Vol. 36 No. 6　　情　報　処　理　　June 1995

解　説

演繹データベースの問合せ最適化技術†

森　下　真　一††

1. はじめに

1980年代から，関係データベースを再帰的に定義される関係を許すように拡張した演繹データベースの研究が盛んになった．演繹データベースの計算は関係データベースに比べてコストが高いため，できるだけ計算の無駄を省くための最適化技術が重要である．本稿では近年明らかになった最適化技術について解説する．

2. Datalog

Coddは関係データベースを提案し，一階述語論理のシンタックスを用いて関係論理を作り，新たな関係を生み出す演算からなる関係代数を定義して実装方法の指針を与え，関係論理と関係代数の等価性を示した[9),10)]．

関係論理は再帰的に定義される関係，たとえば二項関係の推移閉包を表現できない．この表現力の欠如が1970年代後半から強く問題視されるようになり[1)]，より表現力の高い言語がいくつか提案されたが，特にProlog等で使われるホーン節をシンタックスとして利用した言語Datalogが最も利用されるようになった．Datalogは関係論理の自然な拡張であり，関係演算を利用して実装できるので，従来の関係データベースの最適化技術を基盤にして発展してきている．ちなみにDatalogのプログラムは関数記号が現れない論理プログラムである．Datalogを使って推移閉包を表現しよう．

例1：関係e(X, Y)が有向グラフの有向辺の始点と終点を格納しているとする．たとえば点1から点2への有向辺はe(1, 2)，点2から点3への有向辺はe(2, 3)のように表現されているとする．有向グラフの有向辺の推移閉包である2点間のパスを表現する関係p(X, Y)をDatalogでは次のようなプログラムで定義する．

```
p(X, Y)  <-  e(X, Y)
p(X, Y)  <-  e(X, Z),  p(Z, Y)
```

Datalogでは関係を表現するために述語を用いる．最初の節は有向辺はパスであることを意味する．たとえばe(2, 3)からp(2, 3)を推論できる．2番目の節はXからZへ有向辺があり，ZからYへパスがあれば，XからYへパスがあることを意味し，たとえばe(1, 2)とp(2, 3)からp(1, 3)を推論できる．そして新たな定数の組みを生成できなくなるまで推論を繰り返して得られるpの定数の組み全体の集合をpの関係と見なす．□

このような推論の繰返しを最小不動点計算（またはボトムアップ手続き）と呼ぶ．eのようにそれを満たすデータの集合（extension）をすべてデータベース中に明示的に格納する述語（関係名）をEDB（Extensional DataBase）と呼ぶ．一方，pのようにホーン節を使って内包的に（intensional）定義する述語をIDB（Intensional DataBase）と呼ぶ．IDB述語で定義する関係は実体がデータベース中に存在せず，問合せに応じて生成される．

Datalogは関係論理では扱えない問題を表現できるものの，問合せの計算が関係論理に比べて困難になることが知られている[39)]（詳しくはEDBのサイズに関して関係論理がLOGSPACE完全なのに対し，DatalogはPTIME完全であり，前者の計算量は後者より低いことが予想されている）．したがって，問合せQが与えられたとき，DatalogのプログラムをQの計算が少しでも容易なプログラムへと変形してゆく手法があるとありがたい．このような手法は最適化の手法と呼ばれ，以後，代表的な方法について述べる．

† Query Optimization Techniques for Deductive Database Systems by Shinichi MORISHITA (IBM Japan Ltd., Tokyo Research Laboratory).

†† 日本アイ・ビー・エム（株）東京基礎研究所

3. 問合せに関する等価性

ここではp(X, Y)のように，引数がすべて変数である問合せQに関して二つのDatalogプログラムが等価か否かを判定する方法について述べる．p(a, X) のように，引数が定数で束縛されている場合の最適化の手法であるマジックセット法については後に解説する．

まず基本的な場合について述べる．C_1 と C_2 を頭部がQで本体はEDB述語のサブゴールのみからなるホーン節とする．いかなるEDB述語の関係（データの集合）を入力しても，C_1 がQに対して生成する関係が，C_2 がQに対して生成する関係を含むとき，C_1 は C_2 をQに関して包含するという．C_1 と C_2 がQに関して互いに包含するとき，二つは等価であるという．

例2：次の二つのホーン節は等価である．

p(X, Y)　<-　e(Y, X), e(X, Z)

p(X, Y)　<-　e(Y, X), e(X, Z), e(U, X)

上の節は下を包含している．なぜなら上の本体中のサブゴールはすべて下の本体に含まれるからである．一方，下の節も上を包含している．というのも，下の節の変数UをYに置き換えると，本体中のサブゴールはすべて上の節の本体に含まれるようになるからである．□

このように C_1 の変数を C_2 の変数で置き換える写像（ただし頭部の変数は置き換えない）が C_1 の本体のサブゴールすべてを C_2 の本体に含まれるようにできるとき，C_1 から C_2 への包含写像（containment mapping）と呼ばれる．ChandraとMerlinは包含写像を提案し，C_1 が C_2 をQに関して包含するという意味的性質と，C_1 から C_2 への包含写像が存在するというシンタックス上の性質が同値であることを示した[6]．包含写像の存在を判定する問題はプログラムの大きさに関してNP完全である[6]．

例2の下の節からe(U, X)を削除すると上の節が得られる．二つの節は等価なので，e(U, X) は不要なことが分かる．この考え方は一般化でき，本体がEDB述語のサブゴールからなる節が与えられたら，元の節と等価になるようにサブゴールを徐々に除くことで，不要なサブゴールがない極小なホーン節が得られる．

次にDatalogの包含関係の場合へと進む．P_1 と P_2 をDatalogプログラムとする．いかなるEDB述語の関係を入力しても，P_1 がQに対して生成する関係が，P_2 がQに対して生成する関係を含むとき，P_1 は P_2 をQに関して包含するという．P_1 と P_2 がQに関して互いに包含するとき，二つは等価である．まず頭部がQで本体がすべてEDB述語のサブゴールからなる節のみで構成されるプログラムの包含関係の判定法を与える．

例3：次のプログラム について考えよう．

p(X, Y)　<-　e(X, Y), e(Y, Z)

p(X, Y)　<-　e(X, W), e(W, Y)

p(X, Y)　<-　e(X, Y), e(X, U), e(U, Y)

3番目の節を除いても元のプログラムと等価になる．つまり3番目の節は不要である．なぜなら2番目の節のWをUに置き換える写像は3番目の節への包含写像になるので，2番目の節は3番目の節を包含する．したがって3番目の節を除いたプログラムは元のプログラムを包含する．逆の包含関係は明らかである．□

P_2 の各節ごとにそれを包含する P_1 の節が一つ存在するなら，明らかに P_1 は P_2 を包含する．逆は自明でない．なぜなら P_2 のある節を P_1 の一つの節では包含できないが二つ以上の節で共同で包含できる可能性が残されているからである．SagivとYannakakisはこのような可能性がないことを示した[33]．この結果のおかげで P_1 が P_2 を包含することを示すには，P_2 の各節ごとにそれを包含する P_1 の節が一つ存在することをチェックすれば必要かつ十分である．この方法は無限集合についても成り立つが，P_1 と P_2 が有限集合ならば，有限ステップで停止する．

例3で3番目の節を不要な節として除けたように一般には，本体がEDB述語のサブゴールのみからなる節のプログラムから等価性を保ったまま節を徐々に除くことで，不要な節がない極小なプログラムが得られる．

Sagiv-Yannakakisの方法を任意のDatalogプログラムに拡張するには，プログラム P を，頭部がQで本体がすべてEDB述語のサブゴールからなる節のプログラムに変形する必要がある．このためには，Qと単一化可能な節を P から集め，本体がIDB述語のサブゴールを含む限り単一化可能な節の本体で置き換える作業を繰り返せばよい．得られるプログラムは P と等価であり，

Qに対する展開形と呼ぶ．二つのプログラムの等価性を調べるには，二つのプログラムの展開形をつくり，Sagiv-Yannakakisの方法を適用すればよい．

非再帰的プログラムの展開形は有限集合であり，しかも有限ステップで求まるので，二つの非再帰的プログラムの包含関係は決定可能となることが分かる．一方，再帰的プログラムの展開形は無限集合になりうる．したがって，展開形に変形してSagiv-Yannakakisの方法を適用する戦略は有限ステップで停止するとは限らない．実は残念なことに，再帰的プログラム間の包含関係（および等価性）の判定は決定不能である[34]．Shmueliは任意の文脈自由文法からDatalogプログラムを構成する方法を工夫し，再帰的プログラムの包含関係（等価性）の判定を，文脈自由文法が生成する言語の包含関係（等価性）の判定へ帰着し，決定不能性を導いた[34]．

一般には決定不能になるので少し問題を制限し，再帰的プログラムと非再帰的プログラムの包含関係を判定する研究が盛んに行われた．なぜなら再帰的プログラムと等価になる非再帰プログラムが存在する場合があるからである．

例4[24]：次の二つのプログラムは問合せbuys(X, Y)に関して等価である．

```
P1  buys(X, Y)  <-  likes(X, Y)
    buys(X, Y)  <-  trendy(X),
                    buys(Z, Y)
P2  buys(X, Y)  <-  likes(X, Y)
    buys(X, Y)  <-  trendy(X),
                    likes(Z, Y)
```

等価性を理解するために，P_1のbuys(X, Y)に対する展開形をつくろう．

```
buys(X, Y)  <-  likes(X, Y)
buys(X, Y)  <-  trendy(X), likes(Z, Y)
buys(X, Y)  <-  trendy(X), trendy(Z),
                likes(Z1, Y)
      :
```

この展開形とP_2が等価になることはSagiv-Yannakakisの方法を使って示せる．ただし上の展開形は無限集合なので，機械的に有限ステップで等価性を判定できるわけではない．□

幸い，再帰的プログラムと非再帰的プログラムがQに関して等価か否かは決定可能である．CosmadakisとKanellakisは再帰的プログラムが非再帰的プログラムを包含するか否かの判定は決定可能（詳しくはEXPTIME完全）であることを証明し[11]，Ramakrishnan, Sagiv, Ullman, Vardiが分かりやすい判定法を提示した[26]．この判定法は，非再帰的プログラムの各節Cの変数を定数のようにみたて，Cの本体のサブゴールを再帰的プログラムに入力したときにCの頭部が推論される場合に限り，再帰的プログラムはCを包含すると判断する．例4のP_1がP_2の2番目の節を包含することは，この節の変数X, Y, Zに各々定数a, b, cを割り当て，trendy (a)とlikes (c, b)からbuys (a, b)がP_1の中で推論できることから判定できる．

一方，非再帰的プログラムが再帰的プログラムを包含する問題が決定可能であることはCourcelleが示し[12]，ChaudhuriとVardi[7]およびvan der Meyden[20]は独立にこの問題が3EXPTIME完全であることを示した．Chaudhuriらはさらにこの問題を詳細に調べ，判定のための計算がより容易になるサブクラスをいくつか発見している[8]．

このように再帰的プログラムと非再帰的プログラムの包含関係は決定可能だが，最適化の観点からは再帰的プログラムと等価な非再帰的プログラムが存在するか否かを判定し，しかも存在するなら具体的に提示してくれる方法があれば有用である．再帰的プログラムは，それと等価な非再帰的プログラムが存在するとき，boundedと呼ぶ．残念ながら，任意のプログラムがboundedかどうかの判定問題は決定不能であり，Gaifman, Mairson, Sagiv, Vardiらはこの問題を2カウンタ機械の停止性問題に帰着し，決定不能性を導いた[14]．その後Vardiは次の形をしたプログラムに対し，boundedか否かの判定は決定可能（詳しくはNP完全）であることを示した[40]．

```
p(X, Y)  <-  A, p(U, V)
p(X, Y)  <-  B
```

ただしAとBはEDB述語のサブゴールのみを含む．例1や例4はこの形をしたプログラムであり，boundedの判定が可能なクラスは狭いわけではない．boundedの判定が決定可能なサブクラスと決定不能なサブクラスの境界は微妙であり，参考文献15）で詳細に考察されている．

4. マジックセット変形

問合せがp(X, Y)のようにすべての引数が変数のとき，例1のプログラムは無駄なサブゴールを含まず，無駄な節も含まず，等価な非再帰的プログラムも存在しない．ではこれ以上最適化の余地はないのか？

実は問合せがp(a, Y)のようにある引数が定数で束縛されている場合にはさらなる最適化が可能である．このような場合，p(a, Y)を満たすYのデータだけが必要だが，最小不動点計算はp(a, Y)ではなくp(X, Y)を満たすデータをすべて求めてしまう．たとえば例1のeの関係として{e(d, a), e(a, b), e(b, c)}が与えられたとき，最小不動点計算でp(a, Y)を満たすデータを求めようとすると，{p(d, a), p(d, b), p(d, c), p(a, b), p(a, c), p(b, c)} が出力されてしまう．

p(a, Y)を満たすデータだけを求めるのに適した方法として，Prologで使われるトップダウン手続き(SLD-反駁)があり，トップダウン手続きで全数探索すると{p(a, b), p(a, c)}が解として返され，その計算過程でp(b, c)が計算される．しかし最小不動点計算が任意のDatalogのプログラムに対して停止するのに，トップダウン法は必ずしも停止せず，しかも計算に無駄な重複が多いという欠点がある．そこでトップダウン手続きの利点を活かしながら欠点を克服する有力な方法が二つ提案されている．まずTamakiとSatoは，トップダウン手続きの実行時にすでに証明されたサブゴールをテーブルに保持し，その後の計算でこのサブゴールの特殊例が出現したときに計算を打ち切ることで無限ループと無駄な重複計算を回避する方法を提示した[31]．一方，Bancilhon, Maier, Sagiv, Ullmanはプログラムと問合せの組みを，トップダウン手続きでの実行をシミュレートする新たなプログラムへ変形するマジックセット変形を提案した[4]．その後BeeriとRamakrishnanはこの方法を一般化した[5]．マジックセット変形は，例1のプログラムと問合せp(a, Y)から次のプログラムを生成する．

```
mp(a)
mp(Z)   <-  mp(X), e(X, Z)
p(X, Y)   <-  mp(X), e(X, Y)
p(X, Y)   <-  mp(X), e(X, Z), p(Z, Y)
```

始めの二つの節は，問合せp(a, Y)に対して例1のプログラムを使ってトップダウン手続きを実行したときpの第一引数に束縛される定数の集合(マジックセットと呼ぶ)をmpに対して生成する．この集合を用いて，例1の二つの節の推論を絞り込んだのが後の二つの節である．このプログラムを最小不動点計算で評価すると{p(a, b), p(a, c), p(b, c)}が生成される．

Ullman[35]とSeki[30]は独立に，マジックセット変形で得られたプログラムを最小不動点計算で評価すると，トップダウン手続きによる計算と少なくとも同等以上の効率で動くことを示した．このトップダウン手続きに対する優位性がマジックセット変形の利点である．マジックセット変形についてはすでに本学会誌の解説記事[23]で紹介されているので，誌面の都合により本稿での解説は最小限にとどめる．詳細な定義・定理・証明等については参考文献35)，22)等を参照されたい．

5. 否定情報の処理

関係論理では否定記号(本稿では!と記述する)を使用できる．たとえばitem(X) & !broken(X)は，データXがitemに含まれており brokenに含まれていないとき真となる．関係broken(X)に含まれないデータの集合がbroken(X)に含まれるデータの集合より大きい場合，小さい集合の後者がデータベース中に存在すれば，大きい集合の前者はデータベースに格納しなくても単に!broken(X)と記述すればすむ．

Datalogが関係論理の拡張となるためには，否定記号を各節の本体に書けるようにすればよい．ただし，関係論理ではEDB述語だけ否定するが，Datalogの場合はIDB述語も否定することで新たな表現力が得られる．たとえば有向グラフ中の点　X　から点　Y　へパスが存在するものの逆のパスが存在しないことを意図したプログラムは，否定記号!を使って次のように記述できる．

```
one-way(X, Y)   <-  p(X, Y), !p(Y, X)
        p(X, Y)   <-  e(X, Y)
        p(X, Y)   <-  e(X, Z), p(Z, Y)
```

関係論理では否定記号はEDB述語にかかり，EDB述語を満たすデータの集合はデータベース中に格納されているので，あるデータが真か偽はそのデータベースを調べればよい．一方，上のプ

ログラムでは否定記号がIDB述語pにかかり，pを満たす集合は推論して初めて得られる．関係論理での否定記号の取り扱い方を拡張するには，one-wayを推論する前にpを満たすデータの集合を完全に生成してから，あるデータがその集合に含まれるなら真，そうでなければ偽と考えるのが自然である．この考え方は，任意の節を使って推論する前に本体で否定されているサブゴールを満たすデータの集合を完全に生成できるプログラムに一般化できる．Apt, Blair, Walkerらは，各述語が否定記号を通して自分自身を呼び出さないようなプログラムのクラス（通常，層状プログラムと呼ばれる）はこの条件を満たすことを示した[2]．

しかし次のように，否定記号を通して自分自身を呼び出すプログラムの場合うまくいかない．

p(X) <- !p(X)

なぜならこの節を使ってp(X)のデータを推論する前にp(X)を満たすデータの集合を生成できないからである．いま定数aに対して，p(a)にどのような意味を与えたらよいか考えてみよう．p(a)が真と仮定すると!p(a)が偽となるので，真と仮定したはずのp(a)を節から推論できない．一方，p(a)が偽と仮定すると，!p(a)が真になり，p(a)が推論できてしまい仮定と矛盾する．そこでp(a)に真または偽を無理にあてはめるのではなく，この真理値は未定義であると考えればジレンマを回避できる．このような3値のモデルを，述語が否定記号を通して自分自身を呼び出すプログラムに対して提案する研究が1980年代から活発になった．

3値モデルとしては，未定義となるデータの集合ができるだけ小さいモデル，言い換えれば真と偽の領域ができるだけ広いモデルが好ましい．なぜなら，ほとんどのデータに対して真とも偽とも判断を下せないモデルはあまり意味がないからである．しかし，真と偽の領域を広くとり過ぎるとすぐに真と偽の領域が重なり矛盾してしまうので細心の注意が必要である．Kolaitisが作った次の例題は3値モデルの善し悪しを判断するための良いテストケースとなった[19]．

win(X) <- move(X, Y), !win(Y)

この節は，2人のプレーヤが交互に手を指して局面を動かし，次の規則で勝敗が決まるニムゲームを表現している．

●動かす手がない局面に順番が回るプレーヤは負け．

●相手が負ける局面へ動かす手が存在する局面に順番が回るプレーヤは勝ち．

●動かせるすべての局面で相手が勝つ局面に順番が回るプレーヤは負け．

winの節は局面Xから局面Yへ動かす手があり(move(X, Y))，かつ局面Yで順番が回るプレーヤが負ける(!win(Y))ならば，局面Xで順番が回るプレーヤが勝つこと(win(X))を述べており，2番目の勝敗規則を表現している．では1番目と3番目の勝敗規則はどのように表現されているのか？ aを動かす手がない局面とすれば，move(a,Y)を満たすYが存在しないのでwin(a)を推論できない．またaを動かせるすべての局面で相手が勝つ局面とすれば，move(a,Y)を満たすすべてのYでwin(Y)が真となるので，win(a)を推論できない．いずれの場合もwin(a)を推論する可能性のあるすべての場合で推論を妨げる障害があり，この場合 win(a)を偽と考えればwinの節は1番目と3番目の勝敗規則を暗黙のうちに表現していることになる．ただしwin(a)を偽として推論をすすめたときに後にwin(a)を真と推論する矛盾した事態が発生する懸念がある．しかしVan Gelder, Ross, Schlipfはこのような事態が起こらず，推論をすすめると最終的にはニムゲームを表現する3値モデルを作れることを示した[37]．

基礎原子式(ground atom)を推論する可能性のあるすべての節で推論を妨げる障害がある場合にはその基礎原子式は偽と見なすという考え方を一般化して，Van Gelder, Ross, Schlipfはwell-founded model[37]を提唱し，Aptらの層状プログラムのモデル[2]の自然な拡張であることを示し，その後の研究に大きな影響を与えた．

Van Gelder, Ross, Schlipfの提案した初期の方法を使うと，基礎原子式ごとにそれを推論する可能性のあるすべての節で推論を妨げる障害があるかどうかを調べなければならず，現実的でない．Van Gelderは，偽となる基礎原子式の集合の補集合，つまり真または未定義の基礎原子式の集合は最小不動点計算で生成できることに注目し，well-founded modelの簡明な計算方法である

alternating 不動点法を与えた[38]．alternating 不動点法では，真となる基礎原子式の集合に収束する昇順列 $U_1 \subseteq U_2 \subseteq \ldots$ と，真および未定義の基礎原子式の集合に収束する降順列 $O_0 \supseteq O_1 \supseteq \ldots$（ただし O_0 は基礎原子式全体の集合）を次の二つのステップを $i=1,2,\ldots$ について交互に繰り返して生成する．

● O_{i-1} の補集合の元を偽と仮定して最小不動点計算により推論できる基礎原子式の集合を U_i とする．

● U_i の補集合の元を偽と仮定して最小不動点計算により推論できる基礎原子式の集合を O_i とする．

最初のステップでは O_0 は基礎原子式全体の集合なので，U_1 は偽の情報をまったく仮定せずに推論できる，つまり確実に真となる基礎原子式の集合である．この U_1 に含まれない基礎原子式をすべてめいっぱい偽と仮定して推論できる基礎原子式の集合が O_1 であり，それでも推論できない O_1 の補集合の元を偽と見なす．O_1 の補集合の元を偽と仮定して推論される U_2 は，まったく偽の情報を仮定しないで推論される U_1 よりは大きくなる．昇順列 $U_1 \subseteq U_2 \subseteq \ldots$ と降順列 $O_0 \supseteq O_1 \supseteq \ldots$ はある順序数 α 以降は収束し一定になり，U_α を真の集合，$O_\alpha - U_\alpha$ を未定義の集合，O_α の補集合を偽の集合とすれば well-founded model となる（**図-1** 参照）．

2値のモデルの場合，真の集合を求めればその補集合が偽の集合になる．しかし3値モデルの場合，真・偽・未定義のうち二つの集合を求めなくてはならないので，計算のオーバーヘッドが大きい．したがって，与えられたプログラムが2値の well-founded model を持つかを判定できればありがたいが，残念ながらこの判定は決定不能である[25]．そこで，2値の well-founded model を持つプログラムのサブクラスを Ross[32]，Papadimitriou, Yannakakis[25]らが報告している．また3値の場合，2値モデルの場合にも増して問合せ最適化により無駄な計算を省くことが重要になる．そこでマジックセット変形を否定記号を含む Datalog に適用する研究が活発化した．

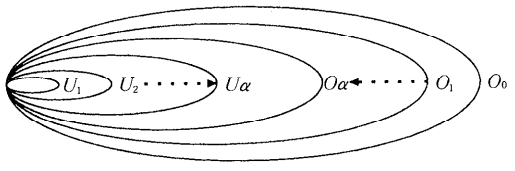

図-1 alternating 不動点法

マジックセット変形を層状プログラムへ適用する場合はさほど大きな問題は起こらないが[16]，一般のプログラムの場合には，マジックセット変形して得られるプログラムと元のプログラムの well-founded model が与えられた問合せの上で必ずしも一致しないという問題が生じる．この問題の原因は，推論を絞り込むために導入されたマジックセットの元が必ずしも真にならず未定義になってしまうことにある．そこでマジックセットの未定義の元を必要な分だけ真の集合に補充する仕組みを Kemp, Stuckey, Srivastava[17]と Morishita[21]は提案し，3値の well-founded model を持つプログラムへもマジックセット変形を適用することに成功している．

6. システム例

これまでに述べた最適化の方法を実装する試みが1980年代後半から数多くなされ，代表的なシステムとしてスタンフォード大学の Glue-Nail[13]，ウイスコンシン大学マディソン校の Coral[27]等がある．ここでは Glue-Nail について触れる．

Glue-Nail は，現実のアプリケーションを構築するのに欠かせない入出力やデータベース更新等の手続きを記述する言語 Glue と，否定つき Datalog を記述する言語 Nail の二つの相補的言語を提供し，これら二つの言語が互いに呼び合うときにできるだけ意味ギャップが起きないように言語設計したシステムである．Glue と Nail で書かれたプログラムは，関係代数の演算に加えて入出力・更新・繰返し文を備えた中間言語 IGlue へ変換され実行される．Nail で書かれた Datalog プログラムは最小不動点計算を実行する IGlue プログラムへ変換されるが，この実行時に IDB 述語の関係は徐々に大きくなる．このように大きさが変化する関係を含んだ演算を効率的に実行するために，IGluc は実行時にジョインの順序を入れ換え，頻繁にアクセスされる属性にはインデックスを張り，インデックスを保守するコストを下回るくらいにアクセスが減ったところでインデックスを除く等の関係データベースで知られている技

術を使って動的な最適化を行う．

なお，10数個の代表的な演繹データベースシステムをサーベイした参考文献として28)があるので参照されたい．

7. おわりに

演繹データベースの問合せ最適化には理論的に深い問題が数多くあるため，これらを解く研究が活発になり現在までに多くの解答が得られた．また演繹データベースのプロトタイプも数多く作成され，配布されつつあるが，商用化されて普及するにはまだ道のりがある．なぜなら，これらのプロトタイプの多くは通常のデータベースシステムが備えているトランザクション処理・障害回復・マルチユーザサポート・制約管理・分散データベースアクセス等の機能を欠いているからである．しかしいくつかのプロトタイプは実用化を意識して作成されている．たとえばGmbHのDeclare[18]は障害回復やトランザクション処理の機能を持ち，メルボルン大学のAditi[36]はマルチユーザをサポートした2次記憶型のシステムである．現在Groupe Bullが商用の演繹データベースシステムの開発を進めているという情報があり[29]，今後の商用化の動きが注目される．

謝辞 貴重な助言をくださった査読者の方々に深謝いたします．

参考文献

1) Aho, A.V. and Ullman, J.D.: Universality of Data Retrieval Languages, Proc. of ACM POPL, pp.110-117 (1979).
2) Apt, K.R., Blair, H.A. and Walker, A.: Towards a Theory of Ddeclarative Knowledge, Foundations of Deductive Databases and Logic Programming (J. Minker ed.), Morgan Kaufmann, Los Altos, CA., pp. 89-148 (1988).
3) Bancilhon, F. and Ramakrishnan, R.: An Amateur's Introduction to Recursive Query-Processing Strategies, Proc. of ACM SIGMOD, pp. 16-51 (1986).
4) Bancilhon, F., Maier, D., Sagiv, Y. and Ullman, J.D.: Magic Sets and Other Strange Ways to Implement Logic Programs, Proc. of ACM PODS, pp. 1-15 (1986).
5) Beeri, C. and Ramakrishnan, R.: On the Power of Magic, Proc. of ACM PODS, pp. 269-283 (1987).
6) Chandra, A.K. and Merlin, P.M.: Optimal Implementation of Conjunctive Queries in Relational Databases, Proc. of ACM STOC, pp. 77-90 (1977).
7) Chaudhuri, S. and Vardi, M.Y.: On the Equivalence of Datalog Programs, Proc. of ACM PODS, pp. 55-66 (1992).
8) Chaudhuri, S. and Vardi, M.Y.: On the Complexity of Equivalence between Recursive and Nonrecursive Datalog Programs, Proc. of ACM PODS, pp. 107-116 (1994).
9) Codd, E.F.: A Relational Model of Data for Large Shared Data Banks, C.ACM, 13, No.6, pp. 377-387 (1970).
10) Codd, E.F.: Relational Completeness of Data base Sublanguages, Data Base Systems, Rustin ed, Prentice Hall, pp. 65-98 (1972).
11) Cosmadakis, S.S. and Kanellakis, P.: Parallel Evaluation of Recursive Rule Queries, Proc. of ACM PODS, pp. 280-293 (1986).
12) Courcelle, B.: Recursive Queries and Context-Free Graph Grammars, Theor. Comput. Sci., 78, pp. 217-244 (1991).
13) Derr, M.A., Morishita, S. and Phipps, G.: The Glue-Nail Deductive Database System: design, Implementation, and Evaluation, VLDB Journal, 3, No.2, pp. 123-160 (1994).
14) Gaifman, H., Mairson, H., Sagiv, Y. and Vardi, M.Y.: Undecidable Optimization Problems for Database Logic Programs, J.ACM, 40, No.3, pp. 683-713 (1993).
15) Hillebrand, G.G., Kanellakis, P.C., Mairson, H.G. and Vardi, M.Y.: Tools for Datalog Boundedness, Proc. of ACM PODS, pp. 1-12 (1991).
16) Kerisit, J.M. and Pugin, J.M.: Efficient Query Answering on Stratified Databases, Proc. of FGCS, pp. 719-726 (1988).
17) Kemp, D.B., Stuckey, P.J. and Srivastava, D.: Query Restricted Bottom-Up Evaluation of Normal Logic Programs, Proc. of Joint Intl. Conf. and Symp. on Logic Programming, pp. 288-302 (1992).
18) Kiessling, W., Schmidt, H., Straussand, W. and Dünzinger, G.: DECLARE and SDS: Early Efforts to Commercialize Deductive Database Technology, VLDB Journal, 3, No.2, pp. 211-244 (1994).
19) Kolaitis, P.G.: The Expressive Power of Stratified Programs, Inf. Contr., 68, No.1, pp. 50-66 (1991).
20) van der Meyden, R.: Recursively Indefinite Databases, Theor. Comput. Sci., 116, pp. 151-194 (1993).
21) Morishita, S.: An Alternating Fixpoint Tailored to Magic Programs, Proc. of ACM PODS, pp. 123-134 (1993).
22) 森下真一：「知識と推論」 共立出版 (1994).
23) 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).
24) Naughton, J.F.: Data Independent Recursion in Deductive Databases, J. Comput. Syst. Sci., 38, pp. 259-289 (1989).
25) Papadimitriou, C.H. and Yannakakis, M.: Tie-Breaking Semantics and Structural Totality, Proc. of ACM PODS, pp. 16-22 (1992).
26) Ramakrishnan, R., Sagiv, Y., Ullman, J.D. and Vardi, M.Y.: Logical Query Optimization by Proof-Tree Transformation, J. Comput. Syst. Sci., 47, pp. 222-248 (1993).
27) Ramakrishnan, R., Srivastava, D., Sudarshan, S. and Seshadri, P.: The CORAL Deductive Database System, VLDB Journal, 3, No.2, pp. 161-210 (1994).
28) Ramakrishnan, R. and J.D.: A survey of Research on Deductive Database Systems, to Appear in J. of Logic Programming.
29) Ramamohanarao, K. and Harland, J.: An Introduction to Deductive Database Languages and Systems, VLDB Journal, 3, No.2, pp. 107-122 (1994).
30) Seki, H.: On the Power of Alexander Templates, Proc. of ACM PODS, pp. 150-159 (1989).
31) Tamaki, H. and Sato, T.: OLD Resolution with Tabulation, Proc. of Intl. Conf. on Logic Programming, pp. 84-98 (1986).
32) Ross, K.A.: Modularly Stratification and Magic Sets for Datalog Programs with Negation, Proc. of ACM PODS, pp. 161-171 (1990).
33) Sagiv, Y. and Yannakakis, M.: Equivalence Among Relational Expressions with the Union and Difference Operators, J.ACM, 27, No.4, pp. 633-655 (1981).
34) Shmueli, O.: Decidability and Expressiveness Aspects of Logic Queries, Proc. of ACM PODS, pp.237-249 (1987).
35) Ullman, J.D.: Principles of Database and Knowledge-Base Systems, Vol. 2, Computer Science Press, New York (1989).
36) Vaghani, J., Ramamohanarao, K., Kemp, D. B., Somogyi, Z., Stuckey, P.J., Leask, T.S. and Harland, J.: The Aditi Deductive Database System, VLDB Journal, 3, No.2, pp. 245-288 (1994).
37) Van Gelder, A., Ross, K.A. and Schlipf, J.S.: The Well-Founded Semantics for General Logic Programs, J.ACM, 38, No.3, pp. 620-650 (1991).
38) Van Gelder, A.: The Alternating Fixpoint of Logic Programs with Negation, J. Comput. Syst. Sci., 47, pp. 185-221 (1993).
39) Vardi, M.Y.: The Complexity of Relational Query Language, Proc. of ACM STOC, pp. 137-146 (1982).
40) Vardi, M.Y.: Decidability and Undecidability Results for Boundedness of Linear Recursive Queries, Proc. of ACM PODS, pp. 341-351 (1988).

(平成6年10月3日受付)

森下 真一 (正会員)

1960年生. 1983年東京学理学部情報科学科卒業. 1985年同大学院修士課程修了. 同年日本アイ・ビー・エム(株)入社, 東京基礎研究所配属. 1990年理学博士(東京大学). 1990～92年スタンフォード大学計算機科学科客員研究員, Nail!プロジェクトに参加. 論理型言語, エキスパートシステム, ジョインアルゴリズム, 演繹データベースの問い合わせ最適化と実装装等の研究を経て現在はデータマイニングの研究に従事. 著書「知識と推論」(共立出版). ACM, 日本ソフトウェア科学会各会員.