

講座

データ・ベースの管理技術 (I)

上 條 史 彦†

1. データとデータ・ベース

コンピュータで取扱うデータにはいろいろな種類がある。事務計算における英数字を主体としたデータ、技術計算における諸定数や観測点データのように浮動小数点数を含むもの、プログラムの原始モジュール、目的モジュール、ロードモジュールなどが例となる。これらデータをコンピュータ内で処理するためには、データがコード化^{††}されていなければならない。コード化の規則には制約はない。コンピュータの内部表現に対応づけられるものとしては、英数字に対するものとして2進法10進法^{†††}を基本としたいくつかのバリエーションがあり、浮動小数点データや機械語プログラムについては、回路構成に即したものがあることは周知の通りである。また処理プログラムのアルゴリズムで解釈できるものならば、表現法を変換することによってハードウェアによる処理が可能となる。実際問題として、コード化がデジタル表現の範囲内で行なわれ、かつ使用する計算機の標準語長 (character, byte, word など) におさまるもの以外は、現在のデータ・ベース・システムでは使用されていない。技術的な問題もさることながら、経済性の面からの制限がそうさせているのであろう。アナログ・データは理屈の上ではコード化できるが、一般に大量の情報量の取扱いがコンピュータに要求されるため、処理コストに問題が残されている。

データに課せられる第2の条件は形式の標準化である。定形式データ^{††††}の特色は、記録媒体上にあるレコードについて、目録に登録できることである。一般にデータ・ベース・システムの要素としてデータの記述部とデータ操作部があげられるが、データ記述部の

仕事は、この目録を作り保守する作業に他ならない。データの標準形式が既知であり、固定的だという制約は大変強い条件である。将来ともこの条件が存続するとは考えられない。現に一部開発者の間では非定形データ[†]の自動処理技術について模索が始まっている。非定形データで最初にとりあげられるものは恐らく文章^{††}のデータであろう。

形式化・非形式化の対比について、誤解をさけるためにつけ加えたい点がある。現在でも文章のような文字連鎖をファイルの中に貯えることはできる。しかしファイルに可変長レコード^{†††}や無指定レコード^{††††}を貯わえるのとは異なり、データ・ベース・システムではデータの間の論理関係を合わせて記憶しなければ意味がないので、単にレコードを貯蔵するという水準でものを考えてはならない。

データ・ベースという言葉の定義が次の話題であろう。必ずしも一定した定義があるとは思えないが、ここでは下記3つの条件を指向するデータの管理技術を指すものとする。

(a) 必要とされるデータを選択・供給する能力を有すること。

データには定期的に使用される場合と、一時的な使用で十分な場合とがある。前者においては処理効率が大きな問題であり、後者では所要データの所在を示す索引が大きくクローズ・アップされる。さらに一時使用の場合にはそのためのファイルを作成・更新するよりは、使用時に望ましいレコードを組合せたものが一時的にあればよいであろう。

実際に存在する与否とを問わず、単に必要なレコードの組合せがあるように見せかけられればよいとする考え方を仮想データ・ベース^{†††††}と呼ぶ。

† 日本アイ・ビー・エム(株)

†† コード化データ (coded datum) なんらかの規則にもとづいて、原始表現とコンピュータ内での表現が対応づけられるものをいうことにする。

††† BCD (binary coded decimal).

†††† 定形式データ (formatted data) データの記憶媒体上の形式があらかじめ知られているものをいうことにする。

† 非定形データ (unformatted data)

†† 文章 (text data) 自然言語にみられるような文字のつらなり。

††† 可変長レコード (variable length record) 長さが最大値しかわからないレコード。

†††† 無指定レコード (unspecified record) 長さの指定のないレコード。

††††† 仮想データ・ベース (virtual data base) 論理構造 データ目

(b) 必要な時にデータを提供する能力を有すること。

ターン・アラウンド・タイム†を所要時間に合わせられるようにすることが(b)項の条件である。定期的な処理の場合にはプログラムはあらかじめ準備されているだろうが、一時的な処理では流れ図、コーディング、検査、データの準備などの工程が入るので時間がかかる。

(c) 必要な人のみに対してデータを提供すること。

データの安全が最も大きな問題である。ファイルの総合化を計ることが、データ・ベース・システムを用いて利益を得る一因とするならば、データの中央集中は避けられない。集中と機密保護は相反する関係にあるので、いままでに増して保護手段を強化する必要がある。

データを積極的に流布することも、(c)項の眼目の一つである。選択的流布‡をデータの管理者側から行えば、例外管理や技術情報検索などの分野における利用者へのサービスの質を高めることができる。

データ・ベースということばは、アルゴリズムに対比して考えるのが最も自然だろう。原始データを所定のアルゴリズムによって目的データの形に変化させる過程で必要となる全てのデータを、そのアルゴリズムのためのデータ・ベースと定義することもできる。コンパイラを例にとる時、識別子、プロダクションなどコンパイラ固有のテーブルがデータ・ベースの一部となる。

アルゴリズムに対比して定義されるデータ・ベースは最も素直な解釈だろうが、アルゴリズムの内容がわからない限りデータの管理のしようがないので、いわゆる汎用データ・ベース・システムでは、データとして

(a) コード化されていること、

(b) データの形式が既知であること

の2つの条件を課し、さらに前述の3つの目的に副うような応用分野を前提として開発することになっているのである。

録などの手段を通じて記憶されているデータの中から必要なものを組合せあたかも所要のファイルがあるかのように見せかける技術。

† ターン・アラウンド・タイム(turn around time) データ処理を行なう時、処理要求を所要のプログラム、データなどと共に提出してから、結果を手許に得るまでの時間、TAT。

‡ 選択的流布(selective dissemination of information) SDI という略語がよく用いられている。

データ・バンク†ということばもよく使われる。データ・ベースがデータ管理の技術的側面から用いられるのに対し、データ・バンクは内容を問題とする。JIS 中にあるハードウェアの試験基準が、データ・バンクから検索可能かどうか、という場合に用いられるのである。素直な形でデータ・ベースということばをデータ・バンクの代りに、あるいはデータ・バンクから得られるデータの集合に対して用いる場合も見受けられ、混乱を招きやすいので、ここではデータ・ベース・システム‡という語句を、本節で定めたデータ管理技術をさすことばとして今後用いることにしよう。

2. データ・ベース・システム

2.1 データ・ベース・システムの充足すべき条件

(a) データ独立性‡‡

データ独立性の確保はいかなるデータ・ベース・システムにも共通して課せられる基本的な条件である。

プログラムが使用するデータのセットについて論理構造‡‡‡を規定することにより、プログラムを記憶装置にあるデータの種々な特性から全く分離することを目的とする。

独立性には現在3つの水準が考えられている。もっとも基本的な水準が記憶装置のハードウェアとしての諸特性からプログラムを切離すもので、装置からの独立性‡‡‡‡と呼ばれる、2番目の水準はアクセス・メソッドからの独立性‡‡‡‡‡である。プログラムはデータ・セットの編成基準とは独立に、仮想データ・ベースを定義できる。アクセス・メソッドからの独立性には、各種プログラム言語に対する共通のインターフェースを用意する問題も含まれている。現在実現されている手法の中ではもっとも高水準にあるということができる。

未解決の分野として残されている問題の中で最大のものは、データの論理構造の変動であろう。特に構造

† データ・バンク(data bank)。

‡ データ・ベース・システム(data base system)

‡‡ データ独立性(data independence) プログラムが記憶装置、記憶構造、プログラム言語の特性などのデータ側の変動に影響されないようにすること。

‡‡‡ 論理構造(logical structure) プログラム中で行なわれるデータ規定に従う仮想データ・ベース。

‡‡‡‡ 装置からの独立性(device independence)

‡‡‡‡‡ アクセス・メソッドからの独立性(access method independence) ここでいうアクセス・メソッドとは OS/360 で考えだされたもので、標準化されたフィジカル・ファイルすなわちデータセットへの論理的インターフェースを指す。

2.2.1 論理構造

論理構造の基本的な規則は、同じセグメントを同一構造の中では重複させないということである。時系列データのように同一種類のデータをくり返す場合があるので、同型のセグメントの形式を統一してセグメント・タイプと呼ぶことにしよう。

レコード（ロジカル・レコード）の概念はそのまま活かすことができる。しかし同一構造体に属するレコードは全体が関連づけられないと意味を持たないことがあるので、構造体の全部のセグメントをまとめて、一つのデータ・ベース・レコードと呼ぶことにしよう。構造の種類にはリスト、リング（環）、木構造（ツリー）、網構造（ネットワーク）などがある。

(a) リスト構造

リスト構造はセグメント・タイプ間に一定の連結順を指定したもので、頭部と尾部にあたるセグメント・タイプが1つずつしかなく、先行するセグメント・タイプと後続のセグメント・タイプが一意的に定まるものをいう。先行・後続の関係をアドレス・ポインタで示すこともあるが、多くの場合アドレス・ポインタのアクセス時間を節約するため、単に一列に並べて直線的なレコードとして扱う。アドレス・ポインタを用いたものを明示連鎖†または単にチェーン、前後関係から判定する方法を暗示連鎖‡またはインプリシット・チェーンという。

セグメントは同一タイプの中で反復して現われることがある。前後が同じ場合にはその部分だけが反復すると考え、1つ1つのセグメントをオカレンス（反復）††と呼ぶ。Fig. 1 にセグメント・タイプ“b”にだけ反復現象が起きた場合のリスト構造を示してある。

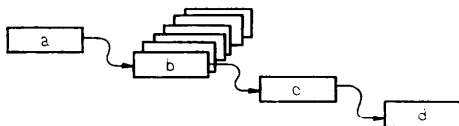


Fig. 1 リスト構造

反復現象が起きた場合には、個々のオカレンス・セグメントから下位のセグメント・タイプが発生し得る。これをサブ・リストと呼ぶことがある。反復されているセグメントに添番号 1, 2, …をつけて、レコードをセグメント・タイプ記号であらわしてみよう。いま b に b₁, b₂, b₃ の3つのオカレンスがあるとする

† 明示連鎖 (explicit chain).

‡ 暗示連鎖 (implicit chain).

†† オカレンス (occurrence) 反復.

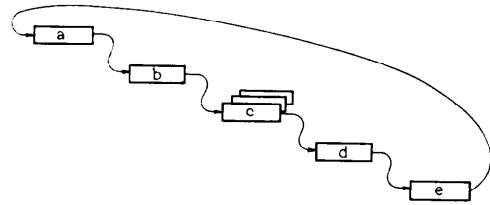


Fig. 2 リング構造 (1)

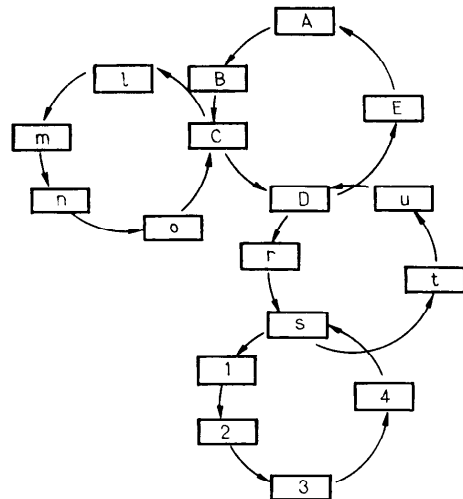


Fig. 3 リング構造 (2)

と、Fig. 1 のリストは下記2つのロジカル・レコードの集合となる。

ab₁cd

ab₂cd

ab₃cd

(b) リング構造†

リング構造はリスト構造の尾部と頭部を論理的に連絡したもので、一つの変型である。Fig. 2 は基本的な構造を示す。

リング構造は、1つのセグメントを他のリングの頭部とすることによって、サブリングをもつものへと拡張することができる。すなわちリングの間で上位・下位の階層関係‡を形成する。

Fig. 2 は単純なリングを、Fig. 3 は階層リングをそれぞれ表現したものである。

(c) 木構造††

木構造はリスト構造に階層の考え方を導入したものである。Fig. 4 では 1, 2, 6 の3つのセグメント・タ

† リング構造 (ring structure) 環構造.

‡ 階層 (hierarchy).

†† 木構造 (tree structure) ツリー構造ともいう.

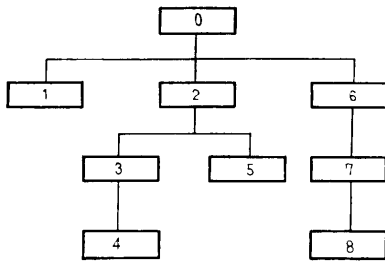


Fig. 4 木構造

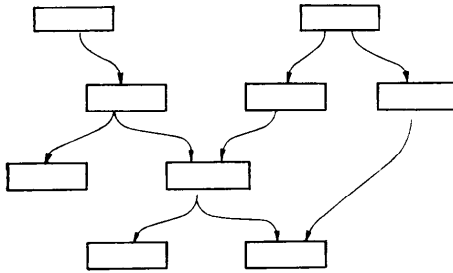


Fig. 5 網構造 (1)

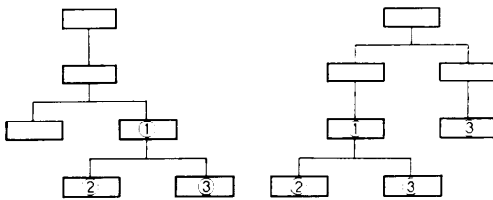


Fig. 6 網構造

イブ,あるいは3,5の2つのセグメント・タイプがそれぞれ同一の階層に属する。

オカレンスは図の表裏方向,すなわち第3次元に追加される。

木構造は次の3つの規則によって展開することにより,線型リストとなることでも知られている。

- (1) 上位のセグメント・タイプから下位へ。
- (2) 同一の枝の中で左から右へ。
- (3) オカレンスを手前から順番につくす。

Fig. 4に付けた番号はオカレンスの無い場合の展開順である。線型リストに直すことにより,記憶構造にブロック・レコードの連鎖が作りやすくなり,データ検索の速度を上げることができる。

なお木構造は,上位セグメント・タイプがただ1つしか許されないデータ構造として定義される。

(d) 網構造†

上位セグメント・タイプを複数個許すと,木構造は

網構造になる。

網構造は応用面では路線図(ルート・マップ)を初めとして多様な機能をもつが,木構造のように簡単に線形リストに展開できないので,記憶構造への変換やデータ保守能率の面では木構造に一步をゆずる。

網構造は一部を重複させることにより木構造に変換できる。Fig. 6はその様子を示したものである。図中,①,②,③としてのセグメント・タイプが2回乃至3回使用されている。

Fig. 6で2個の木構造により網構造が表現できたが,このことは,重複部分をアドレス・ポインタに置き換えることによって,木構造で網構造を表現する可能性を示している。ロジカル・データ・ベース†と呼ばれる技術は定義された木構造のいくつかにまたがって,架空の木構造を定義するもので,多種類のキーの順にデータ・ベース・レコードを並べる機能を持っている。この方法は2次索引を経由するよりも早くデータを検索できる特色を有する。

2.2.2 データ構造の変型

(a) 多重リスト††

前節で考えたリスト構造は一方方向性で単純な形式だった。

多重リストは分岐点を持つリスト構造である。分岐には意味をもたせることにより,データ・ベース・レコードの分類作業をなくすることができる。しかしながら一般にレコードは複雑かつ大きくなり,検索処理が難しい。もともとデータの量が少なく,少数回の入力操作で多量の内部処理ができる索引データ・ベースなどに向いている。

(b) セラ・マルチリスト†††

多重リストの索引を使用すると,大きなデータ・ベースでは装置の特性からくる境界をアドレス連鎖がこえるために,処理速度に大きな影響があらわれる。磁気ディスク装置におけるシリンダ境界はそのよい例で,境界をこえるたびに SEEK 動作が入る。

いまシリンダを1つのセルと考え,アドレス連鎖をシリンダごとに分割しておく,セルの中に入るデータを層化することによってアクセスを早めることができる。

(c) ロジカル・データ・ベース

† ロジカル・データ・ベース(logical data base) アドレス・ポインタを用いて,1つのデータ構造の上にいくつかの見掛けのデータ構造を定義する技術,IBM社のデータ・ベース管理プログラム,IMS/360で開発された。

†† 多重リスト(multi-list)。

††† セラ・マルチリスト(cellular multi-list)。

† 網構造(network structure) ネットワーク構造。

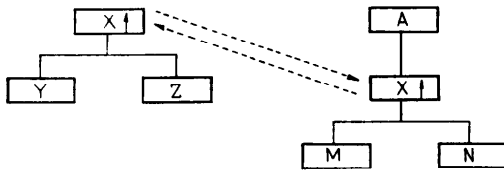


Fig. 7 木構造とアドレス・ポインタの併用

木構造のデータ・ベースを定義し、定義済みのセグメント・タイプを用いて、二重に木構造を定義することができる。基本となる木構造を記憶するのに、レコードごとのアドレス連鎖を利用すると、二重の定義は非常に複雑な多重リストとなり、アクセス時間の面から問題を生ずる。もし基本となる構造の記憶構造に順位による表現（木構造の項を参照）を用いるならば、アドレス連鎖は一重に用いられるだけでロジカル・ツリーが形成される。ここでは IMS/360† で開発された木構造についてふれることにする。理論的には木構造以外にも応用できるはずであるが、利用されていない。

Fig. 7 は 2 個の独立した木構造をアドレス・ポインタによって連絡した図である。連絡するための条件として、両者に重複するセグメント・タイプが存在しなければならない。

両者に共通するセグメント・タイプ X の一方をアドレス・ポインタだけのレコードとし、他方にデータと逆アドレス・ポインタをもたせる。データの重複はなくなり、ポインタを介して、それまでは存在し得なかったロジカル・ツリーが定義できるようになる。

セグメント・タイプ A を新しい木構造へのエントリとしてみよう。展開則により A の直属下位のセグメント・タイプは X である。X は、両方のデータ・ベースにまたがって存在するので、従属するセグメント・タイプとしては Y, Z, M, N が同一階層に並ぶこととなる。Fig. 8 はこのような経路によりできた新しい木構造で、Fig. 7 を使って定義可能なロジカル・ツリーの 1 つである。

X をエントリとしてたどってみよう。直属下位のセグメント・タイプとしては Y, Z の他に A がある。（A の方が M, N より上位。）M, N は A の次にくる。Fig. 9 のデータ・ベース・レコードからなるデータ・ベースでは A と X の位置関係が逆になっている。A をキーとなる項目、X をデータとすると、逆

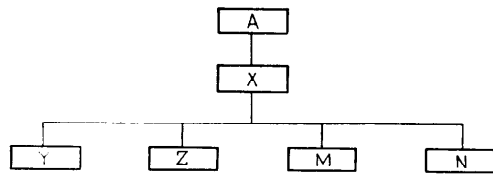


Fig. 8 ロジカル・ツリー (1)

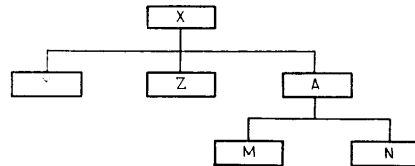


Fig. 9 ロジカル・ツリー (2)

ファイル†††ができて上っている。

逆ファイルあるいは逆索引はデータ検索にあたって便利な道具であるが、データの更新に処理時間を要するので利用し難かった。ロジカル・データ・ベースの技術を応用すると、実際に記憶されるレコードは 1 個所のみであり、かつ基本となる木構造を利用すれば通常のアクセス回数で更新できる。

(d) ポインタ・アレイ††

リスト構造をもととしたデータ構造が全てではない。リスト構造の変形であるところの木構造系に対して、2 方向性を導入したものに網構造があったが、更に一步を進めてレコード内におかれるポインタをアレイの形にまとめたものである。レコードそのものを読まなくてもポインタだけで操作ができる特色を有する。どちらかという記憶構造に近く、アドレス・ポインタに関する操作をデータ・ベース管理システムにまかせないで、利用者が行なう必要がある場合に利用できる。このような特殊な構造までを論理構造に含めるか否かは、データ保全、データ独立性とも関連して議論の余地がある†††。

2.2.3 記憶構造

記憶構造はコア・メモリ中心に開発されたリスト処理手法に負うところが大きい。多くの研究がされており、G. Dodd の論文†††††などに優れた概説と資料表がまとめられている。

† 逆索引 (inverted index).

†† ポインタ・アレイ (pointer array) CODASYL DBTG Report で導入されたデータ形式の 1 つ。

††† CODASYL DBTG Report ではポインタ・アレイは標準の論理構造ではなく、単にリスト構造を実施する上での方法として説明されている。しかしデータ検索を中心して開発されたソフトウェアの中にはこの種のデータ構造を使用しているものがある。

†††† George G. Dodd: Elements of Data Management Systems, Computing Surveys, Vol. 1, NO 2, pp. 117-113 (1969).

† IMS/360 (Information Management System/360) IBM 社のデータ・ベース管理ソフトウェア。

†† 逆ファイル (inverted file) 索引項目を従属フィールドの内容から探しだせるようにしたファイル。

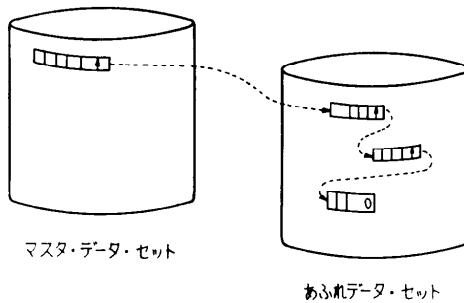


Fig. 10 線型鎖状構造

ところがここで考えているデータ・ベースの記憶装置はコア・メモリのように純粋な等速アクセスではなく、方向性をもった線形メモリの小片をランダムにとり出せるようにしたものである。磁気ドラム、磁気ディスクなどの装置は決して一様な等速アクセス装置ではない。したがって対象となる記憶構造はリストを基本としながらも、基本形からはかけはなれた形になっている。

記憶構造でとりあつかうデータ形式はブロック[†]を単位とする。ブロックの中にはセグメントを1つまたはそれ以上収容するが、セグメントの長さの和はブロックの長さより常に小さい。ブロックにはアドレス・ポインタなどの制御情報や、階層単位情報などを含めることが多い。セグメントの出現回数がタイプごとに異なるため全長が定まらないからである。小型のシステムでは主記憶装置を節約するため可変長ブロックを採用することもある。

(a) 線型鎖状構造

論理構造が線型リストや木構造のように1次元に展開できる場合には、データ・ベース・レコードをマスター・ブロックとそれにつながるいくつかのあふれブロックに分割して、アドレス・ポインタでチェーンする。鎖状構造の場合にはチェーンの回数が検索速度に及ぼす影響が大きいので、実用する際にはデータ・ベース・レコードの長さの統計をとり、それをもとにしてブロック・サイズを適当な大きさに定めてやる必要がある。

アドレス・ポインタとしては装置の絶対番地を使用するものと、ある原点からの相対的位置を使うものがある。後者は2進数表示とできるので、特別に大きなデータ・ベースでない限りアドレス・ポインタに要

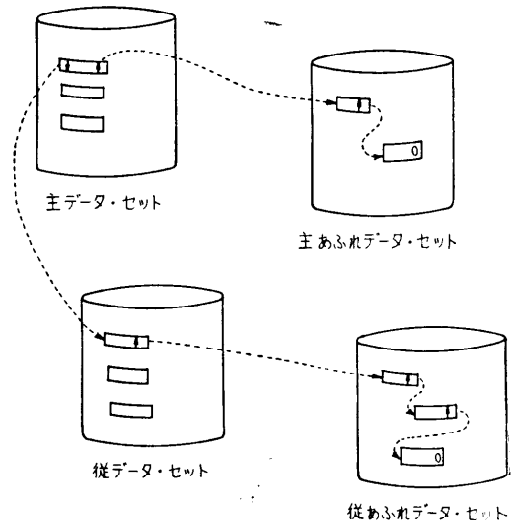


Fig. 11 分岐連鎖

する場所が節約できる。

(b) 分岐連鎖[‡]

一つのデータ・ベース・レコードへのエントリ[‡]をいくつも作る場合には、主ブロックに従ブロックへのアドレス・ポインタを持たせることにより、ブロックによる木構造を作ることができる。大きなデータ構造の一部だけを使用するときには分岐構造を利用するとデータ・ベース・レコードの一部を検索することができるので速度も早くなるし、入出力時間も削減できる。

主ブロックをアドレス・ポインタだけにすると、前出のポインタ・アレイとなる。

(c) 索引つきデータ・ベース

索引にはセグメントに対するものと、ブロックに対するものがある。ここではブロックの索引のみを取上げる。

ブロックの索引はキーと装置アドレスの対比表である。キーはエントリごとに用意する。主データ・セット内のエントリを索引順に並べるとインデックスト・シーケンシャル編成となる。

索引のもち方は大きなテーマなので後章にゆずろう。

(d) その他の記憶構造

磁気テープを対象とした順次編成をデータ・ブロッ

[†] ブロック (blocked record, physical record) レコードはここでは論理データ単位として用いているので、まぎらわしさを避けて単にブロックという語を使用する。ブロックは記憶構造の単位で、それ以上の意味を持たせない。

[‡] 分岐連鎖 (multiple chain)。

エントリ (entry) 入口。データ・ベース・レコードが論理的にいくつもの副レコードに分割できる場合には、頭部と副レコードの組合せの数だけ入口ができる。

クに対して利用することができる。

キーその他のデータを用いてアドレス変換を行ない、ランダム編成にすることも多い。

そのほか記憶構造は装置の特性を活かすために種々

な変型をあみだして利用するので、具体的なシステムについて調べることをおすすめする。

(昭和47年12月4日受付)
