6X-04

メタ質問の並列実行

小倉匠吾　柏川伸悟　松本渉　三浦孝夫

法政大学工学部電気電子工学科

概 要

現在、オブジェクトとメタオブジェクトの継ぎ目の無い操作を行う為、具体化の機能が提案されている。具体化を伴う質問はコストが大きく、これに対し関係代数操作の性能向上, 分散処理, 不要な処理の回避により対応する。本研究では関係代数操作の性能向上の為の属性処理方式の問題点の改善方法, 処理の分散方法, 不要な処理の回避の為の並列化による同期, 不要プロセスのキャンセル, 評価の重複回避の方法について述べる。

1 前書き

メタ質問の実行は、大量なオブジェクトの集合に対する処理を伴わなければならない。このような IceBurg 質問に対しては大量データの高速処理が要求される。メタ質問を扱う上でメタオブジェクトの評価と評価結果の条件反映の処理の対策が欲しい。

我々は逐次処理においてメタ質問の論理的最適化 [3] を既に考えている。これは実行順序によるコストの抑制を狙っているもので、実行単位数の大幅な減少を望むことはできない。本研究ではこの実行単位 (メタ質問) を並列化しコストを分散して、更なる実行時間の減少を目指す。

メタ質問は各メタオブジェクトに対し評価と評価結果の条件反映を行っている。これは単一命令ストリーム-複数命令ストリーム (SIMD) の並列処理をすると、メタオブジェクトの集合に対し一回の評価と評価結果の条件反映を行うことになる。命令を逐次的に行っていくのに対し非常に効率が良くなるので、並列によるオーバーヘッドを上回る処理時間の短縮が望める。

また、ディスクアクセス時間削減とデータの耐久性を目的とした属性単位処理方式の採用について考える。

2 HOME

本研究で扱うシステムを HOME(Harmonizing Objects and Metaobjects Environment) と名付けている。HOME は関係代数に基づく関係データベースシステムであり、メタオブジェクトを扱えることが特徴的である。オブジェクトが単なるデータであるのに対し、メタオブジェクトはオブジェクトの意味, データについてのデータである。スキーマ操作においてはリレーションのメタデータである。

データベース設計では、伝統的なデータベース処理とデータベース設計過程が同時に処理される。これにはスキーマ操作が必要となってくる。今までのデータベースでは、オブジェクトとメタオブジェクトの操作を分離しなければならなかったが、HOME システムでは具体化の概念を組み込みメタオブジェクトをオブジェクトと同様に扱うことによって、データベース設計の途中でもスキーマを変更できる柔軟なデータベース設計を可能にしている。

具体化とはメタオブジェクトの具体化であり、メタオブジェクトを評価することである。具体化の演算には"$"を用いて、リレーション R の具体化は$R と表す。リレーション R=$R(A_1, A_2, \ldots, A_n)$ とすると$R は $< A_1, A_2, \ldots, A_n >$ をタプルとする集合である。メタオブジェクトは射影, 選択及び, 結合等の条件として$を用いてを指定することで、評価される。

3 属性単位処理方式

メタ質問を扱うことによるコストの増加の問題に対し、属性単位処理方式によるディスクアクセス時間の削減による対応を狙う。属性単位処理方式の為に要素を内部的にタプル ID, 属性名, 値のタプル $< Tid, Attr, value >$ で管理する。これよる利点は、関連しない属性のディスクアクセスを回避できるのでディスクアクセスの総データ量が減少することである。更に内部管理にシャドウファイル方式を採用することでデータの耐久性を付与することが可能となる利点も生むことができる。しかし、タプル構築の際には属性数回のディスクアクセスが行われるので負荷が増加してしまう問題が出てしまう。これに対応する為のアイデアとして、索引を採用することを考える。

タプルの構築, 射影, 選択, 結合の Key(親 Key.子 Key) は、$Key(Tid.Attr)$:タプル構築、$Key(Attr.Tid)$:射影, 結合、$Key(Ater.value)$: 選択, 結合となり、3 種類の索引を用意する必要がある。これらは指定された親キーの全子キーを求める。1 度目の探索で先頭のキーを見つければ、次の子キーは次のキーとなるので、探索の必要がなくなり、探索は 1 回で良くなる。$Key(Ater.value)$ において、$value$ のミスマッチはディスクアクセスの必要がない。索引に B-tree を使用すれば更に探索が効率化することが望める。

データはシャドウファイル方式を採用しているが、索引には採用されていない。システム障害やアボート時には、データはセーブポイントの状態に戻るが、索引は戻すことができないので、索引を作り直す必要が出てくる。これに対し、索引の更新もセーブポイントで行い、その間の差分はメモリ領域に保持することで対応する。索引の更新には時間が掛かり、その間はロックされている。もし、その間にシステム障害が起こるとデータと索引の不一致が生じ、索引を作り直さなければならない。これに対応する為にフラグを用意する。索引の更新前にフラグを立てておき、更新後フラグを降ろす。システム起動時に、このプラグが立っていれば、システム障害によりデータと索引の不一致が生じているので索引を作り直す。

4 メタ質問処理

メタ質問は並列化により処理時間の短縮が望めるので、メタ質問の並列処理について考える。

並列化による問題点はプロセスの生成とプロセス間通信のコストである。これを小さくする為には充分な粒度の大きさが必要である。まず、メタ質問における並列プロセスの粒度

Parallel Execution of Meta-Query

Shogo Ogura, Shingo Kashikawa, Wataru Matsumoto, Takao Miura

Hosei University, Dept.of Elec. and Elec. Eng.

Kajino-cho 3-7-2, Koganei, Tokyo, JAPAN

について考える。

データベース操作において、プロセスの最小単位はオブジェクト単位のプロセスである。これではプロセスの粒度が小さすぎて、プロセスの生成とプロセス間通信のコストが大きくなってしまう。メタ質問においてはメタオブジェクト評価と評価の結果を条件に反映させるプロセスの粒度は充分に大きい。したがって、メタ質問をメタオブジェクト単位に並列することで、コスト分散による処理時間の削減が可能となる。メタ質問における並列は、単一プログラム複数データ (SPMD) 構造となるのでマスタ/スレーブ型をとる。

4.1 メタ質問の並列手順

本稿では、メタオブジェクトの評価を伴う射影, 選択, 結合をメタ射影, メタ選択, メタ結合と表記し、メタ質問において評価される属性を$属性と表記する。各メタ質問の具体的なマスタ/スレーブ型の並列を説明する。メタ射影は$属性の評価でタプル毎にスレーブプロセスを生成する。メタ選択は$属性の評価とマッチングでタプル毎にスレーブプロセスを生成する。メタ結合は、まず両リレーションの$属性の評価でタプル毎にスレーブプロセスを生成する。評価完了後、一方のリレーションの$属性タプル毎にスレーブプロセスを生成し、さらにその中でもう一方のリレーションの$属性タプル毎に並列化してマッチングを行う。

【例 1】$\sigma_{\$A\ni"v"}(R(A,B,C)$

$R.A：未評価

タプル毎にプロセス生成

R.A1 評価→マッチング○	例 3 より
R.A2 評価→マッチング○	← Cancel(R.*2)
R.A3 評価→マッチング×→ Cancel(R.*3)	

bcast(R.A 評価完了)

bcast(R.A 作業完了)：$R.A ←評価済

＊プロセス状態,Cancel,bcast() は、後で説明する。

逐次処理ならば並列プロセスが生成されずに順に処理する。並列化によりコストが分散され実行時間の削減が可能となる。

4.2 評価の重複回避

メタ質問を扱うことによるコストの増加の問題に対し、もう一つのアプローチとしてメタオブジェクトの評価の重複回避によりコストの削減を狙う。

$属性が既に評価済みの場合は、評価の重複回避をした処理をする。具体的な処理を説明する。メタ射影は評価のみの処理なので、メタ質問でない射影と同様に扱うこととなる。つまり、スレーブプロセスを生成する必要がない。メタ選択はマッチングのみでタプル毎にスレーブプロセスを生成する。メタ結合はもう一方の$属性が未評価ならば、タプル毎にスレーブプロセスを生成する。評価完了後、単独のメタ結合と同じようにマッチングを行う。

【例 2】$\pi_{\$A,B}$(例 1)

$R.A：評価済

評価が行われているのでメタ質問でない射影

R(A,B,C) → R($A,B)

4.3 不要プロセスのキャンセル

プロセスを並列化したことにより、不要となったプロセスをキャンセルすることが可能となる。これはメタ質問を扱うことによるコストの増加の問題に対し、コストの削減により対応を狙う。

選択や結合によるタプルのミスマッチが起きた時、ミスマッチタプル内のメタオブジェクトのプロセスが不要となる。逐次処理においては不要プロセスの発生頻度を少なくする為に、メタ質問の論理的最適化 [3] がなされている。本研究では、並列処理することでプロセス生成後に不要となったプロセスを処理中にキャンセルすることを可能にすることを考える。

プロセスが完了してしまった後に不要プロセスを発見しても、それをキャンセルすることはできない。そこでプロセスの処理中に不要プロセスを発見することができれば、不要プロセスのキャンセルが可能となる。このために、メタ質問の処理を並列プロセスとして処理の完了を待たずに他の処理を行うことで不要プロセスの発見を可能にする。それ以外の質問はメインプロセスで逐次処理をする。

【例 3】$S(D,E) \bowtie_{D=B}$ (例 2(例 1))

結合：マッチング

R.B1 ○ — S.D1 とマッチング○ / S.D2 とマッチング× / S.D3 とマッチング×

R.B2 × — S.D1 とマッチング× / S.D2 とマッチング× / S.D3 とマッチング×

→ Cancel(R.*2)

R.B3 ○ — S.D1 とマッチング× / S.D2 とマッチング× / S.D3 とマッチング○

逐次処理ならば例 1 が完了してるので R.*2 のキャンセルができない。並列化することで処理の完了を待たずに不要プロセスを発見しキャンセルすることが可能となる。

4.4 同期

メインプロセスは並列プロセスの完了を待たずに処理を行っていくので、同期の必要が出てくる。$属性の関連するメタ質問の完了を待機しなければならないのは、同じ$属性が関連するメタ質問が要求される場合である。ただし、評価済のメタ射影は評価を行わないので除かれる。また、リレーションを出力する時には、リレーションの全属性の全プロセスの処理完了を待機する必要がある。

【例 4】出力 (例 3(例 2(例 1)))

?$S.D:未評価 , ?$S.E:未評価,

?$R.A:評価中 , ?$R.B:未評価

recv(R.B 作業完了)

例 1 で行われている$R.A の作業完了のブロードキャスト (bcast()) による信号を待機 (recv()) している。

出力 (T($A,B,D,E))

5 結び

本研究で、HOME システムで属性単位処理の為の内部管理とデータの耐久性の付与を実現し、メタ質問の並列化による不要プロセスキャンセルと評価の重複回避が可能となった。

参考文献

[1] 北川博之：データベースシステム, 昭晃堂 (1996)

[2] 飯塚肇, 緑川博子：並列プログラミング入門, 丸善株式会社 (2000)

[3] 松本渉, 三浦孝夫：メタ質問の論理的最適化

[4] 松本渉, 田淵勇一郎, 三浦孝夫：オブジェクトとメタオブジェクトの調和