6D－3

# 時制永続オブジェクト管理システム POST/C++ の開発
## — 長大オブジェクト管理機能の実現 —

林 恵将† 北川 博之†† 鈴木 孝幸‡

† 筑波大学 理工学研究科 †† 筑波大学 電子・情報工学系 ‡ 筑波大学 工学研究科

## 1 はじめに

データベース利用の高度化に伴い、データの時間変化の管理を行ないたいという要求が高まっている。データの時間変化の管理については、これまで主に関係データベースをベースとして時制データベース [TCGSS93] の分野で多くの研究が行なわれてきた [HSK95]。

我々の研究グループではより高度な応用を支援する為、オブジェクトデータモデルをベースとした時制永続オブジェクト管理システム POST/C++ の開発を行なってきた。しかし、これまで POST/C++ 中のオブジェクトの長さはページ長未満に制限されており、画像データやテキストデータなどの長大オブジェクトには対応していなかった。

本稿では POST/C++ における長大オブジェクト管理機能の設計と実装について述べる

## 2 POST/C++ におけるオブジェクト

POST/C++ 中でのオブジェクトはその構造から以下の様に分類される:

- **Small OBject (SOB)**
  大きさが1ページ (8K バイト) 未満であり、クラス定義などで規定された内部構造を持つ固定長オブジェクト
- **Large OBject (長大オブジェクト, LOB)**
  大きさが1ページ以上のオブジェクト
  - **Structured Large OBject (SLOB)**
    内部構造を持つ固定長オブジェクト
  - **Binary Large OBject (BLOB)**
    画像データなどの可変長ビット列として扱うことが適当なオブジェクト

以下では、今回新しく追加した長大オブジェクト (SLOB, BLOB) の管理機能について述べる

---

Development of the Large Object Management Scheme in a Temporal Persistent Object Management System POST/C++

Yoshiyuki HAYASHI†, Master's Degree Program in Science and Engineering, Univ. of Tsukuba

Hiroyuki KITAGAWA††, Institute of Information Sciences and Electronics, Univ. of Tsukuba

Takayuki SUZUKI‡, Doctral Degree Program in Engineering, Univ. of Tsukuba

## 3 長大オブジェクトの管理

POST/C++ はメモリマップ方式 [SKW92] を利用したオブジェクト管理システムである。POST/C++ では SOB の管理を以下の様に行なっている*:

- オブジェクトは初めてアクセスされた時点でメモリにマップされる
- オブジェクト内のポインタはマップされた時点で参照先のオブジェクトをマップする領域を予約することによりメモリ上のアドレスに変換される
- 履歴管理はオブジェクト単位で行なわれ、更新後のオブジェクトは現在オブジェクトとして現在ファイルへ、更新前のオブジェクトは履歴オブジェクトとして履歴ファイルへ格納される
- オブジェクトは削除されても Tombstone オブジェクトとして保持される

### 3.1 SLOB

SLOB は構造的には SOB と同種のオブジェクトであると考えて良い。しかし、複数ページにまたがって格納される点で SOB とは異なる。

上記の SOB の管理方式に準じて SLOB の管理を行なう上では、以下の問題点を解決する必要がある:

- SLOB をメモリにマップする上では主記憶上に連続ページを確保する必要がある
- 履歴オブジェクトをそのまま格納すると冗長性が生じる
- オブジェクト削除時にオブジェクト全体を保持すると領域の無駄が生じる

これらを考慮して、SLOB は以下の様に SOB を拡張した管理方法を採ることにする:

- ヘッダと本体を別々の領域に格納する。ヘッダは SOB と同じ形式であるが、本体を格納したページのアドレスの情報を付加する
- 履歴管理はオブジェクト単位で行ない、履歴オブジェクトの本体は更新されたページの Before イメー

---

*詳細については [HSK95] を参照

ジのみを格納する。2次記憶上のページの格納が非連続となるので、そのページ情報を後述のインデックスで管理する

- オブジェクトの削除時はヘッダのみを残し、本体ページの領域は解放する
- SLOBをメモリにマップする上で必要なページ数はハッシュ表により管理する。この表からSLOBへのポインタのswizzle時に必要な連続ページ数を得る

### 3.2 BLOB

BLOBとして例えば画像データが考えられるが、その性質はSOB, SLOBとは以下の様に異なる:

- オブジェクトは更新される度に大きさが変化し得る
- オブジェクトの大きさが数百キロ～数メガバイトの大きさに成ることがあり得る
- オブジェクトに対するアクセスはその一部へのアクセスを繰り返すことで行なわれることが多い
- 本体のイメージはそのまま使用されるのではなく、イメージを解凍して実際のデータを得るといった加工処理が行なわれることがある

BLOBをSOB, SLOBと同様にオブジェクト全体をメモリ上にマップして扱った場合、メモリを必要以上に使用することになり、スワッピングが頻繁に生じたり同時に使用できるオブジェクト数が制限される等の問題が生じる可能性が考えられる。これらの理由から、BLOBは利用者の指定に応じて処理に必要な部分のみをマップして扱うことにする。

この機能を実現する為に、システムはBLOBに対して以下の様なインタフェースを用意する:

- map() : BLOBの指定されたページ数をメモリ上にマッピングする。マップされた部分については、応用プログラムから直接アクセス可能である
- remap() : map()を行なった後にBLOBの別の部分を再マッピングする
- resize() : BLOBの大きさを変更する†

利用者の指定によっては、BLOB全体をマップすることも可能である

BLOBの管理は、SLOBの管理方法にこれらのインタフェースを実現する為の機能を以下の様に追加した方法とする:

- ヘッダと本体を別々の領域に格納する。本体は可変長であり2次記憶上では非連続領域となるので、そのページ情報を後述のインデックスにより管理する
- 上記のインタフェースを実行する上で必要な情報であるメモリ上でBLOBが使用するページ数と現在マッピングされているオブジェクトの開始アドレスはハッシュ表により管理する

### 3.3 ページ情報管理インデックス

本インデックスは、2次記憶上で連続領域に格納されていないLOB(SLOBの履歴オブジェクトとBLOB)のページ情報を管理する。本インデックスはOID、時刻、ページ位置からファイル上のアドレスを得ることができる。インタフェースは以下の様になる:

- insert() : OID、時刻、ページ位置、アドレスからなるエントリを1つ挿入する
- retrieve() : OID、時刻、ページ位置からそのページが格納されているファイルのアドレスを得る。SLOBは履歴ファイル、BLOBは現在・履歴ファイルのアドレスが得られる

インデックスは1データベースに1つ用意される。オブジェクトを再現したい場合、オブジェクトのヘッダからページ数を計算して、そのページ数分だけインデックスを参照して得られたページをファイルから読み出してマップすることにより再現できる

## 4 まとめ

以上、POST/C++への長大オブジェクトの管理機能の実現方法について述べた。現在、本システムは実装作業を行なっている。

今後の課題として、長大オブジェクト処理に関する性能測定を行なうこと、履歴オブジェクトの格納方法の改良(本体の圧縮を行なう etc.)等が挙げられる。

## 参考文献

[TCGSS93] A.U.Tansel, J.Clifford, S.Gadia, A.Segev and R.Snodgrass : *Temporal Databases – Theory, Design, and Implementations* –, The Benjamin/Cummings Publishing, 1993.

[HSK95] 林 恵将, 鈴木 孝幸, 北川 博之 : 時制オブジェクト管理システムにおける履歴管理機構の設計と実装, 情報処理学会 第50回全国大会, 1995.

[SKW92] V.Singhal, S.V.Kakkad, and P.R.Wilson : Texas: An Efficient, Portable Persistent Store, *Proc. 5th Int'l. Workshop on Persistent Object Systems*, Sep 1992.

†map()やremap()による大きさの変更はできない