データベース・システム 82－3
（1991. 3. 11）

ハードウェアログ管理による
並行処理制御および回復処理の高速化

高倉弘喜* 上林彌彦+*

*九州大学工学部 +京都大学工学部

データベース操作の記録であるログは、これまでソフトウェア制御のCPUの処理を経て管理されていた。二次記憶ベースのシステムにおいても、1トランザクションの実行時間の約半分がログ管理に費やされるシステムさえ存在する。この方式をほとんど二次記憶のI/Oのない主記憶データベースで行うとログ管理の占める時間がさらに大きくなってしまう。主記憶データベースでは、ログ管理のオーバヘッドを減らすシステムが必要となる。本稿では、ハードウェア制御により主記憶アクセスを直接監視し、データ処理のバックグランドで物理ログを管理するログ管理システムについて述べる。また、この物理ログを論理ログに変換する方式の概要について述べる。

A Hardware Log Management for Efficient Concurrency Control and Recovery

Hiroki Takakura*
*Faculty of Engneering, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812, Japan

Yahiko Kambayashi+*
+Faculty of Engneering, Kyoto University
1 Yoshidahonmachi, Sakyou-ku, kyoto 606, Japan

The log, which are records of database operations, are managed by software controlled CPU. Even on a secondary storage based system, there exists a system on which the time of log managements takes a half of total execution time of a transaction. When we use a main memory database system, overhead to take log become higher, since no I/O operation for secondary storages are required.

In this paper, we discuss a hardware log management system which snoops the bus and manages physical log in the backgroud of data processing. We also disscuss briefly a method to convert physical log into logcal log.

1. まえがき

現在データベースシステムの高速化のため、様々な手法が比較・検討されている。これらの例として、ファイル構造、質問処理の効率化アルゴリズム、並行処理制御・主記憶データベースシステムの実現、さらにこれらの制御をハードウェア化したデータベースマシン等がよく知られている。

システムの総合的な高速化のためには、このような処理の高速化と共に、障害対策の処理、及び、障害後の回復処理をいかに効率よく行うかもきわめて重要である。なぜなら、通常は高速に動作するシステムでも、一旦障害が発生すると回復に数時間から数日もかかったり、回復不能になるようではデータベースシステムの高速性を十分に生かせるシステムとは言えない。また、信頼性の面から考えて、そのようなシステムは現実には利用できない。また、障害対策のオーバヘッドがシステムの能力を大きく低下させることがあってはならない。例えば、金融・証券のシステムでは数分以内に回復できることが条件であるため、短周期に回復データをとる必要がある。さらにこのためのオーバヘッドが小さいことが要求される。

この問題は主記憶データベースシステムでは特に重要となる。主記憶上のデータは二次記憶上のデータと比べシステムの異常動作の影響を受けやすい。このため、これまで様々な対策が考えられてきた。これらは主として、ある時刻における主記憶データを二次記憶へ書き戻すといったものである。しかし、この方式ではそのある時刻の状態までの回復を行うだけで、主記憶データを障害発生直前の状態には戻すことができない。従って、ある時刻の状態以降は、ログを用いることによりできる限り障害発生直前の状態に戻すことが必要となる。このログとはデータベースに対する処理の記録のことである。さらに、金融システムなどでは1日の取引結果を集計するためにもログを用いる。

また、データ処理に主記憶と二次記憶を使うシステムでは効率向上のため一般に並行処理制御が行われる。並行処理は一般に二相施錠によって制御されていることが多い。しかし、二相施錠では途中で中止されたトランザクションのロールバック（後退復帰）が起こることがある。ロールバック作業にはログを必要とする。

現在のシステムでは、ソフトウェア制御によってデータ操作を論理ログの形式で記録している。ソフトウェア制御ではログ生成のオーバヘッドが大きくなり、システムの動作時間のかなりの部分を占めることになる。あるトランザクション処理システムでは、1つのトランザクションの実行時間の半分がログのために費やされるような例がある。このためデータベースマシンに見られるようにデータベース処理の効率だけを上げても、システム全体の効率はあまり上がらないことになる。

そこで本研究では、このログをハードウェア制御で得る方式を検討している。ログの生成をハードウェア制御で行うとオーバヘッドを小さくできる。原理としては、CPUのつながれたバスを直接監視してバスに流れるアドレスとデータを読み出すことにより物理ログを得ることが可能となる。また同時に、この方式によると、ログのためオーバヘッドがかなり小さくなり、システム効率の改善が期待できる。

しかし、本稿のシステムの物理ログは更新されるデータのアドレスとデータをそのまま記録するため、具体的にどのようなデータ操作を行ったかが判らなくなる。そこで、CPUを追加することにより物理ログを論理ログに変換する方式を現在検討中である。

本稿では、このハードウェア制御によるログシステムに関して、2章で本稿のシステムの基本的事項について、3章でバックアップシステムの概要について、4章でログ管理システムのハードウェア構成について、5章で回復処理の操作について、6章で物理ログを論理ログに変換する方式の概略について述べる。

2. 基本的事項

この章では本稿のシステムを理解する上で必要な基本的事項について述べる。

・ログ

ログとはデータベースに対して行った処理の記録であり、物理ログと論理ログに大別できる[Bern]。物理ログは、「データxに数値vを書き込んだ」といった実際に書き込まれた数値を記録する。これに対して、論

理ログは、「関係Rに組t1を追加した」とか「関係R1とR2の結合を行った」といったデータ操作を記録する。論理ログは複数の物理ログをまとめたものと考えられるので、ログの大きさとしては論理ログの方が小さくてすむし、データ操作も分かりやすい。しかし、論理ログは後退復帰・回復処理の際に、必ず物理的な操作に変換する必要があり、物理ログに比べ処理に時間がかかる。

・主記憶データベース

本来、主記憶データベースシステムとは全データが主記憶に存在し、データの更新操作を主記憶にのみ反映するシステムを意味する。従って、通常の動作では、ディスクなどの二次記憶を必要としない。現在では、扱うデータによっては、このようなシステムも構成可能であるが、実際には巨大な二次記憶から必要なデータのみを主記憶に写して、その後の更新操作を主記憶にのみ反映し、最後に不要になったら二次記憶へ書き戻す方式が一般的である。

本稿のシステムでは、この方式による並行処理や回復処理などハードウェア制御による高速化を研究中である。

・バックアップ

データの更新操作を直ちに主記憶と二次記憶に反映する二次記憶ベースのシステムと比べ、主記憶データベースは障害に弱い。また、主記憶データベースの高速性を損なわないような、高速かつ効率的なバックアップを必要とする。詳細は3章で述べる。

・並行処理制御

並行処理制御は、1つのトランザクションが時間のかかる二次記憶のI/Oを行っている間に、他のトランザクションの処理を行うことで、全体としての処理の高速化を図る方式である。本来の主記憶データベースでは、全データは主記憶に存在するため、I/Oが存在せず、並行処理制御は不要になるという意見がある[Eich]。しかし、次のような問題がある。1つはあるトランザクションが何らかの理由で停止すると処理を続行できなくなる。また、処理中に順次回復処理をするにも並行処理が必要である。もう1つは利用者によるデータの入力速度やシステムに対する応答速度がシステムの処理速度に比べて遙かに遅く、システムの効率を低下させることになる。さらに、主記憶データベースではこの傾向はさらに著しくなるため、本稿では並行処理制御は必要であると仮定して議論を進める。

・二相施錠方式

並行処理制御でよく用いられる二相施錠方式では、施錠段階で必要なデータの施錠を必要に応じて順次行い、解錠段階でデータの解錠を行う。施錠段階での解錠、解錠段階での施錠はできない。本稿のシステムではトランザクションの確定時にまとめて解錠を行う狭義二相施錠方式を利用する。

・障害に関する仮定

本稿では、主記憶で起こる障害として何等かの原因で主記憶データを失うような事態のみを考え、それ以外は考慮しない。また、ディスククラッシュによるデータベースの破壊は他の方法で修復できるものとして、ここでは述べない。

3. バックアップシステム

主記憶データベースでは主記憶のみで更新操作が行われ二次記憶には最新のデータが存在しないため、障害によって主記憶上のデータが消えてしまう恐れがある。ここではこの対策として、効率的なバックアップシステムの概要について述べる。

3.1 二次記憶ベースシステムのバックアップ法

二次記憶ベースのシステムで起こり得る障害としては、

1) システムの異常動作によるバッファなどの周辺回路の破壊
2) ソフトウェアのバグやシステムの異常動作が原因の無効なデータが書き込まれることよるデータベースの破壊

3)ディスククラッシュによるデータベースの破壊

などが考えられる。最も簡単な対策としては、利用者が少ない夜間などにシステムを止めて二次記憶のデータベースを磁気テープなどにバックアップしていた。回復処理では、1)の障害が発生した場合は、周辺回路を修理する。2)の場合は、ログなどを用いて無効なデータの書き込みの取り消しを行うか、破壊が深刻な場合は次に示す3)と同じような処理を行なう。3)の場合は、ディスクを取り替えるなどの原因を取り除いた後、バックアップデータを磁気テープから二次記憶に書き戻して、ログによる処理の再実行を行なって、データベースの状態を障害発生直前まで回復させる。このため、1日に1回しかバックアップを行わなければ最悪の場合、回復処理に長い時間かかることになる。しかし、3)のような事態はそんなに起こるわけではない。

しかし、現実には24時間の常時運転を行なうシステムも存在する。このようなシステムでは、この方式は利用できない。この場合、処理を中断させずにバックアップを行なうなんらかの方式が必要となる。

3.2 主記憶データベースシステムのバックアップ法

主記憶データベースステムでの障害としては、

1)システムの異常動作によるメモリ本体や周辺回路の破壊
2)ソフトウェアのバグやシステムの異常動作が原因の無効なデータが書き込まれることよるデータベースの破壊
3)停電などによる主記憶上のデータの消失

などが考えられる。主記憶データベースの障害対策としては、バッテリバックアップが用いられ、これだけで十分であると考えられていることが多い。これは、主記憶で起こる障害は3)の障害であり、それ以外は滅多に起こらないと考えられているからである。滅多にないからといってなんの対策も取らなければ、1)の場合、メモリ本体が破壊されてしまうとデータは失われるが、メモリが無事であったとしても、バッファなどの周辺回路が破壊されただけでも修理でデータが破壊される恐れがある。2)の場合には、無効なデータをバッテリで保持していても意味がない。

このため、二次記憶ベースの場合と同様に主記憶データベースでもログの記録と主記憶上のデータベースのバックアップが必要となる。ログの記録については、4章で述べる。

主記憶データベースの利点はその高速性にある。通常は高速処理ができるとしても、一度障害が発生すると回復に数時間から数日もかかるようでは、全体の効率が良いとは言えない。高速な回復処理能力のためには、バックアップの周期を二次記憶ベースのシステムの場合より短くして回復処理でのトランザクションの再実行時間を短くする必要がある。さらに、バックアップのたびにシステムを止めるのでは主記憶データベースの高速性が生かせない。従って、主記憶データベースのバックアップは、短周期でかつシステムに及ぼす影響が少ないものでなければならない。

著者らは、バックアップによる影響をできるだけ少なくしたバックアップ方式について研究した[Kamb]。この方式は、図1のように主記憶(DS)が2つの入出力ポートを持ち、データベース処理を行うブロックとバックアップ・回復処理を行うブロックがそれぞれにつながれている。両方のブロックは基本的に独立非同期に主記憶にアクセスを行なう。ここでは、この主記憶をデュアルポートDRAMなどを用いることで実現しようと考えている。デュアルポートDRAMの特性から、バックアップによる処理の中断はなく、バックアップ中に主記憶へのアクセスが若干遅くなるだけですむ。さらに、バックアップ・回復処理はハードウェア制御により高

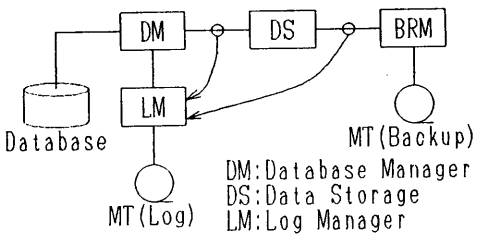


図1 本稿のシステムのハードウェア構成

速に実行できる。

また、図1のようにログ管理を行うブロック(LM)はDM-DS間とDS-BRM間のそれぞれのパスを監視している。

4. ハードウェア制御によるログ管理

この章ではハードウェア制御によるログ管理について述べる。

4. 1 ログ管理のオーバヘッド

ログはデータ更新の記録であり、最悪の場合でもログが残っていればデータベースを修復することができる。このため、最も基本的なログの記録法はトランザクションの確定時にログを二次記憶に記録し、ログの記録が終わってからデータを解錠し、トランザクションを終了させるものである。この方式であれば確定したトランザクションのログのみが確実に記録でき、確定しなかったトランザクションのログは記録されない。また、ログの記録が終了する前に障害が発生すると、そのトランザクションは確定しなかったものとして、記録されたログを利用しなければよい。

しかし、この方式ではログを二次記憶に記録している間トランザクションが待たされるため、データ施錠時間が長くなってしまい、トランザクションの並行性が落ちる。これを解決するには、ログの記録が終了する前にデータを解錠して他のトランザクションの読み出しのみを受け付け、記録が終わったらトランザクションを終了させて、他のトランザクションの書き込みを認める方式が考えられている。

また、確定後から終了までのトランザクションの待ち時間を短くするには、データ更新を行うと直ちにログを記録していく方式が考えられている。しかし、ログの記録は、二次記憶へシーケンシャルアクセスを行うことによって高速化を図っている。従って、トランザクションが中止されても一度書き込まれたログを取り消すことは困難であるし、ランダムアクセスを行なってログを取り消した後で、ログの記録を再開することはシステム効率の面からみて望ましくない。

本来データ操作の補助的な性格を持つログ記録のオーバヘッドが、システムに及ぼす影響が大きいという問題は二次記憶ベースのシステムでも見られる。さらに主記憶データベースではログ生成のオーバヘッドがシステム全体のオーバヘッドに占める割合がさらに大きくなる。これまでの方式では、更新操作を論理ログの形式にソフトウェア制御で変換して記録していた。この方式では、ログを生成するのにデータ処理を行なうのと同じCPUによる処理を必要とし、システムの効率が改善できない。

そこで本稿では、ハードウェア制御によってログの生成をトランザクションの更新操作と同時に行うようなシステムを考えた。

4. 2 ログの種類

本稿のシステムではCPUとしてモトローラ製の68000シリーズを用いる。このとき、ログとして次の５つの形式を考えている。

- ログ Lm : [T_{id}, M, Ad, Val1, Val2]
- ログ Lb : [B , Ad]
- ログ Ls : [T_{id}, S]
- ログ Lc : [T_{id}, C]
- ログ Lr : [T_{id}, R]

[データ更新のログ:Lm]

LmはトランザクションIDがT_{id}のトランザクションがアドレスAdのデータをVal1からVal2へ書き変えたことを表している。並行処理制御のロールバックを行なう際に更新前のデータが必要であるため、データの更新を行う前に元のデータの値Val1をなんらかの方法で知る必要がある。またMは書き変えられるデータのバイト長を表している。トランザクションIDはトランザクションの発生順に与えられる。

[バックアップのログ]

バックアップデータは全体が同一時刻の状態で二次記憶に書き込まれることが望ましい。しかし、これを実現するには様々な問題がある[Kamb]。そこで、バックアップデータをいくつかのブロックに分けて、ブロック毎に二次記憶に書き込む。LbはアドレスAdで指定

されるデータブロックがこの時刻にバックアップされたことを表している。例えば、[B,07FFFF]はアドレスの07FFFFから0FFFFFまでのデータがバックアップされたことを表している。またBにはある予約された数値（例えば、0000）が割り当てられる。

[主記憶データを入れ換える場合の処理]

主記憶に全てのデータが入らない場合、使用頻度の高いものを入れておけば二次記憶へのアクセス時間が減少する。しかし、データの使用頻度は時間とともに変わるため、主記憶のデータを二次記憶のデータと入れ換える場合が考えられる。本稿のシステムでは、この操作をある特別なトランザクションがデータ更新を行ったと見なして、Lmと同じ形式で表す。このときT_{id}にはある予約された数値（例えばFFFF）が割り当てられ、従って、0000、FFFF以外の数値（0001～FFFE）が通常のトランザクションに割り当てられる。

[トランザクションの開始・確定]

LsはトランザクションT_{id}がこの時刻に始まったことを表している。LcはトランザクションT_{id}がこの時刻に確定（コミット）したことを表している。LrはトランザクションT_{id}がこの時刻に後退復帰したことを表している。

5. ハードウェアログシステムの構成と動作

4.2節で挙げたログをどのように管理するかを以下に示す。ログの管理には図2に示すようなトランザクション状態メモリを用いる。このメモリは2ビット×（トランザクション数）だけの大きさを持つ。アドレス方向は各トランザクションIDに対応し、各2ビットでトランザクションの状態を示す。00は初期状態を、10はトランザクションの実行中の状態を、11はトランザクションが確定されたことを、01はトランザクションが後退復帰されたことを示している。

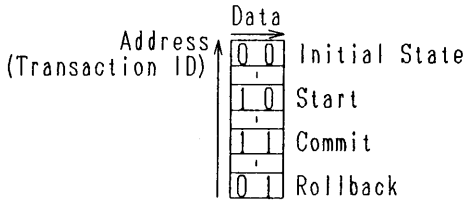


図2　トランザクション状態メモリ

[トランザクションの開始]

いま、あるトランザクションが始まったとする。このとき、図3のようにDMからLMにトランザクションの開始とそのトランザクションIDが通知される。これを受けてLMではトランザクション状態メモリを10に変え、トランザクションの開始時間を示すLsを生成する。

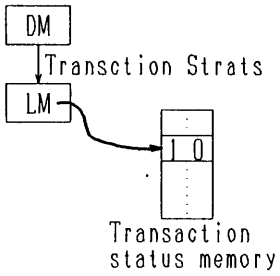
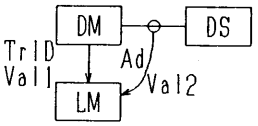


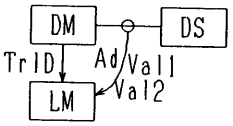
図3　トランザクションの開始

[データの更新]

本稿のシステムでは、Lmを得るにはデータを書き込む前にそのデータの更新前のデータを前もって知っているか、書き込みの直前に読み出す必要がある。前もって知っている場合には、図4(a)のようにデータ更新時にDMからLMへトランザクションIDと更新前のデータを送る。LMはDSへ書き込まれるデータとそのアドレスをバスを監視して読み出す。また、直前に読み出す場合は、図4(b)のようにDMはDSに読み出しアクセスを行う。このとき同時に、DMからLMへそのトランザクションIDを送る。これを受けてLMはDSからDMへ読み出されるデータとそのアドレスを、バスを監視することにより読み出す。その後、DMはDSに対して書き込みアクセスを行う。同時に、バスを監視して、LMはDMからDSへ書き込まれているデータを読み出す。即ち、本稿のシステムでは、DMがリードモディファイライト(RMW)アクセスを行い、LMはそ



(a)データを前もって知っている場合



(b)更新前に読み出す場合

図4　データ更新

のアクセスをバスを監視してログを得る。

また、DMからDSへトランザクションIDを送るには、DMはどのトランザクションがどのデータを更新しようとしているか分からなければならない。ここでは二相施錠を行うので、データをどのトランザクションが施錠しているかを示す施錠表が必要である。本稿のシステムではデータの施錠と解錠をハードウェア制御で行っており、トランザクションIDをアドレスで入力すると、施錠しているデータを出力する施錠メモリを用いている。従って施錠メモリを見ることで、DMはトランザクションIDを知ることができる。

［トランザクションの確定］

このとき、DMからLMにトランザクションの確定とそのトランザクションIDが通知される。これを受けてLMでは図5に示すようにトランザクション状態メモリを11に　変え、Lcを生成する。

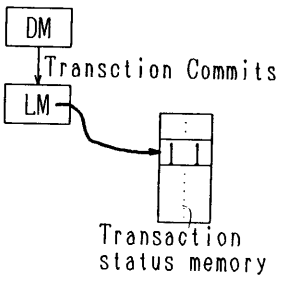


図5　トランザクションの確定

［トランザクションの後退復帰］

4.1節で述べたように、トランザクションが中止されても、既に書き込まれたログを取り消すことは難しい。このため、そのトランザクションが行ったデータ更新が無効であることを示すログを記録する必要がある。このとき、DMからLMにトランザクションの中止とそのトランザクションIDが通知される。これを受けてLMでは図6に示すようにトランザクション状態メモリのを01に変え、Lrを生成する。

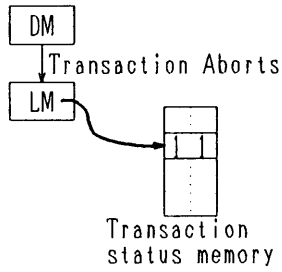


図6　トランザクションの後退復帰

［主記憶データのバックアップ］

バックアップの時間が来ると、BRMがDSに対してアクセスを開始する。図7のようにLMはバスをBRM側の監視して、最初のアクセスが行われると同時にLbを生成する。

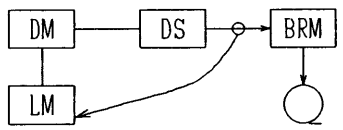


図7　バックアップのログ記録

このようにこの方式では、ログの記録をデータ更新と同時に行える。これは、ログ記録のオーバヘッドを小さくすることができる。

本稿ではログの記録は更新操作のバックグランドで行われ、データ処理に影響を及ぼさない。このためには本稿ではログの記録は更新操作のバックグランドで

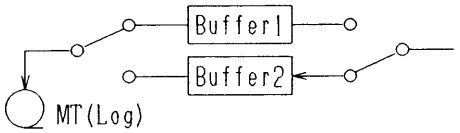


図8　ダブルバッファ

本稿ではログの記録は更新操作のバックグランドで行われ、データ処理に影響を及ぼさない。このためには、ログの記録を直接磁気テープにするのではなく、バッファを通して記録する。このバッファは、一方でログの記録を受付ながら、他方でログを磁気テープに送るような構成が要求される。そこで本稿では、図8のようなダブルバッファを用いる。

6. 並行処理制御

ここでは、確定していないトランザクションのログはまだログバッファに残っているものとして考える。いまあるトランザクションが中止されたとする。この場合、そのトランザクションが行った更新操作を無効にする必要がある。

これまでのシステムでは、論理ログをログバッファから読み出して、これを、データ処理を行なうCPUで解析して更新前のデータを書き戻していた。

これに対して本稿のシステムでは、まず、ログバッファを逆方向に検索し、中止されたトランザクションと同じIDが書き込まれているログLmを探す。Lmが出て来るたびに、元のデータVal1を主記憶に書き戻す。この操作を、そのトランザクションの開始を示すログLsが現れるまで行なう。　全てのデータを元に戻したら、データを解錠する。トランザクションが中止されたことを示すログLrが記録され、また、回復処理ではLrが記録されているトランザクションは無視されるため、ロールバックの操作はログとして記録する必要はない。

このように物理ログを利用しているため、ログを読み出した後は、データ処理も行なうCPUによる処理がいっさいない。従って、後退復帰中も他のトランザクションの処理が続行できる。

7. 回復処理

何等かの原因でデータベースシステムに障害が発生すると、主記憶上のデータは失われてしまっている恐れがある。このため回復処理では、最新の状態にデータを再現する必要がある。本研究のシステムでは、主記憶データはバックアップ用の二次記憶に定期的に保存されている。そこで、初めにこのバックアップデータを主記憶に書き戻す。ただし、バックアップ中に障害が発生することも考えられるので、バックアップデータは最新のものとその1つ前のものの2つを保存しておく。従って、バックアップ中に障害が発生した場合、書き戻すデータは1つ前のものを用いる。こうして主記憶データはある時刻の状態に回復される。しかし、ある時刻であって障害発生直前の状態ではない。また、主記憶データはブロック毎にバックアップされるため、ブロックによってバックアップ時刻が異なる。このため、ログを用いて主記憶ブロックごとの時刻のずれを修正し、さらに主記憶データの状態を障害発生直前に戻す。

8. ハードウェア制御による論理ログの生成

これまでに述べた、ハードウェアログシステムは最もレベルの低い物理ログを記録する。この物理ログではどのようなデータ操作を行ったか判らなくなってしまうため、金融システムのようなログによる集計を必要とするシステムでは、このような物理ログは利用できない。そこで本研究では、バスに流れるデータから論理ログを生成するようなシステムを現在研究中である。

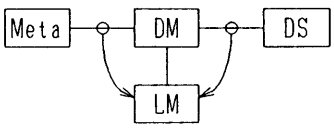


図9　ハードウェア制御による論理ログの生成

このシステムの基本的な動作を図9に示す。データベースシステムでは本来のデータの他に、データが存在するアドレスやデータサイズなどのデータの状態を示すメタデータが存在する。そのため、データベースの処理では、まずメタデータにアクセスして希望のデータのアドレスなどの状態を読み出した後、データのアクセスを行う。また、書き込みを行うことでデータのサイズが変わったりした場合は、さらにメタデータにアクセスして新しい状態を書き込む。

ここでは、このメタデータのアクセスも監視することで論理ログを生成する方式を用いる。メタデータへのアクセスを監視し、データベース操作を解析するCPUをLMに加える。この方式については現在研究中である。

9. まとめ

本稿では、主記憶データベースの効率改善のためのハードウェアログシステムの概要についてのべた。今後の課題は、ハードウェア制御による論理ログを生成するシステムを開発することである。

10. 謝辞

本研究にあたり、有益な御助言をいただいた本学 最所圭三氏、ならびに研究室諸氏に感謝いたします。
尚、本研究は文部省科学研究費試験研究によるものである。

参考文献

[Bern]P.A.Bernstein, V.Hadzilacos, N.Goodman, "Concurrency Control and Recovery in Database Systems," Addison Wesley, 1987.

[Cope]G.Copeland, T.Keller, R.Krishnamurthy, M. Smith, "The Case For Safe RAM," Proceedings of the Fifteenth International Conference on Very Large Data Base, 1989, pp.327-335.

[Eich]M.H. Eich, "Main Memory Database Research Directions," Proc. 6th International Workshop, IWDM'89, 1989, pp.251-268.

[Hagm]R.B.Hagmann, "A Crash Recovery Scheme for a Memory-Resident Database System," IEEE Trans. Comput., C-35, 9, 1986, pp.839-843.

[Son] S.H.Son, "A Recovery Scheme for Database Systems with Large Main Memory," COMPSAC, 1987, pp.422-427.

[Kamb]Y.Kambayashi, H.Takakura : "Realization of Continuously Backed-up RAMs for High-Speed Database Recovery," DASFAA, 1991.