計算機システムの解析と制御 11－2
（1980. 11. 28）

関係データベース専用オペレーティングシステムの構成

大久保 英嗣　津田 孝夫
（京都大学・工学部）

1. はじめに

データベース専用オペレーティングシステムは，データベースシステム自体が，オペレーティングシステムの機能を有すると見ることにより，データベースの統一的管理とシステムの効率化を計ることを目的としている．従来のデータベースシステムにおける，共有のリエントラントコードの実行，制御情報の共有等は，データベースシステム自体がマルチタスキング機能を有する必要性がないという理由によるものであった[1]．即ち，同時トランザクションを操作するために，ホストオペレーティングシステムの機能を使用してきたのである．一方，ユーザ単位にコードを持たなければならないこと，オペレーティングシステムの機能と重複した機能を持つこと等，メモリおよび処理のオーバヘッドは避けられなかったし，オペレーティングシステムがサポートしている機能に制約されて，十分に効果的なものとは言えなかった[2]．さらに，データベースシステムに統一的な機能（RASISの実現等）を持たせるために，オペレーティングシステムとの重複部分は，拡大の一途をたどっている．

このように，データベースシステムはそれ自身，汎用のオペレーティングシステムの構造に近づいてきており，オペレーティングシステムをデータベース専用に構成し，全体としての効率化を計ることが重要になってきている．即ち，データベースマシーンのように個々の機能をハードウェア化し性能を向上させるのではなく，従来の計算機を独立したデータベース専用機とし，システムが制御するデータに着目し，各ユーザが大型データベースを効率良くアクセスする方法を，ソフトウェアで実現しようとするものである．

本稿では，以上のような問題点を踏えて，データモデルを高度のデータ独立性を有する関係モデルに限定して，効率の良い専用オペレーティングシステムを構成することを考える．

2. データベースシステムの問題点

オペレーティングシステムとデータベースシステムは，データベースシステムの処理をより一層効率化しようとしたとき，密接に絡み合い，その管理は複雑になる．本節では，オペレーティングシステムとデータベースシステムの重複部分，データベースシステムとしての問題点等について議論する．

2.1 メモリ管理技法

仮想記憶方式のオペレーティングシステムの下では，データベースシステムに，(1)バッファ管理の非効率性，(2)メモリの多層化，(3)オペレーティングシステムのメモリ管理方式による制約，等の問題点が生じる．

従来のオペレーティングシステムのメモリ管理の方式は，Denningのワーキングセットモデル[3]に代表されるように，プログラムの処理にその主体を置いてきた．しかし，データベースシステムにおいては，データベースが大規模になるに比例してデータ空間が広がり，データに依存したメモリ管理方策が必要になると考えられる．広いデータ空間において，サーチ型の処理を行なうとき，従来のプログラムに見られるような“参照の局所性”は存在せず，従ってデータベース特

有の参照ストリングが生じ，それに見合った方式が必要になる．

2.2 トランザクション・スケジューリング

データベースシステムは，オペレーティングシステムの1つのタスクとして，他の非データベースプログラムと同一レベルにある．従って，データベースシステムの各処理モジュールは，そのサブタスクとしてオペレーティングシステムに管理されることになる．従って，多重プログラミング環境を意識したスケジューリングを行なっても，それは，他のタスクと一括して処理されるため，最適化の意図が反映されなくなる．

2.3 データの保全・機密保護

データベースシステムにおけるアクセス制御は，オペレーティングシステムにおけるプロテクションの理論とは別に考えられてきた[4]．これは，データベースシステムとオペレーティングシステムの機密保護機能の差異を解決するために，オペレーティングシステムを拡張すると非常に複雑なものになってしまうという理由による．さらに，データベースシステムにおけるデータの保全，機密保護のチェックは，オペレーティングシステムの機能を使用して実現しており，非常なオーバヘッドが予想される．

2.4 データの物理構造とアクセス法

現行のデータベースシステムは，ホストのオペレーティングシステムのデータ管理インタフェースにアクセスパスを変換し[5]，オペレーティングシステムにその管理を任せている．データ管理の汎用的な機能には，データベースシステム側から見れば，冗長な部分が存在する．また，データベースを構築する場合には，オペレーティングシステムがサポートしているデータの編成に制約されて十分に柔軟なものとは言えない．

以上のように，従来のデータベースシステムには，解決すべき性能上の種々の問題点が存在する．データベース専用オペレーティングシステムの目的は，オペレーティングシステムをデータベース専用に構成することによって，これらの問題を単純化し，解決することにある．

3. 専用オペレーティングシステムの構成

本節では，データベース専用オペレーティングシステムに必要な機能および全体の構成を概観する．

3.1 機能的分割

専用オペレーティングシステムの各機能は，以下のように要約できる．

(1) スーパバイザ

割込み処理，メモリ管理，タスク管理の機能を実現し，従来のオペレーティングシステムと同様，専用オペレーティングシステムの核となるべきものである．

(2) データベース管理

データベースへのアクセスの管理，データベースの構築，同時アクセスの制御等，従来のデータ管理とデータベースシステムのインタフェース機能をオペレーティングシステムの中へ組み込む．

(3) トランザクション管理およびユーザインタフェース

ユーザ要求の翻訳（インタプリタ方式，コンパイル方式等）および最適化を行なう．さらに，アクセス権限，データの保全性のチェック，ビュー，他検索言語との互換性のためのエミュレータ機能をサポートする．

(4) バックアウト管理およびシステム回復

各ユーザ単位に孤立した矛盾のないバックアウトを構成し，システムの回復を容易にする．

(5) データベースシステムジェネレータ

IPL時に，データベースを仮想空間のデータ領域（3.3参照）にロードすることによって，以降のデータベースへのアクセスを減少させ，データベースアクセスによるオーバヘッドを軽減する．

カタログの管理を行ない，カタログ自体も関係と考えることによって，データベース管理とアクセスモジュールの共用性を計る．

3.2 処理の流れ

ユーザ要求は，以下に示すプロセスを経由して処理される．

(1) ユーザ要求の解析1

ユーザ要求の構文解析を行ない，基本的な関係操作のトリーに構成する．さらに，演算順序の変更による簡単な最適化を行なう．また，ユーザとシステムとのインタフェースのために，作業領域を確保する．

(2) ユーザ要求の解析2

上記の関係操作のトリーを入力とし，並行タスク認識により並行タスクのプログラムグラフに変換する．さらに，同時アクセスによるデッドロック防止のための各種資源のロックを行なう．機密保護，データの保全性のチェックもこの時点で行なわれる．

(3) タスクスケジュール

認識された並行タスク分割をもとに，ある簡単なアルゴリズムによってユーザ要求を多重プログラム化し，スケジュールを生成する．

(4) ユーザ要求の実行

生成されたスケジュールによって，データベースへのアクセス，関係操作を行ない望む結果を得る．

3.3 メモリの構成

主記憶領域は，システム常駐域，データ領域，ユーザ領域，システム非常駐域の4つに分かれる．システム常駐域は，スーパバイザの各処理モジュールの他に，ビュー，セキュリティ等に関する各テーブルが格納される．データ領域は，仮想空間として構成され，データベースの部分集合のみが格納される．ユーザ領域も仮想空間として構成されるが，作業領域等の一時的な領域として使用される．システム非常駐域は，オーバレイ構造として構成され，データベースの生成，削除等のユーティリティ関係，バックアウト，システム回復の各モジュールが格納される．データ領域，ユーザ領域は，別々のメモリ管理方策によって管理される．

4. メモリ管理技法

従来のデータベースシステムの性能上の問題を論ずるための1つの重要な主題として，データベースバッファのふるまいがあった（専用オペレーティングシステムにおいては，前節で述べたデータ領域がこれにあたる）．データベースバッファは，近い将来に再び参照されるかもしれないデータベースの部分集合を保存し，データベースへの付加的なアクセス回数を減少させることを目的としている．本節では，データベースとデータベースバッファ間のバッファページングと，データベースバッファと主メモリ間の仮想メモリページングの2つのページング機

構(“二重ページング(double paging)”と呼ぶ[6])に注目し，新しい置換えアルゴリズムを提案する．さらに，専用オペレーティングシステムのデータ領域の管理方式として，関係生成コストをDenningのGWS(Generalized Working Set)[7]に導入したメモリ置換えアルゴリズムについても考える．

4.1 二重ページングのモデル

Tuel[8]は，二重ページング環境下での最適な性能は，バッファサイズを有効な主メモリサイズに合致するように，縮少することによって達成されるとしている．ShermanとBrice[9),10)]は，有効主メモリサイズとバッファサイズが，データベースサイズとI/Oコストの2つの要素の種々の組合せによって決定されるべきであるとしている．即ち，データベースシステムを設計する場合，データベース，バッファ，主メモリの3つのサイズパラメータが性能を支配する1つの重要な因子となる．

メモリ階層のモデルは，図1のように，Dページからなるデータベースと Nページの仮想バッファおよび主メモリのMページがバッファに有効である3階層から構成される．データベースページが要求されると，最初にそれがバッファに存在するかどうか調べ，バッファにあれば仮想記憶管理を通してそれがアクセスされる．バッファになければ，1つのバッファページと要求したページが置換えられ，その後，仮想記憶管理を通してアクセスされる．このメモリ階層のモデルに以下の(1)から(3)を仮定する．

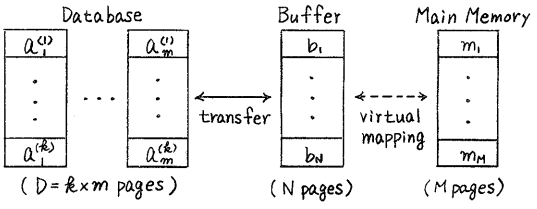

図1. メモリ階層のモデル

(1) データベースの物理構造は，関係データベースの論理構造(表形式)を線型に写像した転置型のファイルとする．

(2) データベース，バッファ，主メモリの各ページのサイズは同一とする．

(3) データベースへのアクセスの間のタスクスイッチは，属性の処理単位で生じ，当該属性の処理中には生じない．

4.2 データベース参照列

関係データベースシステムにおいては，ユーザの要求は，一連の関係演算の系列として内部的に展開される．従って，データベースへのページ参照列は，4.1節で設定したデータベースの物理構造のもとでは，属性方向のページ集合A_iとタプル方向のページ集合T_ℓが，それぞれある長さ又，ℓを持って交互に出現するパターンとなる．集合A_iおよびT_ℓの中では，所謂“参照の逐次性(sequentiality)”[11),12)]が見られ，参照列全体では，ワーキングセットのサイズがウィンドウサイズの線型な関数となってしまう．従って，1つのユーザ要求内では，前に参照したページを再び参照する確率は非常に小さくなる．さらに，データベースのページ参照におけるLRUスタックの深さ分布(stack depth distribution)を考えると，通常のプログラムに関する分布形状は，スタックの深さに関して急速に減少する特性[13)]，所謂，プログラムの“参照の局所性”が存在するが，データベースシステムの場合，プログラムスペースよりもデータスペースが極端に大きくなり，データスペースに関しては参照の局所性は見られない．

データの物理構造を前節のようにした場合のLRUスタックの深さ分布の特性として次のようなことが言える．

(1) 逐次性がある場合は，スタックの深さ1を除けば，属性ページ集合A_iとタプ

ルページ集合Tkの濃度kは，mの線型結合 $n_1 k + n_2 m$（n_1, n_2は正整数）に集中する．
(2)ランダム性が強い場合は，スタックの深さの各点で一様分布となる．

以上の特性が見られるが，これは関係演算のスケジューリングに影響される．1つの演算単位にスケジューリングを行なえば(1)の特性が強くなり，それよりも下位のレベル例えば読み込みと演算処理のような単位でスケジューリングを考えると，(2)の特性が強くなると考えられる．

以上のように，データベースシステムにおけるページ参照列は，“弱い局所性（weak locality）”と“強い逐次性（storong sequentiality）”によって特徴付けられる．

4.3 バッファ置換えアルゴリズム

Lang等[14),15)]は，主メモリ置換え方式をLRU，GLRU（Generalized Least Recently Used），ランダムとしたときの二重ページングに関する解析を行っているが，本節では，LRU置換えを拡張した新しいバッファ置換え方式を提案する．

(1)バッファ分割方式

前節で述べた参照のLRUスタックの深さ分布が属性ページ集合とタプルページ集合の濃度の線型結合に集中することから，属性ページの集合へのアクセスとタプルページの集合へのアクセスを分けて考える．即ち，NページのデータベースバッファをB_1，B_2と2分割し，B_1には属性ページの集合を，B_2にはタプルページの集合を格納して，この2つに関する置換えを統一的に管理する．参照列のLRUスタックの深さ分布は，B_1では$n_1 k$（$1 \leqslant n_1 \leqslant m$），$B_2$では$n_2 m$（$1 \leqslant n_2 \leqslant k$）に集中するので，$B_1$および$B_2$のサイズは各々$n_1 k$（$n_1 < m$），$n_2 m$（$n_2 < k$）ページが望ましい．

さて，バッファ置換えアルゴリズムをLRU置換えを基本として以下のように構成する．

(i)属性ページ集合への参照のとき，B_2に存在するページはB_1に移す．B_2の該当ページは空きページとするか，あるいはB_1の置換えられるページを格納する．
(ii)タプルページ集合への参照のとき，B_1に存在するページは参照するだけとする．

(2)クラスタリング法

データベース参照の逐次性を考慮し，Franaszek等[16),17)]が提案しているように，参照されたページを1つのクラスタとして構成するのではなく，属性ページ集合とタプルページ集合の単位でクラスタリングすることを考える．このクラスタリングに関しても(1)と同様にLRU置換えを基本として考える．2つのアルゴリズムが考えられる．

(i)属性ページ集合へのアクセスの場合，バッファに存在する同一属性ページの集合をバッファの先頭へ移動する．これは，同一属性ページ内の参照の逐次性を考慮しているためである．タプルページ集合へのアクセスの場合は，タプルページの集合に関して行なう．

あるいは，参照の予測性を属性ページとタプルページの参照の切換えに置くと次のようなアルゴリズムが考えられる．

(ii)属性ページへのアクセスの場合，次にタプルページがアクセスされる確率が高いとして，そのページが含まれるバッファに存在するタプルページの集合をバッファの先頭へ移動する．タプルページのアクセスの場合は，そのページが含まれる属性ページに関して行なう．

以上3つのアルゴリズムを述べたが，(1)のバッファ分割方式は，属性ページとタプルページの参照頻度に影響され，(2)のクラスタリング法は，前節で述べた参照の長さ又，Sに影響される．

4.4 関係生成コストによる置換え法

ここでは，前節の置換えアルゴリズムとは別に，関係間の依存性に関する情報を使用したアルゴリズムを考える．置換えの方策は，以下のようになる．

> 関係の維持コストが関係の生成コストより大なるページ（可変サイズ）を置換える．

この置換え法は，関係生成のための処理コスト，関係データベースへのアクセスコスト等をパラメータとして導入しており，メモリ管理のクラスとしては可変サイズの要求時フェッチ法に入る．また，DenningのGWS[7]を関係データベース用に拡張したと考えることもできる．（GWSでは，関係の生成コストがスワッピング・ロードとなる）．

維持コストは，space-time productで表現可能である．関係生成コストは，ユーザ要求によって定義された関係を生成するためのコストであり，データベースアクセスのコストと関係演算の処理コストに分かれる．関係を生成するためのデータがすでに主記憶に存在すれば，データベースアクセスのコストは零となるし，望む関係が生成されていれば，関係生成コスト自体が零となる．即ち，関係生成コストCは，次のように表現することができる．

$$C(X)=\begin{cases}0 & (\text{関係Xがすでに生成されているとき})\\ P(X) & (\text{関係Xに関するデータが有効であるとき})\\ P(X)+A(X) & (\text{上記以外のとき})\end{cases}$$

但し，Pは関係の処理コストであり，Aはアクセスコストである．

このコスト表現に，Casey等[18]が提案している関係間の従属性の概念を導入し，より詳細なレベルで置換えを考える．“Xが Yに従属する”とは，XがYによって生成されることを意味し，Y→Xと表わす．今，Y,Z→Xと仮定すると，関係Xを生成するためのコストは，

$$C(X)=\begin{cases}0 & (\text{関係Xがすでに生成されているとき})\\ C(Y) & (\text{関係Zがすでに生成されているとき})\\ C(Z) & (\text{関係Yがすでに生成されているとき})\\ C(Y)+C(Z) & (\text{上記以外のとき})\end{cases}$$

となる．

この場合，関係生成コストCは時間変化量となり，コスト計算のオーバヘッドが問題となる．従って，従属性の概念を取り入れる場合，関係全体よりも属性単位等の下位のレベルで行なうことが考えられる．

5. タスク分割とスワジューリング

関係データベースシステムでは，ユーザ要求は一連の関係演算の系列として展開される．本節では，これらの関係演算の各々を1つのタスクとして構成し，1つのタスク内のプロセッサセグメントと入出力セグメントをオーバラップして実行することにより，ユーザ要求全体の高速化を計ることを考える．これは，従来の関係データベースシステムにおけるユーザ要求の最適化[19)~21)]とは異なった手法である．

従来の最適化では，多重プログラミングの環境をユーザ要求単位にしか利用し

ておらず，そのスケジューリング自体もオペレーティングシステムに制御を委ねていた．これは，各資源の利用率の向上の目的に反するものである．従って，ユーザ要求内で多重プログラム化することにより，各資源の利用率を向上することが重要となる．

5.1 並行タスクの認識

本節では，ユーザ要求が関係演算のトリーに変換された段階から議論を始める．この認識のアルゴリズムは，Ramamoorthy 等[22)～24)]のアルゴリズムを基本としている．以下，図2の例について考えて行く．図2の関係演算のトリーの各ノードを1つのタスクとして構成し，有向グラフとして表現すると図3のようになる．

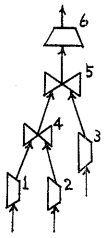

図2. 関係演算のトリー

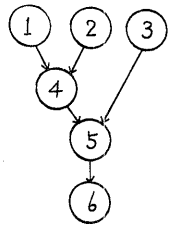

図3. プログラムグラフ

	1	2	3	4	5	6
1	0	0	0	1	0	0
2	0	0	0	1	0	0
3	0	0	0	0	1	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1
6	0	0	0	0	0	0

図4. 隣接行列

(1) 1→2→3→4→5→6
(2) 1→3→2→4→5→6
(3) 2→1→3→4→5→6
(4) 2→3→1→4→5→6
(5) 3→1→2→4→5→6
(6) 3→2→1→4→5→6
(7) 1→2→4→3→5→6
(8) 2→1→4→3→5→6

図5. スケジュール

従って，この有向グラフの隣接行列は図4のようになる．この行列をもとに並行タスクに分割すると，列分割の場合は{1,2,3},{4},{5},{6}，行分割の場合は{1,2},{3,4},{5},{6}となる．この2つの分割から生成される入出力セグメントとプロセッサセグメントのオーバラップスケジュールは，図5に示す8通りとなる．

5.2 コスト計算によるスケジューリング

前節の認識アルゴリズムによって生成されたスケジュールをもとに，時間コストによる評価を行ない最適スケジュールを得る．前節の例で，プログラムグラフの各ノードのプロセッサセグメントと入出力セグメントの時間を表6とする．

node \ time	Processor	I/O
1	3	7
2	5	7
3	3	7
4	8	0
5	6	10
6	5	6

表6. 各ノードの時間

Schedule	time cost
(1)	59
(2)	61
(3)	56
(4)	59
(5)	61
(6)	59
(7)	57
(8)	55

表7. 時間コスト

この場合，図5に示したスケジュールに関する時間コストは表7のようになり，スケジュール(8)が最適であることが分かる．

以上は，入出力セグメントが1つの場合のスケジューリングに関するものであったが，実際の場合は，独立に複数存在すると考えられる．即ち，各関係演算で要求する関係が複数ボリュームに渡って存在する場合，入出力セグメント自体でオーバラップシーケンスを生成することが可能となる．

5.3 スケジューリングコストとメモリ制約

前節までに示したような簡単なユーザ要求に関しては，スケジューリングコストは問題とならない．しかし，ユーザ要求が複雑になると関係演算のトリーの深さが増加し，上記のアルゴリズムによって得られるスケジュールは多項式のオーダーの組合せを越えてしまう．従って，最適に近いスケジュールを生成するための何らかの発見的手法が必要となる．関係演算トリーだけから得られる情報を使用する場合は，全体の演算数に対する2項演算数の割合が1つの尺度として考え

られる．

関係演算を行なう場合のメモリ制約は，スケジュールに著しい影響を与える．従って，メモリ制約をコスト計算の中に組み込む必要がある．現在，プログラムグラフの各ノードにプロセッサセグメントと入出力セグメントの時間の情報の他に，各演算に必要なメモリ量を与え，space-time productによってスケジュールすることを考えている．

5.4 トランザクション管理

ここでは，上記のアルゴリズムをマルチユーザの環境に拡張することを考える．例として，Q1，Q2の2つのユーザ要求が存在するとして，以下の状態を考える．

Q1 ： プロセッサセグメントの状態にある．

Q2 ： 入出力セグメント状態の前にプロセッサが必要である．

この場合，ユーザ要求Q1，Q2はプロセッサを競合して使用することになり，これを解決するために何らかの優先度が必要となる．ここでは，単純に各ユーザ要求の全処理時間に対するプロセッサの利用率で優先度を決定する．即ち，（プロセッサセグメント時間）／（入出力セグメント時間＋プロセッサセグメント時間）の大なるユーザ要求の優先度を上げ，小なるユーザ要求をスワップアウトすれば良い．但し，この場合スワップイン／スワップアウトのオーバヘッドを考慮しなければならないであろう．

6. データの物理構造とアクセス法

6.1 データの物理構造

4節で述べたように，データの物理構造は論理構造の表をそのままの形で線型に写像した転置型のファイルとする．転置型ファイルには，各演算の処理段階で不必要な属性のレコードを読込む必要がないという長所がある．一方，結果のタプルを取り出す場合のコストは，関係それ自体を1つのファイルとして構成する場合と比較して大きくなる．従って，ユーザ要求に即した各サブファイルのサーチ順を見つけることによってアクセスコストを減少させることが必要である．

Batoryは，転置型ファイルのサブファイルのアクセス順序に関する解析解を見つけることが困難であることから，発見的な技法を用いている[25]．それは，以下の単純な直観からきている．

(1) サブファイルのselectivityが同じ場合は，ブロック数が小さいサブファイルからアクセスする．

(2) ブロック数が同じ場合は，selectivityが小さいサブファイルからアクセスする．

この方法の本質は，このselectivityを如何に決定するかである．System-Rでは[5]，単純な決定方式を導入しているが，それが必ずしもすべてのユーザ要求に合致するとは限らない．しかし，各レコードに対してselectivityを持たせると，その情報を格納するための領域に関するオーバヘッドが非常に大きくなり，逆にアクセスコストに影響してしまう．このように，転置型ファイルを構成する場合，selectivityの選択が性能に大きく影響する．

6.2 同時アクセスの制御

データのアクセスを考える場合，同時アクセスの問題，特にデッドロックは重要な問題となる．各資源のロッキングは，トランザクション管理，メモリ管理と結びつけて初めて効率の良い物となる[26]．System-Rでは，デッドロックを許し

ており，デッドロックを検知した場合，速やかにその前の状況にバックアウトすることによって解決している．INGRESでは，ユーザ要求の処理における各プロセス間の絡行は，必要な資源すべてにロックがかけられる場合に制限している[2?]．これらの点を考慮して，専用オペレーティングシステムでは，ロックのレベルを各属性と1つの関係演算のレベルで規定する．

実際，関係全体をロックすれば，少数のタプルのみを扱うユーザ要求の他ユーザ要求への影響が大となるし，タプル単位にロックを考えると，関係全体をスキャンするようなユーザ要求のロックオーバヘッドが大となるからである．

7. 機密保護機能

従来のデータベース管理システムでは，各種の機密保護機能を実現する場合のオーバヘッドのために，単純な機能しか実現されていない．専用オペレーティングシステムの機密保護機能の基本構想には，オペレーティングシステムとデータベースシステムの機密保護機能の次のような差異を解消し，統一化することがある．

(1)データベースに関するオブジェクトの保護すべき期間が非常に長く，半永久的である．

(2)データベースの保護のレベルは，ビュー，関係，属性，タプルのように多岐に渡っている．

8. ユーザインタフェース

ユーザの検索要求言語の処理方式として，大別するとインタプリタ方式とコンパイル方式がある．コンパイル方式（Code Generation 方式）は，ユーザ要求のトランスレータによりユーザ要求すべてを主メモリ内の領域にコード化し，そこに制御を渡して実行する方式である．インタプリタ方式は，手続き的な処理の単位をシステム内の手続き呼び出しによって解釈実行する方式である．

専用オペレーティングシステムのユーザインタフェースとしては，このインタプリタ方式とコンパイル方式の2つの言語が必要であろう．これは，バッチ型の処理にはコンパイル方式の言語が，リアルタイム型の処理にはインタプリタ方式の言語が向いていることによる．

9. おわりに

以上，専用オペレーティングシステムの構成に関して，モデルを関係モデルに限定して述べてきた．従来から言われている関係データベースシステムの処理効率の問題は，データベースシステムをオペレーティングシステムとして構成することによって解消されるであろう．将来，データベースがますます大規模化するにつれて現在のシステムではサポートし切れなくなることが予想される．このような背景の下で，データベースシステムの諸機能を効率化し，システムとして構築して行くことは重要なこととなる．

最後に，貴重な御意見，御討論をいただいた津田研究室の諸氏に感謝の意を表します．

（参考文献）

1) Astrahan,M.M.et al. "System R : relational approach to data base management," ACM TODS, Vol.1, No.2, (1976), 97-137.
2) Stonebraker,M. "Retrospection on a Database System," ACM TODS, Vol.5, No.2, (1980), 225-240.

3) Denning,P.J. "Working Sets Then and Now," in Operating Systems : Theory and Practices(D. Lanciaux, Ed.), North-Holland, (1979), 115-148.
4) Wood,C.et al. "Database Security : requirements, policies, and models," IBM Syst. J., Vol.9, No.2, (1980), 229-252.
5) Selinger,P.G.et al. "Access Path Selection in a Relational Database Management System," IBM Research Rep. RJ2429, (1979), 1-59.
6) Goldberg,R.P. and Hassinger,R. "The double paging anomaly," Proc. AFIPS 1974 NCC, Vol.43, AFIPS Press, Montvale, N.J., 195-199.
7) Denning,P.J. and Slutz,D.R. "Generalized Working Sets for Segment Reference Strings," CACM, Vol.21, No.9, (1978), 750-759.
8) Tuel,W.G. "An Analysis of Buffer Paging in Virtual Storage Systems," IBM J.Res. Develop., Vol.20, No.5, (1976), 518-520.
9) Sherman,S.W. and Brice,R.S. "Performance of a Database Manager in a Virtual Memory System," ACM TODS, Vol.1, No.4, (1976), 317-343.
10) Brice,R.S. and Sherman,S.W. "An Extension of the Performance of a Database Manager in a Virtual Memory System Using Partially Locked Virtual Buffers," ACM TODS, Vol.2, No.2, (1977), 196-207.
11) Smith,A.J. "Sequentiality and Prefetching in Database System," ACM TODS, Vol.3, No.3, (1978), 223-247.
12) Rosell,J.R. "Empirical Data Reference Behavior in Data Base Systems," Computer, Vol.9, No.9, (1975), 9-13.
13) Turner,R. and Strecker,B. "Use of the LRU Stack Depth Distribution for Simulation of Paging Behavior," CACM, Vol.20, No.11, (1977), 795-798.
14) Lang,T.et al. "Database Buffer Paging in Virtual Storage Systems," ACM TODS, Vol.2, No.4, (1977), 339-351.
15) Fernandez,E.B.et al. "Effect of Replacement Algorithms on a Paged Buffer Database System," IBM J.Res.Develop., Vol.22, No.2, (1978), 185-202.
16) Franaszek,P.A. and Bennett,B.T. "Adaptive Variation of the Transfer Unit in a Storage Hierarchy," IBM J.Res.Develop., Vol.22, No.4, (1978), 405-412.
17) Bennett,B.T. and Franaszek,P.A. "Permutation Clustering - An Approach to On-Line Storage Reorganization," IBM J.Res.Develop., Vol.21, No.6, (1977), 528-533.
18) Casey,R.G. and Osman,I.M. "Replacement algorithms for storage management in relational databases," The Computer J., Vol.19, No.4, (1976), 306-314.
19) Smith,J.M. and Chang,P.Y.T. "Optimizing the Performance of a Relational Algebra Database Interface," CACM, Vol.18, No.10, (1975), 568-579.
20) Yao,S.B. "Optimization of Query Evaluation Algorithms," ACM TODS, Vol.4, No.2, (1979), 133-155.
21) Aho,A.V.et al. "Efficient Optimization of a class of Relational Expressions," ACM TODS, Vol.4, No.4, (1979), 435-454.
22) Ramamoorthy,C.V. and Gonzalez,M.J. "A survey of techniques for recognizing parallel processable streams in computer programs," Proc. 1969 FJCC, AFIPS Press, Montvale, N.J., (1969), 1-15.
23) Gonzalez,M.J. and Ramamoorthy,C.V. "Program Suitability for Parallel Processing," IEEE Trans. on Computers, Vol.C-20, No.6, (1971), 647-654.
24) Ramamoorthy,C.V.et al. "Scheduling Parallel Processable Tasks for a Uniprocessor," IEEE Trans. on Computers, Vol.C-25, No.5, (1976), 485-495.
25) Batory,D.S. "On Searching Transposed Files," ACM TODS, Vol.4, No.4, (1979), 531-544.
26) Gray,J.N.et al. "Granularity of Locks in a Shared Data Base," Proc.Int.Conf. VLDB, (1975), Vol.1, No.1, 428-451.
27) Stonebraker,M.et al. "The Design and Implementation of INGRES," ACM TODS, Vol.1, No.3, (1976), 189-222.