

トランスポーズ形ファイルで蓄積した関係に対する関係演算†

佐藤 隆 士** 津田 孝 夫***

関係データベースは、論理的には関係と呼ばれる表形式データの集りである。物理構造(計算機への格納構造)としては、従来、この表を行方向(タプル単位)に蓄積し、検索効率化の手段としてインデックスファイルを用いる方法をとってきた。しかし、このような物理構造には、インデックスファイルのない属性(表の列)に対して検索効率が著しく悪いこと、インデックスファイルのため大きな記憶領域が必要なこと、データ更新の際、インデックスファイルの修正が必要であることなどの問題点がある。本論文では、インデックスファイルを用いず検索を高速化する方法の提案を行う。高速化のための工夫は、トランスポーズ形ファイル(表を列方向に蓄積する方法)を用いることにより、関係演算に含まれる属性のみにアクセス可能としたこと、個々の関係演算だけでなく質問(演算列)としての高速化の工夫をしたことなどである。提案の方法は、今日、ますます必要とされている。小さくても操作性、検索効率のよいデータベースに適当な方法であると考ええる。さらに、本論文では、ページプリフェッチを用いて、検索をより高速化する方法について述べ、提案の方法全体についての計算機実験の結果を示す。

1. ま え が き

関係データベースは、論理構造の明確さ、柔軟なデータ操作などのため注目されている。関係データベースは、論理的には関係と呼ばれる表形式データの集りであるが、物理構造(計算機への格納構造)としては、従来、この表を行単位(行方向)に蓄積し、検索効率化の手段として、インデックスファイルを用いる方法をとってきた^{4),5)}。しかし、このような物理構造に対しては次のような問題点がある。

i) 一般的に、インデックスファイルはすべての属性(表の列)には付けられていないため、インデックスファイルのない属性に対する検索は表全体のサーチを必要とし、著しく効率が悪い。

ii) インデックスファイル記憶のため、大きな記憶領域を必要とする。このため、データベースの大きさが、しばしば、論理的な関係の2倍以上のデータ量となる。

iii) データ更新の際、インデックスファイルの修正が必要となる。このため、データの更新が頻繁なデータベースに対しては好ましくない。

本論文では、インデックスファイルを用いず検索を高速化する方法の提案をする。高速化のための工夫

は、i) トランスポーズ形ファイルを用いることにより、関係演算に含まれる属性だけにアクセス可能としたこと、ii) データベースに対する質問は、一連の関係演算(演算列)で表されるが、各演算の中間結果をタプル識別番号で表現することによりコンパクト化し、中間結果の入出力に要する時間を小さくしたこと、iii) 属性値の取出しを、その属性を必要とする操作の直前まで遅らせることにより、その属性を蓄えるトランスポーズ形ファイル全体へのアクセスを必要とせずに処理できるようにしたことなどである。とくに、ii), iii)は個々の演算だけでなく、演算列としての高速化の試みである。

今日、より大きなデータベースが要求される一方で、ますます、小さくても操作性、検索効率のよいデータベースが必要とされている^{*}。とくに後者に対して、本論文提案の方式は適当であり、インデックスファイルを使用しなくとも十分高速であり、かつインデックスに関する問題を生ぜず、操作性のよいデータベースが作製可能となると考える。

以下、2章は用語の説明である。3章では、関係演算の実行方法について具体的に述べる。4章では、一般によく用いられている関係を行単位に蓄積する方法をとった場合と、3章で述べた方法との比較を行う。なお、比較は仮想記憶でのページフェッチ数をコストとし、詳細に行っている。また、5章では、ページプリフェッチにより演算速度をさらに高速化する方法を

† Relational Operations to the Relation Stored in Transposed Files by TAKASHI SATO (Department of Information Engineering, Takuma Radio Technical College) and TAKAO TSUDA (Department of Information Science, Kyoto University).

** 鹿間電波工業高等専門学校情報工学科

*** 京都大学工学部情報工学科

* たとえば、参考文献 10) p. 100 でいう、数千レコードのデータベース。

検討する。そして、6章では、3, 4, 5章の方法について計算機で実験を行った際の測定結果を示す。

2. 諸 定 義

論理的には、関係は表の型をしており、各列には属性の名がついている。そして、列の記入事項はその属性の定義域から取り出される。表の各行はタプルと呼ばれる。

本論文では関係代数を用いるが、扱う関係演算は、選択、(擬似)射影、結合のみとする^{*}。各演算は次のように定義される。

(1) 選択：関係 R から属性 A の値と定数 c が θ (θ は論理演算子であり、 $=, \neq, \geq, \leq, >, <$ のうちどれか) であるタプルからなる関係を作る演算である。

$$\sigma_{A\theta c}R = \{r | r \in R \wedge (r[A]\theta c)\}$$

ここで、 $r[A]$ はタプル r のうち属性 A の成分を表す。

この場合、属性 A のことをとくに選択属性という。選択は関係 R の二つの属性 A, B に θ が成立するタプルからなる関係を作る場合にも用いられ、

$$\sigma_{A\theta B}R = \{r | r \in R \wedge (r[A]\theta r[B])\}$$

である(この演算は制約と呼ばれることもある)。この場合の選択属性は A と B である。

(2) 射影：関係 R の全属性集合を X とするとき、 $Y \subset X$ のみの属性からなる関係を作る演算である。

$$\pi_Y R = \{r[Y] | r \in R\}$$

Y に含まれる属性を射影属性という。関係はタプルを要素とする集合であるため、属性集合 Y の成分が同じタプルは、射影の結果の関係では一つのタプルにされる(冗長タプルの除去)。

Y 成分の取出しのみを行い、冗長タプルの除去を行わない演算を擬似射影という。この場合、演算の結果は厳密な意味での関係ではなくなるが、本論文ではこれも関係と呼ぶことにする。擬似射影は演算の実行が容易なため、しばしば用いられる。

(3) 結合：関係 R の属性 A の値と関係 S の属性 B の値が θ の関係にあるタプルを連結した関係を作る演算である。

$$R \bowtie_{A\theta B} S = \{(r, s) | r \in R \wedge s \in S \wedge (r[A]\theta s[B])\}$$

A, B をそれぞれ R, S の結合属性という。 R と S の両方に存在する属性について、その属性値間を等号のみとした結合をとくに自然結合という。この際、演算

の結果としては、同じ属性を二重には作成しない。

データベースに対する質問は、関係代数では関係に対する関係演算の列(演算列)として表されるが、質問の答として現れる属性を結果属性という。

そのほか、3章以降で用いる語を定義しておく。

i) 原関係、中間結果の関係

関係に対して関係演算を行った結果も関係となるが、もともとデータベースにある関係と区別するため、前者を中間結果の関係、後者を原関係と呼ぶ。

ii) タプル識別番号(TID#)

関係を表とみなしたとき、表の先頭から数えた行番号を、その行が表すタプルのタプル識別番号(TID#)という。

iii) 水平蓄積法、垂直蓄積法

水平蓄積法(H-storage)とは、関係を計算機の記憶装置にタプル単位(行方向)に格納する方法であり、垂直蓄積法(V-storage)とは属性ごと(列方向)に格納する方法である^{7),8)}。したがって、垂直蓄積法は関係を完全トランスポーズ形ファイル(fully transposed file)²⁾で蓄える方法である。

3. 垂直蓄積法における関係演算の実行方法

本章では、物理構造として垂直蓄積法を用いた場合における各関係演算の実行方法、および、ページ化された仮想記憶計算機における二次記憶から主記憶へのページフェッチ数について述べる。本手法では、質問に対する関係演算列について、各演算の中間結果はすべて原関係のタプル識別番号(TID#)で管理されている。このため、中間結果はコンパクトに表現され、これに要する入力コスト(ページフェッチ数)を極力小さくしている。また、原関係の属性値を蓄えるトランスポーズ形ファイルへのアクセスは、選択演算での選択属性、結合演算での結合属性、質問結果出力時の結果属性についてのみ行う。さらに、質問に対する演算列計算の進行に伴い、原関係における演算、質問結果の対象となるタプル数が減少するため、原関係の属性へのアクセスはこれらを必要とする演算などの直前にを行い、属性を格納するページへのアクセスは必要最小限となるようにしている。

なお、本論文では演算列の最適化についてはとくにふれないが、射影、選択を演算列の最初に実行するという意味での最適化はすでに行われているものとする。

3.1 射 影

垂直蓄積法ではもともと属性ごとに格納されている

* これらの演算よりできる演算列を SPJ 式¹⁾といい、ほとんどの質問が組立て可能である。

ため、演算を擬似射影とすることにより、属性名の管理を要するだけで属性値を蓄えるトランスポーズ形ファイルにはアクセス不要である。

3.2 選 択

3.2.1 属性値と定数間の条件による選択

属性値と定数間の条件による選択 ($\sigma_{A_i \theta c} R$: A_i は関係 R の一属性, c は定数) は、演算列の最適化により原関係の属性に対して行われる。原関係 R の属性 A_i が蓄積されているトランスポーズ形ファイルのページ数を P_i とすると、この演算によるページフェッチ数 (以下、フェッチ数という) は P_i である。出力は選ばれたタプルの原関係での TID# からなる属性一つの関係である (属性が一つでも関係と呼ぶことにする)。そして、その属性名には原関係名を用いる。

3.2.2 属性値間の条件による選択

属性値間の選択を $\sigma_{A_i \theta A_j} R$ とする。 A_i, A_j が異なる原関係に存在する場合には、属性値と定数間の条件による選択の場合と異なり、演算列の最適化後も原関係での選択とすることができない。このため、演算対象となる関係が原関係でなく中間結果の場合には、属性値が原関係の TID# となっているので、あらかじめ 3.4 節で述べる方法により原関係から選択属性を抽出し、実際の属性値に変換しておかなければならない。

関係 R の選択属性 A_i, A_j の占めるページ数をそれぞれ P_{i1}, P_{i2} とすると、条件を満足するタプルを見つけるのに要するフェッチ数は $P_{i1} + P_{i2}$ となる。そして、見つけれられたタプルの (関係 R での) TID# を出力しておく。さらに、関係 R が中間結果の場合には、演算結果を原関係の TID# とするために、選択された R の TID# をもとに、今後の演算、質問結果に用いられる属性をもつ原関係について、 R の属性であるそれらの原関係の TID# を抽出しておく。その方法についても 3.4 節で述べる。

3.3 結 合

本節では、二つの関係の結合 $R_1 \bowtie_{A_{j1} \theta A_{j2}} R_2$ について述べるが、同様な考えにより三つ以上の関係の結合も可能である。

R_1 あるいは R_2 が原関係でない場合には、属性値間の条件による選択の場合と同様、結合属性の抽出をしておかなければならない。

結合の方法には、入れ子ブロック結合 (nested block join), 併合結合 (merge join) が一般的によく知られている⁶⁾ が、本論文では入れ子ブロック結合のみにつ

いて述べる。これは、本論文で提案の方法がとくに適している比較的小さい関係に対しては、併合結合より高速なためである⁶⁾。

関係 $R_1(R_2)$ の結合属性 $A_{j1}(A_{j2})$ の占めるページ数を $P_{j1}(P_{j2})$ とする。一般性を失うことなく $P_{j1} \leq P_{j2}$ とし、演算に使用する主記憶のページ数を B_j と仮定する。入れ子ブロック結合では A_{j1} を $B_j - 1$ ページ主記憶にフェッチすると A_{j2} を走査し、関係 R_1, R_2 の結合されるタプルを見つけるので、 A_{j2} は $\lceil P_{j1}/B_j - 1 \rceil$ 回主記憶に読み込まれる。したがって、フェッチ数は、

$$F_j = P_{j1} + \left\lceil \frac{P_{j1}}{B_j - 1} \right\rceil P_{j2} \quad (3.1)$$

となる。この結果、見つけれられたタプルを関係 R_1, R_2 の結合されるべきタプルの TID# で出力する。それぞれ出力 1, 2 とし、先頭から同じ位置に書かれた TID# は、 R_1, R_2 のそれぞれのタプルが結合されることを表すものとする。なお、属性値間の選択のときと同様、 R_1 あるいは R_2 が中間結果の場合には、さらに、今後の演算、質問結果に用いられる属性をもつ原関係について出力 1 あるいは 2 を R_1, R_2 の属性であるそれらの原関係の TID# に変換しておく。

3.4 属性値の抽出

3.2, 3.3 節で述べたように、演算の対象となる関係が原関係でなく中間結果のときは、演算に先立ち選択属性、結合属性の抽出を行わなければならない。また、このように中間結果を被演算関係とした場合には、計算された結果を原関係の TID# で表現するためにさらに TID# の抽出が必要である (被演算関係が中間結果の場合、その属性値は、原関係名を属性名とする原関係の TID# となっている)。

以上のほか、質問結果の出力を行うときの結果属性についても原関係から属性値の抽出が必要となる。

本節はこのような属性値の抽出について述べる。以後、抽出される属性値の属性を被抽出属性、抽出される属性値を含む関係の TID# を蓄えている属性を抽出属性ということにする。

まず、被抽出属性へのアクセスをなるべく少なくするため、属性値抽出の前にあらかじめ抽出属性を TID# の昇順にソートしておき、属性値取出し後、もう一度もとの順番に並べ換えることにする。このようにして、被抽出属性の占めるページ数を P とすると、このなかには取り出される属性値を含まないページも存在するのでフェッチ数 P' を P 以下とすることが

できる。

被抽出属性値を抽出属性が示すものの順番にもどす必要があるが、このため抽出属性とは別に抽出属性値のものの順番を表すファイル（以後、番号ファイルという。初期値はたんに1, 2, 3, …の値がはいっている）を用意しておき、抽出属性値をソートの過程で移動するごとに番号ファイルの対応する場所にある番号についても同様に移動する。TID# とその番号を（バイトあるいはワードで）同じ長さで仮定すると、このソートに要する入力ページ数は（たとえばマージソートを用いて）、

$$2P_i \lceil \log \lfloor B_s/2 \rfloor P_i \rceil \quad (3.2)$$

となる³⁾。ここで、 B_s はソートに用いる主記憶のページ数、 P_i は抽出属性の占めるページ数である。

被抽出属性値の並べ換えは、番号ファイルの番号が昇順になるようにすればよい。この際、属性値も番号の移動と同様に移動させることはいうまでもない。被抽出属性値の長さを TID# の長さの k 倍とすると、この並べ換えに要するフェッチ数は、

$$(1+k)P_i \lceil \log \lfloor B_s/(1+k) \rfloor P_i \rceil \quad (3.3)$$

となる。

4. 垂直蓄積法と水平蓄積法の比較

本章では、演算あるいは演算列について水平、垂直蓄積法の比較を行う。演算のコストとしては、前章と同じくページフェッチ数を用いる。水平蓄積法とは関係をタプルごとに蓄積する方法であるが、この場合も垂直蓄積法と同様、インデックスファイルは使用しないものとする。

4.1 擬似射影

擬似射影を行う関係の占めるページ数を $\bar{P}$ とすると、水平蓄積法では、すべてのページをフェッチする必要があるのでフェッチ数は $\bar{P}$ である。垂直蓄積法では、3章でも述べたように属性値を蓄えるファイルへのアクセスは不要である。

4.2 選 択

比較を簡単にするため、属性値と定数間の条件による選択についてのみ述べる。垂直蓄積法の場合は、3.2節で述べたように、原関係の選択属性が占めるページ数 P がフェッチ数となる。水平蓄積法では、原関係のページ数 $\bar{P}$ となるが、同一属性値は等長とし、各属性の長さの和と選択属性の長さの比を $m:1$ (m 倍) とすると、

$$\bar{P} \doteq mP$$

となるので、垂直蓄積法の方が約 $1/m$ のフェッチ数で計算可能である。

4.3 葉レベル結合

葉レベル結合とは、原関係どうしの結合である。すなわち、結合を $R_1 \bowtie_{A_{j1} \theta A_{j2}} R_2$ と表すとき、 R_1, R_2 が原関係の場合である。 R_1, R_2 が大きい場合には非現実的な演算となるが、比較的小さい場合には、実行可能な演算である。

R_1, R_2 の占めるページ数をそれぞれ $\bar{P}_{j1}, \bar{P}_{j2}$ ($\bar{P}_{j1} \leq \bar{P}_{j2}$) とし、演算に用いる主記憶のページ数を B_j とする。入れ子ブロック結合では、 R_1 について $(B_j - 1)$ 個のページを読み込むごとに R_2 全体の走査を行うため、水平蓄積法の場合フェッチ数は、

$$\bar{F}_j = \bar{P}_{j1} + \lceil \bar{P}_{j1}/(B_j - 1) \rceil \bar{P}_{j2} \quad (4.1)$$

となる。属性 A の属性値の長さ（同一属性の属性値は等長とする）を $\text{len}(A)$ で表し、属性 A_{j1}, A_{j2} の占めるページ数をそれぞれ P_{j1}, P_{j2} とするとき、

$$\left. \begin{aligned} a &= \frac{\sum_{A \in R_1} \text{len}(A)}{\text{len}(A_{j1})} \\ b &= \frac{\sum_{A \in R_2} \text{len}(A)}{\text{len}(A_{j2})} \end{aligned} \right\} \quad (4.2)$$

ただし、 $A \in R$ は関係 R に含まれる属性を表す。

を用いると、

$$\bar{P}_{j1} \doteq aP_{j1}, \bar{P}_{j2} \doteq bP_{j2} \quad (4.3)$$

であるから式(4.1)は、

$$\bar{F}_j \doteq aP_{j1} + \lceil aP_{j1}/(B_j - 1) \rceil bP_{j2} \quad (4.4)$$

と近似できる。

垂直蓄積法の場合は、3.3節で述べたようにフェッチ数は式(3.1)となる。

例 4.1 $a=b=5, P_{j1}=P_{j2}=10, B_j=6$

のとき、

$$F_j = 30$$

$$\bar{F}_j = 550$$

となる。

4.4 基本演算列

各種関係演算について調べたが、4.3節までの比較では次の点が不十分である。

- i) 垂直蓄積法の場合のみ必要となる属性値の抽出に要するコストが含まれていない。
- ii) 個々の関係演算についての比較であり、質問は演算列となることを考えると、コストの比較は演算列についてのほうが現実的である。

しかし、任意の演算列について調べることは不可能なため、本論文では次式で示される原関係 R_1, R_2 に対する演算列について比較を行う。

$$\pi_{A_1^1 \dots A_{a-1}^1 A_1^2 \dots A_{b-1}^2} [\{ \pi_{A_1^1 A_2^1 \dots A_a^1} (\sigma_{A_a^1 = \alpha R_1}) \} \bowtie \{ \pi_{A_1^2 A_2^2 \dots A_b^2} (\sigma_{A_b^2 = \beta R_2}) \}] \quad (4.5)$$

ただし、 $A_1^1, A_2^1, \dots, A_a^1$ は R_1 の属性、 $A_1^2, A_2^2, \dots, A_b^2$ は R_2 の属性であり、 α, β は定数である。式(4.5)が垂直蓄積法の場合、結合属性、結果属性の抽出を必要とすることはいうまでもない。そして、この式で表される演算列は基本となる関係演算である射影、選択、結合を含むほぼ最小の質問と考えるとよい。この意味において、本論文ではこのタイプの演算列を基本演算列ということにする。以下、基本演算列の計算に要するフェッチ数について述べるが、本節ではコスト解析を容易にするため、属性値はすべて等長で TID# の長さと同じとする。たとえば、属性値が符号化されている場合を考えるとよい。

4.4.1 垂直蓄積法の場合

垂直蓄積法では、射影に関するフェッチは不要であるから、まず、 R_1 と R_2 に関する選択について考える。 $R_1(R_2)$ の一つの属性が占めるページ数を $P_1(P_2)$ とすると、これに要するフェッチ数は $P_1(P_2)$ である。結果 (TID#) の占めるページ数を $P_{j1}(P_{j2})$ とする。

次は結合演算であるが、その前に結合属性 A_a^1, A_b^2 の抽出を行わなければならない。これに要するフェッチ数は、 A_a^1 のほうについて、抽出前後のソートに要するフェッチ数がそれぞれ $2P_{j1} \lceil \log \lfloor B_s/2 \rfloor P_{j1} \rceil$ 、属性値の取出しに要するフェッチ数は抽出される属性値を含んでいるページ数であり $P_1(\leq P_1)$ とする。 A_b^2 のほうについても同様である。

結合に要するフェッチ数は式(3.1)に示すとおりである。結合テーブル番号を格納する出力 1, 2 (3.3 節参照) のページ数を P_i (出力 1, 2 と同じ) とすると、結合結果を原関係の TID# で表すためのフェッチ数は出力 1(2) に対し、 $4P_i \lceil \log \lfloor B_s/2 \rfloor P_i \rceil + P'_{j1}(4P_i \lceil \log \lfloor B_s/2 \rfloor P_i \rceil + P'_{j2})$ となる。ここで、 $P'_{j1}(P'_{j2})$ は $P_{j1}(P_{j2})$ ページのうち、結合される TID# を含んでいるページ数である。

最後に、演算列の結果を出力するために原関係から結果属性を抽出する。関係 R_1 の属性 $A_1^1, A_2^1, \dots, A_{a-1}^1$ の抽出について、抽出前のソートに要するフェッチ数は $2P_i \lceil \log \lfloor B_s/2 \rfloor P_i \rceil$ である。抽出は $a-1$ 個の属性について行うためフェッチ数は $(a-1)P'_1$ となる。ここで P'_1 は、関係 R_1 の一つの属性を蓄えるページ数

P_1 のうち取出し属性値を含むページ数である。抽出後のソートは、 $2(a-1)P_i \lceil \log \lfloor B_s/2 \rfloor P_i \rceil$ フェッチとなる。関係 R_2 からの抽出も同様である。

以上まとめると全ページフェッチ数は、

$$\begin{aligned} F_{BOS} = & P_1 + P_2 + 4P_{j1} \lceil \log \lfloor B_s/2 \rfloor P_{j1} \rceil \\ & + 4P_{j2} \lceil \log \lfloor B_s/2 \rfloor P_{j2} \rceil \\ & + P'_1 + P'_2 + P_{j1} + \lceil P_{j1}/(B_j-1) \rceil P_{j2} \\ & + 8P_i \lceil \log \lfloor B_s/2 \rfloor P_i \rceil + P'_{j1} + P'_{j2} \\ & + 2(a+b)P_i \lceil \log \lfloor B_s/2 \rfloor P_i \rceil \\ & + (a-1)P'_1 + (b-1)P'_2 \end{aligned} \quad (4.6)$$

となる。

4.4.2 水平蓄積法の場合

まず、基本演算列を式(4.5)のとおりに一演算ずつ行う場合について考える。このとき、関係 $R_1(R_2)$ の占めるページ数を $\bar{P}_1(\bar{P}_2)$ とすると、選択に要するフェッチ数は $\bar{P}_1(\bar{P}_2)$ である。選択の結果の占める大きさを $\bar{P}_{j1}(\bar{P}_{j2})$ ページとすると、続く射影(擬似射影とする)に要するフェッチ数は $\bar{P}_{j1}(\bar{P}_{j2})$ となる。この結果 $\bar{P}_{j1}(\bar{P}_{j2})$ ページとなるとする。次に結合に要するフェッチ数は、入れ子ブロック結合を用いるとし、 $\bar{P}_{j1} + \lceil \bar{P}_{j1}/(B_j-1) \rceil \bar{P}_{j2}$ ($\bar{P}_{j1} \leq \bar{P}_{j2}$) となる。さらに、この結果を $\bar{P}_i$ ページとする。最後の(擬似)射影は $\bar{P}_i$ フェッチで計算できる。以上フェッチ数を合計すると、

$$\bar{P}_1 + \bar{P}_2 + \bar{P}_{j1} + \bar{P}_{j2} + \bar{P}_{j1} + \lceil \bar{P}_{j1}/(B_j-1) \rceil \bar{P}_{j2} + \bar{P}_i \quad (4.7)$$

となる。

しかし、演算を組み合わせることで実行することにより、フェッチ数を減少させることができる。それは3回ある射影をその一つ前の演算(選択とか結合)と同時に行うことにより実現される。たとえば選択後の射影は、選択により取り出されたタプルから射影が必要とされている属性値を結果として出力することにより、選択の中間結果を射影の際読み込む $\bar{P}_{j1} + \bar{P}_{j2}$ フェッチが不要となる。結合後の射影も同様であり、 $\bar{P}_i$ フェッチを減少できる。したがって以上の工夫を行うことにより、

$$\bar{F}_{BOS} = \bar{P}_1 + \bar{P}_2 + \bar{P}_{j1} + \lceil \bar{P}_{j1}/(B_j-1) \rceil \bar{P}_{j2} \quad (4.8)$$

となる。

4.4.3 基本演算列での比較

基本演算列について垂直、水平蓄積法の比較を行う。

まず、式(4.6)を、

$$\left. \begin{aligned} F_{BOS} &= F_1 + F_2 + F_3 \\ F_1 &= P_1 + P'_1 + (a-1)P'_1 + P_2 + P'_2 + (b-1)P'_2 \\ F_2 &= P_{j1} + \lceil P_{j1}/(B_j-1) \rceil P_{j2} + P'_{j1} + P'_{j2} \end{aligned} \right\}$$

$$\left. \begin{aligned} F_3 &= 4P_{j1} \lceil \log \lfloor B_s/2 \rfloor P_{j1} \rceil + 4P_{j2} \lceil \log \lfloor B_s/2 \rfloor P_{j2} \rceil \\ &\quad + 8P_i \lceil \log \lfloor B_s/2 \rfloor P_i \rceil \\ &\quad + 2(a+b)P_i \lceil \log \lfloor B_s/2 \rfloor P_i \rceil \end{aligned} \right\} \quad (4.9)$$

とおく. また, 式(4.8)において, R_1, R_2 の属性数 m_1, m_2 等を用いて,

$$\bar{P}_1 \doteq m_1 P_1, \bar{P}_2 \doteq m_2 P_2, \bar{P}_{j1} \doteq a P_{j1}, \bar{P}_{j2} \doteq b P_{j2}$$

と近似すると,

$$\left. \begin{aligned} \bar{F}_{BOS} &\doteq \bar{F}_1 + \bar{F}_2 \\ \bar{F}_1 &= m_1 P_1 + m_2 P_2 \\ \bar{F}_2 &= a P_{j1} + \lceil a P_{j1} / (B_j - 1) \rceil b P_{j2} \end{aligned} \right\} \quad (4.10)$$

となる. ここで,

$$\left. \begin{aligned} m_1 &\geq a+1, m_2 \geq b+1, a \geq 2, b \geq 2 \\ P_1 &\geq P'_1 \geq P''_1, P_2 \geq P'_2 \geq P''_2 \end{aligned} \right\} \quad (4.11)$$

が成立するので, 式(4.9), (4.10)から,

$$\left. \begin{aligned} F_1 &\leq \bar{F}_1 \\ F_2 &\leq \bar{F}_2 \end{aligned} \right\} \quad (4.12)$$

である. F_3 は原関係からの属性値取出しに関するソートによるものであり, 水平蓄積法には相当するものはない. しかし, 一般的にオンラインとかタイムシェアリングシステムでの検索のように質問結果の出力量が少ない場合には, F_3 は $\bar{F}_1$ と F_1 あるいは $\bar{F}_2$ と F_2 の差に比べて小さいため, $F_{BOS} < \bar{F}_{BOS}$ となる.

例 4.2 $m_1 = m_2 = 20, a = b = 5, P_1 = P_2 = P'_1 = P'_2 = 100, P''_1 = P''_2 = 50, P_{j1} = P_{j2} = P'_{j1} = P'_{j2} = 9, P_i = 1, B_j = 5, B_s = 6$ のとき,

$$F_1 = 800, F_2 = 54, F_3 = 172.$$

したがって, $F_{BOS} = 1026$.

$$\bar{F}_1 = 4000, \bar{F}_2 = 585.$$

したがって, $\bar{F}_{BOS} = 4585$ となる. $\square$

例 4.3 $m_1 = m_2 = 10, a = b = 8, P_1 = P_2 = P'_1 = P'_2 = 40, P''_1 = P''_2 = 20, P_{j1} = P_{j2} = P'_{j1} = P'_{j2} = 20, P_i = 2, B_j = 5, B_s = 10$ のとき,

$$F_1 = 440, F_2 = 160, F_3 = 400.$$

したがって, $F_{BOS} = 1000$.

$$\bar{F}_1 = 800, \bar{F}_2 = 6560,$$

したがって, $\bar{F}_{BOS} = 7360$

となる. $\square$

以上の解析より, 次の結論を得る. 従来, 関係データベースの物理構造としては, 水平蓄積法にインデックスファイルを付加したものが使用されてきたが, インデックスファイルはすべての属性に対して付けられてはいないため, それらがなく属性に対する関係演算は効率的でない. これに対し, 垂直蓄積法は, インデックスファイルを使用しなくても関係演算を高速に実

行可能である. とくに, 今日, ますます必要とされている小さなデータベースに対しては十分である. また, インデックスファイルを使用しないので, 効率的な検索のためインデックスの有無を意識する必要がなく, より操作性のよいデータベースになると考える.

5. ページプリフェッチ

主記憶への読み込み開始ページと読み込みページ数を指定することにより, 1回のページフェッチで連続する複数ページを読み込む方法をページプリフェッチという. 一般に, データベースのように大容量で, 記憶の局所参照性が乏しい場合には, ページプリフェッチが有効であるといわれている⁹⁾ が, 本論文で対象としているインデックスファイル等のアクセス手段をもたない比較的単純な物理構造をもつデータベースにはとくに適している. 本章では, 垂直蓄積法の場合を例に, ページプリフェッチによりいかに演算が高速化されるかについて述べる.

前章まではコストの見積りとして, ページフェッチ数を用いてきたが, 本章ではより正確にページプリフェッチのコストを求めるため, 連続する u ページを1回でフェッチするページプリフェッチの時間コストを次のように定義する.

$$\left. \begin{aligned} T(u) &= t_a + ut_i \\ t_a &: \text{アクセス時間コスト} \\ t_i &: \text{1ページ当りの転送時間コスト} \end{aligned} \right\} \quad (5.1)$$

$u=1$ のときは, ページプリフェッチを行わない場合に帰着する.

5.1 連続するページの単純フェッチ

属性値と定数間の条件による選択の場合のように, ある属性を格納するページを連続して読み込む場合には, 演算に使用する主記憶領域のページ数だけ一度にフェッチ (プリフェッチ) することにより時間コストを減少できる. すなわち, 連続する P ページに対して主記憶 B ページ ($P \geq B$) を用いて計算するコストは, B ページずつプリフェッチを行うことにより,

$$C_s^P = \lceil P/B \rceil T(B) \quad (5.2)$$

となる. ページプリフェッチを行わない場合よりコストを減少できるが, これは B ページまとめてフェッチすることにより, アクセスコストが1ページ当たり $1/B$ となるためである.

例 5.1 $P=200, B=10, t_a/t_i=10$ とすると,

i) プリフェッチを行わないとき,

$$C_s = 200 T(1) = 2200 t_i$$

ii) プリフェッチを行うとき,

$$C_3^P = \lceil 200/10 \rceil T(10) = 400 t_i$$

となる。

□

同様な考えは、属性値間の条件による選択の場合にも適用することができる。たとえば、二つの属性値間の場合には $B/2$ ページずつプリフェッチする等である。

属性値の抽出においては、属性値の取出しのため属性の一部をフェッチすればよいが、ページプリフェッチにより属性全体をフェッチしたほうが高速となる場合もある。被抽出属性の占めるページ数を P 、そのうちフェッチが必要なページ数を P' とするとき、

$$\lceil P/B \rceil T(B) < P' T(1) \quad (5.3)$$

ならば、ページプリフェッチにより属性全体をフェッチするほうが高速である。

5.2 入れ子ブロック結合

ページプリフェッチを行わないときは、3.3節で述べたように、演算に使用する主記憶のページ数を B とするとき、これを $B-1$ ページと 1 ページに分けてそれぞれ P_{j1} , P_{j2} ページの読み込みにあてたが、ページプリフェッチが可能な場合にはこの分け方は最適ではない。 P_{j1} , P_{j2} ページの読み込みに対して、それぞれ $b_1, b_2 (b_1 + b_2 = B)$ ページを割り当てページプリフェッチを行うと時間コストは、

$$C_j^P = \lceil P_{j1}/b_1 \rceil T(b_1) + \lceil P_{j1}/b_1 \rceil \lceil P_{j2}/b_2 \rceil T(b_2) \quad (5.4)$$

となる。このうち第 2 項が主要項となるため、第 2 項を、

$$f(b_2) = \frac{P_{j1}P_{j2}}{b_1b_2} (t_s + b_2t_i) \quad (5.5)$$

と近似し、これを最小とする b_2 を求めると、

$$\left. \begin{aligned} b_2 &= \sqrt{R^2 + RB} - R \\ \text{ただし, } R &= t_s/t_i \end{aligned} \right\} \quad (5.6)$$

となる。 b_2 は自然数でなければならないので、実際には式(5.6)に近い自然数を用いることになる。なお、 b_1 は $B - b_2$ より求める。

例 5.2 いくつかの例について式(5.6)の計算値を示す。

- i) $B \ll R$ のとき, $b_2 \doteq B/2$
- ii) $B=5$, $R=10$ のとき $b_2 \doteq 2.2$
- iii) $B=10$, $R=10$ のとき $b_2 \doteq 4.1$
- iv) $B=10$, $R=4$ のとき $b_2 \doteq 3.5$

となる。

□

例 5.3 $P_{j1}=P_{j2}=50$, $B=10$, $R=10$ とすると、

i) プリフェッチを行わないとき、

$$C_j^P = 3850 t_i$$

ii) $b_1=6$, $b_2=4$ でプリフェッチを行うとき (例 5.2 ii) 参照),

$$C_j^P = 1782 t_i$$

iii) $b_1=b_2=5$ でプリフェッチを行うとき、

$$C_j^P = 1650 t_i$$

となる。例 5.2 によると ii) のように $b_2=4$, $b_1=B-b_2=6$ を用いたほうがよいように思われるが、式(5.6)には近似が含まれているため、実際には iii) の $b_1=b_2=5$ ほうがいくぶんコストが小さい。しかし、この程度の差であれば ii) を用いても大差はない。

□

5.3 ソート

垂直蓄積法では属性値抽出の前後にソートを要するが、本節ではページプリフェッチを用いたソートについて述べる。

主記憶のページ数 B で P ページを占める等長データをソートするのに要するページフェッチ数は、ページプリフェッチを行わない場合、

$$F_M = P \lceil \log_B P \rceil \quad (5.7)$$

である。ページプリフェッチが可能で一度に $b (\leq B/2)$ ページずつフェッチしてソート (たとえばマージソート) する場合には、

$$C_M^P = \lceil P/b \rceil \lceil \log \lfloor B/b \rfloor \lceil P/b \rceil T(b) \quad (5.8)$$

の時間コストで可能である。この式を、

$$C_M^{P'} = (P/b) \{ \log (B/b) (P/b) \} T(b) \quad (5.9)$$

と近似し、 $C_M^{P'}$ を最小とする $x=B/b$ を求めると、付録に示すように、

$$\left. \begin{aligned} Sx(\ln x)(\ln kx) - (Sx+1)\ln k &= 0 \\ \text{ただし, } S &= t_s/Bt_i, k = P/B \end{aligned} \right\} \quad (5.10)$$

を満足する x となる。

この結果は、ソートの問題としては興味深いが、本論文でとくに興味の対象としている比較的小さなファイルに対しては、ページプリフェッチによる大きな効果は期待できない。

例 5.4 $P=20$, $b=6$, $t_s/t_i=10$ とすると、

i) $b=1$, すなわちページプリフェッチを行わないとき、

$$C_M^P = 440 t_i$$

ii) $b=3$ でページプリフェッチを行うとき、

$$C_M^P = 273 t_i$$

となる。

□

6. 計算機実験

本章では、計算機 NEC ACOS-450 により各種関係演算を行い I/O 時間を測定した結果を示す。本計算機はセグメンテーションによる仮想記憶計算機であるためページフェッチの概念はない。このため、実験ではページフェッチをデータファイルのブロック転送に置き換えている。また記載された測定値は、

$$\text{I/O 時間} = \text{経過時間} - \text{CPU 時間} \quad (6.1)$$

とした。これは使用計算機の CPU のスピードがあまり速くないこと、入れ子ブロック結合においてブロッ

ク (ページ) 内でソートを行う⁶⁾ 工夫がなされていないこと等の理由により、経過時間に占める CPU 時間が無視できないためである。

6.1 垂直・水平蓄積法の比較

3, 4 章で論じた垂直・水平蓄積法について、計算機実験の結果を表 1 (a) から (c) に示す。属性値の長さはすべて 4 byte, 1 ブロックを 1 kbyte としたため、1 ブロックは 256 個の属性値を格納することができる。表中、 m は属性数、 P は一つの属性をたぐわえるのに必要なブロック数である。理論値として、I/O 回数を用いている*が、これに 20 msec を掛けると測定値に比較的近い値となる。以下、その他の測定条件を演算ごとに述べておく。

A. 選択

属性と定数間の条件による選択 ($\sigma_{A\theta cR}$) である。選択度は 0.1 とした。

B. 葉レベル結合

原関係 R_1 と R_2 の結合 ($R_1 \bowtie_{A_i \theta A_j} R_2$) を主記憶のブロック数 $B_j=5$ について行った。属性数 (m)、一つの属性を蓄えるブロック数 (P) は、 R_1, R_2 について同じとした。また、結合されるタプル数は 10 個としている。

C. 基本演算列

4.4 節の記号を用いて表すと、 $m_1=m_2=20$, $a=b=10$, $P_1=P_2=50$, $B_1=5$, $B_2=6$ の場合について行った。2 回の選択における選択度 S は同じとし、表 1 (c) に示した値について実験した。なお、水平蓄積法では、射影を一つ前の演算といっしょに行う工夫はなされていない。

表 1 より測定値は、すべて理論値から予想されるとおりの結果となっており、全般的に、垂直蓄積法のほうがかなり高速である。

6.2 ブロックサイズの影響

5 章で述べたようなフェッチ開始ページとページ数を指定して複数ページをフェッチする方法を動的プリフェッチという⁸⁾ が、この方法は、本計算機のオペレーティングシステムのもとでは困難であった。このため、一度にフェッチするページ数を一定とした静的プリフェッチについて行った。実験では、ブロックサイズ (BZ) を単位ブロックサイズ (512 byte) の整数倍 (1, 2, 4, 8 倍) とし、単位ブロックを一度に数ブ

表 1 垂直・水平蓄積法の比較

Table 1 Comparison of V- and H-storage.

(a) 選択

m	P	垂直蓄積法		水平蓄積法	
		測定値 (ms)	理論値 (回)	測定値 (ms)	理論値 (回)
3	30	606	33	1,928	99
	100	2,163	110	6,318	330
	200	4,222	220	12,604	660
10	30	594	33	6,554	330
	100	2,164	110	21,800	1,100
	200	4,268	220	43,523	2,200
20	30	595	33	13,256	660
	100	2,223	110	44,123	2,200

(b) 葉レベル結合

m	P	垂直蓄積法		水平蓄積法	
		測定値 (ms)	理論値 (回)	測定値 (ms)	理論値 (回)
3	3	181	8	934	37
	6	447	20	2,332	109
	10	965	42	5,670	271
10	3	155	8	5,077	271
	6	394	20	19,248	951
	10	1,063	42	51,412	2,601
20	3	164	8	19,833	961
	6	379	20	76,823	3,721

(c) 基本演算列

S	垂直蓄積法		水平蓄積法	
	測定値 (ms)	理論値 (回)	測定値 (ms)	理論値 (回)
0.02	7,058	256	40,602	2,144
0.06	8,368	300	49,921	2,572
0.12	11,814	468	71,327	3,564
0.2	18,253	794	113,892	5,604
0.3	29,338	1,089	188,960	9,354

* 3, 4 章では解析を容易にするため、主記憶への入力回数だけについてコストを計算したが、本章では出力回数も含めて理論値としている。

表 2 ブロックサイズの影響

Table 2 Effect of enlarging block size.

(a) 選 択

タプル数	BZ=512		BZ=1,024		BZ=2,048		BZ=4,096	
	測定値 (ms)	理論値 (回)	測定値 (ms)	理論値 (回)	測定値 (ms)	理論値 (回)	測定値 (ms)	理論値 (回)
4,096	587	36T (1)	337	18T (2)	203	9T (4)	127	5T (8)
8,192	1,238	71T (1)	697	36T (2)	398	18T (4)	216	9T (8)
16,384	2,613	141T (1)	1,475	71T (2)	787	36T (4)	442	18T (8)
32,768	5,003	282T (1)	2,836	141T (2)	1,483	71T (4)	806	36T (8)
65,536	10,165	564T (1)	5,659	282T (2)	2,940	141T (4)	1,603	71T (8)

(b) 葉レベル結合

タプル数	BZ=512		BZ=1,024		BZ=2,048		BZ=4,096	
	測定値 (ms)	理論値 (回)	測定値 (ms)	理論値 (回)	測定値 (ms)	理論値 (回)	測定値 (ms)	理論値 (回)
1,024	367	18T (1)	190	10T (2)	177	6T (4)	184	4T (8)
2,048	985	50T (1)	716	26T (2)	388	14T (4)	318	8T (8)
4,096	2,987	130T (1)	1,766	66T (2)	1,245	34T (4)	1,067	22T (8)

ロック転送するのと同様な効果を得ている。属性値の長さは 4 byte とし、表 2 に示すタプル数について実験を行った。理論値としては、I/O 回数×ブロック入出力時間 ($T(BZ/128)$) を用いている。

A. 選 択

属性と定数間の条件による選択であり、選択度を 0.1 とした。

B. 葉レベル結合

演算に使用する主記憶のサイズをすべて単位ブロックサイズの 16 倍とした。したがって、 $B_i=512 \times 16/BZ$ で入れ子ブロック結合を行ったことになる。また、二つの原関係のタプル数は同じとし、結合されるタプル数は 10 個としている。

測定結果は表 2 (a), (b) に示したとおりである。葉レベル結合の $BZ=2,408, 4,096$ byte では測定値が理論的に予想される値より大きくなっているが、これは他の実験と比較し、経過時間に占める CPU 時間が大きくなるため、I/O 時間があまり正確に求まっていないためと思われる。しかし、全体的にブロックサイズの影響は顕著である。

7. む す び

基本的な関係演算について、関係を物理的に垂直あるいは水平に蓄積した場合の比較を行った。その結果、オンライン質問のような出力量の少ない質問に対しては、計算方法の工夫により、垂直蓄積法のほうが水平蓄積法に比べ、数倍以上高速に計算できる可能性

があることが明らかにされた。また、垂直蓄積法のように比較的単純な物理構造をもつ場合には、ページプリフェッチによりさらに高速化が期待できることがわかった。そして、ページプリフェッチはデータベースにとどまらず、大量データのソートに対しても有効であることを述べた。

一般に、汎用オペレーティングシステムのもとでは、本格的なページプリフェッチは困難であるが、今後、データベース専用オペレーティングシステムとすることにより、ページプリフェッチの利点を十分に生かすべきであると考えられる。

参 考 文 献

- 1) Aho, A. V., Sagiv, Y. and Ullman, J. D.: Efficient Optimization of a Class of Relational Expressions, *ACM Trans. Database Syst.*, Vol. 4, No. 4, pp. 435-454 (1979).
- 2) Batory, D. S.: On Searching Transposed Files, *ACM Trans. Database Syst.*, Vol. 4, No. 4, pp. 531-544 (1979).
- 3) Blasgen, M. W. and Eswaran, K. P.: Storage and Access in Relational Data Bases, *IBM Syst. J.*, Vol. 16, No. 4, pp. 363-377 (1977).
- 4) Chamberlin, D. D. et al.: A History and Evaluation of System R, *Comm. ACM*, Vol. 24, No. 10, pp. 632-646 (1981).
- 5) 林, 鴻池: リレーショナル・データベース管理システム AIM/RDB の実現手法, *日経エレクトロニクス*, No. 310, pp. 171-202 (1983).
- 6) Kim, W.: A New Way to Compute the

- Product and Join of Relations, *Proc. ACM SIGMOD*, pp. 179-187 (1980).
- 7) Tsuda, T. and Sato, T.: Transposition of Large Tabular Data Structures with Applications to Physical Database Organization (Part 1) Transposition of Tabular Data Structures, *Acta Inf.*, Vol. 19, No. 1, pp. 13-33 (1983).
- 8) Tsuda, T., Urano, A. and Sato, T.: Transposition of Large Tabular Data Structures with Applications to Physical Database Organization (Part 2) Applications to Physical Database Organization, *Acta Inf.*, Vol. 19, No. 2, pp. 167-182 (1983).
- 9) Smith, A. J.: Sequentiality and Prefetching in Database Systems, *ACM Trans. Database Syst.*, Vol. 3, No. 3, pp. 223-247 (1978).
- 10) Ullman, J. D.: *Principles of Database Systems*, Computer Science Press, Rockville, Maryland (1980).

付 録

$$C_M^{P'} = (P/b) \{ \log (B/b) (P/b) \} T(b) \quad (A. 1)$$

を最小とする $x = B/b$ について述べる. 式 (A. 1) より,

$$\left. \begin{aligned} C_M^{P'} &= P t_1 g(x) \\ g(x) &= (Sx+1) \log_x kx \\ \text{ただし, } S &= t_1/Bt_1, k = P/B \end{aligned} \right\} \quad (A. 2)$$

とおく. $g(x)$ を最小とする x は $C_M^{P'}$ をも最小とするが,

$$\partial g(x) / \partial x = 0$$

より,

$$Sx(\ln x)(\ln kx) - (Sx+1)\ln k = 0 \quad (A. 3)$$

を満足する x のとき $g(x)$ は最小となる.

実際のソートでは, 式 (5.8) からわかるように x は, $2 \leq x \leq B$ の整数値である. したがって, たとえば $S \ll 1(t_1 \ll Bt_1)$ のとき $x = B(b=1)$, $S \gg 1(t_1 \gg Bt_1)$ のとき $x = 2(b = \lfloor B/2 \rfloor)$ とすればよいことになる.

(昭和 58 年 6 月 7 日受付)

(昭和 58 年 7 月 19 日採録)