[A5-3]

XML パーサを考慮した応用向き関係データベース構成法

横山 昌平　太田 学　片山 薫　石川 博

インターネットという緩やかな分散環境の下で XML データ形式は急速に普及している。XML をプログラムで扱う方法は、問い合わせ言語を用いる高レベルのアクセスとパーサを用いる低レベルのアクセスに大別することが出来る。前者については多くの関係データベース製品が XML の格納に対応していることから分かるように、問い合わせ言語を用いて XML データに問い合わせることが可能である。しかしながらパーサを用いる応用に適したデータベースの構成法についてはあまり議論されていない。我々はパーサを用いたシステムに関係データベースのトランザクション等の概念を導入することは非常に有用であると考えた。そこでイベント駆動方式のパーサと関係データベースを用いた WEB アプリケーションを想定し、XML 処理システムを構築している。本稿では XML 文書の関係データベース格納方式である Saxophone およびデータ更新のためのツールである xoDiff と xPatch を提案する。

Relational Database Construction Suitable for XML Parser Oriented Applications

Shohei Yokoyama　Manabu Ohta　Kaoru Katayama　Hiroshi Ishikawa

XML data have spread quickly in the distributed environment of the Internet. There are two methods to access to XML data: One is high level access through query languages over relational databases (RDBs). The other is low level access through XML parsers. Regarding the former, many database products provide XML support so users can query to XML data by using those RDBs. The latter is simple and fast way; however, we can find almost no RDB construction suitable for XML parser oriented applications. We think helpful XML handling system augmented with some concepts of RDBs. Therefore we have designed XML handling system using an event driven parser and RDBs. In this paper, we propose Saxophone which stores XML data into RDBs, and we also describe xPatch and xoDiff for versioning of XML.

1. はじめに

XML は登場とともにインターネットを席捲し、データ交換手段や蓄積方法として広く利用されている。XML が扱う情報が増えると同時に、それら膨大なデータをどのようにコンピュータで扱うかと言うことが問題になっている。XML データを処理する手法は大別して二つある。一つは問い合わせ言語などを利用した高次処理、もう一つはパーサを用いた低次処理である。前者は XML Query など XML に特化した問い合わせ言語が提案されている。後者についてはイベント駆動のパーサである SAX や XML 文書に従い木構造を構築する DOM パーサが様々なプログラム言語から使用できる。

一方、関係データベースは XML 形式よりも長い歴史を持ち、多くの企業内システムあるいは WEB サイトで利用されている。殆どの関係データベースシステムはすでにトランザクションやフォールトトレランスの概念等、データの蓄積や更新に欠かせない機能を実装している。それに対し XML は構造化文書の表現方法のみを規定した規格であることから、それ自体ではトランザクション等の処理に関する機能は持っていない。こうした状況下で関係データベースと XML の連携は前述したような関係データベースの機能を XML に付加する有益な方法であると考えられる。関係データベースと XML の連携はすでに幾つか提案がなされている。SuperSQL[1]を用いれば関係データベースと XML 間での相互変換を行うことができる。すなわち関係データベースに格納されたデータは XML ビューを持ち、そのビューを通し XML 問い合わせ言語による質問を処理することも可能である。XRel[2][3]では XML 文書の持つ木構造をノードで分割し、ノードのタイプに従い関係データベースに格納する。格納された各要素は木構造の経路を ID に持ち、経路表記を基に問い合わせを行うことが出来る。また Oracle 等の商用関係データベースも XML との親和性を謳っている。しかしながらこれらは前述した XML を処理する二つの方法で区分すると高次処理にあたり、パーサを用いた低次処理での関係データベースを用いた応用はあまり議論されていない。

我々はパーサを用いたシステムに、トランザクション等関係データベースが持つ機能を導入する事は非常に重要であると考えた。提案システムでは、XML ファイルをそれが内包するデータとスキーマ関わらず単に SAX イベントの列とみなし、関係データベースに格納する。格納した XML データはデータベース管理システムが持つ機能をそのまま適用出来る。その一方で、利用するユーザから関係データベースの存在を隠蔽し、SAX を用いて、データベースに格納された XML を呼び出すことが可能である。また関連技術は XML の格納と呼び出しを主体とした技術であるのに対し、提案システムは XML の異なるバージョン間の差分データを用い、構成された関係データベースの更新を行うことも考慮している。

東京都立大学大学院工学研究科電気工学専攻
Department of Electrical Engineering, Graduate School of Engineering, Tokyo Metropolitan University.

本稿の構成は以下の通りである。続く2節でデータベース構成法の概要と利用する既存技術について述べる。3節で提案システムについて説明し、4節ではWebアプリケーションへの応用について検討する。最後に5節でまとめと今後の課題を述べる。

2. 既存技術と提案システムの概要

2.1. 既存技術とその問題点

XML データはテキスト形式のファイルであり、それらの処理や更新に利用出来るツールは既に幾つかある。本節では提案システムに関連した既存技術とその問題点について述べる。

2.1.1. XML(eXtensible Markup Language)

XML はテキストファイルに構造化文書を記述する為の規格である。文字列を開始タグと終了タグで囲むことにより、要素とその内容を定義する。開始タグには名前と値からなる属性を付加することが出来る。これらタグは入れ子にすることができるため、階層構造を表現することが出来る。タグの名前や意味付けは完全にXML 文書作成者に任されている。

このように XML は作成の自由度が非常に高いため、複雑な構造化文書、大きな構造化文書を表すことができる。一方で、テキストファイル記述形式の定義であるため、ファイルという単位で制限されてしまう。例えば複数のXML文書はその文書個数分のファイルに分散して存在する。そしてそれらを一元管理する事は考慮されていない。提案システムでは関係データベースに XML 文書スキーマとは独立したテーブルを設け、あらゆるXML文書をそのテーブルに格納することにより複数リソースの一元管理を実現している。

2.1.2. SAX(The Simple API for XML)

SAX は XML のイベント駆動型パーサである。イベント駆動型パーサとは XML ファイルを先頭から読み、タグや文字データの出現に応じてイベントを返す方式で、高速な処理を特徴としている。SAX は XML ファイルの読み込みに特化された API 群で、関係データベースとの関連付けの議論はあまりされていない。提案システムでは SAX で関係データベースに格納された XML にアクセスすることが出来る。

2.1.3. diff、patch

diff は二つのテキストファイルの差異を表示するプログラムである。バージョンが異なるプログラムのソースコードを比較する為などに広く用いられている。また patch は diff の出力に基づき古いバージョンのテキストファイルを更新するツールである。XML データもテキスト形式であるため、diff と patch を用いて更新処理を行うことが出来る。しかしながら、これらは XML 以前の技術であり、行を単位として処理を行っており、XML の構造を理解することは出来ない。本稿では diff と patch に XML の構造を理解させる手法を提案する。

2.2. 提案システムの概要

提案システムは複数 SAX パーサからの呼び出しを考慮した関係データベースの構成法である。

提案システムは4つのツールから成り立っている。それらの相関関係を図1に示す。Saxophone は XML ファイルを関係データベースに格納するためのツールである。xoDiff は異なるバージョン間の差分取得ツール、xPatch は差分を利用したデータベース更新ツールである。また提案手法で構築されたデータベースは Saxophon SAX パーサを用いてアクセスすることが出来る。それらは関係データベースや前述した SAX、patch、diff のような既存のツールを利用し、それらが持っている機能を XML 処理に適用するべく設計されている。

2.2.1. Saxophone

XML データを関係データベースに格納する際に問題になるのはその構造の違いである。XML は木構造であるのに対し関係デ

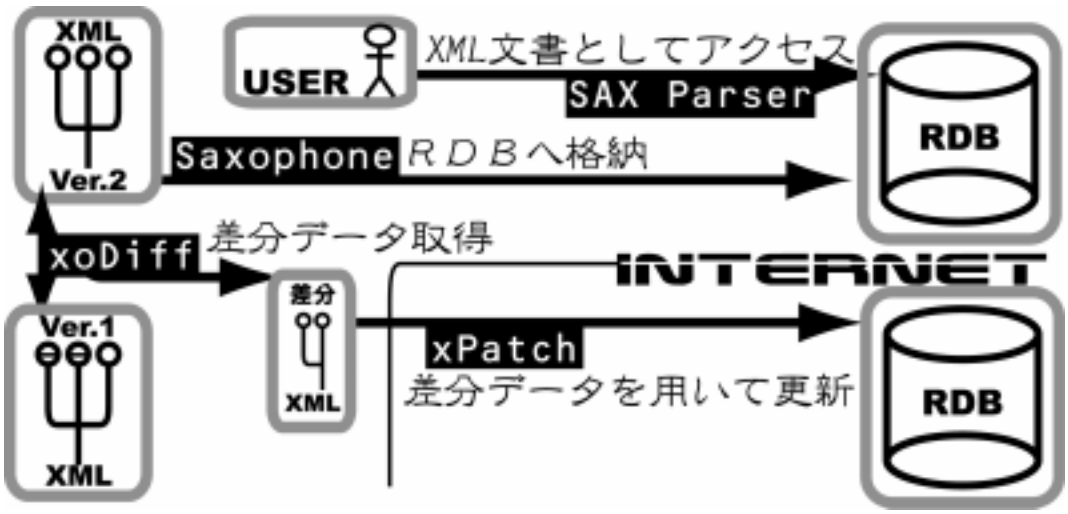

図1 提案システムの相関図

ータベースのデータモデルはテーブル構造である。Saxophone ではSAX イベントの順序に従いXML を時間軸に沿った二次元表形式に変換し、関係データモデルに展開している。この SAX イベントに従った二次元形式のデータモデルを Saxophone データモデル（SaxDM）と呼ぶ。また Saxophone は関係データベースに格納されたXML にアクセスするためにSAX パーサを提供する。

2.2.2. xoDiff

前述したように diff は XML 構造を理解しない。また出力はXML の規格に則らないものとなる。そこで XML を SaxDM に則ったデータモデルに変換しそれを diff の入力として利用すれば、出力として XML ファイルを得ることが出来る。この差分ファイルはXML 形式であるだけでなく、patch プログラムの入力としても動作する。この diff のラッパープログラムを xoDiff と呼ぶ。SaxDM に変換したXML 文書同士なら xoDiff の出力を用いて patch プログラムで更新することができる。

2.2.3. xPatch

xoDiff の出力を基に関係データベースに格納されたXML データを更新するツールである。また関係データベースが持つトランザクションの機能により、更新プロセスは別の更新や呼び出しのプロセスから隔離することが出来る。

3. 提案方式

3.1. XML の関係データベース格納

3.1.1. Saxophone データモデル(SaxDM)

XML データは“要素開始” “文字列データ” “要素終了”の3種類の SAX イベントに分解することが出来る。提案方式ではこれらに加えて“属性出現”も一つのイベントとして扱う。属性情報はSAX では要素開始イベントのプロパティとして現れる。イベントに

イベント	プロパティ
要素開始	要素名
属性出現	属性名 / 属性値
文字列データ	文字列
要素終了	要素名

表1 Saxophone イベントとそのプロパティ

<Dictionary><meaning lang=”Ja” char=”kanji”>辞書</meaning></Dictionary>

イベント ID	イベントタイプ	イベントプロパティ
1	要素開始	名前=Dictionary
2	要素開始	名前=meaning
3	属性出現	名前=char / 値=kanji
4	属性出現	名前=lang / 値=Ja
5	文字データ	文字列=辞書
6	要素終了	名前=meaning
7	要素終了	名前=Dictionary

図2 XML データ例(上)とその SaxDM（下）

分解することにより XML データはイベント順序を軸に持つ一次元構造をとる。表1に各イベントとそのプロパティについて示す。

なお要素に複数の属性値がある場合、属性名で辞書順にソートし、それを順番とする。図2に簡単な SaxDM の例を示す。

3.1.2. データベーススキーマ

SaxDM を関係データベースに格納する。Saxophone は現在PostgreSQL を用いて XML データを関係データベースに格納している。関係データベースのスキーマを図3に示す。

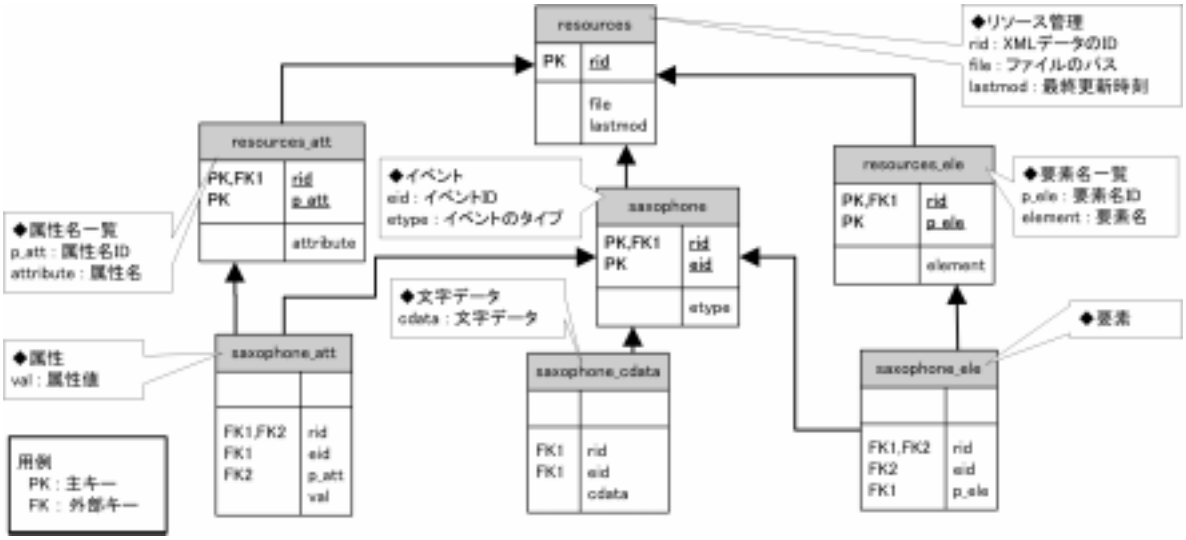

図3 SaxDM を格納する関係データベーススキーマ

●**resources テーブル**

データベースに格納したXMLファイルを管理する為のテーブルである。入力ファイルのパスとリソースID(rid)、最終更新時刻を格納する。提案システムは複数のXMLファイルを同じデータベースに格納する為、入力ファイルを指すユニークな値 rid で各リソースを分ける。

●**resources_ele / resources_att テーブル**

要素名 / 属性名とそれを指すユニークな値を格納する。XML データの要素名や属性名に長さの制限はなく、またその性質上繰り返しが非常に多い。そこで要素名、属性名に固有の番号(p_ele / p_att)を割り振り、本データベースではその番号を持って要素名、属性名のIDとした。これによりディスクストレージのコスト削減を目指す。

●**saxophone テーブル**

イベント ID とそのイベントのタイプを管理する。このテーブルのイベントIDを昇順で順序付けることにより、入力された XML ファイルの SAX イベントの順序がデータベースでも保証される。リソースIDとイベントIDで主キーを構成する。

●**saxophone_ele/saxophone_att/saxophone_cdata テーブル**

イベントのプロパティを格納する。付加されたリソースIDとイベントIDを元に saxophone テーブルと結合させることにより図2のような SaxDM を構成する。

3.1.3. XML文書のデータベース格納

Saxophone は以下の手続きをもって XML ファイルを RDB に格納する。

1. 新しいリソースIDの取得
2. 格納する XML ファイルの絶対パスと最終更新時刻を取得し resources テーブルに格納
3. XML ファイルを SAX パーサで読み込み SaxDM に展開
4. SaxDM を Saxophone のスキーマに沿って分解(要素、属性、文字列IDを付加)
5. 各テーブルに格納する

Saxophone は resources テーブルで XML ファイルのパスと最終更新時刻を管理することにより、入力元となったファイルとリンクしている。詳しくは4節で述べるが、入力元 XML ファイルと Saxophone データベースは同期している。Saxophone に対応していない既存の XML 処理ツールにより入力元の XML ファイルを更新すると自動的に Saxophone データベースの内容も更新される。これはユーザから関係データベースを隠蔽するため仕組みである。

3.2. SaxDM の XML 表現

前節では SaxDM を関係データベースに格納する手法について述べた。ここでは SaxDM を XML 形式に則って表現する方法を説明する。この SaxDM の XML 表現は後述する差分取得ツール xoDiff で用いる。

SaxDM の XML 表現は saxophone 要素を根に持つ XML であらわされる。各イベントはそれぞれ要素であらわされ、イベントのプロパティはその属性、あるいは文字列としてあらわされる。書式と図2の XML データを用いた例は以下の通りである。

●要素開始イベント

<startElement name="*要素名*">

●属性出現イベント

<eventAttribute name="*属性名*" value="*属性値*" >

●文字データイベント

<eventCdata>*文字列*</eventCdata>

●要素終了イベント

<endElement name="*要素名*">

```
01: <saxophone mode="SaxDM">
02: <startElement name="Dictionary" />
03: <startElement name="meaning" />
04: <eventAttribute name="char" value="kanji"/>
05: <eventAttribute name="lang" value="Ja"/>
06: <eventCdata>辞書</eventCdata>
07: <endElement name="meaning"/>
08: <endElement name="Dictionary"/>
09: 0.17659800 1010207196.hi1.eei.metro-u.ac.jp</saxophone>
```

このように、各イベントを改行で分割することにより、XML 表現の行番号から 1 を引いた値は SaxDM のイベント ID に相当し、XML の木構造を行指向で表すことが可能となる。なお文字列に現れる改行は<n/>や<r/>に変換する。また、ルートタグの閉じる直前に作成日時(UNIX 時間)とホスト名を記述する。これは各バージョンの固有値の役割を持つ。

SaxDM の XML 表現は元となる XML と完全に等価であるが、サイズが長くなる傾向にある。インターネットを介した通信時などサイズを考慮する必要がある場合、本研究室で開発した圧縮ツール simplified element XML(seXML) [4]で圧縮する。

3.3. Saxophone データモデルの更新

提案システムで構築されたデータベースを更新する方法について述べる。現在、テキストファイルの更新には diff と patch というツールが良く用いられている。提案システムでは関係データベースをユーザから隠蔽するという目的から、テキストファイルに対する更新モデルと同等のデータベース更新モデルを提案する。

3.3.1. バージョン間の差分取得

前述したようにテキスト形式のファイルである XML データは diff を用いて比較することができる。diff は行指向の差異データ―取得ツールであることから、XML 文書のどの行が変更されているかが出力結果に示される。しかしながら XML 文書はタグを利用して階層構造を表すデータ形式であり、XML の行と階層には関連

```
<saxophone mode="xoDiff"> <saxophone>
— tmp/test_ronbun1.xml Sat Jan  5 23:29:11 2002
+++ tmp/test_ronbun2.xml Sat Jan  5 23:29:11 2002
@@ -3,7 +3,7 @@
  <startElement name="meaning"/>
  <eventAttribute name="char" value="kanji"/>
  <eventAttribute name="lang" value="Ja"/>
-<eventCdata>辞書</eventCdata>
+<eventCdata>辞典</eventCdata>
  <endElement name="meaning"/>
  <endElement name="Dictionary"/>
-0.94588300 1010240951.hi1.eei.metro-u.ac.jp</saxophone>
+0.95184400 1010240951.hi1.eei.metro-u.ac.jp</saxophone>
```

```
— doc/test_ronbun1.xml Sat Jan  5 24:30:11 2002
+++ doc/test_ronbun2.xml Sat Jan  5 24:30:11 2002
@@ -1 +1 @@
-<Dictionary><meaning lang="Ja" char="kanji">辞書</meaning></Dictionary>
+<Dictionary><meaning lang="Ja" char="kanji">辞典</meaning></Dictionary>
```

図5 xoDiff(上)とdiff(下)で生成された差分データの例

性が無い。例えば図 2 で示した XML データと、その中の meaning 要素を“辞書”から“辞典” に変更した XML データの二つのデータを使って差分をとることを考える。diff の出力結果は図 5(下)のようになる。結果の中に変更箇所とは関係のない Dictionary 要素や meaning 要素とそのの属性も現れており、XML のどの要素が変更されたかという情報は非常に得にくい。

提案手法 xoDiff は diff に渡す前に XML データを SaxDM に変換することによって、XML の階層構造に基づいた差異を取得する(図 6)。

まず xoDiff は二つの異なるバージョンの XML ファイルを入力にとる。それらを SaxDM の XML 表現に変更し、変更後の XML ファイルを diff の入力として渡す。diff の出力として得られたファイルはそのままでは XML ファイルの形式をとらない。そこで xoDiff は

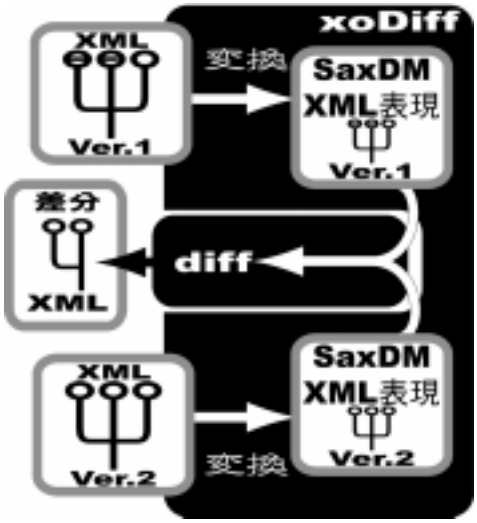

図6 xoDiff による差分データの抽出

その出力の先頭に以下の行を加える☆。

<saxophone mode=”xoDiff”> <saxophone>

前述したように入力データの行番号から 1 を引いた値は SaxDM のイベント ID に相当する。つまり図 2(上)の出力はイベント ID“2”から 7 イベント分の記述で、イベント ID“5”の文字データイベント“辞書”が“辞典”に変更されたことを表している。

3.3.2. 差分ファイルを用いた更新

xoDiff の出力を用いて SaxDM を更新するために xPatch というツールを開発した。本ツールを用いることにより saxophone データベースを更新することが出来る。さらに xPatch は XML ファイルの更新を行うことも可能である。

XML ファイルを更新する為に xPatch は patch プログラムを利用する。patch は、diff 出力の前に差分とは関係の無いデータが含まれていても自動的に無視する。よって diff 出力の前に現れる saxophone 要素の行は無視され、xoDiff の出力をそのまま patch の入力として動作する。patch に入力された SaxDM の XML 表現は差分データに基づいて新しいバージョンにアップデートされる。この際 xPatch は patch のラッパープログラムとして動作する。差分データは SaxDM に基づくものであるため、XML を一度 SaxDM のXML表現に変換する必要がある。また出力も SaxDM の XML 表現であるため、これを XML データに逆変換する必要がある。この変換を xPatch が担当する(図7下)。これらの仕組みにより、ユーザは diff や patch を用いた従来の方法と同じく、XML

☆ 例外として、diff の出力中に入力ファイルの1行目(1 行目は常に saxophone 要素の開始タグ)が現れる場合、xoDiff は saxophone 要素の開始タグを一つのみ付加する。

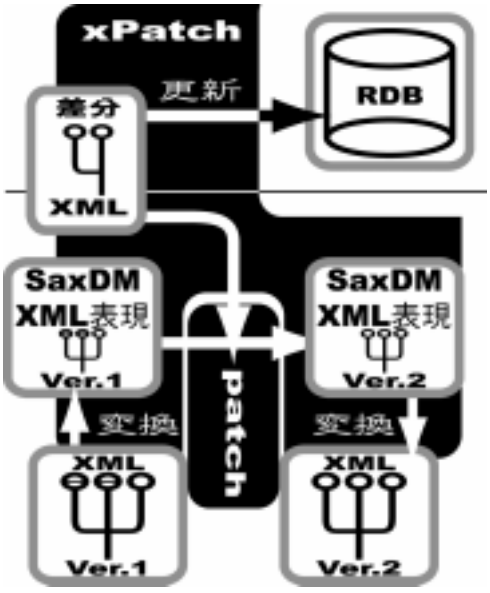

図7 差分ファイルによるRDB(上)とXML文書(下)の更新

の構造的差異に基づく更新ツールxoDiff/xPatchを利用することが出来る。

saxophonデータベースの更新はxoDiffが出力した差分データを基にSQLを生成しデータベースを更新する(図7上)。

xPatchは関係データベースのトランザクション機能を用いて更新を行うため、他のxPatch更新プロセス、或は他の読み出しプロセスとは完全に隔離されて実行される。

SaxDMはイベントIDという連番に依存してイベントの順序を関係データベース内で保証している。この際問題になるのは更新時のイベントID付け替えによる更新速度の低下である。例えばイベントID"1"にイベントが挿入された場合、リソースに関する全てのイベントIDを付け替えなければならない。提案システムでは小数点のイベントIDを用いることでこの問題を回避している。

提案システムの関係データベースでイベントIDはsaxophoneテーブルのeidというフィールドに格納されている。呼び出し時はイベントの順序を決定する為、このeidフィールドを昇順にソートする。例えばイベントIDが5つから成るXMLデータがsaxophoneデータベースに存在し、このデータに対しイベント3つの挿入を行う更新について考える(図8)。差分データがイベントID"2"から"4"への更新であるため、更新前のイベントID"2"以降のイベントID番号はすべて3を足す必要がある。この作業は更新する対象のXMLが大きければ大きいほど、また更新個所がイベントID"1"に近ければ近いほど、saxophoneデータベースを更新する為の作業コストは高い。そこで挿入するデータのeid値を、小数を用いた値に変更することにより、更新対象となるリソースのeid値を変更せずに更新手続きを行うことができる。図8の例では差分データのイベントID"2"から"4"をそれぞれ"1.1"から"1.3"に変更し、それをeid値として格納している。

多くの関係データベースでは4バイト及び8バイトの浮動小数点が扱える、しかしながらこれらの値も有限であり、いつかは枯渇してしまう。そこでアクセスの少ない時間帯、例えばWebアプリケーションで提案システムを利用するなら明け方の時間帯等にeidをイベントIDに同期させる更新を行う。

4. Saxophoneの利用法

前節までにXMLデータを関係データベースに格納し、またそれを更新する方法を述べた。ここでは関係データベースに格納されたデータをアプリケーションから呼び出す方法について述べる。

4.1. Saxophone SAXパーサ

提案システムはWebアプリケーション、特にサーバサイドスクリプトなどHTTPプロトコルを介したアプリケーションへの適用を目的としている。そこで主要なサーバサイドスクリプト言語の一つであるPHPスクリプト用にSAXパーサ(Saxophone SAXパーサ)を開発している。これはPHPに付属のSAXパーサのラッパーとして動作する。したがって、SAXを用いた既存のアプリケーションが提案システムを採用する際に、移行に関するコストはほとんど発生しない。Saxophone SAXパーサの動作手続きを図9に示す。

【更新確認ルーチン】

resourcesテーブル(図3)から対象となるSaxDMの最終更新日時と入力元XMLファイルのパスを取得し、そのXMLファイルの更新日時と比較する。ファイルが更新されている場合、更新ルーチンへ処理を引き継ぐ。更新されていない場合はデータベースをもとにSAXイベント送出する為、イベント発生ルーチンへ進む。

このうように提案システムではXMLファイルを関係データベースに格納した後も、入力元のXMLファイルとデータベースを同期

図8 更新時におけるeidフィールドの変化

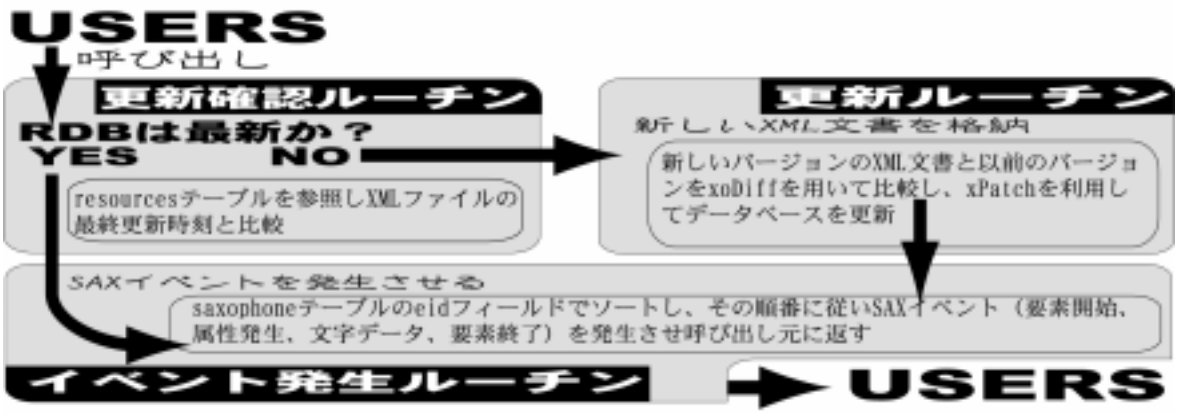

図9 Saxophone SAX パーサの動作手続き

させる仕組みを持つ。この仕組みにより、関係データベースを完全にプログラマから隠蔽出来るのみならず、前述した xoDiff と xPatch を用いた更新以外の、様々な XML ファイル処理システムと連携させることが出来る。

【更新ルーチン】

イベント発生ルーチンと同期して動作することにより入力 XML ファイルを SaxDM に変換し、データベースを更新すると同時に SAX イベントを発生させる。この手続きは関係データベースのトランザクション機能により、他のセッションから隔離されている。

【イベント発生ルーチン】

saxophone テーブル(図3)を元に SAX イベントを発生させ呼び出し側に返す。イベント ID を昇順でソートすることにより SAX イベントの順序を保つことが出来る。

このような手続きにより提案システムは SAX と等価なインターフェイスを持ちつつ関係データベースの機能、XML ファイル・関係データベース間でのデータ自動同期機能を取り入れた処理を実現することが可能である。そして、Saxophone にて関係データベースに格納されたデータは、データベースが持つバックアップ、障害回復などの機能を利用することが出来る。ファイルベースのデータ処理から関係データベースを用いた処理へと発展した動機は、多ユーザの大量アクセス時にデータ処理をスムーズに行う為である。インターネットこそまさしくその最たる例ではないだろうか。XML は自己記述型言語でその応用は様々な分野に及ぶ。しかしながら XML はファイル形式の規格であり、テキストファイルとしての存在しか考慮されていない。提案システムを用いて、XML データを関係データベースに格納し、またそのデータを SAX パーサから利用できるようにすることにより、多ユーザの大量アクセスに対応してきた関係データベースの機能を XML 処理に適用することが可能となる。

4.2. アプリケーションからの利用

XML データを関係データベースに格納することにより、三層サーバ・クライアントシステムを構築することができる。本節ではオンラインショップを例に Saoxphone を利用した3層システム(図10)について述べる。提案手法のサーバ・クライアントモデルは RDB サーバ層、HTTP デーモン層、クライアント層からなる。図10の例では、商品一覧の XML データを格納した公開用の Saxophone データベース、サーバサイドスクリプトや Saxophone SAX パーサと連携する HTTP デーモン、ブラウザソフトの3層と、商品の在庫管理などを行う基幹サーバから成る。基幹サーバはオンラインショップに登録する商品の情報を XML ファイルで Saxophone データベースサーバに送る。ユーザがブラウザを用いて商品一覧を閲覧する時、アクセスを受けた HTTP デーモンはサーバサイドスクリプトを実行させる。サーバサイドスクリプトは Saxophone SAX パーサを用いた XML ファイルを呼び出す。Saxophone SAX パーサはプログラムで指定されたファイル直接読むのではなく、

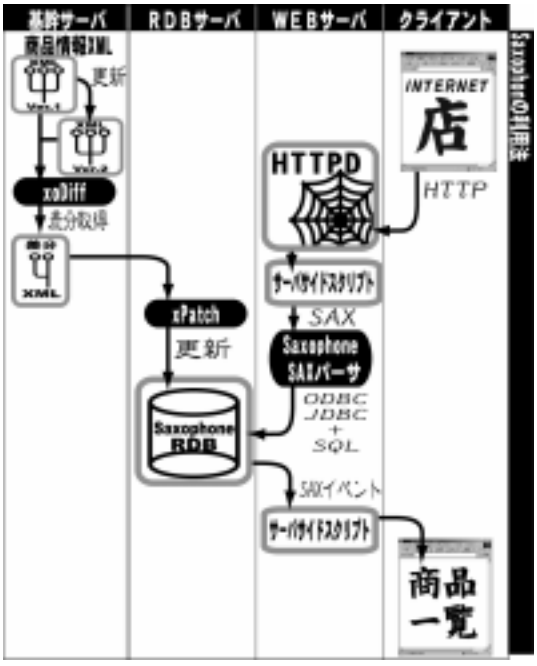

図10 Saxophone を利用したアプリケーションの例

Saxophone データベース上のデータに基づき SAX イベントをサーバサイドスクリプトに返す。この際の手続きは前節で述べた通りである。またこれら関係データベースへのアクセスは Saxophone SAX パーサによりユーザやプログラマから隠蔽されている。サーバサイドスクリプトは SAX イベントに基づき商品一覧の HTML ページを構成し、それをユーザに送信する。

Saxophone データベースの更新の手続きは差分データに基づく。基幹サーバで商品情報 XML ファイルに変更が発生した時、xoDiff を用いてその変更前と後との差分データを取り、それを Saxophone データベースサーバに渡す。データベースサーバは xPatch を用いて差分データを反映させる。

このように提案システムを用いることにより、SAX を利用した XML 処理システムで、関係データベースを用いる3層サーバ・クライアントシステムを構築することが可能である。

5. まとめと今後の課題

本稿では木構造の XML データを SAX イベントに従ってイベントの順序を軸とする SaxDM に変換し関係データベースに格納する方法 Saxophone を提案した。また XML データの差分データを取得し、それをもとに Saxophone データベースを更新するツール xoDiff、xPatch を提案した。また Saxophone にて構築されたデータベースを用いたアプリケーションについて検討した。

本論文で提案した手法の利点としては、

① 現在、関係データベースと XML の連携は XML 問い合わせに特化された問い合わせ言語を用いる方法等、XML や関係データベースのスキーマに依存した方式とであると言える。提案方式ではパーサレベルで XML を関係データベースに格納する。よって個々のデータのスキーマに依存せず、様々なコンテンツを一つのデータベースで管理することが出来る。

② 関係データベースのトランザクションやフォールトトレランス等の機能を XML 処理に容易に適用することが出来る。また提案方式で構築されたデータベースは ODBC 経由で問い合わせ可能であり、分散環境を考慮している。

③ SaxDM の XML 表現を diff の入力にすることにより、XML の構造的変更に則った差分データを得ることができる事を示した。また差分データを用いて、提案方式で構築されたデータベースを更新することができる。

④ Saxophone SAX パーサを使用することにより、プログラマは関係データベースの存在を意識することなく XML を利用したアプリケーションを開発できる。

⑤ SAX パーサを用いたアプリケーションで3層サーバ・クライアントシステムを構築できる。

などの点があげられる。

また、今後の課題として以下の事項があり、今後これらの課題について取り組む必要がある。

- 提案手法の評価を行う為に、通常の SAX 処理との比較を考えている。SAX は高速を特徴とするパーサであるため、提案システムの Saxophone SAX パーサと特に速度面で定量的な比較をする必要がある。またデータベースにおいても読み出し速度に特化した最適化を考慮する必要がある。
- XML をデータベースに格納することによってトランザクション等の機能を受動的に利用する方法は本稿で述べた。今後はこれらの効率化を図るとともに、データベースの機能を能動的に XML 処理に適応する事について考える必要がある。具体的には、データベースのバックアップ機能をもちいて任意のバージョンの復元、あるいは任意のバージョン間の差分データを取り出すようなシステムを考えている。
- 本研究室で提案している XML 圧縮技術 seXML や XML 問い合わせ言語 Quiz[5)6)]と連携させ、XML 統合処理環境へと発展させる予定である。

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。ここに記して深謝の意を表す。

参考文献

1) 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. SIG 8(TOD-10), pp.66-95 (2001).

2) 吉川 正俊, 志村 壮是, 植村 俊亮 ``オブジェクト関係データベースを用いた XML 文書の格納と検索'', 情報処理学会論文誌: データベース第40巻, 第SIG6(TOD3)号, pp. 115-131 (1999)

3) Masatoshi Yoshikawa, Toshiyuki Amagasa, Takeyuki Shimura and Shunsuke Uemura: ``XRel: A Path-Based Approach to Storage and Retrieval of XML Documents using Relational Databases'', ACM Transactions on Internet Technology, Vol. 1, No. 1, pp. 110-141 (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 –Simplified Element XML-, データベースと Web 情報システムに関する IPSJ DBS/ACM SIGMOD Japan Chapter.JSPS-RFTF AMCP 合同シンポジウム(DBWeb2000)

5) 横山昌平, 太田学, 石川博: Quiz: 分散環境を想定した XML 問い合わせ言語, 第62回情報処理学会全国大会, 6W-5(2001)

6) 横山昌平, 太田学, 石川博: エンドユーザ志向の XML 問い合わせ方式, データ工学ワークショップ(DBWS2001), 電子情報通信学会技術研究報告, pp.245-252 (2001)