データベース・システム 30－4
（1982. 5. 20）

リレーショナルDBMSにおけるドメイン概念の実現方式の提案

武藤英男*・中村史朗*・佐藤和洋*・茂木啓次**
（* 日立システム開発研究所，** 日立マイクロコンピュータエンジニアリング）

1. はじめに

リレーショナル・モデル[3]の提案後，このモデルに基づくデータベース管理システム（DBMS）の研究・開発が活発に行なわれている。リレーショナルDBMSの最大の特徴は，システムとしての使い易さであり，従来，高度なユーザ・インタフェースの実現や，それに伴なう性能劣化に対する性能向上のための研究が重点的に行なわれている。しかし，使い易さを十分達成させるためには，DBMS内部の管理情報の充実（高度化）が必須である。

リレーショナル・モデルにおけるドメインはこの管理情報の一種であり，データ・インテグリティ維持のうえで有用な概念である。本稿では，このドメインの実現方式について述べる。

以下，第2章で，ドメインを使用したデータ・インテグリティ維持のための処理方式を述べ，第3章で，ドメインの属性およびドメイン定義言語仕様を示す。

なお，ドメイン管理機能実現対象のリレーショナルDBMSは，リレーショナル言語SQL[2]（structured Query Language）をサポートしていると仮定する。また，本文では，リレーショナル・モデル[1,3]の用語に関する知識を前提として議論を進める。

2. ドメインによるデータ・インテグリティ維持方式

ドメインは，実世界に存在する同一種類の情報（値）の概念的なプールであり，1つ以上のドメインが存在できる。リレーションは形式的に，次のように記述できる。

n個の集合 $D_1, D_2, \cdots, D_n$（$D_i \neq D_j$ の必要はない。ここに $i, j = 1, 2, \cdots, n$）が与えられた時，リレーションRはこれらの集合の直積 $D_1 \times D_2 \times \cdots \times D_n$ の部分集合である。ここで，$D_1, D_2, \cdots, D_n$ がドメインである。リレーションの各アトリビュートは，そのリレーションでのドメインの役割を表現している（図1）。そして，リレーション間の関係は，同一ドメインに属するアトリビュート値で表現される。

このような位置付けにあるドメインをシステム内部の管理データとすることにより，データ・インテグリティ維持のうえで，以下に示す3項目の役割を果たすことができる。

（1）例外的検索処理に対する警告
（2）不正なデータ操作の排除
（3）データベース仕様変更の影響把握

以下，これら各項目について，データ・インテグリティ維持方式を述べる。

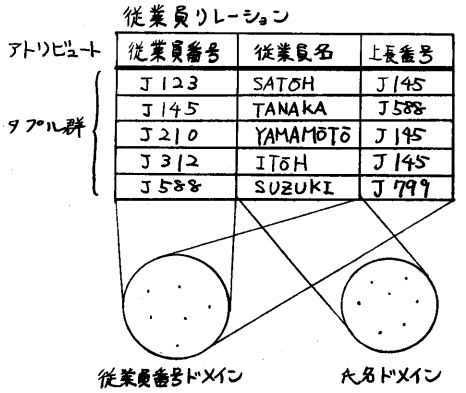

図1. ドメインの位置付け

2.1 例外的検索処理に対する警告

必ずしも無意味とはいえないが，その可能性が大きい例外的検索処理として，以下の項目が考えられる。

(1) 異なるドメインに属するアトリビュートでのジョイン（結合）

例えば，従業員リレーションのアトリビュート"従業員番号"と部品リレーションのアトリビュート"部品番号"とのジョイン。

(2) 異なるドメインに属するアトリビュートでのリストリクション（制限）

例えば，従業員リレーションのアトリビュート"従業員年令"と"給与"間でのリストリクション。

検索処理はデータベースを変更しないため，データ・インテグリティ上の問題はない。よって，システムはユーザに対して，検索結果を出力する。しかし，無意味な情報の利用を回避するために，警告メッセージも同時に出力する。このために，ドメインとそれに属するアトリビュート群との対応関係を，管理データとしてシステム内に保持する。

この他，検索処理に関係して，アトリビュート値に対する関数演算（例えば，平均値，最大値，最小値を求めるもの）や四則演算結果が所属するドメインを検討する必要がある。なぜなら，ネストした検索（キュアリ）やビューを介した検索の場合に，上記(1)および(2)の問題が発生しうるからである。この場合，演算結果はどのドメインにも属さない，すなわちユーザが検索文中で直接記述したものと同様に扱うのが妥当である。

2.2 不正なデータ操作の排除

不正なデータ操作として，次の2項目が考えられる。

(1) ドメイン上に存在を許されないアトリビュート値の登録。

(2) 異なるドメインから検索したアトリビュート値の登録。

ただし，検索結果は，関数演算や四則演算を施されていないとする。

これらの不正なデータ操作は，タプル（群）の追加（insert）あるいはアトリビュート値の更新（update）処理時に発生しうる。

上記(1)を排除するために，ドメイン上に存在できる値の特徴を規定する機能をユーザに提供する。そして，システムは追加や更新処理時，各アトリビュートに登録される値の正当性をチェックする。もしも，不正な値を発見した場合，ユーザに対しエラー・メッセージを出力し，その処理を取り消す。

これと同様の機能は，SQL言語におけるアサーション機能にある。しかし，この機能は，各アトリビュート上に存在できる値を規定するものであり，ドメインに対するものより狭い範囲を対象としている。例えば，年令ドメインには0から150（才）までの値の存在が許されるが，従業員年令アトリビュートには，15から60（才）までの値しか許されない，という場合である。もちろん，アサーションでの規定は，ドメインでの規定に包含されたものでなければならない。

値の特徴としては，データ形式，および存在できる値の範囲（値域）が考えられる。これらのドメイン属性については，第3章で詳しく述べる。

次に，上記(2)を排除するために，ドメインおよびそれに属するアトリビュート群との対応関係を管理データとして保持しておく。そして，追加あるいは更新処理時，ドメインの一致をチェックする。不一致の場合はエラー・メッセージをユーザに出力し，ユーザによる処理を取り消す。

2．3　データベース仕様変更の影響把握

例えば，従業員番号の形式が変更になった場合，従業員番号ドメインに属する全てのアトリビュートの値を変更する必要がある．ドメインおよびそれに属するアトリビュート群との対応関係を管理データとして保持することにより，このようなデータベース仕様変更発生時にも，データベースを正しく変更できる．

3．ドメイン属性および定義言語仕様

前章での説明から，ドメイン管理のためには以下の言語機能が必要なことが分る．

・ドメイン定義機能
・ドメインおよびアトリビュート群との対応付け機能

そして，ドメイン定義機能では，以下に示すドメイン属性を定義する必要がある．

・ドメイン名
　ドメインを識別するためのシステム内でユニークな名称．
・データ形式
　ドメイン上に存在できる値の形式を規定するものである．
・値域
　ドメイン上に存在できる値の範囲を規定するものである．

上記のうち，データ形式と値域について，さらに詳細に検討する．

（1）データ形式

データ形式としてはまず，SQL言語においてアトリビュートの定義で行なっているような，データ・タイプの指定（整数，浮動小数，十進数など）が考えられる。ドメインの属性としてこのような物理的データ形式を導入することは，これまで述べてきたドメイン機能を達成する上で適切でない。例えば，従業員リレーションでの従業員の年令（整数データ・タイプとする）と部門リレーションでの部門毎の平均年令は比較可能でなければならないが，一般に，平均年令は整数値とはならない。

ドメイン属性としては，むしろ，文字データ形式および数値データ形式という抽象的なデータ形式指定に留めるべきである。そして，文字データについては，そのフォーマット，すなわち，英数字の別，文字桁数，予約文字列（例えば，必ず'J'で始まるなら'J'が予約文字列）の指定を行なう．数値データについては，次に述べる値域の指定を行なう．

（2）値域

前章で述べたように，値域とは，各ドメイン上に存在できる値（数値データ）の範囲を規定するものである。この値域は，値の持つ単位（unit）に密接に関連する．例えば，部品の重量（WEIGHT）は，0～10（kg）でなければならないとする．単位の概念を持っていなければ，部品重量ドメインの値域として　0＜WEIGHT≦10　が定義される。しかし，ユーザが，当該ドメインに属するアトリビュートの値として5000（g）を登録しようとする場合も考えられる．この値はシステムにより不正とみなされるだろう．このように，データ・インテグリティを保ち，ユーザの使い易さを向上させるためには，値域と共に単位の概念の導入も必要と考える．

次に，ドメイン管理機能実現のための言語仕様を説明する。

言語機能としては，前に示したドメイン定義機能，およびドメインとアトリビュート群との対応付け機能が必須である。また，単位の概念の導入により，ビュー定義でのアトリビュート記述部，および検索文中での検索アトリビュート・リスト部での単位オプションの指定機能も追加する。

図２に，ドメインに関する言語仕様（シンタクス）をバッカス記法で示す。図中，下線部はＳＱＬ言語に対する変更部分である。そして，図３に，図１に関する定義例を示す。

以下，図２および図３を説明する。

図２（a）～（i）はドメイン定義（define-domain）部分である。

（a）および（b）では，システム内でユニークなドメイン名，およびそのドメイン上に存在できる値の形式（文字あるいは数値）を宣言する。（c）および（d）では，文字（CHARACTER）データについて，そのフォーマットを宣言する。文字データ指定（char-format）での"A"，"9"，および"Z"は，各々英字，数字，および英数字（特殊記号を含む）を表わしている。len1およびlen2では，文字桁数の最小値および最大値を指定する。char-format-spec内で"Z"は，"A"あるいは"9"と混在指定できない。'char-string'では予約文字列を指定する。（e）～（i）では，数値（NUMERIC）データについて，単位名（unit-name）および値域（range-spec）を指定できる。

図２（j）～（m）はリレーション定義（create-table）部分である。

（l）のアトリビュート定義部（field-defn）で，所属するドメイン名を宣言できる。このドメイン名を宣言しない場合は，システムはこのアトリビュートに対して，ドメインに基づく管理を行なわない。

図２（n）～（p）は検索文である。

（p）の検索対象アトリビュートの指定部で，検索結果を出力する時の単位（unit-name）を指定できる。これを指定しない場合，ドメイン定義で宣言した単位を使用する。

図２（q）～（s）はビュー定義（define-view）部分である。

（s）のアトリビュート指定（define-name）部で，ビュー上でのデータの単位（unit-name）を指定できる。指定しない場合は，ベース・リレーションで宣言した単位を使用する。

これらの他の言語機能として単位の定義，および単位間の関係を表現するものも必要であるが，本稿では言及しない。

次に，図３は図１を対象とした，ドメインおよびリレーションの定義例である。

図３（a）は，従業員番号（EMPNO）ドメインおよび氏名（NAME）ドメインの定義である。従業員番号ドメインには文字'J'で始まる３桁の文字（数字）が存在でき，また，氏名ドメインには１～20桁の英字が存在できることを示している。

図３（b）は，従業員（EMP）リレーションの定義である。アトリビュート"従業員番号（EMPNO）"は４桁の空値でない文字列データを含み，従業員番号ドメインに属している。アトリビュート"従業員名（ENAME）"および"上長番号（MGRNO）"は，各々，氏名ドメインおよび従業員番号ドメインに属している。

```
(a) define-domain ::= DEFINE DOMAIN domain-name domain-format
(b) domain-format ::= CHARACTER ( char-format-spec )
                    | NUMERIC [( num-format-spec )]
(c) char-format-spec ::= char-format [char-format-spec]
(d) char-format ::= A [(len1, len2)]
                  | 9 [(len1, len2)]
                  | X [(len1, len2)]
                  | 'char-string'
(e) num-format-spec ::= [unit-name][(range-spec-list)]
(f) range-spec-list ::= range-spec
                      | range-spec-list boolean-op range-spec
(g) range-spec ::= comp-op constant
(h) comp-op ::= = | < | > | <= | >= | ¬=
(i) boolean-op ::= AND | OR

(j) create-table ::= CREATE TABLE table-name (field-defn-list)
(k) field-defn-list ::= field-defn
                      | field-defn-list , field-defn
(l) field-defn ::= field-name (type [, NONULL][: domain-name [(unit-name)]])
(m) type ::= CHAR (integer) [VAR]
           | INTEGER
           | SMALLINT
           | DECIMAL (integer [, integer])
           | FLOAT

(n) select-clause ::= SELECT [UNIQUE] sel-expr-list
                    | SELECT [UNIQUE] *
(o) sel-expr-list ::= sel-expr
                    | sel-expr-list , sel-expr
(p) sel-expr ::= expr [(unit-name)]
               | var-name . *
               | table-name . *

(q) define-view ::= DEFINE VIEW table-name [(field-name-list)]
                    AS query
(r) field-name-list ::= field-name
                      | field-name-list , field-name
(s) field-name ::= identifier [(unit-name)]
```

図2. ドメイン管理のための言語シンタクス

```
DEFINE DOMAIN EMPNO CHARACTER ("J" 9 (3,3))
DEFINE DOMAIN NAME CHARACTER (A (1,20))
```

(1) 従業員番号 (EMPNO) および氏名 (NAME) ドメインの定義例

```
CREATE TABLE EMP ( EMPNO (CHAR(4), NONULL : EMPNO)
                   ENAME (CHAR(20) TAR : NAME)
                   MGRNO (CHAR(4) : EMPNO))
```

(2) 従業員 (EMP) リレーションの定義例

図3 ドメインおよびリレーション定義例

図2で示したように，ドメイン管理のための言語仕様は，リレーショナル言語SQLにおける従来の仕様にほとんど影響を与えない。影響を与える部分も，リレーション定義，検索文，およびビュー定義でのオプションとしてである。これにより，従来のSQL言語を使用しているユーザにほとんど影響を与えずに，ドメイン管理機能を実現していくことができる。

4. おわりに

リレーショナルDBMSの使い易さを向上させるために，データ・インテグリティ維持上有用なドメイン概念の実現方式を述べた。これにより，ユーザからの不正な処理を回避でき，またデータベース管理者の負担も軽減できる。今後，単位の管理方法，およびシステム管理データを保持するデータ・ディクショナリの構造について検討する予定である。

最後に，本研究の機会を与えて頂いた日立製作所ソフトウェア工場高須昭輔部長および石塚忠嗣主任技師に厚く御礼致します。

5. 参考文献

(1) Date, C. J ; An Introduction to Database Systems : Addison Wesley Publishing Company (1981)

(2) Chamberlin, D. D ; SEQUEL 2 : A Unified Approach to Data Definition, Manipulation, and Control : IBM Journal of Research and Development vol. 20, No. 6 (1976)

(3) Codd, E. F ; A Relational Model of Data for Large Shared Data Banks : Communication of the ACM vol. 13, No. 6 (1970)