並列論理型言語によるトランザクション処理*

9F−4

庄野 温夫, 田中 英彦†
東京大学工学部

1 はじめに

オンラインデータ処理に代表されるトランザクション処理は、計算機システムの応用分野として重要であり、高性能化が強く求められている。本稿では、並列論理型言語によるトランザクション処理システムの試作を通じて、言語の特性を考慮しながら、トランザクション処理に内在する並列性を引き出す実現方法について検討する。

2 研究の背景

近年、UNIX 上の商用データベース管理システムが並列・分散処理システムに次々と対応しつつある。これはトランザクション処理への並列計算機の適用の有望性の現れの一面である。分散処理システムにおいては対障害性の観点から分散資源の利用をトランザクションとして処理することが重要となる。

一方、高並列計算機による記号処理・知識処理に関する研究の成果の一つである並列論理型言語は、問題に内在する並列性を損なうことなく処理を記述し、並列に実行される処理間の同期を制御するのに適しているという特徴を持つ。これまで知識ベースに論理型言語を応用する研究がなされてきたが、データの取り出しだけでなく頻繁に更新が生じるトランザクション処理という観点からの検討が十分であったとは言えない。

本稿では、上記の背景に注目し、データベース・トランザクション処理を例題にとり、データの更新を含むトランザクション処理への並列論理型言語の適用について検討する。

2.1 データベース表現

検索のためにインデクスを用いた構造やハッシュを用いた構造が候補として考えられるが、キー値範囲を指定するアクセスや並行制御との適合性から、木インデクスを用いた構造が有利である。

また、並列論理型言語の枠組では、定義節が変数のスコープであり、すべての変数は単一代入である。並列論理型言語の枠組でデータベースを表現するためには、この言語の特性を考慮する必要がある。

*Transaction Processing with Parallel Logic Language
†Atsuo Shono, Hidehiko Tanaka
University of Tokyo

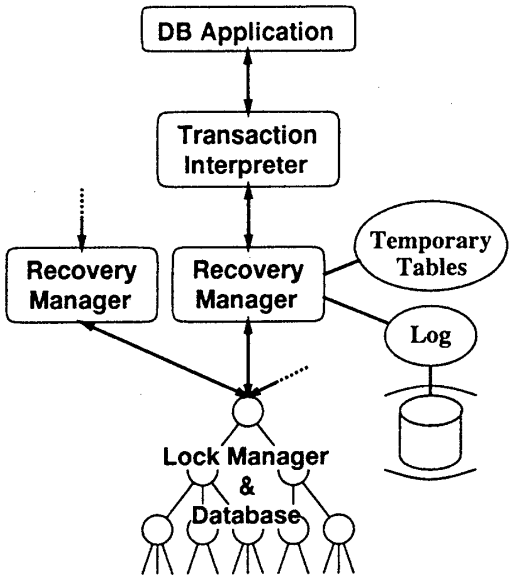

図 1: 試作システムの構成

なお、言語の実行環境である主記憶にデータベース全体がロードされることを基本的には仮定している。

2.2 トランザクション処理

トランザクションとは、データ(共有資源)の利用、処理、変更の一連の操作をひとまとめにした単位である。複数のトランザクションが並行して実行される時、データの更新が競合することがあるので、資源の状態の一貫性を保つための並行制御が必要である。また、最後まで実行されたトランザクションの影響が失われることがあってはならないが、実行途中で取り消されたトランザクションの影響は無効化しなければならない。

3 実装方式

試作システムの構成を図1に示す。

3.1 並列論理型言語 Fleng

実装にあたっては、我々の研究室で開発され利用している言語 Fleng [2] を用いた。Fleng は PARLOG や GHC と

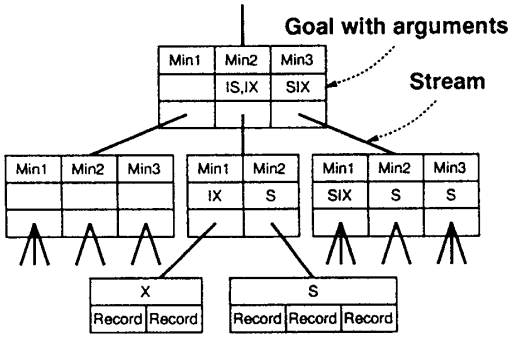

図 2: 木インデクス構造とロック情報

同様 Commited-Choice 型言語の一つである。また Fleng の並列実行を仮想的にシミュレートするインタプリタも開発されており、試作システムの評価にはこれを用いる。

3.2 データベース内部表現

並列論理型言語における変数の特性により、あるゴールの持つデータを更新するには、更新前のデータを持つ変数から更新後のデータを持つ変数を新たに作り、リダクション後のゴールに後者を渡すという方法が自然である。大域的な存在であるデータベースを表現するには、データベースを参照するすべてのゴールが相互に通信のための変数を共有する必要がある。すなわち、データを保持しながらテールリカージョンによって生き続けるゴールとそれらを結ぶ共有変数のストリームによりデータベースを表現する。データの検索や更新の操作のためのメッセージは、木の探索と同様に選択されたストリームに流されて処理される。これはデータベースのアクセスパスを表現したものと考えられるので、木インデクス構造をそのまま表現することができる。試作システムでは木インデクスを原理的に実現するものとして 2-3 木構造を用いた。

さらに、2-3 木への挿入や削除による構造書き換えを木の深さ方向にパイプライン的に並列処理する工夫を行なった。

3.3 トランザクション処理

並行制御とコミット (正常終了)・ロールバック (異常終了) 処理の実装方法について述べる。

試作システムではもっとも基本的な 2 相ロックに基づく並行制御方式を採った。また、木インデクス節にロック情報も記憶することにより、データ操作の範囲に応じた粒度のロックを掛ける方式 [1] を採った。これにより、存在しないレコード (phantom) の問題も解決される。

木インデクス構造を Fleng のゴール・プロセスとストリームで表現し、そこにロック情報を記憶する概念図を図 2に示す。

トランザクションが実行する挿入・削除・更新を記録しておき、ロールバックする際にはそれを逆向きに取消実行する手法を用いる。

3.4 アクセス・インターフェース

試作システムでは、データベースにアクセスするアプリケーションも Fleng で記述することを仮定し、データベース操作をストリームで受け付けるインターフェースを設定した。

- select_into([表 1, 表 2,...], 結果表, 条件) — 結合・選択
- aggregate(元表, 結果表, 演算) — 総和など
- insert(対象表, 要素表), delete(対象表, 要素表)
- update(更新前レコード, 更新後レコード)
- assign(表名,Var) — Fleng 変数への代入

上記のデータベース操作を受け付け解釈実行するモジュールは、これらを下位のデータベース操作に分解し、コミット・ロールバック処理モジュールに引き渡すとともに、コミット時に一括して外部 Fleng 変数への代入などの処理を行なう。

4 おわりに

並列論理型言語で記述することにより、木インデクス構造の更新や、共有データへの競合アクセスの並行制御の際に必要な同期を取りながら、システムの各モジュール内やモジュール間の並列処理が行なえることがわかった。現在、Fleng の仮想並列実行系を用いて定量的評価を行なっている。

また、並列論理型言語におけるゴール・プロセス、ストリームによる表現をデータベースや並行制御のためのデータ構造に用いることによって、古典的なデータベース管理手法をそのまま応用できることがわかった。

今後の課題として、試作システムを実際の並列計算機で効率的に実行する実行系の検討、プロセス・ストリームで表現されたデータベースをディスクに退避・復帰することにより永続的データを並列論理型言語から利用する枠組の確立、試作システムをエレメントとして分散トランザクション処理システムへの拡張などが挙げられる。

本研究を行なうにあたり多大の支援を賜わった田中英彦研究室の諸氏ならびに京大工学部情報工学教室の大森匡氏に感謝致します。

参考文献

[1] J. Gray, et al: Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1993.

[2] M. Nillson, et al: FLENG Prolog - Turning Supercomputers into Prolog Machines, *Proc. of Logic Programming Conf. '86*, 1986.