データベース・システム 48－2
（１９８５・７・１５）

巨大データベースの空間管理方式

三谷政昭　（富士通株式会社）

1. はじめに

近年，企業の扱う情報量は，増大する一方である．また，データベースの利用は益々増加している．このようななかで，現在，データベースのデータを収容する主要な媒体である磁気ディスクの容量も飛躍的に増大してきている．磁気ディスクの容量は，いまや，１装置当りメガバイトからギガバイトの時代となっている．更に，最近では，光ディスク[1]が登場して来ている，光ディスクの場合装置当りの容量はテラバイトのオーダとなる．この結果，企業におけるデータベースの利用方法も大容量化している．すなわち，従来は磁気テープに収めていたデータのデータベース化，時系列データのデータベース上への蓄積等に代表される．

このような大容量データベースでは，以下に示すような，少容量のデータベースにはない新たな問題をもたらしている．

１）　磁気ディスクの容量は飛躍的に増大しているが，磁気ディスクのアクセス速度がこれに伴っていないため，大容量のデータベースは構築できたとしても，データベースの運用，保守等のデータベース全体にわたる処理には，膨大な時間を必要とする（光ディスクの場合はこの問題が更に拡大される）．

２）　大容量のデータベースの処理に伴って，データベースの利用アプリケーションのトランザクション量も増加するため，ＣＰＵ使用率が増大する．

前者の問題の解決方法としては，データベースの区分化がある．後者の問題の解決方法としては，複数システムによる負荷分散がある．

ＡＩＭ／ＤＢは，商用のネットワーク型データベースシステムであり，大容量，高効率を重視したシステムで，従来から，データベースの区分化を可能としている．

また，複数システムによる，データベースの共用[2]も可能となる．

本稿では，ＡＩＭ／ＤＢの従来のデータベースの区分化の手法（レコードの記憶媒体へのマッピング手法）を紹介し，大容量データベース向の区分化手法の拡張について述べる．

2. レコードの記憶媒体へのマッピング手法

ＡＩＭ／ＤＢでは，内部スキーマの定義[3]として，レンジ，サブレンジという概念を用いて，レコードを記憶媒体にマッピングしている．また，レコード実現値の格納領域の管理として，一次領域（プライムと呼ぶ）とあふれ領域（オーバフローと呼ぶ）の手法を用いている．以降，順に説明する．

2.1　レンジ

レンジとは，ＣＯＤＡＳＹＬでいうエリアに該当するが，ＡＩＭ／ＤＢでは，一つのレコードタイプは，一つのレンジにのみ配置可能としている．また，同一レンジへ配置されたレコードタイプは，レコード実現値の格納で，同一の管理をうける．

すなわち，レンジは，レコード格納の仮想的な領域であり，データベースの記憶媒体とは，独立したものである．

レコードタイプは，仮想的な領域であるレンジに配置されるが，記憶媒体への配置は，レンジの全体又はその一部を指定して，記憶媒体へマッピングすることにより行う．

図１は，レコードタイプを記憶媒体にマッピングする方法を，レンジの概念がないとした場合とある場合を対比させたものである．図１のＩは，レンジの概念が仮にないとした場合のマッピング例で，図１のⅡはレンジ概念を導入した場合である．図１では，以下の要件がある場合の配置例である．

－　レコードタイプＡ，Ｂ，及びＣは，レコード実現値の格納で，同一の管理としたい．レコードタイプＤは，Ａ～Ｃとは独立に格納したい．

－　レンジＡはデータ容量の関係又はハードウェア上のアクセス効率から，別ボリュームにも配置したい．

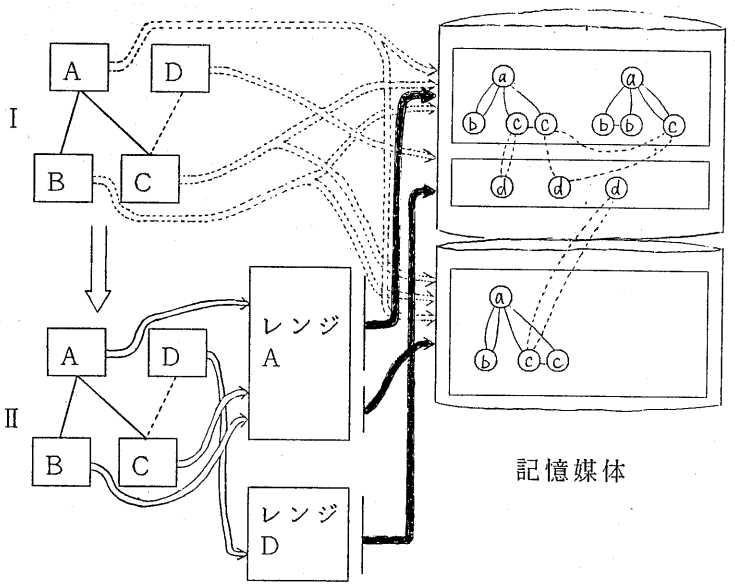

図１　レコードの記憶媒体へのマッピング

2.2　サブレンジ

サブレンジは，データベースを区分化する手法で，レンジに配置されるレコードタイプのうち代表的なレコードタイプのデータをキーとして，レンジを複数の領域に分類したものである．

図2にサブレンジの具体例を示す．

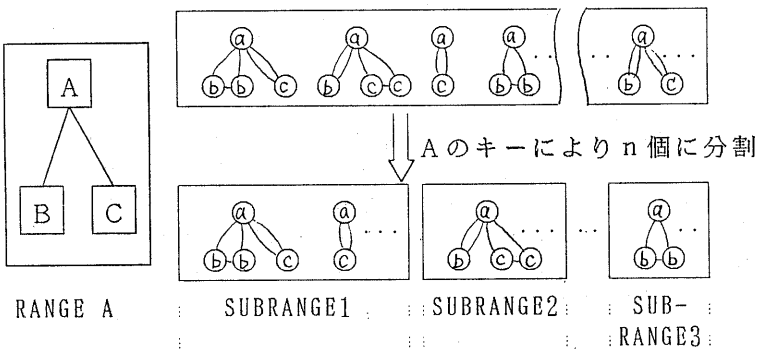

図2　サブレンジによるデータベースの区分化

データベースを区分化することは，データベースのアプリケーションにとっては，必要なことではないが，以下のような場合，データベースの運用，保守にとっては，重要となる．

- データベースが大容量となり，データベースの全体の再編成等が一定時間の間に行うことができないため，いくつかの単位にデータベースを分割して，その単位でデータベースの運用，保守を行いたい．
- データベースのデータが一定単位で独立しているため，インテグリティやセキュリティの観点から独立して管理したい．

2.3　プライムとオーバフロー

前述のレンジにレコード実現値を収容するバケットをＡＩＭ／ＤＢでは論理ページと呼び，前述のレンジはこの論理ページからなる．メンバレコード実現値の追加等により論理ページのレコードがあふれた場合は別の領域のバケットを用意し，そこに格納する．最初のレコード格納域をプライム，あふれのための領域をオーバフローと呼ぶ．

図3にプライム，オーバフローの関係を示す．

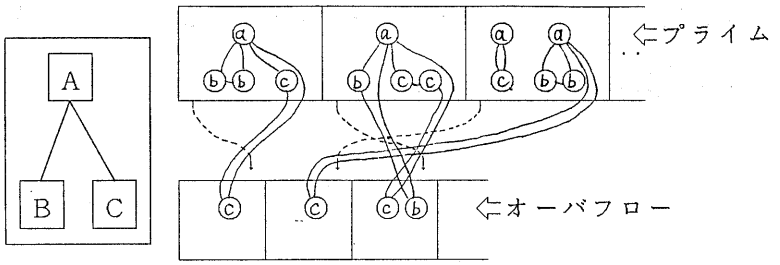

図3　プライムとオーバフローの関係

論理ページはレンジの構成要素であるが，論理ページを記憶媒体に配置（レコードの記憶媒体へのマッピング）するにはデータ管理システムでいうブロックと論理ページを対応させる．すなわち，1又は複数個のブロックと論理ページをレンジ単位に決定する．ブロックに対応したレンジの領域はページと呼ぶ．

図4に，記憶媒体へのマッピング例を用いた，プライム，オーバフロー，論理ページ，ページ及びブロックの関係を示す．

R：レンジ　　P：プライム　，LP：論理ページ
S：サブレンジ　O：オーバフロー　P：ページ　B：ブロック

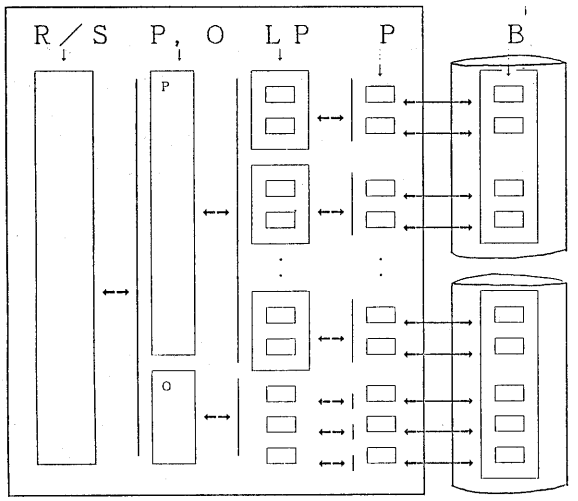

図4　記憶媒体へのマッピング例

3　大容量データベース向きの区分化

データベースの区分化の１手法であるサブレンジについて述べたが，ここでは，大容量向きのデータベースの区分化の拡張方式について述べる．

3.1　ポインタ表現の拡張

ＡＩＭ／ＤＢでは，データベースのポインタを各レンジの先頭を０とする３２ビットの相対バイトアドレス（ＲＢＡ）で表現していた．すなわち，レコードの格納されているページ番号とページ長の積にレコードのページ内変位を加算した値で構成していた．従って，１レンジの最大容量は，約４ギガバイトとなる．また，レンジ容量をＮ倍に拡張する手段としてポインタの１ビットでＮバイトを表すストレッチと呼ぶ係数（図５参照）を用いていた．

この場合，データベースの区分化の１手法であるサブレンジを適用した場合でも，大容量化に関して，以下の問題が発生した。

１）　ストレッチ手法では，データベース中のレコード長をＮバイトのバウンダリに一致させる制約が必要となる．Ｎが大となるにつれ（大容量となるにつれ）バウンダリ損が無視できなくなる．

２）　レンジの先頭を０とする相対バイトアドレスポインタを用いているため，大容量データベースの区分化で，サブレンジが多くなると，サブレンジの追加，削除，統合等がレンジ全体に影響する．

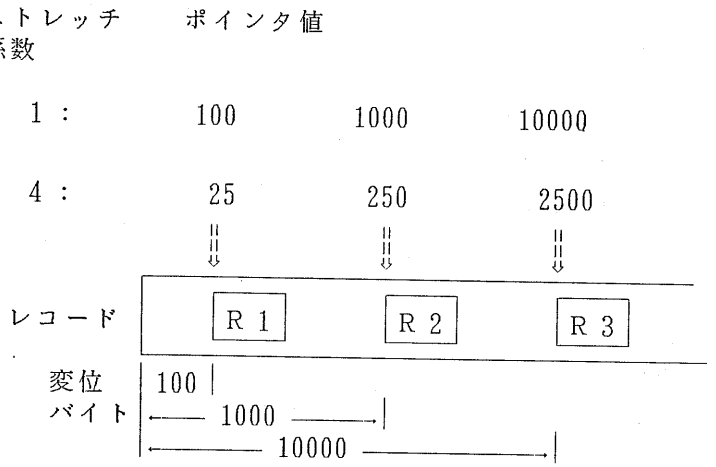

図５　ストレッチ係数によるポインタ表現

前者の解決方法としては，以下の方法が考えられる．

－　ポインタ長を４から６バイト程度に拡張する．

－　ポインタ表現をＲＢＡからＳＬＯＴ方式とすることにより，４バイトでも，表現可能容量を大とする．

従来のポインタが４バイトであること，及びＳＬＯＴ方式であっても４バイトでは表現可能な容量に限りがあることから，大容量化の手段としては，ポインタを概念的には，６バイトとしながらも，４バイトで充分な場合は，４バイトとする可変長と考えることとした．こうすることによって，不必要な１ポインタ当り２バイトの領域の削減と，現状のハードウェア（４バイトレジスタ）に適した処理がたもたれる．

後者の問題は，ポインタを各サブレンジの先頭を０とすることにより，解決可能となる．

3.2　アドレシング方式ＳＬＳ／ＭＬＳ

前述のポインタの拡張方式を採用すると，ポインタは，レンジでなくサブレンジの先頭を０とし，必要に応じ，サブレンジを識別する識別子を用いる方式となる．

ポインタの示す範囲を格納空間（Ｌｏｃａｔｉｖｅ　Ｓｐａｃｅ）と呼ぶと，レンジの先頭を０とする従来のデータベースは，レンジに単一の格納空間を有する．一方，サブレンジの先頭を０とする新しい方式は，レンジに複数の格納空間を有することになるため，オペレーティングシステムの仮想空間に対比して，前者を単一格納空間データベース（Ｓｉｎｇｌｅ　Ｌｏｃａｔｉｖｅ　Ｓｐａｃｅ　ｄａｔａｂａｓｅ），後者を多重格納空間データベース（Ｍｕｌｔｉｐｌｅ　Ｌｏｃａｔｉｖｅ　Ｓｐａｃｅ　ｄａｔａｂａｓｅ）と呼ぶ．

図６にＳＬＳ方式と，ＭＬＳ方式のアドレシングの違いを示す．

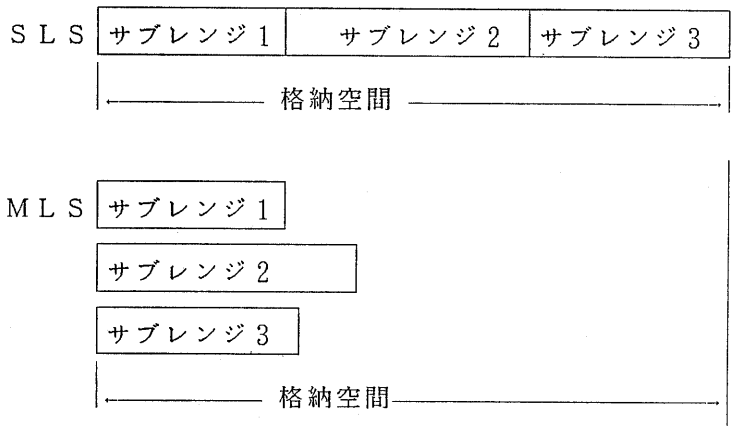

図６　データベースのアドレシング方式

図6から明らかなようにＭＬＳ方式は以下の特徴をもっている。

－　サブレンジの追加，削除，統合等が，他のサブレンジに全く影響を与えない．

－　サブレンジ単位に4バイトのポインタを用いるため，レンジの最大容量はサブレンジ倍増加する．

ＭＬＳ方式で他のサブレンジを示す場合やサブレンジ間の共通領域（例えば，サブレンジ間共通のオーバフロー領域）を示す場合は，ポインタの先頭に相手先を識別する識別子が必要となる．ところが，他のサブレンジや共通のオーバフロー領域を示すポインタの設定は，大容量データベースでは，以下の問題となる．

－　他のサブレンジとのポインタによる結合は，レコード実現値レベルのサブレンジの独立性を失う．

－　共通のオーバフロー領域は，共通のオーバフロー領域の障害に対する信頼性が高くない限りその部分の障害がレンジ全体に影響をおよぼす．

この問題を解決する方法は，サブレンジ間のポインタの代りにサブレンジ間の結合は，レコード中に相手先レコードの索引のキー値をもたせること[4]，サブレンジの共通オーバフロー領域は使用せず，サブレンジ自身のオーバフロー領域を拡張する方式を採用することである．

3.3　ＭＬＳにおける格納空間内の管理

ＭＬＳのアドレシング方式により，サブレンジ間の独立性は，向上したが，サブレンジ内には，プライム領域とオーバフロー領域が存在する．すなわち，サブレンジ（格納空間）にオーバフローが存在する場合，2次元の領域が管理されていることになる．

従来ＡＩＭ／ＤＢ（ＳＬＳデータベース）では，プライムとオーバフロー領域は，図7又は図8に示すようにアドレシングしていた．このため，図7のようにプライム領域の拡張を行うと，オーバフロー領域が影響をうけるか，図8のように，プライム領域に余裕をもたせるために，未使用であるがアドレシングのための予約領域が必要となる．一方，図9に示すようにデータベースの領域をプライムとオーバフローに二分しない方式は，最も柔軟なアドレシング方式であるが，データベースの順処理で，索引順に処理する場合（シーケンシャル処理）に記憶媒体（磁気ディスク）の性能を引きだせない，アドレシングを管理するポインタまたは索引の維持コストが大きい[5]等の欠点がある．

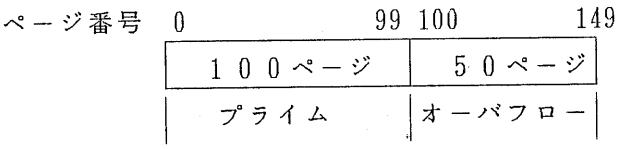

図7　余裕のないアドレシング方式

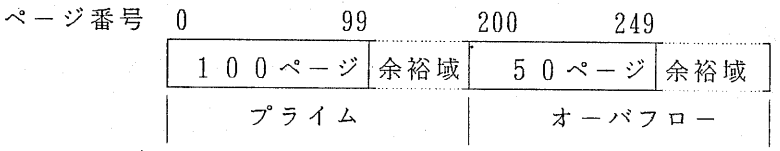

図８　余裕を組み込んだアドレシング方式

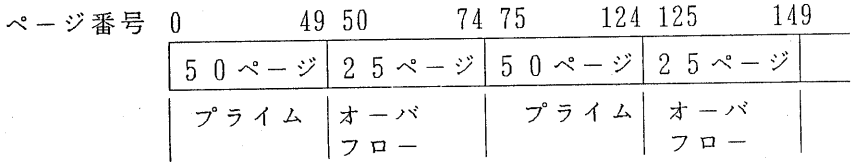

図９　動的なアドレシング方式

ＭＬＳにおける格納空間内のアドレシングは，プライム，オーバフローの相互の影響を除くため，プライムは，ページ番号の小さい方向から，オーバフローはページ番号の大きい方向からアドレシングする方式を採用した．こうすることにより，プライムとオーバフローの干渉をなくすことが可能となった．（図１０）

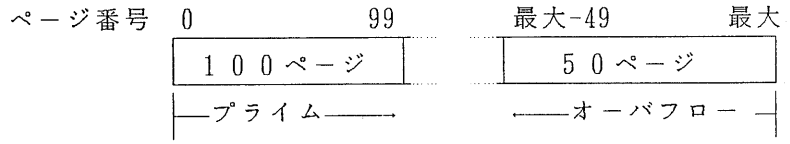

図１０　ＭＬＳにおけるアドレシング

4　大容量データベースの多重処理

データベースの容量が増すにつれ，データベースの運用，保守の役割が極めて重大となる．すなわち，一定時間の間に終了しないデータベースの運用，保守は，システムの再開の停止を意味する．従ってデータベースの運用，保守の多重処理が可能か不可能かは重大な問題となってくる．

4.1　多重処理

ＭＬＳデータベースでは，サブレンジの独立性が高いため，大容量データベースの次の運用が容易となる．

－　複数サブレンジの多重処理による運用，保守が可能

－　サブレンジ単位での再編成が可能

ただし，多重処理は可能でも，１システムでは，一定数以上の多重度は達成されない．

4.2　複数システムによる負荷分散

ＡＩＭ／ＤＢは，疎結合システムによるアクセスが可能となるが，これと同様複数システムによるデータベースの運用，保守も可能となる．従って，データベースの区分化が独立性をもって行われれば，その区分化毎に，複数のシステムで処理可能である．このように，複数システムを用いることにより，データベースは大容量でも，データベースの運用，保守を単時間に行うことができることは大容量データベースにとっては必須の条件と考えられる．

図１１に疎結合システムを用いた，データベースの運用保守の例を示す．

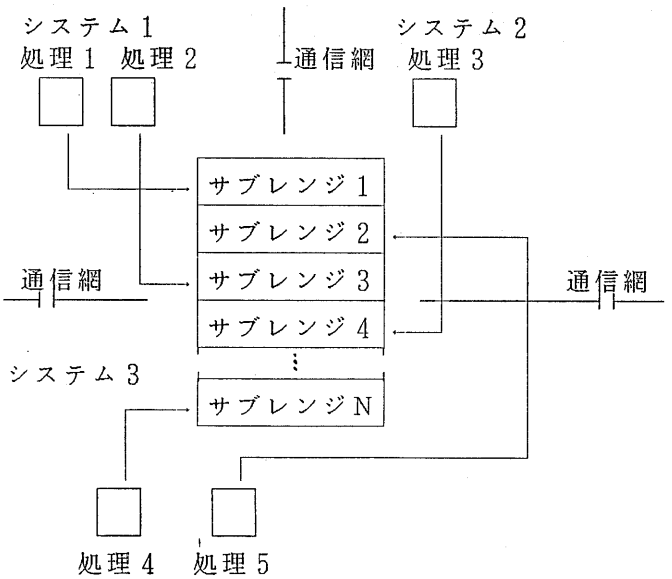

図１１　疎結合システムによるデータベースの運用，保守

5　おわりに

本稿では，大容量データベースに要求されるデータベースの区分化の方式について述べた．区分化の方式で，ＭＬＳ方式を用いると大容量のデータベースの区分化に望ましいこと，ＭＬＳ方式では，独特のアドレシングを用いたこと，及び大容量のデータベースの運用，保守が，複数システムで同時実行可能なことが重要であることについて説明した．今後の課題としては，データベース運用，保守の自動化，システム運用中のサブレンジの追加，削除等があげられる．

本稿では，触れなかったが，データベースの大容量化については，分散処理又は分散データベースで解決する方法が，一般的になりつつあるが[6]，分散処理又は分散データベースであっても，データの重複による分散時は，データベースの大容量化の問題は依然残っている．

［参考文献］

1）　今村：光ディスクメモリ，情報処理，Ｖｏｌ　２６，Ｎｏ．１，ｐｐ．２５－３２（１９８５）．

2）　弘末，他：疎結合計算機によるＤＢ／ＤＣシステムの高信頼性追及の一方式，第２６回全国大会予稿集　４Ｅ－９

3）　穂鷹：データベースの論理設計，ｐｐ．４，情報処理学会（１９８１）．

4）　穂鷹：データベース要論，ｐｐ．１３１－１３２，共立出版（１９８０）

5）　植村：データベースシステムの基礎，ｐｐ．２１４－２１５，オーム社，（１９７９）．

6）　M. Adiba, et al.：Issues in distributed data base management systems：a technical overview, proc. 4th int. conf. on Very Large Data Bases, pp. 89－110（1978）.