KBMS PHIにおける知識ベース演算エンジンの評価

An Evaluation of Knowledge Base Engine in PHI

田中 勉　和田 光教　山下 祥司　宮崎 収兄

Tsutomu TANAKA, Mitsunori WADA,
Shouji YAMASHITA, Nobuyoshi MIYAZAKI,

沖電気工業株式会社

Oki Electric Industry Co., Ltd.

A distributed knowledge base system PHI was developed as an experimental system in the knowledge base research of the Fifth Generation Computer Systems project. The knowledge base engine (KBE) is the attached hardware to PHI. KBE employs the superimposed code word scheme to efficiently perform relational operations.

This paper evaluates the performance of KBE in some relational algebraic commands that frequently appear in compiled queries of PHI. As a result, this paper shows that the KBE performs retrieval efficiently, especially in the case of the retrievals that have two or more retrieval conditions on one relation. It also describes the performance of KBE in set operations and set comparisons that are important in deductive databases.

1. はじめに

筆者らは第5世代コンピュータ・プロジェクトの一環として、分散知識ベース管理システムPHIの試作をした。PHIは、ICOT-LANで結合した複数台の逐次型推論マシンPSI上で稼動する実験システムであり、PHIを構成する各分散サイトは、PSIと知識ベース演算エンジン（KBE）を結合したハードウェア上に実現される。

利用者（ホスト）からPHIに対しては、ホーン節形式の問い合わせが出される。ホーン節の問い合わせは、PHIにおいて関連する内包データベース（IDB）中のルールと統合され、外延データベース（EDB）に対する関係代数演算列にコンパイルされる。KBEは、その関係代数演算列を重ね合わせ符号を用いることにより、高速に実行する専用ハードウェアで、PSIとの転送処理（インターフェスは関係代数演算列/リレーション）、関係演算実行といった2つの処理を行う。

PHIにおけるKBEの有効性を調べるためには、KBE単独（転送処理を含めない）で各関係演算ごとの処理性能の評価を行う。第2章ではKBEの概要について、第3章では、測定に用いたデータおよび測定結果について、第4章では測定結果に対する評価を行う。

2. 知識ベース演算エンジンの概要

PHIにおいて、コンパイルされた問い合わせ中で頻繁に出現する関係演算は、選択演算、集合演算、2つのリレーションの包含関係を調べる処理等の、タプルの内容の比較や特定の属性値についての比較を行う演算である。PHIでは、IDBに比較してEDBが巨大であり、大容量のファクトをリレーションとして格納することを想定しているため、これらの関係演算を高速に処理する必要がある。そのため、PHIでは、関係演算を高速に処理するための専用ハードウェアとしてKBEを付加している。KBEは重ね合わせ符号語（SCW）を用いた索引（SCW索引）により、関係演算の高速化を図っている。　SCW索引は、大量のテキスト・ファイルの検索のために提案された方式であり、[ROBE79], [有川83]等でSCWを用いた検索が検討されている。一方、大容量のリレーションに対する検索を高速化する一般的な方法としてB+木やハッシングがあるが、単一の属性に対する検索条件しか扱えない。KBEでは特に複数属性に検索条件が適用されるような検索処理の高速化を実現するため、SCW索引を採用した。　KBEは、図1に示すように、SCWに対する処理を行う専用ハードウェアのACCE, コントローラ, 作業メモリから構成される［Wada88］。

2.1 SCW索引の生成

N個のタプルを持つリレーションr={t1, …, tN}に対して索引を生成する場合を考える。この時、rのスキーマには、Kr個のキー属性と、PHIが提供する属性に応じて属性値を2進符号語(BCW)に写像する関数hが与えられていると仮定する。あるタプルtiに対する索引値は、次の手順で生成する。

(1) tiの各キー属性値をhによりBCWに写像する。
(2) tiのすべてのBCWを重ねて、索引値（SCW）を生成する。

rのすべてのタプルに対して上記の手順で索引値を生成し、対応するタプルへの識別子と組み合わせることで索引

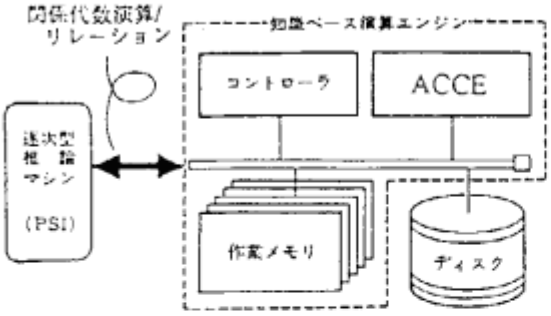

図1 知識ベース演算エンジン

を生成する。

実際のインプリメントにおいては、索引はあらかじめ演算実行前に生成しておき、2次記憶（ディスク）に格納する方式をとっている。又実際の索引は、索引レコードの集合によって構成されており、各索引レコードは、タプルID（リレーションのタプルと1対1の対応をとるための識別子）と対応するタプルのキー属性値から生成した索引値からなる。

2.2 SCW索引方式によるリレーションの検索

問い合わせqに対する検索では、Kq(≦Kr)個のキー属性に対して検索条件を与え、この条件を満足するすべてのタプルを抽出する。KBEでは、その処理を、索引を用いて処理の対象となるタプルを絞り込む段階と絞り込まれたタプルに対する処理の2段階に分けて行う。第1段階では、検索条件として指定された各キー属性値をマッピングしてBCWを作成し、これらを重ね合わせて索引値を生成する。（キュエリマスクQと呼ぶ）

キュエリマスクQと索引値を比較して、索引から式(*)を満たす索引値Siの集合Σを抽出する。

Q and Si = Q ……(*)

第2段階では、Σに対応する候補タプルについて、検索条件を満足するかどうか実際に個々のタプルを検査する。候補タプルの内、検索条件を本当に満足するタプル集合が問い合わせの解である。

実際のインプリメントにおいては、以下の手順で行う。演算実行の際、索引値をメモリ上にロードする。検索条件として与えられたキー属性値をマッピングして、BCW値を作成し、それを重ね合わせてキュエリマスクを生成する。続いて、ACCEにキュエリマスクおよび索引値を渡し、タプルの候補（drops）を抽出する。ACCEからはタプルIDのみが示されるので、実際にそのタプルをディスクからメモリ上に読み込み検索条件を満足しないタプル(false drops)を削除して、その結果を結果リレーションに格納する。

2.3 集合演算と包含関係の検査

集合演算と包含関係の検査では、2リレーション間の同一タプルの検出が主な処理である。KBEでは、これらの処理を2段階で行う。第1段階では、2リレーション間での索引値の比較を行い、同一値を有すると思われるタプル候補の抽出を行う。第2段階では、2リレーションのタプル候補どうしで値の比較を行い、重複するタプルを抽出し、それを基にして目的の演算を処理を行う。

実際のインプメントにおいては、以下の手順て行う。演算実行の際、2リレーションの索引値をメモリ上にロードする。ACCEに2リレーションの索引値を渡し、両索引値で同一の値を有するタプル（タプルの候補）を抽出する。ACCEからはタプル候補(drops)のIDのみが示されるので、タプル候補をディスクから読み込み、タプルの比較処理により重複するタプルを求める。そして、その結果を用いて各演算ごとの処理を行う。

3. 測定

選択演算、集合演算（和、差、積）、包含関係の検査の各演算について、処理性能を調べた。以下に測定に用いたデータ（リレーション,索引）、測定結果を示す。

3.1 データ

(1) リレーション

タプル数2^8～2^{14}、スキーマは選択演算についてはプライマリキーを含め、4, 7, 10, 13属性 、その他の演算については、プライマリキーは含めず、3, 6, 9, 12属性とした。プライマリキーは、タプルIDに相当する属性で属性値はタプルの先頭から末尾へと0から始まる連続な番号を与える。タプル構成については、各属性において特定の属性値が生起する確率（ヒット率）が制御できるようにリレーションの各属性を構成した。

(2) 索引

SCW索引においては、false dropsを小さくしかつ索引のメモリへの読み込みにかかる時間をできるだけ小さくするため、索引値長，索引の重み（SCWにおいてセットされているビットの個数）の設計を必要とする。

KBEにおいて、索引値長は重ね合わせる属性数に依存している。そのため、単位属性あたりの索引値長をパラメータにして設定した。索引値の重みは、索引値長の1/2の場合に絞り込み能力は統計的に最良（false dropsが最小）となる[Robe79]。索引値の重みは索引値を構成しているビットの個数（BCWの重み）に依存している。KBEでは、BCWの重みの上限値を与えると、その重み上限値の近傍にピークをとるようなBCWを生成する関数を用いている。本測定においては、その関数を用いて索引値の重みが索引値長の1/2となる重み上限値を決定し、その重み上限値の近傍を重み上限値のパラメータとした。測定結果の図では、これらの値を索引パラメータとして（属性あたりの索引値長/重み上限値）の形で表す。

3.2 測定結果

(1) 選択演算

2.2で示したように、選択演算は、索引をディスクからメモリ上に読み込むのに要する時間（索引読み込み時間）、ACCEによってdropsを抽出するのに要する時間（ACCE処理時間）、タプルをディスクからメモリ上へ読み込むのに要する時間（タプル読み込み時間）で構成される。図2で

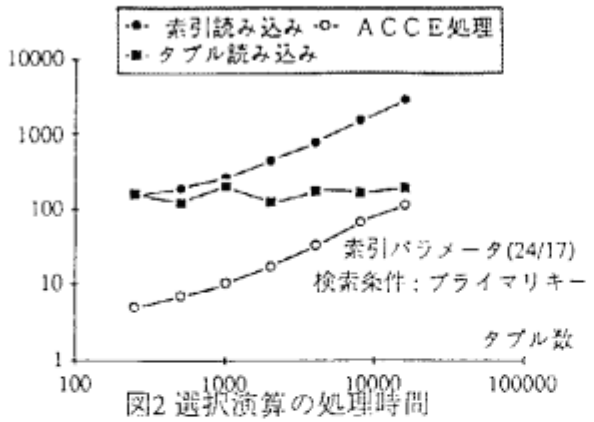

図2 選択演算の処理時間

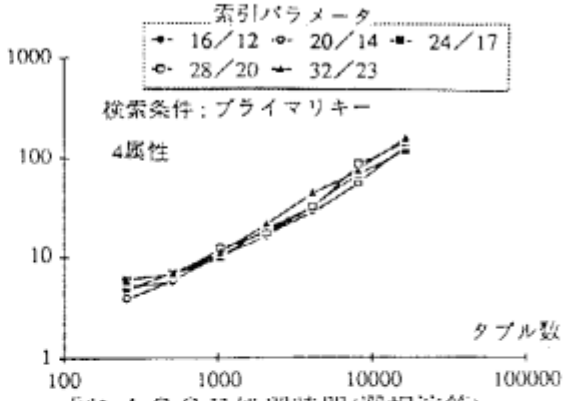

図3 ＡＣＣＥ処理時間(選択演算)

は、その各処理時間の比率を調べるため、タプル数を変化させた場合について測定した。その内、索引値長の変移によるACCE、タプル読み込み処理の影響を調べるため、図3,4に示すように属性あたりの索引値長と重み上限値の組をパラメータにして、タプル数を変化された場合のACCE処理時間、タプル読み込み時間を測定した。図5では、索引値長を固定して重みのみを変化させた場合のACCE処理時間への影響および複数タプルの検索におけるACCE処理時間の影響を見るため、ヒット率をパラメータにして重み上限値のみを変えた場合のACCE処理時間を測定した。又、複数属性に対する検索条件を指定した検索の適性を調べるため、図6には検索条件をプライマリキー、1属性、2属性、全属性を指定した場合のタプル読み込み時間を測定した。

(2) 集合演算および包含演算の検査

2.3で示したように、本演算の処理コストは索引読み込み時間、ACCE処理時間、タプル比較処理時間（タプル読み込み時間を含む）で構成される。演算全体の処理能力は重複するタプルの比較処理であるため、そのタプル読み込みに要するディスクアクセス回数に大きく依存する。そこで、比較処理が最大となる同一リレーションの場合（同一のリレーション間での演算）と一般リレーションの場合（同一でないリレーション間での演算）について、タプル数を変移させた場合の処理コストを測定した。その内、図7,8には集合和演算の測定結果を示した。

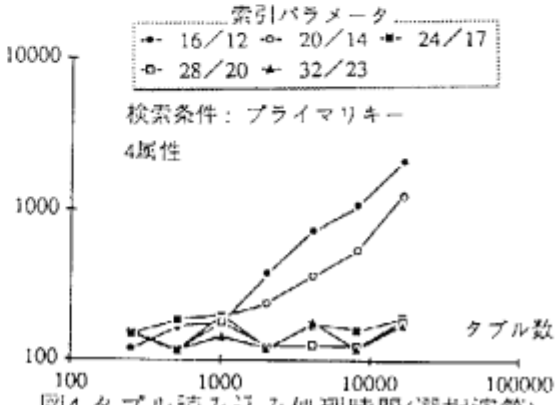

図4 タプル読み込み処理時間(選択演算)

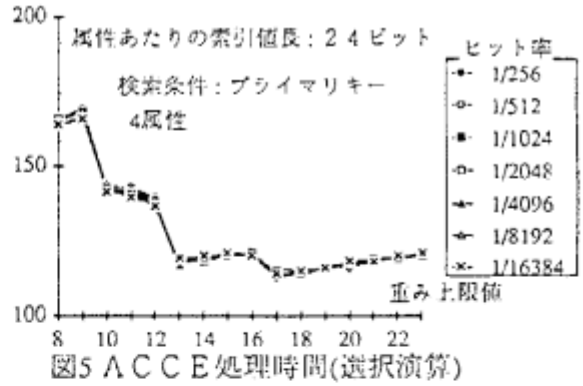

図5 ＡＣＣＥ処理時間(選択演算)

4. 評価

4.1 選択演算

索引読み込み時間、ACCE処理時間はタプル数に対して線形に変移している。処理コストの割合は、索引読み込み処理が処理の大半を占めており、その比率はタプルの増加に伴い、顕著に現われている（図2）。

索引値長によるACCE処理時間の影響に関しては、ACCE処理時間は索引値長に対して影響を受けていないことがわかる（図3）。これは、ACCEでは索引値をワード(16ビット)単位で先頭から順次比較するアルゴリズムを採用しているため、キュエリマスクを満足しないことが判明した時点で次の索引値を比較することによる。

索引パラメータによるタプル読み込み時間の影響については、索引パラメータ(24/17)以上とすることで、タプル

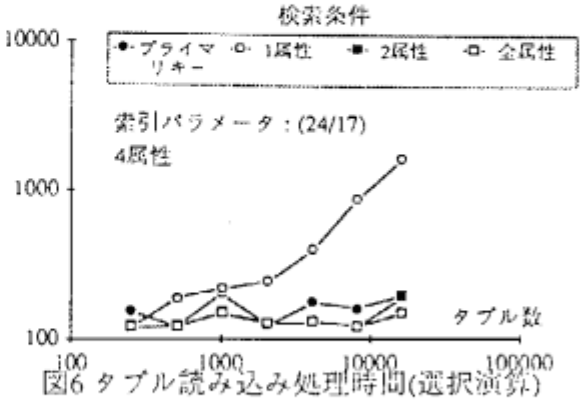

図6 タプル読み込み処理時間(選択演算)

読み込み時間は大きく減少している（図4）。これは、索引パラメータが24/17以上では、false dropsが大きく減少するため、その結果がタプル読み込み時間に反映されていると考えられる。

重みのみを変移させた場合のACCEについては、ACCE処理時間は重み上限値の値が17付近を最小値をして下に凸のグラフを示している（図5）。又、ヒット率によるACCE処理時間の影響はほとんどない。これは、false dropsの個数を見た場合、本パラメータでは、ヒット率の変化させた場合より重みによる変化の場合の方がその値の取り得る範囲がはるかに大きいためであると考えられる。そのため、本パラメータにおいては、複数タプルの検索におけるACCE処理時間は、検索結果のタプルの個数（解の個数）にほとんど影響を受けないことがわかる。

図4.5より、十分な索引値長と適切な重みの設定により、タプル読み込み時間、false dropsを十分小さくできることがわかる。

複数属性に対する検索条件を指定した検索の適性については、検索条件として1属性を指定した場合に比べ、2属性を指定することで、タプル読み込み時間の値はタプル数16384の時で1/8程度小さくなっている（図6）。これは指定する属性の個数を増加させることにより、false dropsが減少し、そのことがタプル読み込み処理時間に反映しているためである。この例では属性数の個数を2以上にすることによりfalse dropsの個数が0となる。

4.2 集合演算および包含関係の検査

処理時間比率に関しては、索引の読み込み処理、ACCE処理、タプルの比較処理の3つの処理の内、索引の読み込みとタプルの比較が処理時間の大半（約95%）を占めている。

索引読み込みとタプル比較については、同一リレーションの時は同程度になっているが、一般リレーションの時はタプル比較に比べて、索引読み込みが大きな比重を占めている。これは、同一リレーションのdropsは全タプルとなるが、一般リレーションのdropsは非常に小さいため、比較処理で読み込むタプルの個数の差がでていると思われる。

ACCE処理については、処理が軽い（dropsが非常に小さい）一般リレーションの値に比べて、最も処理が重い（dropsが全タプル）同一リレーションの値は2.5倍程度であるが、ACCEは処理が重くなっても大きな処理時間の増加はないことがわかる。

実際のPHIにおいては、KBEに対して演算列として複数の演算が与えられる。この場合には、索引読み込み、タプル読み込みで大部分を占めるディスクからメモリへの転送処理は、前の演算でメモリにロードした索引やタプルを利用することによって小さくできる。このことと先に述べたACCEの処理が索引の読み込みやタプルの比較処理に比べて非常に小さく負荷の違いによる影響が小さいことを考慮すると、PHIにおいてはKBEの集合演算は有効であると考えられる。

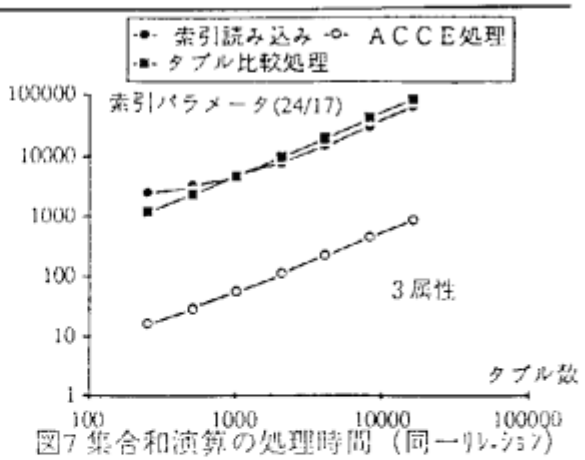

図7 集合和演算の処理時間（同一リレーション）

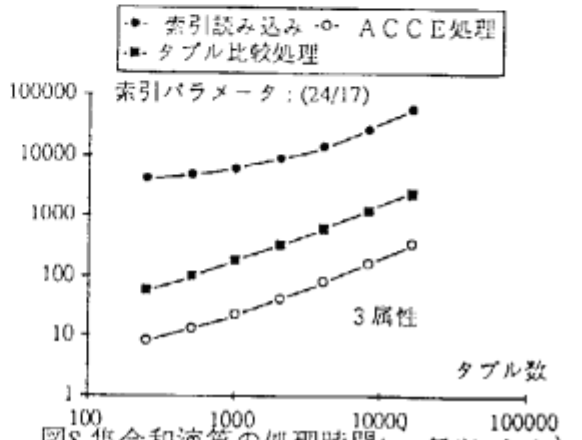

図8 集合和演算の処理時間（一般リレーション）

5. おわりに

KBEにおけるSCW法の評価を行った。選択演算については、十分な索引値長と適切な重みの設定によってタプル読み込み時間、false dropsは十分小さくできた。又、検索条件として指定した属性数の増加に伴い選択率は向上した。その結果、PHIにおけるKBEの目的「複数属性に属性条件が適用されるような検索処理の高速化」に対して、KBEの選択演算が有効であることが確認された。

集合演算および包含関係の検査については、ACCEの処理時間について負荷が軽い場合ともっとも重い場合を比較し、その差が大きくないことからPHIにおけるKBEの有効性が確認できた。今後は、今回の結果をデータベースのシステム実用化の研究に生かしていく予定である。

【参考文献】

[有川83] 有川 他 "重ね合せ符号と逐次サーチを利用する文献情報検索システムについて(2)", 情報処理学会第27回全国大会, 7K-8, 1983

[Robe79] Roberts, C.S. , "Partial-Match Retrieval via the Method of Superimposed Codes", Proc. IEEE, Vol. 67, No. 12, pp. 1624-1642, 1979

[Wada88] Wada, M., et. al. , "A Superimposed Code Scheme for Deductive Databases", in eds. Kitsuregawa and Tanaka, in Database Machines and Knowledge Base Machines, Kluwer Academic Publishers, 1988