データベース・システム 45－2
（1985. 1. 22）

階層型データベース再構成に伴なう新旧ファイル間データ転送プログラムの自動コーディング

磯本征雄，（名市大・計算センター）
溝口理一郎，角所　収，（大阪大・産研）
石桁正士（大阪電通大・工）
竹内　憲（日本電気）

あらまし：　データベース論理設計の際には、最終の論理構造に至るまでの過程において、しばしば試行錯誤による再構成が繰り返される。　本論文では、このような再構成に伴なう新旧データベース・ファイル間データ転送プログラムの自動コーディングの実現手法について述べる。　再構成に関する宣言的知識はＰＲＯＬＯＧ言語で記述し、また自動コーディングの手続き的知識はＦＯＲＴＲＡＮ言語で記述して、全体としてひとつのシステムにまとめ上げる。　ここでの議論は、論理的に明確な処理の自動化（システム化）の実現によって、日々に重要度を増す各分野でのデータベースの構築・管理のための知的支援をめざす試みのひとつである。

１．まえがき

データベース論理設計は、データ記述言語の文法だけでなくデータの値にも関連した複雑な内容を持っている。[1)]　ところがデータベース再構成は、新旧の両論理構造を与えた後の閉じた世界での議論であり、その内訳を目的に合わせて整理することによって、様々の操作手続きを定式化できる。　本論文では、その可能性のひとつとして、データベース再構成に伴って必要となる旧データベースから新データベースへのデータ転送プログラムのコーディングの自動化を試みる。

データベース論理設計では、当然初期の段階から完璧な論理構造にしておくことが望ましい。　ところが現実には、多少の試行錯誤による諸要因の取捨選択により、データベース再構成を繰り返しながら決められることの方が多い。　本論文の直接の目的は、この様なデータベース再構成に伴う新旧サブ・ファイル間のデータ転送プログラムのコーディングを自動化し、データベース管理者（ＤＢＡ）のスキーマ設計・改善が円滑にすすむように支援することである。

さて、この自動コーディングのシステム化にあたって、データベース論理構造の解釈やデータベース再構成の内訳については、これらを規則（ＲＵＬＥ）の形式でまとめるのが望ましい。　そこで、論理構造に関する諸規則の宣言的記述に適した知識はＰＲＯＬＯＧ言語で表現する。　一方、自動コーディング自体は処理手続きであり、規則よりはむしろ手続きとしてまとめたほうがよい。　そこでシステム全体の制御及び自動コーディングに関する手続き的記述の適した知識は、ＦＯＲＴＲＡＮ言語で記述した。　このような議論は単にシステムの実用化の面だけで無く、人工知能やソフトウェア工学における自動プログラミングなどとの関連においても興味深い課題である。[2)]

ところで一般に、計算機プログラムの自動コーディングを実用システムとして実現するには、ジョブ・デック編成法、ファイル呼出し法、ジョブ実行手順など、自動コーディングされたプログラムが処理される周辺の環境整備と様々な機能の標準化が必要である。この点において本論文では、既に完成しているデータベース構築・管理支援システムＫＤＢＭＳの諸機能を基盤にして考える。[3)]　またインプリメンテーションについては、汎用ＤＢＭＳである準関係型のＩＮＱを支援対象にした。[4)]

２．問題提起

本論文におけるデータベース再構成の意味を明瞭にするために、まずは具体例をもとに

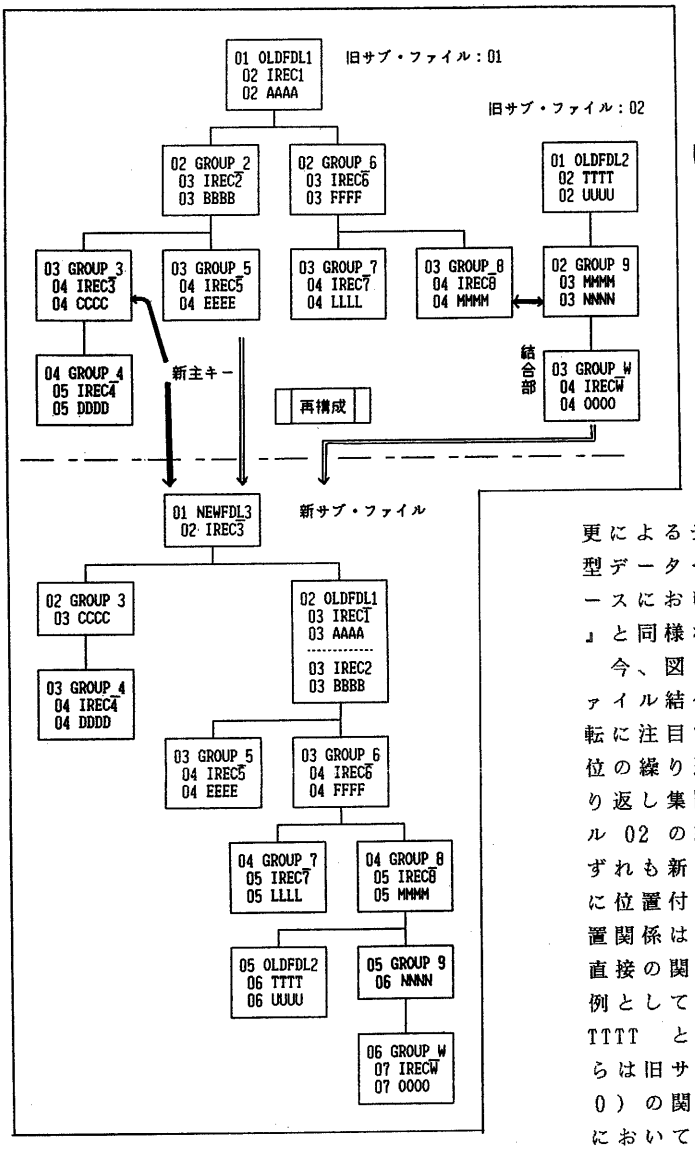

図２．１　データベース再構成の実例
図の中の上側のサブ・ファイル 01 と 02 は、主キーの変更およびファイルの結合によって、下側のサブ・ファイルに再構成される。　各繰り返し集団（四角の枠の中）の第１番目が繰り返し集団名である。　第２番目以後は、基本項目名である。　数字は、レベル番号を示す。
IREC3 は、再構成後に主キーとなる。
２つのサブ・ファイルは、基本項目 MMMMM　で結合されている。

説明する。

図２．１で、上側に２つの旧サブ・ファイルの論理構造が示され、これらを基に下側の新サブ・ファイルが再構成される。　再構成の第１点は、旧サブ・ファイル 01 の左側にある第３階層目の繰り返し集団 GROUP-3の基本項目 IREC3　が新サブ・ファイルで主キーに成っていることである。　第２点は、旧サブ・ファイル 01 と02　に共通に存在する基本項目　MMMM　を結合部として２つの旧サブ・ファイルを結合していることである。　このようなサブ・ファイルの結合や主キーの変更によるデータベース再構成には、単に階層型データベースだけでなく、関係型データベースにおける『関数依存性』や『多値依存性』と同様な問題が発生する。

今、図２．１において、主キーの変更やファイル結合の結果として起こる親子関係の逆転に注目する。　サブ・ファイル 01 の最上位の繰り返し集団　OLDFDL1 や、その下の繰り返し集団　GROUP-2 、およびサブ・ファイル 02 の繰り返し集団　OLDFDL2 などは、いずれも新サブ・ファイルではさらにその下位に位置付けられている。　これらの相互の位置関係は、元々存在していたデータ項目間の直接の関係をもとに再構成されている。　一例として、旧サブ・ファイル 02 の基本項目 TTTT　と MMMM の関係に注目しよう。　これらは旧サブ・ファイル内では　１：ｎ（ｎ＞０）の関係にある。　ところが再構成の過程において　MMMM　の或るひとつの値に対応する親の繰り返し集団内の基本項目 TTTT のすべての値を集めて　MMMM　に関係付ける。　その対応によって親子関係は逆転し、ｍ：１（ｍ＞０）に変化する。　当然これに関連して、論理構造の全体的な変化も起こりうる。　データベース再構成とは、このような関係の変更である。

３．階層構造の表現法

本章では、データベース再構成に関わる諸規則をまとめていくうえで、表記上必要となる基本的事項の準備を行なう。

一般に階層型データベース論理構造は、”データ項目名とその属性”、”繰り返し集団とそれに属する基本項目”、そして”繰り返し集団間の親子関係”に依って表現される。しかし概念スキーマに於けるこれらの記述方法は個々のＤＢＭＳによって異なる。　そこで以下においてデータベース再構成の議論を一般的に展開可能にするために、そしてまた支援システムとしての実現を容易にするために、ここではＰＲＯＬＯＧ言語を用いて、特定のＤＢＭＳに依らない記述法で定義を試みる。

データ項目名とその属性

概念スキーマの中の各々のデータ項目について、項目名、レベル番号、データ・タイプ及び属性（データ長やキー項目の是非など）を次の述語で定義する。

ITEM (*FN,*NO,*LEVEL,*DNAME,*DTYPE,
*ATTRIBUTE).

意味：　サブ・ファイル番号 *FNの行数第*NO行目の項目（ITEM）は、レベル番号 *LEVEL にあるデータ項目名 *DNAME である。　そしてデータ・タイプ *DTYPE でかつ属性　*ATTRIBUTE　である。

INQ の場合には、例えば次のとうりである。

*NO	*LEVEL	*DNAME	*DTYPE	*ATTRIBUTE
1	02	IREC1	X(4)	KEY
2	02	AAAA	X(5)	
3	02	GROUP-2	(N)	
4	03	IREC2	X(9)	

*DTYPEは、基本項目と繰り返し集団を区別して、さらにデータの型、データ長などの属性を与える。　*ATTRIBUTE　には文字列が対応付けられ、検索時にキー項目として扱えるか否かについての情報などその他のものをふくめることが出来る。　ここで以下も含め、特殊文字＊で始まる文字列は、ＰＲＯＬＯＧの変数名である。

繰り返し集団と基本項目の関係

繰り返し集団とそれに属す個々の基本項目との関係を次の述語で定義する。

IS-ELEMENT-OF(*FN,*LEVEL,*DNAME,*GNAME).

意味：　サブ・ファイル番号 *FN　のレベル番号 *LEVEL にある基本項目　*DNAME は、繰り返し集団 *GNAME の要素である

繰り返し集団間の親子関係

階層構造における繰り返し集団の間の親子関係を次の述語で定義する。

IS-CHILD-OF(*FN,*GNAME,*PARENT).

意味：　サブ・ファイル番号 *FN　にある繰り返し集団　*GNAMEは、繰り返し集団 *PARENT の子である。

これらの３つの述語は、階層型データベースの論理構造記述の基本をなす。　ここで、IS-ELEMENT-OF と IS-CHILD-OF　は、いずれも個々のＤＢＭＳに依存して、論理構造記述のなかで暗黙に与えられる場合と明示される場合があるが、ここでは直接に問題にはしない。　ただし、ＩＮＱに即していえば、行番号　*NO を使って論理構造の中の記述の順番をもとに、いずれも述語 ITEM から決められる。　この決定規則（ＲＵＬＥ）は、ＰＲＯＬＯＧで記述されるが、その内訳の大部分がＩＮＱのデータ記述言語の文法のみに依存するので、ここでは説明を割愛する。

４．繰り返し集団間のデータ格納順位

新サブ・ファイルへのデータ格納は、データベース再構成に伴う必要な作業のひとつである。　その際のデータの格納順序は、最上位より逐次下位方向に、繰り返し集団単位で次の５規則に従って行われる（図４．１参照）：

規則０；最上位の基本項目群を最初に格納する。
規則１；親から子の順に格納する。
規則２；子の格納を終われば、未格納の同列の繰り返し集団を格納する。
規則３；同列繰り返し集団がすべて格納を

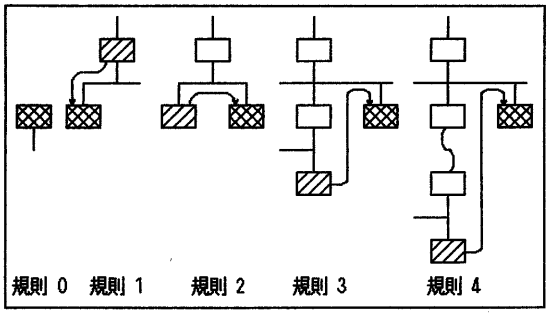

図 4.1　格納順序を決める規則

終了しておれば、その親と同列の未格納の繰り返し集団を格納する

規則４；親の同列もすべて格納を終了しておれば、更にその上の親と同列の未格納繰り返し集団を格納する。

この内で規則４は、ＰＲＯＬＯＧ表現の中では再帰的に記述される。

これら新サブ・ファイルのデータ格納順位は、次の述語に依って記述される。

```
LOAD-SEQ(*SEQNO,*GNAME).
```

この述語は、新サブ・ファイルにおいて第 *SEQNO 番目に格納される繰り返し集団は、*GNAMEであることを表している。　図４．２は、上記の規則０～４をＰＲＯＬＯＧ言語で表したものである。　述語 LOAD-SEQ は、データ転送プログラムの自動コーディングの際に、データ入出力の繰り返し集団間の相対的な順序と関連して、実行順位の制御のための GO TO文の文番号の決定に使われる。

５．データベース再構成の形態

データベース再構成の手順をすべて事実（ＦＡＣＴ）と規則（ＲＵＬＥ）で表現可能にするためには、”再構成”の内容を具体的に定める必要がある。　特に、旧サブ・ファイル群からのデータ転送の方法を一意に定めるためには、しかるべき標準化に伴う或る種の制約が課せられる。　そのために、ここでは次の３つの再構成の形態を仮定することにした；

（１）主キー（レコードを同定する項目）の変更。

（２）サブ・ファイルの結合。

```
/*  ******  (4) RULES FOR THE LOADING SEQUENCE  ******  */
ASSERT_LOADSEQ(*N):-ASSERTA(LOAD_SEQ(1,ROOT)),
          STEP_NO(*SEQNO),SEQUENCE_RULE(*N,*SEQNO).
/*  (4.1) -----  */
SEQUENCE_RULE(*N,*SEQNO):--(*SQ,*SEQNO,1),LOAD_SEQ(*SQ,*G),
  NEXT_GROUP(*N,*L,*G,*LEVEL,*GNAME),
  ASSERTA(LOAD_SEQ(*SEQNO,*GNAME)),!,FAIL.
SEQUENCE_RULE(*N,*SEQNO):--(*SQ,*SEQNO,1),
  DISPLAY("THE LAST GROUP IS AT ",*SQ).
/*  (4.3)  SEARCH THE NEXT GROUP NAME  */
NEXT_GROUP(*N,*L,*G,*LEVEL,*GNAME):-          /* RULE LOAD1 */
  +(*LEVEL,*L,1),
  IS_CHILD_OF(*N,*LEVEL,*GNAME,*G),
  SCAN(*N,*LEVEL,*GNAME).
NEXT_GROUP(*N,*LEVEL,*G,*LEVEL,*GNAME):-      /* RULE LOAD2 */
  IS_CHILD_OF(*N,*LEVEL,*G,*PARENT),
  IS_CHILD_OF(*N,*LEVEL,*GNAME,*PARENT),
  =/=(*PARENT,*GNAME),SCAN(*N,*LEVEL,*GNAME).
NEXT_GROUP(*N,*L,*G,*LEVEL,*GNAME):-=>(*L,3),  /* RULE LOAD3 */
  -(*LEVEL,*L,1),IS_CHILD_OF(*N,*L,*G,*GT),
  IS_CHILD_OF(*N,*LEVEL,*GT,*PARENT),
  IS_CHILD_OF(*N,*LEVEL,*GNAME,*PARENT),
  =/=(*GT,*GNAME),SCAN(*N,*LEVEL,*GNAME).
NEXT_GROUP(*N,*L,*G,*LEVEL,*GNAME):-=>(*L,4),  /* RULE LOAD4 */
  -(*LT,*L,1),IS_CHILD_OF(*N,*L,*G,*GT),
  NEXT_GROUP(*N,*LT,*GT,*LEVEL,*GNAME).
/*   (4.2)  SCANNING OVER NEW ITEMS  */
     SCAN(*N,*LEVEL,*GNAME):-LOAD_SEQ(_,*GNAME),!,FAIL.
     SCAN(*N,*LEVEL,*GNAME):-!.
```

図４．２　格納順序を決める規則のＰＲＯＬＯＧによる記述

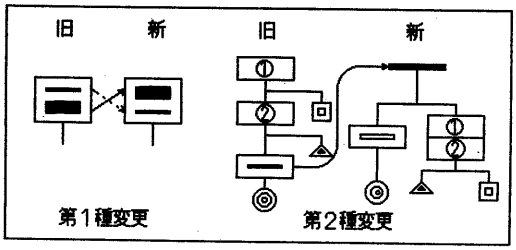

図 5.1　主キーの変更

（3）基本項目の繰り返し集団をまたがる移動。

以下で、個々の再構成の形態について述べる

5．1　主キーの変更

今我々の想定しているＤＢＭＳでは、最上位にある基本項目のひとつをレコード・セットを同定する項目とし、その値はデータベース中に唯ひとつになるように定め、これを主キーと呼ぶ。　当然、主キーは検索の結果として取り出されたレコードを同定する基本項目であり、これを見つけるための条件を与える他の基本項目を検索項目と呼んで、多くの場合主キーの諸属性を与える。　ここで言う主キーの変更とは、この様な主キーと検索項目の役割関係を入れ換えることである。

ここで、主キーの変更に対して、次の述語を定義する。

PRIMARY(*PTYPE,*FN,*DNAME,*GNAME).

意味：　新サブ・ファイルの主キーは、変更形式 *PTYPE によって、旧サブ・ファイル番号*FN の繰り返し集団 *GNAME 中の基本項目 *DNAME を割り当てるものとする。

ただし簡単のために、主キーはサブ・ファイルごとにひとつとする。　新主キーが旧サブ・ファイルの最上位の項目群から選ばれた場合を、第１種変更と呼び *PTYPE=TYPE1 とする。　また第２階層以下の繰り返し集団から新主キーを選んだ場合を、第２種変更とよび *PTYP=TYPE2 とする（図５．１参照）。

第１種変更は、関係型データベースにおけるテーブル内の項目の組み換えに相当する。表５．１は、植物の実（み）とその色の関係を示す。　左側の表は、前者が主キーで後者が属性である。　変更後の右側の表では、色を主キーにし、実（み）を属性にした関係になっている。　これは、関係型データベースにおける正規形と同じ議論である。　このように、第１種変更はバックマン図で示せば見掛け上の変化はほとんどないが、実際のデータを眺めると大きく変更されている。

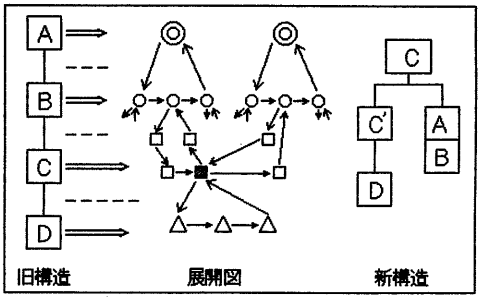

図5.2 第2種変更の状況

第２種変更では、図５．１の右側に示したように、階層型の論理構造自体に変更が加わる。　図５．２の展開図を仲介に再構成の様子を説明しよう。　図５．２で、旧サブ・ファイルの繰り返し集団Ｃの中の基本項目ｃに注目する。　ｃの特定値を持つレコードを集めてこれを束ねると、ｃに対するＣの中のｃ以外の基本項目との対応は

１：ｎ　（　ｎ＞０　）

と成る。　この様にして新たにｃを主キーに選んだならば、その結果として親子関係が変化し、Ｃからｃを除いたＣ’が主キーの直下にくる繰り返し集団を形成することになる。上位にあった繰り返し集団ＡとＢは、やはりｃに対して

１：ｍ　（　ｍ＞０　）

表５．１　果実と色の関係表

果実	色
リンゴ	赤、黄、緑
ミカン	黄、緑
カ　キ	柿色
ナ　シ	褐色
イチゴ	赤、緑
バナナ	黄

色	果実
赤	リンゴ、イチゴ
黄	リンゴ、ミカン、バナナ
緑	リンゴ、ミカン、イチゴ
柿色	カキ
褐色	ナシ
・	

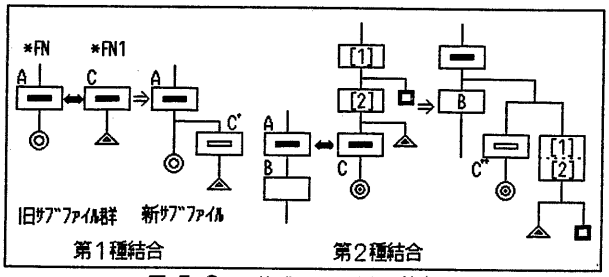

図 5.3　サブ・ファイルの結合

表5．2　＊ＣＴＹＰＥの値とその意味

＊CTYPE	意　味
TYPE1	第１種結合形である.
TYPE2	第２種結合形である.
INVERSE	第２種結合又は第２種変更に伴ない，親子関係に逆転が起っている.
SKIP	繰り返し集団の削除がある.
NORMAL	上記以外，親子関係の変化もない.

と成り、冗長度が大きく成る事を許して、安全度の高い図５．２の右側の構造を取る事にした。

５．２　サブ・ファイルの結合

ここでは、概念スキーマは幾つかの物理的に独立したサブ・ファイルの集まりであるとする。　（ただし、外部スキーマでの統合した利用法は可能ではあるが、ここでは直接関係ないので省略する）。　サブ・ファイルの結合とは、この様に独立に作られた構造を物理構造の上でもひとつに統合する事である。この再構成の結果、ひとつの論理構造が新たに作られることになる。

ここでは、　サブ・ファイルの結合を、次の述語で定義する。

```
LINK-TO(*SEQNO,*CTYPE,*DNAME,
                *FN,*GNAME,*FM,*PARENT).
```

意味：　新サブ・ファイルの格納順位 *SEQNO番目の繰り返し集団は、旧サブ・ファイル番号 *FNの繰り返し集団*GNAMEに対応する。　また、*GNAMEの親は、旧サブ・ファイル番号*FM の繰り返し集団 *PARENT と結合形式 *CTYPE で基本項目 *DNAME　をキーとして結合される。

サブ・ファイルの結合の場合にも、前節の主キーの変更の場合と同じく”親子関係”や”主キーと属性の関係”の変更が付随的におこる。　従って、主キーの変更と対応させると、サブ・ファイルの結合形式も次の２種類に分けて考えられる（図５．３参照）：

第１種結合　（＊ＣＴＹＰＥ＝ＴＹＰＥ１）
第２種結合　（＊ＣＴＹＰＥ＝ＴＹＰＥ２）

この場合、結合された基本項目は親側の繰り返し集団に吸収され、子側の繰り返し集団からは取り除かれる（図２．１参照）。

ここで定義する述語　LINK-TO は、単に旧サブ・ファイル間の結合関係だけでなく再構成の際の親子関係の新旧サブ・ファイル間での対応を示す述語としてそのまま拡張解釈できる。　表５．２は、*CTYPEの値と旧サブ・ファイルから新サブ・ファイルへの再構成の形態の関係を示す。　また図５．４は、*CTYPE=NORMAL の場合の LINK-TOを定める規則の実例を示す。

５．３　繰り返し集団を跨る基本項目の移動

前記２節は、繰り返し集団間の親子関係の変更に対する再構成であった。　ところが、ある基本項目が、再構成後には異なる繰り返し集団に移ることも有り得る。　このような再構成に対して、以下に述べるように、基本項目の値の意味は問わず、手順のうえで可能な変更ならば許すことにした。

```
LINK_TO(*SEQNO,NORMAL,_,*FN,*GNAME,*FN,*PARENT)
  :-LOAD SEQ(*SEQNO,*GNAME),
    IS_CHILD_OF(0,_,*GNAME,*PARENT),
    LOAD_SEQ(*SQ,*PARENT),
    LINK_TO(*SQ,NORMAL,_,*FN,*PARENT,_,_),
    IS_CHILD_OF(*FN,_,*GNAME,*PARENT).
```

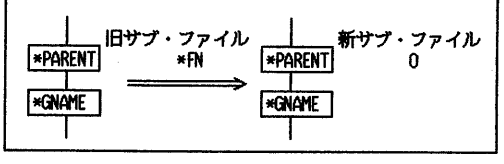

図5．４　＊ＣＴＹＰＥＴがＮＯＲＭＡＬの場合のＬＩＮＫ－ＴＯを決める規則（ＲＵＬＥ）。ここで、サブ・ファイル番号０は、新サブ・ファイルの番号である。

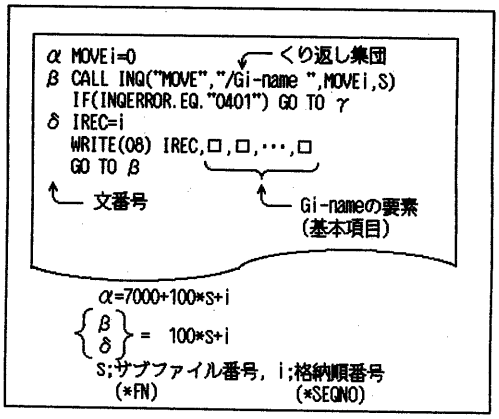

図6.1 データ転送プログラムの基本要素

基本項目の値は、それの属する繰り返し集団が呼び出された時点で初めて与えられる。したがって、再構成の際の基本項目の繰り返し集団間での位置の変更は、次の3規則を充たす場合にのみ許される：

規則1：同じ繰り返し集団に留まる。
規則2：同じ旧サブ・ファイルに属する下位の繰り返し集団に移る。
規則3：異なる旧サブ・ファイルで、結合再構成後の下位の繰り返し集団に移る。

これらの規則は、データの値の意味よりはむしろデータ操作言語によるデータ転送プログラムのコーディングにおける現実的処置に合わせてさだめられた。

6. データ転送プログラムのコーディング規則

データ転送プログラムの自動コーディングには、新しいデータベース・ファイルへのデータ格納順位（第4章参照）や新旧データベース・ファイル間の再構成の形態（第5章参照）の諸規則を、自動コーディングの手続き的な表現と結びつける必要がある。すなわち旧データベース・ファイルからデータを読み込み、これを新データベース・ファイルにデータ転送するために、論理構造の親子関係に合わせたデータ入出力操作を計算機プログラムの形式で表現する必要がある。これら手続きの全体をＦＯＲＴＲＡＮプログラムとして自動コーディングするために、まず論理構造に関する宣言的知識とデータ転送に関する手続き的知識の関係を明確にする必要がある。

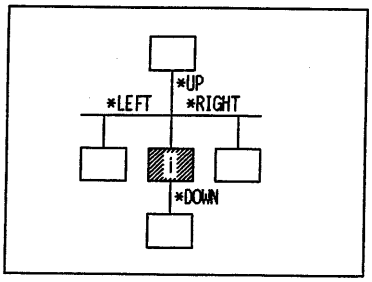

図6.2 第i番目のくり返し集団とその近隣との関係

一方、多くのＤＢＭＳが、多量のデータの一括格納のためのユーティリティ・プログラムを備えている。ここでも、そのようなデータ一括格納用のユーティリティ・プログラムを使うことを前提に、旧サブ・ファイルよりユーティリティ・プログラムへのデータ受け渡しを仲介するプログラムの自動コーディングについて述べる。ただし、自動コーディングされるプログラムは、ＦＯＲＴＲＡＮ言語で記述されるものとする。

まず最初に、データ転送プログラムの構造について述べる。今ある一つの繰り返し集団に注目すれば、そのデータ転送手順の初めの段階で、旧データベースからＤＭＬ（データ操作言語）によりデータを読み出し、これを再編集した後に一時ファイル 08 へ書き込む。前述のように、ここでは一時ファイル08を介して、旧サブ・ファイルからユーティリティ・プログラムにデータを渡す（このことは、単に格納時の処理効率の向上のみが目的である）。以下では、宣言的記述と手続き的記述の関連を明確にすることに焦点をあわせて、図２．１を具体例にしてデータ転送プログラムの構成について述べる。

今、サブ・ファイル 01 において、繰り返し集団 GROUP-2 のデータ転送のみに注目しよう。まず旧サブ・ファイルよりデータを次の命令で読み出す。

```
CALL INQ("MOVE","/GROUP-2/")   (6.1)
```

この命令は、繰り返し集団 GROUP-2 の中の基本項目群をひとまとめにしてプログラムの作業領域に移すためのＩＮＱの命令である。これを受けて、プログラムは再編集後のデータを一時ファイル08に、次の方法で書き込む

```
WRITE(08) IREC2,BBBB          (6.2)
```

データ入出力におけるこれらの命令(6.1) 式と (6,2)式の組み合わせが、繰り返し集団単位でのデータ転送の基本となる。 図６．１は、以上の事柄をＩＮＱのデータ操作言語を用いて一般的に表現したものである。 図６．１の中で、α、β、γ、δは、他の繰り返し集団との相対的な位置関係から決められる文番号である。 （ところで、もし直接に新データベース・ファイルにデータを書き込む場合は、ＷＲＩＴＥ文のかわりに

```
CALL INQ("STORE","/GROUP-2/")
```

とすればよい。 しかしここでは、処理効率の向上のために、ユーティリティ・プログラムを活用した。）

データ転送プログラムの自動コーディングにおける中心的課題は、第４章に述べた述語 LOAD-SEQ に従って図６．１のプログラム・ユニットをモザイク的に組み立て、さらに文番号α、β、γ、δを定めることである。これらを円滑にすすめるために、ある繰り返し集団に注目して、それの階層構造における相対位置をベクトルで次の様に表す（図６・２参照）。

```
(*UP, *LEFT, *DOWN, *RIGHT).
```

この中の各変数名の値と階層構造の中での相対位置の関係は、表６．１のようにまとめられる。 この相対的位置を定めるためパラメータの値を決める述語およびそれらの規則の数は次のとうりである（付録参照）：

```
RELATION-U(*SEQNO,*UP)          ; 4 規則
RELATION-L(*SEQNO,*LEFT)        ; 2 規則
RELATION-D(*SEQNO,*DOWN)        ; 2 規則
RELATION-R(*SEQNO,*RIGHT,*GO)   ; 6 規則
```

第４、５章の結果も合わせ、これらを自動コーディングに必要なパラメータとして編集・出力させたものが、図６．３である（ただし繰り返し集団 GROUP-4のみについての例である）。 ここで *GO の値は、図６．１のγを与える。 特にγの値が０の場合は表６．２の計算に従って文番号が与えられる。

図６．３に示された体裁で与えられるパラメータは、各繰り返し集団ごとにほぼ格納順序に従ってＰＲＯＬＯＧで逐次的に生成され並べられる。 これらは次の段階でＦＯＲＴＲＡＮによって読み込まれ解釈されながら表６．２のコーディング規則に従ってデータ転送プログラムとして自動コーディングされる。ただし、文番号α、β、γ、δは、図６．１

表6.1　繰り返し集団の周辺との関係を示すパラメータ

相対位置	値	意　　味
＊UP	R	直上は，最上位集団である.
	P	親は，第２階層以下である.
	N	自身が最上位で，親を持たない.
＊LEFT	B	同列の先行繰り返し集団あり.
	N	同列の先行繰り返し集団なし.
＊DOWN	C	子の繰り返し集団あり.
	N	子野繰り返し集団なし.
＊RIGHT	B	同列の後続繰り返し集団あり.
	P	同列の後続繰り返し集団はなく，直上の親に帰る.
	R	同列の後続繰り返し集団はなく，最上位集団に帰る.

表６．２　第ｉ番目繰り返し集団のデータ転送プログラムのコーディング規則。表は（＊UP,＊LEFT,＊DOWN,＊RIGHT）を示す。 ただしR̄はR以外、P̄はP以外、＊は任意であることを示す。

行	全文削除	文番号のみ削除	全文与える
α MOVE=i	((R,*,*,*)	(R̄,N,*,*)	(R̄,B,*,*)
β CALL INQ()	(R,*,*,*)		(R̄,*,*,*)
IF() GOTO γ	(R,*,*,*)		注)
δ IREC=i		(R̄,*,*,*)	(R,B,*,*)
GO TO β	(*,*,N,*)		(P,*,C,N)

注) (R,*,*,B); γ=7000+100s+(i+1)
(R,*,*,P); γ=7000+100s+i_p
(R,*,*,R); γ=6000+100s+i_{depth}

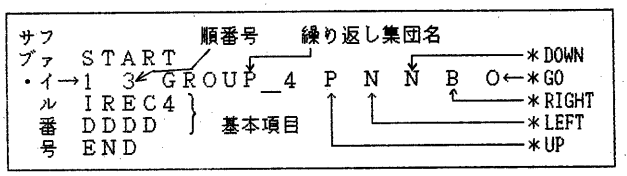

図6．3　文番号 α、β、γ、δ を決めるための
パラメータおよび図6．1との関連データ

及び表6．2の各式に従って算出することによって、プログラム中で唯ひとつの番号に成るように定める。

7．インプリメンテーション

前章までの議論を、実用システムとしてインプリメントする際には、分かり易さと処理効率の両面からＰＲＯＬＯＧとＦＯＲＴＲＡＮを使い分けた。　第3～6章に現れる宣言的知識としての諸規則は、ＰＲＯＬＯＧでコーディングした。　一方でこれらＰＲＯＬＯＧ部の操作コマンドの生成とその起動、最終結果として出力された図6．3の一連のパラメータ・セットからデータ転送プログラムを自動コーディングするなどの手続き的（逐次処理的）な事柄については、ＦＯＲＴＲＡＮ言語を用いた。

インプリメンテーションの際のもうひとつの課題は、マンマシン・インタラクションの形式である。　本論文では、データベース管理者が予め新旧両論理構造を与えておいて、本支援システムがデータ転送の可否をチェックし、論理的に可能なことを確認したうえでデータ転送プログラムを自動コーディングする。　従ってデータベース管理者の初期入力は、論理構造の既に定義されている新旧両データベース名のみである。

この他にも、データベース管理者による新主キーの指定や旧サブ・ファイルの結合における被結合項目の指定による再構成の形態を支援システムに与え、これによる新論理構造の記述の自動生成も可能であるが、ここでの目的から逸れるので割愛する。

8．まとめ

一般に、データベース構築・管理は、操作過程の前後関係や全体の状況を十分に把握しておれば、差ほど難しいことはない。　しかし非熟練者にとっては、これらは決して容易ではない。　また熟練者であっても、論理的に自明な事柄を細々と手作業しなければならない場合は、非常に苦痛を感じる。　本論文では、このような知識を十分に持つ者にとっては当然の事柄を規則の形式で整理し、論理的に自明な操作手順の自動化を図り、データベース管理者への知的支援を試みた。　すなわち、データベース構築・管理の知的支援システムＫＤＢＭＳの機能強化であった。

一方自動化には、しばしば暗黙に標準化・規格化された環境が前提と成っている。　本論文のデータ転送プログラムの自動コーディングにおいても再構成の方式の標準化を行い、これをＰＲＯＬＯＧ言語による事実と規則にまとめることを試みた。　但し主目的が自動コーディングの可能性の確認であったために、関係型データベースにおける『多値従属』や『関数従属』に相当する議論は、図5．2に説明した程度にとどめた。　更に、自動コーディングの論理的背景を明確にするために、モデル化（標準化）の困難な再構成は、議論から除いた。　これらはシステム実用化の段階で、”利用者のニーズをどのように充たすか”といったマンマシン・インタラクションの面から今後の課題となるであろう。

ＤＢＭＳの基本型が定まり、一方で人工知能や知識工学やソフトウェア工学における議論が発展する中で、データベース構築・管理をこのような観点から見直すことは、今後データベースの新しい活用分野を開き、またそのための知的支援を推進していくうえで有意義であろう。またこのような議論は、異なるプログラム・パッケージ間のデータの受け渡し用プログラムの自動コーディングに対しても、同様の議論が可能であろう。

かねてより筆者らは、データベース管理者の不要な苦労の軽減を目指し、経験的知識を論理的に整理したうえでデータベース化し、

これを使って様々の機能の自動化を試みてきた。　本論文の自動化コーディングもそのひとつであった。　多様化する様々のソフトウウェアを、より多くの利用者に能率よく使わせるためには、このような支援の試みも重要であると考える。

参　考　文　献

1）有沢博：データベース理論、情報処理叢書２、情報処理学会、（１９８１）。

2）間野　興：自動プログラミング，情報処理、VOL. 22, No. 11, (1981), PP.1044 ～ 1061.

3）磯本征雄，他：データベース構築・管理のための知的支援システム、『アドバンスト・データベース・システム』シンポジウム，情報処理学会、(1982), PP.49 ～ 58.

4）竹内憲、他：関係データベース・システム　ＩＮＱ、最近のデータベース・システムとその応用、bit 別刷、共立出版、（１９８３），ＰＰ．７８～８７．

付 録：繰り返し集団の相対的位置を決める諸ルール

```
/*--  (5.8.3)  RELATION_STATUS  --*/
RELATION_U(1,N):-!.                                    /* RULE UP1 */
RELATION_U(*SEQNO,R):-                                 /* RULE UP2 */
  LOAD_SEQ(*SEQNO,*LEVEL,_),=<(*LEVEL,2).
RELATION_U(*SEQNO,R):-                                 /* RULE UP3 */
  LOAD_SEQ(*SEQNO,_,*GNAME),
  LINK_TO(*SEQNO,_,_,*FN,*GNAME,*FM,_),
  =/=(*FN,*FM).
RELATION_U(*SEQNO,P):-                                 /* RULE UP4 */
  LOAD_SEQ(*SEQNO,*LEVEL,_),=>(*LEVEL,3).
RELATION_U(*SEQNO,N):-DISPLAY("NOT FOUND A REASONABLE RELATION IN R_1"),
  DISPLAY("      AT SEQUENCE ",*SEQNO).
RELATION_L(*SEQNO,B):-=>(*SEQNO,3),                    /* RULE LEFT1 */
  LOAD_SEQ(*SEQNO,*LEVEL,*GNAME),
  IS_CHILD_OF(0,*LEVEL,*GNAME,*PARENT),
  IS_CHILD_OF(0,*LEVEL,*GNM,*PARENT),
  LOAD_SEQ(*SQ,*LEVEL,*GNM),<(*SQ,*SEQNO).
RELATION_L(*SEQNO,N):-!.                               /* RULE LEFT2 */
RELATION_D(*SEQNO,C):-                                 /* RULE DOWN1 */
  LOAD_SEQ(*SEQNO,*LEVEL,*GNAME),+(*LVL,*LEVEL,1),
  IS_CHILD_OF(0,*LVL,_,*GNAME).
RELATION_D(*SEQNO,N):-LOAD_SEQ(*SEQNO,_,_).            /* RULE DOWN2 */
RELATION_D(*SEQNO,N):-DISPLAY("NOT FOUND A REASONABLE RELATION IN R_D"),
  DISPLAY("      AT SEQUENCE ",*SEQNO).
RELATION_R(*SEQNO,B,0):-                               /* RULE RIGHT1 */
  LOAD_SEQ(*SEQNO,*LEVEL,*GNAME),
  LINK_TO(*SEQNO,_,_,*FN,*GNAME,*FN,*PARENT),
  IS_CHILD_OF(0,*LEVEL,*GNM,*PARENT),
  =/=(*GNAME,*GNM),LOAD_SEQ(*SQ,*LEVEL,*GNM),
  >(*SQ,*SEQNO).
RELATION_R(*SEQNO,R,0):-                               /* RULE RIGHT2 */
  LOAD_SEQ(*SEQNO,2,*GNAME),
  IS_CHILD_OF(0,2,*GNAME,ROOT).
RELATION_R(*SEQNO,R,0):-                               /* RULE RIGHT3 */
  PRIMARY(_,TYP2,_,_),LOAD_SEQ(*SEQNO,3,_).
RELATION_R(*SEQNO,R,0):-                               /* RULE RIGHT4 */
  LOAD_SEQ(*SEQNO,_,*GNAME),
  LINK_TO(*SEQNO,_,_,*FN,*GNAME,*FM,_),=/=(*FN,*FM).
RELATION_R(*SEQNO,R,0):-                               /* RULE RIGHT5 */
  LOAD_SEQ(*SEQNO,_,*GNAME),
  LINK_TO(*SEQNO,_,_,*FN,*GNAME,*FN,*PARENT),
  LINK_TO(*SQ,_,_,*FN,*PARENT,*FM,*GPARENT),=/=(*FN,*FM).
RELATION_R(*SEQNO,P,*IFGO):-                           /* RULE RIGHT6 */
  LOAD_SEQ(*SEQNO,_,*GNAME),
  IS_CHILD_OF(0,_,*GNAME,*PARENT),LOAD_SEQ(*IFGO,_,*PARENT).
RELATION_R(*SEQNO,N,0):-DISPLAY("NOT FOUND A REASONABLE RELATION IN R_R"),
  DISPLAY("      AT SEQUENCE ",*SEQNO).
```