データベース・システム　５３－４
（１９８６．５．１９）

リレーショナルＤＢＭＳにおけるマルチ・オカレンス・フィールド・サポート言語の検討

武藤　英男、中村　史朗、大町　一彦
（日立　システム開発研究所）

リレーショナルＤＢＭＳの普及につれて、第一正規形を前提としたリレーショナル・モデルが実世界の情報構造の自然な表現を出来ない点が指摘されてきた。このため、複数の値の存在を許すマルチ・オカレンス・フィールドを含む非第一正規形リレーションの導入が提案されている。報告者等は、この第一正規形リレーションを操作するために、リレーショナル言語ＳＱＬを拡張する。主な拡張点は、（１）マルチ・オカレンス・フィールド内の集合操作機能、（２）リレーションの構造操作機能、の導入である。
（ＳＱＬ：ＩＳＯ国際標準案のリレーショナル言語）

"Relational Database Language Extensions for Multi-Occurrence Fields"
(in Japanese)

by Hideo MUTOH,Fumio NAKAMURA and Kazuhiko OOMACHI(Systems Development Laboratory,HITACHI,Ltd.,1099 Ohzenji,Asao-ku,Kawasaki-shi,215,Japan)

Recently,relatoinal DBMS is widely accepted for a great variety of applications. However, it has been pointed out that the relational model based on First Normal Form is inconvenient for handling real world information naturally. To solve this problem,we propose Non First Normal Form relations that include set valued attributes,so called multi-occurrence fields. This paper summarizes extensions of the relational language SQL,with main emphasis on set manipulation and structure transform functions.
(SQL : ISO Draft International Proposal)

1．はじめに

リレーショナル・モデルはその出発点として第一正規形という前提を置くことにより、簡潔さと数学的基盤の確立に成功した[1)2)3)4)11)12)13)]。しかしながら、リレーショナルＤＢＭＳが普及するにつれて、第一正規形の前提は実世界の情報の構造に必ずしも合っていない点が指摘されてきた。例えば、次のようなリレーションを考える。

従業員(従業員番号，名前，所属，．．．)

業務歴(従業員番号，業務名，着任日，離任日)

資　格(従業員番号，資格名)

これらは、いずれも従業員に関する情報を表している。業務歴および資格リレーションが従業員リレーションと独立している理由は、所属などが各従業員に対し一つ（以下）の値しか存在しないのに対し、業務歴や資格の各属性値は複数存在しうるためである。この時、以下のような不具合点がある。

（1）実世界における実体（エンティティ）として考えた場合、上記３つのリレーションは１つの実体型”従業員”とその属性としてとらえるのが自然である。従業員番号を除いた業務歴の属性群と資格名とは、それぞれ複数値が許される属性（群）となる。したがって実世界の実体とリレーションとの関連性が必ずしも密でない。しかし、値が複数個存在しうる属性は、必ずしも少なくない。この場合、第一正規形の前提はいたずらにリレーションの数を増加させる結果となる。

（2）上記３つのリレーションに渡って属性値を出力したり条件付けたりする場合、従業員番号によってリレーション間の結合操作を指定する必要があり、ユーザが記述する問合せ内容が煩雑になる。

（3）上述のように、リレーション分割数が増えると結合演算の回数も増える。一方、結合演算はリレーショナルＤＢＭＳの性能に最も大きな影響を与える。したがって、リレーションの分割は性能劣化の原因となる。

実は、コッドが高次の正規形を提案した動機自体は、実世界の実体とリレーションとの対応付けの明確化にあった。しかしながら、彼の目的は、異なる実体型(例えば従業員の情報と部門の情報）が混在している場合には達成されるが、従業員に関する上記の情報が混在している場合には、例示した３つのリレーションを導き出すことになり、上述のような不具合を発生する。

これらの問題を解決する手段として、各タプルのフィールド内に値の集合を許す非第一正規形リレーション[7)10)]の導入が提案され、正規化理論との関係、ネストおよびアンネスト・オペレータの導入による関係代数の拡張[5)9)]、蓄積構造や言語拡張例を含むプロトタイプ開発[8)11)]、などに関して議論されている。本稿では、この集合の存在を許すフィールドをマルチ・オカレンス・フィールドと呼び、これを含むリレーションに対する問合せ、および更新機能に関し、現在ＩＳＯ(International Organization for Standardization)において標準化作業が進められているリレーショナル言語ＳＱＬを拡張して統一的な言語仕様を提案する。拡張内容を大別すると、(1)マルチ・オカレンス・フィールド内の集合値の操作機能、および(2)平坦でないリレーションに対する構造操作機能、の導入である。拡張したシンタックスを付録に示す。

拡張内容の説明の前に、まず、マルチ・オカレンス・フィールド、およびこれを含むリレーションの構造を説明する。

2．データ・モデル

マルチ・オカレンス・フィールドをその基本構造から次の２種類に分類する。

(1)繰返しフィールド

単一フィールドとして集合値を含む構造を持つ。

(2)繰返しグループ・フィールド

複数フィールドが組になり集合値を含む構造を持つ。

一方、第一正規形に基づくリレーションでは、単一フィールドに単一オカレンスを含む構造のフィールドのみ許している。これは繰返しフィールドの特殊形であり、シンプル・フィールドと呼ぶ。同様に、複数フィールドが組になって単一オカレンスのみを許すフィールドを、グループ・フィールドと呼ぶ。我々の扱うリレーションは、これら全ての構造のフィールドを含むことが出来る。図２．１にリレーション例を示す。

また、リレーション内で許す繰返しの段数(ネスト・レベル)を次のように設定する。

(1)リレーション定義時

実世界の情報実体表現では、実用上、ネスト・レベル１までで充分と考える。

(2)ユーザ・ビュー

情報利用時には、ネスト・レベル２以上の構造を許す。

では、このような構造を持つリレーションの操作言語機能を説明する。

３．問合せ機能

３．１　集合比較機能の拡張

マルチ・オカレンス・フィールドを含むリレーションに対する選択条件指定のために、集合値間の比較を行う次の集合比較演算子を導入する。

(1)包含比較子

(a) [IS][NOT] SAME [AS]

(b) [NOT] INCLUDES

(c) [IS][NOT] IN

(2)部分一致比較子

(a) [NOT] OVERLAPS

ここで、[]は省略可能な句であることを示す。NOTは否定を表す。現在のSQLは、上記のIN句のみ許している。提案する拡張SQLでは、例えば、図３．１の文献リレーションに対する問合せ「著者として'A1'と'A2'を含む文献を得よ」を、次のように記述できる。

```
SELECT *
 FROM 文献
 WHERE
      著者名 INCLUDES (*'A1','A2'*)
```

ここで、(*と*)は、集合定数を表現している。上記の問合せの結果、文献リレーションから次の情報が得られる。

文献番号	文献名	著者名
10	PA	A1
		A2
		A3

一方、従来のSQLでは、図３．２の第１正規形の文献リレーションに対し、次のように記述する。

```
SELECT X.*
 FROM 文献 X,文献 Y
 WHERE
      X.著者名='A1'
  AND Y.著者名='A2'
  AND X.文献番号=Y.文献番号
```

そして、この問合せ結果として、次の情報を得る。

文献番号	文献名	著者名
10	PA	A1

以上の例から、次の事が分かる。

(1)拡張したSQLでは、自然な問合せ記述が可能になる。

(2)拡張したSQLと従来のSQLでは、問合せ結果として得られる情報量が異なる。すなわち、拡張したSQLでは、文献番号10の文献の全ての著者名を得られるのに対し、従来のSQLでは、選択条件として与えた著者名を含むタプルしか得られていない。全ての著者名を得るには、ここで得た文献番号を持つ文献情報を得るための記述がさらに必要になる。

３．２　集合間結合演算機能の導入

マルチ・オカレンス・フィールドを含むリレーション間の結合演算のために、集合間の比較に基づく結合演算子を導入する。具体的には、前述の集合比較演算子を全て利用する。例えば、図３．１の文献リレーションに対する問合せ「文献'PB'の著者と同じ著者を一人以上含む文献を得よ」は、次のように記述する。

```
SELECT X.*
 FROM 文献 X,文献 Y
 WHERE
      X.著者名 OVERLAPS Y.著者名
  AND Y.文献名='PB'
```

また、問合せ「文献'PB'の著者を全て著者として含む文献を得よ」も、次のように記述する。

```
SELECT X.*
 FROM 文献 X,文献 Y
 WHERE
      X.著者名 INCLUDES Y.著者名
  AND Y.文献名='PB'
```

前者の部分一致に基づく問合せは、従来のSQLでも、図３．２の文献リレーションに対し次のように記述できる。

```
SELECT X.*
 FROM 文献 X,文献 Y
 WHERE
      X.著者名=Y.著者名
  AND Y.文献名='PB'
```

しかし、集合演算機能において述べたように、得られる情報量が異なる他、後者の問合せについては、コリレーション、GROUP BY句、ネストしたSELECT文などを

用いる必要があり、かなりの熟練を要する。

３．３　タプル内組込関数の導入

現在のSQLは、タプル群に対する組込関数として、MAX(最大)、MIN(最小)、AVG(平均)、SUM(合計)、COUNT(計数)を備えている。マルチ・オカレンス・フィールドに対しても、タプル内の値集合に対する組込関数として、次の関数を導入する。

(a) LMAX
(b) LMIN
(c) LAVG
(d) LSUM
(e) LCOUNT

これらをローカル組込関数と呼び、従来のものをグローバル組込関数と呼ぶ。例えば、図３．１の文献リレーションに対する問合せ「各文献の著者数を得よ」は、ローカル組込関数LCOUNTを用いて次のように記述できる。

```
SELECT 文献名,LCOUNT(著者名)
 FROM 文献
```

また、問合せ「２名以上の著者を持つ文献を得よ」も、次のように記述できる。

```
SELECT *
 FROM 文献
 WHERE
      LCOUNT(著者名)>=2
```

従来のSQLでは前者の問合せに対して、図３．２の文献リレーション上に予め文献毎の著者数を得るビューを作成しておく必要があり、特に、突発的な問合せでは使いづらい。また後者は、GROUP BY句を用いて記述する必要がある、などSQL機能を使い分けることが要求される。

３．４　集合演算機能の導入

複数の集合間での集合演算を行う、次の集合演算子を導入する。

(a) UNION
(b) INTERSECTION
(c) DIFFERENCE

例えば、図３．１の観光地リレーションに対する問合せ「'テニス'、'ゴルフ'、'スキー'の中の２つ以上が出来る観光地を得よ」は、次のように記述する。

```
SELECT *
 FROM 観光地
 WHERE
      LCOUNT(レジャー名 INTERSECTION
       (*'テニス','ゴルフ','スキー'*))>=2
```

また、問合せ「軽井沢で出来る'テニス'や'ゴルフ'以外のレジャー名を得よ」は、次のように記述できる。

```
SELECT レジャー名 DIFFERENCE
                    (*'テニス','ゴルフ'*)
 FROM 観光地
 WHERE
      観光地名='軽井沢'
```

前者の問合せは従来のSQLでは記述できない。また、後者は、WHERE節でブーリアン表現する必要があり記述がめんどうになる。

３．５　サブタプル変数の導入

マルチ・オカレンス・フィールド、特に、繰返しグループ・フィールドに対する正確な条件付けのために、サブタプル変数を導入する。例えば、図３．１の観光地リレーションに対する問合せ「'テニス'を２Ｋ〒以下で出来る観光地を得よ」は、次のように記述する。

```
SELECT *
 FROM 観光地(レジャー  X)
 WHERE
      X.レジャー名='テニス'
  AND X.料金<=2
```

上記のFROM節の( )内がサブタプル記述部であり、Xがサブタプル変数である。このサブタプル変数で選択条件フィールド'レジャー名'と'料金'を修飾することにより、上記の選択条件が繰返しグループ・フィールド'レジャー'の任意のオカレンス(サブタプル)に対するものであることを表現できる。サブタプル変数は従来のタプル変数と相似な概念であり、フィールドの繰返しがネストしていても外側から次々に修飾させることにより、簡潔かつ厳密な条件指定が可能になる。もしも、上記の問合せでサブタプル変数を使用しないと、その意味は「'テニス'が出来、かつ２Ｋ〒以下で遊べるレジャーのある観光地を得よ」となる。

３．６　サブタプル定数の導入

グループになっているフィールドに対する等価(=)

条件は、サブタプル定数で簡略指定できる。例えば、図２．１の文献リレーションに対する問合せ「著者'A１'の寄与順位が１である文献を得よ」は、次のように記述できる。

```
SELECT *
 FROM 文献(著者 X)
 WHERE
      X.著者=('A1',1)
```

上記の( )で囲んだ部分がサブタプル定数である。

３．７　構造操作機能の導入

利用するのに都合の良い構造でリレーションを扱えると便利である。例えば、ある著者の書いた文献を知りたい場合、図３．３(a)に示すリレーションより、図３．３(b)の著者毎に書いた文献群を保持するリレーションが扱い易い。また、第一正規形リレーションをマルチ・オカレンス・フィールドを含むリレーション構造で扱うことにより、拡張SQLを用いた簡潔な問合せを行える。

リレーションを利用し易い構造で見せる機能として、構造化および平坦化という２つの構造操作機能を導入する。構造化は、フィールド(群)を繰返し構造にする機能であり、平坦化は逆に、繰返し構造のフィールドを平坦な構造にする機能である。この様子を、図３．４に示す。

図３．４に示す構造化例は、次の構造操作記述で行うことが出来る。

```
SELECT 著者名,文献名
 FROM 文献(NEST (著者名) BY 文献名)
```

NEST~BY句が構造化の指定であり、BY句の前に繰返し構造にするフィールド名(群)を指定し、BY句の後に構造化の基準となるフィールド名(群)を指定する。上例では、構造化の基準となる文献名フィールドの値が等しいタプル群を、その著者名フィールド値群を構造化することにより、一つのタプルにする。

また、平坦化は次のように記述する。

```
SELECT 文献名,著者名
 FROM 文献(FLAT (著者名))
```

FLAT句が平坦化の指定であり、著者名フィールドの各要素と著者名フィールド以外の全フィールドの値を組み合わせて、各々タプルとする。

上の例のように、構造操作指定をFROM節で行った場合、WHERE節で記述する問合せ条件はこの構造操作結果のリレーションに対して適用される。これにより、例えば、図３．２の第一正規形の文献リレーションに対する問合せ「著者として'A1'と'A2'を含む文献を得よ」は、次のように簡潔に記述できる。

```
SELECT 文献番号,文献名
 FROM 文献(NEST (著者名) BY 文献名)
 WHERE
      著者名 INCLUDES (*'A1','A2'*)
```

構造操作指定は、SELECT文の末尾でも行えるが、ここで指定した操作はFROM節で指定した操作の適用結果のリレーションに対して実施する。構造操作を行った場合、問合せ結果のリレーション構造は、これら一連の構造操作の結果になる。例えば、図３．３(b)のリレーションは、図３．３(a)のリレーションに対し以下に示す操作を行った結果得られる。

```
SELECT 著者名,文献名
 FROM 文献(FLAT (著者名))
 NEST (文献名) BY 著者名
```

３．８　配列操作機能の導入

配列は集合の各要素の並び順に意味を持たせることの出来る構造である。従来、COBOLなどの高級プログラミング言語でサポートされ、集合要素に対するループ処理などに利用されている。例えば、セールスマンの一月分の売り上げ実績を配列の一番目に、二月分を配列の二番目に、などのように蓄積する。

繰返しフィールドや繰返しグループ・フィールドにおいても、各要素を配列データとして操作できると便利である。例えば、図３．５(a)に示す文献リレーションを考える。著者フィールドは、著者名フィールドと寄与順位フィールドから成る繰返しグループ・フィールドである。配列機能がない場合、このリレーションに対する問合せ「'B'が寄与順位１の著者になっている文献を得よ」は、次のように記述する。

```
SELECT *
 FROM 文献(著者 X)
 WHERE
      X.著者名='B'
  AND X.寄与順位=1
```

一方、このリレーションを配列構造を導入して図３．

５(b)に示す構造にすると、前記の問合せは、配列操作機能により次のように記述できる。

```
SELECT *
 FROM 文献
 WHERE
      著者名(1)='B'
```

拡張するSQLでは、繰返しフィールドおよび繰返しグループ・フィールドにおける配列構造をオプションで提供し、配列位置指定による操作を可能とする。

４．更新機能

更新機能には以下の３機能がある。

(1)タプル追加機能
(2)タプル削除機能
(3)タプル更新機能

これらの各機能について主な拡張点を述べる。

４．１　タプル追加機能

繰返しフィールドおよび繰返しグループ・フィールドを含むタプルの追加例をしめす。

(1)繰返しフィールドを含むタプルの追加

図３．１の文献リレーションへのタプル追加は集合定数を用いて、次のように記述できる。

```
INSERT INTO 文献(文献番号,文献名,著者名)
        VALUES (30,'PC',(*'A4','A2'*))
```

この例では、文献番号10、文献名'PC'で、著者名が'A4'と'A2'である文献を追加している。

(2)繰返しグループ・フィールドを含むタプルの追加

図３．１の観光地リレーションへのタプル追加は、次のように記述できる。

```
INSERT INTO 観光地(観光地名,県名,レジャー)
        VALUES ('熱海','神奈川',
                (*('温泉',1),('テニス',1)*))
```

上例は全てタプル定数を代入しているが、問合せ結果を指定することも出来る。

４．２　タプル削除機能

この機能は削除対象のタプルを指定するのに前述の問合せ機能を利用できる他は、従来のSQLと同様である。

４．３　タプル更新機能

繰返しフィールドおよび繰返しグループ・フィールドに対する要素の追加、削除そして変更の機能を追加する。

(1)要素追加

図３．１の観光地リレーションに対する更新要求「'軽井沢'で出来るレジャーとして'乗馬'(料金は５Ｋ¥)を追加せよ」は、次のように記述する。

```
UPDATE 観光地
 ADD('乗馬',5) TO レジャー
 WHERE 観光地='軽井沢'
```

(2)要素削除

図３．１の観光地リレーションに対する更新要求「'軽井沢'のレジャーから'スキー'を削除せよ」は、次のように記述する。

```
UPDATE 観光地
 DELETE レジャー FOR レジャー名='スキー'
 WHERE 観光地名='軽井沢'
```

(3)要素変更

図３．１の観光地リレーションに対する更新要求「'軽井沢'のレジャー'テニス'の料金を３Ｋ¥に変更せよ」は、次のように記述する。

```
UPDATE 観光地
 SET 料金=3 FOR レジャー名='テニス'
 WHERE 観光地名='軽井沢'
```

５．おわりに

本稿では、マルチ・オカレンス・フィールドを含むリレーションの操作のために、リレーショナル言語ＳＱＬを拡張した。この拡張により、第一正規形に基づくリレーショナル・モデルの不具合点を克服でき、自然でかつより強力な問合せ記述能力が実現できたと考えている。

参考文献

1) E.F.Codd :A Relational Model of Large Shared Data Banks;Communication of ACM,vol.13,no.6(1970).

2) E.F.Codd :Further Normalization of the Data Base Relationl Model;Data Base Systems,Prentice-Hall(1972)

3) E.F.Codd :A Data Base Sublanguage Founded on the Relational Calculus;Proc. of ACM SIGFIDET Workshop on Data Description(1971)

4) E.F.Codd :Relational Completeness of Data Base Sublanguages;Data Base Systems,Prentice-Hall (1972)

5) R.Fagin :Multivalued Dependencies and a New Normal Form for Relational Databases;ACM TODS (1977)

6) G.Jaeshke,H.-J.Schek :Remarks on the Algebra of Non First Normal Form Relations;ACM PODS Proceedings(1982)

7) A.Makinouchi :A Consideration on Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model;VLDB Proceedinds(1977)

8) H.-J.Schek,P.Pister :Data Structures for an Integrated Data Base Management and Information Retrieval System;VLDB Proceedings(1982)

9) Patric C.Fisher,Ston J.Thomas :Operators for Non-First-Normal-Form Relations;VLDB Proceedings (1983)

10) H.Arisawa :Operations and the Properties on Non-First-Normal-Form Relational Database;VLDB Proceedings(1983)

11) V.Lum :Design of an Integrated DBMS to Support Advanced Applications;Proc. of the International Conference on Foundation of Data Organization(1985)

12) D.D.Chamberlin et.al :SEQUEL2:A Unified Approach to Data Definition,Manipulation and Control;IBM Journal of Res. and Dev.(1976)

13) C.J.Date :An Introduction Database Systems; Addison-Wesley

文献リレーション

文献番号	文献名	著者		出典			キー
		著者名	寄与順位	雑誌名	ボリューム	ナンバ	
10	PA	A1	1	B1	1	2	K1
		A2	2				
		A3	3				K2
20	PB	A1	1	B1	1	1	K2
		A3	2				K3
							K4
30	PC	A3	2	B2	2	1	K3
		A4	1				K1
							K2

フィールド名

タプル

タプル

タプル

シンプル・フィールド　繰返しグループ・フィールド　グループ・フィールド　繰返しフィールド

図２．１　マルチ・オカレンス・フィールドを含むリレーション例

文献リレーション

文献番号	文献名	著者名
10	PA	A1
		A2
		A3
20	PB	A3
		A1

観光地リレーション

観光地名	県名	レジャー	
		レジャー名	料金
軽井沢	長野	テニス	2
		ゴルフ	3
		スキー	2
箱根	神奈川	テニス	3
		オンセン	1
		ゴルフ	5

図３．１　リレーション例
（マルチ・オカレンス・フィールドを含む）

文献リレーション

文献番号	文献名	著者名
10	PA	A1
10	PA	A2
10	PA	A3
20	PB	A3
20	PB	A1

観光地リレーション

観光地名	県名	レジャー名	料金
軽井沢	長野	テニス	2
軽井沢	長野	ゴルフ	3
軽井沢	長野	スキー	2
箱根	神奈川	テニス	3
箱根	神奈川	オンセン	1
箱根	神奈川	ゴルフ	5

図３．２　リレーション例
（第一正規形:図3.1と等価）

文献リレーション

文献名	著者名
PA	A1
	A2
	A3
PB	A3
	A1

(a)

文献リレーション

著者名	文献名
A1	PA
	PB
A2	PA
A3	PA
	PB

(b)

図３．３　リレーション構造の比較

文献リレーション

文献名	著者名
PA	A1
PA	A2
PA	A3
PB	A3
PB	A1

構造化 ------>
<------ 平坦化

文献リレーション

文献名	著者名
PA	A1
	A2
	A3
PB	A3
	A1

図３．４　構造操作機能

文献リレーション

文献番号	文献名	著　者	
		著者名	寄与順位
10	PA	A1	1
		A2	2
		A3	3
20	PB	A3	1
		A1	2

(a)

配列表現 ------>

配列位置

文献リレーション

文献番号	文献名	著者名	
10	PA	A1	(1)
		A2	(2)
		A3	(3)
20	PB	A3	(1)
		A1	(2)

(b)

図３．５　配列構造による情報表現例

付録　拡張したＳＱＬシンタックス（拡張部分　　）

```
<問合せ記述>：：＝ＳＥＬＥＣＴ［<重複排除>］<検索項目リスト>
                  ＦＲＯＭ<リレーション記述>
                  ［ＷＨＥＲＥ<検索条件>］
                  ［<グループ条件記述>］
                  ［<構造操作記述>］
<重複排除>：：＝［ＡＬＬ｜ＤＩＳＴＩＮＣＴ］
<検索項目リスト>：：＝<対象項目>［，<対象項目>）．．．］｜＊
<対象項目>：：＝<項>｜<対象項目>（＋｜－）<項>
<項>：：＝<要素>｜<項>（＊｜／）<要素>
<要素>：：＝［＋｜－］<基本項>
<基本項>：：＝<フィールド指定>｜<定数>｜<組込関数>｜（<対象項目>）
          ｜<集合操作結果>

<フィールド指定>：：＝［<リレーション名>．｜<タプル変数>．｜<ビュー名>．］
                      ［<サブタプル宣言>］
                      （<フィールド名>［<配列指定>］｜＊）
<フィールド名>：：＝<シンプルフィールド名>｜<グループフィールド名>
                  ｜<繰返しフィールド名>｜<繰返しグループフィールド名>
<配列指定>：：＝（<配列位置>［：<配列位置>］）
<配列位置>：：＝<数値定数>｜ＬＡＳＴ
<サブタプル宣言>：：＝<サブタプル変数>．［<サブタプル宣言>］
<サブタプル変数>：：＝<文字列定数>

<定数>：：＝<単一定数>｜<集合定数>
<単一定数>：：＝<文字列定数>｜<数値定数>｜<空値定数>｜<不定定数>
              ｜<タプル定数>｜<サブタプル定数>
<集合定数>：：＝（＊　<単一定数>［，<単一定数>）．．．］　＊）
<タプル定数>：：＝（<タプル要素>［，<タプル要素>）．．．］）
<タプル要素>：：＝<単一定数>｜<集合定数>
<サブタプル定数>：：＝（<サブタプル要素>［，<サブタプル要素>）．．．］）
<サブタプル要素>：：＝<単一定数>｜<集合定数>
<空値定数>：：＝ＮＵＬＬ
<不定定数>：：＝＃

<組込関数>：：＝<グローバル組込関数>｜<ローカル組込関数>
<グローバル組込関数>：：＝（ＡＶＧ｜ＳＵＭ｜ＭＩＮ｜ＭＡＸ）
                          （［ＤＩＳＴＩＮＣＴ｜ＡＬＬ］（<フィールド指定>｜<ローカル組込関数>））
                          ｜ＣＯＵＮＴ（（＊｜［ＤＩＳＴＩＮＣＴ｜ＡＬＬ］<フィールド指定>））
<ローカル組込関数>：：＝（ＬＡＶＧ｜ＬＳＵＭ｜ＬＭＩＮ｜ＬＭＡＸ）
                          （［ＤＩＳＴＩＮＣＴ｜ＡＬＬ］（<フィールド指定>｜<集合操作結果>））
                         ｜ＬＣＯＵＮＴ（（＊｜［ＤＩＳＴＩＮＣＴ｜ＡＬＬ］（<フィールド指定>
                                                ｜<集合操作結果>））

<集合操作結果>：：＝<対象項目><集合演算><対象項目>
<集合演算>：：＝ＵＮＩＯＮ｜ＩＮＴＥＲＳＥＣＴＩＯＮ｜ＤＩＦＦＥＲＥＮＣＥ

<リレーション記述>：：＝<リレーション指定>［，<リレーション指定>）．．．］
<リレーション指定>：：＝［<所有者名>．］（<リレーション名>｜<ビュー名>）
                        ［（<構造操作記述>）］［タプル変数］
                        ［<サブタプル記述>］
<タプル変数>：：＝<文字列定数>
<サブタプル記述>：：＝（<サブタプル指定>［，<サブタプル指定>）．．．］）
<サブタプル指定>：：＝<繰返しグループ・フィールド名>［<サブタプル変数>］
                                        ［<サブタプル記述>］

<検索条件>：：＝<ブール項>｜<検索条件>ＯＲ<ブール項>
<ブール項>：：＝<ブール要素>｜<ブール項>ＡＮＤ<ブール要素>
<ブール要素>：：＝［ＮＯＴ］<ブール基本項>
<ブール基本項>：：＝<述語>｜（<検索条件>）


<述語>：：＝<比較述語>｜<範囲述語>｜<パターン述語>｜<存在述語>｜<空値述語>
<比較述語>：：＝<値比較述語>｜<集合比較述語>
<値比較述語>：：＝<対象項目><値比較子>（<対象項目>｜<検索文>）
<比較子>：：＝<値比較子>｜<包含比較子>｜<部分一致比較子>
<値比較子>：：＝　＝｜＜｜＞｜＜＝｜＞＝｜＾＝
<集合比較述語>：：＝<包含比較述語>｜<部分一致比較述語>
<包含比較述語>：：＝<対象項目><包含比較子>（<対象項目>｜<検索文>）
<包含比較子>：：＝［ＩＳ］［ＮＯＴ］ＳＡＭＥ［ＡＳ］｜［ＮＯＴ］ＩＮＣＬＵＤＥＳ
                 ｜［ＩＳ］［ＮＯＴ］ＩＮ
<部分一致比較述語>：：＝<対象項目><部分一致比較子><対象項目>
                      ｜ＬＣＯＵＮＴ（<対象項目>ＩＮＴＥＲＳＥＣＴＩＯＮ<対象項目>）
                        <値比較子><数値定数>
<部分一致比較子>：：＝［ＮＯＴ］ＯＶＥＲＬＡＰＳ
<範囲述語>：：＝<対象項目>［ＮＯＴ］ＢＥＴＷＥＥＮ<対象項目>ＡＮＤ<対象項目>
<パターン述語>：：＝<フィールド指定>［ＮＯＴ］ＬＩＫＥ<文字列定数>
<存在述語>：：＝［ＮＯＴ］ＥＸＩＳＴＳ<検索文>
<空値述語>：：＝<フィールド指定>ＩＳ［ＮＯＴ］<空値定数>

<構造操作記述>：：＝<構造化指定>｜<平坦化指定>
<構造化指定>：：＝ＮＥＳＴ［<構造化対象記述>］ＢＹ<構造化基準記述>
<構造化対象記述>：：＝<構造化対象指定>［，<構造化対象指定>）．．．］
<構造化対象指定>：：＝（<フィールド名>［，<フィールド名>）．．．］）
<構造化基準記述>：：＝<構造化基準指定>［，<構造化基準指定>）．．．］
<構造化基準指定>：：＝<フィールド名>［，<フィールド名>）．．．］
<平坦化指定>：：＝ＦＬＡＴ<平坦化対象記述>
<平坦化対象記述>：：＝<平坦化対象指定>［，<平坦化対象指定>）．．．］｜ＡＬＬ
<平坦化対象指定>：：＝（<フィールド名>［，<フィールド名>）．．．］）
```

<グループ条件記述>：：＝［ＧＲＯＵＰ　ＢＹ<グループ化基準指定>］
　　ＨＡＶＩＮＧ<グループ選択条件記述>
<グループ化基準指定>：：＝<フィールド名>［（，<フィールド名>）．．．］
<グループ選択条件記述>：：＝<検索条件>

<データ操作記述>：：＝<タプル追加文>｜<タプル削除文>｜<タプル更新文>
<タプル追加文>：：＝ＩＮＳＥＲＴ　ＩＮＴＯ<リレーション名>
　　［（<フィールド名>［，<フィールド名>）．．．］）］
　　（ＶＡＬＵＥＳ<タプル定数>｜<検索文>）
<タプル削除文>：：＝ＤＥＬＥＴＥ　ＦＲＯＭ<リレーション名>
　　［ＷＨＥＲＥ<検索条件>］
<タプル更新文>：：＝ＵＰＤＡＴＥ<リレーション名><更新記述>
　　［ＷＨＥＲＥ<検索条件>］
<更新記述>：：＝<更新指定>［，<更新指定>）．．．］
<更新指定>：：＝<要素変更>｜<要素追加>｜<要素削除>
<要素変更>：：＝ＳＥＴ<変更記述>
<変更記述>：：＝<変更指定>［，<変更指定>）．．．］
<変更指定>：：＝<フィールド指定>＝<対象項目>［ＦＯＲ<更新対象指定>］
<更新対象指定>：：＝<検索条件>
<要素追加>：：＝ＡＤＤ（<定数>｜<問合せ記述>）ＴＯ<フィールド指定>
<要素削除>：：＝ＤＥＬＥＴＥ<フィールド指定>［ＦＯＲ<削除対象指定>］
<削除対象指定>：：＝<検索条件>