TM-0150

昭和61年度 電子通信学会全国大会
発 表 論 文 集
3件

January, 1986

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

KBMS PHIにおける Prologとデータベースのインタフェース

Interfacing Prolog and Databases in KBMS PHI

宮崎 収兄* 横田 治夫** 阿比留 幸展* 伊藤 英則**
Nobuyoshi Miyazaki Haruo Yokota Yukihiro Abiru Hidenori Itoh
*沖電気工業株式会社 **財団法人 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

1．はじめに　Prologとデータベース管理システム（DBMS）との結合方法や演繹データベースシステムの研究がさかんになっている。ICOTでは関係データベースマシンDelta を利用した演繹データベースシステムIRISの試作[1] などを行った。現在筆者らは知識ベースマシン研究の一環として
(1) 演繹データベースへの問合せのコンパイル方式の改良
(2) Prologプログラムと演繹データベースの結合
などを研究している。本稿では(2)に関して報告する。

2．PrologとDBMSの結合の必要性　Prologプログラムと DBMSの結合は主として次の2つの理由で必要となっている。
(1) Prologの演繹機能によりDBMS機能を拡張する。
(2) Prologから大容量のDBを使用しDBMSの最適化やDB管理機能を利用する。

現在までこの問題の研究の大部分は(1)の観点から行われている。このアプローチの特徴はPrologとDBMSをあわせて演繹DB機能を実現することにある。

これに対し(2)の観点からの研究はPL/Iにおける埋め込みSQLと同様にプログラムからのデータベースへのアクセスの実現を問題としている。従ってプログラムからはデータが必要になる都度問合せがDBMSに対してなされる。現在までこの観点からの研究が少なかったのはプログラム規模が比較的小さくその必要性が明確に認識されていなかったためと、(1)の観点からの研究により演繹DBが構成できPrologのかなりの機能がこの中で実現できたためであると考えられる。しかしながらPrologにおける制御機能の利用や複雑な構造体の使用などは演繹DBに取り込むよりはプログラムと考えたほうが自然である。例えばappendのような機能はプログラムと考えるほうが良い。従って(2)の観点からの研究も重要になってくる。我々の提案するシステムでは(1),(2) を統一的に扱いPrologに演繹DBのインタフェースを埋め込む方法を採用する。

3．Prologと演繹DBのインタフェース　PrologプログラムとDBMSのインタフェースとしてのデータサブ言語としてSQLや関係論理、関係代数なども考えられるが、ホーン節を用いるのが良い。その長所として、
(1) ホスト言語（Prolog）とデータサブ言語のシンタックスがホーン節で統一されプログラマが記述し易い。
(2) ホーン節により演繹型問合せが記述でき、演繹DB機能が自然に利用できる。
ことがあげられる。

Prologプログラムと関係DBとのインタフェースとしてホーン節を使う方法を[2] で提案した。本稿ではDBMSが演繹DBであり[2] の自然な拡張となっている。問合せ処理の方法としてコンパイルド・アプローチとインタープリティブ・アプローチの両者があり、その優劣が論じられてきた。本稿の方法ではPrologプログラムと演繹DBの関係はインタープリティブ・アプローチであり、演繹DB中での問合せ処理はコンパイルド・アプローチによる。この方法により、
(1) 1つの問合せについては効率の良いコンパイルド・アプローチによって実行する。
(2) 独立な問合せはインタプリティブ・アプローチにより発生の都度処理できるので、演繹DBへの問合せが不必要に複雑にならない。また問合せの発生をプログラマが制御できる。

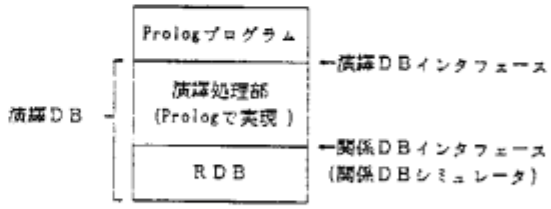

図1　IRIS2の構成

ことになり、両アプローチの長所を生かしたシステムを実現できる。検索や更新はProlog中に演繹DBを起動する述語を使用し、データサブ言語をホーン節により記述することによりなされる。

4．IRIS2　本稿で述べた方法の有効性を実証するためDEC20上に実験システムを試作した（図1）。演繹DBの実現方法としては各種あるが本システムは既にICOTで動作中のIRIS[1] をベースとしたため遅延評価法を用いている。また演繹処理部ではdemo述語によるメタプログラミングの手法を用いている。

IRIS2を使用したプログラムは以下のように書かれる。

```
…, go (idb1, ancestor (X, Y), work_buffer) , …
                    (Prologによるユーザプログラム)
idb1 (ancestor (X, Y)) :- parent (X, Y) .
idb1 (ancestor (X, Y)) :- check_parent (X, Z), ancestor (Z, Y)) .
idb1 (parent (X, Y)) :- edb (parent (X, Y)) .
```

ここでidb1がデータサブ言語の役割をはたす。goは演繹DBへの問合せを起動する述語であり、edb は対象がリレーションとして格納されていることを示している。goが実行される時引数中の変数がインスタンシエートされていれば条件として使われ、変数のままであれば出力変数となる。goによりIRIS2が起動されるとその解は集合で求められるがユーザプログラムには1つずつ変数にインスタンシエートして返される。ユーザプログラム中でバックトラックが起こると次の解が返される。解がなくなるとフェイルしバックトラックする。再度goが実行される時は新しい問合せとしてIRIS2で処理される。

5．まとめ　Prologプログラムと演繹DBとのインタフェースを提案した。本稿で述べた方法を基礎にPHIにおけるKB部とPrologにSP) プログラムとのインタフェースを開発する予定である。

[参考文献]
[1] H. Yokota, et al. : An Enhanced Inference Mechanism for Generating Relational Algebra Queries. ACM-PODS. Apr. 1984.
[2] N. Miyazaki : A Data Sublanguage Approach to Interfacing Predicate Logic Languages and Relational Databases. ICOT TM-001 Nov. 1982.

KBMS PHI における関係データベース管理方式

Relational Database Management in KBMS PHI

山下 祥司* 山中 幸隆* 大場 雅博**
Shouji YAMASHITA Yukitaka YAMANAKA Masahiro OBA
* 沖電気工業株式会社 ** 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

1．はじめに 近年、知識情報処理をめざし、人工知能の研究が広く行われている。筆者等は知識情報処理の要となる知識ベース管理サブシステム研究の一環として、分散知識ベースＰＨＩの研究開発を行っている[1]。本稿ではその基本技術として用いる関係データベースの管理方式の内、コンカレンシィ・コントロールとセキュリティ・コントロールについて報告する。

2．ＰＨＩと関係データベース[2] ＰＨＩの開発目的は、知識情報処理に必要な膨大な量の知識を取り扱う事である。そのため、次の様な機能が必要となる。
・膨大な量の知識を扱う事ができる。
・インタラクティブな使いかたができる。
これらを考慮した結果、ＰＨＩの基本部として関係データベース技術を採用する事にした。

3．コンカレンシィ・コントロール ＰＨＩの基本部である関係データベース（以下基本部と呼ぶ）では複数のユーザがデータを共有するので、データの一貫性を保証する為にコンカレンシィ・コントロールを行う。
分散型のシステムになると、リレーションへのアクセスは増大する事が予想される。特に基本部ではスキーマ情報をリレーションとして保持するので、スキーマ・リレーションへのアクセス要求は飛躍的に多くなる。そこで、それに十分対応できる様に、スキーマを階層化してとらえる。

3.1 データベース構成 基本部のデータベースは、以下の構成をとる。
一般リレーション
一般データが入っている。ユーザが任意に定義、更新できる。
スキーマ情報
一般リレーションのスキーマ情報が入っている。ユーザはデータ定義系コマンドを用いて更新を行う事ができる。
システム情報
システムがもつデータベース管理用の情報が入っている。ユーザに対してはパス・ワードの更新と、自分が定義したリレーションへのアクセス権の付与のみを許している。
図１に構成を図示する。

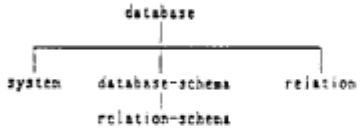

図１．データベース構成

3.2 ロック方式 コンカレンシィ・コントロールとして、基本部ではリレーション名によるプレディケイト・ロックを行ない、特にスキーマ情報に関しては、インテンション・ロック[3]の概念を導入する。
ロックのタイプはX, SIX, IX, S, IS の５種類で、一般リレーションの更新にはＸを、検索にはＳを設定する。スキーマ・アクセスに対してのタイプを表１に示す。但し、database-schema 検索中、更新を行ないたくなった場合は、ロック・タイプをＳからＳＩＸ（S+IXの意）に変更する。ロックの競合を表２に示す。

access type	relation-schema	database-schema
refer	S	IS
update	X	IX
refer db-schema	—	S

表１．スキーマ・アクセスに対する設定ロック・タイプ

	X	SIX	IX	S	IS
X	×	×	×	×	×
SIX	×	×	×	×	○
IX	×	×	○	×	○
S	×	×	×	○	○
IS	×	○	○	○	○

表２．ロック競合表

これによりスキーマ・リレーションに対しては、更新を行う複数のトランザクションからでも同時アクセスが可能となり、増大するアクセス要求にも対処できる。

4．セキュリティ・コントロール 基本部ではデータを統合化して扱うので、データへの不正なアクセスを避けねばならない。そこで、データベース管理システム（DBMS）へのアクセス権とリレーションへのアクセス権をチェックする事によりセキュリティ・コントロールを行う。

4.1 ユーザ・チェック DBMSへのアクセス要求があった場合、正当なユーザであるかのチェックを行う。チェック項目はユーザ名、パスワードである。

4.2 リレーション・アクセス権チェック 4.1 で正当と判断されたユーザであっても、データベース内のすべてのデータにアクセスできるわけではない。各リレーションはその定義者により、定義者以外のユーザに対してのアクセス許可モードを指定されている。これによりリレーションへのアクセス権チェックを行なう。アクセス許可モードは次の３種類である。
C (Close)：検索も更新も許さない。
R (Read)：検索のみ許す。
W (Write)：検索も更新も許す。
また、このチェックにおいては、オーソライズ情報、及びユーザ・グループ情報も考慮する。

5．おわりに ＰＨＩ基本部の管理方式について述べた。基本部は現在詳細設計の段階であり、これからは実現手法についての検討を行ない、より高機能なシステムの開発をめざす予定である。

［参考文献］
[1] 伊藤 他：「KBMS PHI(1)～(4)」，第33回情処全大予稿集
[2] E.F.Codd:"A Relational Model for Large Shared Data Banks", CACM Vol.13 No.6 June.1970
[3] J.N.Gray, et.al.:"Granularity of Locks in a Shared Data Base", Proc.VLDB sep.1975

KBMS PHI における分散問合せ処理方式

Distributed Query Processing in KBMS PHI

吉田勇*　高杉哲朗*　森田幸治**
Isamu YOSHIDA　Tetsuro TAKASUGI　Yukihiro MORITA
* 沖電気工業株式会社　** 財団法人　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

1．はじめに　分散データベースに関する技術は70年代の研究段階から、近年のハードウエア技術等の進歩により、実用段階へと近づいてきた[1]．このため今後のデータベースの構築に当っても信頼性の向上、応答速度の向上等の点から分散化技術をどう実現するかが重要課題となるであろう．筆者等は「KBMS PHI」における分散知識ベース制御メカニズムを検討している．本稿では特に分散問合せ処理方式の検討について報告する．

2．PHI の構成　PHI は図１に示すように複数台のサイトと、これらを結ぶブロードカスト機能を有する高速ローカルネットワーク(LAN) とから構成される．各サイトはTM（トランザクション・マネジャ）とDM（データ・マネジャ）のどちらか一方かもしくはその両方を持っている．１つのトランザクションはユーザサイト内のTMと、そのトランザクションで使用するリレーションを含むサイト内のDMとで構成されるトランザクション通信グループ内で処理される．

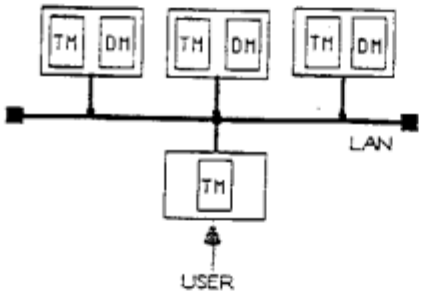

Fig.1 An Overview of PHI

3．モデル　データの分散配置はリレーション単位であり、重複して配置されることはない．リレーションにはユーザ毎に付与できるネームとシステム一意の識別子とが付与される．ネームと識別子の対応表（ディレクトリ）は、リレーションを格納するサイトがそれぞれ分散して保持する非重複分散型とする．この方式はブロードカスト型LAN の利点を活用し、管理オーバヘッドが少なく、サイト自律性の高いディレクトリ管理方式を提供する．

トランザクションは必要なリレーションのネームおよびその使用モードとを付与してスタート・トランザクションをTMが各DMに対してブロードカストすることにより始まり、TMがエンド・トランザクションまたはアボート・トランザクションをブロードカストすることにより終了する．

4．分散問合せ　ユーザからの問合せはTMからトランザクション通信グループに対してコマンド木群としてブロードカストされる[2]．この問合せの中には単一サイト内で処理可能な１サイト内に閉じたコマンド木と複数サイトにまたがる処理を必要とするコマンド木とが通常含まれている．前者のコマンド木については当該サイト独自に処理を行うため一意に決定されるが、後者のコマンド木についてはサイト間のリレーションの転送等を必要とするため、分散データベースにおいては、後者のコマンド木からなる問合せ（分散問合せ）の処理の方式がシステムの性能に大きく影響する．効率のよい分散データベースを構築するためにはこの分散問合せの最適化アルゴリズムが必須である．

分散問合せ処理で重要な点はいつどこにリレーションを転送するかの２点である．従来よりこの分散問合せ処理の最適化については研究されているが、その多くはコンパイル方式などを用いたスタティックなアプローチであり、問合せ実行以前に処理順序を決定している[3),4)]．この方式の利点は問合せの形式が同様のものであれば以前のコンパイル結果を用いることで、コンパイル時間の短縮を図ることが可能である点である．ところがPHI においてはそのディレクトリ管理を非重複分散型としているため、事前にジョイン等の処理の結果リレーションのサイズが不明なため、全体を見渡した最適化には向いていない．そこで筆者らは問合せの実行時に最適な処理順序を決定する．ダイナミックなアルゴリズムを検討している．

5．分散問合せ処理の最適化　筆者等はこの分散問合せのPHI 上での最適化を次の２つの方式で検討した．

(1) ２サイト内で当該リレーションのカーディナリティ（サイズ）を比較し、カーディナリティの小さいものを他方のサイトに転送する．この方式であればそのコマンド処理に関しては当該サイト間のみの情報交換で処理を行うことが可能であり、システム内の他のサイトに影響されないという利点がある．一方でブロードカスト機能を有効利用していないことや、よりグローバルな観点からみれば不必要なオーバヘッドが生じるという欠点があげられる．

(2) まず問合せを解析し幾つかのステージに分解する．その後各ステージにおいて、a)各サイトがリレーションの転送に関する最小コストの処理プランを作成するか、b)各サイトが自サイトに関する最小コストの処理プランを作成しこれを他のサイトのプランと矛盾がないことをチェックする．リレーション転送コストの算出には次の式を用いる．

転送コスト:=(転送回数)* W1 +(リレーションサイズ)* W2
（注：ここでW1, W2はそれぞれの重みを表す）

問合せの処理の実行はこのa), b) のいずれかの方法で作成したプランに従う．これによると(1) で生じるオーバヘッドを減少することが可能である．しかしながらこのプランの作成時間によりこれがオーバヘッドとなる可能性が生じる．

6．おわりに　PHI においてはブロードカスト機能を有するネットワークを持ちかつ、ディレクトリの非重複分散管理を行っているため従来型の問合せ処理とは異なったアプローチが必要である．本稿ではPHI における有効な２つの問合せ処理方式を提案した．今後はこれらの２方式についてそのコスト関数のパラメータの決定などを通して評価しPHI 上に実現してゆく予定である．

［参考文献］

[1] C.Mohan, "Recent and future trends in distributed data base management", Proc. NYU Symposium on New Direction for Data Base Systems, May 1984
[2] 角田　他,「関係代数演算専用エンジンを備えた関係データベースマシンDelta」, 日経エレクトロニクス, No.378, 1985.9.23 pp.235-280
[3] R.Williams, et al., "R*: An Overview of the Architecture", , Improving and Responsiveness, Academic Press, NY, 1981, pp.1-27
[4] D.Rothie, et al., "Introduction to a System for Distributed Databases (SDD-1)", ACM TODS, Vol.5 No.1 pp.1-17, 1980