ISSN 0289-6257

技報

UNIVAC TECHNOLOGY REVIEW

1987年11月　第15号

論　文

技術動向

日本ユニバック

従来，磁気ディスク制御装置には7バイトECCが実装され，11ビットまでのバースト誤りを訂正していたが，記録密度の増加に伴いバースト誤り訂正能力を向上させる必要性が生じてきた．上谷彊輔の**磁気ディスク制御装置における14バイトECCの開発**は，今回開発した14バイトECCの設計，能力評価，および得られた結果を述べるとともに論理回路とマイクロコードの設計方法についても紹介している．

河合昭男の**イメージ・データの圧縮手法**は，日本ユニバックのワークステーションDS 7のイメージ処理システムU-IMAGEのために開発されたイメージ・データ圧縮法について述べている．圧縮法としては，シリーズ1100で採用されているモディファイド・ハフマン符号化が基本となっている．本稿では，開発した圧縮法を紹介するとともに圧縮率の実測データを元に各圧縮法の評価を行っている．

宮坂順之の**衛星通信回線使用時のDCAリンク・レイヤおよびネットワーク・レイヤの伝送効率分析**は，衛星通信回線をDCP/TELCONトランク回線として使用した時のDCAデータ・リンク・レイヤ，およびネットワーク・レイヤの伝送方式を理論式を用いてモデル化し，その伝送効率の分析を行っている．なお，本稿は，昭和60年度より郵政省が推進している衛星通信実験パイロット計画での実験結果をまとめたものである．

K. Brusoらの**再使用可能コード・ライブラリBIBLIOの開発**は，同システムに関する第2回目の報告であり，BIBLIOの開発で遭遇した技術上および手続上の問題を考察し，実際にとられたアプローチの理由付けを行っている．

リレーショナル・データベースは，そのモデルの簡明さと操作の高水準性によって，システム開発の生産性向上に寄与するものとして期待されている．リレーショナル・データベースの実現では，現実世界の要求や性能面への妥協から，本来目指していた目的を損いかねない状況がある．原潔の**リレーショナル・データベースの考察——整合性制約を中心として**は，整合性制約の問題を取り上げ，リレーショナル・データベースが，どこまでそれを実現しているかについて考察している．

大内忍の**ネットワーク型データベース（DMS 1100）からリレーショナル・データベース（RDMS 1100）への変換**は，日本ユニバックのネットワーク型データベースDMS 1100のプログラムをリレーショナル・データベースのプログラムに変換する方法について考察している．

従来，データベースの障害に対するリカバリは，各システムごとに異なった運用手順・方法・ツールなどを用いて行ってきた．小林俊平の**COSTAR（ATD）によるデータベース・リカバリの運用自動化の実現**は，データベースのリカバリ処理の運用管理システムCOSTAR（ATD）の概要について紹介している．このシステムによって，標準の自動リカバリ運用が確立し，コンピュータ室の無人運転に寄与することができたと述べている．

森宗正の**各種バーコードのパーソナル・コンピュータによる印書実験**は，パーソナル・コンピュータとワイヤドット漢字プリンタを利用して行ったバーコードの印書実験について述べている．本実験では，NW-7，CODE-39，および各種バーコードを印書し，バーコード・リーダで読み取り，その可能性を評価し，十分実用に耐えることを確認している．

☆

論文 磁気ディスク制御装置における14バイトECCの開発

Development of 14 Byte ECC for Magnetic Disk Storage Control Unit

上谷彊輔

要約 磁気ディスク制御装置のための誤り訂正符号（ECC）を設計開発した．

従来，磁気ディスク制御装置には7バイトECCが実装され，11ビットまでのバースト(集中)誤りを訂正していたが，記録密度の増加に伴いバースト誤り訂正能力を向上させる必要性が出てきたからである[1]．

今回開発した14バイトECCは，18ビットまでのバースト誤りを訂正し，それ以外の誤りのほとんどを検出することができる．なお，符号設計に当たっては7バイトECC同様，一般化Fire符号を採用した．

本稿では符号理論の概要，ECCに課せられた条件を述べた上で，ECCの設計手段，ECCの能力評価手段および得られた結果を述べ，さらに論理回路とマイクロコードの設計方法の説明を行う．

Abstract For magnetic disk storage control unit, ECC(Error Correcting Code) is designed. In the past, 7-byte ECC, that was able to correct burst error up to 11 bits, was used for the magnetic disk storage control unit. But because of increase of bits density, it needs to improve the error correction capability of ECC[1].

Newly designed ECC that we call 14-byte ECC can correct burst errors up to 18 bits, and detect almost all of other errors. 14-byte ECC belongs to the class of generalized Fire code.

This paper describes the outline of ECC, constraint for the design, development method, evaluation tool and development methodology for circuits and micro codes.

1. はじめに

磁気ディスク記憶装置におけるデータ誤りの発生原因としては，ほこりなどの異物，記録媒体の塗布むら，読み取り時のヘッドの位置ずれ，隣接トラックとのクロストークなどが考えられている[2]．

こうした原因によるデータ誤りは，バースト状（ある場所に塊状に集中的に）に発生するという特徴をもっており，ディスク装置にはシステムに及ぼす影響を考慮し，こうしたバースト誤りを訂正する強力な誤り訂正符号(Error Correcting Code-ECC)装置が実装されてきた．バースト誤りの幅の分布は，データ誤りの原因から容易に想像できるようにTPI（Track Per Inch）やBPI（Bit Per Inch）に依存する[2]．

逆に言うと，ECCは，そのバースト誤り訂正能力を想定したうえでTPI, BPIおよび記憶装置の信頼性目標が設定されるという意味で，記憶装置の信頼性設計といわばトレード・オフの関係にある．

従来，磁気ディスク装置では11ビットまでのバースト誤りを訂正する7バイトECCが使われてきたが，TPI, BPIの増大に伴い，16ビット程度までのバースト誤り

を訂正できるECCを開発する必要性が出てきた[1].

本稿で述べる14バイトECCは，こうした要請に応えるべく設計開発したもので，18ビットまでのバースト誤りを訂正し，それ以上の幅をもつ誤りも極めて高い確率で検出することができる．

14バイトECCでは，7バイトECC同様一般化Fire符号を採用した．回路設計技術およびシステム整合性などの継続性を重視したからである．

以下，ECCの原理，ECCの設計方法と設計結果の評価，論理回路などの実現方法について述べる．

2. 誤り訂正符号の基礎

2.1 誤りのタイプと誤り訂正符号装置のモデル

ECCは対象とするデータ誤りのタイプ（図1）により，以下の三つに分類できる[3].

1) ランダム誤り訂正符号
2) バイト誤り訂正符号
3) バースト誤り訂正符号

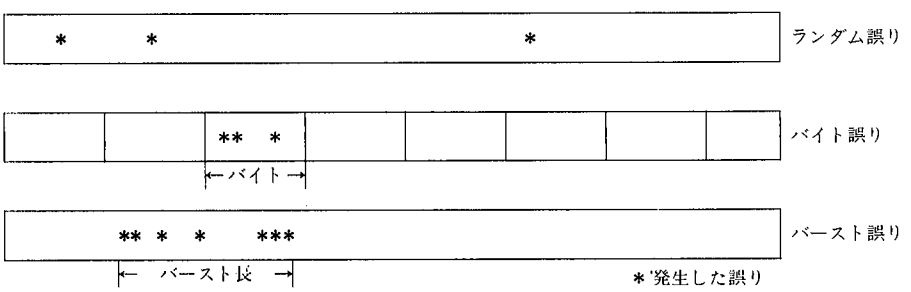


図1 誤りのタイプ
Fig. 1 Type of data error

ランダム誤りは，誤りパターンおよび誤り位置がデータ転送の単位であるデータブロック内にランダムに発生するタイプを指す．バイト誤りは，データ・ブロックをさらに分割した小ブロック(たとえばバイト)単位に発生する誤りを指す．バースト誤りは，かたまって発生する誤りであり，バイト誤りはバースト誤りの特殊な場合である．

メモリーには1）のランダム誤り訂正符号もしくは2）のバイト誤り訂正符号が，テープやディスクや通信装置には3）バースト誤り訂正符号（最近ではバイト誤り訂正符号も）が使われてきた．

メモリーICが1ビット×64Kアドレスのような構成をとっている場合，たとえば，1語＝32ビットとすると，1語は32個のメモリーICにまたがっており，1語内での誤りはランダムに発生する．

またメモリーICが1バイト（＝8ビット）×64Kの構成をとっているような場合は，誤りはバイト単位で発生し，2)のバイト誤り訂正符号が使われる．最近バイト誤り訂正符号がさかんに研究されているが，これはバイト単位のメモリーICの開発普及によるものである．

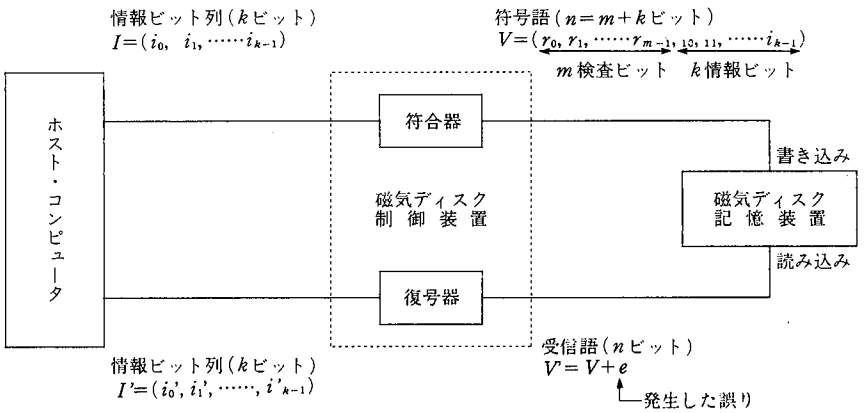


図 2　ECC 装置のモデル

Fig. 2　Model of ECC

ディスク装置はビット単位にシリアルにデータを格納しており，バイトといった切れ目は記録方式上も，読み取り方式上も存在しない．また誤りの主たるものは，1章で述べたようにバースト状に発生する．ディスクの ECC にバースト誤り訂正能力が使われるゆえんである[4]．

図2にディスクを例に ECC 装置のモデルを示した．書き込み時には，符号器が情報ビット列に検査ビットと呼ぶ冗長ビット列を付加する．これを符号化と呼び，検査ビットを付加したものを符号語，符号語の長さ $n=m+k$ を符号長と呼び，こうした符号を (n,k) 符号と呼ぶ．

読み込み時には，復号器が記憶装置から送られた情報系列（これを受信語と呼ぶ）を冗長性に基づいて検査する．

受信語が正しい符号語，すなわち検査ビットと情報ビットとが正しい関係にあれば，情報ビットをとり出してホストへ渡す．

誤りが含まれていると判断したら，（復号器が想定している誤りパターン内で）誤り訂正を試みる．誤り訂正に成功すると訂正したものをホストへ渡す（誤り訂正）．誤り訂正に失敗した時は誤りが検出された旨をホストへ送る（誤り検出）．

ECC とは，正しい符号語の全体（これを符号と呼ぶ）を定めることであり，次の二つの変換を定めることなのである．

情報ビット列　I　⟶符号語　V

受信語　$V'(=V+e)$→情報ビット列 I'

2.2　線形巡回符号と生成多項式[3][4]

定義 2.1（線形巡回符号）

符号 C（正しい符号語の集合）が以下の 1)～3) を満たすとき，これを線形巡回符号と呼ぶ．

1)　$(0,0,\cdots\cdots,0)\in C$

2)　$(v_1,v_2,\cdots,v_n)\in C$，　$(w_1,w_2,\cdots,w_n)\in C$ ならば，$(v_1+w_1,v_2+w_2,\cdots,v_n+w_n)\in C$

3)　$(v_1,v_2,\cdots,v_n)\in C$ ならば，$(v_n,v_1,v_2,\cdots,v_{n-1})\in C$

1) は情報ビットがすべて0ならば検査ビットもすべて0であることを，2) は正しい符号語の和も正しい符号語であることを，3) は正しい（正しくない）符号語はシフトしても正しい（正しくない）符号語であることを示している．

2) でビットの加算を行った．ビット間の演算を定義しておこう．

定義 2.2（二元体）

集合 B＝{0,1} に以下に示す四則演算＋－×÷を導入したものを二元体と呼ぶ．

$1+1=0$　$1-1=0$　$1\times a=a\times 1=a$

$0+1=1$　$0-1=1$　$0\times a=a\times 0=0$

$1+0=1$　$1-0=1$　$a\div 1=a$

$0+0=0$　$0-0=0$　（ただし$a=0$ または 1）

ビット列の多項式表現

ビット列 $(v_0,v_1,v_2,\cdots,v_{n-1})$ に対して，

多項式 $V(x)=v_0+v_1x+v_2x^2+\cdots v_{n-1}x^{n-1}$

を対応させる．

（例）$(101001)\longleftrightarrow 1+x^2+x^5$

多項式表現を用いると，符号すなわち正しい符号語の集合 C を定めることは，“正しい符号多項式”の全体 Cx を定めることである（以後 C,Cx は区別しないことにする）．Cx が線形巡回符号の場合，Cx は次の定理により特徴づけられる．

定理　2.1

(n,k) 線形巡回符号 Cx に対して，次の 1)，2) をみたす多項式 $G(x)$ があって，

1)　$\deg G(x)=m$　（$m=n-k$, $\deg G(x)$ は $G(x)$ の次数）

2)　$G(x)|x^n-1$　（x^n-1 が $G(x)$ で割り切れる）

$Cx=\{G(x)$ で割り切れる $n-1$次以下の多項式の全体$\}$ が成り立つ．

上記の $G(x)$ を符号 Cx の生成多項式と呼ぶ．

2.3　割り算回路と符号器，復号器

1)　符号器……符号器は生成される符号語が生成多項式 $G(x)$ で割り切れるように符号化する．したがって情報ビット $I=(i_0,i_1,\cdots,i_{k-1})$，もしくは情報多項式 $I(x)=i_0+i_1x+\cdots+i_{k-1}x^{k-1}$ に対する検査ビット（検査多項式）は

$$R(x)=x^mI(x)\bmod G(x)\quad (G(x)\text{ で割った余り})$$

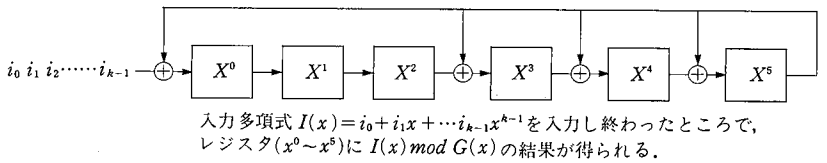
となる．

$$R(x)=r_0+r_1x+\cdots r_{m-1}x^{m-1}$$

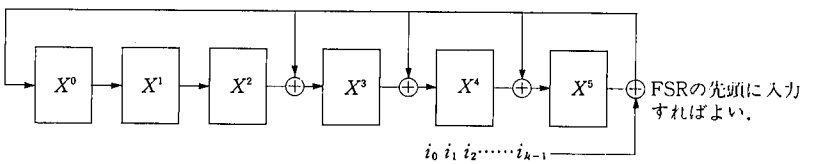
とおくと，符号語は，$(r_0,r_1,\cdots,r_{m-1},i_0,i_1,i_2,\cdots,i_{k-1})$ である．

2)　復号器……復号器は受信語（多項式）$V(x)$ に対し，$V(x)\bmod G(x)$ をチェックし，これが0ならば正しいとする．0でない場合は誤り訂正/検出を行う．誤り訂正の方法は符号方式により異なる．誤り検出のみ行う場合は，$V(x)\bmod G(x)$ の誤り算を行って0チェックを行えばよい（これは，CRC——Cyclic Redundancy Check——として知られているものである）．

3)　割り算回路……これら符号器，復号器では情報多項式 $I(x)$ に対する $x^mI(x)$ や受信多項式 $V(x)$ を生成多項式 $G(x)$ で割るという多項式の割り算回路が必要となる．この割り算回路を図3(a)，(b)に示した．図4には符号器と復号器の一



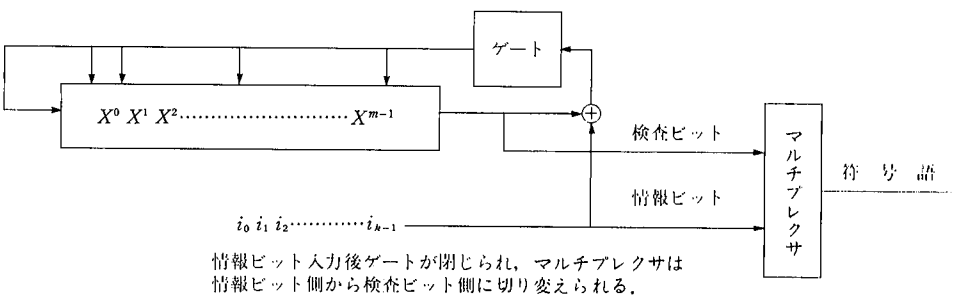
（a）　$G(x)=1+x^3+x^4+x^5+x^6$ による割り算回路



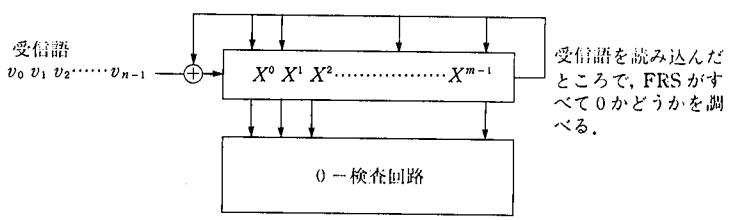
（b）　deg $G(x)=m$ のとき，$x^mI(x)$ mod $G(x)$ を求める回路（図では $m=6$）

図 3　割り算回路

Fig. 3　Circuit for divider



（a）　符号器



（b）　CRC 回路

図 4　割り算回路，符号化回路および CRC 回路

Fig. 4　Decoder and CRC-circuit

部として CRC 回路を示した．図 3, 4 ではフィードバックのかかっているシフト・レジスタを用いている．以後，これを FSR (Feedback Shift Register) と呼ぶ．

2.4　誤り訂正，誤り検出および訂正誤り[3]

ここでは，ECC 設計の基本として，誤り訂正 (error correction)，誤り検出 (error detection) および訂正誤り (miscorrection) の概念を説明する．

ECC の設計は，想定される誤りパターンの中から訂正したい誤りパターンの集合を定めることからスタートする．この訂正したい誤りの集合を E とし，各符号語 x_i に対し，$\Omega i=x_i+E$ とすると，E が誤り訂正できるためには，

$i \neq j$ に対し，$\Omega i \cap \Omega j = \phi$（空集合）

でなければならない．

なぜならば，$\Omega i \cap \Omega j \neq \phi$ とすると，x_i に訂正すべきか，x_j に訂正すべきかが判らなくなる誤りが存在してしまうからである．誤り訂正能力のみを考える分には $\Omega i \cap \Omega j = \phi$ の範囲内であれば，E を大きくすることが可能である．一方，想定される誤りパターンの中には，誤りの存在に気が付いてくれさえすればよいものもある．

これを E' とし，$\Omega i' = x_i + E'$ とすると，E' が誤り検出できるためには，

$i \neq j$ に対し，$\Omega i' \cap \Omega j = \phi$

でなければならない．

この範囲内であれば，誤り訂正もしくは誤り検出されるかというと，実はそうとはならない．ECC には必ず訂正誤りというやっかいな問題がつきまとう．

符号語 x_1 に誤り e'' が付加されたとする（図 5）．$x_1 + e'' \in \Omega_2$ の場合は受信語は x_2 に誤り訂正せざるを得なくなる（もちろん，$e'' = x_2 - x_1$ の場合は誤りが含まれていないと判断される）．これが訂正誤りである．誤り訂正を行う限り訂正誤りの確率は決して 0 にはならないし，誤り訂正能力を余り高くすると訂正誤りの確率が高まる危険性が出てくるのである．

以上から，符号設計においては，誤り訂正能力と同時に誤り検出能力（すなわち，訂正誤りの危険性）の評価が不可欠なことがわかる（3.3 節，4.3 節参照）．

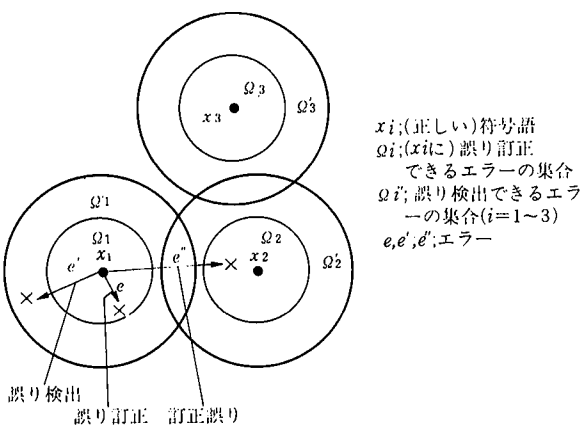


図 5　誤り訂正，検出および訂正誤りの概念図[3]

Fig. 5　Conceptual figure of error-correction, error-detection and miscorrection

3. 一般化 Fire 符号

3.1 数学的準備

ここでは一般化 Fire 符号の理解のために必要となる数学的道具立てと，その意味を述べることにする．

定義　3.1（既約多項式）

'1' および自分自身以外の多項式で割り切れない（すなわち，因子をもたない）多項

を既約多項式と呼ぶ．

定義　3.2（周期）

多項式 $G(x)$ に対し，$G(x)|x^n-1$ なる n の中の最小のものを $G(x)$ の周期と呼び，per $G(x)$ と書く．周期の概念は復号方法の基本をなしているので，その意味を割算回路と関連づけて説明しておく．

データ・ブロック内で，あるビット列を（右へ）α だけシフトすることは，多項式表現では，x^α をかけることに対応する．

たとえば，$(\underline{1,0,1,1,}0,0,0)\longleftrightarrow 1+x^2+x^3$　　$(0,0,\underline{1,0,1,1,}0)\longleftrightarrow x^2(1+x^2+x^3)$

よって，FSR に入力多項式 $F(x)$ を入力後（FSR$=F(x)\bmod G(x)$）入力を閉じて FSR 内で α 回フィードバック・シフトした時の FSR の値は，$x^\alpha F(x)\bmod G(x)$ となる（図6）．

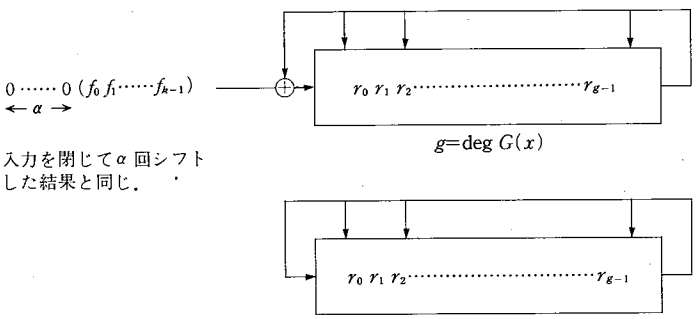


図 6　$F(x)=f_0+f_1x+\cdots f_{k-1}x^{k-1}$ に対する $x^\alpha F(x)$ mod $G(x)$ の計算

Fig. 6　Calculation of $x^\alpha F(x)\bmod G(x)$

このことから，$G(x)$ を既約多項式，$F(x)$ を deg $F(x)<$deg $G(x)$ なる多項式とすると以下の1)，2)が成り立つ．

1)　$x^{\text{per } G(x)}F(x)\bmod G(x)=F(x)$

2)　$0\leqq\alpha,\beta<\text{per } G(x),\alpha\neq\beta$ ならば，$x^\alpha F(x)\bmod G(x)\neq x^\beta F(x)\bmod G(x)$

すなわち，$F(x)$ を入力後周期回フィードバック・シフトすると元にもどり，かつその間の FSR の値はすべて異なっている．これが周期の意味である（図7）．

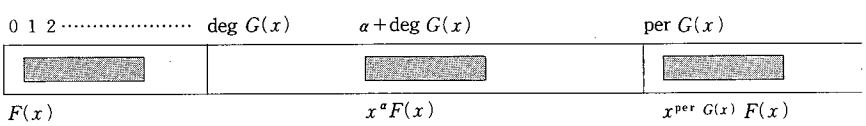


図 7　多項式 $x^\alpha F(x)$ の $G(x)$ の周期内での相対位置

Fig. 7　Relative position of $x^\alpha F(x)$ in the period of $G(x)$

なお，$G(x)=P_1(x)\cdot P_2(x)\cdots Pr(x)$ で，各 $Pi(x)$ の周期が互いに素ならば，

$$\text{per } G(x)=\text{per } P_1(x)\cdot\text{per } P_2(x)\cdots\text{per } Pr(x)$$

となる．

最後に，誤り位置を求めるために必要となる定理を示しておく．この定理により，

互いに素な数 c,e が与えられているとき，$r \bmod c$ と $r \bmod e$ とから $r(<c\cdot e)$ を求める手段が提供される．

定理 3.1（中国剰余定理—Chinese Remainder Theorem）

c,e を互いに素な整数とすると，以下の 1) 2)が成り立つ．

1) $A_e\cdot e+A_c\cdot c\equiv 1 \bmod c\cdot e$ なる A_e,A_c が存在する．

2) $r\equiv r_c \bmod c$, $r\equiv r_e \bmod e$ ならば

$r\equiv A_e\cdot e\cdot r_c+A_c\cdot c\cdot r_e \bmod c\cdot e$ が成り立つ．

なお，1)の A_e, A_c は Euclid 互除法によって求めることができる．また，この定理は互いに素な数 c,e が 3 個以上の場合にも拡張することができる．

3.2 一般化 Fire 符号の性質

Fire 符号は次の形の $G(x)$ を生成多項式とする線形巡回符号であり，高いバースト誤り訂正能力をもつ[7]．

$$G(x)=(x^c+1)\cdot P(x) \quad (P(x)：既約多項式)$$

しかし，Fire 符号は，符号長を長くしようとすると復号時間が符号長に比例して増大するという弱点をもち，このままではディスクのように長い符号長を必要とするものには使えない[6]．この問題を解決したのが R.T.Chien である[5]．

Chien は，生成多項式 $G(x)$ を

$$G(x)=(x^c+1)\cdot \prod_{i=1}^{l} P_i(x) \quad (P_i(x)：既約多項式)$$

の形に一般化し，この一般化 Fire 符号も Fire 符号と同様の性質をもち，しかも $P_i(x)$ を追加することにより，復号時間を変えることなく符号長を長くできることを示した．一般化 Fire 符号の性質を定理の形で述べておく．

定理 3.2（Fire-Chien）

$G(x)=(x^c+1)\cdot \prod_{i=1}^{l} P_i(x)$ が以下の 1)～3)の条件を満たすとする．

1) $P_i(x)(i=1\sim l)$ は互いに異なる既約多項式で，

次数は，$\deg P_i(x)=m_i$，周期 per は，$P_i(x)=e_i$

2) $c,e_1,e_2,\cdots e_l$ は互いに素である．

3) ある $b(>0)$ があって，$c\geqq 2b-1, m_i\geqq b$ $(i=1\sim l)$

このとき，$G(x)$ を生成多項式とし，以下のⓐ～ⓒの性質をもつ線形巡回符号 C_G を作ることができる．

ⓐ C_G の符号長は，$n=c\cdot \prod_{i=1}^{l} e_i$，検査ビット長は，$r=c+\sum_{i=1}^{l} m_i$

ⓑ C_G はバースト長 b 以下のバースト誤りを訂正する．

ⓒ $n\gg r,b$ ならば b ビット以上のバースト誤りに対しても，$(1-nr/2^{r-b})$ 以上の確率で誤り検出できる．

1)～3)の持つ意味を説明しておく．

1)は，Chien の高速復号式を可能にするための条件である．

2)は，必ずしも必要ではないが，符号の効率上このように仮定しておく．

3)は，b ビット・バースト誤りが訂正可能であるための条件（2.4 節における $\Omega i\cap \Omega j=\Phi$ を保証する条件）である．

3.3 Peterson-Chienの高速復号方式

一般化Fire符号の符号器は2.3節に述べた方法で実現できる．

すなわち，$G(x)=(x^c+1)\cdot\prod_{i=1}^{l}P_i(x)$ を展開したものに基づいて割り算回路を構成すればよい．一般化Fire符号の特徴はその復号法にある．Fireの元々の復号法では $G(x)$ を展開したFSRが使われ，最悪ケースで $G(x)$ の周期分のフィードバック・シフトが必要である．Chienの工夫は (x^c+1) および，各 $P_i(x)$ ごとにFSRを用意し，それぞれで，それぞれの周期分のフィードバック・シフトを並行して行うことにより復号時間を大幅に短縮したことにある．

以下，この高速復号方式について説明する．

1) 誤り訂正……簡単のため，$l=1$ の場合を考える．生成多項式を $G(x)=(x^c+1)\cdot P(x)$, $\deg P(x)=m$, $\mathrm{per}\,P(x)=e$ とし，x^c+1 および $P(x)$ による割り算回路をそれぞれ S_0, S_1 とする．S_0, S_1 は各長さ c, m のレジスタであり，これをシンドローム・レジスタと呼ぶ(図8)．符号語 $X(x)$ に長さ b 以下のバースト誤り $F(x)$ が発生したとする．受信語を読み込むと $X(x)$ は，$X(x)\bmod G(x)=0$ であるから S_0, S_1 には $F(x)$ を x^c+1, $P(x)$ で割った結果が得られる．各シンドローム・レジスタの値は $F(x)$ のパターンおよび各周期 c,e 内での相対位置により一意に定まる（周期内での相対位置が同じならば値は変わらない．3.1節参照)．

S_0 は，長さが周期 c に等しく，フィードバックが先頭にしかかかっていない．そのため，$F(x)$ が周期に収まっていれば S_0 には $F(x)$ そのものが現れ，$F(x)$ が周期境界をまたいでいれば $F(x)$ を分割したものが S_0 に現れる(図8)．S_0 をフィ

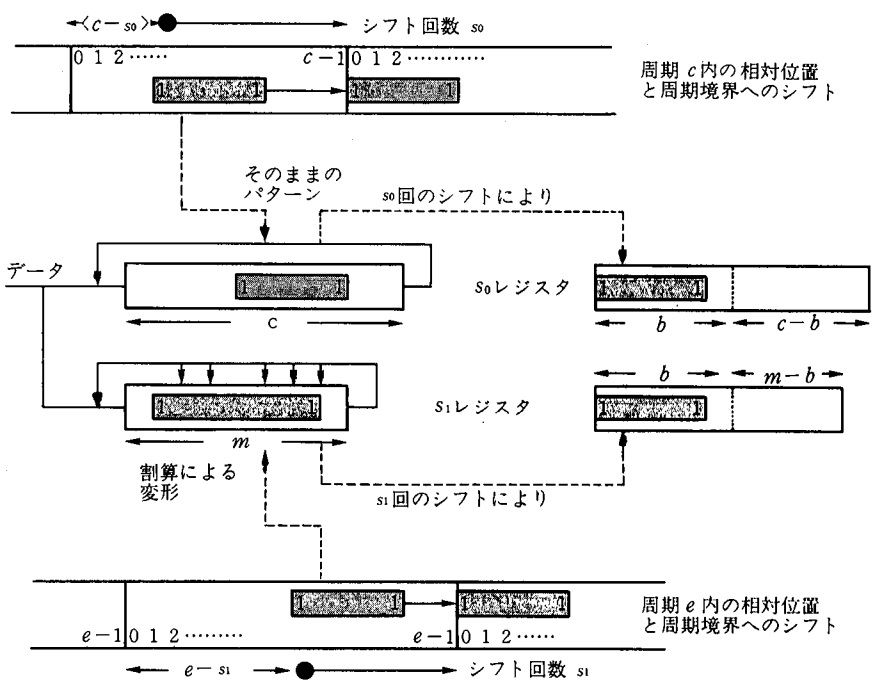


図8　シンドローム・レジスタの動き

Fig. 8　Action of syndrome register

ードバック・シフトすると，周期 c の境を越えたところで上位 $c-b$ ビットはすべて0になり，下位 b ビットに誤りパターン $F(x)$ が現れる．

一方，S_1 をフィードバック・シフトしていくと，やはり周期 e の境を越えたところで S_1 は $F(x)$ に等しくなる(それまでは決して等しくはならない(3.1節参照)．S_0, S_1 のシフト回数をそれぞれ s_0, s_1 とすると，図8に示したように，$c-s_0$, $e-s_1$ が誤りパターン $F(x)$ の周期 c,e 内での相対位置を表すことになる．

すなわち，F の絶対位置 s は以下の式を満たしている．

$s \equiv c-s_0 \mod c \qquad s \equiv e-s_1 \mod e$

c, e は互いに素であるから，中国剰余定理により，s が求まり，S_0 に現れているパターンとあわせて誤り $F(x)$ を訂正できるのである．$l \geqq 2$ の場合はシンドローム・レジスタとして，$S_i(i=0\sim l)$ を用意すればよい（図14)．S_0 に誤りパターンが現れ S_i のシフト回数 $s_i(i=0\sim l)$ より中国剰余定理を用いて絶対位置 s が求められる．

2) 誤り検出……長さ b 以上のバースト誤りを誤り訂正することはできない．しかし，こうした誤りのほとんどに対しても，誤り検出が可能である．以下に誤り検出ケースを想定される発生確率の順に列挙する．

① S_0 を c 回フィードバック・シフトしても上位（$c-b$）ビットを0にできない場合

② $S_i(i=1\sim l)$ のどれかが，周期分シフトしても S_0 と一致しない場合

③ $l+1$ 個のシンドローム・レジスタの中に，0でないものと，0のものが混在している場合

④ 中国剰余定理を用いて求めた誤りの位置がデータブロック長を越えた場合

3) 訂正誤り……逆に，長さ b ビット以上の誤りで誤り検出できないのは以下のケースである．

⑤ 復号器が誤り $E(x)$ を発見できないケース．この場合は，誤り $E(x)$ を含んだままで，データが送られてしまう．

⑥ 誤りパターン $E(x)$ が b ビット以下の別パターン $B(x)$ に化けて，誤り訂正される場合．この時は，発生した誤りとは異なる誤りを含んだデータが送られる．

これらの発生条件を以下に示す．

⑤ 生成多項式 $G(x)$ で割り切れる誤りが発生した場合，復号器はこれを判別することはできない．ただし，この確率はきわめて低く，$1/2^{\deg G(x)}$ である．

⑥ これは $F(x)$ が生成多項式 $G(x)$ と誤り訂正可能である b ビット以下の誤り多項式 $B(x)$ を用いて $E(x)=H(x)\cdot G(x)+B(x)$ と表せるケースであり，この確率も非常に低い（4.3節参照)．

4. 14バイト ECC の符号設計

4.1 問題の設定

ディスクに使える16ビット以上のバースト誤り訂正符号を設計するためには，以下の1)～4)の条件を満たす一般化 Fire 符号の生成多項式

$$G(x)=(x^c+1)\cdot\prod_{i=1}^{l}P_i(x)$$

を求めなければならない．

なお，$P_0(x)=x^c+1$ とおき，$\deg P_i(x)=m_i$，$\mathrm{per}\ P_i(x)=e_i$ すなわち，$m_0=e_0=c$ とする．

1) $b\geqq16$ なる b に対して定理3.2の条件を満たさなければならない．

2) m_i,e_i はなるべく小さいものにする．m_i は16が望ましく，e_i は大きさのそろったものが望ましい．

3) 符号長 n および検査ビット長 r は8の倍数であり，r はなるべく小さく，$n\geqq320000$ でなければならない．

4) 従来使われていた7バイトECCに対して，誤り検出能力の向上と訂正誤り率の減少していなければならない．

以下に1)～4)の条件の説明を行う．

1) 16ビット以下のバースト誤り訂正という仕様の根幹である．なお，回路の並列化（5章で述べる）のためには，1)の制約下での最小値16が望ましい．

2) m_i はシンドローム・レジスタの長さ，すなわち論理規模を決定する．e_i は復号時のレジスタのシフト回数であり，$\max_i e_i$ が復号時間を規定する．

符号長 $\prod_{i=1}^{l}e_i$ を長く，復号時間 $\max_i e_i$ を短くするためには e_i $(i=0\sim l)$ をそろえることが望ましいのである．なお，$b\geqq16$，$c\geqq2b-1\geqq31$ であるから，c および各 e_i は30近辺の互いに素な数でなければならない．

3) ディスク制御装置における処理の関係上，n,r は共に8の倍数でなければならない．検査ビット r を大きくすると，ECCの能力は高まるがディスクに格納できる情報ビットの比率が下がるため，r は小さい方が(当然)よい．また，ディスクの1トラックは4万バイトすなわち32万ビットであり，ディスク装置としては最大レコード長は32万ビットまで許している．1符号語＝1レコードであるから，この条件が必要となる．

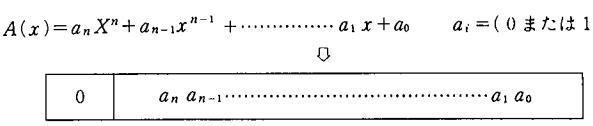
4) 7バイトECCの能力を上回るという当然の要求である．

4.2 符号設計ツールと求められた符号

符号設計には，膨大な多項式の割り算をする必要があり，1回の割り算の時間はなるべく少なくしたい．そのため，多項式を図9のようなビットパターンで表現している．

このデータ構造を用いた多項式の割り算の原理を図10に示す．

$A(x)$ と $G(x)$ のEOR(Exclusive Or)をとり，リーディング0をLSC(Load Shift



先頭の0は割り算時のシフトのための番兵である．

図9　多項式のデータ表現

Fig. 9　Data structure of polynomial

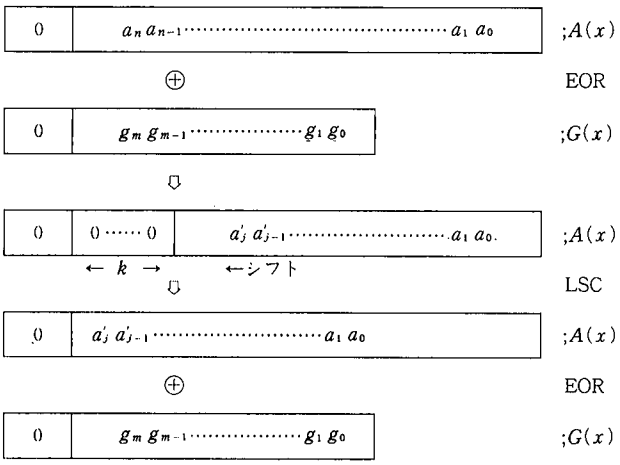


図 10　多項式の割り算の原理

Fig. 10　Principle of division

```
DO 次数k＝2, 34
    FOR all k－次既約多項式F(x)
        FOR all 2^k－1の因子 e
            IF((x^e＋1)mod F(x)＝0) THEN
                eをF(x)の周期として登録する
                EXIT (FOR)
            ENDIF
        ENDFOR
    ENDFOR
ENDDO
```

図 11　既約多項式の周期を求めるプログラム

Fig. 11　Program for the period of irreducible polynomial

and Count) 命令でシフトし，これを $\deg A(x) < \deg G(x)$ になるまで繰り返せばよい．

1)　既約多項式の周期を求めるプログラム……生成多項式を求めるためには，適当な次数と周期を持つ既約多項式が必要であり，そのためには既約多項式の周期を求めなければならない．このプログラムを図 11 に示した．なお，ここでは，次数 k の既約多項式 $F(x)$ の周期が 2^k-1 の約数になるという事実を用いている．

2)　求まった既約多項式と周期……求まった既約多項式のうち，次数 ≦100 のものの一部を以下に列挙する．

(例) 1021 <=> $\overset{x^9}{1}\ 0\ 0\ 0\ 0\ \overset{x^4}{1}\ 0\ 0\ 0\ \overset{x^0}{1}$ <=> x^9+x^4+1

deg＝18 ；　1001001(27)　1341035(57)　1777777(19)

deg＝20 ；　4100001(75)　4102041(25)　6647133(41)　7164555(55)

deg＝21 ；　10040001(49)

deg＝22 ；　34603145(69)

deg＝23 ；　43073357(47)

deg＝28 ；3706175715(87)　3777777777(29)

deg＝30 ；11000100011(99)　16471647235(77)

なお，多項式は8進表現しており，（ ）内は周期を示す*.

アンダーラインを引いた三つの多項式は，次数も周期も小さくかつ周期は互いに素である．これらを$P_1(x)$～$P_3(x)$とし，$c=56$として，生成多項式を構成することにした．これを検査ビット長＝112ビット（14バイト）にちなんで14バイトECCと呼ぶことにする．

改めて7バイトECCと14バイトECCの生成多項式を示しておく．

3) 7バイトECCの生成多項式

$$G(x)=(x^{22}+1)\cdot P_1(x)\cdot P_2(x)\cdot P_3(x)$$

$$P_1(x)=x^{11}+x^7+x^6+x+1 \quad (89)$$

$$P_2(x)=x^{12}+x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1 \quad (13)$$

$$P_3(x)=x^{11}+x^9+x^7+x^6+x^5+x+1 \quad (23)$$

4) 14バイトECCの生成多項式

$$G(x)=(x^{56}+1)\cdot P_1(x)\cdot P_2(x)\cdot P_3(x)$$

$$P_1(x)=x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1 \quad (19)$$

$$P_2(x)=x^{18}+x^9+1 \quad (27)$$

$$P_3(x)=x^{20}+x^{15}+x^{10}+x^5+1 \quad (25)$$

4.3 14バイトECCの能力評価

はじめに14バイトECCの訂正誤り確率を求めておく．一般に訂正誤り確率は誤りパターンのパターン幅に関係なく，$n\cdot r/2^{r-b}$以下であることが証明できる（定理3.2)．ここで，n；符号長，r；$G(x)$の次数，b；バースト誤り訂正能力である．

ここでは，パターン幅$\leqq r$の場合の訂正誤り確率を詳細に求めておく．誤りパターン$D(x)$ ($\deg D(x)<\deg G(x)$) が訂正誤りされるための必要十分条件は

$$D(x)=x^k\cdot B(x) \bmod G(x)$$

なるパターン$B(x)$（$B(x)$のパターン幅$\leqq b$）と位置k($r-b<k\leqq n-b$）が存在することである**.

すなわち，$\deg G(x)$以下の訂正誤りパターンを求めるには，18ビット以内のビット・パターン$B(x)$に，次々にx^kをかけながら$G(x)$で割っていけばよい．プログラムを図12に示す．

このプログラムより，長さ72ビット以内のバースト誤りパターンは正しく誤り検出され，決して訂正誤りされないことがわかった．

以上の事実を含めて7バイトECCと14バイトECCとの性能比較を表1に示す．

---

* なお，deg＝16～30の中に100以下の周期は上記以外には存在せず，deg＝16,17,19,24,25,26,27,29の多項式の中には100以下の周期のものはない．

** 次頁脚注を参照．

```
DO ビット長b=1～18
    FOR すべてのbビット・パターンB(x)に対して
        k←r−b+1
        D(x)←x^k・B(x)
        WHILE k≦n−b DO
            D(x)←D(x) mod G(x)
            D(x)の長さlおよびリーディング0の数zを求める
            誤訂正パターン・テーブル(l)を1インクリメント
            k←k+z
            D(x)←x^k・D(x)
        ENDWHILE
    ENDFOR
ENDDO
```

n：データブロック長　r：生成多項式の次数

図 12　誤訂正される誤りパターン長の分布を求めるプログラム

Fig. 12　Program to obtain the distribution of error pattern length that is miscorrected

---

＊＊（前頁より）　誤りパターン $D(x)$ ($\deg D(x)=d$) が発生したとき，これが誤りパターン $B(x)$ ($\deg B(x)\leqq b$) に訂正誤りされるのは，

$$[x^i\cdot D(x)+x^j\cdot B(x)]\bmod G(x)=0 \quad\cdots\cdots(1)$$

なる $i,j(0\leqq i,j\leqq n-1)$ が存在する時であることがわかる．なぜならば，$[x^i\cdot D(x)+x^j\cdot B(x)]\bmod G(x)=0$ とすると，$x^i\cdot D(x)=x^j\cdot B(x)+G(x)\cdot F(x)$ であり，$G(x)\cdot F(x)$ に対するシンドローム・レジスタの値は0であるから，$x^i\cdot D(x)$ は $x^j\cdot B(x)$ と解釈され，訂正誤りが起きてしまう．

逆に，復号器が誤り訂正プロセスに入り，誤り訂正に成功するのは(1)式が成り立つ時だけであることが復号器の動きから証明されるからである．

いま，$D(x)$ のパターン長が $\deg G(x)$ 以下のケースを考えよう．

1)　$j>i$ の場合は，

$$x^i\cdot(D(x)+x^{j-i}\cdot B(x))\bmod G(x)=0$$

$G(x)$ が x の巾乗の因子を持たず，$\deg D(x)<\deg G(x)$ であるから，

$$D(x)=x^{j-i}\cdot B(x)\bmod G(x)$$

すなわち，訂正誤りされるパターンは，誤り訂正できるパターン $x^{j-i}\cdot B(x)$ を $G(x)$ で割ったパターンなのである．

2)　$j=i$ の場合

$x^i\cdot(D(x)+B(x))\bmod G(x)=0$ となり，$\deg(D(x)+B(x))<\deg G(x)$ であったから，このケースは起こりえない．

3)　$j<i$ の場合

一般に多項式 $H(x)$ の相反多項式 $\tilde{H}(x)$ を次の式で定義しておく．

$$\tilde{H}(x)=x^{\deg H(x)}\cdot H(1/x)$$

14バイトECCの生成多項式は，低次と高次が対称になるように作られており，

$\tilde{G}(x)=G(x)$ である．

式(1)より，

$$[x^{i-j}\cdot D(x)+B(x)]\bmod G(x)=0$$

すなわち，

$$x^{i-j}\cdot D(x)+B(x)=G(x)\cdot F(x)$$

両辺の相反多項式をとると，

$$\tilde{D}(x)+x^{i-j+d-b}\cdot\tilde{B}(x)=\tilde{G}(x)\cdot\tilde{F}(x)\cdot\tilde{F}(x)=G(x)\cdot\tilde{F}(x)$$

となって，これは1)のケースに含まれる．

表 1　7および14バイトECCの性能比較
Table 1　Comparison between 7-byte ECC and 14-byte ECC

	7バイトECC	14バイトECC
最大データ・ブロック長	585,442	718,200
検査ビット長	56ビット	112ビット
同上ブロック内比率*	0.08%	0.16%
回路規模	レジスタ長=56	レジスタ長=112
復号スピード（ハードウェア）	111クロック	83クロック
バースト誤り訂正能力	11ビット	18ビット
訂正誤りの確率	10^{-6}以下	10^{-20}以下

＊ ブロック長は1792×36ビットとした時の比率

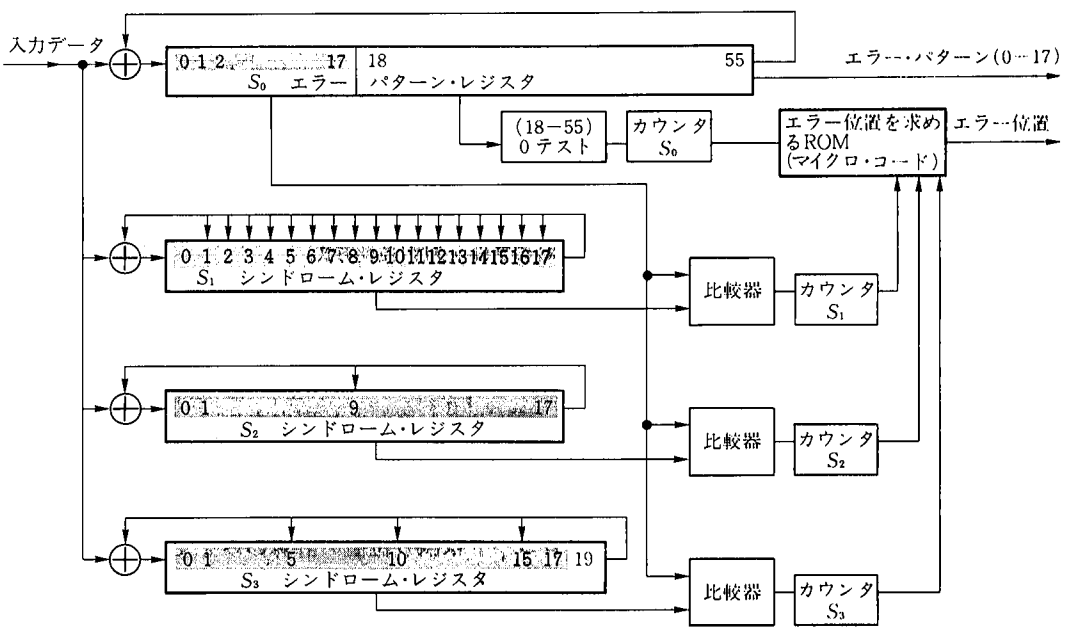


図 13　14バイトECCの復号器(直列方式)
Fig. 13　Decoder of 14-byte ECC(serial circuit)

5. 論理設計

5.1 14バイトECCの論理回路

14バイトECCではシフト・レジスタの並列化を行っている．これは，ディスク装置

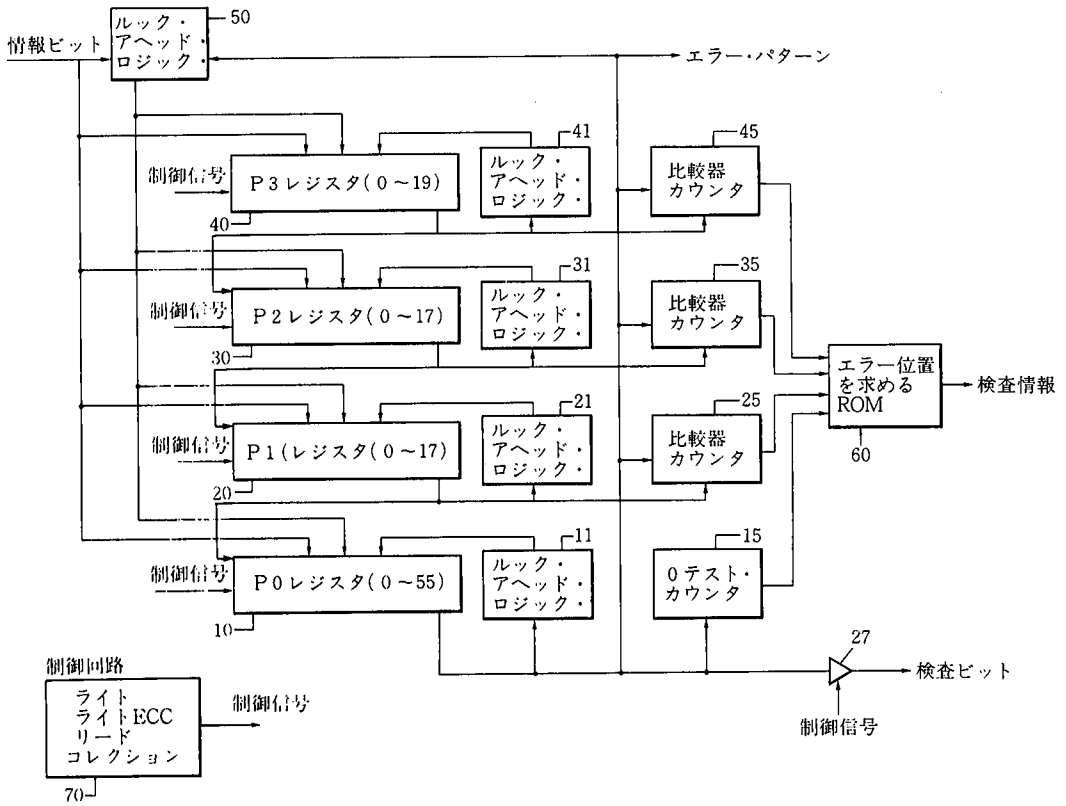


図 14　14バイト ECC のブロック図(並列方式)

Fig. 14　Block chart for 14 byte ECC(parallel circuit)

が高速になるにつれ，1ビット・タイムの間にレジスタのシフトを行うことがますます厳しくなってきているからである．レジスタのシフトを並列化すれば，FSR は一回のシフトに数倍の時間が使えることになり安定した動作が保証される．

図 13 に 14 バイト ECC の論理回路の概要を示す．ここで，ハードウェアはシフト回数をマイクロコードに教え，マイクロコードは誤り位置の計算のみを受け持つ．誤りパターンは S_0 レジスタから渡される．なお，誤り訂正そのものを行う回路は省略した．

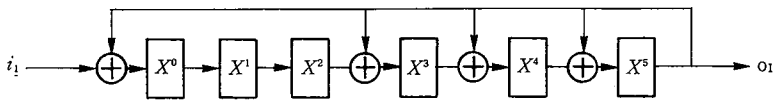
また，14 バイト ECC においては，符号器と復号器とで FSR を共有している．これを並列化したもののブロック図を，図 14 に示した．

5.2　割り算回路の並列化

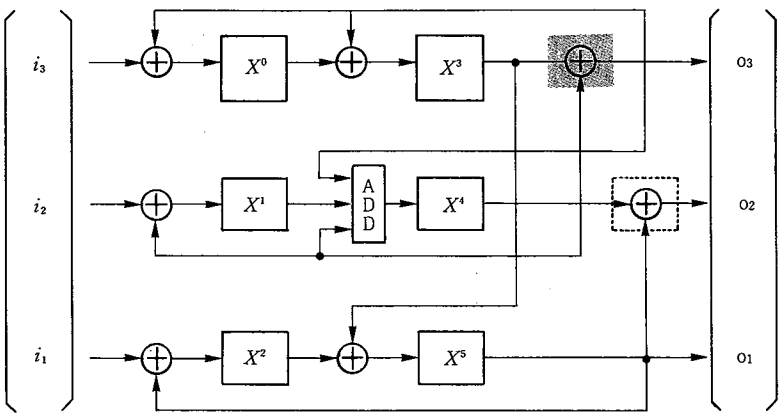
図 15(a)の FSR を 3 ビット並列化することを考える．

FSR は，内部状態を持ち入力 i_1 に対して o_1 を出力し，内部状態 $S_1=(x_1^0,x_1^1,\cdots,x_1^5)$ を $S_2=(x_2^0,x_2^1,\cdots,x_2^5)$ に変換する有限状態機械（finite state machine——FSM）と考えることができる（図 16(a)）．

S_1,i_1 と S_2,o_1 および引き続き i_2,i_3 が入力された時の S,i,o は，以下の式で関係づけられる．



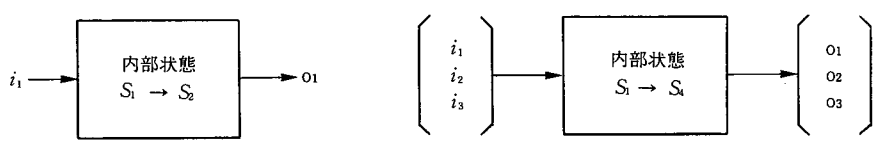
（a）　直列FSR



（b）　3ビット並列FSR

図 15　$1+x^3+x^4+x^5+x^6$ による割り算回路[10]

Fig. 15　Divider by $1+x^3+x^4+x^5+x^6$



（a）　直列FSM　　（b）　並列FSM

図 16　FSRのFSM表現

Fig. 16　FSM representation for FSR

$$\begin{bmatrix} S_{k+1} \\ O_k \end{bmatrix} = \begin{bmatrix} T \end{bmatrix} \cdot \begin{bmatrix} S_k \\ i_k \end{bmatrix} \quad (k=1,2,3) \tag{5-1}$$

ここで，S_i, T は以下の通りであり，T を変換行列，T_{11} をコンパニオン行列と呼ぶ．

$t_{S_i}=(x_i^0\ x_i^1\ x_i^2\ x_i^3\ x_i^4\ x_i^5)$

$$T=\left[\begin{array}{c|c} T_{11} & T_{12} \\ \hline T_{21} & T_{22} \end{array}\right]$$

$$T_{11}=\begin{bmatrix} 000001 \\ 100000 \\ 010000 \\ 001001 \\ 000101 \\ 000011 \end{bmatrix} \qquad T_{12}=\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

フィードバック列—↑

$$T_{21}=(000001) \qquad T_{22}=(0)$$

このFSMを3ビット並列化するには，内部状態 S_1 および入力 i_1〜i_3 から次の内部状態 S_4 および出力 o_1〜o_3 を導く変換 $T^{(3)}$ を求めればよい．

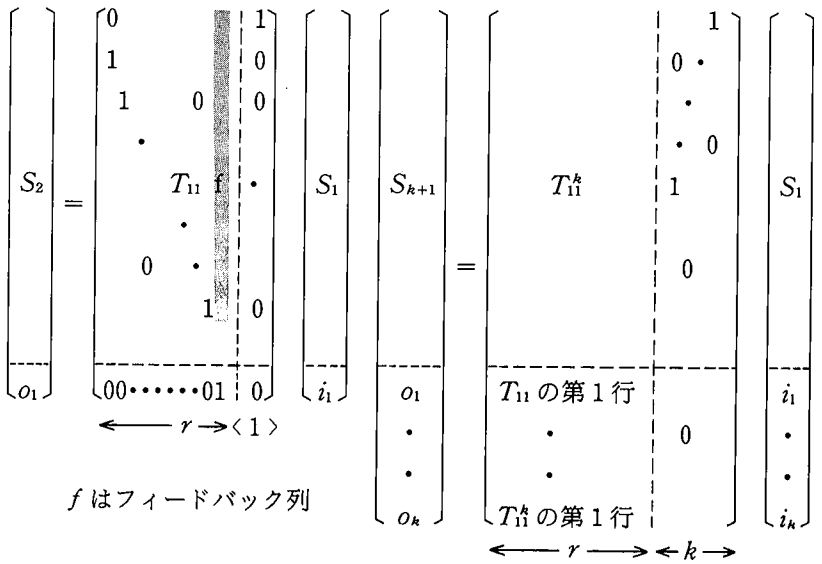
これは，式(5-1) を展開し整理して以下のように求まる．

$$\begin{bmatrix} S_4 \\ \cdots \\ o_1 \\ o_2 \\ o_3 \end{bmatrix} = \left[\begin{array}{c|c} T_{11}^{(3)} & T_{12}^{(3)} \\ \hline T_{21}^{(3)} & T_{22}^{(3)} \end{array}\right] \begin{bmatrix} S_1 \\ \cdots \\ i_1 \\ i_2 \\ i_3 \end{bmatrix} = \left[\begin{array}{c|c} 000110 & 001 \\ 000011 & 010 \\ 000001 & 100 \\ 100110 & 000 \\ 010101 & 000 \\ 001100 & 000 \\ \hline 000001 & 000 \\ 000011 & 000 \\ 000110 & 000 \end{array}\right] \begin{bmatrix} x_1^0 \\ x_1^1 \\ x_1^2 \\ x_1^3 \\ x_1^4 \\ x_1^5 \\ \cdots \\ i_1 \\ i_2 \\ i_3 \end{bmatrix}$$

ここで，$T_{11}^{(3)}$ は T_{11} の3乗，すなわち T_{11}^3 となっている．$T^{(3)}$ を回路化したものを図15(b)に示しておいた．実現の方法は，各行において，1のあるところで排他的論理和をとって，それぞれのフリップ・フロップまたは出力に出せばよい．また，□および　で示したところは共通であるので，図15(b)の回路でも共通にしている．

このように並列化した変換は共通部分をまとめることにより，回路の最小化が可能である．最小化のアルゴリズムについては文献[11]を参照されたい．

最後に，一般にフリップ・フロップが r 個あるFSRを k ビット並列化したFSMの変換行列を示しておく．ただし，$k<r$．



5.3 マイクロコードの設計[9]

図13で示したように誤り位置は，四つのシフト・レジスタにおけるシフト回数 s_0 ～s_3 から求められ，これを行うのがマイクロコードである．

中国剰余定理より，誤り位置は，

$A_i \prod_{j\neq 1} e_j \equiv 1 \bmod e_i$ $(i=0\sim3)$ なる A_i より，

$$p \equiv \sum_{i=0}^{3}(A_i \cdot \prod_{j\neq i} e_j)\cdot(e_i - s_i) \bmod \prod_{j=0}^{3} e_j$$

$$\equiv A_0e_1e_2e_3(e_0-s_0)+A_1e_0e_2e_3(e_1-s_1)+$$

$$A_2e_0e_1e_3(e_2-s_2)+A_3e_0e_1e_2(e_3-s_3) \bmod e_0e_1e_2e_3$$

しかし実際は S_0 レジスタのみは1ビットずつ，S_1～S_3 レジスタは4ビットずつシフトされるのでシンドローム・レジスタに渡されるのは，$s_0, 4s_1, 4s_2, 4s_3$ であり，

$$p \equiv A_0 \prod_{j\neq 0} e_j(e_0-s_0)+\sum_{i=1}^{3} A_i \prod_{j\neq i} e_j(e_i-4s_i) \bmod \prod_{j=0}^{3} e_j$$

$$\equiv -A_0 \prod_{j\neq 0} e_j s_0 - \sum_{i=1}^{3} A_i \prod_{j\neq i} e_j \cdot 4s_i \bmod \prod_{j=0}^{3} e_j$$

適当な $m_i(i=0\sim3)$ を用い，右辺に $\sum_{i=0}^{3} m_i s_i \cdot \prod_{i=0}^{3} e_i$ を足すと，

$$p \equiv s_0(m_0e_0-A_0)e_1e_2e_3+\sum_{i=1}^{3} s_i(m_ie_i-4A_i)\cdot \prod_{j\neq i} e_j \bmod e_0e_1e_2e_3$$

14バイトECCにおいては，中国剰余定理の係数 A_0～A_3 (1, 17, 11, 17——Euclid互除法による[9])を用いて，m_0～m_3 を，$0<m_0e_0-A_0<e_0, 0<m_ie_i-4A_i<e_i$ となるように選んである．

この m_0～m_3 を求めると，$m_0=1, m_1=4, m_2=2, m_3=3$ となり，次のようになる．

$$p \equiv B_0e_1e_2e_3s_0+B_1e_0e_2e_3s_1+B_2e_0e_1e_3s_2+B_3e_0e_1e_2s_3 \bmod e_0e_1e_2e_3$$

ここで，$B_0=55, B_1=8, B_2=10, B_3=7$

以下のような数列$\{r_z, p_z, q_z\}$（$z=0, 1, 2$）を求めていく．

$(r_1, r_2) \leftarrow (a, b)$

$(p_1, p_2) \leftarrow (0, 1)$；$(q_1, q_2) \leftarrow (1, 0)$

$z \leftarrow 1$

REPEAT

$(r_0, r_1) \leftarrow (r_1, r_2)$

$(p_0, p_1) \leftarrow (p_1, p_2)$；$(q_0, q_1) \leftarrow (q_1, q_2)$

$r_0 = c \cdot r_1 + r_2$なるr_2, cを求める．

$p_2 \leftarrow c \cdot p_1 + p_0$

$q_2 \leftarrow c \cdot q_1 + q_0$

$z \leftarrow z + 1$

［ループ不変式として，以下が成り立つ．

$q_1 \cdot r_2 + q_2 \cdot r_1 = b$

$p_1 \cdot r_2 + p_2 \cdot r_1 = a$

$q_2 \cdot p_1 - p_2 \cdot q_1 = (-1)^z$］

UNTIL $r_2 = 0$

$q_1 \leftarrow (-1)^{z+1} q_1 \mod b$

ループを出たところではr_1が1（一般には GCD(a, b)）になっており，

$b \cdot p_1 - a \cdot q_1 = (-1)^z$，$(-1)^{z+1} \cdot q_1 = a^{-1} \mod b$

となり，$q_1 \leftarrow (-1)^{z+1} q_1 \mod b$により，

$0 < q_1 = a^{-1} \mod b < b$

となる．

図 17　GCD$(a,b)=1$ のとき，$a^{-1} \bmod b$ を求めるプログラム

Fig. 17　Program for $a^{-1} \bmod b$ on condition of GCD$(a,b)=1$

これが，図13のマイクロコードが行っている誤り位置の計算方法である．

A_iを求めるツールを示しておく．

$A_i \cdot \prod_{j \neq i} e_j \equiv 1 \bmod e_i$なる$A_i$すなわち，$A_i = (\prod_{j \neq i} e_j)^{-1} \bmod e_i$は，$\prod_{j \neq i} e_j$と$e_i$の間のEuclid互除法により求められる．これを図17に示す．

なお，Euclid互除法は整数に対してだけではなく，多項式に対してもまったく同様の方法で可能である．多項式のEuclid互除法は，ECC回路の各種のテストケース・スタディにおいて極めて重要な役割を果たした．また，電源投入時の診断・初期設定プログラムPOC（Power On Confidence）や診断ソフトウェアの作成にも用いている．

6. おわりに

本稿では，紙数の関係からテストについては述べることができなかった．ここで簡単に触れておく．

テストは，

1)　FORTRANプログラムによる符号能力の検証

2)　DIANA（論理シミュレータ）による機能およびタイミングのテスト

3)　PTS（Peripheral Test System）による符号能力のテスト

の3段階に分けて行った．

1)はいわば数学的証明の検証である．2)では，主に基本機能と基本的タイミングのみを調べた．ロジックのバグのほとんどすべてがここでみつかっている．3)では，主に1)および2)において（計算時間の制約で）できなかったテストを実機で行った．

それぞれの段階で，生成多項式を求めるプログラムに似た大きさのツールをたくさん作った．この中にはテスト・ケースの生成，期待される結果の導出等も含まれている．

これらは，プログラム仕様的には簡単であるにもかかわらず，アルゴリズムを工夫しないと膨大な計算時間を食ってしまうものが沢山ある．符号理論と数学と計算量の考察とがわれわれを助けてくれた．

最後に，この仕事を行う機会を与えて下さったペリフェラル開発一室の紙田室長に感謝の意を申し述べ，謝辞としたい．

参考文献

［1］「装置容量が5Gバイトに達した大容量磁気ディスク装置」，日経エレクトロニクス，No. 378 1985-9-23.
［2］日本電信電話公社，電気通信研究所，「研究実用化報告」．Vol. 28, No. 10(800メガバイト磁気ディスク特集)，1979.
［3］宮川洋，原島博，今井秀樹，「情報と符号の理論」，(岩波講座 情報科学-4)，岩波書店，1982.
［4］S. Lin, D. J. Costello, Jr., "Error Control Coding", Prentice-Hall, 1982.
［5］R. T. Chien, "Burst-Correction Codes with High-Speed Decoding", IEEE Trans. Inf. Theory, IT-1, pp. 109-113, Jan., 1969.
［6］上谷彊輔，「符号理論と一般化 Fire 符号」，開発関連部門，Technical Memorandum.
［7］W. W. Peterson, E. J. Weldon, "Error-Correcting Codes", 2nd ed., MIT Press, 1972.
［8］M. Y. Hsiao, W. C. Carter, J. W. Thomas, Stringfellow, W. R., "Reliability, Availability, and Serviceability of IBM Computer Systems, : A Quarter Century of Progress", IBM J. RES. DEVELOP. Vol. 25, No. 5, Sept., 1981.
［9］E. R. Berlekamp,, "Algebraic Coding Theory", McGraw-Hill, 1968.
［10］F. F. Sellers, Jr., M. Y. Hsiao, L. W. Bearnson, "Error Detecting Logic for Digital Computer", McGraw-Hill, 1968.
［11］H. H. Roth, "Linear Binary Shift Register Circuits Unilizing a Minimum Number of Mod-2 Adders", IEEE Trans. Inf. Theory, April, 1965.

執筆者紹介

上　谷　彊　輔　(Kyousuke Kamiya)

昭和18年生．41年早稲田大学理工学部電気通信科卒業，45年東京大学理学系大学院数学専攻修士．47年日本ユニバック総合研究所入社．52年日本ユニバック(株)入社．技術計算の開発応用を経て，現在電子装置設計のCADの開発応用，およびハードウェアのテスト・ソフトウェア開発を担当．商品開発本部ハードウェア開発3部CAE開発室に所属．

論文 イメージ・データの圧縮手法

Image Data Compression Technique

河 合 昭 男

要 約 本稿は，UNIVAC DS 7 U-IMAGE のために開発されたイメージ・データの圧縮法について述べるものである．圧縮法は，マイクロ・メインフレーム垂直統合の見地から，シリーズ 1100 で採用されているモディファイド・ハフマン符号化（MH 符号化）が基本となっている．これを DS 7 上に実現するに当たって，次の 2 点について考慮した．第 1 は DS 7 で使用するイメージ・スキャナの画像処理方式と MH 符号化の整合性の問題，第 2 はファックスのノーマル/ファイン・モードへの対応である．

本稿では，まず MH 符号化の原理を述べ，次に第 1 の問題点であるディザ法（イメージ・スキャナ本体でハーフトーン処理に用いられている）と MH 符号化の矛盾について論じ，これを解決するために考えたインタリーブ圧縮について述べる．次に，第 2 の問題点への対応として考えた 1/2 圧縮について述べる．これは，MH 圧縮やインタリーブ圧縮とは独立な圧縮法で，これらと併用することもでき，簡単なロジックで圧縮効果も大きい．続いて，ファックスとの互換性の問題とその対策を述べる．最後に，圧縮率の実測データを元に，各圧縮法の評価を行う．

Abstract Image data compression technique implemented for UNIVAC DS 7 U-IMAGE is described.

The compression technique for DS 7 U-IMAGE is based on Modified Huffman (MH) code in consideration of 1100-DS7 micro-mainframe link, since it has been already supported on 1100 series. There are 2 problems to implement this on DS7. The first is the incompatibility between DS7 image scanner and MH coding. The second is to correspond with FAX fine/normal mode.

The author tackled these problems with following approaches:

1) Interleave compression solved this problem, though dither matrix for halftone process in the image scanner is inconsistent with MH code.
2) The 1/2 compression is to correspond with FAX normal mode. It is independent with MH code or interleave compression, and can be used with one of them.

This paper begins with MH coding theory, and explains the interleave compression and 1/2 compression. How to interface U-IMAGE with G3 FAX through Mapper 1100 is then described. Finally the measured compression ratio will be shown.

1. はじめに

UNIVAC DS7 の 2714-00 型イメージ・スキャナで A 4 サイズの原稿を 8 dpm（ドット/mm）の精度で読み取ると，そのデータ量は 500 Kb にもなる．このデータを DS7 のフロッピ・ディスクに収納すると，1 枚に 2 画面分しか入らず，不経済で使いにくい．またホストとのデータ転送も，伝送手順を考えないで単純に計算すると，500 Kb÷9,600 bps＝416 秒にもなり実用的でない．

イメージ・データついては，G 3 ファックスで採用されている MH 符号化[4] による

データ圧縮が歴史的に先行していて実績もある．また，シリーズ1100でもG3ファックス対応のMH圧縮がサポートされていることもあり，マイクロ・メインフレーム垂直統合の見地からDS 7でもこれを採用することにした．

2. データ圧縮法

2.1 冗長圧縮とエントロピ圧縮

データ圧縮法には，**冗長圧縮**（redundancy reduction）と**エントロピ圧縮**（entropy reduction）の二つの考え方がある[1]．冗長圧縮とは，データに含まれる冗長性を取り除くことによりデータ圧縮を行うもので，完全に復元できる**ロスのない圧縮法**である．エントロピ圧縮とは，データに含まれるあまり必要でない情報を削除することによりデータ圧縮を行うもので，完全には復元できない**ロスのある圧縮法**である．文書や一般的データは，圧縮によりデータが歪むことは許されないので，冗長圧縮が必要である．一方，イメージや音声は，人間が認識できれば少々データが歪められても構わないので，エントロピ圧縮でも良い．

本稿で述べるMH圧縮とインタリーブ圧縮は冗長圧縮で，1/2圧縮はエントロピ圧縮である．

2.2 圧縮の限界

冗長圧縮による圧縮の限界は式で与えることが可能である．Shannonの情報源符号化定理によると，情報源 $S=\{s_1,s_2,\cdots,s_m\}$ に対して各 s_i の生起確率を $p(s_i)$ とするとき，S の出力系列を r 元系列に符号化を行ったとき，その平均符号長の下限 $H(S)$ は次式で与えられる[3]．

$$H(S)=-\sum_{i=1}^{m} p(s_i)\log_r p(s_i)$$

エントロピ圧縮による圧縮の限界は，許容されるデータの歪率とのトレードオフである．

圧縮率と圧縮の効果尺度を次のように定義する[2]．

$$\text{圧縮率}=\frac{\text{原データ列の長さ}}{\text{圧縮データ列の長さ}}$$

$$\text{効果尺度}=\frac{\text{圧縮データ列の長さ}}{\text{原データ列の長さ}}$$

2.3 MH符号化

冗長圧縮による2値画像データの圧縮法の基本的な考え方はランレングス符号化[4]である．これは，走査線方向に白または黒のデータが連続することが多いという性質を利用して，その数（ランレングス）を数える方法である．

MH符号化は，これを一歩進めた考え方で，ランレングス符号化では単に数を数えるだけであるが，MH符号化はランレングスの発生頻度を統計的に調べて，頻度の高いものには短いコードを，低いものには長いコードを割り当てる方法である．このため，CCITTで代表的な11種のドキュメント群が選定され，その統計データを元にG3ファックス用として勧告された．

MH符号化は，1走査線1,728ドットを64ドット未満のランレングスを表現するターミネイト符号と，64の倍数を表現するメイクアップ符号を合成して行う．出現頻度

の高い63ドット以下はターミネイト符号のみで，出現頻度の低い64ドット以上は，メイクアップ符号＋ターミネイト符号 で表現する．

《MH 符号化による圧縮率の計算例》

DS7上のA4サイズ(1,728ドット×2,376ドット)すべて0(白)のデータをMH符号化すると，その圧縮率は次のように計算できる．

まず，各行1,728ドットのイメージ・データは，次の計算で29ビットのコードに圧縮される．

- 1,728ドットの白　ラン・メイクアップ符号：　010011011＝ 9ビット
- 0ドットの白　ラン・ターミネイト符号：　00110101＝ 8ビット
- その他……EOL　：000000000001＝12ビット

合計＝29ビット

これに行数を掛けると，

29ビット×2,376行＝68,904ビット

最後にRTC(EOL×6) を加えると，

12×6＝72ビット

(68,904＋72)/8＝68,976/8

＝ 8,622バイト

となる．一方，元のデータは，

1,728×2,376/8＝4,105,728/8

＝ 513,216バイト

したがって，圧縮率および効果尺度は次のようになる．

$$\text{圧縮率} = \frac{513,216}{8,622} = 59.5\text{倍}$$

$$\text{効果尺度} = \frac{8,622}{513,216} = 1.68\,\%$$

3. イメージ・スキャナとディザ法[6]

3.1 イメージ・スキャナの特性

イメージ・データの圧縮を考えるとき，DS 7では，イメージ・スキャナの特性を抜きには考えられない．イメージ読み取り部の仕様は，次のようになっている．

- 原稿サイズ　：最大A4
- 読み取り線密度　：主副走査方向とも8 dpm
- 読み取り操作幅　：主走査方向　216 mm
  副走査方向　297 mm
- 出力線密度　：8，6，4 dpm
- 読み取り濃度指定　：画像データの2値化スレッシュを3段階に選択できる．
- 階調出力　：白黒2階調モードとディザ法による16階調モードを選択できる．
- ディザ・マトリックス：4×4　ランダム・パターン
  4×4　スパイラル・パターン

3.2 ディザ法の原理

ここで，MH 圧縮を行う上でとくに問題になるのはディザ法である．これは，濃淡が表示できる特殊なイメージ出力機構を用いずに，見かけ上の濃淡を表示するために考えられた方法である．DS 7 イメージ・スキャナは，16 階調で読み取った入力画像を 4×4 のディザ・マトリックスをしきい値として 2 値化出力を行う．これを式で表現すると，次のようになる．

$D=(d_{ij})$：ディザ・マトリックス

$0 \leqq d_{ij} < 16$, d_{ij} はすべて異なる

$A=(a_{ij})$：入力画像，16 階調

$0 \leqq a_{ij} < 16$, 数字の大きいほど明るい

$B=(b_{ij})$：2 値化された出力画像

$b_{ij}=0$：表示されない

1：表示される

ただし，D, A および B は，4 行 4 列の行列とするとき，入力画像はディザ法により次のように変換される．

$b_{ij}=0 \quad if \quad a_{ij} < d_{ij}$

$1 \quad if \quad a_{ij} \geqq d_{ij}$

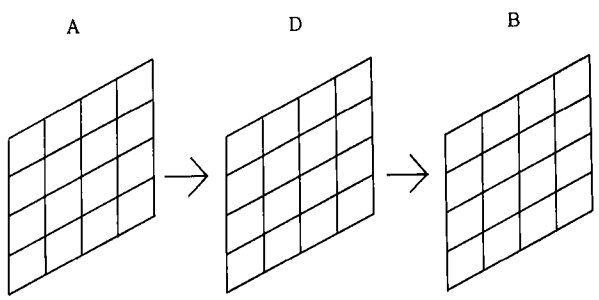


図 1　ディザ法の原理

Fig. 1　Dither matrix

イメージ・スキャナから DS 7 のパンフレットを読み取った例を図 2 に示す．左は 2 値，右はハーフトーンで読み取ったものである．ハーフトーンは，16 階調で読み取ったデータに対してディザ処理を行っている．左と比べて右の方が濃淡がきれいに表現できていることがわかる．

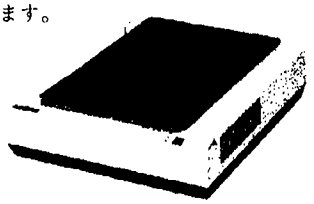
4. インタリーブ圧縮

イメージ・スキャナより，写真をハーフトーンで読み取ったイメージ・データを MH 圧縮すると，そのデータ量は，圧縮しない時に比べて 2 倍くらいに増加してしまう．この原因は，ディザ法によるものである．

MH 圧縮は，走査線方向に白または黒のデータが連続するほど，圧縮率はよくなる．

イメージ・スキャナ

地図、写真等のイメージ情報を読み取り、ワープロ等の文書の中に挿入することができます。原稿はA4サイズまで、原稿の種類により読み取りの解像度を4、6、8本／mmの3段階に切り替えて、鮮明なイメージを画面／印書装置に再現できます。



イメージ・スキャナ

地図、写真等のイメージ情報を読み取り、ワープロ等の文書の中に挿入することができます。原稿はA4サイズまで、原稿の種類により読み取りの解像度を4、6、8本／mmの3段階に切り替えて、鮮明なイメージを画面／印書装置に再現できます。

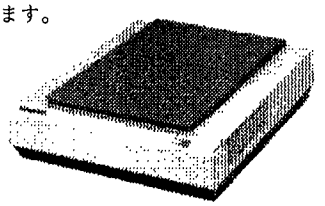


図 2　2値とハーフトーンの比較

Fig. 2　Binary and halftone

DS 7イメージ・スキャナは，4×4のディザ・マトリックスを用いているため，4ドット単位のパターンが連続しやすい．そのため，とくに中間色をハーフトーンで読み取ったデータは，白または黒が連続しない．そこで，4ドット置きにイメージ・データを読み取ってMH圧縮を行うとよいことが予測される．本稿では，この方法をインタリーブ圧縮と呼ぶ．実際，写真をハーフトーンで読み取ったイメージ・データに対してインタリーブ圧縮を用いると，1/2に圧縮できた．

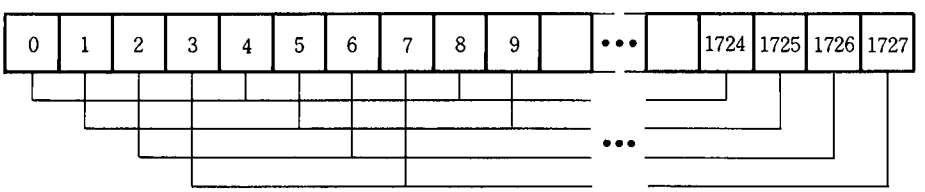


図 3　インタリーブ圧縮の原理

Fig. 3　Interleave compression

5. 1/2　圧　縮

1/2圧縮の基本的アイデアは，グラフ画面をファイルに保存するとき，またはホストに転送するとき，走査線を1本置きに間引きするという考えである．したがって，復元データは原データと異なるエントロピ圧縮である．

U-IMAGEでは，圧縮法として**高精度モード**と**標準精度モード**の二つのモードを設定した．標準精度モードとは，2行分のデータのORを掛けてデータ量を1/2に圧縮して取り扱うモードで，これを1/2圧縮と呼ぶことにする．高精度モードとは，1/2圧縮

を行わないモードである．

間引きしないでORを掛けた理由は，横線が消えてしまわないようにするためである．間引き方式にすると，イメージ・スキャナから読み取ったイメージ・データに対してはインパクトはそれほど大きくもないが，イメージ編集機能を用いてマウスを使って作図するようなケースでは，横線が消えてしまうことがある．

1/2圧縮されたデータを再表示するためには，2行ずつ同一のイメージを表示してゆけばよい．1行置きに表示する方法も考えられれが，テストの結果この方法は，表示された画像がちらついて見にくいものとなり，使えないことがわかった．その理由は，DS 7のディスプレイは精度が高く，画面の操作をインタレースによって行っているためである．これは，1画面の走査を，まず上から1行置きに下まで行い，次に2行目から1行置きに行う方法である．したがって，イメージの表示を1行置きにするなら，インタレースによる前半の走査で実際の表示が行われるのみで，後半の走査では何も表示されないが，この影響を受けて画像がちらつくことになる．この現象はとくに明るい画像，たとえば写真をハーフトーンで読み取って白黒反転表示した画像等に顕著に現れる．奇数行と偶数行の明るさの落差の大きいのが，ちらつきの原因である．

この方法は，簡単なロジックで圧縮効果も大きく，とくに細かい文字や図形でない限り，画像の品位もあまり落ちない．手書き文書や写真等イメージ・スキャナから読み取ったイメージに対しては，ほとんど問題なく使える．

図4の左側は，イメージ・スキャナからDS 7のパンフレットをハーフトーンで読み取った例（図2右側と同一）で，右側は，左のイメージを1/2圧縮（標準精度）でファイル保存したものを再表示したイメージである．文字がやや判読しにくくなり，イメージとして全体的に黒っぽくなるが，十分使える．

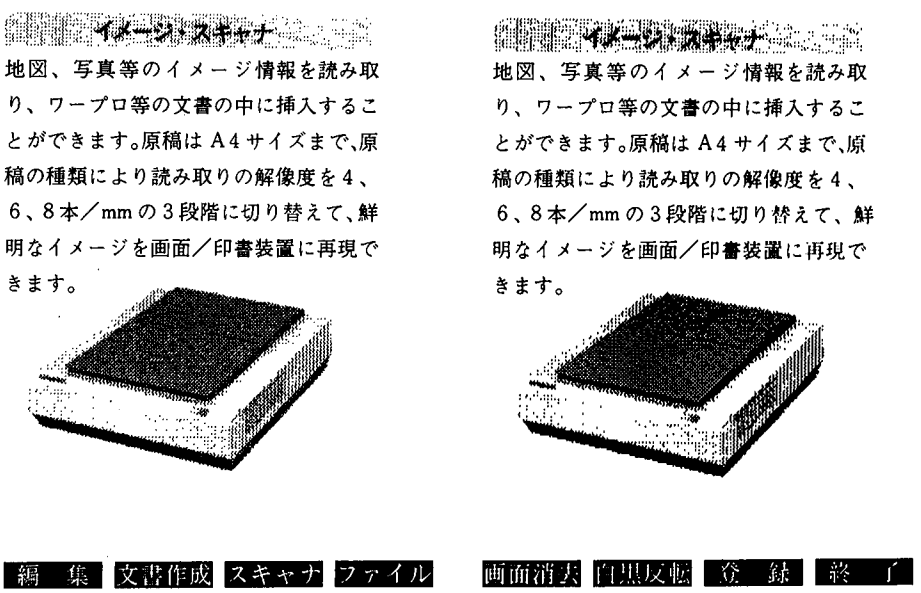


図 4　標準精度と高精度

Fig. 4　Normal mode and fine mode

6. ファックスとの互換性

DS 7 のイメージ・データは，MAPPER 1100 の DKT コマンドによるファイル転送または，IMI/IMO コマンドにより MAPPER 1100 のレポートとして取り扱うことができる．このレポートは FSP 1100 を経由してファックスと通信できる．

ファックスのノーマル・モードとファイン・モードは，DS 7 U-IMAGE で標準精度モードと高精度モードでそれぞれ MH 圧縮を掛けることにより対処できる．

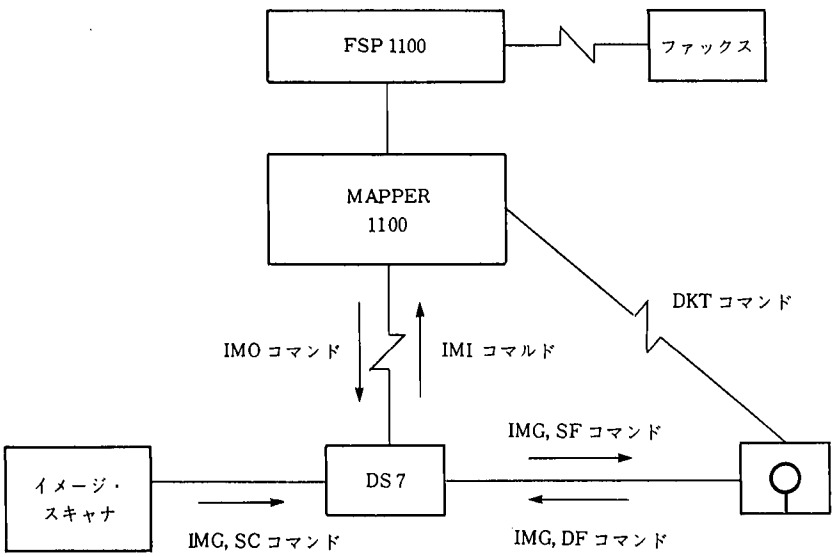


図 5　DS7—MAPPER—ファックスの接続

Fig. 5　DS 7—MAPPER—FAX connection

ファックスから送られてきたイメージを DS 7 で表示するのに，U-IMAGE でとくに考慮すべき問題はない．しかし，逆方向には次の 2 点の考慮が必要である．

1)　1 行 1,728 ドットに満たないデータは，ファックスでエラーとなる……たとえば，画面サイズのイメージをそのまま MH 圧縮してファックスに出力しようとするとファックスでエラーとなる．そのため，U-IMAGE ではこのような場合 1,728 ドットに足りない分 0 を付加した後，MH 圧縮を行っている．

2)　スキャナから読み取ったイメージをファックスに出力すると，A 4 サイズで約 1 cm 長くなる……ファックスの読み取り精度は，主走査方向は 8 dpm，副走査方向はノーマル・モードで 3.85 dpm，ファイン・モードだと 2 倍の 7.7 dpm である．一方，イメージ・スキャナの読み取り精度は，最大で主副とも 8 dpm である．したがって，A 4 サイズの原稿は DS 7 では 297 mm×8 dpm＝2,376 行，ファックスでは 297 mm×7.7 dpm＝2,286 行となり，ファックスの方が 90 行＝約 12 mm 長くなる．この問題については，U-IMAGE レベル 1 R 5 E 以降で，縦方向に拡大縮小を行うことにより同一の大きさになるように考慮されている．

次に，U-IMAGE で解決できない問題として挙げられるのは，インタリーブ圧縮を行ったイメージ・データは，ファックスでは復元できないこと．この問題を解決する

ためには，UNIVAC 1100 側で，いったん MH の復元を行った後インタリーブを元に戻し，もう1度 MH 圧縮を行ってファックスに送信する手続きが必要になる．現在のところこの機能はサポートされていない．

7. 評　　価

この章では，圧縮率の計測を行い，本稿で述べた各圧縮法の評価を行う．

基礎データとして，すべて0のデータ（画面クリアした状態）を MH 圧縮すると，A4サイズ 513 Kb のデータは 8.6 Kb(1.7％)に圧縮される．この数字は実測値であるが，2章で示した計算値と同一の値である．インタリーブ圧縮でも同一である．1/2 圧縮（標準精度モード）と併用すると 4.3 Kb(0.8％）となる．一方，すべて1のデータ（白黒反転した状態）を MH 圧縮すると 12.8 Kb（2.5％），1/2 圧縮と併用すると 6.4 Kb（1.2％）となり，すべて0のデータよりやや大きくなる．これは，MH 圧縮の特徴で，0の方が統計上出現頻度が高いので，より短いコードを割り付けているからである．

次に，手書き原稿の例として活字にする前の本原稿をイメージ・スキャナより2値で読み取ったイメージを MH 圧縮する．高精度モードで 44.7 Kb(8.7％)，標準精度モードで 24.1 Kb(4.7％)となり，MH 圧縮が効果的に働いていることが確認できる．

次に，活字と写真の合成の例として，「ユニバック・ニュース」をイメージ・スキャナより2値およびハーフトーンで読み取ったイメージの圧縮を行う．2値のデータを MH 圧縮すると，高精度モードで 218.1 Kb（42.5％），標準精度モードで 114.2 Kb (22.3％)となり，手書き原稿に比べて圧縮率はかなり悪い．これは，新聞の方が小さく，字数も多いため，その分空白部分が少ないためである．ハーフトーンのデータをインタリーブ圧縮すると，高精度モードで 272.8 Kb（53.2％），標準精度モードで 192.2 Kb（37.4％）であるが，これを通常の MH 圧縮を行うとそれぞれ 475.2 Kb (92.6％)，263.1 Kb (51.3％) となり，インタリーブ圧縮が効果的に働いていることが確認できる．

表 1　圧縮率の計測

Table 1　Compression ratio measurement

原稿種類	読取方法	MH圧縮		インタリーブ圧縮	
		高精度	標準精度	高精度	標準精度
すべて0	—	8,622 (1.7)	4,316 (0.8)	8,622 (1,7)	4,316 (0.8)
すべて1	—	12,780 (2.5)	6,395 (1.2)	12,780 (2.5)	6,395 (1.2)
手書き原稿	2値	44,651 (8.7)	24,052 (4.7)		
活字／写真	2値	218,072 (42.5)	114,216 (22.3)	355,000 (69.2)	196,864 (38.4)
	ハーフトーン	475,164 (92.6)	263,112 (51.3)	272,789 (53.2)	192,156 (37.4)

原稿サイズ：A4
圧縮前のイメージ・サイズ：513,216バイト
手書き原稿：本原稿

バイト
(％)

8. おわりに

一般的に，イメージ・データの圧縮法として，MH 圧縮が必ずしも最良とはいえない．圧縮率について言えば，MH 圧縮は，手書きの文字や図のように白い部分が多いデータに対して，統計的に最大の効果を発揮するように決められたものである．したがって，本稿でも述べたように，イメージ・スキャナから写真をハーフトーンで読み取ってきたようなデータに対しては効果がない．また，CPU 負荷の観点からも，MH 符号化および復元化を行うプログラムは非常に多くのビット操作が必要で，好ましい方法とはいえない．

冒頭にも述べたが，DS 7 のイメージ・データ圧縮法は，シリーズ 1100 とのデータ互換性から，すでに使われている MH 圧縮を基本的に採用した．それに対して本稿で述べたような改良を加えてきた結果，DS 7 U-IMAGE のデータ圧縮は，圧縮率およびマイクロ・メインフレーム垂直統合の観点から，共に十分目標を達成できたものと信ずる．

参考文献
［1］ 渡辺,「効率的なデータ格納法」, INFORMATION, 1986, No. 2, pp. 51～63.
［2］ G. Held, 渡辺訳,「データ圧縮技法入門」,啓学出版, 1985.
［3］ 宮川, 原島, 今井,「情報と符号の理論」,岩波講座, 情報科学 4, 1982.
［4］ 長峰, 杉山, 山田,「最近のデータ圧縮技術」, 情報処理, 1980, Vol. 21, No. 7, pp. 777～785.
［5］ 樋口,「OA に必要とされる画像処理技術」, PIXEL, 1985, No. 37, pp. 63-69.
［6］ 松井, 大屋, 篠原,「パソコン用イメージ・スキャナ」, 日経バイト, 1985, No. 10, pp. 145～162.

執筆者紹介 河 合 昭 男 (Akio Kawai)
昭和 22 年生．45 年大阪大学理学部数学科卒業，日本ユニバック(株)46 年入社，EXEC 保守，シリーズ 1100 性能評価，無人化システム開発を経て，現在ワークステーション・システム部に所属し，DS 7 のイメージ処理を担当する．情報処理学会会員．

論文

衛星通信回線使用時のDCAリンク・レイヤおよびネットワーク・レイヤの伝送効率分析

Performance Analysis of the DCA Data Link Layer and Network Layer Protocol on the Satellite System

宮坂順之

要　約　本稿では，衛星通信回線をDCP/TELCONトランク回線として使用した時のDCA*のデータ・リンク・レイヤUDLC（Universal Data Link Control）およびネットワーク・レイヤDUC（Data Unit Control）の伝送方式をモデル化し，理論式の導出を行った．

ここで得られた理論式の値と実測値とを比較・分析した結果，理論値はよく実測値を近似していることが確認できた．

これにより，DCP/TELCONシステムを衛星通信回線に適用した場合のデータ・リンク・レイヤとネットワーク・レイヤの伝送効率を予測することができ，システムが要求されるパフォーマンスを得るため必要な最適のパラメタの設定が可能となった．また，本分析を通じてDCAデータ・リンク・レイヤ，およびネットワーク・レイヤの問題点・改善点についても明らかにすることができた．

Abstract　In this paper, the author provides a model and analysis of the UNISYS DCA data link layer UDLC (Universal Data Link Control) and network layer DUC (Data Unit Control) protocol. And it introduces the approximate theoretical equations to estimate the throughput for the data link layer and network layer on the satellite system.

The values calculated from these theoretical equations, coincide with the experimental results well. The equations obtained from this analysis, are found to be a useful tool for the DCA data link and the network layer performance estimation.

1. はじめに

本分析レポートは，昭和60年度より郵政省が推進している「衛星通信実験パイロット計画」に参画し，得られた実験結果をベースにまとめたものである．

1.1 目　　的

通信衛星は，地上36,000 kmの赤道上の静止軌道に打上げられているため，往復で500 msの遅延が生ずる．また，降雨・衛星蝕・太陽雑音等の影響で，回線品質が劣化したりあるいは通信不能の状態が発生したりする．

本レポートでは，回線の誤り率，メッセージ長，送信ウインドウ・サイズ，応答返送の条件等によってDCAのデータ・リンク（レイヤ2）およびネットワーク・レイヤ（レイヤ3）の伝送効率がどのように変化するかを分析し，衛星回線を地上回線と同等の条件で使用するための要件を明らかにする．

---

* DCA：Distributed Communication Architecture. Unisysの階層構成のネットワーク・アーキテクチャである．

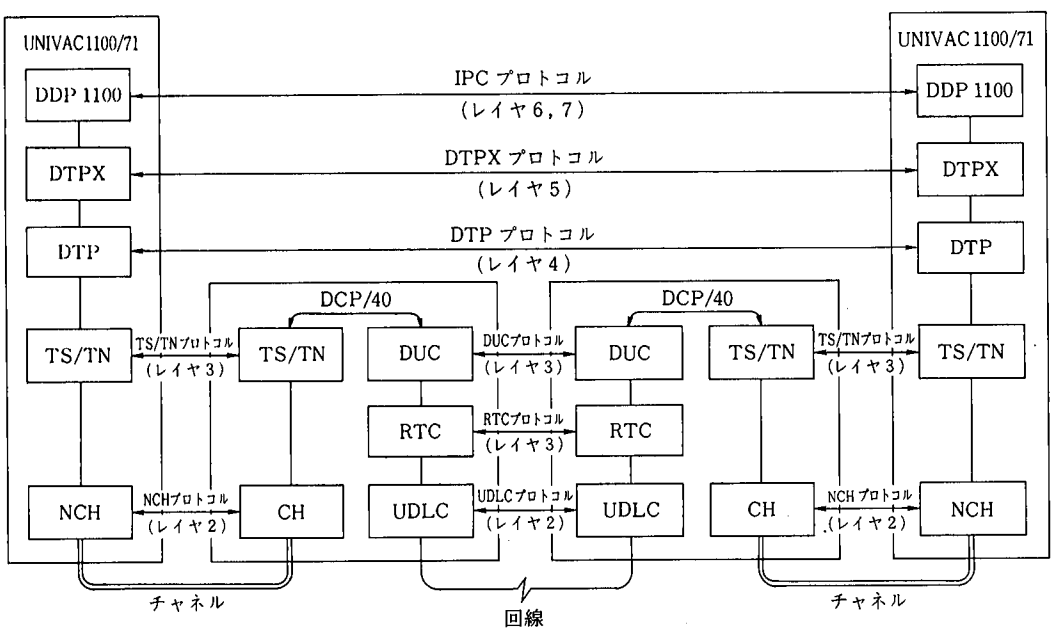
1.2 DCA のレイヤ構成

DCA は，ISO/OSI の 7 層モデルに近い階層構成をしている．図 1 にそのレイヤ構成を示す．

下図のように DCA のデータ・リンク・レイヤ・プロトコルとして，UDLC が使用されている．これは，ISO の HDLC ABM（非同期並行モード）と同等のプロトコルである．

DCA のネットワーク・レイヤ・プロトコルとして，図 1 に示すように，RTC と DUC が使用されている．このうちの RTC は経路制御のためのプロトコルであり，伝送遅延に関係しないので本分析から省いた．一方，DUC は送信側 DCP と受信側 DCP 相互間で送達確認，フロー制御およびロジカル・チャネルの多重化機能を持ったプロトコルで，伝送遅延の影響を受ける．本レポートでは，UDLC プロトコルと DUC プロトコルについての伝送効率の分析を行う．

なお，TS/TN，DTP，DTPX および IPC 等のプロトコルについての分析は今回は行わない．また，これらのプロトコルのパラメタは DUC，または UDLC の伝送効率に影響を与えないように設定した．



IPC : Inter Process Control
(ファイル転送プロトコル)

DDP : Distributed Data Processing
(ファイル転送プログラム)

DTPX : DCA Transport Protocol Extended
(DCA セション・プロトコル)

DTP : DCA Transport Protocol
(トランスポート・プロトコル)

TS/TN : Termination System/Transport Network
(X. 25 相当プロトコル)

NCH : New Channel Handler
(チャンネル・プロトコル)

CH : Channel Handler
(DCP 側チャネル制御プロトコル)

RTC : Routing Control
(経路制御プロトコル)

図 1 DCA プロトコル階層構成

Fig. 1 DCA protocol layers

2. データ・リンク・レイヤ（UDLC）の伝送効率分析

UDLCとしては，伝送遅延の大きな衛星回線に適用するため非同期拡張並行モード(ABME)*を使用した．また，応答方式は逐次応答方式（スライディング・ウインドウ方式：受信フレームに対し個々に応答を返送する）を使用し，再送方式としてREJ (Reject) を使用した．

フレーム長およびウインドウ・サイズは，システム生成時のパラメタとして任意の値が指定可能である．本実験では，フレーム長およびウインドウ・サイズを変化させ伝送効率への影響を測定した．

2.1 エラー・フリー時の伝送効率

ここではエラー・フリーの場合の伝送効率について分析する．

2.1.1 実験結果

衛星通信の実験結果について，地上回線および衛星回線を比較して図2に示す．

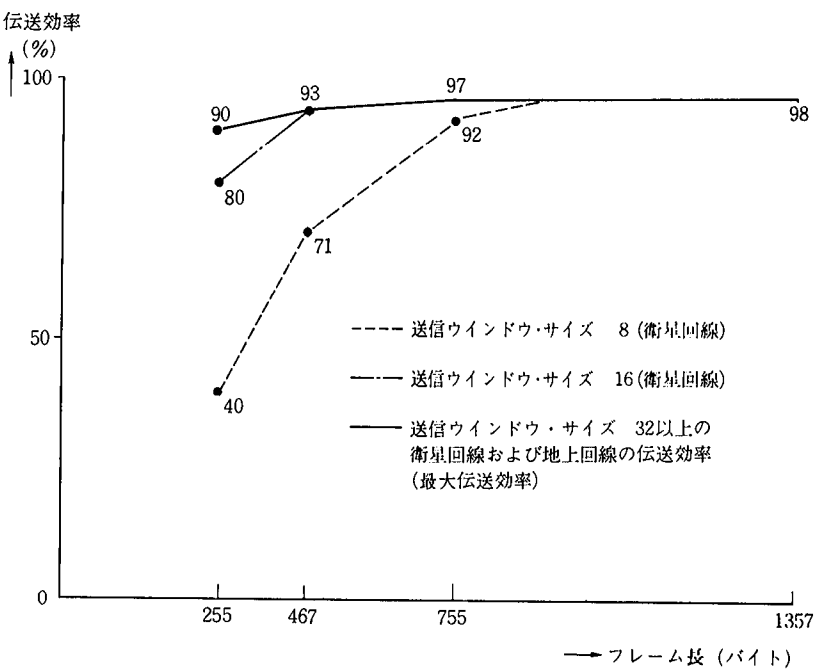


図2　データ・リンク・レイヤの伝送効率

Fig. 2　Utilization of data link layer

図2の実線は，地上回線または衛星回線でウインドウ・サイズが十分大きな場合の伝送効率であり，DCP/UDLCプロトコルを制御しているソフトウェアおよびハードウェアの最大伝送能力を示している．

破線は衛星回線で，送信ウインドウ・サイズが8の場合について示してある．フレーム長が255バイトのとき，伝送効率が40パーセントと低く十分な伝送効率が得られない．これは，衛星回線に伝送遅延時間があり（片道で250 ms），先送りのウインドウ・

* ABME：Asynchronous Ballanced Mode Extended. 主局と従局の機能を併せ持った複合局が，全二重伝送を効率的に行う方式である．

サイズ内に応答が戻らないためである．ウインドウ・サイズが16の場合も同様である．

2.1.2 理論式の導出と分析

送信フレームに対し，応答を受信するまでの時間（応答遅延時間）を T_r ms，送信ウインドウ・サイズを W_s，フレーム長を l バイトとすると，伝送容量 V バイト/秒は，次式で表される．

$$V=\frac{l\cdot W_s}{T_r} \tag{2-1}$$

ただし，V は回線速度を越えないものとする．

この式の V を，回線速度 S バイト/秒と等しいとすると，最大の伝送効率を得るために必要な W_s が得られる．

$$W_s=\frac{S\cdot T_r}{l} \tag{2-2}$$

W_s が本式の値より十分大きな場合，伝送効率は DCP/TELCON の送信フレームの処理時間に依存した値となる．送信フレームの処理時間を α 秒とすると V は次式で表される．

$$V=\frac{\text{フレーム長}}{\text{フレーム送信時間}+\text{フレーム処理時間}}=\frac{l}{\frac{l}{S}+\alpha} \tag{2-3}$$

式(2-1)に $V=6{,}000$ (48 K bps の回線速度)，l にフレーム長として 255，467，755 および 1,357 をそれぞれ代入し，図2の実験結果より T_r を求めてグラフに表すと，図

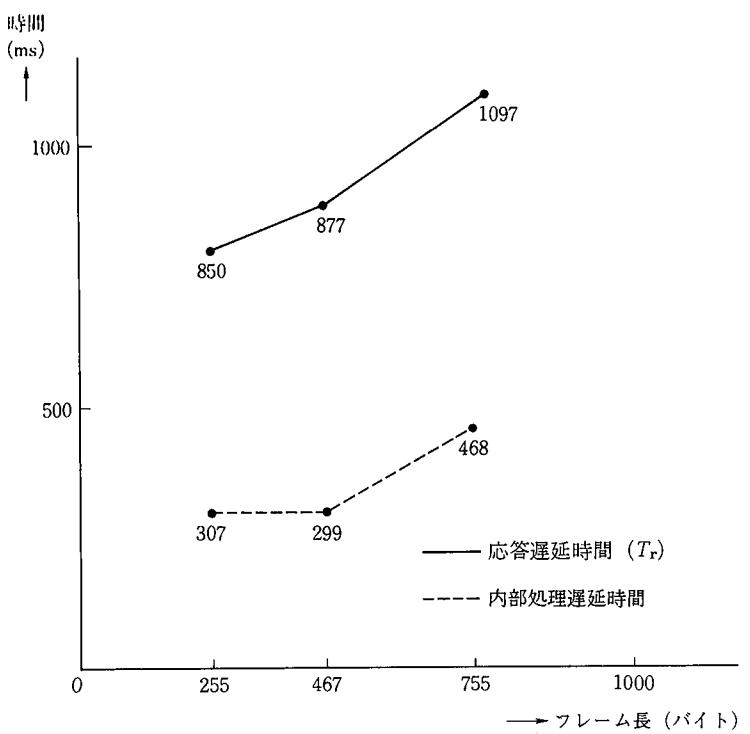


図3 フレーム長と応答遅延時間/内部処理遅延時間

Fig. 3 Frame size and response delay/internal processing delay

3の実線のようになる．T_r より衛星回線の伝送遅延時間(往復)500 msおよびフレームの送出時間を差し引いた値を図の破線で示す．破線はDCP/TELCON内の内部処理遅延時間で300～500 msであり，フレーム長500バイトまではほぼ300 msである．

この値は一般的な処理時間（数10 ms）に比較して大きな値であるが，DCPのハードウェアが分散プロセッサの構成をとっているためで，処理能力が低いためではない．

ここでは，内部処理時間を300 msとみなし T_r を求めると，次式となる．

$$T_r=0.8+\frac{2l}{S} \tag{2-4}$$

ここで，右辺の l を2倍しているのは，両方向同時伝送を考慮したもので，片方向の場合は2倍する必要はない．

本式を式(2-2)に代入すると，DCP/TELCON UDLCを衛星通信回線に適用したとき，最大伝送効率を得るために必要な W_s と S の関係が得られる．

$$W_s=\frac{0.8\,S}{l}+2 \tag{2-5}$$

この式から48 K bpsの伝送速度（S=6,000バイト/秒）についてのフレーム長と送信ウインドウ・サイズを求め図4に示す．

次に，α を求める．式(2-3)に図2の実験結果の実線の値を代入すると α が求められる．α の値は，**フレーム長に依存せず約5 msである．**

この時間は，回線速度が128 K bps程度までは問題とならないが，伝送速度256 K bpsでフレーム長が100バイト，または伝送速度512 K bpsでフレーム長255バイトの時は，最大伝送効率が50パーセント程度となる．したがって，現行V.35 UDLC

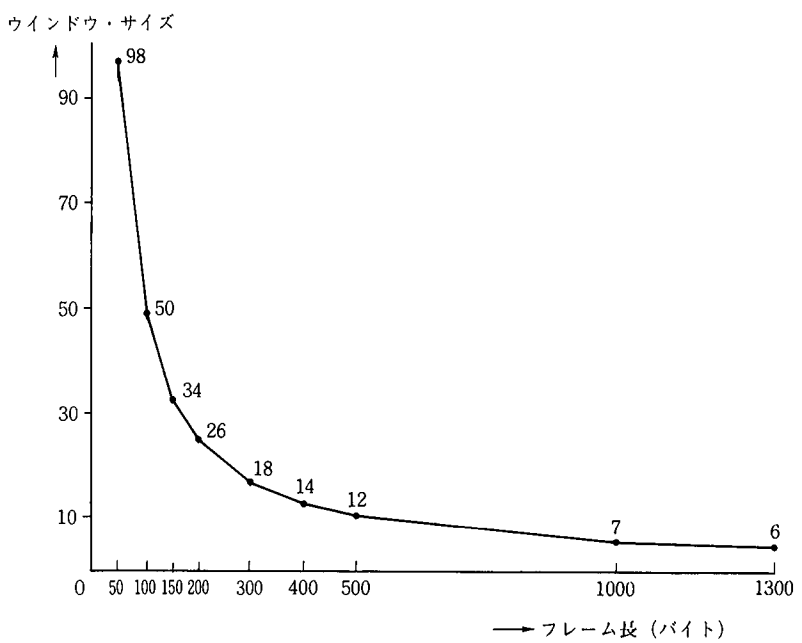


図4　最大伝送効率を得るための l と W_s の関連図（48 K bps回線）

Fig. 4　Relationship between l and W_s for maximum utilization (48 K bps line)

インタフェースでは，最大 128～250 K bps 程度の伝送速度まで適用可能であるが，それ以上では伝送効率が低下し適用困難である．

2.2 回線品質劣化時の伝送効率

2.1 節では，誤りのない場合についての伝送効率について分析した．しかし現実の回線使用時，とくに衛星回線を使用する場合，回線の品質劣化は避けられない．したがって，ウインドウ・サイズ，再送方式等は品質劣下に対応できるよう配慮しなくてはならない．以下に品質劣下時の分析について述べる．

2.2.1 実 験 結 果

図 5 にビット誤り率の変化によるフレーム長とそれに対応した有効回線利用率の変化について示す．ここで，有効回線利用率は次式で定義される．

$$\text{有効回線利用率}=\frac{(\text{伝送された総パケット数}-\text{再送パケット数})\times l}{\text{回線速度}\times T}$$

回線速度：回線の伝送速度（バイト/秒）

伝送された総パケット数：T 時間内に送られたパケットの総数

再送パケット数：上記総パケット数のうちの再送パケットの数

l：パケットのバイト数（HDLC のアドレス，コントロール・フィールドおよび FCS((Frame Check Sequence)を含む）

T：伝送時間

図 5 により，ビット誤り率が 10^{-5} 程度までは実用上十分使用に耐えられるが，10^{-5} より劣化すると急激に有効回線利用率が悪化し，使用できない状態となる．

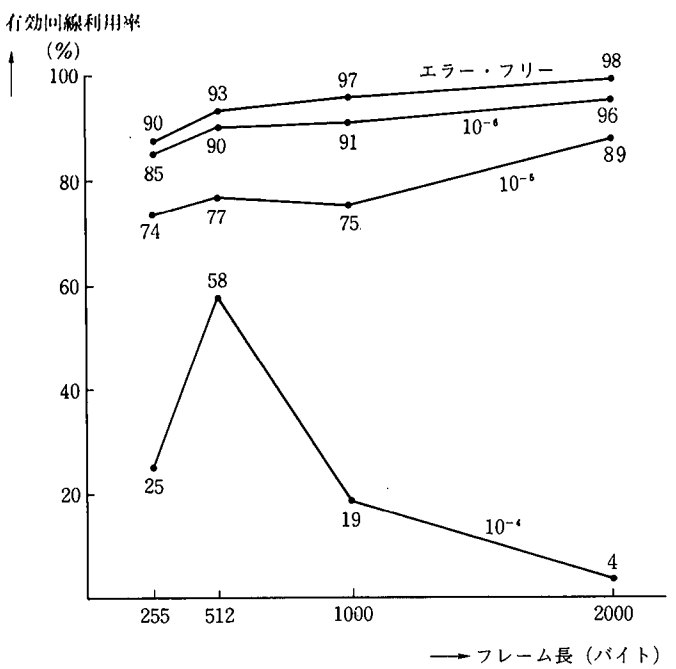


図 5 ビット誤り率と有効回線利用率

Fig. 5 Bit error rate and effective line utilization

2.2.2 有効回線利用率の理論式の導出

ここでは，ビット誤り率と有効回線利用率の関係についてREJ再送方式，MN-SREJ (Multi Number-Selective Reject) 再送方式，およびSREJ (Selective Reject) 再送方式の場合の理論式の導出を行う．なお，ここでSREJとMN-SREJの両方式は，次の通りである．

SREJの方式ではフレームに誤りを検出したとき，同時に1個のフレームの再送しかできない．すなわち，一度に複数のフレームが誤った場合には，最初の誤りフレームのリカバリ後でなければ，次のフレームの再送要求を行うことができない．MN-SREJの方式では，MN-SREJ応答フレームにより，再送フレームの番号と受信フレームの応答番号を相手局に知らせ，同時に複数個のフレームが誤った場合には，複数の再送要求(MN-SREJ)を送出できる．したがって，衛星回線のような伝送遅延の大きな媒体を使用する場合の誤り再送方式として有効である．しかし，この方式はまだISOで標準化されていない．

また，過去の実験結果によると，ビット誤りはバースト的に発生することが多く，ビット誤り率と文字誤り率はほぼ同一である．ここでは，公式化を簡単にするため，ビット誤り率と文字誤り率は同一とみなす．文字が誤る確率を E，フレームが誤る確率を E_f とすると，l 個の符号に誤りのない確率は $(1-E)^l$ で表される．したがって，フレームの誤る確率 E_f は次の式で表される．

$$E_f=1-(1-E)^l \tag{2-6}$$

以下では各再送方式の理論式の導出を行う．

1) REJ再送方式……REJ再送方式では，受信フレームにエラーを検出したとき，当該シーケンス番号以降のフレームを正常に受信されたか否かは無関係に再送要求する．受信局ではフレームにエラーが発生した場合，当該フレームは破棄し，以降の正常フレーム受信時のシーケンス・エラー検出によりREJ応答を返す．

同一フレームが m 回エラー再送される確率を $P(m)$ とすると，$P(m)$ は次式で表される．

$$P(m)=E_f^m(1-E_f) \tag{2-7}$$

また，フレームの応答遅延時間を T_r，フレームの送出時間を T_f，再送フレーム数を R とすると，R は次式で表される．

$$R=\sum_{m=1}^{\infty} m\cdot P(m)\cdot \underline{1回の再送で送信されるフレーム数}$$

$$=\sum_{m=1}^{\infty} m\cdot P(m)\left\{\frac{T_r}{T_f}+\sum_{k=0}^{W_s-1} k\cdot P(k)\right\} \tag{2-8}$$

ここで，{ }内の第2項は，受信局が誤りフレーム受信後，正常なフレームを受信しREJ応答を返送するまでに受信するフレーム数である．通常使用時の W_s は T_r/T_f (これはエラー・フリーの時の最大の伝送効率を得るため必要なウインドウ・サイズである．2.1節参照)より大きく設定するので，ここでは $W_s>T_r/T_f$ とする．$W_s\leq T_r/T_f$ のとき{ }内の値は W_s となる．

式(2-7)を(2-8)に代入して整理すると次式となる．

$$R=\frac{E_f}{1-E_f}\left\{\frac{T_r}{T_f}+\frac{1-E_f^{W_s}}{1-E_f}-(W_s-1)E_f^{W_s+1}\right\} \tag{2-9}$$

ここで，$E_f \ll 1$，$W_s > 1$ なので R は近似的に次式で表される．

$$R \approx \frac{E_f}{1-E_f}\left(\frac{T_r}{T_f}+1\right) \tag{2-10}$$

送信局が W_s 個のフレームを送信後，最旧フレームに対する応答が得られず，応答待ちのため送信が中断する時間 T_s は次式で表される．

$$T_s = \sum_{m=(W_s-T_r/T_f)}^{W_s}\left(m-W_s+\frac{T_r}{T_f}\right)T_f \cdot P(m) + T \cdot \frac{{E_f}^{W_s-1}}{1-E_f} \tag{2-11}$$

ここで，T は送信フレームに対する応答待ちのタイマ値である．本式の第1項は，送信局が W_s 個のフレーム送信内に応答が得られず送信が中断する時間で，これは受信局が $W_s - T_r/T_f$ 個以上のフレームを連続して誤って受信した場合に発生する．第2項は，受信局が W_s 個のフレームを連続して誤り，送信局のタイムアウトのため待つ時間である．

したがって，有効回線利用率 L は次式で表される．

$$L = \frac{T_f}{(R+1)T_f + T_s} \tag{2-12}$$

ここで，$T_s \ll 1$ なので無視すると，L は次式で近似される．

$$L \approx \frac{1}{1+\left(\frac{T_r}{T_f}+1\right)\frac{E_f}{1-E_f}} \tag{2-13}$$

2) SREJ 再送方式……n を正の整数とすると，W_s，T_r および T_f の間に次式が成立する．

$$W_s = n \cdot \frac{T_r}{T_f} \tag{2-14}$$

ここで，n の意味は，W_s 個のフレーム送信時に n 回以上の再送(同一フレームの再送を含む)が発生すると，送信が中断することを意味する．$n-1$ 回以下であれば，中断が発生せず連続してフレームの送信が可能である．また，T_r/T_f は，エラー・フリーの回線で最大の伝送効率を得るための送信ウインドウ・サイズで図4の値である．以降では議論を簡単にするため，W_s の値は n が自然数となるように設定する．

W_s 個のフレームのうち，再送も含めて m 個のエラーが発生する組み合わせの数 $A(m)$ は，m 個のボールを W_s の中に分配する方法の数と同じで，次式で表される[1]．

$$A(m) = \binom{m+W_s-1}{W_s-1} \tag{2-15}$$

ここで，m 個のボールのそれぞれは，フレームの誤り率 E_f に対応している．また，E_f は経歴に依存しない一定の値を持つので，W_s 個のフレームが m 個誤る数 $N(m)$ は，上記 $A(m)$ に ${E_f}^m$ と W_s 個のフレームが誤らないで届く確率 $(1-E_f)^{W_s}$ を乗じた値であり，次式で表される．

$$N(m) = A(m) \cdot {E_f}^m (1-E_f)^{W_s} \tag{2-16}$$

W_s 個のフレームが n 回以上再送される場合，フレーム送信の中断が発生する．

m 個の $(m>n)$ 誤りが発生し中断する時間は $(m-n+1)T_r$ であるので，中断時間 T_s は次式で表される．

$$T_s=\sum_{m=n}^{\infty}N(m)E_f(1-E_f)^{W_s}(m-n+1)T_r \tag{2-17}$$

また，W_s 個のフレームに $n-1$ 回以下の誤りが発生し再送される数 R は次式で表される．

$$R=\sum_{m=1}^{n-1}m\cdot N(m) \tag{2-18}$$

以上から回線の有効利用率 L を求めると，次式となる．

$$L=\frac{W_s\cdot T_f}{(W_s+R)T_f+T_s} \tag{2-19}$$

本式に(2-17)および(2-18)を代入し，式(2-14)の関係を用い整理すると次式が得られる．

$$L=\frac{1}{1+\sum_{m=1}^{n-1}m\cdot N(m)\cdot\frac{1}{W_s}E_f^m(1-E_f)^{W_s}+\sum_{m=n}^{\infty}N(m)E_f^m(1-E_f)^{W_s}(m-n+1)\frac{1}{n}} \tag{2-20}$$

3)　MN-SREJ 再送方式……W_s 内のフレームのうち，少なくとも一つのフレームが m 回再送され，他のフレームは m 回以下（m も含む）の再送でリカバリされる数 $N(m)$ は，次の式で表される．

$$N(m)=W_s\cdot E_f^m(1-E_f)(1-E_f^{m+1})^{W_s-1} \tag{2-21}$$

また，ここでもSREJで使用した式(2-14)を使用する．ただし，ここでの n の意味は，同一フレームが n 回以上エラーとなり再送される場合，送信が中断することを意味する．$n-1$ 回以下であればフレームの送信を継続できる．また T_r/T_f はSREJ方式の場合と同一意味である．

式(2-21)から，SREJと同様にフレームの送信が中断される時間 T_s を求めると次式となる．

$$\begin{aligned}T_s&=\sum_{m=n}^{\infty}(m+1-n)\cdot T_r\cdot N(m)\\&=\sum_{m=n}^{\infty}(m+1-n)\cdot T_r\cdot W_s\cdot E_f^m(1-E_f)(1-E_f^{m+1})^{W_s-1}\end{aligned} \tag{2-22}$$

ここで，$E_f\ll 1$ が成立することから $(1-E_f^{m+1})^{W_s-1}$ の値を2次の項まで近似すると，T_s は次式となる．

$$\begin{aligned}T_s\approx&\sum_{m=n}^{\infty}(m+1-n)\cdot T_r\cdot W_s\cdot E_f^m(1-E_f)\left\{1-(W_s-1)E_f^{m+1}\right.\\&\left.+\frac{(W_s-1)(W_s-2)}{2}\cdot E_f^{2m+2}\right\}\\=&T_r\cdot W_s\cdot\frac{E_f^n}{1-E_f}-T_r\cdot W_s(W_s-1)\cdot\frac{E_f^{2n+1}\cdot(1-E_f)}{(1-E_f^2)^2}\\&+T_r\frac{W_s(W_s-1)(W_s-2)}{2}\cdot\frac{E_f^{3n+2}\cdot(1-E_f)}{(1-E_f^3)^2}\end{aligned} \tag{2-23}$$

また，W_s 個のフレームが $n-1$ 回以内，再送される数 R は次式で表される．

$$R=\sum_{m=1}^{n-1}W_s\cdot m\cdot E_f^m\cdot(1-E_f)$$

$$= W_s \cdot \frac{E_f(1-E_f^{n-1})}{1-E_f} - (n-1)W_s \cdot E_f^n(1-E_f) \qquad (2\text{-}24)$$

したがって，SREJ と同様有効回線利用率 L を求めると次式となる．

$$L = \frac{W_s \cdot T_f}{(W_s+R)T_f + T_s}$$
$$= 1 \Big/ \left[1 + \frac{E_f(1-E_f^{n-1})}{1-E_f} - (n-1)E_f^n(1-E_f) + W_s \frac{E_f^n}{n(1-E_f)} \right.$$
$$\left. + W_s(W_s-1)\frac{E_f^{2n+1}\cdot(1-E_f)}{n(1-E_f^2)^2} + \frac{W_s(W_s-1)(W_s-2)E_f^{3n+2}(1-E_f)}{2n(1-E_f^3)^2}\right] \qquad (2\text{-}25)$$

2.2.3 理論値の評価・分析

式(2-6)および(2-13)の T_r/T_f として図4の W_s を代入し，REJ 再送方式の有効回線利用率を求め，図6に示す．図の破線は実測値を示す．

図6によると，実測値と理論値とはほぼ一致している．実測値が理論値より小さな値を示しているが，これは理論値のビット誤り率が文字誤り率と等しいとしたためで，実際には文字誤り率はビット誤り率より大きな値を示すであろう．ビット誤りがランダムに発生した場合，文字誤り率はビット誤り率の約8倍となり理論値は約一桁左にずれる(10^{-4} の値が 10^{-5} へ)．しかし，実際のエラーはバースト的に発生するので文字誤り率は，それよりかなり良い値を示すと考えられる．したがって，文字誤り率とビット誤り率の関係が明確になれば，理論値は実測値と良く一致するであろう．

式(2-20)，(2-6)および(2-21)を使用して SREJ と MN-SREJ 再送方式の有効回線利用率を求め，図7に示す．図7によると，SREJ 方式は REJ 方式に較べそれほど効果が上っていないが，MN-SREJ 方式を使用した場合，かなり効率の改善が期待できる．とくに 10^{-3} では，REJ および SREJ 方式では使用不可能なほど効率が低下するが，MN-SREJ 方式では60パーセント近い伝送効率が得られ十分に使用に耐える．

また，エラー率 10^{-4} の場合の送信ウインドウ・サイズと有効回線利用率との関係を図8に示す．図によると，エラーの多い回線で2,000バイト/フレームのような長電文を送ると，MN-SREJ 方式の伝送効率は78パーセント以上あがらない，一方255バイト/フレームであれば，送信ウインドウ・サイズを大きくとれば98パーセントまで上げることができる．したがって，エラーの多い回線であまり長い電文とすることは，伝送効率の面からも好ましくない．

3. ネットワーク・レイヤ（DUC）の伝送効率

図1に示したように，DUC プロトコルは，ロジカル・ポート・セションと呼ばれる論理経路を介して伝送制御が行われている．ここでは，一つのロジカル・ポート・セションの伝送能力を分析する．ロジカル・ポート・セションの伝送効率の算定に当たり，回線の伝送効率を測定し伝送量を算出した．また，DUC プロトコルの可変項目として，**送信ウインドウ・サイズ**および**DUC パケット長**を選んだ．受信パケットに対する応答ウインドウ・サイズ W_r は，W_s との関係が $W_s = 2W_r - 1$ となるように設定した．

UDLC プロトコルは，最大伝送効率が得られるように送信ウインドウ・サイズを64と指定した．

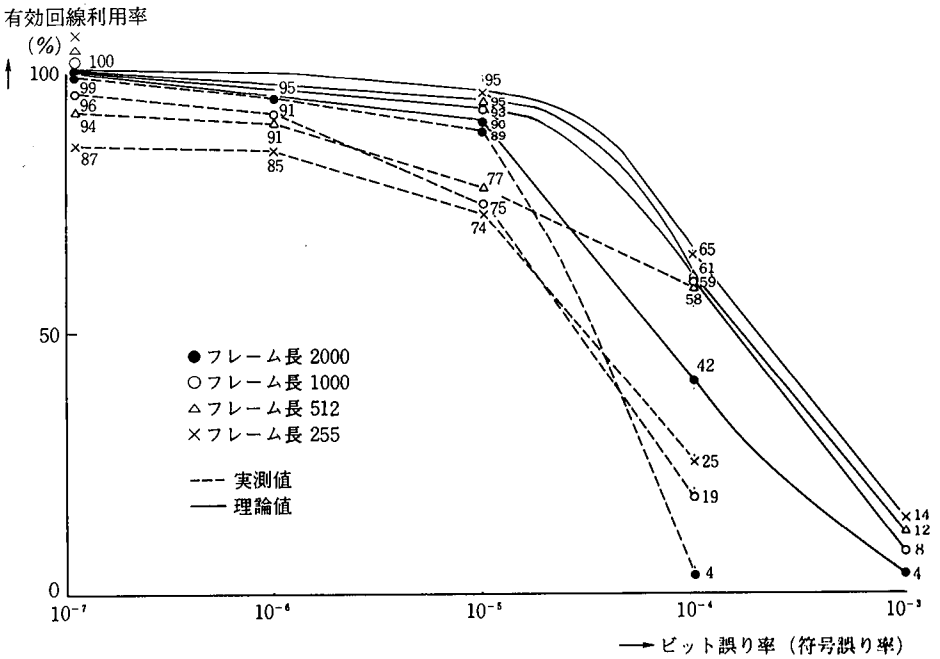


図6　REJ 再送方式の誤り率と有効回線利用率

Fig. 6　Bit error rate and effective line utilization in case of REJ recovery procedure

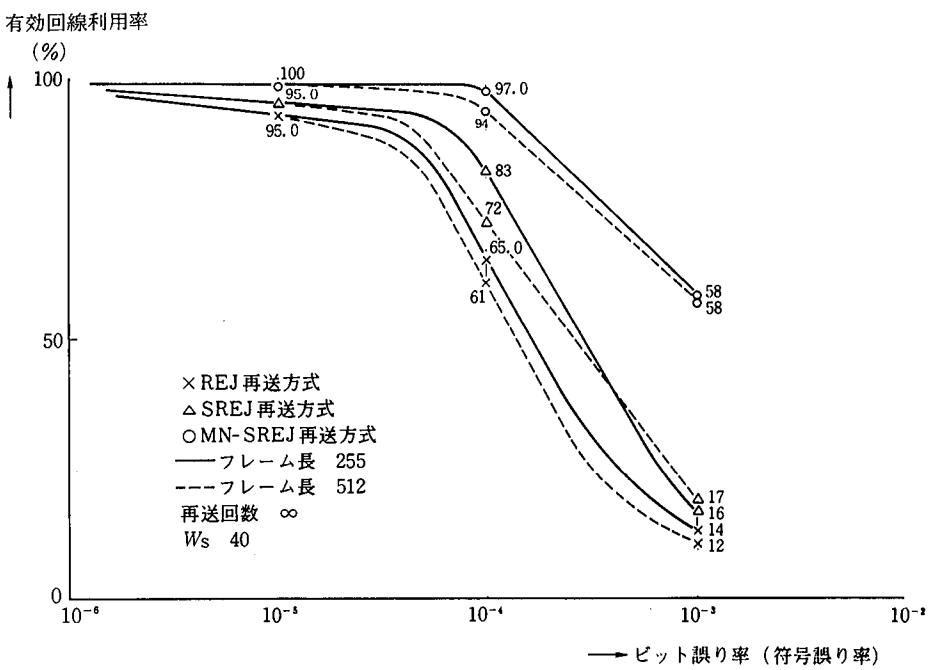


図7　REJ，SREJ および MN-SREJ 再送方式の有効回線利用率比較

Fig. 7　Effective line utilization in case of REJ SREJ and MN-SREJ recovery procedure

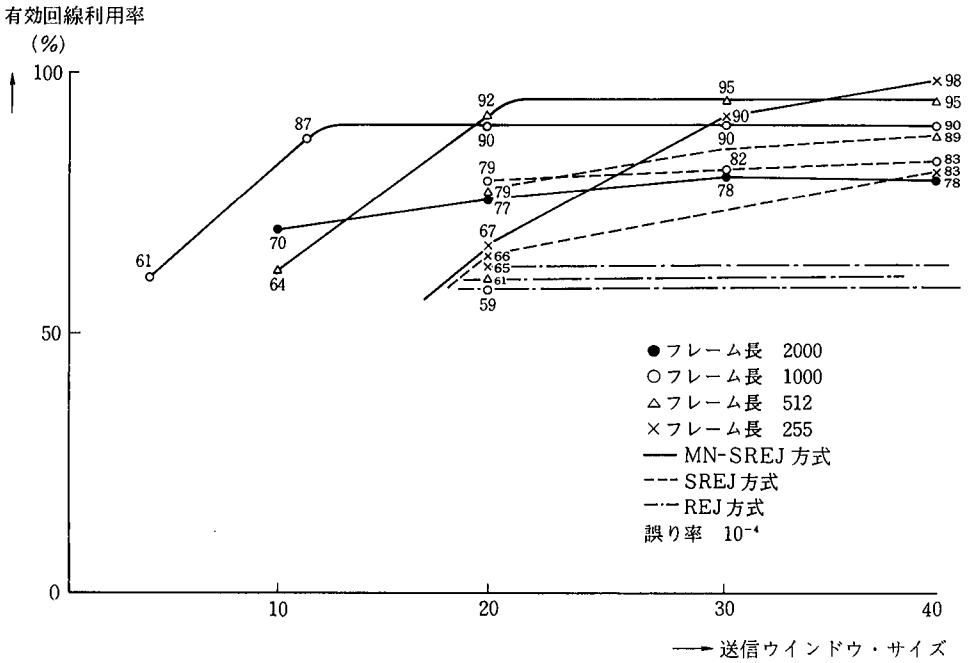


図8 MN-SREJ, SREJ および REJ 再送方式の有効回線利用率と送信ウインドウ・サイズ

Fig. 8 Effective line utilization and window size in case of MN-SREJ, SREJ, and REJ recovery procedure

3.1 実 験 結 果

地上回線1回線構成，および地上回線2回線構成の実験結果を図9に，衛星回線1回線構成，および衛星回線1回線に地上回線1回線を加えた構成の実験結果を図10にそれぞれ示す．

この実験では，一つのロジカル・ポート・セションを使用して，ファイル転送を行った．一般的には一つのデータ・リンクに，複数のロジカル・ポートが多重化される．その場合，必ずしも一つのロジカル・ポート・セションが48 K bps 1回線または2回線相当の伝送能力を持つ必要はない．しかし，本実験のようなファイル転送アプリケーションの場合，回線を最大限使用できるよう配慮する必要がある．

図9によると，地上回線1回線の構成で，一つのロジカル・ポート・セションの能力として，48 K bps 回線を最大限使用できるようにするためには，送信ウインドウ・サイズ8で，パケット長1,300バイト以上，また地上回線2回線の構成では送信ウインドウ・サイズ16でパケット長500バイト以上とする必要がある．

図9，10からファイル転送のような大容量伝送のアプリケーションでは，パケット長を1,000バイト以上，送信ウインドウ・サイズを16程度とした方がよいことがわかる．

一方，デマンドやリアルタイム等のアプリケーションのように短かい(50～200バイト/メッセージ) 電文の多いシステムでは，送信ウインドウ・サイズを16～32と大きくとらなければならない．とくに，一つのロジカル・ポート・セションに多数のシステム・セションを多重化しているような場合，留意する必要がある．

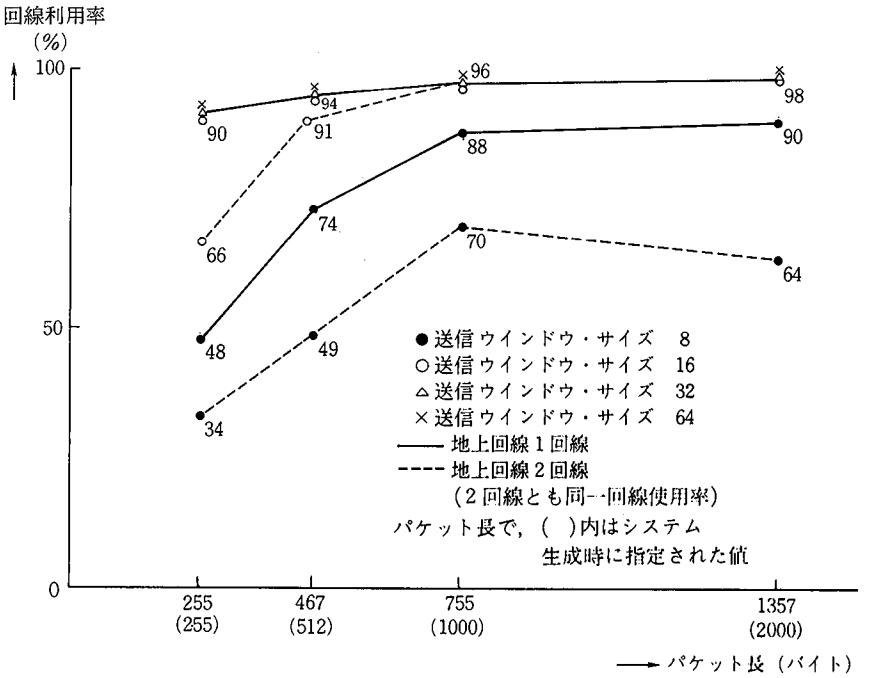


図 9　地上回線 1 回線および地上回線 2 回線の DUC レイヤの伝送効率

Fig. 9　Transmission efficiency of DUC layer in case of terrestrial circuit

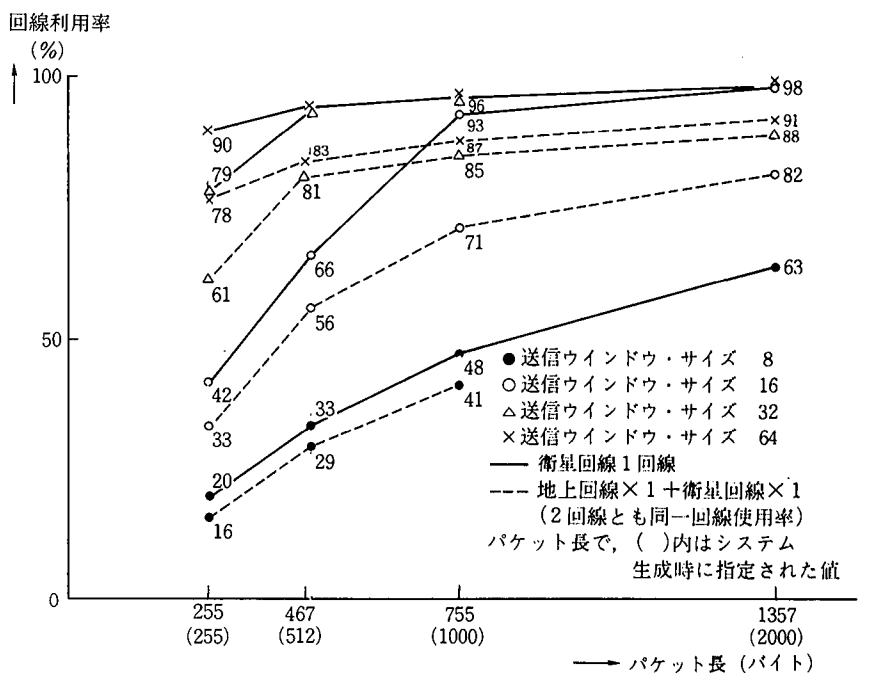


図 10　衛星回線における DUC の伝送効率

Fig. 10　Transmission efficiency of DUC layer in case of satellite and terrestrial circuit

図10の衛星回線1回線構成では，送信ウインドウ・サイズ16で1,000バイト以上，送信ウインドウ・サイズ32で512バイト以上必要である．

図10の衛星回線1回線および地上回線1回線構成では，衛星回線経由の場合と地上回線経由の場合で，パケットの到着時間が異なるため，パケットの順番が逆転する．このため，DUCで順番の並び換えを行っているが，その処理オーバーヘッドでパケット長を長くしても，地上回線2回線構成と等しい伝送効率は得られない．このことから，同一トランクに伝送速度の異なる，または伝送遅延がある回線を混在させることはあまり好ましくないことが判る．これに対応するためには，個々のロジカル・ポート・セションが伝送媒体を選択使用できるような構造にすべきである．

3.2 理論式の導出

一つのロジカル・ポート・セション（ビット・パイプ）を物理的な回線とみなし，タイミング・チャートに表すと図11のようになる．図では W_s が W_r の2倍の場合のタイミング・チャートを示した．本実験では片方向のファイル転送を行ったので，図のタイミング・チャートも片方向伝送の場合について示した．この場合，1ロジカル・ポート・セション（ビット・パイプ）の伝送容量 V_s と W_s およびパケット長 l の関係式は，次式で表される．

$$V_s=\frac{W_s\cdot l}{\delta} \tag{3-1}$$

ここで，δは，図11の遅延時間（秒）である．

また，$W_s<2W_r$ の場合のタイミング・チャートを図12に示す．

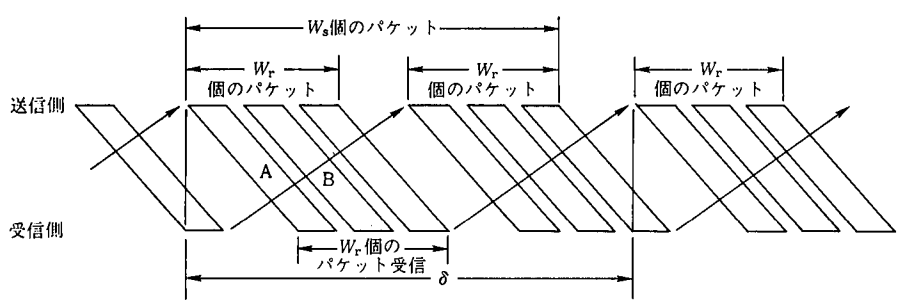
この場合の，V_s は次式の通りである．

$$V_s=\frac{l\cdot W_r}{\delta} \tag{3-2}$$

一つのロジカル・ポート・セションの伝送容量の一般式は，次式で表される．

$$V_s=\frac{l\left[\frac{W_s}{W_r}\right]W_r}{\delta} \tag{3-3}$$

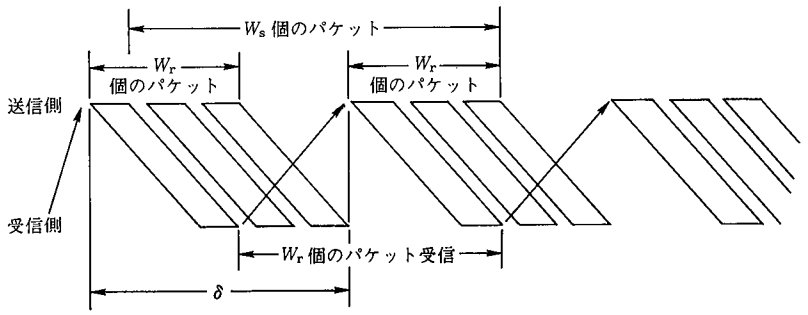
DUCプロトコルでは，受信電文の応答を送信電文に付加して送信する．したがって，ロジカル・ポート・セションを全二重で使用したとき V_s は，ほぼ $W_s=2W_r$ の場合と



δ：W_r個のパケットを送信後，それらのパケットに対する応答が戻るまでの平均時間

図11 $W_s=2W_r$ の時のDUCタイミング・チャート

Fig. 11 DUC timing chart in case of $W_s=2W_r$



δ：W_r 個のパケット送信後，それらのパケットに対する応答が戻るまでの平均時間

図12　$W_s<2W_r$ 時のDUCタイミング・チャート

Fig. 12　DUC timing chart in case of $W_s<2W_r$

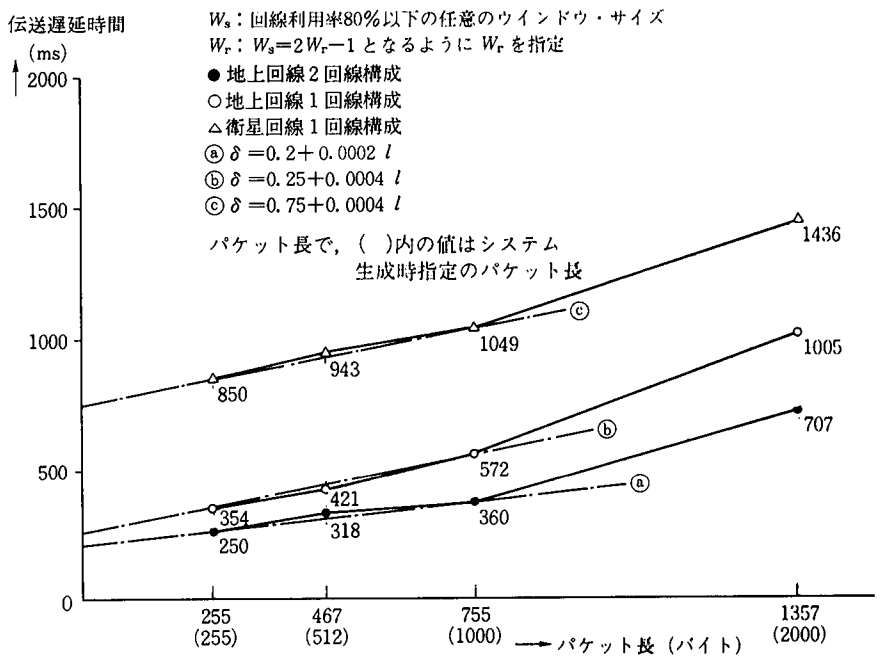


図13　パケット長と伝送遅延時間

Fig. 13　Packet size and transmission delay

同等となる．

本実験では，$W_s=2W_r-1$ の片方向伝送として使用したので式(3-2)を用い，図9および図10のデータ（ただし，回線利用率80パーセント以下の値を使用）から，δ の値を求める．δ の値は W_s に依存しない，パケット長でほぼ一定の値を示す．この値をパケット長を横軸にして図13に図示する．

図13より，パケット長とパケット伝送遅延時間は，パケット長1,000バイトまでほぼ直線的に増加している．また，衛星回線1回線と地上回線1回線との遅延時間の差は約500 msで往復の衛星回線の遅延時間と等しい．図より，近似的に δ と l との関係を求めると，それぞれ図13のⓐ，ⓑおよびⓒのようになる．これらを，式(3-1)に代

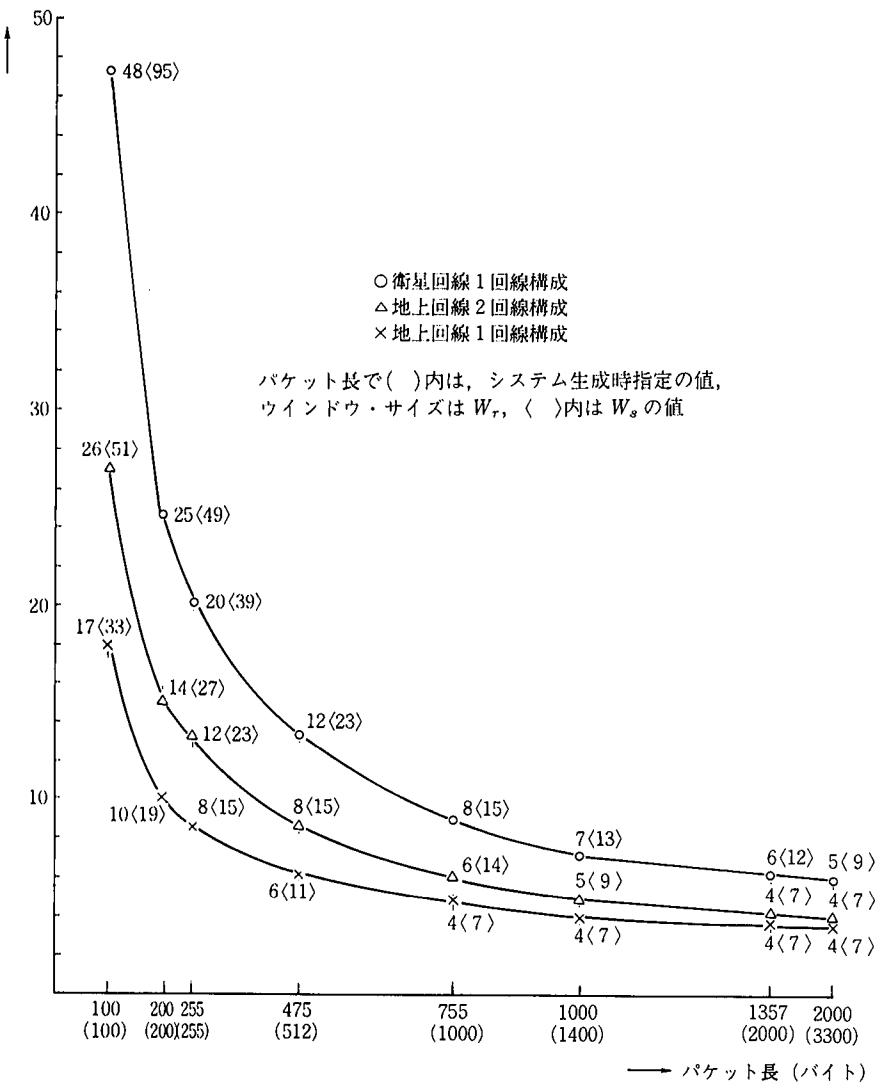


図14　パケット長と応答および送信ウインドウ・サイズ W_r〈W_s〉の関係

Fig. 14　Packet size and window size

入すると，(3-4)，(3-5)および(3-6)が得られる．

$$V_s=\frac{1000\,W_r\cdot l}{0.4l+750}\quad\text{(衛星回線1回線構成)}\qquad(3\text{-}4)$$

$$V_s=\frac{1000\,W_r\cdot l}{0.4l+250}\quad\text{(地上回線1回線構成)}\qquad(3\text{-}5)$$

$$V_s=\frac{1000\,W_r\cdot l}{0.2l+200}\quad\text{(地上回線2回線構成)}\qquad(3\text{-}6)$$

これらの V_s に，それぞれ回線の最大伝送容量として6,000バイト/秒，6,000バイト/秒，および12,000バイト/秒を代入すると，一つのロジカル・ポート・セションで回線を最大限使用するため必要な l と W_r の関係式(3-7)，(3-8)，(3-9)が得られる．

$$W_r=2.4+\frac{4500}{l} \quad (衛星回線1回線構成) \tag{3-7}$$

$$W_r=2.4+\frac{1500}{l} \quad (地上回線1回線構成) \tag{3-8}$$

$$W_r=2.4+\frac{2400}{l} \quad (地上回線2回線構成) \tag{3-9}$$

これを，図14に示す．図の値は実験結果の図9および図10の回線使用率を最大とするため必要な l および W_s の値と一致している．

式(3-7)，(3-8)および(3-9)を使用することにより，パケット長からシステムで要求されるロジカル・ポートの伝送能力を算出し，最適のパラメタを得ることができる．ただし，ここでのパケット長は，システム生成時の値ではなく，上位のプロトコル・データを含んだ実際伝送される平均パケット長であることに注意する必要がある．なお，システム生成時のパケット長は図14の(　)内に示した．

4. おわりに

今回導出したDCAプロトコルのデータ・リンク・レイヤの理論値は実測値のデータと一致しており，実用上十分有効であることが確認できた．

ネットワーク・レイヤ（DUCプロトコル）の伝送効率に関する理論式は，適用範囲が限られており(ノードの数が2個，パケット長が2,000バイトまで，等)，今後はこれをマルチ・ノード（ノードが3個以上）の場合や回線速度の異なる回線が同一トランク内に混在した場合等に拡張してゆく必要があり，これらについても分析する予定である（マルチ・ノード構成の場合の理論式については参考文献[2]参照)．

今回はネットワーク・レイヤまでの伝送効率の分析であったが，今後は上位のレイヤ（トランスポート，セション・レイヤ）の伝送効率についても分析してゆきたい．

なお，この実験は郵政省が推進している「衛星通信パイロット計画」の一環として行われたものであり，電波研究所の指導のもとに実施された．

参考文献　[1]W. Feller 著，「確率論とその応用 I」(上)(下) 紀伊国屋書店，1977.
[2] Mischa Schwarts, "Performance Analysis of the SNA Virtual Route Pacing Control", IEEE Transaction on Communication, Vol. COM30, pp. 172～184.

執筆者紹介　宮坂順之 (Yorihisa Miyasaka)
昭和19年生．43年東北大学理学部物理学科卒業．44年日本ユニバック㈱入社．主として，データ通信システムの設計および開発に従事．現在，商品開発本部ソフトウェア計画部アドバンスト・ソフトウェア室に所属．

論文 再使用可能コード・ライブラリ BIBLIO の開発

BIBLIO——The Implementation

K. Bruso, P. Wright

要 約 再使用コード・ライブラリ BIBLIO（技報 1987 年 5 月第 13 号，pp. 58～67）の開発において，技術上および手続き上の問題に数多く遭遇し，それらに対処することで貴重な経験が得られた．本稿では開発時の問題を考察し，実際にとられたアプローチの理由づけを行う．

その問題は，以下の通りである．

1) コード・カタログの作成
2) コード・ライブラリ検索システムの作成
3) ライブラリの充実
4) システム利用者へのコードの伝達および配布
5) リリースと保守

Abstract Implementation of the BIBLIO reusable code library has raised a number of interesting and challenging technical and procedural issues.

This paper examines some of these issues, the rationale behind the implementation approach.

The issues include:

1) Creating the code catalog
2) Creating a retrieval system to search the code library
3) Evolution of the library's contents
4) Communication/distribution of code to developers
5) Release and support.

1. はじめに

一般にソフトウェア開発の手順は，コードの機能を定義しデザインした後にコーディングをゼロから開始する．しかし，われわれの作ろうとするコードが，すでに他のプロダクトで作成され，しかもそのコードはテストに合格した信頼のおけるものであるといった状況が数多く存在する．この場合ゼロからコーディングするのは，非常に無駄であり，コードを互いに共用できるような手段が必要である．

すでに実用化されている例では，ER 命令や SYSLIB ルーチンのような単純な機能があり，これらは皆が使いやすい形で提供され，ドキュメントも完備している．この方法を，ER 命令のように単純な機能だけでなく，高水準言語で書かれたコード・モジュールにまで応用すれば，より大きく強力なコード・ブロックを多くのソフトウェア・システムで利用できる．

BIBLIO プロジェクトは，コードの再利用を可能にするツールと資源を提供し，再使用可能コード・ライブラリ・システムを構築するものである．

2. プロジェクトの目的

BIBLIO プロジェクトの目的は，「高品質でかつ完全に検証済みのコード・モジュールをライブラリとして登録し，ソフトウェア開発段階でのビルディング・ブロックとして自由に引用できるようにし，プログラマの生産性を高める」ことである．結果として開発および保守費用を減少させ，作業スケジュールの遅延を減らすことができる．

3. BIBLIO の要求仕様

再使用可能コード・ライブラリは，以下の条件を満足しなければならない．

1) **保守の実施**……ライブラリに登録されたコードは，プロダクトとしてサポートする．すなわち，ライブラリ管理者は，SUR（問題報告書）を受け付け，問題解決を図る．そのコード内で他のコードを参照している場合，その部分の保守についても責任を持つ．

2) **伝達・配布機構の設置**……コードは，プロジェクト間，または異なる開発部署間で使用可能にする．プロジェクト・チーム相互，およびプロジェクト・チームとライブラリ管理者が，互いに情報を交換できる機構を作る．その機構によってコードの再利用を促進し，またプロジェクト間で使えるビルディング・ブロックの情報が簡単に得られるようにする．このほか，ツールを用意し，ある条件を満たすコードの検索，ライブラリへのコードの追加，またライブラリへの提案が楽にできるようにする．

3) **品質の保証**……コードは十分テストされ，ドキュメントが整備されていること，高品質（利用者が自分のプログラムに組み入れたいと思う程度）を備えていることが求められる．

4) **内容の充実**……ライブラリに登録されるコードは，それ自身が積み重ね可能で，より強力なブロックを形成することができること．機能を追加することによって，ライブラリ・コードの利用価値（新しくコード作成する費用の節約）が増し，ライブラリ自体をより魅力的なものにする．

4. プロダクトの構成

再使用可能コード・ライブラリを構築するためのツールを開発し，構築手法を確立する．BIBLIO のプロダクト構成は以下の通りである．

1) コード・ライブラリを構築する生成手続きマニュアル

2) 次の 2 点の機能を果たすツール

①ライブラリに登録されたコードの目録(コード・カタログ)の作成

②ある条件を満たすコードのライブラリからの検索

なお，再使用可能コード・ライブラリを効率的に管理するには，ライブラリ管理者の役割が不可欠である．個々のライブラリには，熟練したプログラマが管理者として割り当てられる必要がある．ライブラリ管理者の権限は，手続きの生成・展開，コードの内容変更，コードの配布・保守および必要な場合はライブラリに関するプログラマの教育にまで及ぶ．

5. 使　用　例

BIBLIOの使用例として，こんなケースを頭に浮かべてほしい．ここではBIBLIOの良さを知ることが第1の目的なので，BIBLIO操作の詳細については触れていない．

《使用のシナリオ》

一人のシステム・アナリストが端末にやってきた．彼女に今，割り当てられた仕事は，コンティンジェンシ・ハンドラの上位レベルのデザインを，下位の詳細デザインに落として展開することである．早速，彼女はBIBLIOを呼び出し，コンティンジェンシ・ハンドラが過去に開発されたかどうか，そしてライブラリに入っているかを尋ねた．

あった！．BIBLIOは3件該当するものを知らせてくれた．リアルタイム版，DMS応用版，そしてジェネラル・コンティンジェンシ・ハンドラの3件のソフトウェアで，すべて“受け入れ承認済み”(APPROVED)のステータスである．ということは，コードのドキュメントは完備され，テストも済み，そして何といっても使えるコードがある．彼女がジェネラル・コンティンジェンシ・ハンドラを選ぶと，画面にその説明が映し出された．ここで本当に自分の欲しいモジュールであるかどうか，確認できる．

次の仕事は，彼女が使用しているのとは別の開発言語で書かれたルーチンへのインタフェースをデザインすることとなった．再びBIBLIOを検索してみたが，今度は残念ながら該当するものがなかった．ということは今回は，彼女がライブラリに貢献できる立場となったわけである．彼女はBIBLIOに受け入れられるコードの基準を確認し，その基準を満たすようにデザインを進めた．

しばらくして……．

彼女からデザインを受けとったプログラマは，コーディングを終えたところで，いつにも増して満足感を味わった．というのは，彼が作成したコードは，再使用可能ということを念頭において作られているので，今後多くのプロダクトで使ってもらえるからである．鼻高々に彼はBIBLIOライブラリに，そのコード・モジュールの説明を入力した．

その日の終わりにBIBLIOライブラリ管理者がきて，ライブラリに入力されたコードの説明書を見直し，「システム開発者達は，このライブラリ・システムの要領がわかってきたな．追加されたコードは，みなキーワードを付けるだけで“登録候補”(CANDIDATES)にできる質の高いものばかりだ．登録候補はライブラリ検索の対象となる．そして今週中にテストを終えて，“受け入れ承認済み”のステータスに変えられるだろう」と思った．

6. 開　　発

再使用可能コード・ライブラリを整えるために，いくつかの問題を把握する必要がある．以下でそれらの問題を検討し，解決のための要求仕様と実現方法を説明する．

6.1 コード・カタログにおける説明書の役割

BIBLIOライブラリに入れるため，コード・モジュールの説明記述のテンプレートを作成しなければならない．

6.1.1 要 求 仕 様

1) コードが書かれている言語にかかわりなく，そのコードを説明し得るものであること．

2) システム利用者がその記述内容を読み，コードの機能やインタフェースを理解し，いくつかの選択候補の中からそのコード・モジュールを選択するかどうかを決定できる十分な情報を提供できること．

6.1.2 実 現 方 法

一度，限定した範囲での仕様を書き，次にできるだけ一般化した形での仕様に書き直してみる．そして，各々の段階で，多くの人に査読してもらい，正しく理解できる記述であるかどうかを確認する．モジュールが再使用できるかどうかは，ひとえにユーザが，選択候補としてモジュールの動きを理解できるかどうかにかかっている．

6.2 コード・カタログ処理ツール

コード・カタログを保守するためのツールが必要である．

6.2.1 要 求 仕 様

1) コード・カタログに対して，コードの登録・更新・削除および報告書を得る機能が提供されること．

2) 種々の条件のもとに，コード・カタログから特定の掲載コードを抽出する機能が提供されること．

3) コード・カタログの特定のアクセスを規制するセキュリティ機能が提供されること．

4) ライブラリ内の各掲載コードの利用状況に関する統計情報を持つこと．たとえば，コード別使用者数など．

6.2.2 実 現 方 法

PRIMUS を用いてプロトタイプを作る[11]．PRIMUS はキーワード探索，可変長レコード形式の指定，非常にわかりやすいユーザ・インタフェース，およびアクセス制御などの機能を持っている．また，プロトタイプ作成で得られた経験を活かして，別の実現方法を検討することもできる．

このほか種々の異なる環境 (1100, UNIX* 等) や，さまざまなライブラリ実用例も参考になる．こうやって徐々に BIBLIO ライブラリ操作機能を統合し，フル・スクリーンでメニュ形式のインタフェースを開発していく．

6.3 検 索 機 能

一連の機能要求を用い，それらを条件として，ライブラリ・コードからコード・モジュールを検索できるような方法を開発する．

6.3.1 要 求 仕 様

1) 操作が簡単であること．

2) ソフトウェアを特定して引くか，ソフトウェア名を特定せず機能で引くかを指定できるようにする．たとえば，ファイルの割り当てを行うコードを求めている場合，'ER CSF $' と指定しても同様に検索できるようにする．

---

* UNIX は，米国 AT & T 社の Bell 研究所で開発されたオペレーティング・システムで，AT & T がライセンスしている．

3) ライブラリを展開することが可能であること.

4) 選択条件をきつくすることで，ライブラリからの選択幅を狭められること.

6.3.2 実 現 方 法

キーワードを利用し，次の二段階で開発する.

1) コード・モジュールにキーワードを割り当てる手法を決める．ソフトウェア名による検索と，機能による検索の両方を検討する.

2) 次に，キーワードの自動抽出ツールを開発し，ライブラリ検索および新規モジュールを登録する際のキーワード決定に利用する.

典型的な検索機能には，①要約のみをスキャンする機能，②主題・作者・作成日付・キーワードによる選択機能，③キーワードのマスク化，④同義語の使用（たとえば 'modify' と 'update' を同義に使えるようにする），⑤スペリング検査機能，⑥論理式の使用などがある.

6.4 機 密 保 護

カタログ中の各掲載コードの内容は，ライブラリの利用者が使用する前に，登録設定しておかねばならない．掲載コードの内容の更新は，権限のある者にだけ許される．また，掲載コードの機密レベルとサポート・レベルについても明記しておく必要がある.

6.4.1 要 求 仕 様

1) 開発者は誰でもライブラリに候補コードを提示できる.

2) コードは，ライブラリ管理者によって保守されない場合でも，開発チーム内で共用できる.

3) ライブラリ中の特定掲載コードに対するアクセスを，開発チームのメンバのみに限定することで，局所的ライブラリを形成できる．たとえば，機密アプリケーション担当グループ内で，共通のコードを共有し，部外者はアクセスできないようにすることが必要である.

6.4.2 実 現 方 法

誰でもライブラリに候補コードを出せる．ただし，その候補コードは，ライブラリ管理者が，形式上の情報チェック（たとえば個々の情報フィールドが正しく入力されたか，キーワードが正しく使用されたかのチェック）を完了しカタログに仮登録され，“登録候補”というステータスを付けられ，初めて検索可能となる.

コードがいったんライブラリに入った後は，作成者は自分のコードのカタログの更新はすることができるが，ライブラリ管理者のレビューを受けないと，その変更は正式版に反映されない.

ライブラリ管理者は，コードが受け入れ規準に達するかどうかを調べる．そして，コードが検証されると，そのコードは“受け入れ承認済み”というステータスとなり，ライブラリ管理者はその後の保守の責任を負う．ここからはライブラリ管理者のみが，そのカタログ掲載項目の更新を行う権利がある（ただし，“ユーザ名”というようなフィールドは例外である）.

コードの利用者は，掲載項目に対してコメントを述べたり，修正を提案することができる．また，自分の個人用コピーをとり，一部修正をし，ライブラリ案として再提

示も行えるようにする．情報を分割し，アクセスを段階づけて制限できるようにする（たとえば，ライブラリ管理者のみとか，ライブラリ管理者と開発グループのみ，というように）．

6.4.3 問　題　点

受け入れ規準はいくつかのレベルを設けて，コードのリリース形態によって区別できるようにしたい．たとえば，一般にリリースするコードと，信頼性はあるが限定使用とするコードは，異なった規準を持つべきである．

6.5 配　　布

コード・ライブラリは，多くの人に使用されなければならない．

6.5.1 要 求 仕 様

コード・カタログは，できるだけ多くの人が利用できるように配置する．

6.5.2 実 現 方 法

ライブラリ・ツールの開発方法に合わせて，ライブラリ・データベースの集中/分散の方法を選択する．プロトタイプ・システムとして中央の PRIMUS データベースを使用する．中央の PRIMUS システムで選択されたモジュールは，それを使用するユーザの個別システムに移送される．ユーザのシステムすべてに PRIMUS データベースを持つことは，現状において無理である．なお，本番システムが稼働した場合でも，中央のデータベースを設定し利用することができる．アクセスしやすくするため，中央のデータベースをユーザ・システムにコピーすることも考えられる．

6.5.3 問　題　点

ライブラリの使用を広めるためには，ライブラリ要員を十分に用意する必要がある．一つのデータベースを複数のライブラリが共有する場合，ライブラリ操作の調整も必要である．

6.6 ライブラリの内容

コード・モジュールは，汎用であり，有用かつ再利用できる形で作られている必要がある．

6.6.1 要 求 仕 様

ライブラリのコードはそれ自体，積み木のように積み重ね，より大きなブロックを形成できるものでなくてはならない．

6.6.2 実 現 方 法

初めてライブラリを作る時は，使用範囲を限定したルーチンを集める．たとえば，ER 命令へのインタフェースをとるインライン・コードなどである．まず，モジュールの内容が人々に喜ばれ，飛びつきたくなるような魅力があることを確認する．いったん人々の心をひきつけたら，次はより広い分野に使われるルーチンを指向する．

このようにして次々積み上げて，より大きなコード・ブロックを作ると，ユーザはコードを再利用することで，より大きな効果を得られる．再利用されるコンポーネントの大きさが小さいと，再利用のメリットも部分的で小さく，その労力に値しないかもしれない．しかし，大きなモジュールが再利用できれば，その見返りも大変大きい．また，コンポーネントは抽象的なものであればあるほど，見返りはより大きいと言える．一般的に抽象度が高ければ高いほど，より多くのソース・コードが節約できる．

再使用可能コード・ライブラリは，単に現存プログラムからコードを抜き出し，蓄積すればよいと思っている人が多いが，それではうまくいかない．それはゴミの山から廃車の部品を集めて，新しい自動車を組み立てようとするのに等しい．個々の部品の質の高さと同時に各々が組み合わさって正しく機能するよう，相互のインタフェースをとる中間アダプタが必要である．

ライブラリの中味を組み立てるためには，次のステップが必要である．

1) まず最優先の再使用可能コードのリストを作成する．いくつかのプロジェクトからの代表者で構成されるタスクフォースによって選択を行う．このリスト上の項目を集めることが，ライブラリの目標となる．リストにのせるコードは，クイック・ソートのような古典的アルゴリズムや，ほとんどすべてのプロジェクトでコピーして使用されているコード/ロジック，将来の機能の要求仕様なども対象とする．ライブラリのドキュメントは，SYSLIB や ER 命令のように既存ライブラリにも言及する．

2) 候補のコスト対効果を考慮し，リストのプライオリティ付けを行う．ユーザ数も一つの考え方であるが，再開発にかかるはずの費用も重要である．プライオリティ付けのもう一つの観点は，魅力の度合い（他人が，このコンポーネントをどのくらい使いたいと思うか）である．ソートのように確立された手法もあるし，ユーザ・インタフェースのように非常に創造性の高いものもある．

3) これらのルーチンを開発する際，開発者はルーチンが他人にとっても有益であるように配慮する．すでにできたプログラムのアイデアやデザインはそのまま使えるが，コード本体は多分一般化して整理する必要があるだろう．また，開発を請け負うのが一個人あるいは一グループである場合，オンライン会議（コンピュータ上のファイルを利用しての討議）で他の開発者からデザインをもらったり，交換したりするとよい．このほかライブラリ管理者も開発中にフィード・バックを与えられる．開発者は誰でもライブラリにコードを登録できるが，ライブラリ管理者が一般化の観点から，そのコードを修正することがある．もし別のプログラミング方法がみつかったら，ユーザが組み入れやすい方を選択する．しかし，ある環境のもとでは特定のアルゴリズムを用いるべきという条件つきの場合，複数の形式をライブラリに登録するとよい．この場合は，選択の条件を明記する必要がある．

4) まったく別のケースでは，ライブラリ管理者がプロジェクトで必要とされるコードの開発を請け負う場合がある．この場合，開発コードはライブラリに入れる前に実働環境で試さなければならない．

ライブラリ開発を奨励するためには，コードを利用するプロジェクトが節約できる費用，および短縮できる開発日数を計算する必要がある．ライブラリが予算上オーバーヘッド項目として，どのプロジェクトでも利用できるようになっているか，またはコード使用費を請求できるコスト・センタとして位置づけられれば，各プロジェクトにおける開発スケジュール上の悩みがいくらか解消し，ライブラリの活用・拡張が期待される．

6.7 検証(ステータス確認)

ステータスを“受け入れ承認済み”とするには,コードは高品質でなければならない.

6.7.1 要 求 仕 様

1) 規準を多く決めすぎると,ライブラリ・コードの開発や利用を減退させる.制限は,なるべくコードの利用に対してのみ付けるようにする.
2) 高品質のコード開発を促進するような規準を設ける.
3) 機械的な規準(判断)は理想的だが,制限がきつすぎる場合があるので,時には主観的決定を採用する.

6.7.2 実 現 方 法

コードがコード・ライブラリに正式に登録されるための受け入れ規準を設定しなければならない.これには二つの目的がある.第一にコードが高品質で保守しやすいことを保証する.第二に,コードを開発する際,この規準を目安にしてコードを開発できることである.

コード・ライブラリ管理者は,システムの中のコードに対して最高の権限と責任を持つ必要がある.なお,ライブラリの受け入れ規準とは以下のものを含む.

1) 文書……コード中に十分にコメントが付けられており,保守しやすいようになっているか,またコードとは別に仕様書が存在するか.
2) エラー回復……エラー状態やエラー・データを入力した時にリカバリできる考慮をしているか.
3) 再使用価値……ライブラリの内容とコードの具体的需要にもとづいて主観的に判断する.たとえば,実行サイズや実行速度など.
4) テスト……ユニット・テストのドライバーがあるか.テストがカバーしているエラーの範囲は十分か.テストの結果を判断できる資料が整備されているか.
5) モジュール・インタフェース……モジュールのインタフェースはわかりやすく,また効率的であるか.

モジュール相互でインタフェースがとれ,また相互の積み上げを可能にするには,プログラムの呼び出し,流れの制御および終了の手法が共通であること,また他のモジュールへの情報の受け渡し方の統一がとれている必要がある.

6.7.3 問 題 点

規準を厳しくするあまり,開発者の生産性を妨げてはならない.規準は自由度がないと使えない.たとえば,“読みやすさ”というように抽象的表現を用い,良い実例を参考にさせるのがよい.

6.8 コード・パッケージ

コード・モジュールのサポートと,リリースを成功させるためのパッケージを整備する.

6.8.1 要 求 仕 様

1) コードの採り入れについて,利用者の負荷をできるだけ少なくする.エレメント数や COPY 文がやたら多いのはよくない.
2) システム・ソフトウェアの変更やコードのテストが簡単にできるようにする.

6.8.2 実 現 方 法

コード・モジュールの“パッケージ”とは，以下を含む．

1) コード
2) データ構造や手続きの定義を含んだCOPYエレメント
3) ユニット・テスト・ドライバー，必要であれば入力データ，ユニット・テスト結果，リンク用ラン・ストリーム
4) キーワード等のコード・カタログ上の説明記述

コード・ライブラリにとって理想的なリリース形態は，オブジェクト・コード形式である．というのは，開発の詳細をみせないので，ユーザは修正できないからである．後で別のプログラムに置き換える場合も，利用者であるプロジェクトのことを考えなくてよいし，信頼性を失うこともなく，コーディング形式を各プロジェクトのスタイルに合わせて調整することも必要ない．オブジェクト・コード形式のリリースは，ユーザの手間が省ける．たとえば，モジュール化されたメモリ処理ルーチン群は数多くのエレメントから成るが，一つのユニット上にリンク付けることができる．こうすると，使用する外部定義名の数も減る．

6.8.3 問 題 点

BIBLIOの使用を簡単にするため，コンパイラ，コレクタ，リンキング・システムから使用するファイルを，ソフトウェア・インストレーション時に設定する方法が考えられる．コンパイラが何度も利用するCOPYファイルには，BIBLIOインストレーション・ファイルにUSE名を付ければよい．これはファイル登録プロセッサが実行する．また，リロケータブル・コレクションの際は，コレクタにBIBLIOファイル名を教えればよい．なお，オブジェクト・モジュールのリンクは，インストレーション時にBIBLIOファイル名をリンキング・システムに知らせて行う．

6.9 報奨制度＝教育＝コミュニケーション

ライブラリのコードを利用させたり，コードを集めるためには，BIBLIOの宣伝をしなくてはならない．

6.9.1 要 求 仕 様

コードを再利用することによって，コーディング時間，ユニット・テスト時間，保守時間は減少するが，当初はかなり開発コストがかかる．このため，ライブラリ・コードを作ることを奨励し貢献した場合には，その努力に報いるべきである．

6.9.2 実 現 方 法

ライブラリに登録されたコードは出版物に載せ，各プロジェクトや各使用部所間に情報を流す．

ライブラリ管理者とシステム開発者との間で情報交換するには，中央集中的な電子会議が便利である．ライブラリ管理者は，システム開発者がコードを探すのを手伝ううちに開発者の要求を把握する．システム開発者は，ツールを用いてある要求を満たすコードを検索したり，コード候補を提供したり，ライブラリ中のコードを批評したりすることがたやすくできる．コードの利用者は，意見・提案・感想，および質問を自由に伝えることができる．

ライブラリ中のあるコードを再使用しているプロジェクトは，そのコードに変更が

なされたとき，電子メールによって知らされる．また，システム開発者はソフトウェア・ライフ・サイクルのなるべく初期の段階で，コードの再利用を奨励すべきである．そうすれば，コードの受け入れ規準を事前に考慮してコードを開発できる．このほか，コード提供を行った者に名誉を与えるとともに表彰すべきである．

6.10 リ リ ー ス

ライブラリ・ツールと手法のみでなく，集めたライブラリ・コードもまた収入源となるプロダクトとしてリリースされうる．

6.10.1 実 現 方 法

ライブラリで“受け入れ承認済み”となったコードは，分離したコンポーネントとして商品化する．ライブラリのコードは，単独で価格のついた商品となる可能性が大きい．ライブラリ管理者が責任を持って，ライブラリを分割しパッケージ化する．この実例に ER インタフェース・ライブラリや，グラフ・ライブラリがある．

コード・ライブラリは，カテゴリー1プロダクト，すなわちユーザが問題報告と問題解決要求を正式に出せるプロダクトとしてサポートすべきである．コードのサポートは，利用者の立場であるプロジェクトが行うのでなく，ライブラリ管理者が責任を持ち，定期的リリースを行い，過去のレベルとの互換性を保つ必要がある．

7. 将 来 の 課 題

将来，取り組むべき課題として次のものがあげられる．

1) コード・ライブラリが軌道に乗ったら，コード・テンプレート，デザインおよび文書をライブラリに追加する．

2) DUMP, FSED, COMPARE などのスタンド・アロン・ツールもライブラリに登録する．

3) 再利用上の問題および，その他の観点での問題として以下のものが挙げられる．

① ソフトウェア生成ツールの利用

・言語生成ツール……GSA のように，目的システムを生成するために特定の表記法を用いるツール

・アプリケーション生成ツール……画面ジェネレータ（対話形式で，一部記号化された入力データの展開）など

・仕様を実行可能なプログラムに翻訳する形式的仕様変換システム

② 要素結合（構成要素をいかに組み合わせて一つのシステムを形造るか）

・手続き呼び出しのネストの制限

・UNIX との連結や入出力先の変更

・ADA* のパッケージ記述のような共通のインタフェース・デザイン

・情報隠蔽

8. 開発スケジュール

短期間の開発フェーズを設ける．開発の順序は次の通りである．

---

* 米国政府の登録商標である．

1) プロトタイプ作成……ライブラリ文書と初期の手法設定のため，PRIMUS を使ったプロトタイプを作る．ツールと手法改良のため，小さいライブラリを構築する．内容は限定された分野のもので，すでに受け入れ規準を満たし，一般ユーザにとって価値あるものとする．最初のパッケージは，このように内容を特定する．

2) 一般化……1)が成功した後，対象分野を広げてツールと手法の一般化および拡張をはかる．

3) 開発方法の決定……環境として，1100，UNIX，パーソナル・コンピュータおよびその他のシステムを考える．新しいシステムを一から作るか，それとも既存の文書ライブラリ管理システムを利用するかを考える．プロトタイプを作成した経験が，実システムの決定や要求仕様の決定に役立つ．

4) コード・カタログとコード検索ツールの開発およびパッケージ化手法は洗練されたものを使う．

5) ライブラリ・コードの収集……計画を立ててライブラリを構築する．広範囲のユーザへのライブラリ開放と同時に，奨励制度の実施とプログラマ教育を行う．

9. おわりに

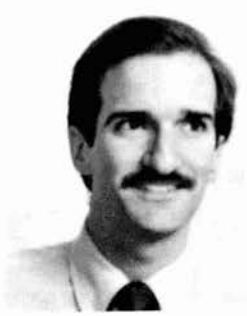
ソフトウェアの再利用は今後の課題であり，いろいろなアプローチがされて行くと思う．本稿がこの問題に関心を持つ人の参考になれば幸いである．

（ソフトウェア3部　下田 和江訳）

参考文献 [1] T. C. Jones, "Laying the Groundwork with Reusable Code", Computer World, Vol. 18, No. 26A, Jun., 27, 1984, pp. 12～16.
[2] T. C. Jones, "Reusability in Programming: A Survey of the State of the Art", IEEE Transactions on Software Engineering, Vol. SE-10, No. 5, Sep., 1984, p. 488.
[3] Y. Matumoto, "Some Experiences in Promoting Reusable Software: Presentation in Higher Abstract Levels", IEEE Transactions on Software Engineering, Vol. SE-10, No. 3, Sep., 1984, pp. 502～512.
[4] L. Bernstein and C. M. Yuhas, "Blood from Turnips?", Datamation, Jan., 15, 1985, pp. 198～112.
[5] E. Horowitz and J. B. Munson, "An Expansive View of Reusable Software", IEEE Transactions on Software Engineering, Vol. SE-10, No. 3, Sep., 1984, pp. 477～487.
[6] S. Thoman, "An Essay on Software Reuse", IEEE Transactions on Software Engineering, Vol. SE-10, No. 3, Sep., 1984, pp. 488～497.
[7] R. Lanergan and C. Grasso, "Software Engineering with Reusable Designs and Code", IEEE Transactions on Software Engineering, Vol. SE-10, No. 3, Sep., 1984, pp. 488～501.

執筆者紹介 Kelsey L. Bruso

1981年にIowa州Luther Collegeより音楽とコンピュータ・サイエンスの分野でB. A. を取得．Sperry社（現Unisys社）でRDPの開発と保守に従事した後，COMUS開発に参加し，システム構成の動的制御と動的コード・パッチングを担当している．



Pam Wright

Wisconsin-La Cross 大学よりコンピュータ・サイエンスで B. S.を取得．Sperry 社（現 Unisys 社）でシステム・プログラマとして 4 年の経験を持つ．ソフトウェア測定ツール SOFMET，データ辞書システム DDS および構造的設計言語 SDL の開発に従事した後，現在 BIBLIO の開発を担当している．

論文

リレーショナル・データベースの考察
——整合性制約を中心として

Consideration of Relational Database
——Centering around Integrity Constraint

原　　潔

要　約　データベース管理システムの目的は，データを利用者が必要とする形にまとめて提供することにある．データの利用を容易に行えるようにするには，データ独立性の実現が重要である．

E. F. Coddの提案したリレーショナル・モデルは，数学におけるリレーションという概念をデータベースに応用したものである．モデルの簡明さと，操作の高水準性によりシステム開発の生産性向上に寄与するとして期待されている．リレーショナル・モデルは，SQL言語を利用者インタフェースとするリレーショナル・データベース管理システムとして多く実現されている．この実現においては，現実世界の要求や性能面への妥協により，本来のモデルの目指していた目的を損ないかねない状況がある．そのようなものの一つに，データの整合性制約の問題がある．データの整合性の維持は，現実世界の意味を捕えるためにも，データベース中のデータを自由に操作してもその整合性が損なわれないためにも，重要な機能である．

本稿では，リレーショナル・モデルの目指したところをCoddの論文を追跡することにより確認し，データベースの整合性制約，とくに参照整合性について述べる．

Abstract　A major role of a DBMS (Database Management System) is to allow a user to deal with a data in abstract terms rather than in physical terms to provide data independence.

The relational model which was proposed by E. F. Codd is represented by the mathematical set-theoretic relation. This model offers higher data independency.

The relational model is often realized as the DBMS which has SQL language as user interfaces.

This survey explores the aim of relational model and what is achieved in its implementation for real applications, when considering Codd's criteria of RDMS.

This paper also refers to the integrity constraint of relational database.

1. はじめに

計算機システムにおける生産性向上の基盤を与えるものとして，また新しいデータベース技術としてリレーショナル・データベースに世の関心が集まってから久しい．最近やっと大規模なデータベース・システムにも適合するリレーショナル・データベース管理システムが利用できるようになってきた．

本稿は，リレーショナル・モデルの提案者であるCoddの論文を追跡することにより，その目指しているところを再確認し，理論の実現であるリレーショナル・データベース管理システム，およびその利用者インタフェースとしてのSQL言語について実現度をレビューする．とくにデータ・モデル化において重要な整合性制約について言及する．

2. リレーショナル・モデル

データベースとは多くの人々から共同利用され，また異なる目的のために利用されることを前提として蓄積されているデータの集まりのことをいう．データベース管理システムは，データを利用者が必要とする形にとりまとめて提供する基本ソフトウェアであり，個々のプログラムを他のプログラムの変更やデータの蓄積環境の変更から切り離すことを目指している．

このことをデータ独立性 (data independency) という．データ独立性を高めることにより，システムのライフ・サイクルを長くし，開発や改訂，保守に要する費用を低減できる．

Codd は，1960 年代終りのデータベース管理システムの状況を振り返り，まだデータ独立性の達成が十分でないと，新しいデータ・モデルを提案する論文を 1970 年に発表した[1]．この論文において，Codd はさらに達成すべきデータ独立性の課題として，次のものを指摘している．

1) データの物理的な配置や順序からの独立……ファイル中でレコードが並ぶ順序や，レコードの物理的な属性が変更されても，業務プログラムに及ぶ変更はないようにすべきである．
2) 索引や転置ファイルなどの性能向上のための手段からの独立……これらの手段は，利用者の目からは隠すべきである．これらを意識して作成された業務プログラムは，データベース・システムの性能向上のための影響を受け，その寿命を短かくする．
3) アクセス経路からの独立……物理的にデータをアクセスする時の経路を，業務プログラム中に表現しないようにすべきである．

このような課題の実現のために，現実世界のデータを把握するモデルとして，Codd は数学におけるリレーション（関係）を考えた．リレーショナル・モデルは，数学におけるリレーションの概念を，ファイルやデータベースに応用したものである．

一般に n 個の集合 $D_1, D_2, \cdots\cdots, D_n$（同じものがあってもよい）に対し，次のように定義される集合 R をリレーションという．

$$R=R(D_1, D_2, \cdots, D_n)\subset D_1\times D_2\times\cdots\times D_n$$
$$=\{(d_1, d_2, \cdots, d_n)|d_i\in D_i, i=1, \cdots, n\}$$

集合 $D_1, D_2, \cdots, D_n$ を定義域 (domain)，n をリレーション R の次数 (degree) という．リレーション R の要素 $(d_1, d_2, \cdots, d_n)$ を n 個組 (n-tuple) という．

リレーション R を構成する定義域 $D_1, D_2, \cdots, D_n$ には同じものがあってもよいので，リレーション R において一意に識別できるように名前 $A_1, A_2, \cdots, A_n$ をつけ，これを属性(attribute)と呼ぶ．$R(A_1, A_2, \cdots, A_n)$ をリレーション R のスキーマという．

リレーション R は集合なので，その要素である n 個組は一意に識別できることが必要である．属性(の集合)のうち，n 個組を一意に識別できるものを候補キー(candidate key) という．複数の候補キーが存在する場合，そのうちの任意の一つを取り出し主キー (primary key) と呼ぶ．リレーション R のある属性（の集合）が，他のリレーション S の主キーになっているとき，これを R の外部キー (foreign key) という．

リレーションは，概念表現として二次元の表形式で表すことができる．このため，リレーションはしばしば 表と呼ばれ，n 個組のことは行 (row)，属性のことは列 (column) と呼ばれる．

Codd は，1971 年にさらにリレーションの操作[2]とリレーションの正規化[3]を論じる二つの論文を発表し，データベースの理論的な基礎を与えた．以後 1970 年代を通じて活発な研究がなされることになる．Codd 自身，リレーショナル・モデルの研究の意義と目的を次のように述べている[4]．

1) 高度のデータ独立性を達成する．
2) コンピュータの専門家でない人から専門家に至るまで，誰もが共通して利用できるような簡単なデータの見方を提供する．
3) データ管理者の負荷を軽くする．
4) データベース・システムの理論的基礎を築く．
5) 事実検索のような人工知能分野とファイル管理とを結合し，将来の応用に備える．
6) データベースの業務プログラムを，現在の手続き的言語の水準から，集合を対象とする非手続き的言語の水準に押し上げる．

リレーショナル・モデルの提案に始まる一連の重要な業績に対し，Codd は，1981 年度 ACM チューリング賞を受賞する．この受賞講演において，彼はリレーショナル・データベースを生産性向上のための実用的基盤として特徴づけている[5]．

この講演において，生産性向上の課題を追求する一方で，リレーショナル・データベースとそうでないデータベース・システムとの間に，明確な一線を画すべき時期がきたことにも言及している．

Codd は，データ・モデルは単なるデータ構造ではないと言い，あらためて少なくとも三つの要素からなると確認している．

1) データ構造型の集まり（データベースを構築するための積み木）
2) 演算子あるいは推論規則の集まり……1)のデータ型のいかなる実現値 (instance) に対しても作用可能であって，1)の構造を任意に組み合わせたものの任意の部分から，データを検索し，導出し，変更できるものでなければならない．
3) 一般性をもった意味制約（整合性）規則の集まり……これは，データベースの整合性のある状態や，状態の変化を暗に陽に定義する．一般性をもつとは，こうした規則が，同じモデルに基づくいかなるデータベースにも通用することを意味する．

この意味で，リレーショナル・モデルはデータ・モデルであり，三者を兼ね備えた最初のものであると Codd は主張している．

構造部分 (structural part) は，定義域，任意の次数のリレーション（その概念表現が表である），属性，n 個組，候補キー，主キーからなる．操作部分(manipulative part)は，選択，射影，結合などの代数的な演算子からなる．演算子はリレーション(表)を別のリレーション（表）に変換する．選択は，一つのリレーション（表）を作用対象とし，そこからいくつかの n 個組（行）を抜き出して新しいリレーション（表）を作る．射影は，一つのリレーション（表）からいくつかの属性を抜き出して新しいリ

レーション（表）を作る．結合は，二つのリレーション（表）を作用対象とし，互いのリレーション(表)の属性に対し指定された条件により，結合したリレーション(表)を作る．整合性部分(integrity part)は，二つの意味的制約からなる．主体整合性(entity integrity）と参照整合性（referential integrity）である．整合性については，後章で述べる．

Coddは，リレーショナル•データベース管理システムと呼べるものは，上記三つを備えているものであるとし，単にデータが表という形式で扱えるシステムと区別している．高いデータ独立性を実現し，生産性向上に寄与するシステムは，そのようなリレーショナル・データベース管理システムというわけである．

3. リレーショナル・データベース管理システムの実装

リレーショナル・モデルは，第2章で概観したようにデータベースの物理的な編成法や実働化上の技法の詳細を一切利用者の目から隠すことによって，高いデータ独立性を達成している．この特徴により生産性向上に寄与するシステムとして期待されるわけであるが，この特徴を維持しつつ大規模なデータベースを能率よく操作できるシステムを実現できるかどうかが課題であった．

リレーショナル•モデルの理論研究が活発になるとともに，システムの実装(Implementation)の実験がいくつか行われた．そのような実装化の実験の中で，一般のデータベース管理システムの基本機能を一通り取り揃えた本格的なリレーショナル・データベース管理システムと呼べるものは，1970年代後半におけるIBM San Jose研究所におけるSystem Rである[6]．System Rの利用者インタフェースとして採用されたのがSQLと呼ばれる言語である[7]．

Coddは，リレーションの集まりからなるデータベースを操作して目的の情報を入手するための演算として，次の二つの手法を提案した．

1) 関係代数（relational algebra）
2) 関係論理（relational calculus）

関係代数では，データベースへの問い合わせに対する操作過程を，リレーションの集まりから新しいリレーションを作り出す集合演算としてとらえる．関係論理では，データベースへの問い合わせを，リレーションの内包的記法を使った命題（論理式）として表現する．

関係論理は，一階述語論理にそって構成されているので数学的な見通しがよく，データベースへの問い合わせを基本的に十分表現できる．そこで，関係論理と同等またはそれ以上の表現能力をもつ問い合わせ言語を関係完備（relational complete）といい，言語の表現能力の基準としている．Coddは，汎用と称する問い合わせ言語は，少なくとも関係完備であるべきであると主張している．関係代数は関係完備であることが証明されている[8]．

関係代数や関係論理は，データベースへの問い合わせの表現能力の高さや数学的な見通しの良さという長所を持つが，一般の利用者にとっては，理解しにくいという短所を持つ．この長所を生かし短所をなくす方法の一つに写像型がある．一般に二項関係は二つの属性の間の写像を定めていると解釈できる．これをリレーションである n

項関係に拡張して，問い合わせに答を出す過程を一連の写像としてとらえるのが写像型言語である．SQL は，写像型言語の一つである．

SQL の基本構造は，SELECT，FROM，WHERE の三つの節から構成されている．SELECT 節は，問い合わせの結果であるリレーションを指定する．FROM 節で問い合わせの対象となるリレーションを指定する．WHERE 節は，問い合わせの条件を指定する．SQL の全般的な解説は，他書に優れたものがあるので，ここでは深く立ち入らない．

1980 年代に入って，リレーショナル・データベース管理システムが数多く商品として提供され始めたが，利用者インタフェースとしては SQL をベースにするものが多い．種々の SQL の氾濫の中で，リレーショナル・データベース言語の標準化が検討され，1987 年末までには ISO 規格 SQL と JIS 規格 SQL が制定されようとしている．ここに，Codd が提案したリレーショナル・モデルは，SQL を利用者インタフェースとするリレーショナル・データベース管理システムとして具現化することになる．しかし，リレーショナル・データベースの実装が商品化されるにあたって，実務世界からの要求あるいは実装上の技術的な問題に関する妥協などにより，具現化したリレーショナル・データベース管理システムは，必ずしも Codd のいう意味での生産性向上の基盤を与えるものにはなっていない．

リレーショナル・データベース管理システムを，本来意図していた方向に導くために，Codd は 1985 年秋の Computerworld 紙に，データベース管理システムが，真のリレーショナル・データベース管理システムであるかどうかを判別するための 12 の規則を発表した[10]．この 12 の規則に対しては，良いデータベース管理システムを設計する上で有効な規則であるという賛成論と，これらの規則を満たすデータベース管理システムは，重たいものになり実用に耐えられないという反対論が出されたが，現在では，システムの設計のガイド・ラインとして有用であるという見方になってきている．

12 の規則とは，次のものである．

1) 情報の規則……すべての情報は，表の中の値によって論理的に一つの方法で表現される．
2) 保証されたアクセスの規則……どのデータも表名，列名およびキー値で得られる．
3) ナル値の系統的な扱い……情報の欠落を表現するための値が扱える．
4) リレーショナル・モデルに基づくアクティブ・オンライン・カタログ……アクティブ・オンライン・カタログもリレーショナル・モデルで表現される．
5) 総合的データ準言語の規則……複数の操作言語によって広範に支援される．
6) 視野（ビュー）の更新……システムで更新可能なデータは，視野（ビュー）を通しても更新可能である．
7) 高水準でのデータ挿入・更新・削除……一つのデータ操作指令で，複数のデータの更新が可能である．
8) 物理的データ独立性……物理的データとアプリケーションの独立が計られている．
9) 論理的データ独立性……論理的データとアプリケーションの独立が計られて

いる．

10）整合性独立性……整合性（integrity）とアプリケーションの独立が計られている．

11）分散独立性……分散するデータとアプリケーションの独立が計られている．

12）非破壊の規則……低水準の言語が，高水準の言語にとって代わってもデータベースが損なわれることはない．

この12の規則に照らし合わせ，既存のリレーショナル・データベース管理システムを評点する方法も公開されており[11]，実装のデータベース管理システムを評価している．この評価で，満足いく高い点数を得ている商品は数少ない．欠点になっている規則のうち一つは，整合性独立性である．

SQL言語の規格制定の審議過程では，時代遅れにならないように標準を早く作成することが求められた．このため，規格化される言語においては，すでにいくつかの問題点や拡張案が指摘されている[12]．ISOではSQL言語の標準化[13]を早期に実現することと，機能強化に応えるために，現在，補遺1の制定を急いでいる．

SQL補遺1[14]で求められている新しい機能の一つは，整合性拡張機能（integrity enhancement feature）である．リレーショナル・モデルにおいては，第2章で述べたように，データ対象として表と行と列をもち，関係演算ができ，整合性制約（integrity constraints）として参照整合性と主体整合性を満たすことが要請される．

データ・モデルの表現能力という面からも，データベースの整合性を維持するという面からも重要な機能でありながら，実装上の困難さ，効率に及ぼす影響などにより，既存のデータベース管理システムでは，実現していないものが多い整合性制約について次章で述べる．

4. 整合性制約

データベース中のデータは，時間の経過とともに内容が変更されていく．このデータの変更を利用者に自由に許しておくと，データの整合性が損なわれる場合が発生する．このため，データの整合性を維持していく機能をデータベース管理システムは，持っていなければならない．

4.1 データの整合性

主なデータの整合性には，次のものがある．

1）値の整合性……たとえば，従業員の年令が18才から60才までの範囲である時に，14才という値は誤りである．

2）行内の整合性……たとえば，従業員の入社年が5年前の人に対し，その勤務年数が10年という値は誤りである．

3）主体整合性（主キー条件）……リレーションRに行を追加するとき，その行の主キー値と同じ値の行がすでに存在している場合には，この行の追加は無効である．主キーに対する整合性は，現実世界における主体（エンティティ）とデータベース中のデータとの整合性を維持しようとするための条件である．

4）参照整合性……あるリレーションの属性が，他のリレーションの属性を参照することがしばしばある．たとえば，従業員というリレーションの所属部署という

属性は，部署というリレーションの部署名という属性を参照している．この場合，従業員のある行を追加しようとするとき，その行の所属部署という属性の値は，部署というリレーションのどれかの行の部署名の値に存在しなければ，この追加は無効となる．このような制約をもつ属性のことを外部キーという（第2章参照）．

外部キーの参照先である主キーを含む行を変更あるいは削除したとき，外部キーをどのように処理するかということは問題である．現実世界の状況に応じて次の三つの場合が考えられる．

1) 波及 (cascade) ……外部キーの参照先である主キーを含む行が変更あるいは削除されたとき，外部キーも合わせて変更・削除する．
2) 制限 (restrict) ……変更・削除の対象となる主キーを含む行を参照している行がない時に限り，変更・削除を許す．
3) ナル値 (nullify) ……外部キーの値をナル値にした後，主キーを含む行を変更・削除する．

参照整合性を宣言するためには，次の指定が必要となる．

1) 外部キーを構成する列名（の並び）
2) 外部キーの参照先である表の名前
3) ナル値を許すか否か
4) 変更・削除のときの扱い

SQL 補遺1に含められている整合性拡張機能は，次の通りである．

1) 主体整合性の制約（主キー）と，満たされなければならない表間の参照整合性の制約（外部キー）
2) 表の行に適用される検査条件（CHECK 条件）
3) 表の中に行が挿入されるとき，列の値を指定しないときにとられる値の指定（DEFAULT 句）

補遺1で導入される参照制約は，指定された制約が満たされない場合には，エラーとすることのみを規定している．SQL 言語のより大きな改良・拡張とされる SQL 2[15] では，エラーとするだけではなく，このときに行うべき有効な動作を追加しようとしている．波及や制限，ナル値化などがそれである．

4.2 リレーショナル・データベース・システムの表現力（参照整合性を中心とした）

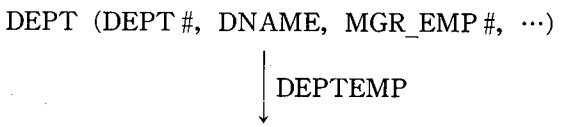
以下 C. J. Date の指摘[16]をもとに，参照整合性を中心に，リレーショナル・データ・モデルの表現能力について誤解されやすいことについて述べる．

1) リレーショナル・データベースでは，データ整合性を提供していない……この指摘が誤りであることは，上記整合性機能の説明から明らかである．しかし，残念ながら既存のリレーショナル・データベース管理システムにおいては，Codd の12の規則による評点の報告にもみられるように，多くのシステムに対してこの指摘は当てはまる．

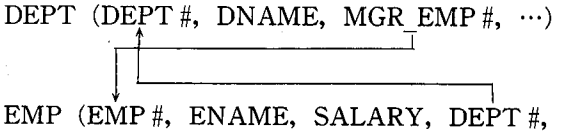
外部キーによる参照整合性の機能は，次のような誤解を生み出しかねない．

2) データ整合性をもつリレーショナル・データベース・システムは，ネットワーク型データベース・システムと同じである……この誤解は，データ構造図と参照図の混乱から生じている．

次の例を考えてみよう．



（a）データ構造図



ここで，DEPT：部署，DEPT#：部署番号，DNAME：部署名，MGR_EMP#：マネージャの従業員番号，EMP：従業員，EMP#：従業員番号，ENAME：従業員名，SALARY：給与

（b）データ参照図

上図で（a）は，ネットワーク型データベースにおける論理データ構造を示している．このデータベースを操作する使用者は，二つのレコードDEPTとEMPおよびそれらを結びつけているリンクDEPTEMPセットを陽に扱うことになる．

上図（b）では，リレーショナル・データベースにおける参照を図示したものである．このデータベースを操作する使用者は，二つの表DEPTとEMPおよびそれらの間の参照関係を陽に扱うことになる．参照関係は，陽には外部キーと主キーにより表現される．

上図（a）と（b）は，極めて似たものであるが，その解釈はまったく異なるものである．

リレーショナル・データベースの利用者は，二つの表のみを操作するだけであり，参照関係を操作する必要はない．これに対し，ネットワーク型データベースの利用者は，二つのレコードの操作以外にDEPTEMPセットを操作するための指令（CONNECT，DISCONNECT）を必要とする．

リレーショナル・データベースにおける参照整合性の働きは，ネットワーク型データベースにおけるセットの働きと非常に似ている．しかし，セットは，参照整合性の機能ばかりでなく，アクセス経路を指定する働きをも持っており，新しいアクセス経路の指定なしに，新しい参照整合性を追加することができない．

セットで表現される参照整合性に比べ，外部キーにより表現される参照整合性の方が柔軟性があり，またデータ操作指令にも余分なものがいらないといえよう．

ネットワーク型データベースにおけるデータ構造図が表現する二つのレコード間の論理関係は，リレーショナル・データベースではどう表現されるだろうか．

3）　ネットワーク型データベースにおけるセットによる親子関係の表現は，リレーショナル・データベースにおいては，結合操作により表現される……この指摘は正しいが，結合操作はセットによる表現以上のデータの表現を可能にしている．

先の例を再度みてみよう．

DEPT（DEPT#, DNAME, MGR_EMP#, BUDGET）

EMP（EMP#，ENAME，DEPT#，SALARY）

BUDGET：予算

EMP表にDEPT#という属性をもつことにより，DEPTとEMP間の関係が表現されており，EMPのDEPT#とDEPTのDEPT#を結合することにより，DEPTとEMP間のセットにより表現されるデータを取り扱うことができる．

データベース中に格納されているデータは，陽に表現される関係以外に，陰な関係をもつ．たとえば，

WORK_IN：ある従業員は，ある部署で働いている．

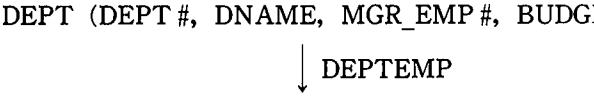
WELL_ PAID：ある従業員は，ある部署の全予算（BUDGET）以上の給料をもらっている．

MANAGES：ある従業員は，他のある従業員の上司である．

このような陰な関係に対し，陽にそれを表現するレコードを設計することができる．しかし，そのような方法は，データの冗長性を持ち込むことになる．このような関係はSQL構文により，次のように陽に表現することができる．

```
WORK_IN : SELECT EMP. *, DEPT. *
          FROM EMP, DEPT
          WHERE EMP. DEPT#=DEPT. DEPT#;
WELL_PAID : SELECT EMP. *, DEPT. *
            FROM EMP, DEPT
            WHERE EMP. SALARY>DEPT. BUDGET
              AND DEPT. DEPT#=EMP. DEPT#;
MANAGES : SELECT MGR. *, EMP. *
          FROM EMP MGR, DEPT, EMP BUKA
          WHERE MGR. EMP#=DEPT. MGR_EMP#
            AND DEPT. DEPT#=BUKA DEPT#;
```

この同じ問題に対し，ネットワーク型データベースにおいては，典型的に次のようなデータ構造が定義されるだろう．



このモデルに対し，先の三つの陰な関係に対し，そのような関係をすべて陽に表現することはできないことは明らかであろう．そのような関係を陽に得るためには，データ構造を注意深くなぞるプログラムを作成する必要がある．

セットで表現される論理関係以上に，リレーショナル・データベースにおける結合操作は，柔軟な論理関係を操作として表現することができる．

リレーショナル・モデルの方が，データ構造に柔軟性があるにしても，一般にデータは自然に階層構造をなしているので，ネットワーク型データベースの方が適しているという意見もみられる．

4) データは自然に階層構造をなす……この指摘は，あるデータに対しては正しい．

たとえば，前述のDEPT/EMPデータベースにおいては，自然に階層構造をなしている．しかし，それゆえに階層構造が良いデータ構造だというのは，必ずしも正しくない．

DEPT/EMPデータベースにおいては，自然にDEPTを親としEMPを子とする階層関係がある．これをそのまま親子関係のデータ構造で表現した場合に，次の二つの問い合わせを考えてみよう．

(Q1)　指定した部署の従業員を読み出す．

(Q2)　指定した従業員の部所を読み出す．

与えられたデータ構造は，Q1の問い合わせに対しては，うまく適合するが，Q2の問い合わせに対してはうまく適合しない．同じデータベースをリレーショナル・データベースで表現した場合，各々の問い合わせは次のように対称性をもっている．

```
Q1：SELECT  *
    FROM  EMP
    WHERE  DEPT#  IN
          (SELECT  DEPT#
           FROM    DEPT
           WHERE  DNAME=与値)；
Q2：SELECT  *
    FROM  DEPT
    WHERE  DEPT#  IN
          (SELECT  DEPT#
           FROM  EMP
           WHERE  ENAME=与値)；
```

自然に階層構造になるデータを，階層構造として定義しない方が，良い設計になる例である．

以上の簡単な例からみられるように，リレーショナル・データベースの方が，ネットワーク型データベースよりも柔軟性やデータの表現能力が高いといえよう．これがいえるのは，高水準の集合操作がデータ操作指令として採用されることと，データ整合性の機能，とくに参照整合性の機能があるからである．

5. おわりに

リレーショナル・モデルの特徴は簡潔さにある．データ構造はリレーションと呼ばれる二次元の表であり，データ操作は，リレーションを操作する集合演算を基礎としている．すなわち，利用者に対し視覚的な高水準インタフェースを提供している．このことにより，エンド・ユーザから専門のプログラマに到るまで，共通のインタフェースを提供しており，データベース利用の生産性向上と利用者の拡大に寄与している．

リレーショナル・モデルは，SQLと呼ばれる（英語の）自然言語に似た言語をインタフェースとするデータベース管理システムとして多く実現されている．リレーショナル・モデルは，データの論理構造についてのみしか言及しておらず，データベース

の実現において必要な物理構造については，実装者の定義に委ねられている．実装においては，記憶域の効率的使用や，処理の高速性の実現のため，多くの工夫が必要とされてくる．それらの工夫は，結果としてしばしば利用者インタフェースに対し特殊な技巧や，物理的概念を操作する機能を持ち込むことになり，理論的な良さをしばしば打ちこわすことになりかねない．

理論と現実の折り合いが問題であり，そこに規則を設け本来の方向を見失わないようにしようというのが，Codd の 12 の規則といえよう．

リレーショナル・データベースは，現実世界をリレーションというデータの型で記憶しようとするものである．現実世界のデータのモデル化においては，データの形式的な面ばかりでなく，データの意味をいかに反映するかも重要である．このための基本概念が主キーであり，それを基にした正規化の理論および整合性の規則である．なお，正規化の理論については他書を参照されたい[9]．

今日，整合性の規則は，実装面においては軽視されており，また理解面においては，ネットワーク型のデータベースにおける論理的なデータ間の関係との誤解も一部にある．ISO の SQL 規格においても，整合性の機能は，補遺 1 としてしか扱われていない．この機能の重要性は衆知の事実であり，今後急速に整理され実現されていくだろう．

リレーショナル・データベース管理システムは，理論と実装の関係が密着した数少ない分野の一つである．高性能化するハードウェア技術に支えられ，使いやすい利用者インタフェースを提供するものとして，リレーショナル・データベース管理システムは，急速にその利用分野を拡大していくと思われる．

本稿では，リレーショナル・モデルの目指した理論的な目標と，それに対する現在の実装との関係をレビューし，とくに実装上重要な課題とされる整合性制約について言及した．

リレーショナル・データベース管理システムの開発および，利用に携わる方々の参考になれば幸いである．

参考文献

[1] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", CACM 13 No. 6, June 1970.

[2] E. F. Codd, "A Data Base Sublanguage Founded on the Relational Calculus", Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control, ACM, Nov. 1971.

[3] E. F. Codd, "Further Normalization of the Data Base Relational Model", In Data Base Systems, Courant Computer Science Symposium Series vol 6, Printice-Hall, 1972.

[4] E. F. Codd, "Recent Investigations in Relational Data Base Systems", IFIP 74, 1974, pp. 1017～1021.

[5] E. F. Codd, "Relational Database: A Practical Foundation for Productivity", CACM vol. 25, No. 2, 1982, pp. 109～117.

[6] M. M. Astraham, et al, "System R: Relational Approach to Database Management", ACM TODS vol 1 No. 2, 1976, pp. 97～137.

[7] D. D. Chamberlin, et. al, "SEQUEL: A Structured English Query Language", Proc. 1974 ACM SIGMOD Workshop on Data Description and Access and Control, May 1974.

[8] E. F. Codd, "Relational Completeness of Data Base Sublanguages", In Data Base

Systems, Courant Computer Science Symposium Series, vol 6, Printice-Hall, 1972.
[9] C. J. Date, "An Introduction to Database Systems", vol I, 4th edition, Addison-Wesley, 1985.
[10] E. F. Codd, "Is Your DBMS Really Relational?", Computerworld, Oct. 14, 1985.
[11] E. F. Codd, "Does Your DBMS Run by the Rules?", Computerworld, Oct. 21, 1985.
[12] C. J. Date, "A Critique of the SQL Database Languages", ACM SIGMOD Record 14 No. 3, Nov. 1984.
[13] ISO SC21 WG3 N291, Final Draft 9075-1987, Database Language SQL, Nov. 1986.
[14] ISO SC21 WG3, Proposed Draft Addendum 1 to DIS 9075 Database Language SQL, 1987.
[15] ISO SC21 WG3 N325, Database Language SQL2, Apr. 1987.
[16] C. J. Date, "Some Relational Myths Exploded", Info IMS 4 No. 2, 1984.

執筆者紹介　原　　潔（Kiyoshi Hara）

昭和44年京都大学理学部数学科卒業．45年日本ユニバック(株)入社．主としてUNIVACシリーズ1100のデータベース管理システムの開発，保守業務に従事．現在，商品開発本部ソフトウェア二部データベース開発室に所属．情報処理学会員・電子情報通信学会員，ISO/SC 21/WG 3委員．

論文

ネットワーク型データベース（DMS1100）から リレーショナル・データベース（RDMS1100）への変換

Translation from DMS1100 Database to RDMS1100 Database

大内　忍

要　約　1980年代に入って，大規模なデータベースを対象とした本格的なリレーショナル・データベース管理システムが実用され始めた．日本ユニバックでも，UNIVACシリーズ1100およびUNIVACシリーズ2200上のリレーショナル・データベース管理システムとしてUDS RDMS 1100を提供している．

リレーショナル・データベースは，正規化されたデータを対象とし，その操作指令は集合演算を用いており，従来のネットワーク型データベースと大きく異なっている．このため，ネットワーク型データベース管理システムであるDMS 1100の十余年に及ぶ使用体験の中で培われてきたデータベース設計の技法や，プログラミング技法がRDMSにも適用できるのか，あるいはRDMSを効果的に利用するためには新しい技能や技術が必要なのか等，未知の部分が多く存在している．

本稿は，既知のDMSプログラムをできるだけ機械的にRDMSプログラムに変換する試みの中で体験した，いくつかの技術について報告する．

Abstract　Recently relational database management systems for a large volume of data begin to be used in real business applications. UDS RDMS 1100 for UNIVAC Series 1100/2200 is a typical of them.

In relational database system, data is represented as normalized table, and its data manipulation is based on set-theoretical operations. It is an open question whether the know-how such as database design and programming methodology for network database is also applicable for relational one or not.

This paper reports the experience of the semi-automatic conversion of network database (DMS program) to relational database (RDMS program).

1. はじめに

1970年代のデータベース・システム技術の進展には，データベース・システム言語協議会（CODASYL）によるデータベース用共通言語体系の完成と普及という，大きな原動力があった．DMS 1100（以下DMSと略する）は，その流れを汲むデータベース管理システム（DBMS）であり，十余年に及ぶ使用体験の中で今やユーザ・システムにしっかり定着し，その使用上の技能や技術も多く蓄積されてきている．1970年代のデータベース時代をもたらした，もう一つのきっかけとして忘れてはならないのが，E. F. Coddが提案したデータのリレーショナル・モデルであろう．しかし，リレーショナル・データベース管理システムは，実現上のいくつかの技術的な問題のために商用化されてくるのは，1980年代になってからになる．

UDS RDMS 1100（以下RDMSと略する）は，シリーズ1100上に実現したリレーショナルDBMSである．RDMSは大規模なデータベースを処理対象とし，トランザ

クション処理にも適用できるよう設計されており，本格的なリレーショナル DBMS といえる．

リレーショナル・モデルは，正規化されたデータを対象としており，その操作指令は集合を対象としている．このため，DMS で蓄積してきたデータベース設計やプログラミングの技法が RDMS にも適用できるものなのか，あるいは RDMS を効果的に利用するための技能や技術に関し新しいものが必要なのか，等に未知の部分が多く存在している．

本稿においては，既存の DMS プログラムを，できるだけ機械的に RDMS プログラムとして変換し実行するという作業の中で得られた，いくつかの技術について報告している．

第 2 章で，簡単にリレーショナル・データベース・システムを概観し，第 3 章で DMS から RDMS への変換の問題について個別的に取り上げ報告する．第 4 章は，RDMS プログラム作成上の補遺的なことがらについて述べる．

2. リレーショナル・データベース・システム

本章では，リレーショナル・データベース・システムの基本的な考え方や，その操作およびシステム構成について概観する．ここで言及する内容は，次章で必要とするものに限られている．

リレーショナル・データベースの考え方は，1970 年に E. F. Codd によって提案された[1]．データベースの利用者が一般に広く増大することを予測し，データ構造は内容表現を使用者の目から隠し，表形式をとっている．データの操作も，データの記憶構造に関係しない上位水準での指令を採用している．また，データベース中のデータが互いに矛盾することがないように，一貫性の保持の主張が組み込まれている．これらの提案を，集合論における関係理論を基礎に行っている．表形式としてのデータ構造と関係代数に基づいたデータ操作，一貫性制約の三つの要件を満たすものをリレーショナル・モデルと呼び，リレーショナル・モデルに基づいたシステムのことをリレーショナル・データベース・システムと呼ぶ（図 1）．

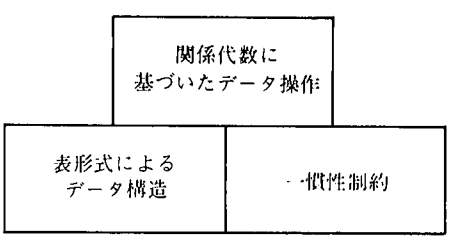


図 1　リレーショナル・モデル

Fig. 1　Relational model

2.1 データ（リレーション）

データ表現の変更が，業務プログラムに及ぼす影響が小さいことを指して，データ独立性が高いという．データベース・システムの目的の一つは，このデータ独立性を高めることにある．データ独立性を高めることは，データ表現の論理的側面と物理的

側面を明確に区分けしていくことにより実現される．これを達成するために，現実世界をデータとして把握するモデルに，集合論における関係 (relation) の概念を応用したのがリレーショナル・モデルである．

一般に，n 個の集合D_1, D_2, ……, D_n の各要素 d_1, d_2, ……, d_n の順序づけられた組 (d_1,d_2, ……, d_n) の全体からなる集合 $D_1 \otimes D_2 \otimes \cdots\cdots D_n$ の部分集合 R のことをリレーション（関係）という．

リレーショナル・データベースは，従来データと呼んできたものをリレーションとしてとらえようとするものである．リレーションは，二次元の表とみなすことができる．表という概念は一般的に良く利用されているもので，親しみやすく理解しやすいので，表の形でリレーションは表現される．以後は，リレーションの代わりに表という用語を使う．リレーションを表で表現するために，組のことを行と呼び，各行を構成する要素のことを列と呼び換える．リレーションを表で表現した例を図2に示す．

表名(リレーション名)

社員表

列名

社員番号	氏　名	年令	家族	所属名
5122	竹内　敦	23	3	経理部
2540	戸田成美	30	4	システム部
1152	西村正彦	42	2	総務部
4801	原田政明	25	5	営業部
1020	稲田博行	45	4	経理部
⋮	⋮	⋮	⋮	⋮

← 行(組)

図2　表によるデータ表現

Fig. 2　Data representation by table

表を構成する各列の値が，それ以上分解できないとき，その表は正規化されている，あるいは第一正規形であるという．正規化されていない表は，複数の表に分解することにより簡単に正規化することができる．正規化された表だけを対象にすることにより，その取り扱いが簡単になるので，リレーショナル・データベースでは，いつも正規化された表だけを考える．

正規化された表だけを対象とするために，従来のファイルで扱ってきた繰り返し項目，配列，集団項目などのデータ構造は直接には扱えない．このことについては，第3章でさらに詳細に述べる．

リレーショナル・データベースにおける表は，集合論を基礎に定義されている概念であり，一般に表と呼ばれているものとまったく同じではない．いわば規範化された表 (disciplined table) である．

リレーショナル・データベースにおける表は，次のような性質をもつ．

1)　リレーショナル・データベースは，複数の表の集まりである．したがって，データベース中の他の表と区別できるように一意の表名をもつ．また，一つの表中の列名も他の列名と区別できるように表中において一意の列名をもつ．すなわち，

表名と列名は必須であり，かつそれらは上記の意味で一意である必要がある．

2) 表の中には，まったく同じ内容の行が2行以上存在しない．すなわち行の重複は許されない．また，各行は一意に識別できなくてはならず，表の各行を識別する列が存在する．この列のことを主キーと呼ぶ．主キーの重複は許されない．これらは，表が集合であることからくる性質である．

3) 表の行は，どのような順序であってもかまわない．すなわち，行の表中における順序は無意味である．これは，従来からのファイルとは大きく異なる点で，やはり表が集合であることからくる性質である．

4) 各列には，一意の名前がつけられるので列の順序にも意味がない．

5) データの問い合わせは，一つ以上の表に対して自由に行うことができるが変更は一つの表に対して許される．このとき，値の変更は列単位で行えるが，追加・削除は行単位で行う．

表をファイルに，行をレコードに，列をフィールドに置き換えることにより，リレーショナル・データベースにおける表は，従来のファイルにうまく対応していることがわかる．上記に述べたいくつかの性質を押えていれば，表をファイルと置き換えて理解してもかまわない．図3に表とファイルの対応を示す．

規範化された表の取り扱いが，従来のファイルの取り扱いと，どう異なってくるかについては第3章で述べる．

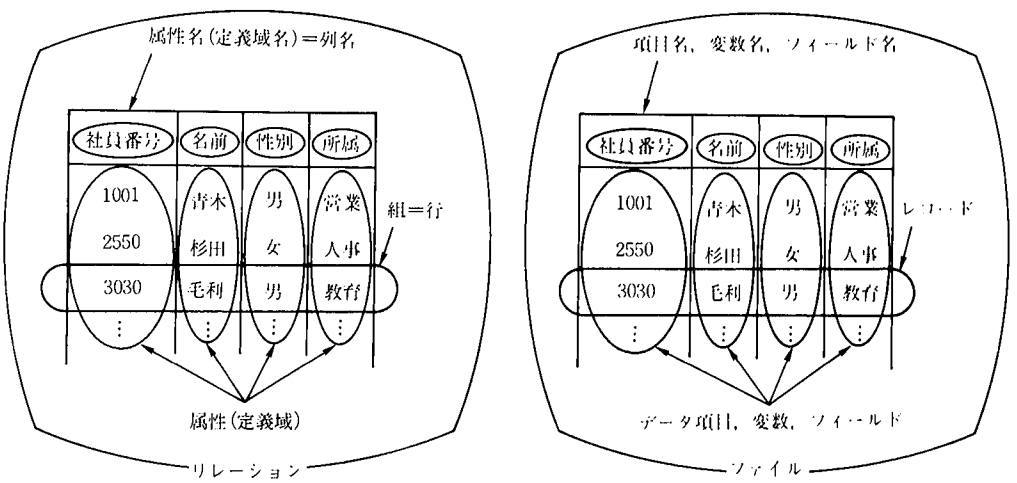


図3　表とファイルの対応

Fig. 3　Table vs file

2.2　データの一慣性

データベースは，多数の利用者から異なる目的で利用されることを前提としたデータの集まりである．したがって，データベース中のデータの信頼性の保持は重要である．

リレーショナル・データベースにおいては，表の中のデータが互いに矛盾することがないように2種類の一慣性制約（integrity constraint）を要請している．

一つは，実体の一貫性（entity integrity）と呼ばれるものである．表の各行を一意に識別するデータ項目の存在の要請である．この項目のことをキーという．表の各行に対し，複数のキー候補がある場合は，そのうちの一つを選び出し，主キー(primary key)とする．キーは，複数の列から構成されていてもよく，そのようなキーのことを複合キー（composite key）という．

他の一つは，参照の一貫性（referential integrity）と呼ばれるものである．ある表の列（あるいは列の集合）が，他の表のキーとして用いられているとき，この列（あるいは列の集合）のことを外部キー（foreign key）という．このとき，互いに一致していなければならない．

参照の一貫性は，ネットワーク型データベースにおけるセット関係と類似しているが，同一ではない．このことに関する議論は第3章で行う．一貫性の保持を図4に示す．

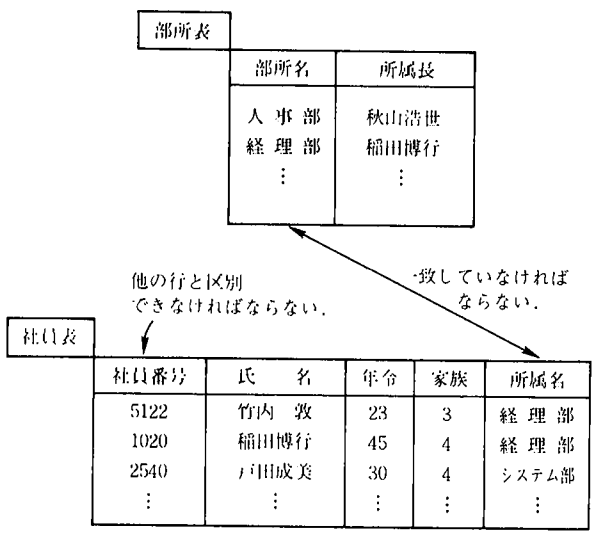


図4　一貫性の保持

Fig. 4　Integrity maintenance

2.3 データ操作

リレーショナル・データベースにおけるデータ操作は，集合操作を基本とした関係代数（relational algebra）を基礎にしている．

リレーショナル・データベースの問い合わせで，最も基本的な操作は，選択(restriction)，射影（projection），結合（join）の三つである．この他に，合併（union），共通（intersection），差（difference）などの伝統的な集合演算が利用される．これらの操作を図5，6に示す．

1)　選択とは，表の中から条件に適合する行を取り出すことである．図5の①に示すように，社員表から経理部に属する社員だけを取り出すのがそれである．

2)　射影とは，表から特定の列だけを取り出すことである．図5の②に示すように，社員表から氏名と家族の列を取り出すのがそれである．取り出した結果の行に重

複があれば，一つだけを取り出す．ただし実装（implementation）では，効率を考慮して重複除去の指示がある場合にのみそうする．

3) 結合とは，二つ以上の表から一つの表を作り出すことである．図5の③に示すように，社員表の所属名と部所表の所属名が同じものを取り出して一つの表にするのがそれである．

4) 合併，共通，差は，集合演算そのものなので，ここでは例を示すのみで説明を省く．

これらのデータ操作の結果は，また表であることに注意してほしい．このため，とくに合併や共通，差の操作においては，結果が表となるよう二つの表間の列同士はまったく同じ特性をもっている必要がある．

データベースの操作は，問い合わせばかりでなく変更の操作も必要である．変更の操作は，問い合わせの操作とは異なり，実在する一つの表に対してのみ可能である．

1) 挿入（insert）は，一つの表に対して行単位で挿入する．ただし，表中のどこに挿入するかを利用者が気にすることはない．

2) 削除（delete）は，一つの表に対して行単位で削除する．ただし，どの行を削除

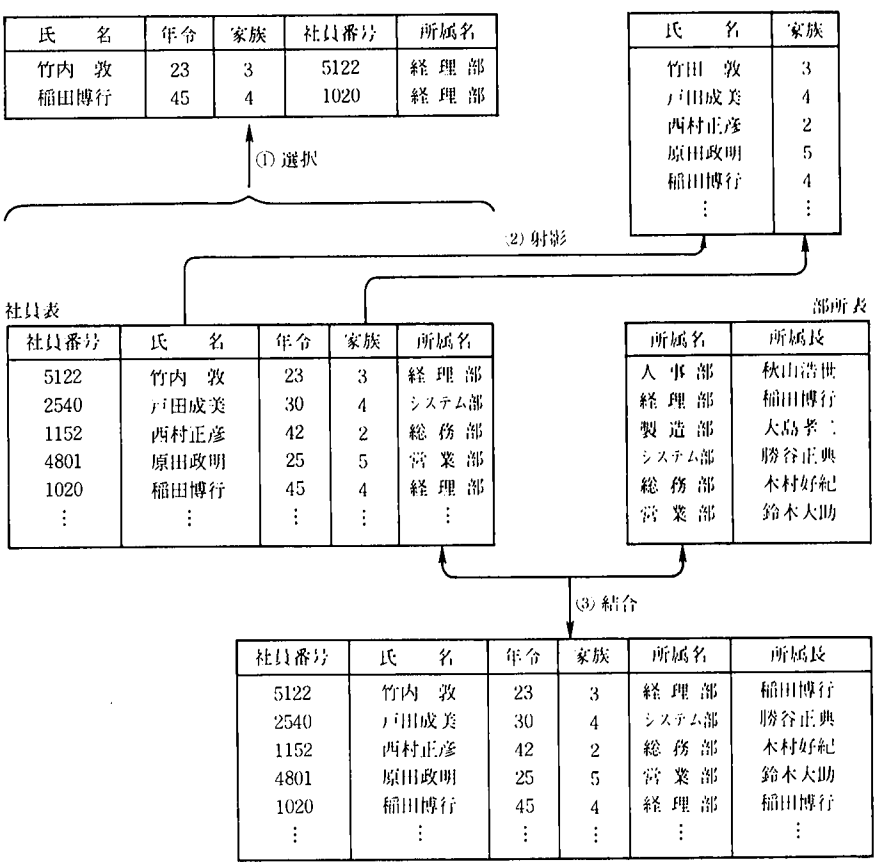


図5　三つの基本データ操作

Fig. 5　Three basic data operations

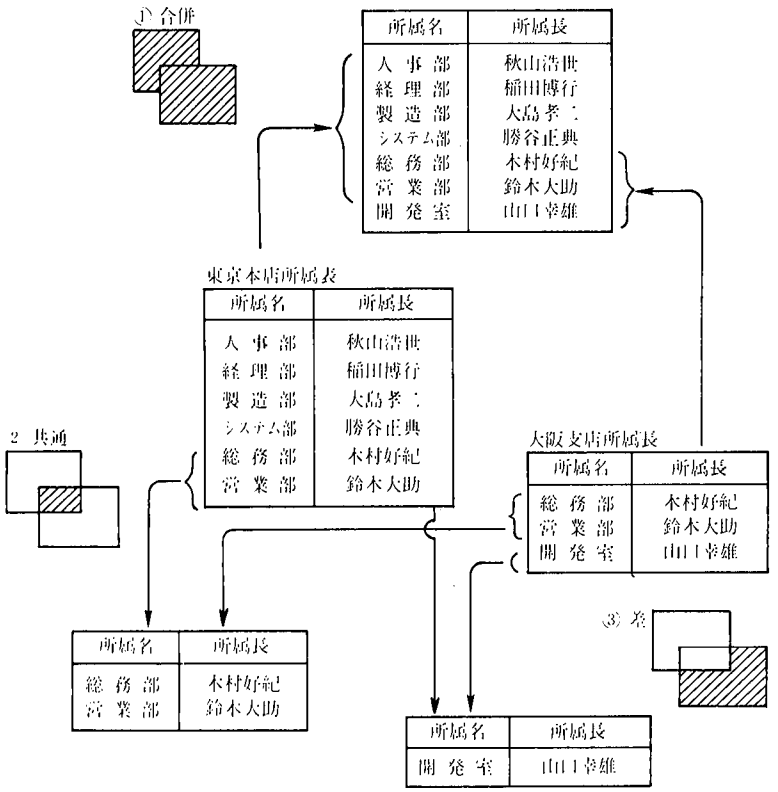


図6　その他のリレーション操作

Fig. 6　Other relational operations

するのかは条件で示されるので，1回の指示で複数の行を削除することがありえる．条件を指定しなければ，表の中のすべての行を削除することになる．

3)　更新（update）は，一つの表に対して列単位で更新する．どの行のどの列を更新するかは条件で与えられ，この場合も1回の指示で複数の行の複数の列を更新することがありえる．

データ操作は，いずれも「何をどのようにするか」だけを指示する．そこで指示するものは，表名，列名，データ値だけであり，このためリレーショナル・データベースにおけるデータ操作は，水準の高いものになっており，データ独立性が高く，理解しやすく，使いやすいものになっている．

集合を対象としたデータ操作であること，物理的な属性に関与していないことにより，いくつかの点で従来のファイル処理とは，様相が異なっている．その点の問題について第3章で議論する．

2.4　リレーショナル・データベース管理システム

Coddの提案したリレーショナル・モデルでは，データの論理的側面だけを言及している．リレーショナル・データベースの実装においては，データの物理的側面や，データ処理的側面に対する考慮が必要であり，誰がいつ使用しても完全に処理できるような仕組みを実現しなければならない．そのような仕組みを提供するのが，リレーシ

ョナル・データベース管理システム（RDMS：Relational Database Management System）である．RDMS の機構は，本稿の範囲外なので言及しないが，参考のために日本ユニバックの提供するリレーショナル・データベース管理システム RDMS 1100 のシステム構成を図 7 に示す．

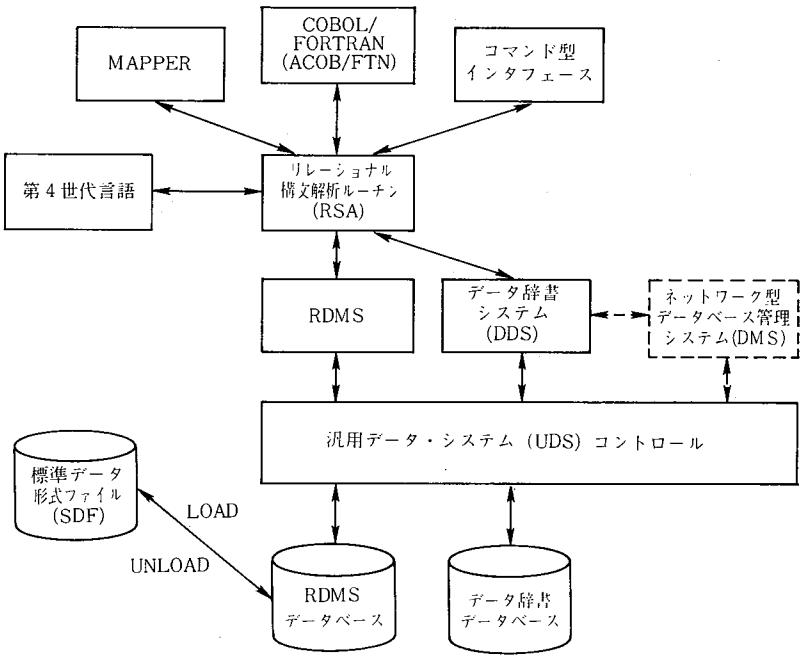


図 7　RDMS 1100 のシステム構成

Fig. 7　System structure of RDMS 1100

3. ネットワークからリレーショナル・データベースへの変換

リレーショナル・データベースにおけるデータ定義，データの一貫性およびデータ操作について，第 2 章で簡単に述べてきた．本章では，ネットワーク・データベースの世界で構築されているデータベース・システムを，リレーショナル・データベースの世界に変換することを目指して，両者の主要な相異点について明らかにし，変換の方法を取り上げる．

3.1 データの定義と一貫性制約

〔項目 1：データ項目には名前が必要である〕

データの操作は，表に付けられた名前，列につけられた名前，およびデータ値だけで指示できるようにすることにより，利用者に対するわかりやすさと，データ独立性を達成している．表名は全データベース中において，列名は関与する表中において一意でなければならない．

ネットワーク・データベースにおいては，名前を持たないデータ項目の存在が許されている．たとえば，FILLER 項目と呼ばれるものがそれである．

リレーショナル・データベースへ変換するためには，すべての項目に対し一意になる名前を付ける必要がある．名前に使用できる文字の集合に差異があるので，規則に

適合しない文字が使用されている場合には，別の文字に置き換えることも必要になる．リレーショナル・データベース・システムにおいては，－（ハイフン）は名前には使用できない．これは別の文字，たとえば下線（アンダスコア）に置き換える必要がある．

〔項目 2：集団項目は定義できない〕

リレーショナル・データベースでは，正規化されたデータのみを取り扱う．こうすることにより，利用者は，いつも表名や列名，データ値だけでデータ処理を指示でき，わかりやすくなっている．

ネットワーク・データベースにおいては集団項目の定義ができるが，リレーショナル・データベースへ変換するに当たっては，集団項目を排除するようにデータ構造を定義し直す必要がある．

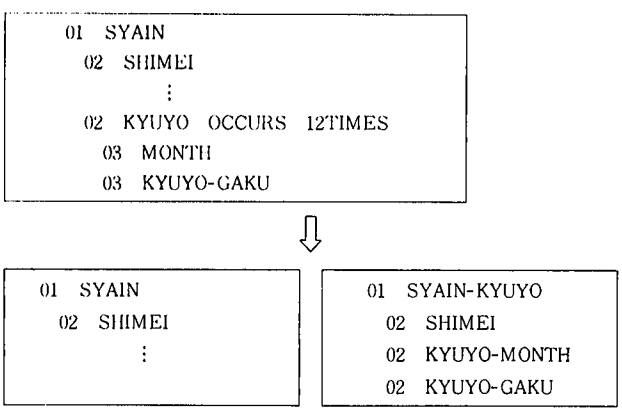
集団項目の排除は，データ処理に対しある不便さをもたらすことになるが，親言語のデータ領域に転送したのち，親言語のもつ集団項目の機能を利用すればよい．

〔項目 3：繰り返し項目は定義できない〕

項目 2 と同様にリレーショナル・データベースでは，正規化されたデータのみを取り扱うので，繰り返し項目の定義をすることができない．

理論的な面，あるいはデータモデルの簡明さからいえば，繰り返し項目の存在は不必要である．しかし，現実には，繰り返し項目は有効なデータ構造になる場合がある．このため，実際の RDMS の実装においては，それを許しているものもある．

繰り返し項目を含むレコードをリレーショナル・データベースに変換するには，そのレコードを複数の表に分割すればよい．分割の例を示す．



この分割は，結合すると元のデータを再現できるものでなければならない．この例では，SHIMEI で結合することにより，それが達成できる．

繰り返し項目全体を一つのデータ項目としてのみ定義しておき，個々の繰り返し項目を扱いたい場合には，親言語の繰り返し項目で定義された作業領域上に転送してから処理するという方法もある．RDMS 1100 では，文字型データの右側の残りの空白は，データベース上に格納するときには圧縮するので，この機能を使えば可変長項目も扱える．ただし，この方法をとるとデータ操作における条件式上，あるいは目標リ

スト上では個々の繰り返し項目の指定はできないことになる．データベース利用の将来性を考慮すると分割する方が望ましい．

〔項目 4：データを一意にするキー項目は必須である〕

リレーショナル・データベースでは，ファイルをリレーションすなわち集合として考えている．集合の要素は一意に識別される．このため表の要素となる各行（レコード）は，必ず行を一意にするキー項目を必要としている．しかも重複キーは許されない．

各行を一意に識別するキー項目があることにより，利用者はデータの物理的な属性，たとえばファイル上での順番あるいは位置などを知ることなく，どんなデータ操作も可能になっており，データ独立性を高めている．

これまでのファイルやデータベースにおいては，キーのないデータを取り扱っていたし，ときにはキーは重複してもかまわなかった．そのようなデータは，リレーショナル・データベースでは取り扱えなくなる．これは，伝統的な実務世界からみれば，欠点にみえるが，むしろデータの完全性，利用の便利さからみれば，そうすべきものととらえるべきである．

リレーショナル・データベースの変換に当たっては，キーのないデータや重複キーを持つデータがある場合には，必ず一意になるキーを見い出してデータベースを設計し直す．

〔項目 5：リレーショナル・データベースは表の集まりである〕

ネットワーク・データベースでは，現実世界の情報をレコード型と親子集合型（セット型）の二つの概念で表現している．これに対し，リレーショナル・データベースでは，すべてをリレーションすなわち表の概念のみで表現している．つまり，レコード間の関係をも表で表現することになる．

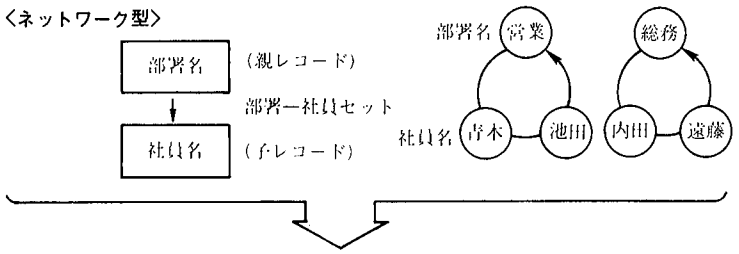
すべてを表という一つの形で表現できるという容易さは，逆からみるとネットワーク・データベースでは，親子集合型という明示的な手段によってわかりやすかった親子関係が，わかりづらくなるという欠点が出てくる．というのは，ネットワーク・データベースでいうところの親レコードも子レコードも，その親子関係を表すのも，みな同じ表という概念だけで表すからである．

ネットワーク・データベースでは，データ操作はデータ間の関係を設計段階で固定してしまうことを前提として指示される．リレーショナル・モデルの目標は，このようなデータ操作がもつ柔軟性のなさを排除することによって，よりデータ独立性の高いわかりやすいインタフェースを提供しようということにある．

ネットワーク・データベースをリレーショナル・データベースに変換するに当たって，レコードは素直に表に対応させていくが，親子関係をどう変換して表現するかは，その親子関係の目的を考慮して決定する必要がある．

ネットワーク・データベースにおける親子集合は，異なるいくつかのことがらを表現している．

1)　子レコードの親レコードに対する従属関係の表現
2)　子レコードの存在条件の規定
3)　レコードのアクセス経路の明示化



〈リレーショナル型〉

(方法1)

部コード	部署名	設置年	
1	営業	S 40	
2	総務	S 41	

親レコードに相当する表

社員コード	社員名	入社年	
003	青木	S 50	
005	池田	S 40	
007	内田	S 53	
009	遠藤	S 60	

子レコードに相当する表

部コード	社員コード	
1	003	
1	005	
2	007	
2	009	

セット関係を表している表
(たとえば，部コード1には，
003 と 005 が所属している)

(方法2) 子レコードに親レコードのキーを属性として含めてしまう．
この場合，部コードが外部キーとなる．

部コード	社員コード	社員名	入社年	
1	003	青木	S 50	
1	005	池田	S 40	
2	007	内田	S 50	
2	009	遠藤	S 60	

子レコードに相当する表

図8 セットの表現

Fig. 8 Representation of own-coupled set

上記の 1)に対しては，まず素直に親子集合を表現する表を導入することでよいだろう．2)に対しては，外部キーの設定を考慮する必要がある．3)に対しては，表定義としては不要であり，リレーショナル・データの物理定義における索引の作成や，データ操作指令における条件式での表現に置き換えればよい．

図8は，ネットワーク・データベースの親子集合をどのように表で表現するかの例を示している．

〔項目6：データの物理構造定義は独立している〕

表の定義は，あくまで論理的なデータ定義である．実際には，表は種々の属性をも

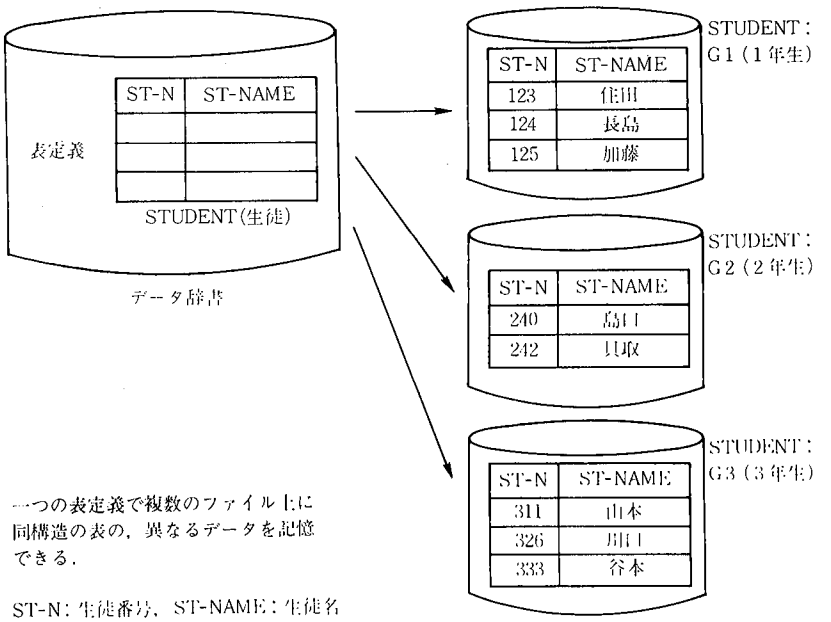


図9　RDMS 1100 における論理的な表と物理的な表

Table 9　Logical table and physical table in RDMS 1100

って二次記憶上に格納される．論理的な表定義と，表を格納する物理的なファイル定義とを対応させることが必要である．

大規模なデータベースにおいては，一つの論理的なファイルを複数の物理的なファイルに分割し格納できることが，運用・管理上必要となる．このために，RDMS 1100 では各表は，バージョンをもつことができるようになっている．ネットワーク・データベースにおける一つのレコード型に対し，複数の WITHIN 句が指定できる機能に対応している．

複数の WITHIN 句をもつレコードに対しては，表を格納する記憶領域に対するバージョンを使用する（図9）．

3.2 データの操作

〔項目1：データ操作の対象は集合である〕

表は，数学における集合の特殊な場合であるリレーションを表現している．リレーショナル・データベースにおけるデータ操作指令は，関係代数を基礎にしたものであり，集合操作である．データとその操作を論理的な局面で定義しているので，いくつかの制約がある．一つは，表と表の間の関連を表現するリンク構造や索引構造を利用者の目にみえる形にしてはならないということである．もう一つは，表における行の順番という形でデータベースの情報を表現してはならないということである．

ネットワーク・データベースにおいては，レコードのつなぎが明示的に利用者にみえており，それを利用したデータのアクセス・プログラムを作成することができる．また，レコードのファイルにおける順番を基礎にレコードをアクセスすることもできる．

このようなデータ処理プログラムは，リレーショナル・データベースをアクセスするプログラムには，直接的には変換できない．このためには，そのようなプログラムにおけるデータのアクセスのアルゴリズムを，集合演算により定義し直すことが必要である．ネットワーク・データベースにおけるつなぎなどによるレコード間の表現は，単にアクセス経路を示しているだけでなく，論理的な親子関係やレコードの存在条件なども含んでいるので，それを集合演算で置き換える方法は，それほど自明の方法ではない．

〔項目 2：データ操作の結果は集合である〕

リレーショナル・データベースに対するデータ操作の結果は，行の集合として得られる．親言語（COBOL や FORTRAN）においては，一時に一つのレコードを処理をする設計仕様になっているので，データ操作の結果としてレコード群が一時に返却されることは，著しくデータ操作と親言語の親和性を欠くことになる．

そこで，データ操作の結果を定義する機能と，データ操作の結果として得られた行の集合から 1 行ずつ親言語に引き渡す機能とが提供される．このために導入された概念がカーソルである（第 4 章参照）．カーソルを使用したプログラムの構造は，次のようになる．

1) カーソルを定義する
2) カーソルを開く
3) 一行づつデータを得る
4) カーソルを閉じる

キーによって一件のデータだけを検索するようなプログラムにおいては，既存のファイル処理プログラムに比べ，リレーショナル・データベースのプログラムは，処理命令数が多くなる．しかし，条件を満たす複数のデータを処理する場合は，後者は集合操作という水準の高い指令を用いるため，少ない命令数で実現できる．これは，開発や保守の生産性を向上させる基盤となる．

〔項目 3：集合操作なので大規模な入出力領域がいる〕

条件式の与え方によっては，一指令で大量のデータを更新することになる．また，表の削除などでは，大量のデータの更新が行われる．このため，処理効率や入出力バッファ領域の大きさの制約から，データ操作自体が正しく作動しない場合がある．

そのような場合には，リレーショナル・データベース管理システムの構成（入出力バッファ領域など）を最適化する必要があるが，それだけでは不十分でアプリケーション・プログラム側で工夫せざるを得ない場合もある．その時は，条件式の与え方を分割し，一指令で処理対象とするデータ量を小さい範囲に納めるなどを行う．

〔項目 4：明示的ロックは表単位である〕

利用者が，明示的に行えるロックは表単位のみである．ネットワーク・データベースにおいては，レコード単位に明示的にロックでき，きめ細かいレコード処理を同時実行環境下で行えるが，リレーショナル・データベースではそれは制限される．

3.3 DMS プログラムから RDMS プログラムへの簡単な変換例

本節では，ネットワーク・データベース・システム DMS 1100 のプログラム（DMS プログラム）を，リレーショナル・データベース・システム RDMS 1100 のプログラム

```
AREA  SECTION
AREA  NAME  IS  CMF-D1                          ──→⑥ファイル RDMS * CMF とする.
      CODE  2021
      ALLOCATE  5600  PAGES
      DYNAMICALLY  EXPANDABLE  6400
      PAGE  896
RECORD  SECTION
RECORD  NAME  M                                 ──→①表 RDMS. M とする.
 CODE  1
 LOCATION  MODE  INDEX
  IN  MULTI-AREA-NAME
  USING  ASCENDING  M-KEY                       ──→④表 RDMS. M のキーとする.
  LINKS  ARE  NEXT
 DUPLICATES  NOT
 WITHIN  CMF-D1  WITH  INDEX  AREA  CMF-I1      ──→⑦ファイル RDMS * CMF に属する.
 MODE  ASCII
 02  M-KEY    PIC  1(36)
 02  M-WORK   PIC  H9(10)   } ②表 RDMS. M の列となる
 02  M-DT     PIC  X(120)
RECORD  NAME  S                                 ──→①表 RDMS. S とする.
 CODE  2
 LOCATION  MODE  VIA  M-S  SET
 WITHIN  CMF-D1                                 ──→⑦ファイル RDMS * CMF に属する.
 MODE  ASCII
 02  S-KEY    PIC  1(36)  ·····                ·····→④表 RDMS. S のキーとする.
 02  S-WORK   PIC  H9(10)   } ③表 RDMS. S の列となる
 02  S-ITM    PIC  X(4)
SET  SECTION
SET  NAME  M-S
     MODE  CHAIN
     ORDER  SORTED
     OWNER  M
     MEMBER  S  AUTOMATIC
     ASCENDING  M-KEY                           ──→⑤表 RDMS. S のキーに加える.
     DUPLICATES  NOT
     SELECTION  CURRENT  OF  SET
```

図 10　DDL プログラム（表中の①～⑦は例 1 の番号と対応している.）

Fig. 10　DDL program

(RDMS プログラム）に単純に変換する例を示す.

この方法で変換されたプログラムは，論理的には変換対象のプログラムと同じ処理を行うが，実行効率の面からみれば，必ずしも RDMS を最適に使用したものにはならないことに注意されたい.

〔例 1〕 図 10 に DDL プログラムがある．これを段階をおって RDMS 化する例を次に示す.

① 1 レコード記述を一つの表に変換；

RECORD NAME M を表 RDMS. M に，RECORD NAME S を表 RDMS. S とする（RDMS は修飾子).

```
ⓐCAT,P  RDMS * CMF., F///4000
〔記憶領域の定義〕
CREATE  STORAGE-AREA  CMF  FOR  SCHEMA  RDMS.
ADD     FILE-TYPE EXEC.
ADD     QUALIFIER RDMS.
ADD     DATA-FORMAT RSM.
ADD     PAGE-SIZE  896.
ADD     MAXIMUM-PAGES  6400.
PROCESS  STORAGE-AREA  CMF  FOR  SCHEMA  RDMS  INSTALL.
〔表の定義〕
CREATE  TABLE    M     IN  RDMS.CMF
COLUMNS  ARE     M_KEY : DECIMAL(10),
                 M_WORK : DECIMAL(10),
                 M_DT : CHARACTER(120)
PRIMARY  KEY  K1  IS  M_KEY  ASC;
CREATE  TABLE    S     IN  RDMS.CMF
COLUMNS  ARE     M_KEY : DECIMAL(10),
                 S_KEY : DECIMAL(10),
                 S_WORK : DECIMAL(10),
                 S_ITM : CHARACTER(4)
PRIMARY  KEY  K2  IS  M_KEY  ASC,  S_KEY  ASC;
```

図11　RDMSプログラム

Fig. 11　RDMS program

② 表RDMS. Mの列は $\left\{\begin{matrix}\text{M\_KEY}\\ \text{M\_WORK}\\ \text{M\_DT}\end{matrix}\right\}$ とする.

③ 表のRDMS. Sの列は, $\left\{\begin{matrix}\text{S\_KEY}\\ \text{S\_WORK}\\ \text{S\_ITM}\end{matrix}\right\}$ とする.

④ 表RDMS. MのキーはUSING句からM_KEY, またUSING句の指定されていない表RDMS. Sのキーは仮にS_KEYとする.

⑤ セット記述からM_S関係を維持するために, 表RDMS. SのキーにM_KEYを加える.

⑥ 1エリア記述を一つのファイルに変換;
ファイル名RDMS * CMFとする.

⑦ 表RDMS. Mも表RDMS. SもWITHIN句で示されているファイルRDMS * CMFに属する.

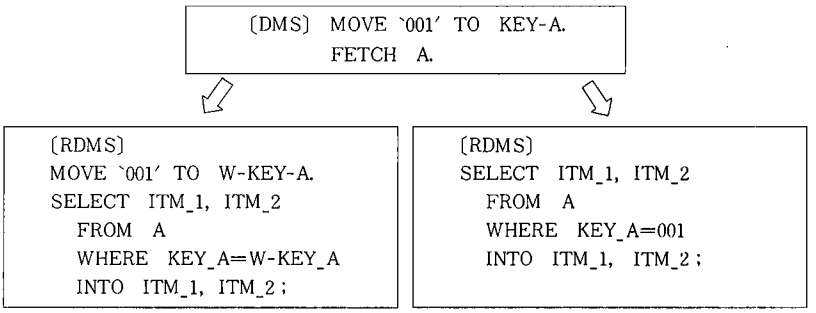
以上のようにして変換すると図11のRDMSのプログラムが作られる.

〔例2〕 DML指令 (DMSで扱う指令) とRDML指令 (RDMSで扱う指令) には, 次のような処理の差があり, それを考慮したのちDML指令をRDML指令に変換する例を示す.

DMSでは, 対象レコードを一意にするためには必要なキー値を指定されたデータ項目へ転送し, その後でDML指令を使用し処理する.

一方, RDMSではDMSと同様の処理のほかに, キー値を指定されたデータ項目へ

転送しないで，RDML 指令の中でキー値を指定し処理する方法もある．次に例を示す．



以上のことを考慮したのち，表 1 の DML 指令と RDML 指令の比較表を参考にして変換すればよい．

表 1　DML 指令と RDML 指令の比較表
Table 1　DML command vs RDML command

DML 指令	RDML 指令
IMPART.	BEGIN THREAD
DEPART.	END THREAD
FREE.	COMMIT
DEPART WITH ROLLBACK.	ROLLBACK END THREAD
KEEP	①なし：更新指令には暗黙的にロックがかけられる．
	②LOCK, UNLOCK
FETCH	SELECT (FETCH)
MODIFY	UPDATE
STORE	INSERT
DELETE	DELETE

4. RDMS プログラム開発上の技法

第 3 章では，ネットワーク・データベースを処理するプログラムを，リレーショナル・データベースを処理するプログラムに，できるだけ機械的に変換する方法について述べた．

本章では，リレーショナル・データベースを処理するプログラムを作成する上でのいくつかの補遺的なことがらについて述べる．本章の内容は，RDMS 1100 を使用する上での留意点である．

RDMS では RDML 指令は，親言語 (COBOL, FORTRAN) のサブルーチン呼び出し文の形式を採用している．

次の RDML 指令は，サブルーチン呼び出し文の引き数として RDMS に渡される．

```
01  DECLARE-CURSOR.                          (＊は全列指定の意味)
    02  FILLER PICX (30) VALUE  ' DECLARE TMP_TBL CURSOR  '.
    02  FILLER PICX (30) VALUE  ' SELECT ＊                '.
```

```
02  FILLER PICX (30) VALUE  ' FROM TABLE1;
:
ENTER MASM  'ACOB$RDMR' USING
                DECLARE—CURSOR
                ERR—STS
                CLM—POS.
```

このことは，RDML 文の構文解析がコンパイル時には成されないことを意味する．また，RDMS では，エラー・メッセージをプログラムにもらう手段が提供されており，エラー発生時に原因を詳細に知ることができる． DMS 1100 では，エラー処理段落やロールバック処理段落の機構が存在するが，RDMS ではそれがないので，エラー処理については利用者自身が処理手続きを指定する必要がある．

RDMS では，ファイルの物理的な定義が完全に論理的な定義と分離しており，いわゆる三層スキーマのデータ・アーキテクチャを採用している．

前章まで幾度も指摘したように，RDMS の操作命令はデータの集合を対象にしている．

以上のような特性から，RDMS のプログラムの作成では DMS 1100 の場合とは違った技法が存在する．本章では，RDMS のプログラムを組む上で有効な手立てを COBOL を例としていくつか採り上げる．

1) GETERROR 指令と構文解析……GETERROR 指令は，RDML 指令操作後のステータス判定に伴ってメッセージを印書するための指令であり，エラー発生時の事後処理として有効である．この指令は，RSA, DDS, RDMS, UDS コントロールに関する全エラー状況の適切な英文メッセージを返してくれる．使用者は基本的には各操作後，コントロールを GETERROR 指令のルーチンに飛ばし，エラー時のメッセージを随時見れるようにするとよい．しかし，RDML 指令の構文解析のみを実行したいならば，次のように DEBUG 指令を BEGIN THREAD 指令の前にコールすればよい．

```
PROCEDURE          DIVISION.
  ENTER    MASM  'ACOB$RDMR'  USING
                 'DEBUG STUB;'
                 ERR_STS
                 CLM_POS.
```

そして，構文のエラーのないことを確認できたなら，DEBUG 指令をとって実行する．また GETERROR 指令を使用しなくても，BEGIN THREAD 指令で次のように UDSMSG オプションを指定すると，

```
ENTER MASM 'ACOB$RDMR' USING
                'BEGIN THREAD FOR UDSSRC READ UDSMSG;'
                ERR_STS
                CLM_POS.
```

RDML 指令を操作した時点で即，UDS からのメッセージを返すようにできる．使用者は，効率の点から考えてこれらの指令を使い分ければよい．

2)　エラー発生時におけるプログラムの配慮……UDS や RDMS の外部エラーが発生してプログラムを終了させるような場合，スレッドを終わらせないまま終了させるべきではない．ROLLBACK，END THREAD 指令を出してから終了させることを勧める．

3)　サブルーチン呼び出し文における引き数の留意点……RDMS プログラム作成者は，レコード（組）の操作で変数を使用する場合，引き数を設定しなければならない．たとえば，ある表（A_TABLE）が列 B，C，D を含んでいたとしたら，UPDATE 指令は次のように書くことができる．

```
UPDATE A_TABLE
    SET C=$P 2, D=$P 3
    WHERE B=$P 1 ;
```

この UPDATE 指令には，表の変えたい列値だけを指定すればよい．しかし次の INSERT 指令の場合，

```
INSERT INTO A_TABLE (C, D)
VALUES ($P 1, $P 2)
```

INSERT 指令では，全項目に対して値を送ってやらなければならないので，列 B の値を指定していないこの INSERT 指令では，エラーとなってしまう．

また，FETCH 指令に関しては，次のような留意点がある．

DECLARE　CURSOR 指令時，SELECT 句に指定した項目の数と順序と FETCH 指令に指定する INTO 句の変数の数と順序は，合致させなくてはならず，その変数に対応する引き数の数も一致させなくてはならない．

たとえば，次のような DECLARE CURSOR 指令があると仮定する．

```
ENTER　MASM　'ACOB$RDMR'　USING
'DECLARE　CURSOR_A　CURSOR
SELECT　*　FROM　A_TABLE ;'
ERR—STS
CLM—POS.
```

このカーソル CURSOR_A は，全列を検索するよう指定してあるため，これに対する FETCH 指令は，以下のように三つの変数と三つの引き数を持っていなければならない．また，INTO 句の変数には，表定義で指定した順序でデータが入れられる．

```
ENTER　MASM　'ACOB$RDMR'　USING
'FETCH　CURSOR_A　INTO　$P1, $P2, $P3 ;'
ERR—STS
CLM—POS
B—ITEM
C—ITEM
D—ITEM.
```

4)　検索操作……検索機能におけるカーソルは，一つあるいは複数の表より条件に該当する列を集め，ある集合体を作り出す働きをするものだが，検索方法には次

の三つの型がある．

(a) DECLARE CURSOR (FOR RANDOM ACCESS)
LOCATE
FETCH CURRENT

(b) DECLARE CURSOR
(OPEN CURSOR)　(DECLARE CURSOR 指令で変数をカーソルで使用している場合)
FETCH NEXT

(c) SELECT

これらは，使用目的に従って使い分けをする．(a)と(b)の相違点は，(a)は主キーで検索する場合のみ使用可能であり，(b)は主キー以外でも検索できるという点である．また，主キーで検索するとき，主キーは重複が許されないことから1レコードのみをもってくる場合，内部的にカーソルを設定し，FETCH してくれる(c)の SELECT 指令を使用できる．しかし，(b)と(c)を比較した場合には考慮する点が異なる．

(c)の SELECT 指令は，(b)の DECLARE CURSOR → (OPEN CURSOR) → FETCH と違って，WHERE 句の条件で1レコードのみをもってくる指令である．次の二つのプログラムの相違点を考えてみる．

(1)
```
01 SELECT--CMND.
   02 FILLER PIC  X(30)  VALUE  ' SELECT *                    '.
   02 FILLER PIC  X(30)  VALUE  ' FROM A_TABLE                '.
   02 FILLER PIC  X(30)  VALUE  ' WHERE C=100                 '.
   02 FILLER PIC  X(30)  VALUE  ' INTO $P1, $P2, $P3;
LOOP.
   ENTER MASM 'ACOB$RDMR' USING
                SELECT--CMND
                ERR--STS
                CLM--POS
                B--ITEM
                C--ITEM
                D--ITEM.
                 ⋮
   IF ERR--STS NOT=6001      (6001 はカーソル指定で集められたデータの EOF ステータスである.)
     GO  TO  LOOP.
```

②
```
01 DECLARE--CURSOR.
   02 FILLER PIC  X(30)  VALUE  ' DECLARE--TMP--TBL CUR-
                                  SOR                       '.
   02 FILLER PIC  X(30)  VALUE  ' FOR RANDOM ACCESS          '.
   02 FILLER PIC  X(30)  VALUE  ' SELECT *                   '.
```

```
  02 FILLER PIC  X(30)  VALUE  ' FROM A—TABLE          '.
  02 FILLER PIC  X(30)  VALUE  ' WHERE C=100;          '.
01 FETCH—CMND.
  02 FILLER PIC  X(30)  VALUE  ' FETCH NEXT TMP_TBL    '.
  02 FILLER PIC  X(30)  VALUE  ' INTO $P1, $P2, $P3;   '.
          ⋮
LOOP.
  ENTER MASM 'ACOB$RDMR' USING
                  FETCH—CMND
                  ERR—STS
                  CLM—POS
                  B—ITEM
                  C—ITEM
                  D—ITEM.
  IF ERR—STS NOT=6001
     GO  TO LOOP.
```

ここで，WHERE 句で検索条件の対象としている項目が主キーではないとすると，データとしては重複があり得る．つまり，C が主キーでないとすると C=100 に相当するレコードが n 件あった場合，①と②の例で，①は②の二つの指令を単に一つの指令にまとめたように思えるが，①の SELECT 指令のループは，②の FETCH NEXT 指令のループとは異なり，レコードが複数件あろうとも，最初のレコードしかもってこない．つまり，何回 SELECT 指令を実行しようとも同じレコードしか返さないのである．しかし，②の FETCH NEXT 指令では，C=100 のレコードを次々ともってくるのである．使用者はこの点に留意し，SELECT 指令にするか，カーソル定義をして FETCH 指令を実行するかを考える必要がある．

5. おわりに

リレーショナル・データベースは，既存のデータベースやファイルと何が違うのか，あるいは何が同じなのかを実体験をベースにまとめてみた．本稿の対象作業は，既存のネットワーク・データベース・システムのプログラムを，なるべく機械的にリレーショナル・データベース・システムのプログラムに変換する際の問題点の把握が目的であった．

実務世界における DMS のシステムを RDMS のシステムに変換する時には，効率や運用・管理の面をもっと考慮した変換を考える必要がある．その意味では，本稿の内容は極めて初歩的なものである．

今回の報告は，体験した個別的事実を述べたものにすぎない．今後は，実際のアプリケーションを想定した有効な RDMS の使用法をテーマとして，さらに研究と実践を積み重ねてゆきたい．

参考文献 ［1］ E. F. Codd, “A Relational Model of Data for Large Shared Data Banks”, CACM 13 No. 6, 1970.

執筆者紹介 大内　忍 (Shinobu Ouchi)

昭和37年生. 60年法政大学工学部経営工学科卒業. 同年日本ユニバック(株)入社. 大型計算機のデータベース関連ソフトウェアの受け入れ, 保守業務に従事. 現在, 商品開発本部ソフトウェア2部データベース開発室 RDMS 課所属.

論文

COSTAR（ATD）による データベース・リカバリの運用自動化の実現

Realization of Automatic Operations to Recover Database by COSTAR (ATD)

小林俊平

要約 従来，データベース障害に対するリカバリ運用は，各データベースごとに運用手順・方法・ツールなどすべて異なった形態で行われていた．一方，データベース・リカバリの運用も複雑・多様化している．これらの問題を解決するために，データベースのリカバリ処理における運用管理システムCOSTAR (Computer Operating Management System for Total Automation and Reliance) (ATD) を開発した．

COSTAR(ATD)は，OS 1100の統合回復システムのもとで，データベースの障害に備え，データベース自体と統合回復システムが作成する更新ログ情報ATD (Audit Trail Disk，監査証跡ディスク) を保存・管理し，データベース破壊時には，すみやかな回復処理を行うものである．

本システムにより，従来の複雑・多様なリカバリ運用から脱却し，標準の自動リカバリ運用が確立し，コンピュータ室の無人運転へ一歩前進した．

本稿では，COSTAR (ATD) の開発の背景・経緯と，今後の自動リカバリ運用について述べる．

Abstract The recovery operations for database troubles caused by such as contingency error and power failure, vary in procedures and tools according to each alternative database.

As application systems are growing rapidly in size and complexity, the problem of database recovery is growing correspondingly. COSTAR (ATD), the acronym of Computer Operating Management System for Total Automation and Reliance (Audit Trail Disk), is a solution for it.

COSTAR (ATD), based on the Integrated Recovery (IR) facility of OS1100 is able to mechanize database recovery using an audit trail, which is a file of updated logs produced by IR.

COSTAR (ATD) realizes a standard automatic recovery of database and makes a considerable progress in unattended operation of computer system.

This paper describes the background of its development, its system structure, and future automatic operation of database recovery.

1. はじめに

情報化社会を支えている情報は，ほとんどがデータベースとしてコンピュータに蓄積されている．これらは，膨大な情報量であり増加の一途をたどっている．今やデータベースが何らかの原因により破壊されると，情報化社会の機能は完全に停止するであろう．

今回開発されたCOSTAR (ATD) は，データベースの障害発生時のリカバリ処理の自動化によって，コンピュータ運用の自動化をさらに前進させるものである．

本稿では，シリーズ1100のオペレーティング・システムOS 1100の統合回復システム（Integrated Recovery, IR）のもとで，データベース・リカバリの自動化が，COSTAR（ATD）によってどのように実現されたか，また今後のデータベース・リカバリ処理におけるコンピュータの運用はどうあるべきかについて述べる．

2. COSTAR（ATD）開発の背景

2.1 統合回復システムの出現

システム制御プログラムEXECレベル39より統合回復システム(以降，IRと呼ぶ)が提供された．従来は，データベースの障害に備え，各システムのデータベースごとにリカバリ運用手順・方法・ツールなどすべて異なった多様な運用形態をとっていたが，IRの出現により統一的手順でリカバリ処理ができるようになった．具体的には，各アプリケーション・プログラムによるデータベースの更新ログ情報(以降ATDと呼ぶ）が，ディスク上に一つの連続したファイルとして保存されるようになった．リカバリ情報の統合化である．このIRにより，リカバリ処理の統合化が大きく前進したと言える．図1にIRの構成を示す．

COSTAR(ATD)は，図の右側のデータベース管理をベースとして構築されている．

2.2 運用上の必要性

IRの出現により，従来とは異なる統合化されたリカバリ処理が可能となった．しかし，機能としては非常に有効なものと言えるが，コンピュータの運用・管理においては人手に頼らざるをえない状況であった．以下に，人手が必須な作業をあげる．

1) ATDの保存・管理……アプリケーション・プログラムからの各データベースへの更新ログ情報およびトランザクション・メッセージが，一括してディスク・ファイルに保存されるようになった．しかし，磁気ディスクの容量の限界や安定性確保を考えると，このATDを保存・管理する必要があり，以下の運用を必要とする．

・ATDのアーカイブ処理（アーカイブとは，ATDをテープへコピーすること）

・アーカイブされたATDのディスク上からの削除．

2) データベースの保存・管理……ATDを保存・管理すると同時に，データベースの保存・管理のために，以下の運用を定期的に行う必要がある．

・データベースのバックアップ（テープへの保存）

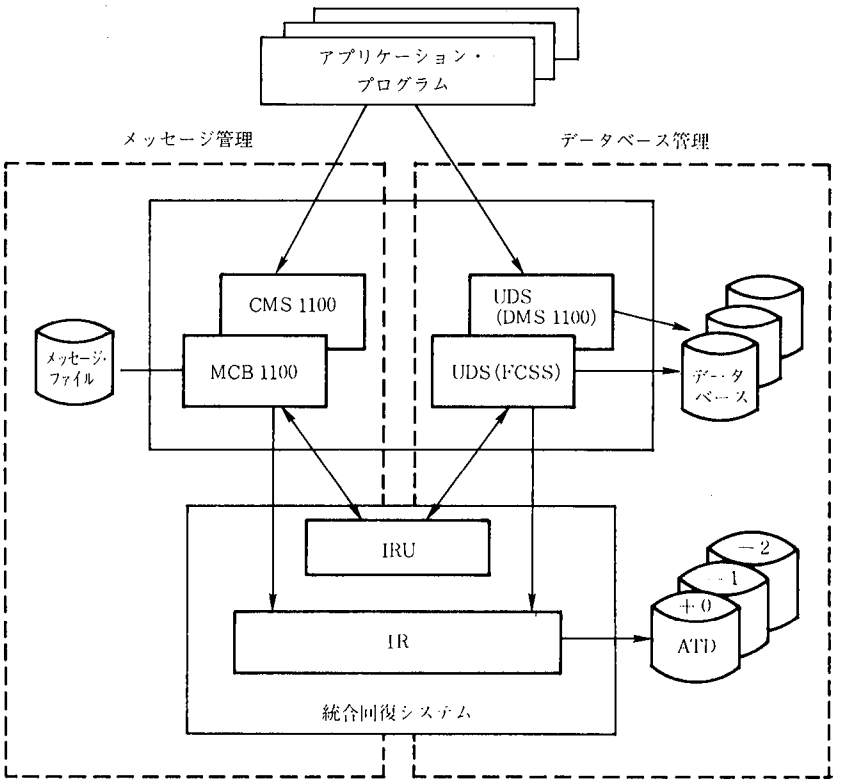
このATDとデータベースの保存・管理を怠るとデータベース破壊時，その復元が不可能となる．また，IRによりATDはディスク上に蓄積され続けるため，システムの停止を引き起こす．

IRでは，これらに対する機能を実行するソフトウェアIRU 1100が提供されている．しかし，運用するのは利用者であり，このような正確かつ迅速を要するリカバリ処理を人手で行えば，いずれはトラブルがトラブルを呼ぶ結果となるであろう．

また，一般的に情報処理システムが抱えている次の三つの問題からも対応が必要であった．

3) 業務量増大の一途（図2）→ 業務処理の複雑・多様化への対応

4) 人の過失による事故・障害の多さ（表1）→ ミスの防止



CMS 1100 : Communication Management System
MCB 1100 : Message Control Bank
UDS : Universal Data System
DMS 1100 : Database Manegement System
FCSS : : File Control Super Structure
IRU : Integrated Recovery Utility

図1　統合回復システムの構成

Fig. 1　Integrated recovery system

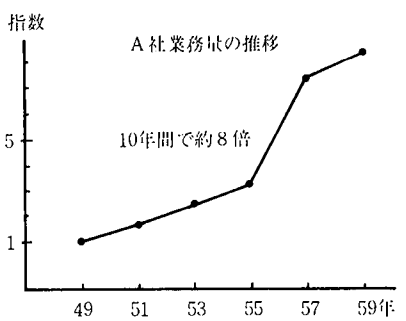


図2　業務量の増加[2]

Fig. 2　Increase of application volume

表 1　システム事故・障害状況[2]
Table 1　System failures

システム事故・障害状況（過去1年の経験）

順位 ＼ 年		58, 59	60	61
1	ハードウェア障害	83.4%	79.4%	80.8%
2	ソフトウェア障害	58.5	54.9	57.9
… …	… …			
5	人の過失による事故・障害	23.4	22.5	21.9

表 2　コンピュータ部門の運用経費
Table 2　Operation cost of EDP department

コンピュータ部門の運用経費　　月額1社当たり金額(百万円)

年 ＼	人件費			人件費＋外注費			機械設備費			合　計		
53	9.6			13.4			17.0			34.9		
55	10.1	% +5.2	% +10.7	15.0	% +11.8	% +18.3	16.4	% −3.9	+10.2	36.6	% +5.0	% +15.6
57	10.6	+5.2		15.8	+5.8		18.8	+14.6		40.3	+10.1	
59	11.8	+10.8	+19.7	22.5	+35.4	+43.7	22.8	+21.3	+30.0	50.7	+25.7	+35.5
61	12.7	+10.8		22.8	+6.2		24.4	+7.0		54.6	+7.8	

5)　コンピュータ部門の運用経費の伸び（表2）→　コストの増加抑制

以上の諸問題点から，データベースのリカバリ処理における運用管理システムCOSTAR(ATD) の必要性が生まれてきた．

2.3　COSTAR(ATD) の要件

COSTAR(ATD)の目標は，データベースのリカバリ処理の運用を自動化することである．この目標を達成するには，以下の事項を満足する必要がある．

1)　ATDの容量を定期的に監視し，必要時ATDをアーカイブする処理ランを起動する．

2)　ATDのアーカイブ・ランに対し，アーカイブ用リールを正確に供給するとともにアーカイブ結果を保存する．

3)　データベースのバックアップ処理に対し，正確なリールの供給を行うとともにバックアップ結果を保存する．

4)　データベース破壊に対し，データベースの復元に必要なバックアップ・リールおよびATDのアーカイブ・リールを正しく供給するとともにリカバリ・ポイントを指示する．

2.4　開　発　経　過

以上の背景から，COSTAR(ATD)は総合運用管理システムCOSTARの第4のサブシステムとして開発された．

開発工数として，約50余人月を要し，61年4月にジェネラル・リリースされた．

3. COSTAR(ATD) の開発

3.1 設 計 方 針

以下の考え方を基本として設計されている．

1) 正確性の保障……ATD およびデータベースをテープに保存し，処理の経過にそって連続的に正確に保存テープを管理する必要がある．また，データベース破壊による復元においても，正しい保存テープおよびリカバリ・ポイントを利用者に供給しなければならない．

2) 運用の自動化……利用者の判断とか操作を極力なくすことにより，操作ミスを防止する．

3) 自衛手段の設定……本システムは，データベースの破壊に備えた障害対策であるが，COSTAR (ATD) 自身の管理情報も，もちろんデータベースであり，自分自身で自衛する以外にない．したがって，障害対策が必要となる．

4) サブシステムの柔軟性……COSTAR のサブシステムとしてすでに提供されているスケジュール管理 COSTAR (SCH)，運行制御 COSTAR (RUN) およびファイル管理 COSTAR (FILE) と連動して，または単独で使用できる．

3.2 適 用 範 囲

もちろん EXEC レベル 39 の IR のもとで実現されたシステムであるが，対象となるデータベースは以下のものである．

1) FCSS ファイル
- オンライン制御プログラム TIP 1100 により生成されるファイル

2) ネットワーク型データベース
- DMS 1100 レベル 8 R 3 により生成されるファイル
- UDS DMS 1100 により生成されるファイル

なお，PCIOS (Processor Common I/O System) 系ファイルは対象外である．

4. COSTAR(ATD) の機能

4.1 概　　要

COSTAR (ATD) は，図 3 の概要図に示すように三つの機能，つまり「ATD のアーカイブ機能」，「データベースのダンプ」，「データベースの回復」から構成されている．

4.1.1 COSTAR (ATD) のねらい

データベースの障害，すなわち破壊に対するリカバリ処理を正確に，かつすみやかに行うためには，使用するテープを正しく選択することである．

1) 上述の三機能に対し，使用すべき正しいリールを提供する．

2) 業務処理の遂行とともに蓄積された ATD を動的監視し，ある一定量の ATD が蓄積されると，ATD のアーカイブおよび削除を行うランを起動実行する．

前者は，COSTAR (ATD) と COSTAR (FILE) が，後者は，COSTAR (ATD) がその役割を果たしている．

3) 上述の三機能の実行用制御文を自動生成することにより，運用部門の負荷を削減する．

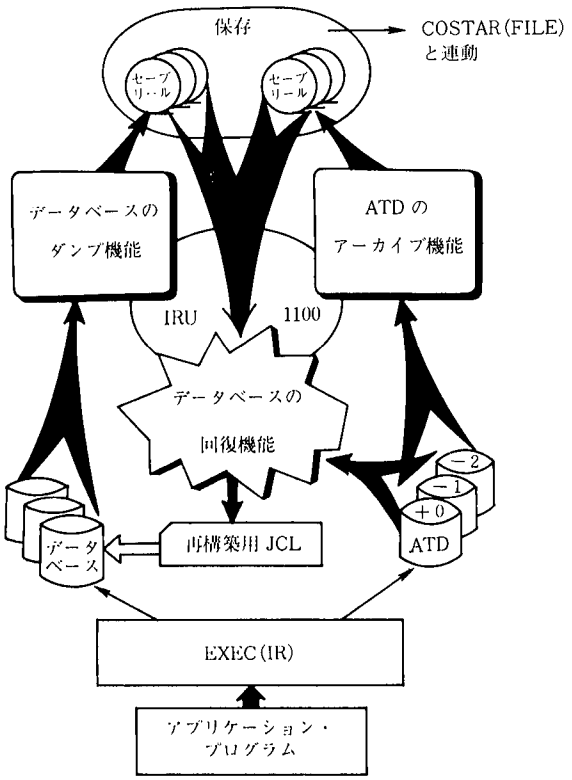


図3 COSTAR (ATD) の概要図

Fig. 3 Overview of COSTAR (ATD)

4.1.2 COSTAR(ATD) の特徴

1) COSTAR (PKG-A)*との連動

① 使用テープの保存・管理……COSTAR (PKG-A) のファイル管理サブシステムと連動し，使用テープ・リールの世代管理を行っている．

② 出庫指示書……COSTAR (PKG-A) のスケジュール管理およびファイル管理サブシステムと連動し，使用テープ・リールの事前出庫を可能にしている．

2) 総合オンライン支援システム AIS 1100 との連動

① データベースのダンプに関する情報の一元化……データベース・ダンプに必要な情報を AIS 1100 の管理マスタ ISD (Integrated System Dictionary) に登録することにより一元管理している．

② ATD の監視ランの統一……AIS 1100 より ATD の監視処理 (VMONIT) が提供されており，これを利用することにより常駐ランを統合している．

4.1.3 全 体 構 成

COSTAR (ATD) は，大別すると次の三機能から構成されている．

1) ATD のアーカイブ機能……データベースの更新ログ情報(ATD)を監視し，自動的にテープに保存し，そのテープを管理する．

---

* COSTAR (SCH), COSTAR (RUN), COSTAR (FILE) を一つに結合したシステム．

・ATDのディスク・エリアの監視
・ATDのアーカイブ用リールの割り付けおよび管理
・ATDのアーカイブ処理の実行

2) データベースのダンプ機能（バックアップ）……データベースの破壊発生に備えて，ダンプ用リールを割り付け，データベースをテープに出力する．
・ダンプ用リールの割り付け
・データベースのダンプ実行JCLの作成または作成と実行

3) データベースの回復……データベースの破壊が発生した場合，1)，2)の処理結果をもとにデータベースのリカバリJCLを作成する．
・ATDのアーカイブ・リールの選択
・ダンプ・リールの選択
・データベースのリカバリ実行JCLの作成

また，管理ファイルとして次のARLMとDGMがある．

① ARLM（Archive Reel Master）……ATDのアーカイブに使用するリール情報を管理している．FCSS非回復型二重化ファイルであり，アプリケーション・グループ*ごとにファイルを設定する．

② DGM（Dump Group Master）……データベースのダンプに関する情報を管理している．ACOB MSAM（COBOLの多重索引順編成）ファイルであり，全アプリケーションに対して1個用意されている．

以上の機能，および管理ファイルとCOSTAR（PKG-A）との連動をとるユーティリティで構成されている（図4）．

4.2 ATDのアーカイブ機能

1) ATDのディスク・エリアの監視

① ATDのサイクル数を一定時間間隔で監視し，あるサイクル以上の世代ファイルが作成されると，アーカイブ・ランを起動する．なお，これは各アプリケーション・グループのレグ**ごとに行う．また，必要に応じてアーカイブ・ランを並行起動させる．

② ATDの状況は，時間の経過とともに変化するため，アーカイブ・ランに対しランの条件ワード***を介して知らせる．条件ワードについては，図5を参照のこと．

③ その他として，オペレータと会話できる機能が用意されており，ATDのアーカイブ状況などを見ることができる．

2) ATDのアーカイブ用リールの割り付けおよび管理

ATDは一世代ごとに独立したファイルではなく，連続した複数世代で構成されたファイルで，終りがない．したがって，テープに保存する場合でも連続した形で管理する必要がある．このため，一つのATDの容量を次の基準で規定した．

---

* ファイル，プログラムの実行およびメッセージの厳密な区切りのこと．各々のアプリケーション・グループは，一意な名前と番号によって識別され，完全に独立したリカバリの単位である．

** 2重ファイルの一つのコピー．

*** ランの起動時，ランにある制御情報を設定し，実行タスクに知らせる．また，タスク間の制御情報としても使用される．

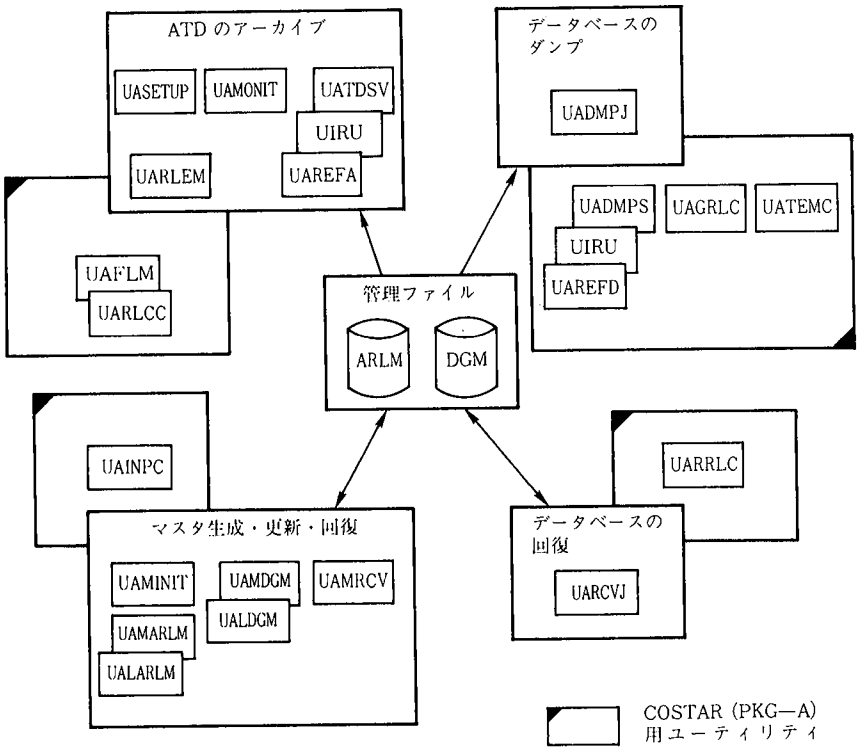


図4　COSTAR（ATD）の全体構成

Fig. 4　Structure of COSTAR (ATD)

2^{11}	2^{10}	2^{9} ～ 2^{0}
a	b	c

a：0　c に指定された一世代のみのアーカイブ
　　1　最新2世代以外のすべての世代のアーカイブ
b：0　レグ1
　　1　レグ2
c：アーカイブ対象の ATD の F サイクル番号

図5　条件ワードの形式

Fig. 5　Format of condition word

ATD の容量を，EXEC システム定義パラメタによって1本のリールに収まるように定義する．こうすることにより，

- テープの連続性保証

  ATD のアーカイブ・ランの並行処理によって，ラッシュ時に対応する．
- テープエラーのとき，とくに出力リール不足時の対応を可能にしている．

なお，ARLM で各アプリケーション・グループのレグごとに，最大500リールが管理可能である．

3)　ATD のアーカイブ処理の実行

① IRU 1100 (IRU プロセッサ)……アーカイブ処理実行は，IRU 1100 により行われる．当システムでは，監視処理から条件ワードを介して知らされる ATD の情報と，ARLM よりアーカイブ用リールを 1 本入手し，次のような IRU 実行制御文を自動生成し@ADD する．

```
@IRU
MOVE    AUDIT   FCYCLE  Fサイクル番号  レグ番号
  TO    REEL    リール番号
  TYPE  装置名；
ACT     APPL    アプリケーション名；
END；
```

なお，ATD の削除は IRU では行わず，アーカイブ結果を正しく保存した後に当システムによって削除される．

② UIRU プロセッサ……IRU によりアーカイブ処理が実行されるが，この実行結果を把握しないことには正しい ATD の管理ができない．したがって，本システムと IRU とを連動する機能が必要となった．これが，UIRU プロセッサである．この UIRU により入手したアーカイブ結果は，ARLM に保存されデータベースのリカバリ処理で使用される．

4.3 データベースのダンプ機能

1) データベースのダンプの範囲

① 対象データベースとしては，FCSS ファイル，DMS EXEC ファイル，DMS TIP ファイルがある．

また，DGM に登録する関係としては，ダンプ・グループ（最大 8 個までのエリア・グループの集まり），エリア・グループ（最大 24 個までのエリアの集まり），エリア（データベース）がある．

② ダンプ方法としては，IRU の DUMP 命令 (IRU 1100)，および HSDMPLD (TIP サポート・ユーティリティ）がある．

③ ダンプの種類としては，静的ダンプおよび動的ダンプがある．

2) ダンプ・リールの管理

エリア・グループ単位にデータベースのダンプが実行される．

① 単独システム……COSTAR (ATD) のみで運用するシステムでは，ダンプ・リールは使用者側で管理する．ダンプ実行時，パラメタにより指定する．

② 連動システム……COSTAR（PKG-A）と連動して運用するシステムでは，ダンプ・リールは COSTAR（FILE）が管理する．

3) データベースのダンプ処理

① IRU 1100 または HSDMPLD……データベースのダンプ処理実行は，IRU 1100 または HSDMPLD により行われる．本システムでは，ダンプ方法およびシステム形態により実行 JCL を自動生成し@ADD する．実行 JCL は 4 ケースある．

なお，IRU でダンプ処理を実行する場合，ダンプ結果を入手するために，UIRU を必要とする．

4.4 データベースの回復機能

1) データベースの回復の範囲

① 回復処理の単位

- 単独システム： エリア・グループ，エリア
- 連動システム： アプリケーション・グループ，ダンプ・グループ，エリア・グループ，エリア

② 回復用リールの入手方法

表3にその方法を示す．

表3 回復用リールの入手方法
Table 3 Obtainment of recovery reel information

リール	ダンプ・リール	アーカイブ・リール
単独システム	パラメタ	パラメタ／ARLM
連動システム	COSTAR(FILE)のファイル・マスタ	同　左／ARLM

③ データベースの回復処理……データベースのリロードおよびリカバリ実行のJCLは，本システムにより自動生成される．表4にその形式を示す．

表4 リロードおよびリカバリ実行JCLの形式
Table 4 Form of reload and recovery JCL

ユーティリティ／JCL	IRU	HSDMPLD
データベースリロード用	DOWN ⋮ RELOAD ⋮	DOWN ⋮ @XQT HSDMPLD ⋮
リカバリ用	RECOVER FILES … FROM REEL…TO REEL… UP ⋮	

自動生成された実行JCLは，

- エリア（データベース）のダウン用とリロード用として，エリア・グループごとに次のファイル名でカタログされる．

  ATD $ RLD * エリア・グループ名

- リカバリ用とアップ用として，次のファイル名で一つカタログされる．

  ATD $ RCV * ALL

この機能は，実行JCLを作成するだけである．実行は，@ADD制御文により使用者側で行う．なお，実行時，JCLの先頭に必ず@IRU制御文を付加しなければいけない．

また，実行終了後，JCLのカタログ・ファイルは削除されないので，使用者側で削除する．

4.5 データベース破壊への考慮（自衛手段）

本システムでは，ARLMとDGMのマスタをもっているが，後者は固定情報のため，前者のみデータベース破壊時の考慮を行っている．

1) 第1の対策……ARLMは，FCSS非回復型二重化ファイルであり，破壊防止のためにTIP 1100の機能を利用している．

2) 第2の対策……第1の方法でも，レグ1，2ともにデータベースが破壊される可能性はある．そのため，次の方法をとっている．

ATDのアーカイブ結果をディスク上に保存する．

この情報は，UIRUプロセッサによりアーカイブ情報としてディスク上に保存される．したがって何らかの原因でARLMが破壊されても，このアーカイブ情報をもとにARLMを復元することができる．

また，使用者がアーカイブ結果を把握していれば，パラメタ入力によりARLMを復元することもできる．

5. COSTAR(ATD) による運用と効果

5.1 今後のリカバリ運用

従来の運用とCOSTAR（ATD）を利用した運用とを比較し，どの範囲が自動化・省力化されたかを見てみる．なお，従来の運用は，DMS 1100 レベル8R2におけるATT（Audit Trail Tape，監査証跡テープ）を使用する．表5に比較表を示す．

COSTAR(ATD)を使用することにより，従来の運用とは比較にならないほど，ほとんどの範囲で自動化されたと言える．なお，自動化された処理と機能を表6にまとめてみた．

5.2 COSTAR(ATD) の運用形態

COSTAR（ATD）のみの運用（単独システム）と，COSTAR（PKG-A）との連動による運用（連動システム）に大別されるが，四つの形態がある．表7に運用形態に対する機能構成および自動化の範囲を示す．

また，AIS 1100と連動する運用では，次の項目が可能になる．

1) データベースのダンプに関する情報の一元化

2) ATDの監視ランの統一

5.3 B社における運用

B社では，61年5月よりCOSTAR（ATD）を利用した運用を開始した．

1) 表7の運用形態5 ＋ AIS 1100の運用……COSTAR-ATDの全機能とCOSTAR（PKG-A）およびAIS 1100とを連動したリカバリ運用の自動化が確立されたシステムである．

2) 一日の運用プロセス（ホット・スタンバイ運用）……オンライン系の運用を図6に示す．また，業務処理オーバーフロー時（1日/月）には待機系にて処理される．図7に待機系の運用を示す．

表5　リカバリ運用の新旧比較

Table 5　Comparison between new recovery operation and old one

処理 ＼ 運用システムの種別	従来運用 (DMS 11008 R 2)	EXECレベル 39	
		IRのみ	COSTAR(ATD)
ATTまたはATDの監視	人　手	人　手	自　動
ATDアーカイブ・ランの起動	――	人　手	自　動
ATTまたはATDのセーブ処理(JCL作成)	DMSがセーブするが，テープのスワップはキーイン	IRUが保存するが，パラメータ作成またはリールの入力は人手	パラメタ作成およびリール入手は自動
ATTまたはATDアーカイブ用のサーボ・ユニット	1日中（1台専有）	ATDアーカイブ時のみ	ATDアーカイブ時のみ
ATTまたはATDアーカイブ・リールの管理	人　手	人　手	自　動
データベース・ダンプ処理(JCL作成)	人　手	人　手	連動システムでは自動
データベース・ダンプ処理の起動	人　手	人　手	COSTAR(PKG-A)による自動
テープの事前出庫指示	人　手	人　手	COSTAR(PKG-A)による自動
ダンプ・リールの管理	人　手	人　手	COSTAR(FILE)による自動
リカバリ処理(JCL作成)	人　手	人　手	自　動
リカバリ処理の起動	人　手	人　手	人　手
リカバリ・ポイントの指示	人　手	人　手	自　動
テープの装着	人　手	人　手	人　手

表6　自動化された処理と機能

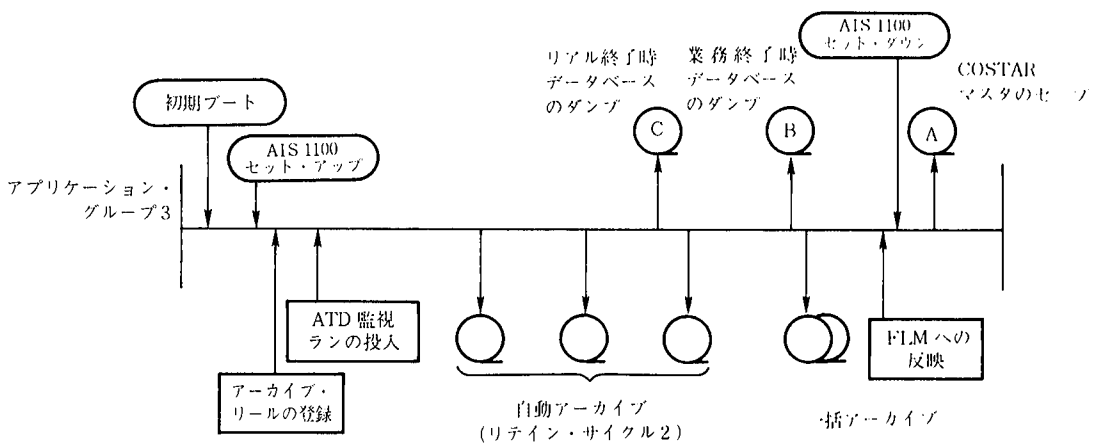
Table 6　Automated process and function by COSTAR (ATD)

処理 ＼ 機能	COSTAR(ATD)			COSTAR(PKG-A)		
	ATDのアーカイブ	データベースのダンプ	データベースの回復	COSTAR (SCH)	COSTAR (RUN)	COSTAR (FILE)
ATDの監視	○					
ATDアーカイブ・ランの起動	○					
ATDのセーブ処理(JCL作成)	○					
ATDアーカイブ・リールの管理	○					○
データベース・ダンプ処理(JCL作成)		○				
データベース・ダンプ処理の起動				○	○	
テープの事前出庫指示				○		○
ダンプ・リールの管理						○
リカバリ処理(JCL作成)			○			
リカバリ・ポイントの指示			○			

表7　COSTAR(ATD)の運用形態

Table 7　Alternative operations of COSTAR (ATD)

形態	使用目的	COSTAR (ATD)			COSTAR (PKG-A)	自動化の範囲
		ATDのアーカイブ	データベースのダンプ	データベースの回復		
1	ATDを保存させる.	○				40%
2	ATDを保存し，データベースの破壊時には，データベース復元（データベース保存情報は使用者より指定）用JCLを作成する.	○		○		55%
3	ATDおよびデータベースを保存し，データベースの破壊時には，データベース復元用JCLを作成する.	○	○	○		70%
4	ATDのアーカイブ・リールの出庫を行う.	○			○	50%
	また，アーカイブ・リールを1日1世代で管理する.	○		○	○	60%
5	データベースのダンプ・リールを管理するとともに，ATDのアーカイブ・リールおよびダンプ・リールを1日1世代として管理する.	○	○	○	○	100%



ATD はレグ1，レグ2ともに保存する.

リテイン・サイクル：自動アーカイブの対象としない ATD の世代数のこと．たとえば，リテイン・サイクル2の場合，3世代（＋0，－1，－2）の ATD が作成されると，－2世代の ATD はアーカイブされるが，＋0と－1世代はアーカイブされない.

FLM：ランで使用するファイル情報を保有しているマスタ・ファイル（リール番号，相対世代数など）で，COSTAR (FILE) で使用される.

図6　B社の運用（オンライン系）

Fig. 6　Application of COSTAR (ATD) (online system)

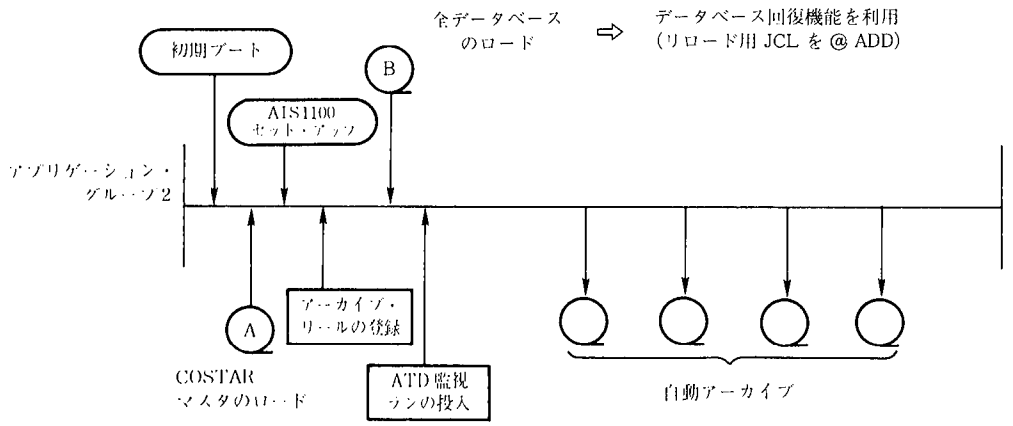


図7　B社の運用（待機系）

Fig. 7　Application of COSTAR (ATD) (backup system)

5.4　COSTAR（ATD）の効果

COSTAR (ATD) により，データベースのリカバリ運用の自動化が実現し，以下の効果が得られた．

1)　リカバリ運用の自動化・省力化……テープの装着およびリカバリ処理の起動を除き，自動化・省力化された（表5）．

2)　正確性の向上……ATDのアーカイブ・リールおよびデータベースのダンプ・リールをCOSTAR（ATD）およびCOSTAR（FILE）で管理するため，正確性が保証された．

3)　迅速なリカバリ……人間の判断が入らないため，すばやいデータベースの復元が可能となった．

4)　運用の標準化が確立……自動化により運用が標準化された．また，業務量の増大に対しても十分対応可能となった．

6.　お　わ　り　に

COSTAR (ATD) の開発により，データベースのリカバリ運用の自動化が実現できたが，コンピュータ運用には種々の形態が存在し，現機能では対応できない場合もある．そこで，今後の課題を数項目あげてみた．

1)　データベースの再編成……データベースの使用効率を良くするために，ユーザでは再編成が行われる．この再編成処理を考慮したデータベースのダンプおよびリカバリ処理が必要である．

2)　PCIOS系ファイル……PCIOS系ファイルは，IRの対象ではないので，現在のシステムは完全なリカバリの統合化とは言えない．

以上への対応が必要であるが，本システムのみでは解決できず，EXECのIRとIRU 1100の改良とともに順次対応していくことになるだろう．

今回開発したCOSTAR（ATD）により，IRをベースとしたデータベースのリカバリ運用が自動化され，コンピュータ室の無人化が一歩前進した．

COSTAR（ATD）は，昭和61年5月に，提供開始されたが，現在は5社で使用されている．今後は，さらに多くのユーザに適用し，標準としてのデータベースのリカバリ運用を確立していきたい．

本稿の最後に，今回の共同開発メンバーであったAIS 1100の開発保守の関係者，およびIRで協力いただいたデータベースの開発保守の関係者に心から感謝の意を表したい．

参考文献　［1］　1100/90シリーズ　システム概説書，資料コード481293001-1，日本ユニバック(株)
［2］　情報化白書1987，日本情報処理開発協会1987年3月31日発刊

執筆者紹介　小　林　俊　平（Shunpei Kobayashi）

昭和46年横浜市立大学商学部卒業．同年日本ユニバック(株)入社，フィールド・サービスを経て，54年以降総合運用管理システムCOSTARの開発・保守に従事．現在，システム第二本部システム企画開発二部運用管理システム開発室に所属．

論文 各種バーコードのパーソナル・コンピュータによる印書実験

An Experiment of Bar-Codes Printing by Personal Computers

森　宗正

要　約　最近バーコードの応用範囲が広がってきている．バーコードを気軽に使うことを考え，手近にあるパーソナル・コンピュータとワイヤドット漢字プリンタを利用し，NW-7，CODE-39およびJANの各種バーコードを印書し，バーコード・リーダで読み取り実験を行って，その可能性を確認した．

その結果，最小黒ドットバーの幅を，NW-7，CODE-39では1ドット以上，JANでは2ドット以上とすれば十分実用に耐えることが確認できた．

Abstract　Currently use of bar-code is growing and found in various areas. Bar-codes (NW-7, CODE-39, and JAN) printing and reading experiment had been done with personal computers and wire-dot Kanji printers, to evaluate its readability and investigate practical use of it.

Bar-code reader can read NW-7 and CODE-39 code even with 1-dot width black bar, while JAN code requires 2-dots.

1. はじめに

低コストで，かつ高性能のバーコード・リーダーが出始めたことにより，バーコードの応用範囲が広がってきている．パーソナル・コンピュータで，バーコードが自由に印書できれば，さらに利用範囲が広がることが予想できる．筆者は，かつてドットプリンタで印書した場合のバーコード幅を理論的に考察し，黒バーと白バーの幅を等しくするには，白バーのドット数を黒バーより1ドット多くすれば良いとの結論を得ている[1],[2]．その結果をもとにワイヤドット・プリンタで実際にバーコードを印書し，バーコード・リーダーで読み取る実験を行った．

手近にあるパーソナル・コンピュータUP 10Eモデル50およびDS-7を使って，NW-7，CODE-39およびJANの3種のバーコードを印書して，オフラインのバーコード・リーダーで読み取りを行い，実用化の確認を行った．

考察に用いた外字を利用するバーコード印書では，バーコードの幅が24ドットに固定されるために，JANのようにセンタバーの幅が他のデータコードの幅と異なるものは印書が不可能であり，また任意のドット幅のバーコードの印書もできない．その解決方法として，ドットイメージ印書を採用することにした．この方法は外字方式と比較して，プログラム量が増え，印書速度が低下するが，文字としての大きさの制約がなく，任意のドット幅でバーコードが印書できる．

今回の実験では，最小黒バーの幅を，1，2，3および4ドットに指定，選択して印字するプログラムをBASICで作成し，印字結果をバーコード・リーダーで読み取って，読み取り可能な最小ドット幅の限界を求めた．その結果NW-7とCODE-39では，1ドット幅の最小黒バーでも，読み取りが可能であり，十分実用性のあることが

わかった．一方，JAN では最小黒バーとして 2 ドットが必要である．

本稿では，まずバーコードについて説明し，そのイメージ印書の方法について述べる．つぎにこの方法で印書したバーコード・パターンの仕様と，印書および読み取り結果について報告する．最後に，バーコード印書を実用化するために必要なプリンタの機能等について考察する．

2. バ ー コ ー ド

2.1 バーコードとは

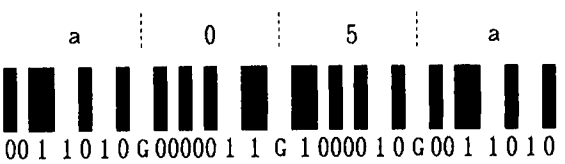
バーコードとは，幅の異なる白と黒の平行バーを組み合わせて，数字や文字を表現する手法である．このバーコードを読み取る機器がバーコード・リーダーである．OCR と比べると，読み取りの認識アルゴリズムが簡単なため可読率が高く，機器コストが安い．このため POS の分野や工業界その他の分野で広く利用されている．

バーコードには多くの種類があるが，現在利用度の高いものとしては，NW-7，CODE-39, JAN, ITF 14 などがある．以下にその概要を示す．各バーコードのパターン構成については，付録を参照されたい．

2.2 NW-7

米国 Pittony Bowes 社が POS 用に開発した CODABAR と呼ばれるバーコードがその原型である．1 字を黒バー 4 本，白バー 3 本の計 7 本のバーで構成する．黒バー，白バーとも狭い（Narrow）バーと広い（Wide）バーがあり，この 7 本の狭・広の組み合わせにより，数字と若干の記号を表現する．ここから NW-7 の名が生まれた．基本的には，広いバーの数を 2 本として，自己検査を可能にしている．当初の仕様では，狭いバーと広いバーの比が，1：2 であったが[3]，最近では 1：2.5～1：3 に変ってきている[4]．文字間には狭い白バーの幅以上の白ギャップを挿入する．現在，NW-7 は，製造業界その他で使用されている．

図 1 に，“05” を NW-7 バーコードで表現した例を示す．



a： スタート，ストップ
G： 文字間ギャップ

図 1 NW-7 のバーコード・シンボル例
Fig. 1 Bar-code symbol of NW-7

2.3 CODE-39

米国 Intermec 社が開発したバーコードで，数字と記号の他に英字も表現できる．さらに記号と文字を組み合わせることにより，すべての ASCII 文字を表現することが可能である．NW-7 と同様，黒白の狭・広バーを使うが，表現可能文字数を増やすため，

黒バーが5本，白バーが4本と計9本のバーを用い，このうち3本を広いバーとしている．CODE-39の名はこのバーの数からきている．パターンの構成上，黒バーと白バーのコードを分離した方が見やすくなる(付録参照)．ANSIで規格化の動きもあり，そこでは3-OUT-OF-9コードと呼ばれている[4]．NW-7同様，狭いバーの幅以上の文字間ギャップを必要とする．

CODE-39も製造業界での利用率が高い．とくに英字が必要な部品等へのアプリケーションに使用されている．英字が表現可能な半面，1字を表現するバーの数が多く，コード長が大きくなる欠点がある．

2.4 JAN

POSで利用するために，製造者が食品や雑貨等の包装に印刷（ソース・マーキーング）するバーコードである．日本では，JAN（Japanese Article Number）コードと呼ばれ，商品が一意的に決まるように（財）流通システム開発センタ内の流通コードセンタがコードを管理している．その寸法等はJIS X 0501-1985(旧B 9550)共通商品コード用バーコード・シンボルとして標準化されている．JANコードを印刷した商品は，1983年7月現在で対象品目の約40パーセントと言われている[6]．JANコードは，また計量商品などに店内で印書する（インストア・マーキング）ことも可能である．

このコードは，最初，米国で雑貨，食品，薬品等を対象としたUPC（Universal Product Code）として規格化された．次いで欧州で，UPCをベースにしたEAN（European Article Number）と呼ぶ世界的な商品コード体系にまで発展した．日本もEANの協会に参加し，国名コードとして49の割り当てを受け，JANは世界的に認められるコードとなっている．現在，UPC，EAN，JANは，一つのバーコード体系と考えられており，POSのバーコード・リーダーは，通常この3種類のバーコードを読み取ることができる．

JANの場合，1，2，3，または4モジュールの黒バー2本と白バー2本で計7モジュールになるように1桁を構成し，数字だけを表現する．ここでモジュールとは，バーコードを構成する基本単位で，その寸法は0.26～0.66 mmの範囲で可変である．JANには13桁の標準バージョン，8桁の短縮バージョンがある．

JANコードには，左右にライトおよびレフト・ガードバーがあり，これがスタート，ストップ・コードの役割を果たしている．また，中央にセンタバーがあるが，その左右でバーコードが異なり，それぞれ左側キャラクタ，右側キャラクタと呼ばれている．左側キャラクタには，黒バーのモジュール数を奇数にした奇数パリティ・キャラクタと，偶数にした偶数パリティ・キャラクタがある．右側キャラクタは，偶数パリティ・キャラクタだけである．

13桁の標準バージョンは，センタバーの両側に，それぞれ6桁のバーコードしかなく，もう1桁を左側キャラクタの奇数パリティ・キャラクタ，偶数パリティ・キャラクタの組み合わせで決めている．この追加の1桁をフラッグ・キャラクタと呼び，第1桁目に置くことになっている（表1参照）．

短縮バージョンでは，左側は奇数パリティ・キャラクタ，右側は偶数パリティ・キャラクタを使用し，フラッグ・キャラクタは，左側のデータ・キャラクタの中に含めて直接表示する．

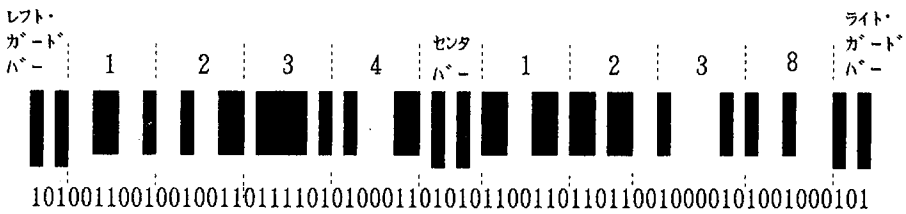


図2 JAN 短縮バージョンのバーコード・シンボル例
Fig. 2 Bar-code symbol of JAN (simplified version)

表 1 フラッグ・キャラクタの構成
Table 1 Construction of flag characters

フラッグ・キャラクタ	左側キャラクタの奇数偶数の組み合わせ					
0	O	O	O	O	O	O
1	O	O	E	O	E	E
2	O	O	E	E	O	E
3	O	O	E	E	E	O
4	O	E	O	O	E	E
5	O	E	E	O	O	E
6	O	E	E	E	O	O
7	O	E	O	E	O	E
8	O	E	O	E	E	O
9	O	E	E	O	E	O

O：奇数パリティ・キャラクタ
E：偶数パリティ・キャラクタ

図2に，短縮バージョンで“1234123（8）”を表した例をあげる．最後の桁の“8”は，モジュラス10によるチェック・キャラクタである．

2.5 ITF 14 (Interleaved Two out of Five 14)

今回実験は行わなかったが，流通関係で広く利用されているバーコードに，ITF 14がある．ITF 14は，JIS X 0502-1987 物流商品コード用バーコード・シンボルとして今年制定された．JIS では14桁の標準バージョン，16桁の拡張バージョンおよび6桁のアドオン・バージョンの3種類のバーコード・シンボルを規定している[7]．商品コードは，JAN 同様流通コードセンタで管理している．標準バージョンは JAN に流通識別コードを1桁追加したものであり，拡張バージョンは流通識別コードを2桁にしたものである．また，アドオン・バージョンは計量等の表示に用い，データ5桁，モジュラス10によるチェック・キャラクタ1桁を表す．

ITF 14では，黒バー，白バーそれぞれに狭・広バーがあり，広いバーが1，狭いバーが0を表している．この点は NW-7 や CODE-39 と同様である．ただ，左からみて偶数桁を黒バー，奇数桁を白バーで表す点が異なる．各桁は5本のバーからなり，このうち2本のバーだけが広くなるようにコードが構成されている．

図3に“3852”を表すバーコードの例を示す．

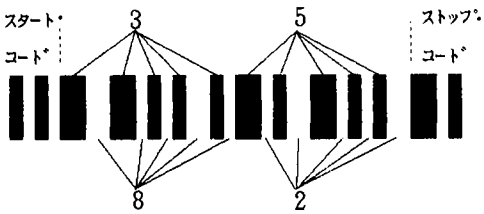


図3　ITF 14 バーコード・シンボル例
Fig. 3　Code-symbol of ITF 14

3. バーコードの印書方法

パソコン等でバーコードを印書する方法としては，外字を利用する方法と，ドットイメージ・モードを利用する方法の二つが考えられる．システム外字の登録の方法や外字コードの付け方は，コンピュータ・システムにより異なる．一方，外字 RAM のロードの方法やドットイメージ・モードの指定方法は，プリンタにより異なる．多くの場合，ESC シーケンスを利用するが，その符号の使い方は千差万別である．

今回の実験では，まず UP 10 E モデル 50 で外字方式による CODE-39 の印書を行って読み取りの可能性を確認し，その後，種々のビット幅のバーコードの読み取りの可能性を試験するために，ドットイメージ・モード方式に変更して UP 10 E モデル 50 および DS-7 で実験を行った．

3.1 外 字 方 式

外字を利用する場合，システムとしてバーコード・パターンをあらかじめ登録しておく方法と，バーコード印書プログラムの最初の部分でバーコード・パターンをプリンタの外字 RAM にロードする方法とがある．一般にユティリティがそろっているなどプログラム作成の容易性では前者が勝り，融通性の面ではドットパターンを任意に入れ換えて印書可能な後者が優れている．しかし両者の間には本質的な相違はない．

外字方式の利点は，バーコードが 2 桁の外字コードとして扱え，印書データが小さく，その結果，実効印字速度が速くなることである．DS-7 の 473 プリンタの場合，次に述べるドットイメージ印書より高速に印書できる．ただし，UP 10 E モデル 50 の QQ 1207 プリンタの場合は，黒の連続印書のときプリントヘッドの温度上昇を抑えるために，1 字ごとに停止，後退，前進，印字が行われ，その効果があまり出ない．

外字方式の欠点は，1 字当たりのパターンが 24 ドットに固定されるため，任意の幅のバーコードが印字できないことである．とくに JAN の場合，文字幅がデータバーと異なるセンタバーがあるため，文字幅が固定されている外字方式では印書が不可能である．

3.2 ドットイメージ・モード方式

ドットイメージ・モードは，任意の図形を印書するためのモードであり，最近のドット方式漢字シリアル・プリンタの多くが採用している．この方式を使えば，任意の文字幅のバーコード・パターンの印書が可能であり，また，JAN コードも印書可能である．半面，縦 24 ドット 1 列を印字するのに，3 バイトのビットイメージ・データをプリンタに転送しなければならない．外字方式で 1 字分のデータは，2 バイトの符号

ですむのに対して，この方式では72バイト（3×24）が必要である．データ量が大きいため，転送時間がかかり実効印字速度が低下する．また，プリンタによってその実現方法が大きく異なり，移植には注意が必要である．

3.3 実験に使用したプリンタの仕様概略

バーコード印書に関係するプリンタの仕様は，次の通りである．

	0473プリンタ	QQ 1207プリンタ
印字方式	ワイヤドット・マトリックス	ワイヤドット・インパクト
ドット径	0.2 mm	明記なし
ドットピッチ	縦 1/180インチ（0.141 mm）	縦 1/180インチ（0.141 mm）
	横 1/180インチ（0.141 mm）	横 1/180インチ（0.141 mm）
最小紙送り量	1/120インチ	1/120インチ

両者の仕様上の差はほとんどないが，印書結果はかなり異なっている．ハードウェア上の相違というよりも，むしろ使用するリボンの特性によるものと考えられる．

4. バーコード・パターン

実験に用いたバーコード・パターンは，最小黒バー幅を1～4ドットに選択できるようにした．実験に使用した各種バーコードのパターンを以下に示す．なお，各表で使用する略号と意味は次に示す通りとする．

Bn	狭い黒バー（NW-7，CODE-39用）
Bw	広い黒バー（NW-7，CODE-39用）
Wn	狭い白バー（NW-7，CODE-39用）
Ww	広い白バー（NW-7，CODE-39用）
Gap	文字間ギャップ（NW-7，CODE-39用）
B 1～B 4	1～4モジュールの黒バー（JAN用）
W 1～W 4	1～4モジュールの白バー（JAN用）

4.1 NW-7のパターン

NW-7の場合，バーコード・パターン要素の組み合わせにより，文字間ギャップを除く1文字の構成は表2に示す3種類になる．黒バーと白バーのドット数が違うこと，広いバーの数の多い記号があることによって，文字幅が異なってくるが，これを一定とするように文字間ギャップで調整した．なお，最小文字間ギャップは狭い白バーの幅と等しくした．NW-7では狭い黒バーとして1～4ドットの4種類のパターンを作成し実験した．各要素の具体的なドット数と文字間ギャップを含めた1字当たりのドット数を表3に，また各サイズの文字間ギャップを表4に示す．

4.2 CODE-39のパターン

CODE-39の場合，バーコード・パターン要素の組み合わせにより，文字間ギャップを除くパターン構成は表5に示す2種類になるので，NW-7同様文字幅が一定となるように，文字間ギャップで調整した．なお，最小文字間ギャップは狭い白バーの幅と等しくした．NW-7同様，狭い黒バーを1～4ドットとした4種類のパターンを作成し実験した．各サイズのパターン要素のドット数と文字間ギャップを含めた1文字当たりのドット数を表6に，また文字間ギャップを表7に示す．

表 2　NW-7 の各パターン要素数

Table 2　Number of pattern elements in NW-7code

単位：個数

	Bn	Bw	Wn	Ww
数字, −, ¥	3	1	2	1
始, 終符号（4種）	3	1	1	2
:, /, ., +	1	3	3	0

表 3　NW-7 のパターン要素のドット数

Table 3　Number of dots of pattern elements in NW-7 code

単位：ドット

最小黒バー幅	Bn	Bw	Wn	Ww	1字
1	1	4	3	6	25
2	2	5	3	7	31
3	3	8	4	9	43
4	4	10	6	12	58

表 4　NW-7 の文字間ギャップ

Table 4　Inter-character gap in NW-7code

単位：ドット

	最小黒バー幅			
	1	2	3	4
数字, −, ¥	6	7	9	12
始, 終符号（4種）	3	3	4	6
:, /, ., +	3	5	4	6

表 5　CODE-39 の各パターン要素数

Table 5　Number of pattern elements in code-39

単位：個数

	Bn	Bw	Wn	Ww
一般文字および記号	3	2	3	1
特殊記号	5	0	1	3

表 6　CODE-39 のパターン要素のドット数

Table 6　Number of dots of pattern elements in code-39

単位：ドット

最小黒バー幅	Bn	Bw	Wn	Ww	1字
1	1	4	3	6	29
2	2	5	3	7	37
3	3	8	4	9	50
4	4	10	6	12	68

表 7 CODE-39 の文字間ギャップ
Table 7 Inter-characters gap in code-39

単位：ドット

	最小黒バー幅			
	1	2	3	4
一般文字および記号	6	5	4	6
特殊記号	3	3	4	6

4.3 JAN のパターン

JAN（UPC，EAN も同様）の場合は前二者と異なり，文字部は 7 モジュール，左右のガードバーは 3 モジュール，およびセンタバーは 5 モジュールからなり，文字間ギャップは使用しない．設計上，左右のガードバーは，不要部分に白バーを挿入して 7 モジュールにした．黒バー，白バーともに 1～4 モジュールの幅のものがある．前回の考察結果から，白バーのドット数は，同じ幅の黒バーより 1 ドット多くしたパターンを採用した．1 モジュールの黒バーのドット数を 1～4 にした時の各要素のドット数を表 8 に示す．なお，1 モジュールの黒バーを 1 ドットとしたパターンでは，各バー幅の比率が許容範囲を越えてしまい，残念ながら読み取りが不可能であった．しかし，1 字当たりのバーの数が少なく，最小黒バー幅の 1 ドットの他のバーコードよりパターン長が短かくてすむ．

表 8 JAN のパターン要素のドット数
Table 8 Number of pattern elements in JAN

単位：ドット

最小黒バー幅	B 1	B 2	B 3	B 4	W 1	W 2	W 3	W 4	1字
1	1	2	3	4	2	3	4	5	9
2	2	4	6	8	3	5	7	9	16
3	3	6	9	12	4	7	10	13	23
4	4	8	12	16	5	9	13	17	30

5. 印 字 結 果

① 最小 1 ドット黒バー

② 最小 2 ドット黒バー

③ 最小 3 ドット黒バー

④ 最小 4 ドット黒バー

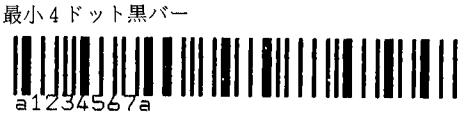


図 4 NW-7 の印字例
Fig. 4 Print sample of NW-7

① 最小1ドット黒バー

② 最小2ドット黒バー

③ 最小3ドット黒バー

④ 最小4ドット黒バー

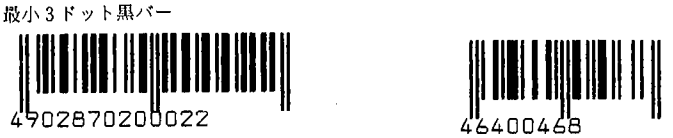
図5 CODE-39の印字例

Fig. 5 Print sample of code-39

① 最小1ドット黒バー

② 最小2ドット黒バー

③ 最小3ドット黒バー



④ 最小4ドット黒バー

図6 JANの印字例（左：標準バージョン，右：短縮バージョン）

Fig. 6 Print sample of JAN (left : standard version, right : simplified version)

以下では，上記のパターンを使用して印字した例を紹介する．

5.1 DS-7による印字例

DS-7と0473プリンタを使った印字例を図4～図6に示す．0473プリンタの場合，黒ドットの連続があっても印字の停止がなく，どのドット幅の場合も比較的きれいな印字結果が得られた．

5.2 UP-10E モデル 50 による印字例

UP-10 E モデル 50 と，QQ 1207 プリンタを使った印字例を図 7～図 9 に示す．QQ 1207 プリンタの場合は，黒ドットの連続があると，一度印字を停止，後退して再び印字を再開する．その頻度は高くパターンにもよるが，1字に1回以上の停止が起こることがある．その結果，印字速度が低下し，さらに位置ずれや印字品質の低下も招いている．今回の実験では，読み取りに大きい支障はなかったが，0473 プリンタに比べて目視でも印字品質がかなり悪いことが判る．

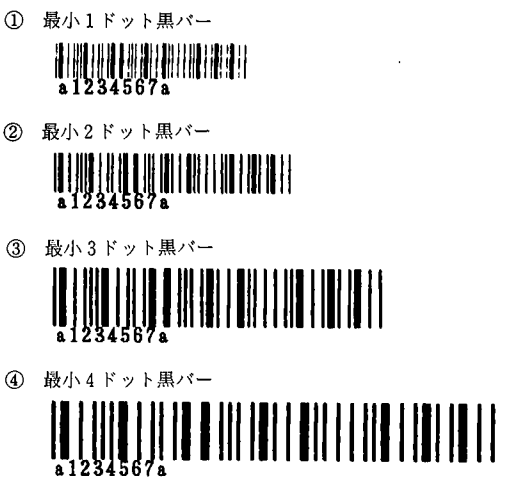


図 7 NW-7 の印字例
Fig. 7 Print sample of NW-7

図 8 CODE-39 の印字例
Fig. 8 Print sample of code-39

① 最小1ドット黒バー

② 最小2ドット黒バー

③ 最小3ドット黒バー

④ 最小4ドット黒バー

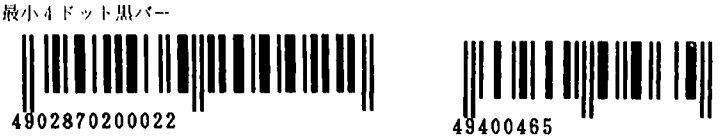


図9 JANの印字例（左：標準バージョン，右：短縮バージョン）

Fig. 9 Print sample of JAN (left : standard version, right : simplified version)

6. 読み取り結果とその考察

6.1 使 用 機 器

読み取り実験には，手近にあったライトペン操作方式で，複数種類のバーコードが読み取り可能なテスコ㈱製のTescom-7000をオフラインで使用した．このリーダーは，今回の実験に使用したバーコード（JAN，NW-7，CODE-39）の他に，標準型，インタリーブ（ITF 14），マトリクスなどの2 OUT OF 5系のバーコードを，人手による切り換えなしに読むことができる．読み取り率などは，ごく普通の機器と考えられる．ただし，リーダーにより最適パターンが異なることがあり，今回の実験結果がすべてのリーダーにあてはまるかどうかはわからない．

オフラインの状態でも，バーコードの読み取りができた時にブザーを鳴らし，同時に読みとったデータを液晶に表示する．液晶は，24字×2行の表示容量がある．ペンは，分解能が0.16～0.3 mmのものを複数種類用いてみたが，ペンによる明らかな相違は見られなかった．

なお，今回は使用しなかったが，ホストとのインタフェースとして，RS 232 CおよびRS 422が用意されている．

6.2 読み取り結果

1) NW-7およびCODE-39の読み取り……DS-7の0473プリンタで印書したNW-7およびCODE-39は，最小黒バーのドット数が1から4まですべて読み取り可能であった．また，UP 10 Eモデル50のQQ 1207プリンタで印書したバーコードでは，最小黒バーのドット数を1にした場合を除いて，読み取り可能であった．QQ 1207プリンタで印書した場合，1ドットの黒バーは，目で見ても明らかに印字品質が良くない．しかし完全に読めない状態ではなく，何回か読み直すと読める程度である．

一方，最小黒バーのドット数が4の場合は，桁数が多くなるとペンを動かす距離が長くなりすぎて，かえって読み取り率が悪くなる．とくに1字当たりのバー数の多いCODE-39の場合，その傾向が強い．今回の実験からは，最小黒バーのドット数を2～3に選ぶのが適当であるとの結論が得られた．

2) JANの読み取り……DS-7の0473プリンタ，UP 10Eモデル50のQQ 1207プリンタのどちらで印書しても，最小黒バー幅が1ドットの場合は，1：2：3：4のバー幅の比率が得られないため極めて読み取り率が悪く，実用化は不可能である．最小黒バーを2ドット以上にすれば，ともに十分に読み取りができ，実用化が可能である．JANの場合，1字当たりのバー数が少ないので，最小黒バーを4にして13桁を表現しても長過ぎるということはない．したがって，JANの最小黒バーのドット数は2～4とするのが適当である．

3) 印字品質……バーコードを印字する場合は，人間を対象とする場合よりも，より高い印字品質が要求されると思われた．今回は印字品質の分析は行わなかったので，主観的な評価ではあるが，意外に要求される印字品質は低くてもよいようである．とくに初期の実験で黒バーと白バーのドット数を等しくとった場合，QQ 1207プリンタでは，新しいリボンよりむしろ使い古したリボンの方が読み取りやすかった．これは新しいリボンの場合，インクのにじみで白バーが小さくなるためと考えている．かなり薄くなったリボンでも読み取りができたのは意外であった．

7. バーコード印書に適したプリンタ

今回の実験から，バーコードを印書する場合，下記の機能が必要であると思われる．

1) ドットイメージ・プリント機能……最近のワイヤドット・プリンタは，大部分がこの機能を持っている．ただし機種によっては，その機能が使いにくかったり，印字速度が極端に低下するものがあるので注意が必要である．また，プリンタの機種ごとに制御コードが異なるので，移植の場合に問題がある．0473プリンタとQQ 1207プリンタの間でも，ドットイメージ・プリント・モードにするための制御コードが，以下のように異なる．

0473プリンタの場合：

$ESC+(E9)_h+(30)_h+n1+n2$

QQ 1207プリンタの場合：

$ESC+(25)_h+(31)_h+n1+n2$

注：ともにこの後にイメージ・データが続く．

n1, n2はドット列の数を表す．実際に転送するデータのバイト数は，この3倍になる．

なお，QQ 1207の場合は，この制御コードを出す前に漢字モードにしておく必要がある．また，両者ともプリント・バッファの幅を，255にセットしておかなければならない．

2) 片方向印字機能……最近の漢字プリンタは，キャリッジの移動方向が左右どち

らでも印字できるが，キャリッジの移動方向によって，位置ずれを起こすことがある．バーコード印書の場合，この位置ずれが問題になる．位置ずれを調節できるプリンタもあるが，調節範囲が1ドット単位であり，最大±0.5ドットのずれが起こりえる．この位置ずれを防ぐために，印字時のキャリッジの移動方向を一方向だけにすることが望ましい．0473，QQ 1207プリンタは，ともにこの機能を持っている．

3) 黒ドット連続プリント機能……プリンタによっては，黒ドットの塊を印書できなかったり，その都度，停止，後退，前進，印字を行うものがある．前者ではバーコードの印書は不可能になる．後者は，印書可能であるが，印書速度が遅く，位置ずれを起こしやすい．QQ 1207プリンタは，その後者にあたる．0473プリンタは，そうした制約がなく，印書が円滑に行われ，位置ずれが起こりにくい．0473プリンタとQQ 1207プリンタの印書結果にも，その差が出ている．

8. おわりに

今回の実験から，24ドットのワイヤドット・プリンタで印書したバーコードは，十分バーコード・リーダーで読み取れることが確認でき，さらに読み取り可能な最小黒バーのドット数が確認できた．

今回の実験では，ワイヤドット・プリンタで印字したバーコードが，実用化できるかどうかに主眼をおいたのでバーコードのパターンの最適化については，ほとんど考慮していない．パターンは，理論的考察をもとに直感的に作成したものである．それでも，大部分が1回の走査で読みとれる程度の印書が可能である．

残っている問題としては，リーダーにとって最も読みやすいパターンを見つけることと，プリンタのリボンの問題がある．

先に述べたように，パターンは理論的に作成したものであり，これが読みやすいパターンであるという保証はない．これについては，さらに検討が必要である．ただしパターンの読みやすさは，バーコード・リーダーの特性と密接な関係があり，特定リーダーに対する読みやすいパターンを求めることになると思われる．

また，新しいリボンと使い古したリボンとでは当然印字結果が異なり，バーコードの読み取り率に変化が出るはずである．さらに，リボンの物理的，化学的および光学的な特性について解析が必要である．

なお，今回の実験の性格上，プログラムの最適化や，印字速度の向上等には手を付けていない．実用化に当たっては，これらの問題についても検討が必要である．

付録　バーコードのパターン構成

NW-7，CODE-39，JAN および ITF 14 のバーコード・パターン構成を以下に示す．

1）NW-7

文字	符号	バーコード・パターン	文字	符号	バーコード・パターン
0	0000011	0 0 0 0 0 1 1	−	0001100	0 0 0 1 1 0 0
1	0000110	0 0 0 0 1 1 0	$(¥)	0011000	0 0 1 1 0 0 0
2	0001001	0 0 0 1 0 0 1	:	1000101	1 0 0 0 1 0 1
3	1100000	1 1 0 0 0 0 0	/	1010001	1 0 1 0 0 0 1
4	0010010	0 0 1 0 0 1 0	.	1010100	1 0 1 0 1 0 0
5	1000010	1 0 0 0 0 1 0	+	0010101	0 0 1 0 1 0 1
6	0100001	0 1 0 0 0 0 1	a	0011010	0 0 1 1 0 1 0
7	0100100	0 1 0 0 1 0 0	b	0101001	0 1 0 1 0 0 1
8	0110000	0 1 1 0 0 0 0	c	0001011	0 0 0 1 0 1 1
9	1001000	1 0 0 1 0 0 0	d	0001110	0 0 0 1 1 1 0

1：黒または白の広いバー
0：黒または白の狭いバー
スタート，ストップ・コードには，a～dを使用する．

2）CODE-39

文字	バーコード・パターン	黒　白	文字	バーコード・パターン	黒　白
1		10001 0100	M		11000 0001
2		01001 0100	N		00101 0001
3		11000 0100	O		10100 0001
4		00101 0100	P		01100 0001
5		10100 0100	Q		00011 0001
6		01100 0100	R		10010 0001
7		00011 0100	S		01010 0001
8		10010 0100	T		00110 0001
9		01010 0100	U		10001 1000
0		00110 0100	V		01001 1000
A		10001 0010	W		11000 1000
B		01001 0010	X		00101 1000
C		11000 0010	Y		10100 1000
D		00101 0010	Z		01100 1000
E		10100 0010	−		00011 1000
F		01100 0010	.		10010 1000
G		00011 0010	(SP)		01010 1000
H		10010 0010	*		00110 1000
I		01010 0010	$		00000 1110
J		00110 0010	/		00000 1101
K		10001 0001	+		00000 1011
L		01001 0001	%		00000 0111

1：黒または白の広いバー
0：黒または白の狭いバー
スタート，ストップ・コードには，*を使用する．

3)JAN(JIS X 0501)

	左側キャラクタ		右側キャラクタ
	奇数パリティ	偶数パリティ	偶数パリティ
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

ガードバーおよびセンタバー	
レフト・ガードバー	101 レフト・マージン
ライト・ガードバー	101 →ライト・マージン
センタバー	01010

1：黒モジュール
0：白モジュール
｜：隣の文字の黒バーの終りのエッジ

4)ITF 14(JIS X 0502)

文字	バーコード・パターン	文字	バーコード・パターン
0	00110	5	10100
1	10001	6	01100
2	01001	7	00011
3	11000	8	10010
4	00101	9	01010

スタート・コード	0000
ストップ・コード	100

1：黒または白の広いバー
0：黒または白の狭いバー
スタート，ストップは左からみて黒から始まる．

参考文献 ［1］ 森 宗正，漢字プリンタ用バーコード・パターンの理論的考察，テクニカルシンポジウム '82 (S7308)，1982.
［2］ 森 宗正，漢字プリンタ用バーコード・パターンの理論的考察（その2），テクニカルシンポジウム '83 (S7510)，1983.
［3］ CODABAR for identification and control, Monarch Marking Systems, 1792.
［4］ Bar Code Reading Systems，東研.
［5］ Code-39 Alphanumeric BAR CODE Specifications, INTERMEC, 1982.
［6］ 最新版 POS システム百問百答，流通システム開発センタ，1983.
［7］ 田中 信夫，JIS X 0502 物流商品コード用バーコード・シンボルの制定にあたって，標準化ジャーナル，Vol. 17, No. 6, 1987, 6, pp. 11～14.

執筆者紹介 森 宗正（Munemasa Mori）

昭和 30 年，早稲田大学第 1 理工学部電気通信工学科卒業．同年，吉沢会計機㈱入社，昭和 33 年日本レミントン・ユニバック㈱（現日本ユニバック㈱）入社．現在，技術研究部主任研究員．主として ISO，JIS 等の標準化関連に従事．現在 ISO/SC 13 国内委員会 WG 2 主査，INSTAC I/O インタフェース標準化調査研究委員会分科会主査．情報処理学会，電子情報通信学会会員．著書に「電子計算機のハードウェア入門」（共立出版，1971）がある．

ICAIの技術動向

Technology Trend of Intelligent CAI

橘田 明, 右近 豊
A. Kitta, Y. Ukon

1. はじめに

コンピュータを利用した教育を一般にCBE (Computer Based Instruction) と呼ぶ. CBEは, 学習者の学習履歴や成績処理などを行う管理システムCMI (Computer Managed Instruction) と教授=学習プロセスへの適用としてのCAI(Computer Aided Education) とに分けることができる.

学習者に対する教材の提示方法をコンピュータ上で実現することから始まったCAIの研究も, もうすでに20年以上経過している. その間, さまざまな研究・実験がなされ, 飛躍的な発展はないにしても着実に実用化が進みつつある.

反面, コースウェア開発に要するコスト負担増から先進技術の成果が十分活かしきれていない点が多く, そのことがCAIの是非(教授=学習プロセスにコンピュータを利用することの是非) やCAIの限界などの議論を呼んでいる. とくに, 実用に供されているシステムにおいては, 個別学習への対応 (学習者の反応にうまく対応して学習を進める諸機能) の不十分さが指摘されているケースが多い.

ところが近年AI (Artificial Intelligence : 人工知能)の研究が盛んになるにつれ, AIのCAIへの応用, すなわちICAI (Intelligent CAI) の試みが見られるようになった.

ICAIの試みは, 個別の学習者への対応, 質問応答など, 従来のCAIの限界を打ち破る可能性を示唆している.

本稿では, これからのCAIの可能性を追求するため, CAIシステムを教授=学習システムの一部としての枠組みであらためてとらえ直し, CAIの発展, ICAIの事例などを取り上げ, その技術動向を明らかにする.

2. CAIの類型

コンピュータを教育に利用する研究は, 心理学・教育学の発展や科学技術の革新等を背景に1960年前後からアメリカを中心に始まった. その後, コンピュータ自体の発達や低価格化などを受けて国内でも徐々に研究・開発され, 1970年代後半からパソコンを中心としたCAIシステムが商品化されている.

現在のCAIシステムは, 大まかにAFO (Ad hoc Frame Oriented)型CAI, 生成型CAI, ICAIの三つに類型化できるが, これらの類型を概観する前にまずCAIシステムの基本構成要素を提示しておきたい.

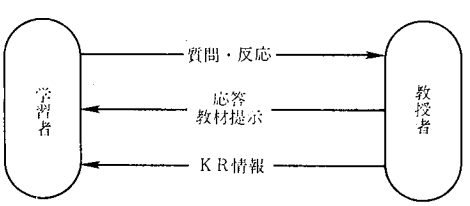
2.1 CAIシステム

CAIシステムは, 教授=学習システムのうち, 教授者側機能のコンピュータ上での実現ととらえることができる.

ところで, 教授=学習システムにおいて学習者と教授者の間には, 次のようなやりとりがある(図1).

学習者の質問・反応 (教材要求などを含む) に対し, 教授者が応答し, 教材を提示し, KR情報を返す等である. こうしたやりとりを可能にするためには教授=学習システムの教授者の側としてコンピュータには, 少なくとも次の四つの機能が必要とされる.

1) 教授内容を管理する機能……学習者に教授したい内容を管理する機能である. 教師が授業に臨む際には, その授業における目標と教授内容に関する知識を持ち, 教授資料としての教材を準備している. CAIシステムにおいても, これに対応する機能が必要である. しかし, その実現レベルと実現方式にはさまざ



KR : Knowledge of Result (評価結果情報)

図1 学習者と教授者のやりとり
Fig. 1 Interface between student and tutor

まなものがある．具体的には単に教材を提示単位順にデータ・ファイルとして持っているだけのものもあれば，構造化・体系化した教材のデータベースとその検索機能という形態もある．また教材を生成するプログラムという形のものもあれば，フレームや意味ネットワークなどの知識表現技術を用いて教授目標や教授内容を管理しているものもある．

2) 教授方略を管理する機能……教師は授業の過程でさまざまな意思決定をしている．たとえば，授業のある場面では表1の授業行動を決定し，同時にその材料となる教材を選び出す．こうした指導手段と系列を決定することを教授方略と呼ぶが，CAIシステムにもこれを管理する機能が必要となる．ただし，これは1)の機能が柔軟に教材を提供できない場合には，あまり意味がない．

3) 学習者の状態を理解する機能……学習者の状態すなわち，教授目標に対する到達度や学習内容に関する理解状態，学習行為の履歴や学習習慣等の特性などを理解する機能をいう．教師が授業を展開するとき，学習者がどのような状態にあるのかをつねに判断しながら進めているように，学習者の個性や理解状態に応じて教授＝学習活動を展開するために必要な機能である．CAIシステムにおいては，単にある問題に対する解答を場合分けするものから学習者の状態を推論によって導き出すものまでさまざまな実現形態がある．

4) 学習者インタフェースの機能……学習者からの入力を内部形式に，あるいは内部形式を学習者への出力に変換する機能．ハード，ソフトの両面を含む．また，教師が学習者との対話の中で学習者の本質的でない誤りを言い直すことがあるように，入力の単純な解析機能を含めることもできる．現在，学習を進めるうえでのシステムとの対話形式（コマンド形式やメニュー形式）や入出力装置（AV関連機器や高解像度画像処理装置およびマウス，タッチパネル，音声入出力装置など）はシステムの利用分野・対象者によってさまざまなものが使われている．いずれにせよ，将来的には自然言語処理技術の進歩に伴い，会話と筆記による学習本来のインタフェースが取り入れられることは間違いない．

以上四つの機能を表す単位をそれぞれ教授内容モジュール，教授方略モジュール，学習者理解モジュール，学習者インタフェース・モジュールと

表1　授業者による授業行動のカテゴリ[11]
Table 1　Category of teaching behavior

教師		子ども
機能	種類	
1. 思考要求 2. ゆさぶり 3. 拡大要請 4. 視点の転換 5. 焦点化 6. 掘り下げ 7. 確認 8. 想起 9. まとめ 10. 評価	A. 学習課題の提示 B. 情報の提示（資料・グラフ・表） C. 実験・実習による D. 観察による E. 発問 F. 発問の繰り返し G. 問い返し H. 行動の指示（話し合い） I. 作業の指示（書く，描く，作る） J. 助言 K. 説明 L. はげまし M. 沈黙	意見発表 a. 並列的意見 b. つけ加え意見 c. 反対の意見 d. はずれた意見 e. 観察（実験・実習）したことについての意見 f. 経験発表 g. グループの話し合いの発表 h. 提案 i. 質問 j. グループの話し合い k. 作業（実験・観察・実習） l. 沈黙 m. 単純反応 私語 n. 体験 o. 連想 p. 質問 q. 解答 r. 不安
その他	N. 指名 O. 指名の切りかえ P. わりこみ Q. 賛否の問い R. 体験の有無の問い S. マネージメント T. 机間巡視	

呼ぶことにすると，これらのモジュールは，CAIシステムに必要な機能を示す概念的な基本構成要素であると言うことができる．ここで「概念的な」といったのは，それが具体的なシステムの中で実際にその機能を実現するための単位でなくてもよいということである．しかし，この基本構成要素は，CAIシステムが学習システムの教授者の側としての機能を，どこまで実現しているかを知るための分析枠組として有効である．次にCAIの類型をこの基本構成要素のもとに概観してゆく．

2.2 AFO型CAI

AFO型CAIは，フレームという単位で構成されている．フレームとは，説明，設問，予想反応，それに対するKR情報．次に制御を渡すべきフレームの情報，などを一組にした単位である．学習順序はこのフレームの系列にかなり依存するが，簡単な分岐，繰り返し，飛び越しを可能にしている場合が多い．

これはCAIの最も初期からある型であり，かつ現在でも実用システムの大部分がそれである．弊社のLEARN UPシステムを初め，国内に約20種のシステムがある．

ところで，AFO型CAIは一定の成果を収めながらも，いくつかの欠点を指摘されている．たとえば，教授内容モジュールに相当するフレームに収められた教材は，教授方略モジュールに相当する反応処理機能とともにフレーム単位で一体化されてしまうため，学習順序が固定化されがちである．分岐や後戻りなどの工夫により，学習順序の固定化はある程度避けることができるが，それはフレームの系列にかなり依存し，本質的な個別対応とはなり得ない．

また学習者モジュールに相当する機能は，学習過程の記録を主としてCMIシステムに活用される形態が多く，上述のフレーム概念からも教授＝学習プロセスの中での個別対応機能は不十分である．

2.3 生成型CAI

学習者の反応に対する教材や解答，あるいはKR情報等をあらかじめ組織的・体系的に準備されたデータベースから再編成したり，教授内容に基づいてあらかじめ設定されているアルゴリズムによって，自動的に生成するシステムを生成型CAIと呼んでいる．

生成型CAIでは，教授方略モジュールの組み立てる教授＝学習活動の展開に従って，学習者に提示する教材をデータベースから柔軟に選択することができる．また，1967年P. Suppesらによるストランド（Strand）算数ドリルなどの方式では，数学的モデルに基づいて教授方略を決定し，さらにさまざまな難易度の教材を生成している．

ストランド算数ドリルでは，学習者のレベルを達成度の関数で決定している．学習者理解モジュールを数学的モデルで実現しているともいえる．

2.4 ICAI

1970形代に急速な発展をみせたAI研究で得られた技術をCAIに適用し，従来のCAIシステムのもつ欠点を克服しようという試みがICAIである．ICAIの最大の特徴は知識の表現にある．

従来のCAIでは，教授内容は学習者に提示する形式に加工された教材として表現されていたが，ICAIでは教授内容そのものをフレーム（2.2のフレームとは異なる）や意味ネットワーク，プロダクション・ルールなどを使って表現している．学習者に提示される教材はここから生成されるが，教材の柔軟性は飛躍的に向上する．

教材が柔軟に提供できることから，教授方略モジュールの自由度も高い．いくつかのICAIシステムでは，次にどのような指導を与えるべきかという知識を，ルールによって表現することで教授方略モジュールを実現している．

さらに，学習者の理解状態を知識ベースによって表現する学習者モデルが考え出され，学習者理解モジュールによって，学習者の知識が欠落しているのか，誤った理解をしているのか，知識の適用を誤ったのか，といったところまで正確に把握できるようになった．

次章でいくつかの具体例を見ながら，その実現技術について述べることにする．

3. ICAIの実現技術

CAIにAI研究から得られた各種技術を適用する試みも，最初からCAIの基本構成要素すべてについて行われたわけではない．

最初に試みられたのは，まず教授内容モジュールであった．1970年にJ. Carbonelらによって開発が始まった南米の地理に関するCAIシステムであるSCHOLARでは，教授内容を意味ネットワークによって表現している（図2）．システムは，この意味ネットワークを用いた推論によって文章

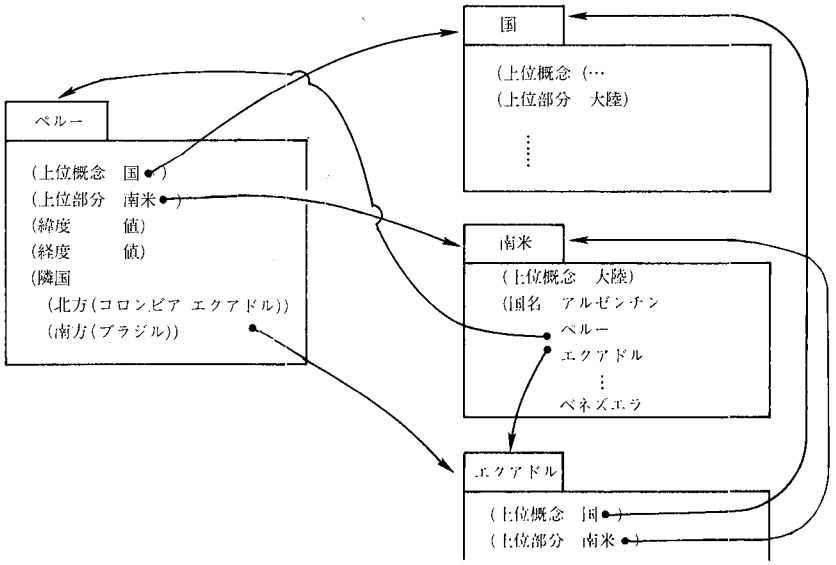


図2　SCHOLARの意味ネットワーク
Fig. 2　Semantic network of SCHOLAR

を生成し，学習者と対話を行う．意味ネットワークで知識を表現することにより，対話という非常に柔軟な"教材提示"が可能となったのである．

次の段階では，教授方略モジュールについてAI技術の適用が試みられた．A. Stevensらによる WHY（1977年）では学生との対話において，学生の思い違いに対しすぐに正解を出さずに反例を示し，学生に自ら誤りを気づかせる方式などを含む約20個のヒューリスティックを用意している．

また，学習者理解モジュールについては，1979年W. J. Clancyらによる GUIDON，1978年J. S. Brownらによる BUGGY において，学習者の理解状態を学習者モデルによって表現することが試みられている．

3.1　教授内容モジュール

教授内容を表現するとき，その対象知識をどのように表現・管理するかが一つの課題であり，各種の方法が提案されている．以下に代表的な方法とシステムを紹介する．

1)　SCHOLAR, WHY……前述のSCHOLARは，教授内容である南米の地理に関する知識を意味ネットワークで表現している（図2）．ネットワークのノードは地理に関するある対象や概念および各種属性を，アークは対象や概念間の関連を表現する．関連には上位属性（Super Attribute），上位概念（Super Concept），上位部分（Super Part）などがある．このネットワークに基づいた推論と格文法を用いて文章を生成し，学習者に提示することで対話を行う．

その後，"ソクラテス問答法"（後述）と呼ばれる教授方略を組み込み，SCHOLARを発展させたWHYが開発されているが，WHYではスクリプト風のデータ構造が用いられている．

2)　GUIDON……Clancyらによって1976年に開発されたGUIDONは，感染症の診断をするための医療データの扱い方を学習するCAIシステムである．GUIDONは，感染症の診断システムであるMYCINの専門家知識を使用している．MYCINの専門家知識は，450のプロダクション・ルールで記述されているが(表2)，これを教授内容の対象知識とみなし（Dルール），さらに教授方略を示すGUIDON個有のルール（Tルール，後述）を用いて学習者との対話を行っている．

3)　BUGGY ……1978年Brown と R. Burtonによって開発された算数の基礎技能（加算・減算）について学習者の誤りの種類と，その原因を特定するシステムである．したがって，CAIシステムではなく誤答診断システムと見なすべきであるが，その知識表現と診断機

表 2　MYCINのプロダクション・ルール[2]
Table 2　Production rules of MYCIN

RULE 050

PREMISE: ($AND (SAME CNTXT INFECT PRIMARY-BACTEREMIA)
(MEMBF CNTXT SITE STERILESITES)
(SAME CNTXT PORTAL GI)

ACTION: (CONCLUDE CNTXT IDENT BACTEROIDES TALLY . 7)

MYCIN's English translation :

IF
1) the infection is primary-bacteremia, and
2) the site of the culture is one of the sterile sites, and
3) the suspected portal of entry of the organism is the gastrointestinal tract,

THEN
there is suggestive evidence (. 7) that the identity of the organism is bacteroides.

もしも 1) 感染症が原発性菌血症で，かつ
2) 培養検体採取部位が通常無菌と考えられる部位で，かつ
3) 問題とする細菌が侵入したと考えられる感染経路が消化管であるならば，
そのとき，問題とする細菌の種類がバクテロイデスである可能性がある（確信度 0.7）

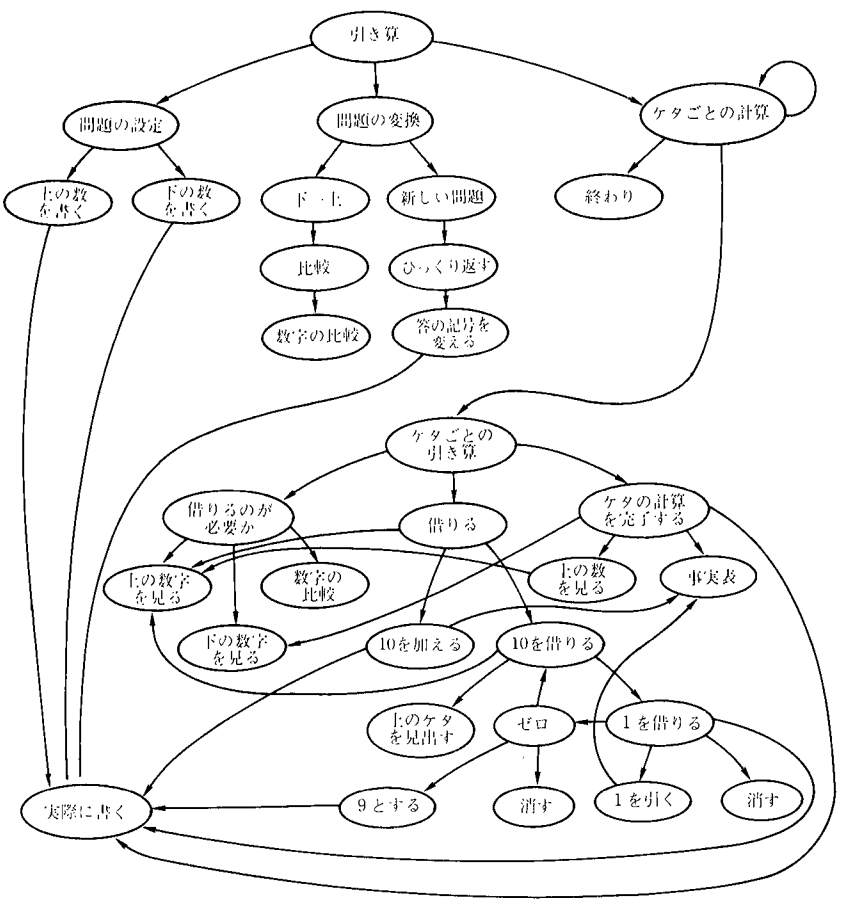


図 3　BUGGY の手続きネットワーク[3]
Fig. 3　Procedural network of BUGGY

構に独自の方法を採用しており，その後のCAIシステムに大きな影響を与えている．引続き，教授機能を付加したDEBUGGYが開発されている．

BUGGYでは専門家知識＝演算技能を下位の副技能に分解し，その副技能をノードとし，副技能間の関連(適用順序)をアークとした手続きネットワークで表現している(図3)．

3.3 教授方略モジュール

学習者を指導するためには，たとえば学習者が誤った理解をしているとき，単に正解を示すだけでは不十分である．もちろん正解を示した方が良い場合もあるが，反例を示し学習者に気づかせる，あるいは学習者にそう考える根拠を説明させることが重要であり，WHY, GUIDONではこれについて興味深い試みをしている．

1) WHY……WHYではソクラテス式問答法による対話を実現するため，約20のヒューリスティックスを組み込んでいる．ソクラテス式問答法とは，相手の誤りを直接指摘せず，いくつかの反問を与えることによって相手に誤りを気づかせようとするものである．ヒューリスティックスには，次のようなものがある[2]．

「もしも，学習者がある因果関係の説明として不必要な一つ，またはそれ以上の要素を与えるなら，その時は誤ったその要素の値を持った反例を選択し，学習者に対してなぜ因果関係はこの場合成り立たないのかを質問しなさい」

こうしたヒューリスティックスにより，システムは学習者に次々と質問してゆく．まずは特定事例に関する予測を質問し，その応答に応じてその根拠を質問する，中間原因を質問する，あるいは引き続き原因を質問する．さらに助言を与えたり，システムが規則を形成して確認をしたり反例を示す．これによって，学習者自ら誤りを気づかせる形態の学習(誘導発見型学習)が可能となる．

2) GUIDON……GUIDONの特徴の一つは，主導権混在型の教授方略を採用している点である．人間の教師と学習者のやりとりにおいて，学習者は教師に細かな質問をするばかりでなく教師の意見を聴いたり，議論の進め方や論点の変更を要求し，教師はそうした学習者のふるまいを参考にして教授方略を決定する．GUIDONでは，"HELP", "FACTORS"など選択機能を使うことによって，学習者主導の学習が可能となっている．

システムが主導権をもつ時の教授方略は，Tルールと呼ばれる規則によって示されている(表3)．Tルールは，後述する学習者モデルを参照しながら対話を生成するためのルールである．これを用いた推論によって，学習者の誤りの指摘，ヒントの提示，反例の提示が可能となる．

3.4 学習者理解モジュール

学習者の能力（学習到達度）に依存した教授＝学習過程を実現するためには，学習者の到達度や理解状況を正確に把握する必要がある．この到達度，および理解状況を表すと考えられる仮説を学習者モデル（Student Model）と呼ぶ．システムは，学習者の反応や学習履歴から学習者モデルの生成および更新を行い，教授方略モジュールは学習者モデルを参照して，次に提示する教材や質問

表3 MYCINのTルール
Table 3 T-rules of MYCIN

T-RULE 5 02	直接的な単一の既知の規則の証明
IF	1) There are rules having a bearing on this goal that have succeeded and have not been discussed,and (成功しているが，いまだ議論していないこの目標に関係している規則が存在し，かつ) 2) The number of rules having a bearing on this goal that have succeeded is 1, and (成功したこの目標に関係する規則の数が一つで，かつ) 3) There is strong evidence that the student has applied this rule (学生がこの規則を適用したという強い証拠が存在するならば)
THEN	Simply state the rule and its conclusion (単に，その規則と結論を表明せよ)

を決定する（図4）.

学習者の示す誤りには，次の3種類が与えられる.

1) 知識の欠落

2) 知識の誤り（誤った理解）

3) 知識の適用の誤り（推論の誤り）

現在提案されている学習者モデル構築の枠組と，上記誤りの種類を対応づけて紹介する.

① オーバレイ・モデル……SCHOLAR やGUIDON で用いられている学習者モデルで，学習者の獲得している知識や推論は，本来獲得しなければならないそれの部分集合とみなす．したがって，先の1)の状態は表現できるが，2)，3)の誤りを表現することができない．しかし，実現が比較的容易であること，何が理解できていて，何が理解できていないかが，明確になるという点から多くのシステムで利用されている.

② バギー・モデル……BUGGY で利用されている学習者モデルで，知識の誤りや知識適用の誤りも学習者が獲得する知識・推論であり，その中にたまたまバグが含まれているものとみなす．したがって，バギーモデルでは，1)に加えて2)，3)の誤りも表現できる．BUGGY では，算数の基礎的技能をいくつかの副技能から構成されているものとみなし（バグも一つの副技能とみなす），2)および3)の誤りは，バグを含んだ副技能の獲得ととらえる.

しかし，バギー・モデルでは，あらかじめこのバグの種類を予測し，それらを記述しておかなければならず，またバグの種類も対象領域に依存するため，汎用化するのが困難で，実現のための負荷も大きい.

③ その他……オーバレイ・モデルやバギー・モデルは，あらかじめ設定された知識や推論の部分集合として学習者モデルを生成している．近年注目を浴びている帰納推論を用いて，動的に学習者モデルを生成する試みがある.

大阪大学で開発されている Prolog のプログラミング教育用 CAI システム[4]は，学習者との対話（設問の提示と反応）を通して，学習者の応答を説明するための Prolog のプログラムを帰納推論（Shapiro の Model Inference System, MIS）を用いて生成する．このプログラムを学習者モデルとみなす．そのプログラムにバグがあれば，それを誤りとみなす．誤りの検出には PDS (Program Diagnosis System) と呼ばれる Prolog のバグを見つけるアルゴリズムを使用している.

2値論理を基礎とする Prolog という言語の特性から，知識の欠落が未学習によるものか否かの判定ができないこと，また帰納推論では，学習者のケアレス・ミスなどによる応答の矛盾の検出が困難なことなどの問題はあるものの，有望な技術と考えられる.

4. おわりに

教授＝学習過程は，人間の行為の中で最も洗練されたものの一つであり，それをコンピュータを利用して実現するためには，人間の学習メカニズ

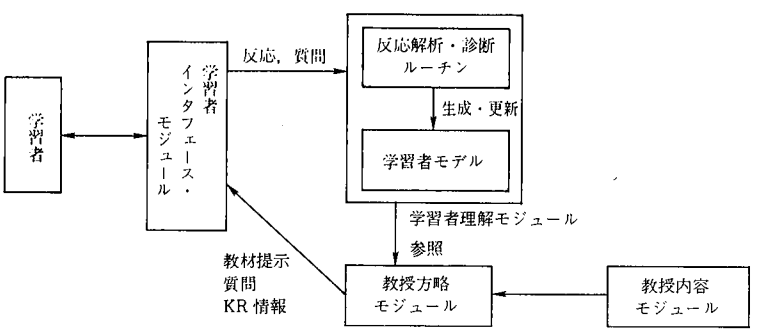


図4 学習者理解モジュール

Fig. 4 Module to understand students

ムの解明，カリキュラム開発，使いやすい学習者インタフェースの開発，信頼性・妥当性の高い到達度診断技術等，さまざまな課題を解決しなければならない．またいずれも困難な課題であり，地道な努力を必要とする．本稿では，CAIの発展過程，また人工知能の研究から得られた各種成果のCAIへの応用という観点から，CAI実現技術を概観した．日本ユニバックでもLEARN UPシステムを始めとする各種CAIシステムを開発してきており，またICAIシステムの開発も行っている．いずれ，他の機会に報告させていただきたい．

参考文献

[1] 西之園晴夫，授業の過程，第一法規 1981.
[2] A. Barr/E. A. Feigenbaum 編，田中幸吉，淵 一博監訳，人工知能ハンドブック第II巻，共立出版，1983.
[3] 安西祐一郎，佐伯 胖，無藤 隆，LISPで学ぶ認知心理学1学習，東京大学出版会，1981.
[4] 願化真志，溝口理一郎他，“帰納推論に基づく知的CAIシステム”，知識工学と人工知能研究会資料38，情報処理学会，1985.

（教育部CAIセンター）

データベース言語SQLの標準化動向

Trend of Database Language SQL Standardization

原　　潔
K. Hara

1. はじめに

データベース関連の標準化活動が，ふたたび活発化している．今日，業務システムの実現においてデータベースとコミュニケーションの技術は欠かせないものになっている．しかし，すでに多くの業務システムがデータベースを中心に構築されているにもかかわらず，最近までデータベースに関する標準は存在しなかった．特異な状態といえよう．

最初のデータベース関連の標準化活動は，1970年代におけるネットワーク型のデータベース言語の仕様開発であった．この活動は，データシステムズ言語協議会（CODASYL：The Conference on Data Systems Languages）におけるデータベース作業班（DBTG：Data Base Task Group）の1969年[1]，71年[2]の活動に始まる．今日のデータベース管理システム（DBMS：Data Base Management System）の大半を占めるネットワーク型DBMSは，これらの仕様をもとに開発されてきた．しかし，種々の理由から結局はこの仕様は標準にはならなかった．

CODASYLによるデータベース言語の体系化は，データベース・システム標準化をめぐって論議を巻きおこした．このため1972年に，米国規格協会計算機情報処理部門（ANSI/X 3：American National Standards Institute/American National Standards Committee on Computer and Information Processing）内の標準化計画委員会（SPARC：Standards Planning and Requirements Committee）にデータベース制御システム研究班を発足させた．この研究班は，1975年に中間報告を，1978年に最終報告[3]を公刊している．この報告書は高く評価され，データベース・システムの研究開発に大きな影響を与えている．たとえば，ANSI/SPARCのデータベース・システム・モデルと呼ばれているものが取り上げられている．しかし，今日では，その3層スキーマの概念だけが残っている．

国際標準化機構（ISO：International Organization of Standard）は，1981年にTC 97（Information Processing System）内のSC 5（Programming Language）にWG 5（Database Language）を設立し，データベース関連の標準化活動をふたたび開始した．その背景には，リレーショナル・データベース・システムの実働化に目途がたってきたことがあげられる．そして，開放型システム間接続（OSI：Open Systems Interconnection）の検討を行っていたSC 16が1985年に再編成され，これに伴ってそれまでのSC 5とSC 15（Labeling and File Structure）でのデータベース関連の標準化作業が統合された．新しくSC 21（Information Retrieval, Transfer and Management for Open Systems Interconnection：開放型システム間接続のための情報検索，転送ならびに管理）が編成され，データベース関連の標準化作業は，WG 3（Database）に組み入れられOSIの一部として位置づけられることになる．

SC 21/WG 3の作業項目は，現在五つである．

1) データ管理参照モデル（DMRM：Reference Model of Data Management）
2) 情報資源辞書システム（IRDS：Information Resource Dictionary System）
3) 遠隔データベース・アクセス（RDA：Remote Data Access）
4) ネットワーク型データベース言語（NDL）
5) リレーショナル・データベース言語（SQL）

本稿では，リレーショナル・データベース言語SQLの標準化の動向について紹介する．第2章でSQL標準化の背景として，他のデータベース関連の標準化作業を紹介する．第3章でSQLの規格制定の経緯を述べ，最後にSQLの拡張について述べる．

2. データベース関連の標準化作業

OSIの基本参照モデル（Basic Reference Model）は，開放型システム間の相互接続のレベルとして7層の階層を設定している．データベース関連の標準化作業が対象としている分野は，この階層の最上位層である第7層の応用層（Application Layer）に位置づけられる．

OSIの基本参照モデルの狙いは，ISO規格作成作業の各々を基本参照モデルに適確に位置づけることにより，各作業間の重複あるいは抜けがないように管理することにある．データ参照モデル（DMRM）も同じ目的を持ち，データ管理に関する種々の規格作成作業の関係を明確に位置づけようとするものである．

OSIの一部として位置づけられたデータベース関連の標準化作業は，図1のように表すことができる．

2.1 DMRM

DBMS自体がデータ管理を目的としたソフトウェアなので，DMRMでの考え方がデータベース関連の各規格作成の内容に深く関係するが，DMRMでの方法論は，まだ完全に固まってはいない．方法論としては，まずデータの種々の側面を明確にし，次にこれらとインタフェースを持つ基本的な機能を洗い出して，規格あるいは既存のデータベース管理パッケージをこれらの機能の組み合せとして表現し，DMRMでの位置づけを行うことをめざしている．

データの側面としては，これまでに現われてきている概念は，次の四つにまとめられている．

① レベル
② 表現ステージ
③ 局ステージ
④ ライフサイクル・ステージ

これらを結びつける方法論として，システム・モデリングに基づくデータフロー・ダイアグラムが提案されている[4]．

1) レベル……DBMSは，利用者の処理の対象となるデータと，そのデータの属性を定めるデータの2種類のデータを取り扱う．後者のデータのことを前者のデータに対するメタデータという．データは常に陽に存在するが，メタデータは常に陽に存在するとは限らない．このデータとメタデータとの関係をDMRMでは，レベルという概念でとらえている．レベル1が，利用者が処理するデータを示し，そのメタデータがレベル2である．リレーショナルDBMSを含め，既存の多くのDBMSは，レベル2のデータの扱いの記述を陽に行う手段を提供してはいない．

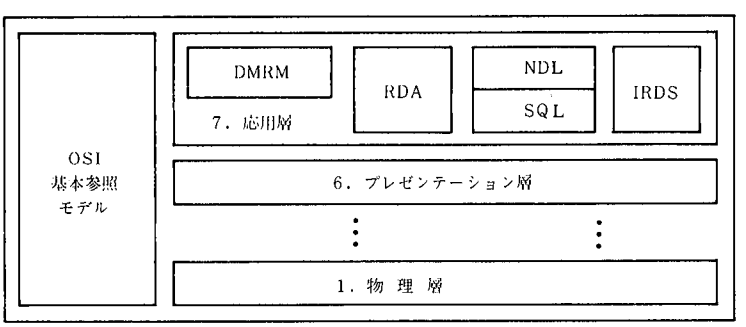


図1 OSIにおけるデータベース関連標準化作業
Fig. 1 Database standardization in OSI

SQLのようなデータベース言語においては，レベル1のデータ操作機能に加え，レベル2のデータ操作機能が統一的な仕様で提供されるべきであるという議論があり，そのような機能拡張がデータベース言語の規格化作業の中で検討されている（スキーマ操作機能）．

2) 表現ステージ……“対象とする世界”に関し，データベース設計者が“認識した世界”を情報論理ステージと呼び，それを“記号化して表現した世界”をデータ論理ステージと呼ぶ．それらのデータが，具体的に“どの物理媒体に納められているかを決めている世界”をデータ物理ステージと呼ぶ．DBMSがまず対象としているのは，データ論理ステージと物理ステージの二つである．

SQLは，データ論理ステージの記述方法として表概念を提供している．実際のデータ処理を行うためには，データ物理ステージの情報が必要であるが，このための機能は現在のSQLの規格化の範囲外になっている．

3) 局ステージ……考察の対象をデータベース全体とするのか，あるいは局所的ないしは特定のものの見方をする範囲とするのかを明確にする概念である．共有データベースでは局所ステージのデータの利用機能は，データ独立性を実現するために必要である．

SQLの提供する，実表（Base Table），導出表（Derived Table），ビュー表（Viewed Table）などは，局ステージにかかわる機能である．

4) ライフサイクル・ステージ……データの生成から消滅まで，データがとるさまざまな状態をライフサイクル・ステージという．ライフサイクル・ステージのどの状態にあるかによって，情報管理の方式は影響を受けるので，IRDSの検討においては，このステージは重要となるが，現在のところこれについては，明確な要求が出されていない．

2.2 IRDS

情報資源辞書（IRD: Information Resource Dictionary）は，メタデータの集合であり，IRDSはその管理や操作を行うシステムである．IRDの考え方は，データベース分野で知られているデータディクショナリ/ディレクトリを発展させたものである．IRDに規定される対象データの範囲は，DBMSだけではなく，あらゆるデータ管理が扱う対象を含んでいる．対象データの把握にはERモデルが使われているが，各データレベルはSQL構文で表現され，辞書自体はSQL表からなっている．SQLからみれば，IRDSは，SQLの1業務プログラムであり，このため，IRDSの利用者インタフェースをSQLで提供するという要求が出されている．

IRDSの標準化作業は，やっと緒についたばかりであり，他の規格化作業との具体的な関連づけはこれからである．分散データベース・システムにおいては，データの地理的配置に関し辞書を必要としており，分散環境の情報の管理をIRDSでどのようにするかも今後の問題である．

2.3 RDA

RDAは，OSI環境にあるデータベースを遠くから処理するためのサービスとプロトコルである（表1，図2）．RDAはOSIの応用層の応用サービス要素（ASE: Application Service Element）の一つに位置づけられる．RDAサービスは，遠隔データベースの処理のための汎用モデルを規定している．それは次のものから構成される．

① アソシエーション管理
② 資源ハンドリング
③ トランザクション管理
④ データ操作

データ操作の処理は，データを検索したり，更新したりするデータ操作言語（DML: Data Manipulation Language）文を転送し，その処理結果を返却する．このDMLは，何であってもかまわないが，現在はSQLに基づいて規定されている．

3. SQL規格制定の経緯

SQLは，リレーショナル・データベースを操作するデータベース言語の一つである．

E. F. Coddが提案したリレーショナル・データベースの理論[5]にもとづいて，1970年代の中期以降いくつかの実働化のための実験が行われた．なかでもIBM社San Jose研究所のリレーショナル・データベース管理システムSystem-Rは，その後のシステムに大きな影響を与えた．このSystem-Rのデータベース言語として提案されたものがSEQUEL[6]である．この言語の名前は，構造

表 1　RDAサービス一覧
Table1　RDA services

サービス	オペレーション	機能既要	入力パラメタ
アソシエーション管理	R-Associate	クライアントとサーバ間のアソシエーションを確立する.	プロトコル・クラス サービス・クラス コミットメント・レベル 利用者識別，課金
	R-Release	アソシエーションを解放する.	強制解放，中断
資源ハンドリング	R-Open	アクセスする資源の利用を要求する.	データ資源名 使用モード
	R-Close	資源の利用を終了する.	データ資源名/資源ID
トランザクション管理	R-BeginTransaction	トランザクションを開始する.	アトミック・アクション ID ブランチ ID
	R-PrepareToCommit	2相コミットメントの第1相処理を要求する.	——
	R-Commit	トランザクションを終了する.	——
	R-Rollback	トランザクションを取り消す.	——
	R-Restart	再開始のためクライアントとサーバの同期をとる.	アトミック・アクションID ブランチID，再開始点
データ操作	R-ExecuteDML	DML文を実行し，結果を返却することを要求する.	データ資源ID 引数型リスト 結果型リスト DML文，繰り返し数 引数値リスト
	R-DefineDML	DML文をチェックし蓄積する（コマンド・ハンドルを返却）.	データ資源ID 引数型リスト 結果型リスト，DML文
	R-InvokeDML	蓄積されているコマンド・ハンドルを実行する.	コマンド・ハンドル 繰り返し数，引数値リスト
	R-DropDML	蓄積されているコマンド・ハンドルを無効にする.	コマンド・ハンドル

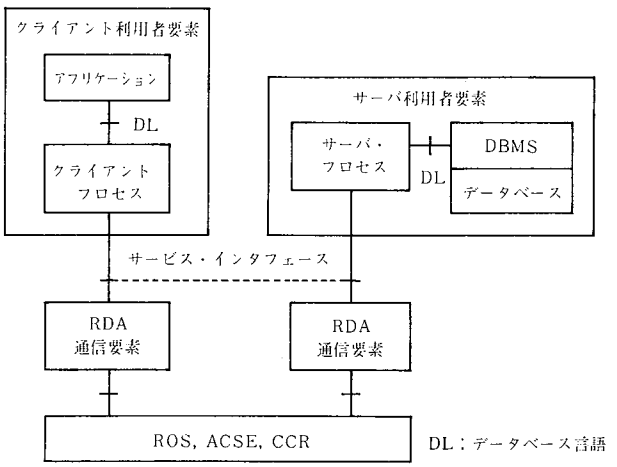


図2　RDA サービス機能
Fig. 2　Structure of RDA

化問い合わせ言語（Structured English Query Language）の頭文字をとって名付けられている．

SEQUEL言語の拡張は，その後SQLと称されるようになり，リレーショナル・データベース管理システムのためのデータベース言語として，多くの実用データベース管理システムで用いられるようになった．

リレーショナル・データベース言語の標準化に向けての作業は，まずANSIによって始められた．ANSI/X 3/SPARC内に作業班を設け，リレーショナル・データベース言語を標準化するための問題点や方向性の検討を行い，1981年に最終報告書を提出した[7]．この報告書を受けてANSIは，X 3内のX 3/H 2でリレーショナル・データベース言語の標準化の作業を開始した．

ISOでは，TC 97/SC 5/WG 5で1982年からデータベース言語の標準化の検討が開始された．1985年のSC 21の再編成により，以後はSC 21/WG 3でデータベース言語の標準化作業は進められている．

ISOやANSIでのデータベース言語の標準化作業は，ネットワーク型のデータベースに対するものと，リレーショナル・データベースに対するものの二つを検討対象とした．前者をNDL(Network Data Base Language)，後者をRDL(Relational Data Base Language）と呼ぶ．

当初NDLとRDLの両データベース言語の標準化を，同一概念で統一して開発していこうとしたが，SC 21が再編成された時期にリレーショナル・データベース言語の標準化の対象は，RDLからSQLに変更された．

実用に供されているリレーショナル・データベース言語の多くがSQLをベースにしたものになっているということが，SQLを標準化の対象にした理由である．リレーショナル・データベース言語がSQLになることにより，NDLとの統一化の議論は薄らいでいる．

ANSIでは，1986年11月にデータベース言語SQLをアメリカ国内規格ANSI X 3. 135-1986および連邦情報処理規格（FIPS：Federal Information Processing Standard）として交布した．

ISOでは，1985年5月にSQLの国際規格案（DP：Draft Proposal）の登録が行われた．1986年11月に国際規格原案（DIS：Draft International Standard）DIS 9075となり，1987年2月にISOの中央事務局に送付され，1987年6月にISO規格として交布された．

日本においては，ISOの規格化作業と併行してデータベース言語の規格化作業が行われており，1987年3月にJIS原案の作成を完了している．早ければ1987年末にはJIS化される見込みである．

CODASYL仕様にもとづくネットワーク型データベース言語の標準化が遅れたことの反省として，SQLに対しては，早期に標準化することを第一の目標に作業を行ってきた．このため最初のSQL規格は，既存のSQL言語の共通部分を採用する傾向が強く表れているので，いくつかの問題点が残されている．

4. SQL言語の拡張

SQL言語の規格化とともに，いくつかの問題点の指摘や拡張に関する提案がなされている[8]．SQL言語の規格化を早期に実現することと，機能拡張の要求に応えるために，SQL規格に対しSQL補遺1[9]の制定を行うことになった．

Coddの提案したリレーショナル・データベースをより忠実に表現するために，SQL補遺1には主な機能として，次のものを含める予定である．

1) 一意性制約の機能拡張（PRIMARY KEY機能）
2) 表間の参照制約の機能付加（FOREIGN KEY機能）
3) 表の行に適用される検査機能（CHECK条件）
4) 表の中に行を格納するとき，値を指定しないときにとられる列の値（DEFAULT句）

標準化の目的の一つとして，今後の製品化に対する方向を与えることがある．この意味からリレーショナル・データベース管理システムをより良いものに導いていくために，SQL言語の改良あるいは拡張の努力が，現在精力的に進められている．

これらの改良・拡張はSQL 2[10]として検討されており，このSQL 2により現在のSQLに対する批判に応えようとしている．

なお，SQL 2で検討されている主な機能には次のものがある．

1) データ型の追加
2) エラーコードの規格化
3) トランザクション処理のための機能拡張
4) カーソル位置づけ機能の拡張

5) スキーマ表に関する規定

6) 集合演算の指定

SQL 2に対しては，多くの改良・拡張案が提案されており，この仕様の開発には時間がかかることが予想される．提案の多さから言語仕様が拡散する危険性があり，1987年6月のISO東京会議において，SQL 2に核という概念を採用することが決定された．これにより，まずどの機能を核として位置づけるかが課題として検討されている．

SQL 2制定に向けて，まだまだ多くの論議が必要とされる．情況を注視し，より良いSQL 2の制定に向けて，規格化作業に積極的に携わっていく所存である．

参考文献

[1] CODASYL, Data Base Task Group Report to the CODASYL Programming Language Committee, Oct. 1969.

[2] CODASYL, Data Base Task Group Report to the CODASYL Programming Language Committee, Apr. 1971.

[3] ANSI/X 3/SPARC, "The ANSI/X 3/SPARC DBMS Framework Report of the Study Group on Database Management Systems", Information Systems 3, 3, 1978, pp. 173～191.

[4] ISO/TC 97/SC 21/WG 3, Technical Outline of the Model of Data Management, N 241, Sept. 1986.

[5] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", CACM vol. 13, No. 6, June 1970.

[6] D. D. Chamberlin, "SEQUEL : A Structured English Query Language", Proc. 1974 ACM SIGMOD Workshop on Data Description, Access and Control, May 1974.

[7] ANSI/X 3/SPARC DBS - SG, Final Report on Relational Database Task Group, ANSI SPARC-81-690, Sept. 1981.

[8] C. J. Date, *A Guide to the SQL Standard,* Addison-Wesley Publishing Company, 1987.

[9] ISO SC 21 WG 3, Proposed Draft Addendum 1 to DIS 9075 Database Language SQL, 1987.

[10] ISO SC 21 WG 3 N 325, Database Language SQL2, Apr. 1987.

[11] 情報処理学会，情報処理学会研究報告 87-DB-59, 1987年6月.

（商品開発本部ソフトウェア二部）

Donald E. Knuth 著

"The METAFONT book"

Addison Wesley Publishing, xi+361 pp., 1986.

本書は，"The Art of Computer Programming"（「基本算法」，サイエンス社）の著者として知られるStanford大学のD. E. Knuth教授による本で，フォントを作るためのシステムMETAFONTのプログラム言語とその処理系について述べたものである．METAFONTは，彼の有名な清書システムTEX姉妹のプログラムで，そのフォントを作成するのに用いられる．

なお本書は，計算機と植字シリーズ（COMPUTERS AND TYPESETTING SERIES）の5冊本の1冊で，同シリーズでは本書のほかにThe TEX Book, TEX : The Program, METAFONT : The Program, Computer Modern Typefacesが刊行されている．

さて，コンピュータによるフォントの作成について馴染みのない方のために，少し解説を加えてから本題に入ることにする．

METAFONTでは，フォントを標準の直交座標のグラフ用紙を使い表現する．このグラフ用紙の1ますをピクセルといい，白か黒のいずれかの色を持つ．したがって，単位長(inch, millimeter)当たりのピクセルの数はフォントの解像度を意味する．METAFONTでは，座標を解像度に依存する量で相対化できるので，解像度の変更は容易である．そして，ペン先の形状（大きさ・姿），ペンの傾き，筆法を指示し，ちょうど習字の練習のように手続き的にペンの動きを与え文字を描く．

なお，曲線の場合は点と接線ならびに線の張り具合によって定義する．基本はペンを移動させながらペン先によって塗りつぶすことにある．

本書を読むに当たっていくつかの注意が必要である．その一つは植字の専門用語が随所に出てくることであり，大きな英和辞典をもってしても解説がない用語がある．これを気にしたらまず初めのほうの章であきらめねばならない．次は，METAFONTプログラム言語である．実際にフォントを作るためにMETAFONTでプログラムを書くことになるのだが，これが伝統的な手続き型言語と少々違うので，例題を読む時は最初はとまどうかも知れない，しかし，初めからていねいに読むに従ってKnuthの考えが解ると同時に，その言語で書かれたプログラムが理解できるようになる．

さらにもう一つは，METAFONTプログラムでは，フォントを作るための非常に詳細な記述を要求されるということである．このため，マクロの作成などの省力化手段が用意されていることが，例題を根気強く読むと解る．基本は非常にきめ細かい指示を与えることである．最後に，高校で習う程度の初歩的なベクトル演算の知識が必要である．

それでは，本書の構成を簡単に紹介する．付録を入れて，全体で三部に分かれる．まず第一部は，METAFONTの基本的な概念と簡単な紹介（第1章から第5章），第二部はMETAFONT言語／処理系の解説（第6章から第27章），第3部は付録（A～Jの10項目）である．

第1章はMETAFONTの由来を述べている．第2章から第4章はフォントを作るための基礎となる座標，曲線，ペンについて説明している．第5章はそれまでの知識で理解できる簡単な例によって，実際にMETAFONTプログラムを実行してみせている．いわば処理系の説明である．

第6章はMETAFONTの語彙，第7章から第24章はMETAFONT言語の説明である．第25章はMETAFONT言語での式のまとめで，第26章はMETAFONT言語の言語のまとめである．第27章はMETAFONT処理系でのエラーとその対処法について説明している．

付録には，A：各章にある問題の解答，B：plain METAFONT（初めから提供されるマクロ）にある基本演算，C：METAFONTの高移植性のための内部コードの定義，D：METAFONT言語設計者の考えなかったMETAFONT言語の利用法，E：METAFONTで作ったフォントとそのプログラムの例，F：METAFONTの，たとえば文字の境界を与える箱に関する数値や隣の文字との間隔というような，フォントに付随する数値の設定法，G：METAFONTの出力であるGeneric Font Fileの形式，H：実際に試しに印字する時に使うプログラム（GFtoDVI）への命令の説明．なおこのGFtoDVIはTEXを経由しないで直接

dviファイルを作るプログラムである，I：本書に出てくる記号や単語などの索引，J：TEXコミュニティへの案内，などが含まれている．

なお，以下では筆者が本書を読んでいるうちに興味を覚えたところをいくつか紹介する．

たとえば，メモリ内のピクセルの表現方法（第13章）や，ピクセルによる表現で不可避な曲線や斜線の見ずらさへの対処（第24章）など，本書は単にフォントを作ろうという人たちの他に，絵書き道具を作ろうという人たちにも参考になる内容が多い。

自明なことであろうが，フォントが同じ書体のものかどうかの判定は，数学の理論のように証明できない．目で確認する必要がある．また，フォントが良いか悪いかは，そのフォントの設計者に依存する．このためには，多くの試し書きが必要である．

フォントを作るとき，できるならばペンの太さや傾きを変えた多くの試し書きがほしいが，METAFONTではこのような試し書きが可能である．また，文字の大きさを変えて試し書きができるといったフォント作成上のさまざまな支援機構がMETAFONTにはある．ただし，これらはすべて利用者がマクロなどを使い，自分で手続きを作ることになっている．

なお，私見ながら毛筆のフォントがあるとおもしろいとひそかに思っている．毛筆のフォントをうまく作る手段がほしいところである．残念なことにMETAFONTは，アルファベット文化圏の産物であり，毛筆のフォントをこのMETAFONTで作成できるかどうかは未知数である，しかし，楷書などは見込みはあると思う．

本書は，卓上印刷(Desktop Publishing)やCAP(Computer-Aided Publishing)に興味を持っている人，フォントに興味を持っている人，ソフトウェアの開発で使うさまざまな記号(文字，図など)を用意しようという人にお勧めしたい．

なお，TEXには参考書や簡単な紹介文献がいくつかあり，大野[1]を参照するとよい．METAFONTの日本語の資料は，現在のところTEXユーザ・グループ例会資料がある程度である．

[1] 大野義夫，"TEX入門1，TEX概説"，ビット，1987年6月号．

（ソフトウェア生産技術1部　加藤潤三）

〝技術文書の書き方〞に関する本あれこれ

文章技術・作文技術に関する本は実に多い．これまでに，優に100種を越える本が出版されているのではないだろうか．その中で，技術的文章の書き方あるいは技術文書の書き方に関するものとなると，比較的少ない．しかし，それでも1割とはくだらないであろう．

誰もが文章を書く時には，間違いなく相手に通じるようにと念じている．しかし，現実の文章がそうなっているとは限らない．そこで作文技術の本を読み漁ることになる．また，伝えたい情報を持ちながらも，まとめあげに自信が持てず，試行錯誤を続け，その結果ついつい書店に行っては手に入れてくる．気がつくと自分の書棚に何冊か並んでいることになる．

さて，本稿では，これら“技術文書の書き方”に関する本の中から筆者の目に触れ，一読されてはいかがかと感じたものをいくつか紹介する．

☆

まず，最初に挙げたい書は，木下是雄著〝理科系の作文技術〞中央公論社（1981）である．いまや，この種の古典ともいえる地位を確保し，多くの読者を得ている本である．いまさら紹介でもないかもしれないが，最も有用の本として取り上げないわけにはいかない．

本書の著者は「スッキリと筋の通ったものを書けるかどうかは，自分の書いたものをきびしく見直す能力と，何度でも書き直す根気とにかかっている」という．〈文章の書き方〉として，この言葉ですべてといっても良いのではないか．しかし本書は，決してこの言葉だけで突き離しているわけではない．見直すための〈作文の原則〉と，書き直すための〈心得〉が随所表れている．

たとえば，「目的規定文(主題に関して主張または否定しようとする意思を明示した文）を書き，それからその目標に収束するように文章全体の構想を練ることが必要だ」と述べ，そのために注意すべきことを挙げている．いわく，〈パラグラフを意識して文章を構成すること．逆茂木型（話の本筋から大枝・小枝の修飾句・節・文が入り乱れて茂り，主題が不明確となった状態）の文や文章を

書かぬこと．飛躍のない記述にすること．はっきりとできるだけ言い切ること．事実と意見を明確に区別して書くこと．短い文で文章を構成すること．破格の文（言葉どおしの関係がきちんと保たれていない文章，あるべき言葉が欠落している文章，言葉のつながり方がねじれている文章）を排除すること．誤解の余地のないように書くこと．なくてもすむ言葉はすべて書かないこと．能動態で書くこと〉．

いずれも，特別に変わったことが書かれているわけではない．しかしつい忘れてしまう注意を，例文を添えて述べているのが有難い．

また，テクニカル・ライタという言葉を実質的に紹介した著者だけに，次のような記述もある．「説明書は，誰が，どういう目的で，いつ，どこで読み，どう利用するのか考え，それに適するように情報を選択・配列することが第1……．それには，いろいろ想定した場面で，いろいろ想定した読者が，何をどの順序で知りたがるかを考えてみる思考実験をしたかによって，説明書のねうちはおよそ決まるのである」とし，「説明書は，一種の教育を目的とするものである．教育のための最良の道は，いちばん無駄の少ない，繰返しのない，論理的に筋が通った道とは限らない」と諭す．

薄い本ではあるが，改めて教えられることの多い，示唆に富む本である．

知られるとおり木下氏は物性論の学者で，〝物理の散歩道〟などの著書で有名な著者群ロゲルギストの一人である．寺田寅彦以来，物理学者には文章を良くなす人が多い．それは，文章に関心の強い人が多いせいだろうか，

日本物理学会には〝JOURNALの論文を良くするために〟第2版（1969）という，知る人ぞ知る書がある．本書は，物理学会誌に連載されてきた〈Jounalの論文を良くするために〉というシリーズを中心にまとめた別刷集である．同学会の欧文誌の論文の質を高めようと企画され，第一級の研究者・教育者16名が寄稿したものである．それだけに，日本人英語のくせについて注意を喚起し，誤りを正したものが多いが，広く良い論文を書くための〈心構え〉や〈ノウハウ〉が随所に書かれている．たとえば坪井忠二氏の，次のような指摘は，添削例とともに挙げられているだけに有用であろう．

〈形容詞や副詞の位置により，文章がわかりやすくも，わかりにくくもなる．形容詞や副詞はそれらがかかる言葉にできるだけ近付けるようにせよ〉，また翻訳の場合などで，〈関係代名詞がないのに，外国的な表現が使われると，たいへんわかりにくい文章になる．関係代名詞直訳型をやめて，文章の構成を叙述的なものにするよう心掛けよ〉，とすすめている．そして最後に，〈良い論文を書くには，自分の言いたいことを間違いなくはっきりと伝えるよう努力するという，決まりきったことを，本気になってやるかやらないか，その心構えが大切だ〉と言っている．

本書は独立して発表された論文を集めたものではあるが，各著者の熱意と編集者の工夫された索引により，まとまりあるものになっている．B5判7ページ約550項目の索引が「英語で何というか」，「英語でどう使われているか」，「表現上の一般的注意」，「文法的な問題」，「日本式英語」，「論文を書くときの注意：一般的注意」，「論文を書くときの注意：科学論文の場合」，「編集的なこと」，「英語学習法」の九つの大項目に整理・分類されて，参照の便を図っている．図らずもというべきか，十分に意図してというべきか，技術文書に必須な索引の有用性を実地に教える型になった．

同じ学会企画のものに，電子通信学会編，〝学術論文の書き方・発表の仕方〟がある．「論文作りの技術とその前提について」，「わかりやすい論文を書くために」，「読みやすい論文の書き方」など，電気通信学会誌に掲載されたものを再録している．先の〝JOURALの論文を良くするために〟よりも全般的である．とくに，中村幸雄氏の「読みやすい論文の書き方」は，〈いかに組み立てるか〉として，事前の準備の仕方から，論文構成の標準形の提示がなされている．また，作文上のノウハウも，実際的例文をのせて示している．

たとえば，〈“主体をハッキリ示さないと論理が不明になるから，主語を必ず書かなければならない”というのは誤解である．前半は正しいが，……．日本語では……，“主格が述語のなかに取り込まれ”ているから，英語の学術的文章をまねて，受動体を濫用するより，主語を明示しなくても能動体の文章を書けばスッキリすることが多い〉．〈修飾の連文節がながながと名詞の前につく翻訳調の文章は，“センテンスを分け，しかも主動詞を連用形にして，形のうえで一つのセンテンスにする方法”を工夫せよ〉．〈“に”，“で”などの助詞は，内

容に応じて“によって”,“に対して”,“において”,“であるから”,“の形で”とかに，言い換えることに気を使う必要がある〉.

さらに，同種のものに宮川松男著〝技術者のための文章作法〟日刊工業新聞社（1979）がある.「機械設計」誌（日刊工業新聞社）に連載した講座で，先の二つと異なり一人が著わしたもので，技術文書を書く意義から，まとめ方，原稿記入作法まで，まとまりの良い少ページの本である．技術原稿を10種類に分類し，それぞれの構想やまとめ方の大要や注意すべきことを示しているが，一度目を通しておくとよいのではないか.

山中秀男著，〝新版，技術文書のまとめ方〟，東洋経済新報社（1971）は，著者の20年来の執筆経験と教育経験に基づいて書かれた本で，適用範囲が広い．上巻は基礎編で，社会人として誰でもが身につけておくべき文章上の基本的な知識が体系的に記述されている．4章の〈用字，用語の使い分け〉は，一読すれば，改めて得るところが多いと思われる．下巻は応用編である．論文のまとめる手順，作成上の注意，構成，式や図・表の取り入れ方など，技術論文を中心に報告・展望・論文抄録・文献紹介・議事録・解説・取扱説明書・特許明細書などのまとめ方を解説している.

最後に，技術文書の一つであるマニュアルについて書かれた本を二冊紹介したい.

文書はまず読まれなければならない．いくら内容的に価値があったとしても，読まれない文書では意味がない．コンピュータの普及に伴って生まれた新たな利用者にとっては，いままでのマニュアルはまさに意味のない文書であった．彼らは，読み始めたマニュアルがわからなかったり，役立ちそうにないと感じたりしたら，即座にそのマニュアルを放棄する．情報社会を一方で支える彼らを無視できないメーカは，彼らを引きつけ，自社製品の有位性を侵透させるべく，読まれるドキュメント作りに目を向けはじめた．制作者達も，読んでくれないものを作るばからしさに耐えられず，読まれるマニュアル作りに動き出した.

このような状況の中で出版されたものに，高橋照男著，〝わかりやすいマニュアルの作成法〟，日経マグロヒル社（1985）がある．テクニカル・ライタの必要性を説き，仕事の範囲を定義し，彼らが身につけておくべき技術・知識を述べている.

技術文書を書こうとしている一般の方々にとっても，この技術・知識の部分「第3章『わかりやすいマニュアル』の文章を書く前に」，第4章『わかりやすいマニュアル』の文章はこう書く」は，十分に役立つ内容である.

もう一つは，海保博之ほか著，〝ユーザ・読み手の心をつかむマニュアルの書き方〟，共立出版（1987）である．認知心理学を専門とする大学教授が〈人に何かをしてもらうための表現，人にわかってもらう表現はどうしたらよいか〉ということをベースに書いた本である．本書は，〈わかりやすいマニュアルを書くには，「ユーザの心理と行動のくせを知れ」〉と主張している．この主張に基づき，何を注意し，どのような工夫をすると良いかを認知心理学のデータで裏打ちし，具体的に述べているユニークな本である.

☆

ここで紹介した本以外にも，良いものは多くあろう．しかし，この種の本を読んだからといって文章が上手になったり，良い文書が作れたりするわけでないことは明らかだ．〝畳のうえの水練〟ではないが，実際に文章を書いてみなければ，どうにかなるわけではない．先に引用したように，木下氏の「スッキリと筋の通ったものを書けるかどうかは，自分の書いたものをきびしく見直す能力と，何度でも書き直す根気とにかかっている」がすべてであろう．ただし，自分の書いたものを見直すときには，上記の本は大いに有用なことも確かである.

（技術情報サービス部　青柳幸久）

87年度発表の製品の中から主なものを選んで紹介します．各製品の詳細についてはマニュアル等をご参照ください．

●通信制御装置 DCP/15 II

DCP/15 II は，通信制御装置DCPファミリ(DCP/15，DCP/20，DCP/40)の新機種で，最大接続回線数24，処理能力DCP/15の2倍という価格性能比のよいシステムである．

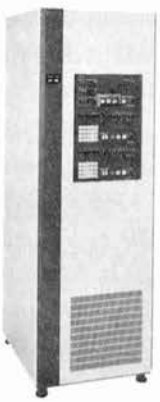
DCPファミリは，ユニシスのネットワーク・アーキテクチャDCA (Distributed Communication Architecture)に基づくネットワーク制御装置で，シリーズ1100および2200/200シリーズのフロントエンド・プロセッサや，リモート・コンセントレータ，ノーダル・プロセッサとして使用できる．

DCPファミリによって，ANSER網やビデオテックス網などの各種データ網，DDX網やVENUS網などの国内外ネットワーク，異機種・異企業間のネットワークにも対応可能で，拡張性の高いネットワークを実現できる．

(資料コード：472825390-0)

●パケット・ネットワーク・プロセッサ PX7250

PX7250は，CCITT勧告X.25の76年版の基本通信プロトコルを採用したパケット・ネットワーク・プロセッサである．情報のパケット化，蓄積・経路制御・速度変換・プロトコル変換に加え，システムの二重化構成も可能で，効率および信頼性の高い伝送を実現できる．特徴としては，①モジュール当たり最大60回線およびノード当たり最大180回線の接続（二重化構成の場合）が可能で，回線の集約により回線コストを大幅に低減できる．②各パケット交換局でのデータ通信状況を遠隔監視できるので，集中管理と分散管理のいずれの形態もとれるほか，障害時の自動切替機構によって，ネットワークの稼働率を高められる．



③ CCITT X.25 76年版の採用により幅広いコンピュータ，および端末の接続が可能であるなどがあげられる．

(資料コード：481203806-0)
(資料コード：481205833-0)
(資料コード：481205834-0)

●構内デジタル PBX システム VX6000 シリーズ

VX6000シリーズは，VX6110とVX6120の2機種からなる，蓄積プログラム制御時分割交換方式の高機能デジタル交換機である．VX6000シリーズによって，高速デジタル網を含めたデータと音声の統合ネットワークの構築を容易にするとともにネットワーク・コストを低減できる．特徴としては，①回線容量は，最大 内線48回線/外線16

回線（VX6110）と最大 内線 120 回線/外線 32 回線（VX6120）で，回転ダイヤル式・プッシュボタン式・多機能電話機が接続可能である．②中継台・局線表示盤・多機能電話機・個別着信・ダイヤルイン等の中継方式に対応できる．③停電時には自動的に予備電源に切り替わり，30 分(VX6110)と 180 分（VX6120）の継続運転が可能である．④デジタル・インタフェース・ユニットを用意しており，同期式端末と非同期式端末を最高 9600 bps の伝送速度で接続/交換し，簡易構内 LAN としても利用できるなどがあげられる．

（資料コード：081891009-0）

●ソリッド・モデリング・システム UNICAD/SOLID

UNICAD/SOLID は，立体の情報を持っているソフトウェアで，シリーズ 1100 および 2200/200 シリーズで稼働する．UNICAD は，これで設計の初期段階から製造まで一貫したシステムになった．

意匠設計，構造解析モデル，重量計算など各種アプリケーションに UNICAD/SOLID で作った形状データを使用できる．従来のソリッド・モデリング・システムには，処理速度が遅い，自由曲面の扱いができない，などの問題があり実用上の障害となっていた．

UNICAD/SOLID は，これらの問題点を解決するとともにシェイディング(陰影による立体表示)等の機能を装備した実用システムを実現している．UNICAD/SOLID は，自動車・精密機器・車輌・電機機械メーカでの意匠設計・機械設計など幅広い分野で利用できる．

UNICAD/SOLID の特徴としては，①形状の局所変更機能による速いレスポンス・タイム，②複合自由曲面によるソリッド・モデリング，③モデリング/マンマシン用の高級処理言語を用いたアプリケーションに応じた専用システムの構築，④シェイディング，透明感表示，隠線・隠面表示，アンチ・エイリアシング（斜線がギザギザになるのを避ける技術），テクスチャ・マッピング（面に模様を張りつける技術）等の立体表現機能，⑤日本ユニバックの汎用設計支援システム UNICAD/SURFACE（サーフェス・システム），UNICAD/DESIGN（ワイヤ・システム），UNICAD/FEM（構造解析プリ/ポスト・プロセッサ APPEX），複合曲面加工ソフトウェア SCULPTOR 等との連動による形状創成から詳細設計，さらには NC データまでの一貫した CAD/CAM システムの実現などがあげられる．

（資料コード：081871910-1）

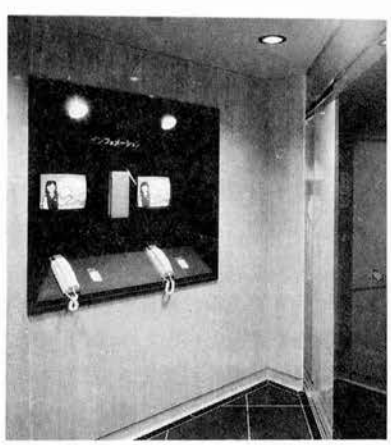
●プラスチック射出成形用流動解析ソフトウェア MELT FLOW

MELT FLOW は，金型の溶融ポリマーの充填から固化・離型までの一連の流れをシミュレートし，冷却解析までを一貫して行うシステムで，宇部興産㈱社と共同で開発された．

このシステムを利用することによって，商品開発の段階で金型を何回も作り直す必要がなくなり，効率が大幅に向上する．

MELT FLOW の特徴として，①樹脂のメルト・フロント（最先端部）に発生する要素の部分充填を考慮した有限要素法解析によってメルト・フロントの進行状況を詳細に追跡できる．②ウェルド・ラインの位置の把握，ショート・ショットの可能性の予測，射出条件の設定など成形特性の予測精度が向上する．③成形品の離型までの温度・圧力・速度の時間的変化を追跡しているので，離型後の変形解析につなぐデータが自動的に用意できるなどがあげられる．なお，MELT FLOW は，シリーズ 1100 および 2200/200 シリーズで利用できる．

●無人受付システム



このシステムは，インテリジェントビル・エンジニアリングの一環として開発されたもので，二つのタイプがある．

タイプ1は，ビデオテックスをベースに開発されたもので，テナントあるいは社内の訪問先部所をテレビ画面の案内に従って選択し，自動的に電話を接続するシステムである．既設の電話を使用するため配線工事が不要で簡単に設置できる．オプション機能として，企業および製品の紹介等を映像装置と連動して表示させることもできる．

タイプ2は，金融機関を始めとするセキュリティ管理を重視する企業に適したシステムであり，(株)泉創建エンジニアリング社と共同開発された．このシステムでは，来客の姿を応待者がテレビ・モニタで確認のうえ，来訪者に応じた適切な応答と必要な部所への案内ができる．オプション機能として映像情報を表示することも可能である．

この二つのタイプを使い分けることにより，事務所ビルはもちろん，研究所，工場，マンション等各種の用途の建物への利用が可能である．

(資料コード：081201828-0)
(資料コード：081201830-0)

●シリーズ8Uファミリ

Uファミリは，32ビットのアーキテクチャを採用したニュー・オフィス・プロセッサで，従来のオフコン機能に加えて分散処理と統合OAの両機能を強化したシステムである．Uファミリは，次の6機種，つまり①U 1000(Uファミリへのエントリ・マシン)，②U 1500（デスクトップ・タイプの強力オフィス・プロセッサ)，③U 3000(充実したデータベース/ネットワーク機能の中型オフィス・プロセッサ)，④U 3500(統合OAシステムの中核機としてあらゆるニーズに応える中型オフィス・プロセッサ)，⑤U 4000(汎用コンピュータに匹敵する処理能力の大型オフィス・プロセッサ)，⑥U 8000(汎用コンピュータの先進機能を装備した超大型オフィス・プロセッサ)，で構成されている．

Uファミリの主な特徴は，①低価格のエントリマシンU 1000から超大型オフィス・プロセッサU 8000まで，一貫した本格的32ビット・アーキテクチャの採用，②OSとしてUNIX*をベースとしてオフコンの使いやすさを継承しながら，数々の機能拡張を行ったDPS 10の採用による下位機から上位機までの完全な互換性，③マルチワークステーションM 8500シリーズからOA-LANに接続された複数のUファミリをアクセスするマルチホスト機能，④Uファミリで稼働しているプログラムから他のUファミリのリレーショナル・データベースを容易に利用できる分散リレーショナル・データベース機能，⑤Uファミリから遠隔地のUファミリを保守・管理する遠隔システム管理機能，⑥Uファミリに接続された端末から入力されたトランザクションを他のUファミリで処理し，結果をその端末に戻す分散トランザクション処理機能，⑦マルチワークステーションM 8500とUファミリ，汎用コンピュータとUファミリとを接続し，階層化された垂直分散システムを容易に実現するマイクロ・メインフレーム・リンク機能，⑧マルチワークステーションM 8500側で作成した文書・図形・イメージ・表データ・グラフ・データ等のマルチメディア文書を，Uファミリ側のファイル上に一元的に管理する電子キャビネット機能，等である．

* UNIXは，米国AT&Tのベル研究所が開発し，AT&T社がライセンスしているオペレーティング・システムである．

(資料コード：081751110-0)
(資料コード：472755331-0)
(資料コード：472755332-0)
(資料コード：472755333-0)

●ビジネス・ワープロUW7E

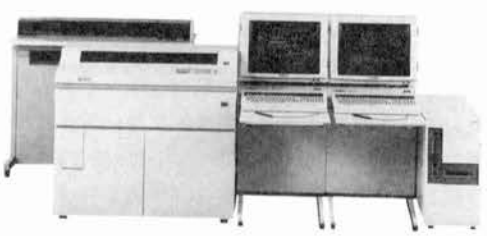
UW7Eは，ビジネス・ワープロにパソコン機能とCAIシステムLEARN UPによる自己学習機能を付加したシステムで，MS-DOS(V 3.1)の標

準搭載によって本格的ビジネス・パソコンとしても利用できる．

UW7Eの特徴は，ワープロ機能としては，①多文節変換/自然かな漢字変換，②豊富な漢字辞書(10万語の国語辞書)，③豊富な文字サイズ（7種)，④多彩な文字修飾等(拡大文字のスムージング，白抜き文字，太字斜体，4倍角の影付文字等)，1/2縮小表示，レイアウト表示，印刷中の編集，演算処理，多彩なグラフ機能（基本グラフ，QCグラフ)がある．このほか，パソコン機能としては，①統合OAソフトウェアSuper REPOの利用，②自己学習機能の提供，③ワープロ文書とのデータ変換などがある．

（資料コード：081811919-0)

●図面自動入力システム Uni-Mudams



Uni-Mudamsは，図面をそのままスキャナから入力し，ベクトル・データに自動変換するシステムである．従来のシステムでは，画像データの処理専用ハードウェアを用いるため高価である，再現品質の悪さから編集作業が必要となるといった問題があった．本システムでは，東京大学生産技術研究所で考案された新しい理論（AI-MUDAMS）に基づき，この問題点を解決している．また，汎用のエンジニアリング・ワークステーションの採用によって，高い再現品質と高速性を達成しながら従来のシステムの約1/2という低価格を実現している．

Uni-Mudamsの特徴として，①複雑な交差部分や結合部分の処理を新しいアルゴリズムと独自の結合処理によって解決しているので，優れた再現品質を実現している．②ソフトウェアで処理するため，ユーザ独自のコマンドや図面の認識などに関しカスタマイジングできる．③わかりやすい日本語画面メニュー，および輪郭線と芯線の重ね合わせ表示によって編集操作が容易である．④プロッタ，磁気テープ，通信インタフェースなどを支援しており，CAD/CAMシステムとのインタフェースやネットワークへの接続が容易である．⑤UNIX搭載の汎用エンジニアリング・ワークステーションを採用しており，低価格であると同時にCAD/CAM，CAEなどのソフトウェアや構造解析のライブラリ等の市販のソフトウェアが使用できる．⑥編集用グラフィック・ディスプレイの複数台接続やNFS LANによる拡張が可能であり，システムの拡張性が高い，等があげられる．

なお，本システムの適用分野は，地図，建築躯体図面，店舗設計図面，住宅間取図，電気回路図，機械図面，アパレル型紙，アナログ計測データ，さし絵等である．　（資料コード：084741006-0)

ISSN 0289-6257

技　報
UNIVAC TECHNOLOGY REVIEW
No. 15

発	行	日	昭和62年11月30日
編	集	人	柳 生 孝 昭
発	行	人	富 田 和 夫
発	行	所	日本ユニバック株式会社 東京都港区赤坂2-17-51 〒107 TEL(03)585-4111（大代表）
頒	布	価	格	1,500円
印	刷	所	三美印刷株式会社