韓國 컴퓨터情報學會 論文誌
第11卷 第5號, 2006. 11.

2006-11-5-1-14

안전한 XML 접근제어에서 효율적인 질의 재작성 기법

안 동 찬*, 변 창 우**

An Effective Query Rewriting Method in Secure XML Access Control

Dong-Chan An*, Chang-Woo Byun**

요 약

XML 데이터베이스에 질의 기반의 접근을 제어하기 위한 효율적인 메카니즘인 2단계 필터링 기법을 제안한다. 문서 단위 접근 범위의 한계를 극복하고 문서의 일부분 단위의 접근을 허용하는 최소 단위의 접근제어 기법 연구에서 안전성에 초점을 둔 연구들은 많았으나 접근제어를 수행할 때의 질의 기반의 효율성에 초점을 둔 연구는 미비하다. 본 논문은 안정성뿐만 아니라 효율성을 고려한 XML 접근제어 수행 메커니즘을 내용으로 하고 있다. 본 논문의 핵심은 사용자의 접근제어 규칙이 아닌 질의 기반의 접근제어 규칙만을 추출하고, 질의 처리기의 부담을 덜어주는 질의 최적화 기법을 통한 접근 제어 정책을 준수하는 새로운 대체 질의로 재작성하는 선처리 방법이다. 본 논문에서 제안하고 있는 방법은 어떠한 XML 데이터베이스 관리 시스템에도 적용가능하며 최소 단위의 접근제어 수행, 낮은 실행시간, 그리고 최적화된 안전하고 정확한 대체 질의 생성을 보장한다. 이와 같은 장점들을 실험을 통해 분석한다.

Abstract

We propose two phase filtering scheme to develop an efficient mechanism for XML databases to control query-based access. An access control environment for XML documents and some techniques to deal with fine-grained authorization priorities and conflict resolution issues are proposed. Despite this, relatively little work has been done to enforce access controls particularly for XML databases in the case of query-based access. The basic idea utilized is that a user query interaction with only necessary access control rules is modified to an alternative form through a query optimization technique, which is guaranteed to have no access violations

• 제1저자 : 안동찬
• 접수일 : 2006.10.16, 심사일 : 2006.10.18, 심사완료일 : 2006.11.18
* 안산공과대학 디지털미디어과 교수 ** 서강대학교 컴퓨터학과 공학박사 수료
※ 이 논문은 2005학년도 안산공과대학 학술연구비에 의하여 연구된 것임.

using tree-aware metadata of XML schemas. The scheme can be applied to any XML database management system and has several advantages such as small execution time overhead, fine-grained controls, and safe and correct query modification. The experimental results clearly demonstrate the efficiency of the approach.

▸ Keyword : XML 데이터베이스(XML database), 데이터 보안(Data Security), 접근제어(Access Control), 최소단위 접근제어(Fine-grained Access Control)

Ⅰ. 서 론

XML 데이터 기반의 정보의 분산과 공유에 대한 요구가 날로 증가함에 따라, 효율적이고 안전하게 XML 데이터에 접근할 수 있는 방법이 매우 중요한 이슈로 부각하고 있다 [1]. 그럼에도 불구하고, 이와 관련된 연구들은 안전성에만 주로 초점을 두어 왔으며 질의에 기반한 효율적인 접근제어를 수행하는 연구는 미비하다. 따라서, 본 논문의 주요 핵심은 질의 기반의 XML 데이터에 대한 안전하고 효율적인 접근제어 수행 메커니즘을 제안하는 것이며, 사용자 질의에 맞는 접근제어 규칙들만 추출하는 방법 및 사용자 질의를 접근제어 정책을 준수하는 최적화된 질의로 변경하는 문제점을 해결하는데 초점을 두고 있다.

사용자 질의에 맞는 접근제어 규칙들의 추출은 사용자의 접근제어 규칙들 중에서 그 사용자의 질의와 관련된 규칙들만 추출할 수 있는 방법을 개발하는 것을 의미한다. 한편, 최적화된 사용자 질의 변경은 추출된 접근제어 규칙들을 이용하여 원래의 사용자 질의를 질의 최적화 기법을 통해 그 사용자의 접근제어 정책을 준수하는 안전하고 정확한 질의로 변경하는 효율적인 질의 재작성 방법을 개발하는 것이다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 관련 연구들과 그들의 단점을 간략히 설명한다. 3장에서는 본 논문의 배경 지식을 설명하고, Document Type Definition (DTD)의 메타데이터를 설명한다. 질의에 해당하는 접근제어 규칙을 추출하는 규칙 필터 및 불안전한 사용자 질의를 안전한 사용자 질의로 바꾸는 질의 필터에 대해 4장에서 설명한다. 5장에서는 여러 실험들을 통해 비교 대상인 Q-Filter보다 성능이 뛰어남을 보이고 최종적으로 6장에서 결론을 맺는다.

Ⅱ. 관련 연구

일반적으로, 현존하는 XML 접근제어 모델은 5-튜플 (주체, 객체, 접근유형, 기호, 타입)로 구성된 접근제어 규칙 명세를 가정한다 [4-13]. 여기서 최소 단위의 접근제어를 수행하는 방법은 객체 부분에 XML 데이터 (혹은 일부분)와 관련되며 XPath를 이용하여 표현함으로써 해결하고 있다 [2]. 접근유형은 연산의 종류를 말하고, 기호는 접근의 허가 (긍정적 접근제어 규칙: +) 혹은 불허 (부정적 접근제어 규칙: -)를 말한다. 타입은 접근의 범위로서 전파허용 (Recursive) 혹은 전파불허 (Local)를 말한다.

전통적인 XML 접근제어 수행 메커니즘[4-13]은 뷰-기반 이행 메커니즘이다. 즉, 질의를 한 사용자와 관련된 접근제어 규칙들에 의해 생성된 문서의 특정 뷰를 기반으로 사용자의 접근을 제어한다. 트리 레이블링 기법으로 뷰를 계산할 수 있는 유용한 알고리즘을 제시하였지만, 뷰 생성의 높은 비용 및 뷰 유지비용의 문제점을 앉고 있다. 또한, 사용자 증가에 따른 확장성 문제를 갖고 있다.

뷰 기반의 문제점을 극복하기 위해 M. Murata et al. 은 접근 제어 정책을 만족하지 못하는 질의를 사전에 걸러낼 수 있는 필터링 방법을 제안하였다 [16]. J. M Jeon et al.은 XML 문서에 대한 모든 접근제어 규칙에 대한 구조 요약 정보에 대한 XML 접근제어 트리 (XACT)를 두어 사용자의 질의를 XACT 접근제어 정보들을 비교하여 변형된 안전한 질의로 변형하는 방법을 제안하였다 [17]. XACT는 질의를 제출한 사용자뿐만 아니라 문서에 대한 모든 접근제어 규칙을 갖고 있기 때문에 불필요한 계산 시간을 갖고 있다. B. Luo et al. 은 [16]의 방법론을 한층 발전시켜 접근제어 정책의 일부분만을 만족하고 있는 질의에 대해 공유 비결정 유한 오토마타 (shared NFA) 방법을 이용해 재작성하는 기법을 제안하였다 [18]. 그러나, 공유 비결정 유한 오토

마타 방법은 사용자의 질의 관점에서 불필요한 접근제어 규칙들에 대한 오토마타까지 포함하고 있어 사용자 질의의 접근허가, 접근불허, 혹은 질의 재작성에 대한 결정을 내리는데 불필요한 시간 낭비를 초래한다. 추가로, 비록 제안된 NFA 기반의 알고리즘은 루트부터 시작하는 경로 질의의 재작성에는 유용하나 "//" 축을 갖고 있는 경로 질의에 대한 재작성에는 비효율적이고 NFA 상의 불필요한 탐색의 문제를 앉고 있다. 또한, 부정확한 재작성된 질의를 만들게 된다.

III. 배경

이번 장에서는 본 논문에서 제안하는 XML 접근제어 방법의 배경 지식과 논문에서 제안하고 있는 가정들 및 기본 용어들에 대해 설명한다.

3.1 XML

XML 문서는 엘리먼트, 속성, 그리고 텍스트 노드로 구성된다. 각 엘리먼트의 내용은 중첩된 엘리먼트들의 일련의 순서 혹은 텍스트 노드이다. 하나의 엘리먼트는 속성 집합을 가질 수 있으며 이들은 이름(name)과 값(value)로 되어 있다. 속성은 엘리먼트에 대한 추가적인 정보를 제공한다.

XML 문서는 잘정형화된(well-formed) 문서와 유효한(valid) 문서, 두 가지 유형이 있다. 유효한 문서는 엘리먼트 혹은 속성 생성의 규칙들을 표현한 문서 유형 정의 (Document Type Definition: DTD) 혹은 XML Schema에 정의된 허용된 타입과 순서 규칙들을 준수한 잘 정형화된 문서를 말한다. 따라서, 유효한 문서 환경에서는 XPath 기반의 질의의 경로 구조(프레디키트는 제외)는 반드시 DTD(혹은 XML Schema)의 구조를 따르게 된다.

[가정 1] *본 논문에서는 DTD 기반의 유효한 XML을 이용한다.*

3.2 XPath와 접근제어

XPath는 XML 문서를 트리 형태로 표현하고 특정 부분을 나타내는 경로 표현 언어이다. 전형적인 경로 표현은 연속적인 스텝(location step)들로 이루어진다 [2]. 각 스텝은 문서 내의 노드 사이의 자식(child), 속성(attribute), 조상(ancestor), 혹은 자손(descendant) 등의 관계를 나타낸다. 경로 상의 어떤 스텝은 또한 선택되는 노드를 걸러주는 필터 역할을 하는 프레디키트(predicate)를 포함하기도 한다. 이와 같은 XPath의 특성을 이용하여 접근 단위의 최소화 요구사항을 만족시키기 위해 XML 문서에 대한 접근을 이행하는 권한부여 모델은 XPath 를 이용한다.

[가정 2] *본 논문에서는 '부모-자식' 관계를 나타내는 '/', '조상-자손' 관계를 나타내는 '//' 과 임의의 엘리먼트를 나타내는 '*' 스텝만을 가정한다.*

3.3. 접근제어 정책

계층적 데이터 모델(예를 들어, Object-Oriented Data Model, XML Data Model 등)에서는 보안 관리자가 명시한 권한부여는 명시적(explicit)이라고 부르며, 명시적 권한부여를 기반으로 시스템이 파생한 권한부여를 묵시적(implicit)이라고 한다 [3]. 저장의 잇점을 얻고자 묵시적 권한부여 방법을 사용하면서 적절하게 '전파 정책(propagation policy)[2]'을 만든다. 여러 환경에 따라 최적의 전파 정책은 다르겠지만 일반적으로 '가장 특정화된 것을 우선으로 하는 정책(most specific precedence)'을 사용한다. 이와 같은 묵시적 권한부여에 의한 전파 정책에 의해 발생될 수 있는 '충돌(conflict) 충돌이라 함은 어떤 노드가 '접근허용' 및 '접근불가' 규칙이 동시에 정의되어 있는 경우를 말한다.' 문제를 해결하는 정책으로는 '거절 우선 정책(denial take precedence)'을 일반적으로 사용한다. 또한, 긍정적(positive) 권한부여와 부정적(negative) 권한부여를 혼합하여 사용하기 때문에 명시적인 권한부여가 없는 노드에 대한 '결정 정책(decision policy)'로 명시적 권한부여가 없는 노드는 접근을허용하는 '개방(open) 정책'과 접근을 불허하는 '폐쇄(closed) 정책'을 사용한다.

일반적으로, 엄격한 데이터 보안을 위해 '거절 우선 정책'과 '폐쇄 정책'을 사용한다.

[가정 3] *본 논문에서의 접근제어 정책은 '묵시적 권한부여' 기반의 '가장 특정화된 것을 우선으로 하는 정책'을 사용하며 충돌 해결 정책으로 '거절 우선 정책' 및 긍정*

2) 한 노드에 대한 접근제어 결과가 그 노드의 후손에 전파되는지 여부를 결정하는 정책을 말한다.

적/부정적 권한부여 혼용을 위해 '폐쇄 정책'을 사용한다.

3.4 문제 정의

본 논문에서 제안하고 있는 XML 접근제어 수행 메커니즘은 질의자가 임의의 XML 문서에 대해 질의를 하였을 때, 그 질의자의 접근제어 규칙들을 살펴 사용자의 질의를 접근제어 정책에 어긋나지 않는 최적화된 질의로 변형하여 질의 처리기에서 변형된 질의를 넘겨주는 작업을 수행한다. 따라서, 긍정적 접근제어 규칙(ACR+)과 부정적 접근제어 규칙(ACR-)이 존재했을 때, 사용자 질의(Q)에 따른 변형된 질의(Q')는 다음과 같은 수식으로 정리할 수 있다.

Q' = Q INTERSECT (ACR+ EXCEPT ACR-)
= (Q INTERSECT ACR+) EXCEPT
(Q INTERSECT ACR-)

XPath 2.0에서는 집합 연산 (즉, UNION, INTERSECT, 그리고 EXCEPT)을 지원한다 [2]. 비록 이들 연산이 기술적으로 경로 표현식은 아니지만, 경로 표현식들 사이에 일정한 방식으로 사용된다. 본 논문에서는 이들 집합 연산을 이용하여 불안전한 사용자 질의를 안전한 질의로 변형하는데 유용하게 이용하고자 한다. XML spy 2006 버전에 이들 집합 연산이 지원되고 있음을 확인하였다 (http://www.altova.com).

이런 환경에서 본 논문에서 제기하는 문제점은 다음 두 가지이다:

① 첫째, 사용자에 대한많은 접근제어 규칙들 중 질의와 관련된 접근제어 규칙만을 추출하는 방법

② 둘째, '*' 및 '//'을 포함하고 있는 사용자 질의 및 접근제어 규칙을 비교하여 최적화된 효율적이고 정확한 XPath 경로 표현식(변형된 질의)을 생성하는 방법

3.5 PRE/POST 구조 (PRE/POST Structure)

본 논문에서 사용하는 DTD 메타데이터를 설명하기에 앞서 기억해야 할 용어 하나를 설명한다.

XPath 기반의 질의는 XPath 경로 표현식에서 가장 마지막 노드의 하부-트리(sub-trees)를 그 결과로 나타낸다 [18,19]. 특히, [18]에서는 이를 '하부-트리 결과(answer-as-subtrees)'라 정의하고 있다.

정의 1. [목적 노드(target node)]: XML 질의 언어 혹은 XML 접근제어 규칙에 사용된 XPath 경로 표현식의 목적 노드는 프레디키트(predicate)를 제외한 경로 표현식의 가장 마지막 노드이다.

예를 들어, '/site/people/person' 의 목적 노드는 person 노드이다. 또 다른 예로, '/site/regions/aisa/item[@id = "xxx"]' 의 목적 노드는 item 노드이다.

그림 1(a)는 XMark로부터 인용된 앞으로 논문에서 예제로 사용하는 auction.dtd를 보여주고 있다 [20]. 그림 1(b)는 DTD를 순차적으로 접근하여 얻은 PRE(order) 그리고 POST(order) 값이 부여된 DTD 트리를 나타낸다. 트리 구조에서 preorder와 postorder 값을 기술하는 방법은 XML 문서를 트리 구조로 만들어 XML 질의 언어를 위한 질의 처리기에서 유용하게 사용하는 방법으로 본 논문에서는 이 방법을 DTD 트리에 인용하고 있으며 이를 PRE/POST 구조 (PRE/POST Structure: PPS)라 한다. 그림 1(c)는 DTD 트리에 대한 관계형 저장 구조 (실제는 해쉬 테이블로 구현함)를 나타내고 있다. LEVEL은 DTD 트리에서의 레벨을 의미하며, SIZE는 임의의 트리 노드에서의 하부-트리의 노드 수를 나타낸다. POST 값은 다음과 같은 수식을 통해 얻는다 [14,15]:

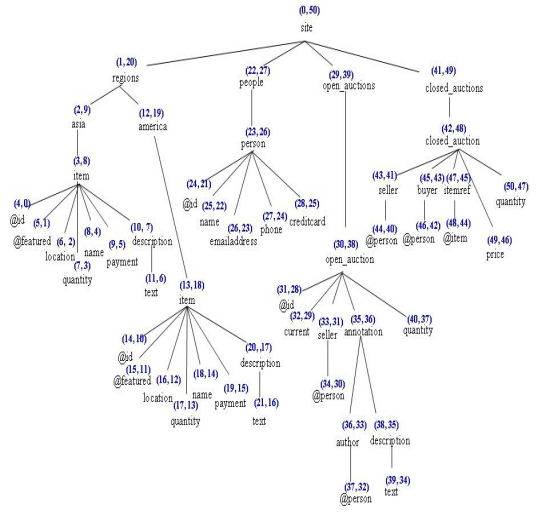
POST = PRE + SIZE - LEVEL

PRE(order)와 POST(order)는 DTD 트리에 대해 한번의 스캔 작업으로 얻게 된다. PARENT는 임의의 노드에서의 부모 노드를 나타낸다.

```
<!ELEMENT site                (regions, people, open_auctions, closed_auctions)>
<!ELEMENT regions             (asia, america)>
<!ELEMENT asia                (item*)>
<!ELEMENT america             (item*)>
<!ELEMENT item                (location, quantity, name, payment, description)>
<!ATTLIST item                id ID #REQUIRED
                              featured CDATA #IMPLIED>
<!ELEMENT location            (#PCDATA)>
<!ELEMENT quantity            (#PCDATA)>
<!ELEMENT name                (#PCDATA)>
<!ELEMENT payment             (#PCDATA)>
<!ELEMENT description         (text)>
<!ELEMENT text                (#PCDATA)>
<!ELEMENT people              (person*)>
<!ELEMENT person              (name, emailaddress, phone?, creditcard?)>
<!ATTLIST person              id ID #REQUIRED>
<!ELEMENT emailaddress        (#PCDATA)>
<!ELEMENT phone               (#PCDATA)>
<!ELEMENT creditcard          (#PCDATA)>
<!ELEMENT open_auctions       (open_auction*)>
<!ELEMENT open_auction        (current, seller, annotation, quantity)>
<!ATTLIST open_auction        id ID #REQUIRED>
<!ELEMENT current             (#PCDATA)>
<!ELEMENT seller              EMPTY>
<!ATTLIST seller              person IDREF #REQUIRED>
<!ELEMENT annotation          (author, description?)>
<!ELEMENT author              EMPTY>
<!ATTLIST author              person IDREF #REQUIRED>

<!ELEMENT closed_auctions     (closed_auction*)>
<!ELEMENT closed_auction      (seller, buyer, itemref, price, quantity)>
<!ELEMENT buyer               EMPTY>
<!ATTLIST buyer               person IDREF #REQUIRED>
<!ELEMENT itemref             EMPTY>
<!ATTLIST itemref             item IDREF #REQUIRED>
<!ELEMENT price               (#PCDATA)>
```

(a)



(b)

Tag-Name	PRE	SIZE	LEVEL	POST	PARENT
site	0	50	0	50	
regions	1	20	1	20	site
asia	2	9	2	9	regions
item	3	8	3	8	asia
@id	4	0	4	0	item
@featured	5	0	4	1	item
location	6	0	4	2	item
quantity	7	0	4	3	item
name	8	0	4	4	item
payment	9	0	4	5	iem
description	10	1	4	7	item
text	11	0	5	6	description
...	...	...	...	...	...
people	22	6	1	27	site
person	23	5	2	26	people
@id	24	0	3	21	person
name	25	0	3	22	person
emailaddress	26	0	3	23	person
phone	27	0	3	24	person
creditcard	28	0	3	25	person

Tag-Name	PRE	SIZE	LEVEL	POST	PARENT
open_auctions	29	11	1	39	site
open_auction	30	10	2	38	open_auctions
@id	31	0	3	28	open_auction
current	32	0	3	29	open_auction
seller	33	1	3	31	open_auction
@person	34	0	4	30	seller
annotation	35	4	3	36	open_auction
author	36	1	4	33	annotation
@person	37	0	5	32	author
description	38	1	4	35	annotation
text	39	0	5	34	description
quantity	40	0	3	37	open_auction
closed_auctions	41	9	1	49	site
closed_auction	42	8	2	48	closed_auctions
seller	43	1	3	41	closed_auction
@person	44	0	4	40	seller
buyer	45	1	3	43	closed_auction
...	...	...	...	...	...
quantity	50	0	3	47	closed_auction

(c)

그림 1: (a) auction.dtd, (b) auction.dtd의 PRE/POST 구조, (c) PRE/POST 구조의 관계형 저장 형태
Fig 1: (a) auction.dtd (b) DTD PRE/POST Structure of (a), (c) Relational Storage of (b)

Ⅳ. 2단계 필터링 기법

2단계 필터링의 목적은 사용자 질의를 처리하기 위해 필요한 접근제어 규칙만을 추출하고, 불안전한 사용자의 질의를 안전한 질의로 재작성하는 것이다. 각 필터링에 대해 자세히 설명하기 전에 사용자 질의와 접근제어 규칙에서 추출되는 정보에 대해 설명한다.

Positive ACRs

(R1): /site/regions/*/item[location="LA"]
(R2): /site/people/person[name = "chang"]
(R3): /site/open_auctions/open_auction
(R4): //open_auction[quantity]/seller

Negative ACRs

(R5): /site/regions/*/item/payment
(R6): /site/people/person/creditcard
(R7): /site/*/open_auction[@id>50]/seller[@person="chang"]

(a)

ACR+-base

rule	path	Pre	Post	P_link
R1	/site/regions/*/item	3, 13	8, 18	P1
R2	/site/people/person	23	26	P2
R3	/site/open_auctions/open_auction	30	38	
R4	//open_auction[quantity]/seller	33	31	P3

ACR--base

rule	path	Pre	Post	P_link
R5	/site/regions/*/item/payment	9, 19	5, 15	
R6	/site/people/person/creditcard	28	25	
R7	/site/*/open_auction[@id>50]/seller[@person="chang"]	33	31	P4

PREDICATES-base

P-id	Parent-PRE	property	operator	value
P1	3, 13	location	=	LA
P2	23	name	=	chang
P3	33	quantity		
p4	33	@person	=	chang

(b)

그림 2: (a) 긍정적/부정적 접근제어 규칙 예제 (b) 접근제어 규칙에 대한 데이터베이스
Fig 2: (a) Sample positive/negative ACRs (b) Sample ACR and PREDICATES databases

일단 보안 관리자 (혹은 정책 관리자)에 의해 그림 2(a)와 같은 사용자 (혹은 역할)별 접근제어 규칙들이 만들어지면, XPath 경로 표현식에서 경로에 대한 정보와 프레디키트에 대한 부분을 그림 2(b)처럼 데이터베이스화 한다.

한편, 사용자 질의로부터 정보(목적 노드의 (PRE, POST) 값, 그리고 프레디키트 정보)를 얻는다.

Q1:

/site/people/person[name="chang"]/phone/

사용자 질의 Q1에 대해 일단 목적 노드 phone을 확인한다. 그런 후, QA는 PRE/POST 구조로부터 목적 노드 phone에 대한 (PRE, POST) 쌍인 (27, 24)를 얻는다. 또한, Q1의 프레디키트 정보를 얻는다. 질의에 따라 (PRE, POST) 쌍은 집합이 될 수도 있다.

사용자 질의가 다음과 같다고 가정하자.

/site/regions/america/item

목적 노드는 item이고, PPS를 통해 (3, 8)과 (13, 18)을 얻게 된다. 만약 america 노드 대신 "*" 노드였

다면 둘 다 질의에 대한 목적 노드의 (PRE, POST) 쌍이 되겠지만, (13, 18)만이 america 노드의 자식 노드의 (PRE, POST) 쌍이 된다. 즉, 목적 노드의 (PRE, POST) 쌍이 집합인 경우, 불필요한 (PRE, POST) 쌍을 제거하는 핵심 개념은 질의의 각 노드의 preorder (postorder) 값은 목적 노드의 preorder(postorder) 값보다 작아야(커야) 한다. 그림 3은 이에 대한 알고리즘을 보여준다.

Input: a user query
Output: suitable (PRE, POST) values of target node of the query
BEGIN
for each (Pr_{tn}, Po_{tn}) value of the target node of the query
{
for (Pr_{step}, Po_{step}) value of each node of the query
if (!($Pr_{step} < Pr_{tn}$ and $Po_{step} > Po_{tn}$))
break;
suitable (PRE, POST) set ← (Pr_{tn}, Po_{tn})
}
END

그림 3. Prune-TNs 알고리즘
Fig 3. The Prune-TNs Algorithm

4.1 일단계 필터링: 규칙 필터

규칙 필터의 목적은 사용자의 접근제어 규칙들 중 질의와 관련된 접근제어 규칙들만을 추출하는 것이다. 사용자 질의 Q1을 다시 예를 들면, 목적 노드 phone에 대한 (27, 24) 값을 얻는다. 그림 7(a)에서 보듯이, 접근제어 규칙들에 대한 PRE/POST 이차평면에 목적 노드 (27, 24) 점을 기준으로 네 개의 사분면이 형성된다.

각 사분면은 질의와 접근제어 규칙 간의 특별한 관계를 의미한다. 이들 관계를 XPath의 목적 노드에 대한 (PRE, POST) 쌍의 값 비교로 구분된다.

정의 2. [1사분면: FOLLOWING 규칙] 임의의 XML 문서에 대해 접근제어 규칙에 표현된 XPath에 의한 영역은 질의 XPath의 영역에 대해 FOLLOWING 영역으로써, 이와 같은 접근제어 규칙을 질의에 대한 FOLLOWING 규칙이라 정의한다.

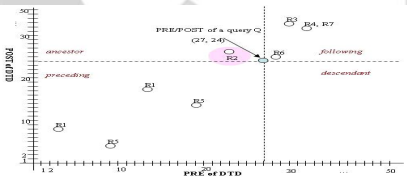
정의 3. [2사분면: ANCESTOR 규칙] 임의의 XML 문서에 대해 접근제어 규칙에 표현된 XPath에 의한 영역이 질의 XPath의 영역에 대해 ANCESTOR 영역(PARENT도 포함)으로써, 이와 같은 접근제어 규칙을 질의에 대한 ANCESTOR 규칙이라 정의한다.

정의 4. [3사분면: PRECEDING 규칙] 임의의 XML 문서에 대해 접근제어 규칙에 표현된 XPath에 의한 영역이 질의 XPath의 영역에 대해 PRECEDING 영역으로써, 이와 같은 접근제어 규칙을 질의에 대한 PRECEDING 규칙이라 정의한다.

정의 5. [4사분면: DESCENDANT 규칙] 임의의 XML 문서에 대해 접근제어 규칙에 표현된 XPath에 의한 영역이 질의 XPath의 영역에 대해 DESCENDANT 영역(CHILD도 포함)으로써, 이와 같은 접근제어 규칙을 질의에 대한 DESCEDANT 규칙이라 정의한다.

정의 6. [SELF 규칙] 임의의 XML 문서에 대해 접근제어 규칙에 표현된 XPath에 의한 영역이 질의 XPath의 영역에 대해 SELF 영역으로써, 이와 같은 접근제어 규칙을 질의에 대한 SELF 규칙이라 정의한다.

여기서 질의와 관련된 사분면은 2사분면(ANCESTOR 규칙)과 4사분면(DESCENDANT 규칙)만이고 추가로 SELF 규칙도 질의와 관련된 접근제어 규칙이다. 즉, 예제에서는 R1에서 R7 중에 R2 만이 질의와 관련된 접근제어 규칙이 된다. 그림 4(b)는 이와 같이 사용자 질의에 대한 ANCESTOR 규칙, DESCENDANT 규칙 혹은 SELF 규칙에 해당되는 접근제어 규칙을 찾는 알고리즘이다.



(a)

Input: (Pr_Q, Po_Q) := QA(query), ACRs
Output: suitable ACRs'
BEGIN
for each rule R1 in ACRs
if ((Pr_Q Pr_{R1} and Po_Q Po_{R1}) *or*
(Pr_Q Pr_{R1} and Po_Q Po_{R1}))
ACRs' ← R1
END

(b)

그림 4. (a) 접근제어 규칙에 대한 PRE/POST 이차평면의 의미 (b) 규칙 필터 알고리즘
Fig 4: (a) Semantics of PRE/POST Plane of Positive ACRs (b) The *Rule Filter* Algorithm

4.2 이단계 필터링: 질의 필터

질의 필터의 목적은 사용자 질의와 그 질의와 관련된 접근제어 규칙들과 XPath 2.0의 집합 연산을 통해 최적화된 안전한 질의로 변경하는 것이다.

규칙 필터를 통과한 사용자 질의는 긍정적 접근제어 규칙과 부정적 접근제어 규칙 이렇게 두 가지 그룹으로 나뉘고 각 그룹은 SELF, ANCESTOR, 그리고 DESCENDANT 접근제어 규칙 이렇게 세 가지의 규칙으로 나뉜다. QE의 처리 과정은 다음과 같다:

1. 사용자 질의와 각각의 부정적 접근제어 규칙 간의 교집합 질의를 만든다.
2. UNION 연산자를 통해 이들 교집합 질의를 합친다.
3. 사용자 질의와 각각의 긍정적 접근제어 규칙 간의 교집합 질의를 만든다.
4. UNION 연산자를 통해 이들 교집합 질의를 합친다.
5. EXCEPT 연산을 통해 4번의 결과에 2번의 결과를 합친다.
6. 5번의 결과 질의를 질의 처리기에 보낸다.

4.2.1 경로표현 최적화 기법

사용자의 질의 혹은 접근제어 규칙의 경로 표현식에 나타나는 '*' 노드를 실제 노드로 바꾸고 "//" 축을 일련의 "/" 축으로 이루어진 단일 경로로 바꿀 수 있다면 정확한 질의로의 변경을 이룰 수 있다.

예를 들어, 변경되는 질의가 다음과 같다고 하자:

Q2: //open_auction[@id<100]

질의 Q2의 목적 노드 open_auction의 (PRE, POST) 값 (30, 38)에 대한 긍정적 접근제어 규칙 R4 (33,31)과 부정적 접근제어 규칙 R7 (33, 31)이 추출되고 이들을 적절히 조절하고, PPS에 의해 부모 노드를 찾으면 다음과 같은 질의로 바뀐다.

긍정적 질의 Q2':

/site/open_auctions/open_auction[quantity][@id<100]/ seller

부정적 질의 Q2':

(/site/open_auactions/open_auction[@id>50]/seller[@p erson="chang"]).

사용자가 질의에서 요구하는 엘리먼트 간의 계층 관계를 해석하는 연산을 구조적 조인(structural join)이라고 정의하는데, XML 질의 처리기 입장에서는 계층 관계가 중첩되어 있는 트리 구조의 XML 질의를 처리하려면 여전히 다수의 구조적 조인을 수행하여야 하기 때문에 질의 처리 비용이 많이 드는 또 다른 문제를 갖게 된다.

긍정적 질의 Q2'이나 부정적 질의 Q2'은 연속된 노드 쌍들이 부모-자식 관계를 이루고 있으므로 각각 3번의 구조적 조인이 필요하다.

XPath 질의나 트리 패턴의 최소화에 대한 연구들이 있어 왔다. XPath 질의의 최소화는 Wood [8]에 의해 최초로 연구되었다. 이 논문에서는 조상-후손 관계를 갖고 있지 않은 선형 XPath 질의와 DTD를 고려하였다. 제한된 조건의 XPath(XP{/, [], *}에 대한 질의 최소화의 복잡도가polynomial 임을 보여주었다. 또한 이와 다른 제한 조건의 XPath(XP{/, [], //})에 대해서도 polynomial 시간에 최소화 될 수 있다는 것이 연구되었다[9]. 그 후 좀더 일반적인 조건(XP{/, //, [], *})에 대한 XPath 질의 최소화에 대한 문제가 다루어졌다[11].

본 논문은 DTD를 이용하여 안전한 XML 문서를 처리하기 위한 질의 최적화를 목적으로 한다는 점이 위와 같은 연구들과의 차이점이다.

XPath 질의의 최적화에 유용하게 사용될 수 있는 DTD의 제약사항 정보에는 다음과 같은 것들이 있다.

① **자식 제약사항**: 어떤 엘리먼트가 다른 엘리먼트를 자식으로 항상 가진다면 이 두 엘리먼트 사이에는 자식 제약이 존재한다.

② **후손 제약사항**: 어떤 엘리먼트가 다른 엘리먼트를 후손으로 항상 가진다면 이 두 엘리먼트 사이에는 후손 제약이 존재한다.

③ **형제 제약사항**: 어떤 엘리먼트가 두 개의 자식 엘리먼트를 가지고 이 두 엘리먼트가 항상 나타나야 한다면, 이 두 자식 엘리먼트 사이에 이웃 제약이 존재한다.

④ **배타적 제약사항**: 어떤 엘리먼트가 다른 엘리먼트와 배타적으로 나타나야 한다면, 즉 두 엘리먼트가 동시에 나타날 수 없고, 적어도 둘 중 한 엘리먼트는 항상 나타나야 한다면, 이 두 엘리먼트 사이에 배타적 제약이 존재한다.

따라서, site 노드와 open_auctions노드는 자식 제약사항이고, open_auction은 후손 제약사항이다. open_auction과 quantity 노드는 자식 제약사항이다. 그래서 최적화된 질의로 변경된 질의는 다음과 같다:

긍정적 질의 Q2′:

//open_auction[@id<100]/seller

부정적 질의 Q2′:

(//open_auction[@id>50]/seller[@person="chang"])

4.2.2 프레디키트 최적화 기법

XML 인스턴스 정보를 모르는 상황에서 질의 재작성 과정에서 원래 질의의 프레디키트 정보와 접근제어 규칙의 프레디키트 정보를 변경되는 질의에 유지되어야 한다. 접근제어 규칙에 포함된 프레디키트의 위치 정보는 그림 2의 PREDICATES 데이터베이스에 저장되어 있고, 사용자 질의의 프레디키트는 질의 정보 추출 과정에서 얻게 된다. 최종 변형된 질의에 접근제어 규칙의 프레디키트를 추가할 때, 같은 경로 위치에 사용자의 프레디키트와 접근제어 프레디키트가 존재하는 경우 최적화를 고려해야 한다. 특히, 부정적 접근제어 규칙의 프레디키트와 긍정적 접근제어 규칙의 프레디키트를 다르게 처리해야 하는데 초점을 두고 있다.

질의와 긍정적 접근제어 규칙의 프레디키트를 서로 비교하여 테이블 1처럼 결과가 나오게 하고, 질의와 부정적 접근제어 규칙은 부정적 접근제어 규칙만을 그대로 유지하게 하고 있다. 그 이유는 부정적 접근제어 규칙은 긍정적 접근제어 규칙의 범위 내에 존재하고, 그것은 바로 질의와 긍정적 접근제어의 프레디키트 최적화에 포함되어 있기 때문에 부정적 접근제어 규칙에 다시 적용할 필요가 없다.

표 1. 완성된 프레디키트 최적화 예제
Table 1. Examples of Predicates Optimizations

질의의 프레디키트	부정적 접근제어의 프레디키트	긍정적 접근제어의 프레디키트	최적화
[@id = "1"]	[@id = "1"]		[@id = "1"]
[@id = "1"]	[@id = "3"]		reject
[@id > "1"]	[@id > "3"]		[@id > "3"]
[@id > "1"]	[@id < "3"]		[@id > "1" and @id < "3"]
[@id = "1"]		[@id = "1"]	[@id = "1"]
[@id = "1"]		[@id = "3"]	[@id = "3"]
[@id > "1"]		[@id > "3"]	[@id > "3"]
[@id > "1"]		[@id < "3"]	[@id < "3"]

4.2.3 질의 필터 수행 과정

질의 필터의 수행과정을 살펴본다. 부적절한 질의를 사전에 거절시키는 방법이 전체적으로 빠른 성능을 내기 때문에 본 논문에서는 질의와 부정적 접근제어를 먼저 처리한다.

단계 1. (부정적 접근제어 처리)

경우 1.1 (SELF 접근제어 규칙). SELF 규칙의 의미는 질의의 목적 노드 (PRE, POST) 쌍과 접근제어 규칙의 목적 노드의 (PRE, POST) 쌍이 같은 경우이다. 만약 사용자 질의에 대해 부정적 접근제어 규칙이 SELF 규칙이라면, 질의의 결과는 질의 거절이다. 그러나, 만약 부정적 접근제어 규칙에 프리티키트가 존재한다면, 같은 경로들 중 프리티키트에 해당되는 경로만 접근 금지이기 때문에 질의는 거절이 되지 않고 최적화 작업을 통해 질의를 재구성한다.

경우 1.2 (ANCESTOR 규칙). 만약 사용자 질의에 대한 부정적 접근제어 규칙이 ANCESTOR 규칙이라면, 질의 필터의 결과는 질의 거절이다. 왜냐하면 부정적 접근제어의 범위에 사용자 질의의 범위가 포함되기 때문이다. 그러나, 경우 1.1처럼 부정적 접근제어 규칙에 프레디키트가 존재하면 최적화 작업을 통해 질의를 재구성한다.

경우 1.3 (DESCENDANT 규칙). 만약 사용자 질의에 대한 부정적 접근제어 규칙이 DESCENDANT 규칙이라면, 질의 필터는 최적화 기법을 통해 질의를 재구성한다. 예를 들어, 질의 Q2 (30, 38)에 대한 R7 (33, 31)는 DESCENDANT 규칙이므로 질의 필터는 질의 최적화 기법을 통해 다음과 같이 변경한다.

Q2′: Q2 EXCEPT

(//open_auction[@id>50]/seller[@person="chang"])

그런 후, 긍정적 접근제어 규칙 R4를 고려하기 위해 단계 2로 간다.

경우 1.4 (Null). 질의와 관련된 부정적 접근제어 규칙이 없으면 질의 필터는 단계 2로 간다.

단계 2. (긍정적 접근제어 처리)

경우 2.1 (SELF 규칙). 만약 질의에 대한 긍정적 접근제어 규칙이 SELF 규칙이라면, 질의 필터는 질의 최적화 기법을 통해 재작성한다.

경우 2.2 (ANCESTOR 규칙). 만약 질의에 대한 긍정적 접근제어 규칙이 ANCESTOR 규칙이라면, 질의

필터는 질의 최적화 기법을 수행한다.

경우 2.3 (DESCENDANT 규칙). 만약 질의에 대한 긍정적 접근제어 규칙이 DESCEDANT 규칙이라면, 사용자 질의는 접근이 허용되지 않는 범위를 포함하게 된다. 따라서, 질의 최적화 기법을 통해 안전한 질의로 변경된다.

예를 들어, 질의 Q2 (30, 38)에 대한 R4 (33, 31)는 DESCENDANT 접근제어 규칙이기에 질의 최적화 기법을 통해//open_auction[@id<100]/seller의 결과를 얻는다. 둘째, 질의 필터는 경우 1.3에서 얻어진 결과를 EXCEPT 연산자를 이용하여 불필요한 부분을 제거한다. 따라서, 최종적으로 변경된 안전한 질의는 다음과 같다:

```
//open_auction[@id<100]/seller
 EXCEPT
(//open_auction[@id>50]/seller[@person="chang"])
```

V. 실험

Q-Filter [18], 2PF(이단계 필터링 기법), 그리고 2PF_NFA(이단계 필터링 기법에 오토마타 적용)을 자바 프로그램 언어를 이용하여 구현하였다(XENON 3.0GHz, 4GB RAM, MS-Windows Server 2003). XMark에 의해 생성된 실험적 데이터 집합 25 개의 접근제어 규칙 (7 개의 긍정적 접근제어 규칙과 18개의 부정적 접근제어 규칙)을 이용하여 두 가지 관점에서 성능을 비교해 보았다 [20].

5.1 실험 I: 거절 질의에 대한 정확한 탐지

거절 질의(Rejection Query)라 함은 항상 거절이 되어야 하는 사용자 질의를 말한다. 접근제어 정책에 의해 각각의 질의 유형에 맞는 20개의 의도된 거절 질의를 만들고 실제 시스템에서 거절 질의의 탐지 개수를 비교해 보았다. 그 결과는 그림 5와 같다. 여기서 "/" 축으로만 되어 있는 질의인 경우는 어느 시스템이나 100% 거절 질의를 탐지하지만, "//" 축으로 되어 있는 질의인 경우 제안하는 기법은 완전히 거절 질의를 탐지하지만, QFilter는 그렇지 않음을 확인하였다. 그 원인은 "//" 축에 대한 경로 항해의 끝이 없기 때문에 질의를 거절하지 못하고 잘못된 질의로 재작성하기 때문이다.

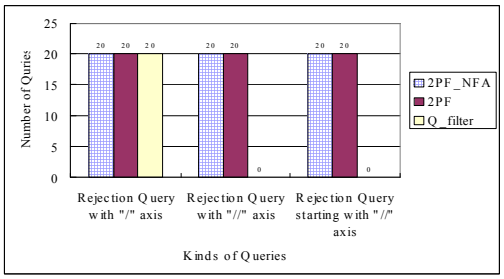


그림 5. 거절 질의 탐지에 대한 결과
Fig. 5 The number of rejecting prohibited queries corresponding to various query types

5.2 실험 II: 수행시간

무작위로 추출한 질의 개수(30, 50, 100, 200, 300, 그리고 500)에 따른 평균 수행시간을 측정하였다.

3.5절의 PPS 정보 및 4.1절의 규칙 필터 컴포넌트에 의해 제안하는 기법은 매우 적은 양의 접근제어 규칙을 이용하여 보다 정확하고 안전한 질의 재작성 과정을 수행한다. 그림 6은 그 결과를 보여주고 있으며 각 수행시간은 각각의 메타데이터를 생성하는 예비조건 생성 시간을 포함하고 있다.

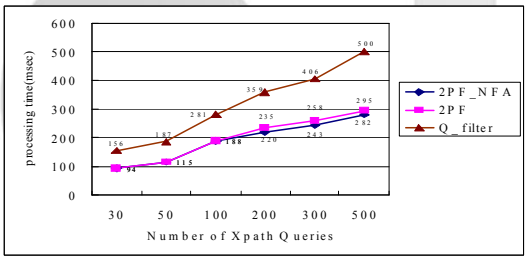


그림 6. 질의에 대한 보안 처리 수행 시간
Fig 6. The Processing time of the security check on XPath queries

5.3 질의 처리기 부담

XMark에 의해 생성된 1G 바이트 크기의 XML 문서를 Berkeley 데이터베이스에 저장하고 XML 문서로부터 300여 개의 XPath 질의문들(중첩 경로가 5에서 9 사이)을 추출하여 동등한 최소화된 질의문들을 생성하였다 [20]. 중첩 경로가 6일 때에는 1/3까지 줄어들었고, 평균적으로 1/2 이하로 줄어들어 최소화의 효과를 확인할 수 있다.

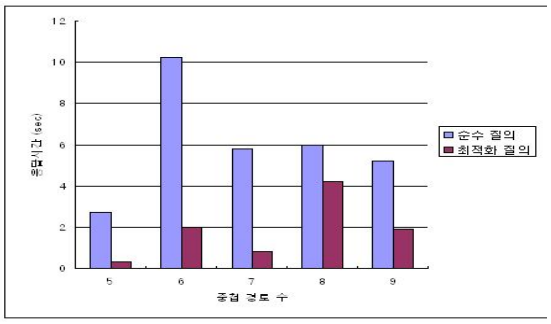


그림 7. 질의 최적화의 응답 시간
Fig 7. The Response time of optimized queries

Ⅵ. 결론 및 향후 과제

본 논문에서 기술한 효율적이고 안전한 XML 접근제어 수행 메커니즘 방법은 사용자의 질의 처리에 필요한 접근제어 규칙들만을 추출하기 위해 PRE/POST 트리 특성을 이용하였다. 특히 PRE/POST 인코딩 기법은 질의 재작성을 하기 전에 사전에 접근 권한이 없는 질의를 거절시키는데 효율적으로 이용된다. 추가로 DTD에 대한 메타데이터 및 제약사항들을 통해 최적화된 질의 재작성 및 XPath 2.0에서 지원하는 집합 연산을 이용하여 사용접근제어 정책을 준수하는 최종적인 질의로 변경하여 질의 처리기의 부담을 최소화하였다. 또한, 실험을 통해 제안하는 방법의 우수성을 보였다.

추후 연구로는 접근 권한 모델(강제적 접근 권한 모델, 임의적 접근 권한 모델, 역할 기반 접근제어 모델)과 조합된 XML 접근제어 시스템 개발을 진행하고 객체-지향 데이터베이스 관리 시스템의 접근제어 시스템에 적용하고자 한다.

참고문헌

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau. Extensible Markup Language (XML) 1.0, World Wide Web Consortium (W3C), 2004. (http://www.w3.org/TR/REC-xml)

[2] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon. XPath 2.0, World Wide Web Consortium (W3C),2005. (http://www.w3.org/TR/xpath20/)

[3] F. Rabitti, E. Bertino, W. Kim and D. Woelk.A Model of Authorization for Next-Generation Database Systems. ACM Transaction on Database Systems, Vol 126, No 1. March 1991, PP. 88-131.

[4] E .Damiani, S.Vimercati, S.Paraboschi, and P.Samarati. Securing xml document. In Proc. of the 2000 International Conf.on Extending Database Technology (EDBT2000), Konstan, Germany, March, 2000, pp.121-135.

[5] E. Damiani, S. Vimercati, S. Paraboachk and P.Samarati.XML Access Control Systems: A Component-Based Approach.In Proc. IFIP WG11.3 Working Conference on Database Security, The Netherlands, 2000. 8.

[6] E. Damiani, S. Vimercati, S. Paraboachk and P.Samarati. Design and Implementation of Access Control Processor for XML Documents. Computer Network, 2000.

[7] E. Damiani, S. Vimercati, S. Paraboachk and P.Samarati.A Fine-grained Access Control System for XML Documents.ACM Trans. Information and System Sec., Vol.5, No.2, May 2002.

[8] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and Enforcing Access Control Policies for XML Document Sources. WWW Journal, Baltzer Science Publishers, Vol.3, N.3, 2000.

[9] E. Bertino, S. Castano, and E. Ferrai. Securing XML documents with Author-x. IEEE Internet Computing, May.June, pp.21-31, 2001.

[10] E. Bertino and E. Ferrari. Secure and Selective Dissemination of XML Documents. TISSEC, 5(3), pp. 237-260, 2002.

[11] E. Bertino, M. Braun, S. Castano, E. Ferrari, and M. Mesiti. Author-X: A Java-Based System for XML Data Protection. In Proc. IFIP WG11.3 Working Conference on Database

Security, Netherlands, 2002. 8.
[12] A. Gabillon and E. Bruno. Regulating Access to XML Documents. In Proc. IFIP WG11.3 Working Conference on Database Security, 2001.
[13] A. Stoica and C. Farkas. Secure XML Views.In Proc. IFIP WG11.3 Working Conference on Database and Application Security, 2002.
[14] T. Grust. Accelerating XPath Location Steps.In Proc. of the 21st Int'l ACM SIGMOD Conf. on Management of Data, pages 109-120, Madison, Wisconsin, USA, June 2002.
[15] T. Grust, M. van Keulen, and J. Teubner. Staircase Join: Teach a Relational DBMS to Watch its Axis Steps. In Proc. of the 29th International Conf. on VLDB, Berlin, Germany, September 2003.
[16] M. Murata, A. Tozawa, and M. Kudo. XML Access Control using Static Analysis. In ACM CCS, Washington D.C., 2003.
[17] Jae-Myeong Jeon, Yon Dohn Chung, Myoung Ho Kim, and Yoon Joon Lee, Filtering XPath expressions for XML access control. Computers & Security, 23, pp.591-605, 2004.
[18] B. Luo, D. W. Lee, W. C. Lee, and P. Liu. Qfilter: Fine-grained Run-Time XML Access Control via NFA-based Query Rewriting. In Proc. of the Thirteenth ACM Conference on Information and Knowledge Management 2004(CIKM'04), November 8, 2004, Washington, USA.
[19] S. Mohan, A. Sengupta, Y. Wu, and J. Klinginsmith. Access Control for XML- A Dynamic Query Rewriting Approach. In Proc. of the 31st International Conf. on VLDB, Trondheim, Norway, 2005.
[20] A. R. Schmidt, F. Waas, M. L. Kersten, D. Florescu, I. Manolescu, M. J. Carey, and R. Busse. The XML Benchmark Project. Technical Report INS-R0103, CWI, April 2001.

저 자 소 개

안 동 찬

2004년 8월 : 서강대학교 컴퓨터학과 공학박사 수료
1996년 ~ 1999년 : SK텔레콤(주) 정보기술연구원
2002년 ~ 현재 : 안산공과대학 디지털미디어과 교수
관심분야 : XML 데이터베이스, 데이터스트림처리

변 창 우

1999년 2월 : 서강대학교 컴퓨터학과 공학사
2001년 2월 : 서강대학교 컴퓨터학과 공학석사
2003년 2월 : 서강대학교 컴퓨터학과 공학박사 수료
관심분야 : XML 데이터베이스, XML 접근제어, 접근제어 모델