XMLによるバージョン管理手法VerteX

3Y−05　吉村 朋美† 横山 昌平‡ 太田 学‡ 片山 薫‡ 石川 博‡
†東京都立大学工学部電子情報工学科
‡東京都立大学大学院工学研究科

1 はじめに

どのようなデータ、文書でも更新が進むにつれ、最新のバージョンだけでなく、以前のバージョンが必要になることがある。その時、用いられるのがバージョン管理という手法である。その代表に、RCSやCVSがあるが、これらはXMLの構造が考慮されておらず、XMLに対するバージョン管理は、いまだ十分ではない[1]。

そこで本稿では、XMLの特性を生かしたバージョン管理手法を提案する。頻繁にアクセスがあると思われる最新バージョンは完全に保存し、以前のバージョンは別途差分データを付加する。この差分データを用いることにより、任意のバージョンを取り出すことができる。

従来のバージョン管理システムでは、差分データ(または編集スクリプト)は各システムによって形式が異なっていたが、本システムで作成されるデータはすべてXMLである。また、本文、差分データ、編集スクリプトが一つの文書にまとまっているので、文書管理が容易である。

2 関連研究

XMLのバージョン管理手法として、UBCC (Usefulness Based Copy Control) [2],[3]がある。

これは、ページ有効性(各バージョンに対する有効なオブジェクトのページ占有率)に基づき、ページ有効性を一定値以上保つようにオブジェクト(ここではタグで囲まれた要素)をコピーして保存する。バージョンの復元は、バージョン毎に、加えた編集のスクリプトを別途保存しておき、そのスクリプトを辿ることで行う。

本研究では、様々な情報を一つの文書に持つことができるというXMLの特性を生かしたバージョン管理手法を目的としており、復元効率に特化したUBCCとは目的を異にしている。

3 VerteX

ここでは、VerteX(A Versioning Technique for XML Documents)の概要を説明する。

3.1 前提条件

最も利用頻度が高いバージョンは、最新のバージョンとする。また、更新は要素毎(開始タグから終了タグまで)、あるいは内容のみの変更とする。変更は、前のものを削除してから、挿入することで実現する。

利用するXMLは整形式(well-formed)文書であるとする。

3.2 基本形式

ルート要素<ROOT>の直下にある子は、<VERSION>要素のみである（図1参照）。

また、<VERSION>タグは"NO"属性を持つ。最も新しいバージョンは、newであり(図1の場合、new=5でもある)、以下1ずつ数字が減っていき、最も古いものが1である。

```
<?xml version="1.0" encoding="EUC-JP">
<ROOT>
    <VERSION NO="new">
        <SECB>こんにちは</SECB>
        <SECC>こんばんは</SECC>
                ⋮
    </VERSION>
    <VERSION NO="4">
        <INSERT start="2" end="4">
            <SECA>おはよう</SECA>
        </INSERT>
                ⋮
    </VERSION>
                ⋮
    <VERSION NO="1">
                ⋮
    </VERSION>
</ROOT>
```

図1 サンプルXML文書

<VERSION NO="new">以下には、最新の文書が完全に保存されている。

"NO"属性が"new"以外の場合、その直下にある子は、<INSERT>要素、<DELETE>要素のみである。これらのタグは、"start"属性および"end"属性を持つ。この属性の値は、タグ以下の要素が挿入(または削除)される位置を示し、SAXがXMLを読み込んだ際のイベントの番号で表される。

3.3 差分データの作成

差分データの作成にはdiffを用いた。以前の最

VerteX : A Versioning Technique for XML Documents
Tomomi Yoshimura†, Shohei Yokoyama‡, Manabu Ohta‡, Kaoru Katayama‡, Hiroshi Ishikawa‡
†Faculty of Engineering, Tokyo Metropolitan University
‡Graduate School of Engineering, Tokyo Metropolitan University

新の状態と、更新した状態を一時的にファイルに保存し、この２つのファイルの差分をとる。図１のサンプル XML を例にとる。diff の出力は図２に示す通りである。

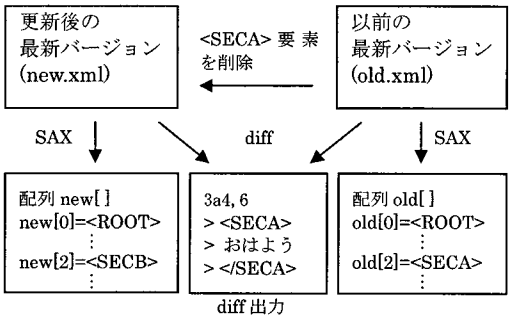

図 2 diff の流れ

先頭の行にある英字は、a ならば挿入を、d ならば削除を表し、数字はファイル内の行数を示す。また、１行目が XML 宣言であり、イベントは０から始まるため、この行数から２を引くと SAX のイベント番号になる。

ここでいう a(挿入)は、以前の最新バージョンにあって、更新後の最新バージョンにないものを表し、d (削除)はその逆である。これは、新しいものから古いものへバージョンダウンするパッチ、すなわちリバースパッチの考え方を取り入れているためである。

diff を適用した後、二つの一時ファイルを SAX で読み込んで配列に保存する。イベント番号が配列の添え字と等しいので、それにより更新された部分を取り出す。取り出された部分は、加えられた変更に応じて<INSERT>タグ、<DELETE>タグで囲われる。その際、これらのタグには位置情報が属性として付加される。

もし、以前の最新バージョンと差分データを比較して、差分データの方が大きかったら、以前の最新バージョンをそのまま保存する。

作成した差分データは、文書内の最新バージョンの後ろに付加する。

3.4 復元処理

復元は以下の手順で行われる。

① <VERSION NO="new">タグ内のものすべてを配列に読み込む。

② 復元したいバージョンが new でないならば、必要なバージョンまで文書を読み込む。

③ <INSERT>タグまたは、<DELETE>タグを読み込んだ場合、位置情報を用いて配列を操作する。<INSERT>タグの場合は、挿入位置以降を後ろにずらし、空いた部分に挿入を行う。<DELETE>タグの場合は、削除部分以降を前にずらす。

④ 必要なバージョンまで読み込み終わったら処理を終了する。

この手順から分かる通り、新しいものほど復元にかかる手間は少なく、処理は速い。

4 おわりに

本研究では、XML の特性を生かした XML のバージョン管理システムの開発を行った。

今後の課題としては、本研究においてシステム開発に PHP 言語を用いたため、ネームスペースに未対応である。これは、言語を JAVA などに移行することで対応が可能である。

また、差分データの作成に diff を用いたが、XML に適しているとは言い難く、変更が多数加えられると余分な差分データが作成され、復元がうまくいかないなどの問題が生じた。本システムでは、以前の最新バージョンより差分データの方が大きくなった場合、以前の最新バージョンを完全に保存するという対策をとった。しかし、これを完全に解決するためには、XML 用の比較関数を用意する必要がある。

差分データは、位置情報を含むため、非常に冗長性が高くなる。そこで、我々の研究室で研究している要素名圧縮手法[4]を利用し、圧縮を行えば、透過的なアクセスができるので、システムはほぼそのまま利用でき、XML の冗長性を省くことができる。

マルチウェイバージョン管理については、Web 上に XML 文書が存在する場合、URL の付加などを用いることを検討している。

謝　辞

本研究の一部は、文部科学省科学研究費特定研究領域(C)(2)「情報学:A02」(課題番号:13224078)による。

参 考 文 献

[1] 石川博: XML とデータベース, 情報処理学会誌, Vol.41, No.1, pp.68-73,2000

[2] S.-Y. Chien, V. Tsotras, and C. Zaniolo: Version Management of XML Documents, In Proc. of Web DB'2000

[3] S.-Y. Chien, V. Tsotras, and C. Zaniolo: XML Document Versioning, SIGMOD Record, Vol. 30, No.3, pp46-53, 2001

[4] 横山昌平, 太田学, 石川博: 要素名圧縮による xml データ圧縮手法の提案, DBWeb2000, pp.331-337,2000