ネットワークサービス向けメモリ常駐型リレーショナル DBMS の設計と実現

中村 仁之輔[†1] 芳西 崇[†2] 梅本 佳宏[†3]
小林 伸幸[†2] 佐藤 文明[†4] 水野 忠則[†4]

RENA（Real-time Database Management System）は，ネットワークサービス分野への適用を考慮されて設計された，メモリ常駐型のリレーショナルデータベース管理システム（RDBMS）である．この分野では，従来のビジネス処理向け DBMS に比べ，扱うデータ量が少なく，データ操作も簡易である．しかし，DB アクセス処理においては，従来の 10 倍以上の高性能が求められている．また，24 時間無中断でサービスを提供する必要があるため，DB バックアップ処理や DB 保守処理についても，サービス無中断で提供する高可用性が求められている．RENA はこれらの要求にこたえて開発された DBMS である．本論文では，RENA で実現したメモリデータベース方式に基づいた高性能化方式，および高可用性を実現するサービス無中断 DB 保守方式の設計，実現方法を述べる．また，従来 DBMS との性能比較を行い，CPU 処理コストおよびスループットが従来 DBMS に比べて格段に優れていることを示す．

Design and Implementation of a Main Memory Resident Relational Database Management System for Network Service Applications

Jinnosuke Nakamura,[†1] Takashi Honishi,[†2]
Yoshihiro Umemoto,[†3] Nobuyuki Kobayashi,[†2] Fumiaki Sato[†4]
and Tadanori Mizuno[†4]

RENA is a main memory resident relational database management system designed for special application fields such as network service applications. In these special fields, in comparison to business or banking applications, the amount of data isn't so large and data handling is simple. However, the level of performance required for handling data is very high and the availability of non-stop processing against system maintenance, e.g. reconfiguring the database, or system failure is very important. Therefore, in these fields, database management systems should have a high level of performance and availability. RENA is an enhanced relational database management system that satisfies both of requirements.

1. はじめに

データベース管理システム（DBMS）は，統一的な形式でデータの定義・蓄積・操作を可能とし，アプリケーションプログラムの開発，およびシステム運用の効率化をもたらすことにより，システムの中核パッケージとして重要なものとなっている．DBMS の適用分野も，ビジネス分野から，ネットワークサービス分野へと広がりつつあり，これらの分野においても，利用可能な DBMS の実現が求められている．

ネットワークサービス分野の場合，たとえば，高度電話サービス等の実現においては，サービス変更，追加の容易なソフトウェア構成が必要なことから，データ管理に DBMS を適用する必要性が出てきている．この分野においては，扱うデータ量は少なく，データの操作も簡易であるが，超高トラヒックな処理が必要であり，レスポンス，スループットとも，従来のビジネス向け DBMS と比べて，一桁以上の性能向上（高性能）が必要である．また，24 時間無中断でサービスを継続する必要があるため，従来は計画停止等を必要とした DB 保守処理やバックアップ処理等についても，オンライン処理を中断することなく行うことが必

†1 株式会社 NTT データゲートウェイシステム本部
Gateway Systems Sector, NTTDATA CORPORATION
†2 NTT サイバースペース研究所
NTT Cyber Space Laboratories
†3 NTT サイバーソリューション研究所
NTT Cyber Solution Laboratories
†4 静岡大学情報学部
Faculty of Information, Shizuoka University

要である．

一方，近年のメモリコストの減少により，全 DB をメモリ上に常駐させるメモリ常駐データベース（メモリ DB）の実現が可能となってきている．メモリ DB では，従来の磁気ディスク（DK）に DB が格納される DBMS（DK-DBMS）で必要であった I/O が不要となり，メモリ上でのみの処理でアクセスが可能となるため，高性能化を図ることができる．

我々は，ネットワーク系のサービスに適用可能な DBMS として，メモリ DB 技術を主体とし，高性能かつ高可用な特徴を持つ RENA（Real-time Database Management System）を開発した．本論文では，RENA の設計および実現方式を示すとともに，性能評価を行った結果，従来 DK-DBMS に比べて CPU 処理コストおよびスループットが約 10 倍以上優れていることを示す．具体的には，2 章で，メモリ常駐型 DBMS の関連研究を述べ，3 章で，RENA が開発目標としたネットワークサービス分野での DBMS への要求条件を，4 章で RENA のアーキテクチャを，5 章でメモリ DB 記憶構造ならびにサービス無中断 DB 保守方式を，6 章で言語処理方式を，7 章で並行制御方式を，8 章でリカバリ制御方式を述べ，最後に 9 章で，RENA の性能評価結果を示し，RENA の有効性を示す．

2. 関連研究

文献 1) では，メモリ DB に関する研究動向がオーバビューされており，プロトタイプ，商用 DBMS が概観されている．ネットワークサービスへの適用を考えると，ショートトランザクションを主に処理する SYSTEM-M が適している．本システムは，リアルタイム OS である MachOS 向けに作成されたもので，2 フェーズコミットによる並行制御，ファジーチェックポイントによるリカバリ手法，固定サイズのセグメント分割，インデックスのバックアップなしによる新たな研究が行われている．しかし，ネットワークサービスでは，単一レコードアクセスの高速化が必要であり，さらに，サービス無中断での DB 保守が求められるため，ユニーク・キーアクセスの高速化と DB 保守方法の改善が必要となる．最近の商用 DBMS に TimesTen [2] がある．本システムは，本格的な商用 DBMS であり，DK-DBMS との接続，SQL，ODBC，JDBC インタフェース，ハッシュインデックス，T-tree インデックス，2 フェーズコミット，チェックポイント，レプリケーションを実現している．しかし，SQL をフル仕様で実現しており，ユニーク・キーアクセスに向けた処理速度の高速化には限界がある．また，ストアドプロシージャ等の定型処理向けの高速化手法は，DK-DBMS に依存しており，高速化の余地がある．DB 保守については，レプリケーションを利用した手法を実現しているが，サービス無中断での DB 保守は実現できていない．

RENA は，ネットワークサービス向けに，アーキテクチャ，記憶構造，アクセス手法，並行制御，ファジーチェックポイント，SQL 仕様のサブセット化，サービス無中断 DB 保守の各方式を実現した実システムであり，高速性，可用性の向上を可能としている．

3. ネットワークサービス分野での要求条件

ネットワークサービス分野において，フリーダイヤルサービスをはじめとする多様な高度電話サービスを，柔軟，かつ迅速に提供するための機構として AIN（Advanced Intelligent Network）が提案され研究が進められている [3),4)]．AIN では，**図 1** に示されるように，多様なサービスを柔軟に提供するため，回線接続を行う伝達層とサービス制御を行う高機能層を分けた多層構成となっている．たとえば，フリーダイヤルサービスにおいては，利用者が論理電話番号（0120-XXXXXX）による発信を行うと，論理番号が伝達層から高機能層に通知される．高機能層では，物理番号（03-XXXX-YYYY）への変換が行われ，伝達層に通知される．そして，伝達層では，物理番号により接続処理が行われる．この例のように，高機能層では，サービス管理を行うために，種々のデータが管理され，様々な契機で検索/更新が行われている．また，サービス追加や仕様の変更にともない，管理されるデータ構造の変更も発生する．このようなサービス管理においては，利用するデータ（顧客情報等）とサービスを実現するアプリケーションプログラムを分離することにより，種々のデータ管理を独立に実施可能であり，アプリケーションプログラムの追加，修正をデータと独立に実施可能な DBMS の導入が有効となる．

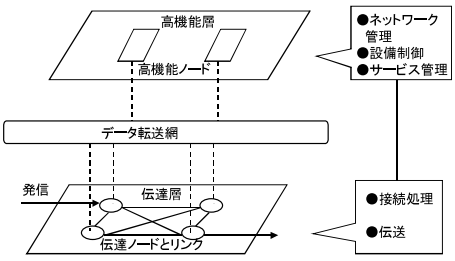

図 1 AIN サービスのアーキテクチャ
Fig. 1 AIN service architecture.

表 1 ネットワークサービス分野のデータベース要求条件
Table 1 Database requirements for network service applications.

項目		ネットワークサービス系 DB	ビジネス処理系 DB
処理能力	応答時間	～数十 m 秒	～数秒
	スループット	～数千 TPS	～数百 TPS
DB 容量		～数 GB	～数百 GB
リカバリ時間（信頼性）		～数秒	～数分
DB 保守時間（連続運転）		サービス無中断（～数百 m 秒）	計画停止（～数時間）

本分野では，従来のビジネス分野と比べ，DBMS への要求条件（**表 1**）および，データベースの利用特性は大きく異なる[5]．これらを，以下にまとめる．

（1）高性能

従来のビジネス処理に使用される場合に比べ，レスポンス，スループットともに，10 倍以上の高速 DB アクセスが要求される．

（2）高可用性

24 時間無中断でサービスを提供する必要があるため，従来は，システムを停止して行っていた DB 保守処理（DB の再編成/再構成），DB バックアップ処理をサービス無中断で可能とする高可用性が要求される．

（3）データ量

対象データ量がビジネス分野に比べて比較的小規模（～数 GB）である．

（4）アクセスパターン

DB アクセスパターンが単純である．大部分の DB アクセスはユニーク・キーによる 1 レコードのみのアクセスである．また，トランザクションは，大部分がショートトランザクションである．

4. RENA のアーキテクチャ

4.1 RENA 設計の考え方

RENA は，ネットワークサービス分野に適用可能な DBMS として設計したもので，3 章で述べた要求条件を満足するために，以下の考え方に基づき設計した．

（1）メモリ常駐型 DBMS

高性能に対する要求条件とデータ量が少ない点から，メモリ常駐型 DBMS による方法を選択した．

（2）高性能化のアプローチ

高性能化を実現するために，ユニーク・キーアクセスに特化したアクセス手法，ショートトランザクション向けの並行制御方式，ログ取得方式，SQL のサブセット化，言語処理におけるコンパイル化を取り入れた．

（3）高可用化へのアプローチ

メモリ常駐型 DBMS では，障害対策として，DB のバックアップを DK 上に取得しておく必要があり，ファジーチェックポイント方式を，また，サービス無中断 DB 保守のために，新旧テーブルの切替えによる DB 保守方式を取り入れた．

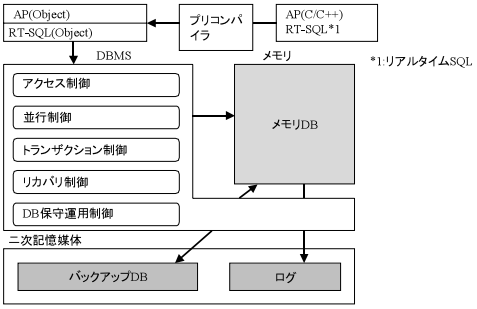

図 2 RENA のアーキテクチャ
Fig. 2 RENA architecture.

4.2 RENA アーキテクチャ

RENA のシステムアーキテクチャを**図 2** に示す．データベースは，主メモリ上のデータベースと DK 等の二次記憶に格納されるバックアップ DB の 2 階層から構成される．主メモリ上のデータ構造は，メモリアクセスに適した構造を採用している．バックアップ DB は主メモリ DB の完全なコピーであり，周期的にコピーが実行される．データベース更新履歴情報であるログは各トランザクション終了時に二次記憶に格納される[6),7)]．

5. アクセス制御方式

ここでは，RENA のアクセス制御方式として，記憶構造，インデックス構造ならびにサービス無中断 DB 保守方式を述べる．

5.1 記 憶 構 造

RENA では，連続メモリ空間上にレコード等を連続配置したメモリセル型記憶構造（**図 3**）を採用した．本構造を用いると，ポインタ操作により高速データアクセスが可能となるとともに，DB 空間の柔軟な拡張，縮小，および可変長データの効率的な格納が可能となる．概要を以下に示す．

（1）固定長レコード用セル

固定長レコード部を格納する空間種別で，固定長レコードが配列構造で格納される．本空間は，テーブル定義時に指定された最大レコード数を格納できるサイズで，連続メモリ空間に確保される．また，固定長レコード内には可変長レコード部を指すポインタを格納する．

（2）可変長レコード用セル

可変長レコード部を格納する空間種別で，様々な長さを持つレコードが保持される．本空間は，テーブル

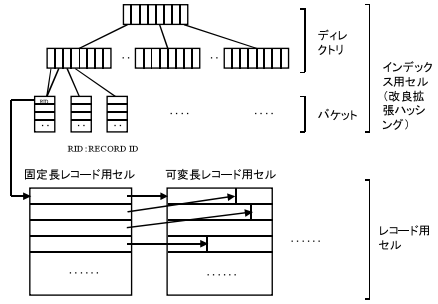

図 3 記憶構造
Fig. 3 Storage structure.

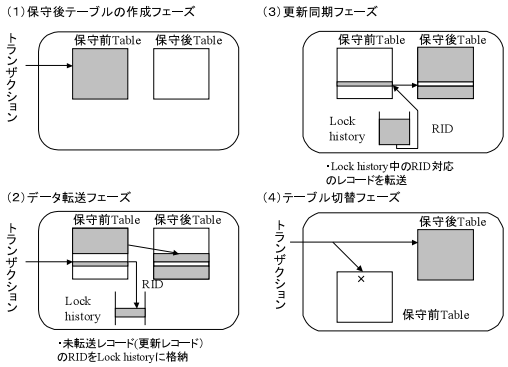

図 4 DB 再構成方式
Fig. 4 DB reconfiguration method.

定義時に指定されたサイズで，連続メモリ空間に確保される．

（3）インデックス用セル

インデックス部を格納する空間種別で，インデックス構成レコードが格納される．本空間はインデックス定義時に指定されたサイズで，連続空間に確保される．

本記憶構造では，定義時に連続メモリ空間を確保するが，空間が不足した場合，5.3 節で示す DB 保守方式により空間サイズを拡大することができる．

5.2 インデックス構造

DB アクセスの高速化手法として，インデックスが一般に有効である．メモリ DB においても，インデックスは高速アクセスのための有効な手段であり，様々なインデックス構造が提案されている[8]．RENA では，ユニーク・キーによる 1 レコードアクセスが主体である点に着目し，ユニーク・キーアクセスに適したインデックス設計を行った．メモリ DB 向けの代表的なインデックス手法に，拡張ハッシング[9]，T-tree[8]がある．しかし，ユニーク・キーによるアクセスを考慮した場合，拡張ハッシング方式が有効である．本方式は，同時実行性が劣る欠点があるが，同時実行性を改善する手法として，平野らにより，改良拡張ハッシングが提案されている[10]．RENA は，処理速度の面から改良拡張ハッシング（図 3）を採用した．ただし，平野らがマルチ CPU 時の同時実行性を向上させるために提案しているバケットのマルチバージョン管理は，トランザクション管理面での負荷を増加させるため，採用していない．

従来 DK-DBMS では，B-tree インデックスへのアクセスならびにレコード格納部へのアクセス時，I/O バッファ内での探索を実施し，I/O バッファに存在しない場合は，DK アクセスを実施する．その後，各ブロック内でのサーチが必要になる．RENA では，I/O に関連する処理が不要であり，また，ハッシュ索引とセル型構造によりサーチ処理コストを削減できており，従来 DK-DBMS に比べて，CPU 処理コストを削減できている．

5.3 サービス無中断 DB 保守方式

リレーショナル DBMS は，スキーマ変更等の DB 再構成機能およびガーベージコレクション等の DB 再編成機能を備えている．ネットワークサービスでは，これらの機能を 24 時間サービス無中断で実現する必要がある．RENA では，この要求にこたえる DB 保守方式を実現した．

RENA の DB 保守方式を**図 4** に示す．本方式は，以下の 4 フェーズから構成される．

（1）保守後テーブル作成フェーズ

処理対象のテーブル（以下，前テーブルと称する）と別な領域に，保守処理後のスキーマ定義情報に基づきテーブル（以下，後テーブルと称する）の領域を確保する．

（2）データ転送フェーズ

前テーブルと後テーブル間で 1 レコードずつデータ転送を行う．前テーブル内で更新ロックされていたレコードおよびデータ転送後に発生した更新結果については，次の更新同期フェーズでデータ転送処理を行う．

（3）更新同期フェーズ

本フェーズでは，データ転送フェーズで転送できなかったレコードについて反映を行い，前テーブルと後テーブル間で同期処理を行う．対象となるレコードは，更新ロックされていて転送されなかったレコードと，データ転送後に更新されたレコードである．転送対象レコードを判別するために，並行制御情報として使用されている更新対象レコードの位置情報（レコード ID）のみロックヒストリとしてメモリ上に保存しておき，この情報を用いて，レコード更新同期を実行する．

（4）テーブル切替えフェーズ

更新同期フェーズにおいて，更新同期の対象レコー

ドは，すべてのトランザクションの更新レコードではなく，また，レコードの転送処理は，トランザクションより短いため，ロックヒストリのエントリは減少しつづけ，最終的にはエントリはなくなる．このロックヒストリがなくなった時点で，テーブル切替え処理を行う．テーブル切替え中は，該当テーブルへのアクセスを防ぐため，DBに対して閉塞処理を行う．閉塞中の処理は，前テーブルから後テーブルへのポインタの変更のみであり，ほとんど処理時間を必要とせず，サービスが中断する時間は，数10 msecオーダであり，実質上，サービス無中断のDB保守が実現できる．

DB再編成処理については，上記と同様の方式で，保守後テーブル作成フェーズにおいて，前テーブルと同じスキーマ情報を持つテーブルを作成し，データ転送時にレコードの詰め直しを行うことにより実現できる[11),12)]．

6. 言語処理方式

RENAにおいては，リアルタイムSQL仕様の制定，言語処理のコンパイル化ならびに最適化により，高速な言語処理を実現している．

6.1 リアルタイムSQL仕様

3章の特性で述べたように，ネットワークサービスにおけるDBアクセスは，大部分が非常に単純である．すなわち，SQLにおける，結合，副問合せ，ソート等の複雑な操作が不要である．そこで，SQLの仕様を制限し，SQLのサブセットで構成されるリアルタイムSQL仕様を制定した．リアルタイムSQLの特徴を以下に示す．

- データ定義言語は，CREATE/ALTER/DROP SCHEMA/TABLE/INDEXに限定する．
- データ操作言語は，INSERT，DELETE，UPDATE，SELECT，COMMIT，ROLLBACKに限定する．
- SELECT，UPDATE，DELETEのデータ操作言語は，ユニーク・キーに対する=条件指定による1レコードアクセスのみが可能である．
- 上記にともない，対象テーブルには，少なくとも1つのユニーク・キーが定義されているとともに，該当ユニーク・キーに対して，ユニーク・インデックスが定義される．
- データ型は，CHARACTER，INTEGER，SMALLINT型に限定する．

6.2 言語処理方式

RENAは，埋め込み型リアルタイムSQL/C，SQL/C++を提供している．プログラマは，ホスト言語C/C++の中に，リアルタイムSQLを記述することにより，アプリケーションプログラムを作成する．埋め込み型リアルタイムSQLの言語処理は以下のように実行される．

（1）プリコンパイル，コンパイル時

- RENAのプリコンパイラにより，ホスト言語に埋め込まれたリアルタイムSQLが展開され，ホスト言語のソースコードに変換される．このとき，リアルタイムSQLの最適なアクセスパスを選択する最適化が同時に行われる．
- 次に展開されたリアルタイムSQLのソースコードがコンパイラにより，リアルタイムSQLのオブジェクトコードに変換される．
- 最後に，これらが，アプリケーションプログラムのオブジェクトコードとリンクされる．

（2）実行時

リアルタイムSQLを含むアプリケーションプログラムの実行時，RENAはアプリケーションプログラムのオブジェクトコードから直接コールされる．

本方式を，コンパイルド SQL方式と呼ぶ．

従来のDK-DBMSにおいても，コンパイル方式を実現している．その方式では，プリコンパイル時に，SQLの解析，最適化を行い，その結果のDBMSへのアクセスコードを生成し，そのコードを二次記憶に格納し，実行時にそのコードを呼び出すことにより，処理の高速化を図っている[13),14)]．この方式で，標準SQLを実装すると，多くの処理単位（数式処理，結合処理，カーソル管理等）を管理しておき，その中から必要な処理単位をダイナミックリンクする必要がある．そのため，実行時の制御用の処理コストがかかる．一方，RENAのコンパイルド SQL方式では，アプリケーションプログラム中に処理単位が実行順序に合わせてリンクされており，実行時の処理時間を短縮できる．

6.3 最 適 化

従来DK-DBMSの最適化[15)]は，DKアクセス回数を最小にするためのものであり，メモリDBMSであるRENAにはそのまま適用することはできない．RENAでは，アクセスの単位は1テーブルのみであり，テーブルには，ユニーク・インデックスのみが定義されているため，条件指定されたカラムのスキーマ情報のみを使用し，利用するユニーク・インデックスを選択する方法をとっている．これ以外の最適化処理として，メモリ上のデータ配置とトランザクションの構成内容の特徴から，以下の方法を実現し，処理の高速化を図っている．

(1) メモリ DB 上のデータは連続配置されていることから，DBMS と AP 間のデータ転送時に，連続配置された複数カラムを一括転送する．この方法により，領域の転送回数を削減できる．
(2) SELECT-UPDATE で構成されるトランザクションにおいては，同一レコードへのアクセスが多いことから，先行 SELECT で検索された結果を更新する UPDATE 処理での同じレコードへの検索処理を削減する．この方法により，UPDATE 処理時，従来は，レコードの探索が必要になるが，その処理を削除することができる．

7. 並行制御方式

一般に DBMS は，複数トランザクションの DB 並行アクセス時のデータ一貫性保証機構として，並行制御機能を有している．2 フェーズロック方式[16]はその代表的手法であり，多くの DBMS で採用されている．しかし，本方式を実現した場合，デッドロックが発生する．そのため，デッドロックを発見し，それを解消する処理が必要になる．

RENA では，DB アクセスが非常に高速であること，およびトランザクション処理が非常に短いことに着目し，スピンロック方式をベースとする並行制御方式を採用した．概要を以下に示す．

(1) ロック単位

ロックの単位は，ショートトランザクションの競合率を下げるため，レコード単位とする．

(2) 制御情報

制御情報には，以下の 2 種類がある．

(a) レコードに，シリアライズ用フラグとロックモード領域を持つ．シリアライズ用フラグは，OS が提供する TEST&SET 命令によりフラグをセットする領域で，トランザクションのシリアライズ制御を実現する．ロックモードは，参照モード / 排他モードを示すもので，正数が参照トランザクション数を，0 が未ロック状態を，−1 が排他ロック状態を示す．

(b) トランザクションごとに自トランザクションがロックしている個所情報を保持する．

(3) ロック制御

各トランザクションにおいて，まず，ロック該当個所に対し，シリアライズフラグのセットを行う．セットできれば，以下のロックモード処理を行う．

(a) ロックモードが排他モードかつ自トランザクションがロック済みであれば，何も処理を行わない．

(b) ロックモードが排他モードかつ自トランザクションがロック未ならば，ロック未取得とする．

(c) ロックモードが参照モードかつ自トランザクションがロック済みであれば，何も処理を行わない．

(d) ロックモードが参照モードかつ自トランザクションがロック未であれば，ロックモードをインクリメントする．

シリアライズ用フラグのセットができない場合とロックが取得できない場合，スピンロックを行い，一定回数以上ウェイトするとエラーで処理を終了する．エラー時以外は，シリアライズ用フラグをリセットする．

ロックの解放は，トランザクション終了時に行い，ロック取得と同様，シリアライズ用フラグのセット後，ロックフラグの減算または 0 セットを実施し，シリアライズ用フラグをリセットする．

(4) ファジーロック処理

ロック処理の削減のために，インデックスサーチ時，バケット同定後，バケット内 RID に基づき複数のレコードを対象に検索条件の確認を行う際，未ロック状態で確認を行った後，該当するレコードのみロック処理を実施し，再度確認を行う．この方法により，チェックを行うたびにロック処理を行う必要がなくなる．

ロック単位を小さくすることによりトランザクション間の競合率を下げることが可能となる．しかし，従来 DK-DBMS で採用されている，待ち制御を行う方式では，ロック単位の細分化を行うと，ロック情報の増大，および待ち制御にともなうロック・キューの増大により，競合検出の CPU 処理コストが非常に大きくなる．一方 RENA で用いた制御法では，待ち制御を行わないためロック・キューが発生せず，競合検出の CPU 処理コストを削減できる．RENA の方式では，競合発生により，ロックエラーとなるトランザクションが増加する可能性があるが，ショートトランザクションがほとんどであるため，その確率は低い．また，ファジーロック処理により，不要なロック処理を排除しており，処理コストを削減できる．

8. オンライン・コンカレント・リカバリ制御方式

メモリ常駐 DB 構造では，電源ダウン等のシステム故障発生時，メモリ上の DB は保存できない．そのため，DB をシステム故障発生前に復旧するための機構が必要となる．RENA では，メモリ空間上の DB と二次記憶上のバックアップ DB の 2 階層記憶アーキテクチャを採用した．メモリ DB は定期的にバックアップ DB として二次記憶上に取得される．あわせて，トラ

ンザクション実行時に取得されるログ（DB 変更履歴）も二次記憶上に取得される．システム故障発生時は，二次記憶上のバックアップ DB とログ情報からシステム故障発生直前の DB に復元する．RENA では，従来 DK-DBMS の更新前情報（BI: Before Information），更新後情報（AI: After Information）の両方を取得する方式と比べ，ログ取得量，I/O 回数，CPU 処理コストを低くおさえることができる方式を実現した．以下にバックアップ方式，ログ取得方式，リカバリ方式を述べる．

8.1 バックアップ方式

バックアップ DB を取得する有効な方式に，ファジーチェックポイント方式[17),18)]がある．本方式は，バックアップ処理とトランザクション処理との排他制御が不要なため，他の処理に比べて，オンライン処理への処理時間の影響が少ないことを特徴としている．しかし，バックアップしたデータの一貫性を保証するためには，バックアップ処理中に走行するトランザクションのログを必要とする．

バックアップは，ファジーチェックポイント方式に従い，バックアップ対象データを二次記憶に出力する．このとき，出力先ファイルは，2 世代化しておき，交互に使用する．また，バックアップした DB データの一貫性を保証するために，バックアップ処理開始以降のトランザクションからのログについては，8.2 節の手順に従いログの取得を行い，バックアップ結果とともに保存する．

本処理は，定期的に実施するが，たとえば，高度電話サービスでは，数分周期で取得が行われる．

8.2 ログ取得方式

ファジーチェックポイント方式において，一貫性が失われる場合には，以下が考えられる．

(a) トランザクションが更新中のデータを，バックアップ処理が取得した場合．

(b) 更新前のデータをバックアップ処理が取得し，その更新トランザクションが正常終了した場合．

(c) 更新後のデータをバックアップ処理が取得し，その更新トランザクションが異常終了した場合．

(d) 更新後のデータをバックアップ処理が取得し，その更新トランザクションが完了以前に障害が発生した場合．

(e) バックアップ処理中に I/O 障害が発生した場合．

これらの問題に対し，一貫性を保証するためのログ取得方式について述べる（**図 5**）．

(1) 通常時のログ取得方式

一般にログは DB 中の BI と AI の 2 種類から構成される．BI は，データ更新前に取得され，ロールバック処理に使用される．AI は，トランザクション終了時に取得され，ロールフォワード処理に使用される．RENA は，バックアップ DB からロールフォワード処理により最新 DB に復元する方式を採用している．通常のオンライン処理時は，トランザクション終了時，AI のみを二次記憶へ取得する．BI については，二次記憶に出力せず，メモリ上でトランザクションをロールバックするのに用いる．この方式により，ログの取得量は 1/2 に削減でき，取得契機は，各トランザクションで 1 回のみでよい．

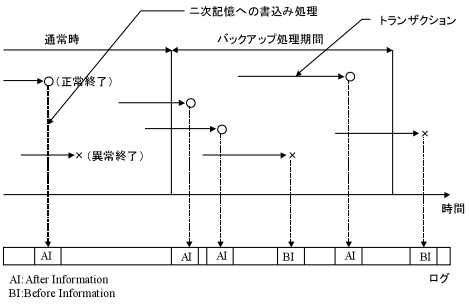

図 5 ログ取得方式
Fig. 5 Logging method.

(2) バックアップ処理期間のログ取得方式

本節の冒頭で示した一貫性が失われる場合のうち，(a), (b), (c) について，一貫性を保証するため，更新前後で BI および AI をログバッファ上に取得しておき，トランザクションが正常終了したときは AI を，異常終了したときは BI を二次記憶に出力する．こうして取得したログにより，一貫性が保証される．また，取得契機は 1 回とすることができる．

(d), (e) については，バックアップ用ファイルを 2 世代化することにより,(d), (e) が発生した場合については，バックアップ処理をアボートし，前世代のバックアップデータを使用することにより一貫性を保証する．

以上のログ取得方式では，バックアップ DB が 2 世代必要になり，二次記憶の容量は 2 倍となるが，トランザクション中におけるログの取得契機は，各トランザクション内で 1 回にすることができ，オンライン処理への影響が少ない DB バックアップ処理が実現できる．

8.3 リカバリ方式

(1) トランザクションのロールバック処理

トランザクションが DB を更新したときに，BI をメモリ上に取得しておき，トランザクションが異常終了したとき，DB を更新前の状態に戻す．

(2) 二次記憶からのリカバリ処理による DB 復元

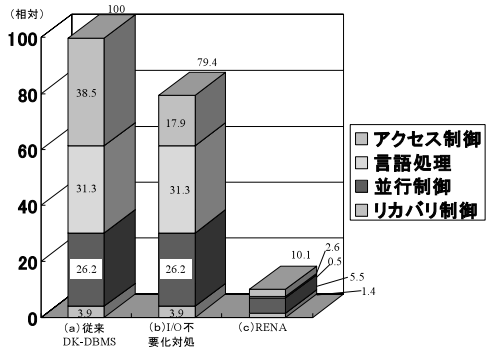

図 6 CPU 評価
Fig. 6 Results of performance tests.

二次記憶上の最新 DB バックアップファイルをメモリ上にロードし，さらに二次記憶上の AI，BI を発生順に反映する．ただし，障害がバックアップ処理中に発生した場合は，1 つ前の世代のバックアップファイルを使用する[11),12)]．

9. 性能評価

RENA で実現した高速性を確認するために，DB アクセス処理の CPU 処理コスト評価およびスループット評価を行った．以下に，評価結果を示す．

9.1 CPU 処理コスト評価

CPU 処理コストの削減効果を確認するため，高度電話サービスの 1 サービス処理を性能評価モデル（IN モデルと呼ぶ）として選び，DK アクセスベースの従来 DK-DBMS と，メモリアクセスベースの RENA の CPU 処理コストを比較した．IN モデル（付録）は，7 つのテーブルからなり，アクセスは，32 個の SQL から構成される．その内訳は，SELECT が 13，UPDATE が 7，COMMIT が 12 である．

従来 DK-DBMS の CPU 処理コストを 100 とした相対評価結果を**図 6** に示す．DBMS の処理は，大きく，アクセス制御，言語処理，並行制御，リカバリ制御に分けられる．図 6 (a) は従来 DK-DBMS の CPU 処理コストを示す．図 6 (b) は，従来 DK-DBMS の処理から，メモリ常駐化による I/O 処理を削減した場合の CPU 処理コストを試算したものである．図 6 (c) は，RENA の CPU 処理コストを示したものである．以下，従来 DK-DBMS と単純に DB をメモリに常駐化した場合と，RENA との比較を示す．

（1）従来 DK-DBMS と I/O 不要化対処との比較

図 6 (b) に示すように，メモリ常駐化のみを実施した場合，CPU 処理コストは，79.4%まで削減可能である．しかし，ネットワークサービスの要求条件を満足するまでには削減できていない．処理部ごとにみると，アクセス制御での削減のみが可能である．アクセス制御では，I/O バッファでのサーチと実 I/O 処理が不要となり，アクセス制御の 47%まで削減できるが，B-tree 構造を意識したインデックス処理とレコードサーチにおけるブロックを意識した処理は必要である．

（2）従来 DK-DBMS と RENA との比較

図 6 (c) に示すように，RENA は従来 DK-DBMS に比べて，1/10 の CPU 処理コストとなっており，高速化が達成できていることが分かる．以下に処理部ごとの CPU 処理コストの削減内容を示す．

（a）アクセス制御

アクセス制御では，従来 DK-DBMS に対し，7%まで削減できている．削減の要因には，I/O バッファでのサーチと実 I/O 処理が不要になる部分が 56%を，RENA で実現したメモリセル型記憶構造と改良拡張ハッシングによる部分が 44%を占める．

（b）言語処理

言語処理では，従来 DK-DBMS に対し，約 2%まで削減できている．削減の要因は，リアルタイム SQL 仕様により，カーソル管理処理が不要になる部分が 20%，コンパイルド SQL 方式により，実行時の処理単位の呼び出し，ダイナミックリンクが不要になる部分と最適化による部分が 80%を占める．

（c）並行制御

並行制御では，従来 DK-DBMS に対し，21%まで削減できている．削減の要因は，待ち制御が不要なスピンロック方式とファジーロックによるものである．

（d）リカバリ制御

リカバリ制御では，従来 DK-DBMS に対し，36%まで削減できている．削減の要因は，ファジーチェックポイント方式の採用によるログ取得量の削減である．

9.2 スループット評価

市販 DBMS に対する RENA の高速性を確認するため，RENA と市販 DBMS を同一計算機の UNIX 環境上で走行させ，スループットを測定した．市販 DBMS は，ビジネス分野で代表的な 3 種類のリレーショナル DBMS を選んだ．評価モデルは，CPU コスト評価で用いた IN モデルのレコード数を 1/1000 にしたミニ IN モデル（付録）と，DBMS の性能ベンチマーク標準の 1 つである TPC-B モデル[19)]を使用した．TPC-B モデルの DB サイズは，計算機のメモリ制限より，スケール 2（テーブルのレコード数 20 万件）とした．メモリ常駐型 DBMS である RENA と，DK-DBMS である市販 DBMS との測定条件を極力合わせるため，市販 DBMS に対しては，以下の性能チューニングを

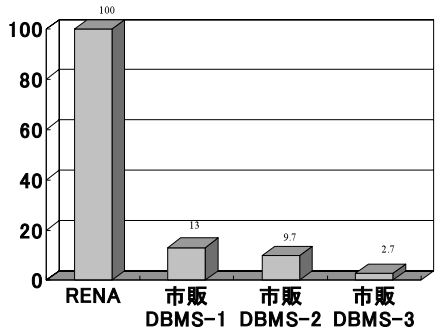

図 7 スループット評価（ミニ IN モデル）
Fig. 7 Results of performance tests (Mini IN model).

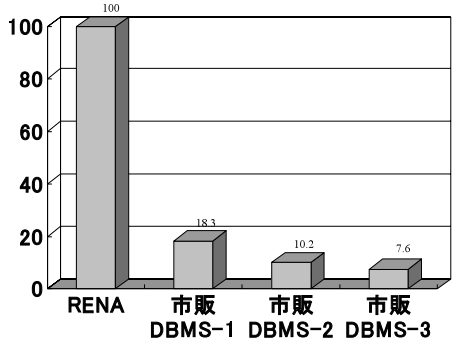

図 8 スループット評価（TPC-B モデル）
Fig. 8 Results of performance tests (TPC-B model).

行った．

- 全 DB のメモリ常駐化
- プライマリキーに対するユニークインデックスの定義
- 高性能が引き出せるストアドプロシージャの利用

また，本評価は，RENA のバックアップ処理は走行させない状態で行った．

試験は，WS: Sun sparc-4/IX (IPX)，メモリ量 64 MB，ディスク容量：520 MB+2 GB × 4，OS: Sun OS 4.1.3 上で実施した．

各種 DBMS を単位時間あたりのトランザクション数（TPS: Transaction Per Second）で比較した結果を**図 7**（ミニ IN モデル）と**図 8**（TPC-B モデル）に示す．これらの図では，RENA の TPS を 100 とした相対性能評価値を示している．図 7 より，RENA は市販 DBMS と比べて，7.7〜37 倍の高スループットを達成していることが分かる．また，図 8 より，RENA は市販 DBMS と比べて，5.5〜13.2 倍の高スループットを達成していることが分かる．図 7 に示したミニ IN モデルによる評価では，参照トランザクションどうしの同時走行が可能なので，図 8 の TPC-B モデルに比べて，スループットの向上効果が高い．

また，各モデルでの RENA の処理時間は，ミニ IN モデルの 12 トランザクションで 10 数 msec，TPC-B モデルで数 msec であり，レスポンスでも十分な高速性を達成している．

バックアップ処理については，高度電話サービスでのバックアップ処理中の CPU 使用率の増加が，数%であり，通常トランザクション処理への影響は少ない．

10. あとがき

ネットワークサービスに必要な高性能かつ高可用な DBMS である RENA の設計と実現方法を示した．RENA は，メモリ常駐型リレーショナル DBMS であり，新たに，メモリセル型記憶構造，改良拡張ハッシングの実装，リアルタイム SQL 仕様，コンパイルド SQL 方式，最適化ならびにサービス無中断 DB 保守方式を提案し，従来提案されているスピンロック方式による並行制御方式，ファジーチェックポイントによるオンライン・コンカレント・リカバリ制御方式をとり入れて実現したものである．

性能評価の結果，CPU 処理コストにおいては，従来 DBMS に比べて，1/10 までコストを削減できており，スループットにおいては，市販 DBMS に比べて，約 5.5〜37 倍のスループットを達成できている．また，リカバリ機能およびサービス無中断 DB 保守機能を実現した．現在，RENA は NTT の高度 IN サービスに導入され，実サービスに供されている．

RENA は，ネットワークサービス向けに特化して開発した DBMS であり，大容量 DB やショートトランザクション以外のトランザクションが多い従来のビジネス分野への適用は困難である．しかし，RENA は，ネットワークサービスと同様のサービス特性を持つ分野での利用は可能であり，高度 IN サービスだけでなく，モバイル環境での位置情報の管理サーバ等への適用の可能性があり，今後，適用先の拡大が期待される．

謝辞 RENA の開発にあたり，ご指導いただいた，NTT ソフトウェア株式会社田中豪部長，西日本電信電話株式会社寺中勝美研究開発センタ所長，ならびに，高度 IN サービスへの導入に尽力いただいた NTT サイバーソリューション研究所板倉一郎主任研究員に深謝いたします．また，数多くの有益な助言をいただいた元 NTT 情報通信研究所，元 NTT ネットワークサービスシステム研究所の関係各位，ならびに，RENA 開発関係者に感謝いたします．

参考文献

1) Garcia-Molina, H. and Salem, K.: Main Mem-

ory Database Systems: An Overview, *IEEE Trans. Knowledge and Data Engineering*, Vol.4, No.6, pp.509–516 (1992).
2) TimesTen Technical Project Managaer, et al.: In-Memory Data Management Technical Paper. http://www.timesten.com/products/wp.html
3) Garrahan, J.J., Russo, P.A., Kitami, K. and Kung, R.: Intelligent network overview, *IEEE Commun. Mag.*, pp.30–36 (1993).
4) Suzuki, S.: IN Rollout in Japan, *IEEE Commun. Mag.*, pp.48–55 (1993).
5) 平野正則, 鈴木孝至, 塩澤恒道, 芳西 崇, 木ノ内康夫：分散処理による高度 IN 用サービス制御ノードの構成, 信学論（B-I）, Vol.J79-B-I, No.8, pp.539–550 (1996).
6) Honishi, T., Kobayashi, N. and Nakamura, J.: Design and Implementation of an Enhanced Relational Database Management System for Telecommunication and Network Applications, *Proc. Pacific Telecommunications Council Eighteenth Annual Conference*, pp.698–703 (1996).
7) 芳西 崇, 小林伸幸, 中村仁之輔, 田中 豪：リアルタイム DBMS 構築技術, *NTT R&D*, Vol.45, No.1, pp.27–32 (1996).
8) Lehman, T.J. and Carey, M.J.: A Study of Index structures for Main Memory Database Management Systems, *Proc. 12th VLDB Conference*, pp.293–303 (1986).
9) Fagin, R., Nievergelt, J., Pippenger, N. and Strong, H.R.: Extendible Hashing – A fast access method for dynamic files, *ACM Trans. Database Syst.*, 4, pp.315–355 (1979).
10) 平野泰宏, 三浦史光, 佐藤哲司：同時実行性を高めた改良拡張ハッシング, 信学論（D-I）, Vol.J78-D-I, No.4, pp.424–433 (1995).
11) 小林伸幸, 芳西 崇：ネットワークサービスへの適用を考慮したデータベース管理システムの実装方式について, 信学技法, 95, No.287 (DE95-50 〜 64), pp.17–24 (1995).
12) Kobayashi, N., Honishi, T. and Umemoto, Y.: A DB Maintenance and Backup Method for Main Memory Resident Database Management Systems for Telecommunication and Network Applications, *The 10th International Conference on Information Networking*, pp.248–253 (1996).
13) Chamberlin, D.D. et al.: A History and Evaluation of System R, *Commun. ACM*, Vol.24, No.10, pp.632–646 (1981).
14) Blasgen, M.W. et al.: System R: An Architectural Overview, *IBM Systems Journal*, Vol.20, No.1, pp.41–62 (1981).
15) Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A. and Price, T.G.: Access Path Selection in a Relational Database Management System, *Proc. '79 SIGMOD Conference*, pp.23–34 (1979).
16) Bernstein, P., Hadzilacos, V. and Goodman, N.: *Concurrency Control and Recovery in Database Systems*, Addison-Wesely (1987).
17) Rosenkrantz, D.J.: Dynamic database dumping, *Proc. ACM SIGMOD Conf. on Management of Data*, pp.3–8 (1978).
18) Hagmann, R.B.: A crash recovery for a memory-resident database systems, *IEEE Trans. Comp.*, Vol.C-35, No.9 (1986).
19) Gray, J. et al.: *The Benchmark Handbook for Database and Transaction Processing Systems*, 2 edition, Morgan Kaufmann (1993).

(平成 13 年 4 月 7 日受付)
(平成 13 年 12 月 28 日採録)

（担当編集委員 増永 良文）

付録 IN モデル

1. DB モデル

テーブル	レコード長（バイト）	レコード数	
		IN モデル	ミニ IN モデル
A	56	40,000,000	40,000
B	53	3,000	3
C	10	600,000	600
D	28	40,000	40
E	1,034	8,000	8
F	9	160,000	160
G	12	8,000	8

2. アクセスモデル
IN モデル，ミニ IN モデル共通

トランザクション	SQL(テーブル名)...
T1	SELECT(A) COMMIT
T2	SELECT(F) COMMIT
T3	SELECT(D) UPDATE(D) COMMIT
T4	SELECT(C) COMMIT
T5	SELECT(C) COMMIT
T6	SELECT(B) UPDATE(B) COMMIT
T7	SELECT(C) COMMIT
T8	SELECT(B) UPDATE(B) COMMIT
T9	SELECT(F) COMMIT
T10	SELECT(G) UPDATE(G) SELECT(E) COMMIT
T11	SELECT(G) UPDATE(G) UPDATE(E) COMMIT
T12	SELECT(D) UPDATE(D) COMMIT

中村仁之輔（正会員）

昭和 52 年愛媛大学工学部電子工学科卒業．同年日本電信電話公社入社．平成 11 年（株）NTT データ技術開発本部マルチメディア技術センタ，現在，ゲートウェイシステム本部．オンライン情報検索システム，リレーショナルデータベース管理システム，データベースマシン，リアルタイムデータベース管理システム，マルチメディアデータベース，放送通信融合システム，コンテンツ管理システムの研究開発に従事．昭和 60 年情報処理学会学術奨励賞．電子情報通信学会会員．

芳西　崇（正会員）

昭和 56 年九州工業大学工学部電子工学科卒業．昭和 58 年東京工業大学大学院工学総合理工学研究科修士課程修了．同年日本電信電話公社入社．以来，主にデータベース管理システムの研究開発に従事．現在，マルチメディア，XML を中心としたデータベースシステムの研究開発に従事．NTT サイバースペース研究所主幹研究員．電子情報通信学会，IEEE 各会員．

梅本　佳宏（正会員）

昭和 61 年大阪大学基礎工学部情報工学科卒業．昭和 63 年同大学院基礎工学研究科博士課程前期修了．同年 NTT 入社．以来，主にデータベース管理システムの研究実用化に従事．現在は，コンテンツ流通システムの研究開発に従事．NTT サイバーソリューション研究所主幹研究員．電子情報通信学会会員．

小林　伸幸（正会員）

昭和 62 年京都大学工学部数理工学科卒業．平成元年同大学院工学研究科修士課程修了．同年 NTT 入社．以来，主にデータベース管理システムの研究実用化に従事．NTT サイバースペース研究所担当課長．

佐藤　文明（正会員）

昭和 37 年生．昭和 61 年東北大学大学院工学研究科電気及通信工学専攻博士前期課程修了．同年三菱電機（株）入社．通信ソフトウェアの研究開発に従事．平成 7 年 1 月より静岡大学工学部助教授．現在，静岡大学情報学部助教授．工学博士．通信ソフトウェア，形式言語，分散処理システムに興味を持つ．電子情報通信学会，IEEE Computer Society，ACM 各会員．

水野　忠則（正会員）

昭和 43 年名古屋工業大学経営工学科卒業．同年三菱電機（株）入社．平成 5 年静岡大学工学部教授，平成 7 年より同大学情報学部教授．工学博士．分散システム，コンピュータネットワーク，モーバイルコンピューティングに関する研究・開発に従事．訳著書としては，「分散システムコンセプトとデザイン」（電気書院，訳），「コンピュータネットワーク」（オーム社），「インターネットとコンピュータネットワーク」（プレンティスホール出版）」等多数．情報処理学会標準化委員会委員，研究会幹事，理事等を歴任．電子情報通信学会，IEEE Computer Society，ACM 各会員．