RDB を利用した XML リポジトリにおけるアクセス管理の実装

横山 昌平† 太田 学† 片山 薫† 石川 博†

†東京都立大学大学院 〒105-0123 東京都八王子市南大沢 1-1

E-mail: †{yokoyama,ohta,katayama,ishikawa}@hikendbs.eei.metro-u.ac.jp

あらまし 本稿では関係データベースを用いた XML リポジトリにおけるアクセス管理の実装について議論する．XML リポジトリは我々が開発中の SAXOPHONE を用いる．SAXOPHONE は SAX イベントに基づいたスキーマを持ち，あらゆる XML 文書を DTD 等の有無に関わらず一元的に管理する事ができる．しかしながら，複数の XML 文書を一つの関係スキーマに格納する場合，関係データベースのアクセス管理機能を個々の XML 文書に対して適用させる事は困難であった. 提案手法ではストアドプロシージャと Prefix Labelling Scheme を用いる事により, XML 文書に対するアクセス管理を，関係データベースのアクセス管理機能を用いて実装している．また本手法は，インタフェースに SAX，実装に SQL を用いており，利用にあたり DBMS やアプリケーションの拡張は必要としない.

キーワード XML，アクセス管理，RDB，プレフィクスラベリング

Access Control Implementation with XML Repository using RDB

Shohei YOKOYAMA† Manabu OHTA† Kaoru KATAYAMA† and Hiroshi ISHIKAWA†

†Tokyo Metropolitan University 1-1 Minami-Osawa, Hachioji-shi Tokyo, Japan 192-0397

E-mail: †{shohei,ohta,katayama,ishikawa}@hikendbs.eei.metro-u.ac.jp

Abstract This paper describes an access control method of the XML repository system, SAXOPHONE, which was implemented at Tokyo Metropolitan University. The main feature of our research is a novel account identifier that is based on the prefix-labeling scheme to realize a hierarchical authorization. SAXOPHONE uses relational databases for XML document storage. Using it, any valid or well-formed XML document is decomposed into events of the SAX parser and is then stored into relational tables using a fixed scheme. Consequently, users can handle the system as a normal SAX parser. This study also illustrates how to realize an access control method of XML documents on relational databases using the account identifier.

Keyword XML，Access Control，RDB，Prefix Labelling Scheme

1. はじめに

半構造のデータを記述する XML(eXtensible Markup Language)はインターネットと親和性の高い言語として，データの交換や蓄積によく用いられている．XML はその汎用性の高さから様々な分野で用いられているが，本稿ではデータ蓄積用途を前提に，アクセス管理を実装する手法について議論する.

データの蓄積に XML を用いる研究は様々な粒度で様々な議論がなされている．我々も関係データベースを用いて XML データをスキーマに関わらず一元的に蓄積する手法 SAXOPHONE を提案してきた. 本稿ではこの SAXOPHONE 上におけるアクセス管理について議論する．本手法は，インタフェースに SAX，実装に SQL を用いており, 利用にあたり DBMS やアプリケーションの拡張は必要としない.

提案手法の特徴はあるアカウントのアクセスポリシーを別のアカウントが継承できるという点である．我々はこれを Hierarchy authorization と呼んでいる．これを実現するために，アカウントの識別子に Prefix Labelling Scheme を利用している.

本稿の構成は次の通りである．続く２節では提案手法の概要と関連研究について述べる．３節では提案するアカウント識別子を説明する．４節ではアカウント識別子を用いて階層的なアクセス管理を行う手法について述べ，５節にて認証プロセスを説明する．また６節では一般的なアクセス管理手法を考慮した議論を行う. 最後に７節でまとめと今後の課題について述べる.

2. 提案手法の概要と関連研究

2.1. 提案手法の概要

アクセス管理は非常に広い意味を持つ言葉である．以下，本研究におけるアクセス管理の定義と，提案手法の概要について記す.

本論文は XML リポジトリ上でのアクセス管理，特にその実装についての提案である．提案手法は XML 文書中の任意の要素，属性あるいは文字列に対し，任意のユーザからのアクセスを拒否する仕組みを提供する. ユーザはアクセスが拒否されたノード以外の XML

文書を SAX パーサによって受け取ることができる．

本研究でアクセス管理を実装する SAXOPHONE は関係データベースを利用した XML リポジトリである．関係データベースはすでに長い歴史があり，大量のデータを効率的に処理するために最適化されている．一方で XML は半構造データの表現形式を定めた規格であり，データの処理については規定されていない．そこですでに大量データの処理に実績のある関係データベースを XML リポジトリとして利用することにより，関係データベースの持つ様々な機能をそのまま XML リポジトリの機能として利用することができる．例えば多くの関係データベースはトランザクションの機能やアクセス管理の機能をすでに実装している．

また，多くの企業において関係データベースはデータ処理の中核として利用されている．データやセキュリティの一元性などの観点からもすでに運用されている関係データベースと XML リポジトリは統合的に利用すべきであると考えられる．我々が開発している SAXOPHONE は関係データベース上に構築された XML リポジトリであり，XML のスキーマにかかわらず様々な XML 文書を一元的に管理する事ができる．

我々はデータの一元化のみならず，アクセス管理の一元化も重要な課題であると考えた．提案手法では DBMS のアカウントを用いて SAXOPHONE にアクセスする．DBMS のアカウントは同時に XML リポジトリのアカウントとして機能する．このことから，RDBMS と XML リポジトリのアクセス管理は一元化される．

一般的にアクセス管理はデータとは別個にアクセスポリシーを記述し，ユーザから要求があった際にアクセスポリシーに従って,データの処理を行っている．それに対し提案手法ではアクセスポリシーはデータと同じテーブルにデータと同じスキーマで格納されている．またポリシーやデータが格納されている関係データモデルから XML 文書として読める形式への変換は SQL の問合せによって表されている．つまり格納された XML 文書へのアクセスは常に一つの SQL ステートメントを介して行われている．そしてアクセスポリシーの処理がこの問合せ中に SQL として記述されているため，アクセスポリシーを適用せずに XML データを得ることは不可能である．

2.2. 関連研究

提案手法の基礎となる SAXOPHONE は関係データベースを用いた XML リポジトリである．XML の格納に関係データベースを利用する手法は多くの提案がある．例えば xRel[1]では XML のノードを分解し，そのノードを指す XPath とともに relational table に保存することにより，スキーマを意識しない XML リポジトリを構築している．木構造である XML 文書を表形式

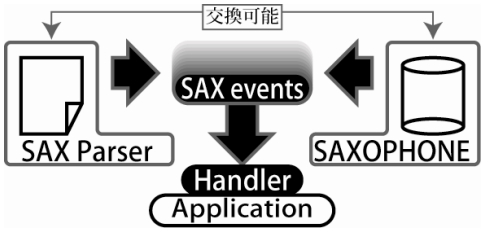

図1: SAXとSAXOPHONEの概要

の関係データベースに格納する際, XML 木を分割して平坦化しなければならない．xRel は木のノードで分割するのに対して SAXOPHONE では SAX イベントで分割している．

XML のアクセス管理に関する研究も議論が多い．XACML[2]は XML 文書全体あるいは個々の要素に対しアクセス管理を行う事を目的として作られたアクセスポリシー記述言語である．この提案はアクセスポリシーの記述に関してのみの議論であり，実際の XML リポジトリに対する実装に関しては触れられていない．

本研究では XACML のような高レベルでのアクセスポリシー記述は議論しない．むしろそれらの技術と利点を補完する研究であると位置づけている．

SilkRoute[3]や XPERANTO[4]では関係データベースに格納されたデータを XML 形式で publish する手法を提案している．これらの研究が扱うのは関係データモデルであり, XML のような順序木を扱う事はできない．

Web アプリケーション等の環境における XML 文書のアクセス管理としては Damiani らの研究[5]が挙げられる．この研究では XML 文書におけるアクセスポリシーの記述と実装手法を述べている．しかしながら，XML 木のノードに対するアクセス管理の実装であり，様々なスキーマの XML 文書が混在する XML リポジトリにおけるアクセス管理の議論がされていない．

XML 文書のアクセス管理を考える際, 我々はポリシーの記述だけでなく，それをどのように XML リポジトリに適用させるのかを同時に実現する必要があると考えている．例えば，XML ファイルに対するアクセス管理を考える．アクセス管理の実装系を用いてユーザがアクセスすると，アクセスポリシーに従った結果が得られる．ところが通常のファイルシステムを利用した場合，アクセスポリシーに関係なくユーザは XML ファイル全体を読み込むことができる．この事を防ぐためにはファイルシステムのアクセス管理と XML 文書のアクセス管理を一元化する必要がある．

XML 文書が関係データベース内にある場合も同じ事が言える．提案手法では，DBMS のアカウントがそのまま XML リポジトリのアカウントとなり，またポリシーで禁止された項目へのアクセスは，通常の RDB フロントエンドプログラムを用いても拒否される．

3. SAXOPHONE：XML リポジトリ

本節では提案手法の基礎となる XML リポジトリ SAXOPHONE について述べる．まず，汎用的な XML パーサである SAX について説明する．

3.1. SAX パーサ

SAXOPHONE は SAX イベントに基づいて XML 文書を平坦化し，関係データベースに保存する．SAX パーサは広く利用されているイベントドリブン型の XML パーサである．アプリケーションはイベントハンドラと呼ばれるクラスに XML の処理を記述し，SAX パーサに渡すことで動作する．イベントハンドラには各 SAX イベントに対応するメソッドが存在し，SAX パーサは読み込んだ XML 文章を先頭から走査し，その構造を発見した順番でイベントハンドラの対応するメソッドに渡す．

この仕組みは，一回の走査で処理が完了する手続きや，CGI 等のセッションレスのプログラムからの利用に適している．また SAX を使って DOM を構築することにより，両者の利点を合わせたプログラムを作成することも可能である．

3.2. SAXOPHONE のスキーマ構成

SAXOPHONE は複数の XML 文書をスキーマに関係なく保存する手法で，プログラムからは SAX パーサとして利用できる．つまり，プログラムは SAX パーサと同様のイベントハンドラを SAXOPHONE に渡すことにより，SAX イベントを得ることができる．このことから，既存の SAX パーサから提案手法への移行は非常に容易である．

XML 文書と SAXOPHONE 内に存在するデータは同値であり互いに変換可能である．SAXOPHONE の仕組みを図 1 に示す．アプリケーションは，呼び出すリソースが XML ファイルであるか，あるいは SAXOPHONE の関係データベース内にあるかを意識せず，イベントハンドラを構築する事ができる．

SAXOPHONE を構成する基本的な関係データベーススキーマを以下に記す．

event (rID, eID, type, property)
resource (rID, URI)

SAXOPHONE にはこの他に圧縮を考慮したスキーマ[6]や版管理機能を持つスキーマ[7]等があるが，本稿はアクセス管理の議論が目的であり，ここでは扱わない．またスキーマに関する詳細は文献[7]に詳しい．

3.3. SAXOPHONE のインタフェース

SAXOPHONE 内の XML 文書は SAX パーサを用いてプログラムから呼び出すことができる．例えば java の SAX パーサを利用して URI で与えられた XML 文書を呼び出す処理は次のように記述される．

```
01. //Obtain an instance of SAXParserFactory.
02. SAXParserFactory spf = SAXParserFactory.newInstance();
03. //Obtain an instance of a parser from the factory.
04. SAXParser sp = spf.newSAXParser();
05. //Parse the document.
06. sp.parse(URI, new mySaxEventHander());
```

SAXOPHONE もユーザからは SAX パーサとして見えるため，まったく同じコードでリポジトリに格納された XML ファイルを呼び出すことができる．

SAX は広く普及した XML パーサであり，多くのユーザ，開発者が利用方法を習熟していると考えられる．提案手法はこの SAX と同じインターフェースを持つため，新たな問合せ言語の習得などを必要としない．

次に与えられた URI に基づいて関係データベースから SAX イベントを再構成する問合せを記す．この問合せは SQL で記述される．

```
01. SELECT      type, property
02. FROM        event
03. WHERE       rID = (SELECT     rID
04.                    FROM       resource
05.                    WHERE      URI = U)
06. ORDER BY    eID
```

これは非常にシンプルな問合せであり，これにより SAXOPHONE は様々な DBMS 上で動作させることができる．XML 文書と関係データモデルの一番大きな違いは，XML が順序木であるのに対し，関係データモデルは行の順序が保障されていない．そのため XML 文書を RDB に格納する際には，XML のノードの順序も記憶させなければならない．SAXOPHONE では SAX のイベント順に昇順の ID を割り当てている．これをイベント ID (eid) と呼んでいる．また問合せの際にイベント順を保障するため eID でソートを行っている．このソートは処理のボトルネックになる可能性があるが，eID に INDEX を作成する等，DBMS の機能を使いクエリを高速化できる．

さて，上記の問合せによりイベント順に並べられた SAX イベントの列を得ることができる．このイベント列に基づき，ユーザから与えられたイベントハンドラの対応するメソッドを呼び出す．このことにより，SAXOPHONE は SAX パーサとして動作する．

4. Hierarchy authorization とアカウント識別子

4.1. 提案手法におけるアクセス管理の概念

提案手法ではアカウント名とパスワード使ったシンプルな認証手続きを行う．例えばインターネットショップサイトで利用する事を考える．商品情報を格納する XML リポジトリにはショップ管理者と顧客がアクセスするとする．ショップ管理者は在庫状況や原価の情報等を顧客から隠す必要がある．そこでショップ

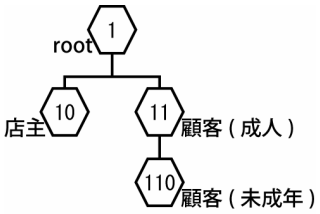

図2: アカウント識別子の構造

table: **event**

field	type	note
rID	integer	resource ID
eID	float	event ID
aID	varchar	account ID
depth	integer	length of account ID
type	integer	SAX event type
property	varchar	SAX event property

table: **resource**

field	type	note
rID	integer	resource ID
URI	varchar	URI of a resource

table: **account**

field	type	note
aID	integer	account ID
account	varchar	account name
password	varchar	password for the aID

図3: データベーススキーマ

管理者と顧客のアカウントを作成し，読み込み可能な情報を制限する．また顧客には未成年がいる，アルコール類を扱っているならば，その情報を未成年から隠す必要があるかもしれない．そこで顧客アカウントが持つポリシーを引き継いで，かつ未成年に適用するアクセスポリシーを付加したアカウントを作成する．

このように提案手法ではアクセスポリシーの継承関係は木構造となる(図 2). 我々はこのようなポリシーの構造とそれを用いた認証を Hierarchy authorization と呼んでいる．

4.2. アカウント識別子

アカウントはアカウント名とパスワード，そしてアカウント識別子(aID)を持つ．システム内部ではアカウント識別子によってポリシーが表現される．アカウント識別子は Hierarchy authorization を扱うため木のラベリング手法である Prefix Labelling scheme を用いている．Prefix Labelling scheme は，木の祖先にあたるノードの識別子は自身の接頭辞であるというシンプルなルールをもったラベリング手法で，木の祖先・子孫関係を探索するのが容易であるという特徴を持つ．

提案手法では根を 1 とする識別子でアカウントを区別する．図 2 の数字は提案手法により付与されたアカウント識別子である．A というノードの x 番目の子供 B_X のアカウント識別子は次の式によって与えられる．

$$B_x = 10^{\left\lfloor \frac{x+9}{9} \right\rfloor}(A+1) + (x \bmod 9) - 10$$

例えば 100 というアカウント識別子を持つノード A の子供 B_8 と B_9 は次のように算出される．

$$B_8 = 10^1(100+1)+8-10 = 1008$$
$$B_9 = 10^2(100+1)+0-10 = 10090$$

識別子には木の兄弟方向に向かって数値が増加し，高さ方向に向かって桁が増加する十進数を用いている．本手法では”9”の文字をオーバーフローフラグとして利用することにより，識別子の枯渇を防いでいる．

4.3. クエリ

Hierarchy authorization ではあるアカウントのアクセスポリシーはそのアカウントの祖先のアカウントポリシーを全て引き継いでいるため，祖先に遡りポリシーを収集する必要がある．これは SQL で容易に表現できる．

```
01. SELECT      aID, property
02. FROM        table
03. WHERE       1 = CHARINDEX(aID,'110')
```

CAHRINDEX(*Substring*, *String*) は Microsoft SQL Server 固有の関数で，Substring の String 中における位置を返す関数である．同様の関数は多くの DBMS で用意されている．例えば postgreSQL には position(*Substring* in *String*)関数が存在する．上記問合せは 110 の接頭辞となる aID を持つ値全てを呼び出している．

5. 提案手法と認証プロセス

本節では提案手法の実装と利用について述べる．まず提案手法の関係データベーススキーマについて述べる．

5.1. 関係データベーススキーマ

提案手法は SAXOPHONE を基にしており，関係データベーススキーマも SAXOPHONE を拡張して実装している．スキーマ構成を図 3 に示す．フィールド depth の値は aID の長さから動的に導き出している．

5.2. アクセス制限

アクセスポリシーはデータに付加される．説明の為，単純な XML 文書を定義する．

```
01. <LIST>
02.     <お取り置き/>
03.     <ジュース/>
04.     <コーラ/>
05.     <ビール/>
06. </LIST>
```

event

rID	eID	aID	depth	type	property
1	1	1	1	要素開始	LIST
1	2	1	1	要素開始	お取り置き
1	2	11	2	NULL	
1	3	1	1	要素終了	お取り置き
1	3	11	2	NULL	
1	4	1	1	要素開始	ジュース
1	5	1	1	要素終了	ジュース
1	6	1	1	要素開始	コーラ
1	7	1	1	要素終了	コーラ
1	8	1	1	要素開始	ビール
1	8	110	3	NULL	
1	9	1	1	要素終了	ビール
1	9	110	3	NULL	
1	10	1	1	要素終了	LIST

図4: アクセス制限を含む格納データ

event

rID	eID	aID	depth	type	property	
1	1	1	1	要素開始	LIST	
1	2	1	1	要素開始	お取り置き	
1	2	11	2	NULL		
1	3	1	1	要素終了	お取り置き	
1	3	11	2	NULL		
1	4	1	1	要素開始	ジュース	
1	4.1	11	2	属性出現	オレンジ味	annotation
1	5	1	1	要素終了	ジュース	
1	6	1	1	要素開始	コーラ	
1	6.1	11	2	属性出現	ダイエット	annotation
1	6.1	110	3	NULL		annotation
1	7	1	1	要素終了	コーラ	
1	8	1	1	要素開始	ビール	
1	8	110	3	NULL		
1	9	1	1	要素終了	ビール	
1	9	110	3	NULL		
1	10	1	1	要素終了	LIST	

図5: アノテーションを含む格納データ

これを図 2 に表されたアカウントからのアクセスを適切に管理する事を考える．店主アカウントは文書全体を見ることができる．顧客は『お取り置き』の品物は見ることができない，未成年の顧客は加えて『ビール』を見ることもできないとする．このアクセスポリシーを提案手法で表すと SAXOPHONE に格納されるデータは図 4 になる．

イベントに NULL とあるのは，SAXOPHONE が拡張した SAX イベントで,何もない事を表すイベントである．これは内部的に利用されユーザあるいはアプリケーションには渡されない．

提案手法はアクセス管理の実装法であり，より上流であるアクセスポリシー定義言語は持たない．本手法でのアクセスポリシーはデータと混在している．eID が同じコラムは，同じイベントとして捉えられる．例えば eID=8 のコラムは要素開始イベントと NULL イベントの二つを持つ．ここでもし呼び出し元のユーザの aID が 110 自身あるいは 110 を接頭辞にもつ場合，システムは NULL イベントを返す.それ以外の場合は“ビール”タグの要素開始イベントが返される．このように提案手法では NULL イベントは『read 禁止』ポリシーとして利用されている．

5.3. アノテーション

前節では特定のアカウントに対して XML 文書の一部にアクセス制限を付与する手法を説明した．提案手法ではその他，特定のユーザのみ読み込むことができるアクセスポリシーを設定する事ができる．これをアノテーションと呼んでいる．アノテーションには二種類あり，あるアカウント以下すべてのユーザが読み込むことができる Descendant local annotation と，任意のアカウントのユーザのみが読むことができる User local annotation がある．

5.3.1. Descendant local annotation

アノテーションはオリジナルの XML 文書に対し文字通り注釈を加えるための機能である．オリジナルの XML 文書に属する SAX イベントは，アカウント識別子のルートである”1”が aID フィールドに格納されている．一方でアノテーションは注釈を付与したユーザのアカウントが aID フィールドとして格納される．例えば，図 4 の XML 文書に対して，顧客(成人)がジュース要素の属性として，”オレンジ味”という属性を設定したとする．その場合，aID フィールドには顧客(成人)のアカウント識別子である”11”が格納される(図５)．

この属性は”11”を接頭辞に持つアカウントすべてから読み込むことができる．つまり，顧客(成人)が設定した”味=オレンジ味”という注釈は顧客(未成年)が呼び出した XML 文書内にも現れる．

5.3.2. User local annotation

一方で，注釈をしたユーザのみ呼び出しを許可するのが User local annotation である．これは上記 Descendant local annotation に加えて，子アカウントのアカウント識別子を持った NULL イベントを挿入する．

例えば顧客(成人)がコーラに User local annotation を行う例を図５に示す．ここでは顧客(成人)のアカウント識別子”11”を持つ属性出現イベント”味=ダイエット”と，顧客(未成年)のアカウント識別子”110”を持つ NULL イベントからなる．

顧客(成人)はこの注釈を呼び出すことができるが，その子孫である顧客(未成年)は呼び出せない．

5.4. SAX イベント列の生成

図５のデータを用いて SAX イベント列の呼び出し手法を説明する．

提案手法は SAX インタフェースを持つ．SAX を使って XML 文書を読み込む際，ユーザはその URL を指定し，コールバックを受けるイベントハンドラを設定する．提案手法も URL によって SAXOPHONE 内の XML 文書を呼び出すことができる．SAX 自体にはユーザ認証の機能は無い．しかし，アカウント名とパスワードを URI に埋め込んでシステムに渡すことにより，関係データベースのユーザ認証と提案システムのユー

ザ認証を行うことができる.

http://***account***:***password***@hostname/path/to/file.xml

この形式は SAXOPHONE の SAX パーサにより処理され, アカウント名, パスワード, URI に分解される.

アカウント名とパスワードを用いて account テーブルから aID を取得し, URI を用いて resource テーブルから rID を取得する.

続いてそのアカウントにおいて有効なイベントを取り出す. あるアカウント A において有効なイベントとは, 同じ eID を持つ複数のコラムがあったとして, A の接頭辞になりえるコラムのうち最も長い aID を持つものである. そしてその{eID,aID}をもつコラムが, A において有効なイベントを保持しているという事ができる.

この手続きは二つの問合せからなる. まず, aID の接頭辞検索を行い, 各イベントにおいて一番長い接頭辞を持つイベントを取り出す. 接頭辞を持つイベントの内, アカウント識別子が一番長いものは, アカウント aID において有効なイベントといえる.

```
01. SELECT      eID, MAX(depth) AS depth
02. FROM        event
03. INTO        #一時テーブル
04. WHERE       1 = CHARINDEX(aID, @aID)
05. AND         rID = @rID
06. GROUP BY    eID, rID
```

MAX(depth)は集約関数で depth すなわち最も長いアカウント識別子を持つものを返す. これをすべての eID に対して行う. 図 6 はそれぞれのアカウントからのアクセスにおいて有効と判断されたイベントを示した図である.

有効なイベントのキーが判明したので, それに基づいてイベントのプロパティを呼び出し, また eID を昇順にソートすることによって,イベント列を構成する.

```
07. SELECT      event.property, event.type
08. FROM        #一時テーブル, event
09. WHERE       #一時テーブル.eID = event.eID
10. AND         #一時テーブル.depth = event.depth
11. AND         1 = CHARINDEX (aID, @aID)
12. AND         event.rID = @rID
13. AND         type <> 0
14. ORDER BY    event.eID;
```

この問合せにより,任意のアカウント aID において, アクセスポリシーに従った SAX のイベント列を生成する事ができる. SAX イベント列は XML 文書と交換可能である. この問合せ結果を XML で表したものが図 7 である.

このように提案システムでは関係データベースの問合せ機能を利用して, アカウントに応じた XML データを提供する仕組みを実装できることを表した.

event

有効なイベント	rID	eID	aID	depth	type	property
● ■ ▲	1	1	1	1	要素開始	LIST
●	1	2	1	1	要素開始	お取り置き
■ ▲	1	2	11	2	NULL	
●	1	3	1	1	要素終了	お取り置き
■ ▲	1	3	11	2	NULL	
● ■ ▲	1	4	1	1	要素開始	ジュース
■ ▲	1	4.1	11	2	属性出現	オレンジ味
● ■ ▲	1	5	1	1	要素終了	ジュース
● ■ ▲	1	6	1	1	要素開始	コーラ
■	1	6.1	11	2	属性出現	ダイエット
▲	1	6.1	110	3	NULL	
● ■ ▲	1	7	1	1	要素終了	コーラ
● ■	1	8	1	1	要素開始	ビール
▲	1	8	110	3	NULL	
● ■	1	9	1	1	要素終了	ビール
▲	1	9	110	3	NULL	
● ■ ▲	1	10	1	1	要素終了	LIST

●= 店主／■= 顧客（青年）／▲= 顧客（未成年）

図6: 各アカウントでの有効なイベント

for 店主

```
<LIST>
  <お取り置き />
  <ジュース />
  <コーラ />
  <ビール />
</LIST>
```

for 顧客(成人)

```
<LIST>
  <ジュース　味=″オレンジ″>
  </ジュース>
  <コーラ　味=″ダイエット″>
  </コーラ>
  <ビール />
</LIST>
```

for 顧客(未成年)

```
<LIST>
  <ジュース 味=″オレンジ″>
  </ジュース>
  <コーラ />
</LIST>
```

図7: 各アカウントによる結果

5.5. DBMS 上における実装

最後に DBMS 上での実装について説明し, 提案手法と DBMS の一貫したアクセス管理について説明する.

図 8 は, データベースとアクセスポリシーインタプリタの二重で認証を行う手法(下)と, 提案手法による統合された認証の流れ(上)を表している.

XML 文書のアクセス管理とデータベースのアクセス管理を個別に行う場合, システムは二種類のアカウントを使用しなければならない. まず XML 文書に対

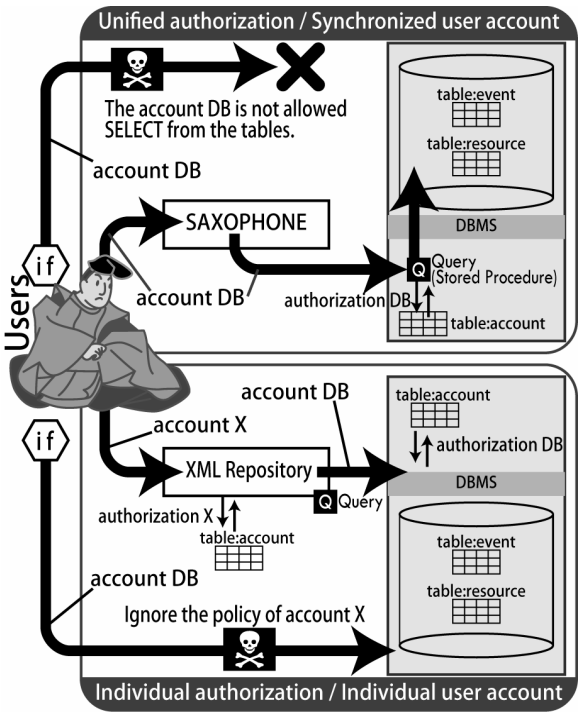

図8: ユーザ認証の仕組み

するアクセスポリシーを呼び出す為のアカウント(accountX)と，そのアクセスポリシーに従って実際のXML 文書をデータベースから取り出すためのアカウント(accountDB)である．

accountDB は適用されるポリシーに従ってデータベース全体から許可されたデータを探す．すなわち実際にユーザが持っている accountX の値とは関係なく，データベース全体を参照可能でなければならない．この事は，この手法を採用する全てのマシンからデータベース全体が参照可能であると言えよう．これはセキュリティー上の脆弱性となりうる．すなわち，ストレージ上での認証と XML アクセス管理の認証は一元化する必要がある．

提案手法では DBMS 上で定義されたアカウント(accountDB)を用いて XML 文書のアクセス管理を行っている．また，前節で述べたクエリは全てサーバ上に Stored Procedure として配置しており，accountDB は DBMS 上でこの Stored Procedure のみを起動できるようポリシーが設定されている．すなわち accountDB では event テーブルや resource テーブルを直接参照することはできない．

また Stored Procedure を起動する際，ODBC 等のフロントエンドによって accountDB の認証が行われる．つまり accountDB のアカウント名・パスワードが正しければ DBMS の認証をクリアした上，対応する aID に基づいて前節で述べた問合せが行われる．そしてその結果，すなわちアクセス管理された XML 文書のみがユーザ側に帰って来る．

6. 議論

関係データベースやオペレーティングシステムのアクセス管理で一般的な手法として Role Based Access Control(RBAC)[8]がある．この手法はユーザとアクセスポリシーを独立させ，アクセスポリシーを役割(Role)毎にまとめる手法である．ユーザは一つまたは複数の Role を持つ事ができる．

この手法ではユーザのアクセスポリシーの変更が容易であり，また複数の Role を用いて会社等の組織構造に合わせた階層的なアクセスポリシーの設定を行うことができる．

本論文ではユーザ個人がそれぞれのアクセスポリシーを持つとしており，アクセスポリシーの階層構造もユーザアカウント自体がノードを形成している．これは本論文が階層的なポリシーをどのように関係データベースで処理するかという部分に特化したためで，提案手法の運用に際しては RBAC の概念を取り入れる必要があると考えている．

RBAC の実装に関しては，role を管理するテーブルを新たに設け，アカウント識別子をロールに割り振り，ユーザアカウントはそのロールと関連づけることによって可能である．その詳細に関しては今後の課題としたい．

7. まとめと今後の課題

本稿では関係データベースを利用した XML リポジトリにおける XML 文書のアクセス管理手法を提案した．提案手法を用いることにより，XML 文書内の任意の要素，属性および文字列を任意のユーザから隠したり，任意のユーザのみに公開したりする事ができる．

また，このようなアクセス管理を行うため，Prefix Labeling Scheme に基づいたアカウント識別子を用いて，アクセスポリシーの処理を行っている．Prefix Labeling Scheme は木へのラベリング手法であり，先祖，子孫の検索が容易とされている．この利点を用いて，提案手法ではアカウント権限の継承を実装している．つまり，アカウントは階層構造をなし，より上位のアカウントのアクセスポリシーを継承する．

提案手法の基礎となる XML リポジトリ SAXOPHONE は関係データベース上に構築されており，DBMS の持つ様々な機能を利用している．例えば，提案システムのユーザ認証は DBMS の仕組みを利用している．

提案手法の特徴は，アクセスポリシーがデータと同じテーブルに同じスキーマで格納されていることである．またアクセスポリシーとデータが混在している SAXOPHONE から XML 文書を取り出す処理は SQL 問合せで表現されている．この SQL 問合せはアクセスポ

リシーの処理を内包しており，またユーザはテーブルに直接アクセスできず，この SQL 問合せを実行する権限のみを持つため，XML 文書を取り出す際，必ずアクセスポリシーの処理が行われる仕組みになっている．このようにデータとアクセスポリシー，および関係データベースと XML リポジトリを統合する事により，セキュリティの一元管理を行うことができると示した．

提案手法のもつアクセスポリシーは，アクセス制限とアノテーションである．アクセス制限を利用すると任意のユーザに対し XML 文書内の一部分を隠すことができる．また，アノテーションを利用するとユーザは自分の為の注釈を XML 文書に加えることができるほか，自分の子孫のラベルを持つアカウントに対して公開された注釈を付加する事ができる．

今後の課題としては，より実用に即したアクセス管理手法へ改良したいと考えている．例えば，多くの DBMS やオペレーティングシステムで採用されている，Role-based access control を提案手法と統合する事などが考えられる．本論文ではユーザとアクセスポリシーが一対一の関係であるが，Role-based access control はアクセスポリシーをロールという独立単位で扱い，アカウントはそれら一つないし複数のロールと関連付けられている．この仕組みではユーザへのロールの付与，剥奪が容易であり，柔軟なアクセス管理を行うことができる．今回提案した Hierarchy authorization を利用して Role-based access control の実装を行いたいと考えている．

また提案手法では問合せ中にソートや一時テーブルの使用などコストのかかる操作が含まれている．今後は実験などを通し，クエリやインデクスの最適化を行い，スループットの向上を図りたい．

さらに，SAXOPHONE を利用した XML 文書の版管理の研究と統合し，時間によって変化する XML 文書に対するアクセス管理について研究を行う予定である．

謝　辞

本研究の一部は，文部科学省科学研究費補助金特定領域研究(2)(課題番号:16016273)による．

文　献

[1] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura, “XRel: A Path-Based Approach to Storage and Retrieval of XML Documents using Relational Databases,” ACM Transactions on Internet Technology, Vol. 1, 1: 110–141.

[2] XACML: OASIS, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml. Accessed 24 Dec. 2004.

[3] Fernandez M.F., et al., "Publishing Relational Data as XML: The SilkRoute Approach," IEEE Data Engineering Bulletin 24(2), 2001.

[4] Carey et al., "XPERANTO: Publishing Object-Relational Data as XML," In Workshop on Web and Databases (WebDB), 2000.

[5] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati, "A finegrained access control system for XML documents," ACM Transactions on Information and System Security, 5(2), pp.169-202, May 2002.

[6] 横山昌平，太田学，片山薫，石川博，“XML パーサを考慮した応用向き関係データベース構成法,” 第 13 回データ工学ワークショップ(DEWS2002)論文集, A5-3, Mar. 2002.

[7] 横山昌平，太田学，片山薫，石川博，“XML 文書管理におけるブランチ機能を有するバージョン系列のための関係データベース構成法,” 信学論 (D1), Feb. 2004.

[8] Role Based Access Control (RBAC), http://csrc.nist.gov/rbac/