JAIST Repository
https://dspace.jaist.ac.jp/

Title	XML問合せにおけるアクセス制御実装方式
Author(s)	紺野，将司
Citation	
Issue Date	2006-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/1972
Rights	
Description	Supervisor:田島　敬史，情報科学研究科，修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

修 士 論 文

XML 問合せにおけるアクセス制御実装方式

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

紺野将司

2006 年 3 月

修 士 論 文

XML 問合せにおけるアクセス制御実装方式

指導教官　田島 敬史 助教授

審査委員主査　田島 敬史 助教授
審査委員　日比野 靖 教授
審査委員　二木 厚吉 教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

410051 紺野将司

提出年月: 2006 年 2 月

Copyright © 2006 by Masashi Konno

概 要

近年，電子カルテ等の個人情報を扱うシステムへの XML データベースの活用が研究されている．個人情報を扱う上で必要になるのがアクセス制御である．そこで，本研究では，XML データに対する問合せ処理の中で，効率良くアクセス制御を行う手法についての研究を行う．アクセス制御の代表的な実装方法としては，あらかじめ問い合わせ変形を用いてアクセス制御を行う手法と，問合わせ処理の中で各データ要素に付加したセキュリティレベルの情報を参照してアクセス制御の判定を行う手法の二つがあるが，本研究では，これら二つの手法を問い合わせの中間解，最終解の大きさや，問合わせ式の複雑さに応じて使い分けることによって，効率の良いアクセス制御を実現する．

目 次

第 1 章　はじめに **1**
1.1　背景と目的 1
1.2　論文の構成 2

第 2 章　基本事項と関連研究 **3**
2.1　XML 3
2.2　XPath 3
2.3　SAX 4
2.4　XMark 4
2.5　DEWEYORDER 4
2.6　関連研究 5

第 3 章　提案する手法 **7**

第 4 章　実験 **9**
4.1　手順 9
4.1.1　XML 形式から CSV 形式への変換 9
4.1.2　テーブルの定義 10
4.1.3　XPath から SQL への変換 12
4.1.4　実験に使用したデータ 12
4.2　実験 1 13
4.3　実験 2 22
4.4　実験 3 26

第 5 章　まとめと今後の課題 **32**

第1章　はじめに

1.1　背景と目的

XML は複雑なデータ構造を可能としたデータ形式であり，近年は電子カルテ等の個人情報を扱うシステムや電子商取引等のシステムに広く活用されている．これらのシステムではアクセス制御の機能が必須になるが，XML データは，関係データベースで扱うデータとは異なり，出現するデータ項目が不規則であったり，タグが任意の深さまでネストしていたりする為，あるデータ項目があるユーザにとってアクセス可能かどうかのアクセスポリシーの定義の仕方が関係データベースに比べて簡単ではない．これまでの研究では，XML データの根から各ノードへのパスのパターンをパス式を使って記述し，これらのパターン毎に，各ユーザがアクセス可能かどうかを定義する方法が主流である．しかし，このようにパス式を使ってアクセスポリシーを定義する場合，各データについて，そのデータをあるユーザがアクセス可能かどうかを判定する計算が関係データベースと比べて複雑になりコストが高くなるという問題がある．XML データベースに限らず，データベースに対する問い合わせ処理においてアクセス制御を行う場合の既存の手法としては,「各データのセキュリティレベルに関するデータをあらかじめ求めておき，与えられた問合せを実行する際に，アクセスした各データがそのユーザにとってアクセス可能かどうかを，そのセキュリティレベルに関するデータを参照して確認しながら実行していく」という方法と,「与えられた問い合わせ式を，実行する前に事前に，アクセス不可能な場所はアクセスしない問合わせ式に変形してから実行する」という方法の 2 つがある．両者の長所，短所を挙げると，前者の方法は，問合わせの実行中に最終的にアクセスされ，アクセス判定を行わなければならないデータの数が非常に少ない場合には，効率が良いという点が長所として挙げられる．一方，逆に，問合わせ実行中にアクセスされるデータの数が非常に多い場合には，それらのデータについて逐一，アクセス判定を行いながら問合わせを実行しなければならないため，非常に効率が悪くなるのが短所である．後者の方法は，問い合わせの段階でアクセスポリシーを適用するので，データ量が多い場合でも時間をかけることなく問い合わせが出来るという長所があるが，問合せ言語の表現力によっては，問合せ式を，アクセスポリシーを反映した問合せ式にうまく変形できない場合があったり [1]，あるいは問合せ式の変形は可能だが，非常に複雑な問合せ式になり，その実行に前者の手法以上に時間がかかってしまう場合がある点が短所となる．

これらがよく知られている手法であるが，XML データベースに対してアクセス制御を行う場合，前述のようにパス式によるアクセス制限の定義が必要なため，上述の両手法の

長所，短所が特に顕著となる．そこで，本研究では，そのような複雑なパス式によりアクセス制御を行う場合の効率的な問合わせ評価方法についての研究を行う．具体的には，基本的には問い合わせ変形を用いてアクセス制御を行うが，要素に付加した情報を用いてアクセス判定を行う方が効率が良い場合には状況に応じて後者の方法も使い分ける手法を検討する．

1.2 論文の構成

本論文は以下の様に構成する．本章で背景，2 章で，基本事項と関連研究，３章で提案する手法について，４章で実験，５章でまとめと今後の課題について述べる．

第2章　基本事項と関連研究

まず，本章では，この研究で前提として考える環境について説明する．

2.1　XML

XML データ形式は，要素にラベルの付いた木構造で表現することができる．例として挙げると図 2.1 の様な XML データは，図 2.2 の様に表現する事が出来る．

```
<root>
  <people id=1>
  <name>A</name>
  <age>24</age>
  <phone>
  <home>111-1111</home>
  <mobile>112-2211</mobile>
  </phone>
</root>
```

図 2.1: XML データの例

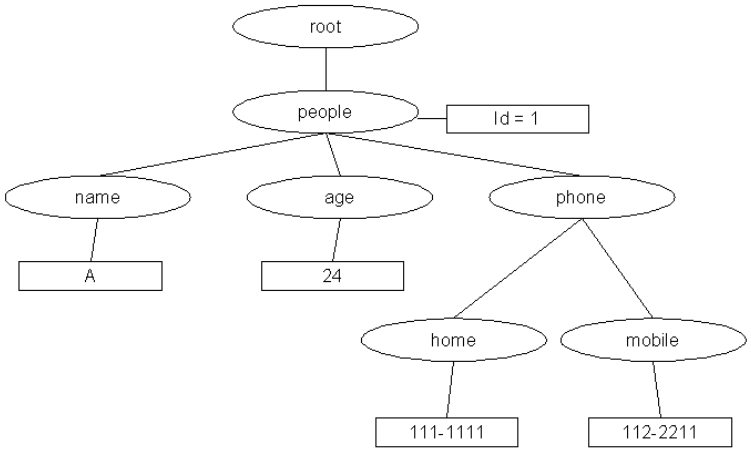

図 2.2: XML データの木構造

2.2　XPath

XPath [4] とは，XML データの特定の部分を示す言語であり，式を記述することにより XML データの任意の部分を示す事が出来る．以下に例を示す．

XPath 式	示すノード
/A/B/C	A の子であるBの子であるC
/A//B/C	Aの任意の深さの子孫Bを親として持つC
/A/B/C[E]	AのBのEを子として持つC

2.3 SAX

SAX[5] とは，W3C が公式に提供したインターフェイスではなく XML-DEV メーリングリストにより開発されたインターフェースである．特徴としては，ドキュメントを読み込みながら処理を行う事が可能である．同様のＡＰＩに DOM が在るが，DOM[6] とは異なり，反応が早く，メモリの消費量が少ないプログラムを書くことが可能である．具体的には，XML 文章を先頭から読み込んで行き，開始タグ，終了タグ，文字列等が出現する度に，対応するイベントを発生させるイベント駆動型のインターフェイスである．

SAX で通知可能なイベント

- ドキュメントの開始，終了
- 要素の開始，終了
- 名前空間の開始，終了
- 文字列の受け取り
- エラー情報

2.4 XMark

XMark[7] は，データベースのベンチマーク用データを生成するツールであり，スケールファクタを設定することにより，ランダムなデータ構造を持つ任意のサイズの XML データを生成する事が可能である．

2.5 DEWEYORDER

DEWEYORDER[8] とは，XML データの各要素に対してラベリングを行う手法のうちの１つであり，図書館学の分類方であるデューイ十進分類法 (Dewey Decimal Classification) を応用したラベリングの手法である．DEWEYORDER は，祖先・子孫関係だけでなく，親子関係の判定も可能であり，兄弟の順番も判定可能である．以下に例を示す．

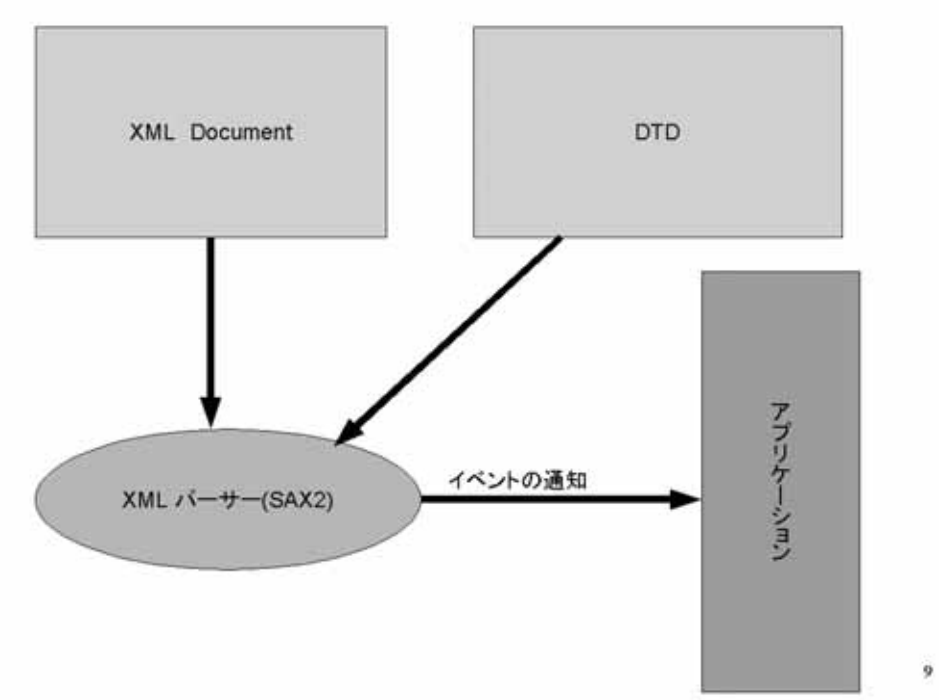

図 2.3: SAX での処理

DEWEYORDER の例

図 2.4 の様な XML データを仮定する．ルートであるノードには 1 を割り振り，兄弟のノードに対しては，i 番目のノードのノードラベルは，親のノードラベル p とする時"p, i"となる．ラベリングを行った結果は図 2.5 である．

2.6 関連研究

関連研究 としては，[2] が挙げられる．この研究は，Path 式にアクセスポリシーを付加し，問い合わせが行われたパス式に対してアクセス判定を行うことでアクセス判定の効率化を行っている．

```
<bibliography>
  <book>
    <year>2005</year>
    <title>research for XML</title>
    <authors>
        <author>ABC</author>
        <author>DEF</author>
    </authors>
    <price>3000</price>
    <publisher>A</publisher>
    <comments>good book</comments>
  </book>
<bibliography>
```

図 2.4: XML データの例 2

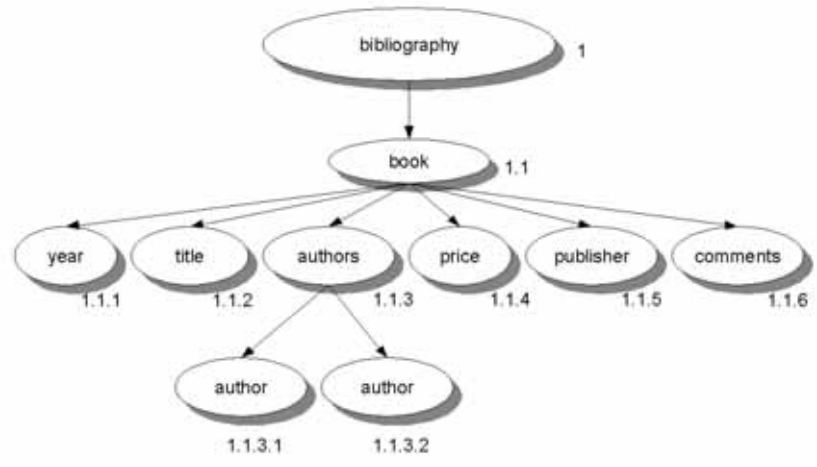

図 2.5: DEWEYORDER の割り振り

第3章　提案する手法

本研究では，効率の良い XML データのアクセス制御として，問合せ式変形によるアクセス制御と，ノードに付加した情報を用いてアクセス制御を行う手法の両方を使い分けてアクセス制御を行う手法を提案する．手法の切り替えは，問い合わせに対する解の大きさや, ユーザーからの問合せの式のパターンやアクセスポリシーの条件により使い分けを行い，両者の長所を利用し，効率的なアクセス制御が実現できるようにする．ノード情報を用いた手法は，ノード自身にユーザーグループ単位でのアクセス判定を行えるよう，グループ数を満たすビット列を割り振る．また，問合せ式変形による手法は，単純なパス式で設定されたアクセスポリシーに対しては, 条件式で, そのパスを除いた条件にし, ノードの値がアクセスポリシーの条件として設定された場合には, ノードに割り振った DEWEYORDER の値を条件に用いて問合せ変形を行う．また, 問合せ変形のアプローチに関しては,1 つの問合せ式に変形して問合せを行う手法と，問合せを実行した解より，アクセス不可の部分を取り除く手法を場合により使い分ける．

問合せ式変形の例

XML のアクセス不可の部分は，パス式用いて設定するので，SQL に変換する際は，条件にアクセス不可のパスを持たないという条件を追加する．また，特定の属性や値を持たないというアクセス制御に関しては，DEWEYORDER を用いた条件式を用いてアクセス制御を行う．以下に例を示す.

単純パス式によるアクセス制御の問合せ変形の例

図 4.1 の XML データを例に挙げる．問合せ式を”/people/person”とし，アクセスポリシーを”/people/person/name”,”/site/person/age” とした場合，SQL は，以下の様になる．

```
PATH LIKE '/people/person%'
AND PATH != '/people/person/name'
AND PATH != '/people/person/age'
```

値によるアクセス制御の問合せ変形の例

前例と同様に，図 4.1 を例に挙げる．問合せ式を”/people/person”とし，アクセスポリシーを”gender が F である person はアクセス不可である”とした場合，gender が F である people の DEWEYORDER を求め，その値の prefix でないものを選択するという条件になる．SQL で記述した場合は，以下のようになる．

```
SELECT f1.* FROM field f1,field f2,field f3
WHERE f2.PATH    ='/people/person'
AND     f3.PATH    ='/people/person/gender'
AND     f3.VALUE  != 'F'
AND     f2.DEWEYORDER   = substring(f2.DEWEYORDER,1,LENGTH(f1.DEWEYORDER))
AND     f1.DEWEYORDER = f2.DEWEYORDER
```

ノード情報を用いるアクセス制御の問合せ変形の例

同じく，前例と同様に，図 4.1 を例に挙げる．問合せ式，およびアクセスポリシーは，前例と同じにする．アクセスポリシーにあたるノードには，仮に 3 という値をアクセスポリシーに関する情報として付加しておく．この場合，問合せは以下の SQL で記述される．

```
SELECT * FROM field f1
WHERE     f1.PATH like '/people/person\%'
AND ACCESSPOLYCY != 3
```

第4章　実験

本章では，提案する手法がアクセス制御の効率化を実現するか，SQL の処理時間を計測する実験を行った．実験環境は，以下に示す．

実験環境

ハードウェア	OS	Memory
サーバ	SunOS5.8	6[Gbyte]
クライアント	Windows2000	1[Gbyte]
ソフトウェア	Oracle9	
	Oracle9i Client	

4.1　手順

XML データは，そのまま形式であると関係データベースに格納できない為，XML データを CSV 形式へエンコードして関係データベースへ格納する．想定した XPath の問い合わせ式とアクセスポリシーを適応し，アクセスポリシーを適用した SQL に変換し，問い合わせを行い，その処理時間を計測する．また，アクセス制御を適応した SQL は，SQL にアクセスポリシーを適用し，１つの SQL にまとめて記述する方式 (以下 A1 とする)，目的のパスを持つ式を検索し，そこからアクセスポリシーでアクセス不可である部分を取り除く方式 (以下 A2)，ノードにアクセス判定を行うための情報を付加し，それによりアクセス判定を行う方式 (以下 A3 とする) をそれぞれ用いて，問合せ変形を行う．

4.1.1　XML 形式から CSV 形式への変換

エンコードには，Java に実装されている SAX を使用して，XML データを読み込み，CSV にエンコードを行う．尚，XMark にてデータを作成した為，ノードによってはテキスト値を持たない場合が多々存在した．その為，CSV に変換するに当たり，テキスト値を持たない場合は ’0 ’を代入した．また，ノードの値，属性は全て文字列として変換を行った．以下に例を示す．

例

図 4.1 の様な XML データの場合，根の部分である”people”や二人目の people の”age”の部分はテキスト値を持っていない．そこで，’0 ’を代入し，CSV に変換した．

```
<people>
  <person ID = 1>
    <name>ABC</name>
    <age>25</age>
    <mailaddress>ABC@mail</mailaddress>
    <gender>M</gender>
  </person>
  <person ID = 2>
  <person>
    <name>DEF</name>
    <age></age>
    <mailaddress>DEF@mail</mailaddress>
    <gender>F</gender>
  </person>
</people>
```

[3]

図 4.1: XML データの例 3

4.1.2 テーブルの定義

作成する表は，”Node”と”Path”とし，各フィールドを次のように設定する．

- NODE(NID,PATHID,VALUE,LENGTH,OFFSET,ATTNAME, ATTVALUE,DEWEYORDER,A_POLYCY)
- PATH(PATHID,PATH)

各変数の説明は以下の表の通りである．
尚，”Node”表の PATHID，A_POLYCY，”Path”表の PATHID にインデックスを付け，検索速度が向上するようにした．データベースへの格納は，SQL*Loder を用いて行い，CSV データを格納した後に，”Node”と”Path”の PATHID にインデックスを貼り，検索速度が向上するようにした．また，値を持たずに’0’として格納した行においては冗長である為，削除を行った．

NID	PATHID	TAG	VALUE	LENGTH	OFFSET	ATTNAME	ATTVALUE	DEWEYORDER
0	0	People	0	0	0	0	0	1
1	1	Person	0	0	0	ID	1	1.1
2	2	Name	ABC	3	29	0	0	1.1.1
3	3	Age	25	2	44	0	0	1.1.2
4	4	Mailaddress	ABC@mail	8	65	0	0	1.1.3
5	5	Gender	M	1	95	0	0	1.1.4
6	1	Person	0	0	0	ID	2	1.2
7	2	Name	DEF	3	135	0	0	1.2.1
8	3	Age	0	0	0	0	0	1.2.2
9	4	mailaddress	DEF@mail	8	169	0	0	1.2.3
10	5	Gender	F	1	199	0	0	1.2.4

5

図 4.2: ”Node”への格納

PATHID	PATH
0	/people
1	/people/person
2	/people/person/name
3	/people/person/age
4	/people/person/mailaddress
5	/people/person/gender

4

図 4.3: ”Path”への格納

変数名	内容
NID	ノードにつけた ID 番号
PATHID	パスにつけた ID 番号
VALUE	各ノードが持つ値
LENGTH	各ノードの値の長さ
OFFSET	値のオフセット
ATTNAME	ノードの属性の名称
ATTVALUE	ノードの属性の値
DEWEYORDER	ルートから割り振ったノードラベリングの値
PATH	ルートからノードまでのパス式
A_POLYCY	ノードに付加したアクセス制御の為の情報

表 4.1: 変数の説明

4.1.3 XPath から SQL への変換

実験を行う為に設定した問い合わせ式とアクセスポリシーを以下に示す．これらの XPath による問い合わせ式及びアクセスポリシーは，手動で SQL に変換する．また，SQL は 4.1 節で記述した A1，A2，A3 の 3 つのアプローチで変換を行う．

4.1.4 実験に使用したデータ

使用した XML データは，XMark を用いて作成したデータを使用した．また，XMark で作成したデータは以下のようになった．

SF(スケールファクタ)	ノード数	データサイズ [Mbyte]
0.01	33,301	4.05
0.03	97,786	12.1
0.05	162,757	20.3

4.2　実験 1

この実験では，単純なパス式によるアクセスポリシーを設定し，アクセスポリシーの比率を変える事により各アプローチにより式変形を行った場合，どのアプローチが最も効率よくアクセス制御を行えるか比較するため，処理時間を計測した．尚，キャッシュ無しにおいての計測は，キャッシュの影響を極力減らすために，キャッシュのクリアを行い，複数の表に複数作成し，キャッシュをクリアした後に同一の表に複数回アクセスしないよう実験を行った．

設定した問合せ式とアクセスポリシー

問合せ式と各問合せに対してのアクセスポリシーは表 4.2 の様に設定した．尚，解とアクセスポリシーの比率は，(1)，(2) はアクセス可能なノードの比率が多く，(3)，(4) は，ほぼ同じ比率，(5)，(6) はアクセス不可なノードの比率が多い．

実験結果

実験結果は，表 4.3 から表 4.8　に示す．

考察

各アプローチ毎に処理時間を計測したが，スケールファクタが小さい場合には，全体的にノード情報を用いた手法 (A3) が処理時間が早いという結果になった．これは，パスを示す”PATH”とアクセスポリシーを示す A_POLYCY”に対して，インデックスを張っているため，パスとインデックスを用いてアクセス可能なノードを検索しているため，パス式のパターン検索を行うＡ１，Ａ２より高速に処理が可能であると考える事が出来る．また，スケールファクタが大きくなるにつれ，A1 の処理時間が短くなっていくという特徴については，スケールファクタが増えていくと，それに伴いデータ量も増加していく為，判定しなくてはいけないノードが増える．そこで, 予めパスにより，アクセス可能なノードを判定しておき, ディスクにアクセスするため，ディスクＩ/O が減少し，処理時間を短くしたと考えられる．また式番号 (2) と（4）においては，A1 の方が処理時間が短いのは，ユーザーからの問合せ式が複数のパスにより構成されているので，判定するノードが増加することにより，アクセスポリシーの判定に時間が掛かってしまうため，処理時間が長くなってしまうと考えられる．よって, データ量が多く，アクセスポリシーが全てパス式で設定され，キャッシュが無い場合では，A1 によるアプローチで問合せ変形を行った方が効率が良いという事が確認できた．また，ユーザーからの問合せ式より，解を出して，そこからアクセス不可の部分を引いた処理をする SQL に関しては，解に対してアクセス不可の部分が僅かであっても，処理時間を短縮出来なかった．これは，問い合わせの中間

式番号	問い合わせ式	アクセスポリシー
(1)	/site/regions	/site/regions//location/site//name /site/regions//mailbox /site//name
(2)	/site/regions /site/people /site/open_auctions	/site/people/person/address
(3)	/site/open_auctions	/site/open_auctions/open_auction/annotation /site/open_auctions//seller /site/open_auctions/open_auction/bidder/date /site/open_auctions/open_auction/initial /site/open_auctions/open_auction/interval
(4)	/site/regions /site/people	/site/regions/africa/item/location /site/regions/asia/item/description /site/regions/australia/item/mailbox /site/regions/europe/item /site/regions/namerica/item/description /site/regions/samerica/item/description /site/people/person/profile /site/people/person/address
(5)	/site/regions	/site/regions/namerica /site/regions/samerica /site/regions//location /site/regions//mailbox /site//name
(6)	/site/regions	/site/regions/namerica /site/regions/samerica /site/regions//location /site/regions//mailbox /site/regions//name /site/regions//payment /site/regions/asia/item /site/regions//shipping /site/regions//quantity /site/regions//item/description /site/regions/africa /site/regions/europe

表 4.2: 実験 1 の問い合わせ式とアクセスポリシー

式番号	A1[s]	A2[s]	A3[s]
(1)	8.00	31:06	6.09
	8.28	31.02	6.08
	7.09	33.07	6.08
(2)	12.01	14.06	13.06
	12.01	14.07	24.02
	12.01	14.06	17.00
(3)	3.02	13.02	3.02
	3.02	13.03	12.00
	3.02	13.04	2.09
(4)	8.07	11.00	21.00
	8.07	11.00	6.05
	8.07	11.00	17.02
(5)	7.08	34.07	5.09
	7.05	34.05	17.01
	7.05	34.07	5.07
(6)	34.07	60.05	5.07
	7.01	60.03	17.04
	7.01	60.09	4.09

表 4.3: SF:0.01, キャッシュ無し

このデータは，スケールファクター 0.01 のデータを用いて，キャッシュのクリアを 1 回の計測毎に行った結果である．

式番号	A1[s]	A2[s]	A3[s]
(1)	7.08	15.05	6.08
	7.08	15.02	6.07
	7.09	15.05	6.07
(2)	12.05	14.07	11.07
	12.01	14.06	11.07
	12.00	18.00	11.07
(3)	3.03	13.03	2.08
	3.02	13.03	2.08
	3.03	13.03	2.05
(4)	8.07	11.00	6.05
	8.08	11.00	6.06
	8.08	11.00	6.04
(5)	7.05	34.08	5.06
	7.05	34.06	5.06
	7.06	34.09	5.06
(6)	7.01	60.01	4.08
	7.04	60.08	4.08
	7.04	60.06	4.08

表 4.4: SF:0.01 キャッシュ有り

このデータは，スケールファクター 0.01 のデータを用いて一度検索を行った後，連続して検索を行い計測を行った結果である．

式番号	A1[s]	A2[s]	A3[s]
(1)	43.09	12.01	17.06
	41.08	11.00	14.00
	53.06	11.00	04.03
(2)	9.01	22.04	40.97
	9.03	20.02	60.08
	12.07	20.03	41.04
(3)	3.05	9.01	44.05
	3.06	8.09	27.00
	3.05	8.00	41.04
(4)	5.02	13.03	4.04
	5.03	12.08	26.01
	5.02	12.08	32.05
(5)	3.06	10.09	18.05
	3.05	10.05	16.07
	3.05	10.05	6.02
(6)	2.03	10.05	39.04
	2.03	10.06	33.09
	2.02	10.05	27.07

表 4.5: SF:0.03, キャッシュ無し

このデータは，スケールファクター 0.03 のデータを用いて，キャッシュのクリアを 1 回の計測毎に行った結果である．

式番号	A1[s]	A2[s]	A3[s]
(1)	4.03	10.03	4.05
	4.03	10.00	4.02
	4.04	10.01	4.02
(2)	10.01	20.01	9.05
	10.01	20.07	9.05
	10.01	20.03	9.06
(3)	3.08	8.00	3.00
	3.05	8.00	2.09
	3.05	8.02	2.09
(4)	4.09	13.00	4.00
	4.09	12.08	4.00
	4.09	12.07	4.01
(5)	3.07	10.08	2.07
	3.05	10.05	2.07
	3.05	10.07	2.07
(6)	2.02	10.04	1.05
	2.03	10.05	1.04
	2.02	10.05	3.01

表 4.6: SF:0.03, キャッシュ有り

このデータは，スケールファクター 0.03 のデータを用いて一度検索を行った後，連続して検索を行い計測を行った結果である．

式番号	A1[s]	A2[s]	A3[s]
(1)	6.07	19.07	21.00
	6.07	17.09	8.09
	6.07	17.06	11.00
(2)	15.09	39.09	96.06
	14.07	37.07	107.02
	14.08	37.08	69.04
(3)	5.07	15.03	58.02
	5.02	12.08	66.05
	5.02	12.09	66.01
(4)	7.08	23.07	77.04
	7.04	23.09	47.08
	7.05	24.00	49.03
(5)	5.06	18.04	47.05
	5.02	18.03	55.00
	5.01	18.02	111.08
(6)	3.09	22.04	31.02
	3.03	19.06	98.06
	3.03	19.03	57.04

表 4.7: SF:0.05, キャッシュ無し

このデータは，スケールファクター 0.05 のデータを用いて，キャッシュのクリアを 1 回の計測毎に行った結果である．

式番号	A1[s]	A2[s]	A3[s]
(1)	6.07	17.09	7.01
	6.07	17.09	6.05
	6.08	17.09	6.05
(2)	14.07	37.01	14.05
	14.06	37.06	14.06
	14.07	38.01	14.05
(3)	5.03	12.08	18.07
	5.02	12.09	4.04
	5.02	12.08	14.01
(4)	7.04	23.06	5.09
	7.04	23.05	5.09
	7.04	23.07	5.08
(5)	5.01	18.04	3.09
	5.01	18.04	3.09
	5.01	18.01	3.09
(6)	3.03	19.04	2.00
	3.03	19.05	2.00
	3.03	19.07	2.00

表 4.8: SF:0.05, キャッシュ有り

このデータは，スケールファクター 0.05 のデータを用いて一度検索を行った後，連続して検索を行い計測を行った結果である．

解の表と，アクセスポリシーで取り除かなくてはならない表の両方を作成し，差分を取っているため，処理に時間が掛かっていると考えられる．

4.3 実験 2

この実験では，あるノードが特定の値を持つ場合，アクセス付加にするアクセスポリシーを設定し，各アプローチによる処理時間を計測する．ただし，この実験では，実験 1 の結果より，A 2 のアプローチは，効率的ではない事が判明したので，A1 と A3 を用いて問合せ変形を行う．尚，実験 1 と同様に，キャッシュ無しの条件においては，キャッシュをクリアした後に，同一データを別の表に複数作り，同じ表に複数アクセスしないよう実験を行った．

問合せ式とアクセスポリシー

問合せ式と各問合せに対してのアクセスポリシーは表 4.9 の様に設定した．

式番号	問い合わせ式	アクセスポリシー
(7)	/site/open_auctions/open_auction	/site/open_auction/open_auction[privacy='Yes']
(8)	/site/people/person	/site/people/person/profile[gender ='female']
(9)	/site//description//	任意の深さの'description' の値に"preventions"を持つノードを表示しない

表 4.9: 実験 2 の問い合わせ式とアクセスポリシー

実験結果

実験結果は，表 4.10 から表 4.15　に示す．

式番号	A1[s]	A3[s]
(7)	17.16	1.02
	22.06	1.02
	23.05	1.02
(8)	32.06	1.00
	22.06	1.02
	32.02	1.00
(9)	11.00	10.08
	10.09	10.08
	10.08	10.09

表 4.10: SF:0.01, キャッシュ無し

このデータは，スケールファクター 0.01 のデータを用いて，キャッシュのクリアを 1 回の計測毎に行った結果である．

式番号	A1[s]	A3[s]
(7)	17.05	1.01
	17.07	1.01
	17.07	1.02
(8)	32.03	1.00
	32.01	1.00
	32.02	1.00
(9)	10.08	10.08
	10.07	10.08
	10.07	10.08

表 4.11: SF:0.01, キャッシュ有り

このデータは，スケールファクター 0.01 のデータを用いて一度検索を行った後，連続して検索を行い計測を行った結果である．

式番号	A1[s]	A3[s]
(7)	44.03	2.08
	44.02	5.04
	44.04	5.00
(8)	41.04	1.03
	41.00	1.04
	41.01	1.07
(9)	19.07	4.05
	20.01	4.05
	20.09	4.05

表 4.12: SF:0.03, キャッシュ無し

このデータは，スケールファクター 0.03 のデータを用いて，キャッシュのクリアを 1 回の計測毎に行った結果である．

式番号	A1[s]	A3[s]
(7)	44.05	1.06
	44.04	1.06
	44.06	1.06
(8)	41.00	1.04
	41.01	1.04
	40.09	1.03
(9)	4.01	4.05
	4.01	4.05
	4.02	4.04

表 4.13: SF:0.03, キャッシュ有り

このデータは，スケールファクター 0.03 のデータを用いて一度検索を行った後，連続して検索を行い計測を行った結果である．

式番号	A1[s]	A3[s]
(7)	171.05	2.06
	221.08	2.05
	174.08	2.05
(8)	120.05	1.06
	123.03	1.07
	122.04	1.07
(9)	33.07	7.00
	34.03	7.01
	35.07	6.09

表 4.14: SF:0.05, キャッシュ無し

このデータは，スケールファクター 0.05 のデータを用いて，キャッシュのクリアを 1 回の計測毎に行った結果である．

式番号	A1[s]	A3[s]
(7)	129.07	2.04
	130.00	2.04
	129.09	2.04
(8)	120.06	1.07
	120.05	1.07
	120.09	1.06
(9)	6.06	7.01
	6.04	7.00
	6.04	7.00

表 4.15: SF:0.05, キャッシュ有り

このデータは，スケールファクター 0.05 のデータを用いて一度検索を行った後，連続して検索を行い計測を行った結果である．

考察

(7)，(8) において，A1 のアプローチを用いた SQL の場合，使用する表を 3 回 JOIN し，かつ DEWEYORDER のパターン検索を行った為に処理が遅くなったと考えられる．また（9）に対しては，A1 の場合，値を格納している”VALUE”を全てパターン検索している為，ノード情報のみを判定している A3 と比較して処理時間が長くなっていると考えられる．

4.4 実験3

この実験では，実験 1，2 の結果より具体的な問合せ変形を用いたアクセス制御の手法とノードに付加した情報を用いたアクセス制御の手法を切り替える条件の調査を行った．実験 1 より，どのスケールファクターのデータにおいても，問合せ式番号 (2)，(4) の時に問合せ変形が早くなるという傾向が見られたため，ユーザーからの問合せを 2 つ以上の式にし，各スケールファクターでの手法切り替えの境界を調査する為, 実験を行った．ユーザーからの問合せ式は，”/site/regions”と”/site/people”とし，各データに対して同じアクセスポリシーを設定した．

アクセスポリシー

アクセスポリシーは以下の通りである．

実験結果

実験結果は，表 4.17 から表 4.19 の通りである．
「比率」は, 解に対するアクセス可のノードの割合を表す．

評価

XMark で作成したデータを用いて実験を行った為，解に対するアクセス不可のノードの割合を細かく設定する事は不可能であったが，目安となる比率を実験によって得る事が出来た．この実験より, キャッシュが無い場合でも, 式変形を用いた A1 の手法は, 安定した処理時間で処理を行うことが確認できた．尚, ノード情報を用いた手法のデータに差が出るのは，データベースのキャッシュをクリアしても OS のキャッシュまではクリアされないため，処理時間に影響を与えると思われる．

式番号	アクセスポリシー
(1)	/site/regions//location /site/regions//mailbox /site/regions//name /site/people//name
(2)	/site/regions//name /site/regions/africa /site/regions/asia /site/people/person/profile /site/people/person/watches /site/people/person/creditcard /site/people/person/homepage /site/regions/australia/item/description/parlist/listitem/text /site/regions/samerica/item/description/parlist/listitem/text /site/regions/namerica/item/description/parlist/listitem/text/keyword
(3)	/site/regions//name /site/regions//description /site/people/person/profile/gender /site/people/person/profile/interest /site/people/person/profile/education /site/people/person/watches /site/people/person/creditcard
(4)	/site/regions//mailbox /site/people/person/profile/gender /site/people/person/address /site/regions//description/parlist/listitem /site/regions//shipping
(5)	/site/regions//mailbox /site/people/person/address /site/regions//description/parlist/listitem /site/people/person/profile
(6)	/site/regions//location /site/regions//mailbox /site/regions//name /site/people//name /site/people/person/address /site/regions//description

(1)

式番号	アクセスポリシー
(7)	/site/regions//location /site/regions//mailbox /site/regions//name /site/people/profile /site/people/person/address /site/regions//description /site/regions//quantity /site/regions//shipping /site/regions/australia /site/regions/samerica /site/regions/namerica
(8)	/site/regions//location /site/regions//mailbox /site/regions//name /site/people/profile /site/people/person/address /site/regions//description /site/regions//payment /site/regions//quantity /site/regions//shipping /site/regions/europe /site/regions/australia /site/regions/samerica /site/regions/namerica

表 4.16: 設定したアクセスポリシー (2)

式番号	A1[s]	A3[s]	比率 [%]
(1)	9.02	29.08	
	9.02	17.07	73.2
	9.01	14.00	
(2)	9.01	8.00	
	9.03	8.00	71.0
	9.02	7.09	
(3)	8.09	30.01	
	8.09	8.02	61.8
	8.08	7.07	
(4)	8.08	8.01	
	9.00	7.04	57.0
	8.08	7.02	
(5)	8.08	6.08	
	8.08	8.01	50.1
	8.07	7.04	
(6)	8.07	8.01	
	8.07	6.04	44.2
	8.06	6.05	
(7)	8.05	7.08	
	8.05	10.01	33.9
	8.04	21.07	
(8)	8.03	6.00	
	8.03	20.06	29.7
	8.03	8.06	

表 4.17: スケールファクタ 0.01 における処理時間の計測

解のノード数：8918

式番号	A1[s]	A3[s]	比率 [%]
(1)	6.03	15.08	
	6.02	18.02	73.6
	6.02	5.07	
(2)	6.03	25.03	
	6.03	18.00	68.5
	6.03	15.06	
(3)	5.02	41.05	
	5.03	18.03	61.5
	5.03	20.04	
(4)	5.03	30.00	
	5.02	23.03	57.2
	5.03	10.01	
(5)	4.07	15.08	
	4.07	37.08	50.5
	4.07	11.09	
(6)	4.05	3.07	
	4.05	21.07	44.3
	4.05	07.05	
(7)	4.03	35.01	
	4.02	3.03	33.9
	4.03	11.04	
(8)	4.01	10.05	
	4.01	18.02	29.8
	4.01	7.01	

表 4.18: スケールファクタ 0.03 における処理時間の計測
解のノード数：26923

式番号	A1[s]	A3[s]	比率 [%]
(1)	10.04	28.02	
	9.07	29.03	73.5
	9.09	12.02	
(2)	10.01	24.09	
	10.00	33.06	69.5
	9.09	36.05	
(3)	8.02	9.06	
	8.02	30.06	61.7
	8.04	31.06	
(4)	8.03	27.09	
	8.02	6.08	57.4
	8.03	20.08	
(5)	7.07	26.05	
	7.05	26.05	50.9
	7.04	29.01	
(6)	7.00	31.09	
	7.01	20.03	44.3
	7.00	19.00	
(7)	6.08	7.03	
	6.06	24.09	33.9
	6.06	24.01	
(8)	6.03	22.01	
	6.02	59.04	29.6
	6.02	25.00	

表 4.19: スケールファクタ 0.05 における処理時間の計測
解のノード数：44673

第5章　まとめと今後の課題

本研究を始めるにあたり，アクセス制御の手法の使い分けは，解に対してのアクセス不可の割合を条件と仮定していたが，実際は, データのサイズ，キャッシュの有無，ユーザーからの問合せ式の数により使い分けを行うと，効率的なアクセス制御が可能であることが判明した．しかし，これは，単純パス式によるアクセスポリシーを持つ場合のみ有効で，ノードの値を条件として持つアクセスポリシーである場合は，ノード情報を用いたアクセス制御の手法の方が，効率的にアクセス制御を行う事が可能である．今後の課題としては, 問合せ式変形の自動化が挙げられる．現段階では，XPath 式から SQL に変換する処理を手動で行っている為，この処理を自動化する事と, が必要であると考える．また，更に考慮しなければいけない点については，現段階では，アクセスポリシーを全てのノードに付加しなければならないが，アクセスポリシーを割り振られたノードを root とするサブツリーに対して，子のノードにアクセスポリシーが付加されなくても，サブツリーの root と同じアクセスポリシーを適用できるように改善する事が必要である．

謝辞

この研究を行うにあたり，熱心にご指導，ご鞭撻を頂いた田島敬史先生に心から感謝し，御礼を申し上げます．また，日常において，有益な議論をして頂いた同講座の皆様にも深く感謝致します．

参考文献

[1] A. Sahuguet, B. Alexe, “Sub-Document Queries Over XML with XSQuirrel,” in WWW Conf., 2005.

[2] 北川直毅, 吉川正俊: 「XML 文章のアクセス制御の効率化に関する研究」,DEWS2005

[3] 天笠俊之, 吉川正俊: 「XML データベース技術概説」, オペレーションズ・リサーチ, 第 50 巻, 第 6 号, pp.365-372(2005)

[4] J.Clark, S.DeRose (eds) :”XML Path Language (XPath) Version 1.0. W3C Recommendation,” 1999.

[5] Simple API forXML(SAX) : http://www.saxproject.org/

[6] Document Object Model(DOM) : http://www.w3.org/DOM/

[7] XMark : http://monetdb.cwi.nl/xml/

[8] I. Tatarinov, S.Viglas, K. S. Beyer, J.Shanmugasundaram, E.J. Shekita, and C.Zhang. ”Storing and queryng ordered XML using a relational database system”,In Proc. SIGMOD 2002, pp.204-215,2002