分散データベースとDD／D

椿　正明
(株) 日本レステミックス

1. まえがき

1970年代はデータベースの時代であった。多くのDBMSが発表され、実用化された。多くの秀れた研究論文が発表され、進歩が見られた。ところで次の1980年代は何の時代であろうか。私はこれを分散データベースとDD／Dの時代と見る。その分散データベースとDD／Dとはどのようなものであろうか、以下検討し概念設計を行なってみよう。

その際我々の開発した汎用ソフトウェアサポートDPLS[10],[11]（Database, Program base, & Language base Support）の経験がやはりその大きなベースとなっている。DPLSではすでに自己記述型のDD／Dデータベースが内蔵され、異なった応用分野の複数個のデータベースが統轄管理されていた。分散データベースはこの自然な拡張として考えられる。これは図－1のパス①②③として表わされる。通常の分散データベースはパス①②③で考えられることが多いであろうから若干異ったイメージをもたれるかもしれない。

応用分野の数 ＼ 処理	集中	分散
1ケ	①	②
nケ	②′	③

図－1　分散処理へのパス

2. 分散データベースの環境と運用

分散データベースの環境としてnケの応用分野にかかわるmケのデータベースがkケ所のノードで処理されるものを考えよう（m＞n, k）。mケのデータベースは相互に関連する1つの組織体の情報をもれなく表現するものとする。そのうちのいくつかは既存のデータベース（OSファイルを含む）であり、そのためいくつかの（異質な）DBMSが稼動される。ノードは本社、支店、出張所などであろう。人事管理、生産管理、プロジェクト管理などのデータベースは、それぞれ応用分野が異るとして独立に生成運用されるであろう。応用分野を決めるのはアトリビュートの集合であると考えられる。これをデータベースタイプと呼ぶ。1つのデータベースタイプについていくつかのデータベースオカレンス（オカレンスはしばしば省略）が生成される。

1970年代のデータベースのスコープは開発の都合などから自然発生的に決められ、組織体のニーズにトータルに応えようというより、その一部をつまみ食いする結果となった。これからの（分散）データベースとしては合理的な基準にもとずいて計画的にそのスコープを決めて行きたい。これはデータベース設計の出発点をも科学的に決定し、トップダウンアプローチを徹底させようとするものである。

データベースのスコープの決定に当って次の3つの観点
(1) アトリビュートの集合（データベースタイプ）
(2) エンティティの集合
(3) タイムフレーム（データの有効期間，更新の頻度など）
が重要である。データベースのスコープの決定基準はDBA（データベース管理者）によるデータベースの運用管理の単純化を目標の1つとして作成されるべき

であろう。データベースは一気開発的に開発されるというより，逐次拡張されるべきものであろう。したがって開発のタイミングによってスコープが決まると不都合が生じ易い。目標の第2はコストであろう。分散処理はコストの要請にもとずくことが多いであろう。

DBAにとって小さいデータベースほど運用管理が容易である。関係のないアトリビュートは極力排除したいし、関係のないエンティティは極力排除したい。また関係はあっても更新（検索ではない）のタイミングがはっきり異るデータは（AREAなどによらず）はじめからデータベースとして分けておいた方が運用管理が簡単になろう。しかしあまり小さくしすぎると1つのジョブの実行にいくつものデータベースが関係し、また関係の表現やインテグリティの保証などせっかくのDBMSの機能が活用できない。とくに1つのジョブが複数個のデータベースを更新するとリカバリ操作が複雑化する。

したがってデータベースのスコープの決定基準として

(1) 密接に関係する（同時に独立にあるいは波及的に更新されるなど）アトリビュートを一括してデータベースタイプを定義する．

(2) 目的とか所在（したがって処理ノード）などを同じくするエンティティを一括してデータベースオカレンスを定義する．

(3) タイムフレームを同じくするデータを一括してデータベースオカレンスを定義する．

などが得られる。要するにできるだけ独立性の高い小さいデータベースを構成すればよいと考えられる。

運用もこの趣旨に沿うべきである。分散データベースシステムは運用によってはいたずらに問題を複雑化させ混乱を招く危険性がある。[3] ノード間の（運用の）独立性をできるだけ高く保つことが重要である。ノードの独立性はDBMSの救済しえない障害によって損われる。データベース間の独立性も障害によって損われるがDBMSによる障害回復時，困現性の欠除によって検出される。したがって困現性を保証する障害回復操作をノードあるいはDBMSとして行なうとき，最も高い独立性を与えるものとして，次の運用基準を提案する。

(1) すべてのノードに共通の次の2種類の運用時間帯を設定する．

時間帯Aに更新が自動的に管理される通常時間帯

時間帯Bに更新が半自動的に管理される夜間、週末などの時間帯

(2) 各データベースに安定度（更新の不必要度）を割つける。最も安定度の低いものを安定度1とする。安定度1のデータベースの更新ジョブにおいて参照されるデータベースは安定度2以上とする。プログラムライブラリにも相当する（比較的高い）安定度を割つける。

(3) 時間帯Aでは安定度1のデータベースのみ更新が許可されるものとする。安定度2以上のデータベースは時間帯BにおいてDBAの管理のもとに更新する（もちろんコピー上での更新テストは時間帯Aにおいて可能）。

(4) データベースのReplicationは安定度2以上のものについてのみ許可する。

安定度＼時間帯	A	B
1	W & R	W & R
2以上	R	W & R

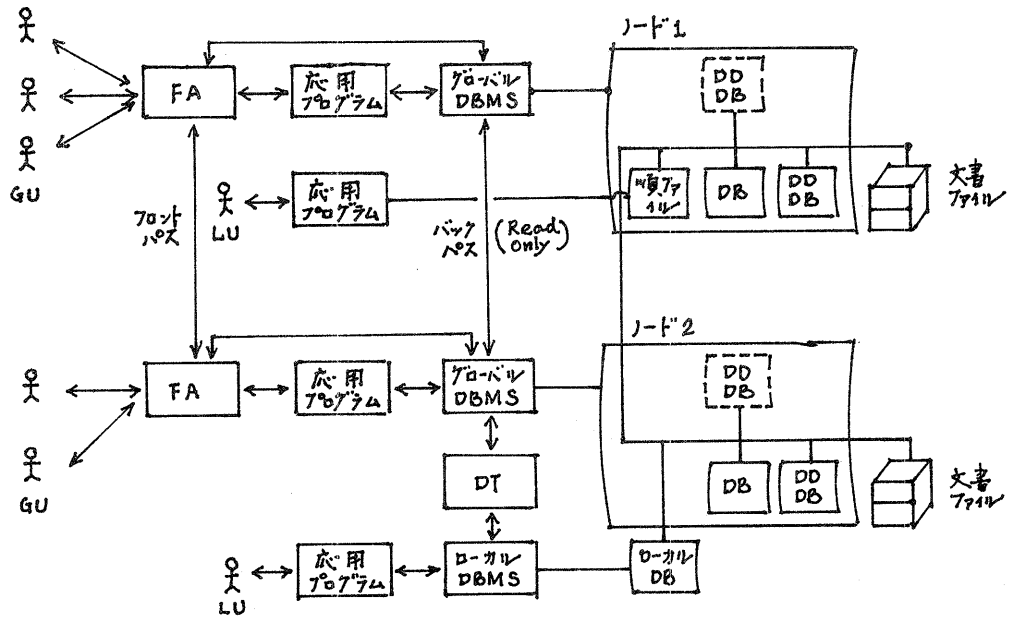
図－2　許可される操作

こうして時間帯Aにおいて更新される安定度1のデータベースは互いに独立とな

り、参照するデータベースの値および走るべきプログラムは変らないから、障害回復操作は他のノードやデータベースと独立に実行でき、両現性が保証される。もし時間帯Aにおいて更新したいデータベースがあり、この更新ジョブにおいて他の安定度1のデータベースが参照されるなら、両者は統合して1つのデータベースとしなければならない。たとえこれらのデータが地理的に離れて発生あるいは使用されるとしても分散処理は許されないことになる。またインテグリティを保つために安定度1のデータベースの更新にともなって安定度2のデータベースに更新を波及させるときもこれを時間帯Bまで延期しなければならない。これはしかしデータベースの感度を下げて、DBAの運用を容易にするために有効であるとも考えられる。

3. 分散データベースのアーキテクチャとDD/D

以上の考察にしたがって想定される分散データベースのアーキテクチャを図-3に示す。簡単のため2ケのノードから成るものとする。各ノードにはDDDB（データディクショナリデータベース）によって統括管理されるデータベース群が配置される。この意味では順ファイルや文書ファイルすら1種のデータベースとして扱われる。（既存の）異質なデータベースもローカルDBとして組みこまれる。DDDBは自己記述的であって、他のデータベースと同等に位置づけされ



記号
FA : Front Analyzer
DT : Data Translator
GU : Global User
LU : Local User

注
IR, レポートライタ, ローダなどのユーティリティは応用プログラムとして表わす

図-3　分散データベースのアーキテクチャ

る。DDDBの安定度は高く設定され、内容の大部分はReplicateされる。更新は時間帯Bに同期をとって行なわれる。

GU（グローバルユーザ）はノードを意識しないで分散データベースにアクセスする。LU（ローカルユーザ）は従来どうりノードを意識して使用する。ノード間には2つのパス，フロントパスとバックパスが用意されている。フロントパスはジョブストリームをそっくり別ノードに送るもので必らずしも回線を意味しない。テープやカードを郵送するケースもあり得る。バックパスは回線を通じて参照のみを行なうものである。もしここで更新を許可すると障害回復操作を両ノード同期の上行なわねばならず、ノードの独立性を失うからである。FA（フロントアナライザ）はユーザジョブをどこのノードで行なうべきか判断し処理する機能である。

グローバルDBMSは常時DDDBを参照して応用プログラムの要求に応える。必要とあらば他ノードのデータやDT（データトランレータ）を介して異質なデータベースのデータを検索する。

DDDBには各データベースの構造，データの所在，冗長の状態，Replicatedか否か，更新波及，アクセスの条件など分散データベースシステムの円滑な運用管理に必要なすべてのデータが整然と収納されている。[5] DDDBに記載すべき情報は大部分システム設計時に発生する。このデータを発生の都度投入し、DDDBをいわゆるシステム開発ワークベンチとして使用するのが今後の方向である。この場合はDDDBの開発者用コンテクスト（コピー）を用意し安定度1のデータベースとして使用する。このワークベンチはDDDBのみを含む退化したノードとして位置ずけられる。

このようにDD/D機能は分散データベースにとって致命的な重要性を持つが次に挙げる種々の用途をとりこんで1980年代大きな飛躍が予想される。

(1) 自己記述型DDDBとしてDBMSの稼動を支援する
(2) DBAのデータベース運用管理を支援する
(3) コンピュータオペレーションやリソース管理を支援する
(4) レポートライタを支援する
(5) ソフトウェア開発を支援する
(6) MISの中枢として企業情報活動を全面的に支援する

4. データモデル

次節でDDDBの構造を概説するに先立ち，記述のルールとしてデータモデルについて若干述べる。概念モデルに限定する。DPLSにおいては情報構造モデル[10],[11]と称したものである。[1] 元来CODASYLの情報代数[2]にもとずいて構築されたが、P.P. ChenのE-Rモデル[3]と見かけ上似ている。E-Rモデルではエンティティタイプとリレーションシップタイプを区別し、後者を図上で菱形などで表示するが、ここではリレーションシップもエンティティの1種としてしか考えない。したがって図表現はエンティティタイプダイヤグラムと呼ばれ短形と矢印のみから成る。この意味ではバックマンダイヤグラムに似ている（矢印がリストの存在を意味することはない）。

本概念モデルの特徴として次の2点を説明する。

(1) アトリビュートは唯一のエンティティタイプに属し，唯一のロールしか持

たない。[7]

(2) エンティティタイプとして通常のもののほか，リレーションシップエンティティタイプおよび汎化エンティティタイプが識別される。

(1) はたとえば従業員ファイル（正しくはエンティティタイプ）中の所属部門番号と部門ファイル中の部門番号とは別のアトリビュートと見なすことである（もちろん両者が無関係ということでない．前者が後者に対する外部キー[4]として定義されていることが概念モデルの中で定義される）。両者は属するエンティティタイプ（従業員，部門），そこでの機能（外部キー，キー）ばかりでなく，更新のタイミング（従業員の異動，組織改革），更新の担当者（人事，総務）などが異るのであるから，いわゆるデータ項目として同一視するより，別属性として扱う方がメタ情報がすっきりと規定できる。アトリビュートの定義によって同時にエンティティタイプが規定されるから，これによって概念モデルが定義される。原則としてアトリビュートは自由に追加されるから，概念モデルは自由に生長変化する。このようなアトリビュートをどう選んでデータベースタイプを構成するかはDBAの裁量である。

(2) はエンティティタイプのキーを構成するアトリビュートと図表現上の規則に特徴がある。リレーションシップエンティティタイプのキーは関係するエンティティに対する外部キーの結合によって構成される。たとえばリレーションシップエンティティタイプ履習のキーは学生，科目に対する外部キーの結合によって構成される。学生あるいは科目のエンティティが削除／変更されると履習の関連するエンティが削除／変更されなければならない。このようにリレーションシップエンティティタイプのエンティティには独立性が欠けている。履習には得点などの属性が追加されうる。このためエンティティタイプの1種として扱った方が都合がよい。nにmの対応関係の表現に多く用いられる。図上では矩形の右側に縦線を1本追加して表現する。

汎化エンティティタイプはいわゆるGeneralization[10]を表現するものである。たとえば自動車のファイル，オートバイのファイルに加えて乗物のファイルを用意したいとき，乗物のキーを自動車およびオートバイに対するエンティティタイプ識別子と自動車あるいはオートバイのキーを結合して構成する。ここでも汎化エンティティタイプのキーの1部に外部キーが用いられているので更新波及が規定される。エンティティタイプの識別子はメタデータである。メタデータがデータベースの通常の値として用いられるのは自己記述型データベースの利点であろう。汎化エンティティタイプは図上では矩形の左側に縦線を1本追加して表現される。

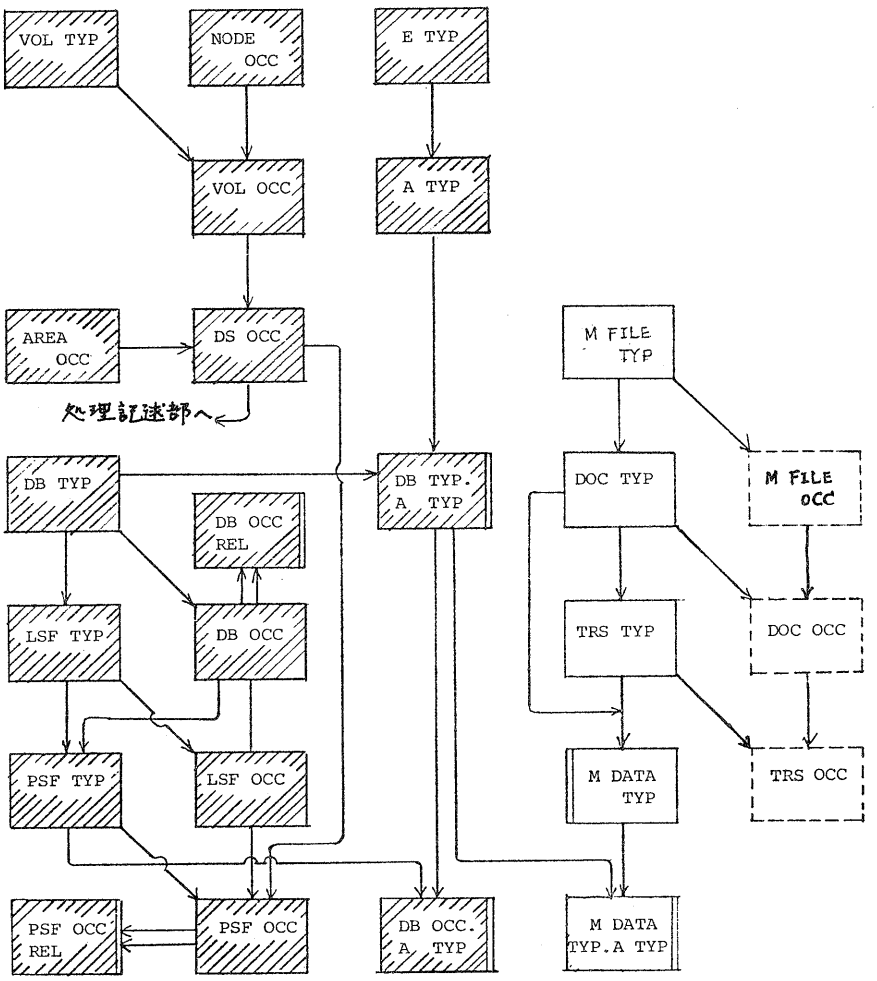
5．DD／Dデータベースの概念モデル

DD／Dデータベースあるいは DDDB の構造を設計し概念モデルとして示そう。概念モデルはアトリビュートの定義によって規定されるが，紙面の制限があるので，エンティティタイプダイヤグラム（図－4(a), (b), (c)）によって全体の構成を示し（エンティタイプ名については図－4(d) 参照），特徴的なエンティティタイプおよびそのアトリビュートについて若干の説明を行なうにとどめる。

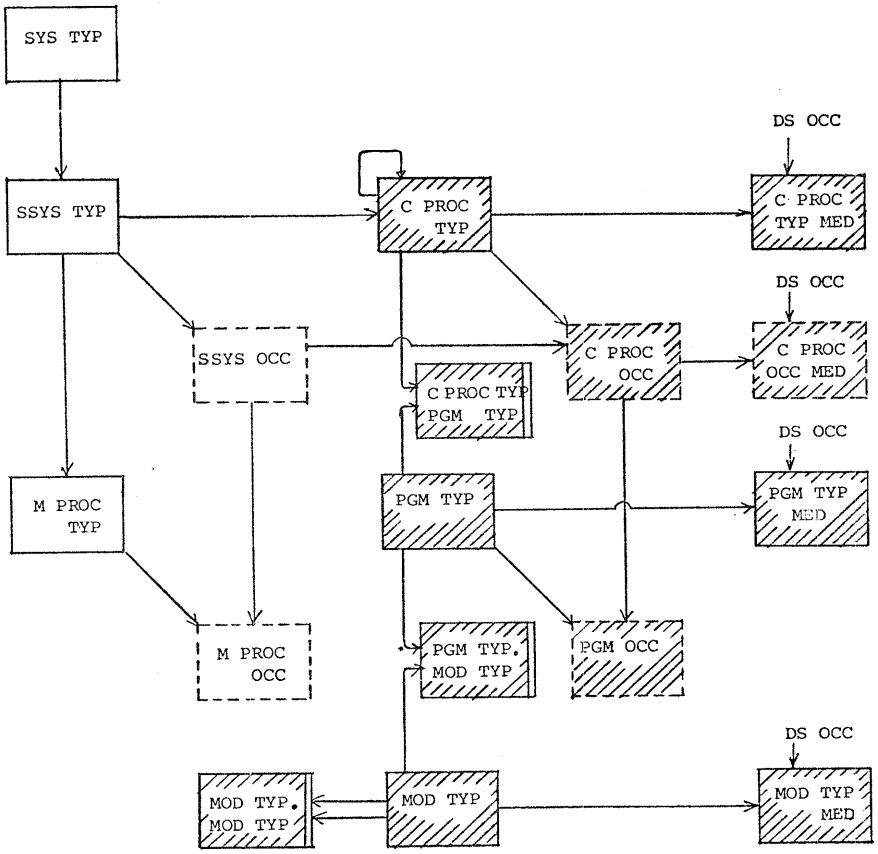
DD／Dデータベースに記述されるエンティティタイプはほぼデータに関するものと処理あるいはプロシジャーに関するものに2分される。前者が図－4(a)に，後者が図－4(b)に示される。どのような処理にどのようなデータが関与す

るが、両者の関係として図－4(c)に示される。処理はモジュールのレベル、プログラムのレベルあるいはさらに大きいレベルでとらえられる。データもアトリビュートのレベル、レコードのレベル、データベースタイプのレベルなど種々のレベルでとらえられる。したがってこれらは一旦汎化エンティティタイプ、PROC TYP, DATA TYPとして汎化され、その上で関係が記述されている。このときリレーションシップエンティティタイプが用いられているが、その属性として、データが入力、出力、参照などどのような形で関係しているかも併記される。

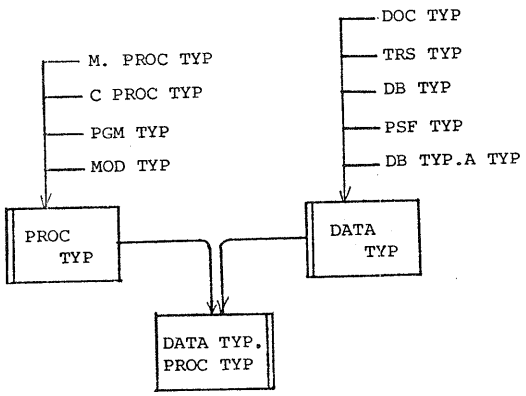
また1つの大きな特徴としてコンピュータに直接関係のない、人間系の情報処理に関するエンティティタイプが含まれている。これらは図－4において斜線のない白い矩形で示されている。したがってDD/Dデータベースによってどのような文書にどのようなアトリビュートが記述されるべきか、それはどのような人によってどんなタイミングに作り出されどこに配布されるかが分る。



図－4(a) データ記述部に対するエンティティタイプダイヤグラム



図－4 (b) 処理記述部に対するエンティティタイプダイヤグラム



図－4 (c) データと処理の関係に対するエンティティタイプダイヤグラム

AREA OCC: Area occurrences.
A TYP: Attribute types or Attributes.
C PROC OCC: Computer procedure occurrences.
C PROC OCC MED: Media of computer procedure occurrences, e.g. journal log.
C PROC TYP: Computer procedure type. The type of computer job/job-step.
C PROC TYP MED: Media of computer procedure type
C PROC TYP. PGM TYP: Relationships between computer procedure types and program types.
DATA TYP: Data type generalizing DB TYP, PSF TYP, DB TYP, A TYP, DOC TYP, and TRS TYP.
DB OCC: Database occurrences.
DB OCC.A TYP: Local attribute types for database occurrence.
DB OCC REL: Various relationships between database occurrences.
DB TYP: Database types.
DB TYP.A TYP: Local attribute types for database types.
DOC OCC: Document occurrences.
DOC TYP: Document types.
DS OCC: Data set occurrences.
E TYP: Entity types.
LSF OCC: Logical subfile occurrences
LSF TYP: Logical subfile types
M DATA TYP: Manual data types generaling DOC TYP and TRS TYP.
M DATA TYP.A TYP: Attribute types which appear in the manual data types.
M FILE OCC: Manual file occurrences.
M FILE TYP: Manual file types
MOD TYP: Program module type
MOD TYP MED: Media of program modules
MOD TYP.MOD TYP: Relationships between calling program modules and called program modules.
M PROC OCC: Manual procedure occurrences.
M PROC TYP: Manual procedure accompanied with subsystem types
NODE: Node Occurrences.
PGM OCC: Particular execution of programs
PGM TYP: Computer program
PGM TYP MED: Media of programs
PGM TYP.MOD TYP: Relationships between program types and included module types.
PROC TYP: Procedure type generalizing M PROC TYP, C PROC TYP, PGM TYP, and MOD TYP.
PROC TYP. DATA TYP: Relationships between procedure type and data type.
PSF OCC: Physical subfile occurrences.
PSF OCC REL: Various relationships between physical subfile occurrences.
PSF TYP: Physical subfile types.
SSYS OCC: Subsystem occurrences. Particular execution of sub-system types.
SSYS TYP: Subsystem type of an application system.
SYS TYP: Application system type.
TRS OCC: Transaction occurrences.
TRS TYP: Transaction types.
VOL OCC: Volume occurrences belonging to a volume type.
VOL TYP: Volume types.

図－4(d)　エンティティタイプ略名の説明

多データベース環境を特徴ずけるエンティティタイプとして、DB TYP, DB OCC DB TYP.A TYP, DB OCC REL などがある。それぞれデータベースタイプ，データベースオカレンス，データベースタイプへのアトリビュート（タイプ）の取り込み，データベースオカレンス間の関係である。データベースオカレンスのアトリビュートとしてその属するデータベースタイプ名，DBA，ノード，安定度，ロック状態などが記述される。データベースオカレンス間の関係としては一方が他方のコピーであるかなど種々の関係の種別が記述される。

分散データベースを特徴ずけるものとしてNODE OCCがある。

E TYP, A TYPからDB TYPまでが概念モデルに属する。エンティティタイプ，アトリビュート（タイプ）についての記述は組織体について1個しか

ない。これはデータマネジャが管理する。

LSF TYP, DB OCC から PSF OCC までは内部モデルであってDBAの管理下にある。したがって異質なDBMS下のデータベースに対しては若干工夫を要するところである。PSFは通常のロジカルレコードに相当し、構成するアトリビュートの順序に応じて値がストアされるものと規定される。LSFはPSFの論理的な概念で、アトリビュートに対する値の並び方は全く規定されない。LSF TYPはデータベースへのアトリビュートの取り込みが決まれば自動的に決まるものである。これらLSFやPSFにおけるタイプとオカレンスの分離は、あるエンティティに対するあるアトリビュートの値が、ケーススタディなどによって何重にも存在する場合にそなえるものである。

VOL TYP, VOL OCC, AREA OCC, DS OCCは記憶モデルに属する。

処理記述部では人間およびコンピュータによる一連の処理単位として定義されるサブシステムが特徴的であろう。サブシステムのアトリビュートとして運用の頻度、トリガーとなるイベントなどが記述される。なお実線で書かれた矩形はDD/Dデータベースの外に定義されるべき関係の深いエンティティタイプである(コンピュータの運用管理システムなどに含まれ時間帯Aに更新される)。

引用文献

[1] Chen, P.P., The Entity Relationship Model-Toward a Unified View of Data, ACM TODS, Vol. 1, No. 1, 9-36 (1976)

[2] CODASYL Development Committee, An Information Algebra: Phase I Report, Comm. ACM, Vol. 5, No. 4, 190-204 (1962)

[3] CODASYL Systems Committee, Distributed Data Base Technology-An Interim Report of the CODASYL Systems Committee, NCC 1978, 909-921

[4] Codd, E.F., A Relational Model of Data for Large Shared Data Banks, Comm. ACM, Vol. 13, No. 6, 377-387 (1970)

[5] EDP ANALYZER, Improving the System Building Process, EDP ANALYZER, Vol. 12, No. 12 (1974)

[6] Hotaka, R. and Tsubaki, M., Self-Descriptive Relational Data Base, Proc. of VLDB Conf., Tokyo, 415-426 (1977)

[7] Mijares, I. and Peebles, R., A Methodology for the Design of Logical Data Base Structures, Modeling in Data Base Management System, North Holland (1976)

[8] Rothnie, J.B. and Goodman, N., A Survey of Research and Development in Distributed Data Base Management Proc. of VLDB Tokyo 48-62 (1977)

[9] Smith, J.M. and Smith D.C.P., Database Abstractions: Aggregation and Generalization, ACM TODS Vol. 2, No. 2, 105-133 (1977)

[10] Tsubaki M., DPLS-Database, Dynamic Program Control, Open-Ended POL Support, Proc. of Workshops on Databases for Interactive Design, Waterloo, Ontario (1975)

[11] Tsubaki, M., Multi-level Data Model in DPLS-Database, Dynamic Program Control and Open-Ended POL Support, Presented at VLDB Conf. Frammingham, Mass. (1975)

(注) 座席予約システムなどのように1つの応用にデータベースのみならず(複数個の)コンピュータ全体が使用されるような分散データベースでは障害の種類、伝播、回復手段が予想されるので、特殊なものとしてここでは対象としていない。