2G－2　並列DBサーバHiRDBに於ける同時実行性の高いインデックス処理方式

梅崎　浩司*　正井　一夫*　宮崎　光夫*　原　憲宏**　河村　信男**
*日立製作所ソフトウェア開発本部　**日立製作所システム開発研究所

１．はじめに

並列データベースは，高いスケーラビリティを大きな特長とする．すなわちプロセッサやディスクの追加に見合ったレスポンスタイムの向上を得られること，また，データ量やトランザクション量が増加してもプロセッサやディスクを追加で一定のレスポンスを得られることが重要である．

日立製作所は，検索だけでなく更新処理や運用を含めて並列実行可能な日立スケーラブルデータベースサーバ「HiRDB(ハイ・アール・ディー・ビー)」を開発した．本稿ではHiRDBにおけるDBアクセスサーバでの並列更新を実現するために特に同時実行性を向上させるインデクス処理方式について示す．

２．HiRDBの構成と特長

HiRDBは，UNIXベースの並列データベースであり，日立3500サーバのクラスタ構成上に開発した．本システムは，各プロセッサに割り当てられた複数のサーバから構成する．

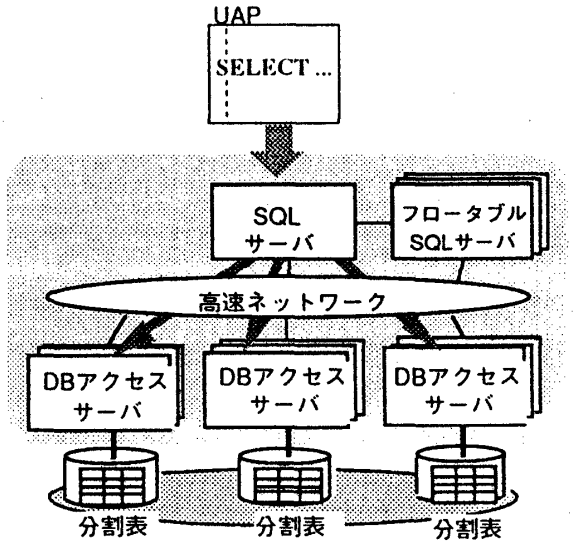

図１　HiRDBの構成

サーバには図１に示すようにSQLを受け付けて解析し並列実行するSQLサーバやその並列化実行指示に従いDBをアクセスするDBアクセスサーバがある．また，ソートやジョインの様なCPU負荷の高い処理を行なうフロータブルSQLサーバがある．

HiRDBには多くの特長があるが，その中でも更新処理の並列実行は，大きな特長である．更新系のSQLに関しては，並列実行をしないものが多いがHiRDBは，検索処理同様に並列化している．

更新系SQLの並列実行を実現するために特に以下の点に考慮した．

・デッドロック
・同時実行性（排他制御）
・トランザクション同期制御

HiRDBでは，シェアド・ナッシング型のアーキテクチャを採用するため，基本的にはノード間をまたがる排他制御は必要としない．しかし，更新処理要求の多重化に伴いデッドロックの危険性が増大し，システムの同時実行性を著しく低下させることになりかねない．デッドロックを検出，回避することも必要であるが，デッドロックを起こりにくくすることが重要である．本稿ではデッドロックを起こりにくくするために特別に強化したインデクスの排他処理について紹介する．

３．HiRDBにおけるインデクス制御

HiRDBでは安定したアクセス性能を提供するB－treeインデクス構造を採用した．更新系SQLを実行すると，データの更新に伴い，そのデータに関連したインデクスキーの更新を必要とする．このインデクスキーの更新と他の検索・更新処理とが，複数のユーザのトランザクション間で衝突しないように，排他機能で制御する．その際，更新処理が保持する排他で検索処理が長時間待たされ，パフォーマンス低下とならないように排他範囲を最小に，排他期間を最短にするように考慮し開発した．以下にHiRDBで強化した点を示す．

High Concurrency Index Management Method for Parallel DB Server "HiRDB"
Hiroshi Umezaki*, Kazuo Masai*, Mitsuo Miyazaki*, Norihiro Hara**, Nobuo Kawamura**
*Software Development Center, Hitachi Ltd., **Systems Development Laboratory, Hitachi Ltd.

■キーロックとページラッチ

排他範囲を最小とするため，インデクスを構成する最小構成要素であるインデクスキーに対して排他を行う．インデクスキーに対する排他は更新処理においてトランザクション完了まで保持され更新結果の論理整合性を保証する．またページに対するラッチはデータベースバッファ中のインデクスページ操作中だけ保持し，操作完了後即座に解放する．上記のキーロックとページラッチにより，インデクスページ操作中（メンテナンス中）の瞬間でなければ同一のインデクスページ内の異なるキー値に対する処理の同時実行が可能である．

■スプリット処理における同時実行性の向上

インデクスキーのメンテナンス時にインデクスページ内のスペースが不足すると，そのページを分割することで挿入領域を確保しインデクスのB－tree構造のバランスを保つ処理を実行する．この処理を「スプリット処理」と呼ぶ．スプリット処理中でもインデクス全体に対するロックを行わずに，スプリット処理対象となるページのラッチだけ行う．このためスプリット処理中でも他のトランザクションからインデクス操作可能となる．スプリット処理中の同時実行を実現するためには，まず，スプリット処理中の崩れたB－tree構造においても，インデクスの操作処理がサーチ対象までたどり着く必要がある．インデクスのサーチ対象キーがスプリット処理で他のページに移動してしまった場合でも，図2で示すように，スキャンキーの値と当該ページで管理している最大キー値とを比較することで，スプリット処理との競合を検知しサーチパスを分岐できる．具体的にはインデクスページ内に右ページへのポインタを持たせることでことで新ページへの移動を可能としている．この方式を「リポジショニング方式」と呼ぶ．

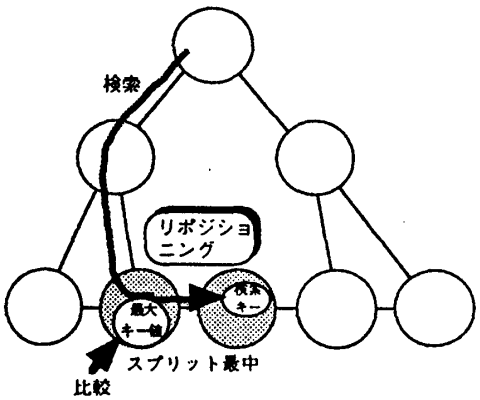

図2　リポジショニング方式

リポジショニング方式でサーチ対象を見失うことがなくなり，インデクスの再サーチが不要となる．また，トランザクションがスプリット処理を行っている最中に中断（ロールバック）しても，他の更新トランザクションは未完のスプリット処理影響範囲にアクセス可能であるため，更新結果を保証する必要がある．このためにロールバックにおいてもインデクスのスプリット処理を完成（ロールフォワード）させることでB－tree構造を決してスプリット処理発生以前の状態に戻さない方式を採用した．この方式により他トランザクションのスプリット処理による影響をなくすことが可能となった．

4．効果

■同時実行性の高いキー排他を採用．

同一ページ内の別キーでの待ちがない．

■キーを更新中でもアクセス可能．

更新の瞬間だけページラッチを行うため，更新トランザクションの終了を待たずにアクセスできる．

■スプリット処理中でもアクセス可能．

スプリット処理中でも更新処理時のページラッチのみを行うことで，トランザクション終了まで待たずにアクセス可能．

■唯一のロック競合が同一キーの更新時のみ．

ユーザの意図しないデッドロックの防止．

5．おわりに

インデクスの同時実行性向上だけがHiRDBが更新系SQLの並列実行を実現したわけではない．多くの更新に関連する同時実行性向上機能及び回復機能が必要であった．しかし，従来からインデクスに関するデッドロックは多く，今回紹介したインデクスの強化は大きな要因といえる．

[参考文献]

1. Lehman, P., Yao, S.B.: Efficient Locking for Concurrent Operations on B-Trees, ACM Transactions on Database Systems, Vol 6, No.4, December 1981.

2. Mohan, C., Levine, F.: ARIES/IM: An Efficient and High Concurrency Index Management Method Using Write-Ahead Logging, ACM-SIGMOD International Conference on Management of Data, San Diego, June 1992.