垂直分散における一時データ管理方式の検討

3R−4

元田　敏浩、　黒川　裕彦、　徳丸　浩二
（ＮＴＴソフトウェア研究所）

1. はじめに

我々はこれまでに垂直分散型のデータベースシステムにおける分散不可視化およびユーザインタフェース向上の手段として、データベース検索結果を一時的に蓄えて利用する一時データ管理方式を検討してきた。またこの方式を、ＭＳ−ＤＯＳのシングルタスク環境で動作する垂直分散型データベース応用システムにデータベース検索結果の格納機能として実装してきた。今回　同システムをマルチタスクＯＳ上に実現するにあたって、これまでの一時データ管理方式をマルチタスク環境下で動作し、しかもユーザインタフェースの高度化に対応した一時データ管理方式とすべく検討を行った。

本稿では、これまで検討を行った一時データ管理方式である『データリスト方式』について述べる。

2. 背景

当初開発した垂直分散型データベース応用システムにおいては、ローカルＤＢは市販ＤＢＭＳを埋め込みＳＱＬ方式で用いリモートＤＢは社内開発したＤＢＭＳを通信回線を介してコマンド方式により利用する２つの異なったＤＢをＡＰ（Application Program）に対して共通化して見せる必要があった。これは、データベースアクセスにＳＱＬを用いる事で、ある程度の分散不可視化を達成したが、ローカルとリモートでアクセス手続きが異なる点や、本システムのＳＱＬの検索系ではバックフェッチができないという制約があった。これらの問題点を解消するために、検索結果を全てローカルマシン上に蓄積する事により、ＡＰにはＳＱＬの発行と一時データへのアクセスという形で共通化したインタフェースで検索結果を自由に取り出せる一時データ管理方式を考案した。さらに、一時データ化する事により複数の検索結果を保持する事も可能となった。（図１）

今回　同システムをマルチタスクＯＳ上へ移植するにあたって検索結果を一時データ化する事が必要かどうかという点も含めて再検討を行った。その結果、ユーザインタフェースを持たないバッチ処理的なＡＰの場合には検索結果の一時データ化の必要性が殆どないが、ユーザインタフェースを持つＡＰの場合には　依然として一時データ化の必要性がある場合が多いという結論に達した。これをふまえ、一時データ管理方式である『データリスト方式』をマルチタスク環境下で矛盾無く動作し、ユーザインタフェースを持つＡＰ向けの機能を強化する事にした。

3. データリスト方式

表１に示すのは、データリスト方式全体の機能構成である。ＡＰからの指示で検索を行いその結果を一時データとして蓄積する基本機能は既に実現した機能である。それ以外の部分がデータリスト方式として新設した機能である。

共有機能は、マルチタスク化に伴う排他制御や複数のＡＰ間のデータリスト受け渡し機能を提供する。

ＤＢ反映機能は、データリストを修正した場合、その修正操作を元のデータベースに反映させるというもので、これまでは各ＡＰが独自にこの機能を作り込んできた。しかし、マルチタスク化に伴い変更情報を一元管理する必要があるためと、ＤＢ反映方法もＤＢＭＳ毎に異なるためＡＰがＤＢ反映を行うよりデータリスト側に反映機能を持たせるべきであるという分散不可視およびオブジェクト指向的観点から、データリスト方式の１機能として取り込む事にした。

変更通知機能は、データリストに加えられた変更操作を、データリストを共有する全ＡＰに通知する機能である。これは特にユーザインタフェースを持つＡＰを意識したもので、例えば画面表示等でデータリスト上のデータを加工して利用する場合は、変更があれば直ちに再加工をする必要があり、本機能はそのトリガを与える。

ｕｎｄｏ機能もユーザインタフェースを意識したものであり、ＤＢ反映前のデータリストに加えられた変更を必要に応じて元に戻す事ができる機能である。

以降は、これらの機能のうち変更通知機能について述べる。

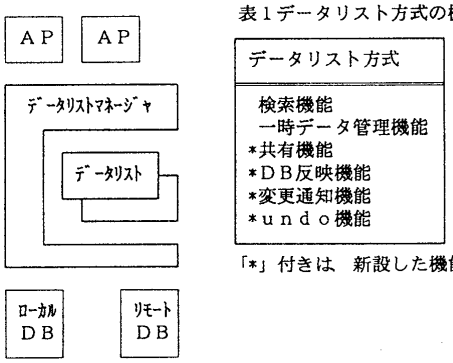

図１　データリスト方式の全体構成

表１データリスト方式の機能構成

データリスト方式
検索機能
一時データ管理機能
*共有機能
*ＤＢ反映機能
*変更通知機能
*ｕｎｄｏ機能

「*」付きは、新設した機能

A study for temporary data storing mechanism on vertical distributed system.
Toshihiro MOTODA, Hirohiko KUROKAWA, Koji TOKUMARU
NTT Software Laboratories.

4. 変更通知機能

変更通知機能は

(1) 通知のタイミングを作り出す排他制御機能

(2) 登録した領域内の変更を監視する監視域管理機能

から構成される。以下各要素機能について述べる。

4. 1 排他制御機能

例えばデータリストに対する変更が複数のＡＰから連続して行われた場合には、変更を通知すべき安定した状態が得られない。また、あるＡＰが変更中に他のＡＰが読みとりを行うと、変更中の一時的に一貫性のなくなったデータを読みとってしまう可能性が大きい。このため、変更通知を行うにあたっては、データリストへの各ＡＰからのアクセスを、データリストに変更を加える期間と加えられた変更を参照する期間とに時間的に分離する必要がある。

そこで、表２に示すようにデータリストに書き込み権および読み込み権を設け、それぞれが排他的になるようにする事によって両者を時間的に分離することができた。ここで、書き込み権取得中に他のＡＰの書き込み権の取得をブロックするのは主にデータリストマネージャの処理の簡単化のためと、ユーザインタフェースを持つＡＰの場合は共用しているデータに変更を加える主体は操作者であり同時に複数の変更を加える事はあまりないと思われるためである。

この排他制御機能によって、書き込み権を取得していたＡＰが書き込み権を放棄した直後すなわち、データリストの変更を行う期間から読みとりを行う期間に入る時に、データリストを共有している全ＡＰに対して、変更通知を送る。

表２ データリストにおける排他制御用権利

権利名	作用
書き込み権	読み込み権取得を失敗させる 他のＡＰの書き込み権取得をブロックする
読み込み権	書き込み権取得をブロックする

4. 2 変更監視域

画面等表示デバイスの大きさの制限のため、一般にユーザインタフェースを持つＡＰで、ある瞬間に同時に表示されているデータは、検索結果データ全体の一部分である場合が多い。このため、発生した変更動作の位置に関係なく変更通知を受け取る機構としてしまうと、受け取る変更通知の大半が再加工の必要がない部分に対する変更によって発生した物となる可能性が大きい。これにより、ＡＰに無駄な処理を行わせる事になり、システム全体としてのパフォーマンス低下を招く恐れがある。

そこで、これらの特性をふまえ、データリストの全レコードのうち特定の範囲を『変更監視域』としてあらかじめ登録し、変更通知時に変更監視域内に変更があったかどうかを示すフラグを付加する。これにより、ＡＰはデータの再加工をする必要があるかどうかを判断でき、ＡＰの変更通知受信処理の負荷を軽減させることができる。

4. 3 失敗を考慮した変更通知処理

データリストの利用法によっては、変更通知がＡＰに伝播するまでの遅延時間が問題となる場合がある。例えば、書き込み権の要求が連続して発生して、書き込み権が放棄されるやいなや別の書き込み権が取得されてしまうという場合である。この場合、変更通知を受け取ったＡＰが読み込み権を取得しようとしても、既に取得された書き込み権のために取得に失敗する。

このように変更通知を受け取った後に読み込み権が取得できなかった場合でかつ、変更監視域内に変更があった場合は、ＡＰは次の変更通知を受けた時に再度読み込み権の取得を試みる。この動作が可能なように、変更通知は変更監視域内の変更の有無にかかわらず毎回ＡＰに送る必要がある。

5. ＡＰから見た変更通知処理

図２は、２つのＡＰ（ＡＰ１、ＡＰ２）が１つのデータリストを共有して動作している場合のタイミングチャートの例である。ＡＰはあらかじめデータリストマネージャに対して、変更監視域を登録しておく。

データリスト内容を変更する場合は、データリストマネージャに対して書き込み権を請求し、書き込み権が得られたなら必要な変更を行う。その後、書き込み権を放棄するとデータリストマネージャは、全ＡＰに対して変更通知を送る。

変更通知を受け取った各ＡＰは、変更監視域内に変更があったかどうかのフラグを調べ、変更があった場合は読み込み権を取得して、必要な領域のデータを読みとり、読み込み権を放棄して、再表示等の処理をおこなう。

6. まとめと今後の課題

以上のように、変更通知機能をデータリスト方式に加える事により、複数のＡＰで１つのデータを共有する環境等において、１ＡＰの加えた変更を速やかに他の全てのＡＰに知らせる事ができる。

今後はこれまでに検討した方式のインプリメントを行い、方式上の問題点の洗い出しを行うと同時に、性能面での評価を進める予定である。

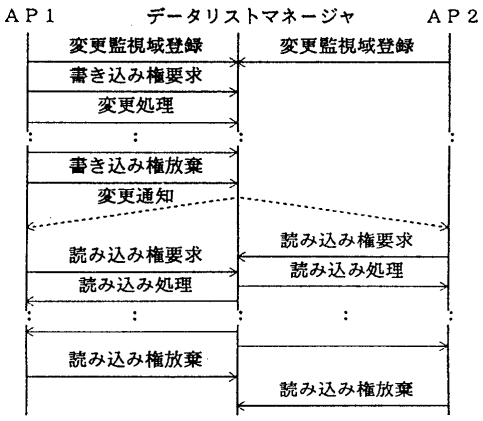

図２ 変更通知機能のタイミングチャート