データベース・システム 34－8
（１９８３．２．１７）

２項データモデルへのデータ入力言語ＮＩＬ

穂鷹　良介（筑波大学）　　蒋　新児（南京大学）

１．目的

筆者等は，ここ２、３年にわたってデータベースの論理設計法並びにそれらの支援システムの開発を行なって来ている（［１］，［２］）。それらの研究の結果，

・データベースの論理設計は，世にいう発想法と方法論的に似通っている

ことが分った。そのため，我々は単にデータベースの論理設計法のためだけでなく一般に発想法を支援するためのシステムＮＳＭ（version １　release １）を開発した（［４］）。その内容については本論文でも次節で簡単に触れるが，利用者があまり組織的ではない発想を行なってもその情報の蓄積に困らないように，柔軟性に特徴のある２項データモデルを用いている。２項データモデルは同様の目的のために例えば［３］，［５］等で用いられている。

［４］で用いたデータ入力の方法は非常に原始的で，モデルの最小要素を一つ一つ入力するレベルのもので，データベースのＤＤＬとして用いるには手数がかかり過ぎ，記述相互間の整合性を保つのも難しい。本稿ではそれに代って関連情報をまとめて入力するための言語ＮＩＬ（ ＮＳＭ Ｉnput　Ｌanguage ）を提案する。ＮＩＬは一般的なある制約を有してはいるが，その制約内で言語を色々に変化させることができる。直接の開発目的としてはデータベースの論理設計への応用があるが，他への応用も考えられよう。

２．２項データモデル

このモデルの構成要素は節（node）と弧（arc ）である。arc は２個の node （一方をcentral node，他方を target node と呼ぶことにする）を結ぶ。 node も arcも必ずある node type と arc type とにそれぞれ属する。arc は有方向（directed）と無方向（undirected）のものがあり，central nodeからtarget nodeに向う向きを正方向と言う。無方向のarc は，それが結ぶ２つのnodeのいずれからも他方に向う向きが正方向である。

ここで後の説明の便宜のために若干の記法を導入しておく。以下¢'A'，¢'x'，¢'x12'など文字列を' 'でくくったものの前に¢をつけたものは，個々のnode，node type， arc ，arc typeを表す。これらから' 'を取去ったものはこれら主体を値として持つ変数を表す。'A'，'x'，'x12'など文字列を' 'でくくったものはここのnode，node type， arc ，arc typeの名前を表す。これらから' 'を取去ったものはこれらの名前を値として持つ変数を表す（図１参照）。

	定数	変数
主体	¢'A'，¢'x'，¢'x12'	¢Ａ，¢ｘ，¢x12
名前	'A'，'x'，'x12'	Ａ，ｘ，x12

図１　　記　　法

nodeとarc を図に表せば図２のようになろう。

図２では node type ¢t1 に属するnode ¢s1 をcentral node，node type ¢t2に属するnode ¢s2 をtarget nodeとするarc type ¢a に属するarc ¢b が描かれている。これを形式的に

arc （¢s1≪¢t1，¢b ≪¢a ，¢s2≪¢t2）

と表現することにする。

上ではnode type，node，arc type，arc のいずれも変数表現を採っているが，この一部に定数

を代入してたとえば図３のように書くことも行う．

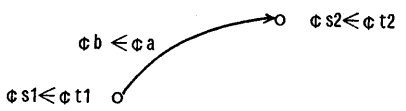


図２　arc のグラフ表現

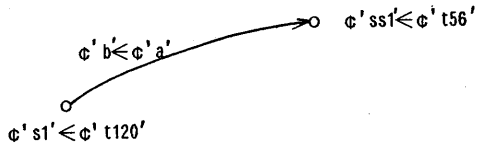


図３　arc（¢'s1'≤¢'t120'，¢'b'≤¢'a'，¢'ss1'≤¢'t56'）

arc type，node typeなどが文脈で明白である場合あるいはそれらを特に明示しなくても良い場合には説明を簡略化して図２と同じことを図４のように書く。またarc ではなくarc typeだけを明確にしたいときには図３と同じことを図５のように書く。

arc（¢s1，¢b，¢s2）が成立するとき，arc ¢b の方向を逆に考えたarc ¢b1を考え，同じ事実がarc（¢s2，¢b1，¢s1）で表わされているとする（図６）。

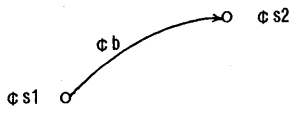


図４ arc（¢s1，¢b，¢s2）

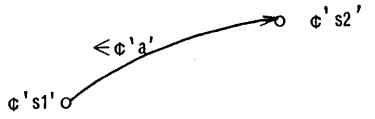


図５ arc（¢'s1'，≤¢'a'，¢'ss1'）

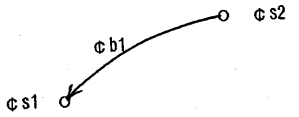


図６　arc（¢s2，¢b1，¢s1）

このような　¢b1　は¢b と全く逆のarc であるから－¢b と表現することも許す（図７）。

無方向のarc の場合には，arc の両端のnodeに向かって矢印がつけられ図８のようになる。

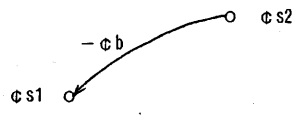


図７　arc（¢s2，－¢b，¢s1）

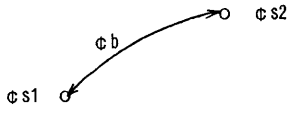


図８　arc（¢s1，¢b，¢s2）またはarc（¢s2，¢b，¢s1）

ＮＳＭの基本機能はデータの入力だけに関して述べると以下の通りである。

・適当なnode　type¢t を生成もしくは削除する。

・適当なarc type¢a を生成もしくは削除する。

・与えられたnode　type¢t に属するnode¢s を生成し，あるいはこれを削除する。

・central node¢s1，target　node¢s2が与えられたとき，arc type¢a に属するarc ¢b を生成し，¢s1，¢s2を結んだり，あるいはこれを削除したりする。

以上ではnodeとかarc の主体の識別については何等問題がないものと仮定している。しかし実際にはこれらの主体を適当な名前で置き換えなくてはならない。名前は複数の主体に対応し得るからあいまいさのない形でこれを行うには若干の注意が必要である。次節以降でこの点に関してより詳細に検討をする。

３．データ入力言語への要求

２項データモデルへのデータ入力ということは，例示すると図９のような情報を名前を用いてデータベース内に投入することである。

このおのおののnodeとarc を１個１個入力するのではなく，グラフが連結しているならば，どこか適当なnodeから出発してそれに連結しているすべてのnodeとarc とを一つのまとまった表現として与えてしまうようなことを考えたい。

たとえば図９のようなデータをnode n1 から出発して図１０のような木状のデータとして一気に書き下すことができたら有難い。

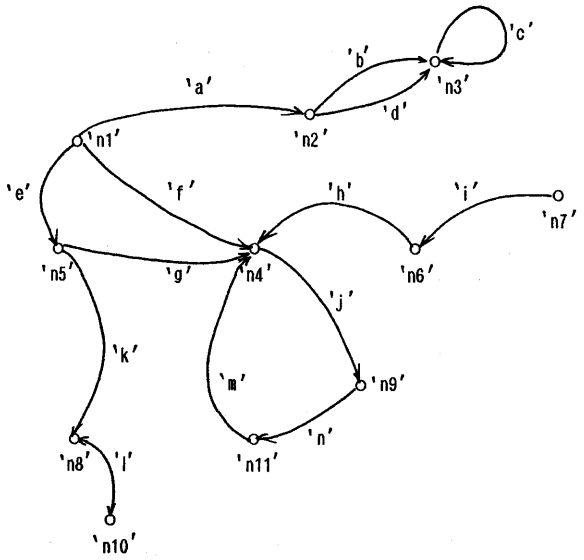


図9　2項データベースのオカーレンス例

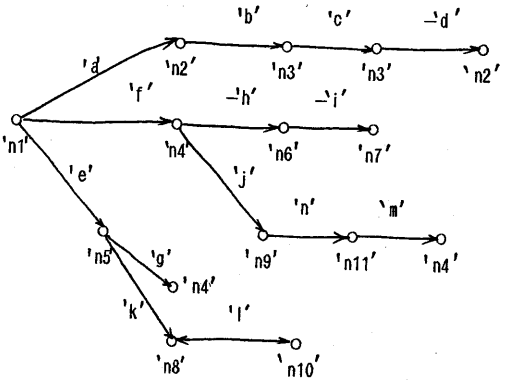


図10　図9を木状にひきのばしたもの

実現の可能性はともかくとして，利用者としては以下のような要求がデータ入力言語ＮＩＬに対して存在する。

Ｒ１：ＮＩＬの記述は表現すべきデータベースのデータが与えられたとして一意であってほしい。

［理由］　ＮＩＬ記述が一意でなければ，複数の設計者の記述が互いに異なる可能性を持ち，検証に困難を来たす。また将来データベースの内容をＮＩＬ記述に変換して戻すとき，入力時の記述と異なるのは好ましくない。

Ｒ２：ＮＩＬは理解し易いものであってほしい。

Ｒ３：ＮＩＬのプロセサは，いわばスクリーン　データベース　エディタであってほしい。

［理由］　スクリーン　エディタは従来型のテキスト　エディタに比し，抜群に使い易い。データベースといえども一種のテキストのように考えば同じインタフェースを考えることができるのではないか。

Ｒ４：コマンドの数はできるだけすくなくしたい。

Ｒ５：node名に文脈依存名を与えたい。

「理由」　たとえばファイル設計を行っているとき，フィールド名は，全域で一意でなくてもそのファイル内で一意であれば十分である。この機能により，node全域での名前の使われ方をチェックしなくてもある程度の並行データ入力が可能となる。

Ｒ６：nodeの文脈依存名と別にnodeを常に一意に識別する一意識別名を維持すべきである。

［理由］　nodeを文脈依存名でしか識別できないとき，常にその局所名の使われた文脈を意識しなくてはならなくなり，後で全体を管理するときに不便である。

Ｒ７：arc の方向は正方向，負方向いずれを利用してのデータ投入もできて欲しい。

Ｒ８：データ投入はどのnodeからＮＩＬで書いても良いようにしたい。

以上のような要求をすべて満すような言語は存在しない。要求同士がどのように矛盾し合って排除され，また残った要求がどのように言語の形を決めて行くかを次節に述べる。

４．ＮＩＬ言語仕様

４．１．要求の検討（arc が有方向の場合）

最初にarc が有方向のものばかりであるケースを考察する。無方向のarc については特別な扱いが必要となるためである。

我々はすでに個々のnode，arc を別々に入力するシステムを有している（［４］）。その入力方法を改良するための方法として考えるからには，１回の入力指定でなるべく多くの関連情報を指定するものでなくてはならない。今２つの arc

（¢x，≪¢a，¢y1）　　（１）

（¢x，≪¢a，¢y2）　　（２）

があったとき，共通記述を括り出して

（¢x，≪¢a，｛¢y1，¢y2｝）　　（３）

のような形にして記述すると¢x，¢a を２回繰り返し書く必要がなく簡便である。これはnode ¢x を根とする木として（１），（２）の情報を一度に表現したことに相当する。

この種の方法としてデータベースの分野でも良く知られている情報の表現方法は，データ間の関連を木構造と見るものである。図１０で図９の情報をnode n1 を根とする木として表現しているのはこの例である。

よって，Ｒ２に応えるＮＩＬの仕様（Ｓpecification）として以下を採用しよう。

Ｓ１：ＮＩＬではarc を木の枝としてデータを表現する。

要求Ｒ１，Ｒ７，Ｒ８は互いに相容れない。すでに見た図４と図７とは同じ情報の２種類の異なった表現である，つまりＲ１に矛盾している。Ｒ１の要求をＲ７，Ｒ８よりも優先させるとＲ７かＲ８かいずれかの要求は認め難い。 arc の方向の正，負は人為的なものであるが，一応人間には正方向が分り易い方向であると考えて，正負両方向をデータ入力に利用することはせず，

Ｓ２：木状にデータを投入して行くときは，arc の正方向に沿って行くこととする。

更に図１１のような情報の表現を考えると，方向を正の方向に限定したとしても図１２のようなすくなくとも３通りの表現がある。よって，Ｒ８の要求もＲ１と相容れない。従って

Ｓ３：木表現の根nodeとなり得るnodeが所属するnode typeをあらかじめ限定する。

根nodeとなり得るnodeが所属するnode typeをindependent node type，それに属するnodeをindependent nodeという。

図１１で仮にnode¢'n1'，¢'n2'がindependent nodeだとしても，木構造の表現が途中のnode（葉でないnode）に independent node が来ることを許すと，依然として図１２の（ａ），（ｂ）は可能であるから次の制約を置く

Ｓ４：木表現でindependent nodeが現れたとき，それは根あるいは葉nodeでなければならない。いいかえるならば，independent nodeが根node以外のnodeとして現れたならば，それは葉nodeである。

図１１で，¢'n1'，¢'n2'がindependent nodeであるとすると，この木表現は図１３のように２個の木表現から成る。

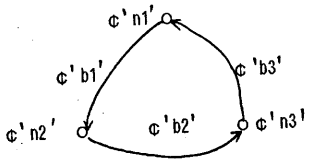


図１１　cycle を有する情報

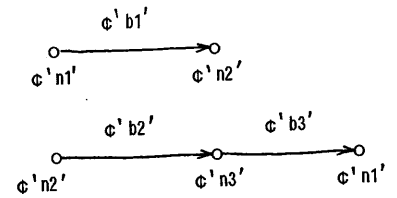


図１３　図１１の木表現

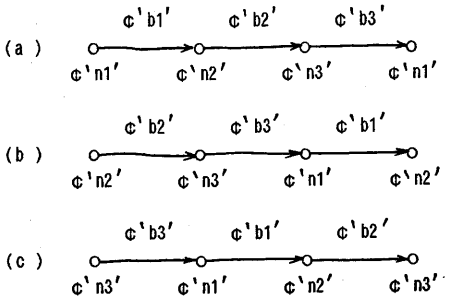


図１２　等価であるが異なる木表現

つまり，連結グラフを１個の木表現で入力することは上述の要求の下では一般的にはできない。

要求６のnodeの一意識別名としては，（node type名Ｔがnode typeを一意に識別すると仮定して），各node typeごとにそれに所属するnodeに通番を振り，node type 名とこの通番の連結名を用いる。この一意識別名をidentifer と呼ぶ。

Ｓ５：あるnode¢ｓ のidentifierは，そのnodeの属するnode type名をＴ，そのnode type内での通番をＮとしたとき Ｔ｜Ｎである。（｜は連結を表示する）

例 node type名 Ｆ のnodeのidentifier例 Ｆ｜１，Ｆ｜５２

identifierに対しては

『すべてのnode¢ｓ に対してあるidentifier ＩＤが存在して

¢ｓ ＝id－node（ＩＤ） （４．１）

となる。ここでid－nodeはidentifier（名前）からnode（主体）へ写像する関数である。』

要求Ｒ５の文脈依存名として２種類のものを考える。そのための準備として次の３種の関数を導入する。

¢ｓ ＝global－node（¢ｔ，ＧＮ） （ここで¢ｓ ≪¢ｔ） （４．２）

¢ｔ ＝target－node－type（¢ａ） （ここで¢ａ はtarget node typeを一意に定める特別のarc type） （４．３）

¢ｓ2＝local －node（¢ｓ1，¢ｔ，ＬＮ）（ここで¢ｓ2≪¢ｔ） （４．４）

関数global－nodeは，node type¢ｔ，global name ＧＮが与えられたとき，そのnode type内のnode¢ｓ を決める。あるnode type¢ｔ の勝手なnode¢ｓ が与えられたとして，（４．２）を満たすglobal name ＧＮが常に存在するとは限らない。それはnode type¢ｔ に依存する。（４．２）を満たすＧＮが常に存在するとき，node type¢ｔ はglobalであるまたはglobal nameを持つという。global name ＧＮはnode type¢ｔ という文脈の下でnodeを識別する名前である。

関数target－node－typeは，arc type ¢ａ が与えられたときtarget node type ¢ｔ を定める。どのarc typeに対しても¢ｔ の値が定まる訳ではなく，arc に伴って指定するtarget nodeをidentifier以外の名前（具体的には上で述べたglobal nameと 以下に述べるlocal name）で指定するときだけである。つまりarc typeはそれに関連するtarget node typeを一意に定めるものとそうでないものがあり，一意に定めるときにその target node type を関数target－node－typeによって表す。target node typeを一意に定めないときには，target nodeはidentifierによって指定される。

arc typeがtarget node typeを一意に定めるとき，もしtarget nodeのglobal nameが最初から分っているときには（４．３）と（４．２）とからtarget nodeを結ぶarc のarc typeが分りさえすればtarget nodeを決定できる（図１４）。

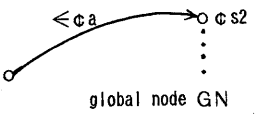


図１４ ¢s2＝global－node（target－node－type（¢ａ），ＧＮ）

実用上は，すべてのnode typeに対してglobal nodeの存在を仮定できなく，利用者はもうすこしせまい範囲（文脈）で通用する名前しか知らない場合がある。

関数local-nodeは，根node¢ｓ1，target node type¢ｔ，local name ＬＮからtarget node s2 を定める（図１５）。

実際には¢ｔ を図１６のように，target node¢ｓ2の直前のarc ¢ｂ の属するarc type ¢ａ から¢ｔ ＝target－node－type（¢ａ）によって決定し，しかる後にtarget node ¢ｓ2を¢ｓ2＝local-node（¢ｓ1，¢ｔ，ＬＮ）によって決定することも良く行う。

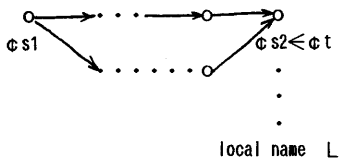


図15　¢s2＝local-node（¢s1, ¢t , LN）

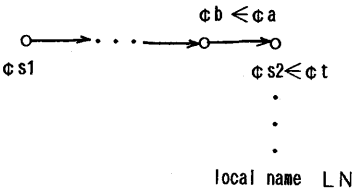


図16　local nameからlocal nodeを知る

target node¢t がこのようにlocal name LNによって決定されるとき，node type ¢t はlocal であるまたはlocal nameを持つという。

S6：どのnode typeもglobal nameを持つかlocal nameを持つかのいずれかである。

簡単のため，global nameとlocal nameの両者を同時に持つケースを排除した。

S7：（¢s1, LN1）≠（¢s2, LN2）ならば常に

local-node（¢s1, ¢t , LN1）≠local-node（¢s2, ¢t , LN2）

が成立する。

local nameによって呼ばれるnodeは異なる根nodeを持つ木状表現に同時に属することはないものとした。S6，S7は local name を認めたときにそれに伴って（簡単さのためもあるが）必要とされた制約である。

S8：根nodeは常にidentifierによって指定される。

根node type さえ判明していれば，global name からも根nodeを指定できるが，これは簡単化のための当分の間の制約である。

関連するtarget node の性質		
target node の指定を	taeget node typeを	
identifierによる	指定しない	local , globalいずれのnode type でも良い場合
identifer によらない	指定する	local node type だけを認める場合
		global name typeだけを認める場合

図17　arc typeと関連するnode typeの性質の分類

要求R3は，NILで書かれた入力データをスクリーン上に表示したもののデータベースへの反映とその逆にデータベースに蓄積されている内容をNILによって表現したものをスクリーン上に見せる機能に集約される。この機能により，一旦データベースに格納された内容を自由にスクリーンで変更することができる。NILによるデータ入力はindependent nodeごとに記述が分かれる。NIL editor は，この単位ごとにデータを入力し，逆にデータベース内のデータをこの単位でスクリーンに表示する。あるindependent nodeを与えたとき，それを根nodeとする記述の範囲（どのarc とどのnodeが一緒に入出力されるか）は上記の作り方によって確定するが，残念なことに，一つのnodeをcentral nodeとするarc が（したがってtarget node もが）複数個あったとき，その順番までは上の制約からは一意に定まらない。しかし実用上はこの程度の一意性の欠如はがまんできるものと予想している。

要求R4は，要求R3と関連がある。NIL editorは，スクリーン編集機能を用いて， データベースの内容を画面上で修正できるからそれだけでデータの挿入，更新，検索，削除ができることになる。つまり編集用コマンドはただ一つ「スクリーン上の内容をデータベースに反映させよ」だけである。以上でarc が有方向の場合のすべての要求の検討を終わった。

４．２．要求の検討（arc が無方向の場合）

arc が無方向の場合，図１８に示すようにnode¢s1，¢s2がどちらもindependent である場合，図１９に示すように異なったＮＩＬの表現が２個現れる可能性を持つ。

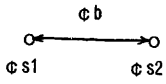


図１８　無方向のarc

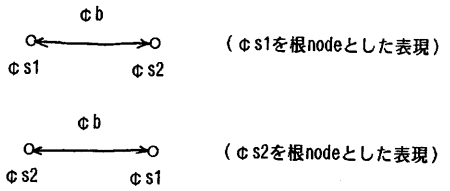


図１９　等価な２通りの表現

簡単のため以下の制約を置く

Ｓ９：arc ¢b が無方向で¢s1，¢s2がindependent nodeかつarc （¢s1，¢b ，¢s2）が成立した場合，ＮＩＬでは冗長ではあるが同時に¢s1を根node，¢s2を根 node にした記述を書くものとする。

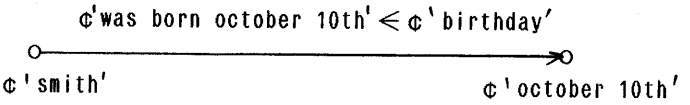
Ｓ１０：independent node間以外の無方向arc は認めない。

４．３．terminal　arc type

あるarc　type¢a によってはcentral nodeに絶対なり得ないtarget　nodeだけを持つ場合がある。このようなnodeはnodeとして扱うよりもむしろarc の付属物として扱うのが自然であることが分った。この種arc typeを，特別のterminal　arc type（terminal arc type ）として扱うことにしてimplementation時にその特質を活かすとことする。このときのcentral nodeはどのterminal arc と結び付いているかということだけを表現するだけで十分情報表現の目的を達している。

Ｓ１１：ＮＩＬに，terminal　arc　typeを設ける。

例．



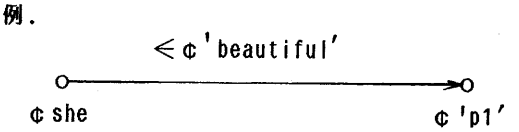
¢'birthday'はterminal arc type である。

４．４．property　arc type

あるnodeはある種のarc typeに属するarc のcentral nodeであるということだけを表現するだけで，十分情報表現の目的を達している場合がある。このようなarc　typeは，特別のproperty arc typeとして扱うのが適切である。

Ｓ１２：ＮＩＬにproperty　arc typeを設ける。

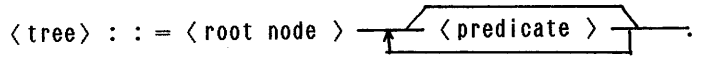
例．

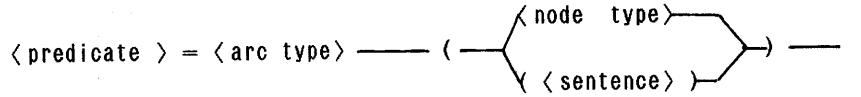


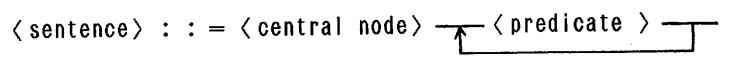
（注）内部的にはすべてのpropertyのtarget　nodeとして特別のnode¢ p1 を一つ設ける。

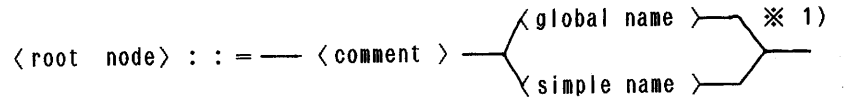
5．ＮＩＬシンタックス

１個の木状のデータ入力形式〈tree〉を示す。









〈comment 〉：：＝／＊・・・・・・＊／

〈node name 〉：：＝〈identifier〉｜〈global name 〉｜〈local name〉

〈central node〉：：＝〈node name 〉

４で述べた各種の制約がこれに加わるが詳細は省略する。

※１）すべての〈tree〉のentry は根nodeをidentifierによって指定された後に書かれる。したがって〈root node 〉ではroot node typeに更にglobal node が存在するときそれを書き，global nameが存在しないときには単なるメモ的な名前〈simple name 〉を書く。

6．終わりに

現在ＮＩＬプロセサの機能を用いてデータベースの論理設計法ＳＤＤＭ　の　ＤＤＬ定義並びにその機能評価，ＳＤＤＭ　ＣＡＤ構築のための試行実験を行なっている。

［　参考文献　］

［１］　穂鷹　良介：データベースの論理設計，情報処理学会，１９８１．

［２］　R．Hotaka and M．Tsubaki：Sentential　Database　Design　Method，Advanced　Database　System　Symposium，ＩＰＳＪ，Dec．１９８１．

［３］　S．Berild，S．Nachmans　and　ＣＡＤＩＳ　group：ＣＳ４－Ａ Tool　for　Database　Design　by　Infological　Simulation，ＶＬＤＢ，Part II，pp．８５－９４，１９７７．

［４］　中村　猛，荒木　誠司，穂鷹　良介：発想法のデータベースによる処理（ＮＳＭ）情報処理学会第２４回全国大会，pp．４５３－４５４，１９８２．

［５］　E．A．Hershey　，Y．Yamamoto，J．Kenning：The　Structure and Contents of a ＰＳＡ　Data　Base（Version　Ａ５．１），June １９７９，ＩＳＤＯＳ　Research Project，Department of Industrial　and　Operations　Engineering，The　University　of Michigan，Ann　Arbor，Michigan　４８１０９，ＩＳＤＯＳ ref．＃７９Ａ５１－０２７３－０．