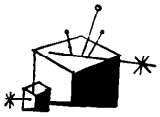
講　座



データベースの論理設計（1）†

穂鷹　良介††

1. はじめに

データベースの設計は大きく分けて論理設計と物理設計になる．両者の境界は必ずしも明快に設定はされていないが，[Tsubaki 82] と同じ立場をここでは採用する．

「すなわち**データベースの論理設計**とは，DBMS に独立な概念スキーマの設計と，DBMS に従属する論理スキーマ（これを論理と呼ぶことに抵抗を覚えるDBMS が多いが，ここでは目をつぶる）の設計を含むものとする．もちろん，この論理スキーマが概念スキーマと一致すれば1つになる．」

以上の概念スキーマと論理スキーマとを区別せず**スキーマ**と呼ぶことにする．論理設計の目標は，与えられた情報システムに対する要求から，データに対する要求（これを**情報要求**と呼ぶことにする）を選び出し，それに答えることのできるスキーマを決定することにある．具体的には，スキーマをなんらかの表現方法によって記述した**スキーマ記述**（通常データ記述文，データ定義文などといわれる）を出力することにある．

スキーマ記述でどのレベルの概念を表現するのかはデータベース設計者に与えられる環境によって個々別別である．前提とするデータベース管理システムが高水準のときには，設計者は便利で使いやすい高水準の概念を利用することができる．使用可能な概念が低水準のときには，設計者が介入して必要なレベルに帰着させなくてはならない．論理設計として，どのレベルまで帰着させるべきか，はっきりとした境界はないが，一応関係データベースの第1正規形レベルを目標に置く．

スキーマは，同じ意味のデータに対しても一意には定まらない．スキーマを導き出す方法もそこで使用される用語も提唱者によってさまざまである．本稿では用語の混乱を避けるために，全編を通じて1つの用語に1つの意味を与えるよう努めた．このため，同じ用語でも異なる意味で使用されていると思われるときで，かつそのことを明確にする必要があるときには，たとえば entity [Chen 76] のように適当な修飾語を付加する．用語は無定義語と定義語に分けられる．無定義語については，もはやその意味を他のデータベースに関する用語で定義しようと試みないものである．本文では厳密な定義なしに若干の説明とともに現れる．

他の用語から説明できる用語はできるだけこれを定義語として，厳密な定義を与えるよう努めた．

本稿は2回にわたる講座として連載される．今回は，論理設計の基礎としてのデータモデル論について述べる．

2. 対象世界とデータモデル

データベースの設計とは**対象世界**を一定の方式で表現し，その表現方式に従って将来データを蓄積，整理し，後の検索に役立てようとするものである．結果として得られた表現のことを**データモデル実現値**という．対象世界とは，よく現実世界，論議領界 (Universe of Discourse) [ISO TC 97/SC 5/WG 3 81] とデータモデル理論でいわれているものである（対象が必ずしも現実的なものでなくても良いので本稿では現実という形容詞を使用しなかった）．

対象世界の構成要素である**対象** (object) は，そのままでは機械的な情報処理対象となり得るものではない．情報処理対象として正確に操作されうるデータモデル実現値の構成要素である**主体** (entity) に一旦変換されなくてはならない（**図-1**）．対象と主体の対応はデータベース設計者の認識によりなされる．ここでいう認識は [Nijssen 760] でいうところの perception と selction とを合せたものに等しい．認識がどのようにしてなされるかについては，これ以上立ち入らない．もしかすると1個の対象を2個の主体として見る

† Logical Database Design by Ryosuke HOTAKA (Institute of Socio-Economic Planning, University of Tsukuba).

†† 筑波大学社会工学系

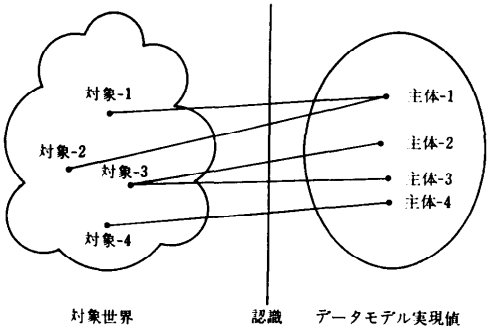


図-1　対象世界とデータモデル実現値

かもしれないし，逆に認識の解像度が悪くて実際には異なる2個の対象を1個の主体として見るかもしれない．どのように認識されようと，これ以降の議論は認識された主体，それを含むデータモデル実現値を前提になされる．

さてデータベース設計者が1つの方式に従って，複数個の対象世界を認識すると，それぞれデータモデル実現値が得られるが，そこで対象世界の表現に用いた概念，枠組は一定している．個々のデータモデル実現値はこの一段高い方式もしくは枠組――これをわれわれは**データモデル型**と呼ぼう――に属していると考える（**図-2**）．しばしばデータモデル型とデータモデル実現値との区別を厳密にしない議論がなされており，これらのいずれかの意味で単にデータモデルといわれることが多いが，本稿では両者を区別できるように異なった呼称を定義しておく．

［定義 2.1］ **データモデル実現値**のことをスキーマという．スキーマを一定の規則に従って具体的に書下したものを**スキーマ記述**という．

スキーマの例としては，たとえば生産管理データベース，人事管理データベース，カリキュラム データベースなどといったものがある．1個のスキーマ，たとえばカリキュラム データベースに対して，経済学部用，理工学部用などといった運用場所，形態等を別にする**データベース実現値**が複数存在し得る．この意味で

［定義 2.2］ **スキーマ**のことを**データベース型**という（[Tsubaki 80]）．

1つのデータベース実現値（[Tsubaki 80]）に蓄えられるデータは，対象世界の変化を反映して（完全には同期しないであろうが）更新される．その結果，無数の異なったデータベース実現値の状態が観察されるであろうが，これらの1つ1つを**データベース状態**ということにする（図-2）．具体的にはデータベース型は複数の**ファイル型**から成る．データベース状態は複数の**ファイル実現値**から成る．

データベースという言葉は通常データベース型，

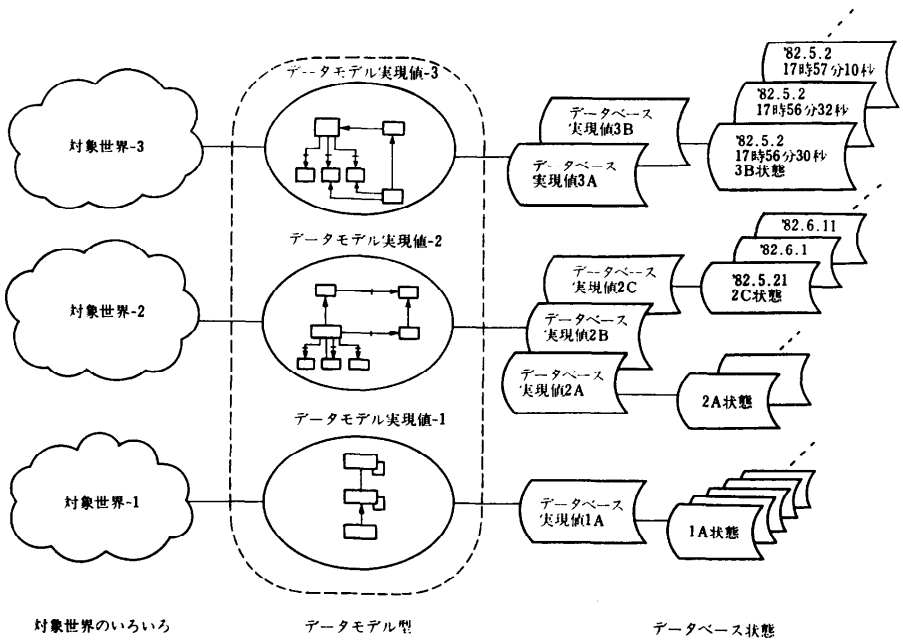


図-2　データモデル型，データモデル実現値，データベース実現値，データベース状態

データベース実現値，データベース状態のいずれかの意味で使われているので，文脈から判断する必要がある．本稿ではこれらを区別する．

データベース実現値として，中に蓄積されるデータをどのようなものと考えるかは，一切データベース設計者の自由とする．必ずしも機械化されていない文書，資料の類を考えてもかまわない．

以上の前提では，データベースの論理設計としてスキーマを決定するためには，データモデル型があらかじめ分かっていなくてはならない．データモデル型を提案するのはデータベース理論家の仕事であるが，極端に言えば100人データベース理論家がいれば100個の異なるデータモデル型が存在するというのが現状である．また1つのデータモデル型を固定したとしても，同じ情報要求の達成に，今度は設計者の数だけのスキーマが設計される．これはデータベース設計者にとっては厄介なことである．3章ではデータモデル型間，スキーマ間の差を説明し，データモデル論とどのように付き合ったらよいか指針を与えたい．

3. データモデル型，スキーマ

3.1 厳密なモデル型の必要性

われわれは日常生活の中で対象世界の記述という行為を頻繁に行っている．記述に用いている方法論の中で最も広く利用されているものの1つは，自然言語によるものであろう．

Jack は東京に住んでいる　　自然文による記述

主体	属性
	現住所
Jack	東京

(a)　主体と属性による記述

Jack は東京に住むプログラマである　　自然文による記述

主体	属性	
	現住所	職業
Jack	東京	プログラマ

(b)　主体と属性による記述

Jack は東京に住むプログラマである　　自然文による記述

'住む'	主体	主体
	Jack	東京

主体と主体間の関連による記述

'職を有する'	主体	主体
	Jack	プログラマ

(c)

注）ここで用いている記述はいずれも説明の便宜上のものであり，厳密なものではない．

図-3　各種のデータモデル型による記述

データモデル型で使用する'主体'，'属性'などという概念は，自然言語による記述を簡略化したものと解釈できよう．

たとえば「Jack は東京に住んでいる」という記述の代りに，Jack を'主体'と考え，その主体の持ついくつかの属性のうち，'現住所'という属性の属性値が'東京'であると考える（図-3(a)）．同様に「Jack は東京に住むプログラマである」という記述は図-3(b)のように表現されるかもしれない．

別のデータモデル型を用いる設計者は，同じ事実を主体と主体の関連として図-3(c)のように表現する．

データモデル型にはいろいろの問題があるが，その1つはここに示したような，記述の多様性の問題である．いずれも意味的には等しいことなのであるが，採用するデータモデル型が異なると一見，別の考え方をしているかのごとく見える．初心者にとっては，このデータモデル型のみかけ上の差違が本質的な差違に見え勝ちであるが，実際はそんなことはない．データモデル型にはいくつかの基本概念があって，それが手を変え品を変えて現れているだけである．

3章ではデータモデル型で使用する種々の概念を統一的な見方を採用して説明し，一方での意味の表現が他方でどのようになるのかを示したい．そのためには概念の定義をできるだけ厳密に行って，自然言語で記述しているときにあり勝ちなあいまいさが混入しないように努める必要がある．以下，記号を用いた形式的な説明方法を採用するのはこの理由による．自然言語の記述能力だけに頼っていては説明が分かりにくくなるような例をあげよう．

在庫管理システムでいうところの'部品'と品質検査システムでいうところの'部品'とでは性質がかなり異なる [Kent 78]．前者でたとえば100点の部品を扱うといったときの1点1点の部品は部品の種類を指している．後者で100点の部品と言ったとき，その1点1点の部品は，実際に検査の対象となったあるいはこれからなる予定の特定の部品を指している．いわば前者の部品の1つは，後者でいう部品の集合に相当する．

「ある著者が1冊の本を著した」というときの本と，「ある著者は昨年100万部の本を売った」というときの本の意味は異なっている．前者が本の種類を指すのに対し，後者は個別の物理的な本を指している．

ある国の就業者というのは職業を有している Charlie, Jack, Mary などの個人の総称である．20代就業

者，男性就業者はこの就業者の一部の個人（個人のグループまたは集合）を指す言い方である．就業者グループという言葉は，このような就業者のグループの総称とする．{男性就業者，女性就業者}というグループは性別就業者グループと名づけられる就業者グループの一部である．10才階級別就業者グループは {0～10，11～20，21～30，31～40，41～50，51～60，61～70，71～80，81以上の各就業者}を表わし，これも就業者グループの一部である．性別就業者グループも，10才階級別就業者グループもともに就業者の分類を示しているから，これらの総称は就業者分類といわれるものに相当する．

最後の例では主体（Jack とか Mary とかの就業者）をまとめて1つの主体（20代就業者とか男性就業者とかのグループ）とし，その主体をさらにまとめて1つの主体（就業者分類）とする操作を行っている．日常使用する自然言語では，このような操作を行うことはあまり頻繁ではないため，操作の途中に現れる種類の異なる概念を明確に区別する言葉を持っていない．単に就業者グループと言ったときに読者が想像するものは一意ではないだろう．上で述べた'部品'もそのような例の1つで，普通はその言葉が使われる環境に応じてその意味が暗黙のうちに仮定される．

これに対してデータベース設計では上述のような操作を頻繁に施す．スキーマはそれが使われる環境とは独立に記述される必要がある．それらは機械的情報処理にも耐え得るように厳密になされなくてはならない．

3.2 基礎概念

1つの対象世界が与えられたとして，それをスキーマで表現する際の問題を考える．

主体全体の集合を ***ES*** (Entity Set) で表わす．主体はデータモデル型の基本単位で，それをさらに分割して考えることをしない原子的なものである．他の著者は object [Falkenberg 76] ともいう．

最終的に人間またはコンピュータが形式的に主体を扱うためには，これを**名前**によって表現しておくことが必要となる．**名前**全体の集合を ***NS*** (Name Set) で表わす．

[仮定 3.1] $NS \subset ES$ と仮定する [Senko 76].

ある集合 ***SS*** (Set Set) を考える．$X \in SS$ のとき，写像 $\boldsymbol{S}$ によって $\boldsymbol{S}(\boldsymbol{X})(\subset ES)$ なる集合が対応するものとする．言い換えるならば集合 $S(X)$ を SS の元 X が代表している．$e \in S(X)$ のとき $\boldsymbol{e \in X}$ と書くことにし，e は X に**属する**という．後に頻繁に出てくるが，データモデル型の議論では集合論のように集合の要素 e と集合 S との関係を論ずるよりは，e と S を代表する何物かとの関係を論ずることが多い．このためにこの記法を新規に導入する．SS の部分集合として特に**主体型**全体の集合 ***TS***$(\subset SS)$ (Entity Type Set) 考える．$T \in TS$ のとき，T は**主体型** (entity type) と呼ばれる．$e \in T$ なる主体 e を T の**実現値**という．T に対しては ES の部分集合 $S(T)$ が対応するが，これは設計者がなんらかの意味で主体の集合を考えることに意義を見出し，それに対応するあるいはそれを代表するものとして T を考えることを意味する．

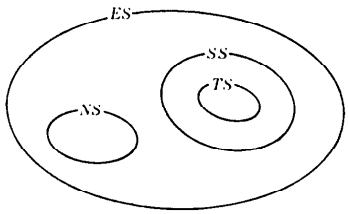


図-4 *ES, SS, TS, NS*

SS と TS との差は本質ではない．SS の要素で，特にデータベース管理者がデータベース型内に登録するものの全体が TS である．一時的に使用する主体型のようなものは TS 以外の SS の要素である．

[仮定 3.2] $SS \subset ES$

この結果，$TS \subset SS$ であるから，主体型自身主体でもある（**図-4**）．主体型は別の著者では **class** [Hammer 81], **object type** [Falkenberg 76], **entity set** Senko 76] などと呼ばれているものに相当する（**3.3** 節では他の概念も含めていろいろな著者の用語法を説明する）．

上記の仮定は SS の内容にかなりの制約を課すことになる．SS の元は ES の部分集合に対応するが，ES の部分集合全体の数（ES の幅集合の濃度）は ES 自身の要素の数（ES の濃度）に比べて圧倒的に多い．ということは，SS の元としては ES の部分集合のうちでごく限られたものだけに対応関係を持つということである*.

さて主体の名前付けの問題に立ち戻ろう．以下では $X(\in SS)$ が指定されたとき，$S(X)$ の元である主体は一意な名前を持つとする．すなわち

* たとえば $\{x \mid x \in SS \wedge \neg(x \in x)\} = S(w)$, $w \in SS$ となるラッセルのパラドックスで使われたような集合と対応関係を持つ w は存在しない．

[仮定 3.3] $X \in SS$ のとき $S(X)$ は NS の部分集合と1対1に対応する．形式的に言い換えるならば1対1写像 **name**$_X$ が存在して

$e \in S(X)$ のとき $\text{name}_X(e) \in NS$

ついでに

[定義 3.4] name$_X$ の逆写像 **ent**$_X$ を次のように定義する．

$s \in \text{name}_X(S(X))$ のとき $\text{ent}_X(s) = \text{name}_X^{-1}(s)$

以上の関係を**図-5** に示す．図を見やすくするために，SS, NS をわざと ES の外に描いている．

以上のようにしておくことによって，とにかく SS の元 X を指示しさえすれば，主体の代りに名前を使用することができるメカニズムが用意された．以後，データベース状態に蓄積されるデータはすべてこの主体の代りの名前の形でなされる．逆に $e \ll X (\in SS)$ となる X を指定しないで主体 e を処理の対象とすることはできない．

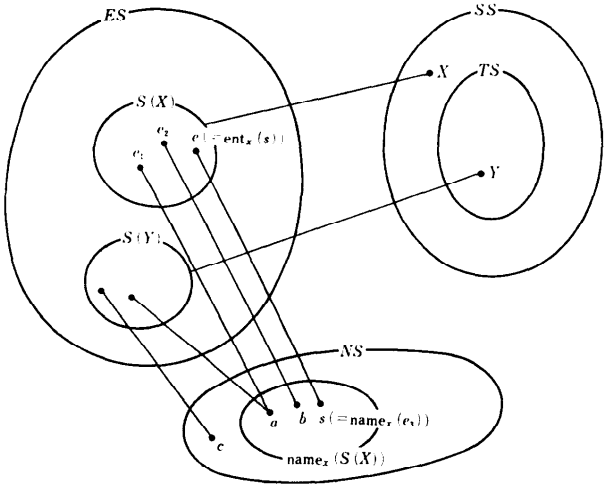


図-5 名前付け機構

注意すべきは，主体と名前の1対1の対応は無条件には得られないということである．なんらかの手段で SS の元 X を明示しておく必要がある．たとえばフォードと単に言ったとしてもそれは X が自動車会社のときと X が歴代米国大統領のときとで異なる主体を指す．

$e (\ll X)$ に名前をつけることは可能になったが，X にも名前をつけることができなくてはならない．次のように仮定する．

[仮定 3.5] TS の要素 **ENTITY-TYPE** が存在して

$TS = S(\text{ENTITY-TYPE})$

このことから，[仮定 3.3] によりすべての主体型は一意に名前づけされることになる．また $T \in TS$ と $T \ll \text{ENTIY-TYPE}$ とは同義になる．さきほど SS と TS との差は，後者の要素をデータベース管理者が登録するところにあると述べたが，$\text{ENTITY-TYPE} \in TS \subset SS$ であるから登録時主体型の名前の一意性が保たれるように管理されなくてはならない．

[仮定 3.6] どのような SS の要素 X に対しても適当な SS の要素 $X_1, X_2, \cdots, X_n$ $(n \geqq 1)$ が存在して

$X \ll X_1, X_1 \ll X_2, \cdots, X_{n-1} \ll X_n, X_n \in TS$

が成立つ．

この仮定により，X の要素を一意に識別できる名前づけの方法がすくなくも1通りは存在することになる．

以下では議論を簡単にするため，1つの主体型に属する1つの主体に複数個の名前があるケース（シノニム）を考えない．議論が複雑になる割には得られる成果が多くないと考えられるからである．

[例 3.7] **図-6**において 犬≪動物，猫≪動物 である．このときの動物は分類学的な意味の主体型を意味する．動物に属する犬とか猫とかは平均寿命とか平均体重とかの集団の持つ性質で特徴づけられるであろう．一方，ポチ≪犬，白≪犬であるが同時に ポチ≪愛玩動物，白≪愛玩動物 である．主体型愛玩動物に属するポチとか白とかは，集団としてではなく個々の犬としての性質，たとえば体

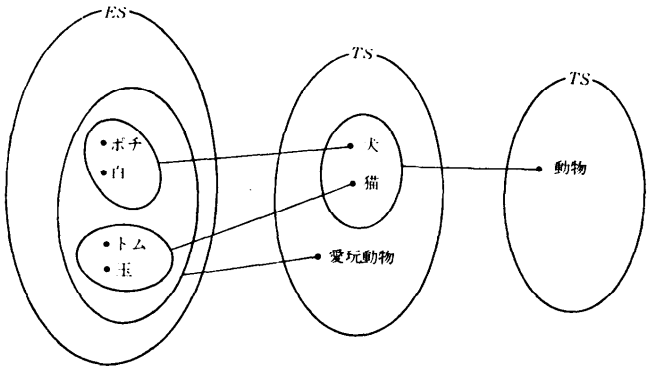


図-6 主体としての主体型

重，背丈，飼主などによって特徴づけられる．

動物も愛玩動物もともに主体型には違いないが，叙述のレベルが上述のように異なっている．レベルの異なる主体もしくは主体型同士は，記号≤で結びつけることによって互いの関係を明示することができる．

データモデル型の基礎概念として最後に必要なのは主体間の関係づけに関するものである．$F \in TS$ で次のような表現がFに関して定義されるときFを**関連型** (Relationship Type) あるいは**文型** (Sentence Type) という．研究分野ではこのFの表現を **frame** と呼んでいる（例えば [Yeh 78]）．ここでは [Yeh 78] での記法を少し変更して示す．

$F[(x_1 \leqslant D_1), (x_n \leqslant D_2), \cdots, (x_n \leqslant D_n):$
predicate$(a_1 : x_1, a_2 : x_2, \cdots, a_n : x_n)]$

ここで $n \geqq 1$, $x_i (i=1, 2, \cdots, n)$ は**変数**，$D_i (i=1, 2, \cdots, n_n)$ は主体型，predicate は変数 $x_1, x_2, \cdots, x_n$ を持つ**述語**，$a_i (i=1, 2, \cdots, n)$ は**属性** (Attribute) を表わす．$i \neq j$ のとき $a_i \neq a_j$ とする．

後の便宜のために

[定義 3.8] F の**属性集合** $A(F)$ を以下のように定義する．

$A(F) = \{a_1, a_2, \cdots, a_n\}$

属性 a_i を指定したとき D_i は一意に定まるものとし，この関係は次の関数 **dom** によって表現される．

[定義 3.9] $D_i = \mathrm{dom}(a_i)$ を満足する $D_i \in TS$ を a_i の**定義域** (Domain) という．

frame による関連型の表現は厳密であるがやや冗長なので，必要に応じて以下のように簡略化した書き方をする．

- 属性 a_i の名前と定義域 $D_i (= \mathrm{dom}(a_i))$ との名前が一致しているときに，D_i を省略する．
- 述語を人間に分りやすい文の形で書く．
- 定義域を明示するときには，属性 a_i が文の中に現れるときに $D_i : a_i$ のように書いて属性との対応を示す．

[例 3.10]

F10：社員xは番地：本籍yで，番地：現住所zである．

F20：学生xは科目yを履修し成績zを得た．

$F \geqslant f$ なる f を**関連** (Relationship) もしくは**文** (Sentence) と呼ぶ．

[仮定 3.11] $F \geqslant f$ なる f は $A(F) = \{a_1, a_2, \cdots, a_n\}$ から $\bigcup_{i=1}^{n} S(D_i)$ の中への関数である．

$v_i = f(a_i)$ のとき v_i を f の a_i に関する**属性値**という．

しばしば，f を表わすのに属性 a_i の順番が分かっているとき $(v_1, v_2, \cdots, v_n)$ と略記する．

[例 3.12]

[例 3.10] の F10 に属する関連の例

f_1＝(山田，小樽，小樽)

f_2＝(山田，小樽，東京)

f_3＝(田中，札幌，大阪)

関連は，データベースでレコード実現値，tuple(組)などといわれているものに等しい．関連を属性集合から定義域の和集合への関数として考える方法は，例えば [Armstrong 74] でも行われた．

[定義 3.13] T を1つの主体型とする．$S(P) \subset S(T)$ となる $P (\in SS)$ を T の **population** という [Nijssen 77]．

F が1つの関連型であるとき，F の population はそれがデータベース管理者の定める一定の制約条件を満すときファイルであるという．言い換えると

[定義 3.14] F を1つの関連型とする．主体型 G は G に属するすべての主体 $F' (\in SS)$ が

（1） $S(F') \subset S(F)$ つまり F' は F の population である．

（2） F' はデータベース管理者の定める制約条件を満す．

を満すとき，Fの**ファイル型**であるといい，Gに属する主体 F' を F の**ファイル**もしくは**ファイル実現値**という（**図-7** 参照）．

[注意 3.15] やや枝葉末節的であるが，F を1つの関連型とするとき，F 自身 $S(F) \subset S(F)$ であるが F は関連型 F のファイルになるとは限らない．

例えば [例 3.12] において，社員の現住所が1つしかない，別の言い方をするならば現住所という属性が社員という属性に従属でなければならないという制約をデータベース管理者が，Fのファイルに対して設けていたとする．このとき，$S(F) = \{f_1, f_2, f_3\}$ である関連型 F は自分自身のファイルではない．

[定義 3.16] 関連型 F の任意のファイル F' に対して，F' の中の関連を一意に識別する極小の属性の集合を**キー**という．

関連型 F はまた主体型でもあるから，F に属する関連（主体）を一意に識別する名前があるはずであるが，これに関して以下を仮定する．

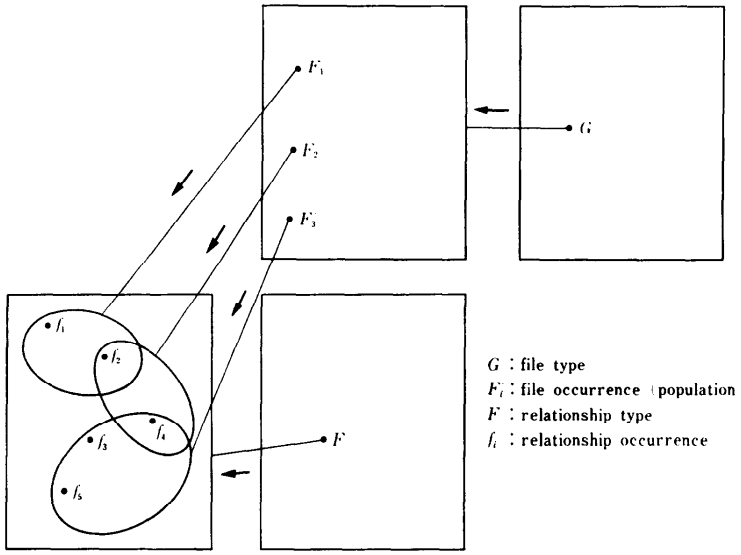


図-7 ファイルとファイル型

[仮定 3.17] F の実現値の名前は F のあるキー $\{a_1, \cdots, a_n\}$ により決定される．正確にはすべての $f(\ll F)$ に対して $\text{name}_F(f)=(f(a_1), \cdots, f(a_n))$ である．[定義 3.18] 上記の F の名前を決定するキーを**主キー**という．

[定義 3.19] ある関連型 F の主キーを $\{a_1, \cdots, a_n\}$ とするとき，各 a_i を**主属性** (prime attribute) といい，F の属性で $a_1, \cdots, a_n$ 以外のものを**非主属性** (nonprime attribute) という．

[定義 3.20] 関連型 F の**キー主体型** (key entity type) とは

(1) Fの主キーが単一の属性 a から成る場合，$\text{dom}(a)$ のことをいう．

(2) Fの主キーが単一の属性から成り立たぬとき，データベース設計者が仮想的に主キーと等価な単一の属性を a とするとき，$\text{dom}(a)$ のことをいう．

またキー主体型に属する主体を**キー主体**という．

キー主体型は常に主体型であるが逆は必ずしも真ではない．主体型であってどの関連型のキー主体型ともならないものもある．

[例 3.21] [例 3.10] の

F10：社員 x は番地：本籍 y で，番地：現住所 z である

において主キー＝{社員}，キー主体型＝dom(社員)で，他に関連型がなければ '番地' はキー主体型ではない．

F20：学生 x は科目 y を履修し成績 z を得た

において主キーは {学生，科目} であるとする．このとき仮想的に 履修＝{学生，科目} と考えて F20 のキー主体型を dom(履修) とする．

3.3 対象志向 vs. レコード志向モデル

データモデル型は [Biller 76], [Kerschberg 76] などでも紹介されているように多くのものが存在する．最近 ISO TC 97/SC 5/WG 3 は数年にわたる活動の結果，概念モデルに関する報告書 [ISO 82] をまとめた．これは参加したメンバの面々，費した年月から考えて現状では最も深く考えられたデータモデル型に関するまとめではないかと思われる．

[ISO 82] では考察したデータモデル型の一部として次のものを挙げている．

- Abstract data types
- Binary relationship models
- Conceptual graphs
- Deep structure sentence models
- Entity-relationship models
- Function-oriented or oprations-oriented models
- N-ary relationships models
- Network models (CODASYL も含む)
- Object-role models
- Process-interaction models
- Relational models

・Semantic nets
・Set theoretic models

これらのモデル型が互いに等価なのか，おのおの違いはどの点にあるのかということは残念ながら，現状では十分クリアになっていない．しかし，[ISO 82] ではこれらのうち同種類の概念を使うものをグループ化して，データモデル型を大雑把に次の3種とする．

・Entity-Attribute-Relationship approaches
・Binary and Elementary N-ary Relationship approaches
・Interpreted Predicate Logic approaches

この大分類のうち，最後のものは形式論理学に基礎を置くもので，[ISO 82] のかつてのまとめ役 T. B. Steel Jr. の強い影響で報告書に載せられたものと思われるが，将来はともかくとしてこの世界ではまだ広く市民権を得ていないように思われる．ここでは上の2つをやや詳細に眺めよう．

3.3.1 Entity-Attribute-Relationship approaches (EAR)

この分類のデータモデル型では主体 [EAR]，**値**，関連 [EAR] を3つの基本的な構成要素とする．値は，主体 [EAR] のさまざまな性質を叙述するものであり，値が主体 [EAR] のどの性質に関するものかを区別するものが属性 [EAR] である．関連 [EAR] は後に見るように主体 [EAR] どうしに範囲を限定した関連である．

[例 3.22] 主体 [EAR]: Jack
Jack の属性 [EAR] '現住所' の値：東京
Jack の属性 [EAR] '国籍' の値：米国

3.2節での概念で言い直すならば，Jack はキー主体で '東京'，'米国' などの主体の間に

・人 'Jack' は現住所 '東京' である
・人 'Jack' は国籍 '米国' である

という関連があるということになる．

'東京'，'米国' は 'Jack' の叙述をしている**局面** (context) では '値' と考えていて '主体 [EAR]' とは考えていない．

局面が変って

・東京は人口 1,200 万人で知事は鈴木氏である

という叙述をしているときは，'東京' を '主体 [EAR]' と考える．このとき，人口，知事などは '東京' の属性 [EAR] で，その値はそれぞれ 1,200 万人，鈴木氏である．

つまり，EAR ではある '主体を主体 [EAR]' と見るか '値' と見るかはある局面 (ある関連を念頭に置いていること) に依存する相対的な見方を採用している．

1つの関連型でキー主体型とそうではない主体型とを判別することは設計の簡素化に役立つ．データベース型の骨格はキー主体型でほぼ決まると考えてよいからである．

まとめると，ある局面での

主体 [EAR] = キー主体
値 = その局面でキー主体でない主体

である．

主体 [EAR] を定めると属性 [EAR] ごとに値が定まるからこれを一まとめにして考えると，これは従来のレコードで表現できることになる．EAR は**レコード志向モデル型**である．

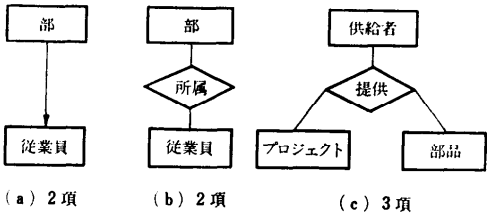
以上は，EAR における主体 [EAR] と値との関連であったが，主体 [EAR] 間の関連の扱いについて次に見よう．EAR では主体 [EAR] 間の関連だけを関連 [EAR] という．関連 [EAR] の扱いには次の2種類が存在する．

・2項
・n項

2項関連 [EAR] の立場を採るものとしては [Bachman 69]，n項関連 [EAR] の立場を採るものとしては [Chen 76] が代表的である．

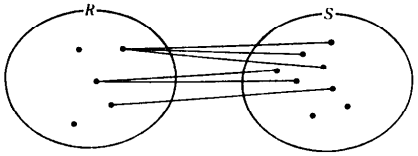
2項関連 [EAR] では関与する主体 [EAR] の数を2個に限定する．n項関連 [EAR] ではこの制約がなく，何個の主体 [EAR] が1個の関連 [EAR] に現れてもよい．

図-8 にこの差を示す．(a) は [Bachman 69] による2項関連型 [EAR] の表現である．2項関連型 [EAR] が矢印によって示されている (**Bachman Diagram**) (**Data Structure Diagram**)．(b) はこれと同様の2項関連型 [EAR] の [Chen 76] による表現である．関連型 [EAR] が菱形の箱によって示されている．[Chen 76] の関連型 [EAR] は3項以上の関連型 [EAR] も (c) のように表現できる．

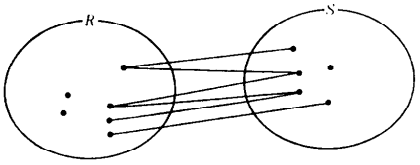


(a) 2項　(b) 2項　(c) 3項

図-8 2項関連とn項関連



(a) (1:n) 関連型



(b) (m:n) 関連型

図-9 (1:n)および(m:n)関連型

一般に R, S 2つの主体型の間の関連型Fは，関連づけられる主体の対応関係に関して次の2種類に分類される．

・(1:n)型または(n:1)型

・(m:n)型

[定義 3.23] F を次のように定義される関連型とする．

$F[(x \leqslant R), (y \leqslant S) : p(a : x, b : y)]$

このとき

(1) R と S とは F に関して **(1:n)** の関連を持つとは，F' を F のファイルとしたときすべての $f_1, f_2 \leqslant F'$ に対して $f_1(b)=f_2(b)$ ならば $f_1=f_2$ が成立つことをいう．

(2) 上記が成立するとき，SとRとは（上で順序が逆に）F に関して **(n:1)** の関連を持つという．

(3) (1)あるいは(2)が必ずしも成立たないとき，R と S とはFに関して **(m:n)** の関連を持つという．(1), (2), (3)に応じてFは (1:n), (n:1), (m:n) の関連型であるという．

図-9 に (1:n) ならびに (m:n) の関連を示す．

[Bachman 69] は (1:n) または (n:1) 関連型 [EAR] のみを認める．[Chen 76] の関連型 [EAR] は，それが2個の主体型間の関連型を表わすとき，(m:n) を許す．

[Bachman 69] のように関連型 [EAR] として2項のものに限り，(m:n) 型を禁止することは [Chen 76] に比べて大きな制約のように見えるが，これには次のような利点がある．

ときとして関連 [EAR] そのものを主体 [EAR] のように考えたくなることがある．[Bachman 69] のような制約を課している場合には，たとえば**図-8**(a)のように部と従業員との間の関連 [EAR] は従業員に属する主体と1対1に対応する．したがって，関連 [EAR] と主体 [EAR] とを同一視することができる．[Bachman 69] で直接は表現できなかった関連 (**図-8**(c)) も，**図-10** のような主体型 '提供' を設けることにすれば，やはり，関連を主体で表現したことになる．したがって，[Bachman 69] のデータモデル型では関連 [EAR] は常に主体 [EAR] 間で定義されたものと考えることができる．

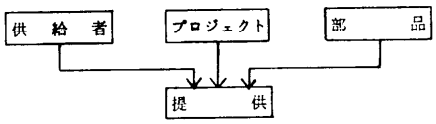
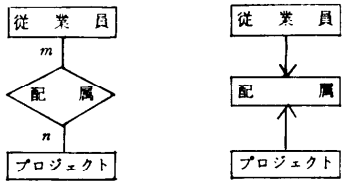


図-10 図3.6(c)と等価な [Bachman 69] のモデル



(a) (m:n) 関連型　　(b) (a)と等価な [Bachman 69] モデル

図-11 (m:n) 関連型と等価な Bachman モデル

これに対して，n 項の関連型 [EAR] または2項であっても (m:n) の関連型 [EAR] を認める [Chen 76] においては，関連 [EAR] を主体 [EAR] と同じように扱いたくなるときどのように考えるべきであろうか．関連 [EAR] と主体 [EAR] との間に関連を考えたくなったとき，それは関連 [EAR] であろうか．関連 [EAR] と関連 [EAR] の関連は関連 [EAR] であろうか．

これを扱うには，別の局面で関連 [EAR] を主体 [EAR] として考えるのが自然である [Schiffner 79]．n 項関連型 [EAR] は図-8(c)から図-10のように変換することによって関連 [EAR] を主体 [EAR] として表現できる．2項 (m:n) 関連型 [EAR] の扱いも同様である (**図-11**)．

つまり，[Bachman 69] のデータモデル型は，[Chen 76] のデータモデル型で，別局面で主体 [EAR] とみなすものを早手回しに主体 [EAR] をみなすことを行っていたものと解釈できる．[Bachman 69] でなにゆえ (1:n) 2項関連型 [EAR] しか考えなかったのか

関連型の項数	2項関連型の対応関係	関連自身の属性の有無	具体的なデータモデル型
2	(1 : n)	有	
2	(1 : n)	無	[Bachman 69]
2	(m : n)	有	
2	(m : n)	無	
n	(1 : n)	有	
n	(1 : n)	無	
n	(m : n)	有	[Chen 76]
n	(m : n)	無	[Chen 80 b]

図-12 EAR データモデル型の分類

キー主体型: X

a	b
(x_1,	y_1)
(x_2,	y_2)
(x_3,	y_3)
(x_4,	y_1)
(x_5,	y_1)

図-13 a が主キーのケース

という理由は，上記の解釈で関連型 [Chen 76] と考えているもの（たとえば，図-8, 図-10 の提供，あるいは図-11 の配属）と関連する主体型 [EAR]（たとえば図-8 の供給者，プロジェクト，部品あるいは図-11 の従業員，プロジェクト）との間の関連が (1 : n) 2項であるのと同じ理由による．つまり関連型それ自身とそれに関連する主体型との間の関連型は，(1 : n) 2項であるからである．関連型を主体型とみなすデータモデル型では (1 : n) 2項関連型が必要でありかつ十分である．

関連 [Chen 76] を別の局面で主体 [Chen 76] とみなすことを許せば，当然それが属性 [Chen 76] を持つことは許される．しかし [Chen 76] では関連 [Chen 76] それ自体が属性 [Chen 76] を持つことを許しており，ここで一体関連 [Chen 76] に属性 [Chen 76] を許すべきかどうかという恣意性 (arbitrariness) が生ずる [Davenport 79]．[Chen 76] のデータモデル型は表現力が豊かで素人向けするデータモデル型であるが，その利点のためにかえって解析が面倒になる．[Chen 76] のデータ・モデル型に多数の変種が存在し，Chen 自身その分類を [Chen 81b] で行っているくらいである．

以上述べた EAR データモデル型を分類して示すと**図-12** のようになる．

3.3.2 Binary and Elementary *N*-ary Relationship Approaches

この分類のデータモデル型では主体，関連を2つの基本的な構成要素とする．関連には何個かの主体が関与するが，その個数を2に限定するか，特に2とは限定しないかでさらに2つに分かれる．2個に限定するものが **Binary Relationship Approach** で [Abrial 74], [Nijssen 77 a], [Senko 73], [Senko 76] などがその例である．この限定がないものは **Elementary *N*-ary Relationship Approach** で，[Falkenherg 76]，ならびに本稿の 3.2 節で述べたデータモデル型は，これであった．[Codd 72] も用語法は異なるがこの一種と考えることができよう．

いずれの場合にも基本的には主体間の直接的な関連だけを問題とするから，**対象志向データモデル型**といえる．ただし，対象志向データモデル型といえども適当に概念を追加して（たとえばキー主体など），これをレコード志向のデータモデル型のように見せかけることは可能である．逆にレコード志向データモデル型を適当に制限して対象志向データモデル型と見せかけることも可能である．[Chen 81 b] では，Entity Relationship モデルの特殊ケースとして [Abrial 74], [Senko 73] を分類している．出発点としていずれのデータモデル型を採用するかは，趣味の問題である．

3.4 抽 象 化

大規模な対象世界を記述しようとするとき，全記述を同一レベルの詳しさで扱うことはできない．細かな単位の対象を適当に寄せ集めて，少し大きな単位の対象として記述するということを何段も繰り返さなくてはならない．より大きな単位の対象を考えて，それを構成する小さな単位の対象が何であるかについては問わないことを抽象化という．

抽象化は，プロセスが対象のときにも考えられるがここでは専らデータを対象とする抽象化について考える．

データモデル型では集合抽象化，列抽象化，導出の3種の抽象化がすでに議論されているのでこれらについて順に述べる．なお，この外に汎化を抽象化として説明する論文 [Smith 77 b] もあるが，すこし意味合いが異なるように思われるのでこれは別型定義として次節で改めて取り上げる．

集合抽象化 (set abstraction) は，**繰返しグループ** (repeating group)，**多価** (multivalue) 等といわれている概念の表現の際現れる．

例えば主体型 X, Y の間の2項関連型 F を考える．

$F : [(x \leqslant X), (y \leqslant Y) : P(a : x, b : y)]$

ここで a が主キーで，1つのファイルが例えば**図-13** のように与えられているものとする．このファイ

キー主体型: Y

a'	b'
$(\{x_1, x_4, x_5\},$	$y_1)$
$(x_2,$	$y_2)$
$(x_3,$	$y_3)$

(a)

キー主体型: Y

a'	b'
$(X_1',$	$y_1)$
$(x_2,$	$y_2)$
$(x_3,$	$y_3)$

(b)

図-14　b' が主キーのケース

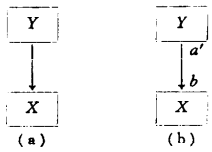


図-15　関連型と属性

キー主体型: XY

a	b
$(x_1,$	$y_1)$
$(x_1,$	$y_2)$
$(x_2,$	$y_1)$
$(x_3,$	$y_4)$

(a)

キー主体型: X

a_1	a_2
$(x_1,$	$\{y_1, y_2\})$
$(x_2,$	$y_1)$
$(x_3,$	$y_4)$

(b)

キー主体型: Y

b_2	b_1
$(\{x_1, x_2\},$	$y_1)$
$(x_1,$	$y_2)$
$(x_3,$	$y_4)$

(c)

図-16　同じ意味の3種の表現法

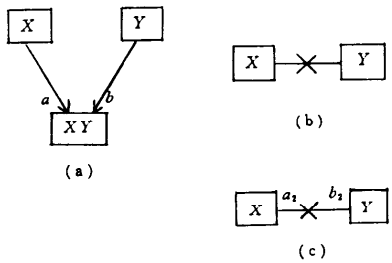


図-17　$m:n$ 関連型と属性による表現

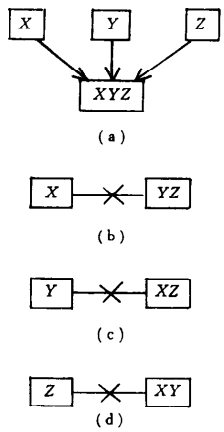


図-18　一般的な関連型の属性 [EAR] による表現

ルが表現していることを別の関連型 F' のファイルとして**図-14**(a)のように表現することもできる.

$$F' : [(x \leqslant X'), (y \leqslant Y) : P'(a' : x,\ b' : y)]$$

F' の属性 a' は，F の属性 b の**逆** [Hammer 81] と呼ばれる．F は X をキー主体型として対象世界を表現したのに対し，F' では逆に Y をキー主体型として対象世界を表現している．図-14(a)の例の $\{x_1, x_4, x_5\}$ のように，逆属性を考えると，属性値として多価のものが現れる．

$\{x_1, x_4, x_5\}$ の代りに $X_1' \leqslant X'$, $S(X_1') = \{x_1, x_4, x_5\}$ なる主体 X_1' を考え，図-14(a)と同じ意味が**図-14**(b)で表現されていると仮定する．X_1' のように主体の集合を1個の主体で代表することを**集合抽象化**といい，集合を代表する主体（X_1' のようなもの）を**集合主体**という．X' のようにその実現値として集合主体が属する主体型を**集合主体型**という．

集合抽象を許すことは表現力を豊かにはするが，反面，同じ意味の別表現を許すことになる．図-13，図-14(a)は同じ意味を表現しているが，なにを関連型のキー主体型と考えるかが異なっている．冗長であるが X, Y 2つのキー主体型を考えると**図-15**(a)のバックマン図式で示されるようなキー主体型間の関連型が存在する．キー主体型 X の方からこの関連型を眺めると属性 b があるように見える．またキー主体型 Y の方からこの関連型を眺めると属性 a' があるように見える．2項関連型がこのように属性でも表現されているとき**図-15**(b)のようにスキーマ図で表わすことにする．$(1:n)$ の2項関連型を表現した場合，一方の属性の定義域は集合主体型になる．

$(m:n)$ の2項関連型を同様に属性を用いて表現すると関連型に現れるキー主体型の双方に集合主体型を定義域とする属性が現れる．例えば2つのキー主体型 X, Y の間の関連型 XY は**図-16**(a)のように表現され，スキーマ図は**図-17**(a)あるいは XY を省略して**図-17**(b)のようである．これを X, Y をキー主体型とし，属性 A_2, B_2 を用いて**図-16**(b)あるいは(c)のように表現することができる．キー主体型 X の方からこの関連型を眺めると属性 a_2 があるように見える．またキー主体型 Y の方からこの関連型を眺めると属性 b_2 があるように見える．属性 a_2 も b_2 もともに定義域として集合主体型を持つ．[Hammer 81] は a_2 が a 上で b と match すると呼んだ．同様に b_2 は b 上で a と match している．彼の考察は，関連型が属性によっても表現し得ること，また，属性で表現したとき

その属性値が関連型を陽に表現する図-16(a)のような表現とどのように関与するかを見たことにある.

さて, frame

$F[x_1 \leqslant D_1), \cdots, (x_n \leqslant D_n) : P(a_1 : x_1, \cdots, a_n : x_n)]$

に属する関連 f は

$(v_1, v_2, \cdots, v_n)$

の形をした列を1個の主体で代表していると考えられる. この意味で関連を**列主体**, あるいは [Smith 77a] にならって**統合化主体**といい, また列を列主体で代表することを**列抽象化**あるいは**統合化**という. [Nijssen 77] は列主体のことを objectified sentence と呼んだ.

以上の集合抽象と列抽象を用いると, n項関連もことごとく属性 [EAR] により表現することができるようになる. **図-18**(a)のような関連型の表現もたとえば同図(b), (c), (d)のいずれかあるいはこれらの組合せによって表現できる.

例えば**図-18**(b)の場合, キー主体型 X の属性としては, Y, Z を統合化した統合化主体型 YZ に属する統合化主体のそのまた集合抽象化の結果得られる主体を属性値とする属性が現れる.

データベース実現値に蓄積されているデータに加工を施し得られるデータを**導出データ**という. データベース実現値に対して検索要求を出すということは, 換言すれば導出データを要求しているということである. 導出データが与えられた場合, 人は, それがどのような基礎データから得られたかということの詳細を議論せずに, それを利用することができる. 導出データの導出は抽象化の一種と考えることができる.

導出データは, それを導出するに要する入力データとデータ加工ロジックの2つから記述することができる. 導出データは, この意味で一部プロセスの抽象化を含んでいると言ってよい.

3.5 別型定義

ある1つの主体は, 観点が違えばいくつかの異なる主体型に属すると考えるのが**別型定義**である. 別型定義にはいろいろの種類のものを考えることができるが, ここでは2種類のものに限定して話を進める.

1つは既存のいくつかの主体型 $T_1, \cdots, T_n$ に対して, 新たに

$S(T_1) \cup S(T_2) \cup \cdots \cup S(T_n) \subset S(T)$

となる主体型Tを考える (定義する) ことであり, **汎化** (generalization) と呼ばれる.

他の1つは既存の主体型Uに対して新たに

$S(T) \subset S(U)$

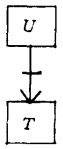


図-19 部分型, 汎型

となる主体型Tを考える (定義する) ことであり, **専化** (specialization) と呼ばれる.

一般に主体型 T, U の間に

$S(T) \subset S(U)$

の関係が存在するとき, TはUの**部分型** (subtype), UはTの**汎型** (generic type) といい, スキーマ図で**図-19** のように表わす.

別型定義は [Smith 77b] が最初に汎化という言葉をデータモデル型の議論に持ち込んだが, 人工知能分野でよく使われる意味ネットワーク (semantic network) の isa 関係も同じ概念を取扱っている [Wong 77]. 役 [Bachman 80] も多少目的意識は異なるが類似の概念である.

汎化は, 多くの主体型をまとめて1つの主体型として扱おうと考えるときに使用できる考え方である. 例えばポンプ, バルブ, パイプ, 歯車を汎化して「建設部品」という主体型を定義することがこれに相当する. いずれの主体型の主体も部品番号, 単価などの共通属性 [EAR] を持つ新しい汎型の主体として考えることができる. すなわち共通属性 [EAR] を持つ主体型同士は汎化することができる.

専化は, これと逆にある主体型の一部をなす主体型で特有の性質を持つものを考えることで, それにより意味の表現をより精密なものにすることができる. 例えば主体型「従業員」の部分型として「管理者」を考えることによって, 管理者特有の属性, 例えば管理職手当, 交際費などについての記述や, 他の主体型との間の関連型の定義を主体型「従業員」に関する記述とは別にできるようになる.

関連型 $F1, F2$ においてキー主体型をそれぞれ $X1$, $X2$ とする. もしも $X1$ が $X2$ の部分型ならば

$A(F1) \supset A(F2)$

となるように属性を定義することができる. つまり, $X2$ の属性 [EAR] をそのまま $X1$ の属性 [EAR] とすることができる. より広い範囲の主体型 [EAR] が有している属性 [EAR] は狭い範囲の主体型 [EAR] に対しても意味を持つということである. 上記の例でいうならば, 建設部品に単価という属性 [EAR] があ

れば，ポンプ，バルブにも単価という属性 [EAR] を持たすことができる．これを**属性遺伝** (attribute inheritance) [Hammer 81] という．

汎化，部分化は設計者のものの見方によって大きく変化するところで実際，スキーマの設計を多人数で行って見るとこれが原因となって設計結果が何種類も現れる．例えば在庫の増減をもたらす事象を表わす主体型として

(1) 仕入，売上，返品，サンプル出荷，……

と細かく原因別にとらえることもできるし

(2) 在庫増，在庫減

と2つに分けることもできる．もっと極端な場合には

(3) 在庫変動

として全部を1つの主体型にまとめてしまうことさえある．

実用上別型定義で気をつけなくてはならぬのは 3.1 節で述べた主体と名前の一意対応性の [仮定 3.3] である．これは仮定というよりもむしろデータベース設計者が気を配らなくてはならない注意のようなものである．

T_1, T_2 が2つの主体型で T_1 が T_2 の部分型であるとき，T_2 での主体の命名規則が T_1 での主体の命名規則として利用可能でなくてはならない．T_1 が専化によって T_2 から得られたものであるならば，単純に T_2 での命名規則を使用すればよいから問題はない．問題は汎化のときである．

例えば T_1 を郵便貯金者，T_2 を銀行預金者として納税者 T_3 を T_1, T_2 の汎型として次のように定義したとする

$$S(T_3)=S(T_1)\cup S(T_2)$$

T_3 は T_1 の汎型であるが T_1, T_2 の主体を識別する命名規則を T_3 上で考えることは難かしい．つまり，汎化は命名規則の存在を自動的には保証しない．

もしも T_1, T_2 が互いに素であるならば ($S(T_1)\cap S(T_2)=\phi$ のとき) これは保証される．$T_3 \ll e$ なる主体の名前は次のように定義すればよいからである．

$$\text{name}_{T_3}(e)=\begin{cases}(T_1, \text{name}_{T_1}(e)) & \text{if } e \ll T_1\\ (T_2, \text{name}_{T_2}(e)) & \text{if } e \ll T_2\end{cases}$$

汎化のとき命名規則がなんらかの方法で得られるならば (一般には複数の主体型に属している主体間の変換テーブルを用意すればよかろう)，まずさきに汎型 T_3 を考えて，T_1, T_2 が専化で得られたものと考えてしまうことができる．したがって，実用上は汎化の代わりに専化だけがあれば十分である．

3.6 拡張性

以上ではいわゆる静的記述という部類に入るデータモデル型のさまざまな概念を説明した．**時間**の経過の記述とか**引金** (trigger) とかいう動的な記述についての直接的な概念は紹介されていない．しかし，これらの概念またさらに別の概念もかなり抽象化の手段を使うことによってデータモデル型を拡張し，表現することができる．

例えば [Foucaut 78] は主体型に関する叙述の他に時間的生起に関する事象 [Foucaut 78], 操作 [Foucaut 78], 等をも関連型の一種として表現することを試みている．

抽象化の手段が残されているということは，データモデル型を拡張することができるということである．

謝辞

データモデル型について講座をまとめるに当り，日本システミックス株式会社 椿正明博士，経済企画庁 佐藤英人氏ならびに査読者から大変貴重な意見を頂いたことを感謝します．

用語索引

参 考 文 献

[Abrial 74] J.R. Abrial: *Data Semantics, in Database Management*, J. Klimbie and K. Koffeman(eds), North-Holland, 1-60 (1974).

[Armstrong 74] W.W. Armstrong: *Dependeny Structure of Data Base Relationships*, Proc. IFIP Congress, 580-583 (1974).

[Bachman 69] C.W. Bachman: *Data Structure Diagrams*, SIGBDP 1, 2, pp. 4-13 (1969).

[Bachman 80] C.W. Bachman, *The Role Data Model Approach to Data Structures*, Proc. International Conf. on Data Bases, 1-18 (1980).

[Biller 76] H. Biller and B.J. Neuhold: *Concepts for the Conceptual Schema*, in [Nijssen 77 b], pp. 1-30 (1977).

[Bubenko 77] J.A. Bubenko jr. IAM: *An Inferential Abstract Modelling Approach to Design of Conceptual Schema*. SIGMOD. pp. 62-74 (1977).

[Chen 76] P.P. Chen: *The Entity-Relationship Model—Toward a Unified View of Data*. TODS 1.1, pp. 9-36 (1976).

[Chen 80 a] P.P. Chen (ed): *Entity-Relationship Approach to Systems Analysis and Design*, North-Holland (1980).

[Chen 80 b] P.P. Chen: *Entity-Relationship diagrams and English Sentence Structures*, in [Chen 80 a], pp. 13-14 (1980).

[Chen 81 a] P.P. Chen(ed): *Entity-Relationship Approach to Information Modeling and*

Analysis, ER Institute (1981).

[Chen 81 b] P. P. Chen: *A Preliminary Framework for Entity-Relationship Models*, in [Chen 81 a], pp. 19-28 (1981).

[Codd 72] E. F. Codd: *Further Normalization of the Data Base Relational Model, in Data Base Systems*, Courant Computer Science Symposia Series, Vol. 6. Englewood Cliffs, N. J.: Prentice-Hall (1972).

[Davenport 79] R. A. Davenport: *The Application of Data Analysis—Experience with the Entity—Relationship Approach. ER-Conference*, pp. 463-481 (1979).

[Falkenberg 76] E. Falkenberg: *Concepts for Modelling Information*. IFIP TC-2, pp. 95-109 (1976).

[Foucaut 78] O. Foucaut and C. Rolland: *Concepts for Design of an Information System Conceptual Schema and its Utilization in the Remora Project*, VLDB, pp. 342-350 (1978).

[Hammer 81] M. Hammer and D. McLeod: *Database Description with SDM: A Semantic Database Model*. TODS 6, 3, pp. 351-386 (1981).

[ISO 82] J. J. van Griethuysen *et al.* (ed): *Concepts and Terminology for the Conceptual Schema*, ISO TC 97/SC 5/WG 3, March 15 (1982).

[Kent 78] W. Kent: *Data and Reality*, North-Holland (1978).

[Kerschberg] L. Kerschberg, A. Klug and D. Tscichritzis: *A Taxonomy of Data Models*. VLDB, pp. 43-64 (1976).

[Nijssen 76 a] G. M. Nijssen: *A Gross Architecture for the Next Generation Data Base Management*, in [Nijssen 76 b], pp. 1-24 (1976).

[Nijssen 76 b] G. M. Nijssen (ed): *Modelling in Data Base Managenant Systems*, Proc. of the IFIP Working Conference on Modelling in Data-Base Managenent Systems, North-Holland (1976).

[Nijssen 77 a] G. M. Nijssen: *Current Issues in Conceptual Schema Concepts*, in [Nijssen77 b], pp. 31-65 (1977).

[Nijssen 77 b] G. M. Nijssen (ed.): *Architecture and Models in Data Base Management Systems*, Proc. of the IFIP Working Conference on Modelling in Data Base Management Systems, North-Holland (1977).

[Schiffner 79] G. Schiffner and P. Scheuerman: *Multiple Views and Abstractions with an Extended-Entity-Relationship Model*, Computer Languages, Vol. 4, pp. 139-154 (1979).

[Schmid 75] H. A. Schmid and J. R. Swenson: *On the Semantics of the Relational Data Model*, SIGMOD, pp. 211-220 (1975).

[Senko 73] M. E. Senko, E. B. Altman, M. M. Astrahan and P. L. Fehder: *Data Structures and Accessing in Data Base Systems*, IBM Syst. J. pp. 30-93 (Dec. 1973).

[Senko 76] M. E. Senko: *DIAM as a Detailed Example of the ANSI SPARC Architecture*, pp. 73-94 (1976).

[Smith 77 a] J. M. Smith and D. C. P. Smith: *Database Abstractions: Aggregation*. CACM 20. 6, pp. 405-413 (1977).

[Smith 77 b] J. M. Smith and D. C. P. Smith: *Database Abstractions: Aggregation and Generalization*. TODS 2, 2, pp. 105-133 (1977).

[Smith 79] J. M. Smith: *A Normal Form for Abstract Syntax*, VLDB, pp. 156-162 (1978).

[Tsubaki 82] 椿 正明: データベース論理設計法構築の試み(I)～(V), Computer Report, Vol. 53～57 (1981-9～1982-1).

[Wong 77] H. K. T. Wong and J. Mylopoulos: *Two Views of Data Semantics: A Survey of Data Models in Artificial Intelligence and Database Management*. INFOR 15, 3, pp. 344-383 (1977).

[Yeh 78] R. Yeh, N. Roussopoulos and P. Chang: *Data Base Design—An Approach and Some Issues*, in Infotech State of the Art Report, Data-Base Technology, Vol. 2: Invited Papers, pp. 443-477 (1978).

(昭和57年3月8日受付)