データベースオペレーティングシステム OPT-R の性能評価†

大久保 英嗣†† 津田 孝夫††

われわれは，データベース処理に目的を限定したオペレーティングシステム OPT-R を設計・開発している．本論文は，現在，開発を終了している OPT-R 第1版の性能評価に関するものである．OPT-R のソフトウェアアーキテクチャについては，既発表の論文を参照されたい．本論文では，OPT-R の特徴的機能であるソフトウェアセグメンテーション技法とトランザクションのスケジューリング技法に焦点をあてて性能評価を行っている．さらに，第1版を実現した計算機 HITAC E-600/3 システム上での実験結果をもとに，OPT-R で提案している各種手法について考察を加えている．

1. はじめに

われわれは，ホストオペレーティングシステム上のデータベース管理システムの性能の問題を解決するための一つのアプローチとしてデータベース処理に目的を限定したオペレーティングシステム OPT-R を設計・開発した[1)-3)]．本論文は，実現を終了した OPT-R 第1版の性能評価[6)-9)]に関するものである．

本論文では，まず，OPT-R のメモリ管理方式であるソフトウェアセグメンテーションに関して，待ち行列モデルをもとに解析を行う．本解析では，メモリ管理モジュール内の各種待ち行列の長さをもとに，アクセスされるセグメントの数と主メモリ容量の関係，同時実行制御のためのロッキングオーバヘッド等について考察する．

次に，OPT-R で独自に提案実現している，1トランザクション内での処理多重度向上策である局所的スケジューリングの評価を行なう．

最後に，OPT-R 第1版を実現している HITAC E-600/3 システム上での実験をもとに，全体的な評価を行う．

2. OPT-R の実現環境

OPT-R 第1版は，HITAC E-600/3 システム上に実現されている．**図1**にその機器構成を示す．OPT-R では，現在，RS 232C 端末 (PIO 転送) 2台をユーザ用とし，コンソール装置 (DMA 転送) を DBA 用端末としている．また，データベース等の各種セグメントの格納媒体として，40 MB のディスク装置1台と 1 MB のフロッピーディスク装置2台をサポートしている．ディスクのデータ転送速度は 470 KB/sec，位置決め時間等のオーバヘッドの合計は，平均で約 40 msec である．フロッピーディスクは，転送速度が 62.5 KB/sec，位置決め時間等のオーバヘッドは，約 80 msec である．

3. ソフトウェアセグメンテーション

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．とくに，ユーザパーティションを 512 B 単位のページに分割し，セグメント (各種カタログ，データベース等) のユーザパーティションへのローディングを管理している．本章では，OPT-R におけるメモリ管理方式について待ち行列モデル[10),11)]をもとに解析を行う．

3.1 セグメントのアクセスパターン

OPT-R のトランザクション記述用言語の各文の実行においてアクセスされるセグメントの一覧を**表1**に示す．これらのセグメントのなかで，以下の4種類のアクセスパス上にあるセグメントが頻繁にアクセスされる．

(1) 関係カタログ—属性カタログ—BMF/PDF
(2) アサーションカタログ—アサーションリスト
(3) トリガーカタログ—トリガーリスト
(4) グラントカタログ

たとえば，PDF をアクセスする場合は，関係カタログ，属性カタログ，PDF の順にアクセスパスをたどらなければならない．

(2)および(3)は一貫性制御のためのものであり，(4)はアクセス権限のチェック用のカタログである．

† The Performance Evaluation of Database Operating System 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

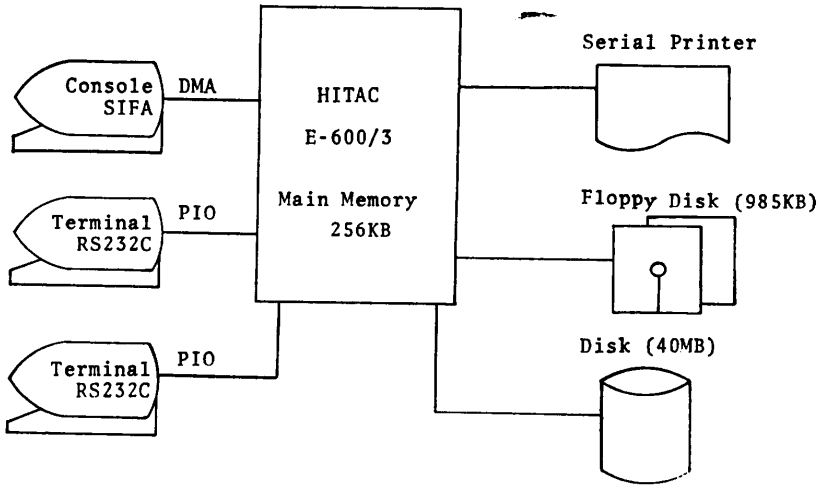

図 1 OPT-R のハードウェア構成
Fig. 1 Hardware configuration in the OPT-R system.

表 1 トランザクション記述用言語実行時のセグメントアクセスパターン
Table 1 Segment access patterns in the execution of transaction language.
1: create, 2: refer, 3: update

Statement \ Segment	ドメインカタログ	関係カタログ	属性カタログ	ビューカタログ	アサーションカタログ	トリガーカタログ	グラントカタログ	データベース	コメントファイル	アサーションリスト	トリガーリスト
Declare-domain	1										
Create-table	2	1	1		1	1	1				
Expand-table	2	3	1								
Define-view		2	2	1			1				
Drop-domain	3										
-table		3	3		3	3	3	3	3	3	3
-view				3			3		3		
-assertion					3				3	3	
-trigger						3			3		3
Comment on-table		3							1		
-column		2	3						1		
-view				3					1		
-assertion		2			3				1		
-trigger		2				3			1		
Assert		2	2		3					1	
Trigger		2	2			3					1
Grant		2	2	2			3				
Revoke		2	2	2			3				
Insert (value type)		2	2		2	2	2	3		2	2
Insert (query type)		2	2		2	2	2	2, 3		2	2
Delete		2	2		2	2	2	3		2	2
Update		2	2		2	2	2	3		2	2
Select		2	2	2		2	2	2			2

これらは，それぞれ各関係対応に作成される．しかし，(2)から(4)の中に現れるカタログはおのおの1ページであり，その読込み時間は，(1)のアクセスパスをたどることに比較すると非常に小さい．また，アサーション/トリガーに関する条件式が格納されている各リストもおのおの1ページであり，各リストの条件式の評価の際には(1)のアクセスパスをたどることになる．したがって，以下では，検索条件の評価あるいは結果の取出し等の際にたどられる(1)のアクセスパスに限定して解析を行う．

3.2 待ち行列モデル

図2に，OPT-R のメモリ管理の待ち行列モデルを示す．本モデルにおいては，以下の三つのキューが存在する．

Q 1: セグメントに関する情報が設定される外部ページ管理テーブル EPCT 確保のためのキュー

Q 2: 主メモリページ確保のためのキュー

Q 3: セグメントアクセスの同時実行制御のためのロック用キュー

したがって，これらのキューにおける待ち時間が，セグメントアクセスのオーバヘッドとなる．しかし，これらの待ち時間を解析的に求めることは，セグメントアクセスの任意性により困難であるから，以下では，GPSS によるシミュレーション実験をもとに考察する．

各セグメントのアクセス確率は，Q 3 の影響を見るために，以下のようにかたよりをもたせた．

P (関係カタログ) = 4/9, P (属性カタログ) = 3/9,
P (BMF) = 1/9, P (PDF) = 1/9

図2中のページイン/アウトおよび各処理モジュールに対応する処理ブロックの時間は，それぞれ

セグメントサイズ×30+40,
セグメントサイズ×20

とした（ただし，単位は msec である）．また，セグメントアクセスの到着時間間隔は指数分布に従うものとして，その平均を 25, 50 の2通りの場合について実験を行った*．**図3**から**図5**に主メモリ容量を 20

* 平均が 75, 100 の場合についても実験を行ったが，ほとんどの場合キューはできなかった．

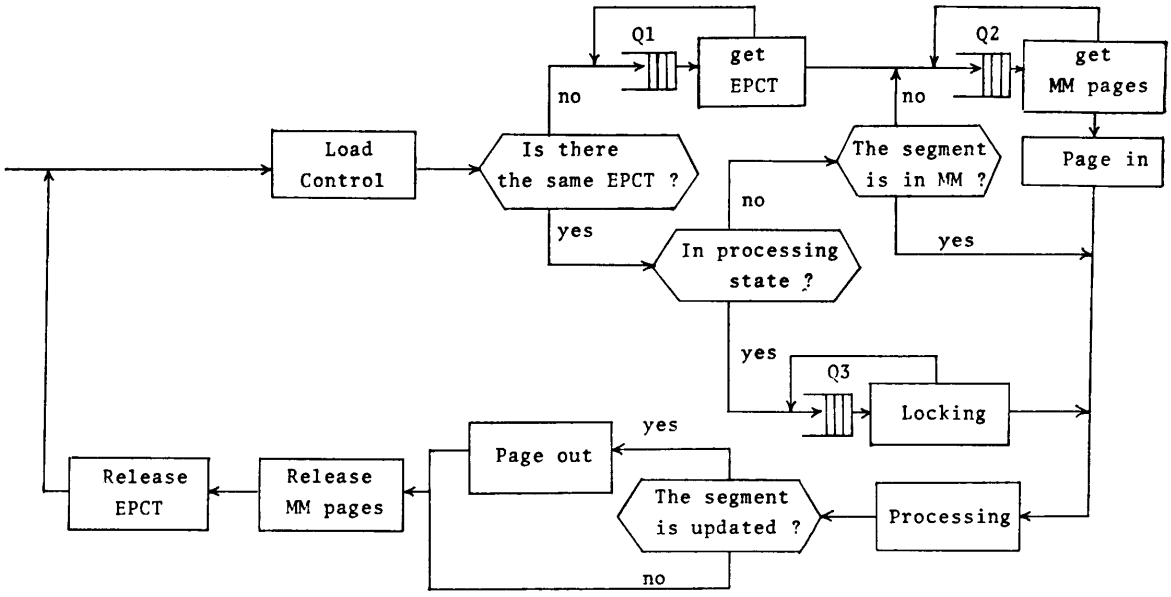

図 2　メモリ管理の待ち行列モデル
Fig. 2　Queueing model of memory management.

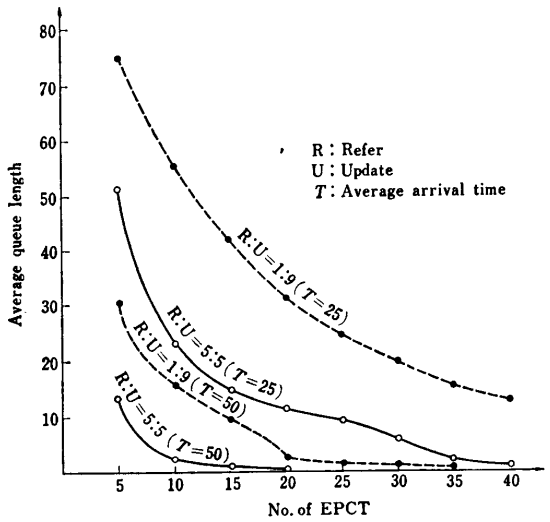

図 3　EPCT 待ちの平均キュー長
Fig. 3　Average queue length of EPCT wait.

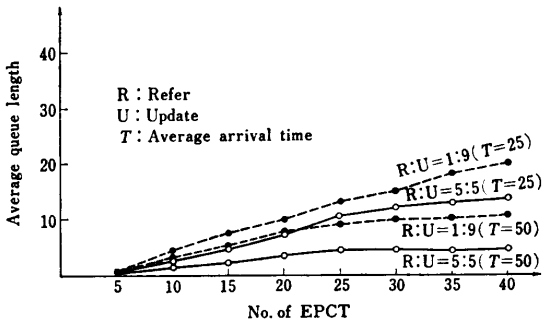

図 4　主メモリページ待ちの平均キュー長
Fig. 4　Average queue length of main memory page wait.

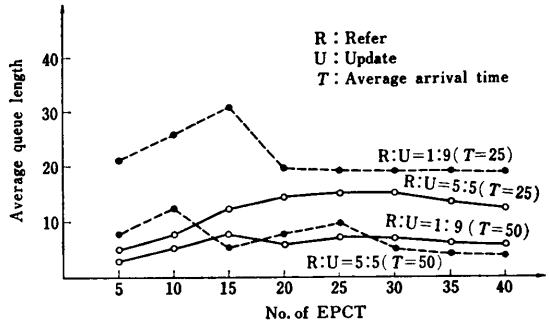

図 5　ロック待ちの平均キュー長
Fig. 5　Average queue length of locking.

ページとしたときの，各キューの平均キュー長と EPCT 数の関係を示す．なお，本実験では，40 回のセグメントアクセスが完了する場合をシミュレーションの終了条件としている．また，参照と更新の比を 5：5 と 1：9 の 2 通りで実験を行っている．

3.3 EPCT 数によるロード制御

OPT-R おいては，セグメントをアクセスする前にセグメント情報を設定するための外部ページ管理テーブル EPCT を確保しなければならない[3]．EPCT に空きがない場合は，当該トランザクションは待ち状態となる．したがって，システム内の処理多重度は，EPCT の数によって左右される．さらに，主メモリ

の利用率は，EPCT を使用中のセグメントのページ数に依存する．すなわち，大きなセグメントによってEPCT を占有している状態では，主メモリ利用率は高くなり，そうでない場合は主メモリ利用率は低くなる．

図3および図4より，EPCT 確保のためのキューQ1のほうが，主メモリページ確保のためのキューQ2よりもトランザクション到着時間間隔の影響を大きく受けることがわかる．すなわち，トランザクション到着率が高くなると，Q1は極端に長くなる．

EPCT 数

$$= \frac{\text{トランザクションを処理する平均時間}}{\text{トランザクションの平均到着時間間隔}}$$

とすることが考えられるが，実際問題として不可能である．したがって，OPT-R では，

EPCT 数＝主メモリページ数

として固定的に割り当てている．図3から図5までのキュー長の和は，EPCT 数が主メモリページ数と等しくなる付近から安定している．さらに，EPCT 数を大きくすると，EPCT 自体のメモリオーバヘッドも問題となるのでこれで十分であるといえよう．

3.4 ロック対象の大きさ

一般に，ロック方式は，物理ロック方式 (physical locks) と述語ロック方式 (predicate locks) に分けられる．OPT-R では，メモリ管理の中にロック機構を実現しており，物理ロック方式を採用している．物理ロック方式では，同時実行トランザクション数がロック対象の大きさ (granule) に影響される．すなわち，granule を小さくすると同時に処理することのできるトランザクション数が増加するが，それに伴ってロック処理のオーバヘッドも増加する．トランザクションの概念を導入しているシステムでは，ロックプロトコルとして二相ロックを採用している．本方式は，トランザクション開始時に当該トランザクションで必要な対象にすべてロックをかけ，当該トランザクション終了時点でロックを解除する方式である．

OPT-R におけるデータベースへのアクセスパスは，3.1 節で示したように階層構造になっているので，階層の上のほうのセグメントにロックをかけると，当該階層下のセグメントは他トランザクションでアクセス不可能となる．したがって，更新トランザクションにおいて各セグメントの物理アドレスを変更しないもの（すなわち，変更中のセグメント以外のセグメントアクセス）に関しては，共用を許す独自のロッ

```
@K000I  ENTER LOGON

@K001I  OPT-R SESSION START : USR01
READY
B
@K006I  BEGIN TRANSACTION#1
TRANS
SELECT S#,P# FROM SPJ WHERE QTY = 200 AND J# = "J1"
TRANS
SELECT S.S#,J.J# FROM SPJ -
       WHERE S.CITY = J.CITY AND S.STATUS = 20
TRANS
E
@K024I  TRANSACTION#1  IS ISSUED
READY



@K031I  TRANSACTION#1 IS FINISHED
READY
FETCH 1

S1  P1
S3  P3

S1  J5
S1  J7
S4  J5
S4  J7
@K720I  FETCH END
READY
```

図 6 トランザクションの実行例
Fig. 6 Example of the transaction.

クプロトコルを実現している．さらに，データベースを複数のエクステント（一つの属性につき最大60個）に分割して格納し[4]，エクステントを単位にロックを可能としている．

図5は，以上に述べた OPT-R の granule をロック対象としたときの，ロック待ちキューQ3の平均キュー長を示してある．EPCT 数が小さい場合は，主メモリ内に存在しうるセグメントが少ないわけであるから，ロック待ちは少ない．しかし，EPCT 数が主メモリサイズに近くなるとロック待ちが多くなり，主メモリサイズを越えた点で安定する．このことからも，前節で述べた主メモリサイズに対する最適 EPCT 数の議論が妥当であるといえよう．

以上，本章では，OPT-R で採用したソフトウェアセグメンテーションについて述べてきたが，本方式では，セグメント情報を設定するための EPCT の数がシステムの性能を大きく左右することがわかった．さらに，最適な EPCT 数は，ロックの granule を小さくすることによって，ほぼ主メモリページ数と同じだけ用意すればよいこともわかった．

4. トランザクションのスケジューリング

OPT-R では，一つのトランザクションをデータベース操作タスク (DOT と呼ぶ) への同期操作系列に変換して実行する[2]．本章では，**図6**のトランザクション実行例をもとに，OPT-R のタスク構成およびトランザクション処理方式について解析する．

(1)

```
POST  sel     (QTY = 200)
POST  sel     (J# = "J1")
WAIT  sel
WAIT  sel
POST  and
WAIT  and
POST  fetch   (S#)
POST  fetch   (P#)
WAIT  fetch
WAIT  fetch
```

(2)

```
POST  join    (S.CITY = J.CITY)
POST  sel     (S.STATUS = 20)
WAIT  join
WAIT  sel
POST  and
WAIT  and
POST  fetch   (S.S#)
POST  fetch   (J.J#)
WAIT  fetch
WAIT  fetch
```

図 7　DOT の呼出し系列
Fig. 7　Calling sequence of DOT.

4.1　トランザクションの生成と実行

本節では，ユーザが端末から入力したトランザクションが，中間コードにコンパイルされ実行される過程を簡単に説明する（詳細は，参考文献2) 参照).

システムは，BEGIN コマンドが投入されることによって，以降の入力がトランザクション記述用言語の入力であることを知る．その後，投入されるトランザクション記述用言語を文単位に構文解析し，中間テキストを生成する．

トランザクションの入力が終了すると，ユーザは END コマンドを投入し，トランザクション入力の完了をシステムに伝える．この時点で，制御がトランザクション管理へ移行する．トランザクション管理は，生成された中間テキストをもとに，データベース操作タスクへの同期操作系列を生成する．この同期操作系列は，**図7**に示すタスク同期用テンプレートの呼び出し命令の系列である．図7の(1)は図6の最初の SELECT 文，(2)のほうは図6の2番目の SELECT 文に対応する．中間テキストをすべて同期操作系列へ変換した後，トランザクション管理は，それを一つのタスク (TRT と呼ぶ) として生成する．この時点で，TRT はユーザから離れ，独自にその処理を行うことになる．すなわち，ユーザの端末では，別のトランザクションあるいはコマンドの入力が可能となる．

4.2　データベース操作の同期系列

図6の二つの SELECT 文は，図7のような系列へ変換される．ここで問題となるのは，図7の四角で囲まれた部分である．(2)は，結合操作 (join) と選択操作 (selection) を並行して実行可能であるが，(1)は選択操作が二つであるから，2番目の選択操作の実行が待たされることになる．すなわち，逐次的に実行されるのと同様になり，処理時間は短縮されない．結果の取出し (answer fetch) 用タスクについても同様のことがいえる．この解決策としては，同一処理を行う複数のタスク（たとえば，(1)の場合には二つの選択操作用タスクを）用意すればよいが，OPT-R を実現する小型計算機では，タスク自体のメモリオーバヘッドが無視できない．さらに，システム処理多重度との関連もあり，今後の課題である．

4.3　各モジュールのローディングオーバヘッド

データベース操作タスクの処理の主要部は，オーバレイモジュールとして構成されている．TRT と同期をとる部分のみが主メモリ常駐となっている．各モジュールはリエントラントコードとして実現されており，2回目以降の呼出しのローディングオーバヘッドが少なくなるようにしている．しかし，オーバレイ領域の大きさには制限があり，同時に主メモリに存在しうるモジュール数にはおのずと限界がある．したがって，前節までに述べた SELECT 文等における条件式の評価を行うための関係操作，論理演算，組込み関数等の17種類のモジュールに関しては，そのローディングオーバヘッドが無視できなくなる．

OPT-R では，現在，最大四つまでの演算に関するモジュールが同時に主メモリ上に存在しうるようにしている．これは，頻繁に使用される選択操作，結合操作，論理積，結果取出し用の各モジュールが同時にローディングされることによる．さらに，OPT-R では，全モジュールの相互呼出し関係に着目して，オーバレイモジュール群を10個のグループに分割し，おのおの独立した領域でオーバレイを行い，ローディングオーバヘッドを軽減している．

4.4　タスクスイッチ等の CPU オーバヘッド

OPT-R のユーザインタフェースは，4.1 節でも述べたように，ユーザと対話しながらトランザクション

記述用言語を各文単位に構文解析し，トランザクション記述が終了した段階でトランザクションとして実行を開始する．したがって，インタプリタ方式と異なり，トランザクションが実行されるに至るまでのタスクスイッチオーバヘッドは問題とならない．また，メモリ管理でセグメントのアクセスを一本化しているので，従来のシステムのようなオペレーティングシステムの空間からユーザ空間へのセグメントコピーによる余分な CPU 時間は必要としない．したがって，CPU オーバヘッドは，データベース操作の各モジュールのローディングと各操作モジュールでのセグメントアクセスの際に生じるタスクスイッチに限られる．しかし，これらに起因するタスクスイッチは当然のものであり，逆に，セグメントアクセスを少なくすることがただちに性能向上につながるといえる．

5. 性能測定

本章では，各種の検索文 (SELECT 文) の CPU 経過時間の測定をもとに，OPT-R で実現している各種機能について評価する．

検索文の構文は，以下のようになっている．

SELECT atrname-list FROM relname-list
WHERE boolean

本検索文の処理は，以下の2段階の処理よりなる．

(1) 条件式の評価

WHERE 句に指定された条件式 (boolean) を評価することによって，条件式を満足する TID (Tuple IDentifier) の集合を得る．

(2) 結果の取出し

(1)で得られた TID 集合に対応する各タプルの SELECT 句に指定された属性 (atrname-list) の実現値を取り出し，結果領域に格納する．

本実験では，検索文を実行するための例題データベースとして，文献5) における以下の四つの関係を使用した．

S (S#, SNAME, STATUS, CITY)
P (P#, PNAME, COLOR, WEIGHT)
J (J#, JNAME, CITY)
SPJ (S#, P#, J#, QTY)

これら四つの関係を表現するために，OPT-R では**表2**に示すセグメントが使用される．なお，各属性の PDF はおのおの一つで大きさは2ページである．

CPU 時間の測定にあたっては，E-600/3 のタイマ機構を使用した．タイマは，16 ビットのカウンタであり 1 msec 単位のカウントを行う．精度は ±2% 以内である．OPT-R のタイマ制御は，タイマ割込み契機を 10 msec として経過時間を計算している．これは，タイマ割込み契機をあまり小さくすると，タイマ割込み処理のオーバヘッドにより，実質的な時間が測定できないことによる．したがって，実験結果はすべて 10 msec 未満の値を切り捨てて各表に記載してある．

5.1 結果の取出しに関するコスト

本節では，以下の検索文について考察する．

(a) SELECT QTY FROM SPJ WHERE QTY=200
(b) SELECT S# FROM SPJ WHERE QTY=200
(c) SELECT S#, P# FROM SPJ WHERE QTY=200
(d) SELECT S#, P#, J# FROM SPJ WHERE QTY=200

これらの検索文を，おのおの一つのトランザクション

表 2 例題データベース
Table 2 Sample databases.

Relation catalog (2 pages)	No. of tuples	Attribute catalog (3 pages)	Length of values (bytes)	No. of BMFs
S	500	S#	2	5
		SNAME	8	3
		STATUS	4	2
		CITY	8	3
P	200	P#	2	4
		PNAME	8	2
		COLOR	8	2
		WEIGHT	4	2
J	300	J#	2	4
		JNAME	8	2
		CITY	8	3
SPJ	1,000	S#	2	4
		P#	2	4
		J#	2	4
		QTY	4	8

表 3 atrname-list のカラム数の影響
Table 3 Effect of the number of columns in the atrname-list. (Computed times are in sec.)

Transaction	Elapsed time	No. of I/Os
(a)	2.06	7
(b)	2.72	8
(c)	3.56	10
(d)	4.06	12

として実行した場合の CPU 経過時間と I/O 回数を表3に示す．各検索文においては，同一の条件式を指定しており，SELECT 句の属性の数が異なるだけである．

(a)から(d)の各場合とも，以下のモジュールおよびセグメントが共通にアクセスされる．

(1) 選択操作モジュール
(2) SELECT 文処理モジュール
(3) 結果取出しモジュール
(4) SPJ の関係カタログ
(5) QTY の属性カタログ
(6) QTY の BMF

さらに，各場合とも SELECT 句に指定された属性の個数分の PDF と BMF がアクセスされる．したがって，各場合の経過時間の差は，SELECT 句の属性数によって生じるといえる．すなわち，単一の関係に対する検索文においては，結果取出しの I/O 時間がその処理時間の多くを占めることがわかる．

条件式の評価に関しては，OPT-R のデータの物理構造である階層転置型ファイルによって，結果となるべき候補の推定が可能であるから，数回の I/O で済む．しかし，結果の取出しにおいては，条件式を満足する TID# の集合が，SELECT 句に指定されている属性の一つの BMF 中にすべて出現するとは限らないので，非常に多くの I/O が必要となる．最悪の場合，当該属性に関する BMF をすべて主メモリに読み込まなければならない．

以上のことは，タプル単位に格納されているファイルの場合は問題とならない．しかし，タプル単位の格納構造では，逆に，条件式の評価の際に問題が生じる．たとえば，副次索引が付加されていない項目に関する条件式の評価においては，当該関係をすべて読み込まなければならない．OPT-R においては，属性単位の格納構造（転置型ファイル）を採用しているので，SELECT 句中に当該関係のすべての属性が指定されない限り，関係をすべて主メモリへ読み込むことはない．

5.2 ソフトウェアセグメンテーションの効果

OPT-R では，セグメントのアクセスをメモリ管理で統一して行い，共有セグメントに対するアクセスオーバヘッドが少なくなるよう工夫している．本節では，このソフトウェアによるセグメンテーションの効果について考察する．

表4に，前節の検索文(b)と同一の検索文を一つのトランザクションに複数回投入した場合の CPU 経過時間を示す．

表4 ソフトウェアセグメンテーションの効果
Table 4 Effect of the software segmentation. (Computed times are in sec.)

Transaction	Segments are not in MM(1)	Segments are in MM(2)	(1)−(2)
(b)	2.72	0.05	2.67
(e)	2.75	0.08	2.67
(f)	2.78	0.11	2.67
(g)	2.82	0.16	2.66
(h)	9.33	3.37	5.96

(e) BEGIN (b),(b) END
(f) BEGIN (b),(b),(b) END
(g) BEGIN (b),(b),(b),(b) END

さらに，以下の複雑なセグメントアクセスパターンをもつトランザクションの経過時間も示す．

```
(h) BEGIN
      SELECT S#,P# FROM SPJ
        WHERE QTY=200 AND J#="J1"
      SELECT S.S#,J.J# FROM S,J
        WHERE S.CITY=J.CITY AND
                          S.STATUS=20
      END
```

表4の(1)は，主メモリ中に当該トランザクションに関するセグメントが何も存在しない場合を示しており，(2)はすべてのセグメントが主メモリ中に存在する場合を示している．

(b)および(e)から(g)の各場合において，(1)と(2)の差は一定である．すなわち，同一トランザクション内で同じセグメントをアクセスする場合は，最初の参照時でのアクセスだけですむことがわかる．したがって，(2)の時間は，選択操作と結果を生成するために必要な CPU 時間となる．(h)の場合の(2)の時間が大きいのは，選択操作，結合操作，論理積，結果取出し，SELECT 文処理の各モジュールのオーバレイによるオーバヘッドが含まれているためである（これに関しては，4.3 節で述べた）．

いずれにしろ，各場合とも必要なセグメントが主メモリ上に存在するときは，大幅な処理時間の短縮が期待できる．しかし，本実験は単一ユーザの場合の結果であり，複数ユーザの場合の実験は行っていない．複数ユーザの場合，とくに主メモリ上にセグメントをローディングするための空きページが存在しない場合は，ページング処理が行われ，そのためのオーバヘッ

表5 スケジューリングの比較
Table 5 Comparison of the schedulings. (Computed times are in sec.)

Sequence	OPT-R scheduling		Serial scheduling		(2)−(1)
	Segments are not in MM(1)	Segments are in MM	Segments are not in MM(2)	Segments are in MM	
1-2-3	4.18	1.42	4.64	1.42	0.46
1-3-2	3.85	1.42	4.64	1.43	0.79
2-1-3	3.69	1.47	4.81	1.46	1.12
2-3-1	3.86	1.45	4.81	1.45	0.95
3-1-2	4.02	1.48	4.82	1.47	0.80
3-2-1	4.19	1.44	4.65	1.45	0.46

ドが問題となる．OPT-R では，各セグメントのページサイズを小さくしてこの問題に対処しているが，セグメントアクセスパターンおよびページ置換えアルゴリズムなどと関連してむずかしい問題であり，今後の課題である．

5.3 局所的スケジューリングの効果

表5に，以下のトランザクションを OPT-R で提案しているスケジューリングに従って実行した場合と，スケジューリングを行わないで逐次的に実行した場合の比較を示す

```
SELECT S.S# FROM S, SPJ
    WHERE  S.S#=SPJ.S#[1]
    AND    SPJ.QTY>=200[2]
    AND    SPJ.P#>="P2"[3]
```

表中の Sequence は，WHERE 句の上記三つの条件式のトランザクション入力時の入力順序である．

表5を見ると明らかなように，すべての場合において，OPT-R のスケジュールのほうが経過時間が小さい．OPT-R のスケジュールによって，逐次的に実行する場合の経過時間の 10% から 20% ((2)/(2)−(1)によって計算) の時間が短縮されていることがわかる．上記のトランザクションは，CPU バウンド処理と I/O バウンド処理が都合よくオーバラップできる例であるが，このほかの実験においても，ほとんどの場合 OPT-R のスケジュールのほうが経過時間が小さくなっている．

表5より，当該トランザクションに必要なセグメントがすべて主メモリ上に存在する場合は，セグメントアクセスが行われないので，CPU バウンドの処理だけとなり，逐次的に実行する場合と同様の経過時間となる．さらに，本実験以外の例において，以下のことがわかっている．条件式に現れる属性に関して，BMF の各ブロックに格納されているタプル数が小さければ，ブロックごとの演算の処理に必要な CPU 時間はわずかであるから，条件式の評価は完全な I/O バウンドの処理になってしまう．したがって，CPU バウンド処理と I/O バウンド処理のオーバラップ可能な時間が小さくなり，スケジューリングの効果は期待できない．

以上要するに，OPT-R のスケジューリングにおいては，ブロック単位の選択操作，結合操作などの処理に必要な CPU 時間と，各ブロックを主メモリに読み込むための I/O 時間が同じになることが理想であるといえる．しかし，このためには，ブロックサイズを非常に大きくしなければならず，一貫性制御，更新操作の場合の処理と関連して問題が生じる．この点は，トランザクション内のすべての文に関する一括スケジューリングとともに今後の課題であるといえよう．

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 の性能に関して，種々の観点から考察した．OPT-R は，小型計算機用に設計されたものであるとはいえ，そのアーキテクチャは，ハードウェアとのごくわずかのインタフェース部分を除けば，多くの計算機で実現可能である．とくに，ソフトウェアによるセグメンテーション方式は，データベース処理に適したものであることがわかった．大容量メモリを備えた計算機では，予想以上の性能を期待できると考えている．

本論文で述べた結果は，システムが飽和状態にある場合の解析を含んでおらず，真の意味でのシステムオーバヘッドの解析は，3章で行ったようなシミュレーションに頼るしかない．しかし，数件の論文で提案してきた，OPT-R の各種機能の実現可能性および有効性は十分証明できたと確信している．今後は，OPT-R 第2版の開発を計画しており，おもに OPT-R を中心としたアプリケーションおよび分散データベー

スへと機能を拡張していく予定である.

謝辞　最後に，本システムの基礎検討から設計開発に至る過程でわれわれのプロジェクトに参加した，津田研究室の学生諸氏に感謝の意を表し，OPT-R 第1版開発の終りとする.

参考文献

1) 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).
2) 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).
3) 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 のメモリ管理方式，情報処理学会論文誌，Vol. 25, No. 4, pp. 552-559 (1984).
4) 大久保英嗣，津田孝夫：階層転置型ファイルに基づく関係操作アルゴリズム，情報処理学会論文誌，Vol. 26, No. 1, pp. 130-138 (1985).
5) Date, C. J.: *An Introduction to Database Systems*, 2nd ed., Addison-Wesley, Reading, Mass. (1977).
6) Stonebraker, M. et al.: Performance Enhancements to a Relational Database System, *ACM TODS*, Vol. 8, No. 2, pp. 167-185 (1983).
7) Chamberlin, D. D. et al.: Support for Repetitive Transactions and AD Hoc Queries in System R, *ACM TODS*, Vol. 6, No. 1, pp. 70-94 (1981).
8) Hawthorn, P. and Stonebraker, M.: Performance Analysis of a Relational Data Base Management System, Proc. ACM SIGMOD 1979, pp. 1-12 (1979).
9) Schmidt, J. W. and Brodie, M. L. ed.: *Relational Database Systems Analysis and Comparison*, Springer-Verlag, Berlin, Heidelberg, New York (1983).
10) Irani, K. B. and Lin, H.: Queueing Network Models for Concurrent Transaction Processing in a Database System, Proc. ACM SIGMOD 1979, pp. 134-142 (1979).
11) Kraemer, W.: Performance Investigation with a DOS/VS-based Operating System Model, *IBM Syst. J.*, Vol. 17, No. 4, pp. 409-443 (1978).

(昭和59年4月27日受付)
(昭和59年7月19日採録)