文章表現を用いたデータベースの論理設計法SDDM

穂鷹 良介（筑波大学）

1 はじめに

データ[ベース]モデル論，あるいはこれと密接な関係を持つデータベース設計論に関して多くの論文が発表されているが，次の2点に関して不満がある。

(1) データモデルあるいはデータベースモデルは具体的なデータベース設計手段と無関係に述べられている。そのため，モデルは現実世界を描写するだけに止まっており，具体的にどのように設計手段と結びつけるのか，またある設計過程から考えてどのモデルを採用するのが得策であるのかといった，「実用的見地」から見た反省に乏しい。

(2) 関係データベースの関数従属性に端を発した，形式的かつ数学的な設計理論は，理論的な設計法を提供したものとなっているが，実用的な適用可能性から考えると大いに問題がある。[1)]

本稿は，上記2点を反省の原動力として，データベース設計手段とデータモデルまたはデータベースモデルとを一体化した体系を考え，現実問題に適用可能な設計方法SDDM（Sentential Database Design Method）を提案する。

この方法の中では，新しい概念の提案がなされ，従来明確には捉えられていなかった諸概念，たとえば導出データ，データフローなどが，データベース設計あるいはその運用管理という側面でどのような関連をたがいに持つかが明らかにされる。

SDDMでの設計手順は大略図1に示す通りである。まずデータベースシステムに対して要求される情報要求が文章の形でなされる（3節）。個々の情報要求は，それに対して単文の形で回答を用意することができるかどうかが設計者によって判断される。可能な場合には単文によって回答が記述され（4節），さもなければより単文によって回答を記述することが容易となるような別の情報要求に変換もしくは分解される。この過程を帰着（5節）と呼ぶ。

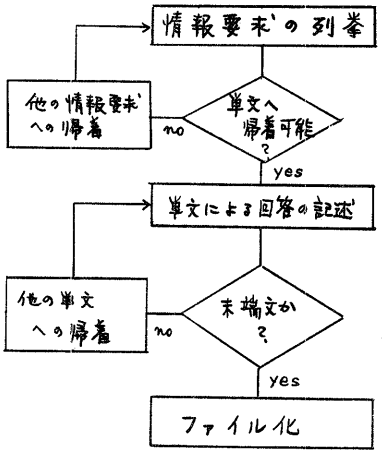

図1 設計手順の流れ

単文を用意するとき念頭におく現実世界のモデル化については2節で述べる。単文を用意したとき，それが直ちにデータベースに入力可能な情報あるいはデータとして入手可能かどうかを判断し，可能な場合にはそれを末端文としてそれ以上の帰着は行なわない。末端文でない場合にはそれを更により基本的な単文に帰着する。他の単文に帰着された単文は，他から計算されることとなるから導出データ（6節）と呼ばれる。

このようにしてすべての情報要求が末端文になった所で，導出データを含めてすべての単文を叙述主体型（7節）に着目してまとめあげファイル化（9節）する。末端文から導出データ，導出データから導出データの導出関係はデータフロー（8節）

となる。

ＳＤＤＭは１９８１年１月現在ではまだ論理設計レベルの設計手段でしかない。本当の意味でのデータベース設計手段となるためには，これに適切な物理設計手段を付加しなくてはならず，しかも作業全体としてはその方がはるかに大きな量となろう。

2　データモデル

ＳＤＤＭは２種類のデータモデルを用いる。一つは概念レベルで現実世界の認識，叙述をいわば設計者の頭の中で行なうためのもの，もう一つはよく知られている関係データベースのBoyce－Codd正規形に従うモデルである。情報要求の列挙から始まり，概念レベルのモデルで現実世界を記述し，論理設計の出力として正規形の関係表（ファイル）を得る。

概念モデルは３種類の基本概念（主体，主体型，名前）と，２種類の操作（主体作成，主体型定義）を有する。

主体（entity）は哲学用語でいうところの特殊者に相当する。設計者が認識する現実世界の対象のことで実体ともいわれる。主体型（entity type）は哲学用語の普遍者に相当するもので，主体の集合をまとまった一つのものとして考えるとき設計者によって定義される。主体はあくまでも認識の対象として識別されるだけで組織的あるいは機械的な情報処理手段の対象たり得ない。そのためにはそれを名前で置き換える必要がある。同様の必要性が主体型についてもいえる。以後の議論はこの主体もしくは主体型をそれを指し示す名前でつつがなく置き換えられたものとして行なう。時として名前による置き換えに問題があるケースもあるが，そのときには別途，名前づけの問題を考える。

主体型の定義は，設計上の必要から設計者が適宜与えるもので，専ら設計者の認識能力，分析能力に頼らざるを得ない。適当な主体の集まりを一つの主体型と把握することである。自然に現実世界を観察することによって認識される主体の他に，既存のいくつかの主体から別の主体を作ることも行なわれる。Smith[2)3)]等が述べた統合化（aggregation），汎化（generalization）はこの例である。他に一つの主体型に属する主体を別の主体型の主体とみなす別型定義，一つの主体型に属する主体をいくつかまとめて別の主体とみなす集合化（grouping）を考える。これらの概念は複雑な対象の叙述に有効であるが，紙数の制約のため，これ以上の説明は省く。より単純なケースに限定してもＳＤＤＭの基本的な発想を伝えるのには支障はない。詳しくは穂鷹[4)]を参照されたい。

3　情報要求

データベース設計の出発点としては色々の種類のものがある。従来の方法は，設計者が利用者の要求を勘案しつつ，現実世界をデータベースモデルの記述能力を用いて叙述するものとか[5)]，あるいは必要とされる出力帳表をはじめに確定してそれを実現するデータベースを用意するものとか[6)]，利用者がデータベースを相手に行なう業務面からデータベースへの要求を拾い出すものとか[7)]，情報要求を文章の形で提出してもらい，それを実現するためのデータ構造を用意して行くもの[8)]などがある。

ＳＤＤＭではBubenko[2)]が行なったように，情報要求は最初文章の形で列挙されるものと考える。熟達した設計者が現実世界を眺めて直ちにデータベースの構造

を定めることができるのは，彼が心の中で利用者になり代って情報要求を自分に対して発し，しかもそれを陽に表ゆすことなくそれへの回答を用意しているからに過ぎなく，とり出す気になれば情報要求は文章の形で書けるのだと考える。

ただこの設計者の自問自答式の手続きは大変効率が良いので，細い部分あまいは利用者がことさら問題にしないような部分でも当然発せられなければならない情報要求については，設計者が適宜補うこととする。

出力帖表をはじめに用意する場合も，情報要求としてそのような出力結果を利用者が要求したと考えれば依然としてＳＤＤＭの情報要求の一種と考えられる。業務分析から要求をとり出す場合も同様である。

情報要求には次のように一連番号を付す。

R1：当社で所有しているバスのおのおのに対して，最近10回の給油記録から算出される燃費を知りたい

R7：別に用意する出力帖表＃５のような出力を得たい

R12：商品の現在の在庫量を知りたい

別々に番号の振られた情報要求ごとにそれを別々の記録として，図２のように整理する。ＳＤＤＭを手操作で行なう場合にはＲ（Requirement）レコードは適当な大きさのカードで実現すればよい。将来，ＳＤＤＭをコンピュータの支援の下に実現することが考えられるが，そのときＲレコードは通常のファイル（データベースの）のレコードになる。

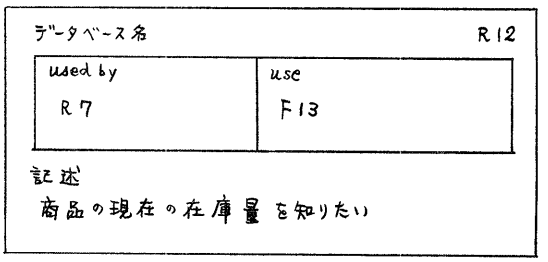

図２　Ｒレコード

４　単文

データベース上で叙述する事柄はすべて単文の形をとるものとする。単文の例としては

(a1) 山崎君は英語ＩＡのクラスで成績Ｂを得た

(a2) 田中君は英語ＩＡのクラスで成績Ａを得た

(a3) 小池君は英語ＩＢのクラスで成績Ｃを得た

(b1) 英語ＩＡのクラスは現代文の科目に属し，２学期，Ｅ07番教室で佐藤先生が担当する

(b2) 英語ＩＢのクラスは現代文の科目に属し，３学期，Ｓ110番教室で斉藤先生が担当する

があげられる。

(a1),(a2),(a3)の３個の単文は，学生，クラス，成績という３種の主体型に属する主体間の関係を文章で表現したもので共通点がある。主体型と主体との関係を明示しながらこれをまとめると

(a) 学生Ｐはクラス ＣＬ で成績Ｇを得た

同様に(b1)(b2)をまとめると

(b) クラスＣＬは科目ＣＯに属し，学期Ｓ，教室Ｒで教師Ｅが担当する

となる。(a),(b)のような叙述はデータベースに蓄えられる数多くの単文のパターンを代表するもので単文型ということにする。

設計の際にはデータベースに蓄える個々の単文を考えるのではなく，それらをいくつかの種類にわけ，その種類を代表する単文型を考える。

任意の文が単文となるのではない。単文は次の制約を満すものでなければならない。

【単文の条件】単文においては1個かつただ1個限りの主体型の主体の叙述がなされていること。この1個の主体型をその単文（あるいはその単文が属する単文型）の叙述主体型という。

単文の条件は，「叙述がなされている」というややあいまいな表現によって特徴づけられているが，その精神は関係データベースのBoyce-Codd 正規条件とほぼ類似の条件を述べている。関係データベースではこれを関数従属性などの全く形式的な表現によって定義しているのだが，かえってそのことにより，実務家が設計を行なうときの直観的な意味づけが不明となるので上のような言い方をした。

関係データベースの言葉でいうならば叙述主体型は，正規形をした関係のキー属性に対応する。

・商品Xは在庫量Yで住所Zに住む生産者Uから提供される

という文型は単文型ではない。この文型においては商品という主体型の叙述のみならず，一部生産者という主体型の叙述（住所がZであるという）も含まれているからである。

データベース名	F13
used by	use
R12	F131, F132, F141 F152

記述

商品Xは在庫量Yである

図3　Fレコード

前ページの例(b)ではクラスが(a)では{学生，クラス}が叙述主体型である。

情報要求の場合と同様に単文型のおのおのにF（File）で始まる識別番号を与え，図3のようにFレコードに記録する。

5　帰着

人間はある問題を解決したいと欲するとき，最初に心の中で「問題Xを解け」と自分自身に命令する。そのゴールに到達するためにいくつかの小問題，たとえばX_1, X_2, X_3が解ければ十分であるというものが心の中に浮かんだとき，今度は問題「X_1を解け」「問題X_2を解け」，「問題X_3を解け」と改めて心の中で自分自身に命令する（図4）。以下同様に小問題X_1, X_2, X_3を解くためのより小さな問題X_{11}, X_{12}, X_{21}, …が得られて，最後にこのようにして展開された全問題が心の中で解決されれば，もとの問題が解決したことになる。ときには，X_1とは別の問題X_2の解決のためにX_1を解決するときに現れた小問題X_{12}が使われることもある（図4）。

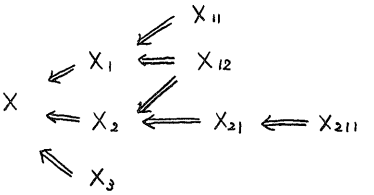

図4　問題解決時のプロセス

このように問題解決のために十分条件をさがして行くことに似た過程を帰着（reduction）と呼ぶことにする。SDDMではたまたまこの問題が情報要求であり，小問題に対応するものが別の情報要求であったり，単文であったりしているが，帰着そのものの考え方は全く同一である。

⇐印で帰着を表現したとき，たとえば図5のような帰着関係があったとする。

このとき，Rレコード，Fレコードはこれらの関係を表現するために図2，図3のように記述される。

すなわち，情報要求R12は上位の情報要求R7によって使われるので R12のRレコードのused by欄にはF7と書き込まれる。R12自身は単文型F13から答を得ることができるから，use欄にF13と書き込む（図2）。同様にF13のFレコードも図3に示したように書き込まれる。

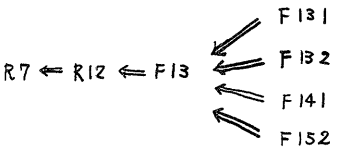

図5　情報要求，単文型間の帰着関係

大規模システム開発のときにはこの帰着関係の数がぼう大となるが，たとえばuse欄に書き込まれているのに対応するRレコードあるいはFレコードがまだ登録されていないときには，未完の設計作業として直ちにそれを見出すことができる。use，used by欄の記入は双方向のポインタのようなもので人手による保守は面倒であるが，将来SDDMをデータベースの機能を利用して半自動化したときには，容易にコンピュータからの支援を受けることができる。

6　導出データ

たとえば

F13：商品Xは在庫量Yである

という単文型を

F131：商品Xは期首在庫量Yである

F132：商品Xを日時Tに数量Y仕入れた

F141：商品Xを客Yに対し日時Tに数量Z売上げた

F152：商品Xは客Yから日時Tに数量Zだけ返品された

という4個の単文型に帰着したとする。このとき，F13に対応するデータはこれをF131，F132，F141，F152の4個の単文型に属する単文から導くことができる。この意味で単文型に帰着される単文型は導出データといわれる。

単文型を帰着するときには必ず導出データが現れるから，単文型以降の帰着は導出データを設計の過程にいかに介在させるかという問題と裏腹である。

上記の例でもそうだが導出データを他の導出データあるいは単文型から導くためには，プログラムの実行，あるいは人手による作業などの何等かのプロセスが必要である（図6）。導出データXを考えるということはその背後にXを作り出すプロセスXを併せ考えるということである。導出データは，したがって，プロセスの抽象化という性格を持っていることになる。

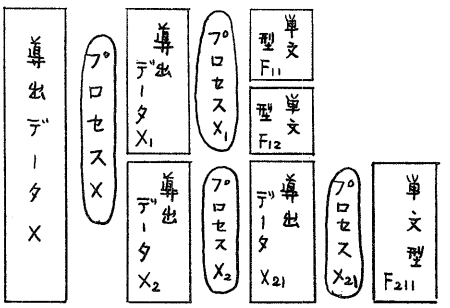

図6 プロセスの抽象化としての導出データ

ソフトウェア工学ではプロセスの抽象化という発想からStructured Programmingを考えるが，SDDMではプロセスではなく，それと一体となった導出データを考えることによってプロセスの抽象化を取扱う。すくなくともデータベースに対する情報要求をこなすという目的のためには，この考え方が便える。

プロセスの抽象化を導出データによって行なうという考え方には更に二つの利点が存在する。一つは，データが利用者の要求するものであって，それをいかにして作り出すかということは本質的な問題ではないということである。つまり，ある一つのデータを利用者が要求したときにそれを作り出すロジックが，物理的効率のことを勘案したり，スピードのことを勘案して変ったとしても利用者として見ればどうでも良い。このとき，設計時に残すべき情報はデータに関するものを中心とすべきである。導出データの考えはこれに合っている。もう一つ，技術上の問題であるが，データの記述とプロセスの記述との両者を比較した場合，データの記述の方が易しいことがあげられる。データはどちらかというと静的なのに対しプロセスは動的で記述しにくい。つまりプロセスを共有財産として使おうと試みるとき，プロセスそのものを対象とせずに，付随する導出データを共有しようとする方が，関係者の共通理解を得やすい。国民所得算出ロジックよりも国民所得データの方が利用しやすい。

7　叙述主体型の明確化，事象

単文型を考え出し，登録するときに次のことに気を配らなくてはならない。単文型は後に見るように，最終的にはファイルとなって，その定義に従ってデータがデータベース内に蓄積される。したがって，単文型を考えるときにまぎれ込んだ誤りは未来永劫データベースの中に含まれる。

単文型には叙述主体が存在する。もし誤って同一主体型なのにも拘らず別の主体型と認識して単文型を形成すると結果は，同一主体のデータを冗長に収集蓄積する破目となる。データ入力費用がかさむと同時に，複数種のデータから得られる複数の情報が同一主体に関して得られることになる。これらが同一の内容であることは必ずしも保証されない。別の誤りとして本来は異る主体型を同一の主体型と認識してしまった場合には一つの叙述が同時に異る２種類の主体型の主体の叙述と解釈されてしまう（図７）。

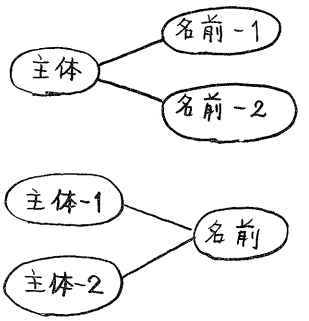

図７　主体の認識に伴う２種の誤り

データベースにおける主体型の認識ということは，それに名前を与えることである。なぜならばそれ以後の処理はすべて名前を基礎になされるからである（図７）。同様の問題は単文型の叙述に用いられる主体型の認識の際にも存在する。データベース開発の際の最も大きな問題であるといって良い。

関係データベースの理論では，叙述主体型は常に存在する。最悪の場合には関係に現れるすべての属性の結合がキーとなるからである。

しかし，実際問題としては単文型の作り方が悪い場合には叙述主体型が存在しない場合がある。たとえばすでに前ページで述べた

F132：商品Xを日時tに数量y仕入れた

は適当と思われる叙述主体型がない。同一商品を同一日時に同一数量だけ仕入れることがありうるからである。現実の事務処理でこのようなケースをどのように取扱っているかを調べてみると，殆ど例外なく，F132 に対応するようなデータに関しては，仕入井というような人為的な属性を一つ追加している。つまり，叙

述主体型は仕入というものを表わす仕入井である。取引井，伝票井，事故処理井などは皆これと同じ性質のものである。

ここで認識された叙述主体型は，普通，事象（event）といわれるものである。事象はある時点に発生するという性質を持っている。この例が示すように正しい設計のためには，叙述主体型を正しく認識する必要があり，そのためには正しい主体型を単文型に取入れる必要がある。叙述対象の意味を深く考えない単なる形式的な設計法はいたずらに実務家をまどわし，余計な時間をとらせるという意味で有害，実用的な利用ができないという意味で無益である。

8　末端文，データフロー

情報要求から出発しての帰着はどの段階まで続くであろうか。帰着を行なっている設計者にとってそれは自然に分るものである。彼はある種の単文型に至るとこれ以上帰着する必要がないと感じる。彼が帰着の必要を認めないのは，その単文型に属す単文（データ）が直接得られるということが分かるからである。通常，これは次の3種類であろう。

(1)　事象を表わす単文型
(2)　現実を観察することによって得られるデータ
(3)　外部から提供されるデータ

これらは，あまり単文のような形をしていないことが多いが，末端文と呼ぶことにする。つまり，帰着は末端文に至って終了する。

今度は逆に末端文から出発して帰着の矢印を逆にたどって次々と導出データに至るルートを考えてみると，そこにデータの流れを見ることができる。これがデータフローであり，SDDMではデータベース設計という作業の中で情報要求，単文型，帰着，導出データ，プロセス，データフローの間に互いに密接な関係を見出したことになる。

データ辞書でデータベースまわりの環境を記述し，管理，開発に役立てようという動きが盛んであるが，データ辞書の設計には上記の関係の把握が最も重要となろう。

9　ファイル化

単文型が上記のように作られ，同時に叙述主体型が1対1に対応する名前によって明確に確定したならば，これをまとめて関係データベースという所の正規化された関係（ファイル）にするのは比較的簡単である。

まず単文型から'てにおは'を取去って

下5（$\underline{E_1}$, E_2, E_3）

のような形にする。たとえば下131ならば

下131（$\underline{商品}$, 期首在庫量）

とする。下線は，叙述主体型を示す。

今仮に

Fi（$\underline{E_1}$, E_2, E_3）

Fj（$\underline{E_1}$, E_4, E_5, E_6）

のような単文型が存在した場合には，これを係合して

Fk（$\underline{E_1}$, E_2, E_3, E_4, E_5, E_6）

のようにする。

F_k はFレコードの識別番号であるが，ファイルの名前は叙述主体型の名前をそのまま利用して混乱はない。F_i の中の主体型 E_2 と F_j の中の主体型 E_3 とが同じ場合には，当然冗長であるから併合のときに一方を取去れば良い。

しばしば同一主体型が2個単文型の中に現れる場合がある。たとえば

F9：部品 x は部品 y を数量 z 使用して組立てられる

というような場合である。この場合には単文型に現れる主体型の名前は主体型を指し示す名前で，両者の役目は異ると考え別々の名前（属性）を与えるというようにすこし単文型の述べ方を変更しておく必要がある。[4)]

最後に併合のとき実際によく現われる例外ケースを紹介しておく。

F10：バス x はガレージ y に所属する

F11：バス x は故障中である

これを先程の記法で簡潔に書くと

F10（バス，ガレージ）

F11（バス）

ということになる。これを併合して

F21（バス，ガレージ）

としたのではF11の意味がファイル化に伴って失われてしまう。これを解決するにはたとえば人為的に待機状態という主体型を設け，

F21（バス，ガレージ，待機状態）

等とする必要がある。

参考文献

1) 中村央朗，P.P.Chen：リレーショナル・データベースにおける多値従属の推移規則の問題点に関する考察，第22回ＤＢＭＳ研究会，55年11月13日，情報処理学会

2) J.M.Smith and D.C.P.Smith：Database Abstractions: Aggregation, CACM 20,6, pp.405-413, 1977

3) J.M.Smith and D.C.P.Smith：Database Abstractions: Aggregation and Generalization, TODS 2,2, pp.105-133, 1977

4) 穂鷹良介：データベースの論理設計，情報処理叢書，情報処理学会，1981出版予定

5) 椿正明：ＳＤＳＰ（System Design/Development Standard Procedure）について，ＤＢＭＳ研究会，52年7月14日，情報処理学会

6) ＩＢＭ：Business Systems Planning（ＢＳＰ）Information Systems Planning Guide, IBM, 1975

7) M.Managaki and K.Kawagoe：A Database Design System with Conceptual Model Description Language, COMPSAC, pp.141-146, 1979