C2-7　メタ質問処理の設計と実現

柏川伸悟 (Shingo Kashikawa)　　小倉匠吾 (Syogo Ogura)
三浦孝夫 (Takao Miura)　　塩谷勇 (Isamu Shioya)

法政大学工学研究科電気工学専攻
産能大学経営情報学部

概 要

本稿では、メタ質問の実現方法を提案する。メタ質問は、操作により複合値が生じてしまう。複合値は、Codd の関係データモデルで考慮されていないため、複合値をどのように処理して実現していくか考慮する必要がある。また、メタ質問処理の過程で一時的な中間関係が生じる。この一時的な中間関係のスキーマを管理し参照できない場合、中間関係ファイルへアクセスすることができない。実現に際して、一時的な中間関係のスキーマ参照がファイルへのアクセスに必要であることを示し、一時的な中間関係スキーマを管理するカタログファイルを導入する。複合値の処理は、カプセル化をすることで単純値のように扱う。これらを用いてメタ質問の処理を実現する。

1　前書き

現在、メタオブジェクトを用いて設計過程でもスキーマの変更できる継ぎ目のないデータ処理が提案されている。その操作方法は、コマンドインターフェイスとして関係代数を拡張している。本研究では、この拡張された関係代数式の動作原理と、実行方法や実行手順を論じる。この拡張された関係代数の質問は、メタ質問と呼ぶ。一般的に、データ操作はオブジェクトデータを対象とするが、メタ質問はその記述に用いるメタデータおよびそれが意味するオブジェクトデータの操作を意味する。メタ質問によるデータベース操作でない場合、同じ操作をするには前段階に実行した結果を置き換え、それから検索する操作を再び入力することで実現できるが、操作が複雑となる。

メタ質問で代表される操作は、具体化と抽象化と呼ばれる関数である。この関数は、普通の関係代数式を評価する際に一階抽象レベルの高い評価を意味する。

メタ質問は、メタデータを動的に評価する。例えば、具体化の表現方法は ”$ ”記号を用いて表現し、メタデータにオブジェクトデータを対応させる。

メタ質問の実行が、従来のメタ質問ではないデータベースの質問の実行と異なり、単純に解決できない理由は、メタデータが意味するオブジェクトデータである集合をどうやって扱うかということである。メタ質問には、操作によりタプル値に単純値ではなく複合値が存在してしまう。その複合値を含む属性が、射影されるという場合が存在する。選択条件も、必ずしも単純値の比較になるとは限らず集合の包含条件となる。その処理の課程では、動的に複合値を保持したファイル生成がある。さらに、メタデータに対応したオブジェクトデータの再評価も認めているため、より複雑なファイル生成となる。処理には処理途中に中間の関係が生じ、そのファイルに対して何を参照してどのようにアクセスするかということが問題となる。

これらの問題点を解決するためには、質問の処理

対象である集合の処理方法と一時的な中間関係のスキーマ管理が必要になる。

具体的な解決方法として、すでに提案しているHOME(Harmonized Objects and Meta-objects Environment) システム用いて [6][7][8]、メタ質問の処理実行にカプセル化の技術を用いる。さらに、データベースのスキーマカタログとは別に、処理途中で生じる一時的な中間関係のスキーマを管理する一時的スキーマカタログを用意する。カプセル化の技術は、オブジェクトデータ集合を保持しカプセル化データとする。そして、それぞれカプセル化されたデータは処理対象として呼び出されて処理されていく。一時的な中間関係スキーマを管理するスキーマカタログは、次の処理で中間関係ファイルへのアクセスに不可欠なため処理実行に必要である。処理途中の中間に生じた関係であるため、中間関係のスキーマを管理するカタログのファイルは最終的に保存する必要はなく、質問処理の終了時に消滅する。

これまでのデータベースの実行処理に関する研究で、データベースのデータ構造に関する研究では、[4] がある。この研究では、データベースのファイルの構造を取り扱い、処理実行の処理単位であるレコード、フィールドについて述べられている。しかし、フィールドのフィールド値に複合値を保持することについては触れられていない。レコード、フィード値については、3.2 章で詳しく述べる。次に、データベース設計や実現に関する研究では、過去に [1] や、[2],[3] がある。この研究では、データベースの構成や処理の実行手順を述べている。これも、[4] と同様に複合値は考慮しておらず、また実行途中に生じた中間関係の処理や管理について触れられていない。

はじめに第 2 章でメタオブジェクトとその操作を示す。第 3 章で研究システムの構成とファイル構造、そしてスキーマの管理について示す。第 4 章では、その処理に際してアイデアであるカプセル化と中間関係のスキーマ管理について述べる。第 5 章でメタ質問処理の操作手順、第 6 章でむすびとする。

2　メタオブジェクトとその操作

オブジェクトとは, データベース中の興味ある情報を含むデータである。オブジェクトは、データベースの設計時に型を定義され分類される。メタオブジェクトとは、オブジェクト間の特性や情報を保持し、データのデータとも呼ばれる。一般的な質問であるオブジェクト質問には、メタオブジェクトの参照が必要で、メタオブジェクトは必要不可欠である。具体的にメタオブジェクトとは、関係や属性などがメタオブジェクトにあたる。質問による結果は、仮想的に結果スキーマ (RESULT) を用いて記述され、一時的なスキーマの管理をする。

メタ質問は、メタオブジェクトを含んだ質問で、関係代数で表現される。メタ質問は、メタオブジェクト、およびそれが表す外延情報を探索対象とする点で、高階の抽象度を有している。

オブジェクトとメタオブジェクトとの両方でシームレスな評価のために、メタオブジェクト `m` に対して`$m` を意味する具体化 (Deification) 機構を導入する [8]。逆に、オブジェクト $\{o_1, .., o_n\}$ に対して、`$o` $= \{o_1, .., o_n\}$ である $\{o_1, .., o_n\}$ の抽象化 (Reification) を意味するメタオブジェクト `o` を導入する。

例 1 関係 CompanyA で'Fab1, 井上' を評価値にもつ、都道府県と具体化された製品を射影せよ。

CompanyA

都道府県	期日	売り上げ	製品
東京	10	40	靴下
千葉	10	30	靴下
東京	11	40	鉛筆
宮崎	10	5	靴下

鉛筆

工場	責任者
Fab1	井上
Fab2	岡田

靴下

工場	責任者
FabA	林
FabB	高橋

次の質問により求めることができる。

```
Peoject[都道府県,$製品]
Select['Fab1,井上' ∈ $製品](CompanyA)
```

(RESULT)

都道府県	$製品 (工場)	$製品 (責任者)
東京	Fab1	井上
東京	Fab2	岡田

製品の各タプルは、メタオブジェクト鉛筆と靴下である。鉛筆と靴下がメタオブジェクトであるか無いかということは、関係のスキーマカタログを参照し判断する。もしも、鉛筆が参照されスキーマカタログに存在しない場合、単なるオブジェクトデータとして判断され具体化されることはない。選択条件は $\in$ というオペレータにより、靴下と鉛筆の評価結果と'Fab1, 井上' の包含関係で判別する。判別は、靴下に Fab1 と井上に含まれるため (RESULT) の結果が得られる。□

3 HOME システムの構成とファイル構造

本章では、実験に用いる研究システムの構造、ファイル構造、そしてファイルへのアクセスを述べる。本システムでは、実際のファイルへのアクセスはスキーマカタログを参照し、参照して得た情報に従いアクセスされる。

3.1 HOME システムの構成

現在、開発中のメタ質問を処理する実験システムである HOME システムは、OS として FreeBSD を用い実装されている。システムのモジュール構成は次の図のようになる。

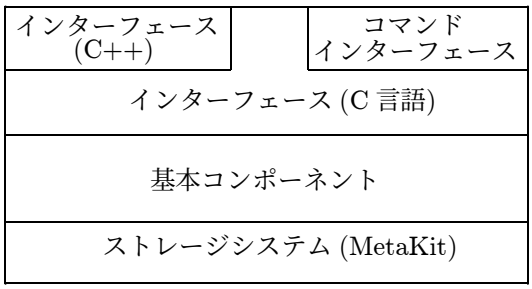

図 1 *HOME* システム

プログラムインターフェースは、C++言語 [9]、C 言語により操作することができる。コマンドインターフェースは、関係代数を拡張した操作言語により操作することができる。インターフェース層は、プログラムインターフェースとコマンドインターフェースから形成される。C++のインターフェースでは、実際には C 言語のインターフェースを呼び実行している。そのため、実行に際して若干のオーバーヘッドが生じている。同様にコマンドインターフェースも同じである。ただし、ユーザーが実際に本システムを扱う場合、C++言語インターフェースを用いた方が単純で扱いやすい。これらのインターフェースを用いてデータベースにアクセスし、HOME システムの基本となるコンポーネントを用いて処理実行を行う。ファイルの I/O に関しては、データの保護を目的とし MetaKit[5] を用いて管理している。データベースファイルのダンピングの場合や、生のバイナリデータを直接参照したい場合は、各種基本コンポーネントを用いてアクセスせずに MetaKit のツール群を用いて参照することが可能となっている。

3.2 ファイル構造

本研究で扱うデータベースのデータ構造は、過去の研究 [4] のような構成になっている。つまり、1 つの関係に対してその関係を構成するファイルは、1 ファイルとしてディスクに格納される。関係のタプルは、内部レベルではレコード (Record) として表現する。レコードの集まりは、1 ファイルとして構成される。レコードは、属性に対応して単数もしくは複数のフィールド値を持つ。このフィールド値は、単純値を持つ。レコードの各フィールド値の割り当ては、スキーマカタログにより管理される。そのスキーマカタログによる情報をもとに、フィールド値へのアクセスがなされ、レコードとフィールドの取捨選択が実行処理でなされる。

3.3 スキーマ管理をするコアセット

メタオブジェクトの完全な意味を記述している本システムのスキーマについて述べる。目的は，我々のスキーマ構造の中だけでスキーマを定義しデータへアクセスすることである。つまり、もし特別なスキーマについて扱うと定義すれば、この構造で全てのスキーマを定義できるということである。我々のスキーマは、これを実現するのに十分であり、これをスキーマ構造のコアセット (*core set*) と呼ぶ。

スキーマは、3 個のリレーションから形成される。

1. RelationCalalog
2. AttributeCatalog
3. DomainCatalog

これで全ての情報を矛盾なく管理できる.
初期状態で RelationCatalog は 3 つのスキーマ関係に相当する 3 つのタプルから構成される。また、Relname,Arity,Width という 3 個の属性を持ち、それらのドメインは name，count，size である。name ドメインの値がを 32 バイト、count，size ドメインを 4 バイトの整数型に仮定すると、Relcat の 1 レコードのサイズは 40 バイトになる。同様に DomainCatalog 関係も定義できる。

RelationCatalog

RelName	Arity	Width
RelationCatalog	3	40
AttributeCatalog	5	104
DomainCatalog	3	40

DomainCatalog

DomName	ValueType	ValueSize
name	CHARACTER	32
count	INTEGER	4
size	INTEGER	4
type	TYPE	32

AttributeCatalog は、3 つのスキーマ関係の全ての属性数にあたる 11 のタプルから構成される。

AttributeCatalog

RelName	AttrName	DomName	Offset	Position
RelationCatalog	RelName	name	0	1
RelationCatalog	Arity	count	32	2
RelationCatalog	Width	size	36	3
AttributeCatalog	RelName	name	0	1
AttributeCatalog	AttrName	name	32	2
AttributeCatalog	DomName	name	64	3
AttributeCatalog	Offset	size	96	4
AttributeCatalog	Position	count	100	5
DomainCatalog	DomName	name	0	1
DomainCatalog	ValueType	type	32	2
DomainCatalog	ValueSize	size	64	3

DomainCatalog において型を表している値 CHARACTER,INTEGER,TYPE は、それぞれドメインとして認められている値の種類を記述している。それぞれ文字列型、整数型、文字列型を意味する。

さらに、ユーザーの関係スキーマを保持することができる。例えば、2 章で示した例 1 の関係 CompanyA は RelationCatalog に次のように保持されている。

〈 CompanyA,4,80 〉

これは、関係 CompanyA のレコードサイズは 80 で 4 つの属性が含まれていることを意味する。それぞれの属性は、AttributeCatalog に次のように保持されている。

〈 CompanyA，都道府県，name，0，1 〉
〈 CompanyA，期日，count，32，2 〉
〈 CompanyA，売り上げ，count，36，3 〉
〈 CompanyA，製品，name，40，4 〉

⟨ CompanyA, 期日, count, 32, 2 ⟩ は、関係 CompanyA に属性として期日があり、それぞれのレコードで 32byte 目から始まり、二つ目の属性ということを意味する。さらに、この属性の領域は count である。この領域名は、DomainCatalog で定義されている。

実際のレコード構成を図示すると、次のように構成され質問処理実行時にアクセスされる。

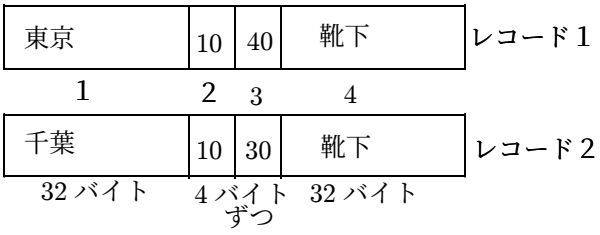

図 2 レコード構成

図 2 では、Company A の 1 つ目と 2 つ目のタプルに対応したレコード構成の図である。レコード 1 とレコード 2 との間に挟まれた 1,2,3,4 の数字はレコードのポジションを表す。レコード 2 の下の 32 バイトは、レコードを構成するフィールドの大きさに対応する。2,3 番目のポジションは、それぞれフィールド値の大きさが 4 バイトずつで合計 8 バイトである。

4 カプセル化と一時的スキーマカタログ

本章では、単純値のように複合値を処理する方法について述べる。複合値を単純値化するための手法として、カプセル化を用いる。また、メタ質問の処理実行により中間関係も出現する。その中間関係のスキーマ管理方法として、一時的スキーマカタログを導入して管理する。

4.1 カプセル化

メタオブジェクトの評価結果は、タプルに複合値が生じてしまう。その生じた集合は、カプセル化して格納する。さらに、評価後の集合から再帰的に評価がなされる場合が起きる。この場合も、カプセル化のカプセル化となる。

評価結果が単純値となった場合でも同様に、カプセル化し格納する。評価される対象の属性は、質問の処理実行時に判別されフラグがたち、区別されて処理される。フラグがたっているため、対応した属性のフィールド値は、カプセル化された値かどうかの判別のオーバーヘッドをなくすことにもなる。カプセル化したデータの名称は、Meta で始まりその関係名、属性名、属性値となる。カプセル化データの再評価では、評価される値はさらにカプセル化され名称にはその評価される値が新たに付け加わることになる。

例 2 カプセル化をした例を示す。例 1 の CompanyA で次の質問により製品を 2 回具体化する操作する。

```
Project[都道府県,$$製品](Company A)
```

製品の属性値である靴下に絞って説明する。靴下はメタオブジェクトであり、靴下の評価値である FabA や、FabB もメタオブジェクトであると仮定する。

FabA

規模	生産量
A class	100

FabB

規模	生産量
C class	40

1 回目の「$製品」による具体化により、関係 CompanyA の 1 つ目のタプルで、次のカプセル化データが生じる。

Meta.CompanyA. 製品. 靴下=

FabA	林
FabB	高橋

そのカプセル化されて得られた Meta.CompanyA. 製品. 靴下が、さらにメタオブジェクトとして評価される。FabA と FabB は、メタオブジェクトであるため具体化される。林と高橋は、メタオブジェクトではないただのデータであるためそのままとなる。具体化されたデータは、また新たにカプセル化データとしてそれぞれ構築される。その結果、Meta.CompanyA. 製品. 靴下は、次のようなデータを保持するカプセル化データとなり、さらに Meta.CompanyA. 製品. 靴下.FabA と Meta.CompanyA. 製品. 靴下.FabB というカプセル化データも存在することになる。

Meta.CompanyA. 靴下=

Meta.CompanA. 製品. 靴下.FabA	林
Meta.CompanA. 製品. 靴下.FabB	高橋

□

4.2 一時的スキーマカタログ

一時的スキーマカタログは、操作途中で生じた中間関係のスキーマカタログである。このスキーマカタログを参照してファイルのアクセスなどが行われる。メタ質問の処理実行により生成される中間関係のスキー

マは、この一時的スキーマカタログをもとに作られることになり、動的なスキーマの構成がこれにより可能となる。一時的スキーマカタログで全てを矛盾無く定義するのに必要なのは、3.3 章で示したコアセットとほぼ同じであるが、DomainCatalog で既に定義されている値は不要である。ただし、メタ質問により中間関係で保持されるデータは、カプセル化データをリンクする単純値のデータだけである。カプセル化データを保持する中間関係のスキーマで、カプセル化データを属性値とする属性の DomainName や ValueType、ValueSize は変更される。そのカプセル化データを属性値とする属性の DomainName は、`Capsul` とし、ValueType は `CHARACTER` である。作られたカプセル化データの消滅を考慮して、中間関係の参照数をカウントするようにする。すなわち、参照数がゼロとなった場合、その中間の関係の必要性は無くなり、削除可能となり削除される。ただし、RelationCatalog、DomainCatalog そして AttributeCatalog はその参照数を数えることはせず"*"と表す。中間関係の消滅は、参照数がゼロとなっとき以外に、メタ質問の実行完了時や異常動作の終了時などを考慮し、データベース操作そのものの終了時にも消滅する。

例えば、例 1 で示した次の質問で

```
Project[都道府県,$製品]
Select['Fab1, 井上' ∈ $製品](CompanyA)
```

で表される質問の選択処理実行後の中間関係のスキーマは、次のように RelationCatalog と DomainCatalog に保持されている。AttributeCatalog は、3.3 章で述べた ComapnyA の値と同じように保持されている。

RelationCatalog

RelName	Arity	Width	RefCount
RelationCatalog	4	44	*
AttributeCatalog	5	104	*
DomainCatalog	3	40	*
Temp.CompanyA	4	80	1
Meta.CompanyA. 製品. 鉛筆	2	64	1
Meta.CompanyA. 製品. 靴下	2	64	1

DomainCatalog

DomName	ValueType	ValueSize
Capsul	CHARACTER	32

選択操作により生成される一時的な中間関係を Temp.CompanyA とすると、もとの関係に比べタプル数は 1 つだけ減少する。この時、具体化操作されて生成されたカプセル化データの Arity は 2 であり、Width は 64 である。参照数は、次で処理される射影の射影属性に「$製品」が含まれているため 1 となる。もしも、次の選択属性に「$製品」が含まれていなかった場合は、参照されていないため、参照数はゼロとなり RelationCatalog には表れることなく生成されたカプセル化データは消滅する。ただし、もしも次のような質問であったならば

```
Select[都道府県 = 東京]
Project[都道府県,$製品](CompanyA)
```

射影操作を処理した後の、中間関係のスキーマは次の選択操作で「$製品」が存在しないが、射影操作の射影属性に「$製品」があるため参照数は 1 となる。射影属性終了時の Attribute カタログには、Temp.CompanyA を中間関係とすると次のように保持されている。

```
〈 Temp.CompanyA, 都道府県, name, 0, 1 〉
〈 Temp.CompanyA, $製品, Capsul, 32, 2 〉
```

関係の「$製品」の属性値が、評価された結果を持つカプセル化データであるため、AttrName は"$製品"に変わり、DomName は"Capsul"になる。

5 メタ質問実行

データベースの処理動作は、与えられた質問に対して構文解析がなされ、得られた解析木から論理的プラン、物理的プランが生成され、最終的に実行プランが生成される。メタ質問の最適化に関しては、[10][11][12] といった研究がありその処理実行があるが、本稿では効率の良い実行の観点から論じる。本章では実際に射影、選択、結果の表示の 3 つの場合に分け、メタ質問実行を論じる。

5.1 射影操作

`Project[Projection List](Relation)` からなる射影操作の場合、処理実行は次の手順からなる。

1. Open(Relation)
2. Create(Temp.Relation)
3. Compile(Projection List)
4. Save(Temp.Relation schema)
5. For(Each Record){
   Select(Projection Field Value);
   Evaluate(Field Value);
   }
6. Save(Temp.Relation)

始めに 1 で、関係をオープンする。次に 2 で、中間関係である Temp.Relation を生成する。次に射影属性は、スキーマカタログによりチェックされる。その際に、具体化操作を行う$が含まれているかどうかも判別される。もしも含まれていた場合、その関係の属性の領域を Capsul に変え、$の回数をカウントし評価回数を判断して回数を保持する。4 で Temp.Relation のスキーマを一時スキーマに保存する。5 では、与えられた Relation のレコードをそれぞれスキャンする。スキャンの際に、Capsul の領域の属性値が格納されたフィールドのフィールド値を評価していく。その評価の回数は、3 で保存されたカウント数と評価結果により得られた値が$を保持しているかに従い変化する。評価時にフィールド値は、次の条件分岐を行う。

```
If(field.value == MetaObject){
  if(Field.value == Relation Name)
     Project[*](Filed.value);
  else
     Project[Field.value](Relation)
}
```

この条件分岐により、実行された結果は 4.1 章で示した関係名として一時的に保存される。フィールド値は、この関係名に置き変えられる。また、さらに具体化操作により評価がなされカプセル化データが生成される場合、カプセル化データはスキーマカタログを参照判別し、1 に戻り再帰的に処理がなされる。最後に処理実行が終わり、6 でその中間の関係を保存する。

5.2 選択操作

`Select[Operand1 comparison Operand2](Relation)` からなる射影操作の場合、処理実行は射影操作と似ているが次の処理動作手順からなる。

1. Open(Relation)
2. Create(Temp.Relation)
3. Save(Temp.Relation schema)
4. CopyRecord(From Relation to Temp.Relation)
5. For(Each Record){
   Evaluate(Operands)
   If(Compare Evaluated(Operand1) with Evaluated(Operand2) == FALSE)
     Drop(Record)
   }
6. Save(Temp.Relation)

選択操作も、1,2,3 は射影操作と同様である。4 で、与えられた関係のレコードを全て一時的な中間関係 Temp.Relation にコピーを行う。次に、5 でコピーされた各レコードに対して与えられた各オペランドを評価し、レコードの取捨選択をする。この評価の時に、オペランドの$の個数をカウントし保持する。評価された結果は、一時的に射影と同様にカプセル化データとされ保存される。射影と同様に、評価回数は、$の個数と評価された結果の$の付加により評価回数が決定される。評価された結果は、`comparison` により指定された条件判別に用いられ、もしも条件に合致しない場合は、そのレコードが落とされる。最後に 6 で、残ったレコードと共に一時的な中間関係が保存される。

5.3 表示の操作

この章では、表示やメタ質問により与えられた結果を保存する場合を論じる。メタ質問の処理途中

で中間の関係は、カプセル化データと共に保持している。メタ質問の処理動作の終了時点では、複合値を考慮せず、カプセル化データを保持しない純粋な単純値を保持した関係に再構成される。そのため、カプセル化データのスキーマから属性の追加とタプルの追加が生じる。既に、`Project[Projection List](Relation)` による射影の処理が終了し、各カプセル化データが格納されていたとすると次の処理動作手順により表示がなされる。

1. Open(Relation)
2. Create(Temp.Relation)
3. Refer(Relation Attribute)
4. Refer(Capsul Domain Attribute)
5. Save(Temp.Relation Schema)
6. For(Each Record){
   Evaluate(Capsul Domain Field)
   Insert(Evaluated Value) }
7. Save(Temp.RelationB)
8. Print(Temp.RelationB)

1,2 は、射影や選択と同様で、3 では与えられた関係のスキーマを一時スキーマカタログから参照する。4 では、3 で参照したスキーマの領域に Capsule が存在した場合、属性値のであるカプセル化データのスキーマを参照する。それらの参照結果を基に、表示される関係の属性スキーマを 5 で決定しそのスキーマを保存する。次に 6 では、関係のレコードをスキャンしカプセル化データを保持するフィールド値を評価する。カプセル化データの属性スキーマにあわせたフィールド値を格納し、各レコードを再構成する。ただし、カプセル化データのレコード数が 2 以上の場合は、フィールド値が単純値とならない。同じレコードを Temp.Relation に追加し、そのレコードにカプセル化データを格納し、Temp.Relation のレコードを構成していく。7 で、全て単純値となる持つ関係を保存し 8 で表示の処理に移る。

5.4 実例を用いた処理動作例

例 1 で示した関係を用いて実際の処理例を示す。始めの処理動作は、選択操作が行われ、次に射影、最後に表示の操作となる。

関係 CompanyA が、データベースから呼び出される。次に一時的な中間関係 Temp.CompanyA が作られ CompanyA と同じ属性スキーマが保存される。CompanyA の各レコードが、Temp.CompanyA にそれぞれコピーされる。コピーされたレコードに対して、製品のフィールド値が評価され、カプセル化データが生成される。この際に評価される回数は、「$製品」というオペランドと評価された結果に$をもつ値が存在しないため評価される回数は 1 回となる。生成されたカプセル化データは、”Fab1, 井上”が含まれるかどうかという条件で判別され、条件に合致しなかったレコードが落とされる。ここで最後に残ったレコードを保存し選択操作は終了となる。ただし、4.2 章で述べたように、カプセル化データは次の操作で用いられるため参照数は 1 となりカプセル化データは消滅しない。この選択操作により、製品の属性値が靴下であるレコードが選択される。

次の射影操作では、選択操作で生成された関係を呼び出し、さらに異なる中間の関係を生成する。射影属性である都道府県と「$製品」により、具体化操作を行う属性である製品の領域を Capsul に変更しその変更した一時的な中間関係スキーマを保存する。各レコード処理では、求める射影属性に従ったフィールド値が取捨選択され、選択操作で生成されたカプセル化データの関係名に置き換えられる。この場合、置き換えられる関係名は、4.1 章のように Meta.CompanyA. 製品. 鉛筆が入る。

この操作の表示では、カプセル化データは一時的スキーマカタログによるスキーマ参照により属性が 2 つあることがわかる。表示する関係スキーマは属性数が 3 となりその属性情報が一時スキーマに保存される。各レコードのフィールド値は、Meta.CompanyA. 製品. 鉛筆であるので、レコードのフィールドが新たに追加され、フィールド値として Fab1 と井上がそれぞれ格納される。また、カプセル化データのレコー

ド数が 2 であることから、同じレコードが追加されされ、追加されたレコードのフィールド値にそれぞれ Fab2 と岡田がそれぞれはいりレコードが構成される。最後に、生成された一時的な中間関係が保存され表示操作が行われる。

6 結び

本研究では、関係代数をベースにしたメタ質問の動作原理とその実行方法や実行手順ができるように試みた。具体的には、一時的な中間の関係スキーマの管理に一時的スキーマカタログを導入しレコードの取捨選択や表示に用いた。また、各レコードのフィールド値には、単純値を保持させるために評価結果を保存した。次に、評価結果のカプセル化データを示す関係名をフィールド値とし、保存されたデータと対応させるという方法を用いて質問の処理実行の実現をした。その結果、これらの方法を用いることでメタ質問を処理実行できることができた。

参考文献

[1] Hector Garcia-Molina, Jeff Ullman, and Jennifer Widom :Database System Implementation

[2] Michael Stonebraker, Lawrence A.Rowe and Michael Hirohama:The Implementation Of Postgres (1990)

[3] Michael Stonebraker, Eric N. Hanson, Chin-Heng Hong: The Design of the Postgres Rules System. ICDE 1987: 365-374

[4] Harbron,TR.: ”File Systems-Structures and Algorithms, ”Prentice-Hall(1988)

[5] Equi4 Software: MetaKit http://www.equi4.com/metakit/

[6] 柏川伸悟, 松本渉, 三浦孝夫, 塩谷勇: データベースにおけるグループ化と集約関数の形式化 DEWS2001

[7] Shingo Kashikawa, Isamu Shioya, Takao Miura: On OLAP Operations Using Meta Data, *Applied Informatics (AI)* 2002

[8] Miura,T.,Matsumoto,W.: Managing Meta Objects for Design of Warehouse Data, proc.*DaWaK* (1999), pp.33-40

[9] Kohei WATANABE,Wataru MATSUMOTO, Takao MIURA,Isamu SHIOYA:Database Programming for Meta Objects, The 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI) 2000 and The 6th International Conference on Information Systems, Analysis and Synthesis (ISAS) 2000, pp.232-237

[10] Ogura Syogo, Shingo Kashikawa,Takao MiuraIsamu Shioya: Parallel Evaluattion of Meta Queries in Databases, IEEE PACRIM Conference on Communication, Computers and Signal Processing (PACRIM01), pp.707-710

[11] Wataru Matsumoto, Takao Miura, Shingo Kashikawa,Shioya Isamu: LogicalOptimization for Meta Queries, Computers and Their Applications (CATA-01)

[12] Wataru Matsumoto, Takao Miura, Shingo Kashikawa,Isamu Shioya: Optimizing Meta Queries in Databases, DATAKON Database Conference 2001, pp.257-266