報　告

データベース言語研究委員会報告 1974—1975年度*

データベース言語研究委員会

データベース言語研究委員会委員

本委員会の委員構成は次のとおりである.

委員長　藤中　恵†(日立)

幹　事　植村俊亮†(電総研)

委　員　池上信男(日立), 伊藤靖彦(日電), 石田番也†(三菱), 岩間研二(三菱), 大石光太郎†(日本ユニバック), 小澤　弥(日電), 川口　充 (富士通), 河津誠一(電電・横通), 坂口　徹(電電・横通), 菅田　光(SSL), 鈴木道夫(電力中研), 高木正英(成蹊大), 田之上拓雄(富士通), 谷内龍輔(東芝), 寺尾　実(日電), 戸沢義夫(東大), 等々力正文(国鉄・鉄研), 中島幸一(日電), 中根清吾(電電), 西村恕彦(電総研), 原　潔(日本ユニバック), 久富博昭(沖電気), 細谷僚一(電電・横通), 穂鷹良介†(SSL), 前田　徹†(日本 SDC), 真名垣昌夫(日電), 渡辺　孝(電電・横通).

† は本稿執筆分担者を示す. (委員氏名五十音順)

1. はじめに

1973年2月に情報処理学会データベース研究会の第1回研究連絡会が開かれた. 研究連絡会はデータベース研究会の運営について協議し, 年間約6回の研究発表会開催のほかに, 特別のテーマをほりさげて勉強する作業グループを作ることになった. 当時データベース研究の中心的位置をしめていた「データベース用プログラム言語」と「データベースのモデル」の二つのテーマが選定され, 二つの作業グループが誕生した.

1974年4月からは, データベース言語作業グループが情報処理学会の正式の研究委員会になり, データベース言語研究委員会として2年間活動した.

データベース言語研究委員会は,

「CODASYL のデータベース用共通言語の堀り下げた検討, 正しい解釈の確立を中心として, データベース用言語に関連する分野を集中的に研究する」

ことを目標として, 1974年6月の第1回委員会を皮切りに, 1976年3月までに 18 回の研究委員会を開催した. この分野に対する関心の高まりを反映して, 多数の委員がこれに参加した.

本稿はデータベース言語研究委員会の2年間の活動の報告である. 委員会の活動はおもに関連する文献資料の紹介, 検討を中心に進められてきた. その成果はすでに委員を通じて直接間接に学会員にフィードバックされていると信じられるが, なおここにその一部をまとめて報告するものである. この報告をまとめるにあたっては,

(1) 1975 年におけるデータベース用言語の現状と問題点とが明確に把握でき,

(2) 今後この分野へ興味をいだく人の参考資料として役に立つ,

内容になることをめざした.

2. はデータベースシステム全般の世界からみたデータベース用言語の位置づけである.

3. は, 今回の研究委員会活動の中心になった CODASYL によるデータベース用言語の解説と現状分析である.

4. は, データベース用言語の標準化の問題を取り上げた. とくに最近の標準化の動向は注目されよう.

最後に以上をまとめて, さらに今後の動向についても簡単に見通しをのべる.

この研究委員会が検討した資料の一部に注をつけて参考文献表に示す.

2. データベース用言語概論

2.1 ファイルからデータベースへ

最近ではファイルという言葉に代ってデータベースという言葉がすっかり定着してきた. 新しい言葉が持

* Data Base Language Study Committee Report (1974—1975) by Data Base Language Study Committee.

つ魅力のせいか，最近の企業等のシステム開発でも従来ならば「…ファイル・システム」といっていた所を必ずといってもよいほど「…データベース」あるいは「…データベースシステム」と名付けている．しかし確かに従来のファイルという概念にはなかった何か新しいものがデータベースという概念にはある．いくつかの相違点が挙げられようが，基本的なものとしてデータベースにおける蓄積データの利用の多様性を挙げることができる．

従来のファイル処理においては蓄積データを利用する主体は，少数にとどまっており関連するプログラムもたかだか数本であった．したがって，COBOL に見る如く，ファイル自身の定義あるいは記述はそのプログラムに埋めておくことで不便さは感じられなかった．初期のデータベースである IDS はこの名残りを止め，データ定義がプログラムの中でなされていた．しかし，業務あるいはそれに関連したデータ処理システムが統合されるにつれ，たとえば，今まで生産管理部門だけが使用していたデータを経営管理部門が経営計画立案のために参照するなど，同じデータを全く別の部門が共用するという必要性が生じて来た．ここに至って，企業あるいはある組織が所有するデータは，その企業なり組織なりの一部門が所有するものでなくなり共有性を持つに至った．共有性を持つ蓄積データについての記述はもはや1プログラムの中に埋没させておくことは不適当であって，これをプログラムから分離させる必要がある．また，蓄積データそれ自身を定義し，記述する言語 (DDL—Data Description Language あるいは Data Definition Language) も必要となった．同時に，蓄積データは種々の分野で多目的に利用されるから，それらの関係を正しく保ち統一的に維持する能力が必要となった．この利用の多様性から帰因される，

データの共有→データ定義のプログラムからの分離→蓄積データの統一的取り扱い

は，いわゆるデータベースと称されるシステムに共通して見られる特徴であって，これが単なるファイルとの性格上の相違点である．この相違点から生ずる技術上の問題点はしたがってデータベース技術の固有問題点であってファイル時代には存在しなかったものが大半である．一例を挙げるならば

(a) デッドロック (複数ユーザに同時更新を許すとき生じる)

(b) セキュリティあるいはプライバシ (多種ユーザが共有データベースにアクセスすることから生じる)

(c) データ定義 (多種ユーザに一般的かつ効率的データ定義をさせるための方法)

などが挙げられる．上記以外にもデータベース固有の技術上の問題点は数多く存在し，またデータベースへの要求が日に日に拡大している現状を考えると，さらに多くの技術的問題点が今後出現することが予想される．データベースは発展過程上，まだその幼年期にあるといえよう．

2.2 データベース管理システム中でデータベース用言語がしめる位置

データベース用言語とはいっても，FORTRAN, PL/I, COBOL 等の汎用言語でないことは明らかであるとしても，GPSS, SIMULA, RPG, APT などの問題向言語とも同列には扱えない特殊性がある．いわば，上記のいわゆる言語においてはその言語を利用してプログラムを書けばそれで完結というような感じがあるのに対し，データベース用言語の場合には，本体はデータベースそのものであって，知識なり事実なりを如何に認識していかに表現するか，いかに維持するかという問題が主であり，言語の問題はユーザとデータベースとがどの段階でインターフェースをどのように持つかという問題の一部として存在する．

したがって，データベース言語を単に言語的な側面から (別な言葉でいえば文法的な諸取り決めの問題から) のみ考察するのは順序として正しくない．あくまでも根本はユーザとデータベースとがどのようにしてインターフェースを (それも一つとは限らない) 持つかという問題が解明された後の問題として位置づける必要があろう．この考察が終らない内にいわゆる言語の問題である文法的細則を定めたとしても，あたかも大木を描写するのに幹から始めずに枝葉から始めるような結果になる可能性が強い．

このような観点は，データベース用言語の標準化の議論がさかんになってから，アメリカ規格協会(ANSI)に設置されたデータベース SG が主唱してきたものである(**4.3** 参照)．CODASYL システムズ委員会でも，親言語方式や独立言語方式などの概念を通してインターフェースの問題の一端をとらえてはいたが，現実のデータベース用言語の研究開発はこのような概念の整理をまたずにどんどん先行してしまった．

いま一度データベース用言語をデータベースとユーザとのインターフェースを中心にしてとらえて，その

位置づけを行ってみよう.

ユーザインターフェースを図-1 に分類して示す.

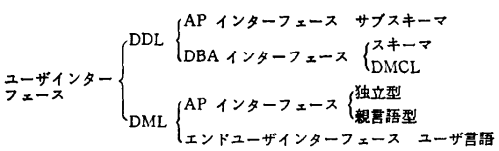

図-1 ユーザインターフェースの分類

データベースのユーザインターフェースは2種に分類できる. その一つは, データベースの定義を行うためのものであって, これを行う部分を通常 DDL (Data Description Language, あるいは Data Definition Language) という. 他方は直接データベースに対してのアクセスを行う部分であって DML (Data Manipulation Language) と呼ばれる. なおデータベースモデルの一つである自己記述的モデルにおいては, この両者の区別はなく DML のみが存在すれば事足りるとされている.

DDL は伝統的には更に2種に分類される. AP (Application Programmer's) インターフェースは CODASYL のデータベース用言語ではサブスキーマと呼ばれているが, これは, アプリケーション・プログラマごとにデータベースを異なった見方で見るための定義である.

DBA (Data Base Administrator, Data Administrator, データベース管理者) は, データベース全体の論理的定義とその物理的配置あるいは構造の定義を行う(3.2.3.1 参照). CODASYL のデータベース用言語では, スキーマと DMCL (Device Media Control Language) に相当する部分である.

さて DML は AP インターフェースを受持つ部分とエンドユーザインターフェースを受持つ部分とに分けられる. ここでエンドユーザとは, データベースを全くプログラムの知識なしに利用する利用者とでもいうべきもので, データベースの発展を考えると, 将来の最も層の厚いデータベースユーザである.

従来のデータベースには AP インターフェースとして独立型と親言語型との2種が存在した.

独立型とはたとえば GIS, MARK IV, ASIST 等に見るようにその言語体系の中でデータベースに対するアクセスをすべて完結させようとするものである. この中には, AP インターフェースとエンドユーザインターフェースの中間をゆくものもある.

親語型としては, CODASYL 言語, IMS, TOTAL, ADABAS, SYSTEM-2000 に見るごとくデータベースにアクセスする手段としては若干のコマンドのみに限定し, 条件判定, 数式演算等他の機能についてはこれを親言語 (そのコマンドが埋め込まれる別の汎用言語——たとえば PL/I, COBOL, FORTRAN) にまかせるという方式を取るもので, 現在のデータベース言語の主流を成している. なお, 親言語型は更にその言語特有のコマンドあるいは動詞を有し, したがって通常はプリコンパイル手段がよく使われる方式のもの (CODASYL 言語, SYSTEM-2000) と単に CALL 命令によってサブルーチンとして呼び出す方式のもの (IMS, TOTAL, ADABAS 等) とがある.

ユーザインターフェースとして大切な性質は, データ独立性, オープンエンデドなることなどである.

本研究委員会がおもに検討した CODASYL のデータベース用言語は, DDL と, AP インターフェースの DML とからなっている (3. 参照). なお 4. にアメリカ規格協会データベース SG によるインターフェース論を紹介する.

3. CODASYL のデータベース用言語

3.1 歴史, 概説

CODASYL は 1960 年にプログラム言語 COBOL を開発して, その改訂拡張を活発に行ってきたが, 1966 年にはリスト処理作業グループを結成して, 2次記憶装置上でのリスト処理作業の機能を COBOL に追加するための作業を開始した.

この作業グループは翌年データベース作業グループ (DBTG) と改称し, 作業目標も, データベース機能を COBOL に追加することと修正した.

DBTG は 1969 年, 1971 年にそれぞれ DBTG 提案[1] と呼ばれる報告書を発表して, 具体的なデータベース用言語を提案した. この提案は特定の計算機システムに依存しない「コンパイラレベルの言語」の文法の提案であり, COBOL とのかね合いもあって, 将来はデータベース用言語の標準化の対象になるのではないかとの観察からも, ひろく世界的な注目を集めた.

1971 年の DBTG 提案をもとにして 1975 年までに,

データベース記述用言語 (DDL)[2]

COBOL データベース機能[3]

が完成し, さらに

FORTRAN データベース機能[4]

の開発が進められている. 本研究委員会の主要な活動目標も, この言語の徹底的な分析にあった. 研究委員

会の活動期間中にこの言語の実動化は急速に進み，1976 年現在すでに十指にあまるシステムが存在するにいたっている（**3.5** 参照）．

CODASYL によるデータベース用言語は，つぎの三つの要素からなる．

（1） スキーマ DDL——データベース全体を記述するための DDL．

（2） サブスキーマ DDL——個々の応用プログラムからみたデータベースの部分を記述するための DDL．

（3） DML——データベースを操作するための命令群．

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と密着していて，それぞれの言語のいわば「データベース機能」を構成する．

本章ではこの言語を概説する．

3.2 DDL（データ記述言語）

3.2.1 概　説

DBTG 提案に含まれていた言語要素のうち，スキーマ DDL はデータベース全体の記述言語であり，とうてい COBOL の一部分になりうる性質ではなかった．CODASYL はこの部分を独立した言語として審議するデータ記述言語委員会（DDLC）を設置して，1973 年には，

Deta Description Language, Journal of Development, June 1973[2)]

を公表した．

ここでいう data description language は「データベース全体を記述する」ためのプログラム言語である．この言語はまた，現存の多くの高水準プログラム言語のいずれからも独立していて，かつ共通に使用できるような共通データ記述言語であることをめざしている．この言語で記述したデータベースを「どんなプログラム言語でいかに操作するか」は，この言語仕様の範囲外とされている．本節では，この資料のごくおおまかな内容を紹介する．

3.2.2 DDL の構造

DDL（正確には，スキーマ DDL）はデータベース全体を記述するために使用する．これはデータ記述言語であって，データを操作する命令などはいっさい含まない．データベースのデータは以下の階層構造をもっている．

データ項目——最小単位．一つの値．

データ群——データ項目の集まり．ベクトルと繰り返し集団がある．

レコード——データ項目，データ群の集まり．データベースの入出力操作における基本単位．

集合（set）——レコードの集まり．DDL のデータ構造表現能力の中心をなす概念．次項参照．

領域（area, realm）——データベースの分割部分．ページの概念に近い．

スキーマ（schema）——データベース全体の構造またはその記述．DDL の記述項からなる．

データベース——一つのスキーマが制御するすべてのレコード，集合，領域からなる．データベースごとに一つのスキーマが存在する．

DDL によるプログラムは以上のそれぞれに関する記述項を上とは逆順に記述していく形式である．その概略を以下に示す．

（スキーマ記述項：スキーマに関する定義を行う）
- SCHEMA 句——スキーマに名前をつける．
  - ON 句——割込み手続きを指定する．
  - PRIVACY 句——機密保護の条件を指定する．

（領域記述項：領域を定義する）
- AREA 句——領域に名前をつける．
  - TEMPORARY 句——一時領域を定義する．
  - ON 句
  - PRIVACY 句

（レコード記述項：レコードを定義する）

レコード副記述項 〔データ副記述項〕… ｝この組み合わせでレコードを定義する

（レコード副記述項）
- RECORD 句——レコードに名前をつける．
  - LOCATION 句——レコードの配置手法．
  - WITHIN 句——レコードを置く領域．
  - ON 句
  - PRIVACY 句

（データ副記述項）
- データベースデータ名句——レコードを構成するデータ項目，データ群に名前をつける．
  - PICTURE 句
  - TYPE 句
  - OCCURS 句
  - RESULT 句
  - SOURCE 句

CHECK 句
ENCODING/DECODING 句
ON 句
PRIVACY 句

(集合記述項：集合を定義する)

集合副記述項 } この組み合わせで集合を
〔メンバ副記述項〕... } 定義する.

(集合副記述項)

SET 句——集合に名前をつける.
OWNER 句——オーナレコードの定義.
DYNAMIC/PRIOR 句
ORDER 句——集合内のレコード順序の定義.
ON 句
PRIVACY 句

(メンバ副記述項)

MEMBER 句——メンバレコードの定義.
KEY 句
SEARCH 句
SELECTION 句——集合選択基準.
ON 句
PRIVACY 句

各句の詳細には立ち入らずに，次項にいくつかの特徴的な概念だけをまとめる.

3.2.3 主要な概念

3.2.3.1 データ管理者，スキーマ，サブスキーマ

データベースは多くのユーザプログラムが共通して使うデータの集まりであるから，ユーザの競合する要求を調整する機能が必要である．この機能を遂行する人（ソフトウェア専門家，システム専門家の集団）をデータ管理者（データベース管理者）という．データ管理者はスキーマすなわちデータベース全体の設計，記述，管理を行い，さらに各ユーザと相談しながらサブスキーマの設計，記述，管理にもたずさわる.

3.2.3.2 データベースの機密保護

データベースの機密保護の必要性もまたその共用性に端を発している（**2.1** 参照）．DDL には，機密保護のための条件を指定する PRIVACY 句があり，この句で指定された条件を満さなければ（パスワードを正しく与えなければ），データ操作を行えない．ただしデータ操作言語がこれを具体的にどう行うかまでは規定していない．いわば錠の文法だけを定めて，かぎの文法は範囲外にしている.

3.2.3.3 データ構造の表現

DDL では，集合 (set) という概念を導入してデータ構造を表現する．そのあらましは以下のとおりである.

さきのレコード記述項はレコード型の定義である．一つのレコード型に対して任意個数のレコード実現値がデータベース中に存在しうる．たとえば「給与レコード」というレコード型の実現値は従業員1人に一つずつ存在するであろう．集合は1個のオーナレコード型と，1個以上のメンバレコード型とかなる．集合にも型と実現値とがある．たとえば「資格」という集合型が，「従業員」というオーナレコード型と，「職位」，「技能」というメンバレコード型とからなるとき，「資格」の各実現値には「従業員」の1個の実現値がかならず含まれ，かつ「職位」，「技能」の任意の個数の実現値が含まれる.

一つのレコード型が複数の集合型のオーナになったり，メンバになったりできるので，これを使って木構造や網構造まで表現できる．ただし実動化を意識した制限がいくつかあるので注意が必要である.

3.3 COBOL DML (COBOL データベース機能)

3.3.1 概 説

1971年の DBTG 提案のサブスキーマ DDL と DML の部分を COBOL 文法にそうように書き直す作業はかなり時間がかかり，結局1975年4月に CODASYL COBOL の正式文法の一部になった[3]．標準化とからんで論争の的になり，慎重に時間をかけた審議が行われたので時間がかかったが，内容はあくまで DBTG 提案を基本としており，大きな変更はなかった.

本節では，CODASYL COBOL の文法にもとづいて COBOL データベース機能を大きく二つに分け，COBOL データ操作言語と COBOL サブスキーマ言語についてその概要を紹介する.

3.3.2 COBOL データ操作言語 (DML)

3.3.2.1 データ操作命令一覧

COBOL データ操作言語は，スキーマの定義に従って実際に構築されたデータベースをプログラマがアクセスするための手段を提供するものである．このデータ操作言語は，COBOL を親言語としており，COBOL プログラムの中で COBOL 動詞と混在した形で手続き部を構成することになる．別の言葉でいえば，手続き向き言語としての COBOL の機能を拡張するものである．**図-2** は，1971年の DBTG 提案と現在の COBOL データ操作命令との対比を示す.

3.3.2.2 制 御 命 令

(1) READY 命令

一つ以上の領域 (realm) あるいは一時領域上の集合

	1971 年 DBTG 提案	COBOL データベース機能
制御用	OPEN	READY
	CLOSE	FINISH
	IF	IF
検索用	FIND	FIND
	GET	GET
	KEEP	KEEP
		REMONITOR
	FREE	FREE
	MOVE	ACCEPT
更新用	STORE	STORE
	MODIFY	MODIFY
	DELETE	ERASE
	ORDER	ORDER
	INSERT	CONNECT
	REMOVE	DISCONNECT
特殊		USE

図-2 データ操作命令 (DML の命令)

(set) について処理できるよう準備を要求し，使用モード (EXCLUSIVE/PROTECTED, RETRIEVAL/UPDATE) の宣言をする．

(2) FINISH 命令

一つ以上の領域あるいは一時領域以上の集合の使用終了を宣言する．

(3) IF 命令

特定の集合において特定のレコードが従属レコードをもっているかどうか，親レコードか子レコードか等の判定を行い，実行順序を制御する．

3.3.2.3 検索命令

(1) FIND 命令

レコード選択式に従って指定されたレコードの位置ぎめをする．FIND 命令は，そのレコードが含まれる領域名，集合名，レコード名，実行単位の現在指示子 (currency indicator) に目的のレコードを指示させる．これは Bachman がかって述べた名言“プログラマは，データベースのネットワークの海を自由に航海する航海士 (navigator) である”を実現化する概念である．レコード選択式には，形式1から7まであり，データベースキーによるもの，レコードの配置方式によるもの，現在指示子そのもの，その NEXT, PRIOR 方向のものなどについて指定ができる．

(2) GET 命令

FIND 命令で位置ぎめされているレコードの内容の全体あるいは指定された項目だけを利用者作業域に転送する．

(3) KEEP 命令

実行単位の現在指示子であるレコードを保持する．

(4) REMONITOR 命令

KEEP 命令により保持されていたレコードをいったん解放し，再度保持する．

(5) FREE 命令

KEEP 命令により保持されていたレコードを解放する．

(6) ACCEPT 命令

各種の現在指示子の内容をプログラム側に転送する．

3.3.2.4 更新命令

(1) STORE 命令

レコードをスキーマ定義に従ってデータベース内に格納する．

(2) MODIFY 命令

レコード内の項目の内容，およびレコードの集合内の関係づけを変更する．

(3) ERASE 命令

一つ以上のレコードをデータベースからロジカルに削除する．

(4) ORDER 命令

レコードの集合内でロジカルな配列順序を変更する．

(5) CONNECT 命令

MANUAL モードの子レコードを集合内に組み込む．

(6) DISCONNECT 命令

MANUAL モードの子レコードを集合から切り離す．

3.3.2.5 特殊命令

(1) USE 命令

ある命令の実行結果がデータベースにとって例外条件であったときに実行すべき手続きあるいは，機密保持のための手続きを指定する．

3.3.3 COBOL サブスキーマ

3.3.3.1 COBOL サブスキーマの構造

サブスキーマは，より高いレベルのプログラムとデータの独立性 (program/data independence) を達成するために考えられた概念であり，ユーザ (プログラム) とデータベースの間にスキーマとサブスキーマの2レベルのマッピングを想定している．サブスキーマは，スキーマの中から特定のプログラムが使用する部分だけを抜き出したものという基本的な性格があり，さらにスキーマよりは，ユーザが自然に認識するデー

タ構造により近い定義が可能である．たとえば，データベース上には物理的にそのような形では存在しない，ユーザ向きのロジカルなレコードを定義できる．

COBOL サブスキーマを定義する方法は，スキーマを定義する言語とよく似ており以下の記述部分から構成される．

TITLE DIVISION
MAPPING DIVISION
  ALIAS SECTION
STRUCTURE DIVISION
  REALM SECTION
  SET SECTION
  RECORD SECTION

3.3.3.2 TITLE DIVISION

このサブスキーマのもとになるスキーマの名前とサブスキーマ自身の名前を指定する．またサブスキーマがプログラムにより使用可能となる前に満足すべき機密保護のルールを指定する．

3.3.3.3 ALIAS SECTION

いわゆるリネーミングの指定をするところで，スキーマで定義された領域名，集合名，レコード名，一意名，作成者名に対し，ユーザが使用したい別の名前をそれぞれ指定する．

3.3.3.4 REALM SECTION

スキーマの領域の中でユーザが使用する領域名を指定する．また各領域についての機密保護のルールを指定する．

3.3.3.5 SET SECTION

スキーマの集合の中でユーザが使用する集合名を指定する．また各集合についての機密保護のルールを指定する．さらに各集合の集合選択基準 (set selection criteria) を指定できる．

3.3.3.6 RECORD SECTION

スキーマのレコードの中でユーザが使用するレコード名を指定する．ある領域上のレコードだけを使用する宣言もできる．また各レコード名についての機密保護のルールを指定する．ここまでは，領域，集合の指定と全く同じであるが，新しいロジカルなレコードを指定することができる．

しかし，スキーマに含まれているレコードの中のユーザが必要な項目だけを任意の配列でまとめて一つの新しいレコードとすることができるというレベルにとどまっている．

なおスキーマ中には，仮想レコード (virtual record)，派生レコード (derived record) など，データベース中に実在しないレコードの定義も行える機能がある．

3.4 FORTRAN DML (FORTRAN データベース機能)

3.4.1 概　説

DBTG 提案からまず COBOL DML が誕生したことは，その開発の経過から考えても当然であろう．CODASYL はさらに 1974 年には FORTRAN データベース機能を検討する委員会 (FORTRAN DBMLC) を設置して，FORTRAN DML と FORTRAN サブスキーマ開発に着手した．

この委員会による FORTRAN データベース機能の文法はまだ正式には公表されていないが，内部資料としてすでに四つの version をかぞえている[4]．

以下この資料をもとに，FORTRAN DML とサブスキーマとの開発方針と概要を紹介する．

3.4.2 開発の方針

この FORTRAN DML の開発に当って委員会はこの言語を，新しく提案されている ANS FORTRAN 改訂案 (FORTREV) と CODASYL DBTG 提案とに基づいたものとすると決定した．

シンタックスは一般にコマンド動詞の後にかっこでくくられたパラメータリストが続く形をとり，入れ子をなすパラメータは，FORTRAN 流にかっこなしで使用される．最近の FORTRAN 言語の設計方針にそって，キーワード形式を多用し，プログラムの見やすさにも注意を払っている．約 20 個の新しいステートメントを ANS FORTRAN に追加することになる．

セマンティクスは COBOL の DML とほぼ同じであるが，特に，ANS FORTRAN での文字式を使用して，レコード名，集合名，領域名のプログラム実行中でのダイナミックネーミングを可能にしている．レコード選択式もプログラム実行中に決定する事が可能である．

その他に，計算機による開発報告の自動編集が可能なように，言語の構文の記法に新しい方法が採用された．これは，BNF 記法に COBOL の言語仕様で使われているかっこの用法を加えたものである (図-3 次頁参照)．

3.4.3 FORTRAN DML の提案

FORTRAN データベース機能の提案 version 0.4[4] は以下の 8 章より構成されている．

第 1 章は，FORTRAN データベース委員会の歴史と，基本方針を述べている．

● FORTRAN DML での retaining 句の表現

```
⟨retain⟩::=
    ⟨retain 1⟩|⟨retain 2⟩
⟨retain 1⟩::=
    RETAINING=([RECORD][,REALM][,SET|{SET=
                ⟨dbnames⟩}])
⟨retain 2⟩::=
    RETAINING=MULTIPLE
```

● COBOL DML での retaining 句の表現

```
RETAINING CURRENCY FOR{ MULTIPLE
                      | REALM
                      | {SETS
                      |  {set-name-1}…
                      | RECORD }
```

図-3 新しい FORTRAN DML 仕様での記法の例

第2章は，データベースの概念と構文の記法を述べている．データベースの概念はCODASYL の DBTG 提案と同じである．記法についてはすでに述べた．DML と，サブスキーマ中で，レコード名，集合名，領域名の代わりに文字式を使用すると定義でき，この文字式を使用することにより，プログラム実行中にダイナミックネーミングが行えることもすでに述べた．レコード選択式は，COBOL の DML でのレコード選択式とセマンティクスは同じである．

第3章はFORTRAN サブスキーマについて述べている．サブスキーマは，SUBSCHEMA, ALIAS, REALM, SET, RECORD, TYPE の6ステートメントから構成され，各々，COBOL サブスキーマの TITLE, MAPPING, REALM, SET, RECORD の各ディビジョンまたはセクションとデータ記述句に対応している．データは，FETCH, FIND, GET, MODIFY, の命令実行時に，スキーマで定義されたデータベース中の型と，プログラムのサブスキーマで定義された型との間の変換が行われる．

第4章は，DML 命令について述べている．15個の命令が (FORTRAN のステートメントと同じ扱いである) 定義された．すなわち ACCEPT, CONNECT, DISCONNECT, ERASE, FETCH, FIND, FINISH, GET, INVOKE, MODIFY, ORDER, PRIVACY, READY, STORE, USE である．

FETCH は，FIND と GET の二つの動作を一つの命令で行う．ほとんどの命令で，文字式を使用したダイナミックネーミングが行える．FIND と FETCH においては，レコード選択式に文字式を使用して，実行時に決定するようにできる．

第5章は，エクセプションの取り扱いについて述べている．次の5個のスペシャルレジスタが，ランユニットごとに設けられる．

DBSTAT　データベースステータスレジスタ
DBRELM　データベースレルムレジスタ
DBSET　データベースセットレジスタ
DBREC　データベースレコードレジスタ
DBPRIV　データベースプライバシレジスタ

以上のスペシャルレジスタ名は，キーワードでも，予約語でもなく，一般の変数名であり，コンパイラにより，プログラム中に DML ステートメントが現われたときに提供されるものである．

第6章は，シンタックスの一覧，第7章は，参考文献のリスト，第8章は用語集である．

今後新 ANS FORTRAN の仕様に基づく FORTRAN がメーカで開発されたとき，この FORTRAN DML とサブスキーマも共に開発されることが期待される．その際，特に，DML とサブスキーマのインプリメンテーションと，データベース管理システムの設計では，ダイナミックネーミングと，レコード選択式の実行時の決定が，いかに効率よく処理されるかで，その FORTRAN DML が評価されるであろう．

3.5 実 動 化

CODASYL のデータベース用共通言語の実動化は，これに関する議論のはげしさに比例するかのように急テンポで行われた．すでに欧米では，

IDS-II (HIS), DMS 1100, DMS 90 (UNIVAC), DBMS-10 (DEC), EDMS (Xerox), PHOLAS (Phillips), IDMS (Cullinane), TOTAL (Cincom), SAAB DMS (SAAB), SIBAS (?), Aberdeen 大学 DBMS などのシステムが現存し，わが国でも，

DBMS 2200 (日電，NEAC 2200 用)，
ADBS (日電，ACOS 4 用)，
AIM (富士通，M シリーズ用)，
EDMS (三菱，MELCOM-COSMO シリーズ用)，
DMS-5 (三菱，MELCOM-COSMO シリーズ用)

などがいずれもすでにリリースを開始，または開発途上にある．さらに前記システムのうち，

IDS-II (東芝，ACOS 6 用)
DMS 1100 (ユニバック，1100 シリーズ用)
DMS 90 (ユニバック，OUK 9000 シリーズ用)
DBMS 10 (DEC, PDP System 10 用)

が日本でただちに利用可能であるし，IDMS の販売も伝えられている．

1971 年ごろの，言語として十分であるかどうかの議論が収束しないうちに，とにかくシステムを作成しようという動きが大勢をしめてしまった感じである．今後の利用者の反応が注目される．

4. データベース用言語の標準化

4.1 概 説

CODASYL によるデータベース用共通言語の開発が本格化するにつれて，データベース用言語やデータベース管理システムの標準化が現実の問題として議論されるようになった．アメリカではいちはやく 1972 年にアメリカ規格協会がそのための研究班 (Study group) を設置し，翌年には国際標準化機構 (ISO) もデータベース管理システム標準化の検討を開始した．本章はこれらの動向の要約と，アメリカ規格協会SPARC データベース SG による提案の解説とからなる．

4.2 標準化に対する ISO の動向

1973 年 8 月，ISO/TC 97 (Computer and Information Processing) の事務局は，オランダから「データベース管理システム及び言語」の標準化活動を開始すべきとの意見 (TC 97 の文書 N 598) を受け，各国にこの意見に対するコメントを要求した．この時オランダは，CODASYL による 3 種の文書：

DBTG Proposals of Oct. 1969 and April 1971[1]，

COBOL Data Base Facility, 1973[3]，

DDL Jour. of Development, 1973[2]

を参考文書としてあげた．これに対して日本国内では，情報処理学会規格委員会 SC 5 専門委員会がこの検討にあたったが，国内の意見がまだ固っていなかったので，コメントの提出を保留し他国の動向を見ることとなった．しかし，十分検討を要する問題なので，オランダも参考文書としてあげている CODASYL のデータベース記述言語仕様書 (DDL JOD, 1973) が，SC 5 専門委員会及び SC 5/COBOL WG の委員に配布された．

1974 年 2 月，ISO/TC 97 は，各国から提出されたコメントの要約を発表した (TC 97 の文書 N 614)．8 ケ国のコメントが集まり，オーストラリア，チェコスロバキア，フィンランド，ドイツ，ハンガリー，スウェーデンの 6 ケ国は，ほぼ全面的に賛成，フランス，イタリアは，時期尚早で反対との結果であった．事務局は，回答が参加国の半数に満たないので，この結論を同年 5 月の TC 97 の総会での討議に持ち越した．

1974 年 5 月，ジュネーブで開かれた ISO/TC 97 の総会では，時期尚早論を述べた日本のコメント (TC 97 文書 N657) も含めて討議されたが，決議事項として，TC 97/SC 5 (Programming Language) に対して，データベースの Study Group を作り，標準化の妥当性を調査するように指示した．

1975 年 6 月 24～26 日，アメリカのワシントンD. C. において，第 1 回のデータベース Study Group の会合が開かれた．日本からは出席しなかったが，フランス，ドイツ，スウェーデン，アメリカの 4 ケ国，計 13 名がこれに出席した．討議の対象となった文書は，スイス，ドイツ，イギリス，フランスからの標準化の進め方の事務的手続きを述べた数ページのものと，アメリカからの技術的な 300 ページもの労作であった．後者は，アメリカの規格協会 (ANSI) の中の ANSI/X3/SPARC Study Group on DBMS が作成した中間報告書 (Interim Report)[5] である (この内容の紹介は，4.3 参照)．この会合の結論は，既存の案をもとにしたデータベース管理システムの標準化は，評価のための基準が確立されていない現在，時期尚早であるので，上記 ANSI/X 3/SPARC の資料を今後の検討の出発点とするということであった．日本国内でも，この線に沿って検討を行うため，SC 5 専門委員会をはじめとして，本データベース言語研究委員会を含む関係委員会に，この ANSI/X 3/SPARC の資料が配布された．

その後，1976 年 1 月 12～15 日，フランスのパリで，第 2 回のデータベース Study Group の会合が開かれている．今回は，フランス，ドイツ，イタリア，オランダ，スウェーデン，イギリス，アメリカの 7 ケ国計 19 名が出席した．今回も日本から出席できなかったのは非常に残念であるが，前回に比べてやや豊富な資料も送られて来ているので，連絡を密にして世界の動向を把握して行けるよう留意すべきであろう．この会合では，Study Group の作業の範囲 (Scope of Work) と作業の進め方 (Program of Work and Schedule) が討議された．

4.3 ANSI/SPARC データベース SG

アメリカ規格協会 (ANSI) には，SPARC (Standards Planning and Requirements Committee) と呼ばれる標準化問題企画委員会がある．SPARC がデータベース管理システム (以下 DBMS と略する) をその検討対象に加える必要を感じ，そのための Study Group (以下 SG と略称する) を発足させたのは 1972 年秋である．この SG は，結局のところ，DBMS に関し標準化の対象となり得るものは，DBMS を構成する要素のそれぞれがどのような働きをなすべきかではなく，それぞれの要素間のインターフェースに関する部分のみである，と判断した．そこで，DBMS はどの

ような要素から構成されるアーキテクチュアを持つべきであるかを描き出すことから始め，その中にどのようなインターフェースが考えられ，それぞれのインターフェースではどのような情報が受け渡しされるべきか，またそのうちどのインターフェースに関する標準化が現実的であり急がれているか，についてレポートをまとめ上げつつある．現時点では，1975 年 2 月付けの 2 次中間レポート[5] が，その活動成果に関し公けにされている最新のものである．

この ANSI/SPARC レポートは，DBMS のフルセットを提示しようとする意欲とその成果において，最も高い価値を見出すことができる．その点で，DBMS のサブセットであることを甘んじつつも，現実的かつ具体的提案としての価値を問うている，CODASYL/DBTG レポートとは，比較次元を異にしている．

ANSI/SPARC レポートにおける，DBMS アーキテクチュアに対する考え方のエッセンスであり，最も特徴づけている点は，そのスキーマ思想にある．そこでは，データの持つ特性，関係，用法を記述するメタ・データの集まりであるスキーマを，概念スキーマ (Conceptual Schema)，外部スキーマ (External Schema)，内部スキーマ (Internal Schema)，の 3 種類に明確に分離して考える必要があることを主張している．

概念スキーマは，データベース中にモデル化して取り込もうとする，現実界のデータに関する，ユーザ企業レベルでの，全体的見地からの本質的記述情報を保持する．概念スキーマはひとつのデータベースに対して，ただひとつのみ存在し，企業管理者と呼ばれる専門的役割の人によって，その内容が統括管理される．

外部スキーマは，データベースをアクセスする個々の利用者（アプリケーションプログラマ，エンドユーザ）にとって，知る必要があり，知る権利があるデータに関する，それぞれの利用目的に適合された形の記述情報を保持する．これは利用者クラスの数だけ，システム内に複数個存在することが許され，それぞれの外部スキーマに対し，その内容を専門的に管理する役割を果す，アプリケーション管理者と呼ばれる存在を必要とする．

内部スキーマは，実際にデータが物理記憶媒体上に，どのように保持され，どのような方法でアクセスされるか，というデータの計算機内部表現に立脚関連した記述情報を保持する．この内容は，パフォーマンスあるいはコストの面から，ユーザのニーズの変化や技術の進歩に応じて，しばしば変更させられるであろう．その内容を専門的に管理するために，データベース管理者と呼ばれる存在を必要とする．

これらのスキーマの中で，最も中核となるものは概念スキーマである．概念スキーマはデータ記述に関する，意味上のすべての情報源を提供するものであり，外部スキーマや内部スキーマの記述がそれと矛盾したり，それからはみ出すことは許されない．概念スキーマは最も変動しない安定したデータ記述を与えるものであり，大部分のシステム変化の必要に際し，概念スキーマと外部スキーマの間の，あるいは概念スキーマと内部スキーマの間の写像ルールを変更するだけで済ませることができる．この方式によって高いデータ独立性の実現を期待することができる．また結局は情報的見地から押えられなければならない，データの安全性や完全性の保護の問題も，この概念スキーマの存在によって解決の道が開かれることになる．

現時点では ANSI/SPARC では，DBMS の中に 41 種類の構成要素間インターフェースを認識している．しかしそのアーキテクチュアを，ストレージ管理機能部分とデータベース管理機能部分に 2 分して，後者に属する 28 種類のインターフェースから標準化の検討を始めるべきであるとしている．中でも，上述の 3 種類のスキーマに関するインターフェースの定義づけに最大のウェイトを置いている．各インターフェースについて，その要求者，応答者，目的，要求者に見える対象，その対象に対して要求者に許されるオペレーション，をそれぞれ詳細に規定することによって定義づける方法をとっている．

前述したように，ANSI/SPARC レポートは，DBMS に関する具体的提案であるよりも，本質議論に基づく要望である．したがって，たとえば，CODASYL/DBTG 提案における DDL に対応する，スキーマ記述言語は理想的にはどのような言語であるか，を具体的に示しているわけでない．それが満足すべき条件を挙げているにすぎない．すなわち，DBMS の標準化問題に関して言えば，ANSI/SPARC レポートは特定の標準形を提示するものではない．あれやこれの標準形を議論する前に，なによりもまずシステム概念そのものを標準化しなければならないという立場に立っている．まさにその点に，ANSI/SPARC レポートの真の貢献がある，と言えるであろう．またインターフェースの検討は，こと DBMS にかぎらず計算機の世界全体からみても重要で，その意味からも得るべきところがおおいと考えられる．

しかしたとえば，プログラム言語の標準化は，プログラム言語の概念，その理想像が完全に明確にならないと行えないといった性質のものとはかぎらない．現実に FORTRAN や COBOL のような現在のプログラム言語の技術水準からみてかならずしも完全でない言語が標準化されている．これを DBMS やその言語にあてはめれば，この SG の方針だけが唯一つのものであるとはいえなくなるかもしれない．中間報告後の SG の難行が伝えられているのも，この意味では当然かもしれない．

5. おわりに

1970 年ごろのデータベース用言語に関する議論をきっかけとして，最近ではデータベースに関する数多くの論文が続々と発表されており，多くの問題が提起されている．その中で特に重要と思われるものを列挙してみよう．

（1） 理論モデル

Codd の RDB (Relational Data Base) モデルを中心として活発な議論がなされている．最近は知識の表現論としての基本的な問題に接近している．

（2） DD/D (Data Dictionary/Directory) メタデータベース（つまりデータベース中に蓄積されているデータ自身の記述），その他をもっと大がかりに取り扱おうとするもの．将来のオペレーティングシステムとも密接な関係を持つことが予想される．

（3） データベースマシン

XDMS（ベル研），RAP（トロント大学）などデータベースをハードウェア的に扱おうという動きである．将来のデータベースシステムにおよぼす影響は大きいであろう．

（4） 非事務用データベース

従来のパッケージ等はすべて事務処理用のものと考えられるが，エンジニアリングアプリケーションを中心に，推論に役立たせようという動きもある．

（5） そ の 他

データベース変換，機密保護，分散型データベース，データセマンティクスなど．

情報処理学会には，データベース言語研究委員会にかわって，1976 年度からデータベースモデル研究委員会が設置された．1973 年から 75 年にかけての 3 年間にデータベース用言語に関する議論はかなり落ち着きをみせるにいたっており，今後はとくに言語に限らないデータベース（システム）のモデルの議論がさらに盛んになることであろう．わが国では，データベースの研究は欧米におけるほど活発化してないが，これらの研究委員会がその火付け役になることが期待される．

参 考 文 献

1) Data Base Task Group Report to the CODASYL Programming Language Committee April 1971, ACM——DBTG 提案といわれるもの．ACM から購入できる．
2) CODASYL Data Description Language, Journal of Development June 1973, NBS Handbook 113——スキーマ DDL の言語仕様書．邦訳を情報処理学会から出版する計画あり．
3) CODASYL COBOL Journal of Development 1976. Canadian Government 110-GP-1d——COBOL データベース機能を含んだ新版．
4) CODASYL FORTRAN Data Base Facility Version 0.4, Jan. 1976——FORTRAN データベース機能の内部資料．1976 年中に完成公表が予定されている．
5) ANSI/X 3/SPARC Study Group on Data Base Management Systems Interim Report 75-02-08, ANSI Doc. No. 7514TS01——アメリカ規格協会 SPARC データベース SG の中間報告．ACM SIGMOD の bulletin FDT Vol. 7, No. 2(1975) に全文収容されている．
6) Klimbie, J. M., Koffeman, K. I., eds.: Data Base Management, Proceedings of the IFIP Working Conference on Data Base Management, North-Holland 1974.
7) Proceedings of BCS Symposium on the April 1971 DBTG Report, Oct., 1971, the British Computer Society.
8) Proceedings of a two-day Symposium "Implementations of CODASYL Data Base Management Proposals", Oct., 1974, the British Computer Society——7, 8 はいずれも CODASYL データベース言語に関する BCS シンポジウムの報告書である．

（昭和 51 年 4 月 12 日受付）