24時間連続運転に向けたDBエリア有効活用

富士通エフ・アイ・ピー 株式会社

■執筆者Ｐｒｏｆｉｌｅ■

江口　淳

1991年４月 富士通エフ・アイ・ピー㈱に入社
1991年７月 現システム本部第二システム統括部
第二ソフトウエア開発部配属
以来８年間バンキングシステムの
ソフトウエア開発（主にデータベースの
保守ユーティリティ開発に従事）

■論文要旨■

大規模バンキングシステムにおいて，２４時間３６５日オンライン連続運転の必要性が叫ばれる中，活性保守の面からも，データベースの保守の見直しが必要となってきた.

データベースの保守運用として，再配置・再構成という運用を行う.

しかし，再配置・再構成は，オンライン停止が前提の場合があり，２４時間無停止運転を行う場合，再配置・再構成が行えないという問題が発生する.

そこで，２４時間無停止運転のデータベースの保守の１つのアプローチとして，ユーティリティの開発を行った.

ユーティリティは，以下の２つのコンセプトを目標に開発を行った.

①オンライン中に起動可能なバッチ処理とする.

②業務的に不要なレコードを削除するだけで，削除後のデータベースエリアを再利用可能にする.

ユーティリティを開発した効果としては,

①オンライン停止時間を考慮する必要がないため，ユーティリティを起動するスケジューリングが容易となった.

②再配置運用をなくし，再構成を削減できた.

今後の課題として，現状ではスキーマの変更を行うデータベースの再構成は，システム停止日に行っているため，今後データベース再構成も，オンライン中にできるようにしていく.

また，複雑な業務条件に対応するための，削除・更新条件のテーブル作成・修正のGUI化等で，エンドユーザでもテーブルが扱えるようする.

■論文目次■

■図表一覧■

1．はじめに

当社では，大規模バンキングシステムにおいて，長年に渡りデータベースの作成・保守に係わるユーティリティの開発を行ってきた．

現在の大規模バンキングシステムを取り巻く状況は，金融ビックバンによる商品の多様性，新商品・新サービスのニーズのたかまりから，ATM，CD 機の取扱時間延長及びデビットカード，電子マネー，インターネットでの電子取引などさまざまな商品が登場してきた．

これらの商品に対応していくため，大規模バンキングシステムでは，24 時間 365 日オンライン連続運転の必要性が叫ばれている．

24 時間 365 日オンライン連続運転を実現するための，活性保守という考えが必要となってきた．

本論文では，「データベースの活性保守」を実現するためのアプローチの一つとして，当社が開発した，オンライン中に起動可能なデータベース保守ユーティリティについて述べる．

2．ネットワークデータベースの問題点と解決策

2．1ネットワークデータベースの問題点

バンキングシステムでは，元帳と呼ばれるデータベースとして，主に基幹データベースで主流のネットワークデータベースを採用している．

図1 にネットワークデータベースの概要を示す．

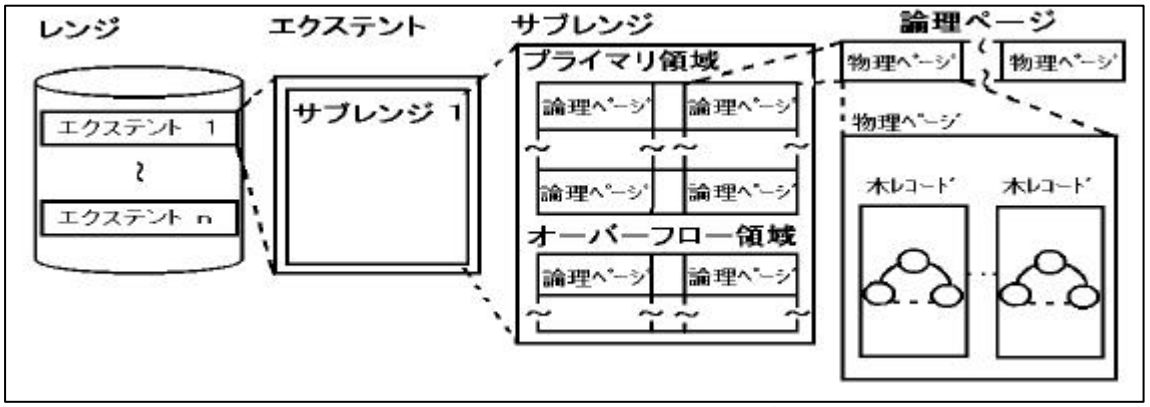

図1　ネットワークデータベース概要

ネットワークデータベースを採用しているシステムでは，長期間の業務（**サービス**）を続けていく中で，問題が生じてくる．

データベースは，長時間の業務（サービス）でレコードの追加を続ける間に以下のような問題が生じる．

・データベースエリア（エクステント）の枯渇

・特定資源（サブレンジ）へのアクセス集中

これら問題点を解決するため，通常では，データベース再配置･ データベース再構成という運用を行う．

２．２再配置

再配置とは，データベース資源の再利用とアクセス効率の向上を目的としているデータベース保守運用である．

(1)資源の再利用

通常は，バッチ処理などで業務的に不要となったレコードの削除を行うが，削除を行っただけでは，レコードの削除が行われたデータベースエリアは再利用できない．

そのため，図2 資源の再利用に示すように，レコードの移動を行ってデータベースエリアの再利用を可能にする．

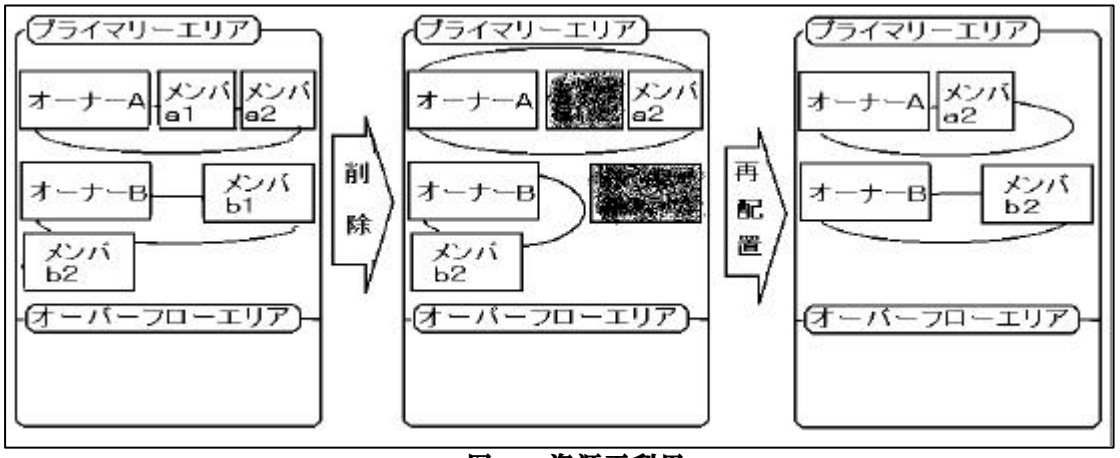

図2　資源再利用

(2)アクセス効率の向上

通常は，プライマリページにレコードを格納するが，エリアが不足してプライマリページに格納できない場合は，オーバーフローページにレコードが格納される．

そのため，レコード検索の際，プライマリページにアクセスし，レコードがなければオーバーフローページをアクセスして該当レコードの検索を行うため，アクセス効率が悪くなる．

このような状況を解消するため，図3 アクセス効率の向上に示すように，プライマリページに空きができれば，オーバーフローページのレコードをプライマリページに移動を行ない，オーバーフローページへのアクセスをなくして，アクセス効率をあげる．

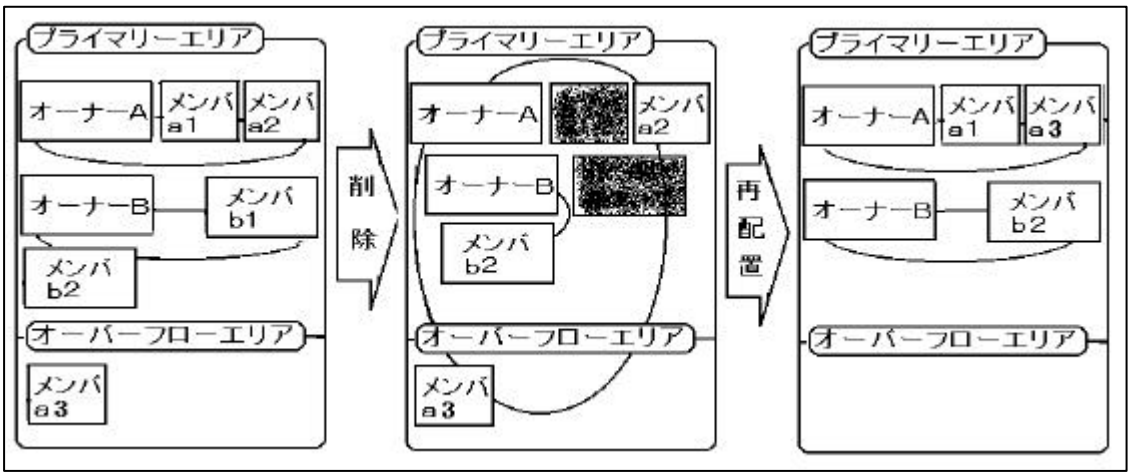

図3　アクセス効率の向上

この二つのケースのように，レコードの移動を行うことを再配置という．

２．３再構成

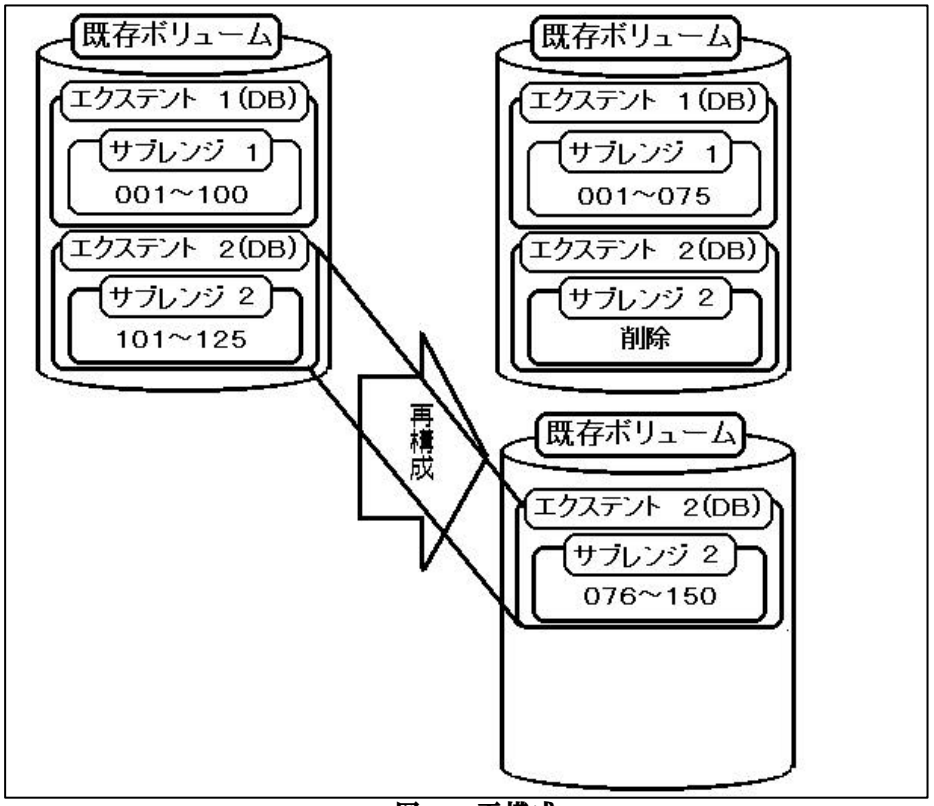

図4　**再構成**

業務運用（サービス）でレコードの追加を続けていくうちに，以下のような場合において，データベースエリアの空きの拡張及び，収容変えが必要となる．

①再配置運用だけでは対処できなくなった場合

②新商品の追加でレコードフォーマットの変更が発生した場合

③データベースエリアの容量に不足が予想される場合

上記事象を解決するために，図4 **再構成に示すように，データベースの論理構造，仮想論理構造及び物理構造すなわち，スキーマに変更が生じるデータベースエリアの拡張及び，収容変えなどを行う．**

これを，再編成という．

これら，再配置･再構成の運用も，24 時間，オンラインシステムを停止しない場合には問題が発生する.

３．２４ 時間連続運転での問題点

再配置については，AIM/OPSFの機能（動的再配置機能及びオンライン中ＤＢ再配置機能）から，オンライン中にできるものもある.

しかし，インデックスエントリのデータベースにおいて，キーの偏りによるオーバーフローページへの書き込みが多発する場合は，インデックスを作り直す必要が生じる.

このような場合に再配置は，DMFというAIMのユーティリティを使用し，データだけを抜き出し，格納構造にあわせて再格納し，インデックスを再作成する運用となる.

また，再構成は，DASDの増設などの物理的及び，収容先DASDの増加，収容先の変更などのため，スキーマの変更を伴う.

24時間連続運転を考慮していく場合，**前述のインデックスの作成及びスキーマの変更は，オンライン中では困難であり，再配置・再構成は，オンライン停止が前提となる.**

すなわち，24時間無停止運転を行う場合，再配置・再構成が行えないという問題が発生する.

そこで，一つの問題の解決策として，オンライン中に業務的に不要なレコードを削除するだけでデータベースエリアを有効活用することができるユーティリティを開発すれば，再配置運用をなくし，再構成運用を減らすことができると考えた.

ユーティリティを開発するうえで，以下のことを重点に開発を行った.

①オンライン中に起動可能にする.

②業務的に不要なレコードを削除するだけで，削除後のデータベースエリアを再利用可能にする.

４．機能概要

４．１オンラインジョブとの競合を極小化

このユーティリティは，サブレンジ単位で起動し，1サブレンジを先頭レコードから最終レコードまでを順アクセスするオンライン中に起動可能なバッチ処理とした.

しかし，実現していくうえで，必ずオンラインジョブとの競合が発生する.

そのため，オンラインジョブとの資源共有で，複数処理からの同時更新による論理矛盾をさけるため，トランザクション排他制御を行うことから，更新データの保証を行うことは必須である.

そこで，トランザクション排他制御による資源の占有範囲を，論理ページ単位と最小の単位とすることで，オンラインジョブとの排他待ちを減少し，オンラインレスポンスへの影響を極小化する.

また，昼間帯のオンライン取扱量の多い時間において，図5インターバル時間に示すように，１論理ページのトランザクション排他終了後から，次の１論理ページのトランザクション排他までの間，インターバル時間を設け，インターバル時間内のオン資源（データベースなど）の占有を行わないことから，オンラインジョブとの頻繁におこりうる資源競合を避けることができる.

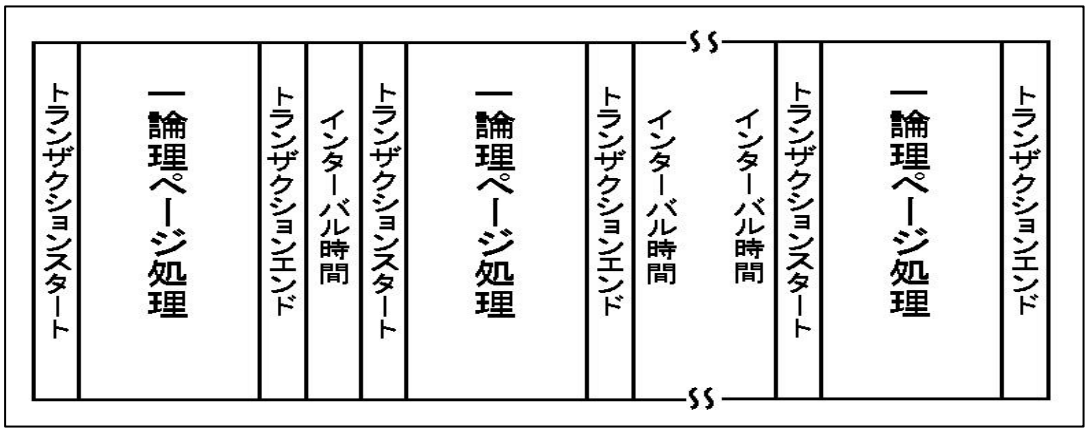

図5　インターバル時間

４．２障害時運用（リスタート型ジョブ）

従来までのバッチジョブの場合，オンラインが停止している夜間帯に起動される．

この場合，異常時には，バックアップからデータベースを起動前の状態まで戻し，再び起動するリラン型が主流である．

しかし，24 時間無停止を実現する上で，オンラインジョブの邪魔をせずに，データベースを元の起動前の状態に戻し，再起動をするような運用形態が取れない．

そのため，トランザクション完結までのデータを保証しつつ，障害復旧後はトランザクション完結後レコードから処理を再開するリスタート型ジョブとした．

以下の動作によってリスタートを可能とした．

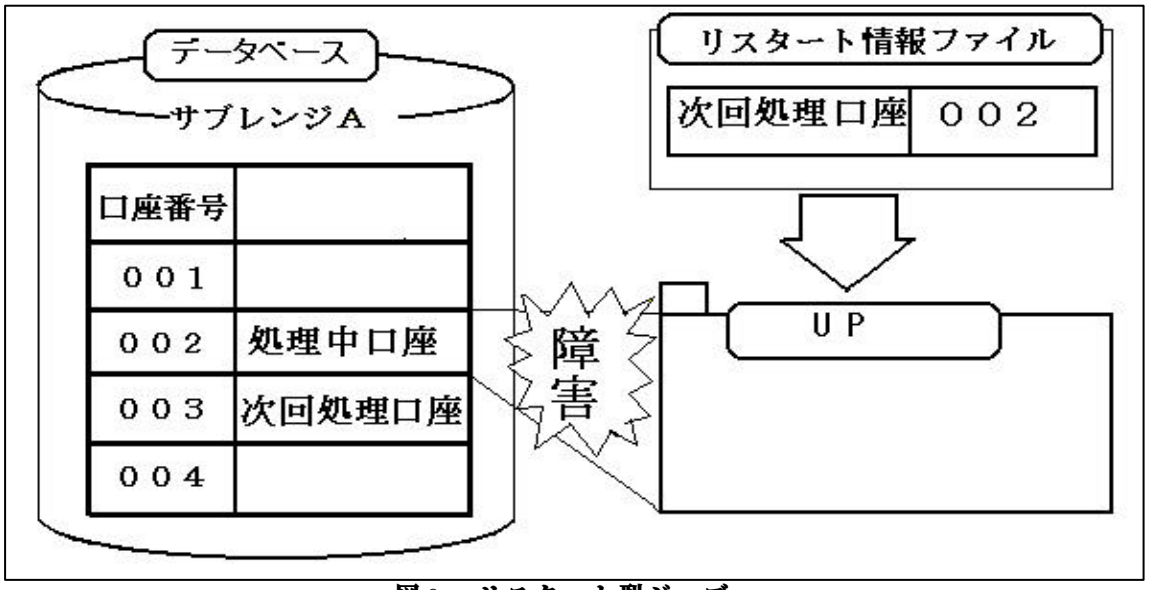

図6　リスタート型ジョブ

①トランザクション終了契機で，リスタート情報用ファイルに，次回エントリレコードを書き込む．

②次回トランザクション開始契機で，リスタート情報用ファイルを読み込み，次回エントリレコードから処理を行う．

③障害発生時には，データベースは，障害発生時前のトランザクション完結時の状態まで AIM のリカバリ機能にから復旧される．

④障害復旧後のリスタート時にリスタート情報用ファイルを読み込み，次回エントリレコードから処理を再開する（図6　リスタート型ジョブを参照）．

４．３座蒲団敷方式採用

業務的に不要なレコードを削除するだけで，削除後のデータベースエリアを再利用可能にするため，データベースの構成に座蒲団敷方式を採用した．

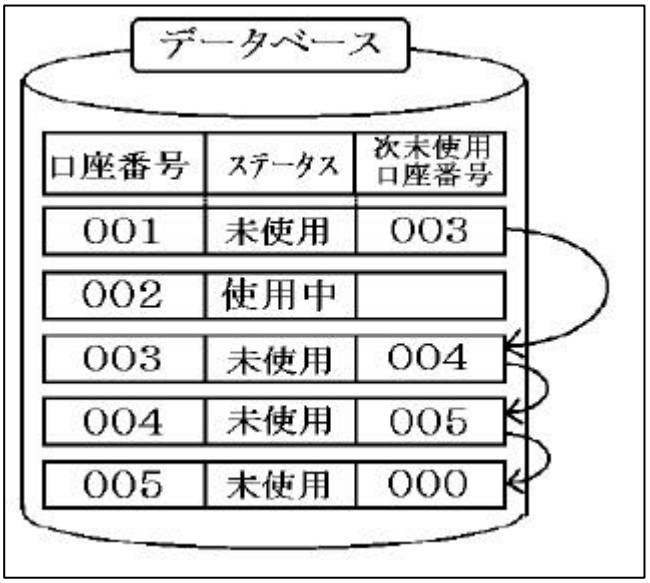

図7　座蒲団敷口座方式

座蒲団敷きとは，図7 座蒲団敷口座方式に示すように，各レコードの保有数を予測しておき，１論理ページ内に格納するエントリレコード（口座）の数を決め，エントリレコード（口座）を座蒲団を敷くようにあらかじめ，１サブレンジ分格納しておく．

このため，オーバーフローページにレコードが格納されず，業務的に不要なレコードが生じた場合，配下のメンバレコードを削除し，エントリレコードを初期フォーマットで更新するのみで，エントリレコードを再利用可能とする．

４．４新番号管理テーブルと新番号テーブル

新番号テーブルと新番号管理テーブルを利用して，新規口座開設を容易とし，オンライン性能を確保し，特定サブレンジへのアクセス集中を防ぐ．

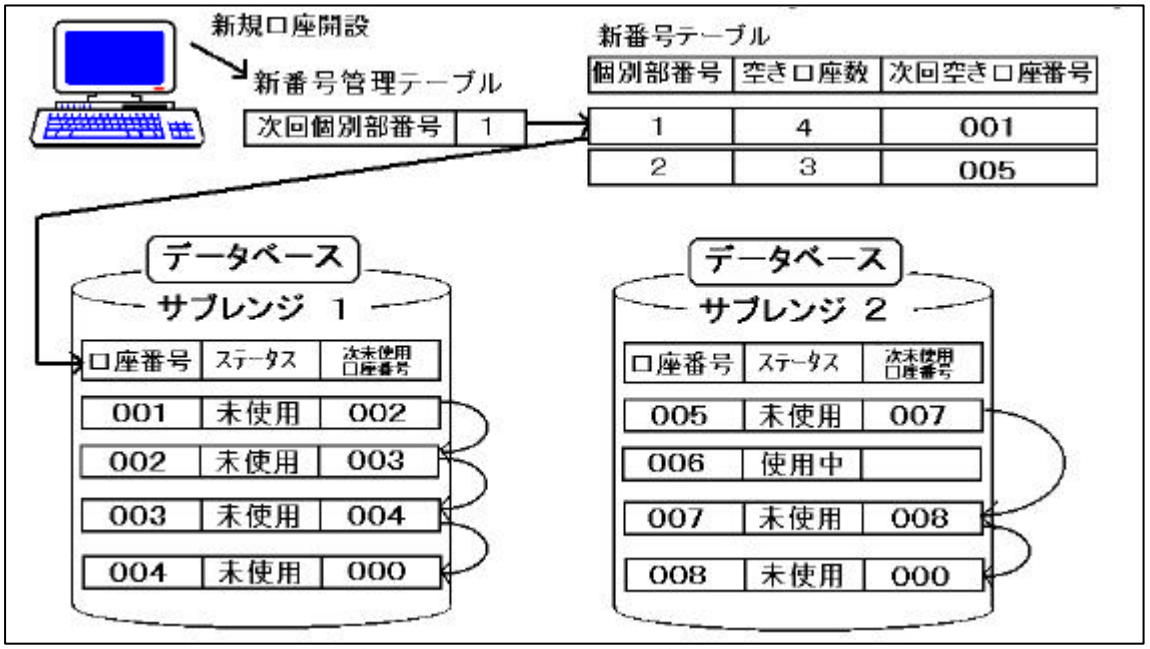

図８　新番号テーブルと新番号管理テーブル

図８ 新番号テーブルと新番号管理テーブルに示すように，新番号管理テーブルは，次回アクセス予定サブレンジ番号を管理しておくことで，一つのサブレンジへのアクセスの集中を防ぐ．

新番号テーブルで，各サブレンジの次回空き口座番号（エントリレコード）と空き口座数を管理しておくことで，口座の新規開設を容易にする．

未使用口座は，次の未使用口座番号を持つことで，点在する再利用可能口座をリンクしており，新規開設時に新番号テーブルの次回空き口座番号に次未使用口座番号の値を設定する．

４．５削除・更新条件のテーブル化

大規模バンキングシステムにおいては，様々な商品ごとの複雑な業務的要件に沿って，不要なレコードを削除する必要がある．

当ユーティリティは，様々な商品のデータベースごとに，別プログラムとはせず，レコード削除・更新条件をプログラムロジックとして固定していない．

図9 削除・更新条件テーブルに示すようなメモリ上に展開されるテーブル上に持つことでプログラムを汎用的にし，プログラムの変更なしで，削除・更新条件の変更が行えるようにした．

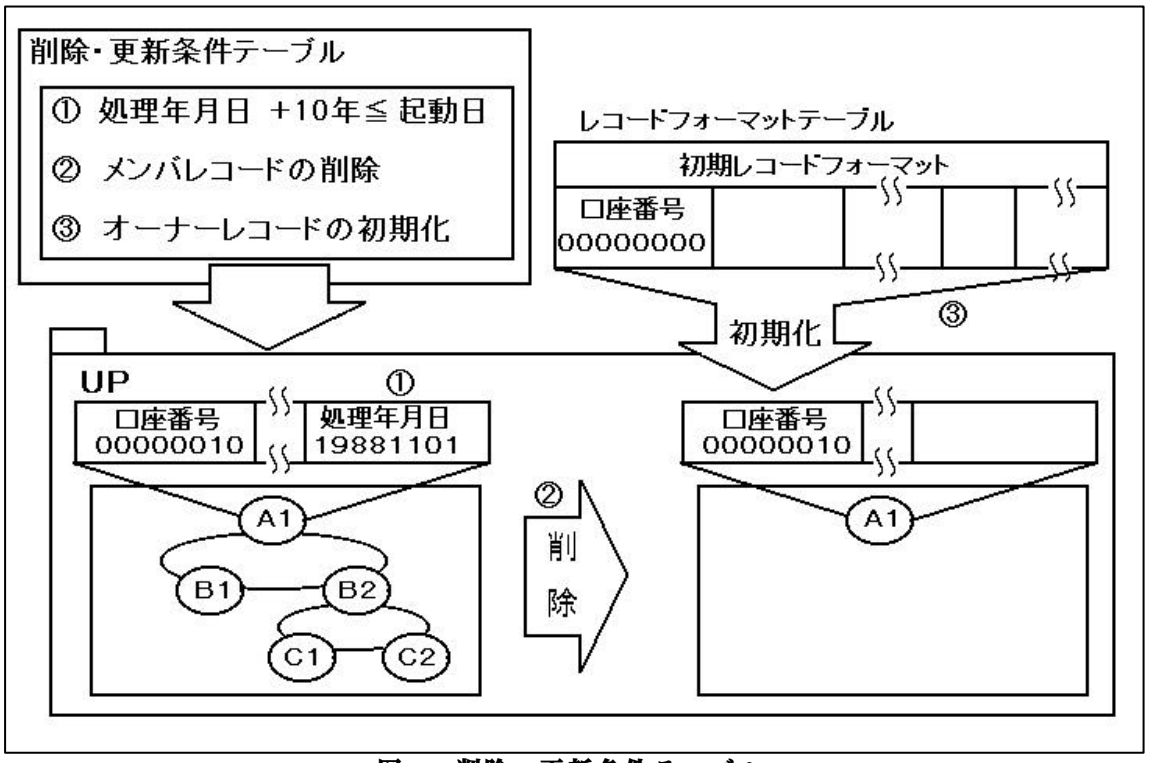

図9　削除・更新条件テーブル

５．効果

当ユーティリティの開発を行うことによって，以下の効果があった．

(1)再配置が必要なくなった

業務的に不要になったレコードを，メンバレコードは削除し，エントリレコードは初期フォーマットで更新することで，新規口座として再利用可能となる．

そのため，再配置運用によるレコード削除後のレコード移動が生じない．

(2)再構成を回避できる

不要レコードを削除し，口座の再利用を行うことによって，口座不足による再構成によるデータベースエリアの拡張を行うという回数が削減できた．

(3)運用の容易性

大量のデータベース（サブレンジ）を処理したい場合でも，オンライン処理中に行える．

そのため，平日の日中帯でもスケジューリングが可能となり，運用が容易に行えるようになった．

６．おわりに

今回，24 時間無停止運転の課題への一つのアプローチとして，データベースエリアの再利用という観点から，ユーティリティの開発を行った．

しかし，再構成の代替ユーティリティではなく，再構成の回数を減らすためのユーティリティのため，現状ではスキーマの変更を行うデータベースの再構成は，システムを停止して行っている．

今後データベース再構成も，オンライン中にできるようにしていく必要があると考える．

また，当ユーティリティは，複雑な業務条件に対応するため，削除・更新条件のテーブル化を行った．

しかし，今後の課題として，現状テーブルの知識のある者だけが修正・作成を行っているため，テーブル作成・修正の GUI 化などで，エンドユーザでもテーブルが扱えるようにしていく必要があると考える．