1Q－9

時制永続オブジェクト管理システム POST/C++ における履歴オブジェクトアクセスの高速化手法の検討

鈴木 孝幸 †　北川 博之 ††

† 筑波大学 工学研究科　†† 筑波大学電子・情報工学系

E-mail : {suzu, kitagawa }@dblab.is.tsukuba.ac.jp

1 はじめに

近年、データベース応用分野の高度化に伴い、複雑なデータ構造や手続きを伴うオブジェクトを扱うことが可能なオブジェクトデータベース管理システム (ODBMS) の利用が進みつつある。また、データベースに対する応用要求として、時間情報管理やオブジェクトの更新履歴管理の要求もある。これまで、主に関係データベースを基礎にした多くの研究が時制データベース [4] の研究分野で行なわれてきている。しかし、ODBMS で時間情報を扱うためには、オブジェクトの状態の時間変化を管理するための時制永続オブジェクトモデルが必要である。そこで、当研究グループでは、時制永続オブジェクトモデルを支援する時制永続オブジェクト管理システム POST/C++ [1, 2, 3] の開発を行なっている。[2, 3] では、システムの設計と実装したシステムの基本性能をベンチマークを用いて測定した結果を報告した。その結果、オブジェクトの現在の状態を表す現在オブジェクトへのアクセスに対して、過去のオブジェクトの状態を表す履歴オブジェクトへのアクセスに 3 倍から 10 倍程度のオーバヘッドがかかることが観測された。時制オブジェクト管理システムでは、履歴オブジェクトへのアクセスにもある程度の高速性が要求されるので、本研究では、履歴オブジェクトへのアクセスの高速化の手法について検討する。

2 章では、時制オブジェクト管理システム POST/C++ の履歴オブジェクトへのアクセスの方法の現在の実装と問題点について述べ、3 章では高速化をはかるための方式について述べる。4 章では、まとめと今後の課題を述べる。

2 POST/C++ システム

2.1 POST/C++ の概要と履歴オブジェクトアクセス機構

[1] で述べた時制永続オブジェクトを管理するためのシステム POST/C++ の概要について述べる。時制永続オブジェクトには、オブジェクトの現在の状態を表す**現在オブジェクト**、過去の状態を表す**履歴オブジェクト**とが存在する。各オブジェクトには、その状態が有効であった期間を示す**カレントインターバル**を持っている。また、時制永続オブジェクトへの参照は現在オブジェクトへの参照として解釈される。そのため、履歴オブジェクトへアクセスするためには snapshot() プリミティブを適用しなければならない。POST/ C++ では、ODBMS の実装方法の 1 つとして知られているメモリマップ方式アーキテクチャを採用している。また、現在オブジェクトと履歴オブジェクトは別々の 2 次記憶上に格納している。より詳細については、[1, 2, 3] を参照されたい。

POST/C++ において、履歴オブジェクトにアクセスするためには、時刻を指定して snapshot() プリミティブを対象とするオブジェクトに適用する。すると、指定された時刻でのオブジェクトの状態を表す履歴オブジェクトが主記憶上にマップされ、そのアドレスが返り値となる。一度主記憶上にマップされた履歴オブジェクトは、更新の操作以外は現在オブジェクトと同様の操作が行なえる。

`snapshot(coid, ts)` として、対象とするオブジェクトの主記憶上のアドレス `coid` と時刻 `ts` が指定された場合、システムの内部では、以下のような手順で対象となる履歴オブジェクトが格納された履歴ファイルのページ (**履歴ページ**) が主記憶にマップされる。各オブジェクトは一意の識別子 (OID) がつけられる。POST/C++ では、OID として 2 次記憶上でのアドレスを利用している。

1. `coid` から現在オブジェクトの OID を調べる。
2. 履歴索引を検索して、指定された時刻 `ts` での履歴オブジェクトの OID(`HOID`) を調べる
3. 仮想記憶マッピング表を調べて、その履歴ページがマッピングされているかを調べる。マッピングされていたら 5
4. `HOID` で示されるアドレスを含む履歴ページを主記憶に読み込み、そのページ中の他のオブジェクトへの参照のポインタ書換えをする
5. マップした主記憶上のアドレスを返す

2.2 履歴オブジェクトアクセス機構の問題点

現在の実装による履歴オブジェクトアクセス機構の問題点としては以下の点がある [2, 3]。

1. 履歴オブジェクトを格納する履歴ファイルは、現在の実装では、更新時間順に格納している。そのために、現在オブジェクトを格納している現在ファイルのページ中でのオブジェクトの格納の局所性が、履歴ファイル中では考慮されない。
2. ポインタ書換えをページ単位で行なっているために履歴ページのマッピングでは同ページに格納されている不要な履歴オブジェクトについてもポインタ書換えが行なわれてオーバヘッドになっている。
3. 永続時制オブジェクトの参照の解釈が現在オブジェクトへの参照となるため、ある時刻での履歴オブジェクト間の参照を辿る場合にも snapshot() プリミティブを適用しなければならない。そのためのオーバヘッドが生じている。

A study on access schemes to historical objects in Temporal Persistent Object Management System POST/C++

Takayuki SUZUKI† and Hiroyuki KITAGAWA††

†Doctral Degree Program in Eng., Univ. of Tsukuba

††Institute of Info. Sci. and Elec., Univ. of Tsukuba

4. コミット時に履歴オブジェクトをマップした主記憶のページを解放しているが、このために一度コミットしてしまうと再び履歴オブジェクトをマップし直さなければならないので、オーバヘッドになっている。

3 履歴オブジェクトアクセスの手法の改良

前章での問題点を解決するために以下のような改良をPOST/C++ に加えることにする。

3.1 履歴ファイルの構成法の改良

現在の実装におけるファイル全体での更新順による履歴ファイルへの構成法を、現在ファイルのページ単位で行なうように拡張する。このことによって、現在ファイル上のページ中での現在オブジェクトの位置の局所性が、履歴ページでもある程度保たれることになる。

3.2 オブジェクト単位でのポインタ書換え

履歴ページをマップする場合、不要な履歴オブジェクトのポインタ書換えが行なわれることによってオーバヘッドが生じるので、履歴オブジェクトはオブジェクト単位でポインタ書換えを行なうことにする。

ところで、履歴オブジェクトへのアクセスを考えると、

- snapshot() プリミティブによるアクセス
- iterator による更新履歴アクセス

の2通り以外はないことになる。したがって、snapshot()、iterator でのアクセスですぐにポインタ書換えを行なう履歴オブジェクトと行なわない履歴オブジェクトに区別して、snapshot() プリミティブによりアクセスされた場合と、ページフォルトが起こった場合に、該当オブジェクトのみのポインタ書換えを行なえば良い。ページ内のポインタ書換えを行なうオブジェクトと行なわないオブジェクトの区別は、ページ毎にビットマップを持ち管理する。また、ページフォルトの時には、マップした主記憶のアドレスを利用しているオブジェクトについてもポインタ書換えを行なう。

3.3 履歴オブジェクト中の参照の解釈の拡張

履歴オブジェクトへのアクセスのパタンの中には、ある時刻におけるデータベースの状態を対象としてオブジェクトのナビゲーションをすることがある。その時刻のことをデフォルト時刻と呼ぶ。そこで、履歴オブジェクトでのオブジェクトの参照をカレントインターバルにデフォルト時刻が含まれる履歴オブジェクトへの参照と解釈するように拡張する。デフォルト時刻は、データベースのオープン時に指定することにする。

すると、履歴オブジェクトへのアクセスには、前節の2通りに加えて、

- デフォルト時刻での参照の解釈によるアクセス

が加わることになる。

参照の解釈が行なわれるのは、ポインタ書換えの時であり、参照先のOID とデフォルト時刻により履歴索引を引き、その結果により参照の解釈が行なわれる。履歴オブジェクト単位でポインタ書換えが行なわれるのであるが、初めてその履歴ページが主記憶に読まれる時は、同じページにある、カレントインターバルにデフォルト時刻が入っている履歴オブジェクトと、マップした主記憶のアドレスをすでに利用している履歴オブジェクトのポインタ書換えが行なわれる。前者は、デフォルト時刻での参照の解釈によるアクセスのためである。

すなわち、snapshot() プリミティブによる履歴オブジェクトのアクセスは2章で示した手順の4が以下のような手順になる。

4. HOID で示されるアドレスを含む履歴ページを主記憶に読み込む。そのページ中の全ての履歴オブジェクトの中で、以下の条件が成立するオブジェクトについては、ポインタ書換えを行なう。
   - その履歴オブジェクトのカレントインターバルにデフォルト時刻が入っているオブジェクト
   - 既にマップした主記憶のアドレスを利用しているオブジェクト

   ポインタの書換えは以下の手順で行なわれる
   (a) 他のオブジェクトへの参照はOID を用いているので、このOID とデフォルトの時刻で履歴索引を検索して、デフォルト時刻での履歴オブジェクトの HOID' を調べる
   (b) HOID' のポインタ書換えを行なう

3.4 履歴ページの解放のオプション

コミット時の履歴ページのマッピングした主記憶を解放しないというオプションを用意する。これによって、同じ時刻でのアクセスが続く場合、同じページの複数回のマッピングがなくなる。

4 結論

本研究では、履歴オブジェクトアクセスの手法を改良する方法を提案した。現在の実装に対して本研究で提案した手法を組み込むことによって、履歴オブジェクトアクセスの高速化が期待できる。

今後の課題としては、本手法を実装したシステムにおいて、どの程度性能改善が行なえるか実測し検証することがある。

参考文献

[1] 鈴木, 北川, 林: 時制永続オブジェクト管理システム POST/C++ の設計と実装 , 情報処理学会研究会報告書, 95-DBS-102, 1995.

[2] 鈴木, 林, 北川, 柳川: 時制永続オブジェクト管理システム POST/C++ の構築と性能評価, 日本ソフトウェア科学会 第12回大会予稿集, 1995.

[3] T.Suzuki and H. Kitagawa: *Development and Performance Analysis of a Temporal Persistent Object Store POST/C++, Proc. of 7th ADC Conf.*, 1996.

[4] Tansel, A.U., Clifford, J., Gadia, S., Jajodia, S., Segev,. A. and Snodgrass, R.: *Temporal Databases – Theory, Design, and Implementations* –, The Benjamin/Cummings Publishing,Inc., 1993.