数理解析研究所講究録
第525巻 1984年 18-41

位相情報空間型データベースシステムのエンドユーザ言語とその処理系について

大阪大学　工学部　上田修功
打浪清一
手塚慶一

1．まえがき

広域的なデータベースシステムを考えた場合、現実世界の情報のモデル化である概念スキーマは、非常に複雑なものとなる。この時、ユーザが容易にデータベースにアクセスできる事が重要で、そのためのユーザインターフェースであるエンドユーザ言語(EUL)の開発が必要とされている。これは、(i)検索要求が、できる限り非手続き的に記述でき、(ii)複雑な処理要求も容易に記述できる能力を持つ言語でなければならない。又、従来の言語では、遠近・包含関係等の位相を有する質問に対しては、システムは一切サポートしておらず、ユーザの責任のもとで記述、管理されている。一方、最近提案された位相情報空間モデルは、こうした位相情報による検索をサポートしているので、このモデルに基づいて(i)、(ii)を満たすべくEULを設計し、了解性、記述能力等でその有効性を

確認した。構成したEULの処理系としては、構文解析とコード生成の機能が必要である。即ち、EULで書かれた質問文を、ファイル情報代数系上の検索式に翻訳し、それをファイル情報代数系プロセッサで検索処理を行うものである。

構文解析については、文法をデータとみなしプログラムと独立な形をとるパーザを試作し、その有効性を確認した。コード生成については、コード生成方法及び、そのコード生成アルゴリズムを設計した。これらは全て、表駆動方式をとっているので、汎用性のあるシステム構成となっている。

2. 位相情報空間

一般に、遠近と包含の2つの性質を記述、処理することは困難であり、この2者を十分取り扱えるデータベース管理システムはまだ見られない。位相情報空間モデルは、この2つの性質を記述、処理可能なモデルである。

2.1 モデル概説

位相情報空間モデルは意味地図であって、包含される概念に対応する領域は包含する概念の部分空間となり、意味的に近い概念は空間内に近い部分空間として配置されている。

位相情報空間モデルでは、ある視点の認識対象がそのレベルでの個体となる。その個体はそのレベルでの空間内の点、

もしくは領域を占める。そして、その空間は個体の属性を示す軸により構成され、個体を軸に射影したものが、その属性の属性値となる。空間は、関係、個体、属性、属性値という階層を有するが、これは相対的なものであり、現在の視点レベルが個体レベルである。

位相情報空間モデルで取り扱うデータタイプは、個体、集合、線、領域、更にこれらが再帰的に入ったものとして定義される。

(i) 個体… 空間内の一点もしくは領域を占める。

(ii) 集合… 個体の集まりを一段上のレベルで個体と見なされたもの。

(iii) 線… ある個体の順序づけられた組が抽象化されて一つの個体と見なされたもの。

(iv) 領域… 空間内の部分空間。

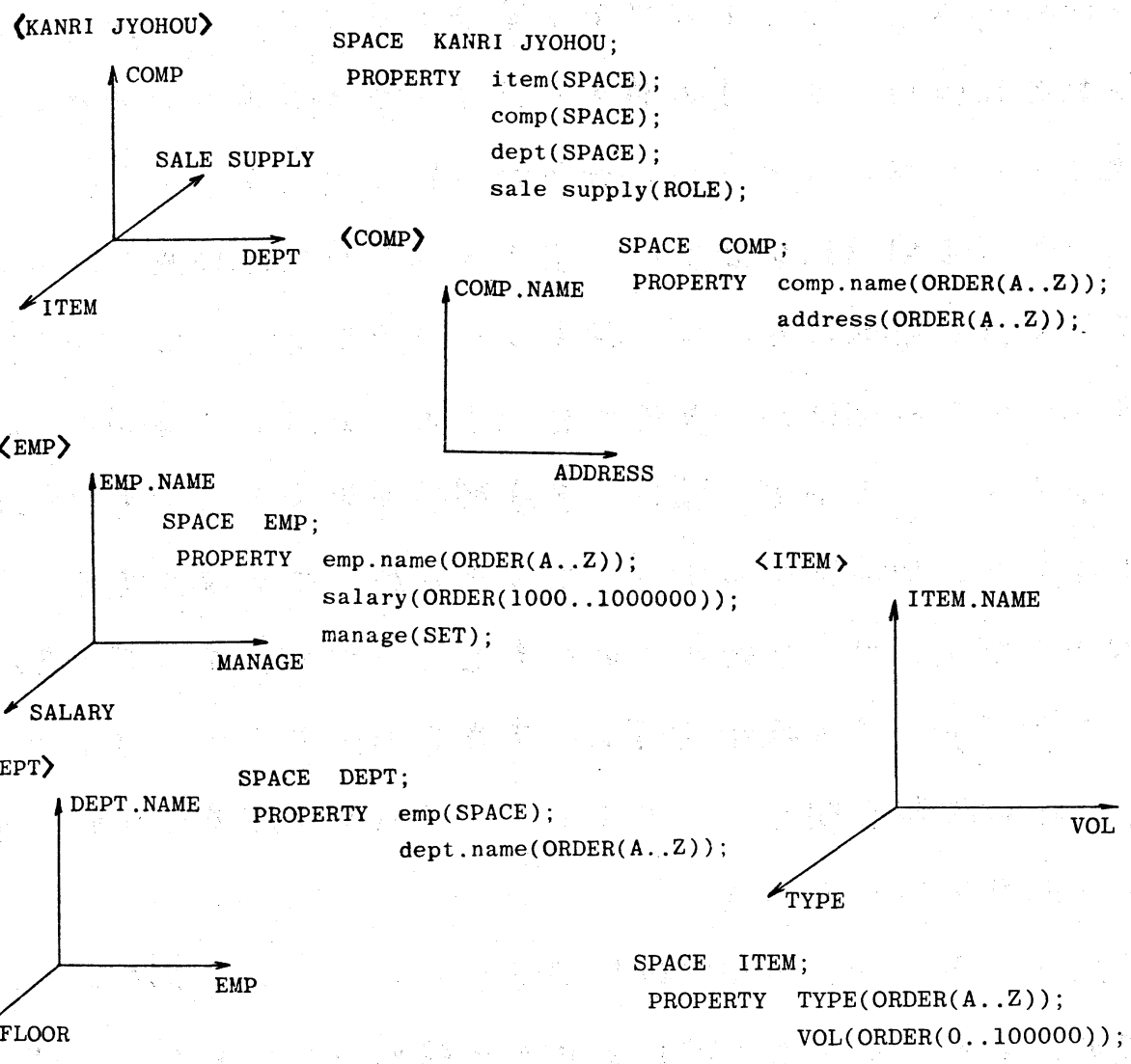
2.2 概念スキーマ

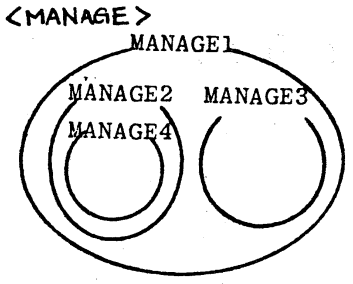
現在、非定型の意志決定用のデータベースシステムとして関係データベースシステムと位相情報空間型データベースシステムの概念スキーマの具体例を記述比較し、両者の特徴について考察する。図1に関係データベースシステムの概念スキーマの例を示す。これに対応する形で、位相情報空間型データベースシステムの概念スキーマを図2に示す。

EMP(NAME,SAL,MGR,DEPT)
SALES(DEPT,ITEM,VOL)
SUPPLY(COMP,DEPT,ITEM,VOL)
SUPPLIER(COMP,ADDRESS)
LOC(DEPT,FLOOR)
CLASS(ITEM,TYPE)

←図1. 関係データベースのスキーマの例

図2. 位相情報空間型データベースのスキーマの概略図






```
SPACE  MANAGE;
 PROPERTY  MANAGE1(SET);
SET MANAGE1;
 PROPERTY  MANAGE2(SET);
           MANAGE3(SET);
SET MANAGE2;
 PROPERTY  MANAGE4(SET);
```

図1を見ればわかるように、関係モデルでは正規形への分解が行なわれる。それ故、個体概念が欠除し、本来同一の個体があちらこちらに点在してしまう。これはデータベースのintegrityをそこなうものであり望ましくない。又、joinによる結合操作は、ユーザが勝手に行なうものであり、その正当性についてはシステムは一切サポートしていない。

一方、位相情報空間では、図2のような空間構成をとり、同一個体が複数存在せず、視点レベルを変えることにより、現レベルでの個体が一意的に定まるのである。又空間は、位相をもっているので、遠近、包含関係を調べることも可能である。又、関係モデルでは、システムが集合を取り扱うことができないので、正規化された表をユーザが、ユーザの責任のもとでjoin等の操作を行ない集合を取り扱っている。ところが位相情報空間モデルでは、システム自身が集合を取り扱える能力を持っている。

以上の事より、位相情報空間型データベースシステムでは、現実世界の情報をある特定の見方を強要することなく、情報を自然な姿で蓄えようとするものである。

3. エンドユーザ言語

データベースシステムは、一般ユーザに広く開かれたシス

テムでなければならず、そのためには情報源としてのデータベースから計算機を全く知らない人間が、必要な時、必要な情報を手に入れることが出来なければならない。大量のデータを数多くのユーザによって共同利用されるためにも、ユーザとデータベースシステムとのインタフェースであるエンドユーザ言語の役割りは非常に重要である。

エンドユーザ言語の満たすべき条件は主に次の3つである。

(i) 検索要求が、非手続き的に容易に記述できる事。

(ii) 複雑な処理要求も記述するに充分な能力を備えている事。

(iii) わかり易い事。

データベース言語を考察する際、上記は重要である。そこで上記をふまえた上で、現在稼動しているデータベースシステムの中で最も有力な関係データベースシステムのエンドユーザ言語について考察する。

3.1 関係データベースシステムのエンドユーザ言語

関係データベースシステムの場合、共通な属性で関係表をjoinすることにより、複雑な処理要求にも対応できる。しかし、ユーザはjoinに必要なtableを正確に知っていなければならず、大きなデータベースになるとjoinの際に、いくつものtableを通さなければならず、その手続きをエンドユーザ言語に課すことは、データベースのユーザの範囲を狭めること

に他ならない。関係データベースでは、数学的な美観を目的として作られているため、semantic constraintが充分反映しておらず、それ故誤った答を出す演算等が許され、正しい答しか出さないようにシステムを設計するのは困難である。これはある意味では、エンドユーザ言語にflexibilityがありすぎるからでもある。

3.2 位相情報空間型データベースシステムのデータ操作言語

本論文の立脚する位相情報空間モデルは、前述したように、データ間の遠近、包含関係を同時に取り扱うことが可能なモデルである。したがってこのモデル上で処理を行なうための言語には、他のモデルで持つ機能に加えて、以下の機能が必要となる。

(i) 意味的に近い、遠いものをアクセスする機能。

(ii) 反対語、同義語、対語的な概念をアクセスする機能。

(iii) 全知識が一つにまとまっているので、知りたいところだけを切り出すwindow的な機能。

(iv) windowからながめたデータをどの視点から見るかというaspectを指定する機能。

これらの機能は、他のデータベースでは充分にサポートすることは恐らく不可能である。

上記の機能をサポートする言語を設計する際に2つのレベルを設けた。

第1レベルは、ファイル情報代数系レベルで、これはファイルを単位とした統合処理を行うレベルである。

第2レベルは、ユーザがスキーマをあまり意識せずに処理を行うレベルである。即ち、ユーザが検索要求を、非手続き的に自然語に近い形で記述するエンドユーザ言語である。

以下に第1，第2レベルの言語について概説する。

3.2.1 第1レベルの言語

これは、ファイル情報代数系レベルであり、ファイルを単位とした統合処理を行う。ファイル基本演算は、異なるファイル同志を合成する演算である。それぞれのファイルは異なったスキーマを有しており、それ故演算結果もまた異なったスキーマとなることが多い。そこでこれを指定する必要があるので、ファイル合成演算子には、ファイル基本演算子の左にスキーマ合成法を指定する添字をつけて、右には統合法を指定する添字をつけて表す。

ファイル基本演算子には次のようなものがある。

(i) マージ (sU_i) : $F_3 = F_1\ sU_i\ F_2$

ファイル F_1 と F_2 をマージし F_3 を求める。

(ii) 交わり (sU_i) : $F_3 = F_1\ s\cap_i\ F_2$

ファイルF_1とF_2の両方に現れているレコードを抽出してF_3を構成する。

(iii)射影（s|i）：$F_3 = F_1\ s|i\ F_2$

F_2を超平面と考え、F_1からF_2への射影として定義される。

(iv)選択（s/i）：$F_3 = F_1\ s/i\ F_2$

ファイルF_1からファイルF_2で指定された属性値をもつ個体等のレコードを選択する。

添字sは、スキーマ合成演算子で、次のようなものがある。

(i)和演算（∪）

両方に現れる全仕様を列挙したもの。

(ii)積演算（∩）

両方に現れる仕様を列挙したもの。

(iii)指定演算（D）

仕様を陽に指定するもの。（射影のみ）

添字nは、統合演算子で次のようなものがある。

(i)条件統合（C）

2つのレコードの指定された属性が与えられた条件を満たしたときのみ統合する。

(ii)個体統合（E）

同一個体を表している（単独キー又は複号キーの一致する）ときのみ統合する。

(iii) View 統合（V）

同一 View に属す個体のみ統合する。

(iv) 非統合（N）

統合は行れない。

(v) 無条件統合（U）

同一レコードは無条件に統合する。

以上を組み合わせる事により、ファイル合成演算が定義される。

3.2.2 第2レベルの言語

第2レベルの言語は、エンドユーザ言語であり、検索要求を、自然語に近い形で記述できるものである。例えば、個体，属性関係については、"OF"，"WHOSE"等を用い、"～の…" という風な記述ができ、又個体間の関係性に関しては、"WHICH"，"THAT"等を用い、"～が…する～"といった記述が可能となり、エンドユーザは検索の手続きを陽に記述することなく検索が容易に行えるようになる。以下にその記述例をあげ簡単に説明する。尚、スキーマは図2を用いている。

EX.1　2階で売っている項目を見つけよ。

PRINT name FIND item WHICH dept WHOSE floor='2' sale.

PRINT は、FIND 以下で求めた個体の属性値を出力するもので、EX.1では、item.nameが出力される。WHOSE は、個体の属

性について条件づけするもので、dept WHOSE floor='2'で、"2階にある部門"が得られ、次いでそのdeptが売っている項目を、WHICH, saleによって表現している。saleは、roleであり、位相情報空間では、属性と同種なものと考えている。つまり、WHICH, <role>, によって、deptとitemの個体を関連づけていると言える。

Ex.2　2階にある全ての部門によって売られている項目を見つけよ。

PRINT name FIND item WHICH ALL dept WHOSE floor='2' sale.

ALLは、個体の集まりを抽象化して一つの集合としてとらえるためのものであり、同様なものに、ONLY(～だけ), OTHER(～の他の)がある。詳細は、後述の翻訳処理で述べる。

Ex.3　おもちゃ部門によって売られている項目だけを供給している会社を見つけよ。

PRINT name FIND comp THAT supply ONLY item
WHICH 'TOY': dept sale.

THATは、WHICHと同様、THATと<role>(例ではsupply)で、個体compとitemを関連づけるものである。'TOY': deptは、'TOY'なる識別名は、deptのそれである事を示す。WHICH, WHOSE, THAT等は一つの節で閉じており、従ってこれらが再帰的に用いられても意味があいまいになることはない。

Ex.4　2階にある部門に少なくとも2つの部門によって売られている項目を見つけよ。

PRINT name FIND item WHICH dept WHOSE floor = '2'

AND COUNT (ITS dept) >= 2 .

ITS は、直前の variable と同じものである事を示すもので、同じ variable があちらこちらに存在する場合は、<range variable> によって示すものとする。

Ex.5　LEVI社が供給している項目、または MEN 部門が売っている項目でかつAタイプである項目を見つけよ。

PRINT name FIND item WHICH 'LEVI' : comp supply ; 'MEN' : dept sale

AND WHOSE type = 'A'.

一つの個体を複数の条件で検索する場合、一つの節中に記述できる場合は、";" または "," で区切って記述し、各々"または", "かつ"の意味をもつものとする。又、一つの節中には書けない場合は、AND, OR を用いるものとする。即ち、AND, OR は1つの variable (上例では item) を複数の節で条件付けするためのものである。

これらの例でわかるように、ユーザは、検索の際、トップダウン的な空間構成のみを意識していればよく、従来の言語のように検索に必要な手続きを言語の中に記述する必要がない。又言語自体も非常に了解性に富んだものになっている。

4. エンドユーザ言語の処理系（構文解析部）

エンドユーザ言語である 2nd level の言語は、翻訳されて 1st level の言語に変換されねばならない。即ちEULの処理系としては、EULで書かれた質問文を構文解析し、次いで、ファイル情報代数系上の検索式に翻訳する機能を持つことが必要である。そこでその前者の構文解析のアルゴリズムの設計として Syntax Free Parser（SFP）を試作した。以下に、SFP について概説する。

4.1 Syntax Free Parser とは.

構成したエンドユーザ言語は、まだ完全なものとは言えず、今後 syntax, semantics 共に変更が加わるのは避けられない。故にエンドユーザ言語の処理系としては、できる限り syntax に独立な形で作られるのが望ましいと考えられる。そこで文法をデータと見なし、処理系は文法に依存しない形をとる、Syntax Free Parser（SFP）を試作した。これは、ユーザがある規則（後述）に従う文脈自由型文法（CFG）を拡張BNF記法（後述）で入力すれば、その文法で構成される言語の acceptorを自動的に生成するものである。それ故、文法に変更が生じても、プログラムには全く影響が及ばないという長所を持ち、汎用性のある処理系である。

4.2 SFPによる構文解析

構文解析で一番問題となるのは、解析アルゴリズムの効率であり、効率何如によっては、解析不可能なことさえある。効率上の基本的な要求として次の2つが考えられる。

(i) 各解析ステップは、現在の計算状態と次に読み込まれる1つの記号にのみ依存するものでなければならない。

(ii) いかなるステップも以後取り消されることがない。

故に、本システムでは、(i)(ii)を満たす文法のruleを制限し、そのruleに従う文法の構文解析アルゴリズムを定義する。これは与えられた文法をparseし易い文法に変換する手段でもある。

Rule 1.

生成規則 $A \to A_1 \mid A_2 \mid \cdots \mid A_n$ $(A_i \in (V_{NT} \cup V_T)^*, i=1..n)$ が与えられたとき、次式が成立しなければならない。

$$\forall i,j : first(A_i) \cap first(A_j) = \phi$$

即ち、A_i $(i=1..n)$ から生成される文の先頭の記号の集合は全て異なっていなければならない。

Rule 2.

空記号(null-string)を生成できる $V \in V_{NT}$ については次式が成立しなければならない。

$$first(V) \cap follow(V) = \phi$$

即ち、空記号列を生成できる記号 V ($\in V_{NT}$) に対しては、その先頭記号集合は、V から生成される記号列の後に続く記号の集合と共通部分があってはならない。

尚、Rule 1, 2 は、トップダウン解析で問題となる左再帰定義をも禁止している。

4.3 SFPへの入力文法表現

本処理系では、入力文法の表現（記述）方法として拡張BNF記法を用いる。以下にユーザの入力文法表現方法の仕様をまとめる。

< A >	… Aは非終端語を表す。
	ex. <START>, < $1 >
"α"	… αは終端語を表す。
	ex. "x", "TOM"
::=	… 左辺を右辺で書き換えることを表す。
	ex. < ZERO >::= "0" % （"%"は文末記号）
/	… "または" 即ち、alternative を表す。
	ex. < A >::= "x" / "y" %
N	… empty, nullstring を表す。
	ex. < A >::= N / < B >< C > %
[θ]	… empty または θ を表す。
	ex. <integer>::= ["−"] <number> %

{0}　　… empty, 0, 00, 000 … を表す。

ex.　<number> ::= <digit> {<digit>}

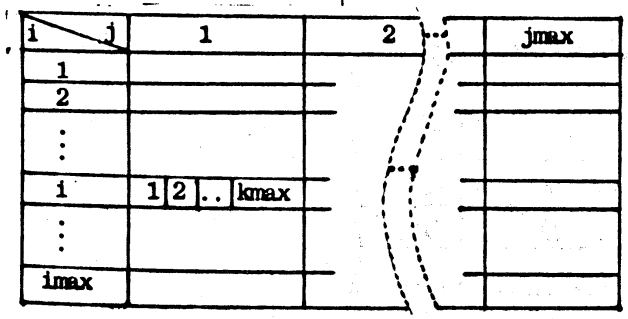
4.4　SFPの制御構造

本処理系は、表駆動方法をとり、特に制御に関しては、syntax table が重要な役割りを果たす。ユーザにより入力された文法は、語い解析がなされ、V_T, V_{NT}, BNF メタ記号等の処理を行い、symbol table と syntax table を作成する。図3に symbol table, syntax table の構造を示す。

SYMBOL TABLE

SYMBOL NUM.	NAME	DEF.	EXIST	
1				non terminal area
:				
ntmaxnum				
ntmaxnum+1				non terminal overflow area
:				
ntovfmaxnum				
tminnum				terminal area
:				
tmaxnum				
tmaxnum+1				terminal overflow area
:				
tovfmaxnum				

SYNTAX TABLE



- SYMBOL NUM. … symbol索引
- NAME … (非)終端語
- DEF. … 非終端語が、再び別の(非)終端語に書き換えられた時 DEF. = 1 となり それ以外は DEF. = 0.
- EXIST … 要素に(非)終端語が存在すれば EXIST = 1, それ以外は EXIST = 0.
- syntax table は i, j, k のインデックスをもつ3次元アレイで、その要素は symbol num. である。
  尚、symbol num. は symbol をハッシュすることにより定まる。

図3　Symbol table と syntax table

3.

次に syntax table のデータ格納法について述べる。

syntax table のインデックス i は、生成規則の左辺の非終端語の symbol num. であり、j は書き換え規則の場合の数を表し、k は各書き換え規則での(非)終端語の順序を表す。以下にその例を示す。

(i) 書き換え規則に空列 (empty) を含まない場合

ex. < START > ::= < A > / "x" < START > %
< A > ::= "y" "z" / "w" %

i \ j	1			2			3
⋮	k=1	2		k=1	2	3	k=1
N(START)	N(A)	0		n(x)	N(START)	0	0
⋮	k=1	2	3	k=1	2		k=1
N(A)	n(y)	n(z)	0	n(w)	0		0

N(START), n(x) 等は、各々、non-terminal < START > の symbol num. terminal "x" の symbol num. を表す。

尚、table 中の "0" は、データの終りを示す。

(ii) 書き換え規則に空列 (empty) を含む場合.

Case 1. N を含む場合

ex. < S > ::= N / < A > "α" / "β" %

i \ j	1			2		3
⋮	k=1	2	3	k=1	2	k=1
N(S)	N(A)	n(α)	0	n(β)	0	0
⋮	...			...		

かつ NULL[(S)] = 1

NULL は、書き換え規則に null string を許す場合、それを示す flag である。

Case 2. [ ] を含む場合

ex. < S > ::= < A > / "x" ["y" < B >] "z" ;

i \ j	1				2					3	
⋮	k=1	2			k=1	2	3	4		k=1	
N(S)	N(A)	0			n(x)	N(*)	n(z)	0		0	
N(*)	n(y)	N(B)	0		0						
⋮											

かつ NULL[N(*)] = 1

Case 3. { } を含む場合

ex. < S > ::= "x" {"y" < A > } "z" ;

i \ j	1					2	
⋮	k=1	2	3	4		k=1	
N(S)	n(x)	N(*)	n(z)	0		0	
N(*)	n(y)	N(A)	N(*)	0		0	
⋮	...						

かつ NULL[N(*)] = 1

以上により構成された symbol table, syntax table, NULL を用いて構文解析が行われるのである。

本処理系は、大阪大学大型計算機上の PASCAL でインプリメントされている。下記の簡単な文法を入力し、次いで入力文が構文解析される様子を以下に示す。

ex. < START > ::= < B > < C > ;
< B > ::= "x" ;
< C > ::= {"y"} ;

この例の場合 symbol table 及び syntax table は、下のようにする。

<SYMBOL TABLE>

SYMBOL NUM.	NAME
⋮	⋮
7	B
8	C
⋮	⋮
24	S
⋮	⋮
139	x
140	y
⋮	⋮

non terminal ↕ terminal

<SYNTAX TABLE>

i \ j	1			2	3
	k=1	2			
7	139	0		0	
8	9	0			
9	140	9	0	0	
⋮					
24	7	8	0	0	
⋮					

NULL[9] = 1

<実行例>

```
 PASCAL * -RANGE                    ← プログラム実行
PLEASE INPUT SYNTAX !!
>< S >::=< B >< C >%                } 文法入力
>< B >::="X"%
>< C >::={ "Y" }%
>E                                  ← 終了記号(E)
① PLEASE INPUT STATEMENT !!
X Y %                               ← 入力ステートメント
RET S
BI=          2
syntax[ 24   1   1]=  ⑦             syntax[i,j,k]
syntax[  ⑦   1   1]=139             とその値のtrace
word=X
RET S
BI=          2
syntax{  7   1   2}=   0            ← 値が0なら
syntax[ 24   1   2]=   8              pop upする。
syntax[  8   1   1]=   9              (k←k+1)
syntax[  9   1   1]=140
word=Y                              syntax[i,j,k]
RET S                               の値がnon-
BI=          6                      terminalの
syntax[  9   1   2]=   9            場合は、その symbol num. とiの
syntax[  9   1   1]=140             値とする。
syntax{  9   2   1}=   0
syntax{  9   1   3}=   0
syntax{  8   1   2}=   0
syntax{ 24   1   3}=   0
CORRECT SYNTAX !!
```

② PLEASE INPUT STATEMENT !!
X %
RET S
BI= 2
syntax[24 1 1]= 7
syntax[7 1 1]=139
word=X ← matching が成功した場合は k←k+1
RET S
BI= 2
syntax{ 7 1 2}= 0
syntax[24 1 2]= 8
syntax[8 1 1]= 9
syntax[9 1 1]=140
syntax{ 9 2 1}= 0
syntax{ 8 1 2}= 0
syntax{ 24 1 3}= 0
CORRECT SYNTAX !!

③ PLEASE INPUT STATEMENT !!
X Y X Y %
RET S
BI= 2
syntax[24 1 1]= 7
syntax[7 1 1]=139
word=X
RET S
BI= 2
syntax{ 7 1 2}= 0
syntax[24 1 2]= 8
syntax[8 1 1]= 9
syntax[9 1 1]=140
word=Y
RET S
BI= 2
syntax[9 1 2]= 9
syntax[9 1 1]=140 ← matching が失敗した場合は j←j+1
syntax{ 9 2 1}= 0
syntax{ 9 1 3}= 0
syntax{ 8 1 2}= 0
syntax{ 24 1 3}= 0
INCORRECT SYNTAX !!

次に%を受理することが書き換え規則で許されない場合、バックトラックすることなくその時点で入力文が入力文法には受理され得ないという事になる。

```
PLEASE INPUT STATEMENT !!
X Y Y %
RET S
BI=          2
syntax[ 24   1   1]=   7
syntax[  7   1   1]=139
word=X
RET S
BI=          2
syntax{  7   1   2}=   0
syntax[ 24   1   2]=   8
syntax[  8   1   1]=   9
syntax[  9   1   1]=140
word=Y
RET S
BI=          2
syntax[  9   1   2]=   9
syntax[  9   1   1]=140
word=Y
RET S
BI=          4
syntax[  9   1   2]=   9
syntax[  9   1   1]=140
syntax{  9   2   1}=   0
syntax{  9   1   3}=   0
syntax{  9   1   3}=   0
syntax{  8   1   2}=   0
syntax{ 24   1   3}=   0
CORRECT SYNTAX !!
PLEASE INPUT STATEMENT !!
E                          ←プログラム停止
*
```

5. エンドユーザ言語の処理系（コード生成部）

前述した様に、第2レベルの言語（EUL）を、第1レベルの言語（ファイル情報代数系）に変換しなければならない。そこで以下では、その変換操作について概説する。

変換操作は節単位に行なわれ、各節で、ファイル統合演算子と、その順序が決まる。EULの記述の主な節についてその変換操作を述べる。

① <property> OF <extended variable>

ここに <extended variable> とは、WHICH, WHOSE, OF 等で条件付けられた variable をいう。1st level の言語は次のようになる。

$$F_{ext.var.} \mid F_T$$

即ち、$F_{ext.var.}$ (<ext.var.> で得られた File) を F_T に射影する。

ここに、F_T は右のような $p = \theta$ (θ:未知) なる temporary file である。

p
θ

F_T

② [ALL] <entity> WHOSE <property> <restriction> <property-value>
[ONLY]
[OTHER]

1st level の言語は次のようになる。

$$F_e. \underset{sp}{/}[\nabla(p)] \; F_T$$

F_T

p
pv

即ち、Fe (<entity> file) を F_T で選択する。ALL 等がある場合、統合演算子 i は $i = \nabla(p)$ となり、選択演算結果と同一 view (p) のもとで同一個体のみを統合する。

ex. 2階にある全ての部門 (ALL dept WHOSE floor = '2')

$$F_{dept} \; /_{\nabla(floor)} \; F_T \; (floor = '2')$$

F_{dept}

dept.name	floor	emp
A	2	α
B	3	β
C	2	γ
D	1	δ

$/_{\nabla(floor)}$

F_T

floor
2

=

dept.name	floor	emp
A, B, C, D	2	α, β, γ, δ

↑ floor により統合

尚、sp は選択演算の方法を示すパラメータである。

③ [ALL] <e.1> WHICH <ext.var.1> <role> [FOR <ext.var.2>]
[ONLY]
[OTHER]

③′ [ALL] <e.1> THAT <role> <ext.var.1> [FOR <ext.var.2>]
[ONLY]
[OTHER]

<e.1>, <ext.var.>, <role> が属す空間 File を F_U, <ext.var.> の空間 File を $F_{ext.var.}$ とすると、1st level の言語は次のとうり。

$$\left(F_U \underset{=}{/}_{\forall(e1)} F_T(role='role') \right) /_{SP}^{[V]} F_{ext.var.1} \left[/_{SP}^{[V]} F_{ext.var.2} \right]$$

SP は、<ext.var.> に、ALL, ONLY, OTHER がある場合下のようになる

	SP
ALL	$\supseteq$
ONLY	$=$
OTHER	$\neq$

即ち、ALL 等は、SP を決定するための伝播条件と言える。

ex. 2階にある全ての部門によって売られている項目

(item WHICH ALL dept WHOSE floor = '2' sale)

<ext.var.>

$$\left(F_{KANRI} \underset{=}{/}_{item} F_T(role='sale') \right) / \supseteq \underbrace{\left(F_{dept} \underset{=}{/}_{floor} F_T(floor='2') \right)}_{F_{ext.var.}}$$

④ PRINT <property> FIND <ext.var.>

$F_{ext.var.} \mid F_T(p=\theta)$ 即ち、$F_{ext.var.}$ を F_T に射影する。

F_T は $p=\theta$ (未知) なる temporary file である。

6. むすび

位相情報空間は、意味地図であってエンドユーザ言語により、一般ユーザは位相を有する検索ができるわけである。又本処理系は、概念スキーマフリーデータベースシステムの言語処理系のアルゴリズムの設計のために試作したものである。

<参考文献>

(1). 打浪清一，手塚慶一："スキーマフリーデータベースシステムにおけるデータ統合について"，電子通信学会論文誌(D) 64-D3 (Mar. 1981)
64-D3 (Mar. 1981)

(2). Uchinami,S;"A TOPOLOGICAL INFORMATION SPATIAL MODEL and ITS APPLICATION to DATABASE MANAGEMENT SYSTEMS",Doctoral dissertation (May.1981)

(3). 上田修功，打浪清一，手塚慶一："位相情報空間型データベースシステムのデータベース言語の構成について" 信学会全国大会予稿 NO 1364 (1983)

(4). 田中他："リレーショナルデータベースの非手続き的検索用言の仕様"，特定研究，学術情報，B-19研究会 (Nov. 1977)

(5). Michel Lacroix and Alain Pirotte: "Example queries in relational language",M.B.L.E. Reserch Laboratory (Jan.1976)

(6). 上田修功，打浪清一，手塚慶一："位相情報空間型データベースシステムのエンドユーザ言語とその処理系について"：電子通信学会 オートマトンと言語研究会資料，AL83-59 (Jan.1984)