解　説

階層構造のデータ・ベース*

渡　辺　純　一**

1. はじめに

組織図は階層構造の表現形式の利用例として，日常なじみ深いものである．コンピュータのデータ記憶構造として階層構造をとり入れた歴史は比較的古く，大多数のデータ処理が純粋に順次処理用の記憶媒体を用い，データの表現形式と媒体上の記憶形式との間に，最小限度の相違しか見られなかった頃に起源をもつとされている（参考文献 1) p. 47).

データ・ベース・マネジメント・システムで階層構造を基礎とするものの数も多く，IBM のプログラム・プロダクトである IMS/VS（情報管理システム——以下 IMS と略す）はその1つである．

IMS はデータ・ベース機能とデータ通信機能を併せ持つが，本稿ではデータ・ベース関係に範囲を限定して IMS の主な機能を述べることにより，階層構造データ・ベースの特徴を概観することとしたい．

2. 論理データ構造

図-1 に見られるように，階層構造は直観に訴える見易さに特徴があるといえよう．ところで，1種類の構造パターンでもって，これを使用するすべての適用業務を満足させるには無理があり，ここにデータ構造の柔軟性が必要となる．すなわち，構造の逆転，不使用データの見かけ上の削除，他のデータ構造との結合等が可能になれば，個々の適用業務の特性に合った形にデータ構造を変更して使用できるというデータ構造の柔軟性が得られることになる．ディスクに記憶したデータ構造を直接変更することなしに，使用者は自分の望む形にデータ構造が変わったと見なして取り扱えるのは，本章で述べる IMS の各種機能の働きによる．つまり IMS は，使用者側が認識するデータ構造（論理データ構造)と，物理的に記憶されたデータ構造(物理データ構造）との間に介在して橋渡しの役を務めるわけである．

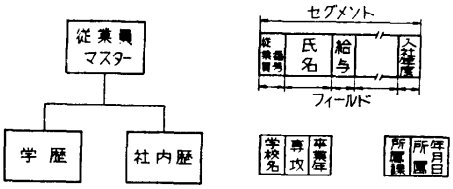

図-1　論理データ構造

2.1 データ構造の論理的な見方

図-1 のような人事データ・ベースの設計に当たり設計者は人事業務に必要なデータ項目を収集し分析して階層構造にまとめ上げる．手順として，まず適用業務の要請を勘案しながらデータ項目を学歴，社内歴関係といったグループ単位にまとめてセグメントを作る．図-1 では3つのセグメント・タイプをもっている．最も使用頻度の高い従業員セグメントを最上位におき，根セグメントとする．他の2種のセグメントは根セグメントの従属セグメントとする．これで，1つの論理データ構造ができ上がる．これは，エンドユーザー，設計者，プログラマーを通して共通に理解し利用するデータ構造の基本的記述となる．IMS へは，この構造を DBD とよばれる制御ブロックに定義して伝える．DBD をもとに，IMS はディスク上にデータと論理データ構造を記憶することになる．

ところで，社内歴や学歴のようにデータの長短に個人差のあるものは，同一タイプのセグメントを複数個持つことにより，多くの場合，セグメントは固定長で設計することができる．これがオカレンスである．任意に抽出した1人の従業員のデータ構造は，論理データ構造にオカレンスを加味したものであり，これを，データ・ベース・レコードとよぶ．**図-2**（次頁参照）は，A～D の4種のセグメントから成るデータ・ベースの2個のデータ・ベース・レコードを図示したものである．

データ・ベースは，1つの論理データ構造を基準に

---

* Hierarchical Data Base by Junichi WATANABE (Systems Science Institute, IBM Japan, Ltd.)

** 日本アイ・ビー・エム(株)教育センター

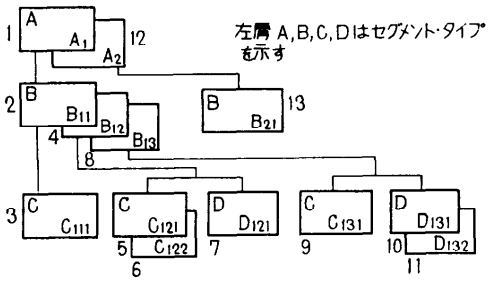

図-2　データ・ベース・レコードと階層順序

して作成され，物理的には固定したものであるが，適用業務ごとに使用するセグメントの選択，構造の倒置，他のデータ・ベースとの結合，更にこれらの組み合わせ等によって，基準のものとは異なる複数個の論理データ構造を定義することができる．つまり，論理的なデータ・ベースの見方を変えることによって，データ・ベースが，あたかも適用業務の要求に合うべく様々な形に変化するかのように柔軟に取り扱えるわけで，データ・ベースの多面的な利用を容易にするものである．いうまでもなく，論理的なデータ・ベースの見方は物理データ・ベースから完全に遊離したものではあり得ず，あく迄もそれを母体として成り立つものである．従って，構造の倒置や他の論理データ構造との結合等に際しては，それに必要な定義の追加記述，特殊用途を持つセグメントの追加挿入など，物理データ・ベース上に事前準備を加える必要がある．

2.2 階層順序

データ・ベース・レコードのセグメントには，階層構造に従った順序づけがなされる．図-2の例では，セグメントの枠外の数字が階層順序を示している．

階層順序は，データ・ベース・レコード内での順次処理の拠り所となるほか，編成法により従属セグメントを乱順にアクセスする際に，IMS が使用する場合がある．階層順序は1つのパスであり，この他に親子セグメント間，兄弟セグメント間のパスがある．図-2では，A_1 と B_{11} が親子セグメント，B_{11}〜B_{13} が兄弟セグメント（オカレンス）の例である．

2.3 セグメントのセンシティビティ

人事部用に作られた論理データ構造に，**図-3**のようなものがあったとすると，給与計算プログラムでは斜線セグメントを使用し，人事異動処理プログラムは従業員マスター・セグメントと斜線のないセグメントを使用するというように，適用業務ごとに使用するセグメント・タイプの組み合わせは異なることが多い．使

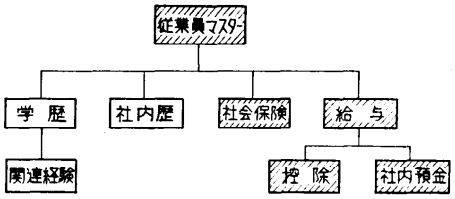

図-3　センシティブ・セグメント

いたいセグメントは，PCB 制御ブロックの中で，センシティブ・セグメントとして定義しておく．IMS は PCB によってプログラムごとの守備範囲（センシティビティ）とし，その論理データ構造が，あたかもセンシティブ・セグメントのみで構成されているかのように取り扱う．この結果，プログラマーは無関係なセグメントを視野から外すことができると同時に，PCB に定義しないセグメントをアクセスすることも出来なくなる．これに加えて，センシティブ・セグメントの各々について，読み取り，追加挿入，内容更新，削除等の入出力機能のいずれを使用するかを定義する．

PCB は DBD と共に，あらかじめ特定の IMS ライブラリーに登録するので，このライブラリー管理を適正に行うことにより，プログラムごとのデータ・ベース処理能力を制御し，機密保護を図ることができる．

2.4 論理関係機能

2.4.1 データの重複排除

給与データ・ベースがあり，これに加えて，スキルズ・インベントリーのために技能データ・ベースを作成するといった場合，技能データ・ベースに従業員氏名をもつと，給与データ・ベースにも従業員氏名が洩れなく記憶してあるので，データの重複保有になってしまう．論理関係機能を使うと，このような重複を避けることができる．**図-4**(次頁参照)ではこれを示している．

技能データ・ベースで，技能氏名セグメントは氏名をもつ代りに，その氏名をもっている給与氏名セグメントのアドレスをもつようにする．この両セグメントの関係を論理関係とよび，給与氏名セグメントは論理親，技能氏名セグメントは論理子となる．論理子セグメントをアクセスすると，IMS は自動的に論理親もアクセスするので，別のデータ・ベースに氏名があることをほとんど意識せずに利用できる．論理関係で設けたこの新しいアクセスのパスを論理パスといい，図-4(A),(B)のように論理親／子ポインターをそれぞれ持たせると，両方向の論理パスが開かれる．同一人が

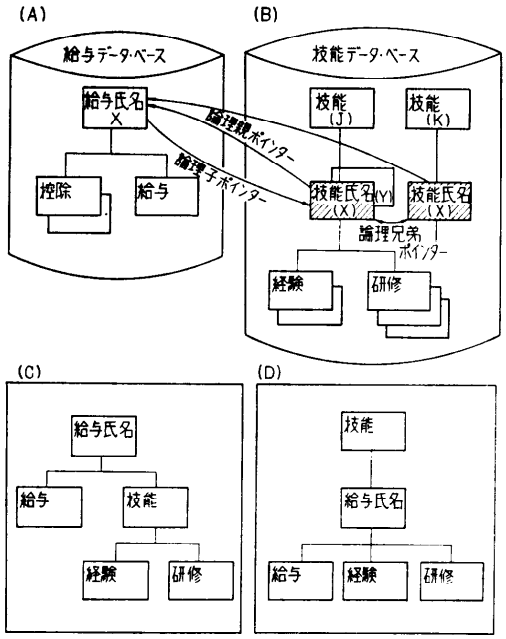

図-4 論理関係とそれによる論理データ構造の創造

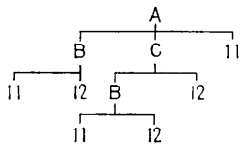

図-5 部品構成表

複数の技能を持つ場合には，その技能セグメント同士は論理兄弟ポインターで結ばれる．

論理関係づけは，1つ以上のデータ・ベースを対象にできるが，部品構成表は1つのデータ・ベース内で論理関係をもつ例である．**図-5** は部品構成表の例で，完成品(A)，組立部品(B,C)，単体部品 (11,12) の間の組立手順における親子関係を表わしたものである．Bは11と12で作られ，CはBと12で，AはB,C,11で作られるという関連がこれから分かる．B,11,12のように製品の中で同じ部品が重複使用されていれば，部品構成表をそのままの形でデータ・ベースに入れると，データの重複が生じる．根セグメントに，部品データを1通りだけ記憶しておき，子部品をもつ部品セグメントは，子部品の種類だけの子セグメントをもつようにする．子セグメントには本来もつべき部品データの代りに，そのデータをもった部品セグメントを指すポインターをもたせる．例えば，部品Aの子セグメントは，部品セグメント B,C,11 のポインターをもつようにする．この場合の子セグメントは，部品セグメントAの物理子であると同時に，それぞれが部品セグメント B,C,11 の論理子セグメントになっている．従業員氏名の場合と同様に，論理子セグメントをアクセスすると，論理親セグメントのデータも併せてアクセスされるので，部品Aの子部品は，部品 B,C,11 であることが分かる．部品Bから展開するのも全く同じ要領でなされる．論理子セグメントのデータとして，Aに直接組付けられるBは m 個，Cの組立てに使うBは n 個というように，論理親との間で固有の意味をもつデータは論理子セグメントが持つ．論理子セグメントは一般的に論理親セグメントに比較して長さが非常に短かいので，データ重複排除の効果をあげることができることになる（**図-6** 参照）．

2.4.2 論理データ構造の結合

論理パスを通じて他のデータ・ベースへアクセスの通路が開けることは，データ構造からすると新しい論理データ構造が創造されることである．

図-4 の例では，(A),(B)の論理データ構造をもとに，論理パスをとり入れることにより，(C),(D)のような論理データ構造を作ることができる．(C)は，給与データ・ベース側から技能データ・ベース側を見て得られる論理データ構造であり，(D)はその逆からみて得られるものの例である．

構造図では，論理親／子セグメントは連結されて，1つのセグメントとして表わされる．**図-7**(次頁参照)では，4つのデータ・ベース間の論理パスを使って，セグメントAからアクセスする場合の論理データ構造の例を示している．論理パスを論理データ構造にとり入れたか否かで，論理データ・ベースと物理データ・ベースと呼び名を区別しており，これを定義する DBD

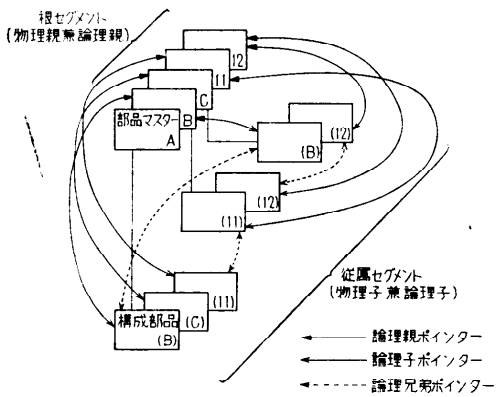

図-6 部品表の論理関係

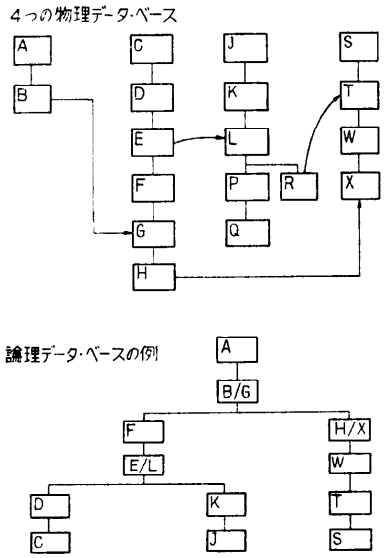

図-7　物理データ・ベースと論理データ・ベース

も，論理 DBD と物理 DBD がそれぞれ対応する．図-7 で論理データ・ベースに一部のセグメントの階層順序の倒置が認められる (D-C ほか)．新しいデータ・ベースが追加作成される場合を含めて，複数のデータ・ベース間に関連づけを行うこの機能は，データの多面的利用に一役買っているということができる．

2.5 副次索引

2.5.1 データ・ベースの多重入口

IMS のデータ・ベース編成には，データ・ベース・レコードが根セグメントのキー項目順に配列され，基本索引がつくものと，乱順で索引をもたないものとある．いずれの場合でも，指定して索引を付加することができ，これを副次索引とよぶ．副次索引は根／従属セグメントを問わず索引の対象にでき，また，1つのデータ・ベースに複数の副次索引をつけることが可能である．副次索引を用いてアクセスするとき，索引の対象となるセグメントからアクセスが開始されるので複数の副次索引を設けることは，データ・ベースに複数の入口，多様な見方を用意することになる．

2.5.2 副次索引の形態

副次索引自体は，索引ポインター・セグメントで構成され，その主な内容は，索引によるアクセスの起点となるセグメント（索引目的セグメント）を指すポインターと，索引項目のデータである．索引データを提供したセグメントは索引原始セグメントとよばれるが，索引目的／原始セグメントは，同一セグメントであっても別セグメントでも構わない．図-8 では，この通りの形態を示している．ここでは，自家用車のカラーが索引項目になっている．

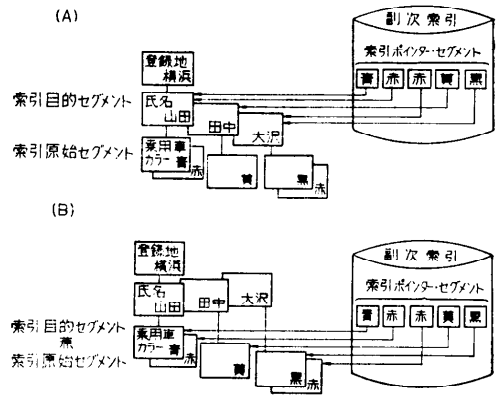

図-8　副次索引の形態

2.5.3 副次データ構造

従属セグメントを索引目的セグメントに選ぶと，そこを入口としてアクセスの対象になり得るのは，上方向に根セグメントに至る階層パス上のセグメントと，索引目的セグメントの従属セグメントである．アクセスの展開順序を考慮に入れると，索引目的セグメントを根セグメントとする新しい論理データ構造（副次データ構造）が生まれる．図-8 の(A)，(B)は図-9 の(A)，(B)の副次データ構造をつくり，これに副次索引を用いない場合の(C)を加えると，1つのデータ・ベースで3通りのデータ構造を使い分けることが可能になる．(A)では，車のカラーを入力して，その所有者名と登録地を調べるのに適し，(B)では，カラーをもとに車の他の特徴，所有者，登録地を知るのに適する．(C)は，市町村別に登録者と車の特徴を一覧するのに便利な構造である．

2.6 論理関係と副次索引の比較

両者の特徴を，表-1(次頁参照)で比較してみた．

3. 物理データ構造

データをディスク上に記憶する形態が物理データ構造で，4種の編成方法がある．セグメントは，編成法

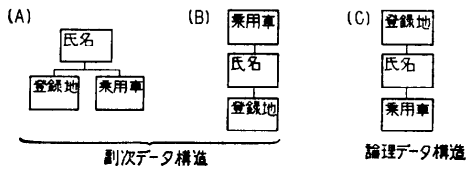

図-9　副次データ構造

表-1 論理関係と副次索引の比較

比較項目	論理関係	副次索引
用途	・複数データ・ベースを関連づける. ・データの重複保有を排除する.	・特定の検索目的を主眼にする.
データ構造と範囲	・1個以上のデータ・ベースを対象. ・ほとんどすべてのセグメントをとり入れる. ・論理的データ・ベースの根セグメントは物理データ・ベースの根セグメントでなければならない. ・1組の論理関係で論理親から論理子へ，またその逆へと2方向のパスにより，大別して2種類の論理データ構造が作れる.	・1個のデータ・ベース内に限定. ・索引目的セグメントの従属セグメントと根セグメントに至る階層パス上のセグメントのみとり入れる. ・根／従属いずれのセグメントでも，副次データ構造の根セグメントにできる. ・1方向のみであり，1種類の副次データ構造を作る.
アクセス	・探索には，すべてのセグメントのデータ項目を利用できる. ・論理親／子の間でのみユニークなデータを保有・利用できる.(部品所要数など)	・副次データ構造の根セグメントの探索は予め選んだ5種類のデータ項目に限定される. ・副次索引データ・ベースのみを対象とした利用ができる.
機能利用の指定	・物理データ・ベースに論理子セグメントを設ける必要がある. ・物理 DBD に加えて論理 DBD が必要.	・副次索引データ・ベースを作成する必要がある.

に共通な記憶の単位で，**図-10** の形式をもつ．セグメント・コードは，セグメント・タイプを識別する数値であり，削除バイトは，物理・論理両面のセグメント削除を把握するビットをもつ．カウンターは，論理子数を持ち，不用意な論理親の削除を防止するのに用いる．ポインター域には，**表-2** から選択されたものが入る．接頭域は IMS が記憶管理し，適用業務プログラムは，データ域のみを処理の対象とする．

3.1 データ・ベースの物理編成

階層順次 (HSAM)，階層索引順次 (HISAM)，階層直接 (HDAM)，階層索引直接 (HIDAM) がある．以下，各編成方法の説明には，図-2 を参照していただきたい．

表-2 ポインターの種類

	ポインターが指す対象のセグメント／カウンターの内容
ポインター	階層順序上，前方向の次のセグメント
	上と逆向きの次のセグメント
	階層順序上，次の物理兄弟セグメント
	上と逆向きの次のセグメント
	論理兄弟セグメント（前方向）
	論理兄弟セグメント（逆方向）
	論理親セグメント
	最初の物理子セグメント
	最後の物理子セグメント
	最初の論理子セグメント
	最後の論理子セグメント

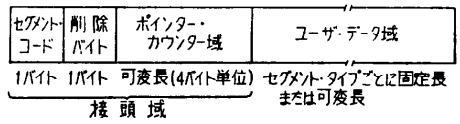

図-10 セグメントの形式

3.1.1 HSAM

各セグメントは，階層順序通りにディスクやテープ上に配列される．追加・削除・置換は直接行わず，更新ずみデータ・ベースを別途に作り出す方式をとる．ポインターは使用しない．従って，論理関係や順次処理にむいたものであるが，乱順処理も可能である．データ・ベース・レコードは，**図-11** のように入る．

3.1.2 HISAM

これも順次編成の一種であるが，索引を持つ．根セグメントから階層順に基本記憶域の論理レコードに入り，残りのセグメントは，あふれ域の論理レコードに収まる．論理レコード間は，ポインターで結ばれる．根セグメントは，キー項目順に配列されるので，順次処理に適すると同時に，索引による乱順処理も可能である．従属セグメントの探索は，その根セグメントから階層順序に従って各セグメントを IMS が調べるため，根セグメントに近い従属セグメントの乱順アクセスの効率はよいが，末端に近い従属セグメントは順次アクセスするのが編成に合った用法となる．

3.1.3 HDAM

HDAM は乱順処理の効率を主眼とする編成法である．副次索引をつけると，順次処理も可能になる．根

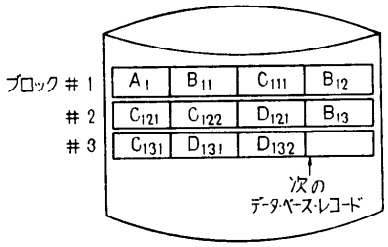

図-11 HSAM 編成の例

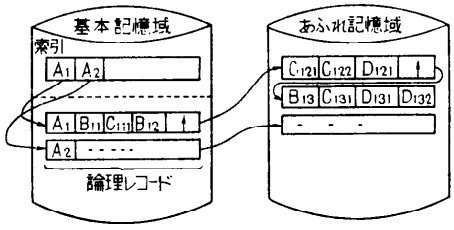

図-12 HISAM 編成の例

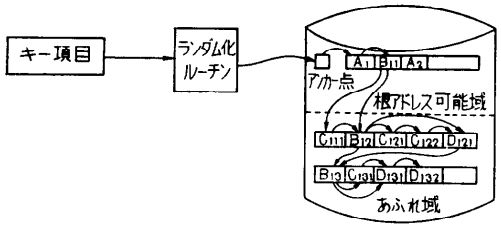

図-13 HDAM 編成の例

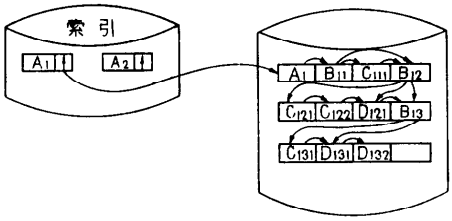

図-14 HIDAM 編成の例

セグメントから階層順に，根アドレス可能域のブロックに入り，残りのセグメントは，あふれ域のブロックに入って両ブロックはポインターで連鎖される．セグメント間は，親子／兄弟や階層順序を示すポインターで結ばれ，これによって論理構造を記憶する．根セグメントは，相対ブロック番号とアンカー点番号でアドレスされる．この番号は，根セグメントのキーをランダム化ルーチンに与えて作り出す．アンカー点はブロック内の根セグメントの所在地を示すもので，これを複数設けると，ブロックを細分化すると同じ効果を持ち，シノニム連鎖を短縮できる．セグメントをポインターで結ぶため，物理的な隣接関係で階層順序を維持する必要がない．追加挿入するセグメントは，効率だけを考慮してできるだけ近隣に空きスペースを見つければよい．空きスペースの管理は，データ・ブロック1個に1ビットを対応させたビット・マップ・ブロックで行い，空きスペースを指すためのアンカー点も設けられる．乱順に従属セグメントをアクセスする場合，親子／兄弟ポインターを用い高速化される．更に，論理データ構造を分割して定義する副次データ・セット機能を使うと，不要な分割部分をスキップして探索するので，一層高速化できることになる．初期作成や再編成時に，全データ域に一定の割合で，空きスペースを均一に分散して設けることができる．これは，追加挿入時等の効率向上の効果がある．

3.1.4 HIDAM

HDAM と同様，乱順編成の1つであり共通点が多いが，データ域は1本化し，索引を持つのが主な相異点である．索引は HISAM のそれと異なり，1個の独立したデータ・ベースで，索引セグメントと根セグメントは論理関係を持っている．他の編成法の長所を集大成した重装備の編成法で，順次・乱順両用の処理に適しているものである（**図-14** 参照）．

3.2 適用業務プログラムとの関連

データに関する記述を適用業務プログラムから取り外して独立させているため，データ・ベースの編成法を例えば HISAM から HIDAM に切り替えても，そのためのプログラムの変更は必要ない．また，新しいセグメント・タイプを追加しても，これをセンシティブ・セグメントにする必要のないプログラムは，そのまま継続使用できる．プログラムの作成・更新に当たっても，編成法に応じて入出力に関する処理部分を書き分ける必要はない．

表-3 DL/I CALL のパラメータ

1. パラメータ数
2. 機能コード
3. PCB 名
4. 入出力域名
5. セグメント探索引数

表-4 機能コード（データ・ベース関連のみ）

機能コード	機　　能
GU	特定セグメントの読み取り
GN	次のセグメントの読み取り
GNP	同一親の次のセグメントの読み取り
GHU	更新のための特定セグメントの読み取り
GHN	更新のための次のセグメントの読み取り
GHNP	更新のための同一親の次のセグメントの読み取り
ISRT	挿入（ロードおよび追加）
DLET	削除
REPL	置換

4. アクセスの方法

4.1 DL/I CALL

IMS のデータ・ベースをアクセスするには，DBD と PSB で構造と範囲を，更に CALL ステートメントで入出力の要求を伝える．CALL パラメータは編成法や論理関係の有無，使用言語に関係なく共通であり，使用者は論理データ構造を基に，適用業務に密着した観点に立ってデータをアクセスできる．**表-3** はパラメータの種類であり，若干の説明を加えたい．機能コードは，基本的な入出力の種類を示すもので，**表-4** がその内訳である．SSA（セグメント探索引数）は，入出力対象のセグメントを指定するための条件を与えるもので，一般形式を**図-15**（次頁参照）に示す．

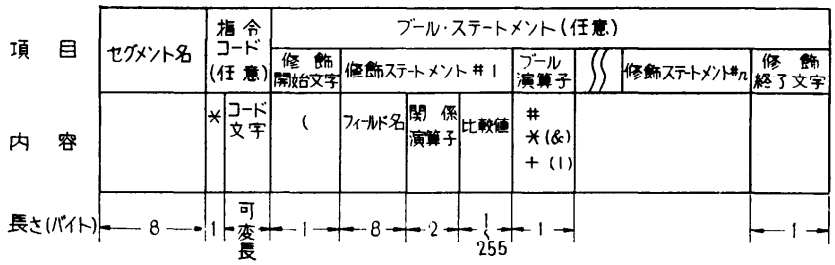

図-15 セグメント探索引数 (SSA)

指命コードは，オカレンス中のアクセス・ポジションの移動，複数セグメントの同時入出力，更新時の排他制御等，特殊な指示を与えるもので，9種類ある．ブール・ステートメントは，セグメント名でセグメント・タイプを指定したあと，更にデータ内容について条件を加えるもので，複合条件は AND, OR 結合で指示する．AND には，独立 AND と従属 AND がある (**4.2.3**)．

4.2 アクセスの例

DL/I の CALL パラメータで，どのような入出力操作ができるかを，読み取りに限定して見てみる．すべて PCB の論理データ構造が処理の基準になる．

4.2.1 順次読み取り

階層順序に従って，次のセグメントを無条件に読み取るには，機能コード GN を用いる．SSA は不要である．データ・ベース・レコードの最後のセグメントの次には，次のレコードの根セグメントが読み取られる．同一タイプのオカレンスを連続して読むには，機能コード GN と，セグメント名だけ指定した SSA を与えればよい．任意のセグメントの従属セグメントだけを順次読み取りの対象とするには，機能コードの GNP を用いる．

PCB で多重位置設定オプションを指定してあるとないとで，GN の読み取り順序を異なったものにすることができる．図-2 の例では，**表-5** のようになる．多重位置設定は，セグメント・タイプごとに，どこまでアクセスしたかを記憶し，他タイプのセグメントへ GN でアクセスが移動しても，後刻そのセグメント・タイプに GN を指定すると記憶してある次のオカレンスから読み取りを継続する機能である．

指令コードFを用いると，処理中のセグメント・タイプの最初のオカレンスにアクセス位置が移動する．最後のオカレンスを指すには指令コードLを用いる．

4.2.2 乱順読み取り

機能コードは GU を用い，セグメント名を常に指定する．他に指定がないと，最初のデータ・ベース・レコード中の指定したセグメント・タイプの最初のオカレンスが読み取られる．特定のセグメントを抽出するには，修飾ステートメントでデータ項目と比較値を，比較の関係演算子 (=<>) で結んで条件を作る．同一セグメント・タイプに複数の抽出条件を与えるには修飾ステートメントをその数だけ増せばよい．修飾ステートメント同士は AND, OR で関係づける．

表-5 単一位置づけと多重位置づけの相違

順序	CALL オペレーション	読み取るセグメント	
		単一位置づけの場合	多重位置づけの場合
1	GU A (KEY=A_1)	A_1	A_1
2	GN B (KEY=B_{12})	B_{12}	B_{12}
3	GN C	C_{121}	C_{121}
4	GN B	B_{21}	B_{13}
5	GN C	なし	C_{122}

従属セグメントを選ぶには，そのセグメントの条件規定の他に，階層パスの上位セグメントに対応する SSA を作り一緒に与える．この上位セグメントも同時に読み取るには，指令コードDを加えればよい．

4.2.3 副次索引の探索

副次索引を用いる場合，SSA に2種の AND 演算子が使用できる．1つは独立 AND (#) で，AND条件を満たすのに同じ索引目的セグメントを指す複数個のポインター・セグメントが必要な場合に用いる．図-8 (A)で，青塗りと赤塗りの2台の車の所有者を検索するには，色を指定する修飾ステートメントを2個用い，この間を#で結ぶと，青と赤のポインター・セグメントが同時に指している目的セグメント（氏名=山田）が検索されることになる．これに対して従属 AND (*)は，索引に登録されたデータ（車の色）とその他のデータとの間の AND 条件の設定に使う．

4.2.4 状況コード

DL/I の入出力操作の結果は，70 数種の状況コードで適用業務プログラムに伝わるため，各種の誤りの有

無，探索の成否，レコード内におけるアクセス・ポジションの移動状況等がプログラムで把握できる．

5. データ・ベースの保全と回復

データ・ベースに，セグメントの追加・削除・置換が行われるつど，変更を記録するログ・レコードが作られる．また，DL/I を CALL して，チェックポイントをとると，このレコードもログ・テープに書かれる．適用業務プログラムが異常終了し，データ・ベースに障害があったとみられるときは，バックアウト・ユーティリティ・プログラムにログ・テープを与えてデータ・ベースを異常終了したプログラムの開始時点か，指定したチェックポイントまで遡って，データ・ベースを復元することができる．

ログ・テープは，ログ累積ユーティリティ・プログラムで累積記録でき，イメージ・コピー・ユーティリティ・プログラムでデータ・ベースのコピーを作成しておくと，これらをデータ・ベース回復ユーティリティ・プログラムにインプットすることにより，長期間遡ったデータ・ベースの回復も可能となる．

6. データ・ベースの再編成

セグメントの追加・削限が度重なると，程度の差はあるが HSAM 以外のデータ・ベースではパフォーマンスを上げるため，また，HISAM では削除スペース再利用のためにも，データ・ベースの再編成が必要になる．論理関係，副次索引を適用している場合を含め，再編成はすべてユーティリティ・プログラムで処理することができる．

7. 階層構造データ・ベース設計の考慮点

7.1 パフォーマンスとディスク・スペース

データ・ベースの設計段階で検討する諸事項は，多かれ少なかれ，パフォーマンス（処理効率）と関連を持つ．編成方法の選択は，何れの編成が諸業務の利用形態によりよく適合するかを数量的に把握した上でなければ，全体として効率のよいものを決定するのは難かしい．また，データ項目を何種類のセグメントにまとめたらよいか，各セグメントを階層構造のどこに位置させるべきかも，パフォーマンスに直接影響する検討項目である．これは人手で行うと，多大の労力と時間を要する作業となる．ところで，パフォーマンスとディスク・スペースの所要量や使用効率とは，しばしば二律排反の関係を持つ．例えば，ポインターには，前後二方向の種類があるが，後方向ポインターは，更新時などに効率よくアクセスするのに役立つ反面，ディスク・スペースはその分だけ多く必要になる．また，両方向の論理関係づけには2通りの方式*があるが，この場合もパフォーマンスとディスク・スペースのいずれを重視するかが，方式選択の唯一の鍵となる．結局は使用者における意思決定に委ねられる問題ではあるが，決定に際して必要となるのは，手軽に試行錯誤ができ，結果を数量的に把握するための設計補助プログラムであろう．

7.2 設計補助プログラム

各種のプログラムが用意されているが，DBDA（データ・ベース設計補助プログラム）は，その1つである．DBDA は，データ項目相互間の使用上の関連を分析し数値化した上で，セグメントにまとめるのが望ましいデータ項目群，親子関係づけ，副次索引の候補セグメント，その他を作表，作図するプログラムである．データ項目の関連分析は，その膨大な作業量のために，人手による場合は自ずと限界があるとされる．しかしながら，基礎的なデータ分析は，効率的，かつ効果的なデータ・ベースの作成に必須のものであり，この意味から DBDA は設計段階での不可欠のツールといえよう．

設計で定めたデータ・ベースの品質水準を維持するには，稼動後のフォロー・アップが必要である．IMS モニターは，実働中のデータ・ベース・システムからパフォーマンス・データを収集するプログラムであるが，これを用いて常態時におけるデータを収集分析しておき，編成，プログラム内容，処理業務の種類等の使用環境に変化が生じた場合に収集したデータと比較することにより，影響の数量的把握ができる．

主記憶装置のバッファ使用に関する統計データは，STAT 機能コードで DL/I を CALL すると得られる．IMS では，一度読み取ったブロックは可能な限りバッファに留め，そのブロック内のセグメントに対する入出力要求は，このバッファ内で処理をすませる方法をとっている．ディスクに対する入出力回数を抑えることにより，パフォーマンスの向上を図るためであるが，適用なバッファ・サイズを定めるには，実際のバッファ使用状況を把握するのが有効である．この点で，STAT 機能はデータ・ベース・システムの維持管理に役立つツールの1つとなっている．

* 論理子を実在するセグメントとして持つ方式と，論理兄弟ポインターで代用しセグメントを持たない方式．

8. むすび

以上，IMS/VS のデータ・ベース機能のあらましと若干の関連事項について述べてきたが，改版や，リリース改定の都度，新機能の付加を続けてきた IMS/VS の V1･M1 をもとにした記述であることを特に付記させていただく．

参考文献

1) Date, C. J.: An Introduction to Data Base Systems.
2) IMS/VS System/Application Design Guide.
3) IMS/VS Application Programming Reference.
44) IMS/VS Utilities Reference Manual.

(昭和51年5月25日受付)
(昭和51年7月9日再受付)