を集合演算として行い，その結果として得られた新しい束縛環境を解とする．

このようなインタプリテーションの処理において，OR ノードが生成する AND ノード群間は互いに独立なので，並列にインタプリテーションを行うことが可能である．このような並列性は OR 並列性とよばれる．また，一つの AND が生成する OR ノード群間では，同一の変数には同じ値が束縛されるという制限の中で並列に実行可能である．このような並列性は AND 並列性とよばれる．インタプリテーション方式には，これらの並列性の抽出の可能性がある．また，葉ノードの OR ノードは，リレーションに対する集合演算である．この集合演算を並列処理により実現することが可能である．

3.2 コンパイレーション方式

この方式は，フォワード・チェイニングによるボトム・アップ法で問い合わせを実行することを基本とする．ボトム・アップ法は，EDB のファクト群の操作を行って，中間結果を生成しながら IDB のルール群に対応して演算を進めていく方式である．しかし，探索空間の軽減のために最適化を行うことが一般的である[1),12)]．

コンパイレーション方式では，**図-2** に示すように，コンパイル時には，コンパイラは，問い合わせと，

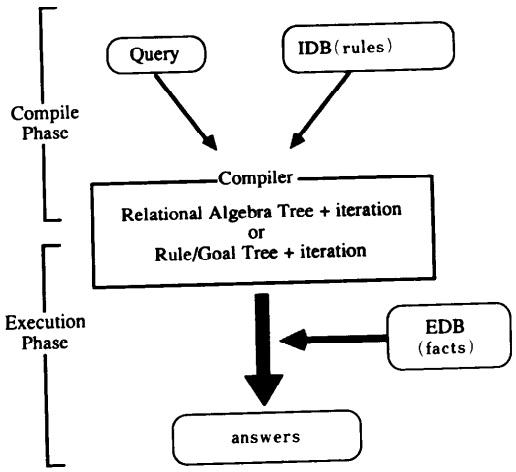

図-2 コンパイレーション方式

IDB だけを参照してオブジェクト・プログラムを生成する．そのオブジェクト・プログラムは，EDB を入力データとして問い合わせを実行する．非再帰的問い合わせの場合には，コンパイレーションにより，関係データベース演算だけから構成されるプログラムに変換できることが知られている[1),5),18)]．再帰的問い合わせの場合には，関係データベース演算群と，それらを繰り返し適用するための制御構造をもつプログラムに変換する必要がある．

オブジェクト・プログラムは，EDB を対象データとする操作群から構成される．EDB を形成するリレーションは，そのプログラムの実行時に参照され，リレーションに対して集合演算が適用される．このように，コンパイレーション方式では，コンパイル時に，問い合わせとルール群を反映した構造の集合演算群からなるプログラムが生成され，そのプログラム実行時に，EDB を形成するファクト群が操作される．インタプリテーション方式が EDB および IDB 両方をデータとして扱うのに対し，コンパイレーション方式は，EDB をデータ，IDB をプログラムとして扱う点が特徴である．

4. 関数型計算による問い合わせ処理

関数型計算の枠組みの中で，演繹データベースへの問い合わせのインタプリテーション，および，コンパイレーション方式を実現する方法がある．それらの研究は，関数型計算モデルにおける並列性抽出，プログラムの記述性，計算機構に関する特徴を，論理型計算を基本とする演繹データベースの問い合わせ処理に適応させようとする試みである．インタプリテーション方式では，インタプリタ自身を構成する演算子が関数として定義され，それらの関数を実行時に動的に適用することにより，問い合わせのインタプリテーションが関数型計算により行われることになる．すなわち，インタプリタ自身が，関数型プログラムとして実現される．

関数型計算によって，演繹データベースへの問い合わせのインタプリテーションを行う場合には，論理型計算を行うインタプリタを関数型プログラムにより記述する．インタプリテーションを関数型計算により実現する方式として，図-3 に示すように，基本演算である AND，OR（中間ノード），OR（葉ノード）を，それぞれ，AND，R_unify，F_unify とよばれる関数として定義し，それらの関数間において，束縛環境およびタプル群をストリームとして受け渡すことにより，並列処理を実現する方式がある[10),11)]．この方式では，関数型計算の評価方式として，要求駆動型評価が用いられ，それと束縛環境群をストリームとしたストリーム処理を組み合わせることにより，並列性の抽出が行われている．

一方，コンパイレーション方式については，コンパイラが関数型プログラムとしてオブジェクト・プログラムを生成することにより，関数型計算によって問い合わせを実行することが可能となる．演繹データベースへの問い合わせが，コンパイレーションにより関係代数の演算列に変換される場合には，関係代数は関数型計算の枠組みに入る代数系であるので，その問い合

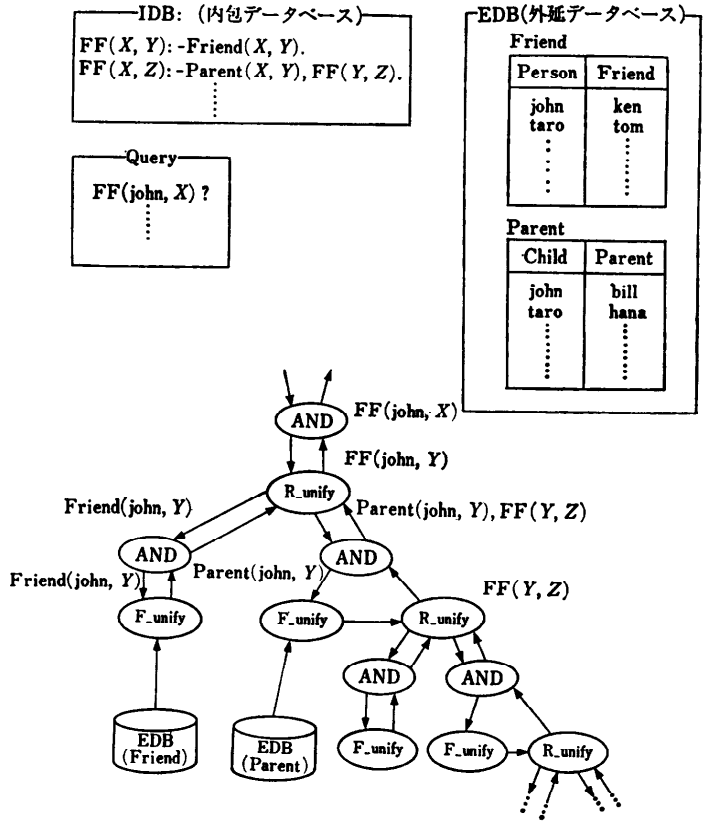

図-3 関数型計算によるインタプリテーション

わせは，関数型計算として実行されることになる．また，図-4 に示すように，演繹データベースの問い合わせにおける各引数への定数の束縛，非束縛の関係から，問い合わせが参照するルール群を関数に変換する方法も提案されている[5]．

5. 演繹データベースの並列処理

インタプリテーション方式において，演繹データベースの問い合わせに内在する並列性の抽出の可能性は，論理型プログラムに内在するホーン節レベルの OR 並列性および AND 並列性に対応する．演繹データベースの問い合わせ処理では，葉ノードの OR ノードにおいて，対象データであるファクト群が大量である点が特徴である．この場合，各ファクト単位の並列性の抽出を行おうとすると，その制御のためのオーバヘッドが並列性をかえって減少させてしまうことになる．したがって，インタプリテーションによる演繹データベースの並列処理方式では，問い合わせに内在する並列性を抽出するために，演繹データベース処理のための基本演算をリレーション（ファクトの集合とみなせる）を対象とした集合演算のレベルに設定し，集合を単位としてそれらの各基本演算あるいは基本演算群を並列に実行することが一般的である．

コンパイレーション方式において，問い合わせが関係代数の演算列に変換される場合には，関係データベースの問い合わせ処理に関する多くの研究によって開発された関係代数の処理アルゴリズム，あるいは，並列処理方式を適用することができる．関係代数の演算列を構成する関係データベース演算は，結合演算，選択演算，射影演算，和演算，積演算，差演算である．これらの演算の処理アルゴリズムおよび並列処理方式は，関係データベース・システム，あるいは，関係データベースマシンの分野において，数多く提案されている．また，演算間に存在する並列性を抽出する方法も提案されている．

一般に，演繹データベースの処理方式では，大きな粒度（グラニュラリティ）の対象データに対する集合演算を行うための並列処理アルゴリズムが用いられている．これらのアルゴリズムは，関係データベースの処理方式の研究により開発された関係データベース演算のアルゴリズム，あるいは，論理型プログラムの並列処理方式を応用あるいは拡張したものとなっている．

IDB: Family-Friend

FF(*X*, *Y*): -Friend(*X*, *Y*).
FF(*X*, *Z*): -Parent(*X*, *Y*), FF(*Y*, *Z*).

EDB: Parent, Friend

Transformation of a query to a function
(query: FF(*a*, *X*)?, a: constant)

Function representation

ff(*X*) = friend(*X*) + ff(parent(*X*))
= friend(*X*) + friend(parent(*X*)) + ff(parent(parent(*X*)))
⋮　　⋮
= friend(*X*) + friend(parent(*X*)) + ••••• + friend(parent^n(*X*)) + •••

Function FF(*X*)

```
Begin
    FF: friend(X);
    DP: parent(X);
  while DP < > φ do
      DFF: =friend(DP);
      FF: =FF+DFF;
      DP: =parent(DP);
  od;
  end;
```

図-4　問い合わせの変換[5]

6. 演繹データベース・システム

演繹データベース・システムは，2. で述べたデータ構造の EDB および IDB を対象として，インタプリテーション方式，あるいは，コンパイレーション方式により問い合わせ処理を実現する．表-1, 2 に，演繹データベースの問い合わせ処理を行うシステムの概要をまとめる．これらの中には，並列処理あるいは分散処理を指向しているシステムも存在する．

関係データベースを対象としたデータベース・システムは，その対象となる基本演算がすでに明確に設定された関係データベース演算であったことから，関係データベース演算の処理の高速化という共通の目標に向けて研究が行われてきた．一方，演繹データベースの場合には，演繹データベース処理の基本演算として明確に設定されたものがないので，演繹データベースを対象としたシステムの設計時には，まず，問い合わせ処理の戦略を決定し，それを実現する基本演算を設定する必要がある．したがって，演繹データベース・システムのアーキテクチャは，関係データベース・シ

表-1-1 演繹データベース・システム（問い合わせ処理方式）

システム名	評価方式	Compilation/ Interpretation	IDB	EDB
Mu-X (ICOT, TOSHIBA) 9)	トップ・ダウン方式．リレーション操作は，Retrieval-By-Unification (RBU) コマンドによって実行される．	インタプリテーション方式．ホーン節インタプリタはGHCによって書かれており，データベースへのアクセスは，RBUコマンドにより行われる．	ホーン節の集合は，2属性（ホーン節頭部，体部にそれぞれ対応している）からなるリレーションとして表現される．	IDBと同様，リレーションとして表現される．
PHI (ICOT, OKI) 8), 9)	ボトム・アップ方式．効率化は，ホーン節変換によるルール書き換えで行う．	コンパイレーション方式．問い合わせとIDBは，関係代数演算および最小不動点演算に変換されて実行される．	IDBは，ホーン節として表現される．これらは問い合わせと統合されたホーン節集合として扱われる．	リレーションとして，複数のLKBM (Local Knowledge Base Manager) に分散して配置される．
DDC (Bull) 7)	ボトム・アップ方式であるが，部分的にトップ・ダウン方式を取り入れている．	コンパイレーション方式．アレキサンダー法によって変換された問い合わせとルールをコンパイルする．問い合わせ結果は最小不動点演算を並列処理することにより得る．	ルール節は，コンパイルされた後，各PCM(Processor, Communication device, Memoryからなるモジュール）ノードにコピーされて保持される．	ファクトは，各PCMノードに，ハッシュ関数により分散されて置かれる．
Tree-structured multicomputer (Northwestern Univ.) 13)	トップ・ダウン方式．Prologと同様に深さ優先探索を行う．	インタプリテーション方式．その動きは，Prologインタプリタとほぼ同じであるが，解を集合として得る．	Prolog-likeな，ホーン節集合として記述される．	リレーションとして表現され，木構造に結合された複数のプロセッサに分散して置かれる．
Surrogate file based knowledge base machine (Syracuse Univ.) 16)	トップ・ダウン方式．ファクトと同様に大量のルールを検索する．コード化による単一化の実行方式が用いられる．	インタプリテーション方式．ただし，単一化は実際のファクトやルールに対してではなく，それらに対応したコード(CCW: Concatenate CodeWord)が格納されたサロゲート・ファイルに対して行われる．	IDBはホーン節頭部，体部に分割されて格納される．すなわち頭部はCCWに変換されてサロゲート・ファイル中に格納され，体部はKBと呼ばれるデータベース中に格納される．	IDBと同じく，CCWに変換され，複数のサロゲート・ファイルに分散して配置される．
OPALE (INPG) 15)	トップ・ダウン方式．処理フェイズをOR, ANDおよびSEARCHの各プロセスに分け，おのおのを組み合わせることによって，問い合わせ処理を実行する．	インタプリテーション方式．各プロセスは，スケジューラ・プロセスによってスケジュールされる．	ルールは，データベース中（ディスク内）に格納される．	IDBと同様，ディスク内に格納されている．
SMASH (Univ. of Tsukuba) 10), 11)	トップ・ダウン方式．問い合わせが参照するルール，ファクト群に対応して，ルール／ゴール木を動的に生成する．	インタプリテーション方式．インタプリテーションは，AND, F_unify, R_unifyと呼ばれる3種類の基本関数を用いて行われる．各基本関数の入出力データは，ストリームとして扱われる．	ホーン節によって記述された各ルールを単位として，表現される．	リレーションとして，二次記憶中に格納され，問い合わせ処理時には複数タプルを粒度とするストリームとして扱われる．
Data/Knowledge Base Management Testbed (Haneywell, National Univ. of Singapore) 14)	ボトム・アップ方式．PCG(Predicate Connection Graph) を生成し，その強連結成分 (clique) を見つけ，評価順序リストにしたがって評価する．	コンパイレーション方式．KM(Knowledge Manager) によって問い合わせをコンパイルし，EDBにアクセスするプログラムを生成し，それをDBMSが実行する．	ルールは，4つのリレーション(isystables, isyscolumns, irulesource, ireachablepreds) として表現される．それらは，Workspace D/KDに移動され，オブジェクト・プログラムに変換される．	リレーションとして表現され，実行時には，Workspace D/KBに転送される．

表-1-2 演繹データベース・システム（システム構成）

システム名	アーキテクチャ構成	ソフトウェア構成	並列性
Mu-X	PIM (Parallel Inference Mechanism) と複数の RE (Retrieval Elements) とこれらを結合するネットワークにより構成される．各 RE は，ホーン節群がリレーションとして格納されている GSM (Global Shared Memory) をアクセスする．GSM はマルチポート・ページ・メモリである．	PKL(Parallel Kernel Language) により記述されたインタプリタの制御下で，リレーション表現されたホーン節が解釈・実行される．また，RBU は，ホーン節内に埋め込まれたコマンドとして実行される．	RBU が RE によって並列に実行される．
PHI	分散処理を指向したシステムであり複数台の PSI を LAN により結合した構成である．	ホスト上のアプリケーション・プログラムから発行された問い合わせを，一台の GKBM (Global Knowlegde Base Menager) と複数の LKBM により分散処理する．GKBM は LKBM が処理する形式の問い合わせ生成をし，LKBM は自サイトで処理を行う．	問い合わせと IDB は，ホーン節変換と強連結成分分解によって内部問い合わせに分解され，それらが複数の LKBM によって並列に実行される．
DDC	DDC は，ネットワークによって結合された PCM ノード群の集合およびホスト・コンピュータからなる．PCM ノードはプロセッサ，コミュニケーション・デバイス，メモリ・デバイスからなる．	ユーザ・プログラムの逐次的な部分は通常の高水準言語のコンパイラによってホストマシンのコードに変換され，並列処理を記述してある部分は，いったん，VIM (Virtual Inference Machine) コードに変換された後，さらに，DDCL (DDC が実行する言語) に変換され，実行される．	ホスト・コンピュータと PCM ノード群に，ファクト群がハッシングにより分散配置され，問い合わせの実行時に複数のルールの処理が並列に実行される．
Tree-structured multicomputer	2分木の構造をもつマルチ・プロセッサ・システムであり，2分木の各ノードがプロセッサに対応している．ルート・ノードは多重プログラミングが可能なミニ／マイクロ・コンピュータであり，その他のノードはマイクロ・プロセッサである．	提案メカニズムを実現するインタプリタは既存の Prolog インタプリタを修正して得られたもので，集合を対象とした推論機構をもつ．	各演算（関係演算に対応）は2分木の構造をもつプロセッサ群によって並列に処理される．
Surrogate file based knowledge base machine	一台の CP (Control Processor) と複数の SFP (Surrogate File Processor) からなる．CPはゴールを各SFPにブロードキャストし，SFP は単一化の結果，得られた結果を共有メモリを介して CP に伝える．ルール節体部のアクセスは CP が行う．	文献16) には明記されていない．	単一化の実行を各 SFP 間で並列に行うことにより，OR 並列性が引き出される．
OPALE	複数のコントロール・プロセッサと複数のプロセッシング・エレメント，それに中間結果格納のための primary memory およびディスク・プロセッサが通信システムを介して結合されている．	OPALE のシステム記述用言語として，実験システムでは Occam が用いられている．	OR 並列性，AND 並列性が引き出される．また複数ディスクへの並列アクセスが行われる．
SMASH	このシステムは，任意のデータベース演算群を関数として定義し，それらを汎用の並列処理システム（複数の汎用プロセッサを高速ネットワークを介して結合したシステム，および，共有メモリ型並列マシン）上で並列に処理する環境を実現する．演繹データベース・システムは，SMASH の一つのアプリケーションとして位置づけられている．	関数型プログラムとして記述された各基本関数をコンパイルする処理系，問い合わせの最適化モジュール，および関数を単位として並列プロセッサへの資源配置を行って問い合わせ処理を行う実行系から構成されている．	ストリームを入力とする関数が要求駆動型制御により，関数型計算の枠組みの中で並列に実行される．AND 並列性，OR 並列性が引き出される．
Data/Knowledge Base Management Testbed	Interface, Knowledge Manager (KM), Run Time Library, DBMS の4モジュールからなる．KM は，ルールを参照しながら問い合わせをCプログラムに変換する．このプログラムはコンパイルされ，Run Time Library とリンクされ，オブジェクト・コードに変換される．DBMS は商用のもので，SQL と embeded SQL インタフェースをもつ．	システムは，商用の関係データベース・システムの上に実現されている．KM は，functional-free Horn clause query を embeded SQL プログラムにコンパイルする．	並列処理システムではない．

表-2-1 演繹データベース・システム（問い合わせ処理方式）

システム名	評価方式	Compilation/ Interpretation	IDB	EDB
CRL (ICOT) 20)	ボトム・アップ方式．sip によりゴールの制約条件を伝播させ，magic set 法によりルールの変換を行い，semi-naive 方式によりボトムアップ評価を行う．	コンパイレーション方式．問い合わせと IDB は，非正規関係の拡張関係代数と最小不動点演算に変換され実行される．	IDB は集合記法をもつ論理型言語 CRL または確定節で表現される．CRL 項は非正規関係の部分クラスである多値関係に対応している．IDB および EDB は，継承関係をもった階層構造として管理されている．	ファクトは，非正規関係データベース管理システム Kappa に格納され，拡張関係代数によってアクセスされる．(ただし現バージョンでは上位層とまだ接続されていない．)
Nail! (Stanford University) 18)	ボトム・アップ方式．	ルール群が，プリプロセッサによりコンパイルされる．問い合わせが発行されると，それに対応するルール／ゴール・グラフを生成し，そのグラフによるファクト群の操作を実現する内部コード (ICODE) を生成する．	ホーン節の形式のルール群によって構成される．ゴールにおけるリテラルには，否定表現が許される．	関係データベースにおいてリレーションとして表現される．
LDL (MCC) 2)	ボトム・アップ方式．	コンパイレーション方式．ルール群は，predicate connection graph とよばれるグラフ（ルール／ゴール・グラフに類似）にコンパイルされる．最適化処理の後に，ルール群は内部表現コードに変換される．	ホーン節の形式のルール群によって構成される．ゴールにおけるリテラルには，否定表現が許される．	リレーションとして表現される．

ステムのそれよりも多様化する．

7. おわりに

演繹データベースの問い合わせ処理を実現するためのデータ構造，および，演算の実行方式について述べた．問い合わせ処理方式をインタプリテーション方式とコンパイレーション方式に分類し，それらの処理系を実現するために必要な基本演算を示し，また，それらの基本演算と並列処理の関連についてまとめた．そして，演繹データベースを対象としたシステムを概観した．演繹データベースの研究は，モデル論，問い合わせ処理方式，問い合わせ処理系の実現方式を中心に行われてきた．現在の段階では，本格的な演繹データベース・システムは実現されていないが，再帰的問い合わせの概念などは，関係データベース・システムにはないものであり，多くのアプリケーションにおいて有効なものとなろう．

謝辞 本解説の記述にあたり，ご討論，ご助言いただいた波内みささん（筑波大学理工学研究科修士課程），宮崎収兄氏（沖電気），ICOT：DOO ワーキング・グループ，情報処理学会データベース・システム研究会，筑波データベース・セミナの皆さまに感謝いたします．また，日頃からご助言いただいている東京大学益田隆司教授，筑波大学中田育男教授，佐々政孝助教授に深謝いたします．

参考文献

1) Bancilhon, F. and Ramakrishnan, R.: An Amateur's Introduction to Recursive Query Processing Strategies, Proc. 1986 ACM SIGMOD, pp. 16-52 (May 1986).
2) Chimenti, D., Gamboa, R. and Krishnamurthy, R.: Towards an Open Architecture for LDL, Proc. 15 th VLDB, pp. 195-203 (Aug. 1989).
3) Gallaire, H. and Minker, J. (eds.): Logic and Databases: Plenum Pub. Co. (1978).
4) Gallaire, H., Minker, J. and Nicolas, J. M.: Logic and Databases: a Deductive Approach, ACM Comput. Surv., Vol. 16, No. 2, pp. 153-185 (June 1984).
5) Gardarin, G. and Maindreville, C. D.: Evaluation of Database Recursive Logic Program as Recurrent Function Series, Proc. 1986 ACM SIGMOD, pp. 177-186 (1986).
6) Gelder, A.V.: A Message Passing Framework for Logical Query Evaluation, Proc. 1986 ACM SIGMOD, pp. 155-165 (May 1986).
7) Gonzalez-Rubio, R., Rohmer, J., Bradier, A. and Bergsten, B.: DDC: A Deductive Database Machine, Proc. 5 th Int. Workshop on Database Machines, pp. 116-129 (Oct. 1987).
8) Miyazaki, N., Haniuda, H., Yokota, K. and Itoh, H.: A Framework for Query Transformations in Deductive Databases, J. Inf. Pro-

表-2-2 演繹データベース・システム（システム構成）

システム名	アーキテクチャ構成	ソフトウェア構成	並列性
CRL	PSI 上の逐次処理システムである．EDB を LAN をとおしてリモート・アクセスできる．現在 EDB については，PIM (parallel inference machine) 上の並列版が開発されている．	ホスト上のアプリケーション・プログラムから，メソッド・コールによって問い合せを出すことができる．知識ベース層は，IDB が CRL か確定節かによって別々に管理されるが，EDB は Kappa で統一されている．	現在，並列版の Kappa が開発されている．そこでは拡張関係代数が並列に実行される．知識ベース層については，階層関係に沿って並列処理可能である．
Nail!	逐次形計算機上で実現されたソフトウェア・システムである．	ルール群をプリコンパイルするプリプロセッサ，与えられた問い合わせを処理するルール／ゴール・グラフから ICODE を生成する Strategy Selection，ICODE のオプティマイザ，および，関係データベースをアクセスする ICODE のインタプリタからなる．	逐次型マシン上に実現されているので，並列性は抽出されない．
LDL	逐次形計算機上で実現されたソフトウェア・システムである．	ルール群を predicate connetion graph へ変換するコンパイラ，そのグラフに対応する関係代数列（特に，結合演算の実行順序）を対象とした最適化により最終的なルール群を生成するオプティマイザ，および，それらのルール群をリレーション (EDB) に適用する処理系からなる．	逐次型マシン上に実現されているので，並列性は抽出されない．

cess., Vol. 12, No. 4 (1989).
9) Itoh, H., Monoi, H., Shibayama, S., Miyazaki, N., Yokota, H. and Konagaya, A.: Knowledge Base System in Logic Programming Paradigm, Proc. 1988 FGCS, pp. 37-53 (1988).
10) Kiyoki, Y., Kato, K., Yamaguchi, N. and Masuda, T.: A Stream-Oriented Approach to Parallel Processing for Deductive Databases, Proc. 5 th Int. Workshop on Database Machines, pp. 102-115 (Oct. 1987).
11) 波内，清木，劉：演繹データベースの並列処理方式の実現と資源割り当て方式，電子情報通信学会 DE 89-19, pp. 9-16 (July 1989).
12) 西尾，楠見：演繹データベースにおける再帰的な問い合わせの評価法，情報処理，Vol. 29, No. 3, pp. 240-255 (Mar. 1988).
13) Qadah, G. Z.: An Interface Model and Tree-Structured Multicomputer System for Large Data-Intensive Logic Bases, Proc. 5 th Int. Workshop on Database Machines, pp. 130-143 (Oct. 1987).
14) Ramnarayan, R. and Lu, H.: A Data/Knowledge Base Management Testbed and Experimental Results on Data/Knowledge Base Query and Update Processing, Proc. 1988 ACM SIGMOD, pp. 387-395 (1988).
15) Sabbatel, G. B. and Dang, W.: Search Strategy for Prolog Data Bases, Proc. 5th Int. Workshop on Database Machines, pp. 654-667 (Oct. 1987).
16) Shin, D. and Berra, P. B.: An Architecture for Very Large Rule Bases Based on Surrogate Files, Proc. 5 th Int. Workshop on Database Machines, pp. 555-568 (Oct. 1987).
17) Ullman, J.D.: Implementation of Logical Query Languages for Databases, ACM Trans. Database Systems, Vol. 10, No. 3 (Sep. 1985).
18) Ullman, J. D.: Database and Knowledge-base systems, Computer Science Press, Vol. II, pp. 987-994 (1989).
19) Yokota, H. and Itoh, H.: A Model and an Architecture for a Relational Knowledge Base, Proc. 13 th Int. Symp. Computer Architecture, pp. 2-9 (June 1986).
20) Yokota, K.: Detabase Approach for Nested Relations, in 'Programming of Future Generation Computers II', North-Holland (1988).

(平成元年 10 月 23 日受付)