関係データベース管理システム RDB/V 1†

牧之内 顕文†† 手塚 正義†† 北上 始††
安達 進†† 佐藤 秀樹†† 泉田 義男††
中田 輝生†† 石川 博††

RDB/V 1は関係（リレーショナル）データモデルに基づいたエンドユーザ向けデータベース管理システムである．その問い合わせ言語 RDB/QL は SQL や SEQUEL と同等の機能をもつ．ユーザはデータ操作を，端末を通して対話的に行うことも，親言語インタフェースを使ったプログラムを通して行うことも可能である．RDB/V 1は開放型 (open-ended) システムとして作られているので，任意の適用業務サブシステムを組み込むことができる．この環境では，ユーザはデータの検索・更新ばかりでなくデータ加工・分析もまた対話的に行うことができる．本論文では，RDB/V 1の設計と実現法について，他の類似のシステムから本システムを区別する面—問い合わせ言語の新しい機能，システム構造，最適化，ログとリカバリ機構—に焦点を当てて述べる．なお最適化についての詳細は別論文に譲り，本論文では考え方のみを記す．

1. 序 論

RDB/V 1は Codd[4]のデータモデルに準拠した関係（リレーショナル）データベース管理システムである．このシステムは，（1）関係モデルの構造的特徴，（2）挿入・更新・削除規則，および（3）関係代数と同等な能力を有するデータ準言語をサポートしている点[5]で完全関係型* (fully relational) である．文献 13)にならってシステムのプロファイルを**表1**に掲げる．

RDB/V 1 の設計目標は，使いやすさ，効率的なデータ操作，広範囲の適用業務への適合性であった．使いやすさに関していえば，三つの側面—データベースの生成，保守および利用—に工夫を払った．すなわち，データベースを構成するファイル構造を単純にしてデータベースの生成・保守作業に携わる管理者の負担を小さくした．また，問合せ言語 RDB/QL にデータ定義，操作および制御の諸機能を統合し[1]，さらにデータ操作とデータ処理とを統合化するための機構を用意した．効率的なデータ操作を保証するためには二つのことがきわめて重要である．一つは，ユーザが適用業務に応じて最適なデータ格納構造が選択できること，また状況に応じてチューニングを行うためのアクセス補助手段（インデックス等）の動的な変更が可能なことである．そしてこれら物理的環境の変化に応じて問合せ評価の最適化をシステムが行うことが2番目である．

RDB/V 1 では到着順，キー順およびハッシュの3種類のデータ編成法を提供している．一方，最適化については，局所的最適化ばかりでなく大局的最適化をも考慮していることと，マージスキャン法[2),20)]の拡張とその適用範囲をコリレーションにまで拡大していることが特長[14),17)]である．この最適化技法により，一つのテーブルに関わる単純検索から複数テーブルに跨がる複雑なものまで法外でない応答速度が保証される．

広範囲の応用を目指して，RDB/V 1 では，プレコンパイル方式による親言語インタフェースを用意するばかりでなくエンドユーザのための対話型インタフェースも備えた．さらに対話型 RDB/V 1は開放型システムとして設計されているので，汎用業務サブシステムを組み込むことによってエンドユーザはデータの検索ばかりでなく，検索されたデータの加工・処理をも一連のコマンドの組合せで対話的に行える．

本論文は RDB/V 1に関する3編の論文[14),18)]の一つである．以下，システム構成，問合せ言語とその評価戦略，ビュー機構，排他制御およびリカバリ方式について述べる．ただし問合せ評価についての詳細は他の論文[14)]に譲る．

2. システム構成

図1にシステム構成を示す．

† Relational Database Management System RDB/V 1 by Akifumi Makinouchi, Masayoshi Tezuka, Hajime Kitakami, Susumu Adachi, Hideki Sato, Yoshio Izumida, Teruo Nakada and Hiroshi Ishikawa (Fujitsu Laboratories Ltd.).

†† 富士通研究所

* 1981 Turing Lecture[6] では完全関係型 DBMS の条件を強めているがこの論文では文献 5)の解釈に従っておく．

表 1 RDB/V 1 のプロファイル
Table 1 Profile of RDB/V 1

名称	RDB/V 1
作動開始年	1979 年
計算機	FACOM M シリーズ
開発言語	SPL/100
現状	継続発展中（分散化）
タイプ	研究用／商用
開発者	富士通研究所

機能	○サポート，×サポートなし
最適化	○
ビュー	○
機密保護	○
インテグリティ	×
競合アクセス管理	○
リカバリ	○
報告書生成	×

対話型 RDB/V 1 は，Command Interface (CI), Interactive Subsystem for Relational Query (ISRQ) および複数の Application Subsystem (APS) からなる．CI の役割りはコマンドを ISRQ か APS かに振り分けることである．APS は ISRQ または Data Storage and Control Subsystem (DSCS) インタフェースを使って開発され，システムに組み込まれる．ISRQ はコマンドを解釈実行する．それはパーサ，意味チェッカ，スケジューラおよびインタプリタの四つのモジュールからなる．問合せ評価の最適化は対象テーブル* に関する情報(データ辞書中に格納される)を参照しながらスケジューラが行う．

RDB/QL の記述を含む Application Program (APP) は Compiler for Relational Query (CRQ) で前処理される．その時点で各 RDB/QL コマンドは一連の DSCS サブルーチン呼出し列に翻訳[19]される．

DSCS は，データ編成とアクセス法，排他制御，ログとリカバリ機能をサポートする．またページ I/O のためのバッファ管理も行う．

RDB/V 1 データベースの論理空間の単位はセグメントとよばれる 4k バイト長のページの集合である．セグメントには，(1)テーブルなどのオブジェクトは複数セグメントにわたらない，(2)セグメント間で共有されるオブジェクト（たとえばシステムカタログ）はない，(3)リカバリの単位であるといった性質がある．ただし，一つのデータベースが複数のセグメントに跨がることはかまわない．このようにセグメントはそれ自身で閉じた存在であり，データの分散の単位でもある．

一つのセグメントは 5 種類のファイルからなる．

(1) マスタファイル—これはセグメントのカタログである．ここには他の構成要素ファイル名やパスワードリストが格納される．

(2) 一対（カレントとバックアップ）のページテーブル (PT) ファイル—これは論理と物理ページ（スロットともよぶ）番号の対の表である．

(3) 4k バイト長の物理ページからなるボリュームスロット (VS) ファイル．

(4) VS ファイル中の各 VS の使用状態を示す 1 対（カレントとバックアップ）のボリュームスロットマップ (VSM) ファイル．

(5) ログファイル（オプション）．

図 2 にセグメントのファイル構成を示す．VS と VSM とは一個のエクステントを形成し，データベース空間の増量単位となる．PT と VSM のファイルが二重化されているのはリカバリのためであり，これについては 6 章で論じる．

RDB/V 1 がサポートするオブジェクトはテーブルとインデックスである．インデックスはテーブルへのアクセス補助手段である．テーブルには 3 種類—到着順，キー順およびハッシュ—がある．

これらオブジェクトは DSCS がサポートするデータ編成とアクセス法で実現される．データ編成法には上記種類のテーブルに応じて 3 種類ある．インデックスは DSCS レベルのキー順編成テーブルを使って ISRQ がサポートする．このキー順編成テーブルはリーフノードがリンクされた B木構造をとっていて，レコードはすべてリーフノードに格納される．またくり返し項目を許しているのでこのテーブルでインデックスを実現するのは容易である．

ハッシュ編成テーブルは連続した $n(\geq 1)$ 個の論理ページをトップノードとする n 個のキー順編成テーブ

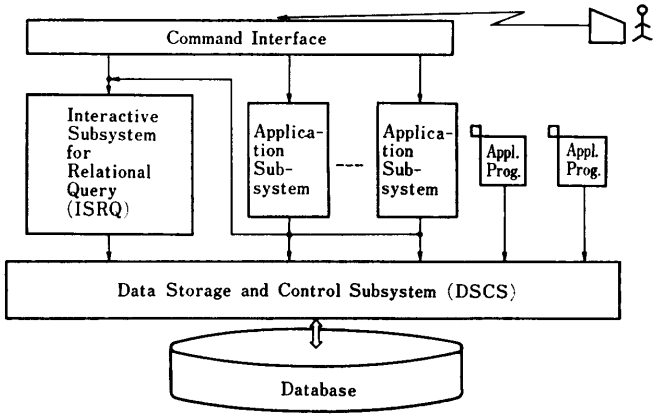

図 1 RDB/V 1 のシステム構造
Fig. 1 Logical structure of RDB/V 1.

* 以下テーブルとリレーション，レコードとタップルは同義で使う．

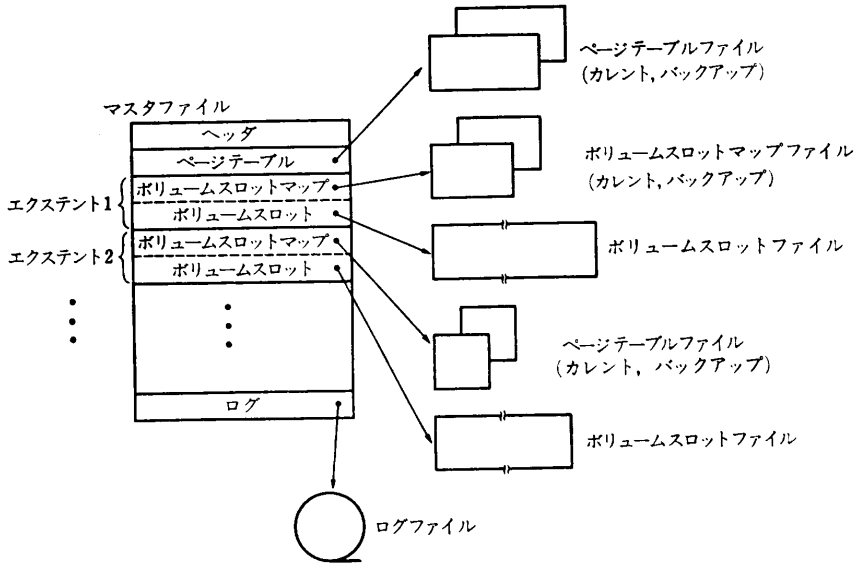

図 2 セグメントを構成するファイル群
Fig. 2 Files for a segment.

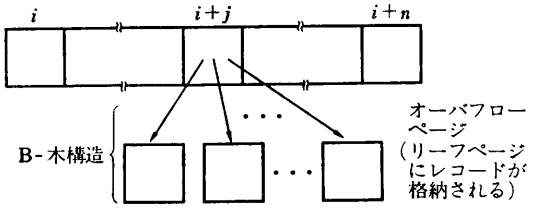

図 3 ハッシュ編成テーブルの構造：n 個のバケツ（i ページから $i+n$ ページ）からなるテーブル．$i+j$ ページはキーの衝突のためオーバフローしている．
Fig. 3 Structure of a hash relation with n bucket pages: page $i+j$ is overflowed and a B-tree is formed with page $i+j$ as the top node.

ルで構成される．したがってハッシュキーの衝突によるオーバフローが生起した場合，最悪でもキー順テーブルと同等のアクセス性能が保証される（**図3** 参照）．ハッシュテーブルのこの構成法の他の利点は，検索時に利用されるキーとしてハッシュとソートの二つのキー*を指定できることである．これを利用すれば国語辞書をその見出し語の最初の1文字で分割格納しかつ各部分を見出し語の辞書引き順に整列するということが可能である．このような方法は辞書のデータベース化に有効である[11]．またセグメントごとに作られるデータ辞書もオブジェクト名をキーとするハッシュテーブルによって実現されている．

* ただしソートキーはハッシュキーの部分でなくてはならない．

3. 問合せ言語とその評価法

問合せ言語 RDB/QL で操作の対象となるオブジェクトはテーブルとインデックスである．これらは任意の時点で任意のセグメント内に（から）定義（削除）できる．テーブル内のレコードの検索，更新，削除および挿入はおのおの GET, UPDATE, DELETE および INSERT コマンドで行う．ユーザはこの操作の対象となるテーブル名と検索（更新または削除）されるべきレコードが満たすべき条件をコマンド中に指定する．これらのレコード操作は対象テーブルのタイプや値をテストすべきフィールド上のインデックスの有無には依存しない．

RDB/QL の構文および意味は SQL[1),12)]や QUEL[22)]によく似ている．しかし二，三の新規機能が追加されている．一つは一個の GET コマンドの結果として複数個のテーブルを得る機能であり，他は一つの DELETE (UPDATE) コマンドで同時に2個以上のテーブルのレコードを削除（更新）する機能である．付録Ⅰに例を示す．

RDB/V 1 では検索された結果のテーブルは，とくに指定されなければ作業用セグメントに格納される．システムの作業エリアとしても使われる作業用セグメントは RDB/V 1 起動時に作られ，終了時に消される．ただし作業用セグメントに名前をつけて保存することも可能になっている．保存されたセグメントをシ

ステム起動時に作業用セグメントとして指定すれば，その中に保存されたテーブルは支障なく再利用できる．

関係データベース管理システムにおける問合せ評価の最適化問題はシステム実用化の面から見て最重要問題と考えられ，多くの研究がなされてきた[10),20),24)]．実際文献 13)に報告されているすべてのシステムは何らかの最適化を行っていることからみても最適化が関係データベースシステムの成功の鍵であるといっても過言ではない．

最適化で考慮されるべきことは次の三つである．

（1） レコード選択条件（述語* を論理演算子で結合した式）の評価順序．

（2） 述語* の評価手法の選択．とくに結合 (join) 述語の評価法は種々提案されている[2),24)]．

（3） 述語を評価するために使用するアクセスパスの選択．

したがって最適化問題は，述語の種類，評価手法の多様さおよび利用可能なアクセスパスの種類の三者の組合せ問題となる．これら三者の可能な組合せのなかから最適なものを選ぶ基準は問合せを評価するために要するコストである．RDB/V 1 では評価のためにアクセスされると予想される(論理)ページ数をコストとして設定している．このコストの算定のために RDB/V 1 では SYSTEM R と同様，テーブルおよびインデックスに関する統計データをデータ辞書中に収集して使っている．ただし，RDB/V 1 では SYSTEM R と違って収集は動的に行われる**．これが可能なのは，RDB/V 1 においてはデータ辞書がセグメントに分散することとオブジェクトのデータがハッシングによって辞書の中で分散することにより，ロック競合による性能低下が比較的軽いと予想されるためである．

一方，最適化における組合せ数の極端な増加を避けるために RDB/V 1 では，(1)に対しては経験則に基づいて述語の種類を分類し，それによって評価順序を一意的に決定する，(2)に対しては結合述語の評価手法としてマージスキャン法[20)]のみに限定している．ただしそれに最大最小法[14),17)]を導入してこの手法だけでは得られないが他の手法(たとえば多重ループ法[20)])では得られるであろう効果が発揮されるようにするとともに結合述語の評価時におけるテーブルアクセスの順序決定の問題も解決している．(3)に対してはテーブル，インデックスとも始点と終点で限った区間の走査のみを採用することによって対処している．たとえばインデックスをダイレクトアクセス手段として使用しないことからくる欠点は最大最小法の適用[17)]によって実際的に回避される．

以上の工夫により，最適化アルゴリズムを大幅に簡素化すると同時に最適化のためのコストを減少させている．

4. ビュー機構

本節では，RDB/V 1 がサポートするビュー (view) についてその機構と最適化のための機能について述べる．

RDB/V 1 では，GET コマンドで得られるどんなテーブルもビューとして定義できる．定義されたビューは仮想テーブルであり，それに対する検索* コマンドが実行されるたびにビュー定義で参照されている実テーブルがアクセスされる．

ビュー定義は解析され，その解析木が内部表現の形でユーザが定義したビューテーブルに格納**される．ビューが問合せの中で参照されると，参照されたビューの解析木が検索される．この時点でシステムは問合せ解析木とビュー解析木の二つを保持することになる．ビュー解析木が問合せ解析木に組み込まれるようなビュー展開を「マクロ型展開」といい，ビューを実行した結果のテーブルを得て，問合せ中のビュー参照をそのテーブルの参照に変更するような展開を「手続き型展開」とよぶ．

どの展開法がとられるかはビューの定義と問合せ中のそのビューへの参照の仕方に依存する．ビュー解析木中の述語と問合せ解析木中の述語を論理演算子で結合して論理式にできれば，スケジューラによる最適化が有効に働く機会は多くなる．したがって最適化の観点からはマクロ展開法が望ましい．この理由から RDB/V 1 では，両解析木を照合して，マクロ展開法が可能かどうかを調べる．もしビュー定義に 'GROUP BY' 句がなくかつ問合せが，ビュー定義中で算術式または集合関数で定義されるフィールドを参照しては

---

* ここでの述語とは〈フィールド名〉〈比較演算子〉〈定数またはフィールド名〉や〈フィールド名〉〈集合メンバシップ〉(検索コマンド) 等のことである．ただし比較演算子は {=, ¬=, >, >=, <, <=}，集合メンバシップは {IN SET, NOT IN SET} である．

** SYSTEM R ではシステムカタログへのロック衝突による性能低下等を理由に統計データ収集を DB 管理者の手にゆだねた[20)]．

* ビューに対しては現在のところ検索のみが許されている．またビュー上にビューを定義することはできない．

** ビューテーブルは任意のセグメントに定義できる．

いないときにはマクロ型展開法がとられる．その他の場合には手続き型展開法がとられる．後者の場合でも，ビューが定義されている実テーブルを制限 (restrict) するような述語を問合せ解析木中に探し，もしあればそれをビュー解析木中に組み込む．こうして，ビュー実行後に作られる結果テーブルのサイズができるだけ小さくするようにする．

RDB/V 1 のビュー機構が備えている他の機能にテーブルの「水平合成」による最適化がある．

いまテーブル $A(a_1, a_2, \cdots, a_n)$ (ただし A はテーブル名，$a_i(i=1, 2, \cdots, n)$ はフィールド名) が $A_1(a_1, a_2, \cdots, a_i=v_1, \cdots, a_n)(+)A_2(a_1, a_2, \cdots, a_i=v_2, \cdots, a_n)$ に等しいとき (ただし v_1, v_2 は定数，'(+)' は append 演算子)，A は A_1 と A_2 に「水平」分解されるという．このとき，フィールド a_i を仮想フィールドという．逆に，Aを実テーブル A_1 と A_2 の上に上記の式で定義されたビューとすることができる．このときAは「水平合成」されたという．このとき，仮想フィールドの値はビュー中にのみ存在し，実テーブル中にはない*．こうすることによりデータベースの格納スペースが節約される (スペースの最適化)．

一方，このように定義されたビューに対する問合せ評価の最適化は，アクセスすべき実テーブルの最適な選択 (いいかえると，不必要なテーブルにはアクセスしないということ) である．この選択は，仮想フィールドに対する述語を解析することによって行われる．もしビューに対する述語に '$a_i=v_j$' があればこの述語を満足する実テーブルのみが検索対象として選ばれる．

RDB/V 1 のこのビュー機構は，統計データベースへの応用できわめて有効である．統計解析においては同じデータをときには時系列データとして，ときにはクロスセクションデータ (年次，月次等) として見たいということがよく起こる．このとき，たとえば年次データを年次ごとにテーブル化してデータベース化し，それらの時系列データは年次フィールドを仮想フィールドとしたビューと定義すればよい．こうすることによる利益は，すでに議論した利点ばかりでなく，発生したデータの格納場所を以前 (すなわち前年度まで) のデータの格納場所と独立にしておけることにもある．さらに新しいテーブルのフィールド構成は必ずしも以前のテーブルとまったく同じでなくてもよい (別の項目についてデータを収集しておいてもよい)．

5. 排他制御

RDB/V 1 では2種類のセグメント—利用と公用—を実現している．私用セグメントは，一人のユーザが使用時に占用する．したがってセグメント内のテーブルのアクセスに関してはロックをかけるというオーバヘッドはない．

一方，公用セグメント内のテーブルは複数のユーザが同時にアクセスしうるのでシステムはロックによる競合管理を行っている．RDB/V 1 で資源のアクセスを競うのはトランザクションである．トランザクションは1個の RDB/QL コマンドかトランザクションの開始，終了コマンドに囲まれた複数のコマンドである．トランザクションは，そのなかでアクセスする資源に対してロック要求を出す．ロックの要求と解放は2フェーズ方式[8] に従っている．

システムはロック要求を集中管理してデッドロックを防止する．それはデッドロック原因者となると予想される最新のトランザクションをロールバックすることにより行われる．

ロック要求の対象となる資源はセグメント，テーブル，インデックスおよびページである．ただしハッシュテーブルはそのなかで2階層に分かれる．それはすでに2章で説明したように RDB/V 1 のハッシュテーブルは複数の (DSCS レベルの) キー順編成テーブルの集りと考えられるからである．

たとえば，テーブルAへのレコード挿入コマンドはデータ辞書を変更する可能性があるので次のようなモード[8] でロックを要求する—データ辞書に IX*．$H(A)$ (ただし Hはデータ辞書アクセスに用いられるハッシュ関数) が示すページをエントリとするキー順編成テーブルに SIX*．さらに変更が必要なデータを格納してページに X*．

このロック要求手順ではテーブルAのデータ辞書を変更するようなコマンドは二つ以上走れないという不満がある．これを解決するには (1) ロック要求の細かさ (granularity) をレコードにまでするか (2) テーブルに関する統計データは動的には収集しないことにし，データ操作コマンドの副作用によってデータ辞書が変更されるということはないようにするかの二つの方法があろう．

---

* より正確にはそのフィールドは定義されてさえもいない．ビュー中で新規に定義されたフィールドとして扱われる．

* IX: インテンショナル エクスクリュシブ，SIX: シェア アンド インテンショナル エクスクリュシブ，X: エクスクリュシブ．

(1)の欠点は，ロックマトリックスのスペースが非常に大きくなる可能性があることである．(2)については RDB/V 1 ではシステム構造上完全には実現できない*．

現在われわれは RDB/V 1 を種々の環境で動かし，現行のロック機構の性能測定を行っている．結果が出るのにはいましばらくの時間がいる．結果（ともし問題があればその解決策）については別の機会に発表したい．

6. リカバリ

RDB/V 1 では，(1)ユーザプログラム（セッション）の失敗，(2)システムダウンおよび(3)ディスク破壊の3種類の事故に対してデータの回復手段を提供する．

(1)のような失敗は，TSS 環境下でシステムが使用されているような場合，作業領域の不足や時間切れ，あるいはデッドロックの原因となることが検知されたというような不測の事態により，当該トランザクションがこれ以上進行できなくなる状態になることが含まれる．このような状況にシステムが素早く，かつ他のユーザにほとんど影響を与えないで対応できることは，RDB/V 1 のようなエンドユーザ指向の対話型システムには強く要求される．このための機構として RDB/V 1 はスロット（物理ページ）の二重化を採用した．この方式は SYSTEM R で私用セグメントのために検討された[1]ものであるが RDB/V 1 では公用セグメントにもその適用を広げている．この方式によるリカバリではページテーブル**内の変更（ポインタの切換え）とボリュームスロットマップの状態変化(不要スロットの解放)が要求されるのみなので非常に高速である．

一方，(2)および(3)のような事態に正しく対処するにはログデータが必要である．ログデータとしては通常，変更されるデータの 'before image' と 'after image' とが取得される．

データの 'before image' を取得する場合，そのログデータは当該データの変更のたびごとに即時にファイルに書き出されねばならない．一方，'after image' をログデータとする場合，ログデータはバッファ中にスタックしておき，トランザクション終了時に一括してファイルに書き出せばよい．

以上のことより，RDB/V 1 では変更データの 'after image' のみをログデータとして取得する方式を採用した*．そのためにページテーブルとボリュームスロットマップを二重化し，おのおののためにバックアップファイルとカレントファイル**とを用意する（図2参照）．

ページテーブルとボリュームスロットマップはセグメントごとに用意されるので，セグメントのチェックポイントが定義される．一つのセグメントにアクセスしていた最後のユーザがそのセグメントをクローズしたとき，そのセグメントのチェックポイントがとられる．外部からのチェックポイントコマンドで強制的にチェックポイントを設定することも可能である．

チェックポイントがとられると，当該セグメントのページテーブルおよびボリュームスロットマップ両者のカレントファイルとバックアップファイルの内容を等しくする．それは，カレントファイルの変更された部分のみをバックアップファイルにコピーすることにより行われる．

セグメント内のオブジェクトの更新が発生すると，そのセグメントに付随するページテーブルおよびボリュームスロットマップに変更が生じる．このとき，その変更はカレントファイルに対してのみなされる．

トランザクションが進行中に取得されるログデータには次のようなものがある．

(1) 論理ページの予約***，解放が行われた場合その論理ページ番号

(2) ページがトランザクション中で初めて書き出された場合（バッファが不足したときやトランザクション終了時）その論理ページ番号とそれに束縛されているカレントスロット番号の対．

(3) 挿入または削除されるレコード，または更新されるフィールドのページ内オフセット番地と長さ．更新の場合，更新値も追加される．

ただし(3)はディスク破壊時のデータ回復に必要なログデータであり，オプションである．

データ回復のための上記機構の特長は以下のように

---

* 到着順テーブルを構成するページ番号はそのテーブルのデータ辞書中に格納されている．これの利点はインデックスがなくてもテーブルサーチは早くできることにある．SYSTEM R の場合，セグメントの全ページをスキャンしなくてはならない．

** ページテーブルの一つのエントリは二つのポインタからなる．一つはカレントスロットを指し，他はバックアップスロット（シャドウスロット）を指す．

* ただしスロットの二重化はページの 'before image' をトランザクション中「1時的に」保存することを意味する．

** System R におけるページテーブルの current version と shadow version とは意味が違う．RDB/V 1 では一つのページテーブル中に current と shadow の両 version が存在する．

*** ハッシュテーブル定義時に連続した複数ページが予約される．

要約される.

(1) ページテーブルおよびボリュームスロットマップの二重化による領域のオーバヘッドは小さい. 実際, 両者のためのファイル容量はセグメントのサイズのおのおの 1/500, 1/16,000 である.

(2) ログデータ量が小さい. たとえば, n ページに分散している n 個のデータを更新する場合 (これは最悪の場合である), データ量は約 $8n+(4+l)\times n$ バイトである. ただし l はデータ長. もしディスク破壊回復オプションが指定されていなければ上式は $8n$ となる.

(3) ログデータはログファイルバッファに蓄積され, しかも量が少ないので, バッファサイズを適当に大きくとれば, ログデータの書き出しはトランザクション終了時に1回行われるだけである.

以上より, ログデータ取得およびそれを使ってのデータ回復作業に関する空間的, 時間的オーバヘッドは小さい.

システムダウン後 RDB/V 1 を正常に起動させるためにはリカバリ用プログラムを走らせる必要がある. このプログラム (quick recovery) は回復すべきセグメントのページテーブルおよびボリュームスロットマップのおのおののバックアップファイルをカレントファイルにコピーし, ログファイル中の当該セグメントに関する論理ページ番号とスロット番号を使って中断されたトランザクションの実行以前の状態のカレントファイルを作り出す. 付録IIにクイックリカバリプログラムの手順を示す. このプログラムは, ページテーブル, ボリュームスロットマップおよびログの各量が比較的小さいため, 文字どおり 'quick' である.

RDB/V 1 でとられたリカバリ方式は

- ページテーブル, ボリュームスロットマップ* の完全な二重化
- 'before image' ログをとらない

点で system R の方式[9]と異なる. しかしスロットの二重化方式は同じなのでそれを原因とする欠点[3),9)] はまぬがれない**.

7. 結 論

RDB/V 1 は当初よりエンドユーザ指向システムとして開発されてきた. したがって対話型システムの実現法としてより柔軟性のあるインタプリタ方式がとられた. これは適用業務サブシステムの統合を容易にした反面, 親言語インタフェースを使った定型業務用プログラムに不用なオーバヘッド* を負わせる原因となる. この欠点を克服するためにコンパイラ方式による親言語インタフェース**の開発がなされた.

しかし対話型 RDB/V 1 は初期のプロトタイプより社内・社外のユーザに試用され, いままでデータベース化が困難であった業務で利用しているエンドユーザから高い評価を受けている.

謝辞 RDB/V 1 の初期のプロトタイプに関心を示し, その応用を積極的に推進しかつシステムの改善, 発展に大きな寄与をした第5システム統轄部の PLANNER 開発グループの方々, とくに神田康敬氏に深謝の意を表したい.

前電子研究部門長代理宮川達夫氏*** にはこのプロジェクト開始当時よりご鞭撻をいただいた. またソフトウェア研究部長林達也氏には種々のご指導を得た. ここにあわせて感謝の意を表する.

参 考 文 献

1) Astrahan, M. M. et al.: System R: Relational Approach to Database Management, *ACM TODS*, Vol. 1, No. 2, pp. 97-137 (Jun. 1976).
2) Blasgen, M. N. and Eswaran, K. P.: Storage and Access in Relational Databases, *IBM Syst. J.*, No. 4, pp. 363-377 (1977).
3) Chamberlin, D. D. et al.: A History and Evaluation of System R, *Comm. ACM*, Vol. 24, No. 10, pp. 632-646 (Oct. 1981).
4) Codd, E. F.: A Relational Model of Data for Large Shared Data Banks, *Comm. ACM*, Vol. 13, No. 6, pp. 377-387 (Jun. 1970).
5) Codd, E. F.: Extending the Database Relational Model to Capture More Meaning, *ACM TODS* Vol. 4, No. 4, pp. 397-434 (Dec. 1979).
6) Codd, E. F.: *Relational Database: A Practical Foundation for Productivity*, Report RJ 3339, IBM Research Laboratory, San Jose, Calif. (Dec. 1981).
7) Eswaran, K. P. et al.: The Notions of Consistency and Predicate Locks in a Database System, *Comm. ACM*, Vol. 19, No. 11, pp. 624-633 (1976).

* P. 52 右段の脚注**を参照.

** トランザクションの進行中は, 変更されるページには2個のスロットが割り当てられたままなので多数のトランザクションが同時に同一セグメントのデータ更新 (UPDATE) を行おうとするときセグメントのスロットが不足する事態が起こりうる. しかしこのような場合はまれであろう.

* 問合せコマンドの解析, 意味チェック, スケジューリングのフェーズ.

** 親言語は社内システム開発用言語 SPL/100 である. このインタフェースは現在のところ社内利用のみに制限されている.

*** 現在は富士通オートメーション取締役開発部長.

8) Gray, J.: Note on Data Base Operating Systems, in *Lecture Notes in Computer Science*, pp. 393-481 Springer-Verlag, New York (1978).
9) Gray, J. et al.: The Recovery Manager of the System R Database Manager, *Comput. Surv.*, Vol. 13, No. 2 (Jun. 1981).
10) Hall, P. A. V.: Optimization of Single Expression in a Relational Database System, *IBM J. Res. Dev.* (May 1976).
11) 石川 博他：リレーショナルデータベースによる日本語辞書の管理とその応用，情報処理学会第24回全国大会予稿集，pp. 1009-1010 (1982. 3).
12) IBM: SQL/Data System Terminal User's Reference.
13) Kim, W.: Relational Database Systems, *ACM Comput. Surv.*, Vol. 11, No. 3, pp. 185-211 (Sep. 1979).
14) 北上 始他：RDB/V 1 に於ける最適化技法，情報処理学会論文誌に掲載予定.
15) 牧之内顕文他：RDB/V 1 に於けるアクセスパスの選択，1979 年情報処理学会全国大会予稿集 (1979).
16) 牧之内顕文他：リレーショナルデータベース管理システム RDB/V 1，情報処理学会データベース管理システム研究会予稿集 (1980. 5).
17) Makinouchi, A. et al.: The Optimization Strategy for Query Evaluation in RDB/V 1 Proc. VLD BInt. Conf. pp. 518-529 (Sep. 1981).
18) 牧之内顕文他：RDB/V 1 による地域情報システムの開発，情報処理学会論文誌に投稿予定.
19) 佐藤秀樹他：RDB/V 1 のホスト言語インタフェースの実現法，情報処理学会第 24 回全国大会予稿集，pp. 491-492 (1982. 3).
20) Selinger, P. G. et al: Access Path Selection in a Relational Database Management System Proc. ACM SIGMOD Conf. pp. 23-34 (Jun. 1979).
21) Smith, J. M. and Cang, P. Y.: Optimizing the Performance of a Relational Algebra Database Interface, *Comm. ACM*, Vol. 18, No. 10 (Oct. 1975).
22) Stonbraker, M. et al.: *The Design and Implementation of INGRES*, Memo. No. ERL-M 577, Univ. of California, Berkeley (Jan. 1976).
23) Wong, E. and Youssefi, K.: Decomposition—A Strategy for Query Processing, *ACM TODS* Vol. 1, No. 3, pp. 223-241 (Sep. 1976).
24) Yao, S. B.: Optimization of Query Evaluation Algorithm, *ACM TODS*, Vol. 4, No. 2, pp. 133-155 (Jun. 1979).

付 録

付 録 I

データベースとして EMP (従業員) テーブルと DEPT (部) テーブルの二つを仮定する.

EMP (ENO, NAME, DNO, SAL, MGR)
DEPT (DNO, DNAME, LOC)

多テーブル創成の例

'EMP テーブルを二つのテーブルに (垂直) 分解したい'

```
GET ENO, NAME, SAL INTO EMPSAL
    ENO, DNO, MGR INTO EMPDNO
FROM EMP;
```

これにより，従業員テーブルが従業員の給与関連のテーブル EMPSAL と所属を示すテーブル EMPDNO に垂直分割される.

多テーブルからの削除

EMP テーブルは削除され，代りに EMPSAL と EMPDNO とが使用される状況を考える.

'従業員番号が 54321 の従業員データを両テーブルより削除せよ'

```
DELETE EMPSAL, EMPDNO
FROM   EMPSAL, EMPDNO
WHERE  EMPSAL. ENO=EMPDNO. ENO
AND    EMPSAL. ENO=54321;
```

このコマンドにより，ENO が 54321 に等しいレコードが両テーブルに存在すれば，それらは削除される．もし片方のテーブルにしか存在しなければ*，レコードは削除されない.

多テーブルの更新

'部番号 10 を 15 に更新する'

```
UPDATE DEPT, EMPDNO
    DEPT. DNO=15, EMPDNO. DNO=15,
FROM   DEPT, EMPDNO
WHERE  DEPT. DNO=EMPDNO. DNO
AND    DEPT. DNO=10;
```

この場合も，DEPT, EMPNO 両テーブル中に DNO が 10 であるレコードが存在する場合にのみレコード削除が実施される.

付録II クイックリカバリの手順

ログデータの種類は五つ—トランザクション名，トランザクションの開始，終了を示すデータ，Put データ，Free データおよび Reserve データ—である. Put (i, j) (i, j は整数) は論理ページ i (の内容) がスロット j に書き込まれたことを示す. Free (i) は，論理ページ i を解放したことを示す (したがってその

* 両テーブルは EMP テーブルを '垂直分解' したものだから，このような状況は望ましくない.

ページに束縛されているスロットも解放される).
Reserve (i, n) は論理ページ i から連続した n ページを予約することを示す.

システムダウン時にはカレントファイルの内容は信用できないのでリカバリはバックアップファイルの内容に基づいて行う.

クイックリカバリの手順

(1) ログファイルの解析

ログファイルを走査し, ダウン時に完全に終了しているトランザクション* のログデータのみを抽出する(他は無視する).

(2) ファイルのコピー

バックアップファイルをカレントファイルにコピーする.

(3) カレントファイルの更新

(1)で抽出されたログデータのすべてに以下の手続きを適用する.

```
if ログデータ=Free(i)
 then
   論理ページ i に束縛されているスロットを解放*
   論理ページ i を解放**
 end
if ログデータ=Put(j, k)
 then
   論理ページ j に束縛されているスロットがあればそれを解放
   スロット k を論理ページ j に束縛***
 end
if ログデータ=Reserve(l, n)
 then
   論理ページ l から連続する m ページを予約
 end
```

(4) カレントファイル (の変更された部分のみ) をバックアップファイルにコピー

(昭和57年4月26日受付)
(昭和57年6月15日採録)

* トランザクションの開始, 終了ログで判定.

* ボリュームスロットマップの当該エントリを 'フリー' 状態にする.
** ページテーブルの当該エントリを 'フリー' 状態にする.
*** ボリュームスロットマップの当該エントリを 'ユーズド' 状態にし, ページテーブルの当該エントリに k を格納.