リレーショナル・データベース管理システムRDB/V1

牧之内顕文，手塚正義，北上始，佐藤秀樹，泉田義男，
安達　進，中田輝生，石川　博
（富士通研究所）

1．はじめに

RDB/V1は，Coddが提案したリレーショナル・データモデルに基づいた汎用データベース管理システムである。彼は最近の論文[4]で，世間に流布している汎用リレーショナル・データベース管理システムを二つのクラスに分類している。彼によれば，次の3条件を満足するシステムが「完全リレーショナル（full relational）」で，条件(1)と(2)は満足するが，(3)は満足しないシステムは「準リレーショナル（semirelational）」である。

(1) リレーショナル・モデルの構造的側面をサポートしている。
(2) 挿入－更新－削除の諸ルールをサポートしている。
(3) データ亜言語（data sublanguage）がリレーショナル代数と同程度に強力である。

この分類に従えば，RDB/V1は「完全リレーショナル」なシステムである。

一方，Kim[7]は発表されたリレーショナル・データベース管理システムを整理して一覧表を作った。彼の整理方法に準拠してRDB/V1のプロファイルを表1に示してある。

name	RDB/V1
year	1979
machine	FACOM Mシリーズ
language	SPL/100
status	active
type	institutional
implementor	富士通研究所

(注) year：システムが稼動しはじめた年
language：開発言語
status：現状
type：institutional / commercial

名称＼機能	optimizing	View/Snapshot	Security	Integrity	Concurrency Control	Recovery	Report Gen.
RDB/V1	Yes	Yes	Yes	No	Yes	Yes	No

表1．RDB/V1のプロファイル

この論文では，ＲＤＢ／Ｖ１のシステム全体にわたって，その設計目標，機能，構造及び実現方法についてその特徴的な面に焦点をあてて概観する。なお，読者はデータモデル，データベース管理システムについての一般的な知識を持っているものと想定している。

設計目標

ＲＤＢ／Ｖ１システムが満足すべき主要な目標は次の三つである。

- 使い易いデータベース管理システム
- 効率のよいデータ操作（検索，更新，挿入，削除）
- 巾広いユーザ，アプリケーション

データベース管理システムの使い易さは三つの側面を持っている。データベースの保守・運用，ユーザインターフェースそれにアプリケーション，特に汎用アプリケーション（パッケージ）とのインターフェースがそれである。データベースの導入，保守，運用の責任を負うデータベース管理者（ＤＢＭ）は，次の三点に於てシステムが使い易くなくてはいけないと考えている。

（ａ）データベース設置

データフォーマットの定義が簡単に行なえ，データをデータベースにロードするのが簡単にできるのが望ましい。特にそのためのユーティリティプログラムは少なければ少ない程よい。

（ｂ）データベースの拡張・変更

データベースを取りまく環境は必ず変化する。この変化は必然的にデータベースの論理的側面にも影響を及ぼす。この時，データベースの論理構造の変更が容易に行なえなければならない。データベースの物理的スペースの拡張，縮少もデータベース運用に影響を与えないでかつスムーズに実施できることが必要である。

（Ｃ）チューニング

チューニングはデータベースの物理的，論理的構造の変更を伴なう。データベース設計時に将来のアプリケーションの特性を確実に認識し，それに適する最適なデータベース設計を行なうことはかなり困難である。かえて不測の変化にすばやく適応できるようなシステムでなければならない。

ユーザとしてはアプリケーションプログラマのみでなく，エンドユーザをも考慮に入れなければならない。特に，山田〔12〕が述べているように，「利用者インタフェースの点からはホスト言語，ユーザ言語型に分けられるがホスト言語型ＤＢＭＳの利用が圧倒的に多数を占める。こうした現況から判断すると，階層型，ネットワーク型，あるいはホスト言語型という１９６０年代の設計思想の商用ＤＢＭＳが１９７０年代にようやく市場に定着し，実用化されるに至ったと見ることができよう。又，このパターンから推すと１９８０年代には，リレーショナル型，ユーザ言語型という１９７０年代の設計思想の商用ＤＢＭＳが一般にも普及し定着することが予想される」ことからもこのことは重要である。従って，ＲＤＢ／Ｖ１では，ユーザに学習，使用しやすい高水準言語インターフェースを提供することと，いわゆるユーザフレンドリインターフェースの充実が重要目標となっている。

次に，汎用アプリケーションパッケージとの関係について述べる。この点についてはDBMSインプリメンターが看過しがちなことであるが，1980年代はDBMSのみがエンドユーザ指向なのではなく，汎用アプリケーションパッケージもそれを目指しているのである。しかもデータベース機能の充実が，これらの汎用アプリケーションパッケージの急務になりつつある。これを考えると，RDB/V1は一方でアプリケーションパッケージの組み込みが容易に行ないうるオープンエンディドな（インタラクティブ）システム構造をとってはりかつ組み込まれたアプリケーションパッケージが提供する各種データ処理コマンドとRDB/V1がサポートするデータ操作コマンドがユーザにとって一元的に操作可能となるようなインターフェースをアプリケーションパッケージに提供することが必須である。

データ操作の高効率を保証するためにRDB/V1では豊富なデータ格納構造を提供しかつ問い合わせの最適化を行なっている。ユーザは，データ処理特性，データそのものの特性に応じて最適なデータ格納構造を選べることが必要である。一方，ユーザの問い合わせ言語であるデータ操作言語RDB/QLはデータ格納構造，アクセス補助手段から独立した高水準非手続型言語なので，データへのアクセス手順の決定はシステムの仕事である。このため，データ格納構造，アクセス補助手段の有無を考慮した最適な手順を決める最適化が必須である。

RDB/V1の対象ユーザについて考えよう。　ユーザは大きく分けて，アプリケーションプログラマとエンドユーザである。アプリケーションプログラマでは，特定のアプリケーションを開発するプログラマと汎用アプリケーションパッケージを開発するプログラマに大別できよう。前者はホスト言語インターフェースを使って特定のデータベース指向のアプリケーションを作る。後者は特定の分野（例えば地域情報システム）で汎用性のあるパッケージを開発する。この場合，データベース中のデータはそのアプリケーション特有の属性情報（意味情報）が付与されたものになるだろう。RDB/V1はそのための機能をも提供しなくてはならない。エンドユーザに対処するためには，RDB/V1はインタラクティブ（TSS）環境下で気楽にかつ容易にデータベースにアクセスできる環境を提供しなくてはならない。

一方，アプリケーションを業務の観点から分類すると大きく二つに分かれる。一つは日常の定型業務をサポートする定型的，トランザクション処理業務，他は研究，学術とか設計，計画に携わっているエンドユーザが主たる対象となる非定型・非トランザクション処理業務である。前者の業務では簡単なデータ問い合わせに対して処理効率が高くなければならない。後者では，どのような型の問い合わせに対しても，ある程度の時間内に答えを返すことが要求される。上述したように，RDB/V1は前者のような業務に対しては豊富なデータ格納構造を提供して，データベース管理者の専門知識が発揮できるようにする。後者に対しては最適化によってその要請に応えている。

2.　問い合わせ言語　RDB/QL

ここでは，始めにRDB/QLの全般的な特徴について述べ，最後にホスト言語インターフェースに関して簡単に触れる。

RDB/QLは，SEQUEL2，QUELに似た言語で一階述語論理に基づいてはいるが∀や∃などの論理記号を取り除き，代りに集合演算を導入したものである。構文的にはSEQUEL2に似ている。機能的には両言語とほぼ同等だがRDB/QLには存在するがQUELやSEQUEL2にはない重要な機能がある。読者は，両言語について基本的な知識を持っていると仮定し，以下では例題を用いてRDB/QLの概要を，主に両言語との差異に焦点をあてて簡単に記す。例となるデータベースは表2に書かれている。

検索

検索結果はXテーブルである。名前がユーザによって指示されない時はシステムが名前を与える。RDB/QLの特徴は1回の検索で複数のテーブルを作ることができることである。

例1 「従業員テーブルから従業員の個人的な情報と組織上の情報とを検索して，別々のテーブルに格納せよ。」

```
GET  ENO, NAME, ADDR  INTO  PEMP,
     ENO, DNO, MGR, SAL  INTO  OEMP
FROM  EMP;
```

この検索が実行されてもEMPテーブルはそのままである。

例2 「プロジェクトの予算を部毎に総計し，それのプロジェクト全体の予算との%を求めよ。」

従業員テーブル
EMP(ENO, NAME, ADDR, DNO, MGR, SAL)

部テーブル
DEPT(DNO, DNAME, LOC)

(注) MGRは従業員のマネージャ

プロジェクト テーブル
PRJ(PRJNO, PNAME, BUDGET, DNO)

作業テーブル
JOB(PRJNO, ENO, JOBDESC)

表2 人員配備データベース

GET DNO, SUM(BUDGET)/SUM(BUDGET

FROM PRJ)

*100

FROM PRJ

GROUP BY DNO;

この検索結果の名前はTEMPTBL(システムがつける。ユーザによる変更可能)で、このテーブルを又検索対象にすることも可能である。

例3 「部がおかれている場所毎の従業員数を求めよ。」

```
GET COUNT(EMP.DNO), LOC
FROM EMP, DEPT
WHERE EMP.DNO = DEPT.DNO
GROUP BY LOC;
```

更新,削除

RDB/QLの1つの特徴として複数テーブルの同時更新,削除の機能がある。これは検索の結果が複数のテーブルになる機能と対をなすものである。

例4 「従業員をEMP,JOBテーブルから削除せよ。」

```
DELETE EMP, JOB
FROM   EMP, JOB
WHERE  EMP.ENO = 111111
AND    EMP.ENO = JOB.ENO;
```

ホスト言語インターフェース

RDB/QLのホスト言語用問い合わせ言語はインタラクティブエンドユーザ用コマンドと基本的には同じであるが次の2点で異っている。

(1) レコード・インターフェースと集合インターフェースを用意している。集合インターフェースにはリスト型式と配列型式がある。この内どのインターフェースが可能かはホスト言語の性質による。

(2) テーブルの順次アクセスインタフェースを用意している。

ホスト言語中の変数名を問い合わせコマンド中に使うことができる。この時，ホスト言語の変数名であることを示すために特殊記号をその先頭に付ける。その変数の宣言も同様な方法で明示される。プリプロセッサはそれらの特殊記号を頼りにソース中のＲＤＢ／ＱＬコマンドを識別し，それをホスト言語のサブルーチン呼び出し列に変換する。

３．　システム構成

システムの論理構成を図１に示す。ＲＤＢ／Ｖ１がサポートするデータベースにアクセスするにはＴＳＳ環境下でインタラクティＲＤＢ／Ｖ１を動かすかホスト言語でプログラムを書くかする。インタラクティブＲＤＢ／Ｖ１は，コマンドインタフェースを軸にしてデータ操作コマンドを解釈実行するInteractive Subsystem for Relational Queryとデータ処理を目的とするインタラクティブなAplication Subsystemとが統合される。ＡＰＳは任意にＲＤＢ／Ｖ１に組み込むことが可能である。即ち，ＲＤＢ／Ｖ１はオープン・エンディドなシステムである。ＩＳＲＱは次の四つの機能をモジュールに分れる。構文解析，意味解析（データ辞書とのつき合わせ），実行スケジュール（最適化）及び解釈実行モジュールである。

ＩＳＲＱ，ＡＰＳ及びApplication ProgramはData Storage and Control Subsystemとインタフェースを持つ。ＤＳＣＳの機能は，(1)データ編成法とアクセス法の提供，(2)排他制御，(3)ログ取得とリカバリ，(4)ページＩ／Ｏ管理である。

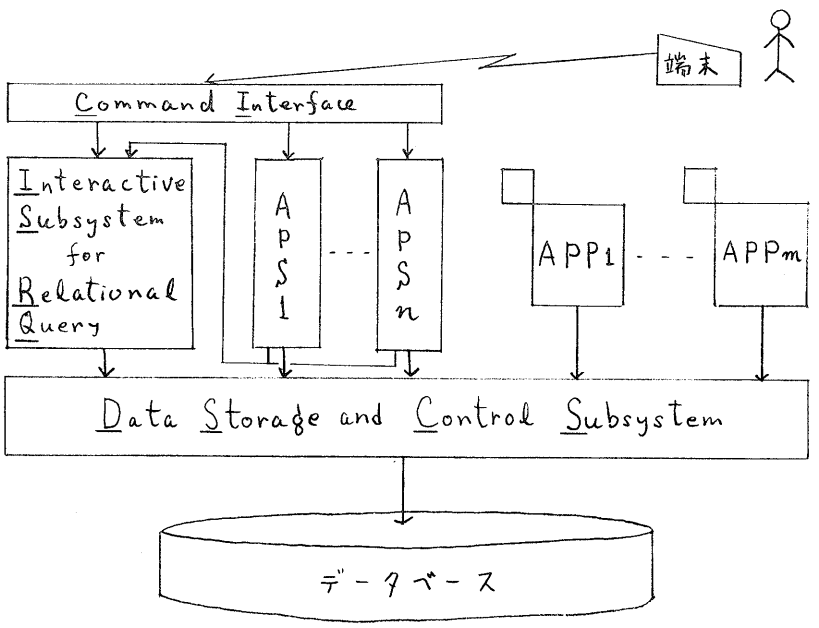

図１　ＲＤＢ／Ｖ１の構成

ホスト言語に埋め込まれたＲＤＢ／ＱＬがプレコンパイルされる時はＩＳＲＱに似たCompiler for Relational Queryが動く。　ＣＲＱはソースを読み込み，そこに埋め込まれたＲＤＢ／ＱＬを認識し，それを一連のＤＳＣＳのサブルーチン呼び出しに変換する。ＣＲＱの機能モジュール構成は，ソース走査モジュールが付加されることと解釈実行モジュールが文字列出力モジュールに代ることとを除けばＩＳＲＱと同じである。図２にプレコンパイルの概念図を示す。
ＣＲＱも実行中対象データベースにアクセスする。それはデータ辞書がデータベース化されているためである。

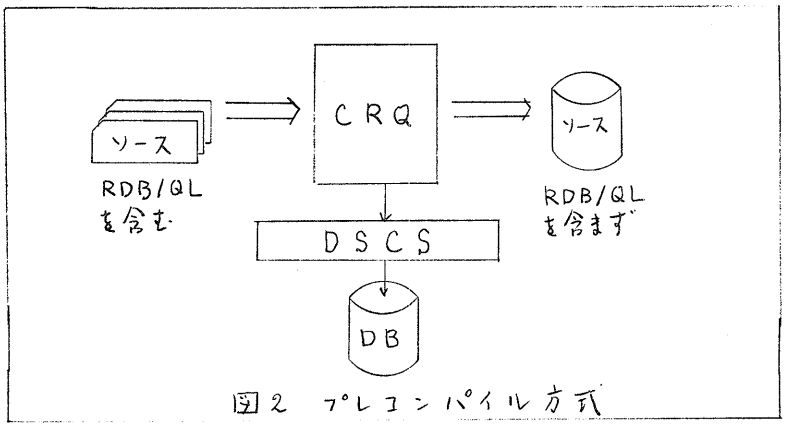

図２　プレコンパイル方式

４．　ＤＳＣＳ

セグメント，ページ

ＲＤＢ／Ｖ１がサポートする一つの格納スペース基本単位はセグメントと呼ばれる。これはユーザ側からみれば一つの独立したデータベースであり，システム側からみれば一群のファイルである。システムから見たセグメントは４種類のファイルからなる。それは，(1)マスタファイル，(2)ボリュームスロットファイル，(3)ボリュームスロットマップファイル，(4)ページテーブルファイルである。ボリュームスロットとボリュームスロットマップは対になって一つのエクステントを形成して，セグメントに複数対な化しうる。これによりデータベースの拡張が容易にできるようになっている。図３はセグメントの構成を示す。

レコード（タップル）が格納されるのがボリュームスロットファイルでそれは４Ｋバイト長のスロット（物理ページ）に切られている。ボリュームスロットマップファイルはスロットの占有状態を示すビットマップである。ページテーブルファイルはセグメントの論理ページと物理ページとのマッピングを行なうためのページテーブルを格納している。ページテーブルのエントリｉ（ｉ論理ページ）に

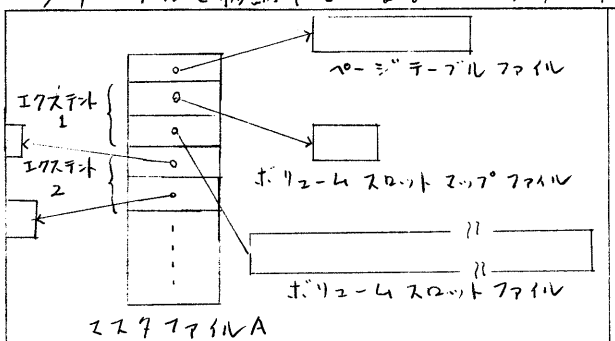

図３　セグメントＡの構造

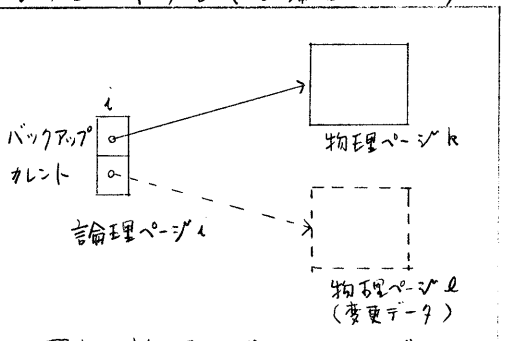

図４　論理ページのマッピング

は二個の物理ページ番地が格納できる（図4参照）。この構造はSystem R[2]でも採用された。この方法の利点は，データが変更されても，元のデータはそのままとっておかれるのでリカバリ用のログデータとしてbefore imageを変更時点でとる必要がないことである。

ページI/O管理

DSCSではページのI/O管理を次の二つの面で行なっている。一つはページの先読み，先書き制御，他はバッファ管理である。データベースがオープンされるとバッファプールが開設される。バッファプール全体はLRU方式で制御される。テーブルがオープンされるたびにバッファプールから適当な個数のバッファがそのテーブル毎に割り当てられる。テーブルのアクセス方法はすべて順次アクセスだからページが処理されると，それが載っているバッファはバッファプールに返される。テーブル毎に割り当てられるバッファの数はそのテーブルの先行読み込みのページ数を決める。一般にnページの先行読み込み制御を行なうためには2n+1個のバッファを必要とする。このnを決定するのは実行スケジュールモジュールで，テーブルのサイズ，スキャン速度を考慮して行なわれる。

テーブル編成法とアクセス法

テーブルの編成法は三種類ある。(1)レコード到着順編成，(2)キー順編成，(3)ハッシュ編成。(2)の編成のテーブルをインデックステーブル，(3)のそれをハッシュテーブルと呼ぶ。テーブルと書けば特にことわらない限り(1)のテーブルのことである。インデックステーブルの構造はB*-tree構造である。RDB/V1がサポートするインデックスはインデックステーブルで実現されている。即ちキーとレコード識別子又は一次キーの対をレコードとするテーブルである。ハッシュテーブルに対してはシステムが標準ハッシュ関数を用意しているがユーザ定義の関数も使えるようになっている。DSCSはこれら3種のテーブルに関するオペレータを提供する。それは(1)テーブルの創成，削除，(2)テーブルのスキャン（レコードの識別子とフィールドの値を返す），(3)レコードのフェッチ（レコード識別子とフィールド番号を指定する），(4)レコードの挿入，削除，(5)フィールドの更新である。原子的でないフィールドを扱えるようにするために(6)レコードスキャンが用意されている。これにより，制約はあるが，集合フィールドをうまく扱えるようにしてある。

リカバリとログ

RDB/V1は次の3種類の事故に対するデータ回復手段を提供する。(1)ユーザプログラムフェイリュア，(2)システムダウン，(3)二次記憶媒体の障害。

データ回復（リカバリ）という概念に付随してトランザクションという概念が必要である。RDB/V1では陽に明示しなければ1コマンドが1トランザクションである。明示すれば複数コマンドが1トランザクションになりうる。(1)の事故は，ユーザプログラム（あるいはプログラム走行環境）が原因でこれ以上データ操作が続けられなくなることをいう。このような状態が生起しかつそれをRDB/V1が検知できる時にはデータは当該トランザクションが開始される直前の状態に戻される。これは，トランザクション中に新規に取られた論理ページ，物理ページを解放し，変更が行なわれたページに対応するページテーブル内エントリのカレントポインタをバックアップポインタに切り換えることにより行なわれる（図4参照）。これはSystem Rでプライベートセグメントのリカバリで用いた

方法と同じであるが，RDB/V1では共有セグメントにまで適用できるようにしてある。これにより，TSS環境下での時間切れ，作業スペース不足などの不測の事態に非常に効率よくかつ他のユーザから独立にデータ回復が行えるので便利である。(2)，(3)のデータ回復にはログをとる必要がある。ログの内容としては普通，データのbefore imageとafter imageがある。before imageを取得すると，取得した時点で必ずログファイルに書き出し，その成功を確認しないとトランザクションは前に進めない。しかもトランザクションが正常に終了しないと既に取られたログを取り消す必要がでてくる。一方，after imageは取得ログデータをバッファ内に一時保存しておいて，トランザクション終了時にまとめて書き出しを行なえる。トランザクション不成功時にはバッファを無効にするだけでよい。この事情を考えて，RDB/V1ではログデータとしてはafter imageのみを取得する。(2)に対するクイックリカバリについて述べる。after imageの取得のみで済ませるために，ページテーブル，ボリュームスロットマップのファイルは2重化する。両ファイルの変更されたブロックは，カレントファイルの対応ブロックに書き出される。バックアップファイルが最新の内容にされる契機は，当該セグメントの最後のユーザがアクセスを終了した時かチェックコマンドによって強制された時である。このコピー作業には2つのファイルが使われ，決して，データが失われないようにされる。セグメント毎のログファイルに取得されるログデータはトランザクション中に新たに取られた論理ページ，物理ページ番号と解放された論理/物理ページ番号である。この情報はファイル内のブロック内レコードの変更情報をログデータとするより大巾に小さい。システムダウンが起こるとログファイルを使って終了していないトランザクションの直前の状態のカレントファイルをバックアップファイルから復元する。この方式の欠点はバックアップファイルが最新の状態を反映していた時刻からダウン時までに行なわれた変更の量に復元作業量が比例することである。しかしデータベースが巨大であってもページテーブル，ボリュームスロットマップは比較的小さいので時間はそれ程問題にならないと考えられる。又同じ理由により定期的にチェックポイントを発行しても，コピー作業の時間は問題にならない。(3)のためのリカバリにはトータルダンプとページ内変更レコードに関するログデータを取る必要がある。

排他制御と機密保護

排他制御の最小単位はページである。必要に応じてテーブル，セグメントにもロックをかけることができる。どの単位にどの種類のロックをかけるかは問い合わせのパターン，選ばれたアクセスパス，スキャンするページ数に依存する。これを決定するのもスケジューラの役割である。ロックスケジュールの目標はロックのオーバーヘッド(スペース，時間)と衝突トランザクションの数の均衡である。RDB/V1ではパスワードによって機密保護を実施する。保護の単位はセグメント，テーブル，ビューである。フィールド単位の保護はビューを通して行なう。アクセス権利の種別はほぼコマンドの数程ある。これにより細かいアクセス制御が実施できる。なおアクセス権利の授与/剥奪権利はただ一つの人(パスワード)にしか許されない。又，誰もがパスワードなしに特定のアクセスが許されるようにオブジェクトに特別のパスワードを付与することもできる。

5. 問い合わせ評価の最適化

リレーショナルな問い合わせを評価する時に最適な方法，手順を選択することが問い合わせの最適化と呼ばれる。Kimの報告[7]によれば，彼の取り上げたシステムはどれもが何らかの最適化を実施している。RDB/V1も例外ではない。以下に，RDB/V1でとられた最適化の考え方について概略する。

Yao[11]はこれまでの成果を分析し，問い合わせ評価方法のモデル化と各モデルのアクセスコストについて論じた。但しジョインについては二つのテーブルの場合であって三つ以上のテーブルのジョインのモデル化はなされていない。

実際のシステムでは，システムが提供するアクセス方法が限られているとはいえ，特定の問い合わせを評価する時に考えうる手順は組み合わせ数学的に増大する。それぞれ一つ一つに対してコストを予測し，その結果によって最適な手段を選ぶことは事実上不可能である。従って実際のインプリメントではインプリメンターの経験，直観によって評価方法のモデルが決定されている([1],[2],[9],[10])。

RDB/V1が採用しているヒューリスティックな方法論は次の主要目標にまとめられる。(1)ジョイン，コリレーションの演算では対象フィールド(ジョインフィールド)がソートされている(されていなければソートする)。(2)リダクションを早めに使ってテーブルのサイズを小さくする。(3)リダクション時に必要なソートして後のオペレーションにそなえる。(4)中間テーブル(一時的に作られる作業用テーブル)の数は最小にする。(5)スキャンの範囲を可能な限り狭くする。(6)同一テーブルのスキャン回数は出来るだけ減らす。

RDB/V1は問い合わせのパターンを認識し，情報を収集して上記枠内でかつ上記の目標を達成する手順を導く。普通複数個の手順が選択対象となるので，その時コストの予測を行い，より小さいコストと予測された手順をみつける。コストの比較が必要な場合の例としてはインデックスを使用するかしないかの決定がある。一般にコストの予測では，予測し切れないパラメータがなれる。このような場合，ユーザ(DB管理者)がパラメータを与える手段を用意されている(そのパラメータは動的には収集されない)。

```
GET NAME, SAL, JOBDESC
FROM JOB, EMP
WHERE JOB.ENO = EMP.ENO
AND DNO = 10 ;
```

表3. 典型的なジョイン例

表3の例をとって説明する。EMPテーブルのENOにインデックスI(ENO, TiD)が設定されていると仮定する。JOBテーブルはスキャンされENOでソートされた一時テーブルTJOB(ENO,JOBDESC)が作られる。次の段階で二つの選択がある。

(A) EMPテーブルをスキャンする。DNO=10でリダクションし，ENOでソートされた一時テーブルTEMP(ENO, NAME, SAL)を作成する。TJOBとTEMPをスキャンして各々のENOフィールドの値の一致を調べる。

(B) I(ENO, TiD)とTJOBとをENOでジョインしてTiDでソートされた一時テーブルT(TiD, JOBDESC)を作る。TiDを使ってEMPテーブルのレコードをフェッチする。(A), (B)の選択時にコスト計算が行なわれる。(A)のコスト予測ではTEMPのサイズが重要なファクターである。これはDNO=10のフィルタリング係数に依存する。DNOにインデックスが張られていれば，その係数の値が小さくなる。(B)ではTのサイズの予測が重要である。この予測にはTJOB作成時に得られたMAX(ENO), MIN(ENO), Iの性質(MAX(ENO), MIN(ENO), 異なりオーダ，分布等々)及びジョインフィルタリング係数が関係する。

6. アプリケーション インターフェース

RDB/V1はサポートしていないが，アプリケーションが必要とするデータ属性をデータ辞書に定義できる。例えばそれはフィールド値の単位とかコード化の種別である。それはテーブル定義時に意味素（SEMANTEME）として与える。この機能はRDB/V1と共に利用アプリケーションパッケージが作動する時重要なものである（図1参照）。例えば，地域情報システム［6］では地域をメッシュに分けてメッシュ毎にデータを保持する。システムがサポートするメッシュコードのタイプは複数個ある。同システムが定義する関数でRDB/V1に組み込まれ，WHERE条件中で呼び出す関数（例えばCIRCLE）では対象メッシュのコード種別を知る必要がある。又RDB/V1では検索結果のテーブルのデータ辞書に検索対象のデータ属性を遺伝させる機能もある。

これらの機能によりRDB/V1がサポートする検索コマンドと組み込まれたサブシステムがサポートするデータ処理コマンドがデータを通して有機的に結合されるので，ユーザに全体として一元化されたコマンド体系を与えることが可能になる。

7. 結論

RDB/V1は現在TSS環境下でシングルユーザモードで稼動中である。あるユーザでは地域情報システムがサブシステムとして組み込まれ，アプリケーションコマンドと共にエンドユーザに強力なコンピュータファシリティを与えている。これらの実働経験からすると，インタラクティブ環境での問題解決手段としてのエンドユーザ指向システムとしては充分ユーザの期待に応えることができると確信する。一方，ホスト言語インターフェースは設計段階であり，RDB/V1のトランザクション処理システムとしての可能性はまだ未知であるがかなりの程度まで行けそうだという感触は得られている。

RDB/V1はエンドユーザ時代の幕開けを告げるにふさわしいシステムであると確信している。

8. 参考文献

[1] Astrahan, M.M. et al. 'Implementation of a Structured English Query Language' C.ACM Vol.18, No.10, October 1975

[2] Astrahan, M.M. et al. 'System R: relational approach to database management' ACM TODS, Vol.1, No.2, June 1976

[3] Blasgen, M.W et al. 'Storage and access in relational data bases' System Journal No.4 1977

[4] Codd, E.F. 'Extending the Database Relational Model to Capture More Meaning' ACM TODS Vol.4, No.4, December 1979

[5] Hall, P.A.V. 'Optimization of Single Expression in a Relational Data Base Syotem'

[6] 神田 et. al. 「関係データベースを核とした地域情報システムについて」

[7] Kim, W. 'Relational Database Systems'
ACM Computing Surveys Vol. 11, No. 3, September 1979

[8] Smith, J. M. et. al. 'Optimizing the Performance of a Relational Algebra Database Interface' C. ACM Vol. 18, No. 10, October 1975

[9] Stonebraker, M. et. al. 'The design and implementation of INGRES'
ACM TODS, Vol. 1, No. 3, September 1976

[10] Wong, E. et. al. 'Decomposition – a sTrategy for query processing'
ACM TODS Vol. 1, No. 3 September 1976

[11] Yao, S. B. 'Optimization of Query Evaluation Algorithm'
ACM TODS Vol. 4, No. 2, June 1979

[12] 山田 et. al. 「日本の商用DBMS利用実態とユーザ評価」
コンピュートピア 3月号 1980