# XML 문서를 위한 효율적 접근 제어 리스트

이용규, 김신우
동국대학교 컴퓨터공학과

# Efficient Access Control List for XML Documents

Yong Kyu Lee, Shin Woo Kim
Dept. of Computer Engineering, Dongguk University

**요 약**

지금까지 XML 문서는 사용자에게 문서내의 모든 내용을 공개하였지만, 전자상거래와 같은 특정 분야의 경우에는 사용자에 따라 문서의 일부만을 공개하는 것이 필요하다. 따라서, 사용자에게 XML 문서의 권한을 부여하고 권한에 따라 접근을 관리하는 접근 관리 시스템이 요구된다. 이를 위하여 사용자 그룹을 권한 주체의 기본 단위로, XML 문서의 엘리먼트를 권한 객체의 기본 단위로 설정하여야 한다. 이러한 권한 주체의 계층 구조는 DAG(Directed Acyclic Graph) 형태로 표현되고 문서에 대한 접근 권한은 접근 제어 리스트를 이용하여 관리된다. 그러나 권한 주체마다 모든 접근 권한을 표시해야 하기 때문에 엘리먼트 단위의 접근 관리를 위해서는 접근 제어 리스트의 크기가 커지게 되는 문제점이 발생한다. 이를 해결하기 위해서 본 논문에서는 DAG를 완전 k-ary 트리로 변환하여 접근 권한을 부모와 자식 노드 간에 상속받을 수 있도록 함으로써 접근 제어 리스트의 크기를 상당히 줄이면서도 권한 주체의 접근 권한을 빨리 알아낼 수 있는 새로운 방법을 제시한다. 성능 분석 결과 새로운 방안이 기존의 접근 제어 리스트에 비해 매우 효과적임을 알 수 있다.

## 1. 서론

XML[3]은 자료의 공유 측면에서 문서의 모든 내용에 대하여 문서에 접근하는 사용자에게 제한 없이 공개한다. 그러나, 전자상거래와 같은 특정 분야의 경우에는 타인에게 공개하지 않아야 하는 개인정보 등에 대한 보안이 필요하며, 이에 따라 문서에 대한 접근 관리가 요구된다. 따라서, 문서를 생성할 때, 문서의 구조와 내용에 대해서 사용자들에게 접근 권한을 부여하고, 접근 권한을 소유하는 사용자만이 문서의 특정 내용에 접근 가능하도록 하기 위한 접근 권한 관리가 필요하다. 이를 위해서는 사용자의 접근 권한을 관리할 수 있어야 하며, 사용자가 XML 문서에 접근할 때 권한에 따라 제어할 수 있어야 한다.

접근 관리에 대한 관련 연구로는 객체 지향 데이터베이스의 권한 관리 모델[4]이 있다. 객체 지향 데이터베이스의 모델은 접근의 대상인 객체와 객체에 대한 권한을 공유하는 사용자 주체 그리고, 주체가 객체에 대해서 행사할 수 있는 권한 유형의 기본적 구성 요소를 갖는다. 그리고, 각각의 권한 주체들에 대하여 권한 연산을 통해서 권한을 관리하고, 권한에 따라 사용자의 객체에 대한 접근을 통제한다. 최근의 XML 스키마 모델[1][2]은 XML 문서를 데이터로 보는 관점에서 객체 지향 개념을 적용하여 XML의 엘리먼트를 문서 구조의 기본 단위인 객체로 표현한다. 따라서, XML 스키마 모델에서 엘리먼트를 객체로 정의하고, 객체 지향 데이터베이스의 접근 권한 모델을 응용한다면 XML 문서의 접근 권한 관리가 가능하다. 그러나, 데이터베이스에서는 클래스의 인스턴스에 대한 접근 관리가 일반적인데 반하여, XML은 데이터의 중복을 최소화하기 위해서는 문서 인스턴스내의 엘리먼트에 대한 접근 관리가 필요하다.

XML 문서의 엘리먼트에 대한 접근 관리를 위해서는 사용자 그룹을 권한 주체의 기본 단위로, XML 문서의 엘리먼트를 권한 객체의 기본 단위로 설정하여야 한다. 이러한 권한 주체의 계층 구조는 DAG(Directed Acyclic Graph) 형태로 표현되고 문서에 대한 접근 권한은 접근 제어 리스트를 이용하여 관리하여야 한다. 그러나 권한 주체마다 모든 접근 권한을 표시해야 하기 때문에 엘리먼트 단위의 접근 관리를 위해서는 접근 제어 리스트의 크기가 매우 커지게 되는 문제점이 발생한다. 이를 해결하기 위해서 본 논문에서는 DAG를 완전 k-ary 트리로 변환하여 접근 권한을 부모와 자식 노드 간에 상속받을 수 있도록 함으로써 접근 제어 리스트의 크기를 상당히 줄이면서도 권한 주체의 접근 권한을 빨리 알아낼 수 있는 새로운 방법을 제시한다.

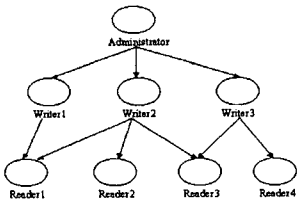
이를 토대로 XML 문서의 접근 권한 및 접근 관리 시스

템을 구현한다. 사용자의 권한을 관리하기 위해서 사용자 그룹을 권한 주체의 기본 단위로 설정하고, 권한 주체별 접근 권한을 본 논문에서 제시하는 접근 제어 리스트에 저장하고, 사용자가 XML 문서에 접근할 때 이 접근 제어 리스트를 사용하여 접근이 허용된 내용만을 사용자에게 보여주도록 한다.

## 2. 접근 권한 관리 및 접근 리스트

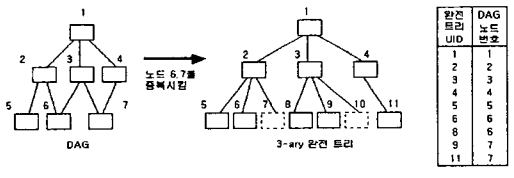
### 2.1 접근 권한 관리

권한 주체는 계층적으로 표현할 수 있으며, [그림 1]은 권한 주체 계층의 한 예로 권한 유형에 해당하는 주체의 역할을 계층적으로 나타낸 그림이다.




[그림 1] 권한 주체 계층의 예

[그림 1]에서 보는 것처럼 권한 주체의 계층 구조는 DAG (Directed Acyclic Graph) 형태를 갖는다. 여기서 어떤 노드의 부모 노드는 자식 노드의 모든 권한을 묵시적으로 행사할 수 있으므로 접근 권한을 중복하여 기술하지 않고 문서에 대한 접근 시 권한 연산을 빨리 수행할 수 있도록 DAG를 K-ary 완전 트리(complete tree)로 변환하여 노드마다 유일한 ID(UID: Unique ID)를 부여한다. [그림 2]는 이를 설명하고 있는데, DAG의 공유 노드는 완전 트리에서 중복되어 나타나며 점선으로 표시된 노드는 가상의 노드이다.



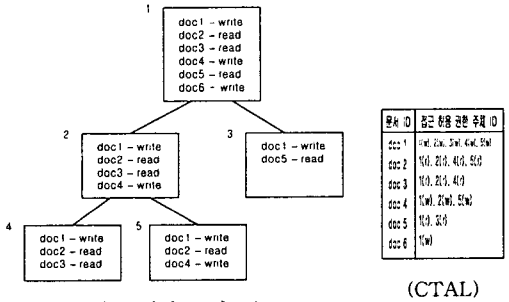

[그림 2] DAG의 k-ary 완전 트리 변환

이와 같이 UID를 부여하면 어느 노드의 자식 노드와 부모 노드의 UID는 (식1)과 (식2)에 의해 바로 알 수 있게 된다.

노드 $i$의 부모 노드 : $Parent(i) = \left\lfloor \frac{(i-2)}{k} + 1 \right\rfloor \cdots$ (식1)

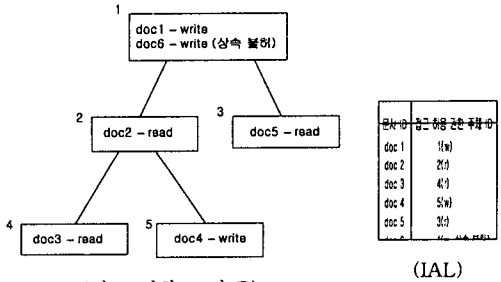
노드 $i$의 $j$번째 자식 노드 : $Child(i,j) = k(i-1)+j+1 \cdots$ (식2)

여기서 어느 노드가 소유하는 접근 권한을 모두 기록할 필요 없이 모든 자식 노드들이 공유하는 권한은 부모 노드에만 기록하여 이로부터 상속받을 수 있도록 하면 접근 권한 리스트의 크기를 많이 줄일 수 있다. 예를 들어 [그림 3]과 같은 권한 계층 구조와 이에 따른 접근 리스트 CTAL (Complete Tree Access List)은 [그림 4]의 권한 트리와 접근 리스트 IAL(Inherited Access List)과 같이 단순화 할 수 있다. 그러나 자식 노드에게 상속되지 않는 권한은 상속 불허를 표시하여야 한다.




(접근 권한 트리 A) (CTAL)

[그림 3] 권한의 중복 등록




(접근 권한 트리 B) (IAL)

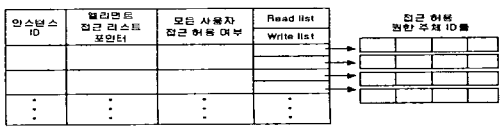
[그림 4] 권한의 상속

[그림 4]에서 어느 노드의 접근 권한은 다음의 (식3)에 의해 구할 수 있다.

접근 권한($i$) = 권한($i$) ∪ 상속 가능 권한(조상 노드들) ∪ 권한(자손 노드들) ····· (식3)

### 2.2 접근 리스트

접근 권한의 관리는 사용자 그룹의 XML 문서에 대한 접근 권한을 유지하고 사용자가 특정 XML 문서에 접근 할 때 권한을 검사하기 위한 것이다. 사용자에게 XML 문서의 엘리먼트 단위까지 접근을 관리하기 위해서는 문서 인스턴트에 대한 접근 리스트(Access List)가 필요하며 엘리먼트에 대해서도 접근 리스트의 관리가 필요하다. 다음의 [그림 5]는 인스턴스 접근 리스트를 설명하고 있으며 엘리멘트 접근 리스트는 이와 유사하다.




[그림 5] 인스턴스 접근 리스트

## 3. 접근 리스트 성능 평가

### 3.1 접근 리스트의 크기 분석

접근 리스트에서 필요로 하는 기억 장소의 크기를 알기 위해서는 접근 리스트에서 관리하여야 할 문서별 접근 허용 주체의 수를 알아야 한다.

먼저, [그림 3]에서 설명한 권한 트리 A의 높이를 $h$라 하고, 분석을 쉽게 하기 위하여 full k-ary 트리라 가정하고, 자식 노드의 어느 권한이 부모 노드에서 기록되는 비율을 $u$라 한다.

권한 트리 A에서 리프 노드의 평균 권한의 수를 $m$이라 하면, 레벨 $i$의 어느 노드의 권한의 수는 $k^{h-i}*m*u^{h-i}$이다. 또한, 레벨 $i$의 모든 권한의 수의 합은 $k^{h-1}*m*u^{h-i}$이 된다. 따라서, 이 트리의 모든 권한의 수는 다음과 같다.

$$\sum_{i=1}^{h} k^{h-1}*m*u^{h-i} \quad \cdots\cdots\cdots\cdots \text{(식4)}$$

이제 [그림 4]의 권한 트리 B의 경우를 분석하기로 한다. 이를 위해 자식 노드의 권한이 부모 노드로 승격되게 될 확률을 $p$라 하자. 그러면 어느 리프 노드의 권한의 수는 $m-m*p$가 된다. 그리고, 레벨 $i$의 어느 노드의 권한의 수는 $m*p^{h-i}-m*p^{h-i+1}$이 되고, 레벨 $i$의 모든 권한의 수는 $k^{i-1}*m*p^{h-i}-k^{i-1}*m*p^{h-i+1}$이 된다. 또한, 루트 노드의 권한의 수는 $m*p^{h-1}$이므로, 트리 B의 전체 권한의 수는 다음과 같다.

$$\sum_{i=2}^{h}(k^{i-1}*m*p^{h-i}-k^{i-1}*m*p^{h-i+1})+m*p^{h-1}$$

$$= k^{h-1}*m-m*\sum_{i=1}^{h-1} p^{i}*(k^{h-i}-k^{h-i-1}) \quad \cdots\cdots\cdots \text{(식5)}$$

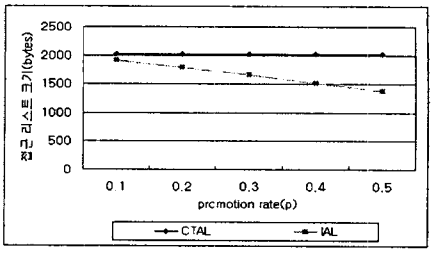
### 3.2 접근 리스트의 크기 분석 결과

접근 리스트 CTAL과 IAL의 크기를 비교하기 위하여 앞 절의 식과 다음 [표1]의 파라미터 값을 이용하였다.

[표 1] 파라미터의 값

| 파라미터 | 값 |
|---|---|
| 문서의 수 | 100 |
| 문서 ID | 4bytes |
| 권한 주체 ID | 4bytes |
| k | 3 |
| h | 5 |
| m | 5 |
| u | 0.5 |

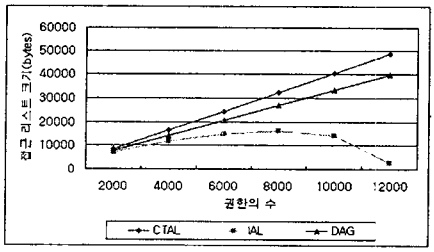
다음의 [그림 6]은 CTAL과 IAL의 기억 공간의 크기를 보여 주고 있다. 그림에서 보듯이 promotion rate가 커질수록 IAL이 CTAL에 비하여 더욱 우수함을 알 수 있다.




[그림 6] 접근 리스트의 크기 비교

### 3.3 접근 리스트 크기 실험

CTAL과 IAL의 크기를 DAG를 사용한 접근 리스트와 보다 정확히 비교하기 위하여 실험을 실시하였다. 먼저, 파라미터 값은 ($m$과 $u$ 제외) [표 1]과 동일하게 설정하였으며, 난수 발생기를 이용하여 필요한 개수의 (문서 ID, 권한 주체 ID) 쌍을 임의로 10회 생성한 후 접근 리스트의 평균 크기를 구하였다. 다음의 [그림 7]은 권한의 수에 따른 접근 리스트 크기의 변화를 보여주고 있으며, CTAL 보다는 DAG를 사용한 접근 리스트가, DAG 보다는 IAL이 더 우수함을 보여주고 있다. 이 결과는 DAG를 트리로 변환할 때 전체 리프 노드 중 20%가 중복되었다고 가정한 결과이며, 다른 파라미터 값(10%, 30%, 40%)에 대한 실험에서도 유사한 결과를 얻었다.



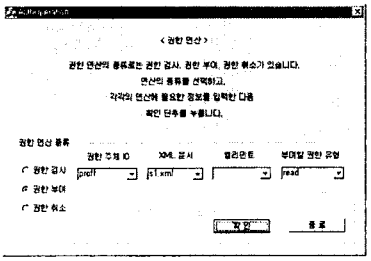

[그림 7] 접근 리스트의 크기 비교

## 4. 구 현

본 연구에서는 XML 문서의 접근 권한 관리 시스템과 XML문서의 접근 관리 시스템을 구현하였다. 시스템 구현 도구로는 Windows/NT 4.0 운영체제에서 마이크로소프트사의 XML-Data[5] XML 파서, ASP(Active Server Pages), DOM(Document Object Model)[6], 인터넷 익스플러러 5.0, Visual C++ 6.0을 이용하였다. 그리고 개발된 시스템을 대학의 성적 관리에 적용하였다.

### 4.1 사용자 권한 관리의 구현

사용자 권한 관리는 사용자인 권한 주체와 XML 문서에 대한 권한을 관리하는 것으로 사용자를 포함한 권한 주체의 등록과 XML 문서의 등록이 있고, 등록된 권한 주체와 XML 문서에 대한 접근 권한 연산으로 구성된다. [그림 8]은 권한 연산 화면을 보여주고 있다.

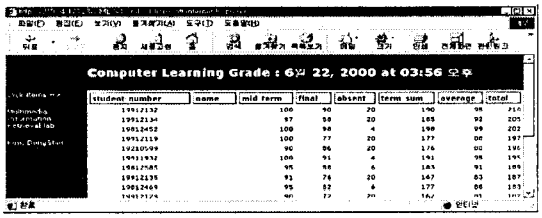


[그림 8] 권한 연산 화면

4.2 XML 문서 접근 관리의 구현

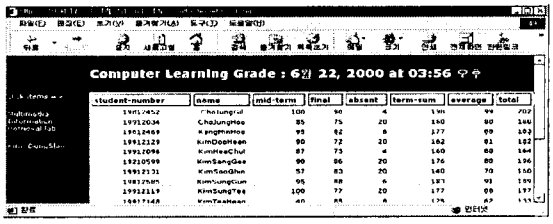
XML 문서의 접근 관리는 사용자인 주체 계층에 대한 인증 단계부터 시작하여 스키마 문서에 대한 처리와 인스턴스에 대한 처리의 두 부분으로 나누어 구현하였다. 사용자는 인증을 마친 후, 접근을 원하는 스키마 문서 또는 인스턴스 문서를 선택한다. 스키마 문서를 선택했을 경우에는 등록된 스키마 문서를 읽거나 삭제할 수 있도록 구현하였으며, 스키마의 삭제는 해당 인스턴스가 존재하지 않을 때만 삭제하도록 하였다.

인스턴스 문서의 접근 관리에서는 사용자가 접근이 허용된 인스턴스 문서를 읽거나 수정, 또는 삭제할 수 있다. 다음의 [그림 9]는 컴퓨터공학과에 속한 사용자가 'sl.xml' 문서의 읽기 기능을 선택한 화면으로 'name' 엘리먼트에 대해서 읽기 권한이 존재하지 않는 경우를 보여주고 있다.




[그림 9] 'name' 엘리먼트의 데이터를 숨긴 화면

그러나, [그림 10]은 'sl.xml' 문서내의 모든 엘리먼트를 읽을 수 있는 권한을 가진 사용자에 해당하는 화면을 보여주고 있다.




[그림 10] 모든 엘리먼트의 데이터를 보여준 화면

5. 결론 및 향후 연구

본 논문에서는 XML 문서의 엘리먼트들에 대한 효율적인 접근 제어 리스트를 설계하기 위하여 접근 주체의 계층 구조인 DAG를 완전 k-ary 트리로 변환하여 접근 권한을 부모와 자식 노드 간에 상속받을 수 있도록 함으로써 접근 제어 리스트의 크기를 상당히 줄이면서도 권한 주체의 접근 권한을 빨리 알아낼 수 있는 새로운 방법을 제시하였다. 성능 평가 결과 우리의 방법이 매우 효과적임을 알 수 있었다.

그리고 이를 이용하여 XML 문서에 대한 접근 권한과 웹에서의 접근을 관리할 수 있는 시스템을 설계하고 구현하였다. 사용자의 계층 구조를 DAG 형태로 정의하고, 각 사용자 그룹에 대하여 XML 문서의 엘리먼트까지 접근 권한을 부여한 후, 이로부터 구성된 접근 리스트를 이용하여 웹에서 사용자가 문서를 접근할 때 권한에 따라 문서의 해당 부분만을 접근할 수 있도록 하였다.

이렇게 구현한 접근 권한 관리 시스템을 대학의 학생 성적 관리에 적용하였다. 사용자 권한 관리로는 학교 구성원들의 그룹을 권한 주체로 설정했고, 권한 주체들을 DAG 형태로 구성하여 XML 스키마와 인스턴스 그리고 엘리먼트에 대한 권한을 검사, 부여, 취소할 수 있도록 하였다. XML 문서에 대한 접근 관리로는 스키마 문서와 인스턴스 문서로 나누어 처리하였고, 사용자가 인스턴스 문서에 접근할 경우 접근 제어 리스트를 이용하여 사용자 주체 별로 접근 권한을 소유한 내용만을 보여주도록 처리하였다. 그 결과 사용자의 권한에 따라서 각기 다른 XML 문서를 작성할 필요없이 하나의 문서로부터 권한에 맞는 내용만을 보여줌으로써, 문서의 중복으로 인한 문제들을 해결하면서 문서를 권한에 따라 보호할 수 있었다.

향후에는 문서의 내용이 변경되었을 때 동일한 문서의 여러 버전을 관리할 수 있도록 확장하는 연구가 필요하다.

[참고문헌]

[1] Andrew Davidson, et. al., "Schema for Object-oriented XML 2.0," http://www.w3.org/TR/NOTE-SOX, July 1999.

[2] David C. Fallside, "XML Schema Part 0: Primer," http://www.w3.org/TR/xmlschema-0/, September 2000.

[3] Charles F. Goldfarb and Paul Prescod, The XML Handbook, Prentice Hall, 1998.

[4] Won Kim, Introduction to Object-Oriented Databases, MIT Press, 1994.

[5] Andrew Layman, et. al., "XML-Data," http://www.w3.org/TR/1998/NOTE-XML-data-0105/, January 1998.

[6] Lauren Wood and Vidur Apparao, "Document Object Model(DOM) Level 1 Specification," http : //www.w3.org/TR/-REC DOM Level-1, 1999.