データベース統合支援システムAdbis(1) : Adbisの概要とDMPの仕様

松尾, 文碩
九州大学大型計算機センター研究開発部

二村, 祥一
九州大学大型計算機センター研究開発部

高木, 利久
九州大学大型計算機センター研究開発部

https://doi.org/10.15017/1474951

出版情報：九州大学大型計算機センター広報. 14 (2), pp.172-217, 1981-06-20. 九州大学大型計算機センター
バージョン：
権利関係：

データベース統合支援システム Adbis(1)
−Adbis の概要とDMP の仕様−

松尾　文碩,* 二村　祥一,* 高木　利久*

1. はじめに

データベース統合支援システムAdbis(A data base integration support)は，筆者らによって開発中の研究者用データベース管理システムである．研究者用として，Adbis の設計には次の２点に重点を置いた．

(1) 観測データを解析するシステムの開発に適する；

(2) 職業的計算機従事者でない研究者が比較的手軽に使うことができる．

Adbis は，勿論，観測データだけではなく一般の数値データベース用として使うことができる．しかし，Adbis には文字列を分解する機能がなく，文献検索などには使用できない．

Adbis は，3節で述べるように次の３つのモジュールから構成される．

DMP(Data Base Manipulation Primitives) ……データベース操作機能

HSI(Horn Set Interpreter) ……論理機能

NLC(Natural Language Converter) ……自然言語通信機能

DMP はデータベース管理システムの必須機能で，単独で使用することができる．他の２つのモジュールの使用は選択的であるが，HSI はDMP の機能を補い，データ解析のための強力な手段を提供する．このうち，DMP は第１版が完成し，既にセンターで使用できる．HSI は今年（１９８１年）秋には第１版を利用者に公開する予定である．そこで何回かに分けてAdbis の使用法を解説する．今回は，Adbis の概要（３，４節）と，DMP の仕様（５，６節）を中心に述べる．

2. データベースシステム

データベースの概念は，計算機システムに関する他の多くの概念同様，ファジィであるため簡潔な言葉で明確に説明できない．

１９５０年代中期までは，機械可読大量データの媒体はカード又は磁気テープであり，データへのアクセスには人手の介入を必要とした．ディスクの開発によって，１９５０年代後半に直接アクセス可能大量データの計算機システムへの常駐化，すなわちオンラインファイルの利用が可能になった．オンラインファイルを扱うための共通モジュールBDAM，ISAM[１]などが開発され，データのアクセスが容易になった．しかし，アプリケーションプログラムは，情報をBDAM，ISAMのレコード単位にしか扱うことができない．これは，基本的にはファイルの読み書きの単位である．必要な情報を取り出すにはその情報を含むレコードを探し出し，レコード形式をもとに情報を抽出する必要がある．つまり，アプリケーションプログラムとファイルのデータ構造とは密着している．計算機業務単位にアプリケーションとファイルの組が作られると，同じ情報が異る形式であちこちのフ

＊九州大学大型計算機センター研究開発部

ファイルに散在することになる．こうなると，ファイルの冗長さもさることながら，あるファイルではデータが更新され，別のファイルは未更新のままといった事態が発生しがちである．1960年代末から1970年代初頭にかけて，統合されたデータベース（integrated data base*）の考えが現われた．これはアプリケーションプログラムごとに分散，重複していたデータを統合し，それらを個々のアプリケーションプログラムから独立したデータの集合体，すなわちデータベースとして構築するものである．データベースと（複数の）アプリケーションプログラムとの仲立ちをするのが，データベース管理システムDBMS（Data Base Management System）である．

DBMSは，複数のアプリケーションプログラムにデータベースを共有させる機能によってディスクスペースの無駄をなくし，データ更新による不整合（inconsistency）の問題を解消するだけでなく，データの機密保護（security）やデータ破壊からの保護（protection），復旧（recovery）の機能を強化している．これらはデータの集中管理により強化が可能になったのである．更に重要なことは，DBMSの傾向として，アプリケーションプログラムに対してデータベースの物理的構造を隠してしまう抽象化（abstraction）への志向がある．物理的データベースは，実世界（real world）の反映である．実世界の構成要素（entity）とその属性（attribute）値，また構成要素間の関連（relationship）などの概念レベルの構造が符号化されたビット列，ポインタなどの物理的レベルの構造となりデータベースとして実現されているのである．一方，概念レベルの構造は，物理的レベルの構造の抽象化とみることができる．DBMSがアプリケーションプログラムに示すのは，データベースの概念レベルの構造である．概念レベル構造のモデル，すなわちデータモデル（data model）によってDBMSは次の3種類に大別される．

i）階層モデル（hierarchical model）
ii）ネットワークモデル（network model）
iii）関係モデル（relational model）

更に，DBMSはアプリケーションプログラム別にデータベースのそれぞれ異る見方（view, subscheme）を与える．大部分のDBMSでは，データモデルとviewの間の抽象化にはそれほど差はなく，viewは（概念的）データベースの一部分となっていることが多い．

現在，非常に多数のDBMSが開発されている．階層モデルのDBMSとしてIBMのIMS（Information Management System），ネットワークモデルのものはCODASYLのDBTG（Data Base Task Group）の提案が有名である．関係モデルのDBMSは，まだ研究段階にあり，実用化されているものは多くないが，研究・試作は活発である．さて，DBMSの一般論についてはこのくらいにしておく．DBMSについて興味のある読者は，成書が出版されているのでそれらをお読み頂きたい（例えば，文献［2，3，4，5］）．

データベースシステムは，企業などの組織体における大量データの統合管理の線に沿って発達してきた．そこでは，データベース化による効果（最終的には経済的効果）が大きいため，現在，企業体の計算機システムではデータベースは完全に定着している．従って，汎用計算機システムで

---

*データベースの英語の綴りは‘database’,‘data-base’,‘data base’と3種類あり，統一されていない．

DBMSを持たないものはなく，富士通でもMシリーズ計算機用にAIM(Advanced Information Manager)というDBMSを提供している．これは，DBTG型のネットワークモデルのDBMSであるが，IBMのIMSとの互換性を有し階層型でもある．だが，残念なことに，AIMは本センターの利用者が気軽に使えるようなシステムではない．まず，AIMは不特定多数の利用者によって使うような構造になっていない；マニュアルは１０冊あり，総ページ数は3,000ページを超える；アプリケーションプログラムは，本センターの大多数の利用者にはなじみ薄いCOBOL，PL/I，アセンブラーでしか作れない；会話型システムは簡単には作れない；等々の理由によりセンター利用者がAIMを使うことはまず絶望的である．

しかし，AIMに限らず手軽に使える研究者用DBMSは現在全くないというのが通説になっている．Adbisは，研究者向DBMSとして，我々が昨年（１９８０年）から開発を始め，その基本部分であるDMPの第１版が完成しセンター利用者の使用に供することができるようになった．

3. Adbisの概要

3.1 Adbisの設計方針

Adbisの開発目的は，次の条件を満たすDBMS作成にあった．

ｉ）観測データなどを解析するアプリケーションプログラム開発に適した研究者向システム；

ii）非識業的プログラマ，特にセンター利用者が比較的手軽に使用できるシステム；

iii）本センターのような共同利用センターの利用形態に適合したシステム．

２節でみたように即存のDBMSは，ビジネス用であり，大量データの統合管理による利点を生すために多くの機能を有し，そのため‘重い’システムになっている．その重さが負担になっているのだから，Adbisは‘軽い’システムでなければならない．又，我々の開発能力からいっても‘軽い’システムしか作れない．そこで，Adbisでは機能を重点的に絞らざるをえない．DBMSの持つ機能のうち，データ解析プログラムを作ろうとする研究者にとって一番有用なものは，データ構造の抽象化ではあるまいか．これによって，データ操作のプログラミングが非常に楽になるからである．Adbisでは，その点に重点をおき，データモデルは研究者に理解しやすいものであるようにした．

Adbisでは，実世界の構成要素とその属性および要素間の関連を，研究者にとってなじみ深い概念である関数（function），ベクトル関数（vector function），関係（relation）によって表現する．その意味では，Adbisは関係モデルのDBMSとして分類される．Adbisのデータモデルについては**4.1**節で述べる．科学的観測データ処理の究極の目的は，法則の帰納にあると考えられる．例えば，ある仮定を立て，観測データについて実際それが成立しているかどうかを調べるプログラムやある条件を満たすデータを検索するプログラムは度々書かれることだろう．Adbisでは，これらを調べるのに，いちいちFortranなどでプログラミングせずに済むように，ホーン集合（Horn Set）とよばれるある形の論理式の集合についての反証（refutation）機能によってこれらを実現できるようにした．すなわち，プログラムを書く代りに，仮定や検索条件を記述した論理式の集合を与えればよい．ホーン集合の能力は第１階述語論理（first order predicate calculus）より弱い．しかし，ホーン集合は非常に簡単であるので数理論理学（mathematical logic）を知

らない人でも，少し，勉強すれば使えるようになる．ホーン集合の反証は第1階の述語論理式の反証よりはるかに高速に実行でき，更に，プログラムによって指定できる条件であればホーン集合によって記述できることが保証されている．* もっとも高速といってもFortranでプログラムした場合に比べると遅い．結局，速度の問題は，プログラムを組み，管理する手間と，ホーン集合による条件設定の容易さ，読み易さなどのトレードオフになる．更に，ホーン集合によって，データベースに実在しない関係，関数，ベクトル関数を仮想的に造り出すことができる．これは，ディスクスペースを節約するばかりでなく，データベース管理者は関係型データベースにおける正規形（normal forms）を全く意識しないで済ますことができる．正規形の概念は，関係型データベース理解の一つの支障になっている．又，ホーン集合は，データベースの整合性（consistency），保全性（integrity）の検査に使用することができる．

一方，幾つかの機能はAdbisでは，少くともAdbis第1版では割愛した．アプリケーションプログラムごとにviewあるいはsubschemeを与える機能をAdbisは持たない．機密保護の機能は，通常のDBMSのようにきめ細くない．又，データ破壊に対する保護，復旧機能はAdbis第1版には全く無い．

Adbisにおけるアプリケーションプログラム作成言語，すなわちホスト言語は，Fortranである．Fortranとした理由は，次のとおりである．

i）大型計算機センターにおけるプログラム言語の使用率は95％以上がFortranである；

ii）観測データ処理用アプリケーションプログラムでは，数学ルーチン，統計ルーチン，図形表示ルーチンの使用は必須である．Fortranをホスト言語とすると，過去に蓄積されたこれらのソフトウェア資源の活用が容易である．

iii）ANSI 3.9—1978規格FORTRAN（通称，Fortran77）では，従来のFortranの欠点はある程度改善され，実用的にはFortran77はPL/I，Pascalより使い易い．

Fortranがホスト言語といっても，我々はFortranの仕様を変えることができないので，Adbisのインターフェースは Fortranのサブルーチン形式になっている．通常，一つのDBMSは，TSSのような感じで計算機システムに唯一つ存在し，データベースのためのデータセットを集中管理している．従って，共用ボリューム上の利用者保存データセットとデータベース用データセットとは全く異る管理方式になる．このことは利用者にとってもセンターにとっても好ましい状況ではない．Adbisは，通常の保存データセットによってデータベースを構築できるように設計した．このため，Adbisは，計算機に一つ存在するシステムではなくプログラムライブラリの構成とした．従って，Adbisは利用者が観測データをデータベースとして統合し，問合せシステム（query system）やデータ解析システム（data analyzer）を作成することを支援するためのシステムと考えることができる．このようにAdbisはオペレーティングシステムとは密着していないため，‘軽く’，手軽に使える．しかし，反面，データセットの管理面からは弱いものになっている．Adbisが管理しているデータセットは，Adbisを経由せずに，TSSコマンドによって更

＊ホーン集合の反証を手続き的に解釈するPROLOG（PROgramming language based on LOGic）というプログラム言語が作られている．

新，消去，改名ができ，これによってデータベースとしての統一性は簡単に壊れてしまうのである．

3.2 Adbisの構成

Adbisは，下記の3つのモジュールから構成される．

（M1） Data Manipulation Primitives(DMP)

（M2） Horn Set Interpreter(HSI)

（M3） Natural Language Converter(NLC)

3.1節で述べたように，（M1）～（M3）はそれぞれFortranサブルーチンライブラリの形式である．DMPは，データベースの構築，消去；レコードの追加，更新；データベースの機密保護などDBMSの必須機能を与える．HSIは，導出原理（resolution principle）によってホーン集合の反証（refutation）を行う．これによって，データベースの検索，仮定の検証，仮想関係の定義，データベースの整合性・保全性の検査などを強力に行うことができる．NLCの機能は自然言語風の文とホーン集合の相互翻訳を行うことであり，この機能によって問合せシステム又はデータ解析システムのマン・マシン・インターフェイスを自然言語風にすることが可能である．DMP，HSI，NLCの相互関連を図3．1に示す．これからわかるようにDMPは単独で使用できる．他の2つは単独では使用できない．DMPは，既に第1版が完成し，HSIは今年（1981年）秋に完成予定であるが，NLCは，研究途中にあり，完成の目途は立っていない．HSIについては，その仕様とホーン集合を続稿で詳述する．

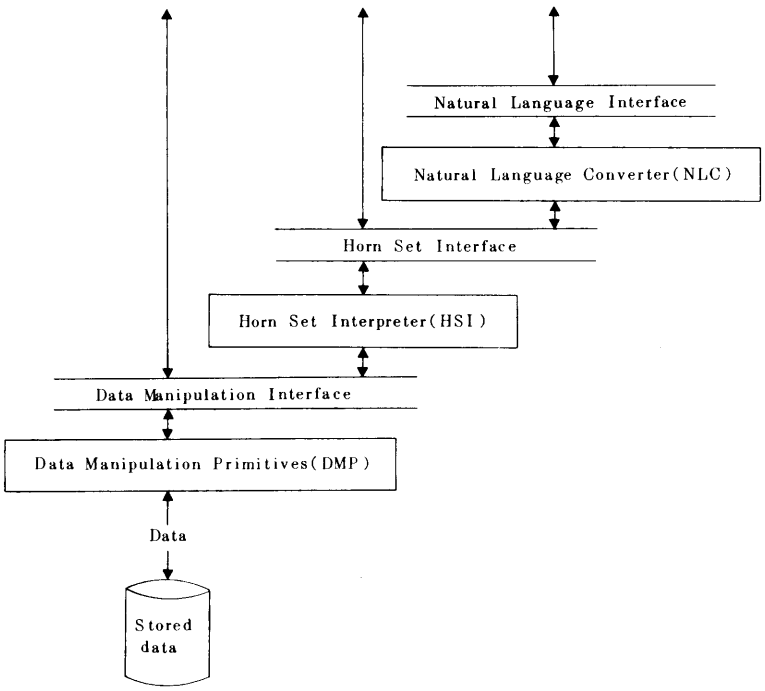


図3.1 Adbisの構成

3.3 Adbisの使用法

3.1節で述べたように，AdbisはFortranサブルーチンライブラリであり，他のFortranサブルーチンパッケージと同じように使用することができる．FORTRAN GE/HE又はFORTRAN 77いずれのFortranプロセッサからもAdbisサブルーチンを呼ぶことができる．

Adbisは，端末とのデータ通信管理機能を持たない．TSSのもとで動作する会話型システムをFortranで作る場合には，端末との通信部分にTAC/LIB[6]の使用をお勧めする．TAC/LIBは，端末との通信用サービスルーチンばかりでなく，システムプログラム開発用のサービスルーチンを多数持っている．

図3.2にアプリケーションプログラムの作成環境を示す．

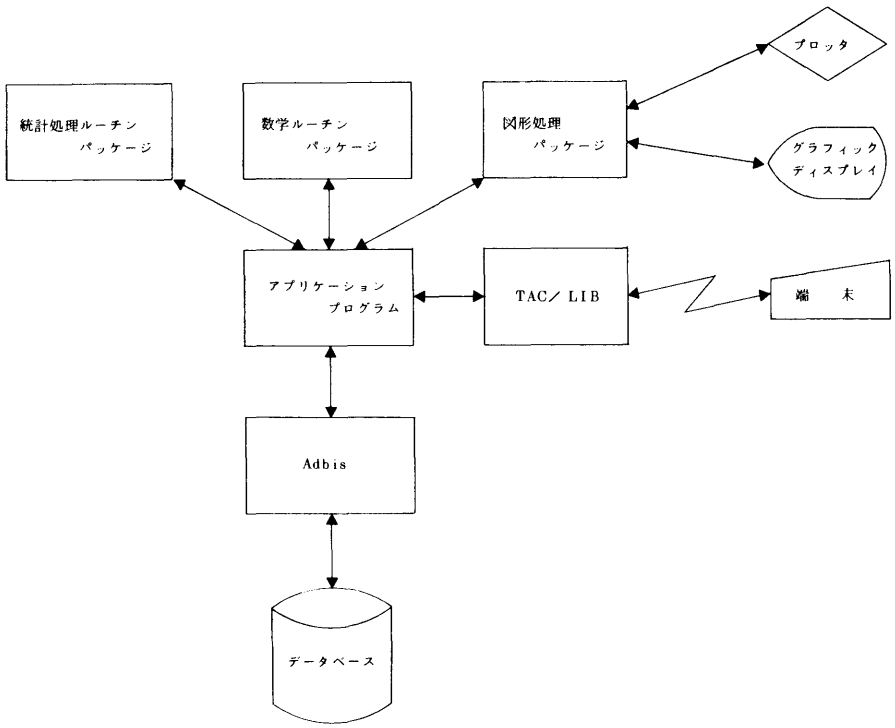


図3.2 アプリケーションプログラムの作成環境

4. Adbis/DMPの概要

4.1 データモデル

Adbis/DMPのデータモデルは，非常に単純である．Adbis/DMPでは，データの構成要素は次の3種類のものである．

1）関係（relation）

2）関数（function）

3）ベクトル関数（vector function）

これらは，いずれも大多数の研究者にとってなじみ深い概念である．

4.1.1 関　係

領域（domain）とは値の集合である．領域D_1, D_2, …, D_nの直積（Cartesian product）は$D_1 \times D_2 \times \cdots \times D_n$と書かれ，それはn－組（n－tuple）($v_1$, v_2, …, v_n）のすべての集合である．ここで，v_1, v_2, …, v_nはそれぞれD_1, D_2, …, D_nの要素である．関係（relation）とは直積$D_1 \times D_2 \times \cdots \times D_n$の部分集合のことである．

関係型データベースでは，関係を表の形式でとらえる．いま，表4.1に示した米陸軍第２０航空軍の出撃表を考えてみる．第２０航空軍は，Ｂ－２９を使って１９４４年６月５日から１９４５年８月１５日まで３８０回の出撃を行った．表4.1は，その最初の７回だけを示している．表4.1は，それ自身関係の表形式表現であるとみることもできる．すなわち，日付，目標，出撃機数，目標爆撃機数，損失，投下爆弾量の６つの領域の直積の部分集合である．そのうち，後半の４つの領域は単純

表4.1　米陸軍第２０航空軍出撃表

日　付	目　標	出撃機数	目標爆撃機数	損失	投下爆弾量(トン)
06／05／44	バンコック鉄道工場等	98	74	5	353
06／15—16／44	八幡・日本製鉄所	68	47	5	107
07／7—8／44	佐世保，大村，戸畑，八幡	18	14	0	31
07／29／44	鞍山製鉄所	96	75	3	140
08／10—11／44	パレンバン製油施設	54	39	—	39
08／10—11／44	長崎工場地帯	29	24	1	70
08／20／44	八幡・日本製鉄所	88	72	14	112

な数値の集合であり表現上の問題はない．１９４４年８月１０～１１日のスマトラ島パレンバン爆撃の損失のような欠損値（missing value）を指定する機能はDMPにはないのでアプリケーションプログラム側で管理しなければならない．まず，日付の表現には２つの問題がある．表１の表現は，‘月／日／年’形式の３個の数字の組である．そこで，日付を，大小比較ができるようにするためには年，月，日の組として表現するか，ある日（例えば，１９００年１月１日）を起点にした日数を１個の数値で示すか，あるいは別の表現法をとるかを決めなければならない．しかし，これは大したことではない．第２の問題は，夜間爆撃の場合は出撃日が２日にまたがり，複数の値が入っていることである．この問題は目標についても生じている．第３回目の出撃は目標が複数である．日付にしても目標にしても，これらを複数の値とみなさないのであれば表4.1はそのままでよい．検索や解析処理のためには，目標は複数の値とした方が都合がよいだろう．更に，それを地名と‘工場’‘都市’，‘軍事施設’などのように対象とに分けた方がよいだろう．関係型データベースでは，複数の値を持つものは別の組としてみる．すると，組は違っても同一出撃であることを示す値が必要になる．こうして，表4.1から表4.2の関係表が作られる．表4.2では，表4.1の出撃機数から投下爆弾量までを省略している．更に，４回目以降の出撃も省いている．日付は，第１

表4.2　表4.1を正規化した表（第1正規形）

MISSION_LIST_20AF

MISSION_NO	DATE	TARGET_PLACE	TARGET_KIND
1	0	BANGKOK	FACTORY
2	10.5	YAHATA	FACTORT
3	32.5	SASEBO	MILITARY_BASE
3	32.5	OMURA	MILITARY_BASE
3	32.5	TOBATA	FACTORY
3	32.5	YAHATA	FACTORY

回出撃日を起点として日で表わしている．2日にまたがる出撃日を目標のように複数の組にすると関係表が冗長になるので，例えば第10日から第11日にかけての出撃を10.5としている．表4.2のように複数の値をまとめないで別の組とした形式を関係型データベースでは，第1正規形と呼ぶ．

関係モデルとは，データベースを表4.2に示したような表の集合としてとらえるものである．各表は，MISSION_LIST_20AFのような関係名を持つ．表4.2のMISSION_NO，DATE，TARGET_PLACE，TARGET_KINDは，領域の名前であるというよりその領域の上の変数の名前*とみなす．この点はDMPも同じである．表4.2では，説明の都合上，関係名，変数名は，PL/Iの名標の形式としたが，Adbis／DMPでは，それらは先頭が英字の8文字以内の英数字列（単純名と呼ぶ；5節参照）である．

DMPでは，領域間が互いに独立な場合にのみ関係とする．例えば，表4.1の領域間には何か従属関係があるのかもしれない．しかし，それは戦況，物資の補給状況，天候等が複雑に絡み合っているだろう．データベースを解析しようとする人に従属関係があらかじめわからない場合を独立とする．従属関係が最初からはっきりしていて，それを利用してアプリケーションプログラムを作るのであれば，関係として定義せずに関数もしくはベクトル関数として扱った方がよい．

4.1.2 関　数

関数とは，次のような関数fが存在する関係$R \subseteq D_1 \times \cdots \times D_m \times D_{m+1}$のことである．

$f : D_1 \times \cdots \times D_m \rightarrow D_{m+1}$.

すなわち，Rの要素であるm＋1組は，$(v_1, \cdots, v_m, f(v_1, \cdots, v_m))$の形になっている．ここで，$v_i \in D_i$ $(i = 1, \cdots, m)$で，$f(v_1, \cdots, v_m)$は，fの関係値であってD_{m+1}の要素である．関数の例としては，表4.3に示すような父親を示す関係は関数とみることができる．

表4.3　関数の例

FATHER_OF

PERSONNEL	PATHER_OF
Ares	Zeus
Harmonia	Ares
⋮	⋮

＊ 属性（attribute）名と呼ばれる．機能は，領域上の変数である．

4.1.3 ベクトル関数

ベクトル関数とは，次のような関数f_1，f_2，…，f_nが存在する関係$R \subseteq D_1 \times \cdots \times D_m \times D_{m+1} \times \cdots \times D_{m+n}$のことである．

$$f_1 : D_1 \times \cdots \times D_m \rightarrow D_{m+1},$$
$$f_2 : D_1 \times \cdots \times D_m \rightarrow D_{m+2},$$
$$\vdots$$
$$f_n : D_1 \times \cdots \times D_m \rightarrow D_{m+n}.$$

すなわち，Rの要素であるm＋n組は（v_1，…，v_m，f_1（v_1，…，v_m），…，f_n（v_1，…，v_m））の形をしている．ここで，$v_i \in D_i$（i＝1，…，m）で，f_j（v_1，…，v_m）（j＝1，…，n）はf_jの関数値であって，D_{m+j}の要素である．ベクトル関数の例として表4.4に示すような人に関する関係がある．ここでは，AGE，SEX，PLACEはPERSONNELの関数と考えられている．

関係型データベースでは，関数，ベクトル関数の表をBoyce—Coddの正規形とよぶ．

表4.4 ベクトル関数の例

PERSONNEL	AGE	SEX	PLACE
Taro	18	M	Tokyo
Jack	21	M	New York
Mary	15	F	Londou
⋮	⋮	⋮	

4.2 Adbis／DMPのデータベース構造

Adbis／DMPは，データベースを関係，関数，ベクトル関数の集合として抽象化する．DMPは各関係，関数，ベクトル関数ごとに表4.2，表4.3，表4.4に対応した表形式（tabular form）のファイルを作る．表形式では，数値データは固定／浮動小数点の2進数で，文字データは4バイトの符号化されたデータとして入れられる．各関係，（ベクトル）関数ごとに領域値から表形式の組を検索するための索引である転置形（inverted form）のファイルが作られる．関係と（ベクトル）関数の違いは，関係ではすべての領域に対して転置形が作られるのに対し，（ベクトル）関数では，特に指定しない限り関数値について転置形は作られない．つまり逆関数としても使うことを宣言したときだけ関数値の転置形が作られる．DMPは，一つの関係，関数，ベクトル関数ごとに一つのデータセットを作る．これを**データファイル**とよぶ．

DMPは，一つのデータベースに対して一つの**マスタファイル**を作る．マスタファイルには，各々の関係，（ベクトル）関数についての表形式，転置形をデータベースとして統合するための情報をもっている．データベース管理者が，データベースを定義するには，Adbis／DMP－DDL（Data Definition Language）を使って行う．DMP—DDLによる定義を書いたファイル（データベース定義ファイル）をTSSのエディタなどで作っておき，DMPのサブルーチンDEFD¥D

（6．2．1節）を呼び出すことによってマスターファイルが作られる．DMP—DDLについては5節で述べる．

Adbis／DMPのデータベースの構造を図4.1に示す．関係，関数，ベクトル関数の表形式，転置形を入れたデータファイルは，マスタファイルを作った後で，CREA¥D（6．2．5節）を使って作る．DMP第1版では，マスタファイルの更新機能はないので，マスタファイルの内容を変えるには，DEFD¥Dによって新たに作り直さなければならない．このとき，既存のデータファイルの構造を変えないような変更であれば，データファイルを作り直す必要はない．

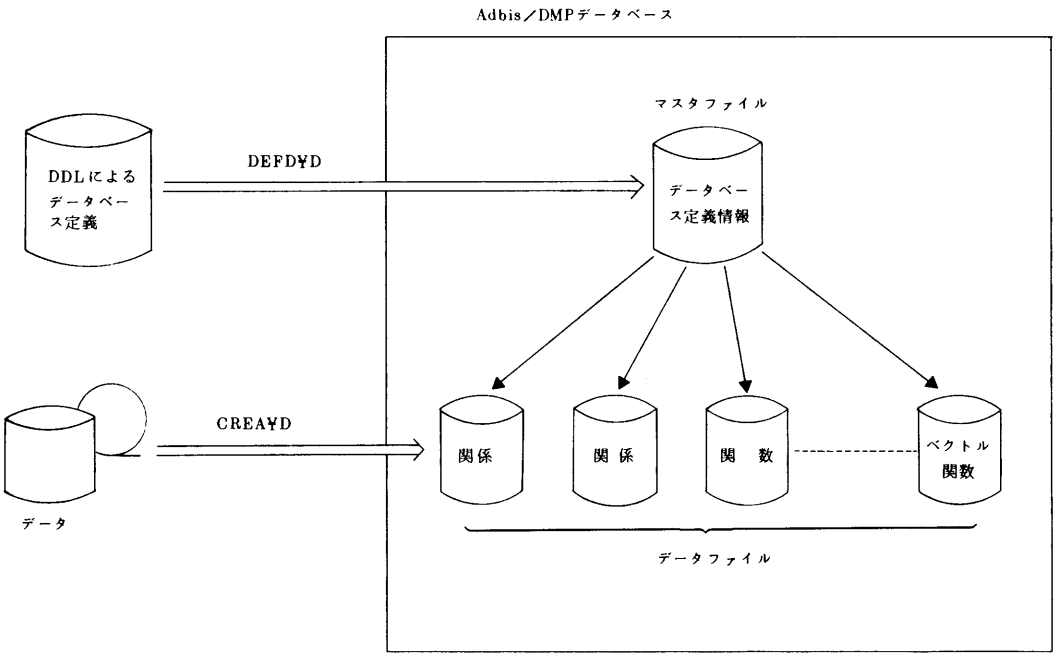


図4.1　Adbis／DMPのデータベース構成

4.3　データファイル

データファイルは，関係，関数またはベクトル関数の表形式と転置形から構成されている．表形式および転置形の構造は，関係，関数，ベクトル関数による差はない．関数，ベクトル関数は，それぞれ4.1.2節，4.1.3節で述べたような関係とみなされる．表形式のレコードは，関係の組である．組を一意的に識別する変数の組を**関係（関数，ベクトル関数）のキー**と呼ぶ．Adbis／DMPでは，関係のキーは，関係を構成する全変数の組であり，関数，ベクトル関数のキーは，引数を構成する変数の組である．表形式の**レコードのキー**は，レコードを一意的に識別するためDMPが作り出す．レコードのキーは，勿論，関係のキーに1－1対応しているが，利用者からは見えない．転置形における組の情報は，レコードのキーであるので，データベースの検索は，まずキー値の集合（これを**キー集合**と呼ぶ）が求まる．DMPにおける論理演算は，キー集合に対する集合演算である．

4.4 機密保護

Adbis／DMPの機密保護機能は強力でない．DMPでは，機密保護はデータベース単位にパスワードが設定できるだけである．パスワードは，権限の強さによって3つのレベルがあり，強い順にマスタパスワード，ライトパスワード，リードパスワードである．リードパスワードでは，データベースの検索だけが可能である．ライトパスワードでは，検索の外，レコードの追加，消去ができる．マスタパスワードでは，すべてのことができる．

4.5 完 全 性

データベースには，関係または関数値が存在するすべての組が入っているとは限らない．この意味での関係，関数またはベクトル関数の完全性の度合は，データベース管理者だけがわかるもので，アプリケーションプログラム側で完全性の度合に応じた処理をしなければならない．Adbis／DMPでは，関係，関数またはベクトル関数単位に完全性の度合を示す整数値をつけることができる（5節のDDLのオペランドCOMPLETENESS）．勿論，整数値の意味付けはデータベース管理者が行い，完全性に応じた対応はアプリケーションプログラムが行う．

4.6 関係型データベース管理システムとしてのAdbis／DMP

Adbis/DMPの特徴は，関係から関数，ベクトル関数を分離したことである．これによる利点の一つは4.2で述べたように無駄な転置形を作らないことである．別の利点は（ベクトル）関数の表現力にある．これはHSIのレベルで活用することができる．

DMPには，通常の関係型DBMSのように表と表から一つの表を作ったり，一つの表を複数の表に分割する機能を持たない．この機能はHSIによって仮想関係を定義することによって補うことができる．仮想関係は実関係を作るのに比べ検索効率の低下はやむをえないが，反面ディスク領域の効率がよく，データベース管理が正規形について気にしなくて済むなどの利点がある．

5. Adbis／DMPのデータベース定義言語

5.1 概 要

この節では，データベースを定義するためのデータ定義言語（data definition language）Adbis／DMP—DDLの仕様を述べる．Adbis／DMP—DDLによるデータベース定義を記述したカードイメージのデータセット（データ定義ファイル）を用意し，Adbis／DMPのデータベース定義用サブルーチンDEFD¥Dを呼ぶことによってデータベースのマスタファイルが作られる．

Adbis／DMP—DDLのコマンドは，DATABASE，RELATION，FUNCTION，VECTFUNC，VARIABLE，ENDの6種である．DATABASEは，データベース名，データベース管理者名，機密保護のためのパスワードなど，データベース全体についての情報を設定する．

データベースの構成要素となる関係，関数，ベクトル関数は，それぞれRELATION，FUNCTION,VECTFUNCで定義する．RELATIONでは，関係名，関係を構成する変数，データファイルのデータセット情報を，FUNCTIONでは，関数名，関数の領域を構成する変数，値域の型，データセット情報，逆関数定義の有無などを，VECTFUNCでは，ベクトル関数名，関数の領域，値

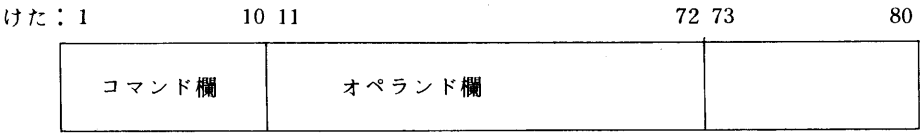
域を構成する変数，データセット情報，逆関数の有無などを指定する．変数は，RELATION，FUNCTION，VECTFUNCとは別にVARIABLEで型，領域を指定する．型は，２バイト長／４バイト長の整数（I2／I4），４バイト長／８バイト長の実数（R4／R8），及び固定長の文字列（C）の指定が可能である．ENDは，定義の終りを示すコマンドである．

5.2 Adbis／DMP−DDLの書式

ここでは，Adbis／DMP−DDL及びデータベース定義ファイルの書式仕様を述べる．

i）データベース定義ファイルは，レコード形式がFまたはFBで，レコード長が８０バイトの順データセットあるいは区分データセットのメンバでなければならない．

ii）Adbis／DMP−DDLのコマンドの書式は下記のとおりである．



・コマンド名は，第１けたから書き始めなければならない．オペランドは第１１けたから第72けたの間に書く．コマンド欄が空白の場合，前行の継続とみなされる．この場合，前行の第７２けたに継続行の第１１けたが続いているとみなす．

・第７３桁から第８０桁までは，任意の文字を書いてよい．

・第１桁に▼＊▼を持った行は，注釈行となる．

iii）オペランドはキーワードパラメータ方式であり，オペランドの指定順序は任意である．オペランドのキーは省略形による指定が可能である．省略形の規約は，IBM OS／VS TSO又は，FACOM OSIV／F4 TSSのコマンドのそれと同じである．すなわち，他のキーと区別可能であれば，任意の先頭部分がキーの省略形となりうる．

iv）データの型は，I2，I4，R4，R8，Ckのいずれかを指定する．I2，I4，R4，R8，Ckはそれぞれ２バイト長の整数，４バイト長の整数，４バイト長の実数，８バイト長の実数，kバイト長の文字列を表す．kは３桁以内の整数である．

5.3 データベース定義用コマンド

ここでは，Adbis／DMPのデータベース定義用の各コマンドについて説明する．コマンドの記述には，次の規約に従う．

1. ［　］で囲まれた部分は省略可能なことを示す．下線部は，省略時に採用される値を示す．
2. {　} で囲まれた部分はいずれか１つを選択すべきことを示す．
3. 英大文字，'＝'，','，'･'，'('，')' 及び '▼'はそのとおり指定すべきことを示す．
4. カナ，漢字，英小文字で表わされたものは，適宜記入すべきものであることを示す．
5. …は，必要ならば直前のものを並べて記入してもよいことを示す．

5.3.1 DATABASE

(1) 機　能

データベース名，データベース管理者名，パスワードなど，データベース全体を記述する．

(2) 記述形式

コマンド	オ　ペ　ラ　ン　ド
DATABASE	NAME（データベース名） ADMINISTRATOR（データベース管理者課題名） ［EXPLANATION（▼データベースの説明▼）］ MASTERPASSWORD（マスタパスワード） WRITEPASSWORD（ライトパスワード） ［READPASSWORD（リードパスワード）］

(3) オペランドの説明

a. NAME（データベース名）

データベース名を指定する．データベース名は，8文字以内の英数字で，先頭は英字（以後，単純名と呼ぶ）でなければならない．

b. ADMINISTRATOR（データベース管理者課題名）

データベース管理者の課題名Fnnnn（F：固定文字，nnnn：課題番号）を指定する．

c. EXPLANATION（▼データベースの説明▼）

データベースの説明を100文字以内の文字列で与える．

d. MASTERPASSWORD（マスタパスワード）

マスタパスワードを単純名で指定する．マスタパスワードは，データベース管理者用のパスワードで，データベースの創成を行うときに必要なパスワードである．マスタパスワードによりデータの追加，変更，削除，検索も可能である．

e. WRITEPASSWORD（ライトパスワード）

ライトパスワードを単純名で指定する．ライトパスワードは，データベースへのデータの追加，変更，削除を行うときに必要なパスワードである．ライトパスワードにより検索も可能である．

f. READPASSWORD（リードパスワード）

リードパスワードを単純名で指定する．リードパスワードは，データベースの検索時に必要なパスワードである．

(4) 使用上の注意

DATABASEコマンドは，データベース定義の最初のコマンドでなければならない．

5.3.2 RELATION

(1) 機　能

データベースの構成要素である関係を定義する．関係名，別名，関係を構成する変数，使用デ

ィスクなどを指定できる.

(2) 記述形式

コマンド	オペランド
RELATION	NAME(関係名) [ALIAS(別名)] [COMPLETENESS(データの完全性)] VARIABLE(変数名[, 変数名]…) [EXPLANATION(▼説明▼)] [SPACE(スペース量)] [UNIT(装置指定名)] [VOLUME(ボリューム通し番号)]

(3) オペランドの説明

a. NAME(関係名)

関係名を単純名で指定する.

b. ALIAS(別名)

関係の別名を単純名で指定する.

c. COMPLETENESS(データの完全性)

データの完全性を整数値で指定する. 省略値は0である.

d. VARIABLE(変数名[, 変数名]…)

関係を構成する変数の名前を指定する. 変数は, VARIABLEコマンドで定義しなければならない.

e. EXPLANATION(▼説明▼)

関係の説明を100文字以内の文字列で与える.

f. SPACE(スペース量)

確保するスペース量をキロバイト単位で指定する. 省略時は, 入力データの大きさからシステムが適当な値を自動的に割り当てる.

g. UNIT(装置指定名)

スペースを確保する装置を指定する. 省略時は▼PUB▼となる.

h. VOLUME(ボリューム通し番号)

スペースを確保するボリュームのボリューム通し番号を指定する. 省略時は, システムが適当に割り当てる.

5.3.3 FUNCTION

(1) 機 能

データベースの構成要素である関数を定義する. 関数名, 別名, 関数の領域, 値域, 使用ディ

スクなどを指定できる.

(2) 記述形式

コマンド	オペランド
FUNCTION	NAME(関数名) [ALIAS(別名)] [COMPLETENESS(データの完全性)] VARIABLE(変数名[,変数名]…) [EXPLANATION(▼説明▼)] [SPACE(スペース量)] [UNIT(装置指定名)] [VOLUME(ボリューム通し番号)] TYPE(関数値の型) [RANGE($\left\{\begin{matrix} a_1 \\ /a_1, b_1/ \end{matrix}\right\}\left[,\left\{\begin{matrix} a_2 \\ /a_2, b_2/ \end{matrix}\right\}\right]$ …)] [INVERSE]

(3) オペランドの説明

a. NAME(関数名)からh. VOLUME(ボリューム通し番号)までは, RELATIONと同じなので, これらについては, 5.3.2節RELATIONのオペランドの説明を参照されたい.

i. TYPE(関数値の型)

関数値の型をI2, I4, R4, R8, Ckから選んで指定する.

j. RANGE($\left\{\begin{matrix} a_1 \\ /a_1, b_1/ \end{matrix}\right\}, \left[\left\{\begin{matrix} a_2 \\ /a_2, b_2/ \end{matrix}\right\}\right]$ …)

関数の値域を指定する. 例えば, RANGE(10,/50,60/)は, 10, あるいは50から60までの値をとることを示し, RANGE(▼RED▼, ▼BLUE▼, ▼YELLOW▼)は文字型の▼RED▼ ▼BLUE▼, ▼YELLOW▼だけを値としてとることを示す. RANGEで指定する値は, TYPEで宣言した型を持たねばならない.

k. INVERSE

逆関数を定義することを指示する. 省略時は, 逆関数は定義されない.

5.3.4 VECTFUNC

(1) 機 能

データベースの構成要素であるベクトル関数を定義する. VECTFUNCでは, ベクトル関数名, 別名, ベクトル関数の領域, 値域, 使用ディスクなどを指定する.

(2) 記述形式

コマンド	オペランド
VECTFUNC	NAME（ベクトル関数名） [ALIAS（別名）] [COMPLETENESS（データの完全性）] VARIABLE（変数名[，変数名]…） [EXPLANATION(▼説明▼)] [SPACE（スペース量）] [UNIT（装置指定名）] [VOLUME（ボリューム通し番号）] VECTOR（変数名[，変数名]…） [INVERSE（変数名[，変数名]…）]

(3) オペランドの説明

a．NAME（ベクトル関数名）から，h．VOLUME（ボリューム通し番号）までは，RELATIONと同じなので，これらについては，5.3.2節RELATIONのオペランドの説明を参照のこと．

i．VECTOR（変数名[，変数名]…）

ベクトルを構成する変数の名前を指定する．変数は，VARIABLEコマンドで定義しなければならない．

j．INVERSE（変数名[，変数名]…）

VECTORで指定した変数のうち，逆関数を定義する変数の名前を指定する．

5.3.5 VARIABLE

(1) 機　能

データベース定義で用いる変数を定義する．変数名のほか，変数の型，領域を指定できる．

(2) 記述形式

コマンド	オペランド
VARIABLE	NAME（変数名） TYPE（変数の型） [DOMAIN($\left\{ \begin{matrix} a_1 \\ /a_1, b_1/ \end{matrix} \right\} \left[, \left\{ \begin{matrix} a_2 \\ /a_2, b_2/ \end{matrix} \right\} \right]$ …)] [EXPLANATION(▼説明▼)]

(3) オペランドの説明

a. NAME (変数名)

変数名を単純名で指定する.

b. TYPE (変数の型)

変数の型をI2, I4, R4, R8, Ckから選んで指定する.

c. DOMAIN($\left\{ \begin{array}{l} a_1 \\ /a_1,\ b_1/ \end{array} \right\} \left[, \left\{ \begin{array}{l} a_2 \\ /a_2,\ b_2/ \end{array} \right\} \right] \cdots$)

変数の取りうる値を指定する. 指定方法は, 5.3.3節FUNCTIONのRANGEオペランドと同様である.

d. EXPLANAION (▼説明▼)

変数の説明を100文字以内の文字列で与える.

5.2.6 END

(1) 機　　能

データベース定義の終りを示す.

(2) 記述形式

コマンド	オペランド
END	なし

5.4 データベースの定義例

Adbis/DMP—DDLを用いたデータベースの定義例を示す.

定義例)

関係MANCARS, 関数COLOR, ベクトル関数 MANUFACTからなるデータベースDBTESTを定義する.

```
①    * THIS IS AN EXAMPLE OF DATABASE DEFINITION.
②    DATABASE  NAME(DBTEST)
               ADMINISTRATOR(F0001)
               EXPLANATION('MANUFACTURERS-CARS DATABASE')
               READPASSWORD(RAED)
               WRITEPASSWORD(WIERT)
               MASTERPASSWORD(MSEATR)

③    RELATION  NAME(MANUCARS)
               ALIAS(MC)
               VARIABLE(MNO,CNO)
               COMPLETENESS(10)
               EXPLANATION('JANUARY,1978--->DECEMBER,1980')
               SPACE(3000)

④    FUNCTION  NAME(COLOR)
               VARIABLE(CNO)
               TYPE(C6)
               RANGE('RED','BLUE','WHITE','BLACK')
               INVERSE
```

```
⑤     VECTFUNC  NAME(MANUFACT)
                ALIAS(M)
                VARIABLE(MNO)
                VECTOR(MNAME,PRODUCT,CITY)
                INVERSE(CITY)

⑥     VARIABLE  NAME(MNO)      TYPE(I4)
⑦     VARIABLE  NAME(CNO)      TYPE(I4)  DOMAIN(50,/100,200/)
⑧     VARIABLE  NAME(MNAME)    TYPE(C4)
⑨     VARIABLE  NAME(PRODUCT)   TYPE(R8)
⑩     VARIABLE  NAME(CITY)     TYPE(C10) DOMAIN('DETROIT','PARIS','TOYOTA')
⑪     END
```

定義例の説明）

① 第1桁が▼＊▼である注釈行である.

② DATABASEコマンドにより，データベースDBTESTを定義している．コマンド名は，第1けたから書き，オペランドは第11けたから第72けたの間に書く．ここでは，データベース名DBTEST，データベース管理者の課題番号F0001，データベースの説明▼MANUFACTURERS—CARS DATABASE▼，リードパスワードRAED，ライトパスワードWIERT，マスタパスワードMSEATRを指定している.

③ RELATIONコマンドにより，関係MANUCARSを定義している．MANUCARSは⑥，⑦で定義されている変数の組であることがVARIABLEオペランドで指定されている．又，データファイルのディスクスペースとして3000キロバイト確保することを指定している．このほか，別名MC，データの完全性10，関係の説明▼JANUARY，1978 …＞DECEMBER，1980▼を指定している.

④ FUNCTIONコマンドにより，関数COLORを定義している．関数の引数は，CNOであり，関数値は，6バイトの文字型で，▼RED▼，▼BLUE▼，▼WHITE▼，▼BLACK▼を値としてとることを指定している．更に，逆関数を用意することを指示している.

⑤ VECTFUNCコマンドにより，ベクトル関数MANUFACTを定義している．ベクトル関数の引数はMNOであり，ベクトルは（MNAME，PRODUCT，CITY）であることが指定されている．そのほか，別名がMで，逆関数はCITYについて用意することが指定されている.

⑥～⑩

VARIABLEコマンドにより，変数MNO，CNO，MNAME，PRODUCT，CITYを定義している．変数MNO，CNO，MNAME，PRODUCT，CITYは，それぞれ4バイト長の整数，4バイト長の整数，4バイト長の文字列，8バイト長（倍精度）の実数，10バイト長の文字列であることが，TYPEオペランドで指定されている．⑦では変数CNOの領域は10，あるいは100から200までの値，⑩では変数CITYの領域は{▼DETROIT▼，▼PARIS▼，▼TOYOTA▼}であることが指定されている.

⑪ ENDコマンドによりデータベース定義の終りを宣言している.

5.5 データベースの再定義について

一度作られたマスタファイルを更新する機能は，DMP第1版にはない．マスタファイルを変えるには，DEFDYDを使って作り直す必要がある．その場合，下記のオペランドの変更であれば，

データファイルを作り直す必要はない.

DATABASE…EXPLANATION, MASTERPASSWORD, WRITEPASSWORD,
READPASSWORD

RELATION…ALIAS, COMPLETENESS, EXPLANATION

FUNCTION…ALIAS, COMPLETENESS, EXPLANATION

VECTFUNC…ALIAS, COMPLETENESS, EXPLANATION

VARIABLE…EXPLANATION

6. Adbis／DMPの仕様

6.1 機能概要

Adbis／DMPで使用できるFortranサブルーチンを表6.1に示す. ここでは, 各サブルーチンの機能を図6.1に添って概説する.

データベースの定義は, DEFD¥Dによる. DEFD¥Dの実行によりデータベース定義情報を含むマスタファイルが作成される. データベース使用の開始と終了は, OPEN¥DとCLOS¥Dで宣言する. OPEN¥Dでは, パスワードによる利用資格チェックが行われ, CLOS¥Dでは, データベースの利用の終了処理が行われる. これ以外のサブルーチンは, すべて, OPEN¥D, CLOS¥Dの間で用いられる.

SHOW¥Dはデータベースの操作を行うために, データベースの定義情報を得るのに用いる. データベースの創成, データの追加, 削除, 更新, 及びデータベースの消去用のコマンドとしては, CREA¥D, INSE¥D, DELE¥D(およびDELS¥D), UPDA¥D, CLEA¥Dがある. CREA¥Dにより関係, 関数, ベクトル関数単位にデータファイルが割り当てられ, そこに表形式, 転置形式のデータが登録される. 創成ずみのデータファイルに対して1レコードずつデータの追加, 削除, 更新を行うものがINSE¥D, DELE¥D, UPDA¥Dである. データの削除については, DELS¥Dを用いることにより, 特定の条件を満たす複数のレコードを同時に削除することも可能である. CLEA¥Dは不要になったデータファイルを消去するときに使う. 更に, データベース創成に分類されるものとして, INVE¥Dがある. INVE¥Dは, データベース定義

表6.1 Adbis／DMPのサブルーチン一覧

サブルーチン名	機　　能
DEFD¥D	データベース定義
OPEN¥D	使用開始
CLOS¥D	使用終了
SHOW¥D	定義情報表示
CREA¥D	データベース創成
CLEA¥D	データベース消去
VALU¥D	求値
FIND¥D	集合検索
MEET¥D	積集合演算
JOIN¥D	和集合演算
DIFF¥D	差集合演算
CARD¥D	要素数評価
EVAL¥D	集合評価
GETN¥D	次値集合検索
GETF¥D	先頭値集合検索
INSE¥D	要素追加
DELE¥D	要素削除
DELS¥D	集合削除
UPDA¥D	要素修正
INVE¥D	転置形生成
SORT¥D	分類
SAVE¥D	集合保存
UNSA¥D	保存集合消去

時には，逆関数が定義されていなかったものについて，データファイル作成後に，新たに逆関数を定義する．

データベースの検索用のコマンドとして基本的なものは，VALU¥D，FIND¥D，EVAL¥Dである．VALU¥Dは，関係に対しては，指定した値の組が関係に有るかどうかを調べ，関数，ベクトル，関数に対しては，関数の値を求める．FIND¥Dは関係，関数，ベクトル関数をいずれも表とみなして，指定した条件を満足するレコードの集合を求めるためのものである．FIND¥Dによりレコードのキーの集合が求まる．キー集合から値の集合を求めるには，EVAL¥Dを使う．

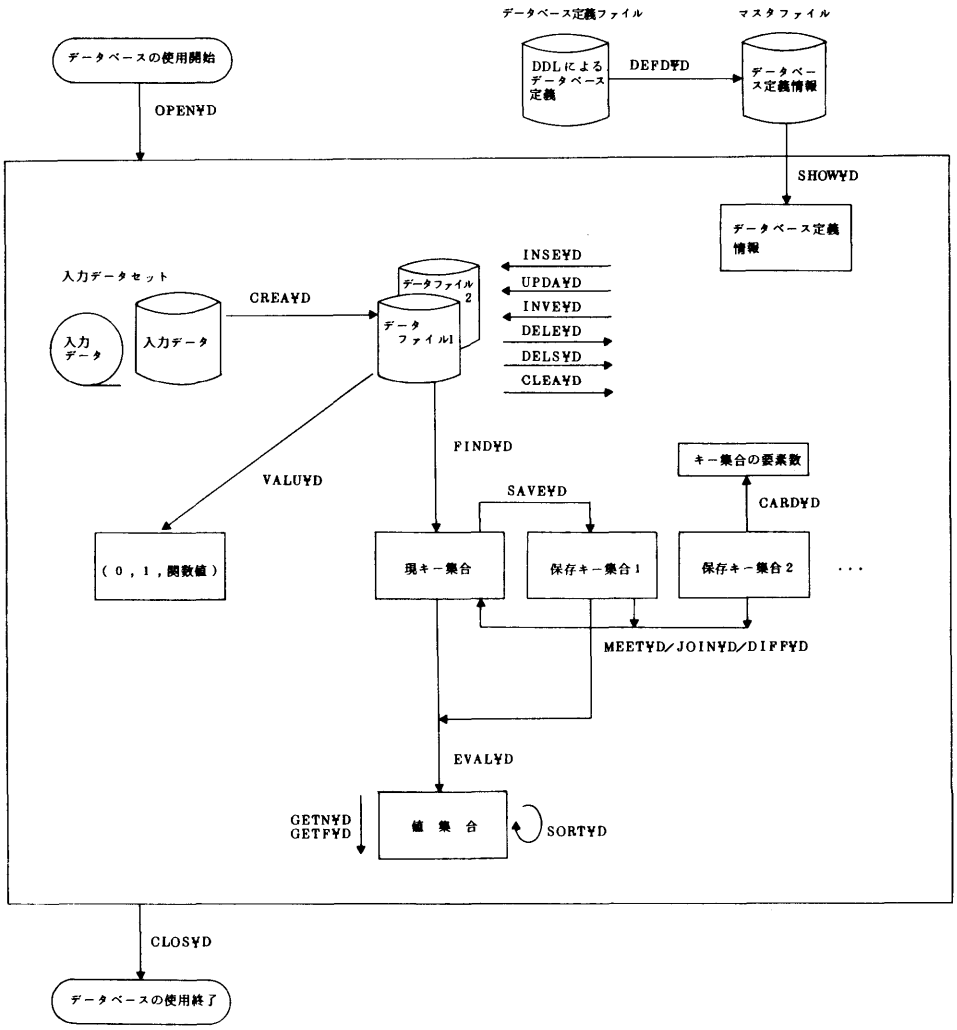


図6.1　Adbis／DMPの処理概念図

キー集合，値集合の操作のために幾つかのサブルーチンを用意した．まず，キー集合操作用として，SAVE¥D,MEET¥D,JOIN¥D,DIFF¥D,CARD¥Dがある．SAVE¥Dは，最も新たに求めたキーの集合（現キー集合と呼ぶ）を保存するものであり，MEET¥D,JOIN¥D,DIFF¥Dは，それぞれキー集合の積集合，和集合，差集合を求めるためのものである．又，キー集合の要素数は，CARD¥Dで知ることができる．次に，値集合操作用のサブルーチンとして，GETN¥D,GETF¥D,SORT¥Dがある．キー集合からEVAL¥Dによって求められた値集合は主記憶の利用者域に入れられるが，値集合が利用者域より大きい場合は，先頭部分だけが入れられる．後続部分をつぎつぎに主記憶にとり出すには，GETN¥Dを繰返し使用する．再度先頭から値集合を見たい時には，GETF¥Dを使えば，先頭部分が再び主記憶に入れられる．SORT¥Dは値集合のソーティングを行う．

6.2 Adbis／DMPサブルーチン

この節では，Adbis／DMPの各サブルーチンの使用法を述べる．使用法の説明において次の記法を使用する．なお，使用例はFortran77で書かれている．

Pn	：復帰時に内容が変化しない引数を表す．
[Pn]	：復帰時に内容が変化する引数を表す．
I	：整数を表す．
R	：実数を表す．
A	：文字列を表す．
array	：配列を表す．
v	：変数を表す．
*n	：nバイト長であることを表す．
[…]	：省略可能であることを示す．
△	：空白を表す．

6.2.1 DEFD¥D(define dd)

(1) 機　能

Adbis／DMPによるデータベース定義に基づき，データベースの定義情報を格納したマスタファイルを創成する．データベース定義は，データベース定義ファイルに，あらかじめ用意しておかなければならない．

(2) 呼び出し形式と引数の説明

```
CALL  DEFD¥D(P1 ,P2 ,P3 ,P4 )
```

[P_1] ：完了コード（Iv＊4）

0＝正常終了．

1＝データベース定義ファイル（DD名＝FT49F001）が割り当てられていない．

2＝マスタファイルのライトエラー．

3＝DDLのコマンド名がおかしい．

4＝DATABASEコマンドが最初のDDLコマンドでない.

5＝関係，関数，ベクトル関数の個数が多過ぎる.

6＝変数の個数が多過ぎる.

7＝未定義の変数が使われている.

8＝データベースがオープンされているので処理しない.

50＝その他のエラー．第2引数にコマンド名，第3引数にオペランド名，第4引数に名前が設定される.

P_2 ：コマンド名（Av＊8，Aarray＊8）

エラーを起したコマンドのコマンド名が設定される．DATABASE,RELATION,FUNCTION,VECTFUNC,VARIABLE,ENDのいずれかが設定される．なお，P_1＝3の場合コマンド名は空白となる.

P_3 ：オペランド名（Av＊16，Aarray＊16）

エラーを起したオペランドの名前が設定される．NAME,ADMINISTRATOR,EXPLANATION,MASTERPASSWORD,WRITEPASSWORD,READPASSWORD,ALIAS,COMPLETENESS,VARIABLE,SPACE,UNIT,VOLUME,TYPE,RANGE,DOMAIN,INVERSE,VECTORのいずれかが設定される.

P_4 ：名前（Av＊8，Aarray＊8）

エラーを起した個所のNAMEオペランドの値が設定される．データベース名，関係名，関数名，ベクトル関数名，変数名のいずれかが設定される.

(3) 使用上の注意

・DEFD¥Dは，他のサブルーチンと違って，OPEN¥DとCLOS¥Dの間で呼び出すものではない.

・DEFD¥Dは，データベース管理者しか呼び出せない.

・DEFD¥Dの入力データは，あらかじめ作成しておき，DEFD¥Dの呼び出し以前に，DD名FT49F001を割り当てておく必要がある.

・データベースの定義方法については，5節を参照のこと.

(4) 適　　用

データベースの定義に使用する.

(5) 使用例

・データベース定義ファイルF0001,DDL.DATAによりマスタファイルを創成する.

i）データベース定義ファイルF0001.DDL.DATAの作成.

ii）データベース定義ファイルF0001.DDL.DATAの割当て.

・JCL（バッチ処理）

```
//FT49F001 DD DSN=F0001.DDL.DATA,DISP=SHR
```

・TSSコマンド

```
ALLOC F(FT49F001) DA(DDL.DATA) SHR
```

iii）アプリケーションプログラム

```
CHARACTER CMD *8,OPER*16,NAME*8
```

CALL DEFD¥D(IRC,CMD,OPER,NAME)

6.2.2 OPEN¥D(open)

(1) 機 能

データベースのオープン処理を行う．オープン処理では，データベースの機密保護チェック，同時利用チェックおよび創成ずみのデータファイルの割当て処理などを行う．

(2) 呼び出し形式と引数の説明

CALL OPEN¥D(P_1,P_2,P_3,P_4)

P_1 ：完了コード（Iv＊4）

0＝正常終了．

1＝マスタファイルが作成されていない．

3＝指定データベースが既にオープンされているか，他のデータベースを既にオープンしている．

4＝既存のデータファイルで形式のおかしいものがある．

5＝パスワードの指定が正しくない．

6＝他の利用者によって更新中のため利用できない．

P_2 ：データベース名（A＊8，Av＊8，Aarray＊8）

データベース名を，前詰めで指定する．

P_3 ：データベース管理者課題名（A＊5，Av＊5，Aarray＊5）

データベース管理者の課題名（Fnnnn）を指定する．

P_4 ：パスワード（A＊8，Av＊8，Aarray＊8）

リードパスワード，ライトパスワード，マスタパスワードのいずれかを指定する．

(3) 使用上の注意

複数のデータベースを同時にオープンできない（P1＝3で通知）．

(4) 適 用

データベースにアクセスする時，最初に呼び出すものである．

(5) 使用例

使用例は，次のCLOS¥Dの使用例を参照のこと．

6.2.3 CLOS¥D(close)

(1) 機 能

データベースのクローズ処理を行う．クローズ処理は，データベース操作のために割当てた各種データセットの解放を行う．

(2) 呼び出し形式と引数の説明

CALL CLOS¥D(P_1,P_2)

P_1 ：完了コード（ Iv＊4 ）

0＝正常終了.

1＝第2引数で指定したデータベース名は，OPEN¥Dで指定した名前ではない.

50＝データベースがオープンされていないのにクローズしようとした.

P_2 ：データベース名（ A＊8 , Av＊8 , Aarray＊8 ）

クローズするデータベースの名前を指定する.

(3) 適 用

・データベースの処理を終了する場合に用いる.

(4) 使用例

データベース管理者F0001のデータベースDB01をパスワードREAD1でオープンする.
使用後，そのデータベースをクローズする.

```
CALL OPEN¥D(IRC,▼DB01△△△△▼,▼F0001▼,▼READ1△△△▼)
      ⋮
CALL CLOS¥D(IRC,▼DB01△△△△▼)
```

6.2.4 SHOW¥D（show dd）

(1) 機 能

データベースの定義を調べる.

(2) 呼び出し形式と引数の説明

CALL SHOW¥D(P_1 , P_2 , P_3 , P_4)

P_1 ：完了コード（ Iv＊4 ）

0＝正常終了.

1＝第2，3引数で指定した情報設定領域が不足している.

2＝第4引数で指定した名前はデータベース又は構成要素として定義されていない.

50＝データベースがオープンされていない.

P_2 ：データベース定義情報設定領域（ I＊4 array）

内容，形式は，(5)に記載.

P_3 ：第2引数の領域のバイト長（ I＊4， Iv＊4 ）

P_4 ：名前あるいは▼#ALLNAME▼（A*8 , Av*8 , Aarray*8）

名前として，データベース名，関係名，関数名，ベクトル関数名，変数名が指定できる.

データベースに登録されている名前の一覧を検索する場合，▼#ALLNAME▼を指定する.

(3) 適 用

データベース操作に先立って，データベースの定義情報を検索するのに使用する.

(4) 使用例

・データベースに登録されている名前を調べる.

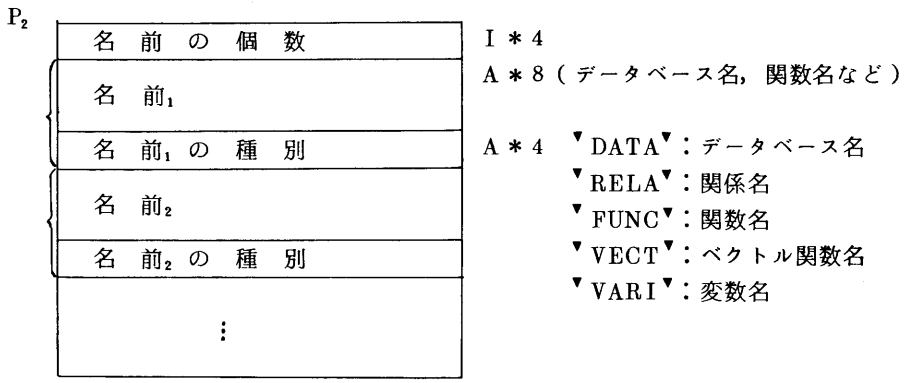
```
DIMENSION MBUF(50)
      ⋮
CALL SHOW¥D(IRC,MBUF,200,▼#ALLNAME▼)
```

・関数FUNC01の定義情報を，検索する．

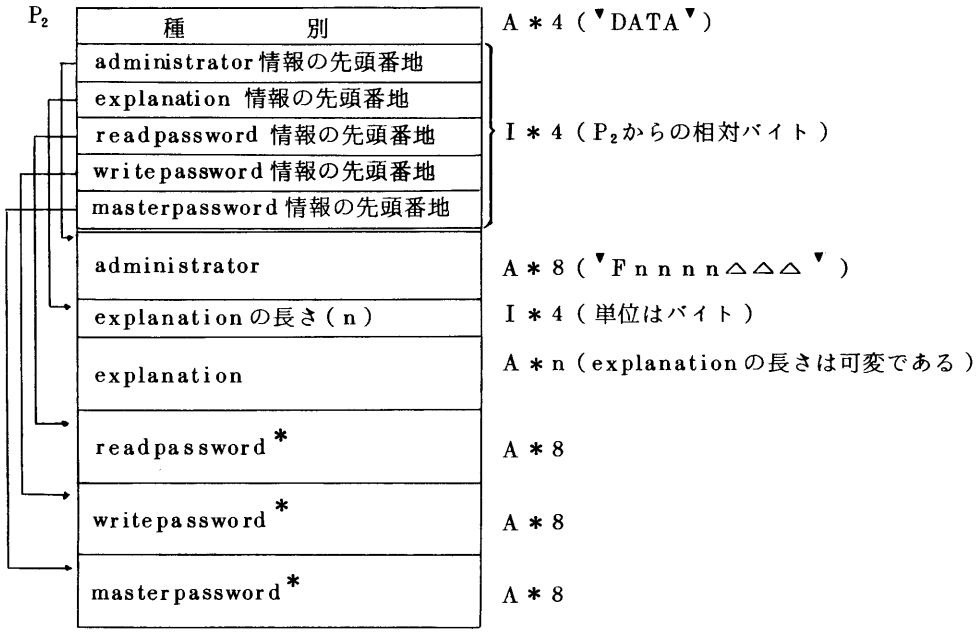
```
DIMENSION MBUF(50)
      ⋮
CALL SHOW¥D(IRC,MBUF,200,▼FUNC01△△▼)
```

(5) データベース定義情報

a．第4引数が，▼#ALLNAME▼の場合

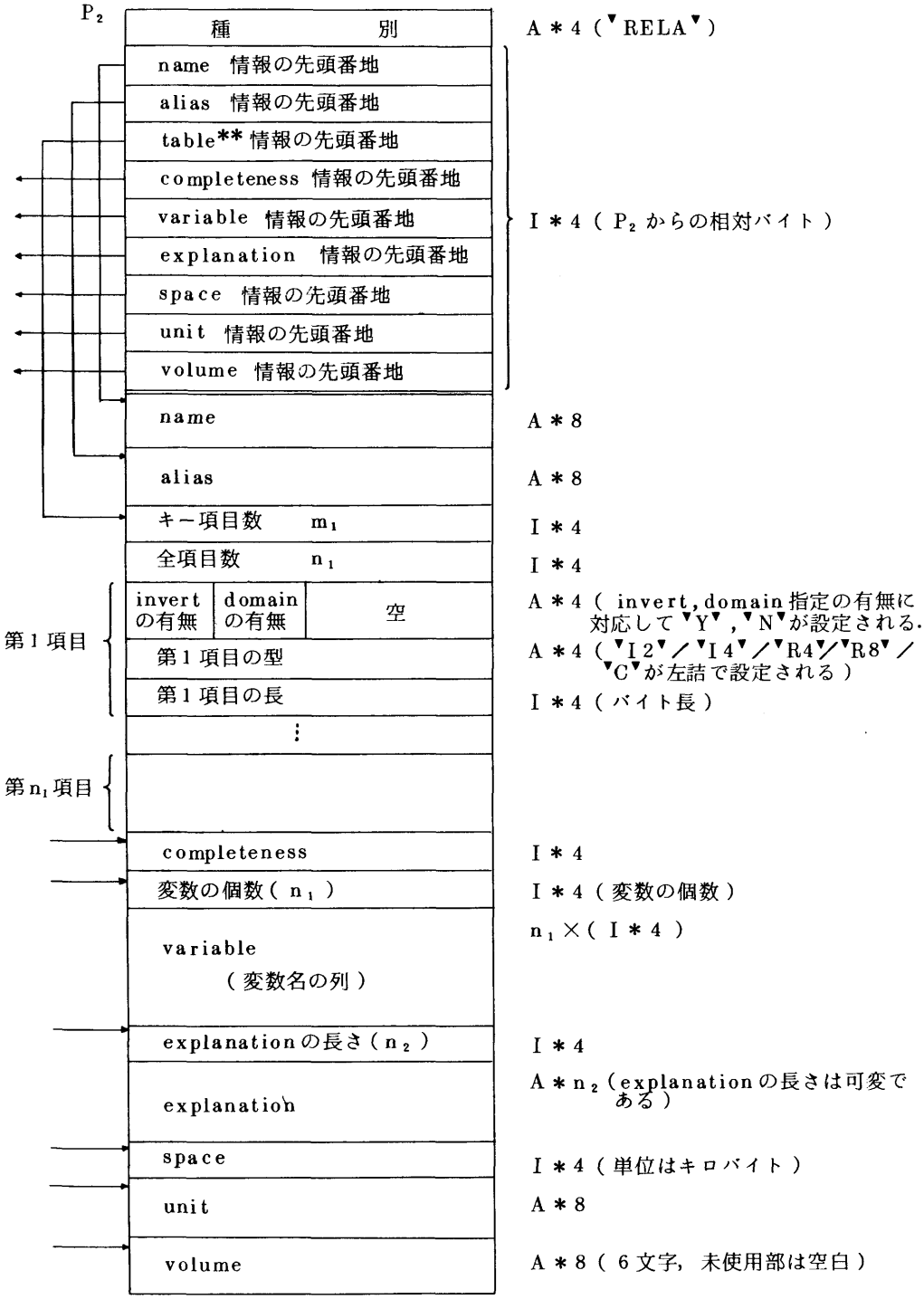


b．第4引数が，データベース名の場合



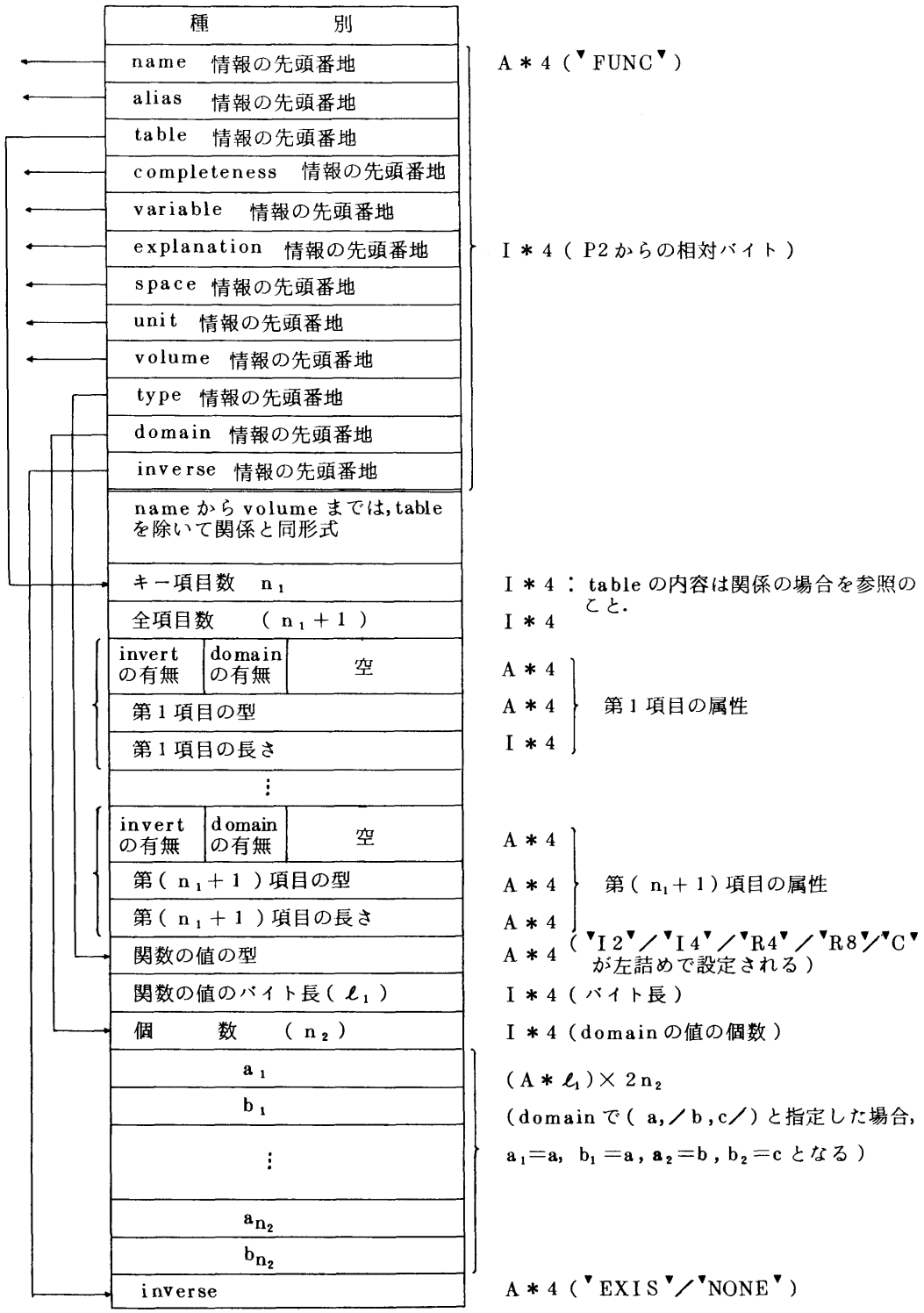
＊データベース管理者でない場合，空白が設定される．

c．第4引数が関係名の場合

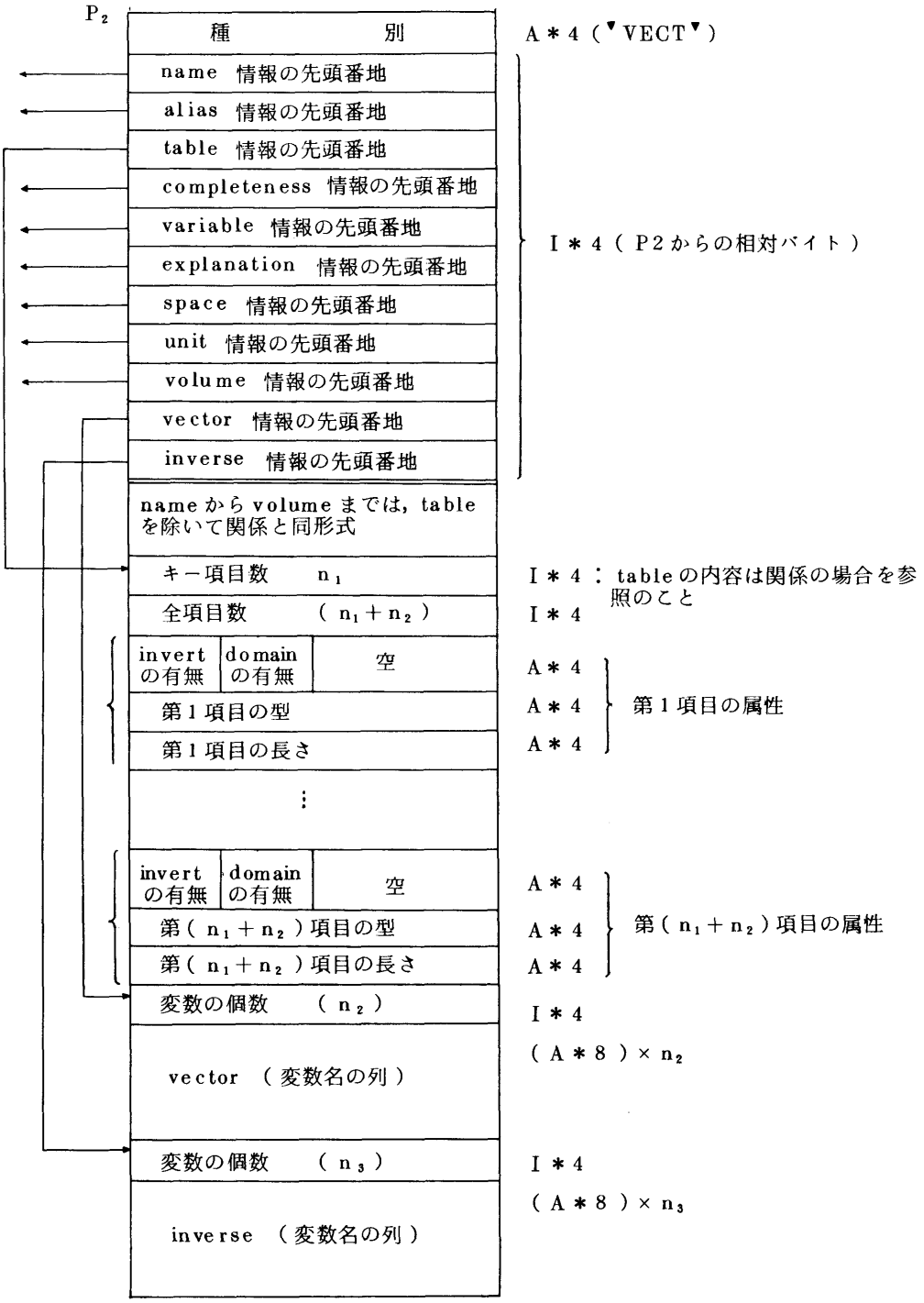


**関係，関数，ベクトル関数の組表現での型，長さなどの情報を与える．キー項目数，全項目数をそれぞれm_1，n_1とすると関係の場合$n_1=m_1$，関数の場合$n_1=m_1+1$，ベクトル関数の場合$n_1=m_1+$（ベクトルの次数）となる．

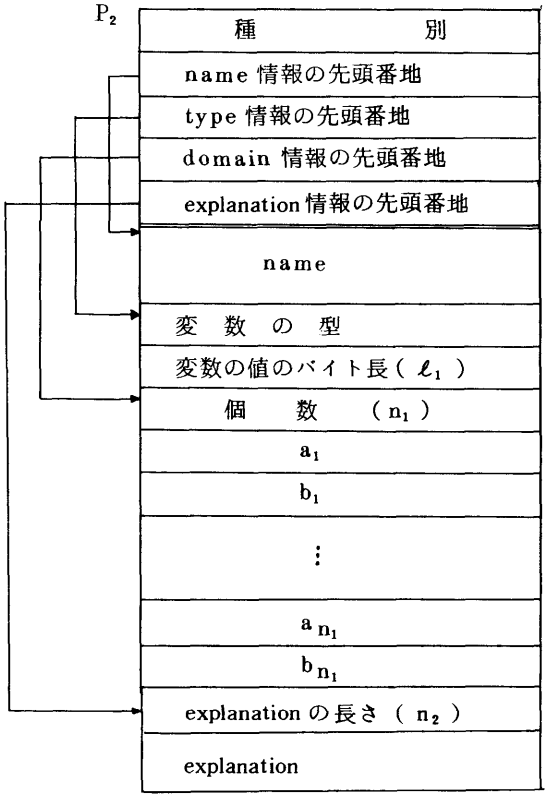
d．第４引数が関数名の場合



e．第4引数がベクトル関数の場合



f．第4引数が変数名の場合



6.2.5 CREA¥D(create)

(1) 機　能

データベースの創成を関係，関数，ベクトル関数単位に行う．CREA¥Dの入力データ（関係，関数，ベクトル関数の実現値で表形式のデータ）の形式は，利用者自身であらかじめ，整えておかねばならない．CREA¥Dによりデータセット名が▼課題名・関係名・データベース名▼の形式のデータファイルが作られる．

(2) 呼び出し形式と引数の説明

```
CALL　CREA¥D(P1, P2)
```

[P1]：完了コード（Iv＊4）

0＝正常終了．

1＝データベース創成の資格がない（データベースのオープン時，マスタパスワードが指定されていない）．

2＝第2引数で指定された名前がデータベース定義に登録されていない．

3＝データファイルが作成ずみである．

4＝データファイルの割当てエラー.

5＝データファイルの領域が不足した.

8＝入力データセットが存在しない.

50＝データベースがオープンされていない.

P_2 ：名前（A＊8，Av＊8，Aarray＊8）

創成する関係，関数，ベクトル関数の名前，あるいはその別名を指定する.

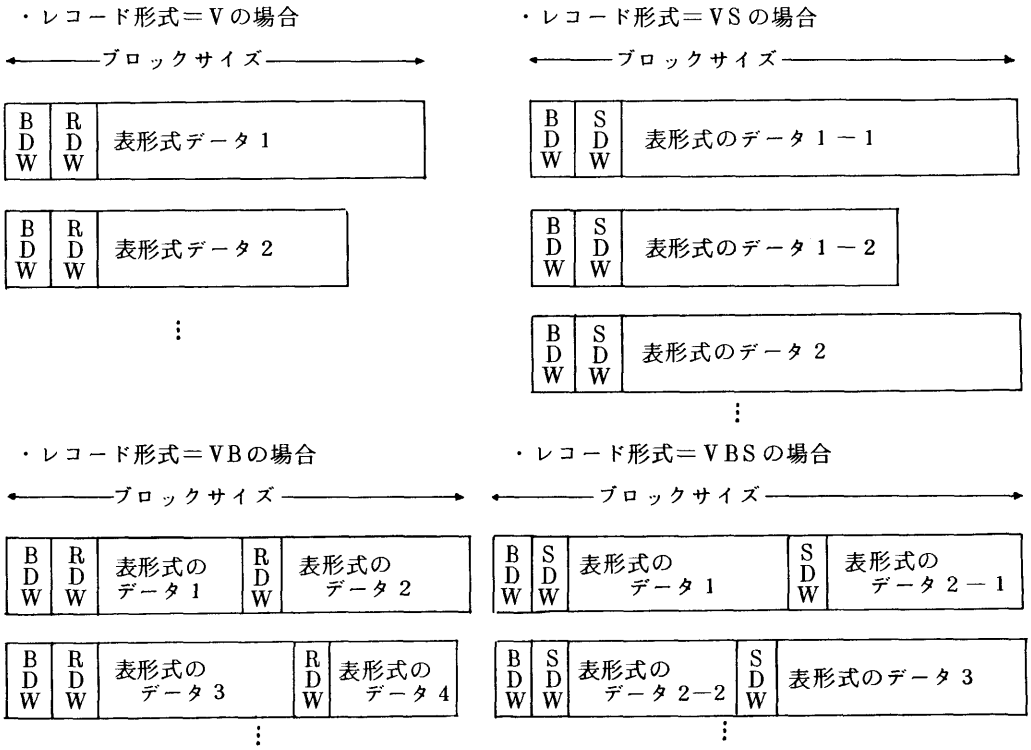
(3) 使用上の注意

・CREA¥Dは，データベース管理者がマスタパスワードを指定してデータベースをオープンしたときのみ有効である.

・CREA¥Dの入力データは，利用者自身でフォーマット化し，CREA¥Dの呼び出し前に，DD名＝FT48F001を割り当てておく必要がある．入力データセットの形式は，次の4で述べる.

(4) 入力データセットの形式

CREA¥Dの入力データセットはレコード形式が，V，VB，VS，VBSの順データセット（あるいは区分データセットのメンバ）でなければならない．それぞれのレコードフォーマットを次にまとめる．BDW，RDW，SDWはそれぞれブロック記述語，レコード記述語，セグメント記述語を表す．なお，Fortranでは，書式なし順次入出力によりレコード形式VS，VBSの順データセットを作成できる.



5.4節の例の関係MANUCARS，関数COLOR，ベクトル関数MANUFACTの入力データを用意する場合，次のFortranのWRITE文を用いればよい．

・関係MANUCARS

```
WRITE(1)MNO,CNO
```

・関数COLOR

```
WRITE(1)CNO,COLOR
```

・ベクトル関数MANUFACT

```
WRITE(1)MNO,MNAME,PRODUCT,CITY
```

(5) 適　用

関係，関数，ベクトル関数のデータファイルを作成するのに使用する．

(6) 使用例

・入力データセットF0001. INPUT.DATAから関係REL01のデータファイルを創成する．

i) 入力データセットF0001.INPUT.DATAの準備．

ii) 入力データカセットの割当て．

・JCL（バッチ処理）

```
//FT48F001 DD DSN=F0001.INPUT.DATA,DISP=SHR
```

・TSSコマンド

```
ALLOC F(FT48F001) DA(INPUT.DATA) SHR
```

iii) アプリケーションプログラムの実行

```
CALL CREA¥D(IRC,'REL01 △△△')
```

6.2.6 CLEA¥D(clear)

(1) 機　能

関係，関数，ベクトル関数を消去する．

(2) 呼び出し形式と引数の説明

```
CALL CLEA¥D(P1,P2)
```

P_1 ：完了コード（Iv＊4）

0＝正常終了．

1＝データベース消去の資格がない（データベースのオープン時，マスタパスワードが指定されていない．

2＝第2引数で指定した名前がデータベースに定義されていない．

3＝データファイルが作成されていない．

50＝データベースがオープンされていない．

P_2 ：名前（A＊8，Av＊8，Aarray＊8）

消去する関係，関数，ベクトル関数の名前，あるいは別名を指定する．

(3) 使用上の注意

・CLEA¥Dは，データベース管理者がマスタパスワードを指定してデータベースをオープンしたときのみ有効である．

(4) 適　用

関係，関数，ベクトル関数のデータファイルを消去するのに使用する．

(5) 使用例

・関係REL01のデータファイルを消去する．

```
CALL CLEA¥D( IRC, 'REL01 △△△' )
```

6.2.7 VALU¥D(value)

(1) 機　能

関係，関数，ベクトル関数の値を求める．値は次のように定義される．

値＝ { 指定レコードがあれば1，無ければ0　関係の場合；
　　 { 関数値　関数の場合；
　　 { ベクトル関数値　ベクトル関数の場合．

(2) 呼び出し形式と引数の説明

```
CALL  VALU¥D( P1 , P2 , P3 , P4 , P5 )
```

[P1] ：完了コード(Iv＊4)

0＝正常終了．

1＝第4引数に指定した名前がデータベースに定義されていない．

2＝データファイルが創成されていない．

3＝データファイルのリードエラー．

4＝第2，3引数で指定した値設定領域が不足している．

5＝関数，ベクトル関数について，第5引数で指定したレコードが存在しない．

50＝データベースがオープンされていない．

[P2] ：値設定領域(Iv＊4 , I＊4 array)

値の設定領域を指定する．値設定領域は，関係の場合，4バイト，関数，ベクトル関数の場合，関数値を設定するだけのバイト長を用意する必要がある．

P_3 ：値設定領域のバイト長(I＊4 , Iv＊4)

P_4 ：名前(A＊8 , Av＊8 , A array＊8)

関係名，関数名，ベクトル関数名，あるいはその別名を指定する．

P_5 ：引数リスト(I＊4 array)

関係については，関係の組を，関数，ベクトル関数については，引数または引数の組を設定する．

(3) 使用上の注意

・関係を構成する変数の属性などは，SHOW¥Dで調べる．

(4) 適　用

関係については特定の関係が成り立っているかどうかを調べるために，関数およびベクトル関数については関数値を求めるのに使用する．

(5) 使用例

・変数A，B，Cからなる関係REL01で，A＝100，B=50，C=▼ABCD▼の組があるかどうかを調べる．A，B，Cの型はそれぞれI4，I4，C4とする．

```
DIMENSION NLIST(3)
        ⋮
NLIST(1)=100
NLIST(2)=50
NLIST(3)=▼ABCD▼
CALL VALU¥D(IRC,IV,4,▼REL01△△△▼,NLIST)
```

6.2.8 FIND¥D(find)

(1) 機　能

関係，関数，あるいはベクトル関数から，引数リストで指定された条件を満たすレコードのキー集合を求める．この機能は，例えば，5項関係（A，B，C，D，E）において，引数リストで（a，b，*，d，*）（a，b，dはそれぞれA，B，Dの値；*は不定を表す）と指定すると，A＝a，B＝b，D＝dであるすべての組に対するキー集合を求める．関数，ベクトル関数については，引数，関数値の組からなる関係と考える．FIND¥Dで求めたものが現キー集合であり，これをSAVE¥Dで保存することにより，集合演算の対象となる．キー集合から値集合を求めるには，EVAL¥Dを使用する．

(2) 呼び出し形式と引数の説明

CALL　FIND¥D(P_1，P_2，P_3，P_4)

P_1：完了コード（Iv＊4）

0＝正常終了．

2＝第3引数で指定された名前が，データベースに定義されていない．

3＝データファイルが創成されていない．

4＝引数リストの指定に誤りがある．

5＝検索項目に対して転置形が作成されていない．

10＝データファイルのリードエラーである．

50＝データベースがオープンされていない．

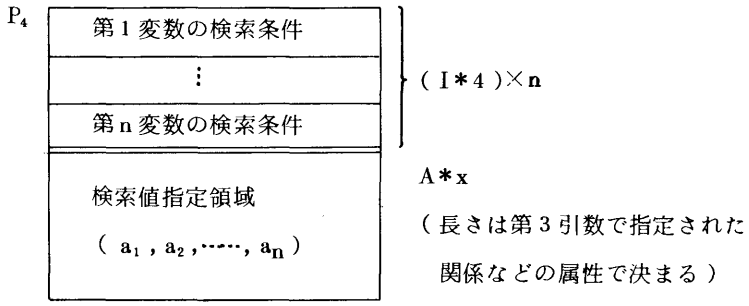
P_2：キー集合の要素数（Iv＊4）

求まったキー集合の要素数が設定される．

P_3：名前（A＊8，Av＊8，A array＊8）

検索対象とする関係，関数，ベクトル関数の名前，あるいは別名を指定する．

P_4 ：引数リスト（ I＊4 array）



検索条件＝
- 0…その変数の値が任意でよいことを示す.
- 1…＝, 指定値と等しいレコードを探す.
- 2…≠, 指定値と等しくないレコードを探す.
- 3…＞, 指定値より大きいレコードを探す.
- 4…≧, 指定値より大きいか等しいレコードを探す.
- 5…＜, 指定値より小さいレコードを探す.
- 6…≦, 指定値より小さいか等しいレコードを探す.

(3) 使用上の注意

・関係を構成する変数の属性などは, SHOW¥D で調べる.

・演算結果の要素数が0でも, それは現キー集合となる.

(4) 適 用

特定の条件を満たすレコードを求めるのに利用する.

(5) 使用例

・変数A, B, Cからなる関係REL01で, 条件A≧100, B＝任意, C＝▼ABCDEF▼を満足するキー集合を求める. ここで, A, B, Cの型はそれぞれI4, I2, C6とする.

```
INTEGER ALIST(10)
      ⋮
ALIST(1)=4
ALIST(2)=0
ALIST(3)=1
ALIST(4)=100
ALIST(5)=▼△△AB▼
ALIST(6)=▼CDEF▼
CALL FIND¥D(IRC,KNO,▼REL01△△△▼,ALIST)
```

ALIST

4
0
1
1 0 0
△* △* A B
C D E F

*任意の値でよい

6.2.9 MEET¥D, JOIN¥D, DIFF¥D(meet, join, difference)

(1) 機　能

2つの保存キー集合の集合積(MEET¥D), 集合和(JOIN¥D), 集合差(DIFF¥D)を求める. 演算結果が現キー集合となる. なお, 同じ関係, 関係, ベクトル関数から得られたキー集合についてのみ演算が可能である.

(2) 呼び出し形式と引数の説明

```
CALL MEET¥D(P1 ,P2 ,P3 ,P4 )
CALL JOIN¥D(P1 ,P2 ,P3 ,P4 )
CALL DIFF¥D(P1 ,P2 ,P3 ,P4 )
```

[P1] ：完了コード(Iv＊4)

0＝正常終了.

1＝第3, 4引数で指定されたキー集合が存在しない.

2＝第3, 4引数で指定されたキー集合が, 同じ関係, 関数, ベクトル関数から得られたものでない.

50＝データベースがオープンされていない.

[P2] ：演算結果である現キー集合の要素数(Iv＊4)

P3 ：集合演算の第1引数となるキー集合名(A＊8 , Av＊8 , A array＊8)

P4 ：集合演算の第2引数となるキー集合名(A＊8 , Av＊8 , A array＊8)

(3) 使用上の注意

・演算の対象となるキー集合は, 前もってSAVE¥Dにより保存された保存キー集合でなければならない.

・引数の2つのキー集合は, 同一関係(あるいは関数, ベクトル関数)から得られたものでなければならない.

・演算結果の要素数が0でも, それは現キー集合となる.

(4) 適　用

検索集合の集合積, 集合和, 集合差をとるのに用いる.

(5) 使用例

・保存キー集合SET1, SET2の集合積をとる.

```
CALL MEET¥D(IRC,KNO,'SET1 △△△△','SET2 △△△△')
```

6.2.10 CARD¥D(cardinal)

(1) 機　能

キー集合の要素数を求める.

(2) 呼び出し形式と引数の説明

CALL CARD¥D(P_1, P_2, P_3)

P_1 ：完了コード（ Iv＊4 ）

0＝正常終了.

1＝第3引数で指定したキー集合が存在しない.

50＝データベースがオープンされていない.

P_2 ：要素数（ Iv＊4 ）

P_3 ：対象とするキー集合名（ A＊8 , Av＊8 , Aarray＊8 ）

現キー集合の場合は，集合名として ▼△△△△△△△△▼ を指定する.

(3) 使用例

・キー集合KEYSET1の要素数を求める.

CALL CARD¥D(IRC,KNO,▼KEYSET1△▼)

6.2.11 EVAL¥D(eval)

(1) 機　能

キー集合から値集合を求める. 値を求める項目は，引数リストで指定する.

(2) 呼び出し形式と引数の説明

CALL EVAL¥D(P_1, P_2, P_3, P_4, P_5)

P_1 ：完了コード（ Iv＊4 ）

0＝正常終了.

1＝第4引数で指定されたキー集合が存在しない.

2＝データファイルのリードエラー.

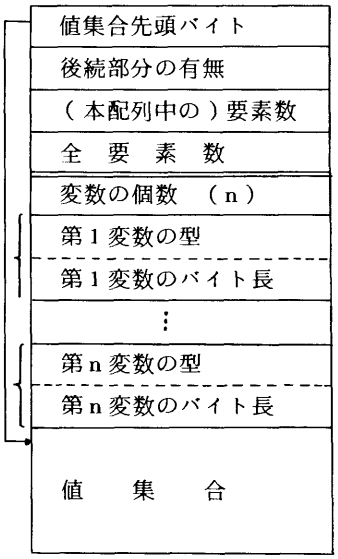
3＝第5引数の引数リストの指定に誤りがある.

6＝第2，3引数の値集合設定領域が不足している.

50＝データベースがオープンされていない.

P_2 ：値集合設定領域（ I＊4 array）

値集合設定領域には次の形式で値集合が設定される.



I＊4（単位はバイト）

I＊4（0＝後続部分なし，1＝後続部分あり）

I＊4

I＊4

I＊4

A＊4（▼I2△△▼／▼I4△△▼／▼R4△△▼／▼R8△△▼／▼C△△△▼）

I＊4（単位はバイト）

変数の型と長さは，引数リストで，1を指定した変数分だけ設定される．

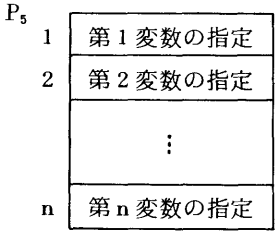
P_3 ：値集合設定領域のバイト長（I＊4，Iv＊4）

P_4 ：キー集合名（A＊8，Av＊8，A array＊8）

値を求めるキー集合名を指定する．現キー集合の場合は，集合名として▼△△△△△△△△　を指定する．

P_5 ：引数リスト（I＊4，Iv＊4 array）

各変数について値を求めるかどうかを（0，1）で指定する．



I＊4 { 1＝対応する変数の値を求める．
0＝対応する変数の値を求めない．

(3) 使用上の注意

・EVAL¥Dを用いた場合，値集合設定領域に入るだけの値集合先頭部分が返される．後続部分を得るには，繰返しGETN¥Dを用いる必要がある．この場合，後続部分があるかどうかは，値集合設定領域の▼後続部分の有無▼に示される．

(4) 適　用

キー集合から値集合を求めるのに用いる．

(5) 使用例

・4変数からなる関係REL01から得られた保存キー集合SET01を評価する．値は，第2，第3変数について求める．

```
INTEGER VSET(100),ALIST(4)
    :
ALIST(1)=0
ALIST(2)=1
ALIST(3)=1
ALIST(4)=0
CALL EVAL¥D(IRC,VSET,400,'SET01△△△',ALIST)
```

6.2.12 GETN¥D, GETF¥D (getnext, getfirst)

(1) 機 能

EVAL¥Dにより求めた値集合が，利用者の用意した値集合設定領域に一度に入りきれないときに後続部分を求める．GETN¥Dは，値集合の次の後続部分を求める時に，GETF¥Dは，再び最初の部分を求めるのに使用する．

(2) 呼び出し形式と引数の説明

```
CALL  GETN¥D(P1 ,P2 ,P3 )
CALL  GETF¥D(P1 ,P2 ,P3 )
```

P_1 ：完了コード（Iv＊4）

0＝正常終了．

6＝第2，3引数で指定した値集合設定領域が不足している．

7＝値集合を入れているファイルのリードエラー．EVAL¥Dを呼んでいない時，後続部分がない時に，GETN¥Dを呼び出した時にこのエラーを起す．

50＝データベースがオープンされていない．

P_2 ：値集合設定領域（I＊4 array）

値集合の表現については，EVAL¥Dを参照のこと．

P_3 ：値集合設定領域のバイト長（I＊4，Iv＊4）

(3) 使用上の注意

・GETN¥D, GETF¥Dは，EVAL¥D呼び出し後，続けて呼び出さねばならない．EVAL¥DとGETN¥D/GETF¥Dの間に他のサブルーチンを呼び出した場合，結果は保証されない．

・GETN¥D, GETF¥D使用時，値集合の次の部分集合が有るかどうかを，利用者が確認しなければならない．

(4) 適 用

値集合が，値集合設定用の利用者領域より大きい場合，つぎつぎに後続部分を求めるのに使用する．

(5) 使用例

・値集合の次の部分集合をMBUFに求める．

```
DIMENSION MBUF(100)
    ⋮
CALL GETN¥D(IRC,MBUF,400)
```

6.2.13 INSE¥D(insert)

(1) 機　能

データベースにレコードを追加する.

(2) 呼び出し形式と引数の説明

```
CALL INSE¥D(P1 ,P2 ,P3 )
```

P_1 ：完了コード(Iv＊4)

0＝正常終了.

1＝パスワードエラー. データベースのオープン時に, マスタパワード, あるいはライトパスワードを指定していないのでレコードの追加は行えない.

2＝第2引数で指定した名前がデータベースに定義されていない.

3＝第2引数で指定した名前のデータファイルが作成されていない.

4＝データファイルのリード／ライトエラー. あるいは, データファイルのスペースが不足した.

5＝同じレコードが既に存在する. 関数, ベクトル関数の場合, 引数の値が同じかどうかがチェックされる.

6＝第3引数で指定した値が取りうる値の範囲内でない.

50＝データベースがオープンされていない.

P_2 ：名前(A＊8, Av＊8 , A array＊8)

レコードを追加する関係, 関数, ベクトル関数の名前, あるいは別名を指定する.

P_3 ：追加レコード(I＊4 array)

追加するレコードの表形式のデータを指定する. 関係については, 関係の組を, 関数, ベクトル関数については, 引数, 関数値の組を指定する. 1レコードのみ指定できる.

(3) 適　用

データファイルにレコードを追加するのに使用する.

(4) 使用例

・関係REL01にレコードを追加する.

```
INTEGER IVALUE(20)
    ⋮
IVALUEの設定
CALL INSE¥D(IRC,'REL01 △△△',IVALUE)
```

6.2.14 DELE¥D(delete)

(1) 機　能

データベースからレコードを削除する.

(2) 呼び出し形式と引数の説明

```
CALL DELE¥D(P1,P2,P3)
```

P1 ：完了コード(Iv＊4)

0＝正常終了.

1＝パスワードエラー. データベースのオープン時に, マスタパスワード, あるいはライトパスワードを指定していないのでレコードの削除は行えない.

2＝第2引数で指定した名前がデータベースに定義されていない.

3＝第2引数で指定した名前のデータファイルが作成されていない.

4＝データファイルのリード／ライトエラー.

5＝消去すべきレコードがデータファイルに見つからない.

50＝データベースがオープンされていない.

P2 ：名前(A＊8 , Av＊8 , A array＊8)

レコードを削除する関係, 関数, ベクトル関数の名前, あるいは別名を指定する.

P3 ：削除レコード(I＊4 array)

削除するレコードの表形式のデータを指定する. 関係については, 関係の組を, 関数, ベクトル関数については, 引数あるいは引数の組を指定する. 1レコードのみ指定できる.

(3) 適　用

データファイル創成後に, データファイルからレコードを削除するのに用いる.

(4) 使用例

・関係REL01からレコードを削除する.

```
INTEGER IVALUE(20)
      ⋮
IVALUEの設定
CALL DELE¥D(IRC,'REL01△△△',IVALUE)
```

6.2.15 DELS¥D(delete set)

(1) 機　能

データベースから, キー集合で指定されたレコード群を削除する.

(2) 呼び出し形式と引数の説明

```
CALL DELS¥D(P1,P2)
```

P1 ：完了コード(Iv＊4)

0＝正常終了.

1＝パスワードエラー．データベースのオープン時に，マスタパスワードあるいはライトパスワードを指定していないのでレコードの削除は行えない．

4＝データファイルのリード／ライトエラー．

8＝引数2で指定されたキー集合が存在しない．

50＝データベースがオープンされていない．

P_2 ：キー集合名（A＊8，Av＊8，A array＊8）

削除するレコードのキー集合の名前を指定する．現キー集合は▼△△△△△△△△△▼を指定する．

(3) 適 用

データファイル創成後に，データファイルから特定の条件を満たすレコード群を消去するのに用いる．

(4) 使用例

・関係REL01から得られた保存キー集合KEYSET1により，そのキー集合が示すレコード群を関係REL01から消去する．

```
CALL DELS¥D(IRC,▼KEYSET1△▼)
```

6.2.16 UPDA¥D(update)

(1) 機 能

関数あるいはベクトル関数の値の更新を行う．

(2) 呼び出し形式と引数の説明

```
CALL UPDA¥D(P1,P2,P3)
```

P_1 ：完了コード（Iv＊4）

0＝正常終了．

1＝パスワードエラー．データベースのオープン時に，マスタパスワードあるいはライトパスワードを指定していないので関数値の更新ができない．

2＝第2引数で指定した名前が，データベースに定義されていない．

3＝第3引数で指定した更新レコードの関数の引数の値がデータファイルにないので，関数値の更新ができない．

4＝データファイルのリード／ライトエラー．あるいは，データファイルの領域が不足した．

5＝第3引数で指定した値が取りうる値の範囲内でない．

6＝第2引数で指定された名前のデータファイルが作成されていない．

50＝データベースがオープンされていない．

P_2 ：名前（A＊8，Av＊8，A array＊8）

値の更新を行う関数，ベクトル関数の名前，あるいは別名を指定する．

P_3 ：更新レコード（I＊4 array）

更新するレコードの表形式のデータを指定する．関数，ベクトル関数の引数および値の組

を指定する．1レコードしか指定できない．

(3) 使用上の注意

・関数の引数の値がデータファイルにない場合，エラーが通知され，更新は行われない．

(4) 適　　用

関数値（あるいはベクトル関数の値）を更新するのに使用する．

(5) 使用例

・関数F1（z＝F1（x，y））で，x＝150，y＝250でのzの値を300から400に変更する．x，y，zはいずれも4バイト長の整数とする．

```
LNTEGER  IVALUF(3)
      ⋮
IVALUE(1)=150
IVALUE(2)=250
IVALUE(3)=400
CALL  UPDA¥D(IRC,▼F1△△△△△△▼,IVALUE)
```

6.2.17　INVE¥D（invert）

(1) 機　　能

逆関数を新規に定義，作成する．

(2) 呼び出し形式と引数の説明

```
CALL  INVE¥D(P1,P2,[,P3])
```

P_1 ：完了コード（Iv＊4）

0＝正常終了．

1＝パスワードエラー．データベースのオープン時にマスタパスワード，あるいはライトパスワードを指定していないので，逆関数は定義できない．

2＝第2引数で指定した名前が関数あるいはベクトル関数としてデータベースに定義されていない．

3＝第2引数で指定した名前のデータファイルが作成されていない．

4＝第3引数で指定された逆関数定義要素に誤りがある．

5＝データファイルの領域が不足した．

8＝既に逆関数が存在する．

50＝データベースがオープンされていない．

P_2 ：名前（A＊8，Av＊8，A array＊8）

逆関数を定義する関数，ベクトル関数の名前，あるいはその別名を指定する．

P_3 ：逆関数定義要素（I＊4，Iv＊4）

ベクトル関数における逆関数定義要素をベクトルにおける順序で与える．関数の場合，こ

の引数は不要である.

(3) 使用上の注意

・INVE¥Dは，データベースの定義では逆関数定義が指定されていなかったものについて新規に逆関数を作る．逆関数はデータベースのクローズ後も保存される.

(4) 適　　用

データベース定義で逆関数が定義されていなかった関数，ベクトル関数について，新規に逆関数の定義，作成を行うものである.

(5) 使用例

・ベクトル関数VECT01について，ベクトルの第3番目の要素について逆関数を定義する.

```
CALL INVE¥D(IRC,▼VECT01 △△▼,3)
```

6.2.18 SORT¥D(sort)

(1) 機　　能

値集合をソートする．ソートの方法はソートリストで指定する.

(2) 呼び出し形式と引数の説明

```
CALL SORT¥D(P1 ,P2 ,P3 )
```

P_1 ：完了コード（Iv＊4）

0＝正常終了.

1＝第3引数のソートキー数の指定が正しくない.

2＝第2引数のソートリストの指定が正しくない.

50＝データベースがオープンされていない.

P_2 ：ソートリスト（I＊4 array）

ソートリストの形式は，次のとおりである．8バイト単位でソートキーを複数個指定する．先に指定したものほど，ソートの優先順位が高い.

ソートキー1	値集合での変数の位置$_1$	I＊4
	ソート内容$_1$	I＊4（1＝昇順，2＝降順）
⋮	⋮	
ソートキーn	値集合での変数位置n	
	ソート内容n	

P_3 ：ソートキー数（I＊4，Iv＊4）

ソートキーの数を指定する．この値が0以下，あるいは値集合の変数の数より大きい場合は完了コード＝1で通知する.

(3) 使用例

・3変数の値集合について，第2番目の変数については昇順に，第3番目の変数については降順に並べる．

```
INTEGER SLIST(4)
      ⋮

SLIST(1)=2
SLIST(2)=1
SLIST(3)=3
SLIST(4)=2
CALL SORT¥D(IRC,SLIST,2)
```

6.2.19 SAVE¥D(save)

(1) 機　能

現キー集合に名前をつけて保存する．

(2) 呼び出し形式と引数の説明

```
CALL SAVE¥D(P1,P2)
```

[P1] ：完了コード（Iv＊4）

0＝正常終了．

1＝現キー集合が存在しない．

2＝引数2のキー集合名の名前がおかしい．

50＝データベースがオープンされていない．

P2 ：キー集合名（A＊8，Av＊8，Array＊8）

保存キー集合の名前を指定する．名前は単純名である．

(3) 適　用

キー集合を，後で参照するために一時的に保存するのに使用する．

(4) 使用例

・現キー集合にキー集合名KEYSET1をつけて保存する．

```
CALL SAVE¥D(IRC,▼KEYSET1△▼)
```

6.2.20 UNSA¥D(unsave)

(1) 機　能

保存キー集合を消去する．

(2) 呼び出し形式と引数の説明

```
CALL UNSA¥D(P1,P2)
```

[P1] ：完了コード（Iv＊4）

0＝正常終了.

1＝第2引数で指定した保存キー集合が存在しない.

50＝データベースがオープンされていない.

P_2 ：キー集合名（A＊8，Av＊8，A array＊8）

消去する保存キー集合の名前を指定する.

(3) 適 用

不要となった保存キー集合を消去するのに用いる.

(4) 使用例

・保存キー集合KEYSET1を消去する.

```
CALL UNSA¥D(IRC,▼KEYSET1 △▼)
```

6.3 Adbis／DMP使用上の制限事項

Adbis／DMP第1版における使用上の制限は次のとおりである.

・関係，関数，ベクトル関数の個数は，合計で最大40個までである.

・変数の個数は，最大100個までである.

・データセット機番47，48，49，50，51，…，50＋xは使用できない．ここで，xは，関係，関数，ベクトル関数の個数の和である．機番47（DD名＝FT47F001），機番48（DD名＝FT48F001），機番49（DD名＝FT49F001）は，それぞれ作業ファイル，データベース創成用入力データセット，データベース定義用入力データに，機番50以降は，マスタファイル，データファイルの割当てに使用している.

・Adbis／DMPを利用するアプリケーションプログラムは，関数副プログラム，サブルーチン副プログラムの名前として，次のものは使用できない.

1）¥Dで終る6文字の名前

2）FACOM OS IV／F4，TACライブラリ［6］のうち，次のもの.

基本ライブラリ……QANUCK，QAPHCK，QBIT，QBITON，QBITOF，QBITCK，QINDEX，QSBSTR，QERROR，QERR01

サービスライブラリ……¥BLNKC，FR¥MAN，GT¥MAN，IO¥SUB，¥OPEN，¥CLOSE，¥READ，¥WRITE，TS¥UID

制御ライブラリ……TSS¥CM，TSS¥ER，TSS¥IN

データセットライブラリ…D¥ATTR，D¥DALC，D¥DDCK，D¥DSCK，D¥FREE

7. おわりに

本稿では，研究者向データベース統合支援システムAdbisの概要とその基本モジュールであるDMPの仕様を述べた．DMPは，関係モデルのデータベース管理のための基本的必須機能を持ち，単独で使用できる．DMPについては，今後，効率面の改善を行うと共に，機能を拡充し使い易いものにしたいと考えている．又，DMPに関連したユーティティプログラムを幾つか作る予定であ

る．これには下記のものを予定している．

- DDLによるデータベース定義の構文チェックプログラム
- データベース創成用データの保全性チェックプログラム
- データベースから創成用形式データへの変換プログラム

参考文献

1. Clark,W.A. The Functional Structure of OS/360 : Part Ⅲ Data Management, IBM Syst. J.,**5**,1,1966,30—51.
2. Date,C.J. *An Introduction to Database System*, Addison—Wesley,Reading, Mass.,1975,1977(2nd ed.).
3. Martin,J. *Computer Data Base Organization*, Prentice—Hall, Englewood Cliffs,N.J.,1975,1977(2nd. ed.).（穂鷹良介ほか訳 データベース管理，日本コンピュータ協会）
4. Tsichritzis, D.C. and Lochorsky, F.H. *Data Base Management Systems*, Academic Press,New York, 1977.（斉藤忠夫訳 データベース管理システム，日本コンピュータ協会）.
5. Ullman J.D. *Principles of Database Systems*, Computer Science Press, Potomac,Maryland, 1980.（国井利泰ほか訳 データベース・システムの原理，日本コンピュータ協会）
6. 計算機マニュアル FACOM OS/IV TAC/LIB解説書，富士通㈱．