유효한 XML 스트림에 대한 효율적이고 안전한 질의 처리

(An Efficient and Secure Query Processing on Valid XML Streams)

변 창 우 † 안 은 주 †† 박 석 †††
(Changwoo Byun) (Eunju An) (Seog Park)

요 약 기존 서버에서 담당하던 접근제어를 서버에 비해 제약을 갖는 클라이언트에서 처리하고자 하는 요구가 생겨남에 따라 자원 제약을 갖는 환경에서 효율적이고 안전하게 질의를 처리할 수 있는 방법이 필요하게 되었다. 기존의 접근제어 연구는 안전성에 초점을 맞추어 왔기 때문에 효율성 측면에 대한 고려가 적었으며, 최근 스트림 환경에서 보안 문제가 대두되면서 접근제어를 포함한 보안 측면의 연구가 시작되었다. 본 논문은 XML 데이터 스트림을 PDA나 휴대용 단말기와 같은 자원의 제약이 있는 클라이언트에서 안전하고 효율적으로 다루기 위한 방법을 제안한다. 본 연구는 유효한 XML 스트림 환경에서 첫째로 한정된 메모리 내에서 안전한 결과를 내기 위해 오버헤드가 매우 적은 접근제어 처리 방법을 제안하고 있으며, 둘째로 접근제어 추가로 인한 오버헤드를 상쇄시키기 위해 처리 단계 마다 최적화가 가능한 부분들을 찾아 성능을 개선하는 방법을 제안한다. 실험을 통해 제안하는 방법의 우수성을 분석한다.

키워드 : XML 데이터, 자원제약 클라이언트, XML 접근제어, XML 질의 처리

Abstract As demands intending to treat an access control on a client side that was conventionally controlled at a server are surged, it needs a way to treat query processing in effective and secure manners in an environment that has limited resources. Because the access control having been previously performed was only focused on safety, there was little effort to consider the access control in terms of efficiency. Researches about security including access control are started as the security issues are cropped up in a recent stream environment. This paper proposes a method for efficient and secure query processing of XML data streams like a PDA and a portable terminal at the client that is in limited resources. Specifically, this study suggests (1) an access control processing that possesses small overhead for attaining a secure result in a limited memory and (2) a way to enhance the performance, finding the parts being capable of optimizing in each processing step for offsetting the overhead caused by an addition of the access control processing. Superiority of the new method was analyzed by experiment.

Key words : XML, resource limited client, XML Access Control, XML Query Processing

· 이 연구는 『2단계 BK21 사업』 및 한국과학재단 특정기초연구(R01-2006-000-10609-0) 지원으로 수행되었음

† 정 회 원 : 인하공업전문대학 컴퓨터시스템과 교수
cwbyun@inhatc.ac.kr
†† 학생회원 : 서강대학교 컴퓨터공학과
dksdw@dblab.sogang.ac.kr
††† 종신회원 : 서강대학교 컴퓨터공학과 교수
spark@sogang.ac.kr

논문접수 : 2008년 9월 19일
심사완료 : 2009년 2월 23일

정보과학회논문지: 데이타베이스 제36권 제3호(2009.6)

1. 서 론

XML[1]이 인터넷 환경에서 데이터 저장과 전송을 위한 표준으로서 부각되면서 많은 사용자 및 응용 프로그램들이 XML 데이터 기반의 정보의 분산과 공유에 대한 요구가 날로 증가함에 따라, 효율적이고 안전하게 XML 데이터에 접근할 수 있는 방법이 매우 중요한 이슈로 부각하고 있다. 특히, XML 문서의 데이터 스트림을 PDA나 휴대용 단말기와 같이 자원의 제약이 있는 클라이언트에서에서 처리하고자 하는 요구가 생겨남에 따라 자원 제약을 고려하면서 클라이언트가 접근할 수 있는 데이터만을 처리할 수 있는 효율적이고 안전한 질

의 처리의 필요성이 대두되었다.

그림 1의 상단은 병원에서 의사들이 휴대용 단말기를 통해 환자의 정보를 알아보는 그림이다. 환자들은 각각 담당 의사가 정해져 있고, 의사의 전공 분야에 따라 여러 명의 담당 의사가 있을 수도 있다. 의사들은 기본적으로 환자들의 정보를 볼 수 있지만 자신이 담당하지 않는 환자의 질병에 관한 정보는 볼 수 없다. 환자 입장에서 정신병이나 사회생활에 어려움을 겪을 수 있는 질병 등은 꼭 필요한 사람들을 제외하고는 볼 수 없어야 한다. 그러나 서버는 정보를 제공받는 클라이언트 모두에게 구별된 데이터를 전송해 주지 못한다. 이는 다양한 사용자에 맞는 접근제어 규칙을 서버에서 모두 관리하기 어려우며, 데이터를 지속적으로 보내야 하는 스트림 환경에서 이러한 여러 가지 경우의 결과를 모두 브로드캐스트(broadcast) 하는데 상당한 비용이 소비되기 때문이다. 따라서 서버는 모든 정보를 브로드캐스트 하지만 클라이언트에서 각자 접근이 허가된 내용만을 읽어 사용자의 질의에 응답해야 한다[3].

그림 1의 하단은 서버가 환자들로부터 받는 정보에 대한 전체적인 DTD(Document Type Definition) 구조와 의사 A와 의사 B의 접근제어 규칙의 예를 보여주고 있다. XML 문서에 대한 접근제어 규칙은 질의와 마찬가지로 XPath 표현식[2]으로 나타낼 수 있으며 상단의 그림과 같이 개인의 단말기에 이러한 접근제어 규칙이 탑재된다.

접근제어를 고려한 질의 처리의 전통적인 방법은 서버 측면에서 사용자의 질의가 입력되면 접근제어 기술을 통해 사용자의 질의 영역이 그 사용자에 대해 접근 가능한 영역인가를 확인하고, 이를 통과하면 질의 처리기를 통해 질의의 결과를 출력하였다. 이와 같이 접근제어와 질의 처리를 독립적으로 수행하는 방법은 자원 제약이 있는 클라이언트 측면에서 수행하기에는 접근제어 엔진과 질의 처리 엔진을 탑재해야 하기 때문에 그대로 적용하는데 무리가 있다.

본 논문은 자원 제약이 있는 클라이언트 측면에서 접근제어 기술과 질의 처리 기술을 동시에 수행하는 방법을 제안한다. 첫째로 한정된 메모리 내에서 안전한 결과를 내기 위해 부담이 매우 적은 접근제어 처리 방법을 제안하고 있으며, 둘째로 접근제어 처리 과정에서 질의 처리를 동시에 수행함으로써 접근제어 추가로 인한 성능 부담을 줄이고자 한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 본 연구와 관련된 기존 연구들을 살펴보며, 이러한 연구 내용들의 특징 및 문제점을 분석한다. 3장에서는 제안하고자 하는 시스템의 구조를 소개하고, 2장에서 정의한 문제점들을 해결하기 위한 방법을 전처리 과정과 질의 처

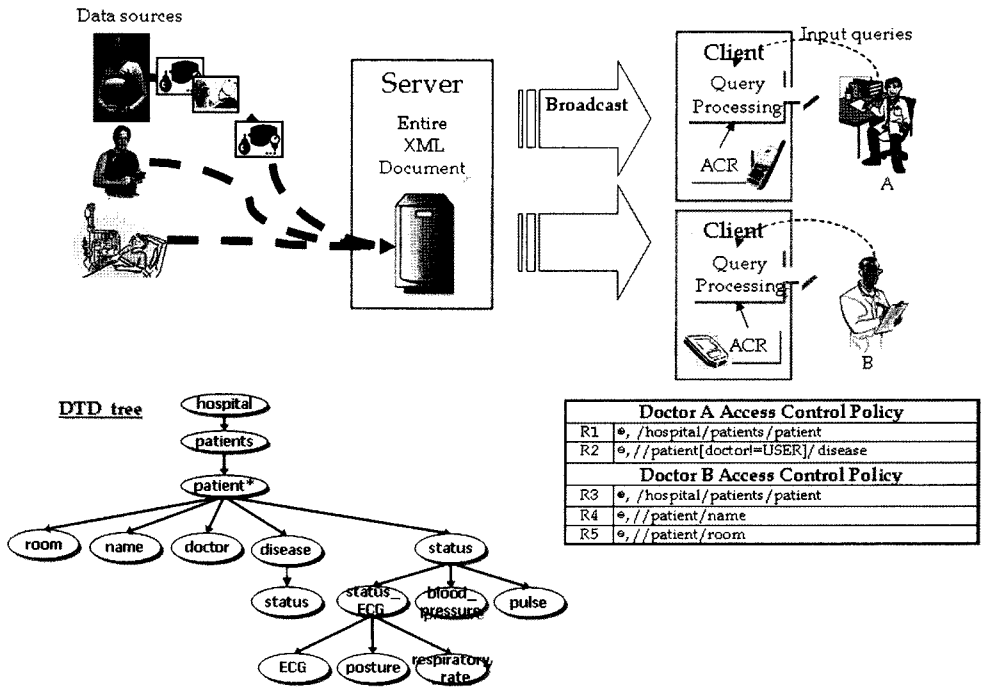

그림 1 자원 제약을 갖는 클라이언트에서 XML 데이터 스트림의 접근제어 처리 예제

리 과정으로 나누어 자세히 설명한다. 4장에서는 실험을 통해 제안 방법의 우수성을 보이며 5장에서 결론 및 향후 연구 방향을 간략히 기술한다.

2. 관련 연구

본 장에서는 XML 데이터 환경에서의 접근제어를 통한 질의 처리 연구를 살펴보고, 기존 연구와 본 연구의 차이점을 설명한다.

뷰 기반의 XML 접근제어 수행 메커니즘. 전통적인 XML 접근제어 수행 메커니즘[4-7]은 뷰-기반 수행 메커니즘이다. 즉, 사용자 별 접근제어 규칙들에 의해 뷰를 생성하고 생성된 뷰 기반으로 사용자의 접근을 제어하면서 질의처리를 한다. 트리 레이블링 기법으로 뷰를 계산할 수 있는 유용한 알고리즘들이 제시되었지만, 뷰 생성의 높은 비용 및 뷰 유지 비용의 문제점을 가지고 있다. 또한, 사용자 증가에 따른 확장성 문제를 갖고 있다.

접근제어를 고려한 질의 재작성. 뷰 기반의 문제점을 극복하기 위해 Murata[8]은 접근 제어 정책을 만족하지 못하는 질의를 사전에 걸러낼 수 있는 필터링 방법을 제안하였다. Jeon[9]은 XML 문서에 대한 모든 접근제어 규칙에 대한 구조 요약 정보를 에지로 구성하고, 각 노드에 접근제어 규칙을 기록한 XML 접근제어 트리(XACT)를 두어 사용자의 질의를 XACT 접근제어 정보들을 비교하여 변형된 안전한 질의로 변형하는 방법을 제안하였다. XACT는 질의를 제출한 사용자뿐만 아니라 문서에 대한 모든 접근제어 규칙을 갖고 있기 때문에 불필요한 계산 시간을 갖고 있다. Luo[10]은 [8]의 방법론을 한층 발전시켜 접근제어 정책의 일부분만을 만족하고 있는 질의에 대해 공유 비결정 유한 오토마타(shared NFA) 방법을 이용해 재작성하는 기법을 제안하였다. 그러나, 공유 비결정 유한 오토마타 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 안고 있다. 또한, 부정확한 질의를 재작성하게 만들게 된다. SQ-Filter 시스템[11]은 유효한(valid) XML 문서 환경에서 DTD의 메타데이타를 이용하여 사용자의 질의에 해당하는 접근제어 규칙들만을 추출하고 질의와 추출된 접근제어 규칙들을 이용하여 사용자의 질의를 안전한 질의로 재작성하는 방법을 제안하였다. Fan[12,13]은 사용자에게 가상 뷰 DTD를 제공하고 이것을 이용하여 질의 재작성하는 방법을 제안하였다. 그러나 사용자 질의별 가상 뷰가 아닌 사용자별 가상 뷰를 제공하고 있기 때문에 질의 재작성 알고리즘은 뷰의 불필요한 부분을 이용하게 된다.

접근제어 기술을 통합한 안전한 질의 처리. 클라이언트 기반 XML 문서 스트림 접근제어 연구[14]는 질의 재작성을 피하고 접근제어 규칙과 질의 계산을 동시에 수행함으로써 현재 입력된 데이터에 대한 접근 여부를 판단한다. 즉, Luo[10] 방법처럼 접근제어 규칙들에 대한 오토마타(Access Rules Automata: ARA)를 생성하고 사용자 질의를 받아 질의 재작성을 하는 것이 아니라 사용자의 질의에 해당되는 XML 문서를 서버로부터 받아 ARA에 통과시킴으로써 최종적인 사용자 질의에 대한 최종적인 안전한 XML 문서를 얻는 기법이다. 추가로 XML 문서에서 ARA에 불필요한 순회과정을 거치는 부분을 건너뛰기 위해 Skip 인덱스 기법으로 제안하고 있다. 결국 접근제어 규칙에 맞추어 사용자의 질의를 재작성 할 필요가 없으며, 접근이 불허된 데이터에 대한 접근이 불가능하므로 안전한 질의 결과를 내보내게 된다. 그러나 이 방법은 사전에 사용자 질의에 대한 질의 처리를 서버에서 한 번 처리하기 때문에 사용자와 서버 간의 질의 처리 채널을 개별적으로 두어야 하기 때문에 서버 부담을 그대로 간직하고 있으며 브로드캐스트 방식의 모바일 환경에 적용하기에는 무리가 있다.

관련 연구와의 차이점. 본 연구는 서버에서는 XML 문서를 스트림 방식으로 브로드캐스트하고 휴대용 단말기와 같은 자원 제약을 가지는 클라이언트 환경에서의 안전한 질의 처리 기법을 제안하고 있다. 접근제어를 고려한 질의 재작성 방법과의 차이점은 질의 처리 엔진을 따로 클라이언트에 두지 않는다. 기존의 접근제어 기술을 통합한 안전한 질의 처리 기법과의 차이점은 서버와 클라이언트 간의 개별적인 채널을 두지 않고 서버에서는 XML 문서를 브로드캐스팅하기 때문에 서버 부하가 없으며 클라이언트에서 질의 처리와 접근제어 처리를 동시에 수행한다.

3. XML 스트림 데이터에 대한 효율적이고 안전한 질의 처리 기법

3.1 시스템 구조

그림 2는 본 연구에서 제안하는 시스템의 구조를 보여주고 있다.

서버: 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 끊

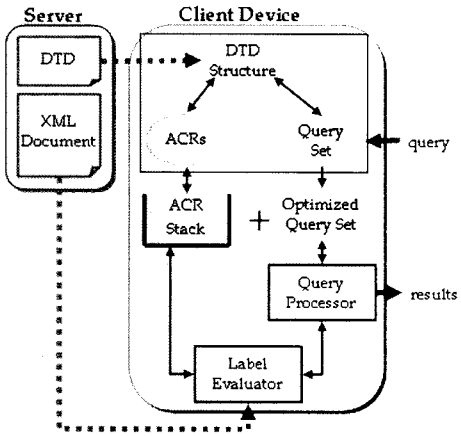

그림 2 시스템 구조

임없이 데이터를 받아 계속해서 갱신 된다.

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

접근제어 규칙(ACRs): 클라이언트 장치는 기본적으로 사용자에 맞는 접근제어 규칙을 탑재하고 있으며 접근제어 규칙이 탑재되는 경로는 다양할 것이다.

질의 셋, 최적화 질의 셋(query Set, optimized query set): 질의는 사용자로부터 입력 받는다. 클라이언트는 사용자로부터 입력 받은 다수의 질의를 저장함으로써 사용자가 원하는 결과를 내보내도록 한다. 이 때 빠른 질의 처리에 적합하도록 최적화된 구조로 변환하여 가지고 있게 된다.

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

레이블 계산기(label evaluator): 질의 및 접근제어 규칙 처리 시 입력 데이터와의 비교를 위해 입력 데이터의 레이블을 계산한다. 레이블 계산기의 적용으로 인해 빠른 계산 및 질의와 접근제어 규칙 유지를 위한 메모리 감소 효과를 가져 올 수 있다.

질의 처리기(query processor): 질의 처리기는 질의와 접근제어 규칙을 동시에 계산한다. 결론적으로 사용자가 원하는 결과를 계산하고 접근제어 스택의 값과 비교하여 사용자에게 결과를 내보낼지 결정한다.

3.2 접근제어 정책

계층적 데이터 모델(예를 들어, Object-Oriented Data Model, XML Data Model 등)[15]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 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 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.

3.3 전처리 과정

3.3.1 해시 구성

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.

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

Tag-name	Pre	Post	Parent
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 그림 1의 DTD 트리의 PRE/POST 해시

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

3.3.2 질의 및 접근제어 규칙 등록

접근제어 규칙 및 사용자 질의 등록은 컴파일 타임 시 DTD 트리에 대한 PRE/POST 메타데이타를 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.

자세한 설명에 앞서, 수식에 표현되는 용어를 다음과 같이 가정한다. 임의의 XPath 경로식에서 두 노드 x와 y에 대한 (pre, post) 쌍을 각각 (PREx, POSTx), (PREy, POSTy)라 하자.

정의 1. [ANCESTOR 관계] 임의의 XPath 경로식에서 두 노드 x와 y에 대해 다음의 조건을 만족하면 x는 y에 대해 ANCESTOR(PARENT 관계도 포함) 관계이다.

$$PREx < PREy, POSTx > POSTy \quad (1)$$

정의 2. [FOLLOWING 관계] 임의의 XPath 경로식에서 두 노드 x와 y에 대해 다음의 조건을 만족하면 x는 y에 대해 FOLLOWING 관계이다.

$$PREx > PREy, POSTx > POSTy \quad (2)$$

정의 3. [PRECEDING 관계] 임의의 XPath 경로식에서 두 노드 x와 y에 대해 다음의 조건을 만족하면 x는 y에 대해 PRECEDING 관계이다.

$$PREx < PREy, POSTx < POSTy \quad (3)$$

정의 4. [DESCENDANT 관계] 임의의 XPath 경로식에서 두 노드 x와 y에 대해 다음의 조건을 만족하면 x는 y에 대해 DESCENDANT(CHILD 관계도 포함) 관계이다.

$$PREx > PREy, POSTx < POSTy \quad (4)$$

다음 P1과 같은 XPath 경로식을 (pre, post) 경로식으로 변환하는 과정을 예로 들어보자.

P1:
/patient/status/status_ECG[ECG>50]/respiratory_rate

먼저 P1의 목적노드인 respiratory_rate에 대한 (pre, post) 값을 그림 3의 DTD 해시에서 탐색하면 (12, 7)이라는 값을 얻을 수 있다. 다음으로 status_ECG의 (pre, post) 값 (9, 8)을 찾고, 이 값을 respiratory_rate의 (pre, post) 값인 (12, 7)과 비교한다. Status_ECG의 pre 값은 respiratory_rate의 pre 값 보다 작고, status_ECG의 post 값은 respiratory_rate의 post 값보다 크다(식 (1)에 의한 ANCESTOR 관계). 따라서 탐색된 값은 이전에 탐색한 respiratory_rate의 조상(부모 포함)이므로 이를 respiratory_rate와 연결한다. status_ECG는 프레디키트를 포함하고 있으므로 다음으로는 ECG의 (pre, post) 값 (10, 5)을 구하여 status_ECG 와의 관계를 확인한 후 status_ECG의 프레디키트로 연결한다. 이제 DTD에서 status의 (pre, post) 값을 탐색하면 두 개의 쌍 (7, 3), (8, 11)이 구해진다. 만약 patient 노드 대신 "*" 노드 혹은 //status 였다면 둘 다 (pre, post) 쌍으로 인정되겠지만, (7, 3)은 (9, 8)을 기준으로 형제(preceding) 관계이고, (8, 11)은 조상(ancestor) 관계이다. 따라서 (8, 11)이 status의 위치에 오게 되고 status_ECG (9, 8)과 연결된다. 즉, 임의의 노드의 (pre, post)

쌍이 집합인 경우, 불필요한 (pre, post) 쌍을 제거하는 핵심 개념은 정의 1의 ANCESTOR 관계를 이용하는 것이다. 동일하게 patient의 (pre, post) 값 (2, 12)을 계산하면 P1의 경로 상에 나타나는 모든 노드에 대한 (pre, post) 정보를 유지하게 된다. 이 과정을 거치면 DTD 형식에 맞지 않는 표현식은 걸러지고 올바른 표현식만 남게 된다. 그림 4는 P1에 대한 (pre, post) 경로식을 보여주고 있다.

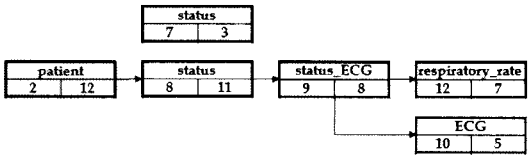

그림 4 P1 XPath 경로식에 대한 변환된 Pre/Post 경로

DTD 해시 메타데이타로부터 임의의 노드에 대한 (pre, post) 값을 알면 DTD 상의 정확한 위치를 파악할 수 있다. 이는 곧 해당 노드에 대한 루트(root) 부터의 정확한 경로를 알 수 있음을 뜻한다. 따라서 XPath 경로 표현식에 나타나는 모든 노드의 정보를 유지하는 대신 몇 개의 필요한 노드에 대한 정보만 유지하여도 XPath 경로를 알아낼 수 있다. 따라서 XPath 경로 변환을 통해 (pre, post) 값을 가지게 된 DTD에 맞는 표현식들에 대한 최적화 과정을 거친다.

절대 경로(full path) XPath 경로식: 이 경우 DTD 상에서 질의에 해당하는 경로는 유일하며, 따라서 목적 노드의 (pre, post) 정보만으로 경로를 결정할 수 있다. P1이 여기에 해당된다.

Descendant-or-Self를 포함한 XPath 경로식: 이러한 경로식은 절대 경로 경로식으로 변환 가능하며 변환된 절대 경로 경로식이 여러 개 존재할 수 있다. 절대 경로 경로식은 위의 조건을 만족하므로 각각의 변환된 절대 경로 경로식에 대한 목적 노드의 (pre, post) 정보로 경로를 표현할 수 있다.

프레디키트를 포함한 XPath 경로식: 이러한 경우는 경로식을 여러 개의 절대 경로로 나누어 생각할 수 있다. 즉, 프레디키트를 제외한 절대 경로와 프레디키트에 대한 절대 경로로 나눌 수 있다. 따라서 나누어진 각각의 경로에 대한 목적 노드의 (pre, post) 정보를 유지해야 하며 프레디키트의 단말 노드와 경로식의 목적 노드가 동일한 질의에 속해 있다는 관계를 파악하기 위해 질의의 분기 노드에 대한 정보가 함께 필요하다. P1이 여기에 해당된다.

그림 5는 P1 경로식에 대한 패턴 분석 결과 즉 절대 경로이면서 프레디키트를 포함한 XPath 경로식에 대한 최적화된 Pre/Post 경로식을 보여주고 있다.

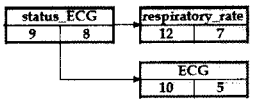

그림 5 P1에 대한 최적화된 Pre/Post 경로

위의 과정을 거쳐 등록된 질의와 접근제어 규칙들은:

- Pre/Post 정보를 바탕으로 질의 계산 시 입력 데이터와 빠른 비교가 가능하며,
- 최적화를 통해 질의 및 접근제어 규칙 유지에 필요한 메모리 사용량과 질의 계산 시 필요한 비교 횟수를 줄일 수 있다.

이제 DTD 해시 등록, 질의 및 접근제어 규칙 등록의 전처리 과정이 끝났으니 본격적인 질의 처리 과정에 들어간다. 다음 절에서는 질의 처리 과정을 설명한다.

3.4 질의 처리(query-processing) 과정

질의 처리 과정에서는 전처리 과정에서 등록된 정보를 이용하여 접근제어 처리와 질의 처리를 동시에 수행한다.

3.4.1 입력 XML 스트림 데이터 정보 계산

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 경로 상에 나타나는 모든 노드를 유지하고 있어야 한다. 예를 들어, 그림 5에서의 P1의 status에 해당하는 (7, 3), (8, 11) 중 현재 입력된 데이터에 해당하는 값을 결정하기 위해 질의에서 patient에 해당하는 데이터가 이전에 입력되었는지 알고 있어야 한다. 만약 patient의 match_flag가 TRUE이면 현재 입력된 데이터의 (pre, post) 값은 (8, 11)이 될 것이고, match_flag가 FALSE 이면 현재 입력된 데이터의 (pre, post) 값은 (7, 3)이다. 이러한 방법을 통해 입력된 데이터의 (pre, post) 값을 결정하려면 질의 경로에 나타나는 모든 노드를 유지해야 하므로 전처리 과정에서 설명했던 질의 최적화가 불가능해진다. 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 것 보다 메모리 활용 면에서 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 루트 노드이다. 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_data의 값을 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 가능해 진다.

3.4.2 접근제어 규칙 계산

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 그림 5와 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.

- 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.

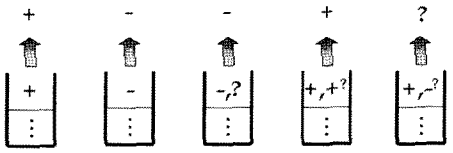

그림 6 접근제어 스택의 계산

- 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.

3.4.3 접근제어를 추가한 질의 계산

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 잠재적

으로 - 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.

자세한 질의 계산 방법은 생략하고 접근제어 규칙과 함께 어떻게 계산되는지 예제를 통해 살펴보도록 하자.

3.4.4 예제 시나리오

서버는 그림 7의 DTD와 XML 문서를 브로드캐스트 한다. 먼저 클라이언트는 DTD를 전송 받아 DTD 해시를 구성한다.

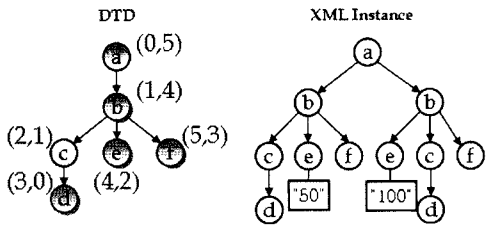

그림 7 DTD와 XML 문서 예제

그림 8에서처럼 사용자가 질의(Q2)를 입력한 후 클라이언트는 위와 같은 접근제어 규칙과 질의에 대해 pre/post 값으로 변환 후 최적화한 결과를 등록한다.

Q2 : /a/b[e = "50"]/c/d
R2: (-) //b/f
R3 : (+) //a
R4 : (-) /a/b[e = "100"]

질의 처리 과정은 다음과 같다.

- 루트 엘리먼트인 a가 입력되면 입력데이터 레이블 계산과정에 의해 현재 입력 데이터의 (pre, post) 값은 (0, 5)가 되고 R3에서 동일한 값을 가지는 노드를 찾을 수 있다. 따라서 접근제어 스택에 + 값을 입력한다. 이제부터는 "전파허용" 정책에 따라 다음 접근제어 규칙이 입력되기 전까지 입력되는 모든 노드에 접근 가능하다.
- b가 입력되면 DTD 해시에서 b의 (pre, post) 값 (1,4)을 찾고 현재 입력 데이터의 정보인 (0, 5)의 자식임을 확인한다.

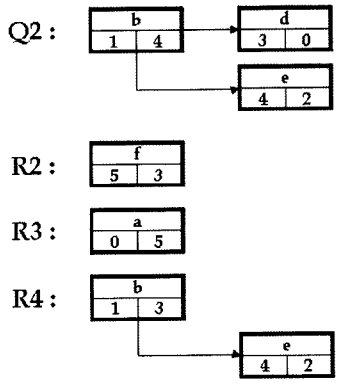

그림 8 등록질의 및 접근제어 규칙

- c가 입력되면 현재 입력 데이터 정보를 (2, 1)로 바꾸고 d가 입력되면 다시 (3, 0)으로 변경한다. 이때 Q2에서 일치하는 노드를 발견한다. 이제 Q2는 프레디키트를 제외한 모든 조건을 만족하므로 d는 잠재적으로 질의의 결과가 되고 이를 Q2에 대한 임시 질의 결과로 저장한다.
- d의 끝 엘리먼트가 입력되면 현재 입력 데이터 정보를 d의 부모인 (2, 1)로 바꾸고 c의 끝 엘리먼트가 입력되면 동일하게 계산한다.
- e가 입력되면 현재 입력 데이터 정보를 (4, 2)로 바꾸고 Q2와 R4에서 일치하는 노드를 발견하게 되는데 이들은 프레디키트 노드이므로 이들의 값을 확인한다. 이 때, Q2에 대한 처리가 끝났으므로 임시로 저장하고 있던 결과를 질의의 결과로 내보낸다. R4에 대해서는 분기 노드인 e에 대한 검사가 해결되지 않았으므로 접근제어 스택에 - 값을 입력 할 수 없다.
- e의 끝 엘리먼트가 입력되면 현재 입력 데이터 정보를 e의 부모인 (1, 4)로 바꾼다.
- f가 입력되면 현재 입력 데이터 정보는 (5, 3)이 되고 R2에서 일치하는 노드를 찾을 수 있다. 따라서 접근제어 스택에 R2에 해당하는 - 값을 입력 한다. "전파허용" 정책에 따라 이제부터 입력되는 모든 데이터는 접근할 수 없고, 이는 데이터가 입력되지 않은 것처럼 보여지며 따라서 질의 계산을 수행하지 않는다. 또한 이후에 입력되는 상태는 모두 -로 변환한다.
- f의 끝 엘리먼트가 차례로 입력되면 접근제어 스택에서 R2의 상태를 출력한다. 이제 접근제어 스택의 top에는 + 값이 존재하므로 입력되는 데이터에 접근 가능하며 질의 계산도 계속 된다.

그림 9는 그림 7의 XML 문서에 대한 접근제어 계산 시 접근제어 스택의 변화 과정을 나타내고 있다.

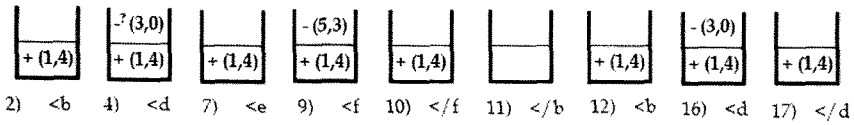

그림 9 XML 문서 입력에 따른 접근제어 스택의 변화

4. 실험 및 분석

본 장에서는 제안 기법과 관련 연구로써 클라이언트 기반의 XML 접근제어 방법[14]의 성능을 비교하고 제안 기법에서 접근제어 적용 전후의 성능을 비교하고, 이를 메모리 사용량과 처리 시간으로 나누어 비교·분석한다.

4.1 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 SigmodRecord.xml

SigmodRecord.xml			
size	467KB	#distinct tags	11
depth	5	#text nodes	8383
		#elements	10022

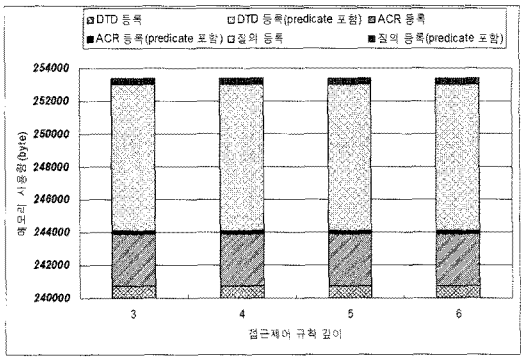

그림 10 규칙 깊이 별 메모리 사용량(제안 기법)

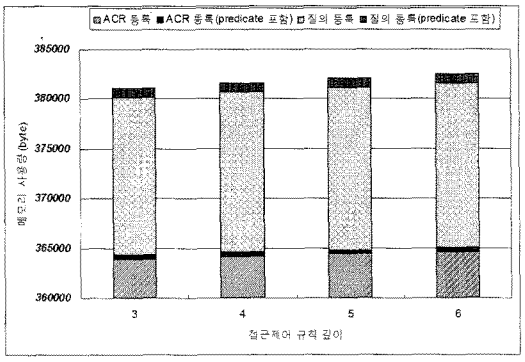

그림 11 규칙 깊이 별 메모리 사용량(관련 연구[14])

4.2 실험 결과 및 분석

메모리 사용량

규칙의 수와 질의의 수는 5개로 고정하였으며 프레디키트를 포함한 규칙의 경우 1개의 프레디키트를 포함하였으며 목적노드는 프레디키트를 가지지 않도록 하였다.

그림 10의 결과에서 DTD 해시 구성에 필요한 메모리는 동일한 DTD를 사용하였으므로 일정하나 접근제어 규칙과 질의 등록 시 추가로 사용되는 메모리 사용량을 알 수 있도록 표시하였다.

그림 10처럼 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, predicate의 단말노드 즉, 3개의 노드를 유지한다. 따라서 질의나 규칙의 깊이가 길어지더라도 이에 따른 메모리 사용량의 추가 비용이 발생하지 않는다.

그림 11은 규칙의 깊이가 깊어질수록 메모리 사용량이 증가함을 볼 수 있다. 또한 그림 9의 전처리 과정에서 사용된 메모리가 255Kb를 넘지 않는 반면 그림 11의 전처리 과정에서 사용된 메모리가 380Kb 이상임을 통해 제안 방법의 경우 최적화를 통해 질의 및 접근제어 규칙 저장을 위해 필요한 메모리를 줄였음을 확인할 수 있다.

수행 시간

그림 12는 두 기법의 접근제어 규칙 처리를 포함한 질의 처리 시간을 접근제어 규칙의 수를 변화시켜 가며 비교한 결과이다. 접근제어 규칙의 개수가 증가할수록 성능의 차이가 줄어드는데 그 이유는 [14]에서 이용한 접근제어 규칙들에 대한 오토마타(ARA)는 XPath로 표

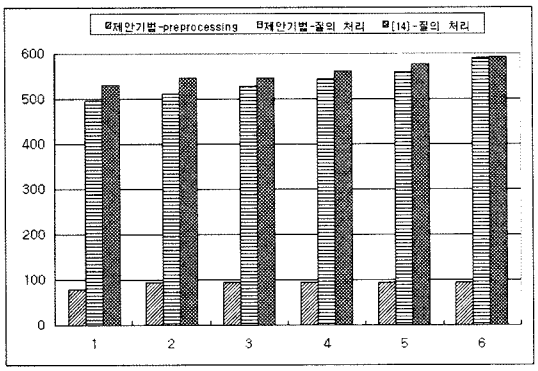

그림 12 규칙 개수 별 처리 시간

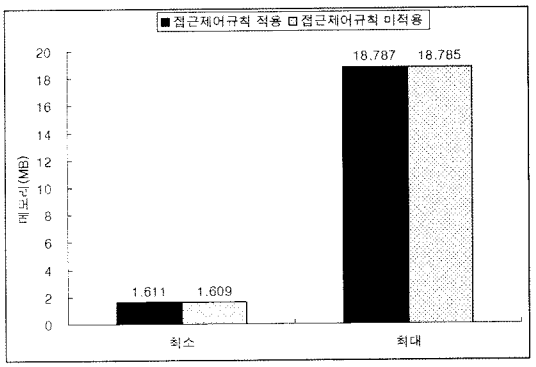

그림 13 규칙 개수 별 메모리 사용량

현된 접근제어 규칙들에 대해 공통된 경로를 공유하는데 실험 데이터의 경우처럼 경로 깊이가 깊지 않은 관계로 질의의 공유되는 부분이 많기 때문이다. 그러나 경로의 공유는 접근제어 규칙의 양이 매우 많은 경우 그 장점이 드러나며 클라이언트 환경의 경우 한 명의 사용자가 등록하는 접근제어 규칙의 개수는 많지 않으므로 접근제어 규칙의 공유는 의미가 없거나 공유로 인해 추가적인 부담이 발생할 수 있다.

본 연구의 접근제어 적용의 부담 분석

메모리 제약이 있는 클라이언트에서 XML 데이터 스트림을 처리하기 위해서는 접근제어 적용 후에도 기존의 빠른 질의 처리와 비교하여 성능의 차이가 매우 적어야 한다. 자원 제약으로 인해 접근제어 처리를 위해 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 13, 그림 14, 그림 15에서는 본 연구의 방법이 접근제어 처리를 위한 추가 비용을 거의 필요로 하지 않는다는 것을 보이기 위해 접근제어 적용 전과 후의 성능을 비교한 결과를 보이고 있다. 메모리 사용량(그림 13), 평균 처리 시간(그림 14), 그리고 파일 크기 별 처리시간(그림 15)의 실험에서 모두 입력 질의와 접근제어 규칙의 수는 각각 5개로 고정된 데이터를 사용하였고, 파일 크기별 질의시간 측정 시 입력 XML 파일은 Sigmod-Record.xml 파일의 크기를 변화시켜가며 측정하였다.

그림 13은 제안 기법의 접근제어 규칙 적용 시와 미적용 시의 메모리 사용량을 보여주고 있다. 최소와 최대 모두 두 경우의 메모리 사용량 차이가 2KB 정도임을 알 수 있다.

그림 14의 결과를 보면 전처리 과정의 경우 평균적으로 접근제어 적용 전과 후가 15msec 정도 차이를 가지며 질의 처리 과정의 경우 평균 100msec 보다 적은 차이를 보이고 있다. 또한 그림 15의 파일 크기 별 처리

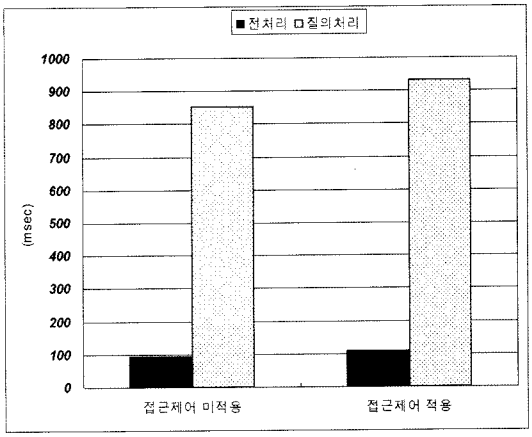

그림 14 평균 처리 시간

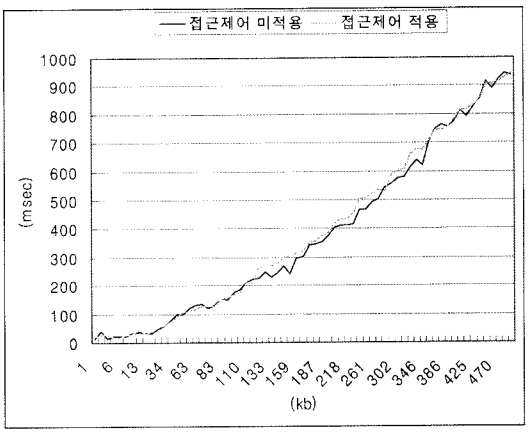

그림 15 파일 크기 별 처리 시간

시간의 결과의 경우, 전체적으로 접근제어 적용에 대한 부담이 그렇게 많지 않음을 확인할 수 있다.

실험 결과에서 확인하였듯이 접근제어 적용으로 인한 추가 비용이 매우 적은 이유는 :

- 우선 전처리 시 접근제어 규칙 등록 시 사용하는 pre/post 구조가 질의 등록 시 사용하는 DTD 해시를 공유하여 사용하므로 접근제어를 위한 추가의 속성 정보를 필요로 하지 않으며,
- 접근제어 규칙 등록 시 최적화 과정을 거침으로써 유지해야 하는 접근제어 규칙 정보의 크기를 줄였고, 이로 인해 질의 처리 과정에서 접근제어 규칙 계산 시 비교할 노드의 수를 줄였기 때문이다.

5. 결 론

인터넷 환경의 보편화와 정보통신의 발달로 인해 많은 정보를 포함한 방대한 양의 데이터가 다뤄지고 있고 이에 따른 보안의 중요성도 날로 증가하고 있으며 개인별로 알맞은 정보 제공을 위해 보안의 기능이 클라이언트로 분산되고 있다. 본 논문은 아직까지 보안의 연구가 미진한 XML 데이터 스트림을 자원의 제약을 갖는 클라이언트에서 처리하기 위한 방법을 제안하였다.

제안하는 기법들의 특징은 다음과 같다.

첫째, 접근제어 처리 시 질의 처리와 속성 정보를 공유할 수 있는 부분(DTD 해시)을 찾아내어 추가적인 비용을 줄였다.

둘째, 입력 데이터의 정보를 계산하여 DTD 상의 정확한 위치를 찾아내고 질의 계산에 필요한 노드를 줄임으로써 질의 최적화를 가능하게 했다.

셋째, XPath 경로 최적화를 통해 전처리 비용을 줄이고 질의 처리 성능을 개선하였다.

이러한 특징들은 자원 제약이 많은 클라이언트 환경에서 기존의 질의 처리보다 안전한 질의 처리를 할 수 있도록 해준다.

본 연구는 유효한 XML 데이터 환경을 가정하고 있어 스키마 정보를 이용하여 접근제어와 질의 처리를 동시에 수행하는 방법을 제안하고 있다. 유효한 XML 데이터 환경에서 DTD를 표현하는 데는 일반적으로 두 가지 방법이 사용된다. 본 논문에서처럼 트리 행태로 표현하는 방법과 DTD에는 엘리먼트를 한 번만 표현해야 하는 규칙 때문에 하나의 엘리먼트에 대한 여러 부모를 가지는 경우가 생겨 DAG(Direct Acyclic Graph) 형태로 표현하는 방법이다. DAG 표현 기반의 연구는 트리 표현 기반의 문제점인 중복된 노드 표현에 따른 저장 공간의 낭비 및 자신의 부모 노드를 파악하기 위해 추가되는 계산 시간의 낭비를 해결해 줄 수 있다. 추후에는 DAG 환경에서 조상(부모)-자손(자식)-형제 관계를 쉽게 알 수 효율적인 레이블링 방법을 연구하고자 하며, 이를 확장하여 스키마가 없는 잘 정형화된(well-formed) 동적인 XML 환경에 적용하고자 한다.

또한, 논문에서 제시된 실험은 467KB인 sigmodrecord.xml 데이터를 사용하였기 때문에 그 양이 작아 클라이언트에서 한꺼번에 문서를 받고 질의 처리는 엘리먼트 단위로 하고 있다. 입력되는 XML 문서의 크기가 큰 경우는 문서 전체가 네트워크를 타고 들어오는데 있어서 bandwidth에 문제가 있어 완전한 XML stream 환경을 위해서는 추가적인 작업이 필요하다. 그것이 XML 단편화(fragmentation) 기법입니다. XML 문서를 단편화하고 서버에서 단편화된 XML 조각들을 보내고 클라이언트에서 질의 처리를 하면서 질의에 해당되는 XML 조각들을 수집하고 원래 구조에 맞게 짜 맞춰야 하는 작업이 필요하다. 본 논문의 개념과 XML 단편화 기법을 결합하여 실제 XML 스트림 환경에 적용하고자 한다.

참 고 문 헌

[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.

[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, (http://www.w3.org/TR/xpath20/).

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

[4] E. Damiani, S. Vimercati, S. Paraboachk and P. Samarati. 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. Samarati. A Fine-grained Access Control System for XML Documents. ACM Trans. Information and System Sec., Vol.5, No.2, 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, Vol.3, No.3, 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] M. Murata, A. Tozawa, and M. Kudo, "XML Access Control using Static Analysis" ACM CCS, Washington D.C., pp. 73-84, 2003.

[9] Jae-Myeong Jeon, Yon Dohn Chung, Myoung Ho Kim, and Yoon Joon Lee, Filtering XPath expressions for XML access control. Computers & Security, pp. 591-605, 2004.

[10] B. Luo, D. W. Lee, W. C. Lec, and P. Liu, "Qfilter: Fine-grained Run-Time XML Access Control via NFA-based Query Rewriting," CIKM'04, pp. 543-552, 2004.

[11] C. Byun and S. Park, "Two Phase Filtering for

XML Access Control," VLDB Workshop on SDM'06, pp. 115-130, 2006.
[12] W. Fan, C. Y. Chan, and M. Garofalakis, "SMOQE: A system for providing secure access to XML," VLDB, pp. 1227-1230, 2006.
[13] W. Fan, F. Geerts, X. Jia and A. Kementsietsidis, "Rewriting Regular XPath Queries on XML views," ICDE, pp. 666-675, 2007.
[14] L. Bouganim, F. D. Ngoc, and P. Pucheral, "Client-based access control management for XML documents," VLDB, pp. 84-95, 2004.
[15] F. Rabitti, E. Bertino, W. Kim and D. Woelk. A Model of Authorization for Next-Generation Database Systems. ACM Transaction on Database Systems, 126(1), pp. 88-131, March 1991.
[16] A. R. Schmidt, F. Waas, M. L. Kersten, D. Florescu, I. Manolescu, M. J. Carey, andR.Busse. The XML Benchmark Project. Technical Report INS-R0103, CWI, April2001.

변 창 우

1999년 서강대학교 컴퓨터학과(공학사) 2001년 서강대학교 컴퓨터학과(공학석사). 2007년 8월 서강대학교 컴퓨터공학과(공학박사). 2007년 9월~현재 인하공업전문대학 컴퓨터시스템과 교수. 관심분야는 XML 질의 처리기, 동적인 데이터베이스에서의 트랜잭션 관리, 역할기반 접근제어 모델, XML, XML 데이터베이스 시스템에서의 필터링 기법, XML 접근제어, 프라이버시 임

안 은 주

2001년 서강대학교 컴퓨터학과(공학사) 2001년 2월~2003년 6월 (주)원클릭테크놀로지스. 2003년 7월~2005년 8월 (주)러시 대리. 2007년 서강대학교 컴퓨터학과(공학석사). 2007년 8월~2008년 10월 (주)온더아이티 정보기술 연구소 차장. 관심분야는 데이터 스트림 관리 시스템, XML 질의 처리, XML 접근제어 임

박 석

정보과학회논문지 : 데이타베이스
제 36 권 제 2 호 참조