解　説

プログラム管理およびユーザ言語管理機能を補強した DBMS——DPLS*

椿　　正　明**

1. はじめに

現代企業など，コンピュータを導入した情報処理組織 S は，**図-1** のように人 ($M_1, M_2, \cdots, M_m$) および応用プログラム ($P_1, P_2, \cdots, P_n$) から構成される．各応用プログラムがかつてのスペシャリストを代替してゆくにつれ，人はゼネラスリトを指向してゆくが，本来いずれもインテリジェンスをもつ独立した主体である．この S がスムーズに運営されるには，個々の要素が正常にその情報生産加工機能を果たすとともに，その間の情報流通が正確かつ効率よく行われなければならない．S が大規模化し，複雑化すればするほど，後者の重要性が増大する．

情報流通の問題は次の4つの関係

(1)　$M_i \rightleftarrows M_{i'}$ (人同士)

(2)　$M_i \rightarrow P_j$ (人から応用プログラム)

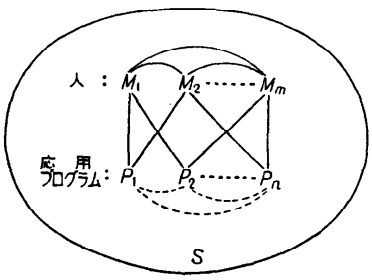

図-1　情報処理組織 S の構成

(3)　$P_j \rightarrow M_i$ (応用プログラムから人)

(4)　$P_j \rightleftarrows P_{j'}$ (応用プログラム同士)

において異なった様相を呈するが，情報流通が正確に効率よく——遠隔地から互いを束縛することなく最小の情報量の移動によって直接的に——行わなければならないことに変わりはない．

情報流通の形式と媒体は，(1)においては自然言語による文書や音声，(2)においてユーザ言語***によるカードなど，(3)においては表図による文書，(4)においては一般にデータ・ファイル形式の磁気記憶媒体が用いられる．

(1)においても作文力，用語や作業の標準化，ファクシミリやテレビ電話など解決すべき点は残されているが，人間の知識，推論力と皆が1つの言語を用いている (S の中で2つの言語が用いられていることはまれである) ことにより，その情報流通は容易である．これに反し，(2), (3), (4) とくに (2), (4) における情報流通は格段と不自由である．(3)にはグラフィックスや COM (Computer Output Microfilm) などの問題があるがその形式と媒体が(1)と変わらないこともあって比較的問題が小さい．

(2)における問題点は (一般論として)

(a)　ユーザ言語が P_j ごとにまちまちである．

(b)　ユーザ言語が what の形式で記述する水準の高いものでなく，情報を where の形式で，その処理を how の形式で記述する水準の低いものであるため，与えるべき情報量が多くまた不安定である．

(c)　ユーザ言語が Open-Ended でなく，新しい用語を次々と追加してゆくことができない．

(d)　Interactive でない．

などがあり，(4)における問題点として

(e)　データ・ファイル形式が P_j ごとにまちまちであり，ファイル変換を施すか，P_j や $P_{j'}$ に

---

* DBMS with the Capability of Program Management and User Language Management——DPLS by Masa-aki TSUBAKI (Chiyoda Chemical Engineering & Construction Co., Ltd.)

** 千代田化工建設(株)

*** ユーザとはここでは M_i すなわち一般ユーザのみを意味する．したがってユーザ言語は一般ユーザが現実世界の問題を記述するために用いられ，これによって記述されたデータは応用プログラムによって処理され，問題の解が出力される．一方プログラマ言語はプログラマが問題の解き方を記述するものでコンパイラによって処理され，応用プログラムが出力される．前者については what の形式が，後者については how の形式が本質的に要求される．従来の固定フォーマットのインプット形式もユーザ言語の1つと考える．

改訂を施してはじめて情報流通が実現する.

(f) データ・ファイル形式が情報を what の形式でなく where の形式で記述する水準の低いものであるため, P_i と P_j' が互いに束縛し合い, また Open-Ended に新しい属性項目を次々と追加してゆくことができない.

(g) P_i と P_j' が互いに遠隔地にあってはならない.

などが挙げられる.

(a), (b)さらに(c)を解決するものとして POL (Problem Oriented Language)[7] が, (d)に対して TSS が, (e), (f)に対してデータベースが, (g)に対してデータベースの分散化が課題となっている. これらのことごとくがソフトウェア界の最重要課題であることは, S における情報流通こそ解決すべき中心課題であることを物語っている.

実際, 人と応用プログラム, また応用プログラム同士間の情報流通は, 人同士間のものに比べてきわめて不自由であり, これがSの情報流通を著しく疎外しているといえよう.

2. DPLS のねらい

DPLS (Database, Program base, Language base Support) はこのような認識のもとに, 図-1 のような情報処理組織の情報流通問題を根本的に解決するために開発されたものである. この組織は管理情報のみならず複雑多様な構造と処理を要求する技術情報を扱うものと想定されている. したがって技術計算によって生産される技術情報が中核的存在であり, 管理情報は大部分これに付随して発生するというケースも一般的なものとして扱われる. データベースは, DPLS においてもその中心課題となっているが, 事務計算の合理化のためにデータ管理の拡張機能としていわゆる DBMS が提供しているデータベースとはその位置づけを異にしている.

DPLS では組織全体が対象である。特殊なモデルを適用してその一部のみを対象にする——たとえばトリー構造によって扱いうる特定の管理情報のみを扱う——ことはその趣旨に反する. したがって基本的なモデルはすべての応用分野の要求をみたす一般的なものでなければならない.

組織はきわめて大規模で複雑である. 組織は絶えず拡張し変化する. したがってその要素が独立であること, 他の要素に束縛されていないことが必須条件となる. こうして応用プログラムが互いに関連し合う場としてのデータ・ファイルは, プログラムのデータ独立性を中途半端にでなく完全に保証するデータベースでなければならない. データベースが常に管理され, 柔軟に忠実に現実世界を表現するためには, 複数データベースから構成されることが有利である. こうして DPLS の構図は**図-2** のように表わされる.

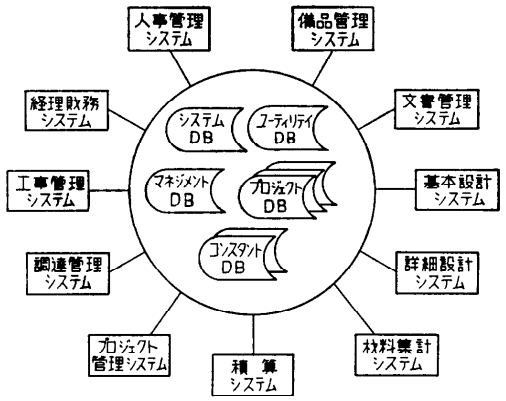

図-2 DPLS の構図

各応用プログラムはデータ・ファイルに対する入出力, ユーザとのインタフェイス, (サブ)プログラムの制御など共通の側面を持っている. これらを汎用的にサポートするモデルをxと書くと各応用プログラム $(P_1, P_2, \cdots, P_n)$ は因数分解されて次のようにその固有部分 (p_1, p_2, p_n) に分割される.

$$(P_1, P_2, \cdots, P_n)=(xp_1, xp_2, \cdots, xp_n)$$
$$=x(p_1, p_2, \cdots, p_n)$$

DPLS や DBMS はこのxに対応づけられるソフトウェアであるが, いわゆる DBMS は P_i として事務計算のみを対象にし, かつデータベース機能のサポートに主眼を置いている. P_i として対象とする範囲が広ければ広いほどxの汎用性は高く, またxとして抽出する部分が大きければ大きいほど, 固有部分 p_i が小さくなる. そしてxが完全なデータ独立性をもつデータベース機能を供給するとき, p_i と p_j' は互いに完全な独立性を保ちつつ自由な情報流通を行うことができる. このとき同時に組織の拡張にともなう新しい固有部分 p_{n+1} の追加も容易に行われる.

xによって情報流通や拡張の問題が解決され, 小さな独立した応用プログラム (固有部分 p_i) を作ればよいことになるとき, 応用プログラム開発はきわめて容易になる. 近年応用プログラムが巨大化し, その開発に2年も3年もかかるケースが増加しつつあった.

このためより精緻なシステム計画が要求され，これに数カ月を要するということも少なくなかった．システム計画は本来試行錯誤を要する最もむずかしい作業であるが，プログラム開発に時間と金がかかるという理由で当初から最終的なものが要求され人間わざでは無理となりつつあったのである．xの導入はしたがってシステム計画をも容易にすることができる．

DPLSはこのようなx，すなわち完全なデータ独立性をもつデータベース機能を中心として，汎用性の高い，より多くの共通機能——たとえばユーザとのインタフェイス，(サブ)プログラムの制御など——を供給すべきものとされている．したがってDPLSのねらいはこのように情報流通の問題を一括して根本的に解決し，

Easy Planning (Application Designer)
Easy Analysis (Application Designer)
Easy Programming (Application Programmer)
Easy Training (Application Programmer)
Easy Expansion (Application Programmer)
Easy Utilization (Application User)

の環境を用意し，技術情報および管理情報を扱う複雑な情報処理組織の円滑な運営を図るものということができる．

3. DPLSの特徴

DPLSの第1の特徴は共通機能xとして

D—データベース
P—プログラムベース
L—ランゲージベース

の3つを抽出し，これらを平等にサポートしようとするところにある．ここでプログラムベースとはプログラム群が，重複のない独立した(サブ)プログラムの集まりとして存在していること，またランゲージベースとはコマンドおよびコマンドマクロ群がやはり重複のない独立したコマンドおよびコマンドマクロの集りとして存在していることを意味している．すなわちデータ管理とともに，プログラム管理およびユーザ言語管理を同様の合理的思想のもとにサポートする．

これは多様な情報処理およびこれに付随した多様なユーザ・インタフェイスを要求する技術計算をサポートするために必然的に備えられた特徴である．OSの機能と比較してみるならばDはデータ管理の，Pはジョブやタスク管理の，またLは言語プロセッサの拡張機能と見ることができるから，いわゆるDBMSとDPLSの相違は図-3のように表わすことができるであろう．またDは応用プログラム同士間の，Lは人と応用プログラム間の情報流通問題を根本的に解決しようとするものである．

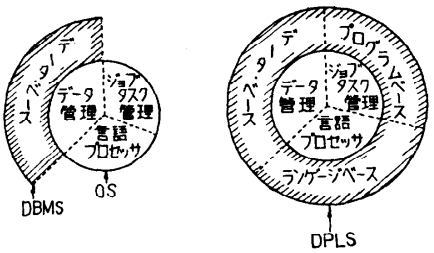

図-3 DBMSとDPLSの機能

DPLSの第2の特徴は完全なデータ独立性を実現するために用意された

情報構造モデル
データ構造モデル
記憶構造モデル

からなる3水準データモデルであろう．これはSenko[8]，石田[9]，Chen[10]らの方向とも一致する．情報構造モデルは最も水準の高い情報表現すなわちwhatの形式の表現を実現するもので情報を属性番号と事象番号で指定する．これによって完全なデータ独立性が実現され，またユーザは記憶媒体の物理的制約を離れて，現実世界の情報を忠実に表現することができる．

現実世界の情報構造モデル同定のさい発生するユーザによる任意性は，DPLSの情報構造モデルの適用によって著しく減少する．このためシステム分析はルーチンワーク化し，きわめて容易なものとなる．すなわちDPLSは従来きわめて困難とされていたシステム分析に対するサポートを提供する．

データ構造モデルでは情報は物理サブファイル番号と記憶レコード番号で指定され，アクセス効率，機密保護，排他制御，可変長データなどデータベースにかかわる煩雑な問題が解決される．記憶構造モデルは所要の記憶スペースを具体的なデータセットの所定サイズ，所要個数のページとして与えるものであり，情報はデータセット，ページ番号，ページ内相対番地により指定される．こうして情報構造モデルによってwhatの形式で表現された情報は順次，データ構造モデルおよび記憶構造モデルによって変換されてwhereの形式となり，具体的な物理媒体に対応づけられる．

このように情報構造モデル，データ構造モデルおよび記憶構造モデルの役割りが明確に分離されているこ

とは，システム分析やデータベース管理を容易にする上で非常に有利である．この3水準データモデルにおける役割りの分離は，情報構造モデルによる完全なデータ独立性とともに，複雑な技術情報を扱う必要によって備えられたものと見ることができる．

4. データベース機能

DPLS は次に示すような通常のデータベース機能すなわち

(1) データ検索，登録，更新
(2) 関係表現，検索，登録，更新
(3) 高速アクセス補助手段
(4) レポートライタ
(5) 排他制御
(6) 機密保護
(7) エラー回復
(8) データベース定義
(9) データベース構成
(10) データベース保守（拡張，再構成，転送，整頓）

などを備えているが，さらに次に挙げる著しい特徴をもっている．

(11) 高水準情報表現
(12) 高性能情報表現
(13) 段階的データベース定義
(14) ページング入出力
(15) 3水準コンテクスト
(16) 多重コンテクスト
(17) 複数データベース環境
(18) DD/D
(19) 単位変換
(20) エラー処理サポート
(21) ファミリー管理

紙面の制約からここでは(11), (12), (13)についてのみ述べる．これらはいずれもデータベースのデータモデル，すなわち3水準データモデルに関連する特徴である．他の項目については，DPLS 概要説明書[4]，および論文[5),6)]を参照されたい．

(11)高水準情報表現とは情報を where の形式でなく what の形式で表現することである．このために情報代数[1)]として提唱された事象 (Entity)，属性 (Attribute)，属性値 (Value) の概念が用いられる．たとえば

"Mr. Smith の給与は3万ドル/年である"
"Mr. Smith の子供は Jack と Betty である"
"経理部のマネジャーは Mr. Brown である"

という情報は図-4——これは情報構造モデルの具体的な姿でもある——のように事象を縦に，属性を横に配置しその交点に属性値をかきこんだ表によって表わされる．この場合 Mr. Smith と経理部は異なった属性の集合と対応づけられるので，異なった事象種 "従業員" と "セクション" に属するものとして扱われる．各事象種ごとの表を論理サブファイルと呼ぶ．ここにはスペースの制約がないので Jack と Betty のように多値属性値（可変長データ）も同様に表わされる．属性にはユニークな番号が与えられる．事象には事象種ごとにユニークな番号が与えられる．これらは属性値のありかを直接示すものすなわち where を記述するものではない．その情報が何者であるか，すなわち what を記述するものである*(* 次頁参照)．したがっ

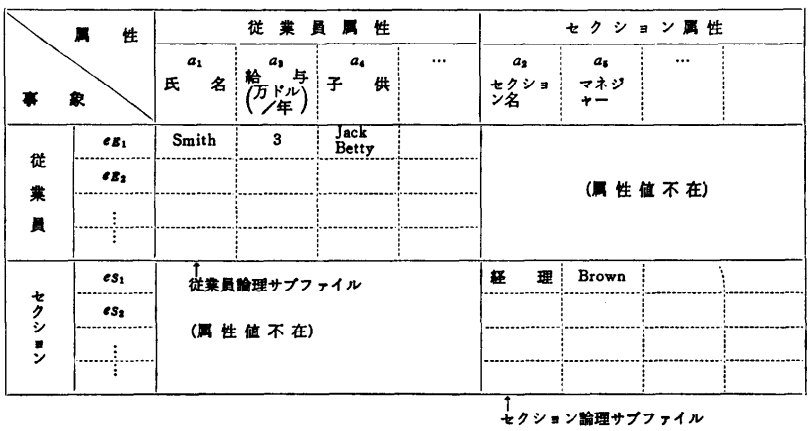

図-4 情報構造モデル

て属性や事象の追加は他に影響を与えることなく自由に行われ，情報構造モデルとインタフェイスを持つ応用プログラムは互いに完全に独立であることができる．なおDPLSの論理サブファイルはRDM[2] (Relational Data Model) のリレーションの働きを果たすことができる．

(12) 高性能情報表現とは

(a) 事象間の関係情報

(b) 事象自身の情報

を忠実，単純かつむだなく表現することである．(a)として同種あるいは異種事象間のネットワークあるいはトリー関係を表現することが要求されるが，RDM[2]のForeign Keyと類似のCPTR法および2項関係によるREL法さらにDBTGモデル[3]のSetと類似のEPTR法によって自由に効率よく表現される．(b)としては多値属性値の配置や冗長性の排除のためのデータの共有などが問題になるが，すべてデータ構造モデルによって合理的に解決されている．

(13)段階的データベース定義とは，まず情報構造モデルを定義し，次にデータ構造モデル，最後に記憶構造モデルと順次段階的に定義してゆくことをいう．応用プログラムの開発は情報構造モデルを定義すれば，直ちに開始できる．データ構造モデルはアクセス効率，機密保護，排他制御などを考慮して定義あるいは変更されるが，これは応用プログラム開発中，開発後，運用中など何時行ってもよい．プログラム開発前に行うべきことは現実世界の情報構造がどうなっているかを分析することだけである．

5. プログラムベース機能

今プログラムAにおいて条件1のときサブプログラムPを，条件2のときサブプログラムQを実行させたいとする．プログラムAを**図-5**(a)のように書き，普通にロードモジュールを作ると，プログラム・ライブラリーは**図-6**(a)のようになる．なおここで別のプログラムBもPとQを呼んでいるものとする．ところがP,Qとの結合を行わずにロードモジュールを作ると図-6(b)のようにA,P,Qはバラバラに存在し，実行時結合が行われる．またプログラムAを図-5(b)のように書くと，当然のことながらAはP,Qなどと結合されずプログラム・ライブラリは図-6(b)のようになる．

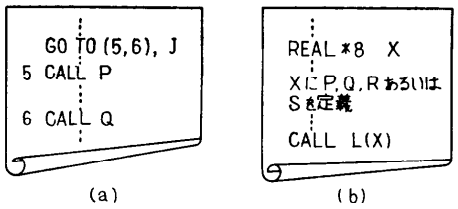

図-5 プログラムAの書きかた

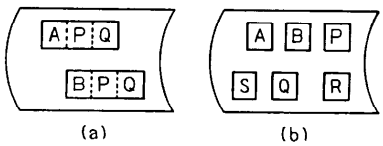

図-6 プログラム・ライブラリの状態

ダイナミックリンク(V)と呼ぶ図-5(b)のプログラム方式はきわめて弾力性に富むもので，ユーザは呼ぶべきサブプログラム名を実行時P,Q,R,Sと任意に指定できる．プログラムベースと呼ぶ図-6(b)のプログラム・ライブラリは図-6(a)に比べ冗長性がなく，保守が容易でかつスペースのむだを生じない．データベースの導入によって独立した小さな単位となった応用プログラムは，ここでさらにサブプログラムとも切り離され，一層独立した小さな単位となり，システム全体のフレキシビリティを増大させる．DPLSはこのようなダイナミックリンク機能によってプログラムベースを実現している．

6. ランゲージベース機能

DPLSの用意するコマンドは次のようなきわめて人間系に近い高水準のユーザ言語である．すなわち，いま，年令30才，年収3万ドルないし4万ドルの従業員をさがしたいとき，一般ユーザは

FIND EMPL, AGE 30, SALARY 30000,～, 40000;

のように指定する．このためにプログラマは事前にコマンドFIND EMPLを

```
ADD COMMAND: FIND EMPL,
  PROGRAM 'FEMPL',
  I (101) AGE, SALARY 0, 0, 100000;
```

のように書き，これをコマンド定義プログラムを通じてコマンド・ライブラリに登録する．同時この処理を実行するプログラムFEMPLを作成しプログラム・ライブラリに登録しておく．なおSALARYにつづくデータはあらかじめ標準値を設定しておくものである．

---

* たとえばFORTRANプログラムにおいてあるデータベース(IDB)のある属性(JAI)のある事象(ID)の値(VAL)をとり出すときは

```
CALL DBI (IDB, JAT, ID, VAL)
```

のように書かれる．VALとしてアレイを用意すれば可変長データが扱える．

コマンドは互いに独立であり，登録は容易にどこまでも追加変更できる．情報およびその処理は what の形式で指定される．このため Open-Ended POL とも呼ばれる．標準値が設定され，対話モードでもバッチでも同様に使用できる．

またユーザは一連のコマンドをログしたりさらにオーソライズしてカタログすることができる．これをコマンドマクロと呼ぶが，これらも互いに独立の要素として記憶される．

DPLS はこのような独立性の高いコマンドおよびコマンドマクロから成るユーザ言語機能，ランゲージベースを提供する．汎用性が高いため，技術計算，事務計算すべてに適用でき，人と応用プログラムの間の情報流通の混乱を解消することができる．

7. おわりに

DPLS は元来千代田化工建設（株）における基本設計，詳細設計，積算，調達，プロジェクト管理，工事管理などを統一的にあつかうべき基本ソフトウェアとして 1970 年ころより研究開発がすすめられて来たものである．種々のプロセス設計システムや機器設計システムまたは文献検索システムをはじめとする種々の管理情報システムへの実験的応用をへて改訂を重ね，当初の目標——プログラム開発期間およびコスト 1/2～1/10，プログラム保守期間およびコスト 1/10～1/20——をほぼ満たしうる状態に到達したといえる．実際通常の管理情報システムの開発は 3 週間程度で可能となっている．

DPLS の応用にあたっては現在次の環境が必要である．

- 機　種: IBM 370 (360) あるいは国産Mシリーズ
- オペレーティングシステム: OS (VS)
- 主 記 憶: 256 KB 以上
- ディスク: IBM 3330, 2314, 2311 相当
- テ ー プ: 不要
- 端　末: IBM 3270, 2741 相当

またサポートするホスト言語は FORTRAN, COBOL, BAL である．

DBMS にはミニコンへの応用，分散化，オペレーティング・システムとの融合など多くのテーマが残されているが，さらに DPLS には従来おくれていた技術計算あるいはトータル CAD (Computer Aided Design) システムへの応用を一層強力にすすめることが要請されていると思われる．

参 考 文 献

1) CODASYL Development Committee: An Information Algebra-Phase I Report, Comm, ACM, Vol. 5, No. 4, pp. 190～204 (1962).
2) E. F. Codd: A Relational Model of Data for Large Shared Data Banks, Comm. ACM, Vol. 13, No. 6, pp. 377～387 (1970).
3) CODASYL Data Base Task Group: April 1971 Report, ACM, New York (1971).
4) 千代田化工建設(株): DPLS 概要説明書(1976).
5) M. Tsubaki: DPLS-Database, Dynamic Program Control & Open-Ended POL Support, Proc. of Workshop on Data Bases for Interactive Design, Waterloo, Ontario (1975).
6) M. Tsubaki: Multi-Level Data Model in DPLS-Database, Dynamic Program Control & Open-Ended POL Support, Presented at International Conference on Very Large Databases, Framingham, Mass. (1975).
7) IBM: PLAN Description Manual
8) M. E. Senko, E. B. Altman, M. M. Astrahan, P. L. Fehder: Data Structures and Accessing in Database Systems. IBM Systems Journal, Vol. 12, No. 1, pp. 30～93 (1973).
9) 石田番也: データベースのセマンティックスとインプリメンテーション，情報処理学会，データベース研究会資料，74—10 (1974).
10) P. P. Chen: The Entity Relationship Model_ Toward a Unified View of Data, ACM Trans on Database Systems, Vol. 1, No. 1, pp. 9～36 (1976).

(昭和 51 年 5 月 19 日受付)
(昭和 51 年 7 月 26 日再受付)