APLとOR（3）
ファイル構築と検索，予測プログラム

三枝協亮・松田寿子

日本アイ・ビー・エムでは計画立案や予測作業を機械化するに当って，担当者自身による機械化を「エンド・ユーザーのためのＡＰＬ利用推進」という形で進めてきた．各担当者が思いのままにＡＰＬを使って業務をこなすには，それなりの利用環境が整っていなければならず，しかるべき体制が必要である．本題の“ファイル構築と検索プログラム”は，こうした命題のもとにＡＰＬ利用推進スタッフにより整備された集中的なデータ・ベースとその検索システムを意味し，“予測プログラム”は予測担当者が検索システムを用いて予測業務を実行することを意味するものである．このような意図のもとに全体をＡＰＬ環境を提供する立場とその受け手の利用する立場とに分け，前者を三枝が，後者を松田が体験にもとづいて記述した．

1. ＡＰＬ利用推進に対する考察

1.1 ＡＰＬについて

1972年頃よりＡＰＬと深くかかわり合い，以来約10年間多くの人々とともにＡＰＬ普及の一翼を担ってきた．ようやく一般の企業においてもＡＰＬを有効に利用していただけるようにコンピュータの環境も整いつつあり，過去４～５年の間に急速にＡＰＬの導入が進んでいる．ＡＰＬがどのようなもので，またどのように使用するべきかということについて信頼できる一般的な結論を述べることはむずかしいことであるが，これら多くのＡＰＬユーザーの事例に直接・間接にかかわり合った経験から，また現在それらのユーザーを技術的にサポートする責任者としての立場から，ＡＰＬに対する意見を述べてみたい．

1.2 ＡＰＬ言語について

ＡＰＬがCOBOL, FORTRAN, PL/I と同様にプログラミング言語の１つであることはその名前（A PROGRAMMING LANGUAGE）に示されたとおりである．本来対話型言語であるから，どの分野の仕事であれ，エンド・ユーザーがコンピュータ端末で，直接操作するエンド・ユーザー言語でもある．

ＡＰＬ言語の基本である関数や作用子には最初から機能が定義されている40種類前後の原始関数と，関数を修飾し，数多くの派生関数を生み出す原始作用子以外に，ユーザーが他の関数と組み合わせて定義する定義関数，または近い将来には定義作用子が含まれる．

これらのＡＰＬの関数は作用するべき引数との相対的位置関係により，Nilladic（引数を取らないもの），Monadic（右側にのみ引数をもつもの），Dyadic（左，右に引数をもつもの）の３通りに分けられ，またそれらは各々，結果値を持つものと持たないものに分けられるので合計６通りのタイプに分類される．これらを表１に示す．

表１ ＡＰＬの関数

関数の例		関数の定義
Nilladic		▽ QUIT
→	（原始関数）	[1] →
QUIT	（定義関数）	[2] ▽
Monadic		▽ *R*←SIGN *A*
× *V*	（原始関数）	[1] *R*← × *A*
SIGN *V*	（定義関数）	[2] ▽
Dyadic		▽ *R*←*A* SIGN *B*
$V_1 \times V_2$	（原始関数）	[1] *R*←*A* MULT *B*
V_1 MULT V_2	（定義関数）	[2] ▽

また，結果値をもつ関数と変数の組合せは，それ自体変数（値）として用いられるので他の関数の引数とすることができる．

例１
× V_1 × V_2
SIGN V_1 × V_2
× V_1 MULT V_2
SIGN V_1 MULT V_2

例２
V_1 ×× V_2
V_1 × SIGN V_2
V_1 MULT × V_2
V_1 MULT SIGN V_2

さいぐさ きょうすけ，まつだ としこ 日本アイ・ビー・エム

何百何千という式や他の関数を用いて組み立てられた一般化プログラムとよばれる複雑な機能をもつものも，これらの単純な原理により組み立てられたものであり，単純なルールに合致するひとつの関数にすぎないのである．ＡＰＬが原始関数についてほとんど知識のない人たちにも使われている理由は，定義関数の仕組みが非常によく利用されているからである．

1.3 ＡＰＬとＡＰＬ補助プロセッサーについて

ＡＰＬの場合，システム環境に存在するさまざまなデータおよびデータ処理資源，たとえばデータ・ベースとか，グラフィック機器の利用等は共用変数プロセッサー（ＳＶＰ）を通してＡＰＬ補助プロセッサーの助けを借りて行うように設計されている．（表２）

表２　ＡＰＬと補助プロセッサー

定義関数	補助プロセッサー	ＡＰ111…QSAM ＡＰ210…BDAM ＡＰ123…VSAM （ファイル・アクセス・メソド） ＡＰ190…ＩＭＳ等 ＡＰ124…端末全画面処理 ＡＰ126…GDDM/PGF（全画面グラフィック） ＡＰ231…ＲＳ23インターフェース ＡＰ100…サブシステムコマンド
ＡＰＬ	ＳＶＰ（共用変数プロセッサー）	

ＡＰＬの環境を作るには一定の規模の投資が必要であり，現在，一般には補助プロセッサーを利用しないＡＰＬ言語だけの小規模な利用は，ほとんどないのではないかと思われる．ＡＰＬ補助プロセッサーは共用変数プロセッサーのマクロ・プログラムを利用して，ＡＰＬユーザー側からのプロトコルを定義し，それぞれの機能をもつようにアセンブラー言語やPL/I言語を使って書くが，メーカーが提供するもの以外にユーザーが作ったものもある．

これらのことを考えると，ＡＰＬの世界が奥行きの深いものであることがおわかりいただけると思う．しかし，定義関数にしても補助プロセッサーにしても必ずしもＡＰＬ言語そのもののように統一のとれたものではなく，それぞれの設計者の意図や対象となる機能の仕組みそのものからくる複雑な要素が大きく作用する．

たとえば，GDDM/PGF により端末の全画面処理やグラフィック機能を利用するための補助プロセッサーＡＰ126においては，GDDM/PGF の190種類以上のコマンドをＡＰＬから直接 CALL する仕組みになっていて，それらのパラメータの与え方などは，ＡＰＬ言語の世界とは本質的に異なるものである．これに似たことはデータ・ファイルの入出力や印刷装置の利用等の補助プロセッサーについても，多かれ少なかれ当てはまることである．

また，これらの機能を組み合わせた適用業務を作成する場合，本来個人がプログラムの設計からコーディング，テストにいたるまですべて１人で行なうことに向いているはずのＡＰＬが，ここにきて突然個人の能力の限界を越えてしまう結果にもなる．

たとえば，時間のかかるくり返しの多い複雑な計算やデータの編集・作表・作図・モデル作りといったコンピュータの利用によって仕事の質も量も大幅に改善できる人々が企業には多数存在するが，これらの人々のコンピュータ資源の利用がより価値のあるものとするためには企業全体としての資源配分が必要で，ユーザーのばらばらな使用形態にまかせていては決して価値ある利用は望めないであろう．

すでに述べたように，ＡＰＬは定義関数による仕組みを通して，言語そのものをよりエンド・ユーザーの実情に合わせて提供することができる．その最も端的な例で，また非常によく行なわれているのが個々の目的に合わせたプログラム・パッケージを利用することであり，このために利用できるいくつかの市販のパッケージがある．しかし，目的にかなうパッケージがなかったり，ユーザーの基準に合わなかったりする場合も多く，各企業の中において，パッケージを作る能力と体制を維持できるにこしたことはない．

2. ＡＰＬ環境の利用

2.1 アァイル構築・検索のためのユーティリティー

ユーザーが自分の欲するデータを苦心して収集・加工し得たにしても，いざ使う場面になって検索や分類・集計作業が思うにまかせず，そのためのプログラム作りに精力を費して目的とする仕事の生産性が損なわれることはよくある話である．特にプログラミング作業に抵抗のある事務系統の人たちの場合などは，余計にこうしたいら立ちを訴える場合が多いのではなかろうか．かといっていちいちプログラマーの応援を求めることも不可能であるから，このようなエンド・ユーザーの人たちにも容易に自分用のファイルが構築できて，好きなように検索できるユーティリティー・プログラムが備わっていることが望ましい．

「ＡＰＬデータ検索集計作表プログラム」はこのような環境を提供するためのユーティリティー・プログラムで，冒頭に述べた「エンド・ユーザーのためのＡＰＬ利用推進」の一環として作成され，歴史的に改良を重ねて広く社内の各担当者に利用され，ＩＢＭ・ＡＰＬユーザ

ーにも解放されている．

以下その概要について紹介する．（引用文献名「ＡＰＬデータ検索集計作表プログラム」ＩＢＭマニュアル）

全体は表３のように４つのグループから成り，各グループはいくつかのユーティリティー・プログラム（ＡＰＬ）で構成される．エンド・ユーザーは，これらのユーティリティー・プログラムを使って各々の目的を達成することができる．以下に主だったプログラムの使い方の例をあげよう．（掲載例は少し古いものであるが，現在は一部改良が加えられている．）

表３　システムの概要

データ・ベースの生成・保守（グループ１）	非対話的実行モード（グループ４）
データの検索（グループ２）	
検索結果をユーザー・プログラムで操作するもの（グループ３）	

①データ・ファイルごとに検索用の索引ファイルを作成し，データ項目の名前，桁数等を登録する．(INDEX 例１)

②索引作成作業の終了したファイルに対して，データの訂正，更新等の変更を行なう．(UPDATE　例２)

③索引ファイルのでき上っているファイルのうち，検索したいファイルを指定して検索する．(SELECT　例３)

④検索結果を分類し，その結果を特定の項目について集計する．(SUMLIST　例４)

⑤検索結果をユーザー・プログラムに読み込み，操作する．（ＧＥＴ　例５）

2.2　予測プログラムへの利用

長期計画や短期計画といった計画策定の基礎となる予測作業では，実際に発生したデータの分析内容によって予測精度が左右されるといっても過言ではない．特に短

［例１］ INDEX 使用例

③ INDEX プログラムを実行する．

④ ファイル・サイズを指定する．

⑤ 索引項目について使用されるコード文字を指定する．

⑥ 項目名，索引項目の指定，項目の場所と長さを指定する．

⑦ 訂正するものを削除する．

⑧ 計算結果等に項目名を与えて同一レコード内に仮想の項目を設定する．

⑨ ⑦と同じ．

```
      INDEX '8001 D080.TEST'                                          ③
PLEASE SPECIFY THE FILE SIZE (NO. OF LOGICAL RECORDS );
0 TO LEASE THE COUNTING TO THE PROGRAM:100                            ④

***CODE CHARACTER SETS:

1) NUMERIC      :0123456789
2) ALPHABETIC   : ABCDEFGHIJKLMNOPQRSTUVWXYZ
3) ALPHANUMERIC: ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

SPECIFY BY NUMBER THE CODE CHARACTER SET FOR THIS FILE :1              ⑤
SPECIFY ANY ADDITIONAL CHARACTERS :_ABC-_____________
DEFINE THE NAME OF EACH DATA FIELD BELOW. ELSE * TO TERMINATE.(I IS INDEX YES OR
      NO.)

NO_:NAME I COL LN COL LN COL LN
  1:FLD1 Y 001 01 000 00 000 00
  2:FLD2 Y 002 04 000 00 000 00
  3:FLD3 Y 006 05 000 00 000 00
  4:FLD4 Y 010 10 000 00 000 00
  5:FLD5 N 021 20 000 00 000 00
  6:FLD4 Y 011 10 000 00 000 00                                       ⑥
  7:FLD6 Y 081 05 000 00 000 00
  8:FLD7 Y 086 05 000 00 000 00
  9:FLD8 N 091 05 000 00 000 00
 10:FLD9 N 096 05 000 00 000 00
 11:*
SPACIFY THE ENTRIES YOU WANT TO CANCEL (0 TO DESIGNATE NONE);
→:0                                                                   ⑦
NO_:NAME LN ___S__T__A__T__E__M__E__N__T_____________________________
  1:FLDX 08 8 1▼100×(▲P[FLD7]) DI ▲P[FLD8]                            ⑧
  2:*
SPECIFY THE ENTRIES YOU WANT TO CANCEL (0 TO DESIGNATE NONE);
→:0                                                                   ⑨
```

［例 2］ UPDATE 使用例

① 引用符を用いてファイル名を指定する．

② 合言葉を■■■■に入れる．

③ 索引項目も UPDATE の対象になり，それぞれ，索引項目についてはＩ，非索引項目についてはＤの見出しで区別される．

④ 同一項目へのデータ更新の区切りに使用する文字は＊○．：→εＴの７個の特殊文字の中で索引コードとして使用されていないものの１つどれを用いてもよい．

⑤ 索引項目に対してデータを更新する場合，INDEX 操作で作られるＧＧテーブルに登録されていない文字が含まれていれば更新は無効となる．

⑥ ０を指定すると UPDATE の操作は完了する．

```
      UPDATE'TEST'                                                    ①
ENTER SECURITY CODE :■■■■                                            ②
ENTER RECORD COLUMNS TO UPDATE.(0 TO QUIT.):FLD2,FLD5
ENTER UPDATING DATA FOLLOWED BY RECORD CONTROL NUMBERS. (ENTER * TO QUIT) :

**IIIDDDDDDDDDDDDDDDDDDDDD: CONTROL NUMBERS HERE                      ③
→:AAAAUPDATED RECORD 1      5 6 7 8
→:BBBB UPDATED RECORD 2     9 10 11 12
→:CCCC  UPDATED RECORD 3    13 14 15 16
→:*                                                                   ④
ENTER RECORD COLUMNS TO UPDATE.(0 TO QUIT.):FLD1
ENTER UPDATING DATA FOLLOWED BY RECORD CONTROL NUMBERS. (ENTER * TO QUIT) :

**I: CONTROLNUMBERS HERE
→:Z 1 2 3
INVALID CODE CHARACTERS USED.                                         ⑤
→:P 1 2 3
INVALID CODE CHARACTERS USED.
→:- 1 2 3
→:*                                                                   ④
ENTER RECORD COLUMNS TO UPDATE.(0 TO QUIT.):0                         ⑥
```

［例 3］ SELECT 使用例

③ SELECT プログラムを実行する．

④ 選択条件を指定する．

⑤ ④と⑤は論理演算を∧（ＡＮＤ）で結ばれている．

⑥ 選択を継続する場合はＹ（ＹＥＳ）を与える．

```
      SELECT 'TEST'                                                   ③
ENTER SECURITY CODE :■■■■
SPECIFY COLUMN(S) : FLD1                                              ④
SPECIFY VALUE(S), FOLLOWING =≠<≤>OR≥ : ≥01
SPECIFY * TO TERMINATE, ELSE ANY APL LOGICAL OPERATOR TO CONTINUE : ∧
SPECIFY COLUMN(S) : FLD2                                              ⑤
SPECIFY VALUE(S), FOLLOWING =≠<≤>OR≥ : ≤AAAA
SPECIFY * TO TERMINATE, ELSE ANY APL LOGICAL OPERATOR TO CONTINUE : *
TOTAL OF  1 RECORDS HAVE BEEN SELECTED.
ANY FURTHER SELECTION ? Y OR N :N                                     ⑥
```

［例 4］ SUM LIST 使用例

① 集計する数値項目名を引用符を用いて指定する．

② 集計の結果は例にあるように制御項目上のコード，レコード数，各数値項目の集計の順に並ぶ．

```
      SORT ON FLD1
SORT START  : 2/8/77 02:16
END OF SORT : 2/8/77 02:16

      SUMLIST 'FLD6,FLD7,FLD8,FLD9'                                   ①
2/8/77 02:17   SUMLIST START
     0              10        40996       18884       48109        -9634
     1               5        27758       27608        1254        -4115
     2               3         8210       11677        7278        -2147
     3               3         9688        -312       12772        -4028
     4               8        23315        3706       38467       36181
     5               8        31430       20610       17302       27478      ②
     6               6       -10551       32701       26205        9917
     7               3         2974        4763       -8647        8079
     8              10        34348        5758       41096       26421
     9              11        28136       45958       38018       -6038
TOTAL****           67       196304      171353      221854      101382
2/8/77 02:17   SUMLIST END
```

［例5］ GET使用例

GETは SELECT—GET. SELECT—SORT—GET という組合せで用いられ，選択されたレコードを1つずつ読み込む．

たとえば，読み込んだデータをリストする機能として備わっている LISTはほぼ次のようなプログラムになっている．

```
    ∇LIST Q
[1] A:⍕(~∧/'*'=10↑REC←GET)/'→A,ρ□←REC[Q]'
    ∇
```

または，

```
    ∇LIST Q
[1] A:REC←GET
[2]   →(∧/'*'=10↑REC)/0
[3]   □←REC[Q]
[4]   →A
    ∇
```

すなわちGETは実行されるごとにレコードを1つずつ読み，最後のレコードまで読み切ると，頭に10個の＊をもつレコードを発生させてEOF（ファイルの終り）を知らせる．上記のサンプルプログラムを変形させて数値項目を集計する場合，たとえば次のようにすればよい．

```
    ∇ADD
[1]   CTR←0
[2] A:REC←GET
[3]   →(∧/'*'=10↑REC)/E
[4]   CTR←CTR+⍕REC[BAMT]
[5]   →A
[6] E:CTR
    ∇
```

または，

```
    ∇ADD
[1]   CTR←0
[2] A:⍕(~∧/'*'=10↑REC←GET)/'→A,ρCTR←CTR+⍕REC[BAMT]'
[3]   CTR
    ∇
```

期予測の場合などは，より多様な観点から収集されたデータを分析した結果による積上げが説得力をもつことが多い．したがって，より豊富なデータと検索・加工能力が予測担当者の仕事の効率および質におよぼす影響度は大きく，2.1で述べたようなユーティリティー・プログラムの存在は，作業担当者にとって大きな福音をもたらすものである．また1人の担当者が使用するデータは，ほとんどの場合担当者固有のデータではなく，同一部門あるいは他部門の人にとっても共通に使用され，各々の目的に応じて加工の形態が異なるという性格をもつ．したがって企業全体として集中的に保有・整備すべきデータ・ベースと，特定の人の間で共同利用される一部データ・グループのデザインをうまく行ない，データへのアクセスのためのインターフェースを容易にすることによってコンピュータ資源の節約をはかり，ユーザーの便益性を高めることが重要である．先に述べたファイル構築・検索のためのプログラムも，このような環境下にあってより有効性が高められるものである．

たとえば，ある特定の製品を使用している顧客について，その特性を製品の利用形態および企業規模から分析しようとする場合を考える．製品関連データは製品ファイルに，企業関連データは企業ファイルに入っているものとする．（図1）

製品中心に仕事をしているユーザーAは，日常製品ファイルを使用し，企業中心に仕事をしているユーザーBは，日常企業ファイルをよく使うわけであるが，両方のファイルを使う立場のユーザーCは，使用のつど両方のファイルを合成しなければならず回を重ねるにつれて繁雑な思いをするであろう．そこで必要部分をある特定の領域に貯えて軽いデータ・ファイルとし，そのファイルを使用すれば余分な操作を経ずに済むし，特定の領域を何人かで共有できる領域とすれば，自分のワーク・スペ

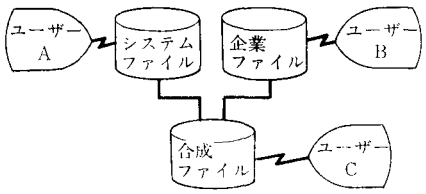

図 1　ファイルとユーザーの関係

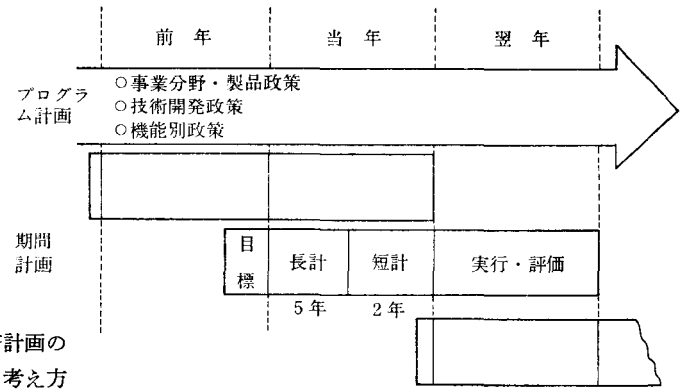

図 2　経済計画の体系と考え方

ースも影響されずに済むという便益性を得ることができる．

以上は予測作業にともなう実績値データの検索部分にまつわる考察であるが，2.1のようなユーティリティー・プログラムを実行させながら途中である条件式等を設定して，その条件式に見合う検索を引続き実行するといった場合にも，ＡＰＬのステートメントを挿入することによってスムーズに作業を継続させることができる．

次に将来の予測を行なう場面におけるＡＰＬの効用はどうか？　ひと口に予測といってもその対象は多岐にわたり，また，いわゆるマクロ的な分野から業界そして各企業へとその領域も多様であるが，ここではその一例として，当社の製品政策にまつわる製品（コンピュータ本体およびその周辺機器，ソフトウェアなど）の販売予測の例をとりあげてみる．

図2は当社の経営計画体系を概念的に示したものである．各国ＩＢＭの製品政策にしたがって，日本，米国，西独，仏などの主要国ＩＢＭの20有余の開発研究所が分担，開発を行なう．こうした製品政策の立案・評価にさいしては，製品ごとの販売予測が重要な鍵になる．

計画立案プロセスは長期計画を基本とし，その後定常サイクルにしたがって製品政策等の見直しによる短期計画が作成されるが，長期・短期計画とも製品の販売予測が計画策定の基礎となる．

この場合，“いつ・どのような製品を・いくらで・発表し・出荷できるか”といった計画項目の各々が，予測のための重要な前提となり，これらの前提の中のいずれかが変更になった場合の予測の変化への対応や，製品が単一でなく種々の製品が複雑に組み合わさった場合の対応，客先の利用形態が賃貸から買取りに変わった場合等，種々の変化に対応するには応答の速い柔軟なコンピュータ・システムの利用が必要で，ＡＰＬはこのような条件にかなった言語といえる．さらに上記のような変更は定型的なパターンとして捉えることはむずかしい場合が多く，そのことがモデル化をむずかしくしている要素が強いわけであるが，逆に真のエンド・ユーザーとしての予測担当者にはその場その場の問題を解決し，切り抜けるための格好の道具としてＡＰＬが重宝であるという感触は否めないと思われる．多少逆説めいてしまうが，その場かぎりの一過性のものであるならば，プログラムそのものを他者に知らしめる必要性はないわけで，他人には複雑きわまりないと思われるステートメントであったとしてもエンド・ユーザーにとって最も効率のよいステートメントであるならばその効用は大きく，その意味で，真のエンド・ユーザー向きの言語としての機能的な便利さをＡＰＬは備えているといえる．以下そのいくつかについて検討してみよう．

行列間の操作

製品ごとの売上高を計算する場合の基本的な形は図3のような行列要素間の操作 $(s(i,j)=u(i,j)\times p(i,j))$ であるが，これをＡＰＬで表現すると，

①…$S\leftarrow U \times P$

と一度に計算させることができる．

製品が中央機種の場合には，付属する周辺機器等を図4のように寄せ集めることになるので，実際の売上高は関連製品の合計として計算される．関連製品が製品ごとに3次元表現で記憶されているものとして，各製品ごとに年別に関連製品を加えたものを製品本体にたし込む．

②…$S\leftarrow S1++/[2]\ S2$

$S2$ の横軸にそって縦軸にたし上げたものを $S1$ の横軸にたし込む

このように数量が予測され，単価が与えられれば，その後の計算は至極単純なステートメントで実行できることがおわかりいただけると思う．

行列の追加・削除

製品の種類は，計画のサイクルごとに変動がおこり得る．すなわち製品政策にしたがって製品の追加，削除の必要性がおこり，そのたびごとに当該製品コードおよびデータを追加，削除するもので，そのような保守も容易

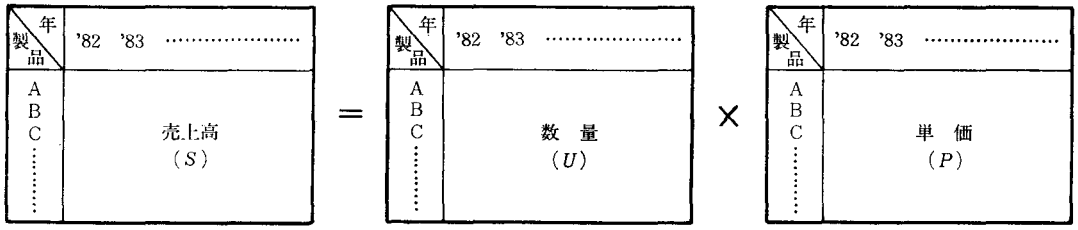

図 3 行列の操作

に行なうことができる．たとえば，6行目に新しいコードを入れるために，6行以下を1行ずらして挿入する場合のAPLステートメントを考える．M種類の製品コードが行列Yに記憶されているものとすると，

③…$Y\leftarrow Y$ [⍳5;], [1] 'NEW', [1] A [10+(⍳(M−5))−5;]

⍳5：1から5まで指標をまわす

により新コード'NEW'を挿入した製品コードになる．

次に，この新コードに対応したデータ（説明を簡略化するために売上高とする）を挿入して，各製品コードとの対応を見るために製品コード行列と売上高の行列を連結して表示させる場合を考える．（図5）

プログラム手順は売上高行列に新製品のデータX（1行分）を6行目に加え(④)，製品コードと売上高行列を連結させる(⑤)．

④…$S\leftarrow S$ [⍳5;], [1] X, [1] S [10+(⍳(M−5))−5;]

新データおよび6行以後を縦軸に連結

⑤…$Z\leftarrow Y$, ⍪ S

Sの数値データを文字化して連結

命令内の複合演算および条件付分岐

いったん予測された売上高について，向こうN年間の平均の伸び率が目標とする伸び率に達しているかどうかによって，結果の良し悪しを判定する場合がおこってくる．ある製品グループ（6行から10行）の売上高合計がN年間で年々10%以上の伸び率にならなければ再試行する場合を考えよう．

⑥…$T\leftarrow$ +/[1] S [5+⍳5;]

合計が T [1]~T [N+1] に入る．

CGR←100×(((T[N+1]÷T[1])*÷N)−1)

→(CGR≥10)/0

RETRY

再試行プログラムのよび出し

CGR（平均伸び率）の計算は数行に分けてもよいし，上記のようにまとめてもよく，どちらにするかはその時の都合による．

すでに定義されているプログラムの使用

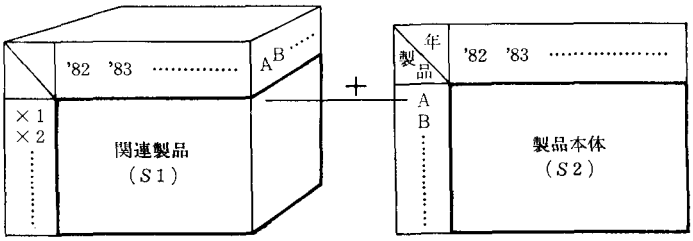

図 4 製品の連結

S1[I:K] I：製品の種類の数
K：年次
S2[I:J:K] J：関連製品の種類の数

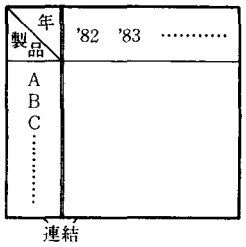

図 5 製品コードと行列データの連結

頻繁に使われる操作プログラムを一度定義して自分のワークスペース内に記憶させておくと，使用のつど定義プログラムの名前をよぶだけで計算を実行させることができる．一例として超小型機種のように値段が安くて数量の多い製品の販売予測の場合はある程度モデル化して確率的に予測する試みも可能である．そのさいによく用いられるパレートモデルの原型である対数正規分布曲線を定義関数として記憶させ，値段の平均値，バラツキをかえて試行錯誤させるケースを考えてみる．定義式は下記にしたがう．

$$f(x)=\frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{(Ln(x)-u)^2}{2\sigma^2}}$$

ここで $u=\sum f(x)\cdot Ln(x)/\sum f(x)$

$\sigma^2=\sum f(x)\cdot(Ln(x)-u)^2/\sum f(x)-1$

$c=e^{2\sigma^2}$ とすると，

平均値 $m=e^{u+1/2\sigma^2}=m_0c^{3/4}$

中央値 $m_e=e^u \quad =mc^{-1/4}$

最頻値 $m_0=e^{u-\sigma^2} \quad =mc^{-3/4}$

分散 $S^2=m(e^{\sigma^2}-1)$

によって $f(x)$ は次のように表現される．

$$f(x)=\frac{1}{m_0c^{1/4}\sqrt{\pi\,Ln(c)}}e^{-\frac{(Ln(x)-Ln(m_0))^2}{Ln(c)}}$$

⑦…LNDTR

上記計算プログラム式をよび出す

m, cを与えてシミュレーション

'INPUT : MEAN, C'

プログラムおよび関数形のプロット図を図6に示す.

以上，担当者にとって日常重宝していると思われるＡＰＬ機能に焦点を当てながら解説を行なってきたが，予測作業そのものは，もっと複雑な様相を呈しており，たとえば従来使用している機種から新機種へ切り換える顧客，あるいは追加導入する顧客の他，はじめて導入する顧客，他社製品に移行したり中断したりする顧客，端末をはじめに導入してだんだん規模を大きくしていく顧客等々の諸現象に対して，予測担当者のセンスとか判断力に負う部分も多い．また，個々の製品予測が集大成された後で外部環境等の状況判断の変化によっていく度か修正したあげく，最終的にトップの意思決定によりある線に落ち着いてから，作業結果の事後調整を行なうといった側面もあり，なかなかに苦労の多い仕事である．それゆえに一般的な予測プログラムという形でのプログラム・ステートメントの紹介を行なわなかったが，このことはむしろエンド・ユーザー個々人が自由に表現し得るプログラミング言語の必要性を意味し，その必要性のもとにＡＰＬがしかるべき体制によるサポートを得て日常よく利用されているという現実は，ＡＰＬの効用を端的に物語っているといえよう．

またデータの分析結果や予測結果をわかりやすいグラフによって図示することも，この種の作業においては重要な要素であり，そのためのエイドもいろいろと用意，提供されているが，本稿の紙幅の関係で割愛した.

```
▽LNDTR[□]▽
      ▽ LNDTR
[1]    'INPUT : MEAN,C
[2]    Y←100⍴0
[3]    Z←100⍴0
[4]    XXX[1]←0
[5]    XXX[4]←1000
[6]    XXX[1+⍳2]←□
[7]    I←0
[8]    I←I+1
[9]    XXX[1]←XXX[1]+10
[10]   Y[I]←XXX[1]
[11]   PM←XXX[2]×XXX[3]*(¯0.75)
[12]   C←XXX[3]
[13]   X←(⍵(Y[I]÷(PM×C*0.5)))÷(⍵C*0.5)*0.5
[14]   A← 0.3193815 ¯0.3565638 1.781478 ¯1.821256 1.330274
[15]   ABSX←|X
[16]   W←(1÷1+0.2316419×ABSX)*⍳5
[17]   FX←0.3989423×*(-X×X÷2)
[18]   PX←1-FX×+/A×W
[19]   →(X≥0)/OUT
[20]   PX←1-PX
[21]  OUT:
[22]   FX←FX÷(XXX[1]×(⍵C×0.5)*0.5)
[23]   Z[I]←FX×1000
[24]   →(XXX[1]<XXX[4])/8
      ▽
       LNDTR
INPUT : MEAN,C
□:
      350,3

 X←Z
       60 PLOT X VS Y
```

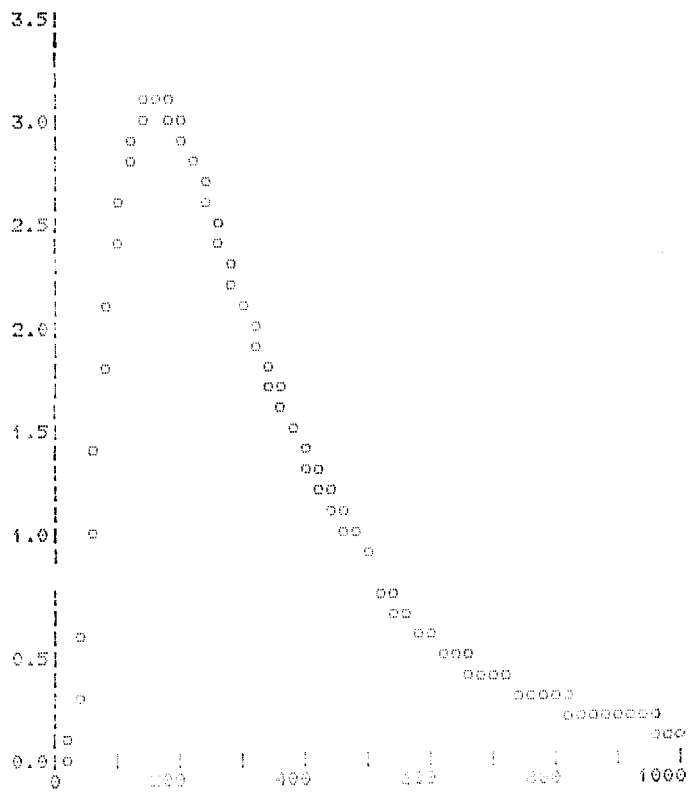

図 6