関係形式モデルデータベースの問合せ言語における関数従属性の導入†

安達 淳††,†††† 斎藤忠夫††† 猪瀬 博††

本論文は，自動的な表結合機能を持つ問合せ言語を提案し，その実働化について述べている．表の正規化により，関係形式モデルデータベースのスキーマは多くの表から構成されるため，従来からの表操作型問合せ言語では表の結合の記述が複雑となりがちな傾向があった．自動的な表結合機能の導入により，問合せは，表名による属性名の修飾や表結合の仕様の記述が不要となり，属性名だけで記述できることになるので，表現が容易になる．この機能は，関数従属性に基づく属性間の決定関係に着目し，問合せの中で利用者から指定された属性を関連づける属性を探索し，そのような属性が存在するか否かによって結合可能性を判定した上で，その属性から利用者指定の各属性を関連づける表の系列を求め，最終的に木状の表結合のグラフを得ることによって実現している．

この手続きではスキーマ構成によっては曖昧さが生じる場合もある．またデータベース本来の機能として任意の形の問合せを記述できる必要もあるため，問合せ言語には表の結合仕様の記述機能もあわせて持たせている．

実働化は PL/I を親言語として，各種のデータ操作・定義機能を備えた処理系として実現し，スキーマ記述に負担をかけずに従来の関係形式モデルのわくぐみの中で問合せ言語を簡単にできることを示した．

1. まえがき

データベースシステムの開発・研究の流れにおいて関係形式モデル (Relational Model)[1),2)]は，

1）集合演算による問合せ (query) の表現によって，複雑な問合せを非手続き的な問合せ言語で表現できる．

2）正規化された表という簡潔なデータ構造を採っており，この性質が完全性 (integrity) の維持にとって好ましい．

等の特徴を持ち，多くの研究者の注目を集めて来た．

関係形式モデルによるデータベースシステムの実働化を見ると，その開発の主眼は主に1)の点に向けられていると言える．たとえば，SEQUEL (System R)[3)~6)] や EQUEL(INGRES)[7),8)] などは，モデルの持つ一般性をいかした複数の表に対する高度の操作機能を持つ問合せ言語であるが，基本的にはその問合せ言語は表操作言語であって，スキーマの意味的な構造とは独立なものであると考えられる．

一方，2)で述べた正規化操作はスキーマ設計と関連しており，データの意味的側面を捕えている．本論文は，正規形の基礎となる属性間の関数従属性を利用して，処理系による自動的な表結合機能を持った問合せ言語を提案している．これによって利用者は表結合の記述なしで属性名だけで問合せを行うことができ，従来からの表操作型言語よりも簡単な形で問合せを表現できる．さらにこの問合せ機能を含んだデータ言語を構成し，その処理系の実働化について述べている．

2. 関数従属性を利用した問合せ機能

2.1 第3正規形の表の作る構造

関係形式モデルではまず表が第1正規形になっていることを前提とする．この表に対し，データ更新等の不都合 (anomaly) を避けるために，正規化 (Normalization) を行い，最終的に第3正規形 (3rd Normal Form, 3NF)[9)] の表を得て，データベースのスキーマが決定される．この正規化は属性間の関数従属性 (Functional Dependency, FD) に基づく表の分割操作である．

一般に属性間の関連は FD によって方向づけられた有向グラフで表現されるが，3NF の表を得るための正規化とは，この有向グラフでまず遷移従属性を排除し，その後1レベルの FD で切った属性の組を1つの表とすることに相当する．以下では関係 R(X, Y) について FD: X→Y があるとき (X, Y は属性あるいは属性の組)，X を決定項 (determinant) と呼び，表 $\underline{X}Y$ と表わすことにする．

正規化によって不都合のない表の集合が得られるが，この副次作用として必然的に表が細分化されるこ

† Relational Query Language with Embedded Functional Dependency by JUN ADACHI, TADAO SAITO and HIROSHI INOSE (Faculty of Engineering, University of Tokyo).

†† 東京大学工学部電子工学科
††† 東京大学工学部電気工学科
†††† 現在は東京大学大型計算機センター

とになる．このため問合せを行う場合に複数の表を結合(join)する操作が必要となり，従来からの問合せ言語では利用者が個々の問合せごとに使用する表の結合を指定するという方法が採られることが多い．たとえば，問合せを部分的な検索のネストで表現したり[4]，あるいは検索の条件を表わす論理式の中に結合を意味する項を付加する[8]という言語形式を採って解決している．

正規化された表の集合体であるデータベーススキーマには，属性間に表というわくを越えた，FD による意味的な関連性が内包されており，これを探索することによって利用者が表結合仕様を指定せずとも，スキーマに内包された形での表の結合を処理系が自動的に行うことが可能であると考えられる．

2.2 表の結合可能性

3NF の表からなるスキーマが与えられた場合の属性間の決定性を次のように定義する．**図 1** は表AB, BCD, BE, EH, CF, DG の属性の関連を示す．表AB のように決定項が1つの属性の場合，A が B に決定性を持つとする．複数の属性が決定項をなす表BCD の場合，B は D を一意に定めないが FD : BC→Dに内包された意味的関連で B は D のいくつかの値を導くので，B は D に決定性を持つとする．すなわち決定項を構成する属性は表のほかの属性および自分自身に対し決定性を持ち，決定項でない属性への決定性（A から B など）をtype 1，決定項の属性どうしの場合（B から C など）を type 2 と区別する．したがってたとえば A と F は A≫B≫C≫F という決定性（≫で表わす）の鎖で関連づけられる．左辺の属性（Fに対しA，B，C）を（決定性の）上位の属性，右辺の属性（Bに対してC，F）を下位の属性ということにする．

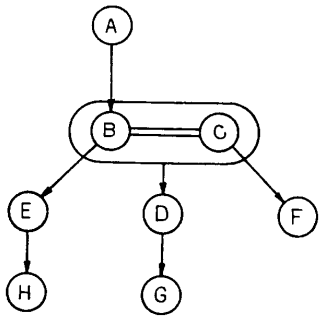


図 1 属性間の関係
Fig. 1 Relation ship among attributes.

属性名の指定だけで自動的に表の結合(joiu)を行うような問合せ処理機能を持つ処理系を考えるために，まず属性名だけが指定された問合せとして，**図 2** のスキーマで属性 MANAGER および EMPLOYEE のみを指定した問合せ（SEQUEL では，SELECT MANAGER, EMPLOYEE WHERE…；や SELECT EMPLOYEE WHERE MANAGER=…；という問合せに相当）について考える．2つの属性を関連づける方法として，

1）2つの属性について可能なすべての組合せを求める．

2）表 α と β を DIVISION で（必要なら表 β と γ を MANAGER で）結合する．

3）表 α と γ を共通の属性 SALARY で結合する．

などいくつか考えられる．1)は意味的関連が不明確で適当でない．2)は指定された属性の上位の属性を探索して関連づける方法（EMPLOYEE は MANAGER の上位属性になっている）であり，3)は逆に下位の属性の探索による方法である．

属性の含まれる表が指定されず，属性名だけから問合せが表現されている場合に採用すべき検索の手法としては，上位の属性によって関連づける 2)の方法が FD の性質とも融合し意味的にも自然であると考えられる．下位の属性による 3)の方法は関連づけの意味が曖昧になったり，あるいは特殊な意味が付加されることが多くなるため不適当である．(3)では，「SALARY が等しい」という条件が付加される．) したがって自動的な表結合機能として次のような手法を採用するのが妥当であろう．

スキーマを構成する 3NF の表の FD に内包された属性間の関連を “system view” と呼び，表を指定せずに属性名で記述された問合せに対しては，指定された属性をすべて含むような部分グラフを system view のグラフから抜き出し，この部分グラフの中で該当す

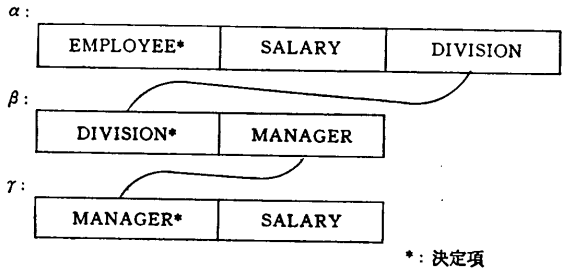


図 2 表の関係の例
Fig. 2 An example of relationship among tables.

る表の結合仕様を求め，属性へのアクセス方法を決める．したがって system view に依存した問合せでは結合の仕様を記述する必要はない．これを行う手順は，

1）属性の結合可能性を判定するために，問合せに指定された属性に対し決定性を持つ上位の属性を探索し，すべての指定された属性を決定性によって関連づけるパスを持つ最上位の属性を求める．

2）最上位属性が求まれば，これを根とし属性をノードとする木状の決定性のグラフが求まり，これが system view の部分グラフである．このグラフに従って最上位属性から指定された属性に至る表とそれらの間の結合の仕様を求める．

となる．以下にそれぞれの手順について述べる．

〔前提〕

結合の対象となる属性とは検索すべき属性および条件節に示された属性である．SEQUEL の場合で言えば，SELECT 節および WHERE 節に現われた属性がすべて結合の対象となる．

〔属性の結合可能性判定手順〕

$\Lambda=\{a_1,\cdots,a_n\}$ を結合の対象となる属性の集合とする（a_i は属性）．すべての a_i に対して決定性を持つ共通の属性 $\tilde{T}$（最上位属性）を求める手順は次のようになる．

step Ⅰ（初期化）

各 a_i について決定性を持つ属性の集合 $\tau(a_i)$ を求める．a_i が決定項の属性なら自分自身を上位属性であるとマークする．

step Ⅱ（判定）

各 $\tau(a_i)$ を調べ，共通の属性 $\tilde{T}$ が含まれていれば，終了．さもなければ step Ⅲ へ行く．

step Ⅲ（探索）

1　すべての a_i について $\tau(a_i)=\phi$ なら，case 2（後述）として終了．

2　$\tau(a_i)\neq\phi$ である次の a_i を求める（$i=1,\cdots,n,\cdots$，の順に繰り返す）．属性の集合 $\Psi=\phi$ とする．$\tau(a_i)$ に含まれるすべての属性 x について，

1′〔x が type 1 の決定性を持つ属性の場合〕

x がすでに a_i の上位属性とマークされていれば case 1（後述）として終了．さもなければ x が a_i の上位属性であるとマークし，Ψ に x に対し決定性を持つ属性をすべて加える．

2′〔x が type 2 の決定性を持つ属性の場合〕

x に対し type 2 の決定性を持つ属性 y がすでに a_i の上位属性とマークされていれば case 1 として終了．さもなければ上記のすべての y および x を a_i の上位属性とマークし，Ψ に y および x に対して決定性を持つ属性を加える．次に Ψ および $\tau(a_i)$ から x および x に対し type 2 の決定性を持つ属性を除く．

3　$\tau(a_i)=\Psi$ として step Ⅱ へ行く

共通の最上位属性 $\tilde{T}$ が求まれば $\Lambda=\{a_1,\cdots,a_n\}$ は結合可能である．step Ⅲ の 2′ は type 2 の場合の上位属性の探索で無限ループに陥るのを回避するためである．この手順は属性数が有限であるから必ず終了する．ただし，case 1 および case 2 は結合に曖昧さの残る場合である．

次に図1の例で，$\Lambda=\{G,H\}$ の場合の適用例を**図3**に示す．

1）step Ⅰ により，$\tau(G)=\{D\}$, $\tau(H)=\{E\}$.

2）step Ⅱ, Ⅲ により，G について①の所をマークし，$\tau(G)=\{B^*,C^*\}$.

3）step Ⅱ, Ⅲ により，H について②の所をマークし，$\tau(H)=\{B^*\}$.

4）step Ⅱ, Ⅲ により，G について③の所をマークし，step Ⅲ の 2′ により $\tau(G)=\{A\}$ となる．

5）step Ⅱ, Ⅲ により，H について④の所をマークし，同様に $\tau(H)=\{A\}$ となる．

6）step Ⅱ により T は B あるいは C と求まる．

〔結合仕様の決定手順〕

最上位属性 $\tilde{T}$ を求めた後，$\tilde{T}$ から各 a_i に至る表とその結合の系列を求める．図3に示したような上位属性のマークを表わすマトリクスを $M(A,a_i)$ とする（A は属性，a_i は結合の対象となる属性．$M(A,a_i)$ は1あるいは0の値をとる）．このための手順は，各 a_i ごとに $U=\tilde{T}$, $M(\tilde{T},a_i)=0$ とした後，次のような

決定性を持つ属性 ≫[2] 属性	結合の対象となる属性 G	H
[1] {B*, C*} ≫ D	[3] 1①	
{B*} ≫ E		1②
{A, C*} ≫ B*	1③	1④
{B*} ≫ C*	1③	1④

1) * は type 2 の決定性を持つ属性であることを示す．
2) ≫は属性間に決定性があることを示す．
3) 1は左欄の属性が上欄の属性の上位属性であるとマークされたことを示す．

図 3　結合可能性の探索

Fig. 3　Search of attributes to examine the possibity of automatic join.

再帰的手続きによって求める.

1 U が a_i に決定性を持つなら, U を決定項の属性とし a_i を含む表を出力して, 終了として戻る.

2 さもなければ, U が決定性を持ち, $M(X, a_i)=1$ であるような a_i の上位属性 X をすべて求める.

〔X が複数個の場合〕

1′ 次に出力する表の位置をマークし, M の a_i の列を待避して, すべての X について $M(X, a_i)=0$ とする.

2′ X を順に選んで, U を決定項の属性, X を属性として持つ表を出力する.

3′ U=X としてこの手続きを呼ぶ. 終了で戻れば, さらに終了として戻る. 無効で戻れば, Mの待避した列および出力した表の位置を回復して 2′ へ行く.

〔Xが1個の場合〕

U を決定項の属性, X を属性として持つ表を出力し, $M(X, a_i)=0$, U=X として1へ行く.

〔Lが存在しない場合〕

無効として, 戻る.

この手順は, 前述の最上位属性の探索を逆にたどるもので, 先の例 $\Lambda=\{G, H\}$ の場合に適用すると (ただし, $\tilde{T}=B$ とする), 属性 G については,

1) U=B だから, X として C*, D が該当する.

2) まず X=C* として表 BCD を出力し, この手続きがさらに呼ばれる. しかし U=C* には該当する X が存在せず無効の探索となる.

3) 次に X=D として表 BCD を出力し, 再びこの手続きを呼ぶ.

4) U=D であるから表 DC を出力して終了.

したがって,

$$\underline{BC}\overline{\underline{D}} \gg \underline{D}G$$

という表とその結合の系列が求まる. Hについても同様で,

$$\underline{B}\overline{\underline{E}} \gg \underline{E}H$$

が求まる.

この例のように各属性ごとに最上位の表が異なる場合には, 最上位属性の含まれる表を適当に選択し (2次記憶へのアクセス法等によって判断される), たとえばこの場合,

$$\underline{BC}\overline{\underline{D}} \gg \underline{D}G$$
$$\gg \underline{B}\overline{\underline{E}} \gg \underline{E}H$$

という木状の系列にできる.

このようにして, 属性ごとの表の系列から, 根となる最上位の表を定め, 表の重複を取り除くことによって問合せの形に応じた木状の表の系列が得られる. これが system view から抜き出した部分グラフとなる.

2.3 検索の曖昧さ

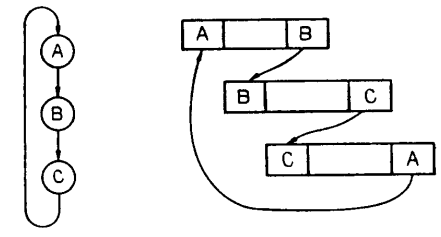
前節の結合可能性の判定で, case 1, case 2 となるのは問合せと system view に曖昧さがある場合である.

〔case 1 の場合〕

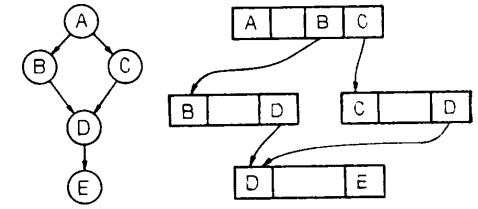
1) スキーマにサイクリックな関数従属性が存在する場合で, これはキー破壊的な関係を一般化した場合である (例, 図 4(a)).

2) スキーマに階属性は存在するが, 属性を結ぶパスが複数存在する場合 (例, 図 4(b). 同じ属性どうしについて2つの表の存在する場合も含む).

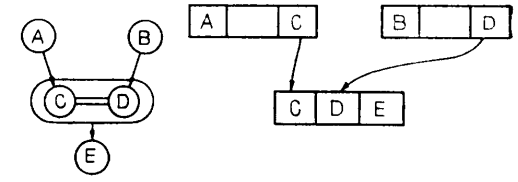
この場合の曖昧さは関数従属性に基づくスキーマ設計の問題[10]と深く関わりを持つが, ここでは与えられたスキーマに対し結合可能性の判定のみを問題として



(a) サイクリックな関数従属性を持つスキーマ



(b) 複数のパスの存在するスキーマ



(c) case 2 の生じるスキーマ

図 4 問合せに曖昧さの生じるスキーマ

Fig. 4 Schema that causes ambiguous queries.

いるのでこれ以上言及はしない.

〔case 2 の場合〕

これはまったく独立したグラフに属する属性どうしや，図4(c)のグラフにおける属性AとDの場合などである．後者の場合は，2.2節で述べたように上位の属性で結合するという手法を採用したために結合不可能と判定される．

2.4 利用者の問合せとの関係

3NF の表からなる階層的なスキーマでは 2.3 節のような場合を除き，2.2節で述べた方法によって処理系の解釈した形で表の結合の仕様を求め問合せに答えることができる．したがってこのような場合は利用者が表の結合を記述する負担がなくなる．

一方，この機能だけではデータの問合せ機能の自由度が不足となることは明らかで，データベース本来の意義からも system view にとらわれず任意にデータ間の関連を設定できることが必要である．ただし，意味的に何らかの価値のある検索は多くの場合ある程度 system view に依存したものであるとは言えよう．しかし次のような問合せでは属性名の指定だけで検索を行うのは難しい．

"学生（学生番号，名前）という表で同名の学生を求めよ".

これは1つの表を2通りに見るという形の問合せであり，表の変数宣言などの機能が言語に含まれていなければうまく記述できない．

以上の観点から，

1）system view に依存した問合せの場合には表の宣言や結合の記述は不要で，処理系が自動的に結合可能性を調べ，表を結合し，問合せに答える機能．

2）上記を補完するものとして，また自由にデータ間の関連を設定するための表の結合の仕様を記述する機能．

を兼備した問合せ言語が適当であると考えられる．

3. データ言語の機能

2章で述べた問合せ処理機能を持つデータ言語の仕様を定め，試作システムとして実働化した．この言語は PL/I を親言語とし，SEQUEL に似たキーワード方式をとっている．ここでは特徴とするデータ定義および問合せ機能について述べる．

3.1 データ定義機能

データ定義においては，属性（attribute）と定義域（domain）の2つを区別し，二層のデータ定義機能を持たせている．

まず定義域に当るものを，DEFINE-VALUE 文を用いて，値集合（value set）として定義する．これはデータ項目のとり得る値に名付けるもので同時にそのデータの型に定める．たとえば，値集合 SALARY，YEAR はともに整数型であるが，異なるカテゴリに属するデータとされる．

属性に相当するものは，DEFINE-TABLE 文によって表とともに，ある値集合上に定義される．たとえば，値集合 EMPNO の上に，1つの表では属性 EMPLOYEE が，またほかの表では属性MANAGER が定義される．このように属性とは同一カテゴリ内の部分集合に付けられる名前であると言える．属性と値集合の区別は SELECT 文の機能と関連を持つ．

3.2 問合せ機能

2.4節で述べた問合せ機能を持つものが SELECT 文で，次のようなキーワードを持つ．

```
SELECT<属性のリスト>
〔INTO   <変数のリスト>〕
〔BY     <変数のリスト>〕
〔USING  <表結合の仕様>〕
〔WHERE<条件>          〕;
```

SELECT 節以外は省略される場合もある．

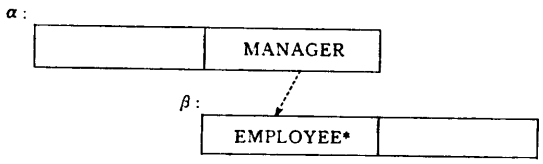
SELECT 節のリストは検索すべき属性を指定する．たとえば表 α の属性 X であれば，$\alpha.X$ と記述するのであるが，表名による修飾は通常省略することができる．

INTO 節，BY 節は親言語との値の授受のために用いられる．INTO 節には親言語の変数リストが記述され，SELECT 節の属性と対応づけられて検索した結果を親言語に渡す．一方 BY 節は親言語から値を受け取って，WHERE 節内の条件の中で値を用いる場合に使用する．

WHERE節では，属性名，BY 節の変数，定数，ストリング等によって検索の条件が論理式の形で記述される．この場合も，通常属性名を表名で修飾する必要はない．

〔USING 節が省略された場合〕

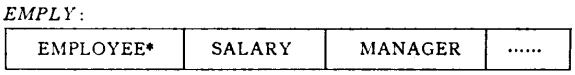
SELECT 文に現われた属性を結びつけるため，2.2節の方法によって処理系が自動的に表の結合を定めて検索する．この場合，結合は同一の属性名（したがって同じ値集合）によって行われる．図5の例で，表 α, β の属性 MANAGER, EMPLOYEEが ともに同じ値集合に属している場合，この2つの属性を軸とし



EMPLOYEE, MANAGER とも同じ値集合 EMP 上の属性である.
*: 決定項

図 5 属性と値集合

Fig. 5 Attribute vs. value set.



EMPLOYEE, MANAGER とも同じ値集合上の属性である.
*: 決定項清定項

図 6 表の例

Fig. 6 An example of table.

て2表は結合できるのであるが, USING 節省略の場合には自動的な結合は行われない.

〔USING 節の機能〕

USING 節は使用する表の宣言, 表の変数宣言, 表の結合の仕様の記述等の機能を持ち,

USING $<T_1>\|<T_2>\|\cdots$

という書式である. <T> は system view に従って結合された木状の表をあらわし, ‖ はそれらが並置されることを示す. <T>の中では表の結合を,

$X_1: N_1(A_1)->(A_2')X_2: N_2(A_2)->\cdots$

ただし, N_1, N_2: カタログされた表の名前

X_1, X_2: この問合せでの表の変数宣言名

A_1: 表 N_1 の属性の名前

A_2, A_2': 表 N_2 の属性の名前

という書式である. N_1, N_2 以外は省略できる. X_1, X_2 の変数宣言は同一の表を2通りに使用する場合や, 木状の表の結合での枝の分岐を明示する必要のある場合に用いれば良い. A_1 と A_2' は表 N_1 と N_2 の結合の軸となる属性を明示したい場合に用いる. 省略された場合には, 同じ属性名, 同じ値集合名の優先度で結合可能性を調べて結合を行う. たとえば, **図6**の表で属性 EMPLOYEE, MANAGER は同じ値集合であるとし, "自分の MANAGER よりも給料の多い EMPLOYEE を求めよ" という問合せでは,

```
USING   E: EMPLY (MANAGER)->M:
                EMPLY
WHERE   E. SALARY>M. SALARY
```

となる. ここで (MANAGER) を省略すると, 表 E と M は同名の属性 EMPLOYEE で結合されるので省略できない. このように USING 節内で結合を表現するので WHERE 節内に結合を表わす式は必要ない.

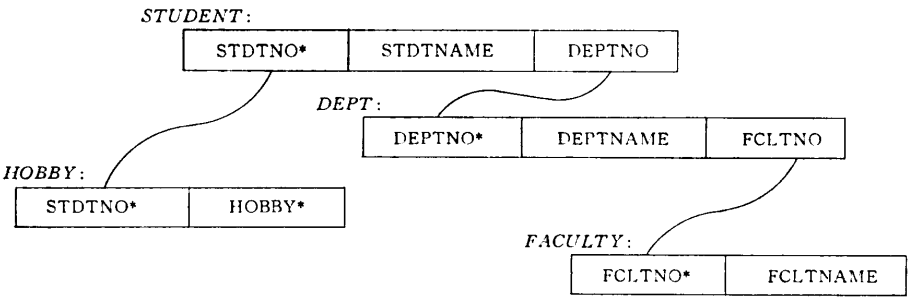
‖ で区切られた<T>の間ではまったく利用者に依存した形で結合が表現される. 同じ値集合による結合等の配慮は必要ない代わりに, WHERE 節内に結合を表わす項を付加しなければならない. たとえば図6の同じ問合せを,

```
USING   E: EMPLY‖M: EMPLY
WHERE   E. MANAGER=M. EMPLOYEE
        & E. SALARY>M. SALARY;
```

と記述することもできる.

3.3 問合せの例

図7に例題のデータベースのスキーマ, **図8**にその問合せ例を示す. 親言語の中でデータ言語部分は1桁目に*を付して示され, データ言語部分を1回通るごとに1タプルずつ目的のタプルが得られて, 結果が親言語部分に与えられる. (a)は "電子工学科の学生名とその趣味を求めよ" という問合せで, 3つの表の結



*: 決定項

図 7 スキーマの例

Fig. 7 An example of schema.

```
EX1: PROC OPTIONS(MAIN);
     DCL (X,Y) CHAR(20);
     DCL  Z    CHAR(12);

 /* NAMES AND HOBBIES OF STUDENTS
    IN ELECTRONICS DEPARTMENT */

     Z='ELECTRONICS ';
LOOP:;
*    "SELECT STDTNAME,HOBBY
*    "INTO   X,Y
*    "BY     Z
*    "WHERE  DEPTNAME=Z;
     IF OK=0 THEN GOTO FIN;
     PUT EDIT(X,Y) (SKIP,X(5),2 A(20));
     GOTO LOOP;

FIN: PUT PAGE;
     END EX1;
```

(a)

```
EX2: PROC OPTIONS(MAIN);
     DCL (A,B,C) CHAR(20);

 /* NAMES AND HOBBIES OF STUDENTS
    WHO HAVE SAME HOBBY       */

LOOP:;
*    "SELECT X.STDTNAME,U.STDTNAME,
*            Y.HOBBY
*    "INTO   A,B,C
*    "USING  X:STUDENT->Y:HOBBY
*            ||U:STUDENT->W:HOBBY
*    "WHERE  X.STDTNAME¬=U.STDTNAME
*          & Y.HOBBY=W.HOBBY;
     IF OK=0 THEN GOTO FIN
     PUT EDIT(A,B,C) (SKIP,X(5),3 A(20));
     GOTO LOOP;

FIN: PUT PAGE;
     END EX2;
```

(b)

図 8 問合せ例

Fig. 8 Query examples.

合を処理系によって自動的に行わせており，属性名だけで記述している．一方(b)は“同じ趣味の学生の名前とその趣味を求める”というもので，1つの表を2通りに見る形の検索であるから，USING 節を用いて表の変数宣言と結合仕様の指定を行っている．

4. 実 働 化

表操作を行うために基本的に必要なデータ記述情報は，表名とその属性名だけであるが，自動的な結合機能を実現するためにさらに必要となるのは値集合に関する情報および表内での決定関係，すなわち決定項に関する情報だけである．したがってデータ記述情報は表ごとに限定されており，これらの情報から問合せに応じて決定関係が解析されるため，スキーマおよびデータ記述情報がそれほど肥大化することなく，また表の追加などの変更に柔軟に対処し得る形で処理系の実働化を行うことができる．

実働化の目的は主に **2, 3** 章で述べた問合せ機能の

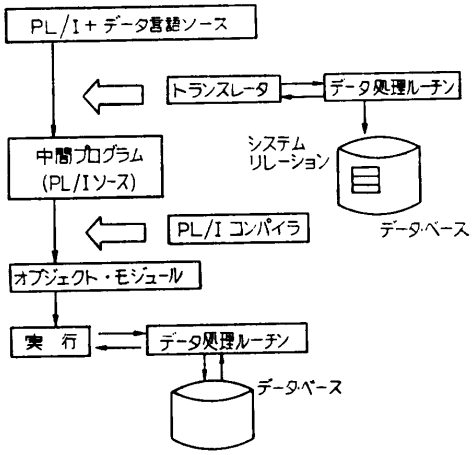


図 9 処理の流れ

Fig. 9 Query Processing.

実現とその有効性の実証にあるが，その他の基本的なデータ操作機能も有しており，データベースシステムとして完備した形になっている．PL/I を親言語としたためバッチ処理向きの処理系として，東大大型計算機センターの H 8800/8700, OS 7 上で稼動するように構成された．その処理プロセスは **図 9** のようになる．

トランスレータはデータ言語部分を，変数の型変換や処理ルーチンの呼出しを行うブロックに変換し，また必要な外部手続きを作成し，PL/I ステートメントだけからなる中間プログラムに落とす．これがコンパイル，実行されて所定のデータ処理が行われる．

記憶域は等長 (1 kB) のページに分割された直接編成ファイルを用い，1) ページマップ，2) システム領域，3) ユーザ領域からなる．2)，3) に収められた表は，表ごとにリスト化されたタプルを収めるデータページと ISAM 的な索引を収めるインデクスページから構成される．トランスレータをはじめすべてのソフトウェアは約 8,000 行の PL/I プログラムで記述されている．

試作システムではデータ言語レベルでのソート機能や組込み関数は，親言語方式を採用するということで割愛している．

5. む す び

関係形式データベースの問合せ言語について，スキーマ記述に余分な負担を加えずに，単なる表操作機能以上の一層使いやすい形の言語の構成を試み，それを試作システムとして実働化した．この言語は，3NF

の表という性質を利用し，表名による修飾や表結合の仕様の記述を不要とし，属性名だけで問合せを行い，表結合は処理系が決定し行うという機能を持っている．

以上の検討および実働化によって判明したことは次のようにまとめられる．

あくまでも従来からの関係形式モデルのわくぐみの中で行ったため，目的の機能が曖昧さのため十分発揮できない場合も生じる．この解決には新たなモデルの開発あるいはスキーマ設計手法の開発が必要となるが，ここでは関係形式モデルのわくぐみの中でもこの機能が部分的に実現し得ることを強調したい．すなわち，実働化した処理系はバッチ向き形態を採っているため曖昧さの生じた場合はメッセージ出力だけで終わっているが，前述の機能の有効性は対話型の利用者インタフェースで一層発揮できると考えられる．この場合対話を通しての曖昧さの解消，利用者への表結合関係の表示と確認等を行いつつ，処理を行っていくことができ，一層使いやすい処理系が実現できると考えられる．

謝辞 本研究に関して熱心にご討議いただいた猪瀬，斉藤研究室の各位に感謝いたします．またさまざまなご指摘をいただいた査読者の方に感謝いたします．なおこの研究は文部省科学研究費の補助を受けたものであることを記して，感謝の意を表します．

参考文献

1) Codd, E. F.: A Relational Model of Data for Large Shared Data Banks, Commu. ACM, Vol. 13, No. 6, pp. 377-387 (1970).
2) Date, C. J.: An Introduction to Database Systems, p. 366, Addison-Wesley, Reading (1976).
3) Chamberlin, D. D. and Boyce, R. F.: SEQUEL: A Structured English Query Language, Proc. ACM SIGFIDET Workshop, Ann Arbor, pp. 249-264 (1974).
4) Astrahan, M. M. and Chamberlin, D. D.: Implementation of a Structured English Query Language, Commu. ACM, Vol. 18, No. 10, pp. 580-588 (1975).
5) Chamberlin, D. D. et al.: SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control, IBM J. Res. Develop., Vol. 20, No. 6, pp. 560-575 (1976).
6) Astrahan, M. M. et al.: System R: Relational Approach to Database Management, ACM TODS, Vol. 1, No. 2, pp. 97-137 (1976).
7) Held, G. D. et al.: INGRES — A Relational Data Base System, Proc. AFIPS NCC, Vol. 44, pp.409-416 (1975).
8) Stonebraker, M. et al.: The Design and Implementation of INGRES, ACM TODS, Vol. 1, No. 3, pp. 189-222 (1976).
9) 穂鷹良介：リレーショナル・データベースにおける第3正規形について，情報処理，Vol. 18, No. 11, pp. 1139-1141 (1977).
10) Tanaka, Y. and Tsuda, T.: Decomposition and Composition of a Relational Data Base, Proc. 3rd Conf. VLDB, pp. 454-461 (1977).
11) 安達 淳，斉藤忠夫，猪瀬 博：表相互の従属性を考慮したデータベースの構成，昭和53年度電子通信学会総合全国大会，5-201 (1978).
12) 安達 淳，斉藤忠夫，猪瀬 博：関数従属性を組込んだデータ言語の構成，情報処理学会データベース管理システム研究会，11-2 (1979).

(昭和55年2月14日受付)
(昭和56年1月22日採録)