数理解析研究所講究録
525巻 1984年 124-143

従属性を考慮したネットワークデータベースの設計

京大工学部　上林彌彦（Yahiko Kambayashi）
京大工学部　古川哲也（Tetsuya Furukawa）

1. まえがき

1970年のCoddによるデータベースの関係モデルの導入により、データベースの理論的研究が大きく進展した。1980年代に入ってからは、関係データベースの商用システムがIBMをはじめとする各社より発表され、従来のデータベースシステムでは応用の困難な分野にも利用されはじめている。[1] しかしながら、効率という面から考えると、従来のネットワークモデルによるデータベースは非常にすぐれており、関係モデルのみならず、ネットワークモデルや階層モデルによるデータベースも今後利用されてゆくであろうと考えられる。

ネットワークモデルによるデータベースの欠点の1つは、組織的な設計方法が存在しないということで、設計者の経験と直感にたよった設計が一般に行なわれている。従来の方法としては、「実体関連図を用いて1対1，1対多，多対多の対

応関係をもとに設計するもの」や「関数従属性を用いたもの」[2][3]が知られている程度で、種々の条件を複合したものは知られていなかった。ネットワークモデルはどのような意味制約を表わしているかは文献[4]で考察しているが、本稿ではその結果を用いて関係データベース理論の成果をどのようにネットワークデータベースの設計に反映できるかについて検討する。

2. 基本的事項

ネットワークモデルにおけるレコード型は、同じ属性集合からなるレコードの集合である。属性はCODASYLではデータ項目と呼ばれている。あるレコード型（親レコード型）の1つのレコード（親レコード）に別のレコード型（子レコード型）の任意個数のレコード（子レコード）を関連づけて全体に名前をつけたものを親子集合といい、親レコード型のすべてのレコードに対して作られる。親子集合は図1(a)に示す多重リンクで表現される。$r_1, r_2, \cdots$ は親レコード型Rのレコード、$q_1, q_2, \cdots$ は子レコード型Qのレコードである。親子集合を個々

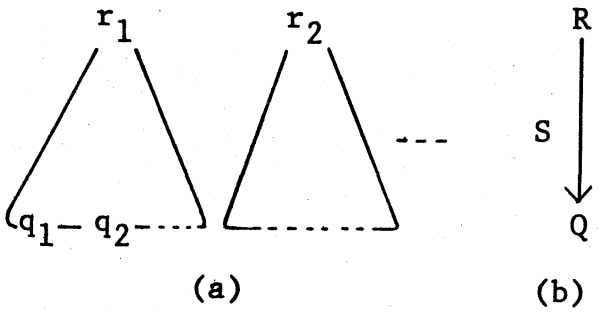

図1　親子集合と親子集合型

のレコードではなくレコード型の関係として記述したものを親子集合型といい、S〈R, Q〉で表わす。Sは親子集合型の名、R,Qはそれぞれ親レコード型、子レコード型である。

ネットワークモデルの構造は、バックマン線図と呼ばれる有向グラフ（N, E）で表わされる。Nは各レコード型に対応する節点集合、Eは各親子集合型に対応し、親レコード型に対応する節点から子レコード型に対応する節点へ向かう有向枝の集合であり、親子集合型名のラベルをつけることがある。S〈R, Q〉は図1（b）で表わされる。

本稿では、関係モデルと同様にR(U)で構成する属性集合がUであるレコード型Rを、t[X]でレコードtの属性集合Xの値を表わす。値が定まれば対応するR(U)のレコードがただ1つ定まるような最小の属性集合KをRのキーと呼ぶ。Rのレコードに複製が存在する場合があるので、必ずしもK⊆Uである必要はない。キーは複数個存在してもよいが、本稿では一般性を失なわず簡単のため1つだけであると仮定している。

従属性としては、関数従属性（FD）、多値従属性（MVD）、存在従属性（ED）を考える。属性集合Xが属性集合Yに従属であるのは、それぞれ X⟶Y, X↠Y, X↣Yで表わす。FD, MVDは空値まで拡張したもの[4][6]とする。

3．スキーマ設計時の問題点

データベースシステムは、個々の利用者の持つデータをまとめることにより冗長度を減らし、全体のデータを統一的に管理する一方、利用者に対しては正しいデータを供給する必要がある。そのため、スキーマ設計に際して次のような条件を考慮する必要がある。

（1） データの冗長性の除去

（2） 質問に対する処理の効率化

（3） 意味制約の保持

冗長性をなくすことは、データベースの本来の目的の1つであり、データの挿入・更新・削除の際にデータ間の矛盾いわゆる不整合を防ぐ。また、予めどういう質問が多いのかがわかっていれば、それらの質問を処理しやすい設計を行なうべきである。質問処理とスキーマの適合性の判断基準も必要となる。[5] しかし、質問処理の効率化のためには冗長性を持たせた方がよい場合が多い。

実世界でデータが持つ意味制約をスキーマ設計に反映させることは、正しいデータの供給の上で重要となる。また、意味制約からデータの冗長性が判断できる場合が多い。だが、意味制約として何を考えるか、例えば、関係データベース理論における各種の従属性や時間的な制約、動的な制約などの

うちどこまでをスキーマ設計に用いればよいのかという問題点もある。

4．親子集合が表現できる意味

ネットワークモデルに対するスキーマ設計を行なう場合、その構造の基本単位である親子集合がどのような意味を持つのか、どのような目的で使われているのかを明らかにしなければならない。この親子集合の意味が明確でないことが、スキーマ設計が経験的・直感的となっている原因の1つである。本節では、親子集合が表わす意味を明確にすると共に、それらの意味の関連についても考察する。

親子集合は、親レコード型の1つのレコード型に対し、子レコード型の複数のレコード型を対応させている。この構造はデータ間の1対多の対応関係だけでなく、いろいろな意味で使われていると考えられ、

(1) 関数従属性の表現
(2) 検索のための索引構造
(3) 冗長性の除去
(4) 存在従属性の表現
(5) 利用者のビューと考えられる値の保持
(6) 属性値に対する一貫性の保持

をあげることができる。

4.1 関数従属性とデータの類別

(1)は親子集合の構造そのものが子レコード型から親レコード型への関数従属性（FD）を表わしていると見ることができることから一般にも指摘されているもので、FD集合を用いたネットワークモデルのスキーマ設計に関する研究もなされている。[(2)(3)] FDの表現は、データ間の1対多の対応を表わすだけでなくデータの類別にも用いることができる。例えば、年令（20才未満，20代，…）や頭文字による類別を行なう場合、グループ名を親レコードとする構造で表現できる。これも値が定まればその属するグループが決定することから一種のFDの表現である。

4.2 検索のための索引構造

データベースの設計を行なう際に質問処理の効率化が1つの重要な要因であることは3節で述べた。(2)の検索のための索引構造は、検索のキーとなる属性集合を親レコード型とする親子集合型である。

[例1] レコード型R(ABC)で、属性Aの値を指定して

Rのレコードを求める質問が多いとき、例えばAの値がaであるBCの値を求めるときは、一般にRにはt[A] = aであるレコードが複数個存在するのでRのレコードをすべて調べる必要がある。ここで、レコード型S(A)を作り、Aの値が一致するRのレコードをまとめて親子集合型<S,R>を作る(図2(a))と、s[A] = aであるSのレコードsを検索すれば、求めたいRのレコードはsを親レコードとする親子集合の子レコードとして得ることができる。Aの値の種類がm、Aの1つの値に対するRのレコードがn(簡単のためすべて同じ数とする)であれば、Rのレコードの総数はm·nとなる。(図2(b)) Sによって索引構造をつくれば、sを得るのに検索するレコードの数は平均でm/2、最悪でもmである。従って、求めたいRのレコードを得るのに平均m/2+n、最悪でもm+nのレコードを検索するだけでよく、もとのm·nに比べると大幅に減らすことができる。

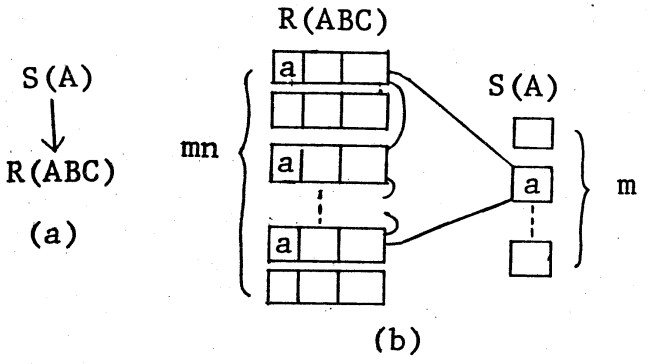

図2 親子集合による検索

4.3 冗長性の除去

親子集合による冗長性の除去は次の場合に分けて考えることができる。

(3-1) 1つの親子集合型で親レコード型と子レコード型に共通の属性集合がある、

(3-2) 1つのレコード型で値が重複する属性集合がある.

(3-3) 複数のレコード型で共通の属性集合がある.

1つの親子集合型で、親レコード型と子レコード型に共通の属性集合があり、各親子集合ではその属性集合の値が一致しているとき、その値は親レコードで代表することができるので子レコード型ではその属性集合は冗長なものとなる。(3-1)はそのような属性集合を子レコード型から除去できる場合である。

(3-2)は値が重複する属性集合からなるレコード型を作り、それを親とする親子集合型を作ることによって(3-1)を適用するものである。例1では、R(ABC)に対し値の重複する属性Aを親レコード型として親子集合型 $\langle S, R \rangle$ を作った。Rのレコードの Aの値はその親レコードと同じであるので、R(ABC)をR'(AB)にすることができる。

(3-2)を複数のレコード型にまで拡張したのが(3-3)である。属性集合Xを含むレコード型 $R_1(Y_1X)$, $R_2(Y_2X)$, …, $R_n(Y_nX)$ ($Y_i \cap X = \phi$, $1 \leq i \leq n$) はレコード型 $S'(X)$

を親とする親子集合型 $\langle S, R_1 \rangle$, $\langle S, R_2 \rangle$, …, $\langle S, R_n \rangle$ を作ることによって属性集合Xを除くことができる。この操作により、1つの属性の値はただ1つだけデータベースに記憶されるような構造にすることも可能である。

[例2] レコード型 R_1(ABCD)、R_2(ABE)で、FD：B→A, C→BD が成り立つものとする。R_1のキーはC、R_2のキーはBEである。R_1とR_2の共通属性はABであり、その値はR_1、R_2で重複して現われる。従って、レコード型R_3(AB)を作り、R_1, R_2をR_1'(CD)、R_2'(E)とすることによってABの値の重複を避けることができる。(図3(a)) R_1'ではCがキーなのでDの値のみが重複し、R_2'ではEのみではキーとはならないのでその値は重複する。R_3ではBがキーとなるのでAの値が重複する。従って、レコード型R_4(D)、R_5(E)、R_6(A)を作り、図3(b)とすることができる。R_2''は、データを持たないポインタのみのレコード型となっている。

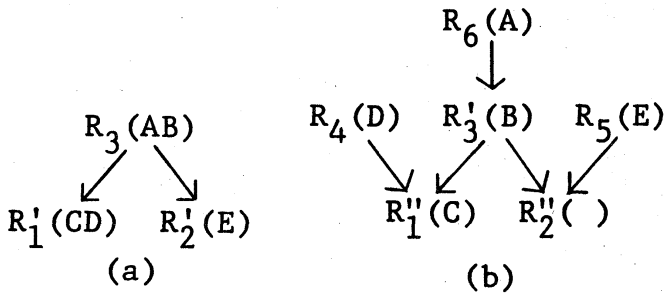

図3 冗長性の除去

冗長性を除くと質問処理が遅くなる場合がある。例えば、

C = c であるようなDの値を求めたいとき、図3(a)ではレコード型R1'でt[C] = cであるレコードtを検索すればよいのに対し、(b)ではR1''でt[C] = cとなるレコードtを検索し次に〈R4, R1'〉の親子集合のポインタをたどりtの親レコードを求めなければならない。いわゆるtime/space trade offであるが、これは設計時に検討を要する問題である。

4.4 存在従属性の表現

親子集合では、子レコードには対応する親レコードが存在する。つまり子レコード型に含まれる属性集合Xの値が存在すると親レコード型の属性集合Yの対応する値が存在するので、存在従属性(ED): X→Yを表わすことができると考えられる。しかし、これは挿入属性や保留属性の指定によって実現されるものであり[4]必然的なものではない。

4.5 ビュー的な値の保持

(5)はレコードの集合から得ることができる値を親レコードに記憶しておくもので、

(5-1) 属性値集合の代表値を記憶する.

(5-2) ある属性の集約値を記憶する.

のふたつを考えることができる。時間的経歴を記憶しているとき

きの最新値や候補の中で現在有効な値（1つの候補集合の中では有効値はただ1つの場合に限る）などは（5-1）であり子レコードの属性の和や平均、レコードの数などは（5-2）である。これらの値は子レコードを検索することによって得られる値である。

［例3］　商品とその価格の移り変わりのデータベースを考える。商品名を親レコード、その価格を子レコードとする親子集合を作ればよいが、子レコードの中での最新の値、即ち現在の価格は商品名と共に親レコードに記憶しておくことが考えられる。（図4(a)）　また、商品を商品群として類別する場合には、4.1よりグループ名を親レコード、商品名を子レコードとする親子集合で表現できるが、商品群ごとの商品の種類の数や平均価格を親レコードで共に記憶することも考えられる。（図4(b)）

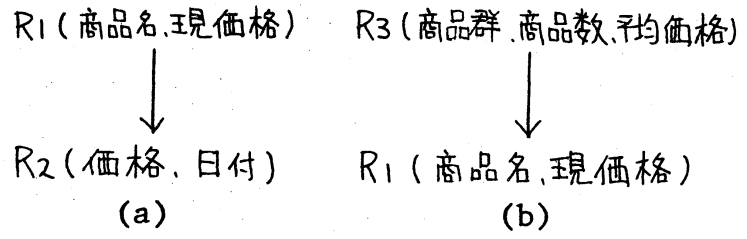

図4　代表値と集約値

4.6　属性値に対する一貫性の保持

一貫性の保持は、同じ属性集合がとりうる値の集合が同一であることを保障するものと、各種従属性の保持とがある。

複数のレコード型に現われる属性集合を1つの親レコード型として まとめることは、一貫性の保持にも用いられる。

[例4]　属性ABを含む2つのレコード型R_1, R_2があり、R_1, R_2では共にAからBへのFDを満足していても全体としては満足していない場合がある。R_1とR_2のレコードが図5(a)であったとすると、それぞれではFD：A→B を満足しているが、全体としてはAの値2に対しBの値は1と2の2つが存在し、この従属性を満足していない。新たにレコード型R(AB)と親子集合型$\langle R, R_1\rangle$, $\langle R, R_2\rangle$を作り（図5(b)）. Rで A→Bを満足させればR1,R2のABの値の集合をRの値の集合の部分集合とすることによって一貫性を保つことができる。

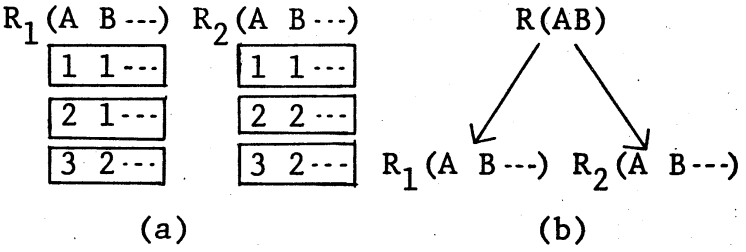

図5　一貫性の保持

一般に、属性集合X,Yを含むレコード型が複数個存在したときのXの値とYの値の対応についての一貫性は、XYからなるレコード型Rを作り、Rを親レコード型とする親子集合によって保つことができる。同様の考え方によって、定義

域間の包含関係も親子集合で扱うことができる。データの更新時には、XYの値が同じレコードはRのレコードを介してすべてつながっているので更新もれを防ぐことができる。また、4.3より子レコード型の属性集合XYは削除することができる。このとき、XYの値の変更はデータベース全般にわたるときはRのレコードの値の変更で、ある特定のレコードに対する変更は親子集合の組換えで可能である。

次に従属性の保持であるが、FDは4.1で述べたように親子集合型に反映させることによって保持される。EDについても親子集合型で保持される場合がある。親子集合の集合によって多値従属性（MVD）も保持される。[4][6] 1対1、1対多の意味制約はFDで表現できる。多対多の表現は、途中に冗長なレコード型を導入した2つの親子集合型で表わされる。従来の実体関連図よりのネットワークデータベースの設計では、1対1、1対多、多対多の意味制約の反映が基本となっている。

利用者のビューと考えられる値も、代表値はその候補となる値の集合を1つの親子集合でまとめることによって、集約値は計算に要する値の集合を親子集合としてまとめることによってそれらの値の一貫性が保たれているかどうかが容易に検査される。

以上のように、親子集合は様々な意味で用いられており、それぞれは独立ではない。これらの意味がうまく合致するようにスキーマ設計を行なわなければならないが、本節で述べたような細い分析はほとんど行なわれていなかったため、スキーマ設計が経験的直感的なものになっている。

5．ネットワークデータベースの設計

本節では4節での議論を基に、ネットワークデータベースの設計法に対する基本的な考察を与える。設計の順序は、

(1) データに対する制約の保持

(2) 質問処理に対する効率を上げるための変更

で行ない、同時に冗長性についても考慮する。

データに対する制約としては各種従属性を与える。質問処理の効率の向上のためには、特定の属性に対する索引の生成、子から親へたどる場合を減らすための設計の変更、親子集合におけるレコードの順序の決定、集約値や特定の値の親レコードへの埋め込み等が考えられる。

FD：$X \rightarrow A$ はその意味によって

typeⅠ：XAを1つのレコード型にまとめておくもの

typeⅡ：親子集合型で表現するもの

の2つに分けることができる。typeⅠは、XとAの値の対応

が1対1に近い場合、Aが集約値を表わしている場合、代表的な質問にAの値を指定して検索を行なうようなものがない場合など、親子集合で表現する必要性があまりないものである。typeⅡは、Aの値を指定して検索を行なう質問が多いと考えられる場合、ERを構造で表現したい場合などがある。

次に実際の例を用いてネットワークデータベースを設計する。

[例5]　属性として次のものを考える。

先生、(先生の)年令、科目、単位数、(科目の)平均点、点数、評価、学生、(学生の)住所、修得単位、入学年度、講座、(講座の)責任者

代表的質問として次のものがあるとする。

q1：ある年度に入学した学生すべてについて、学生順に修得単位と学生名を求めよ

q2：すべての科目についてその担当者と平均点を求めよ

q3：すべての学生について、点のよいものから10科目、科目名と点数を求めよ

q4：すべての講座について、それに所属する先生と学生を求めよ

q5：評価が優の学生がいる科目を求めよ

g_6：ある科目に対し、成績が優秀な者5人を求めよ

次のようなFDが成立している。

先生→年令：type I　先生の人数が少ないと1対1に近く、
年令からの検索もない。

先生→講座：II　講座からの検索(g_4)がある。

科目→平均点：I　平均点は集約値である。

科目→単位数：I　単位数からの検索はない。

学生→住所：I　ほとんど1対1である。

学生→講座：II　講座からの検索(g_4)がある。

学生→修得単位：I　集約値である。

学生→入学年度：II　入学年度からの検索(g_1)がある。

学生・科目→点数：I　点数からの検索はない。

点数→評価：II　評価からの検索(g_5)がある。

講座↔責任者：I　1対1の対応である。

MVD、EDは次のものがある。

科目↠先生：1つの科目を複数の先生で担当する。

先生⊢講座：すべての先生はどこかの講座に属している。

科目⊢先生：すべての科目には教える先生がいる。

学生⊢入学年度：すべての学生は入学年度をもつ。

講座⊢責任者：すべての講座には責任者がいる。

FD、MVDより、図6(a)とすることができる。

レコード型 R(X) のキーを K とする。レコード型 S(Y) で Y ⊇ K であれば、FD: Y → X が成り立つので親子集合型 <R, S> を作ることができる。図6(a)では、R4 は R1 のキー"科目"と R7 のキー"学生"を含むので、それぞれの子レコード型とすることができる。

FD:科目→先生 は R1 のレコードには必ず対応する R3 のレコードが存在し、R3 のレコードには必ず R5 のレコードが存在するという制約になる。子レコード型が親レコード型に存在従属であるのは、挿入属性と保留属性によって表わすことができる。この例では、データの意味からどの親子集合型についても子レコードには対応

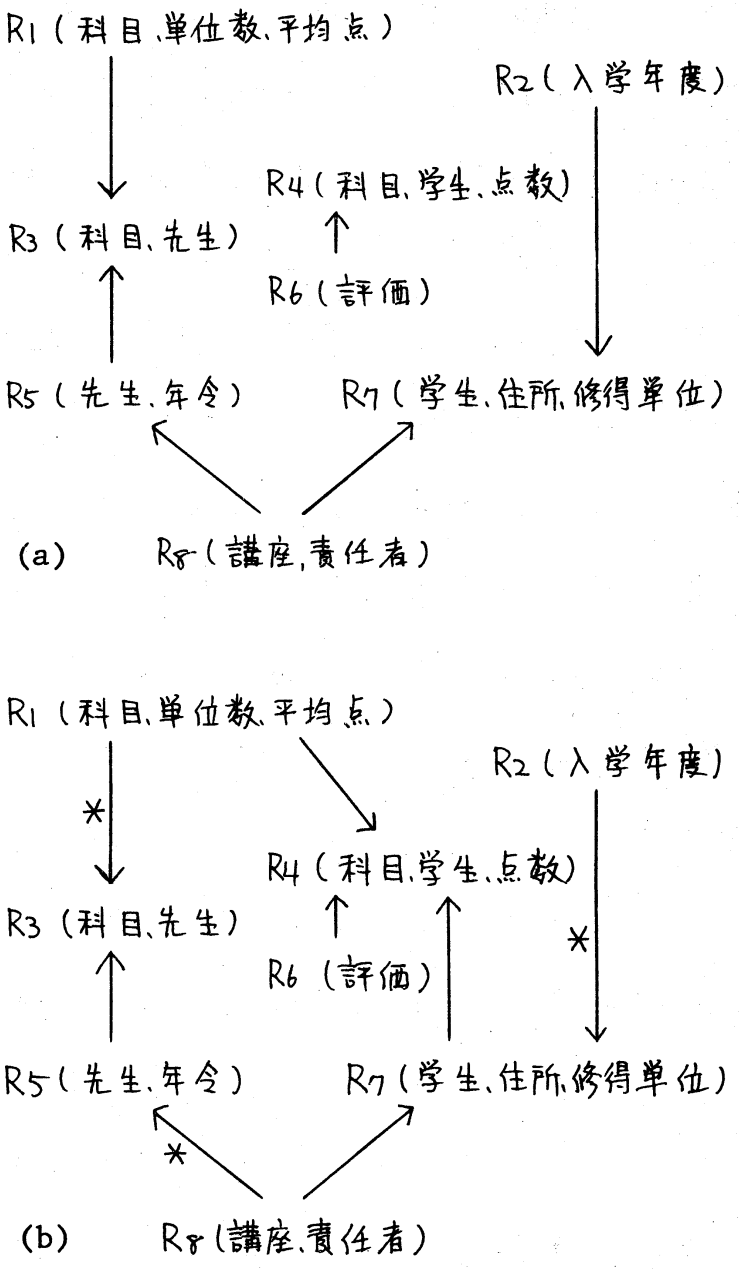

図6 データベースの設計(I)

する親レコードが存在する。親レコードが対応する子レコードを必ず持つ場合を＊で表わしている。これは、レコードの挿入・削除の際に管理を要するものである。ここまでの結果が図6(b)である。

R3では、属性"科目","先生"はそれぞれ親レコード型R1,R7の属性なので、冗長なものとなっている。しかし、g2では科目ごとに先生を求める質問となるので、ここでは"先生"だけを除去する。同様に、R4では"科目","学生"が冗長であるが、質問g3より"科目"は削除せず、"学生"のみを削除する。

次に子レコードの順序であるが、g1では学生順の指定があるので、〈R2,R7〉の親子集合では子レコードは"学生"の値で並べる。g3では点のよいものから選ぶので〈R2,R4〉は点数の順に子レコードを並べる。g6でも成績の優秀なものから選ぶので〈R1,R4〉も点数の順に子レコードを並べる。親子集合型で子レコードの順序を決めたいときに、何の順序によるかを、

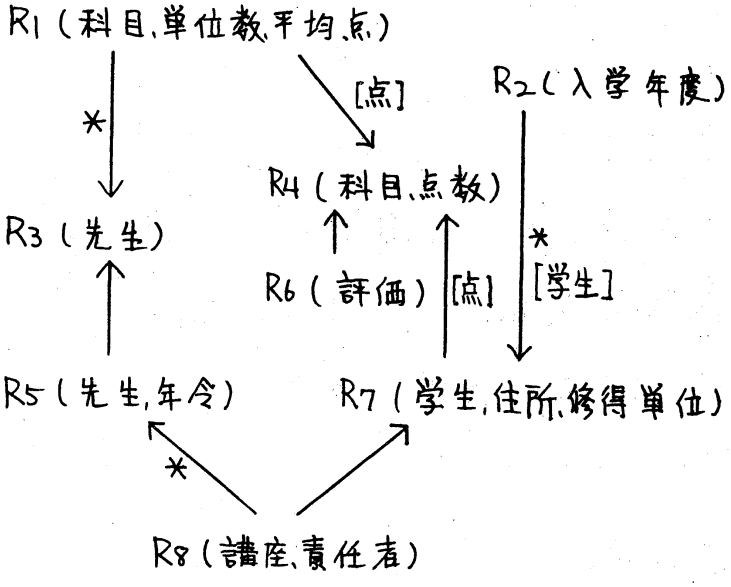

図7 データベースの設計(II)

[ ] で表わすと、最終結果は図7となる。

文献(7)では、例5では扱えなかった質問処理の効率化のための変換（索引構造の付加や親レコード型の属性を子レコード型にも持たせる等）、履歴の管理の問題を扱った例を示している。

6. あとがき

本稿では、ネットワークデータベースの設計のために、ネットワーク構造の解析を行なった。また、その結果から、実際のデータベースの設計を示した。これでネットワークモデルに対するスキーマ設計の問題点が明らかになったと思う。

謝辞

日頃、熱心に御指導、御検討いただく京都大学矢島脩三教授に深謝いたします。また、熱心に御討論いただく研究室の皆様に感謝いたします。

参考文献

(1) 上林，Kim，酒井編，最近のデータベースシステムとその応

田，bit 1月号別冊，共立出版株式会社，1984．

(2) Kuck,S.M., Sagiv,Y., "A Universal Relation Database System Impremented Via the Network Model", Proc. of 1st ACM Symposium on PODS, pp.147-157, March 1982.

(3) Kuck,S.M., Sagiv,Y., "Designing Globally Consistent Network Schemas", Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp.185-195, May 1983.

(4) 古川，上林，矢島，"ネットワークモデルにおける意味制約"，電子通信学会オートマトンと言語研究会，AL83-43，1983．

(5) 上林，古川，"関係データベース理論を用いたネットワークデータベースの設計"，電子通信学会オートマトンと言語研究会，AL83-58，1984．

(6) Lien,Y.E., "On the Equivalence of Database Models", JACM, Vol.29, No.2, pp.333-362, April 1982.

(7) 上林，"関係モデルによる設計と比較"，情報処理学会「データベース」シンポジウム，1983．