# 영역 할당 기법을 이용한 효율적인 경로 테이블 기법

민 준 기†


## 요 약

XML은 인터넷 상에서 데이터 표현 및 교환의 표준으로 떠오르고 있어서 XML 데이터의 양이 급속히 증가하고 있다. 따라서, XML 데이터에 대한 효율적인 저장 및 검색이 필요하다. 관계형 데이터베이스를 이용하는 XRel과 같은 XML 저장 관리 기법에서는 단순히 모든 레이블 경로들을 저장함으로써, 다양한 형태의 경로 표현식을 효율적으로 처리하지 못한다. 본 논문에서는 관계형 데이터베이스를 이용하여 XRel에서 제안된 경로 테이블 기법 보다 효율적인 데이터 저장 및 검색 기법을 제시한다. 본 논문에서 제안하는 기법은 XML 경로 인덱스를 관계형 데이터베이스에 저장하고 레이블 경로를 경로 식별자로 대치함으로써 다양한 형태의 XML 질의들을 기존의 방식에 비하여 보다 효율적으로 처리할 수 있도록 하였다. 또한 제안된 방식은 관계형 데이터베이스 엔진의 수정을 요구하지 않으며, 기존의 방식에 비하여 보다 적은 디스크 공간을 소비한다. 우리의 실험 결과는 제안된 기법이 기존의 기법에 비하여 좋은 질의 성능을 나타냄을 보인다.


**키워드 : XML, 관계형 데이터베이스, 경로 인덱스**

# An Effective Path Table Method Exploiting the Region Numbering Technique

Jun-Ki Min†


## ABSTRACT

Since XML is emerging as the de facto standard for exchanging and representation of data on the web, the amount of XML data has rapidly increased. Thus, the need for effective store and retrieval of XML data has arisen. Since the existing techniques such as XRel which is an XML storage and management technique using RDBMS simply record the existing all label paths, diverse classes of label path expressions could not be efficiently supported. In this paper, we present a technique which supports storage and retrieval for XML data using RDBMS efficiently compared with the existing approaches. Since the proposed technique keeps the XML path index on the relational database and replace label paths with path identifiers, diverse XML queries can be evaluated compared with existing approaches. Also, the proposed technique does not require the modification of the relational database engine and consumes the disk space less. Our experimental result demonstrates the better query performance compared with existing techniques.


**Key Words : XML, Relational Database, Path Index**

## 1. 서 론

다양한 형태의 전자 데이터들의 양이 급속하게 증가함에 따라서, 다양한 분야에서 수집, 생성된 정보들을 공유하고, 활용하고자 하는 상호 운영성에 대한 필요성이 대두되었다. 따라서 W3C(WWW Consortium)에서는 정보 표현 및 교환의 표준 포맷으로 XML(eXtensible Markup Language)[3]을 제안하였다. XML은 표현 능력의 유연성 때문에 인터넷 기반의 많은 응용 분야에서 사용되고 있다. 예로써, 멀티미디어 분야의 MPEG7[11], 지리 정보 관리 분야의 GML[6], 차세대 인터넷인 Semantic Web의 RDF[1], OWL[12]와 같은 다양한 응용 분야에서 XML 기반의 데이터 표현 명세서들을 제안하고 있다.

이러한 XML 데이터의 폭넓은 사용은 XML 데이터의 저장, 검색과 같은 새로운 데이터 관리 요구사항들을 만들어 냈다. 이러한 요구사항을 만족하기 위하여, 다양한 XML 저장 및 검색 기법들이 제안 되었다. XML 문서를 저장하는 방법으로는 일반적으로 파일 시스템을 사용하는 방법, 기존의 객체지향 데이터베이스나 관계형 데이터베이스를 사용하는 방법, XML 만을 지원하는 XML용 데이터베이스를 사용하는 방법이 있다. 이 방법들 중에서 관계형 데이터베이스에 XML 데이터를 저장하는 방법에 대한 연구가 가장 많이

※ 이 논문은 2005년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2005-003-D00280).
† 정 회 원 : 한국기술교육대학교 인터넷미디어공학부
논문접수 : 2005년 9월 1일, 심사완료 : 2006년 2월 17일

진행되어오고 있다[14]. 그 이유는 관계형 데이터베이스가 데이터의 저장 및 관리를 위하여 가장 많이 사용되고 있으며 오랫동안 연구되어서 안정적이며 최적화된 데이터 저장 관리 기법을 사용하고 있기 때문이다.

이 논문에서는 관계형 데이터베이스에서 XML 데이터의 효율적인 검색을 위한 경로 테이블 기법에 대하여 기술한다.

XML 데이터를 검색하기 위하여 XPath[4], XQuery[2] 와 같은 언어들이 W3C에서 제안되었다. XML 질의 언어들은 트리 형태의 모델을 따르는 XML 데이터를 검색하기 위하여 경로 표현식 (Path Expression)을 사용한다. 경로 표현식을 효율적으로 처리하기 위하여 다양한 경로 인덱스들이 제안되었다. 일반적으로, 경로 인덱스는 XML 데이터의 각 레이블 경로(label path) 및 해당 경로로 접근할 수 있는 노드들의 집합을 유지한다. 따라서, 질의 처리 시에 XML 데이터의 관련 있는 부분만을 탐색하도록 제한함으로 질의 수행 성능을 향상시킨다. 이러한 경로 인덱스의 장점에도 불구하고, 경로 인덱스의 자료 구조가 비정형적 그래프에 기반하고 있어서 관계형 데이터베이스에서는 채용되고 있지 못하다.

경로 인덱스 대신에 질의 성능을 향상하기 위하여 경로 테이블[20]이 제안되었다. 그러나 기존의 경로 테이블에서는 단순히 레이블 경로와 경로 식별자 쌍만을 유지하여 레이블 경로 간의 구조적 정보를 활용하지 못함으로써 임의의 경로 표현식을 효율적으로 지원하지 못하는 단점이 있다.

본 논문에서는 영역 할당 기법을 이용하여 각 경로 간의 구조적 정보를 유지할 수 있도록 함으로써 보다 다양한 형태의 경로 표현식을 효율적으로 지원할 수 있도록 하였다. 또한, 제안하는 기법은 각 경로 간의 구조 정보를 유지함에도 불구하고 기존의 경로 테이블 기법과 마찬가지로 관계형 데이터베이스 엔진의 수정을 요구하지 않으며, 기존의 기법보다도 저장 공간의 사용량을 감소시켰다.

## 2. 배경지식

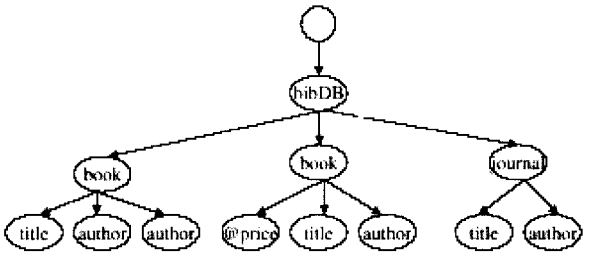
일반적으로 XML 데이터는 트리 형태로 표현될 수 있다. ID-IDREF 애트리뷰트로 표현되는 참조 관계로 인하여 XML 데이터는 그래프 형태로도 표현될 수 있다. ID 타입 애트리뷰트와 IDREF 타입 애트리뷰트는 DTD나 XML Schema와 같은 내용 기술 언어(Content Specification Language)들을 이용하여야 명시할 수 있다. 그러나, DTD나 XML Schema는 XML 데이터의 의무 사항이 아니라서 이러한 참조 관계가 나타나지 않는 문서들이 존재한다. 또한, 이러한 참조 관계는 ID 애트리뷰트의 값과 IDREF 애트리뷰트의 값의 동일-조인(equi-join)을 이용하여 손쉽게 계산할 수 있다. 따라서, 참조 관계를 표현하기 위한 노드 간의 간선은 꼭 요구되는 것은 아니다.

본 논문에서 사용되는 트리 형태의 데이터 모델[1] 에서는 XML 데이터 상의 엘리먼트나 애트리뷰트는 XML 트리 상의 노드로 나타내어지며 각 엘리먼트의 태그는 대응되는 노드의 레이블로 대응된다. 특히 애트리뷰트는 일반 엘리먼트와 구분하기 위하여 노드 레이블에 접미사로 '@'을 추가한다. 또한 XML 트리에서는 XML 문서 자체를 표현하기 위한 루트 노드가 존재한다.

1) 우리는 이것을 XML 트리 라고 칭한다.

```
<bibDB>
        <book>
                <title> title1 </title>
                <author> author1 </author>
                <author> author2 </author>
        </book>
        <book price = 50>
                <title> title 2</title>
                <author> author3 </author>
        </book>
        <journal>
                <title> title3 </title>
                <author> author4 </author>
        </journal>
</bibDB>
```

(그림 1) 간단한 XML Data




(그림 2) XML 데이터의 트리 표현

(그림 1)은 간단한 XML 데이터의 예이고 (그림 2)는 그에 대응하는 XML 트리이다. 단순성을 위하여 (그림 2)에서 데이터 값에 해당하는 노드들은 생략하였다.

## 3. 관련 연구

### 3.1 간선 기반 방법

관계형 데이터베이스를 이용하여 XML 데이터 효율적으로 저장, 검색하고자 하는 첫 번째 시도는 간선 기반 방법[7]이다. 간선 기반 방법은 기본적으로 XML 문서를 표현한 트리 상에서 나타나는 모든 간선(edge)에 대한 정보를 Edge라고 하는 테이블에 저장한다. 이러한 Edge 테이블은 그래프상에서 각 간선의 소스(source)와 타깃(target) 노드 식별자를 저장하기 위한 source 애트리뷰트와 target 애트리뷰트, 간선의 이름을 저장 하기 위한 name 애트리뷰트 등으로 구성된다. 간선 기반 방법은 각 노드 간의 부모-자식 관계를 파악하는데 있어서 효율적이지만, 조상-자손 관계는 부모-자식 관계를 이용한 연속적인 조인이 필요하다.

### 3.2 영역 기반 방법

조상-자손 관계를 효율적으로 유지하는 XML 저장 기법

중에서 가장 잘 알려진 기법은 영역 기반 기법(region based approach) 이다. 영역 기반 기법에서는 각 노드를 하나의 영역 (start, end)으로 표현하며 XML 문서에서 각 노드들의 시작 위치 값(start)과 끝 위치 값(end)으로써, 바이트 수나 단어 수들을 사용하여 노드의 시작과 끝 위치 값을 표현할 수 있다. 또한, XML 트리 상의 각 노드에 대한 전위 탐색(pre-order traversal) 순서와 후위 탐색 순서(post-order traversal)를 이용한 Dietz's numbering을 활용한 방법도 있다[10]. 그러나 이러한 기법들은 개념적으로 위에서 언급한 영역과 유사한 정보를 유지하고 있다.

간선 기반 방식이나, 영역 기반 방식은 노드 간의 관계를 표현한다. 그러나, XML 질의 언어들은 트리 형태의 모델을 따르는 XML 데이터를 검색하기 위하여 경로 표현식 (Path Expression)을 사용한다. 경로 표현식은 질의 대상 노드의 경로 정보에 대한 제약 사항을 나타난다. 그러나 위에서 언급한 저장 기법만으로는 경로 정보를 파악하기 어렵다.

### 3.3 경로 테이블 방식

경로 표현식을 효율적으로 지원하기 위하여 strong Data Guide[8], 1-index[13], APEX[5]와 같은 경로 인덱스들이 제안되었다. 이러한 경로 인덱스 기법은 XML 데이터의 경로 정보를 그래프 형태로 변환하여 이를 색인화하는 기법으로 XML 전용 시스템을 기반으로 제안된 기법이다. 따라서, 이러한 인덱스들을 관계형 데이터베이스에 적용하기 위하여서는 경로 인덱스를 위한 자료 구조 및 관련 API들를 구현하고 관계형 데이터베이스 엔진의 수정을 통하여 데이터베이스 엔진에 반영하여야 하는 단점이 존재한다.

이에 반하여 XRel[17]에서는 관계형 데이터베이스 엔진의 수정 없이 효율적으로 경로 표현식을 지원하는 경로 테이블 기법이 제안되었다. 이 경로 테이블 기법은 영역 기반 방식과 같이 제안되었으나 간선 기반 방식 등과도 직교적(orthogonally)으로 적용이 가능하다.

경로 테이블은 루트부터 시작되는 각 노드의 고유한 단순 경로(simple path)들을 유지한다. 각 경로는 이에 대응되는 고유한 경로 식별자를 가지고 있다.

(그림 3)는 영역 방식에 기반한 경로 테이블 기법의 저장 결과이다. (그림 3)의 경로 테이블에서 ">/"는 각 레이블의 구분자(delimiter)로 사용되었으며, Region 테이블에서는 각 노드의 이름 대신 경로 식별자(path id) 애트리뷰트를 이용하여 각 노드의 경로 정보를 유지하도록 하였다.

Region

| s | e | l | pathid |
|---|---|---|---|
| 1 | 24 | 1 | 1 |
| 2 | 9 | 2 | 2 |
| 3 | 4 | 3 | 3 |
| | ... | | |
| 21 | 23 | 3 | 8 |

PathTable

| Path | pathid |
|---|---|
| >/bibDB | 1 |
| >/bibDB>/book | 2 |
| >/bibDB>/book>/title | 3 |
| ... | |
| >/bibDB>/journal>/title | 7 |
| >/bibDB>/journal>/author | 8 |

(그림 3) 경로 테이블 기법의 예

이제, XML 질의를 관계형 데이터베이스의 질의 언어인 SQL문으로 변환하는 것에 대하여 예제를 이용하여 간단히 설명한다. 다음은 질의 예제 Q1을 경로 테이블 기법에서 처리하기 위한 SQL 문의 예시이다.

```
예제1 (Q1): /bibDB//title
SQL1: SELECT R.*
      FROM Region R, PathTable P
      WHERE P.path LIKE '>/bibDB>%/title'
        AND P.pathid = R.pathid
```

예제 1의 질의문 Q1은 XML 질의어로, 심볼 '/'는 부모-자식 간의 관계[2]를 나타내며 "//"는 조상-자손 관계를 의미한다 ([4] 참조). 예제 1의 SQL1에서 보이듯이 적절한 경로 식별자(pathid)는 SQL-2의 패턴 매칭을 위한 연산자, LIKE를 이용하여 얻어진다.

여기서, XML 질의어의 '/'는 ">/"로 대치되며, "//"는 ">%/"로 대치된다. LIKE 연산자에서 '%' 임의의 패턴을 말하는 것으로 '>'과 '/' 사이에 어떠한 문자열이 와도 된다는 것을 의미한다. 따라서, ">/bibDB>%/title"은 PathTable상에 path 애트리뷰트 값 중에서 ">/bibDB>/book>/title" 과 ">/bibDB>/journal>/title"로 매칭되어 pathid값이 3 및 7인 모든 노드들을 찾을 수 있게 된다.

즉 질의 Q1과 같은 경우, 경로 테이블을 이용할 경우에는 PathTable과 Region간의 2-way Join을 하면 손쉽게 그 결과를 얻을 수 있다. 그러나, 기존의 영역 기반 방식에서는 모든 bibDB, book, title 엘리먼트 정보를 Region 테이블에서 추출하고 이들간의 포함관계를 이용한 조인을 이용하여 그 결과는 얻게 되므로 3-way join을 하여야만 한다.

경로 테이블 기법은 위와 같이 "//"와 '/'의 연속인 경로 표현식을 처리하는데 있어서 효율적이다. 그러나, XML 질의어의 임의문자(wildcard) 인 '*'를 효과적으로 지원하지 못한다. '//'는 위에서 본 것과 같이 '%' 키워드를 이용하여 레이블의 연속(sequence of labels)에 대응된다. 그러나 '*'는 임의의 단일 레이블에 대응되어야 함으로 '%'를 이용하는 것은 비효율적이다. 더욱이, 주어진 경로 표현식에 대응되는 경로 식별자를 얻기 위하여 SQL 처리기는 경로 테이블 상에 있는 모든 튜플들을 검사하여야 하는 단점이 있다. 또한, 일반적으로 관계형 데이터베이스는 문자열을 위한 인덱스를 지원하지 않음으로써, 문자열 검색 비용이 높다.

### 3.4 기타 방식들

이외에도 최근에 XML 데이터에 존재하는 엘리먼트를 (symbol, prefix)로 표현하는 구조-부호화 연속(structural-encoded sequence) 기법을 이용하는 ViST[19] 방법이 제안되었다. ViST에서는 2단 구조의 B$^+$-트리를 활용하여 구조-부호화 연속을 색인화 한다. 따라서, ViST논문에서 제안하

2) 엄밀히 말하며, child 축이 생략되어 있는 것이다.

는 구조를 지원하기 위하여서는 관계형 데이터베이스 엔진을 수정하여야만 한다. 또한, ViST 방법을 확장한 PRIX[14] 방법도 제안되었다. PRIX 방법에서는 ViST 기법과 달리 Prüfer 연속을 이용하여 XML 데이터의 엘리먼트들을 표현한다 (자세한 사항은 [14] 참조). 그러나 PRIX에서도 Prüfer 연속을 저장하기 위하여 ViST에서 사용된 $B^+$-트리 구조와 유사한 구조를 사용한다. 즉, ViST와 마찬가지로 관계형 데이터베이스 엔진을 수정하여야 하는 단점이 존재한다.

지속적인 갱신이 발생되는 XML 데이터를 관계형 데이터베이스에 저장하기 방법으로 Wu 등은 소수(prime number)와 조화 방정식의 성질을 이용한 소수 할당 방법을 제안하였다 [20]. 본 논문에서 제공하는 방법은 XML 트리 상에 존재하는 경로들의 정보를 관계형 데이터베이스를 이용하여 효율적으로 저장, 관리 함으로써 다양한 경로 표현식을 효율적으로 지원하는 방식으로 XML 트리의 노드를 위한 소수 할당 방식과 직교적 (orthogonally)으로 적용 가능하다.

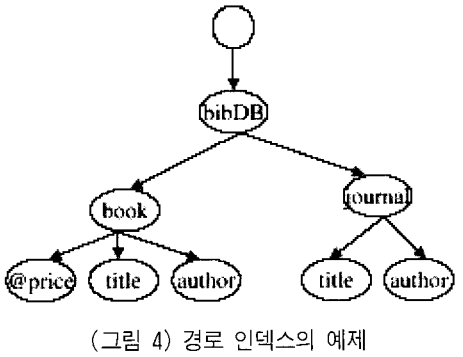
## 4. 경로간의 구조 정보를 활용한 경로 테이블 기법

이 절에서는 우리가 제안하는 관계형 데이터베이스의 수정 없이 관계형 테이블에서 경로 색인을 저장하는 기법에 대하여 기술한다.

### 4.1 제안하는 기법의 관계형 스키마

위에서 언급한 바와 같이, XML 데이터는 엘리먼트 간의 부모-자식 관계를 이용하여 트리 형태로 표현할 수 있다.

우리가 XML 데이터를 트리 형태로 간주한다면, 트리 형태의 경로 인덱스를 구축할 수 있다.경로 인덱스 상의 각 노드는 XML 트리 상의 루트 노드부터 시작되는 레이블 경로(즉, 단순 경로)를 대표한다. (그림 2)에 나타난 XML 트리를 위한 경로 인덱스는 (그림 4)과 같다. 따라서, (그림 2)에 나타나는 모든 고유한(distinct) 단순 경로는 (그림 4)의 경로 인덱스에 나타난다. 예를 들어, (그림 2)에 나타나는 단순 경로 bibDB.book.author는 (그림 4)의 book 노드의 자식인 author 노드로 표현된다. 여기서 book 노드는 bibDB 노드의 자식임을 알 수 있다. 따라서, 단순 경로 bibDB.book도 (그림 4)의 book 노드로 표현된다. 또한, 경로 인덱스 상의 각 노드 i는 루트로부터 노드 i까지의 레이블 경로를 가지고 XML 트리 상에서 접근할 수 있는 모든 노드들의 집합을 유지 한다. 예를 들어 (그림 4)의 book 노드에는 (그림 2)에서 /bibDB/book/을 이용하여 접근할 수 있는 두 개의 book 노드들에 대한 정보가 유지된다.




(그림 4) 경로 인덱스의 예제

3절에서 언급한 바와 같이, 트리는 영역 기반 접근 방식을 이용하여 저장할 수 있다. 따라서, 트리 형태의 XML 데이터의 경로 인덱스도 트리임으로 영역 기반 접근 방식을 이용하여 관계형 테이블에 저장할 수 있다.

본 논문에서 제안하는 기법은 XML 데이터를 저장 관리하기 위하여 기본적으로 다음과 같은 3가지 테이블들로 이루어진다.

PATHINDEX(path_s, path_e, path_l, path)
LABELS(label, lid, path_s)
NODE(s, e, path_s)

테이블 PATHINDEX는 경로 인덱스를 저장 한다. 경로 인덱스에 존재하는 각 노드는 레벨 값(path_l)과 함께 하나의 영역 (path_s, path_e)을 할당 받는다. 또한 이 영역들은 3절에서 언급한 바와 영역 할당 기법을 이용하여 포함 관계를 이용한 조상-자손 관계 정보를 그대로 유지한다. 기존의 영역 할당 기법에서는 각 노드의 주해(annotation)로써, 해당 노드의 레이블을 사용하였으나, 테이블 PATHINDEX에서는 루트로부터 해당 노드까지 도달하는 레이블 경로를 각 노드의 주해로써 이용한다.

위에서 언급한 바와 같이 경로 인덱스의 각 노드는 XML 트리 상의 노드들의 집합을 유지하고 있다. 그러나, 관계형 데이터베이스에서는 일반적으로 집합 타입의 애트리뷰트를 효율적으로 지원하지 않는다. 따라서, 경로 테이블 기법과 마찬가지로 경로 식별자가 필요하다. 경로 인덱스에서 모든 단순 경로는 오직 한번씩만 나타나고, 각 경로 인덱스 상의 노드는 각각의 레이블 경로와 대응된다. 따라서, 경로 인덱스 상의 노드를 위한 영역 중 path_s 값을 레이블 경로의 식별자로 사용할 수 있다.

경로 표현식을 효율적으로 처리하기 위하여 경로 테이블 접근 방식은 SQL의 부분 문자열 매칭을 지원하는 LIKE 연산자를 이용한다. 이러한 부분 문자열 매칭 부담은 스트링의 길이에 비례하여 증가된다. 따라서 이러한 부담을 줄이기 위하여, 테이블 PATHINDEX에서는 레이블 경로 그대로 유지하는 것이 아니라 레이블 식별자(label identifier)의 연속을 이용한다.

각 레이블과 레이블 식별자인 lid에 대한 대응 정보는 LABELS 테이블에서 유지한다. 부가적으로, 테이블 PATHINDEX로부터 관련 있는 path_s 값들을 효율적으로 추출하기 위하여, 레이블 경로들 중 해당 레이블로 끝나는 레이블 경로의 path_s 값들을 LABELS 테이블의 path_s 애트리뷰트에 저장한다.

NODE 테이블은 XML 트리 상의 노드들에 대한 정보를 유지한다. 특히, NODE 테이블은 각 노드에 대한 경로 정보로써 path_s 애트리뷰트를 가진다.

**Property 1** 경로 인덱스 상의 2개의 노드 v = (path_sv, path_ev, path_lv)와 u = (path_su, path_eu, path_lu)가 존재하고, path_sv < path_su < path_eu < path_ev 그리고 path_lu - path_lv = k라고 하자. 이때, XML 트리 상에 경로 식별자 값이 path_sv인 노드 집합 X와 경로 식별자 값이 path_su인 노드 집합 Y가 존재 한다. 여기서 x ∈ X가 y ∈ Y의 조상 노드 이면 두 노드 간의 레벨 차는 k이다.

(그림 5)은 (그림 2)에 보여진 XML 트리를 본 논문에서 제안하는 방식으로 저장한 예를 나타낸다.

우선 LABELS 테이블에서 각 레이블의 고유한 레이블 식별자 (즉, lid) 값을 파악할 수 있다. 예를 들어, author 레이블 식별자 값은 4임을 알 수 있다. 위에서 언급한 바와 같이 (그림 4)의 경로 인덱스는 영역 기반 방식을 적용하여 PATHINDEX 테이블을 이용하여 저장한다. 또한 경로 인덱스의 각 노드에 대한 주해는 단순 경로를 이용하나 부분 문자열 검색 속도를 향상하기 위하여 레이블 식별자의 연속으로 부호화하여 저장 한다. 예를 들어, 단순 경로 bibDB.book.author는 >/1>/2>/4로 표현되며 이에 대응되는 경로 인덱스 노드의 영역은 (7, 8)임을 알 수 있다. 또한, NODE 테이블은 (그림 2)의 XML 트리 상의 노드들의 정보를 나타낸다. 위에서 언급한 바와 같이, 각 XML 트리 상의 노드들의 경로 정보는 path_s를 이용하여 표시한다. 예를 들어, NODE 테이블의 네 번째 튜플 (5, 6, 7)의 path_s 값 7을 이용하여 PATHINDEX 테이블에서 해당 노드의 단순 경로가 >/1>/2>/4로 표현되어 있음을 알 수 있다.

(그림 3)의 경로 테이블 접근 방식에서는 각 노드의 레벨 정보가 Region 테이블에 저장된다. 경로 테이블 방식과는 달리, Property 1으로부터 우리는 PATHINDEX 테이블로부터 노드들 간의 레벨 차를 구할 수 있다. 더욱이, 경로 인덱스의 노드 수는 XML 트리 상의 노드 수에 비하여 상당히 적으므로 제안하는 접근 방식은 더 적은 디스크 공간을 소비하게 된다.

PATHINDEX

| path_s | path_e | path_l | path |
|---|---|---|---|
| 1 | 16 | 1 | >/1 |
| 2 | 9 | 2 | >/1>/2 |
| 3 | 4 | 3 | >/1>/2>/5 |
| 5 | 6 | 3 | >/1>/2>/3 |
| 7 | 8 | 3 | >/1>/2>/4 |
| | ... | | |
| 13 | 14 | 3 | >/1>/6>/4 |

LABELS

| label | lid | path_s |
|---|---|---|
| bibDB | 1 | 1 |
| book | 2 | 2 |
| title | 3 | 5 |
| title | 3 | 11 |
| author | 4 | 7 |
| author | 4 | 13 |
| @price | 5 | 3 |
| journal | 6 | 8 |

NODE

| s | e | path_s |
|---|---|---|
| 1 | 24 | 1 |
| 2 | 9 | 2 |
| 3 | 4 | 5 |
| 5 | 6 | 7 |
| | ... | |
| 21 | 24 | 13 |

(그림 5) 제안한 접근방식의 예

### 4.2 질의 처리 방법

(그림 5)에서 보이는 바와 같이, 제안한 방식은 경로 정보가 레이블 식별자의 연속으로 부호화되어 PATHINDEX 테이블의 path 애트리뷰트에 저장되어 있어서 SQL문 생성 방식은 경로 테이블 방식과 유사하다. 따라서, 이 절에서는 몇 개의 예제를 통하여 경로 테이블 방식의 SQL문과 본 논문에서 제안하는 방법에 기반하여 생성된 SQL 문의 차이점을 설명한다.

이 절에서는 설명하는 SQL 문은 4.1절에서 기술한 관계형 스키마를 기반으로 한다. XML 트리의 노드 간에는 다양한 구조적 관계(예, following 과 preceding)가 존재한다. 영역 기반 방법에서 following, preceding 관계를 계산하는 방법은 [9]에서 제시되고 있으며 본 저장 기법에서도 노드들은 영역 기반 저장 방법을 사용하여 관리하고 있으므로 following, preceding을 지원하기 위한 확장이 용이하다. 따라서, 이 절에서 간결성을 위하여 부모-자식, 조상-자손 관계만을 다룬다.

4.1절에서 언급하였듯이 본 논문에서 제안하는 방식에서는 레이블 경로 대신 레이블 식별자의 연속을 저장한다. 따라서, 주어진 경로 표현식에 나타나는 레이블을 레이블 식별자로 대응시키는 일이 필요하다. 일반적으로 XML 데이터 상의 레이블의 수는 적다. 따라서, 레이블과 경로 식별자(lid)의 대응을 위한 해쉬 테이블을 LABELS 테이블에서 추출하여 메인 메모리 상에서 효율적으로 관리할 수 있으며, 질의 변역기는 경로 표현식 상의 레이블을 직접적으로 레이블 식별자로 변환할 수 있다. 이 부담은 상당히 작으며, 경로 테이블 방식에서도 레이블 경로를 위한 문자열에 '>'와 '%'등을 삽입하기 위하여 유사한 부담이 존재한다.

예제 2.

Q2: /bibDB//title

SQL2:

```
SELECT N1.*
FROM NODE N1, PATHINDEX P1,
LABELS L1
WHERE L1.lid = 3
  AND L1.path_s = P1.path_s
  AND P1.path LIKE ">/1>%/3"
  AND N1.path_s = P1.path_s
```

경로 테이블 방식에서는 예제 1에서 보이는 바와 같이 주어진 XML 질의문에 대하여 경로 테이블의 모든 컬럼을 검색한다. 그러나, 본 논문에서 제안하는 방식은 예제 2에서 보이는 바와 같이 우선 LABELS 테이블에서 title(즉, 레이

블 식별자 값 3)로 끝나는 경로들의 경로 식별자(path_s) 값을 가지고 온다. 그리고 PATHINDEX에서 해당 경로 식별자를 가지는 튜플들 중에서 경로 표현식 조건(즉, ">/1>%/3")을 만족하는 경로들만을 남긴다. 따라서, 제안하는 방식은 경로 테이블 방식과는 달리, PATHINDEX 테이블의 일 부분만을 탐색하여 질의 성능을 향상 시킨다.

예제 3.
Q3: /bibDB/book/*//title
SQL3:

```
SELECT N2.*
FROM PATHINDEX P1, LABEL L1
     PATHINDEX P2, NODE N2;
WHERE L1.lid = 2 AND L1.path_s = P1.path_s
  AND P1.path = ">/1>/2'
  AND P1.ps < P2.ps AND P2.pe < P1.pe
  AND P2.path_l +1 < P2.path_l
  AND P2.path = ">/1>/2>%/3"
  AND P2.path_s = N2.path_s
```

예제 3은 경로 테이블 방식에서는 효율적으로 지원하지 못하는 wildcard가 포함된 XML 질의문과 대응되는 SQL 문이다. 경로 테이블 방식에서는 (그림 5)에서 보듯이 레벨 정보가 Region 테이블에 저장되어 있다. 따라서, 예제 3의 wildcard를 처리하기 위하여서는 각 Region 테이블에서 /bibDB/book로 접근 가능한 book 엘리먼트들을 구하고 /bibDB/book//title로 접근 가능한 title 엘리먼트들을 구하여 이들간의 레벨 차가 2 이상이어야 한다는 조건을 사용하여 조인 연산을 하여야 한다.

이에 반하여 본 논문에서 제안하는 방식은 경로 인덱스 상의 노드 간의 조상-자손 관계 및 각 노드의 레벨 값의 차이를 이용하여 효율적으로 wildcard에 대응되는 경로 정보를 PATHINDEX 테이블 만을 이용하여 찾아 낼 수 있다. 따라서, 각 레벨 정보가 PATHINDEX 상에 존재함으로 적은 수의 튜플 만을 검색하여 질의 상에서 찾고자 하는 질의 결과를 찾아 낼 수 있게 된다. 더욱이, PATHINDEX는 경로 식별자의 연속을 저장하고 있으므로 경로 테이블 방식보다 경로 정보의 길이가 짧다. 따라서 부분 문자열 매칭 시의 부담이 더 작다. 따라서, 본 논문에서 제안하는 방식은 wildcard를 보다 효율적으로 처리 할 수 있다.

예제 4.
Q4: /bibDB/book[.//author]//title
SQL4:

```
SELECT N3.*
FROM PATHINDEX P1, LABELS L1, NODE N1,
    PATHINDEX P2, NODE N2
    PATHINDEX P3, NODE N3
WHERE L1.lid = 2
  AND L1.path_s = P1.path_s
  AND P1.path LIKE ">/1>/2"
  AND P1.path_s < P2.path_s
  AND P2.path_e < P1.path_e
  AND P2.path LIKE ">/1>/2>%/4"
  AND P1.path_s < P3.path_s
  AND P3.path_e < P1.path_e
  AND P3.path LIKE ">/1>/2>%/3"
  AND P1.path_s = N1.path_s
  AND P2.path_s = N2.path_s
  AND P3.path_s = N3.path_s
  AND N1.s < N2.s AND N2.e < N1.e
  AND N1.s < N3.s AND N3.e < N1.e
```

예제 4의 Q4는 predicate를 포함하고 있는 XML 질의문으로 기호 []는 정제 조건(filtering condition)을 나타낸다. 이 예제에서는 3개의 경로 표현식 /bibDB/book, /bibDB/book//author, /bibDB/book//title이 참여하고 있으며 이는 각각 >/1>/2, >/1>/2>%/4, >/1>/2>%/3으로 변환된다.

이 경우 경로 테이블 방식에서는 적절한 경로 식별자 값들을 찾기 위하여 경로 테이블 전체를 3번 검색하여야만 한다. 이에 반하여 본 논문에서 제안하는 방식에서는 위에서 언급한 바와 같이 경로 인덱스 상의 구조적 관계를 이용(즉, P1.path_s < P2.path_s AND P2.path_e < P1.path_e 등과 같은 조건)하여 경로 테이블 전체가 아닌 경로 인덱스 상의 구조 조건을 만족하는 경로 정보들 중에서만 경로 식별자 값을 찾도록 한다. 그리고 세 가지의 경로 표현식에 대하여 각 경로 표현식을 만족하는 노드 정보들을 추출 (즉, P2.path_s = N2.path_s 등과 같은 조건들)하고 각 노드 상의 영역 정보를 이용하여 실제 각 노드 간에 질의 상에 표현된 구조 정보가 만족되는 노드들을 추출하게 된다. 따라서 전체 질의 성능을 향상시킨다.

## 5. 실 험

본 논문에서 제안하는 방식의 효율성을 보이기 위하여, 경로 테이블 방법과 제안하는 방법의 질의 성능을 비교하였다. 이 실험에서 관계형 데이터베이스로 IBM DB2 v8.2 를 이용하였다. 실험에 사용된 하드웨어는 Pentium IV-3.6GHz, 3.25Gbyte 메모리 이고, 운영체제는 Windows XP이다. 프로그래밍 언어는 Java를 이용하였으며 XML Parser로는 JDK 1.5의 JAXP를 이용하였다.

실험용 데이터 세트로서 실제 및 합성 XML 데이터로서 Shakespeare, XMark, 및 GedML을 이용하였다. 실제 XML 데이터로 Shakespeare는 섹스피어의 희곡 모음으로 37개의 희곡을 모아서 하나의 XML 문서로 만들었다. 합성 XML 데이터로 XML 벤치마크 프로젝트을 위하여 만들어진 XMark 데이터[15]를 이용하였다. XMark 데이터는 인터넷 경매 사이트의 내용을 흉내 내고 있다. GedML은 계보학 정

보를 표현하는 마크업 언어의 DTD를 이용하여 만든 합성 데이터이다. <표 1>은 실험에서 사용된 실험 데이터의 특징을 요약한 것이다..

〈표 1〉 XML 실험 데이터 세트의 특징

| Data Set | Size(MB) | Depth | Tags | Paths | Nodes |
|---|---|---|---|---|---|
| Shakespeare | 7.89 | 7 | 22 | 58 | 230426 |
| XMark | 11.88 | 12 | 83 | 536 | 328019 |
| GedML | 10.11 | 10 | 84 | 17540 | 500450 |

<표 2>는 제안한 접근 방법과 경로 테이블 방식을 이용한 데이터베이스의 크기를 나타낸다. <표 2>에서 보듯이 제안한 접근 방식은 경로 테이블 방식에 비하여 적은 디스크 공간을 사용한다. 이는 제안한 접근 방식에서는 각 노드의 레벨 정보가 PATHINDEX 테이블에 관리되고 경로 정보가 레이블의 연속이 아닌 경로식별자의 연속으로 부호화되어 저장하고 레벨 정보를 각 노드 정보와 같이 저장하는 것이 아니라 PATHINDEX 테이블에 경로 정보와 같이 저장하기 때문이다.

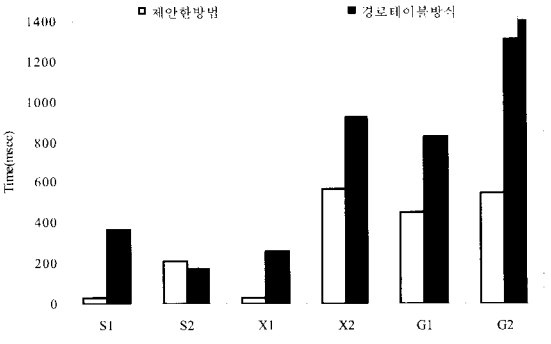
〈표 2〉 데이터베이스 크기(MByte)

| Data Set | 제안방법 | 경로 테이블 방식 |
|---|---|---|
| Shakespeare | 13.92 | 14.63 |
| XMark | 16.84 | 17.64 |
| GedML | 16.52 | 17.91 |

〈표 3〉 질의 리스트

| 질의 이름 | 질의 |
|---|---|
| S1 | //ACT/SCENE/*//STAGEDIR |
| S2 | //ACT[EPILOG//SPEECH]//SEPAKER |
| X1 | //people/*/*/watch |
| X2 | //open_auction[annotation//listitem]/current |
| G2 | //SUBMITTER/*/*//TIME |
| G2 | //END[//PAGE]/FAMC/ADDR |

<표 3>은 본 실험에서 사용된 질의들을 나타낸다. 질의 성능을 측정하기 위하여 각 실험 데이터 세트에 대하여 2개의 질의를 만들었다. 질의 이름의 첫 문자는 질의 수행 대상인 XML 문서를 나타내고 두 번째 숫자는 질의 타입을 나타낸다. 본 논문에서 제안하는 기법은 경로 표현식 간의 구조 정보를 유지하고 있으므로 기존의 기법에 비하여 다양한 종류(e.g., wildcard, predicate)의 경로 표현식을 효율적으로 지원한다. 따라서, 기존의 기법에 대하여 제안하는 기법의 성능을 파악하기 위하여 질의 타입 1과 질의 타입 2 형태의 질의들을 사용하였다. 질의 타입 1의 질의들은 wildcard를 포함한 경로 표현식들이다. 그리고 질의 타입 2의 질의는 predicate를 포함하고 있다.




(그림 6) 질의 성능

(그림 6)은 질의 수행에 걸린 시간을 msec 단위로 나타낸 것이다. 그림에서 보듯이 제안한 방법은 기존의 경로 테이블 방식에 비하여 보다 효율적임을 알 수 있다. 특히 테이블 2에서 보였듯이 제안된 접근 방법은 경로 테이블 방식에 비하여 보다 적은 디스크 공간만을 사용하고, 경로 정보를 레이블 식별자의 연속으로 부호화하여 저장하고 있으므로 경로 표현식 처리 시에 적은 디스크 I/O와 스트링 매칭 부담이 발생된다.

또한, 경로 테이블방식은 wildcard를 경로 테이블만을 이용하여 처리하지 못하고 Region 테이블에 있는 튜플들을 읽어 들여 처리하여야 한다. 이에 반하여 제안된 방법에서는 레벨 정보가 PATHINDEX 테이블에 존재함으로 효율적으로 wildcard를 처리할 수 있다. 따라서, 질의 타입 1의 성능은 모든 경우에 경로 테이블 방식에 비하여 제안된 방법이 보다 효율적이다.

타입 2 질의들을 수행하기 위하여서는 질의 처리기가 경로 정보들이 저장된 테이블(즉, PathTable 또는 PATHINDEX)을 복수 번 탐색해야 한다. 예제 4에서 보듯이 우리의 제안 방법은 경로 인덱스 상에 나타난 구조적 관계를 이용하여 질의에 참여하는 경로 표현식들을 처리 한다. 테이블 1에서 보듯이 Shakespeare 데이터는 아주 적은 수의 레이블 경로를 가지고 있다. 이 경우에는 (그림 6)의 S2 질의 성능 결과에서 보듯이, 단순히 경로 테이블을 여러 번 검색하는 경로 테이블 방식이 효율적이다. 그러나 X2 질의와 G2 질의 성능 결과에서 보듯이 XMark와 GedML과 같이 많은 수의 레이블 경로를 가지고 있는 XML 데이터에 대하여서는 제안된 방식이 더 효율적이다. 특히, X2 결과와 G2의 성능 차이에서 보듯이 구조적으로 복잡할수록 제안된 방법이 경로 테이블 방법에 비하여 보다 효율적임을 알 수 있다.

## 6. 결 론

다양한 XML 저장소 중에서 편리하게 사용할 수 있는 저장소가 관계형 데이터베이스이다. XML 데이터를 탐색하기 위하여 경로 표현식이 사용된다. 따라서, 관계형 데이터베이스 상에서 경로 표현식을 효율적으로 지원하기 위하여 많은

기법들이 제안되었다. 본 논문에서는 XML 문서를 관계형 데이터베이스를 활용하여 저장 하는 방법에 논하였다. 특히, 경로 인덱스를 관계형 데이터베이스의 수정 없이 효과적으로 저장 관리 할 수 있는 관계형 스키마을 제안하였다. XML 데이터는 일반적으로 트리 형태로 모델링 되며 트리 형태의 데이터에 대한 경로 인덱스도 트리 형태로 표현 된다. 따라서 XML 데이터를 저장 하는데 사용되었던 영역 방식을 적용하여 경로 인덱스를 저장한다.

더욱이 경로 인덱스의 경로 정보를 레이블 식별자의 연속으로 부호화 시키고 레벨 정보를 부가적으로 기록함으로써 기존의 경로 테이블 방식보다 디스크 공간을 적게 사용하면서도 보다 효율적으로 경로 표현식을 처리할 수 있다.

향후 연구에서는, 실험 결과에서 보듯이, XML 데이터의 구조적 복잡도에 따라서 선택적으로 효율적인 질의문을 생성하는 방안에 대하여 연구하고자 한다. 또한, 제안한 기법을 바탕으로 시간에 따라서 변화되는 XML 문서를 관계형 데이터베이스의 수정 없이 효율적으로 관리 하고 처리하는 방안에 대하여 다루고자 한다.

## 참 고 문 헌


[1] D. Beckett and B. McBride, "RDF/XML Syntax Specification (Revised)," W3C Working Draft, 2003.

[2] S. Boag, D. Chambering, M. F. Fernandez, D. Florescu, J. Robie, J. Simeon, "XQuery 1.0 : An XML Query Language," W3C Working Draft, 2005.

[3] T. Bray, J. Paoli, C. M. Sperberg-McQueen, "Extensible Markup Language(XML) 1.0," W3C Recommendation, 1998.

[4] J. Clark, S. DeRose, "XML Path Language(XPath) Version 1.0," W3C Recommendation, 2002.

[5] C. Chung, J. Min, K. Shim, "APEX : An Adaptive Path Index for XML Data," In Proceedings of ACM SIGMOD Conf., 2002.

[6] S. Cox, et al., "Geogrphy Markup Language(GML) Implementation Specification 2.1.1," OpenGIS Project Document Number 02-009, 2002.

[7] D. Florescu, D. Kossman, "Storing and Querying XML Data using an RDBMS," IEEE Data Engineering Bulletin, 22(3), 1999.

[8] R. Goldman, J. Widom, "DataGuides : Enable Query Formulation and Optimization in Semistructured Databases," In Proceedings of VLDB Conf., 1997.

[9] T. Grust, "Accelerating XPath Location Step," In Proceedings of ACM SIGMOD, Madison, June, 2002.

[10] Q. Li, B. Moon, "Indexing and Querying XML Data for Regular Path Expression," In Proceedings of VLDB Conf., 2001.

[11] J. M. Martinez, "Introduction to MPEG-7," ISO/IEC, 2000

[12] D. L. McGuinness, F. Harmelen, "OWL Web Ontology Language Overview," W3C Recommendation, 2004.

[13] T. Milo, D. Suciu, "Index Structures for Path Expression," In Proceeding of ICDE, 1999.

[14] P. Rao, B. Moon, "PRIX : Indexing And Querying XML Using Prufer Sequence," In Proceedings of ICDE, 2004

[15] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, J. Naughton, "Relational Databases for Querying XML Documents : Limitations and Opportunities," In Proceedings of VLDB Conf., 1999.

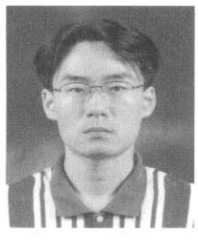
[16] A. Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu, R. Busse, "XMark : A Benchmark for XML Data Management," VLDB Conf., pp.974-985, 2002.

[17] J. Shanmugasundaram, E. J. Shekita, J. Kiernan, R. Krishnamurthy, S. Viglas, J. F. Naughton, I. Tatarinov "A General Techniques for Querying XML Documents using a Relational Database System," ACM Record, 30(3), pp.20-26, 2001.

[18] T. Shimura, M. Yoshikawa, S. Uemura, "Storing and Retrieval of XML Documents using Object-Relational Databases," In Proceedings of DEXA Conf., pp.206-217, 1999.

[19] H. Wang, S. Park, W. Fan, P. S. Yu, "ViST : A Dynamic Index Method for Querying XML Data by Tree Structures," In Proceedings of ACM SIGMOD., 2003.

[20] X. D. Wu, M. L. Lee, W. Hsu, "A Prime number labeling scheme for dynamic ordered XML trees," In Proceedings of ICDE, 2004.


**민 준 기**

e-mail : jkmin@kut.ac.kr

1995년 숭실대학교 전자계산학과(공학사)

1997년 한국과학기술원 전자전산학과 (공학석사)

2002년 한국과학기술원 전자전산학과 (공학박사)

2002년~2003년 한국과학기술원 연수연구원(Post Doc)

2003년~2004년 한국과학기술원 초빙교수

2004년~2005년 전자통신연구원 선임연구원

2005년~현 재 한국기술교육대학교 인터넷미디어공학부 전임강사

관심분야 : Query Processing, XML, Stream Data, Spatio-Temporal DB