結合演算を高速に実行する関係データベースマシン

7X-4

大月知之　　濱田　喬
東京大学大学院工学系研究科　学術情報センター

1　はじめに

関係データベースは従来事務処理に用いられることが多かったが、昨今人工知能、ＣＡＤ／ＣＡＭ等応用分野も広がり、その高速処理への要求は高まる一方である。その演算としては選択、結合、射影等があるが、これらのうちの結合は処理負荷が重く、また、最近の応用分野において繰り返し使われることが多いため、その処理コストの削減が特に重要な課題となっている。現在、このような結合演算を高速に行うデータベースマシンについて研究を進めているが、本稿ではこのマシンに関し、その基本思想、機能、構成、演算実行の流れ、マシンの持つ特徴について述べる。

2　基本思想

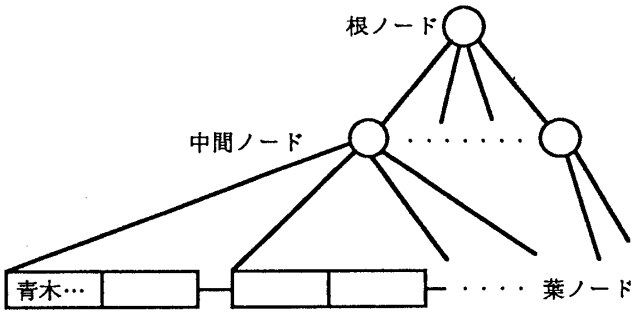
関係データベースマシンにおいて演算の実行にかかるコストには大きく分けて、２次記憶のアクセスコスト、データの転送コスト、処理コストの３つがあり、それぞれのコストを抑えるために、インデックスの利用、高性能ネットワークの採用、並列処理の利用等の方法がとられている。これらのうちインデックスは、データベースマシンに採用されているものの殆どが主に選択演算の実行に威力を発揮するものであり、結合演算等の実行時には２次記憶へのアクセスが無駄に行われていることが多い。しかし、現在提案されているインデックスの中には結合の実行にも有効なものが存在する（[1]、[2]）。本稿で提案するマシンは、このようなインデックスの１つ（Bc-tree[2]）と並列処理（ハッシュとソートを用いる手法[3]）の利用により結合の高速化を実現するものである。以下、このインデックスと並列処理について簡単に説明を行う。

まず採用するインデックスのBc-tree について。これは、インデックスを関係ごとにではなくドメインごとに作るものであり、その内容は図１－ａの表のようなものである。但し、タプル位置の情報とは具体的には属性値と対応する複数のタプル識別子（ＴＩＤ）を記したページ（ＴＩＤページ）へのポインタである。実際のBc-tree はこの表に属性値に基づくナビゲーション用の中間ノードを付した木構造をとる（図１－ｂ）。インデックスとデータの構成は全体として図２のようなものである。

次に、このインデックスを用いた演算の実行例を紹介する。

属性値	関係R		関係S	
	タプル数	タプル位置の情報	タプル数	タプル位置の情報
青木	3	Page A	2	Page B
青山	0	null	5	Page C

(a) 内容



(b) 構造

図１　Bc-tree

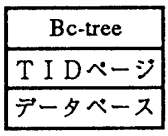


図２　インデックスとデータの構成

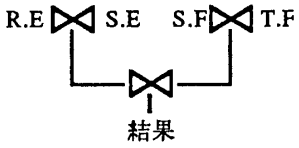


図３　複雑な処理の実行例

＜例１＞選択演算　R.A = const

この場合、まず属性Ａの属するドメインのBc-tree、次にＴＩＤページを参照することにより必要なＴＩＤを獲得し、ＴＩＤに記された格納ページをもとに最終的なタプルを読み出す。

＜例２＞結合演算　R.E = S.E

この場合、まず属性Ｅの属するドメインのBc-tree を参照し、結合可能な属性値を求める。次にこの属性値に対応するそれぞれの関係のＴＩＤページを参照することにより結合すべきタプルを決定し、最後に必要タプルを読み出し、結合を実行する。

上記の２例から分かるように、このインデックスを用いると、選択、結合ともに必要なタプルのみを取り出すので２次記憶への無駄なアクセスが避けられる。

次に、より複雑な問い合わせの実行における並列処理の利用を例により説明する。

A Relational Database Machine for High-Speed Join Operation
Takashi HAMADA (National Center for Science Information System)
Tomoyuki OHTSUKI (University of Tokyo)

＜例３＞R.E=S.E ∧ S.F=T.F

この場合、まずＲとＳ、ＳとＥの結合に必要なタプルをBc-tree、ＴＩＤページを用いて調べる。次に、両方の結合に関し得られたＳのタプルをそのＴＩＤに関し結合する。これにより、最終的に必要なＳのタプルを決定し、Ｓのタプルをもとに R、Ｔに関しても必要なタプルの決定を行い、これらタプルを読み出して結合の実行を完了する（図３）。

例３においてＳのＴＩＤに基づく結合にはインデックスは使用できず、他の何らかの手法で結合を行う必要があるが、このようなＴＩＤに基づく結合は処理の複雑さに比例して多くなるものである。そこで、本研究ではこれらの結合をハッシュとソートを用いた並列処理により実現することとした。以下、この結合手法につき簡単に説明する。

この手法では、まず結合を行う両処理対象をその結合属性値にハッシュ関数を施した結果に従ってクラスタリングする。この際、等しい属性値を持つタプルは同じクラスタに割り振られるため、各クラスタは独立なものとなる。そこで、これらのクラスタを複数のプロセッサを用いてソート、マージし、結合を実行する。このような手法を用いることにより結合の処理コストを削減することができる。

以上に記したように、Bc-tree を用いて２次記憶アクセスコストを削減し、ハッシュとソートを用いた並列処理により処理コストを削減して高速な処理を実現することが本研究における基本思想である。

3　ホストマシンとデータベースマシンの役割分担

現在研究中のマシンはホストマシンに対するバックエンドマシンという位置づけを持つが、それぞれが以下のような機能を持つことを前提としている。

＜ホストマシン＞
- ユーザの権利チェック
- 問い合わせ解析
- 概念スキーマから外部スキーマへの変換
- 障害対策

＜データベースマシン＞
- 各演算の実行
- 内部スキーマ、概念スキーマの取り扱い
- 一貫性制約の実現
- 障害対策

4　マシンの構成

マシンの構成は図４に示すようなものである。但し各略号の意味は以下のとうりである。

- ＣＭ：制御モジュール。他の各モジュールを制御し、ホストマシンからの命令を実行する。
- ＩＭ：インデックス・モジュール。Bc-tree を格納する２次記憶を持ち、インデックス処理の一部分を実行する。
- ＤＭ：ディスク・モジュール。２次記憶を持ち、データベース、ＴＩＤページ等を格納する。
- ＭＭ：メモリ・モジュール。データのステージングを行う。
- ＰＭ：プロセッサ・モジュール。汎用プロセッサとハードウェア・ソータを持ち、各処理を実行する。

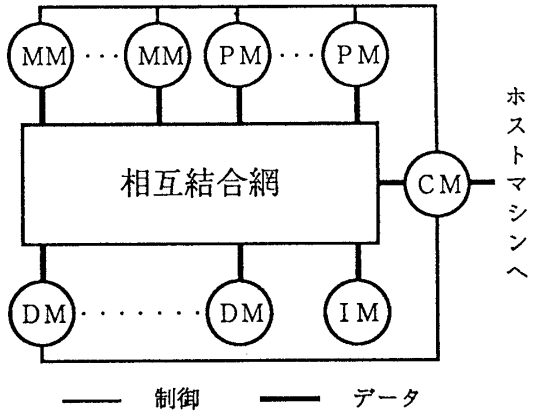


図４　マシンの構成

5　演算実行の流れ

問い合わせは以下のようにして実行される。まずＩＭがBc-tree の必要葉ノードをＭＭを介してＰＭに転送する。ＰＭはこれをもとに必要なＴＩＤページを決定し、これをＤＭに要求する。ＤＭがこれに応じＴＩＤページをＭＭを介してＰＭに転送する。これをもとにＰＭが各素演算（インデックスを用いて完了できる演算）において必要なタプルを決定する。次に、ＭＭを介しながらＰＭとＰＭでデータのやりとりをし、タプル識別子の結合を行って、最終的に必要なタプルを決定する。これらタプルをＰＭがＤＭに要求し、ＤＭからＭＭを介してＰＭにデータを転送することで最終的な処理を完了する。

6　マシンの持つ特徴

本研究で構築するマシンは以下のような特徴を持っている。
- 大規模のデータベースに対応できる。
- 問い合わせに関わる関係のタプルの総和に比べ、最終結果に必要なタプル数が少ない分野で威力を発揮する。
- 選択的結合やマルチウェイ結合を高速に処理する。
- タプルにおける属性値のサイズ（バイト数）の大きい分野で威力を発揮する。

7　おわりに

現在研究中のデータベースマシンに関し、その基本思想、構成、機能、演算の流れ、マシンの持つ特徴について述べた。現在、マシンの持つ具体的性能の評価、本マシンの適する応用分野の検討を実行中である。

＜参考文献＞

[1] P.Valduriez: "Join Indices", ACM TODS, Vol.12, No.2, pp.218-246 (1987).
[2] B.C.Desai: "Performance of a Composite Attribute and Join Index", IEEE T-SE, Vol.15, No.2, pp.142-152 (1989).
[3] 喜連川, 伏見: "データベースマシン", 情報処理, Vol.28, No.1, pp.56-67 (1987).
[4] 上林: "データベース", 昭晃堂 (1986).