DB24－1
（1975. 9. 11）

COBOLデータベース機能

植村俊亮（電子技術総合研究所）

0. はじめに　CODASYL（The Conference on Data Systems Languages）のプログラム言語委員会（PLC；Programming Language Committee）は1975年4月にデータベース機能をCOBOL文法に正式に組み込むことを決定した．このための準備作業は1966年ごろに開始されたので，10年目にこの作業が一応完結したことになる．このデータベース用共通言語の開発の歴史や概要については，すでに報告した[1b]ので，ここでは簡単な年表を図1に示すにとどめる．

本稿では，この言語体系のきわだった特徴，これまでに議論ないしは批判の対象になった言語仕様のいくつかに焦点をしぼって，COBOLデータベース機能の分析を行なう．

1966年	CODASYL PLC（当時はCOBOL小委員会）リスト処理作業グループ正式に発足（非公式には1965年から活動）
1967年	データベース作業グループ（DBTG）と改名
1968年	「データベース操作のためのCOBOLの拡張」を発表．COBOLにはデータベース機能が必要であることが認識された．
1969年	「CODASYLプログラム言語委員会への報告」を発表．DBTG'69提案と略称される．
1971年	同名の報告書の71年4月版を発表[1]．DBTG'71提案と略称される．CODASYL PLCが基本線を承認した．
1974年	CODASYL Data Description Language Journal of Development, June 1973[2] が刊行された．DDL JOD と略称される．スキーマDDLがCODASYLの正式文法になった．
1975年	CODASYL COBOL Journal of Development[3] にデータベース機能を導入．サブスキーマDDLとDMLとがCODASYL COBOLの正式文法になった．

図1　CODASYLによるデータベース用共通言語開発の歴史

1. スキーマDDLとCOBOLデータベース機能

1.1 概説　CODASYLによるデータベース用共通言語は三つの要素で構成されている．すなわち，(a) データベース全体のデータ構造を記述するためのスキーマDDL（Data Description Language for the Schema）(b) 個々の応用プログラムの立場から見たデータベースの一部分を記述するサブスキーマDDL（Data Description Language for the Sub-Schema）(c) データベース中のデータを操作するDML（Data Manipulation Language）である．DBTG提案時代にはこの3者は同一作業グループで一体になって審議されていたが，1971年以降は(a)は完結した言語として独立し，(b)(c)はCOBOLデータベース機能として別個に審議された．データベース全体の記述（固有構造記述，スキーマ）を応用から独立させることはきわめて魅力的な方向であり，今後のデータベース管理システムの主流をしめる概念になると考えられる．CODASYLによるこの方向の徹底的な追求は，これまでに明らかでなかったおおくの問題を発掘提起した．

3者の関係を図2に示す．データ管理者（DA; Data Administrator）と呼ばれるソフトウェア専門家（の集団）がスキーマDDLを使ってデータベース全体を記述し管理運営する．スキーマDDLは既存のどのプログラム言語からも独立で，かつ共通に利用できるデータ記述言語である．応用プログラムはサブスキーマを通してデータベースと連絡する．応用プログラム中では，データベースを操作する一群の命令が使われる．応用プログラムは全体としてはCOBOLプログラムであり，その中にサブスキーマやデータベース操作命令が埋め込まれている．サブスキーマDDLとDMLとは応用プログラムのプログラム言語（親言語という）に依存しており，今回正式文法になったものは正確にはCOBOLサブスキーマDDL，COBOL DMLと呼ばれる．

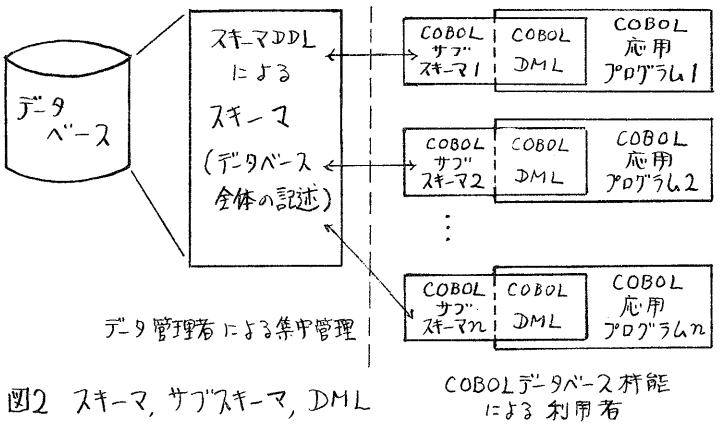


図2 スキーマ，サブスキーマ，DML

1.2 スキーマとサブスキーマとの対応

1.2.1 サブスキーマ環境 サブスキーマの機能については，DBTG'71提案ではかならずしも明確でなかったが，COBOLデータベース機能ではサブスキーマはソースプログラムそのままの形でCOBOLサブスキーマ登録集（ライブラリ）に置かれることがはっきりした．サブスキーマソースプログラムをだれが作るかは指示されていないが，データ管理者が集中管理するものと考えられる．スキーマを知らずにサブスキーマを書くことは不可能である．

1.2.2 対応と結合 一つのデータベースに対してスキーマはただ一つだけ存在する．一つのスキーマに対して複数個のサブスキーマが存在しうる．複数個のサブスキーマがスキーマの部分を重複使用してもよい．COBOLプログラムは5.に例示する方法で登録集からサブスキーマを呼び出す．ただし一つの実行単位はただ一つのサブスキーマしか呼び出して使用できない．すなわち実行時には，スキーマ，サブスキーマ，COBOLプログラムの各一つから成る一意な結合が存在し，この結合は動的には変化しない．具体的に3者の結合を行なう時点としては，COBOLプログラム翻訳時，サブスキーマ翻訳時，実行時などが示唆されている．どの時点にするかで実行時の能率に影響がある．[13]

1.2.3 写像と優先規則 サブスキーマは論理的にスキーマの部分集合でなければならない．スキーマ中の親子組（set）*型やレコード型の一部だけを選択記述でき，論理的に矛盾しない範囲でデータ構造およびデータ形式の写像を行なわせうるが，スキーマ中に記述されていないデータ構造を独自に定義したりはできない．プライバシ指定や親子組選択指定は，サブスキーマとスキーマとの両方で行なわれると，サブスキーマのそれが優先されると規定されている．COBOLデータベース技能がこのような優先規則をみだりに設定することは，混乱をまねくもとになりかねない．なお1.3.4，2.5，3.3参照．

* Setに対する訳語として前稿[16]では「集合」を用いたが，数学の集合論的な扱いと区別するために，本稿ではかりに「親子組」とした．

1.2.4 言語独立なサブスキーマ　サブスキーマは親言語の文法に依存する部分がおおいとの想定にもとづいて，COBOLサブスキーマDDLが設定された．さらにCODASYLではFORTRANサブスキーマDDL，FORTRAN DMLの検討をも進めている．しかし現実には，COBOLサブスキーマDDLのうちのかなりの部分はスキーマDDLの部分集合にすぎず，このことはサブスキーマDDLも大半が言語独立になりうることを暗示している．CODASYLでもこの点を認識しており，サブスキーマDDLがどこまで応用プログラム言語から独立になりうるかを検討している．なおサブスキーマの機能拡張をめざす動きもある（6. 参照）．

1.3 データ操作のデータ定義からの独立

1.3.1 データ独立（data independence）　データを操作する手続き（プログラム）をデータの定義からなるべく独立にして，データ定義に変更があっても操作手続きは変更しなくてもすむようにしたいという要望からデータ独立の概念が生まれた．CODASYLのデータベース用共通言語はこのような概念より以前から設計されていたので，データ独立を欠くとの批判をしばしば受けている．しかし方向としては，データ独立をよりよく達成できるように努力しており，おおくの面で言語仕様改善のあとがうかがえる．

1.3.2 データ定義とデータ操作との分離　この言語体系では，スキーマDDLを完全に分離させて，その言語仕様からデータ操作に関する要素を可能なかぎり取り除こうとしている．さらに応用プログラムはそれぞれの仕事に最低限必要なデータベース部分記述（サブスキーマ）だけを通してデータベースと接触するので，データベースのこれ以外の部分の変更に影響されない．これらの概念を追求した結果すくなくとも次の三つの問題点が明らかになってきたと考えられる．

（1）データ操作をはっきりと想定してはじめてデータ記述言語を合理的に設定できる．データ操作言語（DML）とデータ記述言語（DDL）とを完全に別個に設計し維持することはおそらく不可能である．

（2）データ独立とシステムの能率（低下）との間には密接な関係がある．

（3）完全なデータ独立を達成することもおそらく不可能である．「データ定義に変更があってもデータ操作手続きの変更は最低限にとどめる」というほうが，より現実的なデータ独立の概念である．

1.3.3 配置モード（LOCATION MODE）句　データ独立に関してよく取りあげられる言語要素を四つ以下に論じる．その第1は配置モード句である．データベースにレコードを蓄積し，またそこから呼び出す場合の手がかりとして利用する．配置モード句はスキーマDDL中でレコード型ごとに指定する．次の3種類がある．

（1）直接（DIRECT）配置モード―データベースキーの値を直接指定する方法．

（2）計算（CALCulation）配置モード―データベースキーの値を計算するもとになるデータ項目の値を指定する方法．

（3）親子組（SET）配置モード―データ構造を手がかりにして，システムにデータベースキーの値を決めさせる方法．

このうち（2）はハッシュ化によってデータベースキーの値を定める方法と考えられるので，従来のファイル処理におけるキー変換の手法と同じである．キー変換のもとになるデータ項目の値は利用者がかならず与えなければならない．一方（3）は親子組によるデータ構造操作に便利なようにシステムが独自の手順でデータベースキーを定める方法であって，利用者には介入の余地がない．（1）はDBTG'71提案当時は（2）に近いより利用者よりの手法と考えられていたが，その後

の文法の細かい改訂により，現在では(2)と(3)の中間的な方法とみなされるにいたっている．すなわち直接配置モードでは，原則として利用者がデータベースキーを直接操作する．しかし利用者はNULL(空)値を指定することによって，実質的な値の操作をシステムにまかせてしまうことができる．これを活用すれば，データベースキーの値に全然ふれることなく直接配置モードを実現できる．

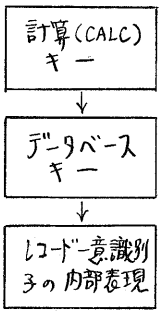


図3 計算キーとデータベースキー

レコードの蓄積，呼出しにあたって，この配置モード句をよく知っておく必要があるのでデータ独立をそこなうという批判がある．一部は誤解にもとづいた批難であったが，現在ではFIND(呼出し)命令と配置モード句の関係もかなりきれいに整理されている．すなわちレコードの配置を直接モードで行なうか，親子組モードで行なうかということは，親子組というデータ構造の設定とはべつの事項である．直接配置モードのレコードの呼出しを親子組をキーにして行なってもよいし，その逆も可能である．計算モードにおいてだけは，利用者がキーを厳重に監視しなければならない．この関係を図4に示す．親子組選択(SET SELECTION)句でも配置モードへの依存をできるだけ取り除く努力が行なわれた．

配置モード ＼ FIND	データベースキー	計算キー	親子組
直接	○	×	△
計算	○	○	△
親子組	○	×	○

図4 配置モード句とFIND命令の組合せ
(△は示すデータ構造の定義があればよいことを)

1.3.4 蓄積の制御とデータ独立 データ独立を達成するにあたっていちばん困難な課題の一つは，データベースに蓄積(STORE)するレコードの制御であろう．データ管理者は当然ながらデータベース中のレコードを蓄積を可能なかぎり厳密に制御したい．ある特定のレコード型について考えても，複数の利用者がそれぞれの立場でつごうのよい方法でレコードを呼び出すことはいっこうにかまわないが，それぞれが勝手にレコードを蓄積することはおおきな混乱のもとになる．したがってこの共通言語体系でも，レコードの蓄積はスキーマ中の指定によって厳重に制御され，スキーマ中の指定が変われば蓄積(STORE)命令の効果がかわる．すなわちこの面ではデータ独立ではない．なおサブスキーマの親子組選択句がスキーマのそれに置きかわるという文法は，蓄積に関してはおおいに疑問である．

1.3.5 データベースキー DBTG'69提案当時のたいへん強いI-D-S色がだんだんうすれて，論理的なレコード意識別子の色合いが濃くなった．かつては，親子組データ構造の物理的表現において，データベースキーをポインタの内容として使用するとまで規定されていたが，この種の文法は削除された．ただしDMLにおける現在指示子(currency indicator)の内容はいぜんとしてデータベースキーの値であると規定されている．またある範囲内でデータベースキーの値の大小順にレコードを順次呼び出す機能(FIND命令)も残されている．

1.3.6 領域(area, realm) ページに相当する概念として批判のまとになった．現在の解釈では，むしろファイルに相当するものとしてとらえられている．READY(準備)命令も親子組単位に指定できるようになったし，WITHIN(内)句もNULL値を柔軟に使用できるので，応用プログラムはどのレコードがどの領

域にあるかを明確に意識しなくてもよくなった．あるレコード型のレコードが複数の領域にわたって書えられる場合でも，領域の指定をしなくてもよい．もちろんこれらの自由化は能率とひきかえに行なわれたのであり，真に能率のよい操作のためには，領域について十分な知識を持っている必要があろう．

2. データ構造

2.1 バックマン線図[5] このデータベース用共通言語体系で記述し操作するデータ構造の説明のためには，I-D-S の創始者 C.W. Bachman によるバックマン線図（Bachman diagram）が使われる．バックマン線図では，レコードの型を一つの長方形で表わし，レコード型の間の親子組の関係を矢印で表わす．長方形の中にレコード型の名前を，矢印のわきに親子組型の名前を書く．簡単なバックマン線図の例を図5に示す．実際のデータベース中には，一つのレコード型に対する実現値（occurrence）が，値の発生した回数だけ存在する．バックマン線図はレコード型についての関係だけを表示し，レコード実現値についての関係をなにも示さない．T.W. Olle は図6のような実現値表記法を提唱している．いずれにせよバックマン線図は単純明確なので，以下にあげる制限をうっかりと見のがしがちである．

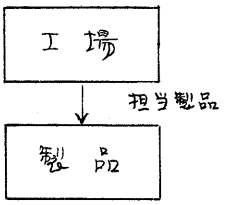


図5 バックマン線図の例

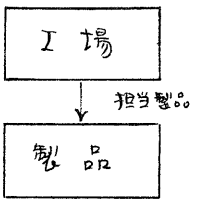
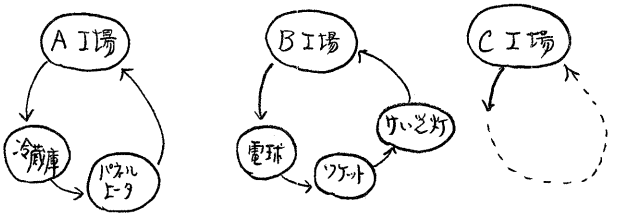
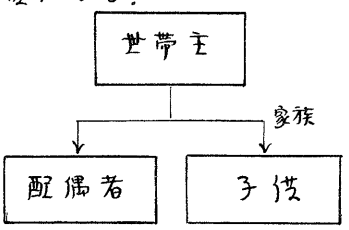


図6 T.W. Olle による実現値表記法

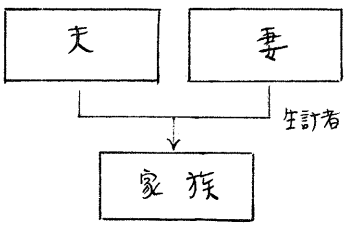


2.2 いくつかの制限事項 このデータベース用共通言語体系では，レコード型のレベルで網（network）構造まで表現できる．しかし以下に述べるような制限事項が文法として定められているので，注意が必要である．

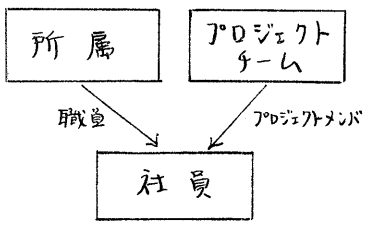
（1） 親子組の宣言において，親レコード型と宣言できるレコード型の個数は1個だけである．これに対して任意個のレコード型を子レコード型として宣言できる．したがって一つの親子組型においては，レコード型のレベルで親と子の1:n対応が成立する．網構造の宣言は複数個の親子組宣言を組み合わせてはじめて可能になる．



許されているデータ構造



許されていないデータ構造



許されているデータ構造

図7 レコード型の1:n対応

（2） 親子組宣言を重ねて行くことによって，自然に輪（cycle）構造が形成されることがあり，これは禁止されていない．しかし一つの親子組宣言の中で，同

じレコード型を同時に親であり子であるという宣言は禁止されている．したがって図8のようなデータ構造を直接には表現できない．

(3) 親子組の一つの実現値において，親レコード実現値と子レコード実現値とは1:n対応をしている．すなわち子レコード実現値は一つの親子組の型内ではたかだか一つの親レコード実現値に対応しているにすぎない．したがって図9のようなデータ構造を直接には表現できない．

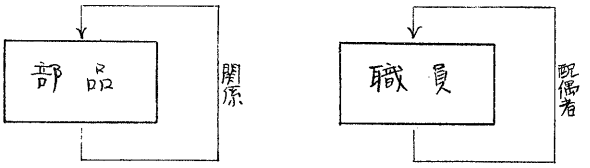


図8 直接的な輪構造は禁止

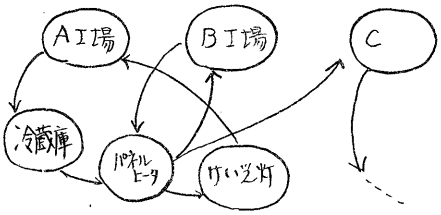


図9 直接的なn:m対応は禁止

2.3 構造レコード，関係レコード

実現値のレベルで親レコード，子レコードのn:m対応を実現させる技法として，構造レコード（structure record）の概念がよく知られている．COBOLサブスキーマDDLには関係レコード（relational record）というデータ項目を含まないレコードの概念が導入されていて，やはり構造レコード的用法を意図しているものと考えられる．図10にこの用例を示す．

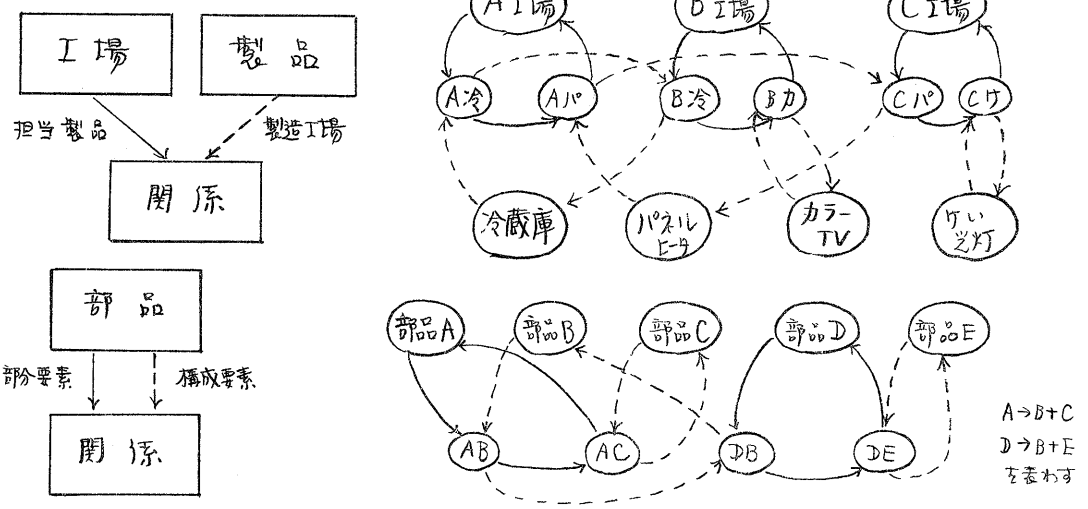


図10 関係レコードによるn:m対応の表現

2.4 特殊な親子組

2種類の特殊な親子組形態が規定されている．その一つは親レコード型をシステム（SYSTEM）と宣言するもので，特異親子組（singular set）と呼ばれる．順編成ファイルは特異親子組として記述できる．もう一つは親レコード型だけを宣言して子レコード型の設定は実行時のDML命令にまかせてしまう親子組で，動的親子組（dynamic set）と呼ばれる．どんなレコード型のレコードでも，実行時には動的親子組の子レコードになりうる．

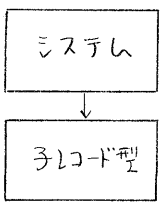


図11 特異親子組

2.5 データ操作命令とデータ構造　利用者はデータ操作命令によってデータベース中のデータを操作し，内容を修正したり，レコードを削除したりできるけれども，スキーマDDLで記述されたデータ構造（親子組の宣言）を改変することはけっしてできない．ただしサブスキーマを通して見えるデータ構造はスキーマ中のデータ構造の部分にすぎないので，単なるレコードの削除でもデータベース全体に思わぬ影響をおよぼす可能性はおおいにある．

3. プライバシの保護機能

3.1 データ操作の制限　データベース中のデータを資格のない人による呼出しから保護するために，プライバシロック，プライバシキーの†機能が用意されている．データ管理者はデータ構造のそれぞれのレベルに想定されるデータ操作ごとに，プライバシロック（PRIVACY LOCK）を宣言できる．これらのデータ操作を行なうためには，適切なプライバシキー（PRIVACY KEY）を与えてシステムの許可を得なければならない．ロックまたはキーとして使用できるのは，文字直定数，データ項目，手続きのいずれかである．たとえばスキーマ中で

```
01 KYUUYO PRIVACY LOCK FOR GET IS "KYUUYO-TANTOO" ...
```

と宣言されているKYUUYOというレコードに対して，GET命令を実行させたい利用者は，そのCOBOLプログラムの宣言節中に

```
KEY-SETTEI SECTION.
    USE FOR PRIVACY ON GET FOR KYUUYO.
DB-P-KEY-KYUUYO.
    MOVE "KYUUYO-TANTOO" TO DB-PRIVACY-KEY.
```

と宣言しておかなければならない．DB-PRIVACY-KEYは特殊レジスタで，USE FOR PRIVACY 句は領域のREADY命令実行時に実行される．なお

```
01 A PRIVACY LOCK FOR GET IS B ...
01 B PRIVACY LOCK FOR GET IS A ...
```

という奇妙なプライバシデッドロックがありうる（東大和田英一氏による）．

3.2 スキーマプライバシロックのアキレスけん　スキーマ全体に対するプライバシロック指定があり，これをなんらかの手段で知るとスキーマ全体を入手できる．たとえばスキーマ全体に対するPRIVACY LOCK FOR DISPLAYがこれである．この弱点を補なうために，プライバシロックの管理はスキーマDDLとは別個の言語体系にまとめるべきだとの議論がある．

3.3 サブスキーマプライバシ指定の優先　サブスキーマ中でスキーマ中におけるのとほとんど同等のプライバシ指定ができる．両方の指定が重なった場合にはスキーマ中のプライバシ指定は無視され，サブスキーマ中の指定が有効になる．これは（1）スキーマ中のプライバシロックがかりに盗まれても，個々のサブスキーマは別個にロックできること，（2）サブスキーマの利用者ごとにきめの細かいロック指定が行なえること，などの利点をもっている．

3.4 より高度のプライバシロック　さきの例で，いったんKYUUYOレコードをGETできた利用者は，それをどんな目的に使ってもよい．実際には，給与統計処理のCOMPUTE命令のためにはこの値を使わせてもよいが，WRITE命令の対象としては使わせたくないといった条件付き許可がありうる．現在の言語仕様ではそこまでは指定できない．

† 今後のDDL JODでは，PRIVACYという用語のかわりにACCESS-CONTROLという用語が使用されることになっている．

4. データ保全（data integrity）

同時実行単位によるデータベース同時更新によって，データの内容が無意味にならないかどうかを管理する機能である．レコードは現在レコードになった時点で監視モード（monitored mode）に入り，現在レコードでなくなるとこのモードでもなくなる．KEEP 命令と FREE 命令とによって，監視モードを拡張（extended monitored）することもできる．これらのモードにあるレコードはシステムの監視をうけ，同時実行単位による変更が行なわれなかったかどうか自動的に検査される．同時実行単位による汚染が検出されると，データベースを変更する命令の実行が行なわれずに，エラー状態表示だけが行なわれる．システムによる監視と検出とは，特定の命令（MODIFY, CONNECT, DISCONNECT, ERASE）の実行時にだけ動作する．

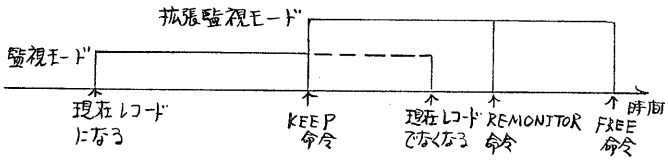


図12　監視モードとその拡張

5. COBOL データベース機能によるプログラムの構造

```
IDENTIFICATION  DIVISION.
PROGRAM-ID.  DBF-KAISETU.
    :
ENVIRONMENT  DIVISION.
    :
DATA  DIVISION.
SUB-SCHEMA  SECTION.
DB  サブスキーマ名  WITHIN  スキーマ名        <- サブスキーマを呼ぶ
    PRIVACY  KEY  IS  "SUKE".
FILE  SECTION.
FD  ...
    :
WORKING-STORAGE  SECTION.
    :
PROCEDURE  DIVISION.
DECLARATIVES.
    プライバシキーのためのUSE  FOR  PRIVACY 宣言節
    :
END  DECLARATIVES.
SIGOTO  SECTION.
    従来のCOBOLの命令のほかに
    以下のDML命令が使える．
    READY    } 領域の準備，後始末
    FINISH   }
    FIND     レコードの発見（位置ぎめ）
    GET      レコードを利用可能にする
    STORE    蓄積
    CONNECT    } データ構造との関連づけ
    DISCONNECT }
    MODIFY
    ERASE    削除
    ORDER    配列
    KEEP       } 監視モード用
    FREE       }
    REMONITOR  }
      ACCEPT   } これらも多少拡張されている
      SET      }
      USE      }
```

```
TITLE  DIVISION.
SS  サブスキーマ名  WITHIN  スキーマ名
    PRIVACY  LOCK  FOR  INVOKING
      IS  "SUKE".
MAPPING  DIVISION.
ALIAS  SECTION.
AD  == ... ==  BECOMES  ... .
    :
STRUCTURE  DIVISION.
REALM  SECTION.
RD       ...
    :
SET  SECTION.
SD       ...
    :
RECORD  SECTION.
01         ...
    02        ...
01
    :
```

```
TITLE  DIVISION.
    :
```

COBOL サブスキーマ登録集

図13　COBOL データベース機能によるプログラムの構造

6. 今後の拡張

ANSI/SPARC データベース SG の提唱する三段階のスキーマ（図14）の概念は今後のスキーマ DDL に影響を与えるであろう[7,9]．サブスキーマの概念を拡張するとい

う動きもある（図15）．DMLの拡大または変更については不明である．

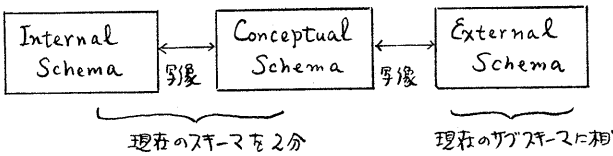


図14 ANSI/SPARC データベースSGによる3層スキーマ

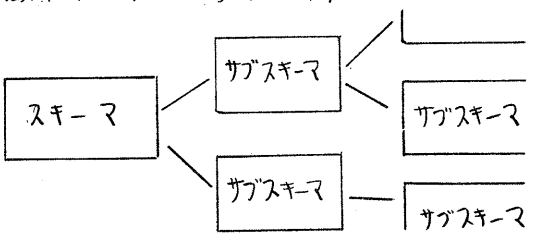


図15 拡大サブスキーマの概念

IDMS	Cullinane,	IBM/360, IBM/370, PDP-11
IDS II	HIS	Honeywell 6000
DMS 1100	UNIVAC	UNIVAC 1108
DMS 90	UNIVAC	
DBMS-10	PDP	PDP-10
EDMS	Xerox	
TOTAL	Cincom	きわめて多数（ただし変形版）
SAAB DMS	SAAB	
SIBAS	（スウェーデン）	IBM/370
PHOLAS	Philips	Philips, ICL
Aberdeen Univ. DBMS		ICL 4-70
DBMS 2200	日電	NEAC 2200
ADBS	日電	ACOS

図16 実動システムの例
（T.W. Olle, F. Manola 氏による）

7. 実動化の動き

この言語体系を実動化する動きはきわめて活発で，1975年9月現在すでに図16に示す数おおくの実動システムが報告されている．ここにみるかぎり，このデータベース用共通言語はすでに学会の研究発表用の域を脱して，実用化が定着しつつあるように感じられる．

8. 標準化の動向

いぜんとして激しい議論のまとになっている．アメリカ規格協会にはすでに前述のANSI/SPARCデータベースStudy Groupができて，標準化の方向をさぐっている．国際標準化機構にも，ISO TC97/SC5内にデータベースシステムStudy Groupができて，今夏その第1回会合が開かれた．ケンブリッジ大学のM.ウィルクス教授はこの問題について，「CODASYLの言語体系は立派な仕事であるが，これをただちに標準化すると，より良い方式の芽をつみ取ってしまう恐れがある」と述べている．これが代表的な慎重論である．べつの立場からは，「ほかに候補がないからといって，よりよい候補ができるまで待つというのは奇妙である．現在の言語体系の標準化にただちに取り組むべきである」という推進論もある．結局は利用者の要求が結論へ導びくことになるのであろう．PDPに始まったミニコンへの実動化の動向はその意味で注目される．なおCOBOLデータベース機能はCOBOLの標準化のわく内で結論を出しうる点にも注目すべきであろう．

9. 関係表モデル（relational model）について

CODASYLのデータベース用共通言語に対する対抗馬として，E.F. Coddによる関係表モデルがよく取り上げられている．関係表モデルによるデータベースシステムはまだ詳細な言語仕様のレベルに達していないので，両者の比較はしばしばデータ構造についてだけ行なわれる．この段階でのおもな論点は，利用者にデータ構造を陽に見せることが正しいかどうかにしぼられよう．筆者は陽に見せるべきものであるとの立場をとる．

この種の議論は今後も活発に行なわれるべきであろうが，議論に先立ってまず

用語を厳密に整理する必要がある．たとえばaccess pathという用語を厳密に定義せずに，「このシステムはaccess path dependentである」といった議論だけを先行させることは無意味であるのみか，誤解のもとになりかねない．

10．結言

C.W. Bachmanが1964年のFJCCでI-D-Sの概念を提唱し，[4]「COBOL, FORTRAN, さらにはNPLの担当者諸氏は，それぞれの言語にこの種の機能が必要でないかどうか十分に検討されるべきである」と論じてから，11年経過した．I-D-Sに始まるCOBOLデータベース機能は着実に進展し，定着しつつある．

参考文献

1. "Data Base Task Group Report to the CODASYL Programming Language Committee April, 1971", ACM (ACM Order Dept. $6.00)
2. "CODASYL Data Description Language Journal of Development June, 1973, NBS Handbook 113", Jan. 1974 (U.S. Government Printing Office, $2.15)
3. "CODASYL COBOL Journal of Development 1973 with revision to May, 1975", 110-GP-1C, Canadian Government
4. Bachman, C.W., Williams, S.B., "A General Purpose Programming System for Random Access Memories", Proc. FJCC, 1964
5. Bachman, C.W., "Data Structure Diagrams", DATABASE (SIGBDP Newsletter), 1-2, 1969 ／あり, bit 1974年12月号
6. Bachman, C.W., "The Programmer as Navigator", Comm. ACM, 16-11, 1973 — 訳
7. Bachman, C.W., "Trends in Database Management", AFIPS Conf. Proc., Vol. 44 (NCC 1975) — 下記の8.の解説．
8. "Interim Report ANSI/X3/SPARC Study Group on Data Base Management Systems 75-02-08", (1975)
9. "Implementations of CODASYL Data Base Management Proposals October 1974, Proceedings of a two-day Symposium", the British Computer Society, 1975
10. Klimbie, J.W., Koffeman, K.I, eds., "Data Base Management, Proceedings of the IFIP Working Conference on Data Base Management", North-Holland, 1974 — IFIP TC2による一連のデータベースWGの最初のもの．
11. Taylor, R.W., "Report on IFIP TC-2 Conference: A Technical In-Depth Evalutation of the DDL", FDT (Bulletin of ACM-SIGMOD), 7-1, 1975
12. "Information Processing 74 Proceedings of the IFIP Congress 74", North-Holland (1974) — データベース関係の論文多数あり．Olle (1098-1106), Codd (1017-1021) など．
13. Managaki, M., et al, "A Structural Model for System Implementation and its Application to CODASYL DBTG", 2nd USA-Japan Computer Conference Proceedings, 1975
14. Manola, F.A., Wilson, S.H., "Data Security Implications of an Extended Subschema Concept", ibid.
15. 植村俊亮，"CODASYLの活動"，情報処理，13-2 (1972)
16. 植村・西村，"CODASYLのデータベース用共通言語"，情報処理学会データベース研究会資料73-4 (1973)
17. "データベースの諸問題"，昭50年電気四学会連合大会 講演論文集 オ6分冊 541 (1975) — 予定