

トータルデータベースの構築 に関する一考察

佐藤 東九男

はじめに

「Database」（以下「DB」と略記する）は、ひとりデータ処理技術者のものであってはならない。それは、何よりもデータ利用者の真のニーズに応えるものでなければならない」といわれてから、すでに久しい。にもかかわらず、今日 DB の名のもとにさまざまな組織体において実用に供されているファイルは、依然としてその大半がデータ処理技術者のものである、というのが実態であるといつてよい。

もちろん、真に最終データ需要者のニーズに照準をおいて設計された DB が存在しないわけではない。だが、それらは、Entity という点からみる限り、特定の領域に Entity を限定して DB 化したものがほとんどであって、言葉の真の意味でトータル DB を実現したものとはいえない。

ここで、“データ処理技術者のもの”とは、データ処理技術者のニーズへの対応、すなわち、EDP 処理生産性の維持向上を第一のねらいとして設計された DB の謂であることはいうまでもない。

小論では、ユーザ志向型 DB の開発にとって、ポイントは何かを考察するとともに、“組織体における基軸エンティティ”なる鍵概念を提唱し、およそいかなる組織体であれそれがトータル DB の開発を志向するならば、まず組織の基軸エンティティの特性に着目して、これをベースに全ファイル体系を展開することが、開発を成功に導く鍵であることを論じた。

さらに、さまざまな組織体について、基軸エンティティの特性を基準とした分類を試み、分類枝のおおの位置づけた組織類型に対して、トータル DB 設計上のポイントを指摘することによって、基軸エンティティ特性の分析から得られる知見が、トータル DB の開発に際して有用なガイドラインとなりうることを示唆した。

1. データベースの現実

DB が、そもそも EDP 室の技術的必要に応じて出現したものであることは、論をまたないであろう。このことは、初期の IDS や DBTG 型 DB に典型的にみられたように、当初から DB システムがデータの Redundancy 排除とそのプログラムからの独立ということを、主要なねらいとしていたことから明らかである¹⁾。

しかし、技術的には高度にソフィスティケートされた DB の構築によって、いかにデータ処理の生産性が向上しようと、それがエンドユーザの広汎なデータ需要に対する十全な対応充足性を欠いているとするならば、その意義は半減することになる。

といっても、これは今日の DB Concept が、エンドユーザーを無視ないしは軽視している、ということでは決してない。

実際、DB システムの Architecture にいう外部ビュー (External view) の概念は、応用プログラマのみならず、エンドユーザーの利便をも配慮したものである²⁾といえるし、また、大多数の Architecture がいわゆるパラメータ利用者 (Parametric User) や Ad-Hoc な非専門家利用者 (Specifier User) を想定している³⁾のも、DBMS の設計者が一般のデータ利用者をも射程において DB Concept の設計を行っているからにほかならない。

しかし、それにもかかわらず、一般エンドユーザの広汎なデータ需要に応える「トータル DB」がなかなか実現をみないのは何故なのか。

幸か不幸か、Database も Databank もいずれも DB と abbreviate される。論者によっては、両者を全く同義とする⁴⁾ものもあるが、小論ではわが国での一般的慣用から受ける印象的差異を踏まえて、一応次のように概念規定をしておく。

Database…………データ提供者 (=EDP 室) の視点から合理化されたデータの集合。技術的合理性がシステム評価の基準となる。

Databank…………データ利用者 (=エンドユーザ) の視点から合理化されたデータの集合。参照可能性 (Referenciability) がシステム評価の基準となる。

したがって、雑多に集められたままのデータ集合であっても、それがエンドユーザのデータニーズを満たす限りは Databank であるが、Database ではない。それが Database であるためには、データ非冗長性、無矛盾性などいくつかの技術的条件を満たしていなければならない。

小論では、トータル DB なる語を用いるときは、叙上の意味での Database 的要件を備えた Databank というニュアンスを含意している。

さて、論旨展開の便宜上、いま「図 1」に示すような簡単なファイルを想定してみる。

CK1	A	CK2	B	C	CK3	D
I	a ₁	1	b ₁	c ₁	ア	d ₁
I	a ₁	2	b ₂	c ₂	イ	d ₂
I	a ₁	3	b ₃	c ₃	ウ	d ₃
II	a ₂	1	b ₁	c ₄	ア	d ₁
II	a ₂	4	b ₄	c ₅	エ	d ₄
II	a ₂	5	b ₅	c ₆	イ	d ₄
II	a ₂	2	b ₂	c ₇	イ	d ₂
III	a ₃	4	b ₄	c ₈	ア	d ₁
III	a ₃	6	b ₆	c ₉	ウ	d ₃

→

CK1	A	
CK2	B	
CK3	D	
CK1	CK2	C

(図 1)

このファイルでローマ数字、アラビア数字およびカナで示した item は候補キー (Candidate Key) であり、アルファベットで示した item は属性 (Attribute) である。いま、これを Boyce/Codd の BCNF 条件に従って正規化すると、4 種類のファイルができる。

この 4 種類のファイルは、いずれも第 2 および第 3 正規型の条件を満たしており、ファイルの DB 的特性という点からみる限り、何らの差異も存在しない。

第 2 正規型………Relation のどの候補キー K についても、「K に属さない、いかなる属性 A も、K の真部分集合に対して関数従属 (Functional Dependence) ではない」ようなファイル

第 3 正規型………第 2 正規型であって、かつ Relation のどの非主要属性集合 (nonkey Attribute) も、全て候補キーに対して推移従属 (Transitive Dependence—— $((A \rightarrow B) \cap (B \rightarrow C) \Rightarrow (C \rightarrow A))$ でないようなファイル⁵⁾

しかし、いま個々の属性にまで立入って見てみると、属性Aと属性Bは候補キー CK 1, CK 2 という単一のキーに対して関数従属しているが、属性Cは CK 1 と CK 2 の直積集合に関数従属している。ここで、単一の item からなるキーを単一キー、2 つ以上の item からなるキーを複合キーと呼ぶことにする。

ここでおそらく、ほとんどの DB Manager は単一キーに係属する属性集合も、複合キーに係属するそれも、いずれも Entity と呼んでとくに区別はしないであろう。すなわち、設例での 4 種類のファイルは全て、それぞれの Entity を表しているという意味で、全く同質であるとして取り扱うであろう。

しかし、意味論 (Semantics) 的視点からデータ属性を考えるエンドユーザの立場からするならば、単一の Key で識別される Entity と複合キーによって識別される Entity とでは、基本的に大きな相違があるといわなければならない。というのも、前者は原則として広義の「モノ」(物理的、観念的存在物) に関係し、後者は「コト」(事象) に関係するからである⁶⁾。

小論では以下、広義のモノを表わす Entity を実体 Entity, コトを表わす Entity を事象 Entity と仮称する。

組織体に関連する若干の具体例をあげるなら、ヒト (自然人のほか、自然人の集団たる各種の法人、権利のない社団などを含む)、モノ (製(商)品、原材料、有形無形固定資産のほか、無体的、非物理的存在物たる法令やプログラムなどを含む)、カネなどは実体 Entity であり、仕入、生産、販売、広報、住民サービスなどの活動 (Activity) は事象 Entity である。

では、これら 2 種類の Entity の特徴とこれらを区別して認識する実益は何か。

(1) 一般にひとつのコト (事象 Entity) には、複数のモノ (実体 Entity) が関与する。すなわち、モノ (複数の実体 Entity) が相互に作用し合い、触発し合うところに生起するのが事象である。

この意味で、事象 Entity よりも実体 Entity のほうが、より根元的、基本的である。

(2) 実体 Entity (の属性値) への関心は、概して組織横断的であり、普遍性が高い。

たとえば、製品という Entity の属性である製品別売上高は、単に製造部門、販売部門の最大の関心事であるばかりでなく、開発部門や長期経営戦略策定スタッフ、さらにトップマネジメントにとっても不可欠の情報である。

しかし、これが営業員別得意先別製品別売上高となれば、関心を持つのは

販売部門に限られるであろう。

(3) 事象 Entity には、通常特定のアプリケーションが関係する。この意味で、任意の事象 Entity への関心は局在的 (Local) である。

一体、組織体において個別部門機能の管理スタッフが第一義的な関心を持つのは、実体 Entity そのものではなくて、事象 Entity である。営業部門の管理スタッフが関心を持つのは「何がいくつ売れたか」より、「いつ、誰が、どこに、何をいくつ売ったか」であり、製造部門の何よりの関心事は、「何を、いくつ造ったか」ではなく、「いつ、誰が、何から、どんな方法で、何をいくつ造ったか」である。

つまり、アプリケーションにとっては、商品や製品などという実体 Entity 自体よりむしろ、営業員、得意先、商品あるいは原材料、生産設備、生産作業員、製品など複数の実体 Entity がからみ合った交点に実現される販売あるいは製造という事象事実こそが関心事なのである。そして、いうまでもないことながら、経営組織体の中で「販売」という事象の動向に決定的な利害を持つのは営業部門であり、「製造」という事象の動向を最大の関心事とするのは生産部門である。

(4) 複数の実体 Entity の主キー (Primary Key) の直積として得られる複合キーを主キーとする事象 Entity は、一般に一種類とは限らない。

これは、実体 Entity が事象に関与する場合、通常はそのある一部の側面 (Facet) だけで関与することの結果である。

すなわち、実体 Entity E_1, E_2, \dots, E_n の個有⁷⁾の属性数を a_1, a_2, \dots, a_n とすると、事象 Af に関与する属性 AfP は a_i のべき集合の直積の部分集合となる。

$$AfP \subset 2^{a_1} \otimes 2^{a_2} \otimes \dots \otimes 2^{a_n}$$

たとえば、販売管理、商品発送管理ともに社員 (営業部員と輸送部門の社員)、得意 (= 送達) 先、商品という 3 つの実体 Entity が関与する点で、いずれも主キーは「社員キー \otimes 得意先キー \otimes 商品キー」となるが実務実際上は、販売にあつては得意先の業種業態、規模、業績、信用状態、当該得意先の売れ筋商品、当該得意先と良好な関係を維持している営業担当者名等々の属性が問題となる半面、商品発送管理では得意先の所在地、商品の物理化学的特性、輸送員の業務キャリアといった属性が関心の対象となる、というようにさい然たる相違がある。

(5) 実体 Entity は、その本性上現に継続して機能しつつある存在であり、絶えざる変化変動の相の下にある。そして重要な点はヒトが (あるいは、少

なくとも当該の実体 Entity を管理する立場にある組織内部の個別機能管理部門が) こうした変動を制御することは、原理上困難ないし不可能であるということである。

実際、社員 (という Entity) の誰が、いつ病欠することになるか、どの得意先 (という Entity) がいつ倒産に至るかは、ほとんど自社のコントロールのらち外にあるし、また商品や原材料 (という Entity) が物理的、価値的に劣化することは、ある程度避けられない。

これは実体 Entity の場合、属性値更新の必要が不可避であり、かつそれが不可予見的に発生することを意味している。

他方、事象 Entity は原則として取引、契約、管理・経営行為その他組織のスタッフが行う何らかの意思決定 (とそれに基づく行為) の結果として生起する。

事象 Entity は、現にその効果を継続中のもの (たとえば、セールスマン別に取り扱い商品群と担当する得意先とが定められている場合の、そうした「決まり」があるという事実) と、すでにその効果が完結しており、この意味で過去の歴史的事実に属するもの (会計上の「取引」事実は、その典型である) とに、大きく類別することができるが、後者についてはそもそも更新の必要そのものが全くなり、前者についても必要となる更新データ (の発生要因) の性格はかなりの程度まで controllable である。

以上、正規化されたファイルの意味論的解釈を足がかりとして、経営組織に関係する Entity を実体 Entity と事象 Entity とに類別し、かなりの紙幅を費やしてその特性を考察したが、叙上の考察からして、次のような結論が帰結することはもはや多言を要しないであろう。

(1) アプリケーション単位の要求分析 (Requirement Analysis) から出発し、これにその基礎を依存する DB 格納データ属性設計の方法による限り、トータル DB (以下「TDB」と略記する) の設計は、不可能ではないまでも困難である。

(2) したがって、アプリケーション単位の要求分析に代わる、TDB 格納データ属性設計のための Design Concept が必要である。

(3) 新しい Concept は、したがって当然、Application free なアプローチに基礎をおくものでなければならない。

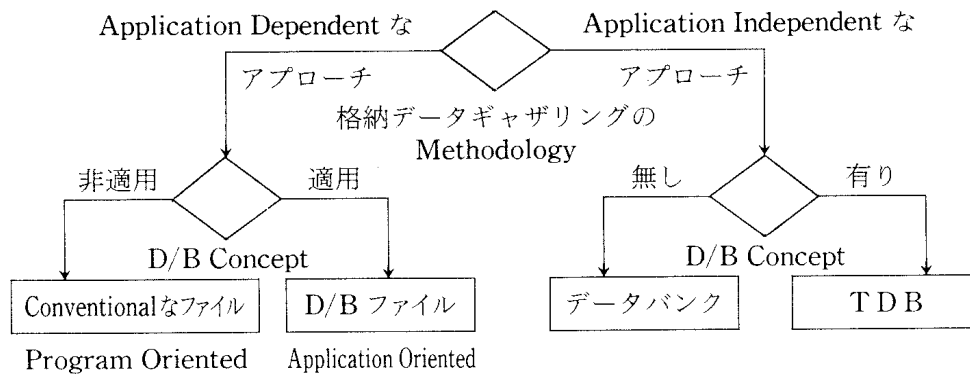


図 2

2. TDB の基本構成

では、この Concept は具体的にどう考えるべきであり、この Concept に基づいて設計された TDB はどうあるべきであろうか。

結論からいえば、理想的にはそれは、組織体におけるあらゆる Entity の動態を、いわば“そのまま、そっくり”ファイル上に写像したものであろう。しかし、これは、いうべくして現実には実現不可能であるといわなければならない。

とすれば、とりうる唯一の方途は、組織体にかかわる全ての Entity の動態を、何らかの形で Model 化することであろう。ANSI-SPARC MODEL にいう、いわゆる Conceptual Model の概念も、“終極的には”こうした意味での Model 化を想定しているはずである⁸⁾。

(1) 管理 Phase と情報類型

ところで、先に望ましい TDB は application free であるべきこと、いわば“データそれ自体の持つ論理”に従って構築されるべきであることを強調したが、これは特定の application というフィルタを通すことによって、バイアスのかかった application dependent なデータ集合になることを恐れるがゆえであって、application とは独立に設計を進めるべきだという趣旨では、決してない。

むしろ、可能な限り多くの application から参照の可能な Datapool の構築を志すがゆえに個別特定の application からは独立であることが要請される、ということである。

そこでいま、組織において発生し(または認識され)、参照される情報の特性と管理 phase との関連、すなわち管理の各 stage においてどういう情報が

参照され、また発生するかについて、若干の考察を試みてみる。

まず、考察の便宜上、経営情報 (Management Information)⁹⁾ を3つの類型に分け、さらにそのおのおのを2つのカテゴリーに分類する。

- ① 実体 Entity 情報……………先に概念規定をした実体 Entity それ自体の内的、構造的な諸特性を示す情報
 - ② 事象 Entity 情報……………先に概念規定をした事象 Entity の諸属性を示す情報
 - ③ 動態情報……………実体 Entity の何らかの“動き”を示す情報
- Ⓐ 事実(fact)情報……………Entity に関する、何らかの事実に係る情報 (Entity の Sein 的側面に係る情報)
 - Ⓑ 基準(Criteria)情報……………
 - …Entity に関する意 (企) 図された、望ましい何らかの状態に係る情報 (Entity の Sollen 的側面に係る情報)

こうすると、これら2つの分類軸を行要素列要素とする matrix ができるが、この各要素に該当する情報例を、実体 Entity として「製品 Entity」、事象 Entity として「売上取引き」にとって示すと、図3のようになる。

	実体 Entity 情報 (製品)	事象 Entity 情報 (売上)	動態情報 (営業実績)
事実情報	製品名, 開発年度, 部品構成, 実際耐用 年数, 実際製造原価	売上傳票 No, 売上 製品名, 売上得意先, 担当営業員, 売上価 格, 売上数量・金額	製品別, 月別, 営業 所別, 担当営業員別, 売上数量・金額
基準情報	設計耐用年数 製造標準原価		製品別, 月別, 営業 所別, 担当営業員別, 売上目標金額

(図 3)

そして、経営情報をこのように Categorize すると、管理の各 stage と、そこで参照され、または発生する情報類型¹⁰⁾ の関係は図4に示すように図式化することができる。

ところで、ここで注目されるのは、動態情報と事象 Entity 情報との関係である。

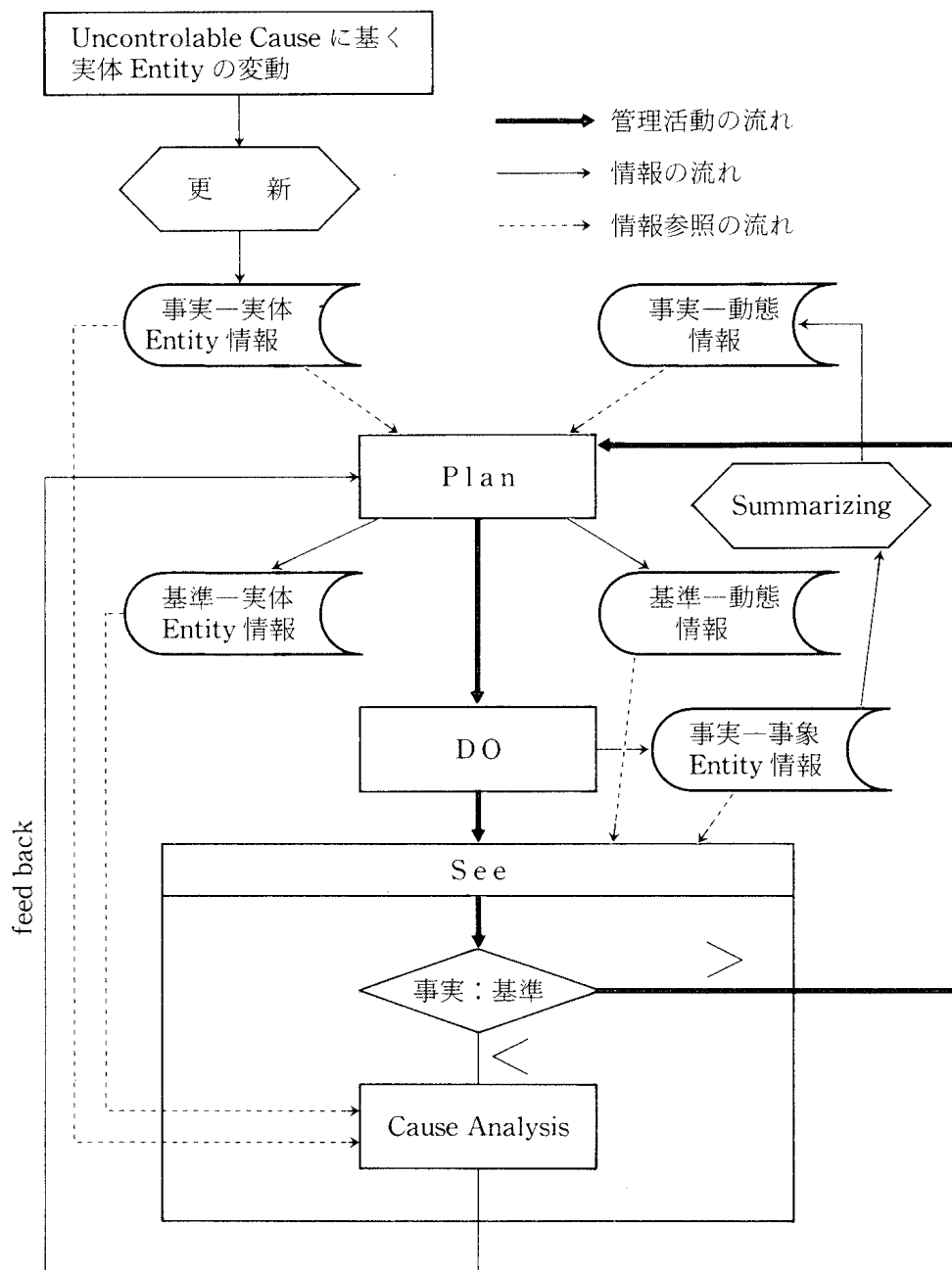


図 4

いま、 n 個の属性 (Property) を座標とする事象 Entity E のデータ空間を考え (これは、 n 次元のデータ空間としてもよいし、あるいはあらゆる事象 Entity の属性を次元数とする m 次元データ空間 ($m > n$) としてもよい)、 E の accountable なデータ値を持つ属性集合の値 v を、key の集合 (先にみたように、事象 Entity は一般に複数の Key を含む—複合 Key—) から任意にとった Key K_1, K_2, \dots, K_n の直積に関数従属する属性軸で構成される超平面上に「1 つずつ」射影 (projection) して、この和をとると、これは K_1, K_2, \dots, K_n の Key 別に識別してとらえられた動態情報 (事象 Entity 情報の Sum-

mary としての) となる。

また、Key 集合の中の任意の単一 Key K_i について同様のことを行くと、当然のことながら、Key K_i によって identify される実体 Entity の動態を示す情報¹¹⁾ が得られる。

換言すれば、事象 Entity 情報を TDB の格納対象情報にさえしておけば、必要な動態情報は、これから演算操作によって入手することができる。

このことは、実体 Entity と事象 Entity に関する情報が整備されてさえいれば、Management Cycle の各 phase で需要される情報ニーズの対応は、“原理上” 十分であることを意味するものである。

(2) 基軸 Entity

ここで、基軸 (core) Entity の概念を導入する。

基軸 Entity とは、次の条件を満たす Entity である。

- ① 営利組織にとっては収益の源泉であり、非営利組織にとっては、それなくしては組織存立の意味がなくなってしまうような、組織にとって最も重要な実体 Entity である (実質的規定)。
- ② ほとんどの事象 Entity の主 Key の中に、構成元のひとつとして入っているような属性を主 Key として持つ実体 Entity である (形式的規定)。
- ③ 場合によっては、組織の中で転態 (Metamorphosis)、統合 (Unification) を行うなど、非恒常性をその特性のひとつとするものもある (特性的規定)。
- ④ 先に、実体 Entity を Control することの一般に困難である旨を述べたが、基軸 Entity はこうした中では相対的に Controlable である。
- ⑤ 上記の特性、とりわけ②および③の特性から、基軸 Entity は、いわば組織体システムの中を貫流する、一種の“流れ” (flow) と見做すことができる。

このように、基軸 Entity の動態を、組織体システムの中を流れる flow としてとらえるとき、これを logistics flow と呼ぶことにする。

実体 Entity の中で、基軸 Entity でないものを“リソース (Resource——経営資源) Entity と名付ける。

典型的には、社員、仕入先、得意先、管理部門、事業用資産などが、これに該当する。

さて、このように実体 Entity を基軸 Entity とリソース Entity に分類するとき、われわれはさらに進んで次の事実を指摘することができる。

- ① 基軸 Entity の特性 (これは、その Entity を認識するのに必要な識別属性

項目のあり様に反映する) と動態は、組織体の種別や業種業態の相違を強く反映する。

- ② これに対して、リソース Entity は、組織体種別、業種業態の相違を超えて、その種類と識別属性項目がほとんど共通である。

たとえば、自動車メーカーと商社では、基軸 Entity の特性と動態に、比較を絶するほどの相違があるが、社員、得意先、仕入先、管理部門といった Entity が経営上の必須のリソースとして関係することは、両者とも全く共通である。

- ③ したがって、組織体特性の相違や業種業態上の特質は、結局のところ基軸 Entity の特質と動態の上に集約的に反映されるということになる。

——TDB の基本構成 Concept——

以上の考察を踏まえると TDB の基本的な Concept を次のように構想することが可能となる。

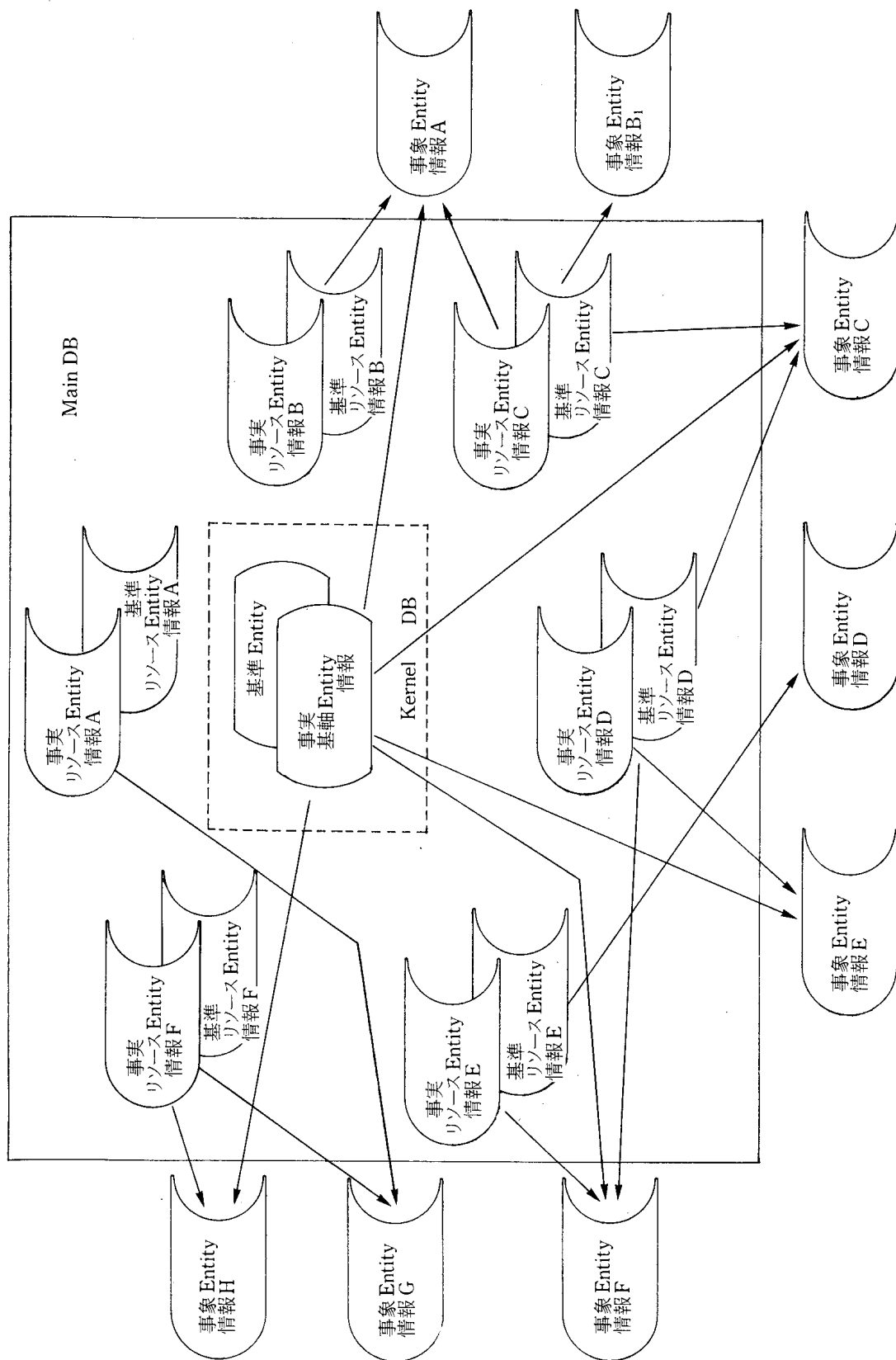
- ① 基軸 Entity のデータ属性値を格納対象とする SubDB を、全 TDB 体系の中核に位置づけ、これを中核 (Kernel) DB と名付ける。
- ② 中核 DB のまわりに、リソース Entity のデータ属性値を格納対象とする SubDB を配し、中核 DB と併せてこれを主 (Main) DB と名付ける。
- ③ さらに、主 DB のまわりに事象 Entity のデータ属性値を格納対象とする SubDB を配置し、①、②および③を併せたものをもって、TDB とする。これを図示すれば図 5 のようになろう。

このように構成された TDB は、次のような特徴とメリットを持つことが期待される。

- ① TDB を構成する各ファイル (それが関係 DB モデルならば “各 Relation”) が DB 理論にいう、いわゆる実在世界 (The World of Reality) のありようと、明確に (いわば、1 対 1 に) 対応する形になる。

このことは、こうして構成される TDB が、Application free であることを意味する。

- ② したがって、原理上 Occurrence Maintenance が容易になる。
- ③ 既述のとおり、組織体の特性は基軸 Entity のあり方に集約的に反映されるので、TDB の設計開発努力の大半を、基軸 Entity 属性の設計に傾注すればよいことになる。
- ④ TDB の Total Design とは別に、実際の implementation は、
- i 業種業態の相違を越えて共通性が高く (すなわち、業種業態ごとの



(図 5) TDB 概念図

特殊性を反映することが少なく)

ii 相対的に Maintenance の頻度が少なく,

iii それゆえに, 開発に先立って検討すべき問題点の少ない

リソース Entity の DB 化から, まず着手することも可能である。

- ⑤ 事象 Entity ファイルについては, 事象の生起後一定期間を経過したら, 既述した必要な演算を行って, 関係する実体 Entity の動態情報として吸い上げ (Summarizing 処理), 吸い上げの終了した事象 Entity tuple は, 磁気テープ等におとす, という処理を行うことによって, TDB を常に Current なものに保ちつつ, DASD の所要 Memory Capacity をミニマムにおさえることができる。

では, このように構築された TDB は有用たりうるであろうか。2, 3 の Application について, モデル実験的に Availability study を行ってみる。

[人事アプリケーション]

一般に, 人事アプリケーションが需要するデータは, そのほとんどが「社員」というリソース Entity に係わるデータであり, 詳細にみると, その内容は大きく 2 つの類型に分類することができる。

ひとつは氏名, 住所, 学歴, 職歴, 特技などのいわゆる属人性のデータ(いい換えれば, Entity に個有の属性データ)である。

このタイプのデータは, 更新の必要が比較的少なく, また組織特性や業種業態の相違を越えて, その管理運営に必要かつ不可欠のものである。この種タイプのデータへの需要には「社員」というリソース Entity SubDB によって対応できる。

いまひとつは, 「いつ, 誰が, 何処で, 何のために, どんな仕事を, どれほど行ったか」を示すデータで, 組織においてヒトが何らかの活動を行った結果として, 発生するデータである。

このタイプのデータは, それゆえ組織体を貫流する Logistics flow と密接に関連し, 当該の組織体にあつて具体的にどんな種類のデータ属性が必要されるかは, その組織体の Logistics flow に強く規定される。が, いずれにせよ, この種タイプのデータ需要に対しては, “事象 Entity 属性データまたはその Summary データ” によって, 対応することができる。

[営業アプリケーション]

営業部門が需要するデータは, 基本的には「売上げ」に係わるデータに尽きるといってよい。そして, 売上データは, 原理上

・誰, またはどの営業所, 営業部門が,

- ・いつ
- ・何を
- ・誰または、どの得意先に
- ・いくつ販売した

という構成をとる。つまり、売上データは、ヒト（販売員および得意先）という実体 Entity とモノ（売上商（製）品）という実体 Entity のいわば直積としてとらえられるデータ属性である。

それゆえ、このタイプのデータは、ここで構想した TDB においては、基軸 Entity の主 Key の直積と主 Key とする一象 Entity の属性データとして、十全にとらえることができる。

3. 基軸 Entity の特性による組織体の類型化

さて、如上から明らかなごとく、組織体における TDB の有り様を基本的に規制するのは、当該組織の基軸 Entity の特性である。それゆえ、当該の組織体にもっとも適合的な TDB を構築するには、先ず自らの組織の Entity 特性を確認することが重要となる。

こうした趣旨から、あらゆる組織体について、その持つ基軸 Entity の特性を基準に類型化を試みると、差し当り次の 7 タイプに類別される。

- ① 基軸 Entity 転態型
- ② 基軸 Entity 構造化型
- ③ 金融型
- ④ 商業型
- ⑤ 資産運用型
- ⑥ 公共サービス提供型
- ⑦ 行政体型

次に、この中のいくつかについて、若干の考察を行ってみる。

(1) 基軸 Entity 転態型

[該当する組織特性]

それが組織体システムに input されてから、output に至るまでの要所要所で形質の変化を伴うような Entity を基軸 Entity として持つ組織類型を、“基軸 Entity 転態型”と呼ぶことにする。

一般に、システムへの入力段階で高いエントロピーを持つ実体 Entity から、Logistics flow の過程でエントロピーを奪い、相対的に低エントロピーの実体 Entity を造り出して、出力するようなタイプの製造業がこの類型に入

る。

具体的には通常、加工型、資源変換型などと呼ばれる、次のような業種類
型がこれに該当することになる。

化学工業

薬品工業

食品加工業

製鉄、非鉄金属工業

土石、ガラス、セメント製造業

電力、ガス製造業

鉱業

農業、林業、牧畜業

[基軸 Entity の特性]

① 転態

鉄鋼業における鉄鉱石、コークス、石灰石→鋼材、高炉ガス、高炉セメント、電力業における石油、石炭、水力、核燃料→電力といった具体例をあげるまでもなく、このタイプの最大の特徴は、Logistics flow を構成する基軸 Entity が、“流れ”の過程で転態 (Metamorphosis) を起こすこと、すなわち形質的に全く別異の物質になってしまう、という点である。

② 基軸 Entity の非多様性

後述する基軸 Entity 構造化型などと比較して、このタイプの場合、基軸 Entity を構成する実体 Entity の種類はきわめて限られている。

実際、たとえばセメント製造に例をとると、入力石灰石、粘土、鉄滓、石こうなど数種、出力は銘柄の相違を別とすればセメント一種だけと、入出力ともに併せてもわずかに数種を数えるにすぎない。

[TDB 開発上のポイント]

このタイプの経営組織体における TDB 設計上の最大のポイントは、転態する Logistics flow の動態を、どのように DB 上に表現するかという問題である。

おそらく、原理上もっとも理想的な形は、システムの中を転態しつつ流れる Material flow の動態を“そのまま、ソックリ”DB ファイル上に写像するというものであろうが、これは原理的にいっても不可能である。

原理上 feasible なモデル化の類型としては、差し当たり次の二つが考えられよう。

ひとつは、システムに流入する連続量としての Logistics flow の入力を可

能な限り微小な単位（これを、Unit Entity と仮称しておく）に分割し、この Unit Entity が“転態”して新たな属性を獲得するごとに、属性項目とその値とを付加していく、という方法である。

この方法では、いわば Bose-Einstein 統計に従がうような特性を持つことになる、きわめて多種類の Unit Entity が、基軸 Entity の基本単位を構成することになる。

いまひとつは、基軸 Entity の流れが、“転態”によって新たな属性とその値とを獲得するごとに、Occurrence を分割していく方法である。この際、Unit Entity は、ある任意の値を持つある属性項目構成を持った Occurrence は、ひとつしか存在しないような形で構成される（この点、第一の方法によるときは、属性項目構成とそのおのこの項目の値が全く相等しい複数の Occurrence が存在しうることになる）

(2) 基軸 Entity 構造化型

[該当する組織特性]

組織システムに input された独立、多種類の Entity（部品材）が、形質的な同一性を保持しつつ、一定の規則に従って順次構造に組み上げられていき、最後に複雑高度な構造体となって、システムから出て行くような Entity を基軸 Entity とする組織類型を“基軸 Entity 構造化型”と呼ぶことにする。

通常、組立型と呼ばれる次のような企業類型が、これに対応する。

造船

各種機械製造

重電

エレクトロニクス

自動車製造

車輛製造

航空機製造

建設

住宅建設

[基軸 Entity の特性]

この類型にあっては、基軸 Entity は次のような特徴を持つ。

①構造化

当初、独立の単体としてシステムに投入された多種多様な Entity は、Substance flow の過程で漸次統合化されていき、最終的にはぼう大な単体 Entity を構成要素とする複雑な構造体 Entity となって組織システムから出

て行く。

②入力 Entity の不変性

システムに投入された個々の Entity は、終始自己同一性 (Identity) を保持したまま、システムから流出していく。

③入力 Entity の多様性

初めに、システムに投入された Entity は、種類、量ともにきわめてぼう大である。

[TDB 開発上のポイント]

① このタイプの場合、Substance flow がシステムの中を流れて行くにつれて、漸次組みあがっていく構造体 (Structure) を、いかに DB 上に表現するかが、最大のポイントとなる。

一般に、この構造関係は木構造 (Tree Structure) により、特殊複雑なもので網構造 (Network Structure) によって表現することができる。

② したがって、採用する DB のタイプが何であれ(実際には、Relational Type が多いと思われる)、implementation 上の問題は“原理上”ないといっ
てよい。

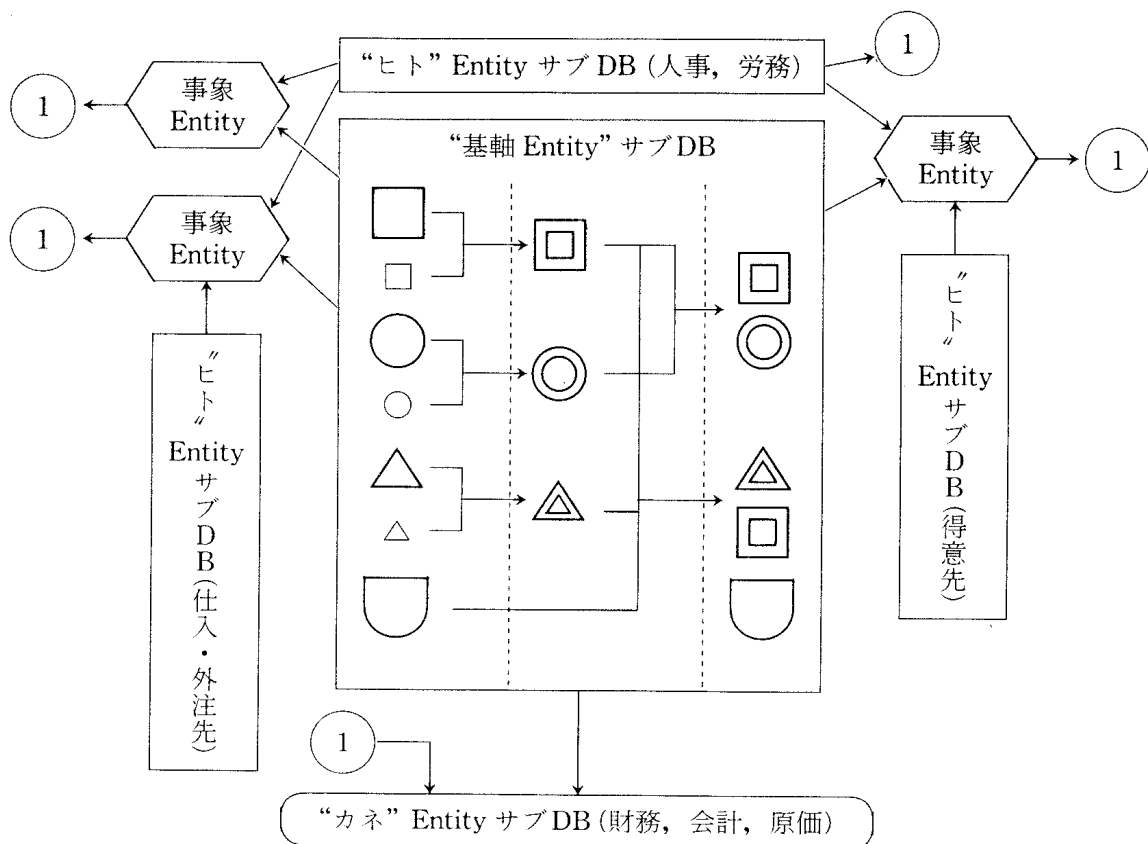


図 6

③ 原素材→部品→半製品→製品と流れる flow の各ステップで発生し、認識されるデータ——消費電力量、工業用水使用量、直接労務費、配賦される各種間接費等——は、各構造化のレベルに正確に対応する属性値として、何らのアイマイさも残すことなく、とらえることができる。

結局のところ、このタイプの場合、問題はデータ構造をいかに Cost/Benefit よく、DB 上に表現するかということに集約される。

(3) 資産運用型

[該当する組織特性]

車輛、船舶、構築物など特定の固定資産の利用提供サービスを収益の源泉としているような経営組織類型を、“資産運用型”と呼ぶことにする。

倉庫

陸運、海運、航空輸送

映画、演劇、スポーツ興業

といった企業種類が、該当する。

[基軸 Entity の特性]

この類型にあっては、基軸 Entity は次のような特徴を持つ。

① 基軸 Entity 管理の二次性

この企業組織類型にあっては、基軸 Entity が、企業システムを貫流する実体 Entity であることには変りない。

すなわち、陸海運送業、航空業にあっては乗客や貨物、興業等にあっては観客、倉庫業では預かり商品等が基軸 Entity を構成することになる。ただ、資産運用型の場合、基軸 Entity がシステム内に滞留する期間はきわめて短い。たとえば、鉄道やバスの乗客、映画の観客などは、ほんの数分～数時間の間システム内にとどまるだけである。

加えて、基軸 Entity が“ヒト”である場合には、下車駅で自分から降りる、閉幕すれば自ずから退場するなど、完全に自己管理を行う。

こうした理由から、このタイプの組織類型の場合、基軸 Entity の動態を DB 上に写像して、管理の対象とすることは、事実上きわめて困難であるし、また実務實際上その必要もない。

② Vessel 管理の重要性

この類型の組織体にとっては、基軸 Entity を収容する、いわば“容れ物”(Vessel) ともいえるべき運用固定資産こそが最重要な経営資源である。

この意味で、運用資産の物的、法的、経理的な側面からする管理こそが、第一の課題となる。

[TDB 開発上のポイント]

叙上のように、この組織類型の場合、基軸 Entity を DB 表現上においても Entity (関係型モデルならば Relation) として扱うことは、原理上はともかくも、実際上は非常にむずかしく、かつ実益も少ない。

こうしたことを考えれば、この類型にあっては次のような Scheme が実際的であると考えられる。

- ① 本来は Vessel たる運用固定資産を基軸 Entity とする。
- ② 本来の基軸 Entity たる実体 Entity の固有属性データは TDB 格納対象データとはしない。
- ③ 本来の基軸 Entity たる実体 Entity の動態データは Vessel Entity の属性データとして取扱う。
- (4) 公共サービス提供型

[該当する組織特性]

個人に対する公共的なサービスの提供を目的として、施設資産の運営を行っている組織体を“公共サービス提供型”と呼ぶことにする。

学校、病院、養護施設などがこの類型に分類される典型的な組織体である。

このタイプの組織体は、建物、施設などの設備資産を運用し、それが事業の収益源泉のひとつになっているという点で、外形上資産運用型に類似している。

しかし、基軸 Entity の特性に着目する限り、この両士は別異の組織体類型として取扱われる必要がある。

[基軸 Entity の特性]

① 個別管理の必要性

資産運用型の組織類型にあっては、基軸 Entity の occurrence 個々に対する個別的な管理は実際上困難であったし、またその必要もなかった。

これに対して、公共サービス提供型にあっては、基軸 Entity を構成する occurrence 個々の個性的特性の相違を踏まえた個別管理こそが、重要なポイントとなる。

このことは、学校の場合は学生生徒それぞれの個性に応じた教育が、病院ならば患者一人一人の病態に即した治療が必要不可欠であることを考えれば、あまりにも明瞭であろう。

② 滞留期間の長期性

学校の場合、一旦入学した学徒児童は 2 ～数年の間システム中にとどまり、病院ならばかなりの割合の患者が、初診後数週間～数ヵ月にわたって入

院ないしは通院をするであろう。養老院に至っては、システム内滞留期間が20年以上に及ぶこともまれではない。

③ 基軸 Entity の恒常性

いうまでもなく、公共サービス提供型における基軸 Entity は“ヒト”である。

したがって、この類型にあつては、当然のことながら転態や構造化といった基軸 Entity そのものの変動を考慮に入れる必要は全くない。

[TDB 開発上のポイント]

このタイプの場合には叙上した基軸 Entity の特性からいって、当然基軸 Entity を Kernel DB とする TDB モデルの標準型に近い形の DB を、大きな問題なしに構築することができる。

注

- 1) S. M. Deen 「Fundamentals of Data Base Systems」 p 7～ James Martin 「Principles of Data-Base Management」 p 35～
- 2) Date 「An Introduction to Database Systems」 p 15, p 179～p 190
- 3) James A. Senn 「Information System in Management」 p 386
- 4) James A. Senn ibid p 338
- 5) S. M. Deen 「Fundamentals of Data Base Systems」 p 138～p 145, Date 「An Introduction to Database Systems」 p 160, p 162
- 6) もっとも、これは作業仮設上の tentative な特徴づけであつて、若干の例外はある。
- 7) 複数の実体 Entity が交錯するところに発生する属性ではない属性
- 8) S. M. Deen 「Fundamentals of Data Base Systems」 p 207～
- 9) ここでは、単に組織体の経営管理に必要な情報という意味である。
- 10) 厳密に言えば、reference される情報と発生する（あるいは認識化される）データとは語用上区別する必要があるが、ここではいずれも“情報”という語で表現した。
- 11) Conventional なファイルにあつては、こうして得られた情報は、実体 Entity の属性値のひとつとしてとらえられた。

参考文献

国勢社編集部編著「商品の科学」国勢社