リレーショナル・データベースの定型業務への適用について

田中　豪　　戸田　博　　池田　哲夫
（日本電信電話公社　横須賀電気通信研究所）

１．まえがき

リレーショナル・データベース（ＲＤＢ）は利用者インタフェースの簡明さ、データベースの定義・構築・維持手段の簡易さより中・大型コンピュータからパーソナル・コンピュータに至る広範囲なシステム環境で利用されつつある。ＲＤＢをサポートするデータベース管理システム（ＲＤＢＭＳ）は

（ｉ）利用者の思考レベルに合った高度なデータ操作を提供する（高水準性）

（ii）データベースのアクセスに伴う物理的情報を隠蔽する（データ独立性）

（iii）システムの変更あるいはチューンアップを容易にする（柔軟性）

などの特徴をもつことから、高スキル利用者のみならずデータ処理に関する専門外の利用者においても容易にＲＤＢＭＳ導入・運用を行なうことができる。

データベースの応用分野は、会話的なアドホック問い合せを中心とする業務処理（非定型業務処理）とアプリケーション・プログラム（ＡＰ）による業務処理（定型業務処理）とに分類される。従来、ＲＤＢＭＳは種々のエンドユーザ・ファシリティ（ＥＵＦ）を充実させることにより、非定型業務処理への適用を主対象として提供されている。しかし最近、上記（ｉ）～（iii）の特徴によりＲＤＢＭＳを定型業務処理に利用する動向が多く見られる。このとき高トラヒックなオンライン・リアルタイム環境にＲＤＢＭＳを適用する場合には、ＲＤＢＭＳの処理効率の問題が顕在化してくる。

本稿ではオンライン・リアルタイム・システム（ＲＴＳ）環境における定型業務処理への適用をねらいとして、主に処理効率を向上させるために具備すべきＲＤＢＭＳの機能および方式について考察する。次章で定型業務処理に適用するＲＤＢＭＳの前提について述べ、３章でいくつかのＲＤＢＭＳ実現上の課題とそれらに対する解決法としての機能、方式の実現方法について述べる。

２．前提

ＲＤＢＭＳを高トラヒックな定型業務処理に適用するためには、ＲＤＢＭＳ自身これら定型業務処理に適した効率のよい処理方式を採用することが第１に必要となる。特に定型業務処理ではＡＰによるＲＤＢアクセスが前提となるため、処理効率上プリコンパイル方式を採用したＲＤＢＭＳが適している。従って本稿では以下のコンピレーション・アプローチを採用したＲＤＢＭＳを考える[1] [2] [3]。

（ｉ）データベース操作言語はＣＯＢＯＬ等の親言語に対する埋め込み型言語（親言語方式）として提供し、ＡＰ中のデータベース操作言語を事前に処理するプリコンパイル機能を提供する。

（ii）プリコンパイル機能ではソース・プログラムの親言語への変換処理に加えて、データベース操作言語ごとに最適なアクセス・パスを含む実行コード（これをＲＤＬオブジェクト・モジュール：ＲＤＬＯＭと呼ぶ）を生成する処理を実現する。

（ⅲ）生成されたＲＤＬＯＭはデータベースの定義変更に対して矛盾のないよう保守するためにＡＰのオブジェクト・コードとは独立に管理する。

データベース操作機能に関してはＲＤＬ[4]で提案されている機能（定義、操作、セキュリティ関連等）の実現を前提とする。なおインデックス定義に関する言語機能はＲＤＬでは規定されていないが、実用レベルの性能を得るためには必要であるため、実現の対象とする。

また、データベース・アクセスに必要なデータベース定義情報の管理法は、

（ｉ）ＲＤＬレベルで提供され、オンライン中に動的に定義もしくは定義変更されうるテーブル、インデックスなどの論理的な情報（これを論理スキーマと呼ぶ）についてはデータベース中に格納し、通常のテーブル・データと同様のアクセス制御を行なう。

（ⅱ）ユーティリティ機能として提供され、オフラインで定義もしくは定義変更されるファイル構成などの物理的な情報（これを物理スキーマと呼ぶ）については独立したファイルとして管理する。

ことを前提とする。

３．ＲＤＢＭＳの実現課題と対処法

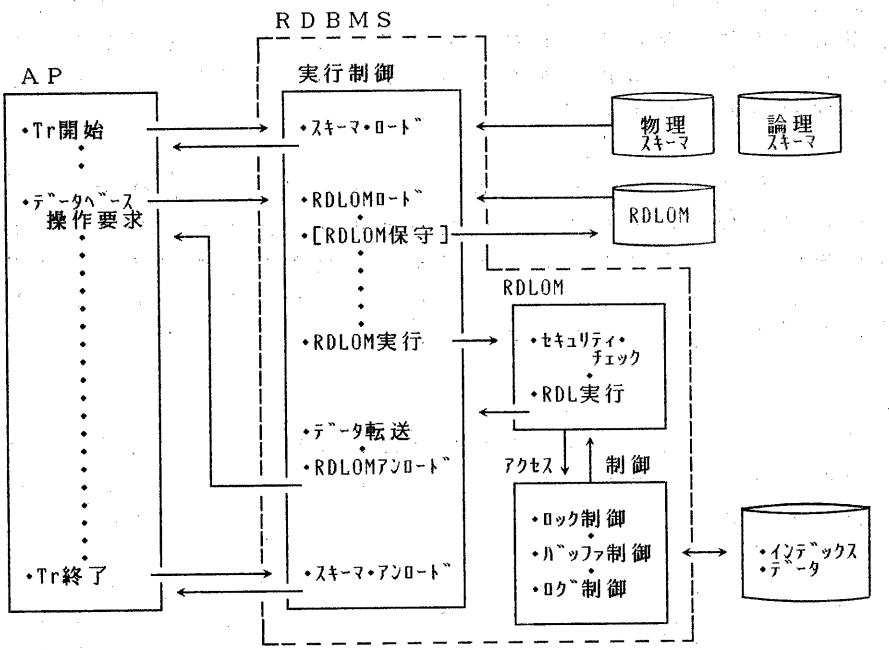

図１　トランザクション処理の流れ

ＲＴＳ環境においてはシステムのスループットを高めるために、個々のトラン

ザクション処理の中で現れるオーバヘッド要因をできるだけ取除くことが必要である。この対処のための観点としてＲＤＢＭＳ内で実行される制御機構のうち、

（ｉ）トランザクション処理の外（例えばＡＰのプリコンパイル時あるいはオンライン開始時など）で実行可能なものは制御の契機を変更する

（ii）（ｉ）の対処がとれないものはチューンアップをはかる

ことを考える。

２章の前提で述べたＲＤＢＭＳによるトランザクション処理モデルは概略、図１のようになる。ここではこの中で観点（ｉ）に該当するものとして

①アクセス・パスの変更に伴うＲＤＬＯＭの保守機構

②セキュリティ・チェック機構

③各トランザクションで共用する資源の管理機構

を、また観点（ii）に該当するものとして

④インデックスに対するバッファ制御

⑤データ転送制御

を取り上げ、それらの対処法について述べる。

３．１　ＲＤＬＯＭの保守機構

テーブルやインデックスを任意の時点で定義あるいは定義変更することのできる機能（ＣＲＥＡＴＥ、ＤＲＯＰ機能など）はシステムの構築、調整あるいは拡張を容易にする機能として重要であり、定型業務においても効果的に利用されると想定される。ＲＤＬＯＭはＡＰのプリコンパイル時にその時点で有効な最適アクセス・パスを選択し、そのアクセス・パターンをコード化した機械語モジュールである。従って利用可能なアクセス・パスの削除（ＤＲＯＰ　ＩＮＤＥＸ）に伴い、関連するＲＤＬＯＭを矛盾のない状態に保守する必要がある。

この保守を自動的に行なうための実現方法には以下がある。

〈方法１〉ＤＲＯＰ　ＩＮＤＥＸ時に、関連するＲＤＬＯＭに不当表示を設定し、そのＲＤＬＯＭの実行時に動的に再生成する[1]。

しかし、この方法は以下の点で不都合がある。

（ｉ）トランザクション処理の背景で再生成処理が発生する可能性があり、このとき処理能力および応答時間に悪影響を及ぼす。

（ii）さらに参照系処理の背景で再生成処理が発生すると、ＲＤＬＯＭの再格納のための更新処理を伴うことになる。これはログ（ジャーナル）の取得契機ともなり、利用者からみて望ましくない事象となる。

上記の問題の解決方法として次の方法がある。

〈方法２〉ＲＤＬＯＭの再生成契機をＤＲＯＰ　ＩＮＤＥＸ時とし、トランザクション処理の背景で実行されるＲＤＬＯＭは常に正当な状態としておく。

この方法により、通常のトランザクション処理に影響を与えることなくアクセス・パス削除時のＲＤＬＯＭの自動保守を実現することができる。なおこの場合、ＤＲＯＰ　ＩＮＤＥＸ処理の速度を劣下させることになるが、定型業務ではこのような定義変更は最繁時を避けて実施されると想定できるため、定義変更時のＣＰＵ負荷は許容できるものと考える。

なおアクセス・パスの追加（ＣＲＥＡＴＥ　ＩＮＤＥＸ）については無矛盾性

の観点からは既存ＲＤＬＯＭの保守は不要である。しかし最適化の効果をトランザクション処理に即時にかつ自動的に反映することが望ましい。このためにはこの契機もＲＤＬＯＭの再生成契機とする対処が必要である。

以上の動的定義変更とＲＤＬＯＭとの関係を表１にまとめる。

表１　動的定義変更とＲＤＬＯＭとの関係

項番	定義変更	既存ＲＤＬＯＭ	
		保守の必要性	保守の契機
1	ＤＲＯＰ ＩＮＤＥＸ	要	定義変更時 →ﾄﾗﾝｻﾞｸｼｮﾝ処理の効率化
2	ＣＲＥＡＴＥ ＩＮＤＥＸ	不要	定義変更時 →ﾄﾗﾝｻﾞｸｼｮﾝ処理への最適化の即時反映

３．２　セキュリティ・チェック機構

権限の委譲、取消し機能（ＧＲＡＮＴ、ＲＥＶＯＫＥ機能）の実現においては、セキュリティ・チェックにはオーバヘッドが伴うため、高トラヒック使用に対応するためにはその効率化が必要である。セキュリティに関する情報はスキーマ内に保存される。データベース操作要求のたびにセキュリティ・チェックを行なうとこれらの照合のためのＩ／Ｏが必要となる。この対処としてＲＤＬＯＭ内にセキュリティ・テェック処理を組み込む方法ではなく、ＲＤＬＯＭの生成（プリコンパイル）時点で対応するチェックを行っておく方法がある。これによりトランザクション処理時のオーバヘッドを最小に抑えることができる。

この実現方法には以下がある。

〈方法１〉プリコンパイルを実行する利用者（プログラマ）がＡＰ中で使用する全ての資源に対するアクセス権を有するときのみプリコンパイル処理を成功させる。この結果生成されたＲＤＬＯＭは、トランザクション処理の中でセキュリティに関するチェックを行なうことなく実行される。

この方法では、任意の端末利用者は特別の権限をもたなくても生成されたＲＤＬＯＭを使用できる。利用者群が特定できる定型業務ではこのようなプログラマ・レベルのチェック機能でも効果があるが、端末利用者レベルでのチェック機能が必要な場合にはさらに次の方法がある[5]。

〈方法２〉論理スキーマ内でＧＲＡＮＴにより委譲されたアクセス権情報に加えてプログラム（ＲＤＬＯＭ）とアクセス権との相関表および各プログラムに対するアクセス統制表を用意し、以下のようにチェックを行なう（図２）。

①ＡＰのプリコンパイル時に、ＡＰで使用する全対象（テーブル等）に対して必要な権限を抽出し（プログラム・アクセス権相関表の作成）、次にそれら権限を全て有する利用者をリスト・アップする（アクセス統制表の作成）。

②ＡＰ（ＲＤＬＯＭ）実行時にはトランザクションを発生させた利用者がアクセス統制表に登録されているか否かチェックし、登録されていればデータベース操作要求毎のチェックを抑止する。

この方法はトランザクション実行中の処理として、②において論理スキーマ参照のためのＩ／Ｏが必要となるが、アクセス統制表のみの参照であるためオーバヘッドの最小化が達成される。またこの方法はＲＤＬＯＭに対する実行権と同様のチェック効果を与えるが、権限の委譲、取消しの対象はテーブル資源等に対するアクセス権が基本となるため、ＡＰを主体としたセキュリティ機構の簡明化がはかれる。

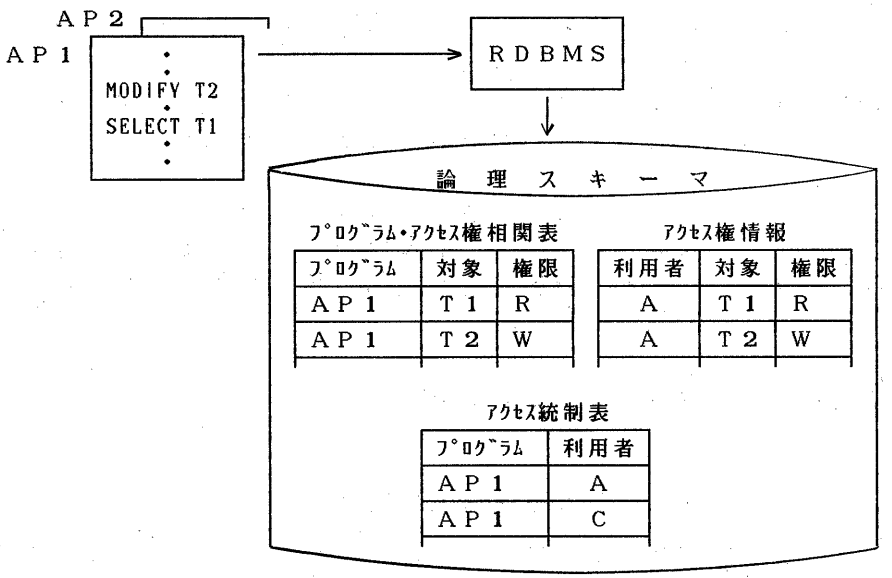

図２　セキュリティ・チェック方式

３．３　共用資源の管理機構

トランザクション処理の中で繰り返し使用され、また共用性の高い資源については不要なＩ／Ｏオーバヘッドをできるだけ回避することが必要である。このような共用資源としては以下がある。

（ｉ）ＲＤＬＯＭ
（ii）物理スキーマ
（iii）論理スキーマ

また、定型業務処理においてはオンライン処理中に頻繁にアクセスされるテーブルに付加された

（iv）インデックス

も共用資源として扱うことができる。以下にそれぞれの資源に対して、主として

メモリ常駐の可能性の観点から考察する。

（１）ＲＤＬＯＭ

ＲＤＬＯＭの総容量は作成されるＡＰの規模に依存する。定型業務においては一般にＡＰ規模が大きいことから、これらはバッファ管理機能の対象となる。しかしトランザクションの中で使用されるＲＤＬＯＭは予め予測することができるため、これらＲＤＬＯＭをトランザクション処理の外、例えばオンライン開始時にプリロードする機能をオプションとして提供する方法が有効である。

（２）物理スキーマ

物理スキーマは小容量であるため、特にバッファ管理の対象とはせず、オンライン開始時にプリロードする機能を提供することのみの対処が可能である。

（３）論理スキーマ

論理スキーマについてはプリコンパイル処理時、および定義または定義変更時のアクセスが主体となる。トランザクション処理中でのアクセスとしてはセキュリティ・チェックのためのアクセスがあるが、３．２節で述べた対処をとることによりこのアクセスを不要化もしくは極小化することができる。従ってこの場合、特別な考慮は不要であり、通常のバッファ管理機能で対処できる。

（４）インデックス

インデックスについてもＲＤＬＯＭや物理スキーマと同様にプリロード機能を提供する方法もある。しかしインデックスは容量的に大きくなる可能性が高いため、メモリ条件の厳しいシステムに対しては次節で述べる専用化したバッファ管理機能で対処する方法が有効である。

以上の共用資源とその管理法を表２にまとめる。

表２　共用資源とその管理法

項番	共用資源	管理法
1	ＲＤＬＯＭ	・バッファ管理　（デフォルト） ・プリロード制御（オプション）
2	物理スキーマ	・プリロード制御（デフォルト）
3	論理スキーマ	・バッファ管理　（デフォルト）
4	インデックス	・専用のバッファ管理（デフォルト）

３．４　インデックスに対するバッファ管理法

ＲＴＳ環境では複数端末からの同時アクセスを可能とするために、業務処理はマルチタスク（タスク・ファミリ）から成るリアルタイム・ジョブとして実行される。あるトランザクションにより起動されるＡＰは、ある１つのタスク上で走

行する。この時バッファ用のメモリ域の持ち方には次の２つの方法がある。

〈方法１〉全タスクで共通的に参照可能な共用空間（システム共用域）上で全データを対象にＬＲＵを行なう。

〈方法２〉各タスク毎の固有空間（利用者固有域）上でタスク内使用データを対象にＬＲＵを行なう。

共用性が高い、即ちジョブ・トータルから見ると参照の局所性が高いインデックスに対しては方法１が効果的である。しかしこのバッファ内に局所性の低いデータ（テーブル・データ）も同時に読み込みＬＲＵを適用すると、局所性の高いデータのバッファ内ヒット率が低下する。この問題は方法１と方法２を併用し、局所性の低いデータは方法２で実現することにより対処することができる。前者のバッファをシステム・バッファ、後者を利用者バッファと呼ぶ（図３参照）。

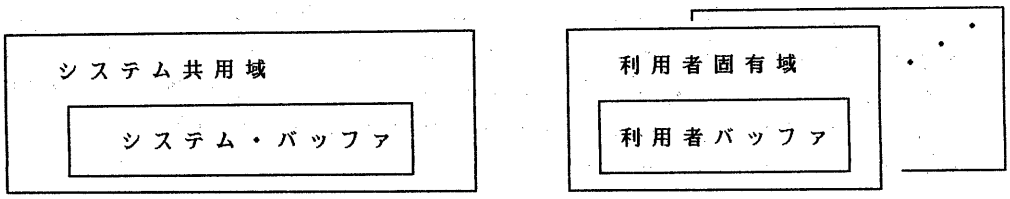

図３　バッファ域の構成例

システム・バッファにおいてはさらに、再参照確率の高いルート・インデックスをなるべく追い出さないようにする常駐化制御（変形ＬＲＵ）を実施する対処も有効である。この方式の概要を以下に示す（図４）。

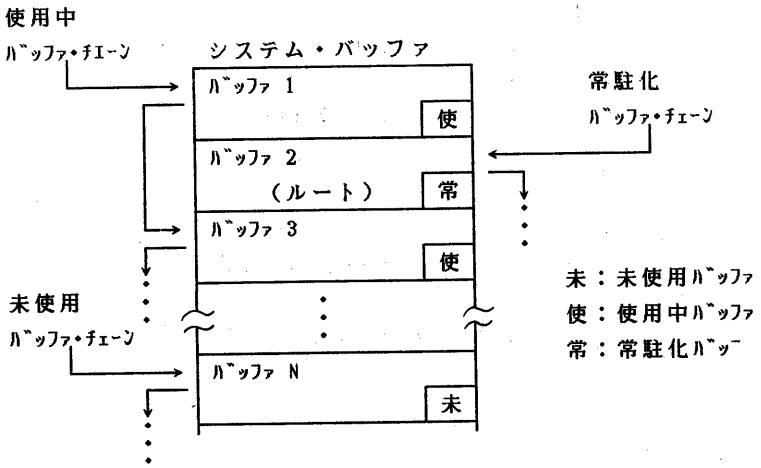

図４　バッファの常駐化制御方式

（ｉ）バッファを使用中、未使用および常駐化という３つの状態で管理する。
（ii）各バッファはチェーン管理を行い、Ｉ／０要求の発生時に読み込むべきバッファの決定は次の優先度に基づいて行なう。
①未使用バッファがあれば、それを使用する。
②未使用バッファがなければ、使用中バッファからＬＲＵで追い出し、使用する。
③使用中バッファもなければ、常駐化バッファからＬＲＵで追い出し、使用する。

３．５　データ転送制御

定型業務処理では以下の理由により、テーブル内のカラム数が多大となることが多い。

（ｉ）入出力帳票には多くの項目が含まれるが、処理効率上、このような帳票類はそのままの形式でデータベース化される。
（ii）意味的（正規化の観点）には別々のテーブルとして管理した方が望ましいが、結合処理に伴う性能劣下を避けるために多くの情報が１つのテーブルとしてデータベース化される。

このようなテーブルに対する操作において、ＡＰとＲＤＢＭＳとの間でカラム単位にデータ転送を行なうと逆に転送オーバヘッドの問題がでてくる。この対処法としてテーブル・タプル全体を一括転送する方法がある。一括転送機能は次のような言語機能の拡張として実現することができる。

テーブル内の全カラムが操作対象となるケース（ＳＥＬＥＣＴおよびＳＴＯＲＥ）において、一括転送を指定するキーワード　”ＲＥＣＯＲＤ”を設ける（図５）。”ＲＥＣＯＲＤ”に続くパラメータ名には、テーブル内の全カラムと並び、属性が対応する親言語の集団項目変数名を指定する。

```
ＳＥＬＥＣＴ    ＡＬＬ
    ＩＮＴＯ    ＲＥＣＯＲＤ　パラメータ名
    ＦＲＯＭ    テーブル名
    ＷＨＥＲＥ  サーチ条件


ＳＴＯＲＥ      テーブル名
    ＶＡＬＵＥＳ  ＲＥＣＯＲＤ　パラメータ名
```

図５　一括転送のための言語構文例

４．おわりに

オンライン・リアルタイム環境での定型業務処理に対してＲＤＢＭＳを適用する際のいくつかの課題の抽出とその対処法について述べた。特にＲＤＢＭＳ内で実行される制御機構のうち、

（ｉ）トランザクション処理の外で実行可能なものはできるだけ外部で実行する。

（ii）（ｉ）で対処できないものはそのチューンアップをはかる。

という観点からいくつかのオーバヘッド要因を抽出し、その効果的な実現法を中心に考察した。

主たる対処案を以下に要約する。

（１）プリコンパイル時に生成されるＲＤＬオブジェクト・モジュール（ＲＤＬＯＭ）に対し、定義変更などによりその保守が必要となったとき、定義変更と同時に保守を行い、トランザクション実行時の性能維持を可能とする方式の提案

（２）プリコンパイル時に、必要なセキュリティ・チェック処理を行ない、トランザクション実行時の処理効率を向上させることのできるセキュリティ・チェック機構の提案

（３）各トランザクションで共用される資源をトランザクションの開始以前に利用可能としておく管理機構の提案

（４）共用性の高いインデックスに対しオンライン・ジョブ（マルチタスク）配下で効率的なＩ／Ｏ管理を可能とするバッファ制御機能の提案

（５）カラム数の多いテーブル・データの転送に対して、効率のよい一括転送を行なう機能とそのための言語拡張方法の提案

これら機能を有するＲＤＢＭＳの提供により、定型業務において処理効率のよいＲＤＢアプリケーションの実現がはかれるものと考える。

〈参考文献〉

[1] D.D.Chamberlin,et al.:Support for Repetitive Transaction and Ad Hoc Queries in System R,ACM TODS,Vol.6,No.1,March 1981

[2] D.D.Chamberlin,et.al.:A History of System R and SQL/Data System, Proc. 7th International Conference on VLDB,Sept.,1981

[3] 稲見：ＭＶＳ環境での関係データベース管理システム－ＤＢ２、ＱＭＦ、ＤＸＴ，情処　第３８回データベース・システム研究会，１９８３

[4] ANSI:Relational Database Language,ISO/TC 97/SC 21/WG5-15, October, 1984

[5] 池田，田中，戸田：データベース管理システムにおける権限チェック方式に関する一考察，情報　第２９回全国大会，１９８４