資　料

データベース管理システム

—その機能と応用—

大阪大学大型計算機センター
研究開発部　磯　本　征　雄

1. はじめに

従来の最も一般的な計算機利用の方法は、FORTRANやCOBOL によってプログラムを作成し、データカードや磁気テープに記録されたデータを処理することであった。処理の手続きについての詳細な記述はプログラムでおこなう。そしてデータは、プログラムによって処理される対象物である。このようなプログラムとデータの関係においては、多くの場合プログラムに重点が置かれ、データはプログラムに合せて作られる従属的関係にあったといえる。

データについて考える前に、プログラムについて振り返ってみる。プログラムの多くは、個々のプログラマーが自分の目的に合せて個別に作るのが基本である。ところが、代数方程式や連立一次方程式の数値解法などのように、明確に定義でき、しかも多くの利用者が共通に使える副プログラムについては、プログラム・ライブラリィとして計算機システムに登録して誰でもいつでも自由に利用できるようになっている。この場合、利用者プログラムへの結合手段は計算機システム側で殆ど面倒を見るために、利用者はあたかも自分の作成したプログラムの如くに引用すればよい。

それでは、もし利用者の必要なデータが予め計算機システム内に貯蔵されていて、データ・ライブラリィとして、いつでも利用可能になっているとすればどうであろうか？　そして、それらのデータが多くの利用者によって頻繁に利用されるのであれば、常時計算機システム内に貯蔵しておくことは十分に意味がある。ここで話題にする内容は、これら計算機システム内に半永久貯蔵するデータとそれらの貯蔵及び管理技術についてである。

単にデータの貯蔵という点では、既に従来からファイル管理として様々の方法でおこなわれ

てきた。ところが、多数の利用者による共同利用のデータ・ファイルの場合には、個別に利用・管理されていたファイルとは異なる様々の問題がある。最近大学その他で、研究者間の話題になっているデータベースとは、これらデータの貯蔵・管理・検索などを経済的かつ安全におこなう情報管理技術に支えられたデータ・ファイルのことである。ただし、情報管理技術を離れて単にデータの内容のみを問題にする場合にもデータベースと呼ぶこともあるようであるが、ここでは上記のものをデータベースと呼ぶことにする。

現在、学術情報サービスや学術研究用のデータベースとは何であるかについて、必ずしも統一された意見があるわけではない。しかし現実には、様々の場所で様々の形式のデータベースが開発され[1]、実用に供されている。[2),3)] 当大阪大学大型計算機センターにおいても、一般利用者の幾つかのグループで、データベース開発の計画がすすめられつつある。そして大型計算機センターにおいても、これらを支援するための準備をすすめている。このような状況を踏まえて、ここでは、データベースについて余り馴染みのない方々に、これについて知っていただくための解説をする。

データベースは、計算機システムにおける重要な機能の内の一つとして、いろいろな文献で解説されている。[4)] ところが、FORTRANやCOBOL と違って、データベースを具体的に構築し稼動させようとすると、現時点では親計算機の機能やデータベース支援ソフトウェアであるデータベース管理システムに強く依存せざるを得ない。したがってここでは、データベースの一般的解説というよりは、むしろ当大型計算機センターの計算機システムであるACOSシリーズ77 SYSTEM 800 との関係を述べる。

2. データベースとデータベース管理システム

データベースは、ある一面から見た場合、異なる目的をもった利用者集団によって、相互に独立に同時アクセスされ得るデータの集まりである。各利用者集団が各々で障害なく目的を達成するためには、データベースとして概略次の4つの機能を備えていることが求められている。[5)]

(1) データ構造の構成法

データをファイル上に記録するための記述又は表現方法と考えてよい。COBOLにおけるファイル定義文の記述や FORTRAN における FORMAT 文に相当する。ただし、記述方法は COBOL や FORTRAN よりは一般に複雑である。

(2) データ検索の方法

データベースへのアクセス手順に関する機能である。データベースは通常多量のデータを扱うため、能率のよいアクセスの方法が必要である。また、アクセスの仕方にも異なる幾つかの機能が要求される。FORTRANやCOBOLでいえばREADやWRITEの機能であるが、これ以外にも多くの機能があると考えてよい。

(3) データ独立性

データベースにおけるデータは、これにアクセスするプログラムとは切り離された、独立した存在である。そしてデータベースの構築・管理とデータベースを利用するプログラムの開発は、多くの場合異なる人によってすすめられる。したがってデータベース利用者にとって、データがファイル上に記録されている物理的な構造や手続きに煩わされることなく、データ名のみでアクセスできることが望ましい。このような理想的状況を実現することがデータ独立性である。

(4) データの安全性確保

データベースが多数の利用者によってアクセスされる以上、望ましくないデータのアクセスをおこなう利用者は常にあり得る。たとえば、悪意の有無や意図的か否かにかかわらず、データベースの破壊や盗難は常に考えられるため、その防止策が必要である。また、データベースの更新・書き込み・読み取りのアクセスが常時発生するような状況下では、アクセス相互間で矛盾せず、内容についての混乱がおこらないように、その保全の配慮が必要である。

データベースは、これらのための機能を備えたデータベース管理システムによって管理される。

データベース管理システムは、特定データベースのための専用システムとして開発されるものもあるが、最近では汎用データベース管理システムとしてサービスされるものも多くなった。[6] 当大型計算機センターでは、ADBS[7]、IDS[8]、INQ[9]などの略称で呼ばれる汎用データベース管理システムが既に利用可能となっている。以下において、上記項目について具体例を混えて解説する。

3. データ構造の構成法

データベースは異なる目的をもつ利用者集団によって利用されるため、様々の利用目的に対して巾広く能率的に対処し得る構造で貯蔵されている必要がある。その具体的表現方法はデータベース管理システムごとに異なっているが、一方でCODASYL DBTGの提案として知られるような標準化への動きもある。[10] 上記のADBSは、このCODASYL DBTGの提案に沿うものである。このような貯蔵データの構造を記述する言語を、一般にデータ記述言語(DDL)と呼ぶ。

データベースをレコードの集まりであると見なして、データ構造の構成法の面から分類すると次の3種類に分けられる。

- 階層モデル
- ネットワーク・モデル

○リレーショナル・モデル

以下で、個々について概説する。

(1) 階層モデル

最上位のレコードを頂点として、下位にゆくほど枝分かれしている。多くの場合、上位レコードに対して下位レコードは、それらの属性などをよりくわしく説明するのに使われる。INQ、HITACのADMはこのモデルに相当する。

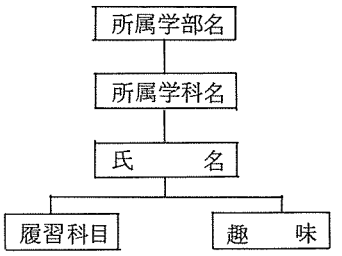


図1　階層モデルによる表現の例

(2) ネットワーク・モデル

レコード間の関係を網構造で記述する。上位レコードと下位レコードの数の対応が、階層モデルでは 1：n であるのに対して、ネットワーク・モデルでは m：n の任意の結合が可能である。

論理構造の上からは、階層モデルよりも複雑な記述が可能である。NEACの ADBSやIDS及びFACOMのAIMはこのモデルである。

図2の例において、ある学部・学科である科目を履習している学生を取り出す場合には、①のパスで検索する。またある学部・学科で、ある趣味をもつ学生を取り出す場合には、②のパスで検索する。①のパスについて具体的に言えば、＜所属学部＝理学部＞＜所属学科＝化学＞＜履習科目＝有機化学＞の順序で条件を与え、最後にこれらの条件を満す総ての学生の氏名が得られる。

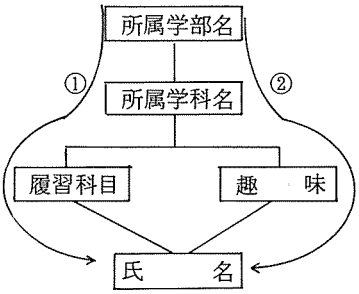


図2　ネットワーク・モデルによる表現の例

(3) リレーショナル・モデル

リレーショナル・モデルでは、情報を n 欄から成る幾つかの表の形式にまとめ、情報間の関係を n 項関係として捉える。

＜所属学部＞	＜所属学科＞	＜氏　名＞	＜履習科目＞
理学部	化　学	谷川	有機科学
工学部	機　械	吉田	力　学
⋮	⋮	⋮	⋮

＜氏　名＞	＜趣　味＞
谷川	登　山
吉田	水　泳
⋮	⋮

図3　リレーショナル・モデルによる表現の例

たとえば、図3の表が与えられているとする。＜氏名＞から＜所属学部＞、＜所属学科＞や＜履習科目＞は決まる。下段の表から＜氏名＞と＜趣味＞の関係が与えられる。そこで、2つの表を合せると、＜氏名＞←→＜履習科目＞と＜氏名＞←→＜趣味＞の関係により表の従属関係を利用することによって＜履習科目＞←→＜趣味＞の関係を導びき出すことができる。このような表の従属関係と表の分解を効率的におこなうのがリレーショナル・モデルの特徴である。ただしこのモデルについては、まだ研究途上にあり汎用データベース管理システムとして実用化されているものは、国内にはまだ無い。

4. データ操作の方法

データベースへのアクセスを行なう命令や記述を一般にデータ操作言語（DML）という。データ操作言語を大別すると次の2種類がある。高級汎用言語であるCOBOL，FORTRAN，PL／I などを基本にして、その上にデータベース・アクセスのための機能を追加する形式になっているデータベース管理システムを親言語方式（親言語型）という。多くの汎用データベース管理システムがこの方式である。親言語が COBOL，FORTRAN，PL／I のいずれであるかはシステムによる。因みに、ADBSではCOBOLが、INQではCOBOLとFORTRANが親言語である。他方、データベース・アクセスのための機能が一つの閉じた言語体系を成すものを独立言語方式（独立言語型）という。

独立言語方式は、適用範囲の制限の厳しさに不安はあるが、可能な処理に対しては簡単な記述で高能率な処理を期待できる。しかし、データベースに対して幅広い応用を考え、相当に複雑な利用が予想される場合には、親言語方式のデータベース管理システムが適する。

プログラマ利用者がデータベース・アクセスのための機能を呼び出す方法は、個々のシステムによる。たとえば、親言語をCOBOLとするADBSにおいてデータを格納する場合には、

```
                •
                •          } COBOL の命令
                •
STORE  RECORD－1.     ADBS の命令
                •
                •          } COBOL の命令
                •
```

のように書かれる。つまり、レコード名が RECORD－1 であるレコードにデータを格納する場合には、必ず動詞STOREを書き、これにつづいてレコード名を書いてピリオドで終了する。ピリオドの後は再度、親言語のCOBOLとしてのプログラムが続く。

一方、親言語をFORTRAN又はCOBOLとするINQにおいては、データを格納する場合には次のようになる。

〔COBOLの場合〕

```
                    .
                    .                          } COBOL の命令
                    .
CALL  INQ USING "STORE" "/RECORD-1/".     INQ の命令
                    .
                    .                          } COBOL の命令
                    .
```

〔FORTRAN の場合〕

```
                    .
                    .                          } FORTRANの命令
                    .
CALL  INQ("STORE", "/RECORD-1/")          INQ の命令
                    .
                    .                          } FORTRANの命令
                    .
```

このように、INQでは副プログラムINQを引用する表現がとられる。

データベース管理システムには、それぞれに固有の制御用、検索用および更新用等の機能のために10数種のデータ操作命令がある。したがって、親言語の中でデータベースをアクセスする場合、適切なパラメターを予め設定したうえでデータ操作命令を実行する。ただし、プログラム・コンパイル時点では、単に親言語をコンパイルするためのジョブ制御カードのほかに、いくつかのパラメターの追加やジョブ制御カードの追加が必要なので注意を要する。

5. データ独立性

データベースにおいては、データがファイル上に格納されている物理的な構造や手続きに関係なしにプログラム側でデータ操作言語の記述が可能になっていることが理想的である。データ独立性とは、この理想を達成するためのデータベースのあり方である。もっと具体的にいえば、処理プログラムでデータの属性をいっさい考えることなく、データ名だけで処理できることが望ましい。データベースにとってデータ独立性は最も基本的な目標であるが、しかし完全なデータ独立性を保障するシステムはまだつくられていない。また、データ独立性を満すための具体的方策となると、システムごとに異なる。

データベース管理システムにおいては、データベースとこれにアクセスするプログラムとをはっきりと分離する。データベースがどんなデータでどのように編成されているかを記述するために、データ記述言語がある。そしてデータベースを利用するプログラムでデータをどんな形で利用するかは、プログラム側の課題であって、プログラム記述言語の対象ではない。このようにしてデータ独立性が満たされたならば、少くともプログラム側から見た場合には、ファイル装置内でのレコードの長さ・ブロックの大きさ・バッファの大きさや数などといった厄介な問題から解放される。

データ記述言語がデータベースの記述であるならば、データ操作言語は処理プログラムの記

述であるといえる。データ操作言語は直接にデータ記述言語を通してデータベースにアクセスするわけではなく、中間に別のデータ記述の機能を介在させる。以下にADBSとINQの場合について個々に説明する。

ADBSにおいては、データベース全体を記述したものをスキーマと呼ぶ。スキーマでは個々のレコードについての属性やレコード間の関係が記述される。スキーマの内でプログラム側で必要な最少限の記述のみを抜き出して、これをプログラムに結合するものをサブスキーマと呼ぶ。スキーマはデータベースについて一つであるが、サブスキーマは利用方法ごとに作られる。図4は、このことに関する概念図である。理解を深めるために、ADBSにおけるスキーマとサブスキーマの記述について、図5と6に具体例を示す。ただし、この例は文献[11]から写したものである。

図5aにおいて、□内はレコード名であり、これについての宣言はスキーマ（図5b）の中のRECORD NAMEの所で、各レコードごとになされる。また、レコード間の関係は図5aにおい

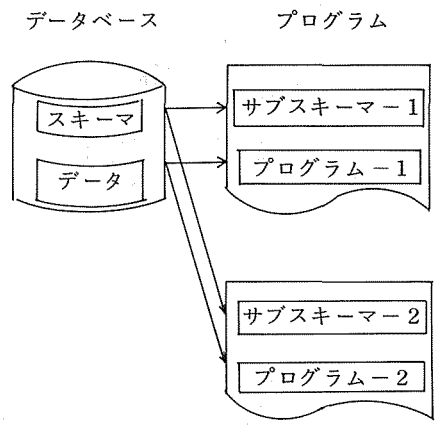


図4　ADBSにおけるデータベースのスキーマとサブスキーマの関係

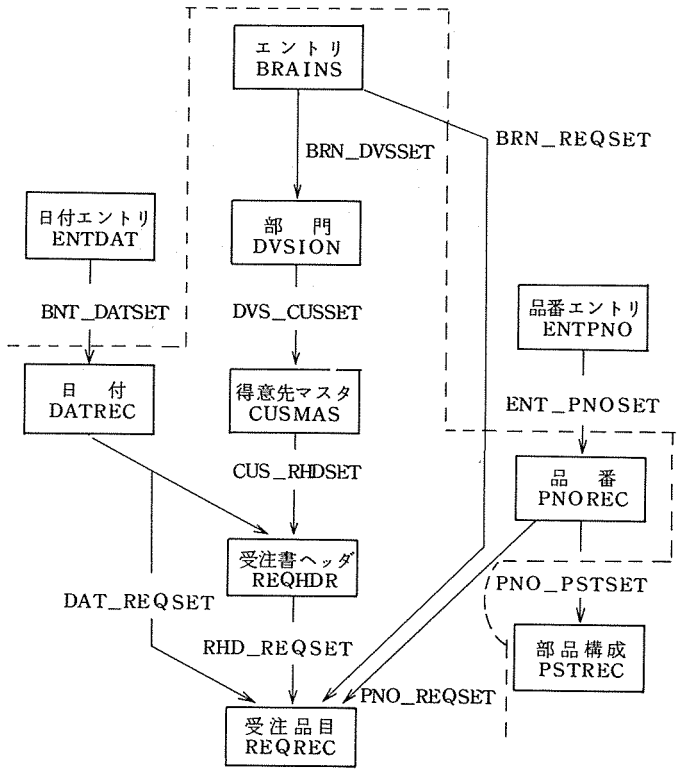


図5a　スキーマデータ構造（ISCHEMA）

```
SCHEMA NAME IS ISCHEMA.
AREA NAME IS IAREA0.
AREA NAME IS IAREA1.
AREA NAME IS IAREA2.
AREA NAME IS IAREA10.
RECORD NAME IS ENTPNO
    LOCATION MODE IS CALC USING PNOORG DUPLICATES NOT ALLOWED
    WITHIN IAREA0.
    01  PNOORG     TYPE IS CHARACTER 10.
    01  PNOWUF     TYPE IS CHARACTER 32.
RECORD NAME IS PNOREC
    LOCATION MODE IS CALC USING PNOPNO DUPLICATES NOT ALLOWED
    WITHIN IAREA0.
    01  PNOPNO     TYPE IS CHARACTER 10.
    01  PNMAXT     TYPE IS DECIMAL 2.
    01  PNNAME     TYPE IS CHARACTER 20.
    01  PNCODE     TYPE IS CHARACTER 1.
    01  PNMAKE     TYPE IS CHARACTER 1.
    01  PNSTCM     TYPE IS CHARACTER 1.
RECORD NAME IS PSTREC
    LOCATION MODE IS VIA PNO_PSTSET
    WITHIN IAREA0.
    01  PSVOLM     TYPE IS DECIMAL 7.2.
    01  PSSUPP     TYPE IS CHARACTER 1.
RECORD NAME IS ENTDAT
    LOCATION MODE IS CALC USING DATORG DUPLICATES NOT ALLOWED
    WITHIN IAREA1.
    01  DATORG     TYPE IS DECIMAL 6.
RECORD NAME IS DATREC
    LOCATION MODE IS CALC USING DRDATE DUPLICATES NOT ALLOWED
    WITHIN IAREA1.
    01  DRDATE     TYPE IS DECIMAL 6.
    01  DRHOLI     TYPE IS DECIMAL 1.
RECORD NAME IS BRAINS
    LOCATION MODE IS CALC USING BRAINE DUPLICATES NOT ALLOWED
    WITHIN IAREA10.
    01  BRAINE     TYPE IS CHARACTER 4.
RECORD NAME IS DVSION
    LOCATION MODE IS CALC USING DVSCOD DUPLICATES NOT ALLOWED
    WITHIN IAREA10.
    01  DVSCOD     TYPE IS CHARACTER 4.
RECORD NAME IS CUSMAS
    LOCATION MODE IS CALC USING CUSCOD DUPLICATES NOT ALLOWED
    WITHIN IAREA10.
    01  CUSCOD     TYPE IS CHARACTER 10.
    01  CUSNAM     TYPE IS CHARACTER 15.
    01  CUSPST     TYPE IS CHARACTER 5.
    01  CUSADR     TYPE IS CHARACTER 35.
RECORD NAME IS RECHDR
    LOCATION MODE IS CALC USING RHDRNO DUPLICATES NOT ALLOWED
    WITHIN IAREA10.
    01  RHDRNO     TYPE IS CHARACTER 11.
RECORD NAME IS REQREC
    LOCATION MODE IS VIA RHD_REQSET
    WITHIN IAREA10.
    01  RRLINE     TYPE IS DECIMAL 2.
    01  RRRVOL     TYPE IS DECIMAL 6.
    01  RRSVOL     TYPE IS DECIMAL 6 SIGNED.
    01  RRMARK     TYPE IS CHARACTER 1.
```

図5b　ADBS スキーマの例

```
SET NAME IS ENT_PNOSET
    OWNER IS ENTPNO
    SET IS PRIOR
    ORDER IS PERMANENT INSERTION IS LAST.
    MEMBER IS PNOREC MANDATORY AUTOMATIC
    LINKED TO OWNER
    SET SELECTION FOR ENT_PNOSET IS THRU ENT_PNOSET
        OWNER IDENTIFIED BY CALC_KEY.
SET NAME IS PNO_PSTSET
    OWNER IS PNOREC
    SET IS PRIOR
    ORDER IS PERMANENT INSERTION IS FIRST.
    MEMBER IS PSTREC MANDATORY AUTOMATIC
    LINKED TO OWNER
    SET SELECTION FOR PNO_PSTSET IS THRU PNO_PSTSET
        OWNER IDENTIFIED BY CALC_KEY.
SET NAME IS PNO_REQSET
    OWNER IS PNOREC
    SET PRIOR
    ORDER IS PERMANENT INSERTION IS LAST.
    MEMBER IS REQREC MANDATORY AUTOMATIC LINKED TO OWNER
    SET SELECTION FOR PNO_REQSET IS THRU PNO_REQSET
        OWNER IDENTIFIED BY CALC_KEY.
SET NAME IS ENT_DATSET
    OWNER IS ENTDAT
    SET IS PRIOR
    ORDER IS PERMANENT INSERTION IS SORTED DEFINED KEYS.
    MEMBER IS DATREC MANDATORY AUTOMATIC
    LINKED TO OWNER
    KEY IS ASCENDING DRDATE
    SET SELECTION FOR ENT_DATSET IS THRU ENT_DATSET
SET NAME IS DVS_CUSSET
    OWNER IS DVSION     SET IS PRIOR

        OWNER IDENTIFIED BY CALC_KEY.
SET NAME IS DAT_REQSET
    OWNER IS DATREC
    SET IS PRIOR
    ORDER IS PERMANENT INSERTION IS LAST.
    MEMBER IS REQHDR MANDATORY AUTOMATIC LINKED TO OWNER
    SET SELECTION FOR DAT_REQSET IS THRU DAT_REQSET
        OWNER IDENTIFIED BY CALC_KEY.
    MEMBER IS REQREC MANDATORY AUTOMATIC LINKED TO OWNER
    SET SELECTION FOR DAT_REQSET IS THRU DAT_REQSET
        OWNER IDENTIFIED BY CALC_KEY.

SET NAME IS BRN_REQSET
    OWNER IS BRAINS  SET IS PRIOR
    ORDER IS PERMANENT INSERTION IS LAST.
    MEMBER IS REQREC OPTIONAL MANUAL LINKED TO OWNER
    SET SELECTION FOR BRN_REQSET IS THRU BRN_REQSET
        OWNER IDENTIFIED BY APPLICATION.
SET NAME IS BRN_DVSSET
    OWNER IS BRAINS     SET IS PRIOR
    ORDER IS PERMANENT INSERTION IS SORTED BY DEFINED KEYS
        DUPLICATES NOT ALLOWED.
    MEMBER IS DVSION MANDATORY AUTOMATIC   LINKED TO OWNER
    KEY IS ASCENDING DVSCOD
    SET SELECTION FOR BRN_DVSSET IS THRU BRN_DVSSET
        OWNER IDENTIFIED BY CALC_KEY.
    ORDER IS PERMANENT INSERTION IS LAST.
    MEMBER IS CUSMAS MANDATORY AUTOMATIC   LINKED TO OWNER
    SET SELECTION FOR DVS_CUSSET IS THRU DVS_CUSSET
        OWNER IDENTIFIED BY CALC_KEY.
SET NAME IS CUS_RHDSET
    OWNER IS CUSMAS
    SET IS PRIOR
    ORDER IS PERMANENT INSERTION IS LAST.
    MEMBER IS REQHDR MANDATORY AUTOMATIC LINKED TO OWNER
        DUPLICATES NOT ALLOWED FOR RHDRNO
    SET SELECTION FOR CUS_RHDSET IS THRU CUS_RHDSET
        OWNER IDENTIFIED BY CALC_KEY.
SET NAME IS RHD_REQSET
    OWNER IS REQHDR
    SET IS PRIOR
    ORDER IS PERMANENT INSERTION IS LAST.
    MEMBER IS REQREC MANDATORY AUTOMATIC LINKED TO OWNER
    SET SELECTION FOR RHD_REQSET IS THRU RHD_REQSET
        OWNER IDENTIFIED BY CALC_KEY.
END_SCHEMA.
```

て矢印で示され、スキーマにおいてはセットと呼ばれる。各セットについての宣言は、スキーマ（図5b）のSET NAME の所で各セットごとになされる。矢印は上位レコードから下位レコードに向い、上位レコードをOWNERと呼び下位レコードをMEMBERと呼ぶ。RECORD NAME，SET NAMEともいずれもそれに引きつづいて属性が記述される。詳細は紙数の関係から割愛する。

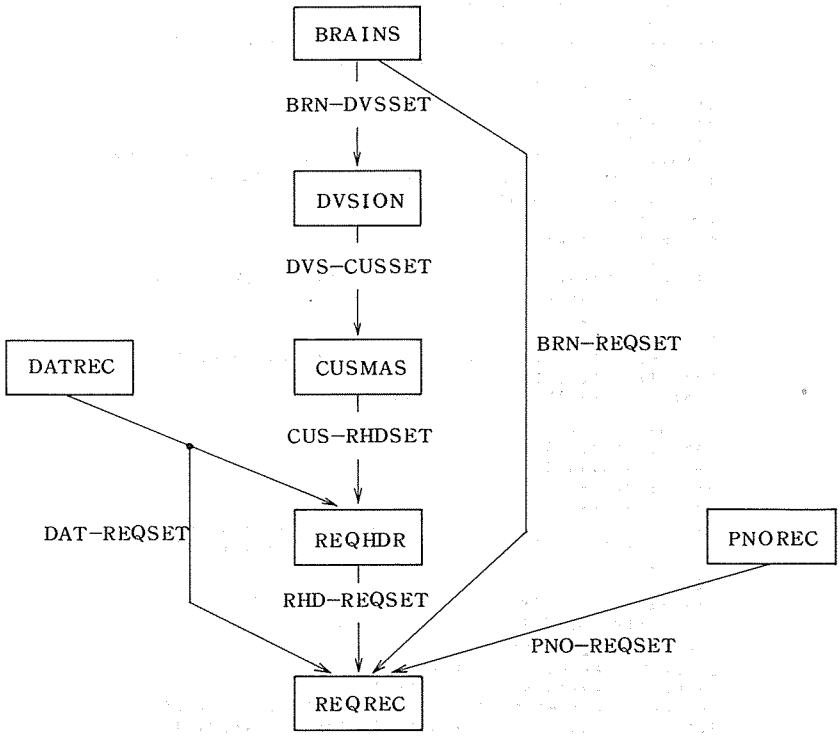


図6a　サブスキーマ構造図
（ISUBSCH）

図6aは、図5aの一部分（図5aの点線内）を抜き出したデータ構造図である。図6aの意味は図5aと同じである。図6bはスキーマ（図5b）に対するサブスキーマである。COBOLプログラムの中で実際にこのとうりに記述される。スキーマ（図5b）と比較してサブスキーマ（図6b）では、SET SECTION，RECORD SECTIONとして最少限の記述に留められている。

INQのデータベースでは、データ独立性に対する方策がADBSの場合とは異なる。INQのデータベースは、相互に独立なファイルの集合として扱われる。そして個々のファイルの属性を記述する言語をファイル記述言語（FDL）と呼ぶ。INQのFDLはADBSのスキーマに相当する。FDLは1つのファイルに1つある。図7には、データ構造図とこれに対するFDL

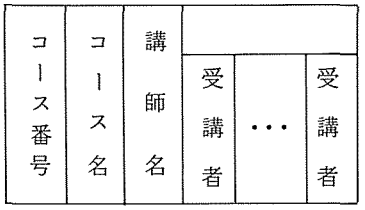
```
TITLE DIVISION.
SS  ISUBSCH WITHIN ISCHEMA.
MAPPING DIVISION.
STRUCTURE DIVISION.
REALM SECTION.
RD  IAREA0.
RD  IAREA1.
RD  IAREA10.
SET SECTION.
SD  PNO-REQSET.
SD  DAT-REQSET.
SD  BRN-REQSET.
SD  BRN-DVSSET.
SD  DVS-CUSSET.
SD  CUS-RHDSET.
SD  RHD-REQSET.
RECORD SECTION.
01  PNOREC.
    02  PNOPNO      PIC X(10).
    02  PNNAME      PIC X(20).
    02  PNSTOC      PIC X(3).
    02  PNVOLM      PIC 9(5)V99 SIGN IS TRAILING.
    02  PNCOST      PIC 9(7)V99.
    02  PNSTOD      PIC 9(7)V99.
01  DATREC.
    02  DRDATE      PIC 9(6).
    02  DRHOLI      PIC 9.
01  BRAINS.
    02  BRAINE      PIC X(4).
01  DVSION.
    02  DVSCOD      PIC X(4).
01  CUSMAS.
    02  CUSCOD      PIC X(10).
    02  CUSNAM      PIC X(15).
    02  CUSADR      PIC X(35).
01  REQHDR.
    02  RHDRNO      PIC X(11).
01  REQREC.
    02  RRLINE      PIC 9(2).
    02  RRRVOL      PIC 9(6).
    02  RRSVOL      PIC 9(6).
    02  RRMARK      PIC X.
END ISUBSCH.
```

図6b　ADBSサブスキーマ
（ISUBSCH）

を示した。[12] FDLの記述方法は、COBOLのDATA DIVISION におけるファイル定義文のレコード記述に類似している。ただし集団項目名の後に（N）の付されているものは、不定繰り返し項目と呼ばれ、データの繰り返しが任意回だけ許される。前述のとうり、データ構造は階層構造である。また、ファイル及びこれに対するFDLは必要ならば何個でも設けることができる。

ADBSにおけるサブスキーマに相当するものは、INQではINQセクションと呼ばれる。図7のFDLに1対1に対応するINQセクションは、図8に示されたものである。ここでは、INQセクションとFDLでデータ名の異なる場合の例を示した。各々カッコ内にFDLのデータ名を書くことによって対応関係は決まる。INQセクションの特徴は、サブスキーマとは異

ファイル名 ： COURSE - FILE （講習ファイル）
パスワード ： COURSE % 2627
データ構造 ：




```
FDL          COURSE-FILE.2  CARD.
PASSWORD     COURSE%2627.
02     ｺｰｽﾊﾞﾝｺﾞｳ                    PIC 9(4)     KEY.
02     ｺｰｽﾒｲ                        PIC X(20).
02     ｺｳｼｰﾒｲ                       PIC X(28).
02     ｼﾞｭｺｳｼｬ  (N).
  03     ｼｬｲﾝ#                      PIC 9(7).
END
```

図７　データ構造及びFDL

```
COBOL   INQ   SECTION.
NAME    COURSE-DATA.
FILE    COURSE-FILE.2,C-TYPE/1/.
            02  COURSE-NO      (ｺｰｽ-ﾊﾞﾝｺﾞｳ)      PIC 9(4).
            02  COURSE-NAME    (ｺｰｽ-ﾒｲ)          PIC X(20).
            02  LECTUREP       (ｺｳｼｰﾒｲ)          PIC X(28).
            02  ATTENDANCE     (ｼﾞｭｺｳｼｬ).
              03  EMP-NUMB       (ｼｬｲﾝ#)         PIC 9(7).
```

図８　INQセクションの例（図７のFDLに対応）

なり、 FDLの段階で分離して定義されたデータ構造を処理プログラム段階で２つ以上を結合して１つの論理構造として定義可能なことである。

このように、ADBSやINQの実例をみたように、データベースの記述と応用プログラムでのデータ構造の記述が分離されている点が、データベースの大きな特徴である。

6. 情報サービス・システムとしてのデータベース

データベースが単なる大容量データファイルとしてのみ存在することは少ない。実際には、データベースを背後に控えて、端末などを通して様々の目的に利用されている。因みに、当大型計算機センターにおいても、蛋白質研究所との共同で蛋白質結晶データのオンライン・サービス・システムを開発している。これは、蛋白質結晶データのオンライン・サービスを直接の目的とし、蛋白質構造データベースとして公開の予定であり、現実に一部稼動中で、アクセス可能となっている。本データベースは、単にデータベース管理システム（INQ）の範囲だけで

なく、オペレーティング・システム（ACOS-6）の中でどのように稼動し得るかを示すための好例であると思うので、ここにその内容を簡単に紹介する。

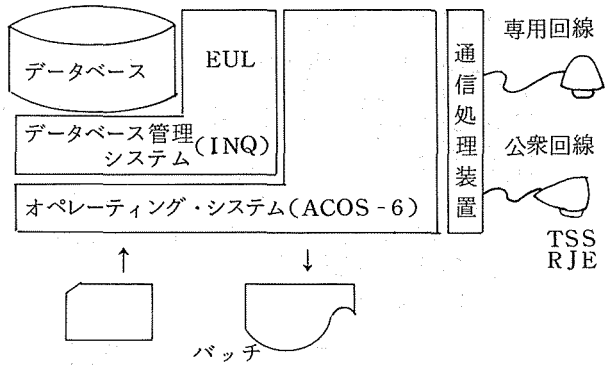


図9　蛋白質構造データベース・システム構成

蛋白質構造データベースは、汎用データベース管理システム INQ を使ってデータの格納や管理をおこなっている。当然、データベースはオペレーティング・システム ACOS-6 の下で管理される。したがって、データベースへのアクセスは、ローカル・バッチ、リモート・バッチ、会話型リモート・バッチ、TSS のいずれの処理モードからも可能である。ただし、機能上の部分的制限や能率の面から本データベースとしては、次のような形態で管理・運用される。

データの格納やデータベースの管理一般については、総てローカル・バッチによっておこなう。なぜなら、これらの作業は非常に重要でありかつ手間のかかるものであり、安全性から見ても管理者が目前で直接確認しながら行うのが最善と思われるからである。応用プログラム開発は、機能上の制限から、ローカル・バッチ、リモート・バッチ、会話型リモート・バッチのいずれかで行わなければならない。

データベースの管理とデータベース・アクセスのための応用プログラム開発は、相互に独立の作業としてすすめる。管理者は、データ構造について INQ セクションとして整理し、プログラム開発者に手渡す。プログラム開発者は、INQセクションをプログラムに組み込みデータベースをアクセスする。データ独立性の充分満たされたデータベース管理システムであれば、このような作業分担は現実場面でも十分に円滑にすすむはずである。

蛋白質構造データベースでは、プログラム開発をしなくても、TSS端末よりデータベースをコマンドによってアクセスできる。これらデータベース・アクセスを直接におこなう TSS コマンドの体系を通常、エンド・ユーザ言語という。

端末からエンド・ユーザ言語によってデータベースにアクセスする方法は2つある。第1は、INQの機能の一つとして備わっているエンド・ユーザ言語 EQL[13] を利用する方法である。EQL は一つの独立したコマンド体系を成す。図10は、実際に蛋白質構造データベースに対してEQLでアクセスした例である。ログオン時のUSERID が何であってもこの例は実行可能である。図10 の詳細な説明は、EQL説明書[13] 及び蛋白質構造データベース利用説明書[14] を参照して下さい。EQLを使うことの利点は、データベースを構築すれば以後はプログラム

```
SYSTEM? INQ
   INQ EQL VERSION 3.4
OPTION FILE ?
INQ SECTION ? INQFNUM,PROTEIN-DB/INQLIB
INQ SECTION ?
INQ FILE ? F-NUMERC
PASSWORD (F-NUMERC     ) ?
INQ CAT/FILE ? PROTEIN-DB/F-NUMERC
INQ CAT/FILE ?
INQ FILE ?

     INQ DATA BASE RETRIEVE START

? RETRIEVE ID-CODE = 2CHA
  304       RECORDS FOUND

? DISPLAY ID-CODE ATOM-NAME COORDINATE

 ID-CODE   ATOM-NAME   X          Y          Z

 2CHA        N          7.726     -6.538     31.638
             CA         8.089     -5.785     30.434
             C          8.997     -6.642     29.562
             O         10.082     -7.008     30.027
             CB         8.777     -4.517     30.943
             SG        10.034     -3.573     30.057

 2CHA        N          8.483     -7.076     28.421
             CA         9.257     -7.930     27.505
             C          9.116     -9.402     27.872
             O         10.110    -10.009     28.295

                    ( 途 中 省 略

  ? DONE

       INQ DATA BASE RETRIEVE END

  SYSTEM ?
```

図10　INQ がその機能の1つとしてもつエンド・ユーザ言語 EQL によって蛋白質構造データベースを検索した実例。この例は、USERID が何であっても実行可能です。

開発することなく検索可能なことである。

もう一つは、COBOLまたはFORTRANで実行形式のプログラムを作成し、H*ファイルに登録し、これを端末より呼び出す方法である。蛋白質構造データベースにおける書誌情報に対

する検索プログラム SEARCH は次のように呼び出される。

SYSTEM ?　PROTEIN-DB／SEARCH，R

このコマンド入力後、端末利用者とシステムの会話が開始され検索がすすめられる。

蛋白質構造データベース固有のエンド・ユーザ言語としては、次の機能を備える予定である。

① 書誌情報，構造上の特徴，化学的性質などを条件とした蛋白質の検索

② 書誌情報，構造上の特徴，化学的性質，原子座標値などの端末又はファイル上への出力

③ 蛋白質結晶の骨格図形，部分的な詳細図形などのグラフィック・ディスプレイ上への出力

④ 上記③の図形に対する回転，部分的平行移動などの図形操作

このような複雑な操作を要するエンド・ユーザ言語については、データベースごとにデータ構造や利用目的に合せてエンド・ユーザ言語を開発する必要がある。

最後に、データベースを作る人への参考のために、データロードの手順について簡単に述べる。限られた頁数で詳細な説明は不可能なので、ここではどの程度の手間がかかるものであるかがわかる程度に留める。

蛋白質構造データベースは、次の手順で作成しデータ・ロードがなされる。

① データベース・ファイルの作成

② データベース・ファイルの初期化

③ FDL の登録

④ INQ セクション・ライブラリィの登録

⑤ } ロード用データの編集
⑥

⑦ ユーティリティ・プログラム（LOADER-1）による実データのロード

⑧ ユーティリティ・プログラム（LOADER-2）による検索用キーワードのロード

これらの手続きは、各々が1つのアクティビティから成っており、全体で1つのジョブ・デックを構成する。ACOS-6 においては、このようなアクティビティの結合や分離は非常に簡単にできるので、作業手順の変更は容易に可能である。

通常、データベースの管理には人手がかかる。そこで、本データベースにおいては、省力化を重点目標の1つとした。実際に上記のデータロードの手順においても、人手の必要なのはカードリーダからのジョブ・デックの投入と、磁気テープからの原始データの投入の2回だけである。しかも1度方式を決めてしまえば、以後は単純なルーチン・ワークとなる。

7. ま　と　め

本解説では、現在話題になっているデータベースの様々な特徴を概説し、さらに情報サービ

スのために開発された蛋白質構造データベースの仕様について簡単に紹介した。正確に解説し得たか否かには多少の不安もあるが、未だデータベースについて馴染みのない方々が、今後この方面での計算機の利用を考えるうえでいくらかの参考になれば幸いである。筆者としては、とかく敷居の高くなりやすい計算機利用において、データベースができる限り幅広く多方面の方々に利用されることを願って、ここに筆をとった次第です。

参 考 文 献

1）文部省科学研究費による特定研究「情報システムの形成過程と学術情報の組織化，総括班：情報システムの形成過程と学術情報の組織化，第２年次報告，1978年３月。

2）山本毅雄，他：TOOL-IR利用マニュアル(上)，文部省特定研究「広域・大量情報の高次処理」，化学班，昭和５０年３月。

藤原鎮男，他：TOOL-IR論文集(1)，文部省特定研究“広域・大量情報の高次処理，開発C班，1975年１２月。

3）鳴海元，他：汎用文献検索システム，HUNDRED,オペレーションガイド，検索編，文部省科学研究費による試験研究(2)，「汎用文献情報検索システムの実用化に関する研究」，昭和５２年９月，昭和５３年１月（補遺版）。

山本純恭，他：数学文献データベース IMP のHUNDREDシステムによる検索の手引、文部省科学研究費による特定研究「情報システムの形成過程と学術情報の組織化B-18班，昭和５３年１月。

4）データベース特集号，情報処理，Vol. 17，№10，情報処理学会，(1976)。

公共情報システム特集号，情報処理，Vol. 18，№10，情報処理学会，(1977)。

大須賀節雄：連載，データ構造１～８(完)，コンピュータ・サイエンス誌，bit，Vol. 8，№5～13，共立出版，(1976)。

5）上條史彦：コンピュータ・サイエンス・シリーズ，データ・ベース・システム，産業図書，昭和５０年。

6）西村恕彦：データベースの選び方，コンピュータ・サイエンス誌，bit，臨時増刊，４月号，共立出版，(1977)。

西村恕彦，他：データベースシステム，共立出版，(1977)。

7）日本電気：ADBS 概説書

8）日本電気：IDS 概説書

9）日本電気：INQ 概説書

10）Jardine D. A.：The ANSI/SPARC DBMS Model, North-Holland Publish-

ing Company, (1976)。
11) 日本電気：ADBS 文法説明書
12) 日本電気：INQ 文法説明書
13) 日本電気：INQ エンドユーザ言語(EQL)説明書
14) 蛋白質構造データベース利用説明書，文部省特定研究，「情報システムの形成過程と学術情報の組織化」C－8班，出版予定。