

TM-0292

分散知識ベース・システム
PHI における問い合わせ処理

羽生田博美, 高杉哲朗, 宮崎収兄
(沖電気工業)
伊藤英則

March, 1987

©1987. ICOT

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191-5
Telex ICOT J32964

Institute for New Generation Computer Technology

分散知識ベース・システムPHIにおける問い合わせ処理 PHI Distributed Knowledge Base Management System: Query Processing Strategy

羽生田 博美*, 高杉 哲朗*, 宮崎 収兄*, 伊藤 英則**

Hiromi HANIUDA*, Tetsuro TAKASUGI*, Nobuyoshi MIYAZAKI*, Hidenori ITOH**

*: 沖電気工業株式会社, **: (財) 新世代コンピュータ技術開発機構

*: Oki Electric Industry Co. Ltd., **: Institute for New Generation Computer Technology

This paper outlines the basic model of query processing and the query optimization strategy of PHI. PHI is an experimental distributed knowledge base management system being studied in the Fifth Generation Computer System project.

In PHI, the user's queries are expressed in Horn clauses. A Horn clause query is processed by a global knowledge base manager (GKBM) and local knowledge base managers (LKBM). This GKBM-LKBM configuration is generated dynamically for a given query.

PHI's query optimization process consists of two phases: propagation of restrictions and subquery optimization. In the first phase of the optimization, restrictions given in a query or definite clauses in an intensional database are propagated to subqueries, then each subquery with propagated restrictions is optimized by selecting the best processing strategy, based on the analysis of its structure.

1. はじめに

第五世代コンピュータ研究開発プロジェクトにおける分散知識ベース・システム PHIの問い合わせ処理方式について述べる。第2章で問い合わせ処理の基本処理モデルについて、第3章で分散型システムでの最適化とその問題点について、第4章で PHIにおける問い合わせ処理の最適化方式について述べる。

2. 問い合わせ処理モデル

本章では PHIにおける問い合わせ処理の基本モデルについて述べる。

2.1 システム構成

本節では、問い合わせ処理の観点から PHIの論理的システム構成について述べる。

PHI は機能的に知識管理部と分散制御部から構成される [伊藤86]。知識管理部は演繹データベース管理を核とする知識管理層と関係データベース管理層から構成される [宮崎86]。また、分散制御部は分散データベース管理部を核とし、1つのトランザクション・マネージャ (TM) と複数のデータ・マネージャ (DM) から構成される [吉田86]。問い合わせ処理の観点から見た PHIのシステム構成はこれらの方式を基本としている (図1)。

図1の構成は、APからの問い合わせに対応して動的に形

成される。APは PHIが管理する共有知識ベースを利用するエキスパート・システム等の知識処理システムである。

グローバル知識ベース・マネージャ (GKBM)、およびローカル知識ベース・マネージャ (LKBM) は、それぞれ分散データベース制御における TM、DMを機能的に拡張したモジュールである。GKBM は問い合わせに関して全LKBM に共通した処理を行う。LKBMは分散データに密着した処理を、自律的かつ必要に応じて他のLKBMと協調して行う。

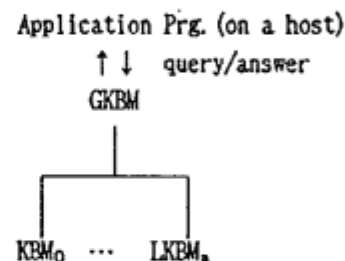


図1 問い合わせ処理におけるPHIの論理システム構成

2.2 問い合わせ処理のスケッチ

本節では上述したシステム構成での問い合わせ処理の概要について述べる。

APから PHIに対してはホーン節形式の問い合わせが送られ [宮崎86, MIYA86]、この問い合わせは1つの GKBMと、複

数の LKBM によって処理される。PHI はコンパイルド・アプローチ [HENS84] に基づいた問い合わせ処理を行う [MIYA86]。また、再帰的問い合わせは最小不動点 (LFP) 演算、すなわち関係代数演算の繰り返し処理で LFP を求めることにより処理される。したがって、PHI の問い合わせ処理は、(a) 問い合わせおよび内包データベース (IDB) に対するホーン節の範囲での処理、(b) その結果に対する LFP 演算の繰り返し処理戦略の決定 [宮崎86] と、関係代数演算の生成処理、(c) その実行の3つの処理に分けることができる。

LFP 戦略の決定と関係代数演算の生成、及びその実行 (b, c) は外延データベース (EDB) に密着した処理であり、LKBM により自律的かつ協調的に処理される。一方、問い合わせそのものと IDB に対する処理 (a) は特定の LKBM だけではなくシステム全体に関係しており、GKBM により処理される。以下、GKBM と LKBM の処理方式について述べる。

2.3 グローバル知識ベース・マネージャ

GKBM は、問い合わせと内包データベースをもとにホーン節の範囲で処理を行い、各ローカル知識ベース・マネージャに対する共通の内部問い合わせを生成する。この内部問い合わせは全 LKBM に放送され、各 LKBM で自律的に処理される。GKBM では次のように処理が行われる。

(1) 問い合わせ及び内包データベースに対する前処理

AP から与えられたホーン節問い合わせと IDB を統合してホーン節集合を生成する。このホーン節集合は AP から与えられる問い合わせとみなすことができ、単にホーン節問い合わせ (HCQ) と呼ぶ。次に、一般に HCQ には引数として定数が含まれるが、HCQ に対する処理を単純にするためにこれらの定数を変数と比較述語に変換する。

(2) ホーン節変換

HCQ に対して、ホーン節変換 (HCT) [MIYA86] を行う。HCT は、一種の部分評価であり、不要な中間述語が削除される。再帰述語は自己再帰的な述語のみが残る。演算データベース処理の効率化は述語数に依存することが多く、HCT により処理の効率化が期待できる。

(3) 強連結成分分解

HCQ 中のルールで、ヘッド述語の評価はボディ部の述語の評価結果に依存していると考えることができ、HCQ はこの依存関係に基づいて有向グラフとして表現できる。このグラフを強連結成分分解する。分解後、各成分間の依存関係は半順序的であり、基本的には個々の成分をゴールから最も遅い側から順に評価すれば解を得ることができる。したがって、この分解により以後の HCQ に対する処理は各成分毎に独立に行うことが可能となる。

また、HCT 後の HCQ に対して強連結成分分解を行うと各成分は、再帰述語数、再帰の線形性、他成分への依存性により幾つかの限られたパターンに分類できる。

(4) 内部問い合わせの生成

HCQ 全体に対して制約条件の伝播に基づいた最適化を、また各成分に対して上述した各強連結成分のパターンに基づいた最適化を行う。この最適化により、HCQ はホーン節形式の範囲で制約を最初に行い限定された空間で探索を行う形式になる。次に、EDB に格納されている関係に対応する述語に対して、その関係の格納サイト情報を付加し内部問い合わせとする。

2.4 ローカル知識ベース・マネージャ

LKBM は GKBM から与えられた内部問い合わせの内、自分に関連する部分のみを自律的に処理する。以下 LKBM の処理について述べる。

(1) 関連成分の識別

LKBM に与えられる内部問い合わせの各成分は、EDB に対する処理、または LFP 演算に関して1つのまとまった処理単位になっている。したがって、各 LKBM はこの成分毎に自分が実行する可能性のある成分を識別する (識別された成分を関連成分と呼ぶ)。この識別は次の方針に従って行われる。

(a) 分散環境では、サイト間のデータ転送コストが全体の処理コストに大きな影響を与えるため、EDB に対する制約処理は格納サイトで行う。

(b) LFP 演算結果の関係の大きさを予測することは困難であるため、LFP 演算の実行サイトは動的に決定する。

(2) ローカル・プランの生成

各 LKBM 毎のデータ・アクセス・プラン (ローカル・プランと呼ぶ) を生成する。まず、関連成分に対して LFP 計算の戦略を決定し、次に各成分に対して関係代数演算列を生成する。ホーン節形式では各ルールのボディ中における述語の順序及びルール自身の順序に意味はないが、これらの順序は関係代数演算列では二項演算の実行順序に対応する。したがって、これら関係代数演算の実行順序の最適化も合わせて行う。

(3) ローカル・プランの実行

(2) で生成されるローカル・プランは、依存グラフの個々の強連結成分に対応する部分問い合わせの集合であり、上述したように半順序関係がある。各 LKBM はこの順序に従って部分問い合わせを処理する。繰り返しを必要とする部分問い合わせに対しては、その問い合わせに関連する LKBM どうしが協調して処理実行サイトを決定する [吉田86]。

3. 問い合わせ処理と最適化に関する問題点

本章では分散型知識ベース・システムでの問い合わせ最適化と、その問題点について述べる。

3.1 分散型システムにおける最適化

演算問い合わせの処理方式には大きく分けて、データベ

スに存在する関係側から順次部分分解の関係を組み立てて行き最終解を得るボトム・アップ方式と、逆に問い合わせ側からルールを展開しながら最終的解を求めるトップ・ダウン方式がある [BANC86]。

PHIのような分散型システムでは、トップ・ダウン方式を基本とした問い合わせ処理方式を採用すると、不必要なサイト間のデータ転送およびインタラクションが増加し、処理コストを増大させる可能性がある。また、問い合わせの効率化が複雑で困難になる可能性もある。したがって、分散環境におけるコンパイル・アプローチに基づくシステムでは、これらの欠点が発生しないボトム・アップ方式を基本とした問い合わせ処理方式が望ましいと考えられる。

先に述べたようにボトム・アップ方式では、問い合わせは互いに依存関係にある部分問い合わせから構成されると考えることができ、最適化の要点は、(a) 制約条件の伝播を部分問い合わせを通じてどこまで行うことが可能か、(b) 再帰的部分問い合わせに対して制約条件をどこまで利用することが可能かの2点である。内包データベースが再帰を含まない場合は、通常の関係代数の最適化と同様の方式が適用できる。再帰を含む場合にも、各部分問い合わせに関して最小不動点 (LFP) 演算子と制約条件が可換な時には [AH079] で述べられている方式が適用できる。また、非可換な場合でも再帰が単純であれば [CERI86, MIYA86] で述べられている方式が可能である。

3.2. ボトム・アップ方式における最適化の問題点

前節で述べた最適化の問題を例で考える。図2に IDB と問い合わせの例を示す (表記は Prolog に準拠)。EDB には IDB 中の各 e_i に対応する関係が存在すると仮定する。

```
?-q(A, constant).
q(A, B):-e1(A, B).
q(A, B):-r1(A, C), q(C, B).
r1(A, B):-e2(A, B).
r1(A, B):-e2(A, C), r1(C, B).
```

図2 十分な最適化が困難な問い合わせの例

このような問い合わせはボトム・アップ方式では、まず述語 $r_1(A, C)$ に対応する関係を求め、次にこれを用いて $q(A, \text{constant})$ に対応する関係を求めることになる。このとき q に対する制約条件 $B=\text{constant}$ は、 q 自身が再帰定義されているので r_1 まで伝播させることができず、 r_1 に対する探索は全探索となり効率的ではない。

また、 q が他の述語と相互再帰的に定義されている、すなわち LFP 演算と制約条件が可換ではない、場合には q に対して全探索を行わなければならない効率的な処理が

できない。

このように、(a) 再帰述語がさらに他の再帰述語に依存している場合には制約条件の伝播を十分に行うことができない、(b) 1つの部分問い合わせに関しても、再帰が相互再帰になる場合には制約条件を探索に利用できないという問題が発生する。

4. PHIの問い合わせ最適化

PHIでは前節で述べた最適化の問題点に対して、(a) ボトム・アップ方式とトップ・ダウン方式の融合による問い合わせ全体の最適化を基本とし、(b) 各部分問い合わせに対しては、前節で述べた方式や制約子付最小不動点 (RLF) 演算等、各部分問い合わせのパターンに応じた最適化を行う。本章ではこの PHI における最適化の基本方式と RLF 演算方式について述べる。

4.1 問い合わせ全体に対する最適化

3.2 節で述べた第一の問題点は、部分問い合わせ間で制約の伝播を十分に行えないことであった。この問題を、図3の性質継承の表現に対する問い合わせを例に考える。ここで、property は isA 関係を通じて継承される。また、edbProperty および edbIsA に対応する関係が EDB に存在すると仮定する。

```
property(Obj, Prop):-edbProperty(Obj, Prop).
property(Obj, Prop):-
    isA(Obj, Class), property(Class, Prop).
isA(Obj, Class):-edbIsA(Obj, Class).
isA(Obj, Class):-
    isA(Obj, SubC), isA(SubC, Class).
```

図3 性質継承の表現例

ボトム・アップ方式では、property に対する問い合わせは isA に対する部分問い合わせと、property 自身に対する部分問い合わせとから構成される。Obj に関する制約条件 (例えば、Obj=penguin) は、単純に isA まで伝播させることができるが、Prop に対する制約条件 (例えば、Prop=canFly) は、そのままでは isA まで伝播させることはできない。しかし、Class の取りうる集合値 (Class|property (Class, canFly)) を与えられれば、これを制約条件として isA まで伝播させることができる。

PHI ではこれを、Obj または Prop に関して、その取りうる値または値の集合を与えられて isA に対する部分問い合わせを解く”問題、として統一的に考える。これにより、問い合わせ $?-property(\text{givenObj}, \text{Prop})$ に対しては、 $?-isA(\text{givenObj}, \text{Class})$ を解き、次に、 $?-property(\text{givenObj}, \text{Prop})$ を解くという完全にボトム・アップ的

動きになる。また問い合わせ ?-property(Obj, givenProp) に対してはproperty(Obj, givenProp)の部分解を求め、次に?-isA(Obj, Class), Class ∈ {Class|property(Class, givenProp)}を求めるプロセスを繰り返すというトップ・ダウン的動きとなる。また、propertyが複数の部分問い合わせに依存している場合にも、propertyが依存する部分問い合わせを、順次トップ・ダウン的に解くことで制約条件を伝播させることができる。

ここで、制約を伝播させられたisA自身に対する問い合わせは、制約条件の形式にかかわらず、3.1節で述べた方式や次節で述べるRLFP方式により、効率的に処理することができる。したがって、PHIではこの融合方式を基本方式とすることにより、問い合わせを効率的に処理することができる。

4.2 部分問い合わせに対する最適化

3.2節で述べた第二の問題点は、各部分問い合わせに関して場合によって制約条件を探索に利用できないことであつた。この問題は、再帰述語が相互再帰の場合や、再帰述語の定義が非線形の場合等に発生する。

PHIではこの問題に対して、制約子付最小不動点演算[宮崎87]を導入することにより対処する。図2のisAに対する問い合わせ?-isA(penguin, Class)を例に考える。この例でisAの定義は非線形であり、このままでは効率的に処理することが不可能である。

各isAを定義するルールの右辺は、isAの探索空間を限定しているとも考えることもできる。したがって、各isAの定義ルールの右辺に、探索空間をさらに限定する節を新たに追加することで、部分問い合わせ?-isA(penguin, Class)を効率化することができる。すなわち、isAの定義ルールの集合に対して、isAの探索空間を限定する節isA*を導入して図4のように修正する。

```
isA*(Obj, Class):-isA*_init(Obj, Class),
isA*_init(penguin, Class).
isA*(SubC, Class):-isA*(Obj, Class),
isA(Obj, SubC).
isA(Obj, Class):-isA*(Obj, Class),
edbIsA(Obj, Class).
isA(Obj, Class):-isA*(Obj, Class),
isA(Obj, SubC), isA(SubC, Class).
```

図4 制約子の導入例

図4におけるisA*の定義ルールは、元のisA定義節のホディ中のisAに対して、他の述語が取りうる値を与えることにより、isAの値の範囲を限定するように定義されている。また、isA自身はisA*により探索空間が狭め

られている。このisA*をisAの制約子と呼び、制約子を導入することにより最小不動点を求める演算を制約子付最小不動点(RLFP)演算と呼ぶ。PHIでは、各部分問い合わせに対して、3.1節で述べたような従来の最適化が不可能な場合、制約子により探索空間を制限することで問い合わせ処理を最適化する。

5. おわりに

分散知識ベース・システムPHIの問い合わせ処理方式について基本処理モデルと最適化方式について述べた。

PHIに与えられるホーン節問い合わせは、1つのグローバル知識ベース・マネージャと複数のローカル知識ベース・マネージャによって処理される。問い合わせ全体に対してはボトム・アップとトップ・ダウンを融合した方式により最適化が行われ、各部分問い合わせに対しては、各問い合わせのパターンに応じて、従来のLFP演算と制約条件の可換性に基づく方式や、制約子付最小不動点演算による方式等最適の戦略を選択することにより最適化を行う。これにより、PHIにおける最適化の問題は、すべての問い合わせパターンに対して戦略決定の問題として対処することができる。今後は、PHIのプロトタイプを試作および評価をPSIにESPを用いて行う予定である。

《参考文献》

- [AHO79] Aho, A. V., Ullman, J. D., "Universality of Data Retrieval Languages," 6th ACM Symp. on Principles of Prog. Languages, pp.110-120, 1979.
- [BANK86] Bancilhon, F., Ramakrishnan, R., "An Amateur's Introduction to Recursive Query Processing Strategies," SIGMOD'86, pp.16-52, 1986.
- [CERI86] Ceri, S., et al., "Translation and Optimization of Logic Queries: The Algebraic Approach," VLDB'86, pp.395-402, 1986.
- [HENS84] Henschen, L., Naqvi, S., "On Compiling Queries in Recursive First-Order Databases," JACM Vol.31, No.1, pp.47-85, 1984.
- [MIYA86] Miyazaki, N., et al., "Compiling Horn Clause Queries in Deductive Databases: A Horn Clause Transformation Approach," ICOT Technical Report TR-183, 1986.
- [伊藤86]伊藤他, "KBMS PHI(1) 分散知識ベースシステムのシステム構成方式," 情処全大会, 2M-5, 1986.
- [宮崎86]宮崎他, "KBMS PHI(2) 知識とデータに関する一考察," 情処全大会, 2M-6, 1986.
- [宮崎87]宮崎, 伊藤, "演繹データベースにおける制約子付最小不動点," ICOT TR-257, 1987.
- [吉田86]吉田他, "KBMS PHI(3) 分散知識ベース制御方式," 情処全大会, 2M-7, 1986.