データフロー概念に基づく関係データベースシステムとその問合せ処理†

馬場敬信†† 斉藤直樹†††

本論文では，関係データベースシステムにおけるデータの流れに着目し，基本機能を定義してデータフローモデルを構築する．次に，より応答時間の短いデータフローを選択するため，データフローの実行時間の評価法と，評価結果を用いてデータフローを選択する手順を示す．さらにデータフロー観念に基づくデータベースシステムの有効性を明らかにするため，ワークステーション上に関係代数演算・インデックス処理など10種類の基本機能を含む処理系を試作し，一変数・二変数の問合せを用いて種々の実験を行った結果を示す．実験によって，本概念に基づくデータベースシステムが正しく動作すること，ならびに処理時間の算出法の妥当性を確かめるとともに，(1) 最適化によって選ばれたデータフローの実行時間が，同一計算機上で同一のファイル構造をもつ既存の処理系の実行時間に比べ35.6%短くなる，(2) プログラムサイズは既存のものと比べ23.5%の削減となる，(3) 問合せ，データ量，選択率などに応じて最適なデータフローが変化し，特にデータベースサイズが大きくなるにつれ，インデックスの使用やソートマージジョインを基本とするデータフローが有利となる，などを明らかにした．

1. はじめに

関係型のデータベースシステムは，データ独立性に優れ，またデータベースを平坦な表としてユーザに提供すると共に，これに対する非手続き的な問合せを可能とするなど，優れた特徴を持っている[1]．このため多くのデータベースシステムにおいて関係モデルが採用されると共に，最近ではこれを基本に知識ベースあるいは統計データベースなどへと発展しつつある[2),3)]．

このような発展と共に，高水準のユーザインタフェースを支える関係データベース演算の処理効率が悪いことが指摘されており，計算機アーキテクチャの立場から並列処理の活用を図るデータベースマシンの研究[4),5)]やインデックス等の補助データの活用を図る研究などが幅広く行われている．

このような背景のもとに，我々は，先に関係データベースシステム内部の処理が，基本機能とその間のデータフローによってモデル化できることを示し，この基本機能を多重プロセッサに分散する方法を示した[6)~8)]．機能の分散化による並列処理の効果は大きいが，この方法は多重プロセッサを前提としているため当然ながら通常の単一プロセッサには適用できない．このため，我々は一つの問合せに対して種々のデータフローの構成の可能性があることに着目し，実行前に各データフローの実行時間を評価することにより，最適なデータフローを選んでこれに沿ってデータベースをアクセスする方法を考案し，単一プロセッサ上で実現した．

このような実現法は，単に処理性能面の利点ばかりではなく，モジュール単位で独立であるためシステムの拡張や変更の容易性，新たなアルゴリズムの取り込み，デバッグや保守の容易性[9)]などの利点が考えられる．また，ネックとなりやすい機能をファームウェア化・ハードウェア化する[10)]際にも対応しやすい．

類似の研究として，問合せ木の解析に基づく最適化[17),18)]あるいはデータフローモデルのデータベース処理への応用[13),19)]などの研究がある．これらの研究に比して，本研究の特徴は，独自のデータフローおよびそのコスト（実行時間）モデルを定義し，このモデルの上での最適なデータフローの決定法を明らかにすると共に，この概念に基づくデータベースシステムを実際に構築することにより，既存のデータベースシステムとの比較において，実験・実証的に単一プロセッサにおけるデータフロー・データベースシステムの有効性を明らかにする点にある．

本手法の発想は非常に簡明ではあるが，これを実際に有効な方法として確立するには，(1) 基本機能を中心としたデータフローモデルを定義し，これをソフトウェアのモジュールとして実現した場合，システム全体が問題なく動作すること，(2) 基本機能の処理時間を予測し，これをもとにデータフロー全体の実行

† Data Flow-Based Relational Database System and Its Query Processing by TAKANOBU BABA (Department of Information Science, Faculty of Engineering, Utsunomiya University) and NAOKI SAITO (Systems Technology Department, NRI & NCC Co., Ltd.).

†† 宇都宮大学工学部情報工学科
††† (株)野村総合研究所技術部

時間の評価が正しく行えること，および，(3) 評価の結果選択されたデータフローが，既存のシステムと比べ十分良好な性能をもつこと，などを明らかにする必要がある．

このため我々は，ワークステーション上で実際に基本機能を作成し，この基本機能を組み合わせて種々のデータフローを実現できるデータベースシステムを試作した．さらに，このシステムを用いて最適化の効果を実験的に確かめた．

以下，本論文においては，データフローモデルとそれに基づいて構築したシステム，および試作システムを用いた実験結果について述べる．

2. 基本概念

データベースシステムに入力された問合せは，通常いくつかのサブシステムによる処理を経て，ユーザへの出力となる．例えば，INGRES[11]は，(i) ターミナル・モニタ，(ii) 構文解析，(iii) コマンド・コントローラ，問合せ最適化，実行，および (iv) データ定義，データ操作，の4つのプロセスから成る．System R[12]における RDS (Relational Data System) と RSS (Research Storage System) はそれぞれ，(i) 構文解析，コマンドコントローラ，問合せ最適化，実行，選択，直積，射影，(ii) データ定義，データ操作，インデックス/レコード操作，マージ，整列，結合，などの処理を含む．したがって，データベースシステムにおける問合せ処理はこれらの処理間のデータの流れとしてモデル化することができる[6),7)]．

この考え方に基づいて，本論文において我々の提案する処理系の概念図を図1に示す．まず，問合せは，構文解析により問合せ木に変換される．

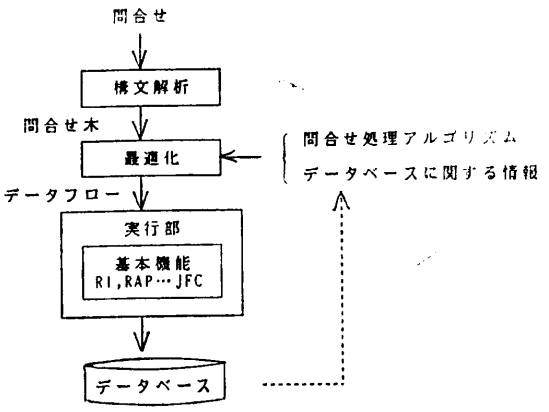


図1 処理系の概念図
Fig. 1 System configuration.

次に最適化のフェーズにおいて，問合せ木からデータフローが生成される．データフローの生成に当っては，問合せの処理アルゴリズムと使用可能な基本機能を考慮する．ここでは，代表的な処理アルゴリズムとしてネステッドループ結合 (NLJ) とソートマージ結合 (SMJ) を対象とする[13)]．この2つのアルゴリズムにおいて，インデックスの有無を考慮して，可能なデータフロー (通常は複数) を生成する．

次にこのデータフローから最適と思われるものを選択するが，ここでは最適性の規範として，データフローを与えてから結果が得られるまでの時間 (応答時間) の最小化を目標とする．このため，データフローを構成する基本機能の単位データに対する処理時間を実験的に求めると共に，データベースに関する情報 (データベースのタプル数，アトリビュート数，制約における選択比など) を用いて，応答時間を算出する方法を求める．

求められたデータフローは実行部に渡され，ここでデータフローによって与えられた順番に基本機能を起動することにより，データベースへのアクセスを行う．

3. 関係データベースシステムにおける基本機能とその組合せ

3.1 基本機能の定義

基本機能の定義に当っては，

(1) 対象とする NLJ および SMJ の両アルゴリズムを実現するための機能を含むこと，

(2) インデックスの処理機能を含むこと，

の2点を条件とした．(2)は，全数探索と共に，インデックスによるアクセスの可能性を考慮することにより，データフローのモデルをより現実的なものとするための方針である．

これらの条件を満たす範囲で選択した基本機能を表1に示す．各基本機能を節点に対応させ，その入力・出力を明らかにした．以下，機能ごとに簡単に説明する．

(1) インデックス処理 (RI, JI, I)

RI (Restriction Index) は，一変数に対する制約条件に基づいて，インデックスのアクセスを行い，条件を満足するタプルへのポインタ (タプルポインタ) を出力する．JI (Join Index) は，二変数の結合のためのインデックスをアクセスする．I (Intersection) は，2つのタプルポインタを入力として，その共通部分を求める．

表 1 基本機能の定義
Table 1 Definition of basic functions.

入力	基本機能	出力
制約条件	-> RI ○ ->	タプルポインタ
アトリビュート値	-> JI ○ ->	タプルポインタ
タプルポインタ（2入力）	-> I ○ -> ↑	タプルポインタ
タプルポインタ	-> RAP ○ ->	タプル値
関係名	-> S ○ ->	タプル値
タプル値	-> P ○ ->	タプル値
タプル値	-> RF ○ ->	タプル値
タプル値（2入力）	-> NLC ○ -> ↑	タプル値
タプル値（2入力）	-> JFC ○ -> ↑	タプル値
タプル値	-> ST ○ ->	タプル値

⟶ タプル　⟶ ポインタ　⇢ パラメータなど

インデックスに関しては，RI に与えられる制約条件によって関係全体のうちどの程度が選択されるか，あるいは JI に対する結合条件によって関係全体のうちどの程度が選択されたかを表す量が必要となり，これを選択率 (selectivity) と呼ぶ[7]．

（2） レコードアクセス (S, RAP)

データベースからタプルをアクセスする．S (Scan) は，関係のタプルをファイルに格納された順にアクセスする．RAP (Record Access and Projection) は，タプルポインタを入力として，タプルをアクセスする．このとき，指定されたアトリビュートのみをアクセスすることにより，射影を同時に行う．

（3） 関係代数演算 (P, RF, NLC, JFC)

いずれも，ある条件に従って1つの関係から別の関係を作り出す機能である．P (Projection) は，必要なアトリビュートのみを取り出して新たな関係とする．RF (Restriction Filter) は，制約条件に従ってタプルの選択を行う．NLC (Nested Loop Concatenation) と JFC (Join Filter Concatenation) は，2つの関係を共通のアトリビュートをもとに結合する．NLC が，2つの関係からのタプルの総当りにより結合を行うのに対し，JFC は，整列された2つの入力をマージしながら結合を行う．RF によって選ばれる割合，ならびに NLC と JFC の結合の成功の割合を表す量として，RI, JI 同様選択率が必要になる．

（4） その他の機能 (ST)

ST (Sort) は，指定されたアトリビュートの値によってタプルの整列を行う．

3.2 基本機能ごとの処理時間

データフローの実行時間を評価するためには，各基本機能の処理時間を見積る必要がある．

特にデータベースを対象とした処理では，各基本機能には，同一の性質（長さ，選択率など）をもったタプルやタプルポインタが繰り返し入力される．このため，個々のデータに対する処理時間の変動は全体としてならされ，処理するデータの量や，計算機システムの性能から処理時間の予測が可能となる[6]．

各基本機能には一般的にデータ量に依存して処理時間の変化する部分と，依存せず一定の値を持つ部分とがある．後者の例は，CPU においては，各基本機能における初期設定部分などデータ量が仮想的に0でも実行する必要のある部分を指し，ディスクアクセスでは，データ・ディクショナリ，ディレクトリのアクセス時間や逐次スキャンにおける最初のシーク・レーテンシ時間などを示す．

したがって，ある基本機能 f の処理時間 $C(f)$ は

$$C(f)=C_0(f)+C(f,n) \qquad (1)$$

と表される．右辺第1項がデータ量に依存しない部分，第2項が依存する部分を表す．n はタプル数，ポインタ数などデータの量を表す．

$C_0(f)$ の値，および $C(f,n)$ の式の定義は，f を実現するプログラムの内部構造や，データベースシステムを実現した計算機システムの性能に依存するため，各 f の作成後に実験を行って定める必要がある．

さらに，n の決定のためには，処理対象とするデータベース全体の量のほか，各基本機能について先に述べた選択率を考慮する必要がある．例えば，整列 (ST) では，入力と出力のデータ量は変わりなく，次の基本機能へと送られるが，制約 (RF) では，入力データから一定の選択率で選ばれたものが出力として次に送られることを考慮して，次の基本機能へのデータ量を決定する必要がある．

3.3 基本機能の組合せによるデータフローの構成

表1中に示したように，基本機能を節点で表し，基本機能間のデータの流れを方向を持った枝で表すと，問合せに対する処理の過程は有向グラフとして表現できる．

基本機能間は，原則的には送り出す側の機能の出力

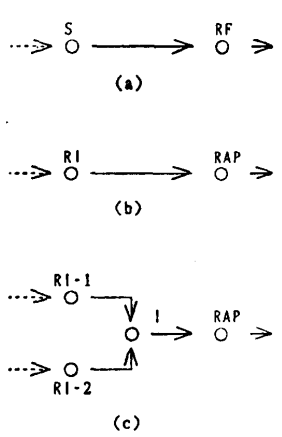


図 2 一変数に対するデータフロー
Fig. 2 Data flows for one-variable query.

データの形式と，受け取る側の入力データの形式が一致すれば，接続することができる．例えば，RI の出力はタプルポインタであるから，これに接続することができる基本機能は I, RAP となる．

したがって，与えられた問合せをもとに基本機能を組み合わせてデータフローを構成する際には，一般に種々の可能性がある．以下，これを一変数の場合と二変数の場合について検討する．

（1） 一変数問合せに対するデータフロー

制約のない場合には明らかに逐次的なスキャン (S) しかない．制約がある場合には，スキャン (S) 後に，RF によって制約を行う (**図 2**(a))か，あるいはインデックスのアクセス (RI) 後にタプルのアクセスを行う (RAP) (図2(b))かのいずれかである．もし，複数のインデックスを持つ場合には，図2(c)に示すように，複数の RI を行いその出力の共通部分を取って (I) から，タプルをアクセスすることもできる．

（2） 二変数問合せに対するデータフロー

二変数の場合には，一変数の問合せに対する可能性に加えて，(i) NLJ と SMJ のどちらのアルゴリズムを基本とするか，(ii) 結合インデックス (JI) を使用するか否か，によってさらにデータフローの数が増える．

今，簡単のために一変数のアクセス部分は図2(b)の RI+RAP に固定すると，(i)，(ii) の組合せによって，**図 3**(a)～(d)の4通りのデータフローが生成される．

図3(a)は NLJ アルゴリズムを実現している．まず，V1 と V2 の2つの変数ごとのアクセス部分が RI-1 と，RAP-1，および RI-2 と RAP-2 の組合せであり，それぞれインデックスのアクセス後に，タプルポインタによって目的のタプルをアクセスしている．それぞれの関係からアクセスされた2つのタプルの集合を入力としてネステッドループにより結合を行い (NLC)，結合されたタプルに射影 (P) を施して出力とする．

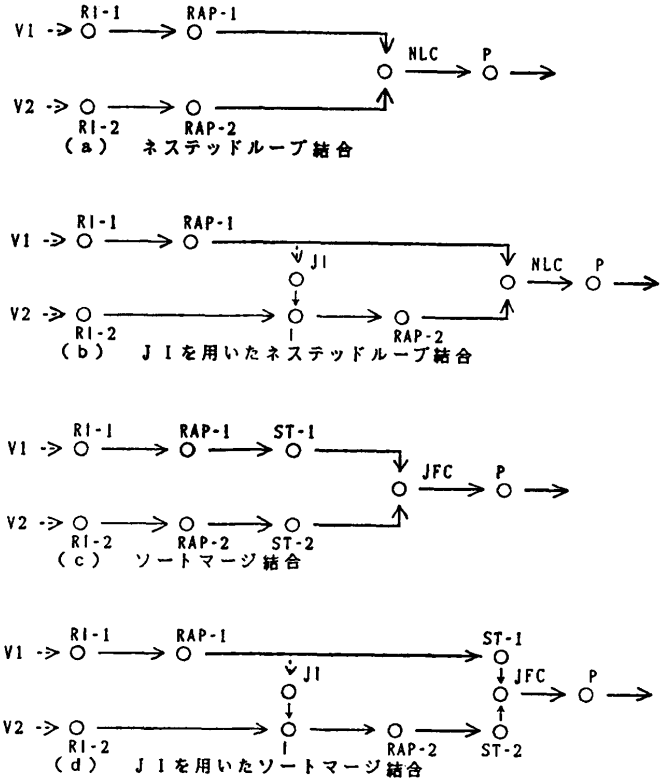


図 3 一変数に対するデータフロー
Fig. 3 Data flows for two-variable query.

図3(b)は，結合インデックスの使用を表している．(a)と異なる点は，RAP-1 でアクセスされたタプル (T1 とする) から結合アトリビュートが JI に送られる点である．JI は，これを入力として，V2 において結合条件を満足するタプルへのポインタを生成する．このタプルポインタと RI-2 でアクセスされたポインタとの共通部分を I によって求める．したがって，I の出力となるタプルポインタは，T1 と結合でき，かつ RI-2 での制約条件も満足するタプルへのポインタである．このポインタ値に従って，アクセスした V2 のタプルと V1 のタプルを NLC により結合し，必要なアトリビュートを P によって選んで出力とする．

図3(c)は，SMJ アルゴリズムを実現している．各変数に対するタプルをアクセスした後，結合アトリビュートについて整列をする．したがって，JFC では，整列された結合アトリビュート同士を比較して結合を行う．

図3(d)は，JI を用いた SMJ アルゴリズムである．JI の使い方は図3(b)と同一である．

以上，二変数問合せに対するデータフローについて述べたが，これらはあくまでも骨組みを示すものであり，表1の機能に限定してもこのほかにも多くの可能性がある．例えば，V1に対する制約条件が RI-1 以外にもあればタプルのアクセス後（すなわち，RAP-1の右側）に RF による制約条件のチェックが必要となる．一変数のアクセス部については，RI+RAP とした部分を図2に示すデータフローの任意のものと置き換えることが可能である．

3.4 データフローの最適化

3.3 節で述べたように1つの問合せに対するデータフローには一般に多くの可能性があり，このため，データフローごとの応答時間を見積り，最小の応答時間を持つデータフローを求める必要がある．

あるデータフローFが与えられたとき，その処理時間Rは，Fを構成する基本機能ごとの処理時間の和として表される．すなわち，(1)式を用いて

$$R=\sum_{f_i\in F} C(f_i) \qquad (2)$$

となる．したがって，多くのデータフローの可能性がある場合には，各々に対してRを計算し，最小のRを示すデータフローを選択すればよい．しかしながら，可能なデータフローの全体を尽くし，その処理時間を比較することは，十分な計算時間を与えられた場合には可能な方法である（例えば，同じ問合せを同じデータベースに繰り返し適用するような場合）が，一般には，問合せを与えられてからデータフローを決定し，その結果に基づいて実行することが考えられ，この場合には何らかのヒューリスティックスな手法が必要となる．

我々は，一変数，二変数の問合せに対して次のようなデータフロー決定の手順を定義した．

(1) 一変数の問合せ
- (i) 制約条件がない時，Sを用いる．
- (ii) 制約条件がありインデックスを使えない時，S+RF を用いる（図2(a)）．
- (iii) 制約条件があり，インデックスが使える時，S+RF と RI+RAP（図2(a)，(b)）の処理時間を比較して，より短いものを選ぶ．

(2) 二変数の問合せ
- (i) 二変数を独立した2つの一変数の問合せに分割し，それぞれに対して(1)による最適化を行う．
- (ii) 結合のためのアルゴリズムの選択を行うため，NLJ アルゴリズムに必要な基本機能 NLC（図3(a)）と SMJ に対する ST+JFC（図3(c)）の比較を行う．さらに，JI がある場合には，JI+I を挿入した場合（図3(b)，(d)）との比較を行う．
- (iii) (i)と(ii)で選択したデータフローを結合して，全体のデータフローを完成する．

本手順中，特に二変数については，これを一変数の問合せに分解してそれぞれの中でまず最適化を行うことにより，処理時間の減少を図っている．

4. 基本機能の実現

4.1 設計・試作の方針

2章および3章で述べた方法の有効性を確かめるために，我々は，この考え方に基づくデータベースシステムを実際に作成した．試作に当っては，我々の基本的な考え方の検証に的を絞り，基本機能を実現して，それを自由に組み合わせてデータベースの処理ができる環境を作ることを目標とした．

このため，我々の手元にある小規模関係データベースシステムである XDB[14]をベースに，

(1) ファイルの構造は，XDB と共通とする．

(2) 基本機能の試作に当っては，ファイルの読込みや文字列の比較など，低レベルのルーチンを除き，3章の定義に基づくモジュールを全く新たに作成する．

の2つの方針のもとに，基本機能を試作した．試作には，ワークステーション Ustation-E 15*[15]を使用した．ハードウェアの主な構成は，CPU が 68000 (10 MHz)，主記憶は 3 MB，ハードディスク 160 MB (ウィンチェスタ型，平均アクセス時間 20 ms，転送時間 1229 KB/S) である．OS は UNIX SYSTEM V をベースにした Uniplus+** であり，プログラムの作成にはCを使用した．

試作した基本機能の全体は，Cで約 2300 行，オブジェクト・モジュール全体の規模は 31 KB である．

---

* Ustation は，ディジタルコンピュータ(株)の登録商標である．
** Uniplus+ は，米国 Unisoft 社の登録商標である．

4.2 データ構造

基本機能の実現のために必要となったファイルの構造，および基本機能間のデータ受渡し領域のデータ構造について述べる．

（1） ファイル構造

ファイル全体は，（i）インデックスをB-トリー形式で格納したインデックスファイル，（ii）インデックス値に対応するタプルポインタが複数個あるとき，これを格納するタプルポインタファイル，（iii）関係に関する情報（関係名，タプル長，アトリビュート数など）を納めた関係ファイル，（iv）アトリビュートの情報（アトリビュート名，アトリビュート長，アトリビュートのタプル内オフセット，アトリビュート値のデータ型，インデックス名，インデックスファイル識別子，タプルポインタファイル識別子）を納めたアトリビュートファイル，および（v）タプルそのものを格納したデータファイル，の5つから構成される．

これらは，前述のように通常のデータベースシステムのファイル構造と同様である[14]．

（2） 基本機能間の受渡しデータの構造

基本機能間のデータの受渡しをする領域を転送領域と呼ぶ．転送領域のデータ構造の決定に当っては，基本機能同士が任意に組み合わされても問題のないよう，受渡しデータと共にそのデータに関する情報を転送領域自身に持たせるようにした．

転送領域のデータ構造をCの型宣言文で記述したものを図4に示す．全体は，データ領域 (data [ ]) とヘッダ部 (data [ ] 以外)から成る．データ領域には，タプルかタプルポインタが置かれるが，これを識別するのが dtup_flag である．タプルポインタを出力とする基本機能 RI, JI, I（表1参照）がこれを1にセットする．

ヘッダ部の dtup_flag 以外の部分には，関係ファイルとアトリビュートファイルから得た情報や転送領域上のタプル数（またはタプルポインタ数），アトリビュート数に関する情報を持つ．

```
struct transfer
 {
  int                dtup_flag;      /*ﾃﾞｰﾀがﾀﾌﾟﾙﾎﾟｲﾝﾀか、
                                       またはﾀﾌﾟﾙかｷ/
  int                tup_len;        /*ﾀﾌﾟﾙ長ｷ/
  int                tup_count;      /*転送領域上のﾀﾌﾟﾙ(ﾀﾌﾟﾙﾎﾟｲﾝﾀ)
                                       の数ｷ/
  int                att_count;      /*転送領域上のｱﾄﾘﾋﾞｭｰﾄの数ｷ/
  struct attr_head   in_att[];       /*ｱﾄﾘﾋﾞｭｰﾄに関する情報ｷ/
  char               data[];         /*ﾃﾞｰﾀ領域ｷ/
 }
```

（a）

```
struct attr_head
 {
  int                del_flag;       /*転送領域上のｱﾄﾘﾋﾞｭｰﾄが有効
                                       かどうかｷ/
  char               att_name[];     /*ｱﾄﾘﾋﾞｭｰﾄ名ｷ/
  int                att_len;        /*ｱﾄﾘﾋﾞｭｰﾄ長ｷ/
  int                att_off;        /*ｱﾄﾘﾋﾞｭｰﾄのｵﾌｾｯﾄｷ/
  int                att_type;       /*ｱﾄﾘﾋﾞｭｰﾄの型ｷ/
 }
```

（b）

図 4 転送領域のデータ構造
Fig. 4 Structure of data to be transferred.

4.3 基本機能

基本機能は，それぞれCの手続きとして実現されている．ここでは，表1に示した順番に，（i）各基本機能の手続き呼び出しの形式を示し，入出力データや入出力パラメータを明らかにすると共に，（ii）各手続き中での処理の要点を示す．引数において，source と destination は基本的に入力および出力の転送領域を表すが，入出力領域を共有できるときは，source に出力が置かれることもある．

（1） RI (destination, attribute_names, comparator, constant)

attribute_names を用いてアトリビュートファイルを検索し，インデックスファイルに関する情報を得た上で，comparator, constant によって表される制約条件を満足するタプルへのポインタをインデックスファイルとタプルポインタファイルよりアクセスする．アクセスしたタプルポインタを destination 転送領域に置き，制約条件を満足するタプルが複数あるときは，タプルポインタファイルを探す．同時に，タプルポインタの数をカウントする．そして，転送領域のヘッダ部の dtup flag を1にして，タプルポインタ数を設定する．

（2） JI (source, destination, attribute_name 1, attribute_name 2)

source には，制約する側のタプルが置かれている．ここからタプル T1 を1つ取り出し，図5のように，結合する相手の関係の attribute_name 2 が等しいものへのタプルポインタを見いだし，destination に置くのが JI の機能である．このため，attribute_name 1 の値（図では A）をもとにインデックスファイルを検索し，RI と同様にタプルポインタを destination 転送領域上に出力する．

（3） I (source 1, source 2)

source 1, source 2 中のタプルポインタ

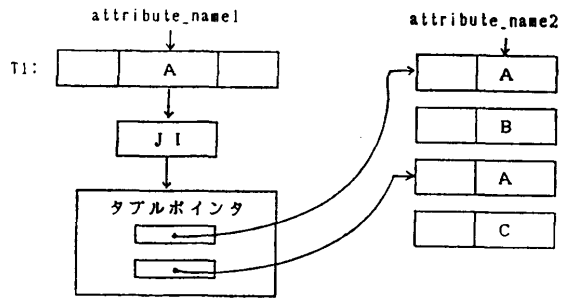


図 5 JI の機能
Fig. 5 Function of join index.

を整列し，それぞれの中で重複するものを除く．source 1 中のタプルポインタで，source 2 のタプルポインタと一致するものだけを残し，そのポインタ数を source 1 の転送領域ヘッダにセットする．

（4） RAP (source, relation_name, attribute_names)

source にはタプルポインタが置かれている．relation_name をもとに関係ファイルをアクセスする．この結果と attribute_names から読み込むべきアトリビュートの情報を得る．これらの情報とタプルポインタとから必要なアトリビュートのみ（この部分が P (Projection) に相当）を source 転送領域に読み込む．また，読み込んだタプルのタプル長，タプル数を計算し，転送領域のヘッダに書き込む．

（5） S (destination, relation_name)

relation_name をもとに関係ファイルをアクセスし，これをもとにデータファイルより対象とする関係の全タプルを読み込み，destination にセットする．

（6） P (source, attribute_names)

アトリビュートヘッダの del_flag（図 4 (b)）を変更して，attribute_names で指定されたアトリビュートのみを有効とする．

（7） RF (source, rest_pred)

rest_pred に指定された制約条件を満足しないタプルを source 転送領域から探し，そのタプルの削除フラグを立てる．そして，削除されずに残ったタプルの数を転送領域ヘッダに書き込む．

（8） NLC (source 1, source 2, destination, attribute_name 1, attribute_name 2)

source 1, source 2 の転送領域ヘッダから，結合後のタプルを置く destination 転送領域のヘッダを設定する．attribute_name 1, attribute_name 2 によって指定されるアトリビュート値を比較し，一致すれば結合して destination に書き込む．ここでは，source 1 と source 2 の全タプルを総当りによって比較する．そして，結合して生成されたタプル数を destination 転送領域ヘッダの tup_count（図 4 (a)）にセットする．

（9） JFC (source 1, source 2, destination, attribute_name 1, attribute_name 2)

基本的な動作は NLC と同じである．ただし，source 1, source 2 中のタプルは，それぞれ指定されたアトリビュートについて整列されているので，先頭から相互に比較しながら結合すれば良い．

(10) ST (source, attribute_name)

attribute_name により指定されたアトリビュートについて，source 中のタプルを整列する．ここでは，Batcher による整列法を使用した．

4.4 基本機能のモジュールサイズ

試作した基本機能のモジュールサイズを表 2 に示す．インデックス処理，レコードアクセス，関係代数演算の中の結合処理などが大きく，他と比較して複雑な処理を行っていることを示している．

モジュールサイズを単純加算すると 55.5 KB であるが，モジュール間の共通ルーチンをリンクして使用する場合には，そのサイズは，31 KB となる．共通のファイル構造，およびプリミティブなルーチンを使用する XDB システムのサイズは 40.5 KB であり，これと比較すると我々の作成したシステムは 23.5% の削減となっている．XDB が NLJ アルゴリズムだけを実現していることを考慮し，我々の処理系で対応する部分だけ (SMJ アルゴリズムで用いる基本機能 ST, JFC を除く）を取り出すと 26 KB となり，35.8% の削減となっている．これは，主として XDB が複雑な制御構造を持っており，モジュール間の関係も整理されていないため，冗長なシステムになっているためである．我々のシステムにおいて，モジュール単位に機能を整理して実現した効果がここにも表れている．

5. 実　　験

5.1 問合せの実行

試作した基本機能を用いて実験を行うため，処理対

表 2 基本機能の大きさ
Table 2 Size of basic functions.

単位：KB

	インデックス処理			レコードアクセス		関係代数演算				その他
基本機能	RI	JI	I	RAP	S	P	RF	NLC	JFC	ST
サイズ	8	7	2.5	7	7.5	2	4.5	5.5	7	4.5

```
player
      *pname char(20)   選手名
       height numeric   身長
      *weight numeric   体重
field
       position char(15) 守備位置
      *fpname char(20)  選手名
                  *：インデックス有り
```

図 6 関係の定義
Fig. 6 Definition of a relation.

象とするデータベースを図 6 に示すように定義した．野球選手に関する 2 つの簡単な関係でアトリビュート pname, weight, および fpname はインデックスを持つ．定義に基づいて 4.2 節(1)に述べた 5 つのファイルを作成した．データファイル中のタプル数は各関係について 72 である．

定義した関係に対する 3 つの二変数の問合せを次のように定義した．記述は，SEQUEL[1]による．

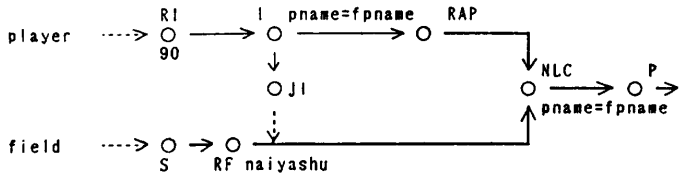
[問合せ 1] 体重が 75 kg 以上の選手に関する全データ（名前，守備位置，体重，身長）を求める．

```
select pname, position, weight, height
from player, field
where weight>75 and pname=fpname;
```

[問合せ 2] 体重が 80 kg 以上の選手の全データを求める．

```
select pname, position, weight, height
from player, field
where weight>80 and pname=fpname;
```

[問合せ 3] 体重が 90 kg 以上で，守備位置が naiyashu の選手の全データを求める．



(a)結合インデックスを用いたネステッドループ結合

```
RI(&rel1,weight,>,90);
 S(&rel2,field);
RF(&rel2,position=naiyashu);
JI(&rel2,&temp,"pname","fpname");
 I(&rel1,&temp);
RAP(&rel1,player,(pname,height,weight));
NLC(&rel1,&rel2,&temp,"pname","fpname");
 P(&temp,(pname,position,height,weight));
```

(b)基本機能の呼び出し列による実行

図 7 データフローの実行
Fig. 7 Execution of a data flow.

表 3 基本機能の処理時間
Table 3 Execution times of basic functions.
単位：ms

基本機能 f	$C(f)$
S, RAP	$70.8+0.061 \cdot t \cdot b$
RF	$0.053 \cdot t \cdot c$
P	$0.0012 \cdot t$
ST	$0.02 \cdot (t^2+20t)$
RI, JI	$224.0+6.7 \cdot r$
I	$14.6+0.4 \cdot p$
JFC	$12.0+0.0012 \cdot \max(t1, t2) \cdot c+0.013(t1 \cdot t2) \cdot s \cdot b$
NLC	$12.0+0.0012 \cdot (t1 \cdot t2) \cdot c+0.013 \cdot (t1 \cdot t2) \cdot s \cdot b$

$t, t1, t2$：入力タプル数，b：タプル長，c：比較バイト数，r：制約条件数，p：ポインタ数，s：選択比

```
select pname, position, weight, height
from player, field
where weight>90 and position="naiyashu"
      and pname=fpname;
```

これらの問合せに対して，3 章で述べたように最適化されたデータフローを生成し，これを実行する．本研究では基本機能による実行効率の評価に重点を置き，最適化部分の試作までは行っていないため，データフローの生成は，3.4 節に述べた手順に従って人手により行った．

例えば，[問合せ 3] に対して，結合インデックスを用いた NLJ アルゴリズムによるデータフローが生成されたとすると，これは，例えば図 7(a)のようになる．これを実行するには，先に 4.3 節で述べた基本機能に対する手続きを図 7(b)のように連続して呼び出せばよい．実行時間の計測には，Uniplus+ のシステムコールである times を手続きの呼び出し間に挿入し，3 回の実行を繰り返してその平均を求めた．

5.2 基本機能の処理時間

最適化の実験に先立ち，(1)式で定義した基本機能の負荷を決定しておく必要がある．このため，我々はまず 4.3 節で述べた各基本機能ごとの処理アルゴリズムから処理の手数が何によって決まるかを推定し，基本機能ごとに式を立てたうえで，処理時間の実測結果と照合して，式の妥当性や係数の決定を行った．表 3 に処理時間の一覧を示す．

例えば，逐次的なスキャン S については，関係ファイル，アトリビュートファイルをまずアクセスして転送領域のヘッダ部を作成した後，ディスクを連続的にアクセスすること

から，処理時間の増加は全体のデータ量に比例すると考え，その処理時間を

$$C_0+C_1\cdot t\cdot b \qquad (3)$$

(t: タプル数　b: タプル長)

としたうえで，実測結果から，$C_0=70.8$ ms，$C_1=0.061$ ms と決定した．他も同様であるが，整列 (ST) については，いわゆる理論的な比較回数 $t\cdot\log_2 t$ では近似できず，実測結果をもとに2次の近似式を定めた．また，JFC と NLC については，(1)式の第2項に当る部分がさらに比較（第2項）と結合（第3項）の2つの処理時間の和となっている．

5.3 最適化の適用例

3.4 節に述べた最適化の手順の適用例を，一変数と二変数の問合せを用いて示す．なお，処理時間の単位は ms である．

(1) 一変数問合せの最適化

問合せ例として「kiyohara 選手の身長，体重」を求める次の問合せを使用する．

```
select height, weight
from player
where pname="kiyohara";
```

最適化の手順 (3.4 節(1)) に従って，データフローとその処理時間を検討する．ここでは，制約条件があり，かつ pname に対してはインデックスがある (図6) ため，図2(a),(b)に示した2つのデータフロー，すなわち，S+RF の処理時間 ($R1$) と RI+RAP の処理時間 ($R2$) の比較を行う．このため，まず表3を用いて各 $C(f)$ を計算する．例えば，

$$C(S)=70.8+0.061\cdot 72\cdot 28=193$$

となる．ただし，タプル数 $t=72$，タプル長 $b=28$ を使用している．同様に $C(\mathrm{RF})=76$，$C(\mathrm{RI})=230$，$C(\mathrm{RAP})=73$ が求められる．

以上の計算結果から，(2)式を用いて $R1=269$，$R2=303$ となり，データフローとしては，S+RF が選ばれる．

なお，表3の定義に基づく $R1$，$R2$ の計算値の妥当性を検証するため，2つのデータフローを実際に実行して処理時間を計測した．この結果 $R1$，$R2$ に相当する実測値の値はそれぞれ 266，283 となり，計算値に近い値となった．

(2) 二変数問合せの最適化

先の［問合せ3］の体重に対する条件を「90 kg 以上」から「80 kg 以上」に変更した問合せを用いる．

最適化の手順3.4節(2)を適用する．

(i) まず，次の2つの一変数の問合せに分割する．

```
[問合せA]          [問合せB]
select             select
from player        from field
where weight>80;   where position="naiyashu";
```

それぞれについて，上記(1)に示した最適化の手順を適用する．weight についてはインデックスがあるので(1)同様 S+RF と RI+RAP の比較となるが，position についてはインデックスがないので S+RF の組合せ1通りだけとなる．結果として，問合せA，Bの処理時間 $T1$, $T2$ は次のようになる．

$$T1=\min(C(\mathrm{S})+C(\mathrm{RF}),\ C(\mathrm{RI})+C(\mathrm{RAP}))$$
$$=\min(269, 313)=269 \qquad (4)$$
$$T2=C(\mathrm{S})+C(\mathrm{RF})=282 \qquad (5)$$

(ii) 二変数を結合するため，NLJ と SMJ アルゴリズムの実現に要する処理時間を比較する．さらに，ここでは関係 player のアトリビュート pname は結合インデックスを持つため，それぞれのアルゴリズムに対して JI を使用した場合と使用しない場合を比較する必要がある（図3参照）．

したがって，JI を用いない NLJ，JI を用いた NLJ，JI を用いない SMJ，JI を用いた NLJ の各処理時間 $R1$，$R2$，$R3$，$R4$ を計算して比較する．

$$R1=C(\mathrm{NLC})+C(\mathrm{P})$$
$$R2=C(\mathrm{JI})+C(\mathrm{I})+C(\mathrm{NLC})+C(\mathrm{P})$$
$$R3=C(\mathrm{ST})+C(\mathrm{ST})+C(\mathrm{JFC})+C(\mathrm{P})$$
$$R4=C(\mathrm{JI})+C(\mathrm{I})+C(\mathrm{ST})+C(\mathrm{ST})+C(\mathrm{JFC})+C(\mathrm{P})$$

ここで，NLC，JFC については選択率 0.1 を使用した．これは，事前にデータフローを実行することにより，実験的に求めた値である．計算の結果，$R1=100$，$R2=303$，$R3=332$，$R4=620$ となり $R1$ が最小となるので JI を用いない NLJ が選ばれ，その処理時間 $T3$ は，

$$T3=R1=100 \qquad (6)$$

となる．

(iii) (i)と(ii)の結果を合わせると，図8に示すデータフローとなり，その処理時間 T は(4)～(6)より，

$$T=T1+T2+T3=651$$

と計算される．

一方，計算値と比較するため，図8のデータフローを実際に実行して得られた値は633であり，近い値と

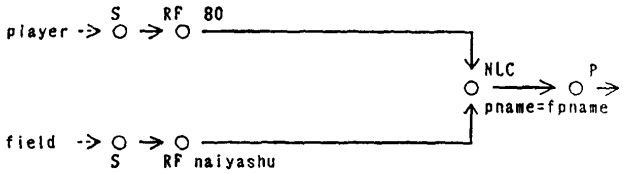


図 8 二変数の最適化で選択されたデータフロー
Fig. 8 Data flow selected by the optimization phase.

なっている．さらに，(ii) で棄却した3つのデータフローについても実測値との比較を行い計算値の妥当性を確かめている．

5.4 最適化の効果

最適化の効果を明らかにするため，同じ二変数の問合せを，既存のデータベースシステム XDB[14] と，我々が試作したシステムの両方で実行して比較した．さらに，データフローの最適化の効果を明確にするため，(A) NLJ, (B) JI 付き NLJ, (C) SMJ, (D) JI 付き SMJ の4通りのデータフローの中でそれぞれ最適化を行い，その処理時間を実測し，比較した．

表4にその結果を示す．使用した問合せは，5.1 節の問合せ1～3である．A, B, C, D の欄がそれぞれのアルゴリズムにおいて最適化されたデータフローの値であり，下線を引いた値は，その中からさらに最小なものとして選ばれたデータフローの実測値である．また，平均の欄は，A, B, C, D の平均であり，最適化の範囲を A, B, C, D のそれぞれの基本機能アルゴリズム内に留めた場合の実行時間の期待値を表している．この結果から次のことが言える．

(1) 既存のデータベースシステム XDB と比べて最適化で選択されたデータフローの実行時間は平均 35.6% 短くなっている．

(2) 最適化の範囲を各 A, B, C, D の基本機能アルゴリズム内に留めた場合の実行時間の期待値と比べて，A, B, C, D 全体を考慮すると平均 27.6%実行時間が短縮されている．

さらに，表4の A, B, C, D を比較すると，JI を使用するBとDの実行時間が長いことがわかる．この原因を調べるため，問合せ処理において基本機能がどのように使われているかを調べたのが表5である．JI の全体に占める割合が大きく，この程度のデータベースサイズではインデックスの効果が出ないこと，および RI, S, ST, NLC などの負荷が比較的大きいことなどがわかる．

表 4 最適化の効果
Table 4 Effect of optimization.

単位：ms

問合せ	XDB	(A)	(B)	(C)	(D)	平均
1	1611	1011	1561	1067	1644	1321
2	1200	711	1333	811	1310	1041
3	761	544	811	716	949	755

5.5 データ量と選択率の影響

5.1 節で定義したデータベースのサイズは小さく，最適化のアルゴリズムの検証や，計算値と実測値との比較などの目的には十分でも，データ量が大きくなった場合の影響については，別に検討が必要となる．

本節では，実測値との比較で妥当性を確かめたデータフローのモデルを使用し，これを比較的多数のタプルに適用して，データ量の変化が基本機能の選択にどのような影響を与えるかを検討する．

(1) RI+RAP と S+RF の比較

player のタプル数を $10 \sim 10^3$ まで変化させたときの両者の処理時間の変化を図9に示す．この結果から，タプル数がある値（図では100近辺）をこえると

表 5 基本機能の動的負荷の割合
Table 5 Execution time ratios for basic functions.

単位：%

	(A)	(B)	(C)	(D)
RI	34.4	20.3	31.4	18.0
S	23.5	12.7	24.6	14.1
JI	—	40.4	—	40.0
I	—	1.3	—	1.3
RAP	11.7	7.6	11.2	7.7
ST	—	—	2.1	1.3
ST	—	—	17.6	10.0
JFC	—	—	4.1	2.5
NLC	18.7	11.4	—	—
P	11.7	6.2	9.2	5.1

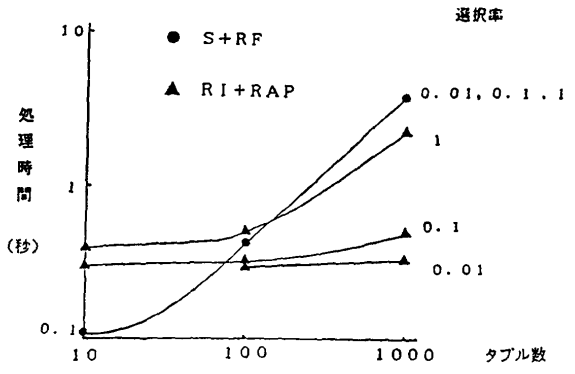


図 9 S+RF と RI+RAP の比較
Fig. 9 Comparison between S+RF and RI+RAP.

インデックスの効果がでてくること，および選択率が小さいほどその効果が大きくなるという常識的な結果が得られている．タプル数が少ない所ではインデックスアクセスの負荷が大きいため，逐次スキャンの方が良い．このことは5.3節の最適化の適用例で，タプル数 72 に対する S+RF と RI+RAP の比較を行ったとき，常に S+RF が選択された理由になっている．

（2） NLJ と SMJ の比較

NLJ と SMJ を比較するため，結合のための基本機能の処理時間を求めた．二変数の一方 (player) を 100 タップルに固定し，他方 (field) を 10～10^3 タップルとして，選択率を 1～0.01 に変化させた．結果を**図 10** に示す．

この結果，データ量が多くなると SMJ が有利となり，一般的に得られている結果と一致する[16]．この原因を基本機能の処理時間から見ると，NLJ においては，ネステッドループによる結合 (NLC) がデータ量の増加に応じて大幅に大きくなるのに対して，SMJ では，整列 (ST) が大きくなるものの JFC の処理時間はあまり増加していないことによる．

（3） JI の効果

JI の効果が大きい NLJ で2入力の一方 (player) を 100タップルに固定した．もう一方 (field) を 10～10^3 タップルとし，選択率を 1～0.01 に変化させた．結果を**図 11** に示す．

この結果から，制限されるタプル数が大きくなければ JI の効果がないことがわかる．このことは，比較的少ないタプル数を処理した結果を表す表4において

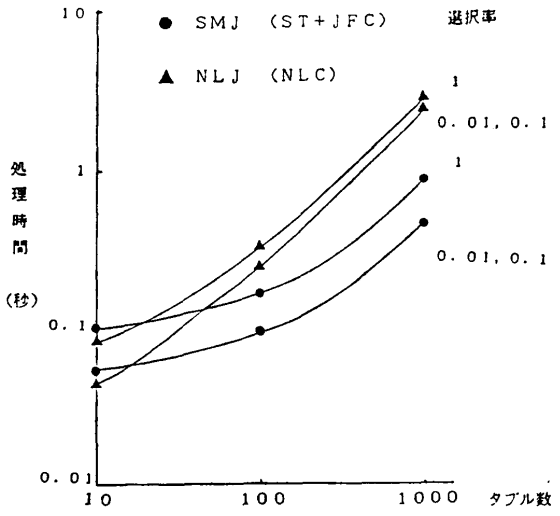


図 10 NLJ と SMJ の比較
Fig. 10 Comparison between NLJ and SMJ.

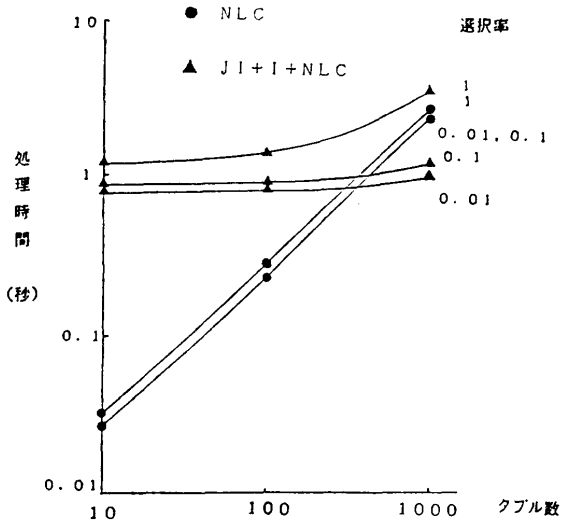


図 11 JI の効果
Fig. 11 Effect of JI.

JI を含む B, D が不利となったことと符合している．さらに，これと(2)の結果から，タプル数が増えて NLJ が SMJ より不利になるような場合に，NLJ に JI を組み合わせる方法が有効になることがわかる．

6. おわりに

関係データベースシステムにおけるデータフローに着目し，基本機能を定義してデータフローモデルを構築するとともに，その処理時間を計算して最適化を行う手順を示した．さらに，この新たな概念に基づく処理系の有効性について検討するため，ワークステーション上に処理系を試作し，これを用いて種々の実験を行った．

実験により，データフロー概念に基づくデータベースシステムが正しく動作すること，および実測値との比較から処理時間の算出法の妥当性を確かめるとともに，つぎのような結果を得た．

（1） 最適化によって選ばれたデータフローの処理時間は，同一計算機上に同一ファイル構造を用いて実現された既存のデータベースシステムと比べ，平均 35.6% 短縮された．また，最適化を4通りの基本アルゴリズム全体にわたって行った場合，各基本アルゴリズムの枠内でのみ最適化を行った場合と比較して平均 27.6% の短縮となった．

（2） プログラムのサイズについては，既存のシステムと比べてモジュール間の関係が整理されたため 23.5% の削減が達成された．

（3） 問合せ・データ量・選択率などによって，最

適なデータフローは変化する．一般にデータ量の少ない所では，一変数問合せについては逐次スキャン，二変数問合せについてはネステッドループ結合アルゴリズムが有利だが，データ量が多くなると一変数問合せについてはインデックスの使用二変数問合せについてはソートマージ結合や結合インデックスを用いたネステッドループ結合が有利になる．

特に，(1)，(3)の結果は，最適なデータフローが，処理アルゴリズムや問合せ，データベース・サイズによって変わることを意味しており，最適化によるデータフロー選択の有効性の裏付けとなっている．

本論文で示した方法は，モジュール間の独立性が高く，情報隠ぺい，モジュール単位の拡張・変更の容易性など，抽象データ型・オブジェクト指向の概念などと共通の利点を持っている．機能間の独立性が高いため，負荷の大きい基本機能のファームウェア化・ハードウェア化も行いやすい．また，基本機能を多重プロセッサに分散することにより，基本的なソフトウェアの変更なしに，並列処理を取り込める可能性もある．これらの発展の可能性については，今後の課題として検討を続けていきたい．

謝辞 XDB システムを提供いただいた Maryland 大学 S. Bing Yao 博士，貴重な御意見をいただいた本学奥田健三教授，山崎勝弘博士，ならびに処理系の試作に協力いただいた卒研生宗和洋一朗君に感謝する．本研究は，一部文部省科学研究費（課題番号 61550727, 62550255）の援助による．また，試作には，ディジタルコンピュータ(株)より貸与されたUステーションを使用した．

参考文献

1) Date, C. J.: *An Introduction to Database Systems* (3rd Edition), p. 574, Addison-Wesley, Reading, Massachusetts (1982).
2) 田中克己：リレーショナルデータベースと知識ベース，*Computer Today*, No. 7, pp. 13-17 (1985).
3) 佐藤英人：リレーショナルデータベースと統計データベース，*Computer Today*, No. 7, pp. 18-21 (1985).
4) Hsiao, D. K.: Super Database Computers: Hardware and Software Solutions for Efficient Processing of Very Large Database, *Proc. IFIP 86*, pp. 933-944 (1986).
5) 清木，劉，益田：関係演算のストリーム指向型並列処理における資源割り当て方式，情報処理学会論文誌，Vol. 28, No. 11, pp. 1177-1192 (1987).
6) Baba, T., Yao, S. B. and Hevner, A. R.: Design of Functionally Distributed, Multiprocessor Database Machine Using Data Flow Analysis, *IEEE Trans. Comput.*, Vol. C-36, No. 6, pp. 650-666 (1987).
7) 馬場，Yao, S. B., Hevner, A. R.：データフロー解析に基づく多重プロセッサデータベースマシンの設計，第 29 回情報処理学会全国大会論文集，pp. 757-758 (1984).
8) 斉藤，馬場：多重プロセッサデータベースマシンにおけるデータフローの最適化，信学技報，EC 85-48, pp. 85-96 (1985).
9) Goguen, J. A.: Reusing and Interconnecting Software Components, *IEEE Comput.*, Vol. 19, No. 2, pp. 16-28 (1986).
10) 飯塚，田中：ソフトウェア指向アーキテクチャ，p. 409，オーム社，東京 (1985).
11) Stonebraker, M., Wong, E., Kreps, P. and Held, G.: The Design and Implementation of INGRES, *ACM TODS*, Vol. 1, No. 3, pp. 189-222 (1976).
12) Astrahan, M., et al.: System R: Relational Approach to Database Management, *ACM TODS*, Vol. 1, No. 2, pp. 97-137 (1976).
13) Yao, S. B.: Optimization of Query Evaluation Algorithms, *ACM TODS*, Vol. 4, No. 2, pp. 133-155 (1982).
14) Software System Technology: XQL User's Manual (version 1.0) (1982).
15) ディジタルコンピュータ(株)：Ustation マニュアル (1984).
16) Valduriez, P. and Gardarin, G.: Join and Semijoin Algorithms for a Multiprocessor Database Machine, *ACM TODS*, Vol. 9, No. 1, pp. 133-161 (1984).
17) Smith, J. M. and Chang, P. Y-T.: Optimizing the Performance of a Relational Algebra Database Interface, *Comm. ACM*, Vol. 18, pp. 568-579 (1975).
18) Wong, E. and Youssefi, K.: Decomposition —A Strategy for Query Processing, *ACM TODS*, Vol. 1, pp. 223-241 (1976).
19) 田中 譲：データベース・マシン，情報処理，Vol. 23, No. 10, pp. 939-947 (1982).

(昭和 62 年 4 月 2 日受付)
(昭和 63 年 9 月 5 日採録)

馬場 敬信（正会員）

昭和22年生．昭和45年京都大学工学部数理工学科卒業．昭和50年同大学院博士課程修了．同年より電気通信大学助手，講師を経て，現在宇都宮大学工学部情報工学科助教授．工学博士．昭和57年より1年間メリーランド大学客員教員．計算機システム，特に計算機アーキテクチャ，並列処理，データベースなどの研究に従事．著書「マイクロプログラミング」(昭晃堂)，「Microprogrammable Parallel Computer」(MIT Press)．電子情報通信学会，IEEE 各会員．

斉藤 直樹

昭和37年生．昭和60年宇都宮大学工学部情報工学科卒業．昭和62年同大学院修士課程修了．同年野村コンピュータシステム(株)入社．現在，野村総合研究所技術部勤務．在学中，データベース・システムの研究に従事．現在はソフトウェア開発環境，特にCASEを含む上流工程に興味を持っている．