データベース・システム 67－4
（１９８８．９．２２）

データベース設計支援システム
ＳＤＤＭ－ＣＡＤ

穂鷹良介，北原　匡（筑波大学）

著者等は長年に亙ってデータベースの論理設計を支援するシステム（ＳＤＤＭ－ＣＡＤ：Sentential Database Design System - Computer Aided Design）の開発に従事してきた。その特徴は方法論的には、

（１）データベースで表現したいと考える対象の利用者側からの記述

から始まって

（２）上記の記述のデータベース設計者による分析、精密化

を経て最終的に

（３）あらかじめ設定していたデータモデル*)に従ってのデータベーススキーマの作成

の三つの段階を明確に区別するものである。

*) データモデル型あるいは最近はデータモデル機能といわれているもの

本報告では、データベース設計方法論並びにその支援システムがどうあるべきかということについて、実際の使用、開発経験の結果、我々が現在までにたどりついた考え方、実現方法を紹介し、更に将来の発展の方向について触れる。

A Database Design Support System
SDDM-CAD

Ryosuke HOTAKA and Masashi KITAHARA

Institute of Socio-Economic Planning
University of Tsukuba

We have been involved in developing a logical database design method called SDDM-CAD (Sentential Database Design Method - Computer Aided Design) for a long time. The characteristic of the method is clear distinction of the following three stages:

(1) A description of the object world from the stand point of database users

(2) An analysis and elaboration of the above description by database designers

(3) Construction of a database schema under the predetermined data model*)

*) sometimes called data model type or recently called data modelling facility

In this report, we would like to explain our findings on how a logical databasedesign method and its support system should be designed based on our experience of usage and development of the system. Also, future directions of this work are discussed.

1　はじめに

著者は１９８０年にデータモデル並びにデータベース設計法について徹底的なサーベイを行なって一つのデータモデル（ＴＨデータモデル）を前提としたデータベース設計方法論ＳＤＤＭを［Hotaka1980］、［Hotaka1981a］、［Hotaka1981b］で提案した。引続き設計過程を支援するシステムＳＤＤＭ－ＣＡＤの開発が行なわれ、その内容の一部は［Hotaka1986］に紹介されている。［Hotaka1988］に方法論の説明、システムを使うためのマニュアルが用意されている。

本報告では、データベース設計方法論並びにその支援システムがどうあるべきかということについて、実際の使用、開発経験の結果、我々が現在までにたどりついた考え方、実現方法を紹介し、更に将来の発展の方向について触れる。

ＳＤＤＭはデータベースの設計過程を

（１）データベースで表現したいと考える対象の利用者側からの記述

から始まって

（２）上記の記述のデータベース設計者による分析、精密化

を経て最終的に

（３）あらかじめ設定していたデータモデル*)に従ってのデータベーススキーマの作成

*)　データモデル型あるいは最近はデータモデル機能といわれているもの

を行なう三つの手順として明確に捉え、その設計支援システムＳＤＤＭ－ＣＡＤは最終結果を得るまでになされる様々な設計作業をコンピュータプログラム等の手段によって支援するシステムである。

データベースの設計手順が上の三つの要素をもつことは、データベース設計方法論で必ずしも明確には意識されていないが、データベース設計の本質ではないかと筆者は考えている。

先づ（１）は設計の対象とその範囲を定めるものであるからその必要性は明らかである。

（１）で挙げられる情報要求は利用者がそのまま用いるような形で要求されるが、それは情報の大きなかたまりあるいは利用者向けに加工されたものであることが多い。これをそのままの形でデータベースに蓄積することも理論上は可能であるが、情報の共通利用を考えるときはそれら複数の情報要求に共通に含まれている事実を独立させて一個所で管理できるようにしなくてはいけない。したがって情報要求を分解して基本的な事実に分解する必要がある。これが（２）のステップで、化学でいえば物質の分子レベルあるいは原子レベルまでの分析に相当する。物質の正体を明らかにしようとするときそれを細かく分析することによって、物質間の共通性を見抜くことができる。情報のときに分子もしくは原子に相当するのは意味を失わない最小の「事実」で、ＳＤＤＭの場合それを「単文」としている。この分解のときに必要なことは情報要求を組立てるときにこの単文がどのように、組み合わされるべきかという情報である。（２）のステップでは、分解のときにこの情報も同時に記憶される。

このようにして得られた原子的な事実は非常に細かいだけに数が多過ぎて管理に不向きなので、最後に（３）のステップでもうすこし使いやすいように同種類のものをまとめることが必要となる。（２）のステップが分析もしくは解析に対応するとすれば、このステップは統合に対応する。

以下、この３段階の設計手順で我々が遭遇した問題、解決方法について述べる。

2　ＳＤＤＭ基本概念

［Hotaka1988］から体系的な説明なしにＳＤＤＭの基本概念を例で説明する。

２．１　情報要求

［例］

R1: 最後の定期点検後、各バスは何キロメータ走っているか？

月人別売上げ: 各セールスマンごとの月別の売り上げ集計を知りたい。

(注) 'R1', '月人別売上げ' は情報要求の識別名である。

２．２　文型

［文の例］

'山田' 君は '数学' で 'A' を得た。
'田中' 君は '数学' で 'B' を得た。
'田中' 君は '英語' で 'A' を得た。
'中山' 君は '英語' で 'C' を得た。
'山田' 君は '英語' で 'B' を得た。

［文型］

履修状態: '学生'x は '科目'y で '成績'z を得た.

［単文型でない文型］

F1: '部品'x は '重量'y で '住所'z に本社のある 'メーカ'u が提供している.

［上記の文型から得られる単文型］

F11: '部品'x は '重量'y である.
F12: '部品'x を 'メーカ'u が提供している.
F13: 'メーカ'u の本社は '住所'z である.

3　情報要求ステージ

ＳＤＤＭでは情報要求の提出の仕方については特に制約を設けていない。しかし出された情報要求は識別番号が与えられて識別される。情報要求間に重複があっても構わないが、相互に矛盾があってはいけない。

情報要求の出し方は、設計方法論によって以下のように色々の形態のものがある。

（１）対象となるシステムが造りだすと仮定されている出力帳票を用いる（［Tsubaki1987］）

（２）実体型（entity type）の列挙（［Chen1979］, ［Curtice1984］）

（３）利用者のビューの列挙（［Navathe1982］）

（４）データフロー ダイアグラムおよび処理要求からのデータの洗出し　（［Fong1985］）

実務上現れるデータベース設計は既存の何等かのシステムの再構築となる場合が多いが、その場合には現行システムの用いている帳票を情報要求とする（1）のアプローチは極めて説得力がある。勿論その際に不要な帳票を削除することもまた新規に新たな帳票を加えることも可能である。その反面全く新規のデータベースの設計時に十分安定した出力帳票を設計の早い段階で用意するのは難しいように思われる。

実体型を列挙させる（2）の方法はできれば申し分ないが、ではどうやって実体型の認識に至るのかということを別途説明してやる必要が残るように思われる。

（3）の方法は情報要求の提出者にあるデータモデルにしたがった方法で要求を出させるものでやはり利用者側にデータモデルに対する深い知識を前提とする。

（4）は企業等で実際にデータをどのように処理するであろうかという場面を想定することにより必要なデータを想起させる方法でシステムを実現する側の人間からみると具体的で分りやすい。

ＳＤＤＭでは極めてデータ中心に設計を進めるので始めから処理を想定した（4）のような情報要求の出し方は採らないのが普通である。というよりは情報要求が出されたところで、それをどのようにデータベースから作りだすかという処理を別途考えるステップを置くので設計作業が重複してしまう。

ＳＤＤＭでは情報要求並びに次の分析ステージを通じてデータ処理を直接考察の対象とすることはない。このステージの情報要求も次のステージの単文もどちらもデータの一種と考えることができるが、常に一つのデータが他のデータから導出されるかどうかということだけを考え、具体的にその導出のロジックを考えることをしない。つまりデータベース設計の前過程を通じてデータだけを中心に考えて処理に関しては思考を最小にしている。この方法によって設計を速かに進めることができるだけでなく、設計結果が特別の処理ロジックに依存する度合を低め、データの独立性を高めることができる。この方法の背後にはシステムの要所要所を押えるものはデータであるとする認識がある。

いま一つの情報要求をＲとし、それをより具体的な他の情報要求Ｒ１，Ｒ２に分解したとする。これを次のように表示し、ＲはＲ１とＲ２とに帰着されたという。

Ｒ<－Ｒ１．
　<－Ｒ２．

この意味は一度情報要求Ｒが出されたが、その内容を精密に検討したところ別のより具体的な情報要求Ｒ１、Ｒ２が満たされればよいということが分ったということである。情報要求ステージでは可能な限り帰着を続け、それ以上具体的な要求にするには具体的な文に帰着する以外はないというところまで手続きを進める。

現在の情報要求ステージは要求の述べ方について何等制約をつけないという自由がある反面、その提示の仕方が任意でかえって設計者に余計な判断を強いる結果となっている。上で述べたような特別な情報要求の提示の仕方をいくつか用意し、設計者に選択させるようにするほうがよいように思われる。

４　分析ステージ

４．１　設計結果の一意性

情報要求を文の形に帰着したところから分析ステージに入る。分析ステージは更にＲＧ、ＧＳ、ＳＳの３ステップからなる。ＳＤＤＭでは途中に一般的な文型を介在させているが目的は情報要求をいくつかの単文型に帰着させることで、単文型の目的は更にデータベース設計結果の一意性を高めることである。しばしばデータモデルで売物にする性質は、同じ対象の表現に対して如何に自然な表現を提供できるかということのようである（[Hammer1981]，[Chen1976]）。これに対してＳＤＤＭでは対象が同じならばできるだけ同じ表現になるように努める。

データベースの設計を行なうときに異なる二つの立場があって、これを区別するのが有益と思われる。一つは各利用者の立場からデータベースはこのように使えてほしいという要望を述べる立場で、もう一つは大局的な立場から個々の利用者の要望だけではなくデータベースの本来の目的である共有性を第一に尊重する立場である。データベース設計に当たっては個々の利用者のいわば局所的な要望をそのまま認めるわけにはいかない。共有データに対して局所的な立場ごとに異なる設計結果をもたらす可能性があるからである。

ＳＤＤＭでは従って先づ大局的な設計方針でデータベースの設計を完成させて、しかるのちに局所的な個々の利用者の要望をサブスキーマもしくは外部スキーマなどによっての満足させようとする。このことによって大局的なデータベースの設計結果をできるだけ客観的なものとしようとする。実際、情報要求ステージで集められた要求はある局面ある局面ごとの個々の利用者の要求であった。ある情報要求をＲとして、これが文型Ｆ１、Ｆ２、Ｆ３によって満足されるとする。つまり

Ｒ<－Ｆ１
　<－Ｆ２
　<－Ｆ３

であるときには後日プログラムを開発してＦ１、Ｆ２、Ｆ３のデータからＲに応える情報を作りだすことができる。したがって分析ステージにおいては大局的な設計結果になることを第一に考えて分析を行なうことにしても局所的な要求をないがしろにしたことにはならないと思われる。

大局的な設計を行なうために設計結果を一意にする条件についてであるが、一般に制約条件を付ければ付けるだけ表現の自由が減り一意性が増す。このときできるだけ自然で設計結果を良くするような制約条件が見付かると都合がよい。

単文型に要求される条件としてはＳＤＤＭを当初提唱したときには

（1）one fact in one place の原則を満たすもの

という性質しか分っていなかったが、その後以下のような条件がつけ加えられた。

（2）単文型にはキーが存在すること
（3）単文型の各単文の属性値は必ず値を持つこと
（4）単文型に属する文の属性値は常に単数値であること
（5）単文型の各属性は純粋であること

（1）の条件は関係データベースでよく知られた第３正規形またはボイス コッドの条件である。（2）のキーの存在は単文型がどの実体型と対応するかを明確にすることによって設計者に自分の表現がどのような意味をもつかを自分自身明確に意識するために、またキーを指定することによって単文型の意味制約の表現に役立つ。（3）は［Smith1979］が提唱した（3，3）正規形と趣旨は同じでしかも属性値にナル値を許さぬことにより一意性を高める。ナル値を許す設計では多くの種類の設計結果が可能である。（4）はやはりコッドが関係データベースで唱えた第１正規形の条件である。第１正規形は従来はデータベース設計の指針としてはあまり注目されていなかったのではないかと思うが、非第１正規形がリピーティンググループを許すモデリングで、リピーティンググループが階層構造と同じであることを考えれば、階層構造の局所性を持ちこませぬためには、この制約が非常に大切である。（5）については付録で説明する。

４．２　定義域、nested aggregation

１つの文型を考える。その文型の属性の属性値は色々の値を取るが、その値の集合を代表する実体型を考え、それを定義域と呼ぶ。

［例１］

F12: '部品'x を 'メーカ'y が提供している.

において、x が取りうる値の集合を代表する実体型は属性 '部品' の定義域である。

通常は、属性と定義域には同じ名前を与えて混乱はないので、設計者が特に断らない限り、ＳＤＤＭは属性の定義域名は属性名と一致しているものとして扱う。

［例２］

F10: '部品構成'('組立部品'x は'構成部品'y から成る)z
がある.

この［例２］で '組立部品'、'構成部品' の定義域が共に同じ '部品' だと設計者が考えたとすると、別途定義域名を与える必要がある。定義域名△△△は

'○○○'<△△△>

のように文型に現れる属性○○○の直後に<･･･>で括って示す。

［例３］

F10-1: '部品構成'('組立部品'<部品>x は '構成部品'<部品>y から成る)z がある.

ここで'部品' は二つの属性の共通の定義域である。

いくつかの属性をまとめて一つの属性と考えたとき、まとめてできる属性をＳＤＤＭではブロック属性という。たとえばＣＯＢＯＬあるいはＰＬ／Ｉでいうところの構造に対応する。

［例４］

ＣＯＢＯＬでいえば

```
02 A
  03 B
  03 C
02 D
```

のようなデータ定義において属性'A'はブロック属性である。上記の構造を持つ属性'A'を定義するには次のように属性'A'の定義域をあらかじめ定義しておく。

A: ('B'x ･･･'C'y).

(注) ここでは属性名と定義域名は一致している。両者が異なるときについては下の例７を参照のこと。

上記のように定義したとき、Ａは定義域名としてデータベース全域で一意でなくてはならない。Ｂ、Ｃ等はブロック定義域の構成を示すもので、このブロック定義内だけで一意であれば良い。

［例５］

部品提供：('部品'x を 'メーカ'u が提供すること).
F12: '部品提供'x がなされる.

部品提供というブロック定義域が '部品' と 'メーカ' の両方の列の結合を実現値として持つ定義域として定義され F12 でブロック属性 '部品提供' が使用されている。

５　統合

５．１　汎化

(註) 汎化の概念を正確に述べるだけの十分な紙数の余裕がないので以下の説明はやや不完全である。

５．１．１　キー実体型の汎化

s,t をあるファイルのキーの定義域となる実体型とする。s は t のキー以外のすべての属性を継承し、また s の実現値と等しいキー値をもつ実現値が t の中に存在するとき s は t の部分型である、あるいは t は s を汎化した実体型である等と呼び

s isa t あるいは s <-･- t

と書く。以下汎化を例題を用いて説明する。

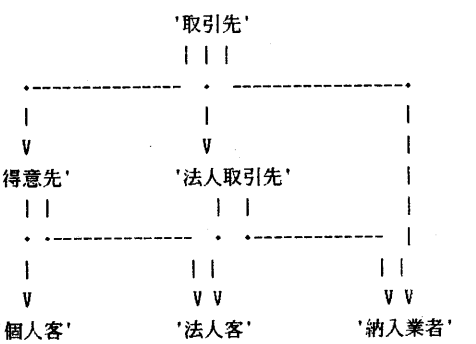

図５．１　汎化関係

上例では '納入業者' isa '取引先' という関係は、'納入業者' isa '法人取引先', '法人取引先' isa '取引先' という別の汎化関係から結論できるから冗長である)。ＳＤＤＭ－ＣＡＤでは汎化関係をＧＥＮステップで指定するが、そこではｔ isa ｔ のような自明な関係あるいは上記のように他の汎化関係から結論できる関係は指定しない。

また汎化関係をたどって行ったときにある実体型が自分自身の部分型となるサイクルを生じるような指定は冗長もしくは矛盾した実体型を定義していると考えられるからＳＤＤＭ－ＣＡＤでは警告が出される（複数キーの場合などにこのようなことがあり得る)。

５．１．２　汎化関係と専化ファイル

以上は実体型同士の汎化関係を見たものである。データベースで更にこれがファイルとの関係でどのように利用されるかを見よう。上記の実体型がそれぞれキーの定義域となっているファイルを考える。各々に適当な非キー属性を補ってデータモデル間の関係を再度表示すると図５．２のようになる。

図５．１の実体型をキーの定義域とするファイルの実例を図５．３に掲げる。

部分型に対応するファイルを専化ファイルと呼ぶことにする。キー属性は継承されないことにしているが、これはＳＤＤＭではファイルとキーとは１対１に対応するように簡単化しているので、ファイルが異なればキーの名前も変っている。

得意先	得意先担当者	得意先担当者電話	住所
a	加藤	123-4567	大坂
b	徳川	456-7890	東京

図５．３　部分型に対応する専化ファイル

汎化を行うときこのように部分型の数だけファイルが現れ、それらの間の isa 関係が別に記憶される。これらを物理上も別々のファイルとして実現するかあるいは他の実現方法を取るかは、物理設計の問題である。

同様に'個人客' は '得意先' と '取引先' の部分型であるので、専化ファイル '個人客' には '得意先担当者','得意先担当者電話','住所' などの非キー属性が継承されて存在する。

専化と汎化とを比較すると汎化は異なる実体を観点をかえて同一のものと見る考え方であり、ものごとを統一して見るときに便利である。その結果部分型の持っていた細かい意味の差は失われてしまう。これに対して専化は区別のされていない概念を細分化し意味を明確にするのに役立つ。

［例１］

従業員は社長を除いて何れかの管理職の下に配属されており、かつどの管理職も同時に従業員であるということを示すには以下のように考えることができる（図５．４）。

データベース設計において専化ファイルを定義するときには非キー属性がその汎型に対応する汎化ファイルから継承されるから、全部の属性について定義する必要はなく汎化ファイルとの差だけを指定すればよく概念の説明の経済化に役立つ。

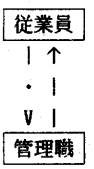

図５．４　専化による意味の明確化

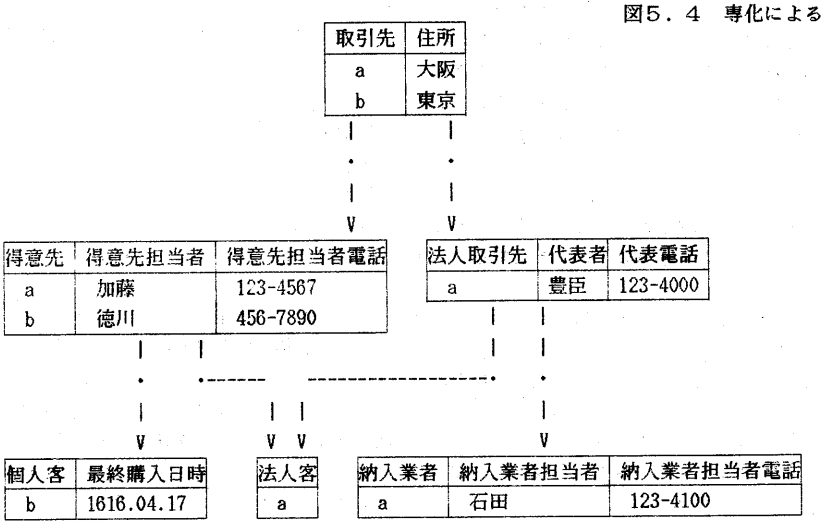

図５．２　ファイルの汎化関係

５．１．３　上位汎型の存在と名前の問題

一般に二つの 実体型 s,t が与えられたとしてもそれら両者の汎型が存在するとは限らない。　しかし、ＳＤＤＭにおいては、もしも一つの実体型 v にたいして

v isa s, v isa t

となる実体型 s, t が存在するときは

s isa u, t isa u

となる両者の汎型 u を同時に定義するものとする（ここで u は s または t の一方と一致していてもよい）。

ＳＤＤＭでは更に任意の実体型 t に対してその実現値を一意に識別する名前が存在することを仮定する。

以上の二つを仮定しておくことにより、複数個の実体型の実現値となっている実体を一意な名前で参照することができることになる。概念的に汎化を考えることは常に可能であるが、上記の条件を実際に満たすことができるかどうかは、データベースを導入する組織の実務上の体制に依存する（実務で遭遇する名寄せの難しさを考えよ）。

［例］

実体型 '法人客' を '得意先', '法人取引先' の部分型として定義するときは '得意先', '法人取引先' の汎型として '取引先' を同時に定義しておく。これにより '取引先' の実体を識別する名前で '法人客', '得意先', '法人取引先' すべての実体を識別することができる。

５．２　ファイル化

ＳＤＤＭでは情報要求を次々に分解して行って単文型を得る。設計の最終段階として、これらの単文型で実際にファイルとして残すことが指定されたものを（この指定はＫＥＥＰステップで行う）次のようにファイルとしてまとめ上げる。

(注) ファイルと文型とは厳密には異なる概念であるが、ＳＤＤＭでは簡単のためにそれらが１対１に対応することを仮定する。よってファイルは対応する文型を定めることにより決定されると考える。この文型をファイル文型と呼んでおく。

（１）　ＳＳステップで列挙された単文型のうちＫＥＥＰステップで指定されたものだけを残し、他の単文型をファイル化の対象からはずす。

（２）　上で残った単文型をキー名の等しいもの同士をグループ化する。

（３）　各々のグループごとに、ファイル文型を以下のように作る。

(a) ファイル文型名とキー名を同じにする。

(b) グループの単文型に現れる非キー属性を全部、非キー属性としてファイル文型に加える。

(c) 非キー属性で同じ名前のものが重複して出現したときには冗長なものを取り去る。

(注) 新しく得られる文型は最早単文型であるとは限らない。

［例１］

F1: 'A' 'B'.
F2: 'B' 'C' 'D'.
F3: 'A' 'E'.
F4: 'B' 'C' 'E'.

という単文型が定義されているものとする。ファイル化の結果は以下のようになる。

A: 'A' 'B' 'E'.
B: 'B' 'C' 'D' 'E'.

ファイル化された結果は、次のファイル文型を持つシステム ファイルにまとめられる。必要に応じてこの文型を利用者の文型で用いることができる。

ET: 実体型 'ET'x は 'ENTITY-TYPE-NAME'y の名を有し、'COMMENT'z を有する。

LT: 列実体型 'LT'x は属性の列である.

F: ファイル 'F'x は属性 'KEY'z をキーとして持つ.

A: 属性 'A'x は 'ATTR-NAME'y の名を持ち列実体型 'LT'z の定義に使われ、定義域は'D'u 'COMMENT'v である.

D: 定義域 'D'x はフィールド長 'LEN'y フィールドデータ型 'DATA-TYPE'z である.

BD: 複合定義域 'BlockDomain'x はブロック属性の定義域である.

ISA: 'ISA'('SUBTYPE'<ET>x は 'GENTYPE'<ET>y) の部分型である.

LT isa ET.
D isa ET.
F isa LT.
BD isa LT.
BD isa D.

ＳＤＤＭでは対象とするデータベースがどのようなものであっても、最終的な設計結果は上記のスキーマをもつＳＤＤＭ内の（メタ）データベースに蓄積される。この意味で上記のスキーマは分析に使うデータモデルの型（データモデル機能と呼ぶこともある）を決定していることになる。

ＳＤＤＭで用いているデータモデル機能はＴＨエンティティモデルと呼ばれているものである。

ＳＤＤＭがその設計結果を表現するのは大局的なデータベース設計を強く意識した表現の一意性を重要視するＴＨエンティティモデルであるが、情報要求段階では特別なデータモデルを仮定していない。情報要求を洗いだすには利用者の協力が必要で、そのときに表現の仕方に制約のある厳格なデータモデル機能を義務付ける必要はない。表現力の大きいデータモデル機能を利用するほうが有利である。しかしそのようなデータモデル機能で表現された結果をそのままにしておくと局所的な表現の欠陥が残ってしまう。

6　支援システムの実装

ＳＤＤＭ－ＣＡＤは、ＩＢＭシステム／３８上に３回にわたってインプリメントされた。最初は実現の労力を減らすためになるべく一般的な支援ｔｏｏｌを作ったが、設計者の発送を容易にするためには、個々の場面ごとに特殊処理をする必要があるということを学んだ。

現在は設計を合計１４ステップに分け夫々にメニュー画面を用意する形となっている。記憶に残る問題点は次の通りである。

（１）テキストを直接設計者が更新することがある。そのためユーザがどのような操作をしたかということをシステムは知る必要があり、テキストエディタ類似のものをつくることになった。

（２）設計作業がある程度進んだ段階で設計者が誤りに気付き一度行なった設計をやり直したときに、蓄積されている途中の結果をどのように活かしどのようにキャンセルするかが難しかった。この問題は完全に解決されてはいない。

（３）開発が長期に亙ってなされたために、他の魅力的なハードウェアが登場し、開発済みのユーザインタフェースを陳腐化してしまった。

初期化ルーチン 1309 ｽﾃｯﾌﾟ, 更新作業 7262 ｽﾃｯﾌﾟ, 後始末ルーチン 2293 ｽﾃｯﾌﾟ, 制御部分 1022 ｽﾃｯﾌﾟ ,言語は最後はRPGIIIであった。

7　将来の方向

設計作業の入力情報、出力情報は他の設計方法論あるいは開発方法論とリンクしなければいけない。それを行なうにはインタフェース部分に再びデータベースがなくてはいけないということが分った。今後は情報資源辞書の観点からこれらの問題を統一的に再編成していく。

付録　純粋属性

属性の選びようによってはデータベースの設計が一意にならないことがある。付図１はそのような例である。

図の(b)、(c)の文型は次のように考えると(a)のように変形できる。たとえば図の(b)の '1期売上げ' という属性は '期' という属性の属性値が 1 であるときの '売上げ'属性という意味を持っている。図の(c)の 'Ａの売上げ' という属性は 'セールスマン' という属性の属性値が 'A' であるときの '売上げ'属性という意味を持っている。

この事実に着目し、次のように表現を分解する。

'1期売上げ'=>'期' 1、'売上げ'
'2期売上げ'=>'期' 2、'売上げ'
'3期売上げ'=>'期' 3、'売上げ'
'4期売上げ'=>'期 '4、'売上げ'

'Aのうりあげ'=>'セールスマン' 'A'、'売上げ'
'Bのうりあげ'=>'セールスマン' 'B'、'売上げ'

つまり、属性の中に紛れ込んでいた二つの属性をそれぞれ独立させるようにすると、(b)、(c)からそれぞれ '期'、'セールスマン' という属性が新たに析出され、(b)の '１期売上げ', '２期売上げ', '３期売上げ', '４期売上げ' の４属性と(c)の 'Ａの売上げ', 'Ｂの売上げ' の２属性は上記のように分解されて、それぞれ二つずつの属性によって(a)のように表現されることになる。

このような変形を属性の純粋化といっている。一つの属性に２個の属性が混在しているときにはそれを分離するというように、あらかじめ定めておくことによって、等価な表現が無規律に複数個現れるのを制限する。ＳＤＤＭは大局ステージのデータモデルを設計するのが目的であるから、できるだけ設計結果が一意になるように、単文型を考えるときには属性を純粋化することとする。関係データベースの正規化理論とは異なった別の正規形といってよい。

純粋化はまた属性の定義域の実体型を汎化していることに対応する。汎化は実体型の数を減らす概念操作であるため純粋化によって、用いる概念の数を減らすことができたのである。

参考文献

[Chen1976] P.P.S.Chen: The entity-relationship model: Toward a unified view of data, ACM TODS, Vol.1, No.1, pp.9-36.

[Chen1979] P.P.S.Chen: The Entity-Relationship Approach to Logical Data Base Design, QED Monograph Series, No.6, 1979

[Curtice1984] R.M.Curtice: An Automated Logical Data Base Design and StructuredAnalysis Tool, A quarterly bulletin of the IEEE computer siciety technical committee on Database Engineering December 1984 Vol.7 No.4, pp.40-45.

[Fong1985] E.N.Fong, M.W.Henderson, D.K.Jefferson and J.M.Sullivan: Guide on Logical Database Design, NBS Special Publication 500-122, 1985

[Hammer1981] M.Hammer and D.McLeod: Database Description with SDM: A Semantic Database Model, ACM TODS, Vol.6, No.3, September 1981, pp.351-386.

[Hotaka1980] 穂鷹良介：文章表現を用いたデータベースの論理設計法ＳＤＤＭ，データベース管理システム研究会報告23-3,情報処理学会1980.1.16

[Hotaka1981b] 穂鷹良介： データベースの論理設計，情報処理叢書 6，情報処理学会1981

[Hotaka1981a] R.Hotaka and M.Tsubaki: Sentential Database Design,「アドバンスト・データベース・シンポジウム，1981年12月，情報処理学会，pp.53-60.

[Hotaka1986] 穂鷹良介：データ・モデルとデータベースの論理設計，日経データプロ・ソフト，日経マグロウヒル社，1986.2

[Hotaka1988] 穂鷹良介：データベースの論理設計法ＳＤＤＭ－ＣＡＤ説明書，第３．２版，1988.7.28

[Navathe1982] S.B.Navathe and S.G.Gadgil: A Methodology for View Integration in Logical Database Design, VLDB 1982, PP.142-164.

[Smith1977] J.M.Smith and D.C.P.Smith: Database Abstractions: Aggregation and Generalization, ACM TODS, Vol.2, No.2, 1977, pp.105-133.

[Smith1979] J.M.Smith: A Normal Form for Abstract Syntax, VLDB 1979, pp.156-162.

[Tsubaki1987] 椿正明：システムのデータ統合を図るデータ分析方法論，ＰＬＡＮ－ＤＢ，アプリケーション・デザイン NIKKEI COMPUTER 別冊 1987.6.22 pp.278-284

売上げ統計: 'セールスマン'x は'期'y に'売上高'z の売上げを記録した.

'A'	1	100
'A'	2	120
'A'	3	135
'A'	4	150
'B'	3	90
'B'	4	104

(a)

ｾｰﾙｽﾏﾝ売上げ:
'ｾｰﾙｽﾏﾝ'xは'1期売上げ'y1,'2期売上げ'y2,'3期売上げ'y3,'4期売上げ'y4を記録した.

'A'	100	120	135	150
'B'	-	-	90	104

(b)

期別売上げ: '期'x は 'Aの売上げ' がy1, 'Bの売上げ' がy2 であった.

1	100	-
2	120	-
3	135	90
4	150	104

(c)

付図１　等価な文型