データベースシステム 93－5
システムソフトウェアと
オペレーティング・システム 59－5
（1993．5．27）

パネル討論:
オペレーティング・システム, データベース・システム, プログラミング言語の役割と接点

パネリスト: 鈴木則久 (IBM 東京基礎研究所)
牧之内顕文 (九州大学工学部)
上林弥彦 (京都大学工学部)
大蒔和仁 (電子技術総合研究所情報ベース研究室)
司会: 清木 康 (筑波大学電子・情報工学系)

概要

ネットワーク・システム, 分散処理システム, 並列マシン, マルチメディア・システムなどの新しい計算機システムの普及が, オペレーティング・システム、データベース・システム、プログラミング言語の分野にも大きな影響を与えている. そのような新しい計算機システムの環境においても, オペレーティング・システム、データベース・システム、および、プログラミング言語 (処理系) は, システム内の最も主要な構成要素である。このパネル討論は、これらのシステムの接点となる技術の動向を探ることを目的とする。次の項目を対象とする。

1. オペレーティング・システムとデータベース・システムの接点の技術動向
2. データベース・システムとプログラミング言語の接点の技術動向
3. マルチメディア処理に向けての各システムの役割
4. 超並列システムにおける各システムの役割

Panel Discussion:
Future Direction on the Technology of Operating Systems, Database Systems and Programming Languages

Panelists: Norihisa Suzuki (IBM Tokyo Research Insutitute),
Akifumi Makinouchi (Kyusyu University),
Yahiko Kambayashi (Kyoto University),
Kazuto Ohmaki (ETL)
Coordinator: Yasushi Kiyoki(University of Tsukuba)

Abstract

Network systems, parallel machines and multimedia systems have been widely spread to provide new computing environments. These new environments strongly influence the research areas of operating systems, database systems and programming languages. Even in such new environments, operating systems, database systems and programming languages are the most important components. In this panel session, we try to find future directions on the fundamental and essential technology in those systems. The following issues are discussed:

1. operating systems and database systems
2. database systems and programming languages
3. roles of those systems for multimedia processing
4. roles of those systems in parallel machines

データベースシステム 93－5－1
システムソフトウェアと
オペレーティング・システム 59－5－1
（1993．5．27）

プルーバブル情報ベース技術の確立を目指して

大蒔和仁　電子技術総合研究所 *

1　データベースから“プルーバブル”情報ベースへ

本稿はでデータベースの技術をより広く捉えソフトウェア技術全体を支えるための汎用技術であると捉えたい。[1]

一般に計算機システムのソフトウェアは応用プログラム、言語処理系、OSなどに階層化される。これらのソフトウェアはいずれもある種の構造をもったデータを効率よくまた信頼性高く参照更新できる技術を必要とする。さらに、現在の計算機環境では、分散していて且つ複数の利用者から共通に利用される状況を避けて通ることができないため、構造を持ったデータを信頼性高く参照更新できる技術がより一層重要でまた複雑になる。

本稿ではこの「計算機ソフトウェアから共通に利用される構造を持ったデータ」を情報ベースと呼ぶことにする。

応用プログラム、言語処理系、OSなどはいずれも、すなわち情報ベースをデータ操作のためのソフトウェアエンジンとして使うことによって情報ベースのデータアクセスに固有の問題解決をすべてこのエンジンに委ねることができる。そして、データアクセス以外の各ソフトウェア階層固有のアルゴリズム研究に傾倒できる。図1に情報ベースシステムとソフトウェア各階層の相互関係を示す。

このような情報ベースを信頼性高く構築するためには、あれば便利だからという機能を次々にアドホックに付け足すのではなく、理論的に何らかの意味で動作が保証されている機能を使って構築すべきである。本稿ではこの「理論的に何らかの意味で動作が保証する手だてが有り得る情報ベース」をプルーバブル(provable)情報ベースと呼ぶことにする。情報ベースはすべからくプルーバブルであるべきである。

2　情報ベースの静的部分と動的部分

個々の情報ベースはその中に蓄えているべき情報をある種の構造をもって保持している。これを情報ベースの静的部分と呼ぶことにする。複数の情報ベースが分散環境に配置されていて複数の利用者から利用されるとアクセスに関する時間順序制御をしなければならない。これを情報ベースの動的部分と呼ぶことにする。

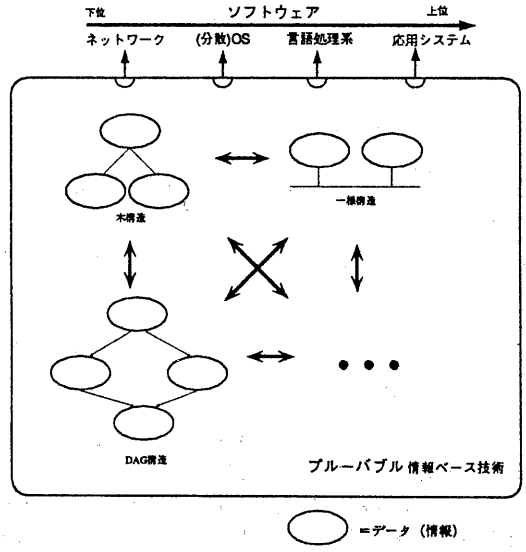

図 1: 情報ベース技術の位置付け

プルーバブルの性質はこれら静的部分と動的部分の両方が持たなければならない。これら二つの部分を無理矢理一つの理論によってプルーバビリティを実現しなければならない必然性はない。

我々はプルーバブル情報ベースを支えるための理論となり得るものの中で著名なものの幾つかについて調査してみた[1]。そして静的部分と動的部分のプルーバビリティに関係すると思われる理論で著名なものを分類すると次のようになるであろう。

静的部分：型理論は数学者の研究としてかなり前から研究が行われてきた。しかし計算機言語への応用が考えられたのは[2]による1970年代の後半であろう。そして情報ベースへの応用が考えられ始めたのは[3]からである。一方（おそらくこれらとは独立に）演繹オブジェクト指向についての研究がなされてきており、オブジェクトの構成方法には理論的な支えが必要であるとの認識から[4, 5, 6]などにおいてF-Logic、O-Logic、C-Logicなどが提唱されてきている。

*〒305 つくば市梅園 1-1-4 電子技術総合研究所情報アーキテクチャ部情報ベース研究室, E-mail: *ohmaki@etl.go.jp*

[1]筆者はデータベースについてはまったくの素人である。データベースに関する技術用語の不適切さはご容赦願いたい。

動的部分： 状態遷移機械をモデルとする並列記述に関する等価性などを詳細に調べることができる体系としてCCS[7]やπ-calculus[8]などに代表される並行プロセスの代数的アプローチがある。特に通信プロトコルの分野での応用が顕著である。情報ベースへの応用は現在あまり考えられていない。時間を考慮にいれなければならない情報ベースの研究では利用可能かどうかの検討をすべきモデルである。また、線形論理 (linear logic)[9] は従来の古典論理にかわる論理として、オブジェクト指向や並列動作の意味を規定するために利用し得るらしと考えられている非常に新しい体系である。

筆者らの研究室ではアクティブデータベースシステム[10]について並行プロセスの代数的アプローチの研究を行っている[11]。また、情報ベースのデータ（情報）として利用者インターフェイスを構成するプロセスを考えその結合関係の論理構造に対して代数的な理論の成果がどのように生かし得るかについての研究も行っている[12]。

3 将来への展開

最近ではネットワーク環境が構内や広域に対して非常な勢いで発達しており、理論的基盤を与えるのが追いついていないのが現状である。分散環境が発達すればするほど理論的基盤に基づいた情報ベースづくりの必要性が増す。前節で述べたたようにデータベースの研究とは独立に発達してきた理論の最新の成果を情報ベースの高信頼化のための理論的基盤として利用可能かどうかについて検討を行い、実際の問題に適用して理論的基盤の妥当性を検証すべき時期に来ている。

理論的研究はともすれば製品としてのソフトウェアの開発の後追いになってしまうことが多く、いわゆる「現実に役に立つ」理論が少ない。「情報ベース」は「プルーバブル」であると同時に

- 理論的裏付け
- 現実的裏付け

の両面から研究を展開すべきである。ともすれば理論は理論、現実は現実と分業され相互の関係が疎遠になってしまうが、それは避けるべきである。これ以上踏んではならない轍である。

謝辞

本稿の多くの部分は当研究所情報ベース研究室および言語システム研究室のメンバーによる当研究所調査報告[1]作成時の議論に基づいている。調査報告の執筆に当たった小方一郎、磯部祥尚、高橋孝一、海老原一郎、小島功、佐藤豊の各氏に感謝する。

参考文献

[1] 小方、磯部、高橋、海老原、小島、佐藤、大蒔：“プルーバブル情報ベースシステムに関する調査報告,” 電子技術総合研究所調査報告、第223号、March 1993.

[2] R.Milner : “A Theory of Type Polymorphism in Programming,” J. Computer and System Sciences, Vol.17, pp.348-375, 1978.

[3] A.Ohori : “A Study of Semantics, Types, and Languages for Databases and Object-Oriented Programming,” Ph D Thesis, University of Pensylvania, 1989.

[4] M.Kifer and G.Lausen : “F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance, and Scheme,” Proc. of the 1989 ACM SIGMOD Int'l Conf. on the Management of the Data, pp.134-146, 1989.

[5] D.Maier : “A Logic for Objects,” Proc. of the Workshop on Foundations of Deductive Databases and Logic Programming, pp.6-26, Washington DC, 1986.

[6] W. Chen and D.S.Warren : “C-Logic of Complex Objects,” Proc. of the 1989 ACM SIGMOD Int'l Conf. on the Management of the Data, pp.369-378, 1989.

[7] R.Minler : “Communication and Concurrency,” Prentice-Hall, 1989.

[8] R.Minler, J.Parrow, and D.Walker: “A Calculus of Mobile Processes, Part I and II,” Dept. of Computer Science, University of Edinburgh, LFCS Report, ECS-LFCS-89-85 and 86, 1989.

[9] J.-Y.Girard : “Linear Logic,” Theoretical Computer Science, 50, pp.1-102, 1987.

[10] U.Dayal: “Active Database Systems,” 3rd International Conference on Data and Knowledge Bases, pp.150-169, 1988.

[11] 小島、磯部、大蒔：“並行記述代数とアクティブデータベースの関連について,” 情報システムの能動化・協調化・自己組織化に関するObase workshop in Kobe, Feb., 1993.

[12] 佐藤、大蒔：“対話型ソフトウェアの動的な部品間接続の方式について,” 情報処理学会研究報告, SE89-7, pp.49-56, 1992.

データベースシステム　93－5－2
システムソフトウェアと
オペレーティング・システム　59－5－2
（1993．5．27）

オペレーティング・システム、データベース・システム、プログラミング言語の役割と接点

データベースからオペレーティングシステムへの機能要請

上林　弥彦　　京都大学工学部

まえがき

コンピュータの利用形態の変化とハードウェア技術の進歩によって既存のソフトウェアに対しても大きな変更が要求されつつある。ヨーロッパのエスプリプロジェクトによる予想によれば、システムの扱うべき情報量は年３０％の割当で増加し、トランザクション数は今後５年で１０倍の割合で増加するといわれている。また、ダウンサイジングによって各利用者ごとに１台の端末を利用できるようになったため、計算以外に利用者間の協調といった新しい用途が増えてきている。すなわちデータベース、知識処理、分散処理、利用者間通信といった用途の比重が増しつつある。ハードウェア技術面を考えると、マイクロプロセッサの処理能力は年に５０％の割合で向上しており、ＤＲＡＭの容量は３年で４倍になっている。ディスクのスループットのみが過去１０年間で２倍にしかなっておらず、これが大量データ処理のボトルネックとなりつつある。このような応用面や、ハードウェア技術を反映してデータベース研究の面からオペレーティングシステムはどうあるべきかという要請について思い付くまま列挙してみる。

分散並列への対処

ダウンサイジングは必然的に分散処理への対処を必要とする。特に、分散データベースに対処するためには、遠隔データベースをＳＱＬで操作できるだけでは不十分である。データ更新があれば処理の正当性（直列可能性）が要求される。直列可能性はデータ共有がなければ、全く問題とならないものであったが、全域的な正当性であるため局所性を生かして効率を向上する立場の分散処理とは相入れない。システムによって採用している並行処理方式やバックアップ／回復方式が異なるときにどのように統合するかは、処理の統合よりはるかに困難である。オペレーティングシステムはこのような統合のための基本的な機能を提供するべきである。まえがきで述べたトランザクション数の増大に対処するためには現在主流であるメインフレームによる処理では不十分である。メインフレームのトランザクション処理能力は今後１０年で５倍程度にしか増えないことが予想されている。ディスクのスループットの問題もあるため、ＣＰＵの並列化（特にマイクロプロセッサを用いるものがコスト／パフォーマンスが高い）、大容量主記憶の利用、ディスクアレイの利用が基本となる。複数のＣＰＵやディスクを扱う必要があるためこの場合にも並行処理の問題が生じる。並列処理マシンの問題はＣＰＵ数が増加したときにオペレーティングシステムが対処できないことのあることで、ハードウェア上のＣＰＵ数とオペレーティングシステムのサポートできるＣＰＵ数の異なることがある。ＳＱＬ等のデータベース言語は集合的操作が主であるため並列処理との適合性は高いといえる。

高信頼性・高可用性への対処

共有データがからむと、信頼性の向上が非常に重要となる。これは、いったん間違えたデータが記憶されると、その影響がどんどん波及していく可能性があるためである。また、ユーティリティーとしての情報システムは２４時間休みなく働くことが必要である。２重系のシステムではシステムの故障修理中も動くことが要求される。ソフトウェアやオペレーティングシステムのバージョンアップ時にも使える必要がある。メインフレームで作られた銀行システムではログを取るためのオーバーヘッドが４０％にも達することがあるといわれており、このオーバーヘッドを減らすことができればシステムの効率がかなり向上する。回復処理による停止時間もできる限り短いことが望まれる。

データベース向きオペレーティングシステム

ＵＮＩＸを用いた高可用性データベースシステムを開発して十分なパフォーマンスが出ずＵＮＩＸを書き換えたセコイヤシステムの例でわかるように現状のオペレーティングシステムは必ずしもデータベース向きではない。また、オペレーティングシステムの上にデータベースを実現するのが一般的であるが、基本的データベース機能を実現し、その上にオペレーティングシステムを作るといった方式も採用されている。オペレーティングシステムの中には可搬性を重視して中にデータベース的な表を用いたものもあるが、データベースの上にオペレーティングシステムを実現するのが実際的かどうかは今後の問題であろう。プラント制御に用いる実時間オペレーティ

ングシステムにおいても、データ量が増え、実時間データベースを実現する必要が生じている。現状では読み出し専用と書き込み専用のトランザクションを分けて特殊なトランザクションのみを考えることにより、デッドロックや直列可能性で生じる問題を回避しているが、実時間オペレーティングシステムがより一般的なデータベース機能を効率よく実現できるようになる要請は大きい。

セキュリティ

オペレーティングシステムのセキュリティモデルでは、データを扱うオブジェクトが一般にプログラムであるという仮定があった。すなわちオブジェクトは自分のセキュリティレベル以下のデータを読み込め、自分のセキュリティレベル以上のデータに書き込めるという制限により、レベルの高いデータの内容がレベルの低いデータに混入することを防いでいた。しかし、オブジェクトとして利用者を考えると1番レベルの低い利用者はすべてのレベルのデータに書き込み可能で、データの汚染が起こる。このセキュリティ管理についてはより強力な方式が必要といえる。

並行処理

複数個の要素からなるシステムで各要素をできる限り同時に働かそうとするのが並行処理である。並行処理は効率向上のために導入されたが、現実には全域的正当性を必要とするためこの対処のためのオーバーヘッドが高くなる。このため目的別にトランザクションを定義しそれぞれに対する並行処理方式の開発や、大きな仕事に対するトランザクションを分割しトランザクション間に従属性を導入する等の方法が取られている。このような機能を効率よく扱えるようにするべきである。

応用統合

各種システムの統合には通信レベルで行なうものや、共有データベースによるものが考えられ後者の方が結合度は高い。オープンシステムは今後重要で、通信面はオペレーティングシステムがサポートして各種応用プログラムで共有するべきである。

ハードウェア

過去にコアメモリから半導体メモリに移行したとき、ログの先書きへの変更等を必要とした。新しいハードウェアを有効に使うにはオペレーティングシステムレベルでの対処が必要である。例えば高速のＤＲＡＭはバッファを有しているため、あるバッファサイズ内の順アクセスを行なうとかなり高速となる。フラッシュメモリがディスクを置き換える場合には、オーバライトできないこと、読み出しと書き込みの実質的な時間が大幅に異なること等に対する対処が必要である。

オブジェクト指向

オペレーティングシステムがオブジェクト指向となり、プログラミング言語も利用者インタフェースもオブジェクト指向となったときに、データベースや知識処理もオブジェクト指向となれば統一がとれてよいという話がある。しかし、実際にオブジェクト指向データベースを開発している人々の楽観的な予測でも、９５年にデータベース全体の７％を占めるに過ぎない。これは利用者の保守性（すでにあるデータや応用プログラムから離れられない）だけが原因でなく、データの操作にＳＱＬ以上の言語があまり必要でない用途が広いためである。今後も、データの電子化による利点（検索コストが安い、コピーが容易、発送も簡単等）を利用して単に情報をデータベース化するだけの利用が大きく増えると考えられる。したがって、オブジェクト指向オペレーティングシステムで全て解決と言うわけにはいかない。オブジェクト指向データベースについてもデータベースの立場とプログラミング言語の立場はかなり異なっておりその２つの立場の融合と言った試みもある。しかし、分散オブジェクト管理は異種分散の自律的システム統合のための技術として有望視されておりトリガー等のＯＳレベルでの実現も必要であろう。

むすび

以上思い付くままにまとめたが、大きく分けてデータベースの機能を実現しやすくするためのものと効率向上のためのものとがある。

データベースシステム　93－5－3
システムソフトウェアと
オペレーティング・システム　59－5－3
（1993．5．27）

オペレーティングシステム，データベースシステム，プログラミング言語の役割と接点

清木 康

筑波大学電子・情報工学系

1　はじめに

ネットワーク・システム，並列マシン，超並列マシン，マルチメディア・システムなどの新しい計算機システム環境の普及は、オペレーティング・システム、データベース・システム、プログラミング言語の研究分野に多大な影響を与えている．そのような新しい計算機システムの環境においても，これらの研究分野は，計算機システムにおける基幹的な構成要素であり、新しい環境に適合した機能、性能に関する対応が不可欠である．

このパネル討論では，新しい計算機環境におけるオペレーティング・システム、データベース・システム、プログラミング言語の関連について概観し，それらの接点に関する本質的な技術を探り，今後の技術的動向について議論する．

2　オペレーティング・システムとデータベース・システム

オペレーティング・システムおよびデータベース・システムに関連する新しい計算機システム環境として，次のような項目がある．

1. ネットワーク・システム
2. 並列マシン，超並列マシン
3. 分散処理
4. コンティニュアス・メディア，マルチメディア
5. リアルタイム・システム
6. ユーザ・インターフェース

これらは，オペレーティング・システムとデータベース・システムに関する研究分野に共通の項目である．データベース・システムとオペレーティング・システムに関する研究は，それぞれの高性能化，高機能化，高信頼化を目指して，互いに独立に行われてきた．オペレーティング・システムでは，多様な応用プログラムを対象として，それらをハードウェア・アーキテクチャおよびシステム・ソフトウェア・アーキテクチャから独立させるための抽象化の枠組みを規定することを目指してきた．また，データベース・システムにおいては，データベースの応用に対して，データ構造およびデータの内容に関する抽象化を行い，データ独立性を実現する枠組みを規定することを目指してきた．

最近では，データベース・システムの研究の主要な対象は，データ構造だけでなく，データベースの操作系に関する柔軟な記述を応用側に提供することに向い，データベース・プログラミング言語，オブジェクト指向データベース，関数型データベース，演繹データベースの研究が活発になっている．オペレーティング・システムの研究においても，永続的データの扱い，データの高信頼化など，データの構造および内容を対象とした機能に関する研究が行われるようになっている．

このように，両者の境界は，以前よりも明確ではなくなってきている．また，ネットワーク・システム，並列マシン，さらには，マルチメディア・システムなどによるハードウェアの環境の変化は，オペレーティング・システムおよびデータベース・システム両者に共通のものであり，それらの環境への対応に関しては，両者のシステムの設計，および，実現が独立に行うのではなく，むしろ，それらを一体化して検討して行く必要があると考えられる．

3 データベース・システムとプログラミング言語

データベース・システムの普及に伴い, その応用分野は多様化し, また, 知識ベースへの拡張, および, 複雑な構造をもつデータの統合に関する研究が活発に行われている. また, 応用分野の多様化への対処, 知識ベースへの拡張, 複雑なデータ構造のデータベースへの統合を実現するために, データモデル, データ記述言語, データ操作言語, および, それらの処理系に関する新たな研究が行われている. それらの研究は, 理論的, あるいは, 技術的に整った計算体系に基づいているものが多く, 特に, 論理型, 関数型, あるいは, オブジェクト指向型計算モデルなどのプログラミング・パラダイムに対応したデータモデリングが試みられている. 知識ベースへの拡張に関しては, 論理型と関係データベースの統合により実現される演繹データベースの研究がある. 複雑なデータ構造の支援に関しては, オブジェクト指向型の概念に基づくデータモデルの提案が行われている. また, データ構造およびデータ操作に関する宣言的な記述を目的として, 関数型の概念を適用する試みが活発である.

このように, データベース・システムとプログラミング言語は, プログラミング・パラダイムを接点として, 強く関係付られるようになってきた. データベース・プログラミング言語の分野では, プログラミング言語とデータ永続性に関する議論が活発に行われている. 今後, プログラミング言語処理とデータベース・システムの接点に関する研究は, さらに重要になっていくものと考えられる.

4 まとめ

以上の観点から, 新しい計算機システム環境におけるオペレーティング・システム, データベース・システム, および, プログラミング言語の技術的関連について討論を行う. さらに, マルチメディア処理, および, 超並列システムにおける各システムの技術について, 今後の動向を探る.

参考文献

[1] Cattell, R.G.G. "Next-generation database systems: Introduction," Comm. ACM, Vol.34, No.10, pp. 30-33, Oct. 1991.

[2] Kiyoki,Y., Kurosawa,T., Kato,K. and Masuda,T., "The Software Architecture of a Parallel Processing System for Advanced Database Applications," Proc. the 7th IEEE international conference on Data Engineering, pp.220-229, April 1991.

[3] 清木 康, "データベースとオペレーティング・システム," コンピュータ・システム・シンポジウム論文集, pp.31-40, 情報処理学会, Oct. 1992.

[4] Shinjo,Y. and Kiyoki,Y. "Harmonizing a distributed operating system with parallel and distributed applications," Proc. 1st IEEE Int. Symp. on High-Performance Distributed Computing, pp. 114-123, Sept. 1992.

データベースシステム　93－5－4
システムソフトウェアと
オペレーティング・システム　59－5－4
（1993．5．27）

データベースとプログラム言語とオペレーティング・システム

日本アイ・ビー・エム東京基礎研究所
鈴木則久

計算機が人類にもたらした最大の恩恵は、大量のデータの蓄積とその蓄えられた大量のデータの中から欲する情報を高速に取り出すことを可能にしたことであろう。

今回の合同研究会では、このデータベースのプログラム言語とオペレーティング・システムとの接続点について検討していただいている。データベースの研究は応用面ばかりに集中しているわが国の研究の中で、基礎的な面での研究が盛んになることを願って止まない。

さて、データベースとプログラム言語との関係であるが、良い言語と言うのは、データベースが自然な延長として存在することである。プログラム言語の中でのデータベースに関する演算は、データ宣言、データ演算、トランズアクション処理の三つに分けられる。データ宣言では、データベースは基になるプログラム言語の型宣言として扱えなければならない。またデータ演算では、プログラム言語の演算の一つとして、その型理論の中で扱われる、すなわち型チェックも普通と同じ様に可能でなければならない。またトランズアクション処理も無矛盾な形で、プログラムの意味論の中で扱われており、トランズアクションが完了したか中止したかに関わり無く再現可能な演算でなければならない。

また、良いオペレーティング・システムは、プログラム言語の延長として、良いプログラム環境を成していなければならない。

オペレーティング・システムでのシステム変数はプログラム環境での大域変数と同じ操作で扱われ、同じ様に取り扱えなければならない。システム・コールは、プログラム環境でのライブラリ・プロシージャ・コールと同じ機構で呼ぶことができなければならない。これが唯一の要求であり、大変簡単なものである。

さて、これをオブジェクト指向データベースを設計する場合に当てはめて考えてみよう。まず、オペレーティング・システムのコマンド・レベルでは、データベースはシステム変数に代入される。そして、データベース演算はコマンドレベルで行え、結果はまたシステム変数に代入される。また同様なことが、プログラム環境に入ったトップ・レベルでも、プログラムの中でも実行できなければならない。

データベースシステム　93－5－5
システムソフトウェアと
オペレーティング・システム　59－5－5
（1993．5．27）

データベーストランザクションとオペレーティングシステム

牧之内顕文　白光一
九州大学工学部情報工学科

トランザクション管理の問題はデータベースシステムの中心問題である。この問題には並行処理制御及びリカバリという二つの大きな問題がある。現在までの商用化されたデータベースシステムを見ると、そのトランザクション管理はオペレーティングシステム上のユーザレベルで独立的に実現されてきた。これらのシステムでは、オペレーティングシステムのファイルシステムに基づき、バッファプール、ロックテーブル及びログなどをユーザレベルで構築して並行処理制御とリカバリを行うため、トランザクション処理にはかなりのオーバヘッドが存在する。従って、データベースシステムのトランザクション管理の効率向上にはオペレーティングシステムとのインタフェースが大きな課題である。

十年程前から、トランザクション管理をオペレーティングシステムの基本機能の一部として実現しようというアプローチがあった。このアプローチでは、従来のバッファプールを使わず、オペレーティングシステムがデータベースファイルをユーザの仮想アドレス空間にマップする。ユーザプログラムは仮想アドレス空間にアクセスすることによりファイルをアクセスできる。また、オペレーティングシステムはそのマップされた仮想アドレス空間に対して、ページロック及びページログにより並行処理制御とリカバリを実現する。

このようなアプローチの主な問題点はトランザクション管理とデータベースのデータ管理との間には壁が存在することである。データベースのデータは構造化され、またセマンティクな関連を持っている。従って、並行処理制御はデータ構造や、セマンティクなどを反映する必要がある。例えば、B+木によりデータベースをアクセスする場合、インデックスに対するロック方式はそのデータに対するロック方式と異なる場合がある。また、階層化されたデータに対して、上位データをロックする場合、その下位データもロックする必要がある。しかし、このアプローチでは、仮想アドレス空間上で全てのデータが同じページデータとして一元的に扱われるため、上位に於けるデータ構造が反映されない。また、一般的に、オブジェクトサイズはページサイズより小さい場合が多い。従って、ページ単位でのロックはプログラムの並列性が低くになり、分散システム環境上でのページ単位でのコンシステンシを保持するには、通信コストが高くなるなどの問題もある。

一方、このアプローチには、様々な利点がある。特に近年、データベースの応用範囲が段々広がりつつあり、また、マルチプロセッサ、大容量メモリ及び高速ネットワーク等といわゆる新しい計算機システム環境も実用化され始めた現状で、このアプローチの利点はもっと強調されてよい。このアプローチでは、データベースは仮想メモリの一部になっているため、永続データは通常のプログラムの揮発データと機能的にも効率的にも同様に扱える。また、データベースファイルのデータ形式はメモリ上の形式が同一になり、永続データとそれを扱うプログラムが同じ仮想アドレス空間にあるため、複合オブジェクト及び長大オブジェクトも効率的で簡単に扱える。さらにオブジェクトポインタ変換ためのオーバヘッドがほとんどない。

しかし、上で述べたように、トランザクション管理をオペレーティングシステムの基本機能の一部として実現すると、トランザクション管理とデータベースのデータ管理との間の壁が問題となる。この問題を避けるために、トランザクション管理を（オペレーティングシステムの）ユーザレベルで実現すると、ユーザレベルで構築したデータベースのデータ構造などをトランザクション管理へ反映できることになる。しかし、一方、トランザクション管理をユーザレベルで実現すると、並行処理制御ためのロックなどのオーバヘッドが大きくなる。従って、問題はどうようにオペレーティングシステムの仮想アドレス空間管理メカニズムを利用して、データベースシステムのデータ管理とうまく接続するかということである。ここでは二つの問題がある。一つはオペレーティングシステムがどのようなプリミティブを提供するか。もう一つはデータベースシステムはどのようにこれらのプリミティブを利用し、

トランザクション管理を実現するかということである。

実用化されたオペレーティングシステムを見ると、SunOSでは、メモリマップド(mmap())機能を提供した。ユーザはこの機能を利用し、指定したファイルを自分の仮想アドレス空間にマップできる。現在、注目を集めているオブジェクト指向データベース管理システムObjectStoreはこの機能を利用して実現した。しかし、この機能を利用しても、ユーザはそのマップされた空間のキャッシュを管理することができない。即ち、ユーザはそのキャッシュを直接ロックすることができない。そのため、並行処理制御を行うには、ObjectStoreのようにユーザレベルでロックテーブルを構築し、ロック管理を行う必要があり、オペレーティングシステムのメモリ管理機能をまだ十分に生かしていない。また、ObjectStoreでは、ページ単位にロックするため、並列度が低い、通信コストが高いなどの問題は依然存在する。

カーネギメロン大学で開発された分散型オペレーティングシステムMachは新しい計算機システム環境を考慮した非常に興味深いものである。特に、Machでは、ユーザプログラムが仮想メモリを管理することができるようなプリミティブ(EMMI: External Memory Management Interface)を提供している。従って、Machに基づいて上で述べたアプローチを実現すると、ユーザレベルで実現できるし、また、Machの実メモリ管理メカニズムを生かすことが可能になる。我々はMachを利用し、上述したアプローチによりWAKASHIサーバを実現した。ユーザはWAKASHIサーバを使って、指定したファイルを自分の仮想アドレス空間にマップすることができる。また、WAKASHIサーバはそのマップされた空間を分散共有メモリとして扱い、メモリコヒーレンスを保持する。WAKASHIサーバをデータベースベンチマークにより実行評価したが、マルチメディアデータを扱うデータベースシステムとして使う時に、WAKASHIサーバはバッファ方式より性能面で特に有利になることを確認した。また、WAKASHIサーバはユーザレベルで実現したので、ユーザが構築したデータベースのデータ構造などをWAKASHIサーバへ反映することが可能である。しかし、そのデータ構造をどのように反映するか。また反映されたデータに対して、どのようにオブジェクト単位で扱えるかはまだ解決すべき課題である。そのデータ構造を反映するには、マップされた仮想アドレス空間にオブジェクトの割り当て及びその参照関連を保持するため、オブジェクトのオフセット、サイズ及び参照のオフセットなどからなるオブジェクト参照テーブルを構築することが考えられる。だが、このようなオブジェクト参照テーブルはかなりの記憶容量が必要があり、避けられないオーバヘッドが存在する。また、オブジェクト参照テーブルを利用し、オブジェクト単位でデータを扱うことは非常に困難である。オブジェクト単位でデータを扱うには、オブジェクトフォールト、オブジェクトのデータリクエスト、オブジェクトのロックリクエストなどのオブジェクト単位のプリミティブが必要である。しかし、Machには、ページフォルト、ページのデータリクエスト、ページのロックリクエストなどのページ単位での仮想メモリ管理メカニズムしか提供していない。そのため、オブジェクト参照テーブルにより、ページフォールトをオブジェクトフォールトへ、ページデータリクエストをオブジェクトデータリクエストへ、ページロックリクエストをオブジェクトロックへ効率的に変換することが可能であるかどうかは大きな問題であろう。要するに、Machのように、オペレーティングシステムが仮想メモリ管理をユーザレベルに開放することは、データベースシステムに非常に役立つことである。しかし、データベースシステムでのオブジェクト単位でのデータ管理とオペレーティングシステムでのページ単位での仮想メモリ管理との間の関係をどのようにおりあいをつけるかは課題である。

もう一つの大きな問題は、分散計算機システム環境上でのトランザクション管理の通信コストの問題であろう。分散環境上で分散化されたデータのコヒーレンスを保持するには、遠隔データのロック及びデータ伝送のため、かなりの通信コストが存在する。この通信コストを減らすにもオブジェクト単位でのコヒーレンスが必要になる。また、そのデータ伝送は、TCP/IPに基づいたRPC(Remote Procedure Call)により実現されるため、大容量のデータ伝送するにはUDP/IPと比べると性能面で非常に不利になる。従って、データ伝送のメッセージを他のメッセージと区別して、もっと効率良いプロトコルで実現するほうが考えられる。