U.D.C. 681.322.067.004.14

複数データベースの統合・利用技術

Integration and Migration of Database Systems

米田　茂*　*Shigeru Yoneda*
吉田郁三*　*Ikuzô Yoshida*

計算機システムの拡張，独立に構築されたデータベースの相互利用に際し，重要となってくるデータベース変換システムについて実験システムTRVSを例にとり説明した。TRVSは，データ構造を記述するDCDL，変換規則を記述するMPDLの2種の高水準の言語機能によってデータベースの変換機能を実現する。DCDL，MPDLの内容を具体例を挙げて説明した。

本TRVSを用いると，データベース変換プログラムが従来のプログラム言語を用いた場合に比較して数十分の一のステップ数で作成される。MPDLで設定したデータ，メタデータ変換機能は他変換システムにない特長である。

1　緒　　言

データ定義言語DDL(Data Description Language)の利用は，データベースシステムで非常に重要な役割を果たしている。すなわち，データに対する共通の見方を定義するDDLの導入によって，データの独立性，共用性が従来のシステムに比較し，飛躍的に強化された。しかし，現在開発されているDDLは，ハードウェア性能，ソフトウェア技術の現実的な制限から，データベースを管理しているDBMS(Data Base Management System)の構造，あるいはハードウェアの構造に依存したものになっている。この結果，計算機システムのソフトウェア，ハードウェアの拡張，新規のアプリケーションの開発で，既に開発・蓄積しているプログラム，データなどの財産をいかに守り，有効活用していくかという問題が発生する。同一ユーザーの計算機システムで，複数のDBMSを用いた計算機システムの構築が行なわれている点，及び異なる計算機で構築されているデータの共用が要求される点が，この問題をいっそう複雑にしている。

本論文では，上記の問題に対し提案されている種々のアプローチについて述べるとともに，複数データベースの統合・利用に際しての基本技術であるデータベースの変換システムに関して，日立製作所が研究しているシステムを例にとり，その持つべき機能・実現方法について検討する。

2　データベース統合の方法と分散データベース

前述した問題は，従来，データの独立性として議論されてきたものである。これに対し種々の側面からアプローチされている。その代表的なものを以下に述べる。

(1)　各種のデータモデルを内蔵し，データの独立性を保証するDBMSを開発する。

このアプローチは，ANSI/X3/SPARC(ANSI：American National Standard Institute, X3：計算機と情報処理分野，SPARC：Standards Planning And Requirements Committee)DBMSスタディグループが提案したユーザーのデータに対する見方を表わす外部スキーマ，データの物理的蓄積構造を表わす内部スキーマ，そして実世界のモデルを表わす概念スキーマの3レベルのスキーマ構造をもつDBMS機能[1]を実現しようとするものである。

上記提案に触発され，ユーザーのデータに対する見方を表わすサブスキーマ，データベースの全体構造を表わすスキーマを導入したCODASYL(Conference on Data Systems Languages)のDDLC(Data Description Language Committee)がスキーマを記述するDDL JOD(Journal of Development)の改訂[2]（データの物理構造スキーマ概念の導入など）を施し，G. M. Nijssenらが異種DBMSの共存を実現するアーキテクチャの提案[3]を行なっている。

(2)　各種のデータモデルを表現できる統一したデータ操作言語を開発する。

C. J. DateがUDL(Unified Database Language)として提案したもの[4]で，三つのデータモデル(リレーショナル，階層，ネットワークの各モデル)の操作機能を一つの言語体系の中に実現している。

(3)　データベースを要求された形に抽出したり，変換したりするデータベース変換システムを開発する。

CODASYL SDDTTG(Stored Data Description and Translation Task Group)の活動[5]，ミシガン大学の実験システムの開発[6]など。SDDTTGでは，データの論理構造及び物理構造の表現に，M. E. Senkoの提案したDIAM(Data Independent Accessing Model)モデル[7]を修正した方法を適用している。

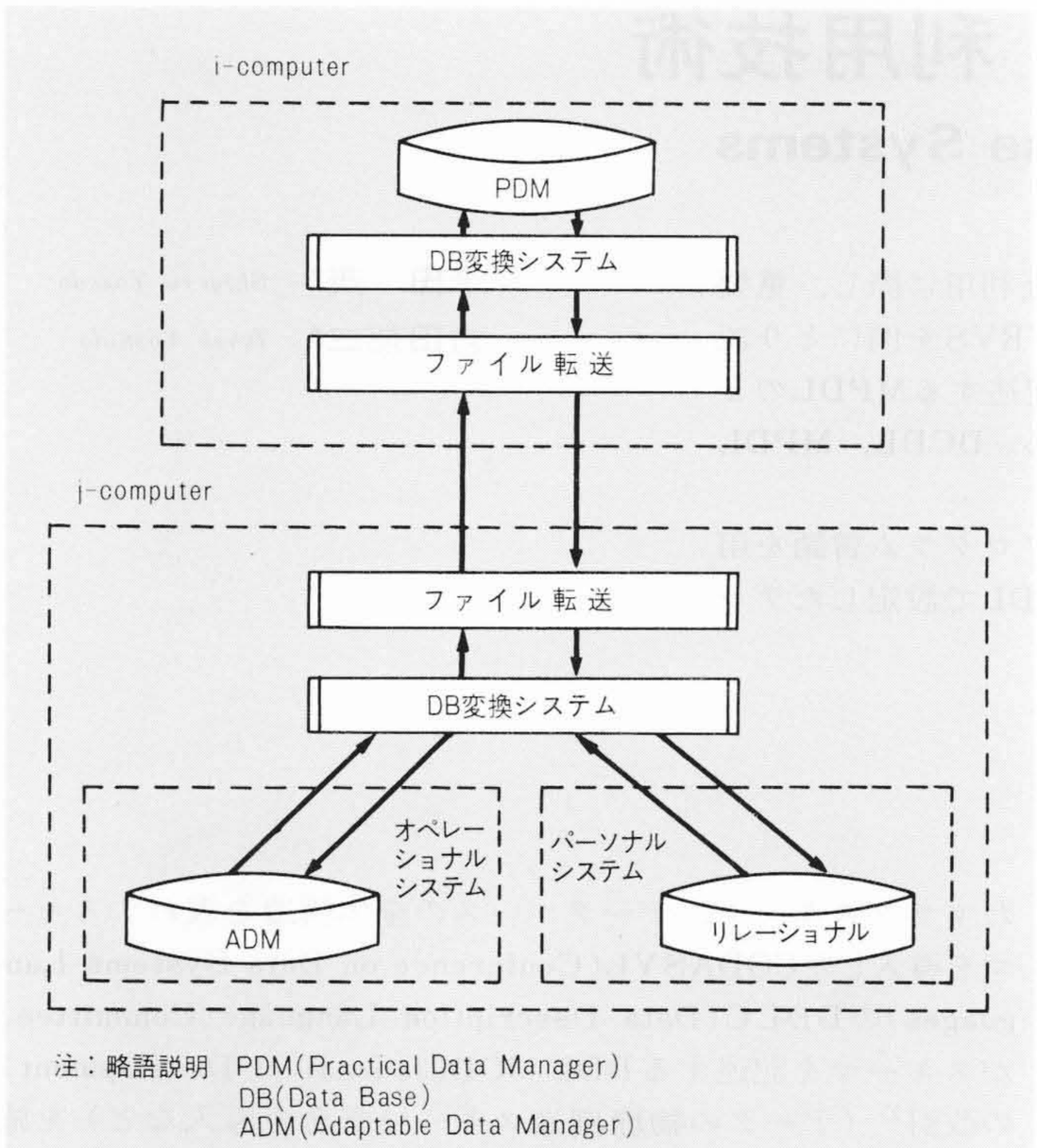
上述したアプローチの中で，既に構築されたデータベースの統合利用・移殖を可能とし，現実的な解決方法を与えることを前提に，上記(3)のデータベース変換システムを取り上げることにする。

データベース変換システムに関連した問題としてプログラム変換の問題があるが，ここでは，あるデータベースから他のデータベースへデータベースを変換する，あるいは，必要なデータを抽出して別データベースを作り出す，というデータベース自体の変換・統合の問題に対象を限定する。

データベース変換システムは，次の問題領域をカバーするものである。

(1)　DBMSの変更によるデータベースの変換
(2)　OS(Operating System)ファイルのデータベースへの変換
(3)　データベースの再構成
(4)　オペレーショナルシステムで構築されたデータベースの意思決定支援システム，OA(オフィスオートメーション)など

* 日立製作所システム開発研究所



注：略語説明 PDM(Practical Data Manager)
DB(Data Base)
ADM(Adaptable Data Manager)

図1 DB変換システムの分散システム・個人システムへの適用
異種・複数データベース環境下でのデータベースの相互利用・概念を，データベース変換システムを核として示している。

の個人用データベースへのデータ抽出
(5) 分散システムでのデータ交換

上記の中で，(4)，(5)に対するデータベース変換システムの適用の概念を**図1**に示す。変換システムの詳細に入る前に，同図に関連して，データベース変換システムの分散システムへの適用について簡単な説明を与える。なお，分散システム，特に分散データベースシステム自体の現状については，参考文献8)，9)を参照されたい。

分散システムに対するユーザー要求をみると，次の両極に大別できる。

(1) 銀行のオンラインシステムのように，ユーザーにとってシステムの独自性・個別性が必要でないシステム(権限集中システムと呼ぶ)。

(2) 意思決定支援システム，OAなど個人あるいは部署単位を指向したシステムのように，データの相互利用のために緩く計算機システムが結合され，各システムの独自性を認めることが必須なシステム(権限分散システムと呼ぶ)。

上記二つの分散システムの特徴をまとめてみると**表1**に示すようになる。この中で，前述したデータベース変換システムの機能は，(2)のタイプの分散化環境に適した技術と言える。ユーザーが要求するデータの構造・データ形式に合致するように，オペレーショナルデータベースに対し，**図1**のデータベース変換システムが，選択条件の設定，ファイルの結合，データ構造の変換などの操作を施しデータを抽出する。この抽出したデータ集合をファイル転送システムによってユーザーシステムに提供する。上述の方法により，部署間で合意されたデータの交換機能の実現，部署間のセキュリティの確立，一括データ送信による通信オーバヘッドの削減など，権限分散システムで必要な諸機能が実現される。

分散システムの議論で，従来，技術的要因，経済的要因が強調されてきたが，これに加えてユーザーの利用特性(コンピュータ利用組織の特性)に対する配慮が重要と考えている。すなわち，分散システムを自分たちが責任をもって管理し，自分たちの業務に密着した作業を行ない，その中の情報を自分たちで保護するというユーザーの自主性，独自性を実現する“do it yourself”のシステムを実現するものとしてとらえることが，分散システムを考える際の本質的な一要素である。

3 データベース変換システムの機能

前章で述べたデータベース変換を行なう実験システムとして研究しているTRVS(Translation, Restructuring and Validation System)の概要，言語機能について説明する。

3.1 TRVSの概要

データベース変換機能の技術開発項目の明確化を目的に，TRVSでは，次の基本方針に基づいて機能を設定している。

(1) 変換用原始データ，目的データは，データベース初期作成データの形式で順次ファイル媒体上に登録されている。

(2) データベース変換処理の自動化は困難であるため，原始データ，目的データに対し，(a)構造記述及び(b)両者間の変換規則の記述の2種類の記述をTRVS利用者が行なうことによって，データベース変換機能を実現する。

(3) データベース変換に際しては，原始データを目的データへとバッチ処理的に一括して変換する。

上記(2)で述べた2種類の記述に対応して，データ構造記述言語DCDL(Data Characteristics Description Language)，変換規則記述言語MPDL(Mapping Procedure Description Language)の言語機能を導入した。

TRVSのデータベース変換処理の概念を**図2**に示す。

DCDL，MPDLの定義内容に従って，原始データを目的データへ変換する。この変換に際し，DBMSあるいはユーザーが独自の方法で構成しているデータをTRVSで定める内部形式のデータに変換する。Restructurer(再構成器)はこの内部形式のデータを操作することによって，各種システムで構築されたデータとの独立性を実現している。具体的には，DBMSであるADM(Adaptable Data Manager)などで定義される階

表1 分散形態の相違による分散システムの特性 権限集中形の分散システムと権限分散形の分散システムの各々の特徴を，特にデータベース利用の面から比較している。

分散形態	トラフィック	一度に転送されるデータ量	DBの構築の形態	データ管理の方法	オペレーショナルDBとの区分	オペレーショナルDBへのアクセス	機密保持
権限集中システム	大	小(レコード単位が大部分)	各システム均質に作成	一括して集中的に管理	同一である場合が多い。	自由にアクセス	システム全体として保証
権限分散システム	小	大(ファイル単位が多い)	全く独立に作成	各システム個別に管理	全く独立な場合が多い。	非常に制限を受ける(合意要)。	各個別システム対応に保証

層構造及びOSファイル，リレーショナルモデル，ネットワークモデルの初期作成用データとして用いられるフラット構造の2種類のデータ構造を内部的に表現・操作する機能をもっている。

図2で示したTRVSの構成要素の概要は，以下に述べるとおりである。

(a)　SFORM：データベースあるいはOSファイルからアンロード，ダンプされた原始データを再構成器が取り扱う内部形式のデータに変換する。

(b)　TFORM：再構成器によって作成された内部形式の目的データをユーザーが要求したデータ形式に変換する。

(c)　Restructurer：MPDLで定義された内容に則して，データの分割，結合及び構造変換を行ない，目的データの内部形式を作成する。

以下にTRVSの言語機能を具体例に基づいて説明する。ここで用いるデータベースを**図3**に示す。

3.2　データベース構造記述言語“DCDL”

DCDLは，(1)データの構造体・型定義，(2)データの妥当性チェック定義，(3)データが果たす役割定義，を目的とし，PL/I(Programming Language/I)の構造体宣言，DBMSのデータ定義言語を参考に記述方式を設定している。

DCDL定義の基本構造を**図4**に示す。

変換システムの入力データ（原始データ）の定義をSDD(Source Data Definition)，出力データ(目的データ)の定義をTDD(Target Data Definition)としてDCDLで記述する。**図4**で示した各原始，目的データの詳細情報を定義するfile記述，segment記述，item記述の内容は，データベース変換処理で必要とされる以下の項目を含んでいる。

(a)　file記述

順次ファイル上に登録される原始データ，目的データと以降の定義内容との対応づけを行なうため，ファイル名，レコード・アクセスに利用する論理レコード長さ，キー項目を指定する。

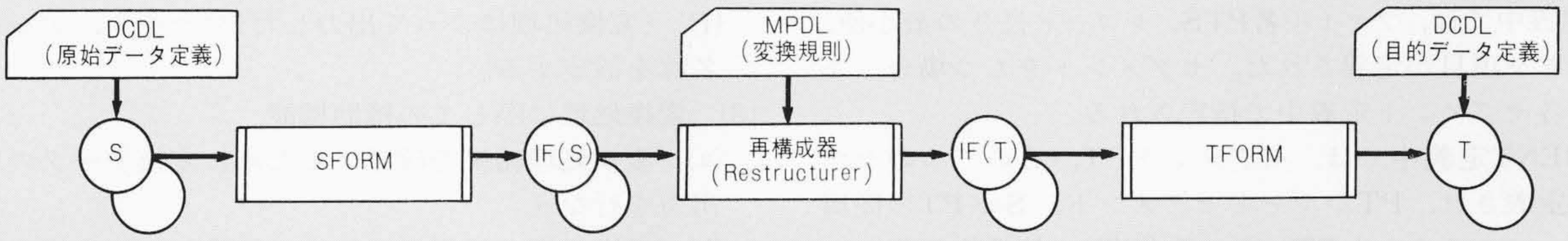
(b)　segment記述

セグメント構造を定義する。セグメント名，セグメント発生条件，セグメント間の親子関係，親セグメントに対する子セグメントのキー項目を定義する。

(c)　item記述

ファイルあるいはセグメントのデータ項目を定義する。レベル番号によってデータ項目の構造を表わし，データ項目名，長さ，形，繰返しグループ情報，データの妥当性チェック条件，データ項目がもっている役割などを指定する。

図3のPTSファイルに対するDCDL定義の例を，(1)繰返しグループとして定義する場合，(2)セグメントとして定義す



注：略語説明　DCDL(Data Characteristics Description Language)，MPDL(Mapping Procedure Description Language)，
S(原始データ)，T(目的データ)，IF(内部形式データ)

図2　データベース変換処理の概要　　データベース変換システムTRVS(Translation, Restructuring and Validation System)の変換過程の概念を，ユーザーの入力情報と合わせて示している。

PTS

PID	P#	PN	S			
			SID	S#	SN	UC
01	2	X	02	4	AB	5
			02	2	BB	4
01	3	XX	02	4	AB	2
			02	1	XB	3
01	6	Y	02	7	AC	7

F2

SN	P#	SALES					
		JAN	FEB	MARCH	APRIL	MAY	JUNE
AB	1	30	23	35	32	40	42
AB	2	31	39	33	27	35	45
			⋮				

INV

P#	QH
2	10
3	17
4	5
6	20

SUP

SN	S#	AD
AB	4	SJ
AC	7	MV
BB	2	SF
DX	5	LA
XB	1	SJ

F1

SN	P#	MM	SALES
AB	1	1	30
AB	2	1	31
AB	3	1	29
AB	4	1	40
	⋮		

注：略語説明
PID(部品データID)
P#(部品番号)
PN(部品名)
SID(業者データID)
S#(業者番号)
SN(業者名)
UC(単価)
QH(在庫量)
AD(住所)
MM(月)
SALES(売上量)

図3　例として用いるデータベース　　本稿第3章以降のデータ構造記述言語DCDL(Data Characteristics Description Language)，変換規則記述言語MPDL(Mapping Procedure Description Language)の説明で用いるデータベースの構造図である。

```
SDD;…原始データの構造定義を表わす。
     (file記述)
 ID;…データ項目の定義を表わす。
     (item記述)
SDD;
     (file記述)
 SEGMENT;…セグメントの定義を表わす。
     (segment記述)
 ID;
     (item記述)
 SEGMENT;
    :
TDD;…目的データの構造定義を表わす。
     (file記述)
 ID;
     (item記述)
TDD;
     (file記述)
SEGMENT;
     (segment記述)
ID;
     (item記述)
SEGMENT;
 :
/*…DCDL定義の終了を表わす。
```

図4 DCDL定義の基本構造 データ構造記述言語DCDLを用いて，原始データ定義，目的データ定義を行なうに際しての基本的な定義構造を示している。

る場合の2種類について**図5**(a), (b)に示す。

SDD定義中では，ファイル名PTS，レコード長さの最小値，最大値，キー項目が定義された。セグメントをもつ場合，キー項目は各セグメント定義中で指定される。

SEGMENT定義中では，セグメントPT，Sの二つのセグメントが定義され，PTがルートセグメント，SがPTの従属セグメントであることをPT(0)，S(PT)の括弧内の親セグメント情報によって示し，当該セグメントの長さ，キー項目，発生条件をSEGSIZE，IDENTIFIER，CASEにより定義する。

ID定義中では，レコード上のデータ項目の構造をレベル番号によって指定し，各データ項目の形(文字形式，10進パック形式，2進固定小数点形式など)，長さ，データ項目のもっている役割(RL)，チェック条件(CH)，繰返し条件(OCCURS)などを定義している。また，例としては出ていないが，DCDLを利用することによって，データ形の変換，演算機能を使用した新データ項目の導入がTR句，DR句によって定義できる。

3.3 変換規則記述言語“MPDL”

変換規則の記述言語MPDLが実現すべき再構成機能を分類すると以下に述べるようになる。

(1) データベース再構成の基本機能

(a) ファイル中の必要なデータ項目を選択，抽出し別ファイルを作成する(ファイルの分割機能)。

(b) 複数ファイルを結合して一つのファイルを作成する(ファイルの統合機能)。

(c) データ構造に操作を施し，他のデータ構造を作る(構造変換機能)。

(i) 階層構造・フラット構造間の変換を行なう。

(ii) データ項目の出現値からデータ項目名称を新たに設定したり，逆操作を行なったりする(データ，メタデータ変換機能)。

(2) 上記基本機能に付随して必要となる機能

(a) ファイル内，ファイル間にわたる選択条件，結合条件を与える。

(b) データの並び順の制御を行なう。

(c) データ出現値のユニーク性の制御を行なう。

(d) データ項目の演算に基づいて，新データ項目を導入したり，既データ値の変更を行なう。

(e) 同一の値をもつデータ別に分類，グループ化する。

(f) (変換処理によって出力した)ファイル，データ項目に名称を設定する。

(3) 変換処理に際しての補助機能

(a) 変換処理結果を確認するために変換データのプリント出力を行なう。

(b) 変換処理の過程で一時的に作成した作業用中間ファイルを削除する。

上記の機能分類に則してMPDLの言語機能を設定している。MPDLの高機能化と単純化を目的として各オペレータは1以上のファイルに操作を施し，一つのファイルを生成する方式にしている。MPDLは現在11種類のオペレータと5種類の組込み関数から構成されている。**表2**にMPDLオペレータの一覧を示す。

MPDLオペレータの中から，データの選択を行なうSELECT，

(a) 繰返しグループ定義

```
SDD;
   PTS;RECSIZE(38,108);P#:IDENTIFIER(A);
  ID;
   02:PT;
     03:PTID        CHAR(2);
     03:P#          P(3);
     03:PN          CHAR(20);
     03:S           OCCURS MAX=6,
                    END=SID¬=`02´;
       04:SID       CHAR(2);
       04:S#        P(3);
       04:SN        CHAR(8);
       04:UC        P(3),
                    CH((GT 0) AND (LT 12));
/*
```

(b) セグメント定義

```
SDD;
   PTS;RECSIZE(32,32);
 SEGMENT;
  PT(0);SEGSIZE(24);P#:IDENTIFIER(A);
  CASE(PID EQ `01´);
 ID;
   02:PID         CHAR(2);
   02:P#          P(3);
   02:PN          CHAR(20);
 SEGMENT;
  S(PT);SEGSIZE(14);S#:IDENTIFIER(A);
  CASE(SID EQ `02´)
 ID;
   02:SID         CHAR(2);
   02:S#          P(3);
   02:SN          CHAR(8);
   02:UC          P(3);
                  CH((GT 0) AND (LT 12));
/*
```

図5 DCDL記述例 図3のPTSファイルに対するDCDL定義を，繰返しグループの場合及びセグメントの場合2種について示している。

表2　MPDL一覧　変換手続記述言語MPDLで準備しているMPDLオペレータの一覧と，その概略機能を示す。

項番	MPDL	機　　能
1	←	ファイル名称，データ項目名称の割当て
2	SELECT	条件に合致したデータの抽出
3	BIND	複数ファイルの結合
4	MERGE	複数ファイルのマージ
5	FLAT	階層構造をフラット構造へ変換
6	CONSTRUCT	フラット構造の階層化
7	STRUCT	データ値をメタデータに変換
8	DESTRUCT	メタデータをデータ値に変換
9	HNORMAL	階層パス上の重複排除
10	SORT	データのソート
11	DROP	一時ファイルの削除
12	組込み関数	SUM/MAX/MIN/AVG/COUNT

ファイルの結合を行なうBIND，データ値から項目名称を導入するSTRUCTのオペレータについて，以下の例題に基づいてその記述方法を示す。ここで使用するデータベースの構造は，**図3**に示したものである。

(1)　SELECTオペレータの例(その1)

PTSファイルの情報を，その住所(AD)がSJの業者が供給している部品の情報に限定する。この限定された情報により発生するファイルをNFとし，NF中にはPTSファイルのデータ項目P#，S#，P#，PN，SNの各情報が含まれるものとする。要求されたファイルNFは**図6**(a)で示したSELECT文によって得られる。同図(a)の指定により，PTSファイルからデータ項目P#，PN，SN，S#が抽出される。このとき，要求された部品供給業者の住所がSJという条件を指示するデータ項目がSUPファイル中に存在するため，PTSファイルとSUPファイルを結び付けるデータ項目S#(業者コード)を利用し，データ選択条件の実現を行なっている。

(2)　SELECTオペレータの例(その2)

PTSファイルの各部品ごとに，その部品を提供しているメーカーの数(NS)を求めるとともに，各部品の単価を1上げる。このファイルをNPTSとする。NPTSは，個数を数え上げる組込み関数COUNT及びデータ選択に際するデータ項目の演算機能を用いることによって**図6**(b)のように書くことができる。

(3)　BINDオペレータの例

PTSファイルとINVファイルとを結合し，部品－業者－在庫量の関連を示す新たなファイルPTSINVを作成する。結合は，同一のPTS，INVファイル中に含まれる同一の部品番号をキーとして行なう。この記述例を**図6**(c)に示した。このBINDオペレータによるファイルの結合は，リレーショナルデータベースで言う結合操作(join)と等価な動作が階層構造を保存する形で得られる。

(4)　STRUCTオペレータの例

ファイルF1でデータ値として登録されている月の情報をファイルF2で示すように項目名称として変換する。この記述の方法を**図6**(d)に示した。

STRUCTオペレータは，データ値として既に登録されているデータベースに対し，それをデータ項目名称(メタデータ)として処理したいというデータに対するユーザーの見方(ユーザービュー)の変化に対処するものである。この機能は通常の事務作業で慣れ親しんでいる帳票などで，月別の生産・販売実績，店別の販売実績など月単位に収集されたデータを一覧性を向上させる目的で，1枚の帳票上にまとめて書くという機能に対応するものである。オペレーショナルシステムで構築したデータベースをOA，DSS(Decision Support System)など最近話題になっているエンドユーザー指向のシステムに取り込む際，重要となる機能である。

STRUCTオペレータの対となるオペレータとして，メタデータをデータ値に変換するDESTRUCTオペレータを設定している。

STRUCTでデータ値を見出しデータ項目名称に変更する場合，その見出しに対応するデータ値としてどのデータをもってくるかが問題となる。見出しとなったデータ値に対応して，例えば，在庫量，販売量など複数項目のデータを関連づける場合と単一の項目だけを関連づける場合の2種の場合が発生する。STRUCTオペレータでは，この二つの場合に対応して二つのフォーマットを与えている。**図6**(d)の例では，データ値MMのメタデータ(JAN.，FEB.など)に対応して販売量を示す単一のデータ(SALES)値を取り扱うことを示している。具体的には，データ項目MMの値1，2，3，…6を，データ項目名称JAN.，FEB.，MARCH，…JUNEに変換し，各見出しに対応する月データ値とマッチするSALES値を該当項目の値とする。

```
(a)
   NF←SELECT<P#,PN,SN,S# FROM PTS:
     (PTS.S# EQ SUP.S#)AND(SUP.AD EQ `SJ')
(b)
   NPTS←SELECT<P#,PN,(NS)=COUNT<S#
     BY P#>,S#,SN,UC+1  FROM PTS>
(c)
   PTSINV←BIND<P#,PN,S,S#,SN,UC
     OF PTS1 QH OF INV:(PTS.P# EQ INV.P#)>
(d)
   F2←STRUCT<P#,SN,SALES(MM=(1=JAN,
     2=FEB,3=MARCH,4=APRIL,5=MAY,6=JUNE))
     FROM F1>
```

図6　MPDLの記述例　本文中の例題に対する変換規則記述言語MPDLの書き方を示したものである。

4　TRVSの期待効果

TRVSで設定した言語機能が，どの程度データベース変換プログラム開発で工数削減に貢献するかを，TRVSとPL/Iとの比較により検討する。

表3は，1MPDLオペレータがPL/Iの宣言文(DCL文)，手続文何ステップに対応するかを示したものである。すなわち同表は，PL/Iでファイル変換プログラムを個別に作成し

表3　MPDLのPL/Iの展開　TRVSによる工数削減効果をみるため，変換規則記述言語に対応して，PL/Iステートメントが何ステート必要かを表わしている。

項番	MPDL	PL/I 文		
		DCL文	手続き文	総　計
1	←	11	19	30
2	SELECT	25	57	82
3	FLAT	22	45	67
4	BIND	32	85	117
5	CONSTRUCT	22	75	97
6	SORT	30	110	140
7	MERGE	32	85	117
8	HNORMAL	25	68	93
9	DROP	1	14	15

た場合必要となるステップ数と，同等の機能をTRVSで記述した場合のステップ数との比較を表わしている。手続き文のステップ数をみると，PL/I のほうが数十倍多くのステップ数が必要なことが理解できる。

この結果から，TRVSのデータベース変換に際する変換プログラム作成の工数削減が大幅に可能なことが把握できる。

5 結　言

複数データベースの統合・利用に対し，提案されているアプローチについて説明した。この中で，データベース自体を一括して変換する実験システムTRVSの言語機能を，主に例を用いて説明した。現在，TRVS言語仕様の最初のバージョンを実験中である。現在までの実験を通じ，実システム適用上の留意事項として次の2点が重要と感じている。

(1) 変換データの正当性の保証

(2) 変換処理速度の保証

また，今後の課題として実DBMS，計算機ネットワークシステムとの結合による分散システムへの適用・発展を考えている。

参考文献

1) ANSI/X3/SPARC Study Group on DBMS：Interim Report(1975)
概念スキーマ導入によるデータの独立性実現の提案を，最初に行なった資料である。本資料の他の特徴の一つは，統合データディクショナリの考えを導入している点にある。

2) CODASYL DDLC：DDL JOD(1978)
スキーマDDLの構文を詳述している。付録として，格納構造記述言語DSDL(Data Storage Description Language)の構文が示されている。

3) G.M.Nijssen：A Gross Architecture for the Next Generation DBMS：Modeling in DBMS, G. M. Nijssen eds. North Holland Publishing Company, 1～24(1976)
実世界のデータのモデル化を考察し，概念スキーマを導入している。この概念スキーマに対し，種々のユーザービュー(データモデル)の共存を図るDBMSアーキテクチャが提案されている。

4) C. J. Date：An Introduction to the Unified Database Language(UDL)：Proc. 6th. VLDB (1980)

5) CODASYL SDDTTG：Stored-Data Description Approach to File Translation：A Model and Language：Information Systems, **2**, 3, 95～148(1977)

6) J. P. Fry eds：An Assessment of the Technology for the Data-and-Program-Related Conversion：Proc. NCC(1978)
データベース及びプログラム変換に関する広範囲の調査内容に基づき，技術課題の整理を行なっている。

7) M. E. Senko eds：Data Structures and Accessing in Data-Base Systems：IBM Systems J., **12**, 1, 30～93(1973)
データの論理レベルの表現及び物理(装置)レベルの表現について，その記述方法をレベルに分類し示している。

8) 日本電子工業振興協会，データベース専門委員会：データベース，システムに関する調査－分散データベースおよびデータベース・マシーン：(1981)

9) M. Adiba eds：Issues in Distributed Data Base Management Systems：A Technical Overview：Proc. 4 th VLDB (1978)

論文抄録

順序回路に対するテスト発生の一手法

日立製作所　林　照峯・五嶋　将・他1名

電子通信学会論文誌D　J64D-9，869～876（昭56-9）

電子計算機技術，半導体技術の発展に伴い，論理装置の故障診断はますます重要な問題となっている。なかでも，プリント板などの論理回路に対するテストパターンを電子計算機により自動的に発生させる問題は重要な問題の一つである。これに関して，従来から種々の研究が行なわれ，幾つかの成果が得られているが，順序性が比較的強い場合，例えば高次のカウンタを含む場合のテストパターン発生能力については明らかでない。

論理回路のテストパターン発生手法としては，Dアルゴリズムに代表される経路活性化法という手法がよく用いられているが，順序回路に対しても効率良くテストパターンを発生させるには種々の工夫が必要となる。本論文は，この立場に立ち，比較的順序性の強い回路に対しても有効なテストパターン発生手法を提案している。

第一に，フリップフロップ，カウンタ，シフタ，ROM，RAMなどをゲート展開せず一つの論理素子として扱えるようにし，素子の機能を考慮しながらテストパターン発生ができるようにしている。

第二に，すべての素子出力線及び外部入力線に対し，3種のヒューリスティックなコスト関数を定義し，これらのコスト関数を用いて，テストパターン発生手続き中に生ずる選択を行なうようにしている。コスト関数は，回路中の素子の動作機能及び接続関係により決めている。小規模回路での実験によれば，この方法は従来の一般的な方法に比べ，処理時間，故障検出率の両方をかなり向上させており，有効であることが確認できた。

第三に，カウンタに対し，その状態遷移の規則性を利用した新しい手続きを導入することにより，高次のカウンタを含む回路に対するテスト発生能力を高めている。すなわち，カウンタを動作させるための長い入力パターン系列を，短いパターン系列の繰返しという形で容易に求まるようにしている。ほかにも，ROM，RAMに対してアドレスパターンの選択優先度を，テストパターン発生の成否により学習的に変えていくようにしている。このようにして，フリップフロップよりも高度な機能素子を含む順序回路に対するテストパターン発生能力を高めている。

また，このような手法を主にFORTRAN言語を用いてプログラム化し，比較的複雑なプリント板回路で実験を行なったところ，故障検出率，処理時間の性能で良好な結果が得られた。特に，カウンタ，シフタ，RAMを含む2,300ゲートの回路に対し，故障検出率97%，CPUタイム21分(3Mips計算機)が得られた。

以上により，本論文の手法の有効性が確認された。しかし，カウンタ，シフタなどを含む複雑な多重ループがある場合など，非常に順序性の強い回路に対しては十分な性能が得られないこともあり，この点が今後の研究課題である。