セキュリティを考慮した RDB の XML ビュー機構の検討

品川　徳秀[†,††]　　北川　博之[††,†††]

† 科学技術振興機構 戦略的創造研究推進事業
†† 筑波大学 システム情報工学研究科　〒 305–8573 茨城県つくば市天王台 1–1–1
††† 筑波大学 計算科学研究センター　〒 305–8577 茨城県つくば市天王台 1–1–1
E-mail: †siena@kde.cs.tsukuba.ac.jp, ††kitagawa@cs.tsukuba.ac.jp

あらまし　近年、XML でのデータ統合が広く利用されるようになってきている。このデータ統合において RDB を利用するために、XML ビューを提供する XML 出版技術が開発されてきた。一方、XML データ処理においては、ネットワークを通じてデータ交換が行なわれ、複数のサービス提供者を通じて一連の処理が行なわれる事もあり、アクセス制御や部分暗号化、電子署名等によるセキュリティの確保が重要である。本研究では、セキュリティを考慮した効率の良い XML ビュー機構の実現を目指す。XPERANTO 方式をベースとし、XQuery で定義された XML ビューに対して定義されたセキュリティポリシーを XML 出版過程において適用する方式を検討する。
キーワード　XML ビュー, RDB, セキュリティ, アクセス制御

Proposal of An XML View Mechanism Considering Security

Norihide SHINAGAWA[†,††] and Hiroyuki KITAGAWA[††,†††]

† Core Research for Evolutional Science and Technology, Japan Science and Technology Agency
†† Graduate School of Systems and Information Engineering, University of Tsukuba
1–1–1 Tennodai, Tsukuba, Ibaraki, 305–8573 Japan
††† Center for Computational Sciences, University of Tsukuba
1–1–1 Tennodai, Tsukuba, Ibaraki, 305–8573 Japan
E-mail: †siena@kde.cs.tsukuba.ac.jp, ††kitagawa@cs.tsukuba.ac.jp

Abstract　XML-based data integration has been used widely. Schemes to publish XML data from RDBs are important to use RDBs in such data integration frameworks, and they have been developed. In actual XML data processing, data are exchanged through networks, and they are processed by one or more service providers. Therefore, security mechanisms such as access control, data encryption, and desital signature, are also important. In this paper, we discuss a scheme to apply efficiently security policies to XML views over RDBs provided by XML publishing. The basis of XML publishing is the XPERANTO approach. Security policies are represented as security queries, and then applied in the cource of view query processing.
Key words　XML View, RDB, Security, Access control

1. はじめに

現在、XML は、ネットワークサービスやソフトウェアコンポーネント間のデータ表現や交換、統合処理において重要な技術の一つとなっている。一方で、多くの実データは、技術が高度に成熟され、多くの実績が培われてきた RDBMS で管理されている。このため、XML 技術を中心としたシステムにおいても、RDB を利用するための機構が必要である。その一つとして RDB の XML ビューを提供する XML 出版と呼ばれる技術が有用であり、これまで様々な研究が行なわれてきた。特に代表的な XML 出版の研究として、XPERANTO [1] や SilkRoute [2] 等が知られている。

また、ネットワーク利用の一般化により、ネットワークを介して利用可能なシステムが構築され、様々なサービスが行なわれるようになっている。これに伴い、データの実利用の場面では、アクセス制御や暗号化、電子署名等のセキュリティ機構が重要になっており、XML においても標準化が行なわれている [3]〜[5]。一般に、ネットワークを介したオープンシステムでは、異なる組織によって運営される複数のサブシステムを連携させて一連の処理が行なわれる。また、時として、その参加シ

department		
did	dname	address
D001	人事	addr-1
D002	経理	addr-2
D002	研究開発	addr-3
D003	営業	addr-4
...	...	...

employee				
pid	pname	did	jobcode	salarycode
P0001	Jack	D001	4 (manager)	M-13
P0002	Michael	D003	3 (vice manager)	V-10
P0003	Robert	D001	1 (regular)	R-11
P0004	Thomas	D002	1 (regular)	R-08
...	...	...	...	...

図 1　リレーション例
Fig. 1　Relational tables.

```
<db>
  <department>
    <row>
      <did>D001</did>
      <dname>
          Personnel</dname>
      <address>
          addr-1</address>
    </row>
    ...
  </department>
  <empleyee>
    <row>
      <pid>P0001</pid>
      <pname>Jorn</pname>
      <did>D001</did>
      <jobcode>4</jobcode>
      <salarycode>
          M-13</salarycode>
    </row>
    ...
  </empleyee>
</db>
```

図 2　デフォルト XML ビュー
Fig. 2　Default XML view.

```
create view employee_list as {
  <list>
    for $dep in view("default")/
                  db/department/row
    return
      <department did="$dep/did">
        <dname>$dep/dname</dname>
        <address>$dep/address</address>
        <members>
          for $emp in view("default")/db/
               employee/row[did=$dep/did]
          return
            <member pid="$emp/pid">
              <pname>$emp/pname</pname>
              <jobcode>$emp/jobcode
                  </jobcode>
              <salarycode>$emp/salarycode
                  </salarycode>
            </member>
        </members>
      </department>
  </list>
}
```

図 3　ユーザ XML ビュー定義
Fig. 3　View definition.

```
<list>
  <department did="D001">
    <dname>
        Personnel</dname>
    <address>
        addr-1</address>
    <members>
      <member pid="P0001">
        <pname>Jorn</pname>
        <jobcode>4</jobcode>
        <salarycode>
           M-13</salarycode>
      </member>
      ...
    </members>
  </department>
  ...
</list>
```

図 4　ユーザ XML ビュー
Fig. 4　Defined XML view.

ステムは動的に候補群から選択されるため、事前に限定できない事もある。それゆえ、多くの参加者が共通に利用可能で、柔軟かつ標準化されたセキュリティ機構を利用する事が望ましい。

従来の XML 出版機構ではセキュリティ機構を統合していないため、より上位の層でセキュリティポリシーを適用する必要がある。このため、まず XML 出版層で完全な XML ビューを実体化し、上位層のミドルウェアでアクセス制御が行なわれ、その結果に対して更に XML 暗号化や電子署名といった処理が適用される。しかし、このように段階的に XML の処理する方式では、XML のパースとシリアライズ、探索を繰り返し行なう必要があり、非効率的である。

本研究の目的は、XML 出版において、XML ビューに対して定義されたセキュリティポリシーを XML 出版過程において効率良く適用する事にある。本稿では、XML 出版方式として XPERANTO を基本機構とする。セキュリティ機構としては、XACML のようなポリシーベースのアクセス制御方式と、XML Encryption 及び XML Signature による暗号化・電子署名を想定し、これらの適用について検討を行なう。

表 1　XQGM 演算子
Table 1　XQGM operators.

演算子	機能
table	リレーションの指定
project	射影 (計算による結果の形成)
select	制約によるタプルの選択
join	複数入力の結合
groupby	集約関数によるグループング
orderby	属性値によるソート
union	複数入力の和集合
unnest	ネストを展開したリスト
view	ビューの指定
function	XQuery 関数の指定

2. XPERANTO

2.1 概要及び例

XPERANTO は RDB の XML ビューを実現するミドルウェアシステムである。XPERANTO では、RDB を直接的に表現

表 2　XML 関数

Table 2　XML functions.

XML 関数	機能	出現可能演算子	SQL2003 対応関数
cr8Elem(Tag,Atts,Clist)	要素名 Tag、属性リスト Atts、内容 Clist の要素生成	project	xmlelement
cr8AttList(A_1,$\cdots$,A_n)	属性群から属性リストを生成	project	xmlattriute
cr8Att(Name,Val)	属性名 Name、属性値 Val の属性生成	project	xmlattribute
cr8XMLFragList(C_1,$\cdots$,C_n)	要素内容群から XML 断片リストを生成	project	xmlforest, xmlconcat
aggXMLFrags(C)	XML 断片リストを生成する集約関数の適用	groupby	xmlagg
getTagName(Elem)	要素名の取得	project, select	
getAttributes(Elem)	属性リストの取得	project, select	
getContents(Elem)	要素内容の XML 断片リストを取得	project, select	
getAttName(Att)	属性名の取得	project, select	
getAttValue(Att)	属性値の取得	project, select	
isElement(E)	要素か否かを判定	select	
isText(T)	テキストか否かを判定	select	
unnest(List)	ネストを展開	unnest	

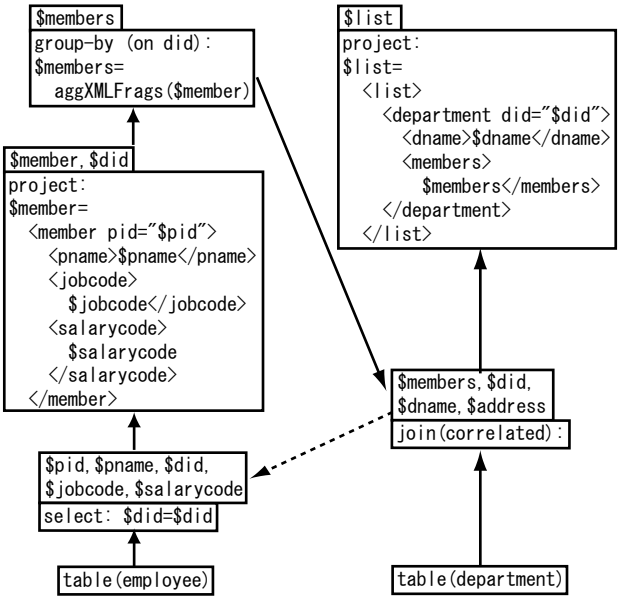


図 5　XGML グラフ

Fig. 5　XQGM graph.

した平坦かつ一様な構造のデフォルト XML ビューがシステムによって自動的に定義される。ユーザ XML ビューはデフォルト XML ビューに対する XQuery 問合せによって定義する形式を取る。このビューの生成は、後述するように RDBMS における問合せ能力を有効に活用し、不足する機能をミドルウェア側で処理する事で実現される。

図 1 に示す部門と社員の RDB を例とし、そのデフォルト XML ビューを図 2 に示す。更に、これに対して、部門別の構成員リストに再構成したユーザ XML ビュー定義が、図 3 の XQuery 問合せである。これにより導出される XML ビューの一部を図 4 に例示する。

2.2　問合せ処理

XPERANTO では、与えられた問合せを次のように処理する。その際、部分問合せを可能な限り RDBMS 側で処理し、RDBMS 側で処理できない残りの問合せを XPERANTO ミドルウェアで処理する。

表 3　XML 関数合成規則

Table 3　XML function composition rules.

Function	Composition Target	Result
getTagName	cr8Elem(Tag,Atts,Clist)	Tag
getAttributes	cr8Elem(Tag,Atts,Clist)	Atts
getContents	cr8Elem(Tag,Atts,Clist)	Clist
getAttName	cr8Att(Name,Val)	Name
getAttValue	cr8Att(Name,Val)	Val
isElement	cr8Elem(Tag,Atts,Clist)	True
isElement	except for cr8Elem	False
isText	PCDATA	True
isText	except for PCDATA	False
unnest	aggXMLFrags(C)	C
unnest	cr8XMLFragList(C_1,$\cdots$,C_n)	$C_1 \cup \cdots \cup C_n$
unnest	cr8AttList(A_1,$\cdots$,A_n)	$A_1 \cup \cdots \cup A_n$

（1）**問合せ解析**

ビュー問合せとビューに対するユーザ問合せを解析し、表 1 に示す演算子を用いて、内部表現モデル **XQGM (XML Query Graph Model)** のグラフとして表現する

（2）**問合せ変換**

XQGM グラフを再構成し、SQL 問合せとして RDBMS によって処理可能な部分問合せと、それ以外のミドルウェアで処理しなければならない部分問合せに分離する

（3）**問合せ処理**

分離された部分問合せを RDBMS で処理した結果に対し、問合せの残りをミドルウェアによって処理して最終的な XML データを生成し、ユーザに提供する

図 5 に、図 3 のビュー問合せを表すビュー定義 XQGM グラフを示す。各ノードは、表 の XQGM 演算に対応し、上部の枠内には出力の属性リストが示されている。実線矢印はノード間の入出力関係を、点線矢印は結合における依存関係を表す。例えば、右側中央の `join(correlated)` と書かれたノードは、`employee` タプルを対応する `department` タプルに結合する事

を意味している。

演算子中では、表 2 に示す XML 関数が利用可能である。尚、XPERANTO 発表時には、一般にこれらを RDBMS で処理できなかったが、SQL 2003 において XML 対応が行なわれた事により、SQL 中で同様の関数が利用可能となった [6]。本稿では、SQL 2003 を前提とし、SQL 2003 に相当する関数が存在する XML 関数は、RDBMS 側で処理可能とする。また、XML 関数の冗長な組合せは、表 3 に挙げた合成規則により、合成結果と等価な XML 関数に置き換える事で除去される。

XQGM グラフは、以上の演算子および XML 関数で表現される。RDBMS で処理可能な処理をプッシュダウンし、それ以外の処理をプルアップする事で、SQL で表現可能な部分問合せが分離される。これにより、RDBMS の能力を有効活用しつつ、XML 出版が実現される。

3. セキュリティ機構

3.1 アクセス制御ポリシー

XML を交換形式としたネットワークを介したシステム連携では、異なる組織によって運営される複数のサービスを通じて一連の処理が行なわれたり、時としてその参加者を事前に限定できなかったり、アクセス制御の方針が複数の組織で決定・運用されたりする。それゆえ、事前に定義された固定的なビューを既定のロールに対して提供する方式のものより、動的にアクセス者の持つ属性に応じて選択されるポリシー群を収集・適用したビューを提供する方式のものが有用であると考え、本稿では後者のアクセス制御方式を想定する。

例えば、このようなポリシーベースのアクセス制御記述言語として、OASIS によって策定中の XACML がある。詳細は割愛するが、XACML のセキュリティモデルでは、**主体**であるデータの利用者は認証システムによって認証された後、利用者自身の持つ属性に基づいて、**対象**のデータに関する適切なアクセス制御ポリシー群が選択される。これらポリシー群はデータ提供システムにおいて適用され、その処理結果が利用者に提供される。また、その際にデータ提供システムにおいて果たされねばならない**責務**も指定可能となっている。

本稿では、XML データ中のアクセス制御対象を XPath で指定し、利用者の属性に応じてアクセスの可否、およびデータ提供時の責務として暗号化・電子署名の指示を与える。例として、表 4 に示すポリシーを適用する。

3.2 セキュリティ問合せ

ポリシーは XML ビューに対して適用され、その結果がユーザ問合せの対象となる。XPERANTO では、ビュー問合せとユーザ問合せを合成して処理を効率化する。XML 出版過程において、ビュー問合せおよびユーザ問合せと統一的に処理可能とするために、適用されるポリシーを問合せとして表現する。これを、**セキュリティ問合せ**と呼ぶ。セキュリティ問合せを表現するため、幾つかの演算子を導入し、その処理の具体的な方法について説明を行なう。

ここで、XML ビューにポリシーを適用した結果をユーザ問合せと合成されるべき XML ビューとみなせば、これとユーザ問合せとの合成は XPERANTO における通常の問合せ合成と同様に実現可能である。このため、以下では XML ビュー問合せとセキュリティポリシーの合成のみに注目して説明を行なう。

表 4 アクセス制御ポリシーとセキュリティ問合せ (SQ)
Table 4 Access control policies and their security queries.

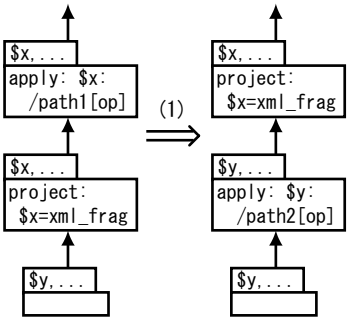
Policy 1	
説明	人事部の者のみが従業員の給与を知る事ができる
対象	`/list/department/members/member/salarycode`
主体	人事部に所属しない
可否	拒否
SQ	$\texttt{Apply}_{\texttt{list:/list/department/members/member}}$ $[\texttt{Proj}_{\texttt{not(salarycode)}}]$
Policy 2	
説明	従業員以外には役職を持つ者のみの名簿しか読めない
対象	`/list/department/members/member[jobcode<2]`
主体	従業員でない
可否	拒否
SQ	$\texttt{Apply}_{\texttt{list:/list/department/members}}$ $[\texttt{Mask}_{\texttt{member[jobcode<2]}}]$
Policy 3	
説明	人事部の者は従業員の給与情報にアクセスできるが、それは暗号化されねばならない
対象	`/list/department/members/member/salarycode`
主体	人事部に所属
可否	許可
責務	`encript(/list/department/members/member/salarycode)`
SQ	$\texttt{Apply}_{\texttt{list:/list/department/members/member}}$ $[\texttt{Enc}_{\texttt{salarycode}}]$

表 4 中に、ビュー `employee_list` に対する各ポリシーを表現したセキュリティ問合せを SQ の項に併記した。ここで、`Apply` は XML 断片を値とする属性に対して適用され、指定属性中の XPath で指定される XML 断片に対して `[]` 内の演算子を適用する高階演算である。`Proj` における `not(`属性`)` は、指定属性以外の全ての属性を意味する。`Mask` は `[]` 内の条件を満たすタプルの指定属性を *null* とする (即ち、XML 断片を除去する) 演算、`Enc`、`Sign` はそれぞれ、指定属性の XML 断片の暗号化、電子署名を行なう演算を表すものとする (表 5)。

表 5 拡張 XQGM 演算子
Table 5 Extended XQGM operators.

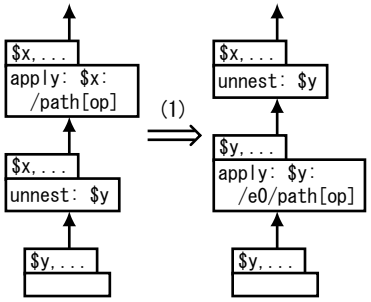
演算子	機能
apply	与えられた演算を指定 XML 断片に適用
mask	条件を満たさない時、指定属性を *null* で置換
enc	指定属性中の XML 断片の暗号化
sign	指定属性中の XML 断片の電子署名

ここでは便宜上、`Mask`、`Enc`、`Sign` を演算子として表記するが、ユーザ定義関数や RDBMS の提供する暗号化機能等を利用して実現できる。例えば、`Mask` は、真偽値判定によって与えられた値そのものか *null* を返す関数 `maskfunc()` が利用可能と仮定すると、指定属性以外はそのまま残し、指定属性を `maskfunc()` の適用結果で置き換える `Proj` 演算として記述す



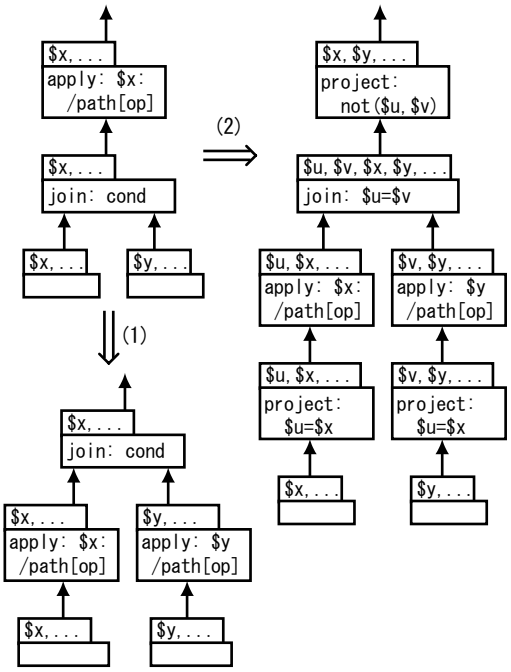
(1) : path1 = e1/e2/.../ek/path2
∧ xml_frag = <e1><e2>...<ek>
$y</ek>...</e2></e1>

図 6　Apply-Proj の交換
Fig. 6　Exchange of Apply and Proj.



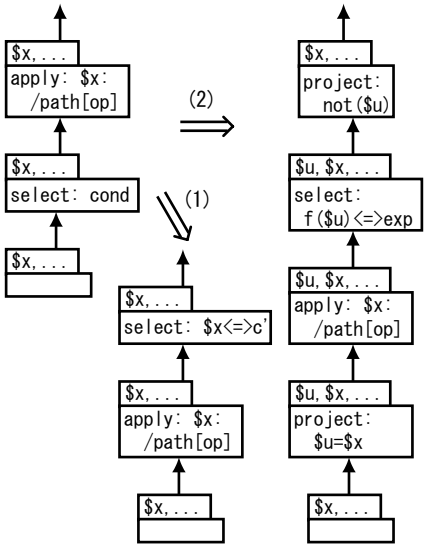
(1) : $y = <e0>$x</e0>

図 7　Apply-Unnest の交換
Fig. 7　Exchange of Apply and Unnest.



(1) : cond: $x に依存しない
∨ cond: $x = $y ∧ op ∈ {enc, sign}
(2) : cond: $x = $y ∧ op ∈ {proj, mask, enc, sign}

図 8　Apply-Join の交換
Fig. 8　Exchange of Apply and Join.



(1) : cond: $x に依存しない
∨ cond: $x と定数式 c の比較
∧ op ∈ {enc, sign}
(書替え後は op 適用後の値 c' と比較)
(2) : cond: $x に関する比較
∧ op ∈ {proj, mask, enc, sign}
(書替え後は $u を用いて比較)

図 9　Apply-Select の交換
Fig. 9　Exchange of Apply and Select.

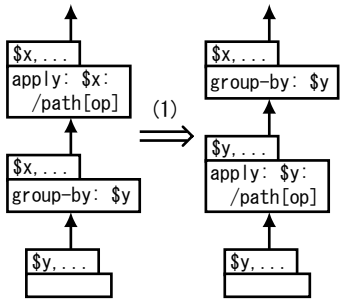
る事ができる。このような場合、これらは RDBMS において処理可能であるとみなして問合せ最適化を行なう。尚、暗号化や電子署名には、実際には様々なパラメータが使用されるが、表記上省略する。

前述したように、ビュー問合せとセキュリティ問合せの合成を行なう事で、ポリシーの適用を XML 出版過程において行ない、効率化をはかる。この効率化は、主に問合せ変換時に Apply を適切な位置までプッシュダウンする事で実現される。

3.3　演算子のプッシュダウン

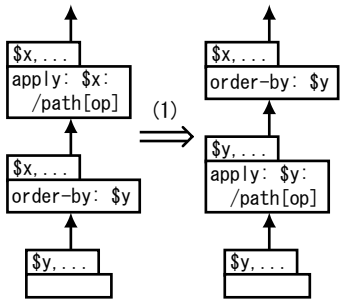
Apply 演算子を適切にプッシュダウンする事で、適用対象をより内部の小さな XML 断片とする事ができ、XPath 式は短くなる。完全にプッシュダウンした場合、Apply は除去され、[] 内の演算を直接適用する XQGM グラフになる。また、Apply の適用対象属性が入力属性リストに存在しない場合や、存在しても XPath が適合しない事が確定した場合には、Apply は効果を持たないため、除去する事ができる。

Apply を完全にプッシュダウンできない場合には、RDBMS



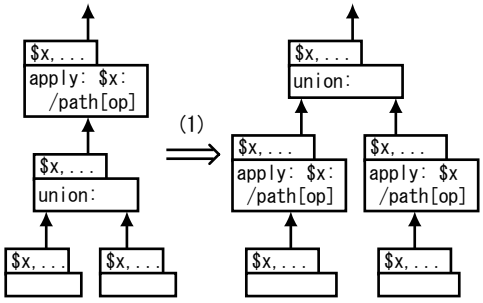
(1) $x が $y のグルーピング結果でない
∨ $x が $y のグルーピング結果
∧ op ∈ {enc, sign}
(2) $x が $y のグルーピング結果
∧ op ∈ {proj, mask, enc, sign}
(図中では割愛)

図 10　Apply-GroupBy の交換
Fig. 10　Exchange of Apply and GroupBy.



(1) $x が $y の並び替え結果でない
∨ $x が $y の並び替え結果
∧ op ∈ {enc, sign}
(2) $x が $y の並び替え結果
∧ op ∈ {proj, mask, enc, sign}
(図中では割愛)

図 11　Apply-OrderBy の交換
Fig. 11　Exchange of Apply and OrderBy.



(1)：常に可

図 12　Apply-Union の交換
Fig. 12　Exchange of Apply and Union.

で処理可能な部分グラフの直上に留め、ミドルウェア側で対象属性中の XML 断片をパースして、XPath で選択される部分木に対して処理を適用する。この時も、あらかじめ Apply を可能な限りプッシュダウンしておく事で、パース対象の XML 断片を小さく抑える事ができ、断片数は多くなるため並行処理をしやすくなる事も期待できる。また、Apply で適用される演算によって一部のデータが除去されれば、生成される XML 断片が小さくなる等、XML 断片生成コストを抑える事に繋がる。

各演算子との交換による XQGM グラフの書き替え規則を図 6 から図 12 に示す。演算子の交換可能条件は、各図の (1), (2) の通りである。但し、これらの条件が成り立つ時に必ず演算子を交換する事が望ましいわけではなく、問合せの意味が変わらない事を保証できる範囲で行なわなければならない。また、交換によって処理コストが増加する場合もあるため、その見積りを行なって適用する必要がある。

図 6 の Apply と Proj の交換は、XPath の先頭部分と Proj によって構成される構造が一致する時に行なわれ、Apply の適用対象を XML 断片の内部へ限定する。これは、前述のプッシュダウンの効果に直結し、ほとんどの場合に有効である。

図 7 の Apply と Unnest の交換は、Apply 適用対象の XML 断片を大きくしてしまう。これは、前述の Apply のプッシュダウンに期待する効果と反する。しかし、下位の演算子との交換によって Apply の適用対象をより小さくできる場合や、Apply を除去できる場合がある。このような場合には、この交換を行なうのが望ましいと考えられる。

また、図 8 の Apply と Join の交換は場合分けが少々複雑であるため、それぞれについて説明を加える。

(1a) 結合条件が Apply の適用対象である属性 $x に依存しない場合には、両項へ Apply をプッシュダウン可能である。実際には、演算子の交換により処理コストを低くできる場合に限り、プッシュダウンを行なうべきである。これには、結合前の両項のタプル数の合計が結合後のタプル数より少ない場合や、下位の演算子との交換によって Apply の適用コストを小さくできる場合、Apply によって適用される演算が以降の XML 断片の生成コストを小さくする場合等が該当する。

(1b) 結合条件に $x が使用され、かつ Apply で適用される演算が Enc または Sign である場合も、(1a) と同様にプッシュダウン可能である。一般に暗号化・電子署名の処理は高コストであり、入力 XML 断片のパース等の処理と合わせて、タプル数が処理コストに大きく影響する可能性がある。この交換は、暗号化・電子署名は入出力がともに単一の XML 断片であるような全単射関数であるため、これらの適用前の値での比較 $p = q$ と適用後の値での比較 $f(p) = f(q)$ は等価な結果を導く事による。Proj や Mask がこの交換規則に含まれないのは、このような性質を持たないためである。一方、XML 暗号化・電子署名の結果は、しばしば入力に対してかなり大きな文字列になる。このため、Enc や Sign の適用結果の値を持ちいた比較はコストが高くなりがちである事にも注意が必要である。タプル数は減るものの比較コストの方が高くなる場合には、交換を行なわないか、(2) の交換の妥当性を検討する。

(2) それ以外の Apply をプッシュダンする事によって適用対象となるタプル数が大きく減少する場合を考える。(1b) に述べたように演算子の交換によって比較コストが高くなったり、

Proj や Mask が適用されると比較結果が変わったりする。この交換規則は、比較前の値も別途保存して、演算を適用し、保存した属性で結合した後に削除するという余分なコストを払っても、全体のコストが低くなるような場合に適用される。

図 9 の Apply と Select の交換についても、基本的な考え方は Apply と Join の交換と同様である。(1) の場合、$x と定数式 c の比較は、$x に対して適用される処理と同等の計算を c に対して行なった結果の c' との比較に置き換えられる。$x を直接利用しない比較の場合には、比較値の計算のために元の値を必要とするため、演算子の交換を行なわない、または、(2) の交換規則の適用を検討する。

図 10 から図 12 に示す Apply と Group-By, Order-By, Union との交換は、これらの演算子は対象属性の構造を直接的に変形せず、タプル数も変化させないため、交換は容易である。図中では割愛したが、Apply と Group-By, Order-By との交換においては、Join, Select との交換における (2) と同様に、元の値を保存して処理を行なう事も考慮の対象となる。

XML 暗号化処理のように入力として XML 断片を必要とする場合は、XML 関数によるタグ付けが事前に必要となる。これは前述の通り、SQL2003 の相当する関数を呼べる場合に RDBMS で処理可能である。それゆえ、Enc, Sign を RDBMS 側で実現可能な場合には、入力の構成に必要なタグ付けの処理とともにプッシュダウンする事も可能である。

3.4 セキュリティ問合せ合成例

Policy 1 のプッシュダウンの過程を、関与するノードに限定して図 13 に例示する。Apply と Proj, Join, Group-By との交換が行なえ、最終的に Apply は除去されている事、これによりアクセス制御を実現する Proj が直接適用され、アクセスの拒否対象となる属性を早期に削除できる事が分かる。

project による $member の生成時に参照される$salarycode が削除されるが、このような存在しない属性を参照した場合、その値は *null* 値として扱われ、*null* 値からは要素が生成されないものと定める。Proj による属性削除等、問合せ最適化過程で明らかに確定できる場合は、同様の結果を生成するように上位の演算子のパラメータを書き替えておく。

この他の Policy 2 および Policy 3 のセキュリティ問合せでも、同様に Apply をプッシュダウンして、Mask および Enc を直接適用する XQGM グラフに再構成される。また、セキュリティポリシーの適用を問合せの合成に帰着させたため、複数のポリシーを適用する事も可能である。

以上の方式により、アクセス制御と、暗号化・電子署名の指示からなるセキュリティポリシーを、XML 出版過程において可能な限り早い段階で適用する事が可能となる。これは、生成された XML 断片の再処理を最小限に抑える事に繋がり、処理コストを抑える働きを期待できる。

4. ま と め

RDB の XML 出版は、XML データ統合における RDB の利用に重要である。また、実システムでは、XML ビューに対するセキュリティ機構が必要とされる。本稿では、セキュリティポリシーを XML 生成後ではなく XML 出版過程において適用し、処理コストを抑える手法を提案した。今後、高度な最適化方式の考案、コスト見積り手法の定式化、プロトタイプシステムの実装、効率の測定等を行なっていく予定である。

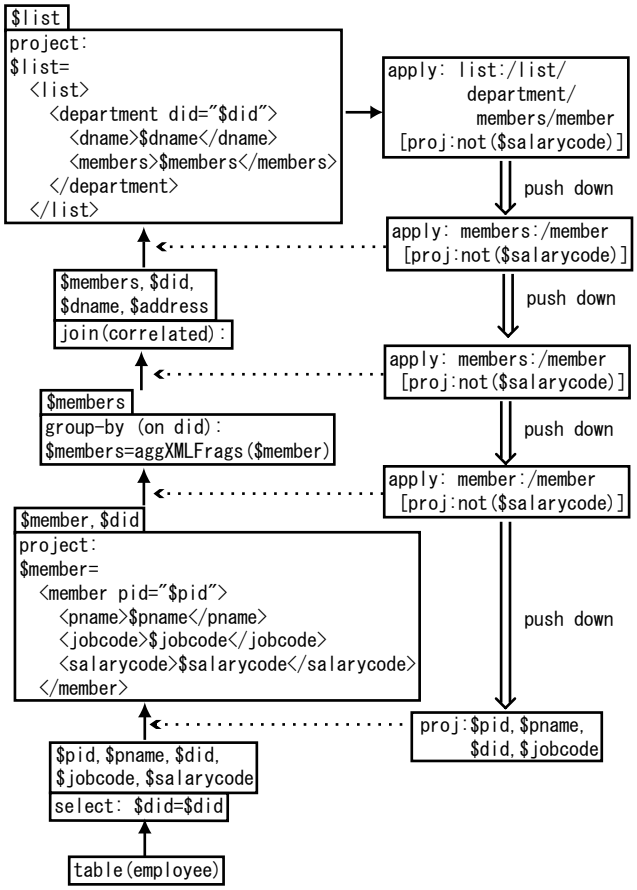


図 13 プッシュダウンの例
Fig. 13 Pushing down Apply.

[謝 辞]

本研究の一部は、CREST「自律連合型基盤システムの構築」、科学研究費補助金特定領域研究 (2)(#16016205)、基盤研究 (B)(#15300027) による。

文 献

[1] M. J. Carey, D. Florescu, Z. G. Ives, Y. Lu, J. Shanmugasundaram, E. J. Shekita, S. N. Subramanian. XPERANTO: Publishing Object-Relational Data as XML, WebDB 2000 (Informal Proc.), pp. 105–110, Dallas, USA, May, 2000.

[2] M. F. Fernandez, W. C. Tan, D. Suciu, SilkRoute: trading between relations and XML, Computer Networks, Vol. 33, No. 1-6, pp. 723–745, 2000.

[3] O. S. Godik, E. T. Moses (eds.). Extensible Access Control Markup Language (XACML) v1.0, http://www.oasis-open.org/specs/index.php#xacmlv1.0, OASIS, Jan, 2003.

[4] D. Eastlake, J. Reagle (eds.). XML Encryption Syntax and Processing, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/, W3C, Dec, 2002.

[5] D. Eastlake, J. Reagle, D. Solo (eds.). XML-Signature Syntax and Processing, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/, W3C, Feb, 2002.

[6] A. Eisenberg, J. Milton, SQL/XML is Makinkg Good Progress, ACM SIGMOD Record, Vol. 31, No. 2, Jun, 2002.