Vol. 33 No. 12　　情　報　処　理　　Dec. 1992

解　説

データベースプロセッサ

データベースプロセッサ GREO†

伏見信也††　武田保孝††　岩崎浩文††
小宮富士夫††　中込宏††

1. まえがき

GREO は付加プロセッサ (attached processor) 型のデータベースプロセッサである (図-1). GREO は東京大学生産技術研究所と三菱電機㈱による共同研究の成果を実用化したもので, 1989 年に製品化されて以来, 当社オフィスコンピュータを中心に, 汎用コンピュータ, ミニコンピュータなどに装着され, 関係データベース処理, バッチ処理などのビジネスアプリケーションにおいて広く使用されている. GREO は, データベース処理におけるソート処理の重要性に着目し, これを専用ハードウェア化(ハードウェアソータ)することにより, データベース処理を中心とする広範囲のビジネスアプリケーションの高速化を実現する.

本稿では, GREO の基本構成要素であるハードウェアソータを中心に, GREO の開発背景, 処理の基本モデル, ハードウェア及びソフトウェアの構成, 専用コンパイラなどについて解説する.

2. 背　景

2.1 ソフトウェアデータベースの問題

従来のソフトウェアによるデータベース処理は, 予想されるアクセスパターンに対してあらかじめ索引 (index) を作成し, 各レコードへのアクセスパス (access path) を提供することを基本としている. データベース処理のオプティマイザは, 与えられた問合せに対し, 使用可能なアクセスパスを選択し, これを適切に組み合わせてデータベースへのアクセスプランを生成する[2]. このような方式では, 定型的な問合せ (単純な検索, トランザクション処理など) に対しては妥当な性能を得ることができるが, 非定型 (ad hoc) な問合せに対しては, 一般に索引の存在を期待できず, ファイルの全数走査などが必要となるため, 十分な性能を実現することが困難であった. また, この方式ではアクセスの基本単位がレコードであり, 多数のレコードを広範囲に走査する必要がある場合には, 個々のレコードに対する索引操作のオーバヘッドが蓄積し, 処理時間の低下を招いた.

2.2 データストリームモデル

GREO は, これら索引の効果が期待できない場合において, 処理対象となるデータ全体 (ファイルなど) を基本処理単位とし, これを高速に一括処理することを目指す. このために, 入力されてくるレコード群をデータの流れ, すなわちデータストリーム (data stream) と捉え, その流れに沿って, ソート, 選択, 射影, 結合などの基本演算を互いに重畳して並列実行する. 複雑な問合せは, 一連の基本演算群をパイプライン的に適用し, 必要に応じてデータストリームを分割/併合することにより実行される. これらデータベース処理の基本演算は, データストリームに対するデータのふるい落とし, 変形などの機能とみなされ, フィルタ (filter) と総称される. データストリームとフィルタによるデータベースの処理モデルはデータストリームモデル (data stream model)[3] と呼ばれる. 図-2 に本モデルによる SQL[14] の問合せの実行例を示す. このように, データベース処理をデータストリームの流れに沿って実行することは, ディスクからのデータの読出し時間に重畳して処理が終了することを意味し, この意味では本モデルが目指す処理時間は理論上最良となる.

2.3 データストリームモデルの実現

データストリームモデルでは, 全てのフィルタがデータストリームに沿って動作可能であることが要求され, したがってその時間計算量は入力レ

† A Database Processor GREO by Shinya FUSHIMI, Yasutaka TAKEDA, Hirofumi IWASAKI, Fujio KOMIYA and Hiroshi NAKAKOMI (Mitsubishi Electric Co.).

†† 三菱電機(株)

コード数 N に対して線型時間，すなわち $O(N)$ であることが必要となる．しかるに，データベース処理の基本演算のうち，結合，除算，グループ化，ソートなどは，単一 CPU による処理では $O(N)$ よりも大きい $O(N\log N)$ などの時間計算量を必要とする．したがって，このモデルの実現には複数プロセッサによるアーキテクチャ的支援が必須となる．

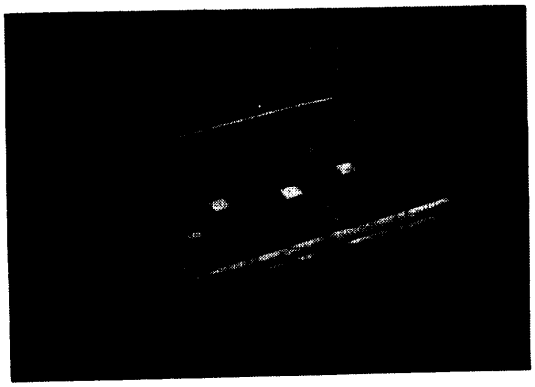


図-1 データベースプロセッサ GREO

SQL 文: SELECT * FROM T, U
WHERE T.A=U.B AND U.B>100
AND T.A+U.A=10
ORDER BY T.C

データストリームモデルによる処理：

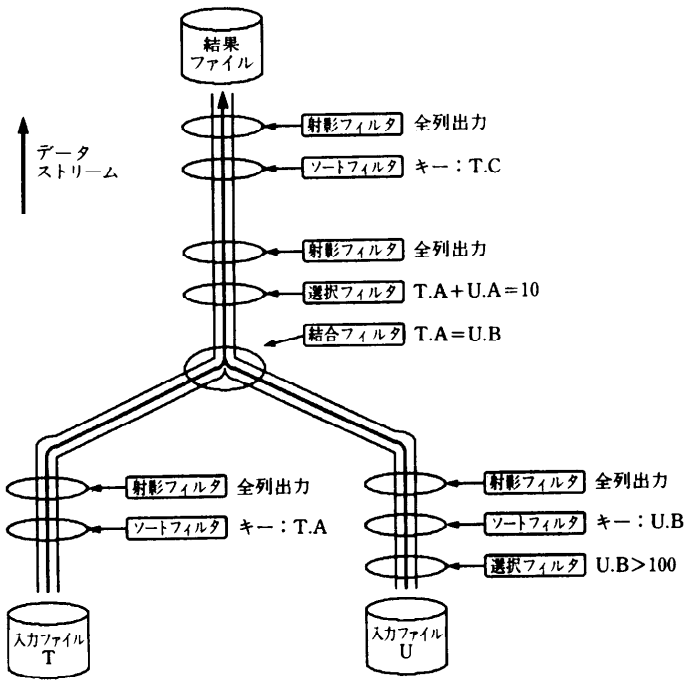


図-2 データストリームモデルの基本概念

一方，これら負荷の重い演算の時間計算量の支配項は，ソート，またはそれと等価な処理であることが知られている．すなわち，これら演算も処理対象データがあらかじめソートされていれば $O(N)$ 時間で実行可能である．たとえば，結合 (join) は結合対象のファイルが結合キーに関してそれぞれソートされていれば，各ファイルをそれぞれ一度走査することで処理が完了し，この際に必要とされる時間計算量は $O(N)$ となる．

このように，少なくともソート処理を $O(N)$ で実行可能ならば，理論的には全ての基本演算が $O(N)$ で処理可能となり，データストリームモデルが実現可能となる．このための専用ハードウェアが $O(N)$ 時間ソート可能なハードウェアソータ (hardware sorter) である．

3. 基本設計

GREO の設計に際しては，ハードウェアによる高速ソートを中心とした処理アルゴリズムとアーキテクチャの採用を基本方針とした．ここで，最近では汎用マイクロプロセッサを用いた並列データベース処理との比較において，ハードウェアソータのような特殊プロセッサの有効性が議論されている[4]が，われわれは

(1) 上記のように，ソートの高速化はデータストリームモデル実現のための中心的課題である

(2) バッチ処理などの既存のビジネスアプリケーションにおいてもソート処理が主要なボトルネックである

(3) ソート処理の実体はデータの比較，移動などの単純な処理の繰り返しであり，汎用マイクロプロセッサが提供するような複雑な機能を必要としない

(4) データベース処理におけるソートとは，雑多なデータを一定の順序に並べ替え，その後の処理を高速化する処理であ

り，いわばデータの配列に関するエントロピの最小化処理とも考えられる．この立場では，汎用マイクロプロセッサが目指す computationally intensive な，いわばエネルギ指向の処理とは本質的に異なる処理とも考えられる

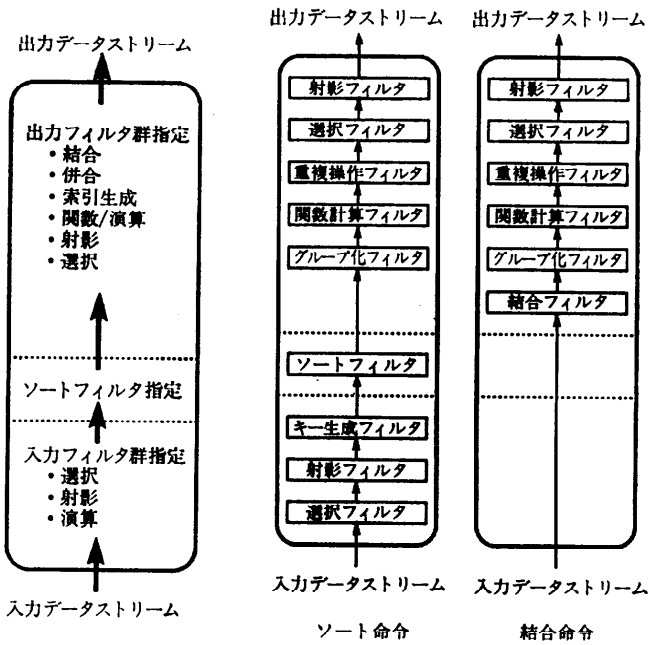
などの点から，専用ハードウェア化によるソート処理の高速化が妥当であると結論した．

この方針の実現のため，ソートを中心とした基本命令構造と，それに従う命令を定義した(**図-3**)．ここでは，図-2 に示すような各種のフィルタを，ソート，ソートの前処理，ソートの後処理に分類し，おのおのソートフィルタ，入力フィルタ，出力フィルタと呼ぶ．GREO の命令はこれらを適宜組み合わせることによって構成される．

この構造により広範囲の問合せ処理が一度のデータストリーム転送，すなわち一命令で実行可能となる．たとえば SQL による以下の問合せ：

```
SELECT NAME, ADDRESS
FROM EMPLOYEE
WHERE AGE>30
ORDER BY NAME
```

は，入力フィルタとして選択(WHERE AGE>30) 及び射影(SELECT NAME, ADDRESS)，ソートフィルタによりソート(ORDER BY NAME) をおのおの指定して得られるソート命令により，一度のデータストリームの転送で処理が完了する．図-2 の例のような複雑な問合せに対しては，**図-4** に示すようにこれら命令を木構造に多段接続してパイプライン的に実行する．



（a）命令の基本構造　　（b）命令の例

図-3 GREO の命令

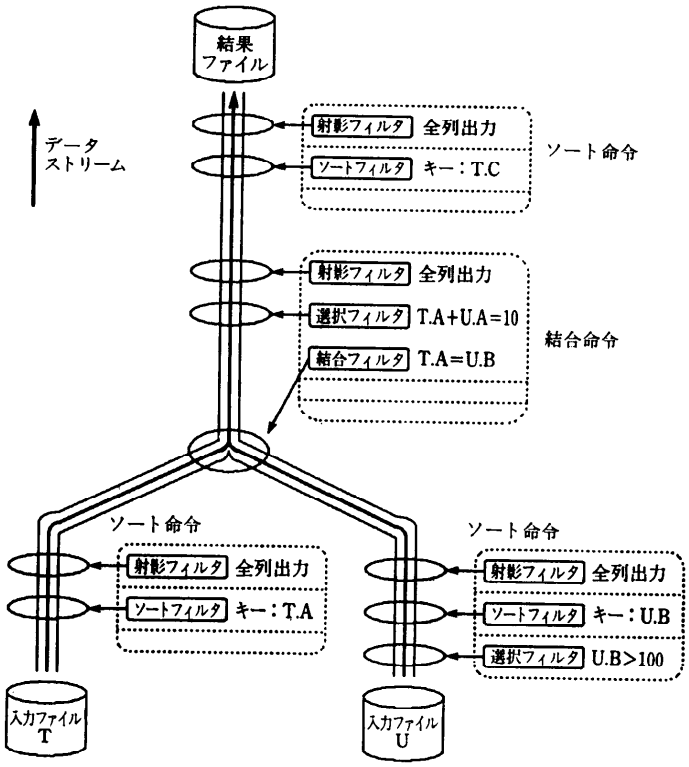


図-4 GREO の命令とデータストリームモデル

4. ハードウェアアーキテクチャ

4.1 概　要

図-5 に GREO のハードウェアの内部構成を示す．GREO は約 30×30 cm² の基板一枚に実装されており，ソートを行うハードウェアソータと，ソート以外のデータベース演算を実行するデータストリームプロセッサより構成される．ハードウェアソータは上記のソートフィルタを，またデータストリームプロセッサは入力/出力フィルタを，おのおの実現する．

ハードウェア実現に際しては，実製品としてのライフサイクルを考慮し，以下のような設計方針を定めた．

（1） ハードウェアソータの設計においては，論理と記憶を一体化したいわゆる logic-in-memory のアプローチではなく，大容量の汎用 RAM にプロセッサを接続したアーキテクチャとし，将来における汎用の RAM の集積度向上をそのまま活用できるアプローチを目指す．

（2） データストリームプロセッサに関しては，必要とされる汎用性（式/述語の数，入れ子など）や特殊性（日本語処理の例外規則など）を考慮し，また将来のマイクロプロセッサの性能向上を考慮したうえで，特殊専用なハードウェアを用いるのではなく，汎用マイクロプロセッサを必要な数用いたマルチプロセッサ構成として，所要の性能を実現する．

4.2 ハードウェアソータ[7)~9)]

ハードウェアソータのアルゴリズムやその実現方式については従来より数多くの研究があり[5),6)]，またこれらハードウェアソータを実際に実現し，それをアーキテクチャの基本要素としたデータベースマシンも RINDA[10)]，DBE[11)] など，GREO 以外にもいくつかの例がある．

GREO のハードウェアソータは，パイプラインマージソートアルゴリズム[5),6)] により $O(N)$ 時間ソートを実現する．本ハードウェアソータは，最大 8 MB/秒のソート速度を有する専用のソートプロセッサ[7)] をパイプライン接続することにより構成される．本プロセッサを用いて構成したハードウェアソータの例を図-6 に示す．n 台のソートプロセッサを一次元接続することにより，$N=2^n$ 個までのレコードをソート可能なソータを実現することができる．すなわち，本ハードウェアソータは N 個のレコードのソートに $\log N$ 台のプロセッサを使用する型のハードウェアソータである[5),6)]．GREO では $n=19$ としてソータを構成する．

本ソータの動作は，概略以下のようになる．i 番目のプロセッサ P_i は，2^{i-1} レコード分のローカルメモリを持つ．P_i は，前段のプロセッサ P_{i-1} から送られてくる 2^{i-1} 個のレコードからなるソートされたストリング（長さ 2^{i-1} のストリング）を自メモリにロードし，続いて P_{i-1} から送られてくる長さ 2^{i-1} のストリングを on-the-fly にマージして長さ 2^i のストリングを生成し，次段のプロセッサ P_{i+1} に送出することを繰り返す（図-7）．この結果，入力データストリームの最初のレコードの到着時点からソートされたストリームの出力が終了するまでの全ソート時間は $2N+$

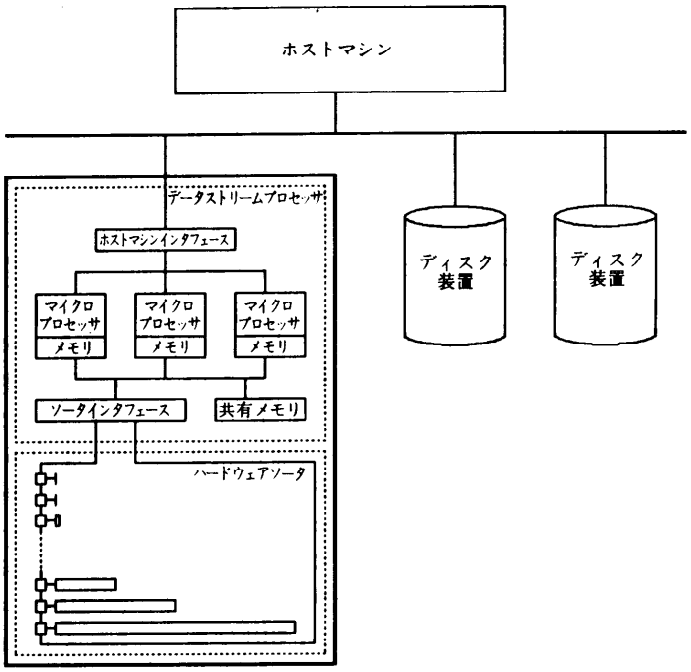


図-5 GREO のハードウェア構成

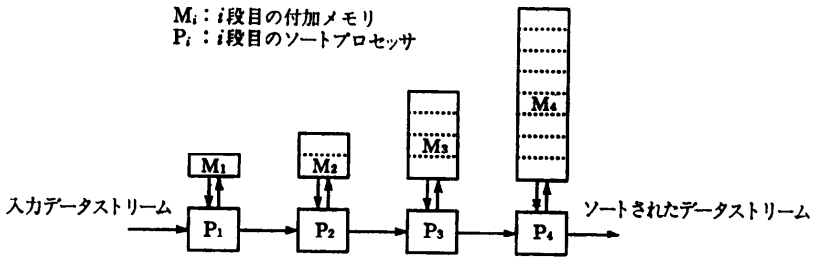


図-6　ハードウェアソータの構成

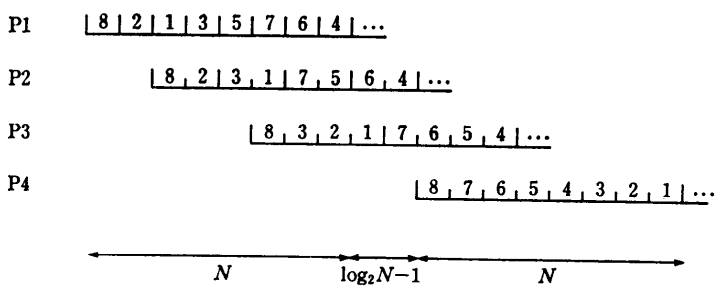


図-7　パイプラインマージソート

$\log_2 N-1$ となり，$O(N)$ 時間でのソートが実現される．

また，本ソータは String Length Tuning 機能[7),9)]など，レコード長，レコード数などの変動に対して柔軟に対応できる拡張機能が実現されている．これら拡張機能により，実際の利用環境下において，ソータの資源を有効に利用した処理が実現されている．

4.3　データストリームプロセッサ

データストリームプロセッサは，3個の汎用マイクロプロセッサ (MC 68020) とローカル/共有メモリ，ハードウェアソータとのインタフェースプロセッサ，ホストマシンとのインタフェースなどから構成される．

汎用マイクロプロセッサ群は，ハードウェアソータによるソート処理の前処理/後処理をおのおの入力/出力フィルタ群として実行する．上記のように，GREO の設計においては，選択，射影など，ハードウェア化が可能である処理に対しても，実用上必要となる汎用性の観点から高速の汎用マイクロプロセッサとそのソフトウェアによる処理を採用した．これらマイクロプロセッサ群は，GREO への入力データストリーム，GREO からの出力データストリームに対し，その処理負荷に応じて適当な数が動的に決定され，おのおのに割り付けられる (**5.2** 参照)．

また，ソータインタフェースプロセッサは，ホストマシン上でのデータの形式とハードウェアソータのデータ形式を on-the-fly に変換する機能，マイクロプロセッサ群からハードウェアソータにアクセスするための制御機能などを有する．さらに，ハードウェアソータの持つメモリを汎用マイクロプロセッサ群の共有メモリとして利用できる2ポート構成とし，ファイルのマージ処理，結合処理などの際に大容量バッファとして使用する．

5. ソフトウェアアーキテクチャ

5.1　概　要

図-8 に GREO のソフトウェア構造を示す．GREO が装着されるホスト計算機上には，SQL などの言語プロセッサ，ソートなどのユーティリティ群，GREO の命令群の実行スケジューラ，GREO の動作を制御する GREO サーバなどが存在する．ユーザからの問合せは図-4 のような木構造を有する GREO の命令群にコンパイルされ (**6.** 参照)，ユーティリティ群からの処理要求とともにスケジューラに送られる．これら命令群に対し，GREO のスケジューラはデータフローの発火制御の原理[4)]に従って実行可能な命令を識別し，

これをサーバに送って命令の実行を行う．GREOが複数台装着されている場合にはこれらGREOが同時に起動され，複数のGREOによる並列処理が実現される．

GREO内部には，データストリームプロセッサを構成する汎用マイクロプロセッサ上にSYと呼ばれるマルチプロセッサ/マルチタスクOSが搭載される．SYは，プロセスの生成/消滅，プロセッサ間/プロセス間通信，マルチプロセッサ間の排他制御，割込み制御，ホスト計算機との通信などの機能をシステムコールとして提供する．GREOの諸機能は，これらシステムコールを用いたSY上のプロセスとして実現される．このうち，ホストマシンとGREOを接続するための機能（入出力コマンドの交信など）はSY上に常駐するプロセスとして実現され，個々のデータベース機能は要求に応じて動的に生成されるフィルタプロセスとして実現される．

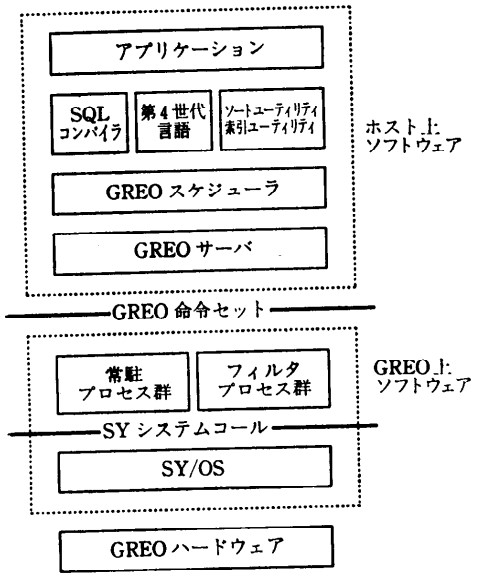


図-8 ソフトウェア構成

5.2 フィルタの実現

GREOは，表-1に示される基本コマンド群をホストマシンに提供することにより上記データストリームモデルとフィルタの概念を実現する．すなわち，GREOのサーバは，スケジューラからGREOに対する命令が与えられると，まず起動すべきフィルタ群を識別し，（必要であれば）entryにより所要のプログラムをフィルタとしてGREOに登録（ホストからのダウンロードなど）する．次にopenによりこれらをGREO内部でプロセスとして起動し，フィルタを実体化する．起動されたフィルタに対し，xferにより連続的にデータを送り（あるいはフィルタから連続的にデータを受信し），closeによりフィルタプロセスを消滅させる．

openの際には，init引数により選択処理や射影処理の詳細が指定される．この引数はGREOの命令中のフィルタ指定の実体である．これらは，open実行時にGREOのマイクロプロセッサ上で動的にコンパイルすることが可能であり，以降のxferでは指定条件に対して最適化されたコードにより処理が行われる．

また，一般に一つのデータストリームの処理にはGREO内部では入力/出力の双方のデータストリームの同時処理が必要となる．SYは，これら二つのデータストリームに対して入力フィルタ群，及び出力フィルタ群を起動し，これらを同時に動作させる．この際，SYはこれら同時実行される二つのフィルタ群に対し，その処理負荷に応じて割り付けるマイクロプロセッサ数を変化させることにより負荷分散を制御する．

表-1 GREOの基本インタフェース

インタフェース	機能	説明
entry (Prog, io, id)	フィルタの登録	progで指定されるプログラムをioで示される種別（GREOへの入力データストリームに対するものか，出力データストリームに対するものか）の番号idのフィルタとしてGREOに登録する．
open (io, id, init)	フィルタの生成	ioで示される種別のidで登録されたプログラムをフィルタプロセスとして起動し，これを初期化パラメタinitにより初期化する．初期化パラメタとしては，選択処理を起動の際の選択条件や射影条件などがある．
xfer (io, id, data)	フィルタへ/からのデータストリーム転送	ioで示される種別のidで指定されるフィルタへ（から）dataをデータストリームとして入力（出力データストリームをdataに格納）する．データ量が多い場合には，これを必要な回数繰り返して実行する．
close (io, id, stat)	フィルタの消滅	ioで示される種別のidで指定されるフィルタを構成するフィルタプロセスを停止，消滅させ，それまでにxferにより実行したデータストリーム処理に関する統計情報をstatに取り込む．

6. 言語コンパイラ[13]

ここでは図-3に示す二つのGREOの命令を仮定して，SQLにより記述された処理をGREOの命令群に変換するためのコンパイルアルゴリズムの概略を述べる．

6.1 実行木の生成

GREO用のSQLコンパイラは，まず実行木(Execution Strategy Tree)を生成する．実行木とは，問合せが参照しているファイルの集合を結合演算により結合し，一つの出力ファイルとするための結合順序を決定するための木構造である．また，これら各結合演算に付随して実行される選択演算などを確定するためにも用いられる．木の各ノードは，処理対象のファイル(FROM句に指定された表)，及び処理の(中間)結果のファイルを表し，またそのファイルを生成する結合述語，結合結果に対する選択述語，結果に対するソートのキーなどを表現する．実行木の各ノードに対し，ファイルまたは結合結果を構成するファイルの集合がラベルとして付加され，ノードに付随する結合述語，選択述語，ソートキーがノードの属性としておのおの「J:」，「S:」，「K:」により表現される．

実行木の例を図-9に示す．図-9は，図-2と同様のSQL文に対し，FROM句で参照されているファイル({T}と{U})をWHERE句の述語「T.A=U.B」で結合({T,U})する実行木を示す．{T,U}ノードの「J:」フィールドがT.A=U.Bであることによって結合の構造が示されている．また他の述語，キーは以下のように各ノードに分配されている．

①単一の表の選択
(U.B>100) →{U}の「S:」
②結合の前処理のソート →{T} {U}の「K:」
③結合結果に対する選択 →{T,U}の「S:」
④ORDER BY句に対応するソート →{T,U}の「K:」

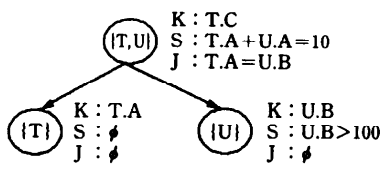


図-9 実行木

表-2 性能評価

ベンチマーク	対ソフトウェア処理性能向上比	
	応答時間向上比	CPU時間減少比
選択処理1	14.1	1/16.0
選択処理2	46.0	1/116.1
選択及び結合処理	77.1	1/421.0

6.2 コード生成

以上により生成された実行木が，コード生成によりGREOのソート命令，結合命令の組合せに変換される．ここで，ノードのうち「J:」フィールドが空のものはソート命令に，そうでないものは結合命令に変換される．変換の際，「J:」フィールド以外のフィールドに指定されている種々の述語，式などは対応する命令の内部の基本機能フィルタ群に写像され，フィルタのopen時の初期化パラメタとしてGREOに指示される．図-4は図-9の実行木から生成されたコードの例である．

7. おわりに

データベースプロセッサGREOについてその概要を解説した．GREOはすでに多数の使用実績がある．各種のベンチマークによる性能評価の例を表-2に示す．これらの評価から，関係データベース処理，既存のバッチ処理などを含む広い範囲のビジネスアプリケーションに対してGREOが有効であることが実証的に確認された．

参考文献

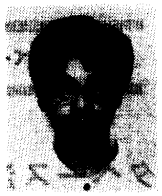
1) 喜連川優，伏見信也：データベースマシン，情報処理，Vol. 28, No. 1, pp. 56-67 (1987).
2) Selinger, P., Astrahan, D. and Chamberlin, D.: Access Path Selection in a Relational Database Management System, Proc. of ACM SIGMOD Conference (1979).
3) Fushimi, S., Kitsuregawa, M. and Tanaka, H.: An Overview of the System Software of a Parallel Relational Database Machine GRACE, Proc. of 12th Intl. Conf. on Very Large Data Bases, pp. 209-219 (1986).
4) DeWitt, D. and Gray, J.: Parallel Database Systems: The Future of High Performance Database Systems, Comm. ACM Vol. 35, No. 6, pp. 85-98 (1992).
5) Bitton, D., DeWitt, D., Hsiao, D. and Menon, J.: A Taxonomy of Parallel Sorting, Comput. Surv., Vol. 16, No. 3 (1984).
6) 喜連川優，楊 維康，鈴木慎司：VLSIソートプロセッサ，情報処理，Vol. 31, No. 4, pp. 457-465 (1990).

7) 伏見信也，喜連川優他：LSI ソートプロセッサ，信学技報，DE 88-2, pp. 9-16 (1988).
8) Kitsuregawa, M., Yang, T. and Fushimi, S.: Implementation of LDI Sort Chip for Bimodal Sort Memory, Proc. of VLSI 89, pp. 285-294 (1989).
9) Kitsuregawa, M., Yang, T., Suzuki, T. and Takagi, M.: Design and Implementation of High Speed Pipeline Merge Sorter with Run Length Tuning Mechanism, Proc. of 5th Intn'l Workshop on Database Machines, pp. 144-157 (1987).
10) 井上 潮，速水治夫，福岡秀樹，鈴木健司，松永俊雄：リレーショナルデータベースプロセッサ RINDA の設計と実現，情報処理学会論文誌，Vol. 31, No. 3, pp. 373-380 (1990).
11) 松田 進，東郷一生，島川和典，岩崎孝夫：データベース演算処理装置のアーキテクチャ，信学技報，Vol. 91, No. 259, DE 91-38, pp. 33-38 (1991).
12) 安藤隆朗，小宮富士夫，中込 宏，伏見信也，喜連川優：リレーショナルデータベースプロセッサ GREO の構成，信学技報，DE 89-37, pp. 8-15 (1989).
13) 伏見信也，岩崎浩文，安藤隆朗，佐藤重雄，小宮富士夫，樋口雅宏：高速ハードウェアソータを用いた SQL システムの実現，信学技報，Vol. 91, No. 259, DE 91-38, pp. 1-8 (1991).
14) JIS：データベース言語 SQL/X 3005, 1990.
15) Ullmann, J.: Principles of Database Systems, Computer Science Press (1982).

(平成4年8月19日受付)

伏見 信也（正会員）

昭和34年生．昭和56年東京大学理学部情報科学科卒業．昭和61年同大学院情報工学専門課程博士課程修了．工学博士．同年三菱電機(株)入社．データベースシステム，データベースマシンの研究，開発に従事．元岡記念賞，本会学術奨励賞受賞．ソフトウェア科学会，ACM，IEEE 各会員．



武田 保孝（正会員）

昭和35年生．昭和58年新潟大学工学部情報工学科卒業．昭和60年同大学院工学研究科情報工学専攻修士課程修了．同年三菱電機(株)入社．以来，第5世代コンピュータプロジェクトの一環として，逐次型推論マシン (PSI) および並列推論マシン (PIM) のハードウェア開発に従事．現在，関係データベース処理プロセサの研究開発を行っている．

岩崎 浩文（正会員）

昭和35年生．昭和58年東京工業大学理学部情報科学科卒業．同年三菱電機(株)入社．以来，ビジネルコンピュータ向データベース・データ管理システム，障害回復システム等の研究開発に従事．

小宮富士夫（正会員）

昭和29年生．昭和51年国立東京高専電子工学科卒業．同年三菱電機(株)入社．以来，ミニコンピュータ，オフィスコンピュータ等の OS 開発を通じ，データ管理，リレーショナル・データベース，ネットワーク・ファイルシステムの研究開発に従事．

中込 宏

昭和28年生．昭和52年東北大学工学部電気工学科卒業．同年三菱電機(株)コンピュータ製作所入社．以来，オフィスコンピュータの入出力制御装置，入出力チャネル，リレーショナルデータベースプロセッサの開発に従事．