データベース・システム 39－2
（１９８４．１．２６）

リレーショナル・データベースマシン　ＦＲＥＮＤ

正田 定幸，川上 英，岸田 巧，羽生田 博美

沖電気工業株式会社　総合システム研究所

１．　はじめに

データベース・マシンは、ホスト・コンピュータの処理負荷を軽減するとともに、データベース処理能力の向上を大きな目標にしている。しかしながら、データベース・マシンの提案は多いものの、その総合的な性能評価は少ない。すなわち、データベースマシンでは、情報の検索だけでなく、データベースの更新や障害回復をも含めた、制御方式を確立せねばならず、その性能評価は、これらの処理オーバヘッドを含めたものでなければならない。

筆者らは、小規模データベース・マシン－ＦＲＥＮＤ－を開発し、その適応領域、提供するコマンド機能、処理方式及びいくつかの性能評価を報告してきた[5][6][7][8][9]。本稿では、より詳細な性能評価を文献[1]を参考にして行ったので、その結果を報告する。

２．　基本構成とコマンド仕様

ＦＲＥＮＤは１６ビットマイクロプロセッサと小型ディスクから成っており、この上で関係データベース管理システムが実現されている。一方、ＦＲＥＮＤのホスト・コンピュータとなるパーソナル・コンピュータには、ＦＲＥＮＤにアクセスするためのインタフェース・ルーチンがあり、これを介してアプリケーション・プログラムはデータベースを利用することが出来る。ＦＲＥＮＤとパーソナル・コンピュータは標準的な通信回線により結合されるため、様々な異種のパーソナル・コンピュータでも、共通のデータベースを利用することが可能となる。システムの基本構成を図２．１に示す。

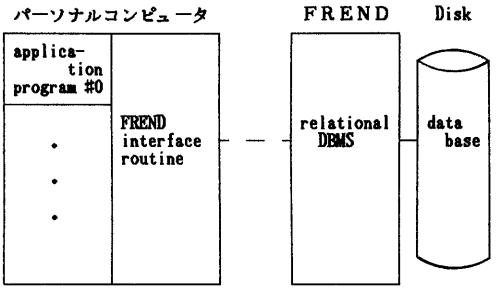

図２．１　FREND システム構成

ＦＲＥＮＤを利用するための各種コマンドはパーソナル・コンピュータ内にあるインタフェース・ルーチンで提供されている。パーソナル・コンピュータ内のアプリケーション・プログラムは、この非手続的な検索コマンドを使って、容易にデータベースを利用することができる。このため、パーソナル・コンピュータはデータベース管理の負荷から解放されるとともに、複数のパーソナル・コンピュータがＦＲＥＮＤ内のデータベースを共有することが出来る。検索コマンドで指定する条件は以下に示す形式（Conjunctive Normal Form）で表現される。

$$\text{Condition} ::= \bigcap_{i} \left(\bigcup_{j} P(i,j) \right)$$

$$P(i,j) ::= (\text{attribute})(\text{operator})(\text{value})$$

ＦＲＥＮＤでは現在、タプル単位の操作機能を提供している。アプリケーション・プログラムはカーソル識別子を使って、必要とするタプルを取り出すことができる。以下に検索コマンドと更新コマンドの例を示す。

例１　"職種がプログラマである者の氏名と所属を検索せよ"

```
Conditional retrieve  EMP.NAME,
                      DEPT.DNAME
From                  EMP, DEPT
Where                 EMP.DEPT#
                        =DEPT.DEPT#
AND                   EMP.JOB=
                        "PROGRAMMER"
CURSOR                C1

Get Next              C1
```

Conditional Retrieveコマンドにより条件を満足するタプルの一つが出力され、以後はGet Nextコマンドによりカーソル C 1 を指定することにより順次１タプルずつ出力される。

例２

```
Conditional Retrieve  *
From                  EMP
Where                 JOB=
                        "PROGRAMMER"
Cursor                C1

Delete                C1
```

この例では、JOB="PROGRAMMER"であるタプルを検索し、カーソルC1を使ってタプルを削除している。

表２．１にＦＲＥＮＤで提供する代表的コマンドを示す。

表2．1　FREND アクセス・コマンド

classification	command name	abbreviation	function
database session	Begin Database	BD	start session
	End Database	ED	end session
relation definition	Define Relation	DF	creation of relation
	Define key	DK	creation of inverted index
	Delete Relation	DR	drop of relation
transaction management	Start Transaction	ST	declaration of transaction start
	End Transaction	ET	declaration of transaction end
	Abort Transaction	AT	transaction abnormal end
data manipulation	Conditional Retrieve	CR	retrieval with condition
	Sequential Retrieve	SQ	retrieval like a file
	Get Next	GN	get one tuple
	Insert	IN	insert a tuple
	Delete	DL	delete a tuple
	Replace	RP	replace a tuple
access right control	Register User	RU	registrate a user name
	Delete User	DU	delete a user name
	Change Password	CP	change password
	Authorize	AU	authorization of access right
maintenance	Initialize	II	initialize database
	Copy Database	CD	database backup

３．　評価システムとデータベース

データベース管理システム（ＤＢＭＳ）の性能評価として、System R [2]や、INGRES [3][4] の報告があるが、必ずしも標準的な評価方法が確立しているわけではない。ＦＲＥＮＤはホスト・コンピュータとして、パーソナル・コンピュータを想定したデータベース・マシンであり、その適応領域は従来のＤＢＭＳとは、必ずしも一致しない。したがって、単純な性能評価だけで、ＦＲＥＮＤと他のシステムとの比較は困難であるが、ここでは文献［１］で行われた、ＤＢＭＳの評価を参考にして、本データベース・マシンの性能を実測した。

３．１　評価システム

評価に用いたＦＲＥＮＤのディスク容量は４０Ｍｂｙｔｅである。また、ディスク・アクセスの際のページ・サイズは１Ｋｂｙｔｅである。

ＦＲＥＮＤはバックエンド・タイプのデータベース・マシンなので、ＦＲＥＮＤと、ホスト・コンピュータ間のデータ転送速度によってレスポンスが変動する[5]。したがって今回は、ＦＲＥＮＤ自体の性能評価を行うためにＦＲＥＮＤとホスト・コンピュータ間の通信時間を含まない、ＦＲＥＮＤの内部処理時間を測定し、評価を行った。また、ＦＲＥＮＤは基本的にデータベース管理者が、専任されていないアプリケーションを想定しているため、データベース・メンテナンスが必要となるような２次記憶上のクラスタリング手法を採用していない。したがって、ここでの測定データは非クラスター状態におけるものである。評価には、ＦＲＥＮＤ内部のハードウェア・タイマと、システム内部に組み込んだ評価用プログラムとを用いて、ＦＲＥＮＤのＣＰＵタイム、及び２次記憶アクセス・タイム等を測定した。また、ＦＲＥＮＤでは、ホスト・コンピュータ（ユーザ）を８台まで接続して処理できるが、この評価では、１ユーザとした。このホスト・コンピュータから検索等のコマンドを投入し、このコマンドに対する処理時間を測定した。また、ＦＲＥＮＤは３２Ｋｂｙｔｅのデータベース・バッファ（ディスク・キャッシュ）を持っているが、この効果を明確にするために、処理を行なう前にバッファをクリアーしてから測定を行った。

３．２　評価用データベース

検索処理、及び更新処理に関する評価に使用した２種類のリレーションを次に示す。

（１）　検索用リレーション

検索処理の評価用リレーションＲｉはタプル長 182byte、タプル数 10000で、リレーションの主キーとなる項目（項目名　ＵＮＩＱＵＥ）及び他のいくつかの項目から成る。項目ＵＮＩＱＵＥは項目長 4 byte であり、0から9999 の整数値をとる．この項目ＵＮＩＱＵＥに対して、各種の条件を加えることにより、条件検索、または、ジョイン検索の際の、タプル選択率を制御する。また、Ｒｉと同じ構造で項目ＵＮＩＱＵＥに対して、Ｂ＋treeのインデックスを定義したリレーションＲＢｉも作成した。

（２）　更新用リレーション

タプルの追加等の更新処理の評価用リレーションとして、９種類のリレーションを用意した。これらのリレーシ

ョンはすべて、タプル長 182byte、タプル数 10000 であるが、それぞれのリレーションはＢ＋treeのインデックスを持つ項目数（ 0 ～ 8 個 ）が異なる。インデックスが定義された項目は、Ｒｉの項目ＵＮＩＱＵＥと同様に4byte の整数とし、値は 0 から 9999 とした。タプルの追加には 1000 タプル、削除置換には 10000 タプルをあらかじめ追加したリレーションを用意した。

４．　性能評価

４．１　条件検索

種々のアプリケーションを想定して、評価用リレーションは項目ＵＮＩＱＵＥにＢ＋treeインデックスが定義されたＲＢiを使用した。また、ＲＢiの項目ＵＮＩＱＵＥに対して各種の条件を加え、タプル選択率が、0.01% ～ 10%（ 出力タプル数 1 ～ 1000 ）と異なる条件検索を行い、処理時間を測定した。表４．１に測定結果を示す。

表4．1　条件検索の処理時間

n	1	2	10	20	100	1000
t	357.5	186.9	55.9	41.2	29.7	27.7
γ	1.000	0.523	0.156	0.115	0.083	0.077

n: 出力タプル数
t: 処理時間／n （m sec.）
γ: 処理時間（t）の比

１タプル当りの処理時間を見ると、タプル選択率が、0.01% ～ 0.1%（出力タプル数 1 ～ 1000 ）で急激に減少し、1%（ 出力タプル数 100 ）でほぼ一定となっている。これは、ＦＲＥＮＤでは、結果のタプルが複数個ある問い合せに対しては、バッファリングの効果が大きく現れているものと考えられる。リレーション自体のデータ量は約 1.8Mbyte で、データベース・バッファの約 56 倍であるが、ＦＲＥＮＤでは、システム・ページ、データ・ページ等の種類の異なるデータを、独立にバッファリングしているためバッファリングが効果的に行われていると考えられる。また、リレーションＲＢiの代りに、項目ＵＮＩＱＵＥにインデックスが定義されていないリレーションＲｉを用いて、同様の測定を行なうと、リレーション・スキャンを行なうため全処理時間はほぼ一定で、117.9 秒～134.2 秒であった。Ｂ＋treeインデックスが定義されている場合と比較すると、Ｂ＋treeインデックスが検索に対して、非常に効果的であることがわかる。

４．２　ジョイン検索

ここでは、単一リレーションの検索に用いたものと同様の、項目ＵＮＩＱＵＥにＢ＋treeインデックスを定義されたＲＢiを用意して測定を行い、２つのリレーションを結合するジョイン１と３つのリレーションを結合するジョイン２の、二種類の問い合せについて評価を行った。次に評価に用いた問い合わせをＳＱＬでの例で示す。

ジョイン１

```
SELECT *
FROM    RB1, RB2
WHERE   RB1.UNIQUE = RB2.UNIQUE
AND     RB2.UNIQUE < n
```

ジョイン2

```
SELECT  RB1.*, RB2.*
FORM    RB1, RB2, RB3
WHERE   RB1.UNIQUE = RB2.UNIQUE
AND     RB2.UNIQUE < n
AND     RB1.UNIQUE < n
AND     RB1.UNIQUE = RB3.UNIQUE
```

この二種類の問い合せで、nの値を1～1000まで変化させることにより単一リレーション検索の評価と同様、種々のアプリケーションに応じた、ジョイン検索を、評価することができると考えられる。またnは結果の出力タプル数であり、n/10000は条件を加えられたリレーションのタプル選択率となる。表4.2及び4.3に測定結果を示す。

表4.2 ジョイン1の処理時間

n	1	2	10	20	1000
t	721.4	401.1	159.2	134.9	130.1
γ	1.000	0.556	0.221	0.187	0.180

n: 出力タプル数
t: 処理時間／n (m sec.)
γ: 処理時間（t）の比

表4.3 ジョイン2の処理時間

n	1	2	10	20	1000
t	1108.4	683.9	396.4	355.7	342.9
γ	1.000	0.617	0.358	0.321	0.309

n: 出力タプル数
t: 処理時間／n (m sec.)
γ: 処理時間（t）の比

ジョイン1、ジョイン2ともに、条件を加えたリレーションのタプル選択率が増加するにしたがって、1タプル当りの処理時間が、急激に減少している。このことから単一リレーションの場合と同様に、問い合せ結果のタプルが複数の場合に、バッファリングの効果が大きいと考えられる。また、単一リレーション検索も含めて、表4.1、表4.2、表4.3を比較すると、結合処理の多い複雑な検索ほど、処理時間が結果のタプル数の影響を受けていない。FRENDで採用しているジョイン検索処理が、単純なネステッド・ループ方式であることから、これによるCPUのオーバーヘッドはそれほど多くないと考える。むしろ、ジョインするリレーション数が増すに従って、外側のループで検索されたリレーションが使用したバッファをループの内側でスワップすることが多くなり、バッファリングの効果が低くなるためであると考えられる。ジョイン2では、処理の対象となるデータの総量はデータベース・バッファサイズの約170倍である。特に、2次記憶アクセス回数の少ない出力タプルが1個であるジョイン1、ジョイン2の処理時間は、単一リレーション検索に比較して各々約2倍、3倍で処理されている。したがってデータベース・バッファを増加させることにより、ジョイン検索の処理時間を、単一リレーション検索の"結合するリレーション数"倍程度まで減少させることができると予想される。

4.3 更新

更新処理の評価として、タプルの追加、削除、置換処理の評価を行った。FRENDではタプルの削除及び、置換は例2の様に、対象となるタプルの検索と、削除、置換を組み合せて行な

うが[6]、本評価では検索に対する処理時間を除いた削除または、置換コマンドだけの処理時間について測定を行った。FRENDは更新処理に対して、シャドー方式によるデータの二重化処理を行っており[9]、測定結果にはこの処理に関する負荷が含まれている。二重化されたデータを一重化するEnd Transactionコマンドに対する処理時間は含まれていない。

（1） タプル追加、削除

4．2節で示した様に、インデックスは、検索において非常に有効である．しかしながら、更新では逆に、インデックスを持つことがオーバーヘッドとなる。したがって、リレーションを設計する上では検索と更新のトータルな割合を考慮して、インデックスの数を決定しなければならない。したがって追加、削除処理では、インデックの個数と処理時間という観点からインデックスの項目を各々0～8個持つ9個のリレーションを用いて評価を行った。測定結果を表4．4、及び表4．5に示す。

インデックスが定義されている場合、タプルの追加、削除はインデックス処理が処理全体の大部分をしめている。文献[1]でも指摘されているように、タプルの追加、削除処理では、リレーションに対して定義されているインデックスの有無、個数が性能に最も大きく影響すると考えられる。インデックス数が増すにしたがって、処理時間中のDisk・I/O時間の割合が増えている。これは、インデックス数が増すにしたがってバッファリングの効果が低くなっていくためであると考えられる．したがって、データベースのバッファが増大すると、この場合も性能が向上するものと予想される。

表4．4 タプル追加の処理時間

n	0	1	2	3	4
t	45.2	129.6	238.9	395.8	696.5
γ (i/o)	0.082	0.076	0.235	0.368	0.540

5	6	7	8
869.7	1324.3	1600.5	2006.6
0.576	0.668	0.684	0.711

n：インデックス数
t：処理時間／n (m. sec.)
γ (i/o)：処理時間に対するdisk i/o時間の比

表4．5 タプル削除の処理時間

n	0	1	2	3	4
t	43.6	151.3	242.5	249.2	446.7
γ (i/o)	0.092	0.394	0.438	0.293	0.395

5	6	7	8
786.0	913.4	1106.1	1408.8
0.625	0.634	0.642	0.670

n：インデックス数
t：処理時間／n (m. sec.)
γ (i/o)：処理時間に対するdisk i/o時間の比

（2） タプル置換

タプルの置換処理では、以下に示す項目に対して評価を行った。

a．インデックスを持たない項目に条件を加え、この項目の値を変更する。
b．インデックスを持つ項目に条件を加え、他のインデックスを持たない項目の値を変更する。

c．インデックスを持つ項目に条件を加え、この項目の値を変更する．

d．インデックスを持つ項目に条件を加え、他のインデックスを持つ項目の値を変更する。

表４．６に測定結果を示す。

表4．6 タプル置換の処理時間

タプル置換の種類	a	b	c	d
処理時間／タプル数	16.2	19.9	57.0	159.8

（m sec.）

インデックスに対する処理が発生しないａ、ｂと、インデックスに対して何らかの処理を行なうｃ、ｄを比較すると、追加削除の場合と同様、インデックスに対する処理時間が置換処理全体で大きな負荷となっている。また、ａ、ｂ、ｃと、ｄの２グループを比較すると、ｄの処理時間が他の処理に対して、非常に大きくなっている。これは、ａ、ｂ、ｃの場合、一度アクセスされたタプル、インデックス等に更新処理が行われるため、バッファリングが非常に効果的になっているが、ｄでは、この効果が比較的低いためであると考えられる。削除、追加処理はインデックスの使用に応じて負荷が比較的大きくなるが、置換処理は、処理ａのようにインデックスを持つ項目をタプルの識別子的使用をすれば、負荷はほとんど増えない。

５．　結論

ＦＲＥＮＤについて、主にバッファリングの効果、及びインデックスの効果を中心に評価を行った。データベース・バッファは、３２Ｋｂｙｔｅと大きくはないが、検索及び更新処理において非常に有効であることがわかった．ＲＡＭは、今後益々低価格化、高速化していくので、より性能の良いデータベース・バッファの構築が期待できる．インデックスは、検索において、選択率が１０％程度と比較的高い場合でも、インデックスを使用しない場合とと比較して４.５倍程度速く、非常に有効であることを示している。また、データベース・バッファ及びインデックスは二次記憶へのアクセス回数を減少させるために処理時間を短縮する事ができるが、バッファリングの効果が上がってくると、処理のボトル・ネックは二次記憶へのアクセス処理から、ＣＰＵの処理速度へと移ってくると考えられる。しかしながら、２次記憶のアクセス時間を大幅に短縮することは、現在のディスクでは困難である。一方、ＣＰＵはクロックの高速化や、高性能の３２bitプロセッサの採用等により、現在のＦＲＥＮＤアーキテクチャでかなりの程度対応していくことが可能と考えられる。今後は二次記憶のクラスタリングの効果や他のアルゴリズムを使った結合処理等の評価を行って、より高性能な方式の開発を行なう予定である。

[参考文献]

(1) Bitton, D., DeWitt, D.J., Turbyfill, C., "Benchmarking Database Systems A systematic Approach", VLDB'83

(2) Chamberlin, D.D., et.al. "Support for Repetitive Transactions and Ad Hoc Queries in System R", ACM TODS Vol6 No1, 1981, pp70-94.

(3) Stonebraker, M., et.al. "Performance Enhancements to a Relational Database System", ACM TODS, Vol.8, No.2, pp167-185

(4) Stonebraker, M., "Retrospection on a Database System", ACM TODS Vol5 No2, 1980, pp225-240

(5) 川上 他 "データベース・マシン FRE ND の性能評価", 電子通信学会, 昭和58年度 情報・システム部門全国大会, 548

(6) 疋田 他 "データベース・マシン FRE ND のコマンド機能", 電子通信学会, 昭和58年度 情報・システム部 門全国大会, 547

(7) 疋田 他 "分散処理向データベース・マシン (FREND)", 電子通信学会 電子計算機研究会, EC83－18, 1983, pp29－35

(8) 疋田 他, "小規模OA向きデーターベース・マシン FREND", 情報処理学会, 第26回全国大会, 4F－10, 昭和58年前期, pp705－706

(9) 川上 他, "データベース・マシン FRE ND の処理構造", 情報処理学会, 第26回全国大会, 4F－11, 昭和58年前期, pp707－708