関係データベースを利用したXMLリポジトリのためのアクセス管理手法

横山昌平[†] 太田学[††]
片山薫[†] 石川博[†]

本論文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アクセス管理の実装について報告する．XMLリポジトリに関係データベースシステムを利用する利点は，関係データベースがすでに培ってきた大量のデータ処理に関する機能，たとえばインデックスやトランザクションを，XML処理に容易に適用できるからである．XMLには一番基礎となる整形式のデータ，そしてスキーマでモデルが限定される妥当なXML文書に大別される．提案手法は整形式のあらゆるXML文書を格納する目的で設計されている．提案手法が目指すことは関係データベースの持つユーザ認証機能をXMLリポジトリと統合することである．ユーザはすでに利用しているDBMSのアカウントを利用して，提案するXMLリポジトリにセキュアなアクセスをすることができる．提案手法の特徴は，アクセス権の処理がすべてSQLで記述されている点である．つまりほとんどのRDBMS上で提案システムを容易に構築できる．本論文では提案手法をMicrosoft SQL Server，MySQL，PostgreSQLという3つのDBMS上で実装し性能を調査した．本論文は我々が提案するSAXOPHONE上での実装について報告する形をとっているが，本論文が議論する問題点は関係データベースを利用したXMLリポジトリすべてに関連する問題点であり，それを解決する提案手法はそれらXMLリポジトリ上でアクセス管理全体に適用できる．

An Access Control Method for XML Repositories Using Relational Databases

Shohei Yokoyama,[†] Manabu Ohta,[††] Kaoru Katayama[†] and Hiroshi Ishikawa[†]

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 such as Microsoft SQL Server, MySQL, PostgreSQL.

1. はじめに

半構造データを記述するExtensible Markup Language（XML）[1)]はインターネットと親和性の高い言語として，企業間のデータの交換の共通形式として発展した．さらに現在ではデータの交換だけでなくデータの蓄積用途にもよく用いられている．XML文書はインターネットの発展とともに日々増え続け，様々な分野で用いられているが，XML自体はあくまでファイル形式のみを規定した規格であり，ユーザがどのようにそれにアクセスするかはアプリケーションに完全に依存する．XML文書を扱うための最も基本的なツールとしてDOMとSAXがある．これらはAPIであり，ユーザはそれらのメソッドを通してXML文書にアクセスすることができる．しかしこれらは1つのXMLファイルを処理するためのツールであり，データの管理に関する機能は持っていない．

かつて大量のデータを管理する必要性から関係デー

† 東京都立大学大学院工学研究科
Graduate School of Engineering, Tokyo Metropolitan Universityy
†† 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

タベースが登場したように，XML 文書のデータを効率良く蓄積・管理するための仕組みは非常に重要な課題である．

また，インターネットが世界中に張りめぐらされ，個々のコンピュータをつないでいる現代において，データを利用する人のアクセスは適切に処理されなければならない．そこで本論文では XML リポジトリ上でのアクセス管理手法を提案する．

XML のデータ管理の仕組みは大きく分けて 2 つの手法がある．1 つは同じく大量のデータ管理をする目的で研究されてきた関係データベースを利用して XML のデータ管理も行う手法である．また，もう 1 つはネイティブ XML データベースと呼ばれる，XML 文書のためだけに作られた独自のシステムでデータ管理を行う手法である．前者はすでにデータベースが持っている様々な機能，たとえばアクセス管理・インデクス・トランザクション・レプリケーション・バックアップ・グリッド等の機能をそのまま XML データ処理にも適用できるという利点を持っている．しかしながら，関係データベースの管理するデータは表形式であり，XML 文書は木構造であるため何らかの変換が必要である．XML 文書の木構造を処理するという点では，初めから XML の木構造を扱う目的で作られたネイティブ XML データベースの効率が良いとされる．

この 2 種類の異なる手法ともに異なる利点があるため一概にどちらか優れていると決めることはできないが，我々は現時点での普及度の点で考え関係データベースを使った手法を採用した．関係データベースはすでにほとんどの企業で利用されており，また無料のレンタルサーバ等を通して個人での利用も容易である．その点に注目し，我々は SAXOPHONE[2] という関係データベース上に構築する XML リポジトリを提案してきた．

本論文の主題はこの SAXOPHONE 上でアクセス管理を実現することである．SAXOPHONE に関しては 4 章で詳述するが，これは SAX を用いて XML 文書をシリアライズし，固定された関係データベースのテーブルへ格納する手法である．この手法の利点は XML の最も基礎的な形である整形式の XML を一元的に格納できることである．格納先の関係データベーススキーマは固定されており，関係データベースから XML データを復元する問合せも一元化できるため，どのような構造の XML 文書も，またそれにどのような更新がかかってもデータベーススキーマの更新は必要なく，また格納した XML 文書を呼び出す問合せも変化しない．

次にアクセス管理を考える．大量のデータを管理し，また大勢のユーザにそれを公開する場合，ユーザごとに閲覧可能なデータを規定することはセキュリティの観点から重要である．たとえば，商店の在庫管理を考える．オーナは原価等の情報を在庫データと一緒に管理したいと思うかもしれない．しかし，同じ在庫データを使ってオンラインショッピングを行っている場合，その顧客からは原価情報を隠したいと考えるだろう．あるいは，そのオーナが同時に実際の店舗販売も行っていたとしたら，実店舗では販売するがオンラインでは販売しない商品があるかもしれない．この場合，データに対するそれぞれのユーザからのアクセスは適切に処理されなければならない．

従来の関係データベースで管理されるデータ場合，このようなアクセス管理は適切な条件式（アクセスポリシ）を含む view で解決される．オーナ，実店舗の顧客，オンラインの顧客とユーザごとにそのユーザしかアクセスできない view を用意し，ユーザはその view に対して問合せを行う．関係データベースのようにあらかじめスキーマもデータ型も決定されており，表という単純な構造をしているデータを扱う場合は view に規定される条件も単純な条件で済むかもしれない．しかしながら整形式の XML はどのようなデータ構造もユーザが自由に作り出すことができる．また整形式の XML 文書ではスキーマは明示されないため，スキーマを大幅に変えるような変更も簡単に行うことができる．もし関係データベースでいう view のような条件式で整形式 XML 文書をアクセス管理する場合，スキーマの変更のたびに条件式を見直して，必要ならば更新しなければならない．これは非常に困難である．そこで我々は，条件としてアクセスポリシを規定するのではなく，データ単位でアクセスポリシを規定する方法を提案する．データ単位とは，たとえば上記例でいうなら原価情報のデータそのものに「オーナのみアクセス可能」というフラグ（アクセスポリシ）を付ける方式であり，スキーマの変更も含むデータの変更においてもそのデータが失われない限りアクセスポリシは見直しや変更の必要がない．本論文では整形式 XML 文書のアクセス管理においてデータ単位でアクセス管理を行う手法を提案する．

提案する XML リポジトリは関係データベース上に構築することは前述した．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おり，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．提案システムのユーザ認証は関係データベースのアカウントおよびユーザ

認証と統合されている．すなわちユーザは現在利用している関係データベースのアカウントで提案システムを利用できるということである．このことはセキュリティの一元化の観点からも，ユーザの利便性の観点からも利点であるといえる．

本論文は実際に実装した結果の報告を主眼としており，極力擬似コードや汎用的なアクセスポリシの記述を避けている．XML における汎用的なアクセスポリシの記述については最近議論の多い点[3)~5)]ではあるが，提案手法は関係データベースの問合せ・アクセス管理機能を用いて，XML リポジトリのストレージレベルでのアクセス管理の実装を目的としており，XML に対するアクセスポリシの定義を実際のストレージへ射影する議論とは粒度が異なる．本論文の主眼は実際に SAXOHONE が利用する DBMS ストレージへ格納するための手法である．しかしながら，提案手法が解決を試みる問題点は関係データベースを用いた XML リポジトリにおいては共通の問題点であり，その意味では実装の一例を報告するだけでなく，関係データベース上で XML 文書を管理する際の一般的な問題点に関する議論とその解決手法の提案と位置づけることができる．

本論文の構成は以下のとおりである．続く 2 章では提案手法で解決すべき問題点と関連研究について述べる．3 章では研究の背景および関係データベースを利用した XML リポジトリにおける一般的な問題点について議論する．4 章ではアクセス管理を実装に利用する SAXOPHONE と Prefix Labeling Scheme を利用したアカウント識別子について述べる．5 章では関係データベースの機能とアカウント識別子を用いた Hierachy Authorization の実装について記し，続く 6 章ではユーザ認証プロセスを説明する．7 章では実装により得られた知見を実験結果を示しつつ述べ，最後に 8 章で本論文をまとめる．

2. 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．提案手法におけるアクセス管理とは XML 文書インスタンス中の任意のノード（要素・属性・文字列）を単位として，それらに対し任意のユーザからのアクセスを拒否することによってアクセス権をコントロールする仕組みである．本論文ではあるアカウントにおける「アクセス否定」ルールの集合をアクセス権と呼ぶ．ユーザはアクセスが拒否されたノード以外の部分で構成された XML 文書を SAX APIs を通し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．多くの関係データベース，たとえば商用の Microsoft SQL Server や無料で利用可能な PostgreSQL 等は，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等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ことができる．

我々はデータの一元化だけでなく，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．

提案手法ではアカウントのアクセス権限が階層構造になっている．ユーザアカウントはツリーとして表すことができ，親ユーザのアクセス権を子ユーザが継承する．我々はこれを Hierarchy Authorization と呼んでいる．アクセス権限が階層構造であるため，アカウントを識別する ID（アカウント識別子）として，木に対するラベル付け手法の 1 つである Prefix Labeling Scheme を利用している．なお，本論文では説明の簡便さのためにアクセス権限のツリーに直接ユーザのアカウントが関連付けられているが，実際にはロールが関連付けられた Role Based Access Control 手法[6)] を念頭に開発されている．

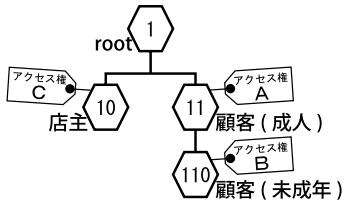

図 1 アカウント識別子とアクセス権
Fig. 1 Account IDs and right of access.

表 1 アクセス権
Table 1 Right of access.

アカウント	Root	オーナ	顧客（成人）	顧客（未成年）
アクセス権		C	A	A
				B

図 1 はインターネットショップの商品データベースを想定したアカウント ID とアクセス権の例を示している．それぞれのアカウントにはアクセス権が設定されている．アクセス権はアカウント識別子によって階層化されている．子のノードは親のノードのアクセス権を継承する．すなわちアカウントとそれに適用される実際のアクセス権の対応は以下のようになる．

この例では顧客（未成年）は顧客（成人）のアクセス権 A を継承し，未成年に付与されたアクセス権 B と同時に適用される（**表 1**）．たとえば「顧客（成人）は商品のコストを見ることができない」および「顧客（未成年）はタバコ・アルコール商品を見ることができない」と設定されていた場合，顧客（未成年）はその両方のルールに従う．つまりあるアカウントに適用されるアクセス権は，そのアカウントの先祖すべてのアクセス権を集約することによって得られる．

2.2 関 連 研 究

データの蓄積に XML を用いるための研究は様々な粒度で様々な議論がなされている．たとえば Apache プロジェクトの Xindice[7] はネイティブ XML データベースに分類される XML 専用のデータベースである．ネイティブ XML データベースは現状では XML 特有の柔軟なデータ構造に対応しているものの，関係データベースと比べ，スケーラビリティに欠けるといわれ，まだ普及も進んでいない．我々はすでに関係データベースが実装している機能をそのまま適用する目的，および関係データベースの普及度を考え，独自の実装ではなく，関係データベース上に XML リポジトリを構築する手法を選んだ．提案手法の基礎となる SAXOPHONE[2] は我々が開発した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．たとえば xRel[8] 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．

SilkRoute[9] や XPERANTO[10] 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ことはできない．

XML のアクセス管理に関する研究も議論が多い[3),4)]．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[11] があげられる．この研究では XML 文書におけるアクセスポリシ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，スキーマの知識によらない様々なスキーマの 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．

XACML[5] 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ことを目的として作られたアクセスポリシ記述言語である．これらの手法は基本的には XPath 等で指定された要素に対してアクセスの可否を記述する方式である．しかしながら XPath 等で表される条件文は DTD 等によってスキーマが記述されていない場合，さらにはユーザの知識としてもスキーマが規定されていない場合，記述が困難である．特定のスキーマに限定されない整形式の XML 文書では，毎回スキーマの変更をともなうような更新の可能性も考慮しなければならない．XPath 等の条件文はスキーマが変更されるたびに，変更後のスキーマに合わせて条件を書き換えなければならない．この方法では更新のコストが非常に大きくなってしまう．提案手法では XML 文書を SAX のイベント列に分解し，そのイベントごとにアクセス可否を設定する方法をとっている．この方法では，XPath 等を用いた条件文は必要とされずデータに直接アクセス権が付与されているため，スキーマの大幅な変更をともなうような更新の際もアクセスポリシの見直しは必要とならない．さらに，非常に複雑な XPath で示されるデータをアクセス拒否したい場合も，実際にそのデータに対してアクセスの可否を設定すればよいので，シンプルに実現可能である．

3. 研究の背景

本章では提案手法の背景となった問題点とそれに関する議論について述べる．

3.1 アクセス権同期の問題

関係データベースを利用したXMLリポジトリの提案が多数なされていることは前述した．この手法の利点は大規模データ処理に関する関係データベースの機能をXML処理に適用できるという点である．関係データベースが得意とするアクセス管理をXMLリポジトリ上で利用するための手法が本提案である．このようなシステムではアクセスポリシの定義やその記述手法でだけでは解決のできない根本的な問題をかかえている．ここで注意すべきことは，関係データベースのアクセス管理機能をXMLリポジトリへ適用させると同時に，格納するXML文書が必要とするアクセス管理を関係データベースのアクセス管理へ同期させなければならないという点である．

たとえば一般的なサーバクライアントシステムを考える．関係データベースのテーブル T に格納したXML文書 X があるとする．X は部分文書 a と b から構成される．ユーザ A は b へのアクセスが拒否され，ユーザ B は a へのアクセスが拒否されるというアクセスポリシが設定されていたとする．ここでは原始的な形として，アプリケーションがポリシを解釈し，ODBC等で接続したDBMSに，アクセスが許可された部分文書のみを取得するクエリを発するシステムを仮定する．ユーザ A，B はそれぞれ関係データベースのアカウントを持っており，それを使ってDBMSに接続する．クライアントアプリケーションはポリシを解釈してユーザごとに異なる問合せを発する．このようなシステムではユーザインタフェースがXQueryであれDOM，SAXであれ最終的にはSQLに変換されDBMS上で処理が行われる．議論の見通しのために非常に簡単な例を記す．たとえばユーザ A がXMLリポジトリから文書を取り出す問合せは次のように表されるシステムがあったとする．

```
01: SELECT   *
02: FROM     T
03: WHERE    部分 = 'a';
```

この問合せを実行するには，DBMS上で各ユーザに対してテーブル T に対するSELECT権限を付与しなければならない．このことは次の問合せを許容することを意味する．

```
01: SELECT   *
02: FROM     T
```

後者の問合せによってユーザ A はXML文書のアクセスポリシにかかわらず，XMLリポジトリに格納されたすべてのXML文書全体を閲覧することができてしまう．我々の知る限りでは関係データベース上に構築するXMLリポジトリの研究において，この問題に言及した議論はない．特にXML文書の構成要素が，固定された関係データベースのスキーマに格納される手法においては，テーブルのタプルごとにアクセス許可・不許可を設定する必要があるが，現在のほとんどのDBMSにおいてはタプル単位でアクセスを管理することは困難である．

3.2 Viewの問題

前節で提示した問題点を避けるために，関係データベースを用いたデータ処理では，適切にアクセス権が設定されたViewを用いるのが一般的である．つまりユーザごとにそのユーザがアクセス可能なデータのみを取り出すことのできるViewを設定し，そのViewに対して問合せを行うという仕組みである．

多くのXML文書のアクセス管理の研究では，関係データベースのViewの代わりにXPath等で条件を指定し，それをアクセスポリシとしている．XPathで条件を指定するということは，ユーザがスキーマを規定している必要がある．この場合のスキーマとはDTD等の実体だけでなく，ユーザの頭の中にのみ存在するスキーマに関する知識も指す．しかし整形式のXML文書を扱う場合，スキーマがまったく規定できないようなデータの存在可能性も考慮しなければならない．たとえば所有する本のリストは次のようなXML文書で表されるかもしれない．

```
01: <Book-Shelf>
02:     <誰がために鐘は鳴る/>
03:     <カラマーゾフの兄弟/>
04: </Book-Shelf>
```

このようなXML文書は本が増えるたびにスキーマが変更される．スキーマが変更されるということは，スキーマに基づいているXPathによる条件文も見直さなければならない．その条件文がユーザごとに存在するとなるとXML文書更新の際にかかるコストは膨大なものとなる．

固定されたスキーマを作ることができないシステムの例としては我々が行っている創発的XMLの研究[12)]があげられる．このフレームワークは，分散した複数のXML文書が内包するキーに基づいて自発的に結合する，いわばアクティブデータベースの概念を持った提案であり，結果として得られるXML文書の形はあらかじめ予想がつかない．つまりXPath等を用いて

条件式としてアクセス権を記述することは不可能である．このようなシステムでアクセス管理を考える場合，提案手法のようなデータ単位でアクセスを制御する手法が現実的な実装手法であると考えられる．

3.3 提案手法

これらの問題を解決する手法を開発し，我々の提案している XML リポジトリ上での実装を報告することが本論文の目的であるが，それに先立ちどのようなアプローチで解決するかをここで考える．

まずアクセス管理の仕組を示すために，一番原始的なモデルを考える．関係データベースへのアクセスである以上，ユーザはそれぞれ DBMS のアカウントを持っていることは前提である．また，すでに多くのユーザが DBMS を身近に利用しているものと考えられる．もし DBMS のアカウントを持っているならそれを利用することにより，関係データベースと XML リポジトリのアカウントを一元化することができる．XML リポジトリには一般的な関係データベースへの接続と同じようにアカウントとパスワードによって接続する．また XML リポジトリから取り出したい XML 文書を URI で指定する．アクセス管理のための手続きを P とするなら，アカウント $account$ によりアクセス管理された XML 文書 $X_{account}$ を得る処理は次の式で表される．

$$X_{account} = P(account, password, uri-of-X)$$

さらにアクセス権同期の問題を解決するには，手続き P はユーザにとって改変不可能でかつ DBMS 内に存在する必要がある．また，アクセス権の適用はどのアカウントからのアクセスにせよ P 内で動的に解決されなければならない．これを実現させる仕組みとして多くの DBMS が持つストアドプロシージャと呼ばれるサーバサイドの実行環境を利用する．ストアドプロシージャは中に SQL 問合せ手続きを含み，ユーザは要求された引数を与えることによりこれを起動することができる．ユーザはこのストアドプロシージャの実行権限のみを持つことにより，ストアドプロシージャ内に記述された問合せや，XML 文書が実際に格納されているテーブルはユーザが改変できないブラックボックスとすることができる．

またこのストアドプロシージャはユーザ数にかかわらず 1 つであるので，ユーザごとに View が乱立するといった問題も避けることができる．

この仕組みによりユーザは DBMS によりアクセス禁止されているテーブルへストアドプロシージャ経由でのみアクセスが可能となる．このストアドプロシージャが単一の結果セットを返すなら，ユーザからは通常の SELECT 文を用いた問合せと同じように見える．提案手法では各ユーザは XML が実際格納されているテーブルへのいっさいの権限を持たずに，アクセス管理を行うストアドプロシージャの実行権限のみを持っている．この仕組みにより，アクセス管理プロセスを通さずに XML 文書を取得することができない仕組みを提供している．そしてストアドプロシージャを起動するには DBMS 上での適切なアカウントが必要であり，またストアドプロシージャ内でさらにアカウントのチェックを行うため，ユーザはクライアントの種類によらず，格納された XML 文書を不正に得ることはできない．

この仕組みで問題となるのは，アクセス管理の手続きを SQL を用いて記述しなければならないということである．このレベルでは XACML に代表される高度なアクセスポリシ記述言語を直接解釈し適用することは非常に困難である．そこで提案手法では XML 文書の各構成要素に対して，各ユーザがアクセス可能か，不可能かという 2 つの状態でアクセス権の管理を行っている．さらに SQL で処理ができるように，これらアクセス権の定義は XML が格納されているテーブルと同じ場所にリレーショナルデータとして保存している．とはいえ本研究は高度なアクセスポリシ定義によるアクセス管理を否定するものではない．提案手法はストレージレベルでのアクセス権の付与を目的としており，高度なアクセスポリシ定義は適切にストレージレベルのアクセス権へ射影することができると考えている．しかしながら，それは XML 木と関係データベースに格納された XML 文書とのマッピングの議論であり，本論文の目的を超えてしまうため，ここでは述べない．

4. SAXOPHONE とアカウント識別子

提案手法であるアクセス管理の説明に入る前に，その前提となる XML リポジトリ SAXOPHONE とアカウント識別子と呼ぶユーザとそのユーザが持つアクセス権を保持する ID の仕様について簡単に説明を行う．SAXOPHONE は我々が提案してきた SAX イベントに基づいて DMBS 上に構成される XML リポジトリである．またアカウント識別子は，提案システムのユーザに付与される ID である．本手法ではアクセス権が階層的に構成されるため，木へのラベリング手法に基づいた識別子を利用している．

4.1 SAXOPHONE：XML リポジトリ

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

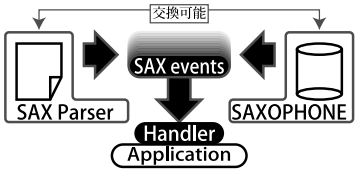

図 2 SAXOPHONE の仕組み
Fig. 2 SAXOPHONE framework.

event

rID	eID	type	property
1	1	1	kiosk
1	2	1	cigarettes
1	3	2	name="menthol"
1	4	1	cost
1	5	3	200
1	6	4	cost
1	7	1	price
…	…	…	…

resource

rID	URI
1	http://hostname/exaple.xml
2	http://hostname/test.xml
…	…

図 3 SAXOPHONE に格納されたデータ例
Fig. 3 An example of data in the SAXOPHONE.

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 アプリケーションはそのまま利用可能である．

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 2** 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ことができる．

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)

また実際に格納された XML 文書の例を**図 3** に示す．このデータは次の XML を意味している．

```
01: <kiosk>
02:   <cigarettes name="menthol">
03:     <cost>200</cont>
04:     <price>...
```

SAX は XML ファイルの先頭から順に読み込んで，XML の構成要素を発見するたびに利用者側にイベントを送る．つまり SAX を使えば XML ファイルは木ではなくイベントのストリームとなる．そこで eID フィールドにイベントの発生順の番号を格納し，そのイベントのタイプを int 型として type フィールドに格納，イベントの中身，つまりタグ名や属性および文字列を property フィールドに保存することにより，1 つの固定されたテーブルにすべての XML 文書を格納することができる．XML 文書を取り出すときには eID を昇順でソートすることにより，元の SAX イベントストリームを再構成することができる．SAX は DOM と並んで，XML 文書を扱うための最も基本的なツールであるが，提案手法はこの SAX との親和性が非常に高い．

SAX イベントを再構成は SQL 問合せにより高速に実行される．

```
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るため，SAXOPHONE は様々な DBMS 上で動作させることができる．さらにこの SAX イベントストリーム再構成のための SQL 問合せは提案システムによって自動的に行われ，ユーザはあくまで SAX パーサを用いて XML 文書にアクセスする．たとえば 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: SAXParserFactory spf
02:     = SAXParserFactory.newInstance();
03: SAXParser sp = spf.newSAXParser();
04: sp.parse(URI, new mySaxEventHander());
```

Java では SAX パーサのインスタンス呼び出しにファクトリーメソッドを使っている．このため既存の SAX アプリケーションはいっさいのコードの変更なしに，提案手法への乗り換えが可能となっている．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 と同じインタフェースを持つため，新たな問合せ言語の習得等を必要としない．

以上が我々がすでに提案している SAXOPHONE の仕組みである．本論文ではこの SAXOPHONE 上でのアクセス管理の実装の報告である．提案手法は SAXOPHONE の特徴である SAX との親和性の高さを維持し，かつ柔軟なアクセス管理の実装を目的としている．

4.2 アカウント識別子

本節ではユーザを識別するために提案システムが利用している識別子について説明する．前述したようにアカウントのアクセス権の構造は木である．よってアカウントの識別子には木へのラベリング手法を用いている．木へのラベリングは多くの提案があり，最近では XML 木へのラベリング手法の議論が非常に活発である[13)~15)]．提案手法が必要とするラベリング手法の要求を次に記す．

1. 一般的な DBMS のデータ型に適用
2. SQL 問合せで先祖検索が可能
3. 子孫・兄弟方向ともにラベルが枯渇しない

これらの条件から，我々は Prefix Labeling を採用した．Prefix Labeling はあるノード A の先祖のノードはすべて A の接頭辞になるラベルを持つというルールを持つ．我々は以前，SAXOPHONE 上で版管理を行う手法を提案した際この Prefix Labeling を利用した識別子を提案している[16)]．アカウント識別子はその研究と同じ手法を用いているが，用語等が多少異なるので，本節で簡単に説明する．

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 1 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．

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

たとえば 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．

$$B_8 = 10^1(100+1) + 8 - 10 = 1008 \quad (2)$$
$$B_9 = 10^2(100+1) + 0 - 10 = 10090 \quad (3)$$

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．つまり識別子 1008 の同じレベルの弟ノードは 1009 ではなく 10090 になり，その次に 10091 へと進む（**図 4**）．この仕組みにより木の幅方向ラベルの枯渇を防いでいる．

あるアカウントにおけるアクセス権は祖先すべてのアクセス権を集約する必要があることはすでに述べた．その問合せは SQL で次のように表現される．

```
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．たとえば postgreSQL には position（Substring in String）関数が存在し，MySQL では LOCATE（Substring，String）関数が存在する．

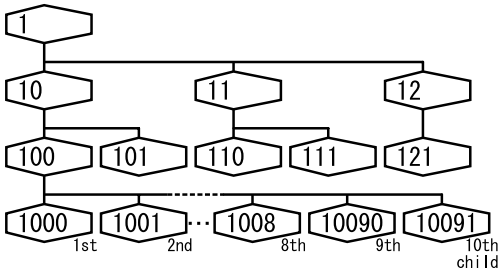

図 4 アカウント識別子の例
Fig. 4 An example of our account identifiers.

上記問合せは 110 の接頭辞となる aID を持つ値すべてを呼び出している．WHERE 句が TRUE となるのは，テーブル中で aID フィールドが 1 か 11 か 110 の項目である．つまり図 1 の例で考えるならこの問合せにより root，顧客（成人），顧客（未成年）すべてのアクセス権を得ることができる．

5. アクセス管理手法

3 章で本研究で議論する問題とその解決へのアプローチについて述べた．また 4 章では我々が研究してきた XML リポジトリ SAXOPHONE や識別子の構成法を紹介した．本章では提案するアクセス管理手法を行う我々の実装系について詳述する．

5.1 関係データベース構成法

本実装の関係データベーススキーマを**図 5** に示す．続いて event テーブルの各フィールドについて説明する．rID は格納する XML 文書の識別子である．SAXOPHONE は複数の XML 文書を同一のテーブルに格納する．そのため文書ごとに異なる値を割り当てている．rID は resource テーブルで URI と関連付けられている．eID は XML 文書を SAX で読み込んだ際，イベントが発生する順序に 1 から昇順で与えた番号である．文章内のイベントの位置を表し，このフィールドを昇順にソートすることにより，イベントの順序を再構成することができる．aID にはそのイベントが属するアカウントのアカウント識別子が格納される．type フィールドはイベントの種類を表す．提案手法が扱う 5 つのイベントとそれに対応する type 値，property 値の意味を**表 2** に示す．

table: **event**

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

CONSTRAINT:
PRIMARY KEY (**rID**,**eID**,**aID**)
FOREIGN KEY **rID** REFERENCES **resource.rID**

table: **resource**

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

CONSTRAINT:
PRIMARY KEY **rID**
UNIQUE **URI**

table: **account**

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

CONSTRAINT:
PRIMARY KEY **rID**
UNIQUE **URI**

図 5 関係データベーススキーマ
Fig. 5 The RDB schema.

表 2 イベントの種類
Table 2 Event types.

イベントの種類	type	property
アクセス拒否	0	NULL
エレメント開始	1	タグ名
アトリビュート出現	2	属性名=”属性値”
文字列データ	3	文字列
エレメント終了	4	タグ名

$type = 1$，2，3，4 は実際に SAX パーサが扱うイベントである．$type = 0$ は本研究で独自に拡張したイベントで，その eID を持つイベントがアクセス拒否されることを表す．ここでは XML 文書中の 1 つのイベントに注目して，実際にこれら SAX のイベントとアクセス拒否のイベントがどのように格納されるかについて示す．$rID = 1$ かつ $eID = 4$ である ⟨cost⟩ という開始タグがあったとする．図 1 で示したアカウントからアクセスを想定する．商品原価はオーナは知りたいが，顧客からのアクセスは拒否したい．この場合 event テーブルへは**表 3** のように格納される．

このデータは，11 が接頭辞となる aID を持つ利用者は『アクセス拒否』され，それ以外のケースで 1 が利用者の aID の接頭辞になる場合は『エレメント開始イベント』が発生する，ということを意味している．よって顧客（成人），顧客（未成年）は $eID = 4$ の ⟨cost⟩ 要素を参照することはできないが，オーナは見ることができる．

表 3 ストレージ構成例
Table 3 An example of data in the storage.

rID	eID	aID	type	property
1	4	1	1	cost
1	4	11	0	NULL

表 4 Descendant local annotation 例
Table 4 An example of descendant local annotation.

rID	eID	aID	type	property
1	3.1	11	2	smell=”cool”

つまりあるアカウントにおいて有効なイベントとは，そのアカウント自身か，さもなければ直近の祖先のイベントである．アカウント識別子 = 11, 110 にとって直近の祖先は $aID = 11$ のアクセス拒否イベントであり，アカウント識別子 = 10 にとっては $aID = 1$ のエレメント開始イベントが有効なイベントとなる．

SAXOPHONE に格納するオリジナル XML 文書のイベントにはすべて 1 というアカウント識別子が付加されているが，SAX イベントすなわち $type = 1$，2，3，4 のイベントは必ずしも $aID = 1$ である必要はない．たとえば **表 4** には $aID = 11$ を持つアトリビュート出現イベントが存在する．

これはすなわち 11 が接頭辞となるアカウント識別子を持つユーザがアクセス可能なイベントである．よって顧客（成人）と顧客（未成年）はこのイベントを取得できるが，root およびオーナは取得できない．イベント ID はそのイベントを挿入したい前後のイベント ID を参照して間の値をつける．イベント ID は XML 文書中の絶対的な位置を示すため，他のイベントや注釈と一致してはならない．この時点ではオリジナル文書のイベント ID はすべて整数値なので，必然的に少数値を有するイベント ID となる．このようなオリジナルの XML 文書に root 以外のユーザが独自のイベントを挿入することを我々は注釈（annotation）と呼び，表 4 にあるような，注釈をしたユーザとツリー上でその子孫にあたるアカウントが閲覧できる注釈を Descendant local annotation と呼んでいる．挿入するイベントの ID として少数値を利用した理由としては，イベント ID の更新や再計算の必要なしに新しいイベントを任意の場所に挿入することができるためで

表 5 User local annotation 例
Table 5 An example of user local annotation.

rID	eID	aID	type	property
1	3.1	11	2	smell=”cool”
1	3.1	110	0	NULL

ある．この方法ではイベント ID の偏りが発生するため，システムの利用が少ない時間帯，たとえば夜間等に整数値のイベント ID を振りなおす作業を行う．

提案手法ではさらに，アクセス拒否イベントを併用することにより注釈した本人のみが取得可能なイベントも作成することができる（**表 5**）．

この表の例では，アカウント識別子 11 を持つ顧客（成人）は注釈を取得できるが，その子孫の顧客（未成年）はアカウント拒否イベントを受け取る．また root やオーナアカウントは 11 および 110 の接頭辞ではないためこの注釈は取得しない．このように注釈した本人のみ取得可能なイベントを User local annotation と呼ぶ．

注釈はあくまで利用方法に基づいた名前であり，処理方法はアクセス管理手法と同じである．では次にアクセス管理された SAX イベント列を生成する実装系を具体的に説明する．

5.2 アクセス権処理と SAX イベント列の生成

実装系の具体像を描くために，以下の単純な XML 文書を定義し，どのようにリポジトリに格納され，それが呼び出されるかについて実際の SQL コードを交えて説明する．提案手法は多くの DBMS 上で動作することを目的としているが，DBMS の機能には多少の差があり，また SQL 文も厳密には異なることもあるため，説明の一貫性に留意して本論文では Microsoft SQL Server の用語を用いる．

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

これを図 1 に表されたアカウントからのアクセスを適切に処理することを考える．オーナ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．このアクセス権を含んだ XML 文書を提案手法で表すと格納されるデータは**表 6** になる．

さらに顧客（成人）が，ジュースタグに Descendant local annotation（eID=4.1），コーラタグに User local annotation（eID=6.1）を行ったとする．注釈後の格納データは**表 7** になる．

表 6 関係データベースに格納された XML 文書
Table 6 An XML document in RDBs.

rID	eID	aID	type	property
1	1	1	1	LIST
1	2	1	1	お取り置き
1	2	11	2	
1	3	1	3	お取り置き
1	3	11	2	
1	4	1	1	ジュース
1	5	1	3	ジュース
1	6	1	1	コーラ
1	7	1	3	コーラ
1	8	1	1	ビール
1	8	110	0	
1	9	1	3	ビール
1	9	110	0	
1	10	1	1	LIST

表 7 注釈された XML 文書
Table 7 An annotated XML document.

rID	eID	aID	type	property
1	1	1	1	LIST
1	2	1	1	お取り置き
1	2	11	2	
1	3	1	3	お取り置き
1	3	11	2	
1	4	1	1	ジュース
1	4.1	11	2	味=”オレンジ”
1	5	1	3	ジュース
1	6	1	1	コーラ
1	6.1	11	2	味=”ダイエット”
1	6.1	110	0	
1	7	1	3	コーラ
1	8	1	1	ビール
1	8	110	0	
1	9	1	3	ビール
1	9	110	0	
1	10	1	1	LIST

以上が XML 文書から関係データベースにアクセス権を付加して格納する手法である．次に，格納されたデータに対し，ユーザが持つアクセス権に従った適切な XML 文書を得るための問合せを説明する．

ユーザは提案システムにアカウント名・パスワードおよび URL を指定してアクセスを行う．関係データベースおよび提案システムによってユーザ認証が行われる．このプロセスは 6 章で詳述する．ユーザ認証の結果ユーザのアカウント識別子 $@aID$ が得られる．また URI からは，リソースの識別子 $@rID$ が得られる．この $@aID$ と $@rID$ を用いて event テーブルに問い合わせることによりアクセス管理を行う．取り出すデータはアクセスが許可されているイベントのみで成り立つ SAX イベントの列であり，元々のイベント

の順序が再現されていなければならない．そのようなデータを取り出すための用件を次にあげる．

1. type と property をユーザに返す
2. event テーブルを走査する
3. rID フィールドは@rID である
4. aID は@aID の接頭辞である
5. eID ごとにグループを作る
6. 個々のグループの中で NULL イベントが存在する場合はそのグループ全体を結果から除外する
7. eID の昇順でソートする

ここで着目したいのは，eID の値は再利用されないということである．SAX イベントと同じ eID にはそのイベントへのアクセス拒否を意味する NULL イベント以外は設定できない．またある aID でアクセス拒否されたイベントを下位のアカウントで再び有効にすることはできない．

つまり，アクセスするユーザの持つアカウント識別子 $@aID$ の接頭辞となるイベントを取り出したとき，個々の eID においては type=1，2，3，4 を持つイベントが 1 行だけヒットするか，もしくはそれに type=0 の NULL イベントが加わった 2 行ヒットするか，その 2 つの場合しかない．つまり event テーブルを eID ごとにグループ化したとき，各グループに NULL イベントが 1 つでもあった場合，そのグループすなわちイベントは $@aID$ においてアクセスが拒否されたと見なすことができる（6 行目）．

この用件を満たす問合せは次の単純な SQL 文で表現することができる．SQL 文の各行の役目は上記用件の対応する行に書かれている．

```
01. SELECT   MAX(type),MAX(property)
02. FROM     event
03. WHERE    rID = @rID
04. AND      1 = CHARINDEX(aID,@aID)
05. GROUP BY eID
06. HAVING   0 <> MIN(type)
07. ORDER BY eID;
```

3 行目および 4 行目の条件で，そのユーザが取り出すべきイベントを指定している．また 6 行目の HAVING 句で，アクセス拒否されたイベントを除外している．アクセス拒否は NULL イベントで表されるため type の値として 0 を持つ．すなわち他のすべての SAX イベント（type=1，2，3，4）よりも小さい値を type フィールドに持っている．よって eID でグループ化された集合の中で type の値が一番小さいものが 0 でないグループのみを取り出せば，アクセスが拒否されていないイベントのみを取り出すことができる．

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 6 アクセス管理された XML 文書
Fig. 6 Access controled XML documents.

SELECT 句でフィールドに MAX 関数が指定されているのは，問合せがグループ化されている場合，結果には一意となる値を指定する必要があるためである．結果を取り出す時点では HAVING 句がすでに処理されているため，各 eID においては SAX イベント（type=1，2，3，4）が 1 行ずつ存在する状態である．この問合せでは MAX 関数を利用することによって，その値を取り出している．各イベントはそれぞれ 1 行の結果のみを持つため，最大値を得る集約関数を設定することで，その 1 行を取り出すことができる．このほか，最小値を取り出す MIN 関数を使用しても同じことができる．

この問合せを実行することにより，各ユーザは設定されたアクセス権に従った，それぞれ異なる XML 文書を得る（**図 6**）．

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のみを利用して，アカウントに応じた XML データを提供する仕組みを実現した．

本論文で提案するシステムでは上記の問合せを利用しているが，同じデータを得るための問合せは複数考えられる．ここでは我々が提案手法と同時に研究を進めている版管理機能を持った SAXOPHONE[16] 向けに開発したアルゴリズムを紹介する．版管理とアクセス管理の違いは eID でグループ化した際の各グループの大きさである．

アクセス管理では前述したように，1 つの SAX イベントか，それに NULL イベント加わった 2 行のデータしか存在しえない．しかし版管理では，同じイベントが何回も書き換えられることがある．つまりあるユーザ $@aID$ が取り出すデータはそれぞれの eID におい

て，@aID の接頭辞になる値を持つ複数のイベントのうち @aID 直近のイベントのみを取り出さなければならない．そのため問合せは多少複雑になる．ここでは自己結合を用いた問合せを説明する．

```
01: SELECT    a.property, a.type
02: FROM      (SELECT *
03:             FROM event
04:             WHERE rID = @rID
05:             AND 1 = CHARINDEX(aID,@aID))
06: AS        a
07: LEFT JOIN (SELECT *
08:             FROM event
09:             WHERE rID = @rID
10:             AND 1 = CHARINDEX(aID,@aID))
11: AS        b
12: ON        a.eID=b.eID
13: AND       a.aID < b.aID
14: WHERE     b.aID IS NULL
15: AND       a.type <> 0
16: ORDER BY  a.eID;
```

この自己結合は左外部結合でありかつ非等価結合である．まずは JOIN に入力するためにユーザの持つ @aID を接頭辞とするイベントをすべて取り出す．その結果を自己結合する．結合条件は 12 行目および 13 行目である．13 行目の条件により，エイリアス a の各行について，その行が持つ aID の値よりも大きい aID を持つエイリアス b の行が結合される．外部結合であるため，もしエイリアス a のある行が持つ aID よりも大きな値を持つ行がエイリアス b に存在しない場合．結果としてエイリアス b のすべてのフィールドに NULL が設定された行と結合される．すなわちエイリアス b の NULL が許されていないフィールドいずれかが NULL を示している場合，そのエイリアス a に格納されたイベントは @aID の自分自身も含めた直近の先祖で定義されたイベントと見なすことができる．この問合せによってユーザはアクセス管理が適切に行われて SAX イベント列を得ることができる．7 章でこれら 2 つの方式の性能を評価する．

5.3 更　　新

ここでは前節で示した商品の在庫情報に関する XML 文書を更新する例を考える．たとえばリストの末尾に ⟨ ワイン/⟩ を加えるとする．ワインはアルコール飲料なので子供からのアクセスを禁止する必要がある．そこでポリシを付加した SAX イベント列を作成する（**表 8**）．

eID を決定するために挿入箇所の前後を見る．挿入箇所の前は ⟨/ビール ⟩ であり eID は 9，後は ⟨/LIST⟩ で eID は 10．差分の XML データはこの 2 つのイベントの間に挿入されなければならない．そこで，この更新データには 9 と 10 の間の値を振り event テーブルに挿入する．更新された event テーブルを**表 9** に示す．

表 8 差分データ
Table 8 An annotated XML document.

rID	eID	aID	type	property
1	?	1	1	ワイン
1	?	110	0	
1	?	1	3	ワイン
1	?	110	0	

表 9 更新された XML 文書
Table 9 An annotated XML document.

rID	eID	aID	type	property
1	1	1	1	LIST
1	2	1	1	お取り置き
1	2	11	2	
1	3	1	3	お取り置き
1	3	11	2	
1	4	1	1	ジュース
1	4.1	11	2	味=”オレンジ”
1	5	1	3	ジュース
1	6	1	1	コーラ
1	6.1	11	2	味=”ダイエット”
1	6.1	110	0	
1	7	1	3	コーラ
1	8	1	1	ビール
1	8	110	0	
1	9	1	3	ビール
1	9	110	0	
1	9.1	1	1	ワイン
1	9.1	110	0	
1	9.2	1	3	ワイン
1	9.2	110	0	
1	10	1	1	LIST

このように変更点のみの差分データをリポジトリ中の XML 文書の適切な箇所に挿入することでデータを更新する．更新は差分データに基づいて行われる．変更箇所の特定は手動で指定するほか，Longest Common Subsequence（LCS）[17] アルゴリズムでの自動化も考えられる．LCS はテキストファイルの差分抽出によく用いられる diff が利用しているアルゴリズムで，2 つの列の差分を抽出することができる．XML は木であるが提案手法ではそれを SAX イベントの列として扱っているため，LCS による差分抽出が可能となる．

更新やアノテーションでデータの挿入が起きるたびに少数値を含む eID を割り振ると，eID に偏りが生じる．この偏りはシステムの利用頻度が低い時間帯，たとえば夜間に，イベント順に 1 から始まる整数のみの

eID を振りなおすことによって解消することができる．

個々のユーザによる XML 文書の更新は，読み込みに特化した SAX インタフェースを持つ提案システムでは，現在のところ考慮していない．

6. ユーザ認証プロセス

前章では DB サーバ内部で実行されるアクセス権処理手法を述べた．本章ではその処理プロセスを起動するためのユーザ認証プロセスを説明し，システム全体像を描く．

ユーザから見て提案手法は SAX パーサとして利用できる．SAXOPHONE を利用するための SAX パーサは提案システムが提供する．提案システムの SAX パーサはユーザのアカウント名，パスワード，取り出したい XML 文書の URI をサーバに渡し，アクセス権が適切に処理された XML 文書を受け取る．

前章で述べた処理はすべて DBMS 内にストアドプロシージャとして格納されている．つまりクライアントプログラムから提案手法はアカウント名，パスワード，URI を引数にとる関数のように見える．

$$X_{account} = P(account, password, uri)$$

この DBMS 内のストアドプロシージャを起動するにはそのユーザが DBMS のユーザでなければならない．DBMS の接続にも同じアカウントを使う．このように提案手法では 1 つのアカウントで DBMS とその中に構築されている XML リポジトリと両方の認証を行う．このためすでに DBMS のアカウントを持っているユーザはそのまま提案手法も利用できる．またストアドプロシージャを使う提案手法の仕組みにより，XML リポジトリ上でも，その DBMS アカウントでのみ XML 文書のインスタンス単位でのアクセス権が適切に処理される．

さてアクセス管理を行うストアドプロシージャの引数はアカウント名，パスワード，URI と述べたが，SAX パーサは URI のみ引数を持つ．提案手法は完全な SAX 互換を目指しているため，SAX の処理に特別なオプションを設けるのは極力排除したい．そこで，URI にアカウント名，パスワードを埋め込んで SAX パーサに渡す手法を提供する．これには非常に一般的な URI の文法を利用する．

http://accout:password@foo.bar/foofile.xml

この形式の URI を SAX パーサに渡せば，そのアカウントに基づき DBMS に接続し，またストアドプロシージャ内でアカウント名はアカウント識別子に，URI はリソース ID に変換され，前章で述べたアクセス権処理の手続きへと進む．このプロセスにより XML リポジトリからアクセス管理された XML 文書を取り出すことができる．

この仕組みにより event テーブル，resource テーブルおよび account テーブルはすべてのユーザから隠すことができる．DBMS のどのようなフロントエンドをユーザが利用しても event テーブルに格納されたデータを取り出すには，このストアドプロシージャを経由しなければならない．ストアドプロシージャを起動するには，正しいアカウントで DBMS に接続し，同じアカウントと正しい URL を引数に設定する必要がある．ここまで述べてきたように，提案手法を用いることにより XML リポジトリと DBMS のユーザ認証は一元化され，またユーザはアクセス権が適切に処理された XML 文書しか呼び出すことができない．

7. 実験と考察

本章では前章までに述べたアクセス管理手法を実際に実装しその性能を実験によって確かめる．提案する手法がデータの量に対してどのような性能を描くか，複数のテストセットを用意して確かめる．実験は複数の DBMS を用いて行っている．

7.1 テストデータと実験環境

実験に用いるテストデータには実際に巨大なデータベースとして機能している DBLP の論文書誌情報 XML 文書を利用した．その DBLP の情報を基に約 800 KB から約 50 MB までの 4 つのテストセット S, M, L, LL を作成した．それぞれの構成を**表 10** に示す．

このデータを用いて XML リポジトリを関係データベース上に構成している．参考として各テストセットを SAX で読み込んだ際にかかる時間を**図 7** に示す．この際 SAX のイベントハンドラは単にイベントを読み捨てているだけであり，この図は純粋に XML 文書を読むのにかかる時間を示している．

さらにテストデータでは，アカウント識別子 11 に対し，すべてのアトリビュート出現イベントにアクセス拒否を設定した．実験機材の環境を**表 11** に示す．

実験に使用する DBMS は Microsoft SQL Server

表 10 テストデータ中の SAX イベント数内訳
Table 10 An amout of SAX events in the test data.

type	S	M	L	LL
1	22509	108346	256836	1282176
2	4512	33566	49290	250413
3	20265	95643	232635	1161450
4	22509	108346	256836	1282176
SUM	69795	345901	795597	3976215

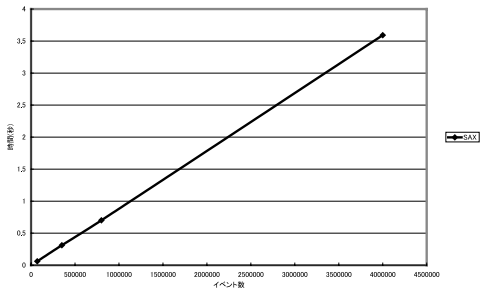

図 7 SAX による XML 文書読み込み時間
Fig. 7 Time for reading XML document.

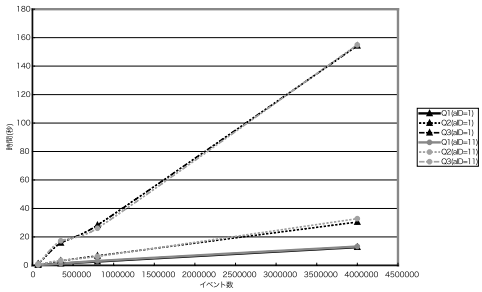

図 8 問合せにかかる時間（MS SQL Server）
Fig. 8 Query time (MS SQL Server).

表 11 実験環境
Table 11 Experimentation environment.

	環境
OS	Windows XP Professional
CPU	Intel Pentium 4 (3.6 GHz)
メモリ	1 GB
ディスク I/F	シリアル ATA/RAID01

2005 CTP，PostgreSQL8.0 および MySQL Server 4.1 である．また，処理時間やコストの取得は，Visual Studio .net で作成した C#によるプログラムを用いている．DBMS には現時点でストアドプロシージャを利用できないものもあり，実験では公平のためにクライアントアプリケーション上で直接クエリを実行している．

7.2 実　　験

本節では前節で述べたデータを利用して実験を行い，提案手法の性能を示す．データ単位でアクセス管理する手法は，XPath 等を利用した条件式によってアクセス権を記述する手法に比べて記述量が多くなってしまうことが考えられる．そこで提案手法では異なるユーザが持つアクセスポリシのうち同じアクセス権をまとめる仕組み Hierarchy Authorization を導入した．Hierarchy Authorization ではアカウントの権限構造が木をなすため，アカウント識別子として木のラベリング手法である Prefix Labeling を利用している．一般論として，関係データベースは木の扱いが苦手とされている．本実験では木の操作をともなうアカウント識別子の処理にかかるコストを計測する．

まず，5.2 節で提案した問合せ手法と比較のため，アカウント識別子の木を操作せず，単純に SAX イベント列のみを取り出す問合せを定義する．

```
01: /* @aID = 1 の問合せとして */
02: SELECT   type,property
03: FROM     event
04: WHERE    type <> 0
05: ORDER BY eID
06:
07: /* @aID = 11 の問合せとして */
08: SELECT   type,property
09: FROM     event
10: WHERE    type <> 0
11: AND      type <> 2
12: ORDER BY eID
```

$@aID = 1$ のときは event テーブルからすべての SAX イベントをイベント順で取り出す．また，$@aID = 11$ のときは event テーブルからアトリビュートに関するイベントを取り除いた結果をイベント順で取り出している．この問合せに比べ，Hierarchy Authorization の木構造データを処理する問合せがどれくらいの時間かかるのかを調べ，それが現実的に許容できる範囲内かどうかを考察する．ここでは説明のため，木の操作を含まない問合せを $Q1$，提案手法の問合せを $Q2$，自己結合による問合せを $Q3$ として扱う．

実験は 4 つのテストデータに対して，$@aID = 1$ および $@aID = 11$ のユーザからの問合せをそれぞれ 10 回行い，実行時間の平均値を結果とした．この手順で前述した 3 つの DBMS すべてにおいて実験を行った．

まず Microsoft SQL Server 2005 CTP を用いて $Q1$，$Q2$，$Q3$ の問合せを行った．結果を**図 8** に示す．$Q3$ は他の問合せ手法に比べ効率が悪い．これは自己結合の際に行どうしの直積が発生するためと考えられる．

次に $Q1$ および $Q2$ に関して他の 2 つの DBMS 上で問合せを行った．MySQL Server 4.1 での結果を**図 9** に，PostgreSQL8.0 での結果を**図 10** に示す．

MySQL の $Q2$ では SELECT 句で MAX 関数の代わりに MySQL 独自の関数 bit_and 関数，group_concat 関数を利用している．bit_and 関数は指定されたフィールドとの値すべて and 演算で足し合

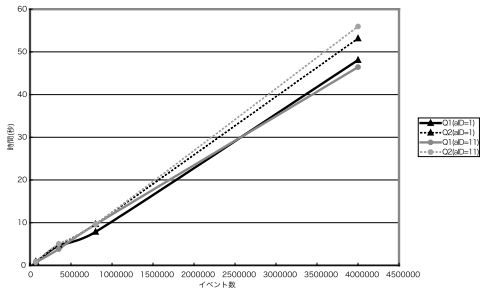

図 9 問合せにかかる時間（MySQL）
Fig. 9 Query time (MySQL).

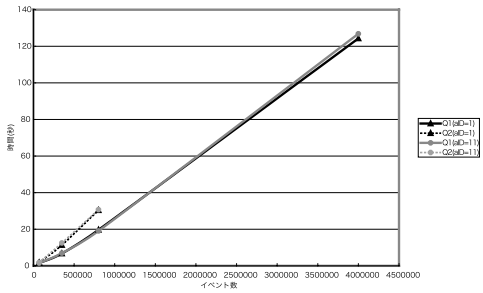

図 10 問合せにかかる時間（PostgreSQL）
Fig. 10 Query time (PostgreSQL).

わせる関数で type フィールドを引数に設定している．また group_concat 関数は指定されたフィールドの文字列をすべて連結する関数で peorperty フィールドを引数に設定した．5.2 節で述べたが，提案手法の規定と HAVING 句に指定された条件により，SELECT 句で判断されるグループはそれぞれ 1 行のデータだけを持つことが保障されている．つまり and 演算においても，文字列を足し合わせる演算においても，その行が持つ type 値と proeprty 値が返される．これらの関数は MySQL 上でともに MAX 関数よりも速く動作するため，利用している．

テストデータ LL において，PostgreSQL の $Q2$ は我々の実験環境では現実的な時間内で終了しなかった．そのため図にプロットしていない．問合せ中にメモリを消費してしまいスワップ操作が頻繁に発生したため，それにかかる時間が結果に含まれており，問合せの性質を正確に現すことは困難であった．参考として，ユーザ（@$aID = 1$）からのアクセスが 771.5 秒，ユーザ（@$aID = 11$）のからのアクセスが 655.4 秒かかった．

7.3 考察および議論

DBMS による性能としては，同じ問合せにおいては Microsft SQL Server が一番速く動作する．これは Microsoft SQL Server の機能であるクラスタ化インデクスが有効に働いているものと考えられる．提案手法では eID フィールドに対してクラスタ化インデクスを昇順で作成している．クラスタ化インデクスを作成した場合そのフィールド値の昇順あるいは降順で，実際のストレージ上でもデータが並ぶ．つまり Microsoft SQL Server のストレージにおいては，すでに SAX のイベント発生順序でデータが並んでおり，問合せ中の ORDER BY 句で指定した eID の昇順ソートにかかるコストを抑えられる．

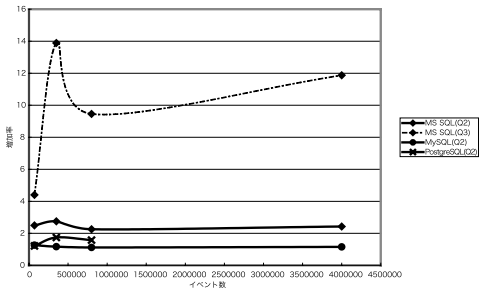

図 11 クエリ時間増加率
Fig. 11 The rate of time over increase.

次に Hierarchy Authorization の処理を行わない問合せ $Q1$ とその他の問合せの比較を行う．**図 11** は，$Q1$ の実行時間に対してその他の問合せが何倍の時間がかかったのかを描いている．

この図から $Q3$ に比べ $Q2$ の手法が総じて優れていることが分かる．また DBMS の種類によらず，同じ程度の結果を得ている．特に MySQL を利用した場合，平均して問合せ時間の増加率は 1.18 倍であり，アクセス管理を行わない場合と比べ，非常にわずかな時間で Hierarchy Authorization の木構造の処理を行っていることが分かる．

一般論として，木構造の処理は関係データベースで行うことは困難とされている．提案システムはユーザの持つ権限構造がデータ量増加を防ぐために木構造をとっている，しかしながら我々の提案する問合せ手法おいては，木構造をとるアクセス権の処理を非常に効率的に行えることを示した．

8. まとめと今後の課題

本論文では関係データベースを利用したスキーマによらない 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ことができる．

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を継承する．

3 章で指摘したように，固定された関係データベーススキーマに，あらゆるスキーマの XML 文書を格納するシステムでは，関係データベースのアクセス権と XML リポジトリのアクセス権が同期していることが重要である．提案手法では同一テーブル内に XML 文書のデータと一緒にアクセス権の情報も格納し，ストアドプロシージャを用いてのみ問合せを行う仕組みによって，アクセス権管理の一元化を行っている．

また実験により，提案手法の動作速度およびコストを計測した．提案手法のアクセス管理は Prefix Labeling Scheme の処理に基づいている．ユーザの権限構造は木となるが，一般的に木の操作は関係データベースには向かないとされている．しかし提案手法では，実験の結果が示すように，ユーザの権限構造を示す木の処理にかかるコストはわずかである．

提案手法の特徴は，アクセスポリシ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，および関係データベースと XML リポジトリを統合することにより，セキュリティの一元管理を行うことができることを示した．

提案手法の持つアクセス権は『アクセス拒否』のみである．XML インスタンス中の SAX イベント 1 つ 1 つにアクセス権を設定できる仕組みによって，アクセス管理を行っている．また利用方法からの命名として 2 種類の注釈を定義した．注釈を利用するとユーザは自分のための情報を 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ことができる．

今後の課題としては，提案手法を利用したアプリケーションの開発を考えている．たとえば我々が提案している創発的 XML[12] のストレージとして本手法を利用することを検討していきたい．また，SAXOPHONE 上で版管理を行う研究[16] との統合も視野にいれている．また，提案手法の改良点としては，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こと等が考えられる．

また SAXOPHONE の機能として XML 文書の内部に対して直接問合せを行う仕組みも検討したい．SAXOPHONE の最も大きな利点は SAX APIs を通してユーザがアクセスできることであり，その利便性やプログラミングスタイルを阻害することなく，XML 文書の内部に直接アクセスする手法について SAX APIs の拡張も含めて議論していきたい．

参 考 文 献

1) The World Wide Web Consortium (W3C): Extensible Markup Language (XML), Web. http://www.w3.org/XML/ (Accessed 1 June 2005).
2) 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 (2002).
3) Kudo, M. and Hada, S.: XML document security based on provisional authorization, *Conference on Computer and Communications Security*, pp.87–96 (2000).
4) Anutariya, C., Chatvichienchai, S., Iwiahara, M., Wuwongse, V. and Kambayashi, Y.: A Rule-Based XML Access Control Model, *Rules and Rule Markup Languages for the Semantic Web: 2 International Workshop* (*RuleML*) (2003).
5) OASIS: eXtensible Access Control Markup Language (XACML), Web. http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf (Accessed 1 June 2005).
6) The National Institute of Standards and Technology: Role Based Access Control (RBAC), Web. http://csrc.nist.gov/rbac/ (Accessed 1 June 2005).
7) The Apache Software Foundation: Apache Xindice, Web. http://xml.apache.org/xindice/ (Accessed 1 June 2005).
8) Yoshikawa, M., Amagasa, T., Shimura, T. and Uemura, S.: XRel: A Path-Based Approach to Storage and Retrieval of XML Documents using Relational Databases, *ACM Trans. Internet Technology*, Vol.1, No.1, pp.110–141 (2001).
9) Fernandez, M.F., Kadiyska, Y., Morishima, A., Suciu, D. and Tan, W.-C.: SilkRoute : A Framework for Publishing Relational Data in XML, *ACM TODS*, Vol.27, No.4, pp.438–493

(2002).
10) Carey, M., Florescu, D., Ives, Z., Lu, Y., Shanmugasundaram, J., Shekita, E. and Subramanian, S.: XPERANTO: Publishing Object-Relational Data as XML, *Workshop on Web and Databases* (*WebDB*) (2000).
11) Damiani, M., di Vimercati, S.D.C. and Paraboschi, S.: A finegrained access control system for XML documents, *ACM Trans. Internet Technology*, Vol.5, No.2, pp.169–202 (2002).
12) 石川 博, 片山 薫, 高橋俊介, 長屋未来, 布施貴義, 横山昌平：創発的 XML の提案, 夏のデータベースワークショップ（DBWS2005）論文集 (2005).
13) Li, Q. and Moon, B.: Indexing and Querying XML Data for Regular Path Expressions, *Proc. 27th International Conference on Very Large Data Bases* (*VLDB*), Roma, Italy (2001).
14) Abiteboul, S., Kaplan, H. and Milo, T.: Compact labeling schemes for ancestor queries, *12 Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*), Washington, D.C., USA, pp.547–556 (2001).
15) Silberstein, A., He, H., Yi, K. and Yang, J.: BOXes: Efficient Maintenance of Order-Based Labeling for Dynamic XML Data, *Proc. 21st International Conference on Data Engineering* (*ICDE*) (2005).
16) 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を有するバージョン系列の為の関係データベース構成法, 電子情報通信学会論文誌, Vol.J87-D-I, No.2, pp.97–112 (2004).
17) Leeuwen, J.V.: *Algorithms and Complexity*, Elseveir Science Publishers (1990). 廣瀬 健ほか（訳）：アルゴリズムと複雑さ, 丸善株式会社 (1994).

(平成 17 年 6 月 21 日受付)
(平成 17 年 10 月 13 日採録)

（担当編集委員 天笠 俊之）

横山 昌平

1977 年生．2001 年東京都立大学工学部電子情報工学科卒業．2003 年東京都立大学大学院工学研究科電気工学専攻修士課程修了．2003 年 4 月より同大学院博士課程に在籍中．RDB と XML データ処理の連携に興味を持つ．

太田 学（正会員）

岡山大学工学部情報工学科助教授．1999 年東京大学大学院工学系研究科電気工学専攻博士課程修了，博士（工学）．東京都立大学大学院工学研究科助手を経て 2005 年より現職．情報検索，データマイニングとその Web 応用の研究に従事．日本データベース学会会員．

片山 薫（正会員）

東京都立大学大学院工学研究科助手．2000 年京都大学大学院情報学研究科社会情報学専攻博士後期課程修了，博士（情報学）．データベースシステムに関する研究開発に従事．日本データベース学会会員．

石川 博（正会員）

東京都立大学大学院工学研究科教授．東京大学理学部情報科学科卒業．富士通研究所を経て 2000 年より現職．東京大学博士（理学）．著書に “Object-Oriented Database System”（Springer Verlag），“Database and Data Communication Network Systems: Techniques and Applications”（共著，Elsevier），『e-ビジネス技術入門教科書—ビジネスモデルと情報技術（IT）IT TEXT』（CQ 出版），『次世代データベースとデータマイニング—DB&DM の基礎と Web・XML・P2P への適用』（CQ 出版）等．国際論文誌 ACM TODS，IEEE TKDE，国際学会 VLDB，IEEE ICDE 等，学術論文多数．1994 年情報処理学会坂井記念特別賞，1997 年科学技術庁長官賞（研究功績者）受賞．現在，情報処理学会データベースシステム研究会主査．情報処理学会論文誌（データベース）共同編集委員長．International Journal Very Large Data Bases Editorial Board．日本データベース学会理事．仏国ナント大学招聘教授．電子情報通信学会，ACM，IEEE 各会員．