XSLT により定義された XML 実体化ビューのインクリメンタルな管理手法

宮坂　集策† 　石川　佳治†† 　北川　博之††

† 筑波大学システム情報工学研究科　〒 305–8573 筑波大学つくば市天王台 1–1–1
†† 筑波大学電子・情報工学系　〒 305–8573 茨城県つくば市天王台 1–1–1
E-mail: †syusaku@kde.is.tsukuba.ac.jp, ††{ishikawa,kitagawa}@is.tsukuba.ac.jp

あらまし　XML はインターネット上の汎用の情報交換フォーマットとして広く用いられており，ストリーム的な情報配信などでも利用されている．XML を加工するために利用される代表的な言語として XSLT がある．本研究では，継続的に配信される XML データを蓄積管理する XML リポジトリ上において，XSLT により定義された加工結果（本研究ではこれを XML 実体化ビューと呼ぶ）を効率的に維持管理するための手法を提案する．配信される XML データの更新情報は XUpdate 言語により記述されるものとする．提案手法では，XSLT による XML ビュー定義と発生し得るデータ更新要求の情報を解析し，更新情報の受信時に用いられる更新スクリプトを生成する．データの到着時には，更新スクリプトを実行することで，XML 実体化ビューをインクリメンタルに更新するための更新文を効率的に生成する．

キーワード　XML，XSLT，ビュー管理，情報配信，ストリーム処理，XUpdate

Incremental Maintenance of Materialized XML Views Defined by XSLT

Shusaku MIYASAKA†, Yoshiharu ISHIKAWA††, and Hiroyuki KITAGAWA††

† Graduate School of Systems and Information Engineering, University of Tsukuba
†† Institute of Information Sciences and Electronics, University of Tsukuba
1–1–1 Tennoudai, Tsukuba, Ibaraki, 305-8573 Japan
E-mail: †syusaku@kde.is.tsukuba.ac.jp, ††{ishikawa,kitagawa}@is.tsukuba.ac.jp

Abstract　Today, XML is widely used as a general-purpose format for information exchange on the Web. It is also used in stream-oriented data dissemination and other purposes. In this paper, we propose an efficient method to maintain XSLT transformation results, called *XML views*, on an XML repository. XSLT is a popular transformation language for XML documents. We assume that update requests for the maintained XML documents are described in the XUpdate language and delivered continuously through the network. In our approach, the view management system analyzes XML view definitions described in XSLT and specifications of XML data updates that may occur, then generates update scripts. When a new XUpdate document is arrived, some of the update scripts are executed and view update specifications for the corresponding XML views are generated. Using this approach, we maintain XML views incrementally as possible.

Key words　XML, XSLT, View maintenance, Information dissemination, Stream processing, XUpdate

1. ま え が き

XML [1] はインターネット上の汎用の情報交換フォーマットとして広く用いられているが，近年，情報配信・データ放送における配信情報の記述や，ネットワークに接続された各種機器間の通信でも利用されている．XML は，さまざまな形でデータの変換・加工処理が可能という特色を有している．XML データの変換処理において用いられる言語の代表的なものとして，XSLT [2] がある．XSLT では，それ自体が XML 文書である XSLT スタイルシートにより XML の変換処理を記述する．それを XSLT プロセッサが処理することで，変換結果のテキストが生成される．本稿では，蓄積された XML データに対し，XSLT スタイルシートの記述に基づき生成される XML データのことを，元の XML データに対する **XML ビュー**（XML

view）と呼び，また，ここで XML データの変換に用いた XSLT スタイルシートのことを **XML ビュー定義**と呼ぶことにする．

XML の利用の拡大につれ，ニュース記事・株価情報などの配信やネットワークを介したセンサ情報の送信など，情報配信においても XML データの利用が今後さらに増加していくと考えられる．配信された XML データは受信者により取捨選択され，蓄積・利用されるが，その処理の効率化は重要な課題である．特に，XML データを継続的に配信する状況においては，毎回の配信においてサーバが保持する大量の XML データをクライアントに送付することはオーバヘッドが大きい．そのため，前回までに配信された情報との差分のみを配信することによって効率化を図ることが考えられる．

本研究では，このような情報配信環境における **XML 実体化ビュー**（XML materialized view）の管理手法を提案する．蓄積された XML データに対して XML ビューが定義されており，それらの定義をもとにビューが実体化されているものとする．一方，蓄積された XML データに対する更新要求を行う差分の情報を表すデータ（以下では，このようなデータを**データ更新情報**と呼ぶ）が，ネットワークを介してストリーム的に次々と配信されるとする．本研究では，そのような情報が XML データの更新を表現するための提案の一つである，XUpdate 言語 [3] で記述されていると想定する．

継続的なデータ更新情報の配信に対応する最も直接的な手法は，XUpdate の形式によるデータ更新情報が配信されるたびに対応する XML データに更新内容を反映し，それをもとに再度全ての XML 実体化ビューを再生成する方法である．しかし，この方式にはオーバヘッドが大きいという問題があることから，インクリメンタルな更新を低コストで行う機構が求められる．本提案手法では，XSLT による XML ビュー定義と配信されうるデータ更新のパターンを表す情報を活用することによって，データ更新情報を示した XUpdate 文の受信時に起動される更新スクリプトを生成する．更新スクリプトは，配信されたデータ更新情報をもとに，対応する XML 実体化ビューに適用する更新処理を生成する．本稿では，提案手法の概要とその実現手法について述べる．

2. 研究の目的

この節では，本研究の目的を例を用いて説明する．

2.1 システムの構成

本研究で想定しているシステムの構成は図 1 のようになる．図中央の XML ビュー管理システムは，XML リポジトリに格納されているデータを XSLT スタイルシートによる XML ビュー定義に従って変換し，XML 実体化ビューの生成を行う．基本的には，データ更新情報を示した XUpdate 形式の文書が配信されるたびに，XML ビュー管理システムは XML リポジトリの XML ファイルに更新情報を反映し，XML ビュー定義に基づいて XML ビューを再生成する．図中に示されている更新テンプレートおよび更新スクリプトの役割及び詳細については後述する．

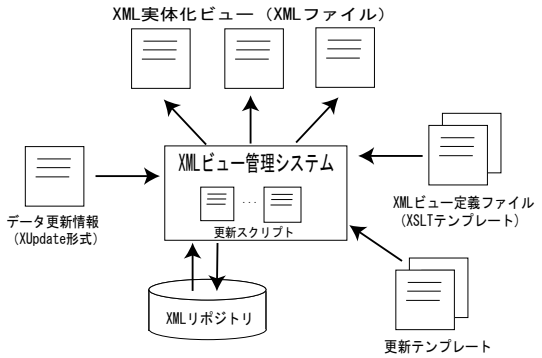

図 1　システム構成

2.2 XML 実体化ビューの具体例

まず，以下の説明で用いる XML 実体化ビューの例について説明する．図 1 に示した XML ビュー管理システムの XML リポジトリに，図 2 の国名と都市名に関する情報を表す XML データが蓄積管理されているとする．一方， XML ビュー管理システムには，図 3 に示す XML ビュー定義が記述された XSLT スタイルシートが登録されているとする．このスタイルシートを図 2 に適用すると，以下の処理が行われることになる．

- 入力 XML の `country` 要素の属性 `name` の値を，出力 XML では `country` 要素の子要素の `name` 要素の値とする．
- 入力 XML の `city` 要素の属性 `name` の値を，出力 XML では `city` 要素の値とする．
- 入力文書中に現れる属性 `id` はすべて取り除く．

以上の処理の結果として得られる XML 文書（XML ビュー）を図 4 に示す．

```
 1: <?xml version="1.0"?>
 2: <world id="1">
 3:  <country id="2" name="Germany">
 4:   <city id="21" name="Berlin"/>
 5:   <city id ="22" name="Bonn"/>
 6:  </country>
 7:  <country id="3" name="France">
 8:   <city id="31" name="Paris"/>
 9:   <city id ="32" name="Sanary"/>
10:  </country>
11: </world>
```

図 2　XML データ

```
 1: <?xml version="1.0"?>
 2: <xsl:stylesheet
 3:  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 3:  version="1.0">
 4:  <xsl:template match="/world">
 5:   <world>
 6:    <xsl:apply-templates select="country"/>
 7:   </world>
 8:  </xsl:template>
 9:  <xsl:template match="country">
10:   <country>
11:    <name><xsl:value-of select="@name"/></name>
12:    <xsl:apply-templates select="city"/>
13:   </country>
14:  </xsl:template>
15:  <xsl:template match="city">
16:   <city><xsl:value-of select="@name"/></city>
17:  </xsl:template>
18: </xsl:stylesheet>
```

図 3　XML ビュー定義 （XSLT スタイルシート）

```
 1: <?xml version="1.0"?>
 2: <world>
 3:  <country>
 4:   <name>Germany</name>
 5:   <city>Berlin</city>
 6:   <city>Bonn</city>
 7:  </country>
 8:  <country>
 9:   <name>France</name>
10:   <city>Paris</city>
11:   <city>Sanary</city>
12:  </country>
13: </world>
```

図 4　スタイルシートの適用結果（XML ビュー）

2.3　XUpdate による XML データの更新

本研究では，XML データを継続的に配信する情報源から，図 5 に示すような XUpdate 形式のデータとしてデータ更新情報が逐次配信されてくるものとする．XUpdate [3] は，XML:DB Initiative が提案している XML データベースへの更新要求を記述するための言語である．図 5 では，図 2 に示した XML データに対し，“/world” というパスで表される要素の末尾の子として，イタリアに関する新たな country 要素を追加する指示を表している．

```
 1: <?xml version="1.0"?>
 2: <xu:modifications version="1.0"
 3:  xmlns:xu="http://www.xmldb.org/xupdate">
 4:   <xu:append select="/world">
 5:    <xu:element name="country">
 6:     <xu:attribute name="id">4</xu:attribute>
 7:     <xu:attribute name="name">Italy</xu:attribute>
 8:     <city id="41" name="Roma"/>
 9:     <city id="42" name="Milano"/>
10:    </xu:element>
11:   </xu:append>
12: </xu:modifications>
```

図 5　XUpdate データの例 (1)

図に示すように，XUpdate データのルート要素は modifications 要素である．その子要素に更新命令の XML 要素のシーケンスを含めることにより，更新内容を表現する．そのような更新命令の要素には，図に示した append 以外に，insert-before, insert-after, update, remove, rename がある．いずれの要素も，属性として変更箇所を特定するための XPath 式を値としてとる select 属性を有する(注1)．append 要素はデータ追加処理を指示する記法であり，要素，属性，テキスト，処理命令，コメントの追加を記述できる．上記の例では country 要素（属性値 id, name および要素内容の追加を伴う）の追加が指定されている．XUpdate には XML データの生成や変数などに関する構文も存在するが，本論文の内容には直接には関連しないので，ここでは省略する．

以下の議論では，簡単化のため，modifications ルート要素の子孫として，append 要素がただ 1 つ出現するような XUpdate データのみを考える．modifications 要素は，一般に任意個の更新命令の要素のシーケンスを含むが，それを単独の更新命令を含む複数個の XUpdate データのシーケンスに切り分けることは容易であるため，上記の制約を入れても本質的な影響は小さい．一方，追加処理だけに限って議論する理由は，先に述べたように，ストリーム的な更新情報の配信においては，データの追加に伴う更新処理が特に頻繁に生じると考えられるためである．他の更新処理への対応は今後の課題とする．

なお，本研究は XUpdate 言語に基づく更新情報の記述を前提とはしているが，XUpdate 言語のサポート自体が研究の目的ではない．本手法では，提案されている XUpdate の機能のうち，XML データの更新要求記述に基本となる機能の一部を利用しているのみであるため，XML データの更新要求を記述可能な他の言語にも適用可能である．

ここで，XUpdate データが与えられたときの最も直接的なビュー更新処理のアプローチについて触れておく．直接的にビューの更新を行う方式では，図 5 の XUpdate データが与えられたときには，図 2 のデータに更新を行い，再び図 3 の XSLT スタイルシートを適用し，全ての XML ビューを再生成することによってデータの更新内容を実体化ビューに反映させることができる．しかし，このようなアプローチには次のような問題がある．

- 配信された XML データをローカルな XML リポジトリに蓄積する必要があるための格納コストが発生する．
- XML リポジトリ中の XML データに更新処理を毎回適用して XML 実体化ビューを再生成するため，一つ一つの更新内容はわずかでも，XML データが大きくなったり，管理しているビューの数が大きくなるにつれて実体化ビューの再生成にかかる時間が増大する．
- データ更新情報の内容によっては，既存の XML 実体化ビューに影響を及ぼさない場合もある．しかし，上記のアプローチではそのような場合にも実体化ビューの再生成処理が発生してしまう．

このような問題点を踏まえ，本研究では XML 実体化ビューのインクリメンタルな更新を行う手法を提案する．次節にて提案手法の詳細について述べる．

3.　提案手法の概要

3.1　前提となる考え方

本研究の前提となるのは，情報配信においては，しばしば定型的な更新処理が行われるというという考え方である．たとえば，ニュース情報の配信では，新たなニュース記事を前回までの記事に追加するような形で配信することが，自然なアプローチである．また，ネットワークを介したセンサ情報の送信などにおいても，新たにセンサ情報が検出された時点で，その情報のみを追加情報として送信する方が，センサ側の処理が軽量で済むことから，一般的なアプローチと考えられる．以上のような更新処理はデータの追加を主体とするもので，更新内容のコンテンツは毎回異なってはいても，そのフォーマットは一定であり，図 5 で示したような比較的単純な更新処理からなると考えられる．

継続的に XUpdate によりデータ更新情報が配信される他の

(注1)：なお，append 要素のみ，データをどの位置に追加するかを指定するための child 属性を持つが，ここでは省略する．省略時のデフォルトは「末尾に追加」である．

応用シナリオとしては，XML 形式で管理されている株価データ，在庫データ，カタログ情報などに，ネットワーク経由で動的に更新を行うようなケースが想定される．そのような場合は，内容の追加だけでなく，更新，削除などのすべてのパターンが生じうるが，更新・削除に関しては ID 属性や絶対パスの指定などを利用して対象要素を明確に特定する場合が一般的と考えられる．よって，このような応用の場合にも，データの更新処理をいくつかの定型的なものに限定できることが多いと考えられる．

以上のような考え方をもとに，本手法では定型的な更新が継続的に行われる状況を対象とする．

3.2 ビュー更新処理の概略

前節で述べたように，本研究では定型的なデータの差分情報が更新要求として継続的に配信されるという状況を想定している．このような状況において，本手法ではまず前処理段階として，データ更新情報としてどのようなパターンのものが配信されうるかという情報を**更新テンプレート**として保持しておく．また，更新テンプレートに含まれる情報を解析することにより，XML 実体化ビューの更新処理プランを記述した**更新スクリプト**を生成する．これを用いることにより，データ更新情報が配信された際のビュー更新処理の効率化を図る．以上のような情報を用いた XML ビュー管理システムにおける処理ステップの概略は次のようになる．

（1） 配信されたデータ更新情報（XUpdate データ）のパターンに対応する適切な更新スクリプトを選択する．

（2） 選択した更新スクリプトを配信された XUpdate データに適用する．

（3） 適用結果を対応する XML 実体化ビューに反映する．

3 番目の XML 実体化ビューの更新に関しては，実体化ビュー更新用の XUpdate データを生成し，次いで，その XUpdate データに基づく更新処理を実行するものとする．このような処理によって，XML 実体化ビューのインクリメンタルな更新を実現する．ここで，例として図 6 のデータ更新情報が与えられた場合には，図 4 の XML ビューを更新するために，上記の処理ステップを経て XML ビュー管理システムは最終的に図 7 のような XML 実体化ビュー更新用の XUpdate データを生成する．すなわち，上記の説明における「適用結果」とは，図 7 のような XUpdate データのことを指している．場合によっては，適用結果が空，すなわちビューの更新が生じない場合があるため，その場合には上記 3 番目のステップは実行されない．

```
 1: <?xml version="1.0"?>
 2: <xu:modifications version="1.0"
 3:  xmlns:xu="http://www.xmldb.org/xupdate">
 4:  <xu:append select="/world/country[@name='Italy']">
 5:   <xu:element name="city">
 6:    <xu:attribute name="id">43</xu:attribute>
 7:    <xu:attribute name="name">Napoli</xu:attribute>
 8:   </xu:element>
 9:  </xu:append>
10: </xu:modifications>
```

図 6　XUpdate データの例 (2)

このように XML 実体化ビューの更新処理についても XUpdate を用いて記述する理由としては，以下の理由を挙げることができる．

```
1: <?xml version="1.0"?>
2: <xu:modifications version="1.0"
3:  xmlns:xu="http://www.xmldb.org/xupdate">
4:  <xu:append select="/world/country[@name = 'Italy']">
5:    <xu:element name="city">Napoli</xu:element>
6:  </xu:append>
7: </xu:modifications>
```

図 7　XML 実体化ビュー更新用 XUpdate データ

- 更新処理が明確に記述できる．
- XUpdate 対応のツールを用いて更新処理が容易に実行できる．
- 更新処理の自由度が増す：更新処理をその都度適用するのではなく，システムの負荷が低い時点でまとめて更新処理を適用するなどの，柔軟な管理が行える．
- XML 実体化ビューを他のサイトに置く自由度が存在する：その場合には，実体化ビュー更新用の XUpdate データを対応サイトに送付し，更新はそのサイトに任せることになる．このような機能は，ストリーム的に配信される情報を加工して再配信する仲介者的な機能を実現する際に有用であると考えられる．

更新処理手続き *update* は図 8 のようになる．この手続きは配信された XUpdate ファイル f を受け取り，その `select` 属性の値（XPath 表現）を抽出する．2〜8 行目のループでは，各更新スクリプトに対するマッチングを行う．*u.select* で示している各更新スクリプトに対応する XPath のパターンを，3 行目で照合することにより，XPath のパターンと入力 XUpdate のパターンがマッチするかを判定する．マッチした場合は 4 行目でスクリプトを適用し，その結果である XUpdate ファイル *out* が空でなければ，6 行目で更新処理を行う．

以下の節において，ビューの更新処理の際に本手法で使用する更新テンプレート，更新スクリプト及び XML 実体化ビュー更新用 XUpdate データを生成するために必要となる**更新用 XSLT スタイルシート**の生成処理について述べる．

procedure *update*(f)
f: 入力 XUpdate ファイル
1 $s \leftarrow$ *xpath*(`//xu:append/@select`); // XPath 表現を抽出
2 **foreach** update_script u **do**
3 **if** *match*(*u.select*, s) **then** // マッチした
4 *out* $\leftarrow$ *apply_script*(u, s); // 更新スクリプト u を適用
5 **if** *out* $\equiv$ NULL **break**; // 適用結果が空なら次の処理へ
6 *update_view*(*u.view*, *out*); // u に対応するビューを更新
7 **end**
8 **end**

図 8　更新処理の手続き

3.3 更新テンプレート

前節でも触れたように，本手法では定型的なデータ更新処理のパターンに関する情報は**更新テンプレート**として保持する．これは，配信されうる XUpdate データのパターンを記述した

ファイルである．これに対して XML ビュー管理システムは事前に解析を行うことで，XUpdate データが配信された際に影響を受ける（可能性がある）実体化ビューを特定し，その実体化ビューに対する**更新スクリプト**を事前に作成する．

図 5 に示したデータ更新要求に対応する更新テンプレートは図 9 のように表される．更新テンプレートはデータ更新要求のパターンを一般的に表現したものであり，図 9 の例は,「データに対してある 1 つの国に関する情報を新たに追加する」というパターンに当たる．また，図 9 中の “*” は任意のテキストにマッチするワイルドカードである．図 5 の XUpdate データが与えられた場合，“*” には country 要素の属性と要素内容に関する記述がマッチする．

```
1: <?xml version="1.0"?>
2: <xu:modifications version="1.0"
3:  xmlns:xu="http://www.xmldb.org/xupdate">
4:  <xu:append select="/world">
5:   <xu:element name="country">*</xu:element>
6:  </xu:append>
7: </xu:modifications>
```

図 9　更新テンプレートの例 (1)

より一般的な更新テンプレートの例を図 10 に示す．このテンプレートは,「国名を指定し，その国の 1 つの都市に関する情報を新たにデータに追加する」というパターンを示している．この更新テンプレートに対応する XUpdate データの例を図 6 に示す．図 10 のテンプレートにおいては，追加位置を指定する select 属性中の XPath 表現の中に**テンプレート変数** $x が用いられている．テンプレート変数は任意のテキストにマッチするワイルドカードとして扱われるが，図 10 の例のようにテンプレート内に現れる XPath 表現に関して，文字列などの定数部の値の束縛にも用いられる．これによって，1 つのテンプレートが対応するパターンの表現能力が向上し，図 6 に示したような，特定の要素を指定した追加処理にも対応することができる．図 6 の例に対し更新テンプレートを適用する場合には，変数 $x には文字列 “Italy” が束縛されることになる．

```
1: <?xml version="1.0"?>
2: <xu:modifications version="1.0"
3:  xmlns:xu="http://www.xmldb.org/xupdate">
4:  <xu:append select="/world/country[@name='$x']">
5:   <xu:element name="city">*</xu:element>
6:  </xu:append>
7: </xu:modifications>
```

図 10　更新テンプレートの例 (2)

3.4　更新スクリプト

更新スクリプトとは，XML 実体化ビューの更新処理記述（具体的には XML 実体化ビュー更新用 XUpdate ファイルの生成処理）のことである．XML リポジトリにおいて登録･管理されている図 2 のような XML データの集合を $Entries$ で表し，$e \in Entries$ に対し XML ビュー管理システムに登録されている XML ビューの定義（XSLT スタイルシート）の集合を $Views(e)$，更新テンプレートの集合を $Templates(e)$ と記す．各 $e \in Entries$ について，それぞれのペア $(v, t) \in Views(e) \times Templates(e)$ に対し，更新テンプレート t による更新が実体化ビュー v に更新を及ぼしうる場合のみ，これらのペアに対して更新スクリプト u を生成する．更新スクリプト u に対応する実体化ビューを $u.view$ で，また，u に対応する更新テンプレートの select 属性の値を $u.select$ で表す．図 8 ではこれらの記法を用いている．

新たな XML ビュー定義が登録されたとき，XML ビュー管理システムは，既存の各更新テンプレート t に対し，t に従う更新がその XML 実体化ビューに影響を及ぼしうるかどうかを判定し，影響を及ぼす可能性がある場合には更新スクリプトを生成する．逆に，新たな更新テンプレートが登録された場合には，XML ビュー管理システムは既存の XML ビュー定義のそれぞれに対し同様の判定処理を行い，必要があれば更新スクリプトを生成する．一方，XML ビュー定義もしくは更新テンプレートが削除された場合には，対応する更新スクリプトを削除する．

更新スクリプトは，実際には XSLT プロセッサを呼び出しながら処理を進めるドライバプログラムである．図 3 の XML ビュー定義と図 10 の更新テンプレートのペアに対する更新スクリプトの例を挙げて更新スクリプトの働きを説明する．この更新処理では，対応する XSLT スタイルシートである図 11 を呼び出すことになる．具体的な XSLT スタイルシートの導出法については次節で述べる．

（1） まず，図 6 のような入力 XUpdate ファイルから，

```
<?xml version="1.0"?>
<city id="43" name="Napoli"/>
```

という，追加される XML フラグメントの情報を構成するため，XSLT プロセッサを呼び出す．この処理は，XML ビュー定義や更新テンプレートに依存しない，一般的な XSLT スタイルシートで記述される．上記の XML フラグメントを一時ファイルに格納する．

（2） 次に，図 11 に示す更新用 XSLT スタイルシートについて，変数の置換を行う．このスタイルシートの場合，8 行目にテンプレート変数 $x が埋め込まれている．このような XSLT スタイルシートについては，図 8 の手続きの 3 行目の照合処理の際に得られた束縛情報によりテンプレート変数の置換を行う．この例の場合は $x を定数 “Italy” に置き換える．

（3） 置換後の XSLT スタイルシートを，上記 XML 一時ファイルに適用することにより，図 7 に示すようなビュー更新用 XUpdate データを得る．

3.5　更新用 XSLT スタイルシートの生成処理

本節では例として，図 3 の XML ビュー定義と図 10 の更新テンプレートからどのように更新用 XSLT スタイルシートを生成するかについて述べる．更新用 XSLT スタイルシートの生成処理のステップは以下のようになる．

Step 1: 更新テンプレートにより追加される要素のパスが，XML ビュー定義の XSLT スタイルシートのどの箇所に対応するかを特定する．

- まず，更新テンプレートからパス情報を抽出する．最初

```
 1: <?xml version="1.0"?>
 2: <xsl:stylesheet version="1.0"
 3:  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 4:  xmlns:xu="http://www.xmldb.org/xupdate">
 5:  <xsl:template match="/">
 6:   <xu:modifications version="1.0"
 7:    xmlns:xu="http://www.xmldb.org/xupdate">
 8:    <xu:append select="/world/country[name='$x']">
 9:     <xsl:apply-templates select="city" />
10:    </xu:append>
11:   </xu:modifications>
12:  </xsl:template>
13:  <xsl:template match="city">
14:   <xu:element name="city">
15:    <xsl:value-of select="@name"/>
16:   </xu:element>
17:  </xsl:template>
18: </xsl:stylesheet>
```

図 11　更新用 XSLT スタイルシートの例

に，図 10 の 4 行目の更新テンプレート `select` 属性の内容を変数 p_1 に代入し，p_1 = ”/world/country” となる．

- 次に，更新テンプレートの本体部で追加される要素のパス（図 10 の 5 行目）を変数 p_2 に代入し，p_2 = ”city” となる．

これら 2 つを連結し，パス $p = p_1/p_2$ = ”/world/country/city” を得る．

Step 2: パス p に対応するテンプレート（`xsl:template` 要素）集合をビュー定義 XSLT スタイルシート（図 3）の上位から探索する．具体的には，`xsl:template` 要素の `match` 属性をトップレベルから順にチェックする．まず図 3 の 4 行目のテンプレートの `match="/world"` がパス p の先頭部にマッチし，次に適用される 7 行目のテンプレートの `match="country"`，そして 13 行目のテンプレートの `match="city"` がパス p の残り部分にマッチする．その結果，図 3 の 15〜17 行目のテンプレートがパス p に対応する XSLT のフラグメントとなる．

Step 3: Step 2 でパスの情報がマッチしない場合には，XML ビュー定義と更新テンプレートが対応していないことになる．この場合はここで処理を終了する．

Step 4: 図 11 の 1〜12 行目および 18 行目からなる，更新用 XSLT スタイルシートの雛形を作成する．この内容は更新用テンプレートの情報を用いて作成できる．

Step 5: ステップ 2 で抽出した XSLT スタイルシートを，XUpdate の形式へ対応させるために若干のフォーマットの変更を行い，図 11 の 13〜17 行目に埋め込む．

他の場合も同様に，更新テンプレートの記述内容をもとにして

（1） XSLT テンプレートの雛形の作成

（2） XML ビュー定義用の XSLT スタイルシートから対応する箇所の抽出と加工

（3） （2）で得られた結果の（1）への埋め込み

というような一連の処理を適用する．

4. 議　　論

4.1 XSLT，XPath に対する制約

XSLT 言語 [2] は汎用の XML 変換用言語であるため，多くの機能を有しており，データ変換に関する議論は容易でない．そこで，本研究では XSLT のサブセットである XSLT_0 言語の利用を想定している．XSLT_0 言語は [4] で提案され，XSLT の骨格を抜き出したものであり，`xsl:template`, `xsl:apply-template`, `xsl:variable`, `xsl:if` などの基本的要素からなり，`xsl:for-each` などは含まれない(注2)．しかし，`xsl:for-each` は XSLT による変換処理において非常に様々な場面で利用されるものであるため，今後本研究においても支援を検討したいと考えている．`xsl:for-each` を用いた変換処理の記述は構造が複雑なものとなることが多いと考えられるため，構造が比較的簡単なものに対しては以下で述べる XPath に対しての制約と合わせて考えることによって支援が可能となるケースがあると考える．

一方，XUpdate の `select` 属性で指定される XPath 表現と，XSLT の `xsl:template` の `match` 属性で指定されるロケーションパターンに関しては，本稿では XPath フラグメントの基礎となる 4 つの構成要素 [5] である “`/`” (child axis), “`//`” (descendant axis), “`*`” (wild card), “`[]`” (branch) のうち，“`//`” を除く 3 つのパターンから構成されるパターンを支援する．また，XPath などの関数群などについては省略する．“`//`” パターンなどの支援については今後の課題とする．

4.2 XML ビュー管理の分類

前節の内容とも関連するが，本研究においての XML 実体化ビューの管理は，扱う XML データ，XML ビュー定義，データ更新のパターンといった情報の構造の複雑さによって以下のようなものに分類できる．

（1） データの蓄積が不要なもの

（2） 一部のデータの蓄積が必要となるもの

(1) は構造が最も単純なものであり，本稿において紹介した XML 実体化ビュー管理の例は全て（1）に属する．これに関しては，本研究において提案している手法を用いて XML 実体化ビューの更新を行う際には XML リポジトリへのアクセスを必要としない．従って，(1) に属するビュー管理のみを行うようなケースでは XML データを蓄積しておく必要がなく，データの格納コストを抑えることができる．

次に，(2) に対応するビュー管理においては XML データに対する条件指定の記述はより複雑なものとなっている．例として，図 2 のデータに対して「都市データを 1 つだけ持つ国の国名を抜き出す」というビューが定義されている場合を考えると，データ更新情報として図 6 のようなパターンがシステムに与えられた場合には，都市データが追加された際にどの国のデータ

(注2)：また，関数などもいくつかの基本的なものに限定されている．なお，`xsl:template` の `match` 属性で指定されるロケーションパターン（XPath 的な表記）については，XSLT 仕様のままブラックボックス的に扱われており，XSLT_0 で特に限定はされていない．

をビューの内容としてに抜き出すべきであるかを判断しなければならない．すなわち，(2) のビュー管理については，ビューを生成・更新する際に XML データへの参照が必要となり，XML データに対する格納コストが発生する．しかし，このような場合においてもビューの中で指定されている条件に関係する要素についての情報のみを保持しておけば十分であり．ビュー定義の内容に関連しないデータは保持しておく必要はない．

以上の点について，本手法を用いてどのような範囲の支援が可能となるかなどの詳細については今後研究を進めていく予定である．

5. 関 連 研 究

半構造データベースのインクリメンタルビュー更新に関しては，OEM モデルに基づくビューの更新を扱った [6] などがある．また，XQuery により定義されたビューのインクリメンタルな管理に関しては [7] の研究がある．

XSLT に関しては，その利用が一般化しているにも関わらず，インクリメンタルなビュー管理に関する研究はいまだ見られない．しかし，XSLT の処理を RDBMS の処理と融合する研究 [8] のように，大量のデータの利用環境における XSLT 処理の効率化に関する研究が進められている．また，XSLT のセマンティクスの分析に関する研究も [4] などで行われており，XSLT の骨格となる基本的な機能に絞った XSLT_0 言語の提案などもなされている．

6. まとめと今後の課題

本稿では XSLT により定義された XML 実体化ビューのインクリメンタルな更新処理方式のアプローチについて述べた．XUpdate 言語に基づくフォーマットにより，ネットワーク経由で XML データに対する更新情報が送付されてくる場合に，元の XML データにいったん更新を行って再びビュー導出をするのではなく，更新情報をもとに直接的にビュー更新を図ることを目的としている．新たに配信された更新情報のみを用いたインクリメンタルな処理により，効率的なビュー更新を目指している．

今後の課題は以下のようになる．

- 対象となる XSLT 言語のサブセットの明確化：XSLT 言語は豊富な機能を持った汎用の XML 変換言語であり，フルセットの言語に対するインクリメンタルなビュー管理は困難であると考えられる．本稿では XSLT_0 言語 [4] の使用を想定したが，ビュー管理に与える影響をより明確に議論するためのサブセットの構築が必要であると考える．
- 更新処理方式の詳細化：更新処理アルゴリズムや更新スクリプト導出アルゴリズムを詳細化する．
- XML データの更新言語に関する検討：本稿では XUpdate 言語 [3] を議論の中心としたが，XUpdate 言語の仕様化はドラフト段階で止まっており，必ずしも XUpdate が今後主力の XML 更新言語となるとはいえない．XML データの差分計算ツールの出力フォーマットなど，関連する情報の調査に基づく，基本的な XML データ更新機能のサポートが必要である．

謝　　辞

本研究の一部は，文部科学省科学研究費特定領域研究 (2)（15017207），日本学術振興会科学研究費基盤研究（B）（15300027），若手研究（B）（14780316）によるものである．また，本稿を執筆するにあたり，ご多忙の中非常に興味深い御指摘・御意見を頂きました査読者の方々に厚く御礼を申し上げます．

文　　献

[1] XML, Extensible Markup Language. http://www.w3.org/XML/
[2] XSLT, XSL Transformations version 1.0 http://www.w3.org/TR/xslt
[3] XUpdate - XML Update Language, Working Draft, Sept. 2000. http://www.xmldb.org/xupdate/
[4] G.J. Bex, S. Maneth, and F. Neven, “A Formal Model for an Expressive Fragment of XSLT”, *Information Systems*, 27, pp. 21–29, 2002.
[5] G. Miklau and D. Suciu, Containment and Equivalence for an XPath Fragment, *Proc. ACM PODS*, 2002.
[6] S. Abiteboul, J. McHugh, M. Rys, V. Vassalos, and J. Wiener, Incremental Maintenance for Materialized Views over Semistructured Data, *Proc. VLDB*, 1998.
[7] K. Dimitrova, M. El-Sayed, and E.A. Rundensteiner, Order-Sensitive View Maintenance of Materialized XQuery Views *Proc. ER*, pp. 144-157, 2003.
[8] G. Moerkotte, Incorporating XSL Processing into Database Engines, *Proc. VLDB*, pp. 107-118, 2002.