データベースシステム 128－30
（２００２．７．１８）

SuperSQLのXML生成部の実装方式

慎 祥揆† 有澤 達也‡ 遠山 元道§

†慶應義塾大学大学院 理工学研究科 開放環境科学専攻
‡慶應義塾大学大学院 理工学研究科 開放環境科学専攻 / 慶應義塾大学 ITC 本部
§慶應義塾大学 理工学部 情報工学科
E-mail: † shin@db.ics.keio.ac.jp, ‡ ari@db.ics.keio.ac.jp, § toyama@ics.keio.ac.jp

データベース出版に利用されるSuperSQLでは様々なメディアに関する電子出版が出来る。SuperSQLでは構造化データの取得が全体の処理時間に大きく影響するため、この処理時間の高速化の方法について研究を続けている。本研究ではSuperSQL処理系の中でXML生成系に注目し、新しいXML生成処理アルゴリズムを提案している。SuperSQLの質問処理において、このアルゴリズムを用いることによりSQLによるデータの取得からグルーピング操作までの一連の処理時間を短縮する効果的なシステムを提案する。

キーワード : SuperSQL 、 データベース出版 、 XML

The Implementation proposal of the XML generation part of SuperSQL

Sang-gyu SHIN † Tatsuya Arisawa ‡ Motomichi TOYAMA §

†‡School of Science for OPEN and Environmental Systems,
Faculty of Science and Technology, Keio University.
‡Keio University ITC
§Department of Information and Computer Science, Faculty of Science and Technology,
Keio University.
E-mail : † shin@db.ics.keio.ac.jp ‡ ari@db.ics.keio.ac.jp § toyama@ics.keio.ac.jp

The electronic publishing about various media is made in SuperSQL used for database publication. In SuperSQL, since acquisition of structure-ized data influences greatly at the whole processing time, research is continued about the method of improvement in the speed of this processing time. In this research, new XML generation processing algorithm is proposed in a SuperSQL processing system paying attention to an XML generation system. In question processing of SuperSQL, the effective system which shortens a series of processing time from acquisition of the data based on SQL to grouping operation is proposed by using this algorithm.

keyword : SuperSQL , Database Publishing , XML

1 はじめに

SuperSQL[1][2] は、データベース出版を目的とした問い合わせシステムである。SuperSQL では、取得する属性間に階層構造に関する情報を付加することによって関係データベースからの問い合わせの結果をグルーピングし構造化する。また、同時に記述されたメディア指定や次元情報、装飾情報により、この構造化データはHTML、XML 等の様々な媒体用の応用データに変換される。

この構造化データは属性値の等しいものをまとめた簡潔な状態で表現される。しかしながら、関係データベースからの検索結果はもともとフラットなものであって、構造化データを生成するために必要な処理は大きなものとなる。

そこでSuperSQL 処理系の質問処理において、結果のデータ構造を示すグルーピング木を利用することで$ORDERBY$ 節を付加した複数のSQL に質問文を分割し結果を組み合わせるといった手法を提案する。その手法を実装することにより、内部中間結果の抑制およびグルーピング操作の簡略化による構造化データ取得のコストを低減することを目指す研究 [4] やハッシュテーブルを作成し構造生成時間を速くする研究なとがされている。

しかし、いままでのSuperSQL 処理系はLisp 版をそのままJava にマッピングしたため、Java のオブジェクトプログラミング機能など様々な利点を有効に利用することができなかったため、本稿ではまず、SuperSQL 処理系のなかでXML 処理系を変更し、その効果を比較する。

その新しいXML 処理系を設計する際はJava のオブジェクト概念とXQuery を支援することを基本に作成した。そこで本研究ではSuperSQL のXML 出力アルゴリズムについて、これまでとは異なるアルゴリズムを提案する。以下、2 章でまずSuperSQL について述べ、3 章で提案するSQL アルゴリズムについて述べ、4 章で本研究で提案する構造生成アルゴリズムについて説明し、5 章でシステムの流れに関して述べる。最後に 6 章で結論を述べる。

2 SuperSQL

2.1 SuperSQL の構文

SuperSQL の質問文は、SQL のSELECT 節を拡張し‘GENERATE＜メディア指定＞＜TFE＞’という構文を持つGENERATE 節で置き換えたものである。＜メディア指定＞にはXML,HTML,LATEX 等の出力したい媒体の指定ができ、＜TFE＞には出力のレイアウトを記述する。TFE は、表の構造とレイアウトを規定する式の一種であり、オペランドと演算子から構成されている。オペランドはSQL のターゲットリストの要素であり、演算子は結合子及び反復子からなる。結合子は属性及びTFE をレイアウト上で並べるために用いられ、そのレイアウト方向を示す為に「,」(縦方向)、「!」(横方向)、「例えば、表 1、表 2、表 3、に示した 3 つのリレーションがあるとき、以下のように各支店各売場毎に従業員とその給料を一覧するというSuperSQL 質問文を、SuperSQL 処理系に入力する。

```
GENERATE XML [ S.city, [ D.name, [ E.name,
    E.salary ] ! ] ! ] !
FROM  STORE S, DEPT D, EMPLOYEE E
WHERE  S.id = D.store
        and  D.manager = E.manager
```

表 1: リレーション STORE(店)

属性名	内容
id	支店番号
city	支店名
address	住所

表 2: リレーション DEPT(売場)

属性名	内容
id	売場番号
name	売場名
store	売場のある支店番号
manager	売場のマネージャーの従業員番号

表 3: リレーション EMPLOYEE(従業員)

属性名	内容
id	従業員番号
name	従業員名
salary	従業員の給料
manager	従業員のマネージャーの従業員番号

```
01:  <city>Tokyo
02:    <name>huku
03:      <name>Ryuichi Tanaka </name>
04:      <salary>4000 </salary>
05:      <name>Mitsunobu Tada</name>
06:      <salary>15000 </salary>
07:    </name>
08:    <name>car
09:      <name>Humiki Kawasaki</name>
10:      <salary>17000 </salary>
11:      <name>Satoshi Asai </name>
12:      <salary>14000 </salary>
13:    </name>
14:    ......
15:  </city>
```

図 1: 結果 XML ビュー

2.2 SuperSQL 処理系

関係データベースから XML に出版するためには初めに関係データベースからの問い合わせのフラットな検索結果を XML の木構造に変換することが必要である。

SuperSQL 処理系では、与えられた質問文に対して図 2 に示すように「構文解析部」において構造情報と装飾情報を抽出し、「データベースインタフェース部」において分解された情報をもとに SQL を関係データベースに発行する。そして、その検索結果を「データ構造化部」において構造情報に従ってグルーピングを行い、「レイアウト編集部」において装飾情報とメディア指定にあわせて、目的の応用データを生成する。[4] ではデータ構造化部で特に着目し、SuperSQL の質問文処理に対して、ORDERBY 節を SQL へ自動的に付加することにより、グルーピング時に必要なソート処理を、データベースからの検索結果を得る時点で行うといった手法を提案した。

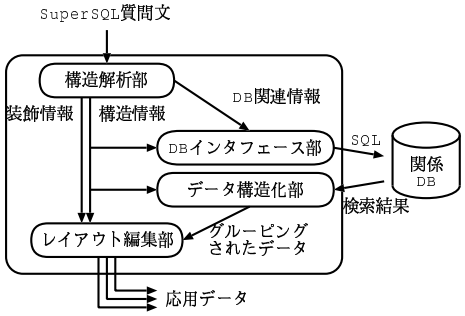

図 2: SuperSQLO 処理系処理図

例えば、前の SuperSQL 質問文では、次のような SQL 文

```
SELECT S.city, D.name, E.name, E.Salary
FROM  STORE S, DEPT D, EMPLOYEE E
WHERE  S.id = D.store
          and  D.manager = E.manager
```

と、レイアウト式

```
(g2 (c1 1 (g2 ( c1 2 (g2 ( c1 3 4 ))))))
```

に分解される。

装飾のレイアウト情報に関しては、例えば、SuperSQL 質問文の Generate 句に

```
[S.city@{tag=cityName},
   [D.name@{tag=DepartName},
      [E.name@{tag=EmployeeName}, E.salary
       ]!@{tag=MEIBO}
    ]!@{tag=DEPARTMENT}
 ]!@{tag=STORE}
```

のように修飾子がついていたとする。このとき、D.name@{tag = DepartName} の@{tag = DepartName} は D.name という基本項目に対する装飾を表しており、また、[E.name, E.salary]!の@{tag = MEIBO} は [E.name, E.salary]!という反復子を用

いたターゲットリストに対する装飾である。また装飾情報がないデータ項目や反復項目には、null として情報が格納される。これらから、

((tag STORE)((tag cityName) null ((tag DEPARTMENT) (null (tag DepartName)((tag MEIBO) (null (tag EmployeeName) null))))))

のようなリストが生成される。このリストの 2 番目の要素 (tag cityName) は、2 番目の基本項目に対する装飾情報、後者のリストの 9 番目の要素 ((tag EmployeeName)) は、3 番目から 4 番目にかけての基本項目すなわち E.name、E.salary の反復 (群化された項目) に対する装飾情報という意味である。このような 2 つのリストによって、全ての装飾演算子による装飾情報を表すことが出来る。

```
01:  <STORE>
02:    <cityName>Tokyo
03:      <DEPARTMENT>
04:        <DepartName>huku
05:          <MEIBO>
06:            <EmployeeName>Tanaka </name>
07:            <salary>4000 </salary>
08:        ......
09:          </MEIBO>
10:        ......
11:        </DepartName>
12:        ......
13:      </DEPARTMENT>
14:      ......
15:    </cityName>
16:      ......
17:  </STORE>
```

図 3: 結果 XML ビュー

2.3 従来の処理アルゴリズム

前節で述べた処理方法のため以下のようなアルゴリズムで作成される。

表 4: 将来のメソッドの一部分

メソッド名	用途
le	構造情報を保時する
le2	装飾情報を保時する
car()	始めの要素を返す (Lisp car)
cdr()	最初の要素を取り除いた残りのリストを返す
createC1	連結演算子を処理する
createG2	反復演算を処理する
media	出力媒体の指定を保持 (Lisp cdr)

```
01:  XMLC1(){
02:    manager(creater) を生成
03:    if(le2.car()!= null) {
04:      タグ名前 =
05:        タグ名前を取る ((List)le2.car())
06:      element を生成
07:    }
08:    if(le が属性を持っている) {
09:      属性を追加
10:      element を生成
11:    }
12:    return C1
13:  }
```

図 4: 連結演算子 , (c1)

```
01:  XMLG2(){
02:    manager(creater) を生成
03:    if(le2.car()!= null) {
04:      String result =
05:        c1 に作成された element を
06:        List 情報によってグループ化して
07:        element を生成
08:    }
09:    return g2
10:  }
```

図 5: 反復演算子 [ ]! (g2)

3 装飾情報(タグ)処理

いままでタグはレイアウト編集部によって抽出された装飾演算子によるレイアウト情報から付けている。このシステムではSuperSQL をSQL と装飾情報と構造情報に分ける際、基本的なタグ情報を処理することを提案する。

```
01:  RegularExp(){
02:    //子共を持っている element
03:    Pattern P1
04:    //自分のタグ情報
05:    Pattern P2
06:    while(result){
07:      //子共を持っている element を抽出
08:      if(pattern が子共を持っている element)
09:        情報と貯蔵
10:      //自分のタグ情報
11:      if(pattern が自分のタグ情報)
12:        as 節に変える
13:    }
14:  }
```

図 6: 正規表現アルゴリズム

```
01:  DB 結果処理 (){
02:    //結果を読む
03:    結果 = 関係データベースの問い合わせ結果
04:    //結果から基本 element を作成
05:    while(結果){
06:      //各 column の結果と column の名前から
07:      //基本 element らを作成
08:      createElement(column 名前,
09:                    column の結果)
10:    }
11:  }
```

図 7: 基本 element 作成アルゴリズム

すなわち、パーザする際、$SELECT\ \ AS$ 節を利用し、タグ情報をカラムとしてデータベースに入れる。まず、!@{$tag =''$ タグ名''} 装飾演算子は子供をもっているタグと考えることが出来る。これについてはグループ情報として持って残り @{$tag =\ \ $} については AS に変える。それで、前の例の質問文は以下のアルゴリズムによってこのように変えられる。

```
SELECT  S.city AS cityName,
        D.name AS DepartName,
        E.name AS Employee
        E.salary
```

関係データベースから結果を読む際にカラムの名前を読んでタグを付けることが出来る。タグが定義されていない場合は基本カラムの名前がタグになる。これによってリスト情報を簡単化出来る。

4 構造作成処理

4.1 element 処理

従来の XML 文書の作成は前節に並べたようなSQL を関係データベースに発行して、その検索結果を「データ構造化部」において構造情報に従ってグルーピングを行い、「レイアウト編集部」において装飾情報とメディア指定にあわせて、目的の応用データを生成する。XML はSuperSQL が持っている演算子のなかでC1(連結演算子,) と g2(反復演算子[ ]!) をグループのため利用している。

本稿ではelement や属性や name space などを処理するため、新しいメソッドを提案している。従来のSuperSQL の XML 処理部はLisp 版をそのままJava にマップしているので、今回、XML 処理エンジンの設計ではJava の機能を有効に利用し、ユーザ問い合わせ言語にある XQuery を支援する際の処理も考慮して設計する必要がある。

そのため、まず、element を作成するオブジェクトを作成した。このオブジェクトは表5 のようなメソッドを持っている。図8 はelement を作成する基本アルゴリズムである。elemet 作成部分はタグの名前、属性、Namespace を処理する必要があるし、コンテンツを含む処理が必要である。

表 5: createElement クラスのメソッド

メソッド名	用途
setElement(str)	新しい element(str) を追加する
setAttribute(str, str)	element の属性を追加する
setNamespace(str, str)	element の namespace を追加する
getElement(target)	ターゲット element を抽出する
delElement(target)	ターゲット element を取り消す
replaceElement(target, source)	ターゲット element をソース element に変える

```
01:  createElement(){
02:    setElement(String) {
03:      新しい名前を作成結果構造に入れる
04:      return name;
05:    }
06:    setAttribute(String, String) {
07:      createAttribute At =
08:        createAttribute.set(name, value)
09:      return At;
10:    }
11:    setNamespace(String, String) {
12:      createNamespace Ns =
13:        createNamespace.set(String, String)
14:    }
15:    addText() {
16:      element にコンテンツを含まれる
17:    }
18:    createEle() {
19:      子供 element を含まれる
20:    }
18:    ......
19:  }
```

図 8: createElement アルゴリズム

4.2 属性処理

SuperSQL 質問文は@{att = “属性名”} のように記述する属性に関する装飾演算を持っている。それに対してある element に属性を付く attribute オブジェクトを作成した。全ての element が属性を持っていることはなく、属性を持っている element とこの属性情報のみを貯蔵し処理する。属性のオブジェクトは表 6 に述べる。

表 6: createAttribute クラスのメソッド

メソッド名	用途
setAttribute(str, str)	element の属性を追加する
getAttribute(target)	ターゲット属性をを抽出する
delAttribute(target)	ターゲット属性を抽出を取り消す
replaceAttribute(target, source)	ターゲット属性をソース属性に変える

```
01:  createAttribute(){
02:    名前と値をもらう
03:    setAttribute = getNamespace(prefix)
04:      Namespace の prefix があるとき
05:      element の属性として処理
06:    }
07:    setAttribute(String, String) {
08:      At = createAttribute.set(name, value
09:      return At
10:    }
11:    delAtt(String, String) {
12:      while(!null)
13:        当たる属性を探す
14:        属性を取り消す
15:    }
16:  ......
17:  }
```

図 9: createAttribute アルゴリズム

4.3 Namespace 処理

表7 に Namespace のメソッドを述べる。Namespace の処理にはまず、prefix と uri という情報から作る。例えば、prefix は $< db : column >$ の db である。すなわち、$prefix$ を利用し $< xsl : templatematch ='' ...'' >$ のように XSLT 表現作成ができる。この処理には element に prefix を追加するアルゴリズムが必要である。それと、Namespace は prefix を取り消すとき element に入っている全てを検索し、消す必要がある。

表 7: createNamespace クラスのメソッド

メソッド名	用途
getNamespace(target)	ターゲット Namespace を抽出する
delNamespace(target)	ターゲット Namespace を取り消す
repNamespace(target)	ターゲット Namespace を変える

```
01:  createNamespace(){
02:    setBase(基本 Namespace)
03:    //XML と W3 の Homepage である
04:      return name;
05:    prefix と uri を namespace として取る
06:    ns = creNamespace(preficx, uri) {
07:      createAttribute At =
08:        createAttribute.set(name, value)
09:    setNS() {
10:      return getNS(prefix)
11:    }
12:      createNamespace Ns =
13:        createNamespace.set(String, String)
14:    }
15:    delNS() {
16:      while(!null) {
17:        NS(prefix) を持っている element から
18:        全ての prefix を取り消す
19:      } }
20:    ......
21:  }
```

図 10: createNamespace アルゴリズム

5 システムの流れ

全体的なシステムの流れを説明する。まず、図11のようにSuperSQL の質問部はSQL に変更される。図12 はタグ情報を持っている構造情報である。図11 の質問部によって図12 のような木構造が作成される。図11 と図12 を比べて見れば (数1,2,3,4) との結果にとのタグが付けているか分かる。このシステムでは基本的なタグは関係データベースから結果タプルを読むとき付けられているのでこの木構造はタグが付いているものである。それを図13 の子孫関係テーブルを用いて element として作成する。つまり、処理されている子供 element は親 element にタグが付けているストリングとして含まれる。図12 の A は子孫関係を表す。この関係を図13 で見れば、MEIBO は3 と4 番目の結果タプルを子供として持っているし、DEPARTMENT は2 番目と複数 (*で表現されている) の MEIBO を子供に持っている。タグの名前は AS 節に作成されたカラムの名前を結果を読む際読んで作成する。タグの名前が定義されていない場合は基本カラム名前がタグの名前になる。$ORDERBY$ 節が $DISTINCT$ による問い合わせ文プランは本稿の扱う範囲からはずれるためここでは議論しない。

6 まとめ

本稿では SuperSQL における XML 生成系に注目し、新しい処理系に関するアルゴリズムを提案た。これは SuperSQL 処理系を変更し、高速化することと XQuery を支援するための設計を志向している。又、新しい関数なとを作成する際、その機能追加を簡単に行うため、オブジェクトを利用出来るシステムとして設計する。前のシステムと比べて新しいエンジンを設計するときの応用するため、様々な設計に対して実験をする予定である。

今後は、このアルゴリズムの応用し、SuperSQL

```
GENERATE XML [S.city@{tag=cityName},
                [D.name@{tag=DepartName},
                  [E.name@{tag=EmployeeName}, E.salary
                  ]!@{tag=MEIBO}
                ]!@{tag=DEPARTMENT}
              ]!@{tag=STORE}
FROM STORE S, DEPT D, EMPLOYEE E
WHERE S.id = D.store and D.manager = E.manager
```

↓

```
SELECT S.city AS cityName, 1
       D.name AS DepartName, 2
       E.name AS EmployeeName, 3
       E.salary 4
FROM STORE S, DEPT D, EMPLOYEE E
WHERE S.id = D.store and D.manager = E.manager
GROUP BY D.name
```

図 11: 質問部の変更

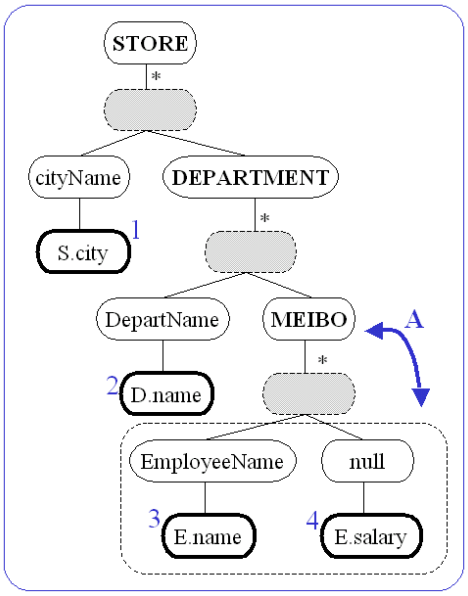

図 12: 構造情報

MEIBO (3, 4)
DEPARTMENT (2, MEIBO*)
STORE (1, DEPARTMENT*)

図 13: 子孫関係テーブル

の他のメディアに対するエンジンの設計についても適用することを考えている。それと、最適化や外部関数の処理なとの機能の拡張についての研究を勧める予定である。

参考文献

[1] SuperSQL: http://ssql.db.ics.keio.ac.jp/

[2] Motomichi Toyama, "SuperSQL: An extended SQL for Database Publishing and Presentation," *ACM SIGMOD '98* , pp. 584-586, 1998

[3] Michael carey, et. al. : Efficiently Publishing Relational Data as XML Documents, *Proc, of the 26th VLDB Conference,* pp. 67-76, 2000.

[4] 有澤 達也, 遠山 元道 : SuperSQL 処理系におけるグルーピング操作の効率的な実装, DEWS 2001

[5] 赤堀正剛, 有澤達也, 遠山元道: SuperSQL による関係データベースと XML データの統合利用, 情報処理学会論文誌:データベース, Vol. 42, No. SIG8(TOD 10), pp. 66–95, 2001.

[6] Igor Tatarinov, et. al. : Updating XML, *ACM SIGMOD '01,* 2001.

[7] Java: http://java.sun.com/

[8] S. Abiteboul, et. al. : The Lorel query language for semistructured data, *In Proceedings of International Journal on Digital Libraries,* volume 1(1), pp. 68-88, Aril 1997.

[9] J. Shanmugasundaram, et. al. : Efficiently publishing relational data as XML documents, *In VLDB '00,* 2000.

[10] eXcelon: The XML application development environment, http://www.odi.com/excelon/main.htm

[11] M. J. Carey, et. al. : XPERANTO : Publiching object-relational data as XML, *In ACM SIGMOD WebDB Workshop '00,* 2000.