韓國컴퓨터情報學會 論文誌
第16卷 第8號, 2011. 8.

2011-16-8-1-3

자원제약적인 모바일 단말기에서 효율적인 질의처리를 위한 접근제어

안 동 찬*

Access Control for Efficient Query Processing on Limited Resource Mobile Terminal

Dongchan An*

요 약

과거 접근제어에 대한 연구는 효율적인 측면보다 안전성에 더 많은 초점이 맞추어져 있었다. 본 연구는 배터리나 메모리 등의 자원이 제약적인 개인 휴대용 단말기와 같은 환경에서 XML 데이터 스트림의 효율적이고 안전한 질의처리에 대한 방법을 제안한다. 특히, 본 연구는 접근제어 처리과정에서 발생할 수 있는 추가적인 오버헤드를 각 질의처리 단계별로 찾아내고 최소화하여, 자원의 사용을 최적화하며 안전한 결과를 얻기 위한 최소한의 오버헤드를 갖는 접근제어 처리 방법을 제안한다. 끝으로, 실험을 통해 본 연구에서의 제안방법에 대한 우수성 분석을 하였다.

▸ Keyword : 접근제어, 질의처리, 자원제약, XML

Abstract

Access control that has been previously performed mainly on safety, and thus not much effort has been done to consider access control in terms of efficiency. This paper proposes a method for an efficient and secure query processing of XML data streams, such as a personal digital assistant and a portable terminal, at the client side with limited resources. Specifically, this paper proposes an access control processing that possesses a small overhead for attaining a secure result with limited memory and a method to enhance the performance, finding the parts capable of optimizing each processing step for offsetting the overhead caused by the addition of access control processing. The superiority of the new method is analyzed through an experiments.

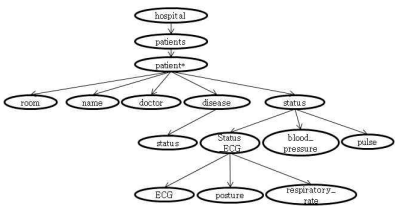
▸ Keyword : Access Control, Query Processing, Limited Resource, XML

• 제1저자 : 안동찬 • 교신저자 : 안동찬
• 투고일 : 2011. 04. 12, 심사일 : 2011. 05. 03, 게재확정일 : 2011. 05. 11.
* 신안산대학교 멀티미디어컨텐츠과(Dept. of Multimedia Contents, Shin Ansan University)
※ 본 논문은 2010학년도 신안산대학교 학술연구비에 의하여 연구된 것임

I. 서 론

XML[1] 문서의 접근제어 규칙은 질의와 유사한 XPath[2] 표현식으로 나타내며, 접근제어 규칙은 휴대용 개인 단말기에서 적용된다. 그림 1은 서버가 환자로부터 전송받는 정보의 DTD(document type definition) 구조와 의사 A와 의사 B의 접근제어 규칙(ACR: access control rules)을 보여주고 있다.

병원의 의사는 휴대용 단말기를 통해 환자의 정보를 수시로 체크한다. 각각의 환자는 최소 한명의 담당의사가 있으며, 환자의 상태에 따라 여러 명의 담당의사가 있는 경우도 있다. 담당의사는 환자의 의료정보를 수시로 체크할 수 있지만 담당이 아닌 의사 즉, 허가되지 않은 환자의 의료정보는 체크할 수 없는 것이 원칙이다. 환자의 입장에서 보면, 환자의 정신질환이나 사회생활에 영향을 줄 수 있는 의료정보는 엄격히 허가된 담당자에게만 열람되어야 한다. 그러나 서버는 정보 수신을 원하는 모든 클라이언트에게 일일이 구별된 정보를 전송할 수 없다. 왜냐하면 다양한 사용자별로 다양한 접근제어 규칙을 모두 관리 할 수 없기 때문이다. 또한, 연속적인 데이터의 브로드캐스팅이 이루어지는 환경에서 다양한 경우의 결과를 고려할 때 많은 비용이 필요하기 때문이다. 그러므로 서버가 모든 정보를 브로드캐스팅 한다고 해도 클라이언트는 사용자의 질의에 맞는 응답과 접근 가능한 내용에 대해서만 허용되어야 한다[3].



Doctor A Access Control Policy	
R1	+, /hospital/patients/patient
R2	−, //patient[doctor!=USER]/disease
Doctor B Access Control Policy	
R3	+, /hospital/patients/patient
R4	−, //patient/name
R5	−, //patient/room

그림 1. 자원제약적인 클라이언트 환경에서 XML 데이터 스트림의 DTD 구조와 접근제어 정책
Fig. 1. DTD structure and ACR of XML data streams by a client with limited resources

질의처리의 전통적인 방법은 사용자의 질의 도메인이 접근제어 규칙을 통해 접근가능한 도메인인지 확인한 후 실행된다. 사용자로 부터 질의가 들어오고 접근이 허용되면, 질의의 결과는 질의처리기를 통해 제공되어야 한다. 마찬가지로, 이러한 기법의 응용은 독립적인 접근제어의 실행을 가능하게 하지만 질의처리의 실현은 불가능할 수 도 있다. 왜냐하면 접근제어 엔진과 질의처리 엔진은 제약적인 자원을 가진 클라이언트에서의 실행을 필요로 한다. 그러므로 우리는 자원제약적인 모바일 단말기 환경에서 효율적인 질의처리를 위한 접근제어를 필요로 한다.

본 논문에서 우리는 자원제약적인 모바일 단말기에서 접근제어 기술과 질의처리 기술을 동시에 실행 가능한 방법을 제안한다. 첫 번째, 제한된 메모리 자원을 갖는 환경에서 안전한 질의처리를 위해 오버헤드가 거의 없는 접근제어 처리 방법을 제안한다. 두 번째, 본 연구는 접근제어 기능의 추가로 인한 질의처리시 오버헤드를 최소화 하고 동시에 자원 사용의 효율적인 질의처리를 목적으로 한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서 XML 접근제어와 클라이언트 기반 질의처리 기법에 대한 관련연구를 살펴보고, 3장에서 자원제약적인 모바일 단말기에서 효율적인 질의처리 기법에 대해 소개한다. 4장에서 제안하는 기법의 효율성을 실험을 통해 입증하며 5장에서 결론 및 향후 연구방안을 제시한다.

II. 관련 연구

전통적인 XML 접근제어 기법[4-7]은 뷰(view)-기반 접근제어 기법이다. 접근제어의 의미는 관련 접근제어 정책에 따라 적절한 사용자에게 허용된 문서의 특정 뷰를 허용하는 것이다. 이 기법은 트리 레이블링을 이용해 뷰 계산을 위한 유용한 알고리즘을 제공하기도 한다. 그러나 뷰 유지를 위한 고비용이 문제이며 사용자 수가 증가하게 되면 뷰를 유지 하는데 많은 문제가 발생한다[8,9].

뷰-기반 접근제어의 문제점을 극복하기 위해 [10]는 접근제어 규칙에 위배되는 질의를 제거하는 필터링 기법을 제안하였다. [11]은 질의처리를 위한 질의에 관련된 접근제어 규칙을 적용한 질의로 재작성하는 추가적인 단계를 제안하였다. 그러나 접근제어 규칙의 공유 비결정적 유한 오토마타(NFA: nondeterministic finite automata)는 사용자 또는 사용자의 역할(role)에 의해 작성된다. 따라서 공유 NFA는 사용자의 질의 관점에서 접근제어에 불필요한 접근제어 규칙까지도 포함하게 되며, 사용자의 질의에 대해 허용, 거절, 재작성

의 결정을 위한 많은 시간을 낭비하게 된다.

클라이언트 기반 XML 데이터 스트림의 접근제어[12]는 질의의 응답을 위한 접근제어 규칙의 평가와 동시에 실행되고, 질의 재작성 없이 현재 입력되는 데이터의 접근제어 상태를 결정한다. 결국, 오토마타(ARA: access rules automata)에 의한 사용자의 질의에 대하여 [11]의 방법처럼 질의처리를 위한 질의에 관련된 접근제어 규칙을 적용한 질의로 재작성하지 않고, [12]에서는 XML 문서는 통과되고 최후 사용자의 질의에 대해 최후의 안전한 XML 문서를 포함하게 된다.

한편, 기존 접근제어 환경에서는 신뢰성 있고 성능이 뛰어난 서버가 개개의 사용자에 맞는 정보를 제공해주는 역할을 담당했다. 그러나 최근 인터넷이나 개인 휴대 단말기 등의 보급이 널리 확산되면서 정보에 대한 접근이 용이해짐에 따라 다수의 다양한 사용자가 생겨나게 되었고, 다음과 같은 여러 가지 요인들로 인해 접근제어 환경이 빠르게 변하고 있다[12].

(1) 내부/외부 공격 증가로 인한 데이터베이스 서버 취약성 증가
(2) 분산된 데이터 공유/처리 문제
(3) 인터넷 등의 접근 시 부모가 자녀들의 접근을 제어하고자 하는 요구

위와 같은 요인들은 서버에서 담당했던 접근제어 프로세스를 클라이언트로 이동하게끔 하고 있다. 실제로 Microsoft사의 Windows Media Player는 개인 PC에서 소프트웨어적으로 출판된 자산을 보호할 수 있도록 하는 솔루션을 제공하고 있으며, PC나 PDA등에서 보안 토큰이나 스마트카드 등을 탑재함으로써 하드웨어적으로 보안 기능을 제공하는 경우도 있다. 그러나 이러한 보안 기능은 거의 데이터 암호화에 의지하고 있으며 사용자마다 다르게 접근할 수 있는 해결책으로 부족함이 있다. 그 이유는 첫째, 클라이언트에서의 접근제어는 다양한 사용자에 맞는 접근제어 정책을 적용할 수 있어야 하나 암호화는 사용자 마다 다른 암호 키를 필요로 하므로 수많은 암호 키를 관리해야 하는 어려움을 갖는다. 둘째, 스트림 환경에서 이러한 암호화 방법은 서버와 클라이언트 간에 암호 키를 주고받을 때 비동기화 문제가 발생할 수 있다. 셋째, XML이 데이터 저장과 전송의 표준으로 부각되면서 접근제어 연구는 세밀한 접근제어(fine-grained access control)로 확대되고 있다. 세밀한 접근제어란 문서 자체의 접근을 제어하는 것이 아니라 보통 문서 내의 엘리먼트(element) 단위의 접근제어를 말한다. XML은 계층적 구조로 이루어지므로 세밀한 접근제어가 가능하다. 하지만 암호화 방법은 세밀한 접근제어를 수행하지 못한다.

결국, 클라이언트와 같은 자원제약적인 환경에서 적용이 가능한 효율적인 질의처리를 위한 접근제어가 필요하다. 효율적인 질의처리를 위한 접근제어란 자원의 사용을 최소화하며 빠르고 정확한 질의처리를 위한 접근제어를 의미한다.

III. 자원제약적인 모바일 단말기에서 효율적인 질의처리

1. 시스템 구조

그림 2는 본 연구에서 제안하는 시스템의 구조를 보여주고 있다.

서버는 수많은 데이터 소스들로부터 정보를 제공받고 이들 정보를 취합하여 하나의 XML 문서로 만든다. 따라서 클라이언트에 브로드캐스트 할 DTD 문서와 XML 문서를 가지고 있으며 이들 문서가 변경될 때마다 클라이언트에 정보를 보낸다. DTD 문서의 경우 거의 변경이 없으나 XML 문서는 데이터 소스로부터 끊임없이 데이터를 받아 계속해서 갱신 된다.

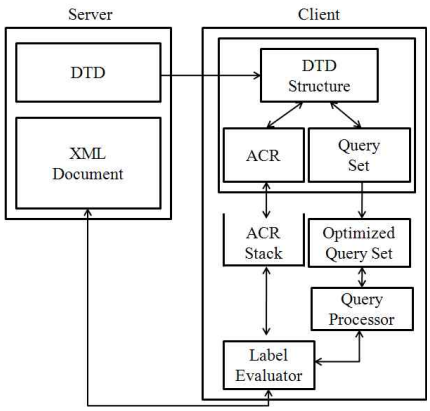


그림 2. 시스템 구조
Fig. 2. System Architecture

클라이언트는 서버로부터 전송 받은 DTD를 저장하며, 접근제어 규칙 및 사용자로부터 입력 받은 질의, 그리고 끊임없이 입력되는 XML 파일을 처리하기 위한 모듈들을 가지고 있다. 본 연구에서는 휴대용 단말기와 같이 메모리 자원이 적은 클라이언트를 가정하고 있다.

DTD 구조는 질의 및 접근제어 처리를 위해 서버로부터 전송 받을 XML 문서의 구조 정보를 알고 있어야 한다. 따라서 클라이언트는 서버로부터XML 문서를 전송 받기 전에 먼저 DTD를 전송 받아 질의처리에 알맞은 구조로 변환하여 저장한다. 본 연구에서는 DTD에 (pre, post) 값을 부여하여 해시 형태로 저장한다.

클라이언트 장치는 기본적으로 사용자에 맞는 접근제어 규칙을 탑재하고 있으며 접근제어 규칙이 탑재되는 경로는 다양할 것이다.

질의 셋은 사용자로부터 입력 받는다. 클라이언트는 사용자로부터 입력 받은 다수의 질의를 저장함으로써 사용자가 원하는 결과를 내보내도록 한다. 이 때 빠른 질의처리에 적합하도록 최적화된 구조로 변환하여 가지고 있게 된다.

접근제어 스택(ACR stack)은 접근제어 규칙 계산 결과 현재 입력되고 있는 데이터에 대한 접근 여부를 판단하기 위해 접근제어 스택에 조건을 만족하는 접근제어 규칙을 유지한다.

레이블 계산기(label evaluator)는 질의 및 접근제어 규칙 처리시 입력 데이터와의 비교를 위해 입력 데이터의 레이블을 계산한다. 레이블 계산기의 적용으로 인해 빠른 계산 및 질의와 접근제어 규칙 유지를 위한 메모리 감소 효과를 가져올 수 있다.

질의처리기(query processor)는 질의와 접근제어 규칙을 동시에 계산한다. 결론적으로 사용자가 원하는 결과를 계산하고 접근제어 스택의 값과 비교하여 사용자에게 결과를 내보낼지 결정한다.

2. 접근제어 정책

계층적 데이터 모델(예를 들어, Object-Oriented Data Model, XML Data Model 등)[13]에서는 보안 관리자가 명시한 권한부여는 명시적(explicit)이라고 부르며, 명시적 권한부여를 기반으로 시스템이 파생한 권한부여를 묵시적(implicit)이라고 한다. 저장의 잇점을 얻고자 묵시적 권한부여 방법을 사용하면서 적절하게 '전파 정책(propagation policy)'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)'문제를 해결하는 정책으로는 '거절 우선 정책(denial takes 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 접근을 허용하는 '개방(open) 정책'과 접근을 불허하는 '닫힘(closed) 정책'을 사용한다.

일반적으로, 엄격한 데이터 보안을 위해 '거절 우선 정책'과 '닫힘 정책'을 사용한다. 본 논문에서의 접근제어 정책은 '묵시적 권한부여' 기반의 '가장 특정화된 것을 우선으로 하는 정책'을 사용하며 충돌 해결 정책으로 '거절 우선 정책' 및 긍정적/부정적 권한부여 혼용을 위해 '닫힘 정책'을 사용한다[14].

3. 전처리

3.1 DTD 해시 테이블

서버가 DTD 문서를 전송하면 클라이언트는 질의 계산 및 접근제어 규칙 계산에 이를 활용하기 위해 DTD를 순차적으로 접근하며 엘리먼트 정보를 Pre/Post 값으로 변환하여 DTD 해시를 구성한다. DTD 해시를 구성할 때 Pre/Post 구조를 사용하는 이유는 조상(혹은 부모)과 자손(혹은 자식)을 빠르게 탐색할 수 있으며 XML 문서 표현 시 임의의 노드에 대한 Pre/Post 값을 통해 루트(root)부터의 경로를 파악할 수 있기 때문이다.

그림 3은 그림 1의 DTD 트리에 대한 Pre/Post 해시를 보여주고 있다. 우선 Pre(order)는 DTD를 순차적으로 접근하여 얻은 값을 말하고, Post(order)는 다음과 같은 수식을 통해 얻는다. : POST = PRE + SIZE - LEVEL

LEVEL은 DTD 트리에서의 레벨을 말하며 SIZE는 임의의 트리 노드에서 하부-트리의 노드 수를 말한다.

그림 3에서 나타나듯이 DTD 해시는 태그 이름(tag-name)을 키로 가지며 네 개의 정보({tag-name, pre, post, parent})를 유지한다. 따라서 DTD 해시의 탐색시에는 태그 이름으로 검색하는데, 만일 DTD 문서에 중복 정의된 엘리먼트가 존재하면 해시에서 여러 개의 (pre, post) 값을 찾게 된다.

Tag-name	Pre	Post	Parent
hospital	0	14	-
patients	1	13	hospital
patient	2	12	patients
room	3	0	patient
name	4	1	patient
doctor	5	2	patient
disease	6	4	patient
status	7	3	disease
status	8	11	patient
status_ECG	9	8	status
ECG	10	5	status_ECG
posture	11	6	status_ECG
respiratory_rate	12	7	status_ECG
blood_pressure	13	9	status
pulse	14	10	status

그림 3 DTD 트리(Fig. 1)의 PRE/POST 해시테이블
Fig. 3. PRE/POST hash table of the DTD tree (Fig. 1)

3.2 질의와 접근제어 규칙 등록

접근제어 규칙 및 사용자 질의 등록은 컴파일 타임시 DTD 트리에 대한 PRE/POST 메타데이터를 이용하여 처리된다. 본 절의 내용은 XPath 경로식을 일차적으로 (pre, post) 경로식으로 바꾸고, 이차적으로 XPath 경로식 패턴 분석을 통해 최적화된 (pre, post) 경로식으로 바꿔 등록하는 과정이다.

XPath 경로에 해당하는 Pre/Post 값을 탐색하는 과정은 다음과 같다. 탐색의 순서는 목적 노드로부터 경로 표현식의 시작 노드로 거슬러 올라간다. 이 때 프레디키트의 분기 노드를 만나면 프레디키트의 값을 먼저 탐색한 후, 다시 분기 노드의 이전 노드에 대한 탐색을 계속한다. 탐색된 값이 XPath를 만족하는지 알기 위하여 먼저 탐색된 노드의 (pre, post) 값과 새로이 탐색된 노드의 (pre, post) 값을 비교하여 새로이 탐색된 노드가 조상 관계에 있는 노드인지 확인한다.

트리 구조에서 임의의 노드에 대한 부모는 유일하므로 만약 탐색된 (pre, post) 값 중 조상 관계를 가지는 값이 있으면 이는 조상 노드가 맞고, 조상 관계를 가지는 값이 탐색되지 않으면 이는 XPath 표현식이 DTD를 맞게 표현되지 않았음을 뜻한다. 따라서 이러한 올바르지 않은 질의나 접근제어 규칙은 계산에서 제외한다.

4. 질의처리

질의처리 과정에서는 전처리 과정에서 등록된 정보를 이용하여 접근제어 처리와 질의처리를 동시에 수행한다.

4.1 XML 데이터 스트림의 처리

서버로부터 XML 문서를 전송 받으면 질의와 접근제어 규칙 계산을 위해 입력된 데이터의 (pre, post) 정보를 알아야 한다. DTD 해시는 입력된 데이터의 태그 이름으로 알맞은 (pre, post) 값을 빠르게 탐색할 수 있게 해준다. 그러나 별도의 정보를 유지하지 않으면 여러 개의 (pre, post) 값이 탐색될 경우, 입력 데이터에 대한 DTD 상의 정확한 위치를 알 수 없게 되므로 알맞은 (pre, post) 값을 결정하기 위해 질의 경로상에 나타나는 모든 노드를 유지하고 있어야 한다. 따라서 본 연구에서는 새로이 입력될 데이터의 정확한 정보를 알아내기 위해 이전에 입력된 데이터의 정보를 유지한다. 새로운 데이터가 입력되면 바로 이전에 입력된 데이터의 정보를 이용하여 (pre, post) 값을 계산한다. 이 방법은 입력된 데이터의 정보 계산을 위해 이전에 입력받은 데이터의 (pre, post) 정보만을 유지함으로써 질의 최적화를 가능하게 하므로 질의 경로 상의 모든 노드를 유지하는 것보다 메모리 활용면에서 효율적이다.

입력 데이터의 정보는 두 가지 경우로 나누어 계산된다. 입력 데이터 계산을 위해 유지해야 할 정보를 Current_data라 하자. 먼저 엘리먼트의 시작(start element)을 알리는 태그가 입력되면 이는 바로 이전 데이터의 자식이거나 루트노드이다. Current_data의 값이 NULL이면 입력된 데이터는 루트 노드이고, DTD 해시에서 루트 노드에 대한 정보를 쉽게 찾을 수 있다. 입력 데이터가 루트 노드가 아닌 경우, DTD 해시에서 입력된 데이터의 태그 이름으로 (pre, post) 값을 탐색한다. 찾은 결과 중 기존에 유지하고 있던 데이터의 자식에 해당하는 값을 찾으면 Current_data의 값을 새로 찾은 값으로 바꾼다. 엘리먼트의 끝(end element)을 알리는 태그가 입력되면 다음에 입력될 데이터를 위해 Current_data의 값을 이전 값으로 돌려놓는다 즉, 엘리먼트가 끝나면 다음에 입력될 데이터는 현재 입력 받은 데이터와 동일한 부모를 갖는 동일한 레벨의 엘리먼트이다 따라서 DTD 해시에서 Current_data의 부모에 해당하는 pre, post 값을 탐색하고, 찾은 결과와 Current_data의 pre, post 값을 비교하여 부모 관계를 갖는 값으로 Current_data의값을 수정한다. 이렇게 Current_data를 계산하면 질의 계산 시 빠르고 정확한 비교를 가능하게 하며 질의의 모든 정보를 유지할 필요가 없으므로 질의 최적화를 할 수 있고 좀 더 가벼운 처리가 가능해진다.

4.2 접근제어 규칙의 처리

질의 계산과 접근제어 규칙의 계산은 동일하나 입력되는 데이터의 접근 가능 여부를 판단하기 위해 접근제어 규칙을 계산 할 때에는 접근제어 스택을 유지한다. 접근제어 규칙이 명시된 엘리먼트가 입력될 때마다 해당 접근제어 규칙의 접근 가능 여부를 스택에 입력하게 되는데, 접근제어 스택은 다음과 같이 네 가지 값을 가질 수 있다.

(+) : 접근 가능한 상태
(+?) : 잠재적으로 접근 가능한 상태
(-) : 접근 불가능한 상태
(-?) : 잠재적으로 접근 불가능한 상태

위의 네 가지 상태 중 +와 - 상태는 프레디키트를 포함한 접근제어 규칙을 만족하여 해당 규칙의 접근 여부가 적용되는 상태이며, +?와 -? 상태는 프레디키트를 제외한 규칙은 만족되지만 프레디키트의 조건을 만족하는지 따져본 후 접근 여부를 적용할 수 있는 상태를 말한다.

본 연구에서 사용하는 접근제어 모델에 맞추어 접근제어 스택의 계산 방법을 정의하면 다음과 같다.

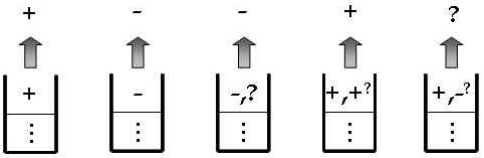


그림 4 접근제어 스택
Fig.4.Stack of access control

본 연구는 '닫힘 정책'에 따라 스택이 비어있을 경우, 기본적으로 데이터에 접근할 수 없다. 따라서 질의를 만족하더라도 스택의 top에 + 값이 오기 전까지 결과를 내보내지 않는다.

스택의 top에 + 값이 있는 경우, 또 다른 접근제어 규칙을 만족하면 해당 규칙을 top에 입력한다. 이 경우 입력 데이터에 대한 접근이 허용되므로 질의를 계산하고, 질의를 만족하면 결과를 내보낸다.

스택의 top에 - 값이 있는 경우, 사용하는 접근제어 모델은 접근이 불허된 노드의 하위 노드는 접근할 수 없다. 따라서 이후에 입력되는 값들은 모두 -로 변환한다. 이 경우 입력 데이터에 대한 접근이 허용되지 않으므로 질의를 계산하지 않는다. 즉, 질의 입장에서는 데이터가 입력되지 않은 것처럼 처리하게 된다.

스택의 top에 +? 값이 있는 경우, 이는 잠재적으로 접근 가능할 수 있으므로 만약 질의를 만족하면 그 결과를 임시로 저장한다. 만약 나중에 프레디키트를 만족하면 +로 값이 바뀌고 임시로 저장했던 질의 결과를 사용자에게 내보낸다. 프레디키트를 만족하지 않으면 스택에서 출력하고 임시로 저장했던 질의 결과는 버린다.

스택의 top에 -? 값이 있는 경우, 이는 잠재적으로 접근 불가능하거나 아예 접근제어 규칙을 만족하지 않을 수도 있다. 따라서 질의를 만족하더라도 결과를 내보낼 필요가 없으므로 그 결과를 저장하지 않는다. 만약 이후에 프레디키트를 만족하게 되면 -로 바뀌고, 이 경우 스택에서 상단에 입력되었던 값들도 모두 -로 변경된다. 상단에 +? 값이 있을 경우 그 값이 -로 바뀌면서 임시로 저장했던 질의 결과는 사용자가 접근할 수 없는 정보가 되므로 버리게 된다.

4.3 접근제어를 포함하는 질의처리

질의의 계산은 접근제어 규칙의 계산과 유사하나 다음 네 가지 경우로 나누어 처리한다.

첫째, 접근제어 스택의 값이 -이면 입력된 데이터에 대해 질의를 계산하지 않는다. 이는 접근제어 규칙의 계산에 의해 접근이 불허된 노드는 사용자 입장에서 입력 사실을 모르므로 입력되지 않은 것처럼 처리하기 위함이다.

둘째, 접근제어 스택의 값이 -? 상태인 경우 잠재적으로 -의 가능성이 있으나 질의를 만족하지 않을 수도 있다. 두 가지 경우 모두 질의의 결과를 내보낼 필요가 없으나 -의 경우와는 달리 질의를 계산한다.

셋째, 접근제어 스택의 값이 +이면 입력되는 데이터에 접근 가능하므로 질의를 계산하고, 질의를 만족하는 데이터가 입력되면 결과로 내보낸다.

넷째, 접근제어 스택의 값이 +? 상태인 경우 잠재적으로 +의 가능성이 있으나 아직 그 결과를 알 수 없으므로 질의를 계산하되 질의를 만족하는 데이터가 입력되면 내보내지 않고 임시로 저장한다. 이후에 +? 값이 +로 바뀌면 임시로 저장한 질의의 결과를 내보낸다.

IV. 실험 및 평가

본 장에서는 제안 기법과 관련 연구로써 클라이언트 기반의 XML 접근제어 방법[12]의 성능을 비교하고 제안 기법에서 접근제어 적용 전후의 성능을 비교하고, 이를 메모리 사용량과 처리시간으로 나누어 비교·분석한다.

1. 실험 환경

본 실험은 펜티엄 IV 3.0Ghz 프로세서와 1GB DDR2 메모리가 장착된 마이크로소프트 윈도우 XP 운영체제에서 자바 가상 머신 1.5.0을 이용하였다. 또한 자바 가상 머신이 사용하게 될 가상 메모리의 최초 및 최대 힙(heap) 크기를 512MB로 설정하여 2차 보조기억장치의 I/O가 발생하지 않도록 하였다. 실험을 위한 입력 문서로는 SigmodRecord.xml(467KB) 파일을 사용하였으며, 접근제어 규칙과 질의의 수는 클라이언트 환경임을 고려해 최대 10개를 넘지 않도록 하였다. 표 1은 SigmodRecord.xml에 대한 자세한 정보이다.

표 1 SigmodRecordxml
Table 1 SigmodRecordxml

Size	467KB	# distinct tags	11
Depth	5	#text nodes	8,383
		# elements	10,022

2. 실험 결과 및 분석

2.1 메모리 사용량

규칙의 수와 질의의 수는 5개로 고정하였으며 프레디키트를 포함한 규칙의 경우 1개의 프레디키트를 포함하였으며 목

적노드는 프레디키트를 가지지 않도록 하였다.

그림 5와 그림 6의 결과에서 DTD 해시 구성에 필요한 메모리는 동일한 DTD를 사용하였으므로 일정하나 접근제어 규칙과 질의 등록 시 추가로 사용되는 메모리 사용량을 알 수 있도록 표시하였다.

그림 5와 그림 6처럼 제안 방법은 질의와 규칙의 등록 시 메모리 사용량을 줄이고자 최적화 과정을 거치게 되는데, 프레디키트를 포함하지 않는 규칙의 경우 규칙의 깊이에 상관없이 목적노드 즉, 1개의 노드만을 유지하며 프레디키트를 포함하는 규칙의 경우 깊이에 상관없이 목적노드, 분기노드, 프레디키트의 단말노드 즉, 3개의 노드를 유지한다. 따라서 질의나 규칙의 깊이가 길어지더라도 이에 따른 메모리 사용량의 추가 비용이 발생하지 않는다.

또한 그림 5의 처리 과정에서 사용된 메모리가 255KB를 넘지 않는 반면 [12]의 처리 과정에서 사용된 메모리가 380KB 이상임을 통해 제안 방법의 경우 최적화를 통해 질의 및 접근제어 규칙 저장을 위해 필요한 메모리를 줄였음을 확인할 수 있다.

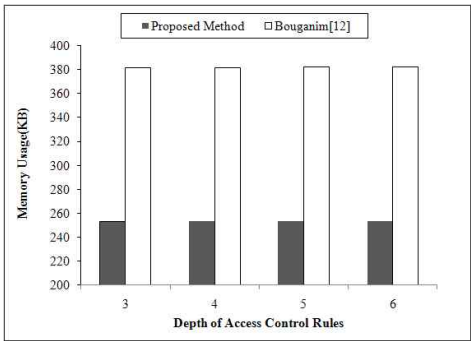


그림 5 접근제어 규칙 깊이에 대한 메모리 사용량
Fig 5 Memory usage for the depth of access control rules

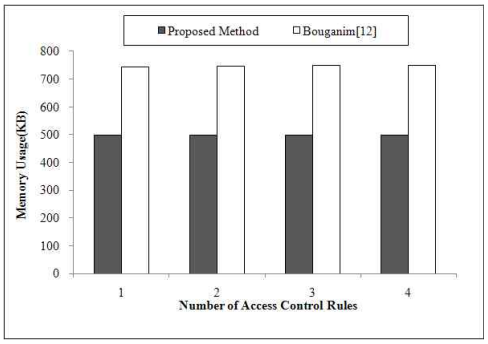


그림 6 접근제어 규칙 수에 대한 메모리 사용량
Fig 6 Memory usage for the number of access control rules

2.2 처리시간

그림 7은 제안하는 방법과 [12]의 방법에 대해 접근제어 규칙 수를 변화시켜가며 질의처리시간과 접근제어 규칙 처리시간을 비교한 결과이다. 클라이언트 환경의 경우 한 명의 사용자가 등록하는 접근제어 규칙의 개수는 많지 않으므로 접근제어 규칙의 공유는 의미가 없거나 공유로 인해 추가적인 부담이 발생할 수 있다.

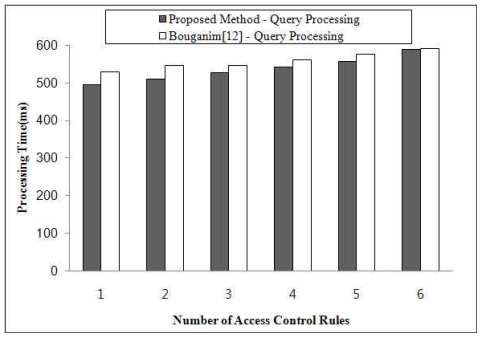


그림 7 접근제어 규칙 수에 대한 처리시간
Fig 7 Processing time for the number of access control rules

2.3 접근제어 적용에 대한 오버헤드 분석

자원제약이 있는 클라이언트에서 XML 데이터 스트림을 처리하기 위해서는 접근제어 적용 후에도 기존의 빠른 질의처리와 비교하여 성능의 차이가 매우 적어야 한다. 자원제약으로 인해 접근제어 처리를 위한 많은 공간을 소비할 수 없으며 사용자 입장에서도 보안상의 이유로 질의의 결과를 느리게 받는 것은 경우에 따라 보안 적용 전보다 안 좋게 느껴질 수 있다.

그림 8에서는 본 연구의 방법이 접근제어 처리를 위한 추가 비용을 거의 필요로 하지 않는다는 것을 보이기 위해 접근제어 적용 전과 후의 성능을 비교한 결과를 보이고 있다. 평균 메모리 사용량(그림 8)과 파일 크기 별 처리시간(그림 9)의 실험에서 모두 입력 질의와 접근제어 규칙의 수는 각각 5개로 고정된 데이터를 사용하였고, 파일 크기별 질의시간 측정 시 입력 XML 파일은 SigmodRecord.xml 파일의 크기를 변화시켜가며 측정하였다.

그림 8은 제안 기법의 접근제어 규칙 적용 시와 미적용 시의 평균 메모리 사용량을 보여주고 있다. 두 경우의 메모리 사용량 차이가 2KB 이내임을 알 수 있었다.

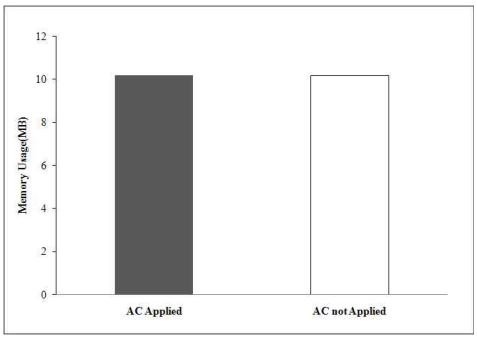


그림 8 평균 메모리 사용량
Fig 8 Average memory usage

그림 9의 파일 크기 별 처리시간의 경우, 전체적으로 접근제어 적용에 대한 부담이 그렇게 많지 않음을 확인할 수 있었다.

실험 결과에서 확인하였듯이 접근제어 적용으로 인한 추가 비용이 매우 적은 이유는 첫째, 접근제어 규칙 등록 시 사용하는 pre/post 구조가 질의 등록 시 사용하는 DTD 해시를 공유하여 사용하므로 접근제어를 위한 추가의 속성 정보를 필요로 하지 않으며 둘째, 접근제어 규칙 등록 시 최적화 과정을 거침으로써 유지해야 하는 접근제어 규칙 정보의 크기를 줄였고, 이로 인해 질의처리 과정에서 접근제어 규칙 계산 시 비교할 노드의 수를 줄였기 때문이다.

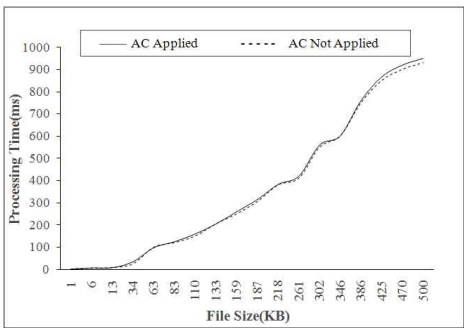


그림 9 파일 크기에 대한 처리시간
Fig 9 Processing time for file size

V. 결론

기존의 접근제어 환경은 서버에서 보안의 문제를 다루는데 집중해 왔다. 그러나 최근 급속도로 변하는 환경에서 서버중심이 아닌 자원제약적인 클라이언트 환경에서 효율적인 질의처리와 안전한 접근제어에 대한 요구가 증가하고 있다. 따라서, 본 논문은 유효한 XML 데이터 스트림 환경을 가정하고 스키마 정보를 이용하여 접근제어와 질의처리를 동시에 수행하는 그래서 자원의 사용이 효율적인 방법을 제안하였다.

특히, 자원제약적인 클라이언트 환경에서 접근제어 기술과 질의처리 기술을 동시에 실행 가능한 방법을 제안하였으며, 먼저 제한된 메모리 자원을 갖는 환경에서 안전한 질의처리를 위해 오버헤드가 거의 없는 접근제어 처리 방법을 제안하였다. 다음으로 접근제어 기능의 추가로 인한 질의처리시 오버헤드를 최소화 하고 동시에 자원 사용의 효율적인 질의처리방법을 제안하였다.

향후 연구로는 유비쿼터스 센서 네트워크 환경에서 다양하게 변경이 가능한 접근제어의 위임 및 안전한 질의처리에 대한 연구가 진행될 것이다.

참고문헌

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Mal er, F. Yergeau eXtensible Markup Language (XML) 1.0, World Wide Web Consortium (W3C), 2004.

[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), 2007.

[3] W. Lindner and J. Meier "Towards a Secure Data Str eam Management System," VLDB Workshop TEAA 2005.

[4] E. Damiani, S. Vimercati, S. Paraboachk and P. Sam arati, "Design and Implementation of Access Control Processor for XML Documents", Computer Network, pp. 59-75, 2000.

[5] E. Damiani, S. Vimercati, S. Paraboachk and P. Sa marati, "A Fine-grained Access Control System for XML Documents", ACM Trans Information and System Sec., Vol5, No2, pp. 169-202, May 2002.

[6] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti, "Specifying and Enforcing Access Control Policies for XML Document Sources", WWW Journal, Vol3, No3, pp. 139-151, 2000.

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

[8] J. Cai, C. K. Poon, "OrdPathX: Supporting Two Dim ensions of Node Insertion in XML Data", in proceeding of DEXA, pp. 332-339, 2009.

[9] W. Liang, A. Takahashi, H. Yokota, “A Low-Storag e-Consumption XML Labeling Method for Efficient Structural Information Extraction”, in proceeding of DEXA, pp. 7-22, 2009.
[10] M. Murata, A. Tozawa, and M. Kudo, “XML Access Control Using Static Analysis”, in proceedings of ACM Conference on Computer and Communications Security, pp. 73-84, 2003.
[11] B. Luo, D. W. Lee, W. C. Lee, and P. Liu, "Qfilter: Fine-grained Run-Time XML Access Control via NFA-based Query Rewriting," CIKM'04, pp. 543-552, 2004.
[12] L. Bouganim, F. D. Ngoc, and P. Pucheral, "Client-based access control management for XML documents," VLDB, pp. 84-95, 2004.
[13] F. Rabitti, E. Bertino, W. Kim and D. Woelk, "A Mo del of Authorization for Next-Generation Database Systems", ACM Transaction on Database Systems, 126(1), pp. 88-131, March 1991.
[14] E. Damiani, M. Fansi, A. Gabillon, S. Marrara, “A general approach to securely querying XML”, Computer Standards & Interfaces 30(6), pp. 379-389, 2008.

저 자 소 개

안 동 찬
2011 : 서강대학교 컴퓨터공학과 공학박사
현 재 : 신안산대학교 멀티미디어컨텐츠과 교수
관심분야 : 이동데이터베이스, 의료정보 보호, 접근제어
Email : channy@sau.ac.kr