4K－4

複合トランザクションモデル —入れ子モデルとのシミュレーション比較—

本田 治　多田 知正　樋口 昌宏　藤井 護

大阪大学 大学院基礎工学研究科 情報数理系専攻

1 はじめに

近年のデータベースにおいては，トランザクションの生存時間が長いものが存在するため，アボート時の再実行のオーバーヘッドが大きくなるという問題が存在する．この解決策の１つとして，入れ子トランザクション(nested transaction)が提案されている[1]．入れ子トランザクションは，複数の部分トランザクションからなり，部分トランザクション毎の再実行が可能である．しかし，部分トランザクション単位のスケジューリングができないため，高い並行性が得られない．そこで，我々は複合トランザクションモデルを提案している[2]．複合トランザクションは，それを構成する要素トランザクション単位でスケジューリング可能なため，高い並行性が期待できる．

本稿では，複合トランザクションの有効性をシミュレーションにより評価する．

2 トランザクション

トランザクションとは，データベースを処理するプログラムの実行のことであり，読み出し操作と書き込み操作によってデータベースを操作する．トランザクションは単体で実行された場合，データベースの無矛盾性を保存する．これをトランザクションの一貫性という．また，トランザクションは，正常に終了した場合にトランザクションがデータベースに及ぼした影響は保存され，異常終了した場合にはまったく影響を与えない．これをトランザクションのアトミック性という．トランザクションが正常に終了し，その結果を保存することをコミットという．正常に終了しなかった場合に，その結果を無効にすることをアボートという．

3 入れ子トランザクションとその問題点

入れ子トランザクションは，部分トランザクションを含むことができる．部分トランザクションは，さらに部分トランザクションを持つことができる．入れ子トランザクションは図１に示すような木構造で表される．木の各頂点は部分トランザクションに対応している．

入れ子トランザクションは，全体として一貫性及びアトミック性を満たす必要があるが，個々の部分トランザクションは，一貫性を満たす必要はない．入れ子トランザクションは，ある部分トランザクションがアボートした際に，その部分トランザクションと，その子孫の部分トランザクションのみを再実行するだけでよいため，再実行のオーバーヘッドが小さくなる．しかし，部分トランザクション単位のコミットは，行うことが出来ない．入れ子トランザクションの全ての部分トランザクションはまとめてコミットされなければならない．

入れ子トランザクションの全ての部分トランザクションが，一貫性を満たす場合を考える．このとき，各部分トランザクションを別々にスケジューリングすれば，高い並行性が得られる．しかし，入れ子トランザクションでは，部分トランザクションが一貫性を満たすことを仮定していないため，部分トランザクション単位でスケジューリングを行うことはできない．入れ子トランザクションではなく複数のトランザクションを用いて実装することも考えられるが，この場合は，全体のアトミック性が失われてしまう．これを解決するためには，部分トランザクション単位でスケジューリングでき，かつ全体でアトミック性を保証するようなトランザクションモデルが必要である．

4 複合トランザクション

前述の条件を満たすモデルとして，複合トランザクションモデルを提案している．複合トランザクションは，一貫性を満たすトランザクション(要素トランザクションと呼ぶ)の集合である．複合トランザクションは，入れ子トランザクションと同様の木構造を持つ．複合トランザクションでは，要素トランザクション単位でスケジューリングが可能である．また，複合トランザクションは，アトミック性を満たさなければいけない．

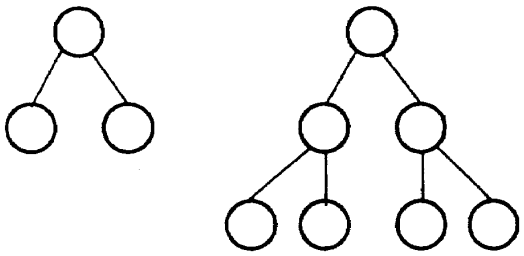

図 1: 入れ子トランザクションの例

複合トランザクションにおいても,要素トランザクション単位のアボートと再実行が可能である.しかし,複合トランザクション全体としてのアトミック性を満たすため,要素トランザクション単位のコミットはできない.複合トランザクションの全ての要素トランザクションはまとめてコミットされなければならない.

5 シミュレーション

5.1 シミュレーションの目的

入れ子トランザクションの中には,全ての部分トランザクションが一貫性を持つものが存在する.その際,そのトランザクションを,複合トランザクションと、して実装することにより,トランザクションの高い並行性が得られ,これにより,データベースのトランザクション処理の性能が向上すると考えられる.複合トランザクションのデータベースの性能に与える影響を評価する際,スケジューリングを含めたそのふるまいを数学的モデルで記述することは困難であるので,シミュレーションによってデータベースのスループットを測定し,比較を行った.

5.2 シミュレーションの方法

各トランザクションは,いくつかのデータ項目からランダムに選んだデータ項目に対しアクセスする.アボートされたトランザクションは必ず再実行されるものとする.並行に処理されるトランザクションの数をマルチプログラミングレベルという.

シミュレーションでは,通常のトランザクションと入れ子トランザクションをランダムに次々と発生させ,スループットを測定する.入れ子トランザクションは,図 1 に示した 2 種類のいずれかの構造をもつ.次に,入れ子トランザクションの一部を,同じデータにアクセスする複合トランザクションに置きかえて同様に実験を行い,両者のスループットを比較する.

6 結果

シミュレーションでは,部分トランザクションは,それぞれ 2000 のデータ項目の中からランダムに選んだ 4 つの項目にアクセスさせ,通常のトランザクションは 8 つの項目にアクセスさせた.トランザクションのスケジューリングは,並行性の高いことで知られる逐次化グラフスケジューリング (Serialization Graph Testing)[3] を用いて行った.マルチプログラミングレベルを変化させ,各レベルでスループットを測定し比較した.その結果,マルチプログラミングレベルが高いほど複合トランザクションを導入した場合のスループットが,入れ子トランザクションのみの場合より高くなった.これは,マルチプログラミングレベルが高いときには,トランザクションの衝突が頻繁に発生し,その結果,並行性の低い入れ子トランザクションでは,トランザクションのアボートが頻発するためであると考えられる.

7 おわりに

本稿では,複合トランザクションモデルを導入することにより,データベースのトランザクション処理の性能が,入れ子トランザクションモデルのみを用いた場合より向上することをシミュレーションにより確かめた.

参考文献

[1] J. Boss, J. Eliot : "Nested Transaction: An approach to reliable distributed computing", MIT Press(1985).

[2] 多田 和正, 樋口 昌宏, 藤井 護 : "逐次化グラフを用いた複合トランザクションの並行制御", 情処研報, 98-DPS89-2(1998 年 6 月).

[3] P. A. Bernstein, V. Hadzilacos and N. Goodman : "Concurrency Control and Recovery in Database System", Addison-Wesley(1987).