索引順編成ファイルの排他制御

4P−2

松井　浩二　　石橋　英次
（株）東芝　府中工場

１．はじめに

索引順編成ファイル(ＩＳＦ)の構成法には、レコード部を主キーでソートして格納する方式と、到着順に格納する方式がある。一般には主キーでソートする方式が用いられるが、我々は複数のキーによるアクセスを考慮し到着順に格納する方式を採用しており、インデックスにはB*ｔｒｅｅを採用している。

本稿ではＩＳＦのアクセスをトランザクションとして実行するときの排他方式の一実現法について述べる。本方式は、インデックス、レコード共にブロック単位でロックし、同時実行性を高くすることを特徴とする。

２．レコード検索

図１．Ａはレコードの検索処理におけるロック手順を表している。インデックスのルートブロックから始め、葉に相当する転置ブロック、最終的にレコードブロックのロックを確保する。

データの一貫性・独立性を保証するために、一度確保したレコードのロックはトランザクションが終了するまで解放することができないが、インデックス部のロックは検索が終了した時点で解放することができる。同様に、レコードブロックは更新の有/無に応じて共有/占有モードでロックするが、インデックス部は共有モードでロックすればよい。ただし、インデックスの正当性を保証するために、次のレベルのロックが確保できた後に現レベルのロックを解放しなければならない。

しかし、既に述べたようにレコードは複数アクセスにまたがってロックされるため、レコードブロックのロック確保待ちの間、転置ブロックのロックを確保しておくと、デッドロック発生の要因となる。一方、レコードは削除されることはあっても、インデックス部のようにブロックの一部が他のブロックに移動されることはない。この特性により、レコードブロックのロック確保前に転置ブロックのロックを解放することができる。ただし、ロック待ちの間にレコードが削除された場合には再びインデックスを検索する必要がある。

３．レコード追加・削除

図１．Ｂはレコード追加時のロック手順を表している。レコードはファイルの最終位置に追加される。レコードの同時追加制御については報告済みなので[1]本稿では割愛する。

インデックス部のロック手順は検索フェーズと更新フェーズの２つに分けることができる。検索フェーズではルートブロックからキーを挿入すべき転置ブロックに至るパスを全てロックする。このとき、転置ブロックは占有モードでロックするが、それより上位のインデックスは必ずしも変更が生じないので共有モードでロックする。

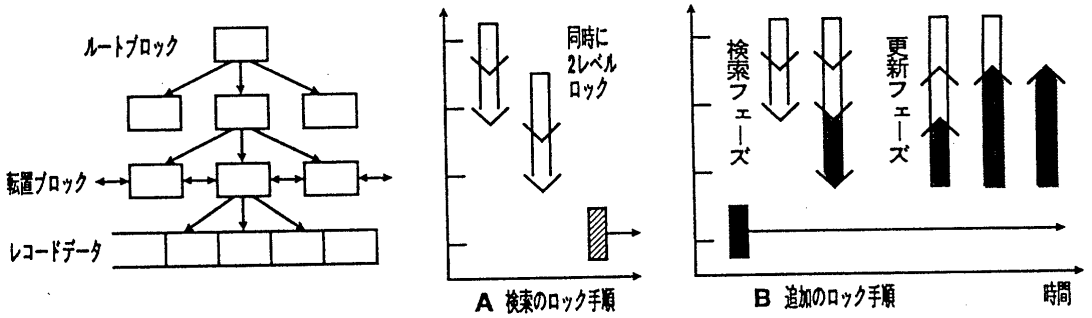

図1 索引手順編成ファイルの構成とロック手順

A Lock Mechanism for Transaction processing of Indexd Sequential File.
Koji MASTUI,　Eiji ISHIBASHI
TOSHIBA CORPORATION

更新フェーズでは、インデックスの分割が必要な場合に検索フェーズで確定したパスを逆順に占有モードでロックする。一段上の占有モードロック確保後、同レベルの隣接ブロックを占有モードでロックし、再配分あるいは分割処理をおこなう。更新フェーズは転置ブロックから再帰的に上位へ波及する。最終的に、更新のあった部分は占有モードでロックされトランザクションが終了するまで保持され、更新のない部分の共有モードロックは解放される。

レコード削除処理は追加処理と同様の手順で実行することができる。

4．デッドロックに関する考察

本方式では、ＩＳＦに対してインデックスの整合性を保ったままレコードの検索、追加、削除を同時実行することができるが、デッドロック発生の可能性がある。

レコードの追加、検索・更新、検索・削除をＩＳＦに対する基本処理と考える。トランザクション内で１回の基本処理のみ実行する場合にデッドロックが発生するのは、図２に示すように、インデックスの分割・併合処理途中に発生する２つのケースだけである。

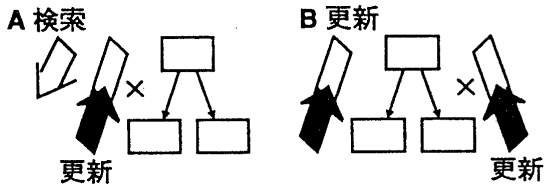

図2インデックス処理におけるデッドロック

１つは更新フェーズと検索との間に発生するもので、もう１つは更新フェーズのロックモード変換同士で発生するものである。いずれのケースもデッドロック検出されなければ、インデックスを正しく検索できない、あるいはインデックスに矛盾を引き起こしてしまう。

このようなインデックス処理におけるデッドロックはシステムの内部処理の都合で発生するものであり、アプリケーションへの影響を少なくするためデッドロック発生時には再試行される。検索におけるデッドロックの場合は、現レベルのロックを解放した後ルートブロックから検索を行い、更新フェーズでのデッドロックはそのまま再びロックを要求する。

インデックスの分割・併合の発生確率が低いことから、１回の再試行により内部処理によるデッドロックのほとんどを回避することができるが、再試行の結果がデッドロックとなった場合はエラーリターンし、トランザクション全体のロールバック再試行に期待する。

トランザクション内で複数の操作を行う場合はインデックスの複数部分が占有モードでロックされるためデッドロックを回避することは難しい。しかし、インデックスの更新が１つの転置ブロックに集中する場合は、分割・併合が発生しない限りインデックス処理に関するデッドロックは発生しない。転置ブロックのロック確保待ちのタスクが持っているのは上位インデックスの共有モードロックなので、複数回更新するときの検索フェーズとは競合しないためである。

5．インデックスの再分配に関する考察

本方式では、インデックス処理におけるデッドロックは主に分割・併合処理で発生する。デッドロック発生を少なくするには、分割・併合の発生回数を少なくすればよい。

我々は、B*ｔｒｅｅの分割処理で、新しくブロックを追加するか、隣接ブロックへキーを均等に移動するかの判定において、移動できなければ追加するではなく、移動量が少なければ分割することにした。

レコードをランダムに追加する実験を行った結果、いき値をインデックスの分岐数の１０％としたとき、移動回数が４０％減少し、分割回数にはほとんど差がないことを確認した。

6．おわりに

本稿ではＩＳＦのアクセスをブロック単位の排他により実行する方式を紹介し、さらに、デッドロック発生条件や、その対策について考察した。

我々は、マルチプロセッサ構成のミニコンピュータ上に、典型的な受発注システムを構築し、同時実行性が十分に引き出され、デッドロックが問題にならないことを確認した。

＜参考文献＞

[1] 鈴木、松井、森「ﾐﾆｺﾝﾋﾟｭｰﾀでのｼﾞｬｰﾅﾙﾘｶﾊﾞﾘ高速化手法」情処全大第41回、5D-3