データベースオペレーティングシステム OPT-R のメモリ管理方式†

大久保 英嗣†† 津田 孝夫††

本論文では，データベースオペレーティングシステム OPT-R の主要な構成要素の一つであるメモリ管理の処理方式について述べる．OPT-R におけるメモリ管理の特徴としては，単一レベル記憶とソフトウェアによるファイルのセグメンテーションがあげられる．これら二つの特徴を実現することによって，ユーザから主メモリおよび2次記憶といったメモリ階層の管理を解放し，システムの移行性も高めている．

1. はじめに

データベース等の大規模ソフトウェアは，大量のメモリを必要とし，大型計算機上でのみ効率的に実現されうるアプリケーションの例であろう．しかし，データベースを初めとして，グラフィックス，コンパイラ，統計計算等のアプリケーションが，主メモリ容量の限られている小型計算機上で実現される傾向が強くなってきている．通常，これらアプリケーションを実現する場合は，オーバレイ手法やそのアプリケーションに固有の方法が採用されている．したがって，それらの手法を使用することによるソフトウェアの移植性が問題となっている．さらに，大型の計算機では，仮想メモリを管理するハードウェアが標準的に装備されており，しかもそれらを円滑に運用するための研究が長年に渡って研究されている（例えば文献1）参照）．しかし，小型計算機では，仮想メモリを実現するハードウェアは標準的には付加されておらず，ハードウェアの違いによるオペレーティングシステム（以下 OS と記す）自体のアーキテクチャの変更も大きな問題となっている．

以上のような背景から，われわれは，異なる計算機上で一般に通用するソフトウェアによる仮想メモリ管理技法を開発した．本論文で提案する方式の特徴は，従来のソフトウェアによるセグメンテーション方式[9]を基礎として，それを，プログラムおよびプログラムで使用する局所的データの範囲から，データベース処理にまで拡張した点であるといえよう．ここでいうデータベース処理とは，データの共有に伴う一貫性制御や回復管理をも含めた処理を意味している．すなわち，本方式は，ハードウェアでの管理では非常に困難である選択的な2次記憶への書込み（その必要性は文献2）で述べられている）が，柔軟に行えるという利点をもっている．以下，本論文では，OPT-R プロジェクト[12),13)]の一環として開発したソフトウェアによるメモリ管理方式について述べる．

† The Memory Management Technique in the Database Operating Systems OPT-R by EIJI OKUBO and TAKAO TSUDA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学科

2. 設計思想

データベースは，大規模なるがゆえに大型計算機上で実現されている．とくに，仮想メモリ方式の OS 上にデータベース管理システム（以下 DBMS と記す）を構築する場合，データベース用のバッファは仮想空間上に置かれページングの対象となる．したがって，主メモリとバッファ間のページング処理（仮想メモリページング）とデータベースをバッファに読み込むための入出力処理（バッファページング）とによって，2次記憶から1ブロックをプログラム領域へ転送するためのオーバヘッドは数千命令にもなるといわれている[2)]．SQL/DS[3)]，INGRES[4)]等を含めて，多くの DBMS は，このオーバヘッドを小さくするために，バッファをユーザ空間上に確保し，DBMS 自体がそれを管理するという困難に陥っている．

上記二つのページング処理は，double paging と呼ばれ，数多くの解決がなされている[5)~8)]．しかし，それらの解析は，ページング処理のための置換えアルゴリズムに重点を置いており，システムの効率を向上させるための本質的解析にはなっていない．従来の OS は，仮想メモリ内のプログラムに関してはメモリ管理を使用し，プログラムでデータをアクセスするときにはデータ管理を使用している．このようなデータとプログラムの2元管理が，そこでの本質的な問題であると思われる．この2元管理は，おもに性能上の理由に

よるものであるが，その結果，データがプログラムに依存してアクセスされることになるといった弊害が生ずる．プログラム内の局所的なデータベース，すなわち，バッファ，ユーザ宣言の定数，作業領域的配列がプログラムに依存するのは当然であるが，共有可能なすべてのデータは，プログラムに依存するものではないといえる．さらに，プログラムおよびデータの区別が明示的になされている場合でも2次記憶上に共存して格納されており，その意味では同等のレベルにある．

以上の考察から，われわれは，プログラムおよびデータを OS の統一化されたアクセスルーチンによって一元的に管理することとした．これは，IBM の System/38 等で実現されている単一レベル記憶 (Single Level Storage あるいは One-Level Store) の思想[10]に基づくものである．そこでは，プログラムおよびデータはともにページングの対象となり，とくにデータに関しては，従来のデータ管理の入出力に代わって，ページング機構で入出力が可能となり，単純な管理方式が実現できる．しかし，System/38 のようなハードウェアによる仮想マッピング法は，ハードウェアの制限による他システムの移行性が問題となっている．さらに，システムクラッシュなどの障害に対しては非常に弱いという欠点がある．したがって，われわれは，使用する計算機からはなるべく独立したソフトウェアアーキテクチャを実現するために，ソフトウェアによって上記管理を行うこととした．これは，Software Segmented Virtual Memory と呼ばれている[9]．本手法は，データを格納するファイルをすべて仮想空間へ結合し，実空間と仮想空間の間のマッピングをセグメントを単位に行うものである．しかし，ファイル容量が大きくなると，そのためのページテーブルの容量も大きくなり，ページテーブル自体を主メモリに常駐させることが困難になる．したがって，この問題を解決するために，ページテーブルを二つに分類し管理することにした（これは，Stonebraker が文献2）で述べている問題の解になっている）．それは，主メモリのページ対応に一つのエントリをもつ内部ページ管理テーブルと，主メモリ上に存在するセグメント対応に一つのエントリをもつ外部ページ管理テーブルである（次章以降で詳述する）．外部ページ管理テーブルは，文献9）のセグメントマップとフレームテーブルの二つを合わせたものであると考えることができる．文献9）では，フレームテーブルはすべて主メモリ上にあることを前提にしているので，テーブル自体のメモリオーバヘッドは無視できないと考えられる．しかし OPT-R では，外部ページ管理テーブルのエントリ数を制限しており，この問題は存在しない．すなわち，テーブルのエントリに空きがない場合は，当該セグメントを要求しているタスクは待ち状態に入り，空きができた時点で再びそのタスクがスケジュールされるようになっている．したがって，外部ページ管理テーブルの限られたエントリ数で2次記憶上のすべてのセグメントが管理可能になる．さらに，将来的には，この外部ページ管理テーブルによって，システム内の処理多重度の管理を行うことも考えている．

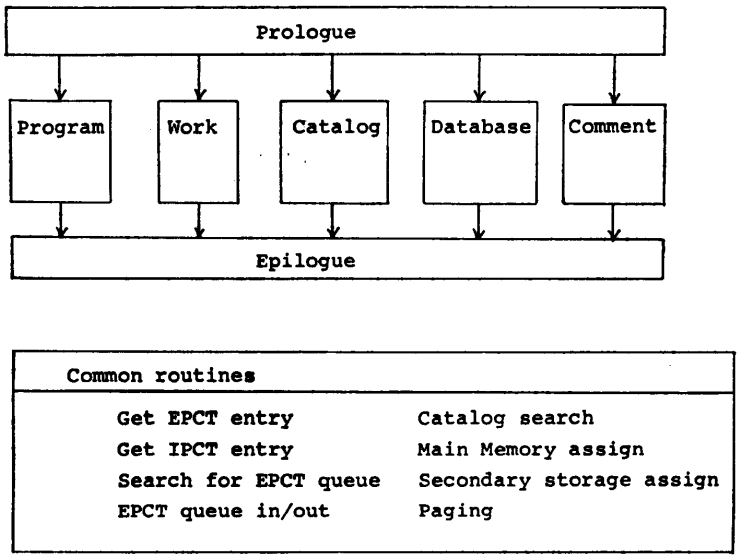

図 1 メモリ管理のモジュール構成
Fig. 1 Module configuration of the memory management.

3. メモリ管理モジュールの構成

OPT-R では，主メモリ領域をシステムパーティションとユーザパーティションの二つに分けて管理している．メモリ管理は，システムパーティション内に常駐し，各セグメントのユーザパーティションへのローディングを制御している．OPT-R におけるセグメントとしては，以下の5種類がある（詳細は，文献 12) 参照)．

（1） プログラム(ユーザプログラム，UP)

（2） 作業領域

（3） カタログ

（4） データベース (PDF, BMF, CTF)

（5） コメント（関係，属性，ドメインに付加）

メモリ管理は，これらセグメントの種類に対応した五つの処理ルーチンと，それら処理ルーチンで使用する共通ルーチン群よりなる（**図1**に構成を示す）．共通ルーチン群は，メモリ管理に必要なテーブルの確保および解放を行うもので，各ルーチンは臨界領域として構成されている．

OPT-R の各モジュールは，セグメントを要求する場合，**図2**に示すパラメタリストに情報を設定し，以下に示す二つのマクロを用いてメモリ管理を呼び出す．

（1） セグメントの要求 (Get SEGment)

メモリ管理は，要求されたセグメントを主メモリ上にローディングし，その先頭アドレスを呼び出し元へ返す．

（2） セグメントの解放 (Free SEGment)

GSEG マクロで要求したセグメントに関する処理が終了したことをメモリ管理へ知らせる．この時点で，メモリ管理は当該セグメントの主メモリページを解放する．

OPT-R では，現在，エンドユーザ言語のみをサポートしており，ユーザが直接メモリ管理を呼び出すことはできないが，将来的にはホスト言語型のデータベース操作言語をサポートする予定であり，そのなかから呼び出すことは可能である．

4. メモリ階層の管理

本章では，ページング処理のためのメモリ置換えアルゴリズム等，主メモリおよび2次記憶間のメモリ階層の管理方式について述べる．

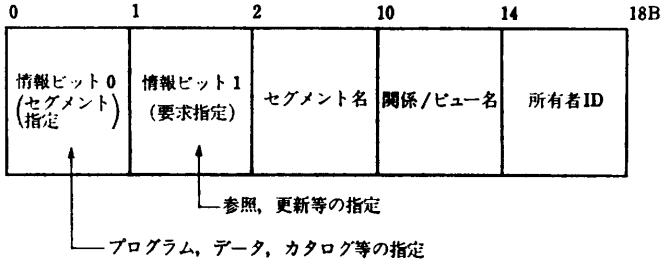

図 2 パラメタリストの形式
Fig. 2 Format of parameter list.

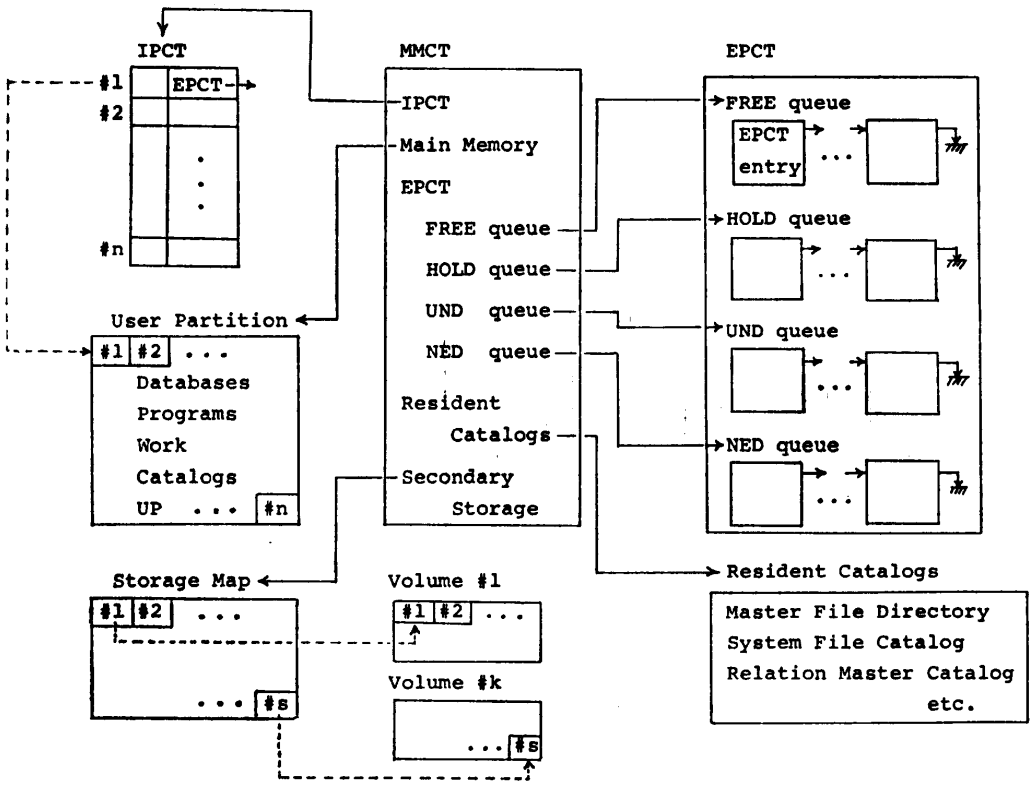

図 3 メモリ管理のためのシステムテーブルの関連
Fig. 3 Relationships of the system tables for memory management.

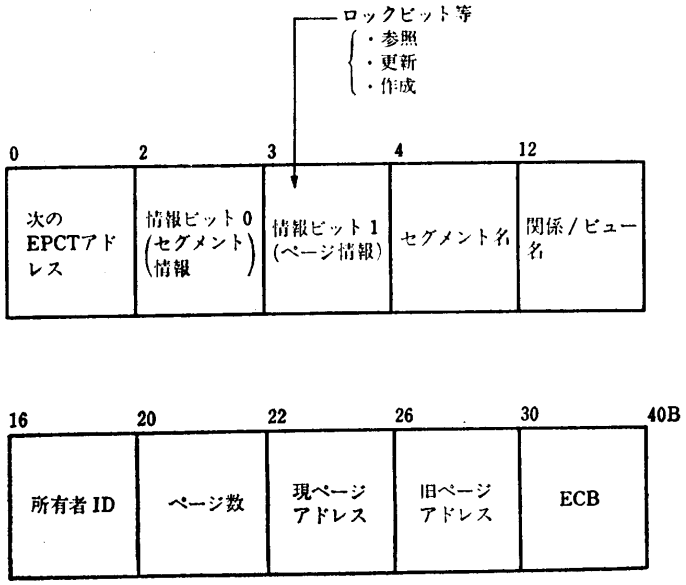

図 4 EPCT エントリの形式
Fig. 4 Format of EPCT entry.

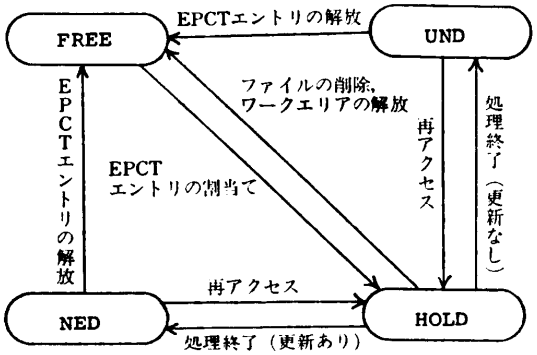

図 5 EPCT キューの状態遷移
Fig. 5 State transition of EPCT queues.

メモリ管理は，以下の四つのテーブルを用いてセグメントのローディングを制御している（**図 3** 参照）．

（1） 主メモリ制御テーブル (MMCT)

主メモリおよび2次記憶の管理に必要なテーブル類のアドレス，さらに各種カタログに関する情報を保持しており，各テーブルへの参照は，このテーブルを経由して行われる．

（2） 内部ページ管理テーブル (IPCT)

ユーザパーティションをページ（512バイト/ページ）単位に分割して，そのページ数分のエントリをもつテーブルであり，主メモリの使用状況を管理するためのものである．

（3） 外部ページ管理テーブル (EPCT)

セグメント要求に対応して一つ割り当てられ，各セグメントに関する情報を保持する．

（4） ストレージマップ (SMAP)

OPT-R では，2次記憶媒体をすべて一つのアドレス空間とみなし，逐次的な番号 (PBN) によってその空間をアドレス付けしている．本テーブルは，この空間の使用状況をビットのオン-オフで表現するもので，各ビットは PBN に対応している．

セグメントが要求されると，メモリ管理は以下の段階を経て主メモリに当該セグメントをローディングし，その先頭アドレスを要求元へ返す．

（1） 要求されたセグメントに対して EPCT を一つ割り当て，セグメントに関する各種情報を設定する．

（2） セグメントのページ数分の連続した主メモリ領域を捜し，IPCT の各種情報を設定する．空き領域がなければ，ページアウトの処理が行われる．

（3） 割り当てた主メモリに要求されたセグメントをページインする．

以下，本章では，これらについて具体的なアルゴリズムを述べていくこととする．

4.1 EPCT の管理

EPCT は，1エントリ 40 バイトからなりセグメント対応に一つ作られる（**図 4** に EPCT の構成を示す）．EPCT は，以下に示す四つのキューによって管理される．

（1） FREE：未使用
（2） HOLD：使用中
（3） UND：処理終了，ページアウト不要
（4） NED：処理終了，ページアウト要

処理中のセグメントに関する EPCT は，すべて HOLD キューにつながれ，処理が終了した時点で UND あるいは NED キューにつなぎかえられる．以後同一のセグメントに関する処理要求があると，UND あるいは NED キューを再び HOLD キューにつなぎ，処理を行うことになる（EPCT の各キューの状態遷移を**図 5** に示す）．

EPCT を2次記憶上のセグメントすべてに対応して用意すると，その領域のオーバヘッドは大きく，主メモリに常駐させることが困難になる．したがって，OPT-R では EPCT のエントリ数を制限している．それ以上の数のセグメントが同時に要求される場合に，EPCT を使用するタスクのキューイング操作に

よって対処している．EPCT は，以下に示す優先順位によって割り当てられる．

（1） FREE
（2） NED，現ページが2次記憶上にある
（3） UND
（4） NED，現ページが主メモリ上にある

すべての EPCT が HOLD キューにある場合は，セグメントを要求しているタスクを待ち状態にする．上記四つの状態のうち，（4）の場合は，すでに変更されたセグメントが主メモリ上にあるので，当該セグメントを2次記憶へページアウトした後，EPCT を要求セグメントに割り当てなければならないので優先順位を低くしてある．

4.2 主メモリの管理（IPCT の管理）

IPCT は，1エントリ6バイトからなり，ユーザパーティションのページ対応に一つ存在する．IPCT は，**図6**に示すように，ページ情報部と EPCT アドレス部から構成されている．メモリ管理は，ページ情報部のビット構成により，当該ページの使用状況を把握する．当該ページが使用中の場合は，EPCT アドレス部によって当該ページにローディングされているセグメントの EPCT を得て，その情報を調べることができる．

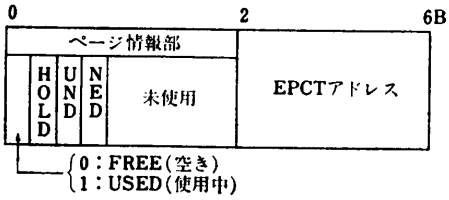

図6 IPCT エントリの形式
Fig. 6 Format of IPCT entry.

主メモリ容量にはおのずと制限があり，新たなセグメントの要求に答えうるに十分な主メモリページが存在しない場合，主メモリ上の未使用セグメントのなかから要求されたセグメントのページ数分のページをページアウトし，空ページを確保する必要がある．OPT-R のメモリ管理では，主メモリの置換えの優先順位を以下のようにして，ソフトウェアによってページング機構を実現している．

（1） FREE
（2） FREE, UND
（3） FREE, UND, NED

上記三つの条件のいずれかに合格した連続したページ集合が存在しない場合は，当該タスクは待ち状態になる．すなわち，HOLD 状態のページ集合の処理が終了し，それらのページが UND あるいは NED 状態になるのを待つことになる．NED キューにあるページは更新されているので，当該ページを解放する場合には，そのページの内容を2次記憶へページアウトしなければならない．これらの処理の流れの例を**図7**に示す（図7では主メモリを5ページとしてある）．

以上のように，OPT-R のメモリ管理では，処理の終了したセグメントのページ集合を置換えの対象としているが，更新されたセグメントに関するページ置換えの優先順位を低くすることにより，2次記憶への入

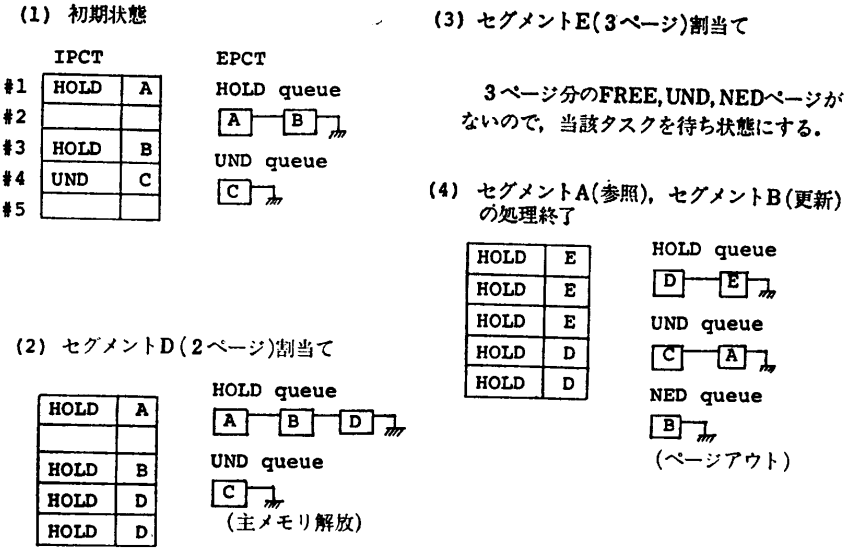

図7 主メモリページ割当て処理の例
Fig. 7 Example for the processings of main memory page assign.

出力がなるべく少なくなるように配慮している．

4.3 2次記憶の管理 (SMAP の管理)

OPT-R におけるプログラムおよびデータは，統一してアドレス付けされた2次記憶上格納にされている．すなわち，2次記憶上のセグメントは，記憶媒体を区別せず，物理ブロック番号 (PBN) と呼ばれる逐次的な番号によってアドレス付けされている．各 PBN は，セグメントに関するカタログに格納されており，メモリ管理はセグメントの要求があると，この PBN を各記憶媒体の物理アドレスへ変換して当該セグメントをアクセスする．

2次記憶の使用状況の管理は，ビットマップ形式のテーブルである SMAP をもとに行われる．SMAP の各ビットはページ（すなわちブロック）に対応しており，'1' が使用中を '0' が未使用を表している．ページアウト等2次記憶への書込みが必要になると，メモリ管理は SMAP のビット列を走査して連続した空き領域を捜す．空き領域が見つかると SMAP 内のエントリ番号（すなわち PBN）を得て，実際の物理アドレスへ変換し書込みを行う．

本管理法の利点としては，2次記憶媒体の変更によるソフトウェアの変更が，物理ブロック番号と記憶媒体の物理アドレス間の変換法を変更することに吸収できることがあげられる．

5. セグメントの識別と共用

5.1 セグメントの識別

前章で述べたように，メモリ管理では，アクセスすべきセグメント対応に EPCT を割り当てる．EPCT を割り当てられたセグメントは主メモリにローディングされ，処理終了後も UND あるいは NED キューにつながれる．すなわち，当該 EPCT が他セグメントの EPCT として使用されることになるまで主メモリ上に置かれる．したがって，メモリ管理は，セグメント要求を受け付けると，当該セグメントの EPCT がすでに HOLD, UND あるいは NED キュー中に存在しているか否かを調べる必要がある．要求されたセグメントの検索は，図2に示したパラメタリストと図4に示した EPCT の情報ビット，対象セグメント名，所有者 ID の各項目の比較により行われる（データベースの場合はさらに，対象セグメント所属関係／ビュー名の比較も加わる）．この検索は，HOLD, UND, NED の順に行われる．

要求されたセグメントの EPCT が HOLD キューに存在する場合，当該セグメントはすでに処理中であるから共用可能か否かを調べる（セグメントの共用については次節で述べる）．共用可能でない場合は，要求元タスクは待ち状態になる．また，UND あるいは NED キューに存在する場合は，そのエントリを HOLD キューに再びつなぎかえ，当該 EPCT 内の現ページアドレス（主メモリ上のアドレスあるいは PBN）を用いて，要求されたセグメントにアクセスする．キュー内に要求されたセグメントに関する EPCT がない場合は，FREE キューより EPCT を一つ割り当て当該セグメントをページインした後，HOLD キューにつなぐ．

以上のように EPCT を設ける理由としては，以下の三つがある．

（1） セグメントに一つ EPCT を割り付けることによって（主メモリ中に同一セグメントを唯一つしか存在させないことで），主メモリの利用率を高める．

（2） データベース処理の矛盾をなくする．すなわち，同時実行制御を容易にする．

（3） 要求されたセグメントへのアクセスパス[12]をたどることによるオーバヘッドを極力少なくする．

5.2 セグメントの共用

OPT-R では，1パーティション（システムパーティション）内でマルチユーザおよびマルチタスク制御を行っているため，複数のタスクが同一のセグメントをアクセスする可能性がある．したがって，メモリ管理では，並行制御を可能にし処理の一貫性を保つために，EPCT 内にロック用のビットと ECB と呼ばれる事象を制御するキューを用意している．同一セグメントへの参照に対しては同時アクセスを許すが，更新要求の場合には要求元タスクを ECB にキューイングして，当該セグメントへのアクセスが許可されるまで要求元タスクを待ち状態にする．EPCT 内のロックビットと処理内容によるセグメント共用の関係を**表1**に示す．本方式は，共有ロック (shared lock) 方式の一つであり，lost-update の問題は解決される．しかし，

表1 ロックビットと処理内容の関係
Table 1 Relationships between lock bits and request.

ロックビット ＼ 処理	参照	更新
参照	○	△
更新	△	△
作成	△	△

○: 共用許可，△: ECB に登録

本管理方式ではデッドロックの生じる可能性がある．OPT-R では，これを解消するために，デッドロック処理用のシステムタスクを一つ用意してバックアウトすることにしている．バックアウトとは，要約すれば

（1） すべての変更を元の状態へ戻す，

（2） ロックされている資源を解放する，

（3） 処理要求を再スケジュールする，

よりなる．バックアウトは，次節に示す two-pages mechanism により高速に行うことが可能である．

5.3 two-pages mechanism

メモリ管理は，2次記憶からセグメントをページインするときに，その PBN を旧ページアドレスとして，主メモリ上のアドレスを現ページアドレスとして EPCT に設定する．当該セグメントの処理が終了すると，そのセグメントの EPCT は HOLD キューから UND あるいは NED キューにつながれる．その後，当該セグメントの主メモリが解放される．このとき，UND キューに関しては，その EPCT 内の現ページアドレスが削除されるだけである．一方，NED キューに関しては，その EPCT に対応したセグメントが2次記憶上の別の位置にページアウトされ，さらに，EPCT の現ページアドレスがその2次記憶上のアドレスである PBN に更新される．

以上のように，旧ページ (old pages) はデータに変更が加えられる前のページイメージとして，現ページ (current pages) は変更後あるいは処理中のページイメージとしてとらえられる．したがって，前節で触れたようなバックアウトは，現ページを削除して旧ページを主メモリにページインすることによって簡単に実現される．このような2重化は，4章で述べた SMAP および各システムオブジェクトごとのマスタカタログにも適用されている．

以上に述べた方式は，System R の方式に近い概念を採用しており，文献 11) にその実現法が詳細に述べられている．本論文の方式の特徴は，2次記憶の管理，一貫性制御，回復管理等をメモリ管理と結びつけて実現していることである（文献 11) では，メモリ管理との関連はあまり述べられていない）．すなわち，上記の各種の管理のためのセグメントに関する情報を，3章で述べた四つのシステムテーブルをもとに管理している．文献 11) における Map の2重化は，本方式の SMAP の2重化に対応しており，オープンされているセグメントを示す Map Switch は，当該セグメントに関する EPCT が各キューにつながれているか否かに対応する．さらに，セグメントの2次記憶上のスロットの位置を示す Vector の2重化は，EPCT 内の旧 PBN と新 PBN で対処しており，各種情報を極力分散させないようにしている．

6. おわりに

本論文では，データベースオペレーティングシステム OPT-R のメモリ管理方式について述べた．OPT-R のメモリ管理方式として，プログラムおよびデータを一つのパーティションで一元的に管理するための，ソフトウェアによるセグメンテーション方式を提案した．本管理方式を採用することによって，資源の限られた小型計算機システムにおいても，マルチユーザデータベースシステムが実現可能となる．

今後の課題としては，外部ページ管理テーブルおよび主メモリ領域の効率のよい割付け法の開発がある．たとえば，EPCT の割付けにおいて，処理中のセグメントによってすべての EPCT が占有されることは，要求時ページングにおけるスラッシングと同様の問題であると考えられる．このような確率は低いと思われるが，これを回避するために，システムの多重度による制御方法を考えておかなければならない．また，主メモリ領域の管理において，IPCT をページ対応に用意するのではなく，セグメント対応に用意してメモリオーバヘッドを軽減するための方策も考える必要があろう．

謝辞 最後に，貴重なご指摘を数多くいただいた査読者に感謝の意を表します．

参考文献

1) Denning, P. J.: Working Sets Then and Now, in Lanciaux, D. (ed.), *Operating Systems: Theory and Practices*, pp. 115-148, North-Holland, Amsterdam (1979).
2) Stonebraker, M.: Operating System Support for Database Management, *Comm. ACM*, Vol. 24, No. 7, pp. 412-418 (1981).
3) Chamberlin, D. D. et al.: A History of System R and SQL/Data System, Proc. Very Large Data Bases, pp. 456-464 (1981).
4) Stonebraker, M. et al.: The Design and Implementation of INGRES, *ACM TODS*, Vol. 1, No. 3, pp. 189-222 (1976).
5) Tuel, W. G.: An Analysis of Buffer Paging in Virtual Storage Systems, *IBM J. Res. Dev.*, Vol. 20, No. 5, pp. 518-520 (1976).
6) Sherman, S. W. and Brice, R. S.: Performance

of a Database Manager in a Virtual Memory System, *ACM TODS*, Vol. 1, No. 4, pp. 317-343 (1976).
7) Lang, T. et al.: Database Buffer Paging in Virtual Storage Systems, *ACM TODS*, Vol. 2, No. 4, pp. 339-351 (1977).
8) Fernandez, E. B. et al.: Effect of Replacement Algorithms on a Paged Buffer Database System, *IBM J. Res. Dev.*, Vol. 22, No. 2, pp. 185-202 (1978).
9) Baxter, A. Q. and Hart, J. M.: Software Segmented Virtual Memory, *Softw. Pract. Exper.*, Vol. 12, No. 2, pp. 185-194 (1982).
10) Reynolds, D. N. and Henry, G. G.: The IBM System/38, *Datamation*, Vol. 25, No. 9, pp. 141-143 (1979).
11) Lorie, R. A.: Physical Integrity in a Large Segmented Database, *ACM TODS*, Vol. 2, No. 1, pp. 91-104 (1977).
12) 大久保英嗣, 津田孝夫: データベースオペレーティングシステム OPT-R の設計目標とアーキテクチャ, 情報処理学会論文誌, Vol. 25, No. 4, pp. 535-543 (1984).
13) 大久保英嗣, 津田孝夫: データベースオペレーティングシステム OPT-R のタスク管理とトランザクションのスケジューリング技法, 情報処理学会論文誌, Vol. 25, No. 4, pp. 544-551 (1984).

(昭和 58 年 5 月 20 日受付)
(昭和 59 年 1 月 17 日採録)