データベース・システム 77－5
（１９９０．５．１７）

ＪＤＭＦについて

穂鷹良介

情報スキーマ調査研究委員会

データベースが世のなかにはじめて出現した当時より様々なデータモデルの提案がされてきた。現在日本規格協会／情報技術標準化研究センター(INSTAC)内に設けられた情報スキーマ調査研究委員会では規格としてのデータモデルの検討を進めており、委員会としての一応の成果に達した。今後は本案を日本国内のパブリックレヴユーにまわし改良について一般からの意見を反映させたのちに日本工業規格案として上位委員会に提案する予定となっている。本報告は提案データモデルについてその開発思想、概略を説明する。

ON JDMF
JIS Data Modelling Facility

Ryosuke Hotaka

Study and Research Committee on Information Schema

From the beginning of the database history, various data models have been proposed by many authors. Currently, in the Study and Research Committee on Information Schema established in Information Technology Reserch and Standardization Center (INSTAC) of JAPANESE STANDARDS ASSOCIATION (JSA), a data model has been studied and proposed as a standard. The proposal will be distributed in Japan to gather public opinion. And after reflecting the various improvements, the committee is planning to present it to the upper committe to be authorized as a Japanese Industrial Standard. This report gives the underlying philosophy and the overview of the proposed data model.

1　ＪＤＭＦとは

データモデルの議論が始められて十数年が経過した。データモデルの基本的な概念は個々のデータモデルごとにそれほどの違いは認められないにも拘らず、データベース設計者ごとに使用するデータモデルは個々別々で、使われている概念、その概念を表す用語も別々である。

その結果、データベース設計に携わっている専門家同士でもお互に相手の言っていることを理解するのに困難が生じる。異なったデータモデルを適用して得られる設計成果はたとえ同一の対象を表現するものであっても互いに共通性はなく、データベースが目指すデータの共有が阻害されているのが現状である。

これは、あたかもネジの目的はある部品をある場所に固定するのが目的であるのにも拘らず、５mm、６mm、７mmのネジでは満足せず５．４mm、５．４９mm、６．０２mmとネジの長さに多種類の変化を求めている状態にたとえることができよう。

ＪＤＭＦ（JIS Data Modelling Facility）は、ネジの規格で行なうのとほぼ同じことをデータベース設計のために用いるデータモデルに対して行なおうとするものである。

１９８８年データベース関連のＪＩＳ規格案を検討する日本規格協会情報技術標準化研究センター内に設けられたデータ管理調査研究委員会では、世界に先んじてデータモデルの規格化の必要性を認め日本国内で使用するデータモデルの規格化の可能性の検討を開始した。

１９８９年には、更にこの活動を専門に取り扱う情報スキーマ調査研究委員会が独立した委員会として設立され、活発に活動を行なった。その結果委員会内でほぼ一致した仕様をＪＩＳ規格案としてまとめたものがＪＤＭＦと名付けられたデータモデルである。１９９０年４月現在までに１１版になるまで改定されてきている。

なおここでデータモデルのことをデータモデル機能（ＤＭＦ）と呼んでいるが、その理由はデータモデルという言葉で具体的な応用分野を表現する応用データモデルを指す場合があってそれとの混乱を避けるためである。

ＪＤＭＦはデータベース設計作業の上流工程で用いられることを念頭においたデータモデルである。個々のデータベースの実現方法に関係することについては敢えて定めていなく論理的な段階で止めている。その結果設計結果は機種その他の制約を越えいかなる利用者でも有効活用することができる。これはある意味で非常に大切な性質である。

抽象的なモデルの仕様を定めるという試みは、具体的に実現されるソフトウェアあるいはハードウェアの仕様を定める規格と基本的に異なり、多くの関係者にその必要性、有効性を認めて貰うのはなかなか難しい。データ処理が高度に発展した社会においてのみこのような規格の必要性が理解され、かつ有効に利用されるといえよう。ＪＤＭＦと同程度の抽象的あるいは論理的水準にありかつ完成度の高い規格はデータ処理分野に関しては世界にその類を見なくこの分野に関して我国がはじめて世界の水準に達したと判断している。

2　基本思想

データベースには

（１）データの共有性　そこに蓄積されるデータを共有したい

（２）表現の容易性　対象を容易に表現したい

（３）利用の容易性　蓄積されているデータを容易に利用したい

という期待が持たれている。これらの異なる目的を同時に満足させるモデルを作り上げるのは現状では仲々技術的に難しいものがある。もし作業をマネージできる範囲に分解してすこしずつ解決していくとしたらどのような順番でこの目標を達成したら良いであろうか。

一見して（１）を先に解決したならば（２）、（３）の目的はその後に機能追加の形で考えればよいと分る。これに対して（１）の考慮無しに（２）、（３）を定めた後に（１）の問題を解決するのは容易ではないように思われる。

したがってＪＤＭＦでは（１）のデータの共有性を最優先に考えた。データの共有のためには更に具体的にどのような性質が必要とされるか考えてみる。

共有のためにはあるデータが蓄積されたときそのことが各利用者に容易に理解されなくてはならない。共有データはその性格上いくつもの異なった見方が可能である。もしもデータがある特定の利用からみて便利な形で蓄積されているのだが他の利用目的からはすぐには利用しにくい形を取っていたとすると後者の目的のために別にデータの蓄積をせよという要求がでることになる。あるいはデータの整理の仕方がマチマチだとそもそも利用可能なデータがデータベースのなかに存在するということを見抜けなくて、新規に同じデータをデータベースに蓄積することになる。

このように同じデータを別のデータと思って複数個データベースのなかに蓄積してしまうと各々のデータの更新手続きは当然同期しないことになり、結果として同じ対象を記述するデータに相違が生じる。これは正しい意味でのデータの共有ができていないことを意味する。したがってデータベースに新しくデータを蓄積しようとする利用者は既存のデータと同じものを重複しようとしていないか何等かの方法で確認する必要がある。

つまりデータを本当の意味で共有するためにはデータの蓄積のしかたあるいはデータの設計結果が一意であることが望ましい。したがってＪＤＭＦでは設計結果ができるだけ一意

になるように仕様を定めている。

このためにＪＤＭＦでは基本的と思われる概念から出発してすこしずつ新概念を基本概念として追加していくが、既存の概念だけでは新概念の代りを務めることには極めて困難があると見極められる場合にのみそれを認めるという立場をとった。

データモデル機能に取り入れる概念の数を可能な限り少なくするという方針を採用すると機能不足が心配されるが、それは別途共有データの表現という目的から切り放して基本概念ではなく追加概念として導入すればよい。その際にデータの共有のためには追加概念と基本概念との関連を明快に定義しておく必要がある。このことにより追加概念によってデータベースに蓄積されたデータは基本概念だけを用いて蓄積されたデータと同様に扱うことができデータの共有性を損なうことはない。

基本概念として導入する概念を極力少なくするようにしたのだが、そのためにかえって強力な機能をもつことになった点がある。ＪＤＭＦはあらゆる対象を表現することを目指しているから、特別の場合としてデータ辞書それ自身もＪＤＭＦの基本概念を用いて記述される。その結果データベースとデータ辞書の両方が同じ考え方で扱われることとなりメタデータの扱いに柔軟性を増した。この性質はデータモデル機能の自己記述性と呼ばれる。データモデル機能の自己記述性もデータの共有性を追求した結果得られた性質である。

３　基本概念

以下ＪＤＭＦの基本概念の説明ならびにその必要性について述べる。ここでの説明は分りやすさを主眼としているので厳密なものでも完全なものでもない。規格を使用するときには必ず規格本文を参照されたい。

３．１　モデル化の基本部品

応用データモデルで記述の対象を表現する概念が必要である。これを実体実現値という。実体実現値が応用データモデルの中でどのような意味をもつか分類するのがデータの整理上意味をもつ。この分類に対応する概念を実体型という。データモデル機能によっては実体という言葉で実現値の意味をもたせる場合と以下に述べる実体型の意味をもたせる場合とがあるが、その誤解が生じないように実現値か型かを明示する。

実体実現値として如何なる対象を代表させるのも応用データモデル設計者の自由である。したがって実体実現値自身は特別の意味を有していない。意味が定まるのは設計者がその実体実現値がどの実体型に分類されるかを定めたときである。

３．２　概念の構造化

実体実現値をモデル化に際して用いる部品とすればその部品を適当に組合せてより複雑な構造物を作ることができる。

３．２．１　アグリゲーション

たとえば山田太郎氏と田中良子さんが結婚したとする。山田太郎氏を実体実現値ｘとして表現し、田中良子さんを実体実現値ｙとして表現したとする。ｘもｙも人間を表す実体型Ｍの実現値であると考えておく（図１）。

二人の結婚はｚ＝（ｘ，ｙ）と表現できるであろう。このように複数個の実体実現値を組合せてできる概念を再び実体実現値と考える考え方をアグリゲーションといい、組み合わされてできた実体実現値をリスト実体実現値という。ｚはリスト実体実現値である。

リスト実体実現値のときも、同様にその分類を表現する概念として実体型を考える。これをリスト実体型という。ここでは結婚という概念に対応するリスト実体型をＺで表現したと考えよう。

リスト実体実現値ｚが与えられたとしてその構成要素であるｘあるいはｙを識別できる必要がある。ｚ＝（ｘ，ｙ）でｘの占める位置、ｙの占める位置を識別する概念を列と呼ぶ。ｘ，ｙに対応する列をａ，ｂとする。

このときａはＺを構成する、ｂはＺを構成するということにする。列がリスト実体型を構成している。またａはｘを識別する、ｂはｙを識別するということにする。列がリスト実体実現値の構成要素である実体実現値を識別している。またｚでのａの値はｘである、ｂの値はｙであるという。

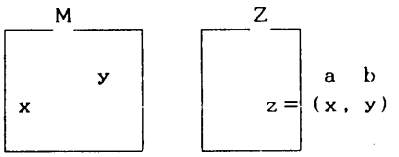
列は実体実現値がどのような役割をもってリスト実体実現値に参加するかまた列が識別する実体実現値がどの実体型の実現値であるかをしめす。上の例ではａは「夫」という役割をｂは「妻」という役割を持ち、いずれもＭという実体型の実現値を識別する。列が識別する実体実現値に対応する実体型をその列の定義域という。

３．２．２　表

リスト実体実現値の集合を表という。リスト実体実現値では組合せられる実体実現値の１個１個に列が対応していたが、表ではただ単にリスト実体実現値を集めたものである。図２に表の構造を例示する。

図２の表 T1（従業員）の構成要素は z1, z2 などのリスト実体実現値で特に行と呼ばれる。各々の行は ta1(主ｷｰ)、ta2(名)、ta3(住所)、ta4(電話) などの列によって行の構成要素が更に識別される。

後の説明のために図２の表を構成する列の定義域をそれぞれ原子実体型 e1(社員)、e2(氏名)、e3(住所)、e4(電話番号)としておく。したがって各列は表原子列である。



列は	リスト実体型を構成する
a	Z
b	Z

列は	リスト実体実現値zの構成要素を識別する
a	x
b	y

列は	定義域を持つ
a	M
b	M

列は	役割を持つ
a	夫
b	妻

図1　アグリゲーションの構造

	T1 従業員			
	ta1 主キー	ta2 名	ta3 住所	ta4 電話
z1 = (	123,	太田,	東京,	1234)
z2 = (	456,	田中,	大阪,	4567)

図2　表（表原子列からのみ構成される場合）

図3の表には定義域がリスト実体型である表複合列 ta9（入社日）が表を構成している。ta9 の定義域を e5（日付）とする（図4参照）。

	T2（従業員）				
	ta5 主キー	ta6 名	ta7 住所	ta8 電話	tc9 入社日
					b1 b2 b3 年 月 日
z1 = (	123,	太田,	東京,	1234,	z11)
z2 = (	456,	田中,	大阪,	4567,	z21)

図3　表（表複合列が表を構成する場合）

e5（日付）		
la1	la2	la3
Y	M	D

図4　リスト実体型

リスト実体型 e5 を構成する3個のリスト原子列 la1, la2, la3 はリスト実体型 e5 の実現値 z11 の構成要素を識別する。たとえば

z11 = (89, 10, 24)

とすると la1 は 89 を、la2 は 10 を、la3 は 24 を識別する。

表 T2 の b1、b2、b3 は表間接原子列で tc9 の定義域を構成する原子列 la1、la2、la3 に対応して自動的に設けられる。このように表を構成する列の値が再びリスト実体実現値でありそれをまた構成する構成要素がまたリスト実体実現値であるというようなことが、不定回生じる。

T3（部品展開）		
tc10 主キー		ta11 必要個数
b4 親	b5 子	
A	B	1
A	C	2
A	1	4
B	2	1
B	1	2

図5　定義域

図５の表で列 b4 の値「B」と列 b5 の値「B」とは同じ実体実現値を指しているがそのことは単に列の値を比較して分るものではない。これは列 b4 と列 b5 の定義域をあらかじめ同じ定義域 e6(部品) と定義しておくことによって判別がつく。たとえば列 b5 の値「1」と列 tal1 の値「1」とは二つの列の定義域が異なるため値が等しくても同一の実体実現値を指しているとは考えない。ＪＤＭＦでは列の定義域と列の値の対が一致するとき指している実体実現値が同一であるとする（副型が関連するときはもうすこし精密な述べ方が必要である)。 定義域概念は複数の表の列の値の参照関係を形式的に扱うために必要とされる。

表 T1 の ta1、表 T2 の ta5、表 T3 の tc10 の３個の列はいずれもその値が表の中の行を一意に識別するという共通の役割をもっていると考えられる。このことを明示的に表現するために役割という概念を設け、各列毎にその役割を与える。役割は列の意味を識別するものであるから、ある一群の役割に対して適用可能なロジックはその役割をもつ列がどの表を構成するものであっても等しくあてはまる。役割概念は表を越えて共通処理ロジックを構築するのに役立つ。

３．３　対象の管理、参照

対象世界を単に表現するだけではなく、対象世界を構成する要素のうち現在応用データモデルに取り入れられているものが何であるかを管理することが応用データモデルあるいはデータベースに期待される。ＪＤＭＦでは実体型を管理実体型と想定実体型の二つに分類する。管理実体型の実体実現値はそれに１対１に対応する行によって管理を行なう。管理実体型に対応する表を実表という。実表にはそれを構成する主キーがあるが、主キーの値は管理の対象となっている実体実現値を表現する。実表の主キーの定義域は対応する管理実体型である。たとえば上記の図２で表「従業員」は実表で、e2 (社員) は主キー ta1 の定義域であるから管理実体型である。

応用データモデルのある列の値が管理実体型の実体実現値を指している場合その値は管理実体型に対応する実表のある行の主キー値として現れなければならないものとする。管理されている実体型の実現値を参照したときそれが応用データモデルにとりいれられていない実体実現値である場合にはその参照は意味を失うと考えられるからである。管理実体型を定義域としてもつ列を参照キーあるいはＲキーという。ＪＤＭＦでは表間の関連はすべて参照キーによって表現するものと想定している。そのように関連の表現方法を制約することにより表間の関連の取扱いを単純にすることができる。

応用データモデルに取り入れられている一つの実体実現値が同時に二つの実体型 e、fの実体実現値になっている場合がある。特殊な場合として一方、たとえば e の実現値がすべて同時に他方、たとえば f の実現値であるという場合が実務上よく出現する。このとき e は f の副型であるという。

副型概念は必ずしも定義域が一致しない列の値が共通の実体実現値を参照していることを形式的に判別するために必要である。たとえば図２で示す表のほかに管理実体型 e8(管理職) に対応する実表（図６）があるものとする。３．２．２図５に関連して述べた判断規準だけに従うと ta12 の定義域は e8(管理職)、ta1 の定義域は e1(社員) で異なるから、ta1 の値 123 と ta12 の値 123 とは異なる実体実現値であると判断される。しかし管理職の実体実現値は同時に社員の実体実現値であると考えて e8 が e1 の副型であるとした場合には、副型 e8 の実体実現値は同時に汎型 e1 の実体実現値と考えるので ta1 の値 123 と ta12 の値 123 とは同じ実体実現値を指していると判断される。

管理職
ta12
主キー
======
123
500

図６　副型

３．４　副表

二つの表 TA1 と TA2 とがあって TA1 を構成する列がもつ役割はすべて TA2 を構成するいずれかの列の役割として現れ（役割の継承）、逆に TA2 のどの行に対してもその行を識別する主キー値と等しい主キー値をもつ行が表 TA1 に現れるとき TA2 は TA1 の副表であるという。

管理実体型 e が管理実体型 f の副型であるとき e に対応する実表は f に対応する実表の副表である。

３．５　実体実現値表現形式

同じ実体実現値を表現するのにたとえば単位系が異なると処理ロジックはそれを意識しなくてはならない。ＪＤＭＦではあらかじめ応用データモデルの設置者が定めた実体実現値の表現形式を原子実体型ごとに指定することができる。

４　設計支援辞書

ＪＤＭＦを用いてもモデル化の際に使用する用語等に自由度が高ければ、設計結果は一意ではなくなる。ＪＤＭＦでは、設計の根幹は列の決定にあり、列の決定は表、役割、定義域（実体型）、論理表現型のいずれを選択するかという問題になる。表のなか、役割のなか、実体型のなか、論理表現型のなかのいずれにおいても類似のものがあっては設計の一意性を損なう。したがって応用データモデル設置者ごとにこれらの標準を設定するのが適当と思われる。標準は設計支援辞書となるであろう。

5　将来の課題

ＪＤＭＦはかなり安定してきたとはいえまだまだ以下に示すような将来の課題を多く残している。

5．1　データモデル機能上の問題

（1）基本概念と追加概念の関係の明確化

ＪＤＭＦの概念の殆どは基本概念であるが、たとえば副表概念は厳密には基本概念とはいえない（副型概念があれば不要のようである）。副表概念も含めてＪＤＭＦには将来追加概念が導入される可能性があるのでその取扱い方に一定の方針が必要とされる。

（2）非ボイス・コッド正規形への対処

ＪＤＭＦは関係データベースの言葉でいうならばボイス・コッド正規形に限定された表の設計結果に対しては、原子的な役割概念だけできれいに設計できる。しかし他の表から結合操作によって冗長に持ち込まれた推移従属列の役割は、いわば表と原子的な役割の混合した複合役割のようなものになってしまう欠点が指摘されている。

（3）プロセス系との連係

現状ではＪＤＭＦで設計した結果をすぐ既存のデータベース管理システム等を利用して実現する容易ができていない。ＳＱＬなどＪＤＭＦと相性の良いデータベース管理システムへの変換支援システムの開発が必要である。

（4）オブジェクト志向

ＪＤＭＦは基本的な概念の提示に止どまっている。次のステップとして２で述べたように今後表現の容易性、利用の容易性を追加概念の形で基本概念と調和させつつ導入する必要がある。特に複雑な対象を手直ししながら設計していくようなときには複合対象を扱うオブジェクト志向的なアプローチが有益である。

5．2　ＪＤＭＦ設計支援システム

ＪＤＭＦではデータモデル機能を述べたのに止どまり、具体的にどのような設計手続きで応用データモデルを設計していくかの指針は述べられていない。

5．3　ＩＳＯとの関係

現在ＩＳＯにはＪＤＭＦのようなプロジェクトはない。しかし近い将来何等かの形で同様な試みが開始されることは殆ど確実である。我国としてはとりあえずＪＤＭＦをＩＳＯの場に提案することを考えているが、ＪＤＭＦが直ちにベースドキュメントの形にならない場合でもＩＳＯ案とＪＤＭＦとの相違の解消に力を注ぐ。ＩＳＯ案にＪＤＭＦを凌駕する良い提案がなされたときはしかるべきタイミングでＪＤＭＦもＩＳＯにあわせて改定すべきである。

[参考文献]
日本工業規格（案）Ｖ１１　データモデル機能ＪＤＭＦ，情報資源スキーマ調査研究委員会，１９９０．４．２