동적 프레디킷: 히포크라테스 XML 데이타베이스를 위한 효율적인 액세스 통제 방법

(Dynamic Predicate: An Efficient Access Control Mechanism for Hippocratic XML Databases)

이 재 길 † 한 욱 신 †† 황 규 영 †††
(Jae-Gil Lee) (Wook-Shin Han) (Kyu-Young Whang)

요 약 최근에 Agrawal 등이 제안한 히포크라테스 데이타베이스는 관계형 데이타베이스에 프라이버시 보호 기능을 추가한 데이타베이스 모델이다. 본 논문의 저자들은 히포크라테스 데이타베이스 모델을 XML 데이타베이스에 적용할 수 있도록 확장한 **히포크라테스 XML 데이타베이스** 모델[4]을 제안하였다. 본 논문에서는 **동적 프레디킷**(*dynamic predicate*)이라는 새로운 개념을 제안하고, 히포크라테스 XML 데이타베이스 모델에서의 액세스 통제에 이 개념을 적용한다. 동적 프레디킷은 권한에 의해 액세스가 허용되는지를 결정하는데 적용될 수 있는 질의 처리 도중에 동적으로 생성되는 조건을 나타낸다. 동적 프레디킷은 권한 검사를 질의 계획에 효과적으로 통합하여 액세스가 허용되지 않은 엘리먼트들이 질의 처리 과정에서 일찍 제외될 수 있게 해준다. 합성 데이타와 실제 데이타를 사용하여 기존의 액세스 통제 방법과 질의 처리 시간을 비교하는 다양한 실험을 수행한 결과, 본 논문에서 제안한 액세스 통제 방법은 하향식 액세스 통제 방법에 비하여 최대 219배, 상향식 액세스 통제 방법에 비하여 최대 499배 성능을 향상시킴을 보였다. 본 논문의 주요 공헌은 히포크라테스 XML 데이타베이스 모델 상에서 동적 프레디킷을 사용하여 액세스 통제 방법을 질의 계획에 효과적으로 통합할 수 있도록 한 것이다.

키워드 : 보안, 프라이버시, 액세스 통제 방법, XML 데이타베이스, 히포크라테스 데이타베이스

Abstract The Hippocratic database model recently proposed by Agrawal et al. incorporates privacy protection capabilities into relational databases. The authors have subsequently proposed the *Hippocratic XML database* model[4], an extension of the Hippocratic database model for XML databases. In this paper, we propose a new concept that we call the *dynamic predicate(DP)* for effective access control in the Hippocratic XML database model. A DP is a novel concept that represents a dynamically constructed condition that can be adapted for determining the accessibility of elements during query execution. DPs allow us to effectively integrate authorization checking into the query plan so that unauthorized elements are excluded in the process of query execution. Using synthetic and real data, we have performed extensive experiments comparing query processing time with those of existing access control mechanisms. The results show that the proposed access control mechanism improves the wall clock time by up to 219 times over the top-down access control strategy and by up to 499 times over the bottom-up access control strategy. The major contribution of our paper is enabling effective integration of access control mechanisms with the query plan using the DP under the Hippocratic XML database model.

Key words : Security, Privacy, Access control mechanism, XML database, Hippocratic database

· 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단으로부터 지원을 받았음

† 정 회 원 : 한국과학기술원 전산학과/첨단정보기술연구센터
jglee@mozart.kaist.ac.kr
†† 종신회원 : 경북대학교 컴퓨터공학과 교수
wshan@knu.ac.kr
††† 종신회원 : 한국과학기술원 전산학과 교수, 첨단정보기술연구센터 소장
kywhang@mozart.kaist.ac.kr

논문접수 : 2004년 8월 26일
심사완료 : 2005년 6월 9일

1. 서 론

최근 들어 사적인 데이타가 데이타베이스에 점점 더 많이 저장되고 있으며[1], 프라이버시(privacy)의 중요성이 크게 대두되고 있다[2]. 데이타베이스에 저장되어 있는 사적인 데이타를 제공한 데이타 제공자의 프라이버시를 보호하기 위해, Agrawal 등은 프라이버시 보호

기능이 데이타베이스 시스템의 주요 기능에 추가되어야 한다는 점을 강조하고, 관계형 데이타베이스에 프라이버시 보호 기능을 추가한 히포크라테스 데이타베이스라는 새로운 개념을 제안하였다[3].

히포크라테스 데이타베이스는 관계형 데이타베이스에 기반한 모델이므로 최근에 널리 사용되는 XML 데이타베이스에 적용하기 위해서는 확장이 필요하다. 본 논문의 저자들은 히포크라테스 데이타베이스 모델을 XML 데이타베이스에 적용할 수 있도록 확장한 **히포크라테스 XML 데이타베이스**(*Hippocratic XML database*) 모델과 이 모델에서의 기본적인 액세스 통제 방법을 제안하였다[4]. XML 데이타는 관계형 모델과 달리 트리 형태의 계층 구조를 가진다. 따라서, 히포크라테스 데이타베이스 모델에서 제시한 개념들을 트리 형태의 계층 구조에 맞게 확장하며, 확장된 개념들을 정형적으로 정의하였다.

참고문헌 [4]에서 제안한 액세스 통제 방법은 권한 인덱스와 최근접 질의[5] 기법을 사용한다. 이 방법은 먼저 XML 데이타 트리에서 권한이 부여된 엘리먼트를 2-차원 공간상의 점으로 매핑시켜 다차원 인덱스[6]로 구현된 권한 인덱스에 저장한다. XML 데이타 트리의 특정 엘리먼트에 대한 권한은 가장 가까운 조상 엘리먼트에 부여된 권한에 의해 결정되므로[7-9] 이 권한을 찾아내기 위해 최근접 질의 기법을 활용한다.

본 논문에서는 참고문헌 [4]에서 제안한 히포크라테스 XML 데이타베이스 모델의 기본적인 액세스 통제 방법을 개선한다. 참고문헌 [4]의 방법은 기존의 XML 액세스 통제 방법들[7-9]과 달리 권한 검사를 위해 XML 데이타 트리를 탐색하지 않기 때문에 기존의 XML 액세스 통제 방법들에 비해서는 좋은 성능을 보인다. 하지만, 이 방법은 질의 처리 과정에서 전체 질의 결과를 먼저 검색(retrieve)하고 후처리로서 각각의 질의 결과가 액세스가 허용된 질의 결과인지 검사하기 때문에 액세스가 허용되지 않은 질의 결과까지 불필요하게 검색하는 단점이 있다.

이러한 단점을 해결하기 위해, 본 논문에서는 액세스 통제를 질의 계획에 통합하여 불필요한 질의 결과 검색을 미연에 방지할 수 있는 **동적 프레디킷**(*dynamic predicate*)이라는 새로운 개념을 제안한다. 동적 프레디킷은 권한에 의해 액세스가 허용되는지를 결정하는데 적용될 수 있는 질의 처리 도중에 동적으로 생성되는 조건을 나타낸다. 동적 프레디킷을 사용하여 액세스 통제 방법을 질의 계획에 효과적으로 통합할 수 있으며, 이로 인해 액세스가 허용되지 않은 엘리먼트들은 질의 처리에서 일찍 제외되어 질의 처리 성능을 크게 향상시킬 수 있다.

본 논문의 공헌을 요약하면 다음과 같다.

- 액세스 통제를 질의 계획에 효과적으로 통합할 수 있게 해주는 동적 프레디킷이라는 새로운 개념을 제안한다.
- 히포크라테스 XML 데이타베이스 모델에서의 권한 인덱스, 최근접 질의 기법, 동적 프레디킷을 사용하는 질의 처리와 통합된 액세스 통제 방법을 제안한다.
- 다양한 실험을 통하여 제안한 액세스 통제 방법이 기존의 방법에 비해 매우 성능이 우수함을 입증한다.

본 논문의 구성은 다음과 같다. 제2절에서는 관련 연구로서 XML 데이타의 프라이버시 보호에 대한 기존의 연구를 설명한다. 제3절에서는 히포크라테스 XML 데이타베이스에서의 개선된 액세스 통제 방법을 제안한다. 제4절에서는 제안하는 액세스 통제 방법의 성능 평가 결과를 제시한다. 마지막으로, 제5절에서는 결론을 내린다.

2. 관련 연구

본 절에서는 관련 연구로서 XML 데이타의 프라이버시 보호에 대한 기존의 연구를 설명한다. 제2.1절에서는 기존의 XML 액세스 통제 방법[7-10]을 설명한다. 제2.2절에서는 히포크라테스 XML 데이타베이스 모델[4]을 설명한다. 제2.3절에서는 히포크라테스 XML 데이타베이스 모델을 위한 기본적인 액세스 통제 방법[4]을 설명한다.

2.1 기존의 XML 액세스 통제 방법

XML 데이타의 보안을 위하여 여러 가지 XML 보안 모델[7,8]이 발표되어 있다. XML 보안 모델은 데이타베이스에 저장된 XML 데이타에 권한을 부여 및 취소하는 방법과 부여된 권한을 통하여 XML 데이타에 대한 액세스를 통제하는 방법을 제공한다.

XML 문서의 특정 엘리먼트에 대한 권한은 가장 가까운 조상 엘리먼트에 부여된 권한에 의해 내포될 수 있으므로, 권한 검사를 수행하기 위해서는 해당 엘리먼트 뿐만 아니라 조상 엘리먼트에도 권한이 부여되어 있는지를 검사해야 한다. 이를 위해, 기존의 XML 액세스 통제 방법[7,8]은 XML 문서의 루트와 검사하려는 엘리먼트 사이의 패스 상의 각각의 엘리먼트마다 권한이 부여되어 있는지 검사한다. 기존의 방법은 패스를 탐색하는 방향에 따라 루트로부터 검사하려는 엘리먼트의 방향으로 패스를 탐색하는 **하향식**(*top-down*) **방법**과 검사하려는 엘리먼트로부터 루트의 방향으로 패스를 탐색하는 **상향식**(*bottom-up*) **방법**으로 구분된다[7]. 두 가지 방법은 모두 검사하려는 엘리먼트의 조상 엘리먼트까지도 권한이 부여되어 있는지를 검사한다. 검사하려는 엘리먼트가 데이타베이스 전체에 분포되어 있으면, 최악

의 경우 권한 검사를 위해 전체 데이타베이스를 액세스할 수 있다[7].

최근에 일반적인 XML 보안 모델을 다소 변형시킨 보안 모델에서의 액세스 통제 방법이 몇 가지 발표되었다. 참고문헌 [10]에서는 권한이 후손 엘리먼트에 대한 권한을 내포하지 않는 보안 모델을 가정하였다. 이러한 종류의 보안 모델에서 모든 엘리먼트마다 권한이 명시적으로 부여되어야 하므로 명시적으로 부여된 권한의 개수가 많아지는 문제가 있다. 이와 같은 문제를 해결하기 위해 인접한 엘리먼트에 부여된 권한들을 압축하여 표현하는 방법인 Compressed Accessibility Map(CAM)을 제안하였다. 참고문헌 [9]에서는 인스턴스 레벨의 권한이 미리 지정된 패턴(security annotation)으로만 부여되는 특수한 보안 모델을 가정하고, 실행 시간(runtime)에 수행될 필요가 있는 권한 검사의 횟수를 최소화하는 최적화 기법을 제안하였다.

2.2 히포크라테스 XML 데이타베이스 모델

히포크라테스 XML 데이타베이스 모델[4]은 Agrawal 등이 제안한 히포크라테스 데이타베이스 모델[3]을 XML 데이타베이스에 적용할 수 있도록 확장한 모델이다. 히포크라테스 XML 데이타베이스는 Agrawal 등의 히포크라테스 데이타베이스와 같이 데이타 수집 과정과 질의 처리 과정에서 프라이버시 보호를 위한 검사를 수행한다. 본 절에서는 논문에서 중점을 두는 질의 처리 과정에서의 프라이버시 보호 방법을 설명한다.

질의 처리 도중에 히포크라테스 XML 데이타베이스 모델은 Agrawal 등[3]에 의해 정의된 바와 같이 속성 액세스 통제(attribute access control)와 레코드 액세스 통제(record access control)를 연속해서 수행함으로써 데이타 제공자의 프라이버시를 보호한다. 질의에는 질의를 수행하고자 하는 사용목적이 함께 명시되며, 이는 속성 액세스 통제와 레코드 액세스 통제에 사용된다. **속성 액세스 통제**는 데이타 관리자가 지정한 데이타의 사용목적과 질의 사용목적이 일치하는지 스키마 레벨에서 검사하는 과정이다. **레코드 액세스 통제**는 데이타 제공자가 명시한 데이타의 사용목적과 질의 사용목적이 일치하는지 레코드 레벨에서 검사하는 과정이다. Agrawal 등의 히포크라테스 데이타베이스 모델에서는 속성 액세스 통제만을 권한 모델을 도입하여 정의한 반면, 히포크라테스 XML 데이타베이스 모델에서는 속성 액세스 통제와 레코드 액세스 통제를 모두 권한 모델을 도입하여 정의하였다.

정의 1과 2에서 두 가지 종류의 권한인 **관리자 프라이버시 권한**(*administrator privacy authorization*)과 **제공자 프라이버시 권한**(*provider privacy authorization*)을 정의한다. 관리자 프라이버시 권한은 속성 액세스 통제에 사용되며, 제공자 프라이버시 권한은 레코드 액세스 통제에 사용된다.

정의 1. [4] 데이타 관리자에 의해 부여되는 권한을 **관리자 프라이버시 권한**(*administrator privacy authorization*)이라 부르며, 3-투플 (s, o, p)로 정의한다.

- s: 데이타 사용자;
- o: DTD의 엘리먼트를 지정하는 경로식(path expression)[1];
- p: 권한 타입 혹은 데이타의 사용목적.

이 권한은 모델의 간결성을 위해 항상 후손 엘리먼트에 대한 권한을 내포한다고 가정한다.

정의 2. [4] 데이타 제공자에 의해 부여되는 권한을 **제공자 프라이버시 권한**(*provider privacy authorization*)이라 부르며, 2-투플 (o, p)로 정의한다.

- o: XML 문서의 엘리먼트를 지정하는 경로식(path expression);
- p: 권한 타입 혹은 데이타의 사용목적.

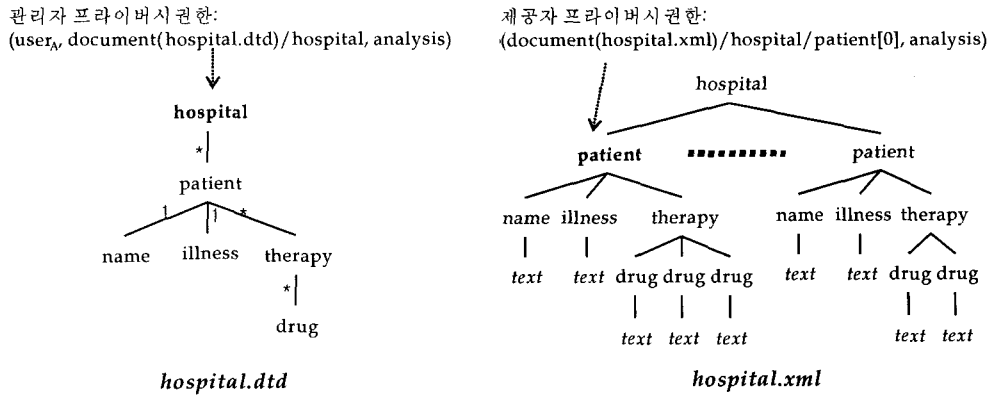
데이타 사용자는 관리자 프라이버시 권한과 제공자 프라이버시 권한이 모두 부여된 엘리먼트만을 액세스할 수 있다.

예 1. 그림 1은 관리자 프라이버시 권한과 제공자 프라이버시 권한이 부여된 예를 나타낸다. 그림 1(a)의 *hospital.dtd*는 그림 1(b)의 *hospital.xml*의 DTD이다. 그림 1(a)에서 관리자 프라이버시 권한 ($user_A$, *document*("*hospital.dtd*")/*hospital*, *analysis*)은 *hospital.dtd*에 부여되므로, $user_A$는 이 DTD의 인스턴스인 *hospital.xml*의 *hospital* 엘리먼트 및 후손 엘리먼트들을 *analysis* 사용목적을 위해 액세스할 수 있다. 그림 1(b)에서 *hospital.xml*에 부여된 제공자 프라이버시 권한 (*document*("*hospital.xml*")/*hospital*/*patient*[0], *analysis*)은 *hospital.xml*의 첫번째 *patient* 엘리먼트 및 후손 엘리먼트들이 *analysis* 사용목적을 위해 데이타 제공자로부터 제공되었음을 나타낸다. 그러므로, 데이타 사용자 $user_A$는 관리자 프라이버시 권한과 제공자 프라이버시 권한이 모두 부여된 *hospital.xml*의 첫번째 *patient* 엘리먼트 및 후손 엘리먼트들을 *analysis* 사용목적을 위해 액세스할 수 있다. □

2.3 히포크라테스 XML 데이타베이스에서의 액세스 통제 방법

본 절에서는 히포크라테스 XML 데이타베이스 모델의 기본적인 액세스 통제 방법을 설명한다. 기존의 XML 액세스 통제 방법[7-9]은 XML 데이타 트리를 탐색하면서 가장 가까운 조상 엘리먼트에 부여된 권한을

1) 본 논문에서는 경로식의 표기법으로 XPath[11] 표준을 사용한다. 경로식의 결과로 엘리먼트의 집합이 나올 수 있는데, 이러한 경우에는 각각의 엘리먼트마다 권한이 부여된다고 가정한다.



(a) 관리자 프라이버시 권한의 부여 (b) 제공자 프라이버시 권한의 부여

그림 1 프라이버시 권한 부여의 예

검색하므로, 최악의 경우 XML 데이타 트리 전체를 액세스하는 비용이 소요된다. 참고문헌 [4]에서는 이러한 문제를 해결하기 위해 권한 인덱스와 최근접 질의를 활용하는 액세스 통제 방법을 제안하였다. 제2.3.1절에서는 액세스 통제 방법에 사용되는 권한 인덱스[4]의 구조를 설명하고, 제2.3.2절에서는 액세스 통제 알고리즘을 설명한다.

2.3.1 권한 인덱스의 구조

권한이 부여된 엘리먼트는 많은 XML 질의 처리 방법에서 사용되는 번호화 방식(numbering scheme)[12-14]에 의해 2-차원 상의 점 (*start*, *end*)로 표현되므로, 이들을 저장하기 위한 권한 인덱스로서 2-차원 인덱스[6]를 사용한다. 번호화 방식에서 (*start*, *end*) 값은 엘리먼트 간의 조상-후손 관계를 나타내며, 다음의 두 조건을 만족한다[12]: (1) 엘리먼트 v가 엘리먼트 u의 부모일때 ($start_u$, end_u) 범위(interval)는 ($start_v$, end_v) 범위에 포함되며, (2) 두 형제(sibling) 엘리먼트 u, v에서 엘리먼트 u가 엘리먼트 v 전에 나오면 $end_u < start_v$ 이다. 따라서, $start_a < start_d \wedge end_a > end_d$이면 엘리먼트 a는 d의 조상이다.

2.3.2 권한 인덱스와 최근접 질의를 사용하는 기본적인 액세스 통제 방법

XML 문서의 특정 엘리먼트에 부여된 권한은 후손 엘리먼트에 대한 묵시적 권한을 내포하며, 묵시적 권한은 후손 엘리먼트에 부여된 다른 명시적 권한에 의하여 오버라이드된다. 따라서, 검사하려는 엘리먼트에 대한 권한은 가장 가까운 조상 엘리먼트에 부여된 권한에 의해 내포된다. 이러한 조상 엘리먼트에 부여된 권한을 **최근접 조상 권한**(*nearest ancestor authorization*)[4]이라 부르고 정의 3에서 이를 정형적으로 정의한다. 그리고, 보조정리 1에서 이를 구하는 방법을 제시한다.

정의 3. [4] 엘리먼트 e의 **최근접 조상 권한**인 $auth_{naa}$는 다음의 두 조건을 만족하는 권한이다: (1) $auth_{naa}$는 엘리먼트 e 혹은 e의 조상 엘리먼트에 부여된 명시적 권한이며, (2) 엘리먼트 e와 $auth_{naa}$가 부여된 엘리먼트 사이의 패스 상의 엘리먼트에 다른 명시적 권한이 존재하지 않는다.

보조정리 1. [4] 번호화 방식을 사용할 때, 엘리먼트 e의 최근접 조상 권한인 $auth_{naa}$는 $start(auth) \le start(e) \wedge end(auth) \ge end(e)$를 만족하는 권한 $auth$ 중에서 $|start(e) - start(auth)|$의 값을 최소로 하는 권한이다. 여기에서 $start(e)$, $end(e)$는 엘리먼트 e의 *start*, *end* 값을 나타내며, $start(auth)$, $end(auth)$는 권한 $auth$가 부여된 엘리먼트의 *start*, *end* 값을 나타낸다.

증명. 보조정리 1에 따라 구해진 권한은 정의 3의 두 조건을 모두 만족함을 보인다. 첫째, 번호화 방식의 정의에 따라 권한 $auth_{naa}$는 엘리먼트 e 혹은 e의 조상 엘리먼트에 부여된 권한이므로 조건 (1)을 만족시킨다. (권한이 부여된 e의 조상 엘리먼트를 e_{naa}라고 부른다.) 둘째, $|start(e) - start(auth)|$의 값을 최소로 하는 권한이 $auth_{naa}$이고, 엘리먼트 e와 엘리먼트 e_{naa}의 패스 상의 엘리먼트에 다른 명시적 권한 $auth_{exp}$가 존재한다고 가정한다. 그러면, 번호화 방식의 정의에 따라 $start(e_{naa}) < start(auth_{exp}) < start(e)$이다. (여기서 $start(auth_{naa}) = start(e_{naa})$ 이다.) 이는 *start* 값의 차이가 최소인 권한이 $auth_{naa}$라는 가정에 위배된다. 따라서, 엘리먼트 e와 엘리먼트 e_{naa}의 패스 상의 엘리먼트에는 다른 권한이 존재하지 않으므로 조건 (2)를 만족시킨다. □

참고문헌 [4]에서는 권한 인덱스와 최근접 질의를 사용하는 액세스 통제 알고리즘을 제안하였으며, 이를 *Nearest Ancestor Filtering*이라 부른다. 본 알고리즘은 질의를 수행한 다음 매 질의 결과마다 보조정리 1에

```
Algorithm Nearest Ancestor Filtering
Input: (1) a set of XML query results R and a usage purpose p of a query
       (2) the authorization index storing provider privacy authorizations
Output: authorized query results
Begin
01:  for each query result r ∈ R do
         /* search the authorization index */
02:      Find the nearest ancestor authorization auth_naa of r according to Lemma 1;
03:      if the usage purpose of auth_naa implies the usage purpose p then
04:          output r;
End
```

그림 2 액세스 통제 알고리즘 Nearest Ancestor Filtering[4]

따라 최근접 조상 권한을 구하여 권한 검사를 수행한다. 이때, 최근접 조상 권한은 최근접 질의[5] 기법을 활용하여 구할 수 있다. 마지막으로, 질의 결과에 대한 액세스가 허용되는지 검사하기 위해 최근접 조상 권한의 사용목적이 질의 사용목적을 내포하는지 검사한다.[2] 최근접 조상 권한을 찾기 위해, Nearest Ancestor Filtering은 권한 인덱스를 한번만 액세스하는 반면, 하향식 및 상향식 방법[7]은 많은 조상 엘리먼트를 액세스하고 액세스한 조상 엘리먼트마다 부여된 권한을 찾는다. 그림 2는 알고리즘 Nearest Ancestor Filtering을 나타낸다.

3. 개선된 액세스 통제 알고리즘

본 절에서는 히포크라테스 XML 데이타베이스에서의 액세스 통제 알고리즘 Nearest Ancestor Filtering의 성능을 향상시킨 개선된 알고리즘을 제시한다. 제3.1절에서는 개선된 액세스 통제 알고리즘의 개요를 설명한다. 제3.2절에서는 **동적 프레디킷**이라는 새로운 개념을 제안하고, 이를 사용하여 권한 검사를 질의 계획에 통합하는 최적화 기법을 제안한다. 마지막으로, 제3.3절에서는 갱신 연산 처리 시 발생할 수 있는 문제와 그 해결 방안을 논의한다.

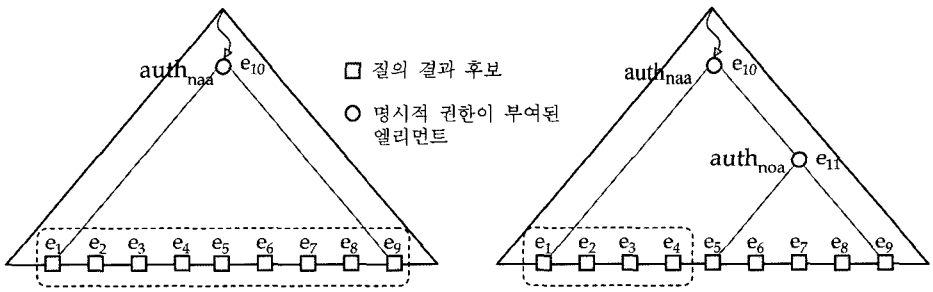
3.1 개요

제2.3.2절에서 설명한 알고리즘 Nearest Ancestor Filtering은 액세스가 허용되지 않은 질의 결과까지 불필요하게 검색(retrieve)하는 문제가 있다. 왜냐하면, 이 알고리즘은 질의 처리를 수행하여 전체 질의 결과를 먼저 얻어내고 후처리로서 각각의 질의 결과가 액세스가 허용된 질의 결과인지 검사하기 때문이다. 이러한 문제를 해결하기 위해, 권한 검사를 질의 계획에 통합하여 액세스가 허용되지 않은 엘리먼트들은 질의 처리 대상에서 미리 제외하는 최적화 기법을 제안한다.

위에서 언급한 최적화는 질의 처리 도중에 동일한 액세스 가능성(accessibility)을 가지는 엘리먼트들의 그룹을 알아내어 수행할 수 있다. 이러한 엘리먼트들의 start 값은 범위(interval)를 형성하며, 이를 본 논문에서 *start* 범위라 부른다. start 범위는 액세스가 모두 허용된 엘리먼트들의 그룹 혹은 액세스가 모두 허용되지 않은 엘리먼트들의 그룹을 나타낸다. 질의 처리 시에는 질의 계획에서 액세스하려는 엘리먼트의 start 값이 이러한 start 범위에 포함되는지를 검사함으로써 액세스가 허용된 엘리먼트인지 결정할 수 있다. 또한, 액세스가 허용된 엘리먼트인지 알아내기 위해 알고리즘 Nearest Ancestor Filtering처럼 매번 권한 인덱스를 검색할 필요가 없다.

start 범위는 동일한 최근접 조상 권한을 가지는 엘리먼트들의 start 값의 범위로서 구성할 수 있다. 정의 3에 따라 특정 엘리먼트 e의 액세스 가능성은 엘리먼트 e의 최근접 조상 권한에 의해 결정되기 때문이다. 그림 3은 동일한 최근접 조상 권한을 가지는 엘리먼트들을 나타낸다. 그림 3(a)에서 엘리먼트 e_1의 최근접 조상 권한은 e_{10}에 부여된 $auth_{naa}$이며, $auth_{naa}$를 오버라이드하는 권한이 존재하지 않는다. 따라서, 엘리먼트 e_{10}의 후손 엘리먼트인 $e_1 \sim e_9$는 모두 최근접 조상 권한으로 $auth_{naa}$를 가진다. 그러나, 그림 3(b)에서는 e_{11}에 부여된 $auth_{noa}$가 $auth_{naa}$를 오버라이드하므로, 엘리먼트 $e_1 \sim e_4$만 최근접 조상 권한으로 $auth_{naa}$를 가진다. 이러한 경우까지를 모두 고려하기 위해, 최근접 조상 권한 $auth_{naa}$의

2) 액세스가 허용되는 질의 결과인지를 결정하는 정책에는 두 가지가 있다. 첫번째 정책[7,8]은 질의 결과로 반환되는 인스턴스에 대해서만 부여된 권한을 검사하는 것이며, 두번째 정책[9]은 질의 결과로 반환되는 인스턴스 뿐만 아니라 질의 결과 인스턴스를 만들어내는 다른 모든 인스턴스까지도 부여된 권한을 검사하는 것이다. 본 논문에서는 단순히 첫번째 정책을 택한다. 두번째 정책을 위한 방법은 향후 연구에서 제시한다.



(a) 최근접 조상 권한 ($auth_{naa}$) (b) 최근접 오버라이드 권한 ($auth_{noa}$)

그림 3 동일한 최근접 조상 권한을 가지는 엘리먼트들

최근접 오버라이드 권한(*nearest overriding authorization*) $auth_{noa}$를 $auth_{naa}$로부터 XML 데이타 트리를 프리오더 순으로 탐색할 때 가장 먼저 $auth_{naa}$를 오버라이드하는 권한으로 정의한다. 정의 4에서 이를 정형적으로 정의하고, 보조정리 2에서 이를 구하는 방법을 제시한다.

정의 4. 엘리먼트 e의 최근접 조상 권한을 $auth_{naa}$라고 하고, $auth_{naa}$가 약성 권한이라고 하자. 이때, $auth_{naa}$의 **최근접 오버라이드 권한**인 $auth_{noa}$는 다음의 두 조건을 만족하는 권한이다: (1) $auth_{noa}$는 사용목적에 관계없이 $auth_{naa}$가 부여된 엘리먼트의 후손 엘리먼트에 부여된 명시적 권한이며, (2) 엘리먼트 e부터 $auth_{noa}$가 부여된 엘리먼트까지 프리오더(preorder) 순으로 방문할 때 두 엘리먼트 사이에 다른 명시적 권한이 존재하지 않는다. 그러나, $auth_{naa}$가 강성 권한인 경우는 정의에 따라 $auth_{noa}$는 존재하지 않는다.

보조정리 2. 번호화 방식을 사용할 때, 엘리먼트 e의 최근접 조상 권한이 $auth_{naa}$이고 $auth_{naa}$가 약성 권한이라 하자. 이때, $auth_{naa}$의 최근접 오버라이드 권한인 $auth_{noa}$는 $start(auth) > start(e) \geq start(auth_{naa}) \wedge end(auth) < end(auth_{naa})$를 만족하는 권한 $auth$ 중에서 $|start(e) - start(auth)|$의 값을 최소로 하는 권한이다.

증명. 보조정리 2에 따라 구해진 권한은 정의 4의 두 조건을 모두 만족함을 보인다. 첫째, 번호화 방식의 정의에 따라 권한 $auth_{noa}$는 $auth_{naa}$가 부여된 엘리먼트 e_{naa}의 후손 엘리먼트에 부여된 권한이므로 조건 (1)을 만족시킨다. (e_{naa}의 후손 엘리먼트 중에서 $auth_{noa}$가 부여된 엘리먼트를 e_{noa}라고 부른다.) 둘째, $|start(e) - start(auth)|$의 값을 최소로 하는 권한이 $auth_{noa}$이고, 엘리먼트 e부터 엘리먼트 e_{noa}까지 프리오더로 방문할 때 다른 명시적 권한 $auth_{exp}$가 두 엘리먼트 사이에 존재한다고 가정한다. 그러면, 번호화 방식의 정의에 따라 $start(e) < start(auth_{exp}) < start(e_{noa})$ 이다. (여기서 $start(auth_{noa}) = start(e_{noa})$ 이다.) 이는 start 값의 차이가 최소인 권한이 $auth_{noa}$라는 가정에 위배된다. 따라서, 엘리먼트 e부터 엘리먼트 e_{noa}까지 프리오더로 방문할 때 다른 권한이 두 엘리먼트 사이에 존재하지 않으므로 조건 (2)를 만족시킨다. □

엘리먼트 e와 동일한 최근접 조상 권한을 가지는 엘리먼트들의 start 범위는 보조정리 3과 같이 구할 수 있다. 이 범위에 start 값이 포함되는 엘리먼트들은 엘리먼트 e의 최근접 조상 권한의 권한 타입에 따라 모두 액세스가 허용되거나 금지된다.

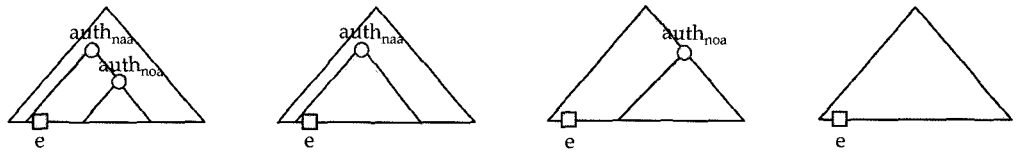
보조정리 3. 엘리먼트 e의 최근접 조상 권한이 $auth_{naa}$이고 $auth_{naa}$의 최근접 오버라이드 권한이 $auth_{noa}$일 때, start 값이 아래의 범위(interval)에 포함되는 엘리먼트들은 엘리먼트 e와 동일한 최근접 조상 권한을 가진다.

- $auth_{naa}$가 존재하는 경우
  - $auth_{noa}$가 존재하는 경우: $[start(auth_{naa}), start(auth_{noa}))$
  - $auth_{noa}$가 존재하지 않는 경우: $[start(auth_{naa}), end(auth_{naa}))$
- $auth_{naa}$가 존재하지 않는 경우에 루트 엘리먼트의 최근접 오버라이드 권한을 구하여 이를 $auth_{noa}$로 사용함
  - $auth_{noa}$가 존재하는 경우: $[start(root), start(auth_{noa}))$
  - $auth_{noa}$가 존재하지 않는 경우: $[start(root), end(root))$

증명. (sketch): 그림 4는 $auth_{naa}$와 $auth_{noa}$의 존재 여부에 따라 분류한 네 가지 경우를 나타낸 것이다. 음영처리된 영역에 위치하는 엘리먼트들은 엘리먼트 e와 동일한 최근접 조상 권한을 가지며, 이러한 엘리먼트들의 start 범위는 보조정리 3의 범위와 같다. □

3.2 동적 프레디킷(dynamic predicate)

본 절에서는 동적 프레디킷이라는 개념을 제안한다. **동적 프레디킷**(*dynamic predicate: DP*)은 질의 처리 도중에 동적으로 생성되는 조건을 의미한다. 인스턴스 레벨의 권한으로 인해 보조정리 3에 의해 구해진 start 범위는 **데이타-의존적**(*data-dependent*)이며 실행 시간



(a) 첫 번째 경우 (b) 두 번째 경우 (c) 세번째 경우 (d) 네번째 경우

그림 4 엘리먼트 e와 동일한 최근접 조상 권한을 가지는 엘리먼트들이 위치하는 영역

에 구성되어야 한다. 질의 계획에 삽입된 동적 프레디킷은 이러한 데이타-의존적인 start 범위가 실행 시간에 검사될 수 있도록 해준다. start 범위를 검사함으로써, 액세스가 허용되지 않은 엘리먼트들을 질의 처리 대상에서 미리 제외할 수 있으므로 질의 처리 성능을 높일 수 있다. 따라서, 본 논문에서는 동적 프레디킷이라는 개념을 액세스 통제와 질의 계획을 밀결합하는데 적용한다. 정의 5에서 액세스 통제에 적용되었을 때의 동적 프레디킷을 정형적으로 정의한다.

정의 5. 엘리먼트 *e*의 **동적 프레디킷**(*dynamic predicate*: *DP*)은 질의 처리 도중에 동적으로 생성되는 조건으로서, 엘리먼트 *e*와 동일한 액세스 가능성을 가지는 엘리먼트들의 start 범위를 나타낸다. 동적 프레디킷은 ([$start_{begin}$, $start_{end}$), *sign*)으로 표기한다. [$start_{begin}$, $start_{end}$)는 start 범위를 나타내고, *sign*은 start 범위에 포함된 엘리먼트들이 액세스가 허용됨(+) 혹은 허용되지 않음(-)을 나타낸다. 동적 프레디킷은 다음과 같이 구성한다:

(1) [$start_{begin}$, $start_{end}$): 제공자 프라이버시 권한에 대해 보조정리 3에 따라 구해진 start 범위;

(2) *sign*: + 제공자 프라이버시 권한에 대해 구해진 최근접 조상 권한의 사용목적이 질의 사용목적을 내포하는 경우
- 그 밖의 경우.

예 2. 그림 5의 XML 문서에 *//patient//drug* 질의를 *analysis* 사용목적을 위해 수행한다고 가정한다. 질의 처리 도중에 질의 결과로서 *drug* 엘리먼트 (11,13)이 검색되면 정의 5에 따라 엘리먼트 (11,13)의 동적 프레디킷을 구한다. (1) 엘리먼트 (11,13)의 최근접 조상 권한은 (2,46)에 부여된 권한이고, 이 권한의 최근접 오버라이드 권한은 (22,45)에 부여된 권한이다. 따라서, 보조정리 3에 의해 [$start_{begin}$, $start_{end}$) = [2,22)를 얻는다. (2) 엘리먼트 (2,46)에 부여된 권한의 사용목적은 ¬*analysis*이며, 이는 질의 사용목적인 *analysis*를 내포하지 않는다. 따라서, *sign*은 -가 된다. 이와 같이 구해진 동적 프레디킷 ([2,22), -)을 질의 계획에 추가하면 *drug* 엘리먼트 (11,13)~(17,19)는 질의 결과에서 미리 제외되므로 질의 처리 성능을 높일 수 있다. □

*sign*이 -인 동적 프레디킷이 질의 계획에 추가될 때만 [$start_{begin}$, $start_{end}$) 범위에 start 값이 포함되는 엘리먼트들을 질의 처리에서 제외할 수 있어 질의 처리 성능을 개선할 수 있다. *sign*이 +인 동적 프레디킷은 [$start_{begin}$, $start_{end}$) 범위에 start 값이 포함되는 엘리먼트들을 질의 처리에 포함시켜주는 역할을 한다. 하지만, 이 경우에도 동적 프레디킷을 질의 계획에 추가해야 한다. 왜냐하면, *sign*이 +인 동적 프레디킷도 다음으로 동적 프레디킷을 생성해야 하는 엘리먼트를 찾는데 필요하기 때문이다. 이 엘리먼트는 start 값이 [$start_{begin}$, $start_{end}$) 범위에 포함되지 않는 첫번째 엘리먼트이다.

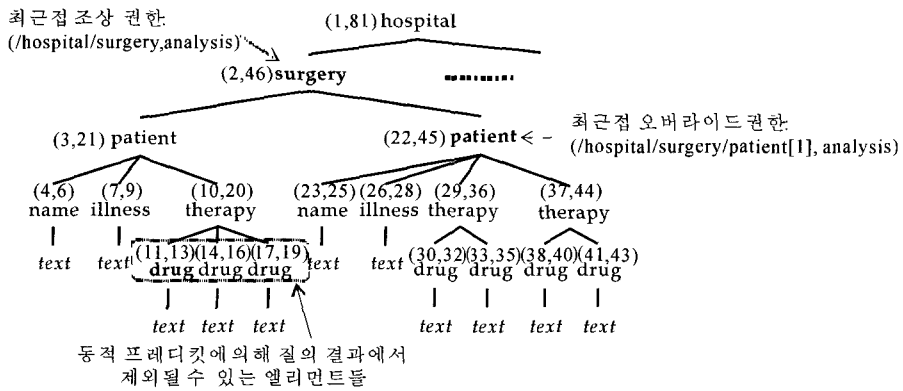


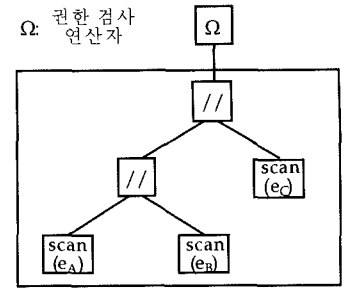
그림 5 동적 프레디킷에 기반한 질의 처리 성능 향상의 예

동적 프레디킷은 질의 계획에서 빨리 적용될수록 액세스가 허용되지 않은 질의 대상이 질의 처리에서 빨리 제외되어 성능을 더 높일 수 있다. 이를 위해, 질의 처리 시에 동적 프레디킷을 질의 계획에서 **푸시-다운** (*push-down*)한다. 이는 관계형 DBMS의 질의 최적화에서 실렉션 프레디킷을 푸시-다운하여 처리되는 투플의 개수를 가능한 빨리 줄이는 질의 최적화 방법과 유사하다고 볼 수 있다. 그러나, 관계형 DBMS의 질의 최적화는 질의에 명시된 실렉션 프레디킷을 질의 분석 단계에서 한번만 푸시-다운하는 반면에, 본 논문의 최적화는 프레디킷을 질의 처리 도중에 동적으로 생성하여 수시로 푸시-다운하는 새로운 방법이다.

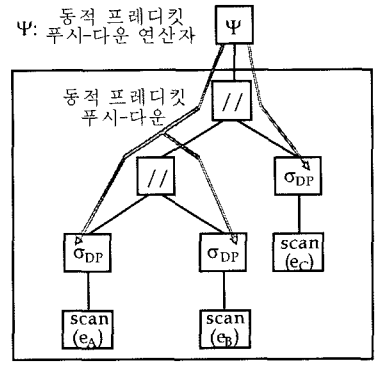
동적 프레디킷의 **푸시-다운**은 질의 계획에서 질의 대상 엘리먼트를 스캔하는 연산자의 바로 위에 실렉션(σ) 연산자를 삽입하고, 질의 처리 도중에 동적 프레디킷을 삽입된 실렉션 연산자의 프레디킷으로 할당(assign)하는 방법이다.

그림 6은 동적 프레디킷을 사용하기 전과 후의 질의 계획을 $//_{e_A}//_{e_B}//_{e_C}$ 형태의 질의를 수행하는 예로서 나타낸다. 질의 계획은 Wu 등[15]에 의해 제안된 방법을 사용하여 구성된다. 여기에서 질의 계획은 관계 대수(relational algebra)의 질의 계획과 유사하다. 즉, 질의 계획은 질의 대상 엘리먼트가 리프 노드이고 연산자가 내부 노드인 트리이다. 동적 프레디킷을 사용하지 않는 기본적인 질의 계획은 그림 6(a)와 같다. 권한 검사 연산자 Ω는 제2.3.2절에서 설명한 알고리즘 Nearest Ancestor Filtering을 수행하여 권한에 의해 액세스가 허용된 질의 결과를 반환하는 연산자이다. // 연산자는 조상-후손 관계에 있는 엘리먼트의 쌍을 반환하는 연산자이다. // 연산자를 처리하기 위한 가장 대표적인 방법으로는 구조적 조인[12-14]이 있다. $scan(e_A)$, $scan(e_B)$, $scan(e_C)$ 연산자는 구조적 조인을 수행할 수 있도록 start 값에 따라 정렬된 순서로 각 엘리먼트를 스캔하는 연산자이다. 기본적인 질의 계획은 $//_{e_A}//_{e_B}//_{e_C}$ 질의의 각 결과를 얻어내고, 각 질의 결과마다 권한 검사 연산자 Ω에 의해 권한 검사를 수행하도록 구성된다. 질의 계획에서 각 연산자의 출력은 상위 연산자의 입력으로 파이프라인(pipeline) 된다[16].

동적 프레디킷을 사용하도록 확장된 질의 계획은 그림 6(b)와 같다. 확장된 질의 계획은 $scan(e_A)$, $scan(e_B)$, $scan(e_C)$ 연산자의 바로 위에 σ_{DP} 연산자를 삽입하고, 동적 프레디킷 푸시-다운 연산자 Ψ에 의해 동적 프레디킷을 생성하여 이를 σ_{DP} 연산자로 푸시-다운하도록 구성된다. 여기에서 화살표는 Ψ 연산자에서 생성된 동적 프레디킷이 σ_{DP} 연산자로 푸시-다운됨을 의미한다.



(a) 기본적인 질의 계획



(b) 동적 프레디킷을 사용하도록 확장된 질의 계획

그림 6 동적 프레디킷을 사용하기 전과 후의 질의 계획

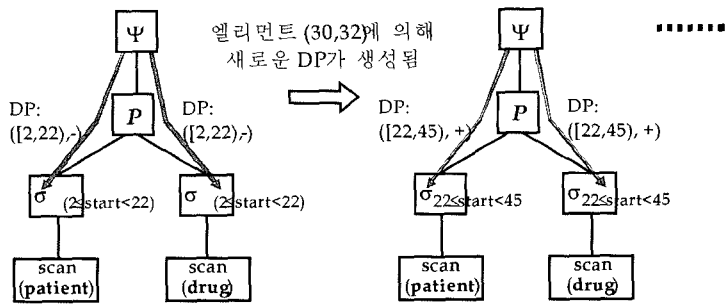
푸시-다운된 동적 프레디킷에 의해 엘리먼트 e_A, e_B, e_C 중에서 액세스가 허용된 인스턴스들만이 // 연산자의 입력으로 전달된다.

동적 프레디킷 푸시-다운 연산자 Ψ는 구해진 질의 결과의 최근접 조상 권한이 이전 질의 결과의 최근접 조상 권한과 다른지 검사한다. 다를 경우에만 연산자 Ψ는 동적 프레디킷을 새로 구성하여 질의 계획의 σ_{DP} 연산자로 푸시-다운한다. Ψ 연산자의 알고리즘은 상용 DBMS의 질의 처리기에서 사용하는 *iterator* 모델[16]을 사용하여 표현된다. Iterator 모델에서 질의 계획을 구성하는 연산자는 하위 연산자의 결과를 입력으로 전달 받아 처리한 후, 처리 결과를 다시 상위 연산자의 입력으로 전달해 주는 방식으로 질의를 처리한다. 이때, 하위 연산자로부터 결과를 하나씩 얻어올 수 있도록 각각의 연산자는 인터페이스로 GetNext() 함수를 제공한다.

그림 7은 연산자 Ψ의 GetNext() 함수를 나타내며, 이를 *Dynamic Predicate Filtering*으로 부른다. 이 알고리즘은 크게 두 단계로 구성된다. 첫번째 단계에서는 연산자 P의 GetNext() 함수를 호출함으로써 현재의 DP를 질의 계획의 하위 연산자 P에 푸시-다운하고 액세스가 허용된 질의 결과를 하나씩 얻어온다(라인 2~6). 이때, DP는 GetNext() 함수의 인수(argument)를

```
/* DP: a Dynamic Predicate */
01:  Ψ.GetNext() {
        /* Ψ has only one child */
02:     Let P be the child operator of the DP push-down operator Ψ;
        /* initially, no DP has been constructed */
03:     if Ψ.GetNext() is first called then
04:        Retrieve the first candidate query result e_candidate;
05:        Construct an initial DP for the element e_candidate using Definition 5;
        /* get an authorized query result and push down the current DP */
        /* P.GetNext() returns a result of P and
           recursively transmits the DP to σ_DP operators */
06:     e := P.GetNext(DP);
07:     if ( start(e) is not contained in [start_begin, start_end) of DP ) then
           /* reconstruct the DP */
08:        Construct the DP for the element e using Definition 5;
09:     else return e;   /* return an authorized query result */
10:  }
```

그림 7 개선된 액세스 통제 알고리즘 Dynamic Predicate Filtering



(a) 질의 결과(11,13)에 의해 생성된 DP의 푸시-다운

(b) 질의 결과(30.32)에 의해 생성된 DP의 푸시-다운

그림 8 알고리즘 Dynamic Predicate Filtering의 수행 예

통하여 하위 연산자로 전달된다. 이 *DP*는 모든 σ_{DP} 연산자까지 연속적으로 전달된다. 연산자 Ψ의 GetNext() 함수가 처음 호출된 경우에는, 첫번째 질의 결과 후보인 $e_{candidate}$를 검색한 후에 $e_{candidate}$에 대하여 정의 5에 따라 초기 *DP*를 구성한다(라인 3~5). 두번째 단계에서는 구해진 질의 결과 *e*에 대한 *DP*를 다시 구성해야 하는 경우에만 정의 5에 따라 다시 *DP*를 구성한다(라인 7~8). *DP*의 $[start_{begin}, start_{end})$는 이전 질의 결과와 동일한 최근접 조상 권한을 가지는 엘리먼트들의 start 범위이므로, 질의 결과 *e*의 start 값이 이 범위에 포함되는지 검사함으로써 *DP*를 다시 구성해야 하는 엘리먼트를 쉽게 결정할 수 있다(라인 7).

예 3. 그림 5의 XML 문서에 *//patient//drug* 질의를 *analysis* 사용목적을 위해 수행하는 경우, 알고리즘 Dynamic Predicate Filtering의 수행 과정은 그림 8과 같다. 그림 7의 라인 4에서 첫번째 질의 결과 후보인 *drug* 엘리먼트 (11,13)을 검색한다. 라인 5에서 정의 5에 따라 예 2에서와 같이 *DP* = ([2,22), -)를 얻는다. 라인 6에서 *P*.GetNext()를 호출함으로써 *DP* = ([2,22), -)를 연산자 σ_{DP}에 모두 푸시-다운한다. 여기에서 연산자 *P*는 // 연산자이다. 이로 인해, start 값이 [2,22)에 포함된 *patient*, *drug* 엘리먼트인 (3,21), (11,13), (14,16), (17,19)들은 질의 처리 대상에서 제외된다. 따라서, [2,22) 다음에 위치한 *drug* 엘리먼트 (30,32)를 읽어온다. 라인 8에서 *DP* = ([22,45), +)이 새롭게 구성되며, 이 *DP*는 다음 번에 Ψ.GetNext()가 호출될 때 사용된다. □

Dynamic Predicate Filtering은 (1) 엘리먼트가 start 값의 순서로 스캔되고 (2) 질의 결과가 start 값의 순서로 반환되는 조건을 만족하는 질의 계획에 최적으로 적

용된다. 대부분의 구조적 조인 알고리즘[12-14]은 이 조건을 만족한다.[3] 따라서, 본 논문의 방법은 XML 질의 처리를 위해 가장 널리 사용되는 알고리즘인 구조적 조인의 특성을 충분히 활용한다.

Dynamic Predicate Filtering은 $//e_A[.//e_B//e_C]//e_D$와 같은 트위그(twig) 질의[17]에도 적용될 수 있다. 트위그 질의는 연속적인 두 단계를 거쳐 처리된다[17]. 첫번째 단계에서는 개별적인 선형 패스(앞의 예에서는 $//e_A//e_B//e_C$와 $//e_A//e_D$)를 추출하고, 각각의 선형 패스에 대한 질의 결과를 얻어낸다. 두번째 단계에서는 각각의 선형 패스 별로 얻어진 질의 결과를 병합-조인한다. Dynamic Predicate Filtering은 오직 선형 패스 질의만을 다루는 첫번째 단계에서 수행하면 된다.

3.3 갱신(update) 연산 처리

XML 데이타가 갱신되면 권한이 부여되어 있던 데이타 엘리먼트의 (*start*, *end*) 값이 바뀔 수 있으며, 이를 권한 인덱스에도 반영해주어야 한다. 특히, 데이타 엘리먼트가 삽입되면 같은 XML 문서에 있는 많은 데이타 엘리먼트들의 (*start*, *end*) 값이 바뀔 수 있다. 이로 인해, XML 데이타 및 권한 인덱스를 갱신하는 비용이 커지게 된다. 이와 같은 문제는 액세스 통제 방법의 문제라기 보다는 번호화 방식에 내재하는 문제이다. 최근 들어, 기본적인 번호화 방식을 갱신 연산을 효율적으로 처리할 수 있도록 개선하는 연구가 활발히 진행되고 있다. 대표적인 연구로는 인접한 데이타 엘리먼트의 (*start*, *end*) 번호 사이에 여유 간격을 남겨두는 방법[12]과 데이타 엘리먼트의 (*start*, *end*) 번호로 정수를 사용하지 않고 부동-소수점수(floating-point number)를 사용하는 방법[18]이 제시되었다. 이들 방법들에서는 새로운 데이타 엘리먼트가 삽입되어도 다른 데이타 엘리먼트의 (*start*, *end*) 값에 영향을 주지 않는다. 본 논문에서 제안한 액세스 통제 방법은 Li와 Moon[12] 혹은 Amagasa 등[18]의 개선된 번호화 방식을 그대로 사용할 수 있으므로, 갱신 연산에 소요되는 비용을 최소화 할 수 있다.

한편, XML 데이타 엘리먼트에 권한이 부여되거나 권한이 취소될 경우에는, 해당 엘리먼트의 (*start*, *end*) 값을 권한 인덱스(즉, 2-차원 인덱스)에 추가하거나 삭제하면 된다.

4. 성능 평가

본 절에서는 제안한 액세스 통제 방법의 성능을 기존의 방법과 비교하여 성능 평가 결과를 제시한다. 제4.1절에서는 성능 평가를 수행한 실험 데이타와 실험 환경을 소개하고, 제4.2절에서는 실험 결과를 설명한다.

4.1 실험 데이타 및 실험 환경

본 실험에서는 하향식(top-down) 및 상향식(bottom-up) 액세스 통제 방법[7], CAM 방법[10], Nearest Ancestor Filtering[4], Dynamic Predicate Filtering을 비교한다.[4] 비교를 위한 척도는 각 방법을 사용하여 액세스 통제를 수행할 때의 질의 처리 시간이다.

데이타셋의 종류에 따른 성능 변화를 측정하기 위해, 합성 데이타인 XMark[19] 벤치마크 데이타와 실제 데이타인 TreeBank[20] 데이타를 사용한다. XMark 벤치마크 데이타는 하나의 큰 XML 문서로 구성되며, 본 실험에는 **데이타베이스 크기** 변화에 따른 성능 변화를 측정하기 위해 10MB, 100MB, 1GB의 XML 문서를 사용한다. 이 데이타는 유사한 형태의 서브트리가 동일한 레벨에서 여러번 반복되어 나타나는 특성을 가진다. TreeBank 데이타는 약 86MB의 XML 문서로 구성되며, XML 엘리먼트가 여러 레벨로 반복적으로 중첩되어 있는 특성을 가진다.

명시적 권한의 개수에 따른 성능 변화를 측정하기 위해, 데이타셋의 전체 엘리먼트의 개수에 대한 명시적 제공자 프라이버시 권한이 부여된 엘리먼트의 개수의 비율을 0.01%에서 100%까지 변화시킨다. 권한 (o, p)에서 o를 결정하기 위해, 데이타셋에서 0.01%~100%의 엘리먼트를 랜덤하게 선택한다. 이때, 모든 엘리먼트에 명시적 혹은 묵시적 권한이 부여되도록 보장하기 위해 루트 엘리먼트를 먼저 선택한다. 권한 충돌이 발생하지 않는다면 여러 개의 권한이 동일한 엘리먼트에 부여될 수 있다. 권한 충돌은 동일한 사용목적의 긍정적 권한과 부정적 권한을 한 엘리먼트에 함께 부여하려 할 때 발생한다. 권한 (o, p)에서 p를 결정하기 위해, 5개의 사용목적 계층 구조에서 균등하게 사용목적을 선택한다. 또한, 사용목적 계층 구조의 개수를 40까지 늘려가며 실험을 수행한다. 긍정적 프라이버시 권한과 부정적 프라이버시 권한은 9 : 1의 비율로 부여한다.

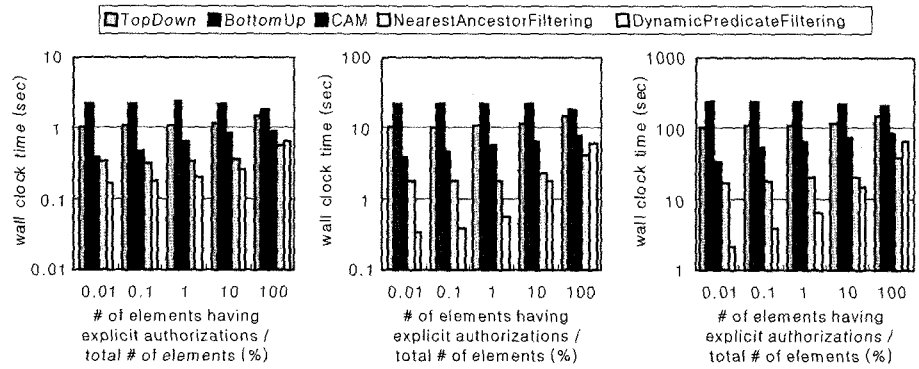
CAM 방법을 위해서는 본 모델에서의 묵시적 권한을 CAM에서의 명시적 권한으로 간주한다. 이는 CAM에서는 모든 권한이 명시적으로 부여되어야 하기 때문이다. 그러면, 압축 후에 본 모델의 명시적 권한에 해당하는 권한들만이 CAM에 의해 색인된다. 따라서, CAM에 의

3) 이 조건을 만족하지 않는 경우에는, 새로운 동적 프레디킷이 이전의 동적 프레디킷을 대치하는 대신 이전의 동적 프레디킷을 축적하도록 Dynamic Predicate Filtering을 간단하게 확장할 수 있다. (하지만 성능은 약간 감소할 것이다.)

4) 참고문헌 [9]의 방법은 본 모델에 적용할 수 없기 때문에 본 실험에서 비교하지 않는다. 참고문헌 [9]의 방법에서는 인스턴스 레벨의 권한이 미리 정의된 패턴에 따라 부여되고, 이 패턴이 액세스 통제 성능을 향상시키기 위해 사용된다. 반면, 본 모델에서는 인스턴스 레벨의 권한이 데이타 제공자의 다양한 의도를 나타내기 위해 사용되므로 인스턴스 레벨의 권한에 특정한 패턴이 존재하지 않는다.

dataset	simple queries	complex queries
XMark	//person//interest	(1) //site//open_auctions//open_auction//bidder//increase (2) //open_auctions[.//bidder]//seller
TreeBank	//NP//NN	//SBAR//S//VP//PP//NP

그림 9 실험에 사용한 XML 질의



(a) 10MB database. (b) 100MB database. (c) 1GB database.

그림 10 XMark 벤치마크 데이타에서 *//person//interest* 질의의 처리 시간

해 색인된 권한은 본 모델에서 권한 인덱스에 의해 색인되는 권한과 동일해진다. CAM은 권한을 색인하기 위해 권한이 부여된 엘리먼트의 패스 문자열을 저장하는 트라이(trie)를 구축한다.

질의 형태에 따른 성능 변화를 측정하기 위해, $//e_A//e_B$ 형태의 단순한 질의와 $//e_A//e_B//e_C//e_D//e_E$ 형태(선형 패스 표현식) 혹은 $//e_A[.//e_B]//e_C$ 형태(분기 패스 표현식)의 복잡한 질의를 수행한다. 같은 형태의 질의들은 유사한 경향을 가지므로, 각각의 데이타셋 별로 대표적인 경향을 나타내는 그림 9의 질의에 대해서 실험 결과를 제시한다.

실험은 450MHz CPU와 512MB 메모리를 가진 SUN Ultra 60 워크스테이션에서 수행한다. 권한 인덱스를 위한 다차원 색인으로는 MLGF[21,22]를 사용하며,[5] 데이타 및 색인 페이지의 크기는 4096 바이트로 한다. CAM 방법을 위한 트라이(trie)를 구현하기 위해서는 SP-GiST[23]를 사용한다. 하향식 및 상향식 방법을 위해서는 참고문헌 [24]와 같이 권한을 권한 테이블에 저장하고, 주어진 엘리먼트에 부여된 권한을 찾기 위해 B^+-트리 인덱스를 통해 권한 테이블을 액세스한다.

5) 본 실험에서는 MLGF를 사용하나, 다차원 점을 다룰 수 있는 다른 색인도 마찬가지로 사용할 수 있다. 그러한 인덱스의 예로는 R-tree, buddy tree, quad tree 등이 있다[6].

4.2 실험 결과

데이타베이스 크기와 명시적 권한의 개수에 따른 효과

그림 10은 데이타베이스 크기와 명시적 권한의 개수 변화에 따른 *//person//interest* 질의의 처리 시간을 나타낸다. Dynamic Predicate Filtering은 다른 방법들에 비해 크게 향상된 성능을 보인다. Dynamic Predicate Filtering은 기존의 XML 액세스 통제 방법에 비해 그림 10(a)에서는 2.3~13.4배의 성능 향상을 보였고, 그림 10(b)에서는 2.4~66.7배의 성능 향상을 보였으며, 그림 10(c)에서는 2.4~109.0배의 성능 향상을 보였다. 기존 방법과의 성능 차이는 데이타베이스 크기가 커질수록 증가한다. 이는 데이타베이스 크기가 커질수록 같은 레벨에서 제거된 서브트리에 포함된 엘리먼트의 개수가 증가하므로, 하나의 동적 프레디킷에 의해 질의 처리에서 제외되는 엘리먼트의 개수가 증가하기 때문이다. 이러한 효과는 명시적 권한의 개수가 적을때 더 두드러진다. 이는 오버라이드되는 묵시적 권한의 개수가 더 적어지므로 하나의 동적 프레디킷에 의해 제외되는 엘리먼트의 개수가 더 커지기 때문이다.

명시적 권한이 모든 엘리먼트(100%)에 부여되어 있을 때, Dynamic Predicate Filtering은 Nearest Ancestor Filtering에 비해 약간(1.6배 이하)의 성능 저하를 보인다. 이는 Dynamic Predicate Filtering에서 동적 프레디킷을 구성하는 것은 아무 이득 없이 최근접 오버라이드 권한을 검색하는 오버헤드를 유발하기 때문이다.

XML 데이타베이스에서는 데이타 계층 구조를 활용하여 많은 수의 엘리먼트가 묵시적 권한을 부여 받는다. 모든 엘리먼트마다 명시적 권한을 부여하는 경우는 매우 드물다. Yu 등[10]에 의해 보여진 CAM의 높은 압축률은 이러한 주장을 뒷받침한다. 또한, 명시적 권한이 70% 이하의 엘리먼트에 부여되어 있을 때, Dynamic Predicate Filtering은 Nearest Ancestor Filtering보다 좋은 성능을 보였다. 그림 10에서 Nearest Ancestor Filtering의 질의 처리 시간이 100%일 때 다소 증가하였는데, 이는 권한 인덱스의 깊이가 1 증가하였기 때문이다.

Nearest Ancestor Filtering은 CAM과 비슷한 성능을 보인다. 이는 제4.1절에서 논의한 바와 같이 CAM은 압축 후에 권한을 색인하는 트라이 구조로서 권한 인덱스와 유사한 역할을 수행하기 때문이다. 약간의 성능 향상은 인덱스 구조의 효율성과 최근접 조상을 찾는 효율성에 기인한다. 한편, Dynamic Predicate Filtering은 CAM에 비해 1.3~14.8배의 성능 향상을 보였다. 왜냐하면, Dynamic Predicate Filtering은 액세스가 허용되지 않은 엘리먼트를 질의 처리에서 일찍 제외하여 질의 처리 시간을 줄여주는 반면, CAM은 그러지 못하기 때문이다.

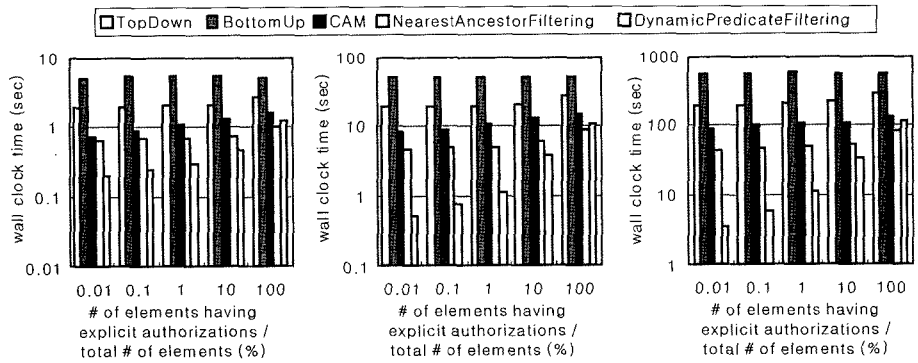
질의 형태에 따른 효과

그림 11은 데이타베이스 크기와 명시적 권한의 개수 변화에 따른 길이가 긴 경로식을 가지는 보다 복잡한 형태의 질의인 *//site//open_auctions//open_auction//bidder//increase* 질의의 처리 시간을 나타낸다. Dynamic Predicate Filtering은 단순한 질의보다 복잡한 질의에서 다른 방법들에 비해 더 큰 차이로 좋은 성능을 보인다. 이는 Dynamic Predicate Filtering이 액세스가 허용되지 않은 엘리먼트를 질의 처리 도중에 일찍 제거하므로 복잡한 질의에서의 질의 처리 시간을 보다 더 효과적으로 줄여주기 때문이다. Dynamic Predicate Filtering은 기존의 XML 액세스 통제 방법에 비해 그림 11(a)에서는 2.2~26.2배의 성능 향상을 보였고, 그림 11(b)에서는 2.6~101.4배의 성능 향상을 보였으며, 그림 11(c)에서는 2.4~160.4배의 성능 향상을 보였다.

트위그(twig) 질의 *open_auctions[.//bidder]//seller*의 처리 시간은 그림 10과 유사한 경향을 보인다. 트위그 질의에서 Dynamic Predicate Filtering은 각각의 선형 패스에 적용되므로, 다른 방법들에 비하여 Dynamic Predicate Filtering의 전체 성능 향상은 각각의 선형 패스로부터 얻어진 성능 향상의 합이 된다. 100MB의 XMark 데이타에서 Dynamic Predicate Filtering은 CAM에 비해 1.1~17.2배의 성능 향상을 보였고, Nearest Ancestor Filtering에 비해 0.9~7.3배의 성능 향상을 보였다.

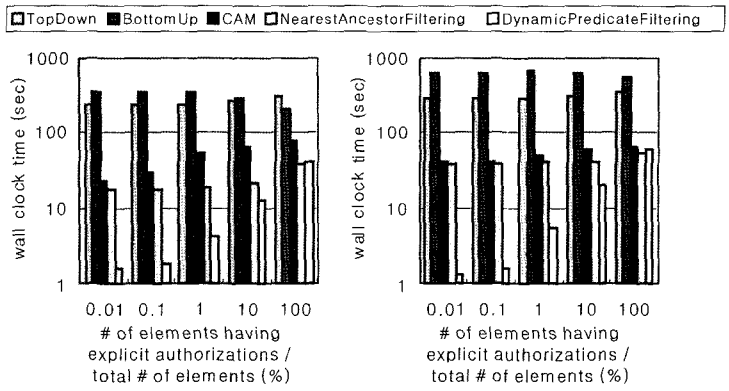
XML 데이타의 깊이에 따른 효과

그림 12는 TreeBank 데이타(86MB)에 대한 명시적 권한의 개수 변화에 따른 *//NP//NN* 질의와 *//SBAR//S//VP//PP//NP* 질의의 처리 시간을 나타낸다. Dynamic Predicate Filtering은 다른 방법들에 비해 TreeBank 데이타와 같이 XML 엘리먼트가 여러 레벨로 반복적으로 중첩되어 있는 데이타에서 특히 더 좋은 성능을 보인다. 그림 12 (b)에서 Dynamic Predicate Filtering은 하향식 방법에 비해 6.0~219.1배의 성능 향상을 보였고, 상향식 방법에 비해 9.4~499.2배의 성능 향상을 보였고, CAM에 비해 1.1~31.2배의 성능 향상을 보였으며, Nearest Ancestor Filtering에 비해 0.9~



(a) 10MB database. (b) 100MB database. (c) 1GB database.

그림 11 XMark 벤치마크 데이타에서 *//site//open_auctions//open_auction//bidder//increase* 질의의 처리 시간



(a) *//NP//NN* query. (b) *//SBAR//S//VP//PP//NP* query.

그림 12 TreeBank 데이타(86MB)에서 단순한 질의와 복잡한 질의의 처리 시간

29.2배의 성능 향상을 보였다. 그림 12에서 Dynamic Predicate Filtering과 다른 방법들간의 성능 차이가 그림 10과 그림 11에 비해 더 커진것을 알 수 있다. 이는 많은 레벨을 가지는 XML 데이타에서는 상위 레벨에서 제거된 서브트리에 포함된 엘리먼트의 개수가 증가하므로, 하나의 동적 프레디킷에 의해 제외되는 엘리먼트의 개수가 증가하기 때문이다.

권한 타입과 사용목적 계층 구조의 개수에 따른 효과

본 논문에서 수행한 모든 실험에 대하여, 긍정적 권한과 부정적 권한의 비율을 9:1에서 1:9로 변화시키며 동일한 실험을 수행하였다. Dynamic Predicate Filtering의 성능은 평균적으로 약 10.8%가 향상됨을 보였다. 이는 질의 처리에서 제외되는 대부분의 엘리먼트가 질의 사용목적과 다른 사용목적의 권한이 부여된 엘리먼트이기 때문에, 부정적 권한이 성능에 그리 큰 영향을 미치지 못하기 때문이다. 또한, 사용목적 계층 구조의 개수를 5에서 40으로 변화시키며 역시 동일한 실험을 수행하였다. Dynamic Predicate Filtering의 성능은 사용목적 계층 구조의 개수가 증가함에 따라 최대 13.1%까지 향상됨을 보였다.

5. 결 론

본 논문에서는 히포크라테스 XML 데이타베이스 모델에서의 질의 처리 성능을 개선하기 위해 **동적 프레디킷**(*dynamic predicate*)이라는 새로운 개념을 제안하였다. 동적 프레디킷은 질의 처리 도중에 동적으로 생성되는 조건을 나타내며, 권한에 의해 엘리먼트의 액세스가 허용되는지를 결정하는데 적용될 수 있다. 동적 프레디킷은 액세스 통제를 질의 계획에 효율적으로 통합할 수 있게 해줌으로써, 액세스가 허용되지 않은 엘리먼트를 질의 처리 과정에서 일찍 제외시켜 질의 처리 성능을 크게 향상시킨다. 동적 프레디킷을 사용함의 정당성을 보조정리 3에서 증명하였다. 그리고, 개선된 액세스 통제 방법을 알고리즘 Dynamic Predicate Filtering으로 구현하였다.

제안한 액세스 통제 방법의 효율성을 입증하기 위해, 합성 데이타셋과 실제 데이타셋을 사용하여 많은 실험을 수행하였다. 실험 결과, Dynamic Predicate Filtering은 하향식 액세스 통제 방법에 비해 최대 219배, 상향식 액세스 통제 방법에 비해 최대 499배, CAM에 비해 최대 31배, Nearest Ancestor Filtering에 비해 최대 29배까지 성능을 향상시킴을 보였다. 그리고, Dynamic Predicate Filtering의 장점은 (1) 데이타베이스의 크기, (2) 질의에 사용된 경로식의 길이, (3) XML 데이타의 깊이가 증가할수록 부각됨을 보였다. Dynamic Predicate Filtering의 성능은 긍정적 권한에 대한 부정적 권한의 비율에 거의 영향을 받지 않고, 큰 비율에서 약간 성능이 향상됨을 보였다. 이러한 결과들은 동적 프레디킷을 사용한 질의 처리 방법의 효율성을 입증한 것이다.

본 논문의 주요 공헌은 히포크라테스 XML 데이타베이스 모델 상에서 동적 프레디킷이라는 개념을 사용하여 액세스 통제 방법을 질의 계획에 효과적으로 통합하는 기법을 제안한 것이다. 본 연구는 상용 XML DBMS에 구현될 수 있는 실용적인 방법으로 사료된다.

참 고 문 헌

[1] Information and Privacy Commissioner of Ontario, "Intelligent Software Agents: Turning a Privacy Threat into a Privacy Protector," Apr. 1999.

[2] Information and Privacy Commissioner of Ontario, "An Internet Privacy Primer: Assume Nothing," Aug. 2001.

[3] Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y., "Hippocratic Databases," In *Proc. 28th Int'l Conf. on Very Large Data Bases*, Hong Kong, China, pp. 143-154, Aug. 2002.

[4] 이재길, 한욱신, 황규영, "히포크라테스 XML 데이타베이스: 모델 및 액세스 통제 방법", *정보과학회논문지: 데이타베이스*, 제31권, 제6호, pp. 684-698, 2004년 12월.

[5] Roussopoulos, N., Kelley, S., and Vincent, F., "Nearest Neighbor Queries," In *Proc. 1995 ACM SIGMOD Int'l Conf. on Management of Data*, ACM SIGMOD, San Jose, California, pp. 71-79, June 1995.

[6] Gaede, V. and Gunther, O., "Multidimensional Access Methods," *ACM Computing Surveys*, Vol. 30, No. 2, pp. 170-231, June 1998.

[7] Bertino, E., Castano, S., Ferrari, E., and Mesiti, M., "Specifying and Enforcing Access Control Policies for XML Document Sources," *World Wide Web Journal*, Vol. 3, No. 3, pp. 139-151, 2000.

[8] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., and Samarati, P., "A Fine-Grained Access Control System for XML Documents," *ACM Trans. on Information and System Security*, Vol. 5, No. 2, pp. 169-202, May 2002.

[9] Cho, S., Amer-Yahia, S., Lakshmanan, V. S., and Srivastava, D., "Optimizing the Secure Evaluation of Twig Queries," In *Proc. 28th Int'l Conf. on Very Large Data Bases*, Hong Kong, China, pp. 490-501, Aug. 2002.

[10] Yu, T., Srivastava, D., Lakshmanan, V. S., and Jagadish, H. V., "Compressed Accessibility Map: Efficient Access Control for XML," In *Proc. 28th Int'l Conf. on Very Large Data Bases*, Hong Kong, China, pp. 478-489, Aug. 2002.

[11] Berglund, A., Boag, S., Chamberlin, D., Fernández, M. F., Kay, M., Robie, J., and Siméon, J., XML Path Language (XPath) Version 2.0, W3C Working Draft, Nov. 2003.

[12] Li, Q. and Moon, B., "Indexing and Querying XML Data for Regular Path Expressions," In *Proc. 27th Int'l Conf. on Very Large Data Bases*, Rome, Italy, pp. 361-370, Sept. 2001.

[13] Al-Khalifa, S., Jagadish, H. V., Koudas, N., Patel, J. M., Srivastava, D., and Wu, Y., "Structural Joins: A Primitive for Efficient XML Query Pattern Matching," In *Proc. 18th Int'l Conf. on Data Engineering*, San Jose, California, pp. 141-152, Feb. 2002.

[14] Chien, S.-Y., Vagena, Z., Zhang, D., Tsotras, V. J., and Zaniolo, C., "Efficient Structural Joins on Indexed XML Documents," In *Proc. 28th Int'l Conf. on Very Large Data Bases*, Hong Kong, China, pp. 263-274, Aug. 2002.

[15] Wu, Y., Patel, J. M., and Jagadish, H. V., "Structural Join Order Selection for XML Query Optimization," In *Proc. 19th Int'l Conf. on Data Engineering*, Bangalore, India, pp. 443-454, Mar. 2003.

[16] Graefe, G., "Query Evaluation Techniques for Large Databases," *ACM Computing Surveys*, Vol. 25, No. 2, pp. 73-170, June 1993.

[17] Bruno, N., Koudas, N., and Srivastava, D., "Holistic Twig Joins: Optimal XML Pattern Matching," In *Proc. 2002 ACM SIGMOD Int'l Conf. on Management of Data*, ACM SIGMOD, Madison, Wisconsin, pp. 310-321, June 2002.

[18] Amagasa, T., Yoshikawa, M., and Uemura, S., "QRS: A Robust Numbering Scheme for XML Documents," In *Proc. 19th Int'l Conf. on Data Engineering*, Bangalore, India, pp. 705-707, Mar. 2003.

[19] Schmidt, A. R., Waas, F., Kersten, M. L., Carey, M. J., Manolescu, I., and Busse, R., "XMark: A Benchmark for XML Data Management," In *Proc. 28th Int'l Conf. on Very Large Data Bases*, Hong Kong, China, pp. 974-985, Aug. 2002.

[20] Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B., "Building a Large Annotated Corpus of English: The Penn Treebank," *Computational Linguistics*, Vol. 19, No. 2, pp. 313-330, June 1993.

[21] Whang, K.-Y. and Krishnamurthy, R., Multilevel Grid Files, IBM Research Report RC11516, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, Nov. 1985.

[22] Whang, K.-Y. and Krishnamurthy, R., "The Multilevel Grid File - A Dynamic Hierarchical Multidimensional File Structure," In *Proc. Int'l Conf. on Database Systems for Advanced Applications*, Tokyo, Japan, pp. 449-459, Apr. 1991.

[23] Aref, W. G. and Ilyas, I. F., "SP-GiST: An Extensible Database Index for Supporting Space Partitioning Trees," *Journal of Intelligent Information Systems*, Vol. 17, No. 2-3, pp. 215-240, Dec. 2001.

[24] Rabitti, F., Bertino, E., Kim, W., and Woelk, D., "A Model of Authorization for Next-Generation Database Systems," *ACM Trans. on Database Systems*, Vol. 16, No. 1, pp. 88-131, Mar. 1991.

이 재 길
정보과학회논문지 : 데이타베이스
제 32 권 제 3 호 참조

한 욱 신
정보과학회논문지 : 데이타베이스
제 32 권 제 4 호 참조

황 규 영
정보과학회논문지 : 데이타베이스
제 32 권 제 4 호 참조