Journal of Korea Multimedia Society Vol. 11, No. 4, April 2008(pp. 415-422)

관계형 DBMS 기반의 XML 문서 경로 통합 시스템

이범석[†], 황병연[††]

요 약

XML의 사용이 증가함에 따라 XML로 표현된 문서를 효율적으로 저장하고 검색하기 위한 XML 문서 관리 시스템에 대한 많은 연구들이 활발하게 진행되고 있다. 최근에는 주로 관계형 DBMS의 장점을 이용한 XML 문서의 저장과 검색에 대한 연구가 이루어지고 있다. XML Parser를 이용하여 문서 내용을 관계형 테이블에 매핑(Mapping)하면 안정적이고 효율적인 XML 문서 관리 시스템을 구축할 수 있다. 본 논문에서 제안하는 X-Binder 시스템은 관계형 DBMS기반의 역인덱스 기법을 사용한다. 역인덱스 기법은 빠른 검색 속도를 보장하지만, 많은 저장 공간을 낭비하는 단점을 가진다. 이 문제점을 해결하기 위해 XML 문서 저장 시 형제 관계를 가지는 경로들을 통합하여 저장한다. 제안하는 X-Binder 시스템은 XRel과의 성능 평가에서 저장 공간을 줄이고, 검색 시간을 단축하는 성과를 보였다.

Path Combining System of XML Documents based on Relational DBMS

Bum-Suk Lee[†], Byung-Yeon Hwang[††]

ABSTRACT

With the increasing use of XML, considerable research is being conducted on the XML document management systems for more efficient storage and searching of XML documents. Depending on the base systems, these researches can be classified into object-oriented DBMS (OODBMS) and relational DBMS (RDBMS). OODBMS-based systems are better suited to reflect the structure of XML-documents than RDBMS-based ones. However, using an XML parser to map the contents of documents to relational tables is a better way to construct a stable and effective XML document management system. The proposed X-Binder system uses an RDBMS-based inverted index; this guarantees high searching speed but wastes considerable storage space. To avoid this, the proposed system incorporates a path combining module agent that combines paths with sibling relations, and stores them in a single row. Performance evaluation revealed that the proposed system reduces storage wastage and search time.

Key words: XML Retrieval(XML 검색), Path Combining(경로통합), Inverted Index(역인덱스)

1. 서 론

XML(Extensible Markup Language)[1]의 기본 목적은 웹상에서 일반적인 정보를 단순한 방법으로 유지하고 전달하며 처리할 수 있게 하는 것이다. 이러한 XML의 사용이 빠르게 증가함에 따라서 이와 관련된 다양한 분야에 대한 연구의 필요성도 함께 증대되고 있다. 특히 최근에는 XML로 표현된 문서를 효율적으로 저장하고 검색하기 위한 XML 문서 관리 시스템에 대한 많은 연구들이 활발하게 진행되

※ 교신저자(Corresponding Author) : 황병연, 주소 : 경기도 부천시 원미구 역곡2동 산 43-1(420-743), 전화 : 02)2164-4363, FAX : 02)2164-4777, E-mail : byhwang@catholic.ac.kr
접수일 : 2007년 4월 3일, 완료일 : 2007년 8월 30일
[†] 준회원, 가톨릭대학교 컴퓨터공학과 박사과정 (E-mail : leez79@gmail.com)
[††] 종신회원, 가톨릭대학교 컴퓨터정보공학부 교수
※ 이 논문은 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2007-521-D00400)

고 있다[2]. 이러한 연구는 XML에 적합한 객체지향 DBMS(OODBMS)를 사용하는 것[3,4]과 기존의 관계형 DBMS(RDBMS)를 사용하는 것[5,6]으로 나눌 수 있다. 관계형 DBMS를 이용하는 방법이 OODBMS를 사용하는 것에 비해 XML 문서의 구조를 반영하기에 크게 충분하지는 않지만, XML 파서(parser)를 이용하여 문서의 구조와 내용을 테이블에 매핑(mapping)하면 안정적이고 효율적인 XML 문서 관리 시스템을 구축할 수 있다. 이에 따라 최근에는 관계형 DBMS를 이용한 XML 문서의 저장과 검색에 대한 연구가 점차 늘어나고 있다.

XRel[5]은 기존의 관계형 DBMS를 이용하여 XML 문서를 저장하고 검색하기 위해 개발된 시스템이다. 이 시스템은 XML 문서를 관계형 테이블에 매핑하기 위한 스키마를 제안하였고, 경로검색 방법을 제시하였다. 그러나 시스템의 빠른 검색 속도를 보장하기 위한 방법을 제시하지는 않았고, 실제 성능 평가 실험에서 부분 매치 질의 및 복합 질의는 느린 결과를 보여주었다. 또한 모든 경로에 대해 저장하기 때문에 경로 테이블의 크기가 지속적으로 증가하는 단점을 가지고 있다.

이러한 단점들을 해결하기 위해 본 논문에서는 관계형 DBMS를 기반으로 하는 X-Binder 시스템을 제안한다. 이 시스템은 경로 통합 모듈과 질의 변환 모듈을 가지고 있다. 경로 통합 모듈은 XML 문서가 저장될 때, 공통의 조상 경로를 가지는 형제 노드를 함께 저장함으로써 경로 테이블 크기가 급격히 증가하지 않도록 하였다. 또한 제안하는 시스템에는 사용자의 질의를 SQL로 변환하기 위한 질의 변환 모듈이 있다. 검색을 위해 사용자가 입력하는 질의의 형태는 XPath[7]를 이용해 정의한 선형 경로 표현식(linear path expression)이다. 이 형태의 질의는 전체 매치 질의(full match query)와 부분 매치 질의(partial match query), 그리고 복합 질의(complex query)로 나뉜다. 이 세 가지 경로 질의 형태에 대해 질의 변환 모듈에서 관계형 DBMS의 SQL로 변환한다.

그리고 사용자 질의에 대해 빠른 검색 속도를 보장하기 위해 정보검색에서 사용되는 역인덱스 기법[8]을 적용하였다. 역인덱스 기법은 구현이 용이하고, 속도가 빠르다는 장점을 가진다. 반면에 일반적인 역인덱스 기법의 단점은 기억 공간을 많이 차지한다는 것이다. 역인덱스를 구성하는 데이터베이스 테이블의 여러 항목에 대해 많은 정보를 유지할 경우 저장공간이 데이터 파일 크기의 300%까지 커지는 경우도 존재한다. X-Binder 시스템의 경로 통합 모듈은 역인덱스의 장점을 그대로 이용하면서, 단점을 보완해준다. X-Binder는 경로 통합 모듈을 이용함으로써, XRel과의 성능평가에서 경로 테이블에 저장된 행의 수가 줄어들었으며, 경로 검색 속도 비교에서도 우수한 성능을 보였다. 또한 질의 변환 모듈은 사용자로 하여금 XML 문서의 검색을 위해, XPath의 선형 경로 표현식을 사용할 수 있게 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서 XRel 시스템과 X-Binder 시스템에 사용된 XML 질의 모델에 대하여 설명한다. 3장에서는 본 논문에서 제안하는 X-Binder의 시스템 구조와 테이블, 그리고 경로 통합 모듈과 질의 변환 모듈에 대해 기술한다. 4장에서는 구현 및 실험을 통해 기존의 방법과 제안한 방법을 비교하는 성능평가를 수행하고 그 결과를 제시한다. 마지막으로 5장에서는 성능평가 결과에 대한 고찰과 향후 연구 계획에 대하여 제시한다.

2. 관련연구

XRel은 기존의 관계형 DBMS를 이용하여 XML 문서를 저장하고 검색하기 위해 개발된 시스템이다. XRel은 4개의 관계형 테이블에 XML의 트리 구조를 매핑한다. 각각의 테이블은 {Path | pathID, pathexp}, {Element | docID, pathID, start, end, index, reindex}, {Attribute | docID, pathID, start, end, value}, {Text | docID, pathID, start, end, value}의 구조를 가진다.

Path 테이블은 문서들 내의 모든 경로들과 그들의 고유한 id를 저장하는데, pathexp에 저장되는 경로는 '#/movie#/director#/fullname'의 형태를 가진다. 여기서 '#'이 사용된 이유는 XRel의 SQL 질의에서 문자열 매치를 위해 LIKE 연산자와 '%'를 사용하기 때문이다. '/movie/director'의 결과를 얻기 위한 질의에서, LIKE '#/movie#%/director'의 형태로 표현된 질의가 있을 때, '#'를 넣지 않는다면, '/movie%/director'로 질의가 수행 될 것이고, 이것은 '/movielist/director'나 '/movies/director'와 같은 잘못된 결과도 함께 불러오게 된다. 이러한 잘못된 결과를 검

색하지 않기 위해 경로를 저장할 때 '#'을 함께 저장한다. Element와 Attribute 테이블은 요소들에 대한 정보와 속성들에 관한 정보를 각각 저장한다. 여기서 Attribute는 테이블 내부에 value 값을 저장하지만, Element는 value 값을 Text 테이블에 따로 저장한다. 따라서 Element와 Text 테이블은 구현에 따라 하나의 테이블로 구성할 수도 있다. XRel은 기존의 안정적이고 보편적인 관계형 DBMS를 이용하여 쉽게 XML 문서의 저장 및 검색을 가능하게 한다.

XPath[7], XQuery[9], Quilt[10] 등과 같은 주요한 XML 질의 언어들은 어떤 XML 트리에 대한 질의를 명시하기 위하여 경로 표현식을 사용한다. XML 연구에 많이 사용되는 것은 선형 경로 표현식(linear path expression, LPE)과 분기 경로 표현식(branching path expression, BPE)인데, 본 논문에서는 선형 경로 표현식(LPE)을 사용한다. 경로 질의 함수 *PathFinder*(*p*)는 선형 경로 표현식으로 나타내어진 경로 p를 입력받아 해당 경로를 가진 문서와 경로에 해당하는 내용을 출력한다. 연산의 정의는 다음과 같다.

정의 1. XML의 경로 질의는 경로(p)를 입력하여 해당 경로를 가진 문서의 이름과 해당 경로의 내용 정보를 사용자에게 보여준다.

PathFinder(*p*) = {(*document, text*) | *document*는 *p*를 포함한다. 그리고 *text*는 *p*에 존재한다.}

Q1과 Q2는 *PathFinder*(*p*)가 입력받는 기본적인 두 가지 형태의 선형 경로 표현식이고, Q3은 복합질의이다. 두 질의 중 Q1은 경로 상에 명시된 부모-자식 관계를 의미하는 '/'를 가진다. 이러한 형태의 질의를 전체 매치 질의라고 한다. 반면, Q2는 경로 상에 명시된 '/' 이외에, 조상-자손 관계를 의미하는 '//'를 가지며, 부분 매치 질의라고 한다. 이 두 가지 형태의 질의를 통칭하여 선형 경로 표현식이라 한다. Q3은 선형 경로 질의와 내용 질의가 혼합된 형태로 복합질의의 예를 보여준다.

Q1 ::: /movie/director/fullname

Q2 ::: /movie//player/role

Q3 ::: /movie[/@year='2003']//player/name

정의 2. 전체 매치 질의는 $/l_0/l_1/l_2\cdots/l_n$의 형태로 정의된다. 이때, l_0은 문서의 루트이고, $l_i(i=1,2,\cdots,n)$은 그 경로 내에 존재하는 i번째 레이블이다. l_{i-1}과 l_i사이의 관계는 '/'로 표현되고, 부모-자식 관계로만 이루어진 경로 표현 질의임을 의미한다. 또한 l_n은 말단 노드(end node)를 의미한다.

정의 3. 부분 매치 질의는 $/l_0/l_1//l_2\cdots/l_n$으로 표현된다. 경로 내에 존재하는 i번째 레이블이 있을 때, l_{i-1}과 l_i사이의 관계에 '//'가 필수적으로, '/'는 선택적으로 포함되어 사용된다. 이 질의는 조상-자손 관계가 포함된 경로 표현 질의이다.

정의 4. 복합 질의는 $/l_0[/l_1='c1']/l_2\cdots/l_n$으로 표현된다. 이 질의는 l_0/l_1의 내용이 *c1*을 포함하고, $l_0/l_2\cdots/l_n$의 경로를 가지는 문서를 선택하는 것이다. 복합 질의는 전체 매치 질의를 의미하는 '/'와 부분 매치 질의를 의미하는 '//'를 모두 사용할 수 있다.

3. X-Binder: XML 문서 경로 통합 시스템

3.1 시스템 구조

X-Binder의 시스템 구조는 XML 문서의 효율적인 저장을 위한 경로 통합 모듈(path combining module), 선형 경로 질의로 입력받는 사용자 질의를 관계형 DBMS 검색을 위한 SQL로 변환시켜주는 질의 변환 모듈(query translation module), 그리고 데이터베이스에 저장된 정보에 대해 빠른 검색 속도를 위한 역인덱스 관리 모듈(inverted index management module)로 나누어진다. 역인덱스 관리 모듈은 저장된 역인덱스 테이블에 대해 항상 정렬된 상태를 유지하도록 관리하는 기능을 한다.

그림 1은 X-Binder의 시스템 구조를 보여준다. 사용자 인터페이스(UI)를 통해 XML 문서가 입력되면, 경로 통합 모듈과 역인덱스 관리 모듈을 통해 데이터베이스로 저장된다. 저장된 XML 문서에 대해 선형 경로 질의가 입력되면, 질의 변환 모듈을 통해 SQL 형태로 변형된다. 변형 과정을 거친 다음 역인덱스를 이용하여 데이터베이스에 저장된 문서 정보를 검색

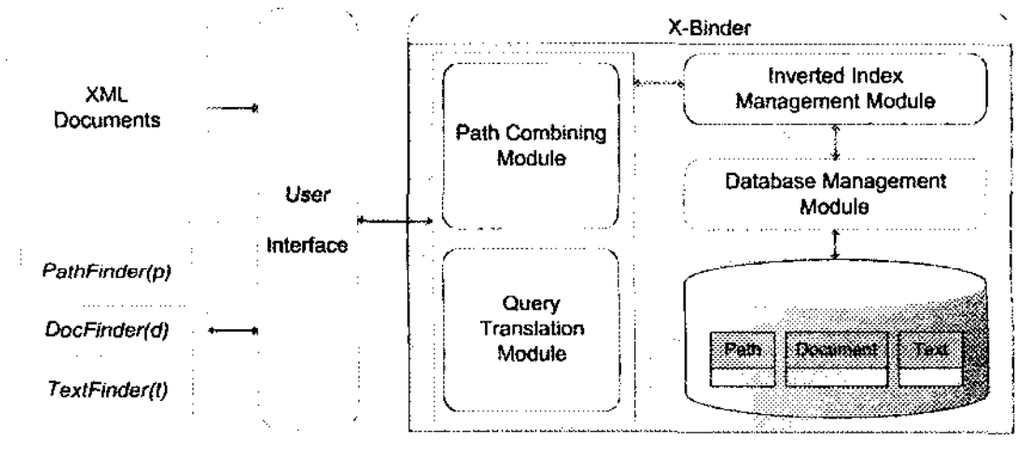

그림 1. X-Binder의 시스템 구조

하고 UI를 통해 사용자에게 검색 결과를 보여준다.

X-Binder에서 경로 통합 기법과 역인덱스 검색 기법을 사용하기 위하여 그림 2와 같이 Document, Path, Text로 이루어진 세 개의 관계형 테이블 스키마(relational table schema)를 설계하였다. 그림 2는 예제 XML 문서를 설계한 데이터베이스 테이블에 저장한 것을 보여준다. 데이터베이스의 docID, pathID, textID는 각각 문서 식별자와 경로의 식별자, 그리고 내용 정보에 대한 식별자를 의미한다.

XRel과 확연히 다른 것은 바로 경로 테이블에 저장되는 경로들이 통합된 경로라는 점이다. XRel은 레이블 경로 매치를 위한 질의 수행 시 LIKE 연산을 이용한다. 이 방법은 Path 테이블의 모든 정보를 문자열 매치로 검색하기 때문에 검색해야 할 열의 수가 증가하면 시간도 따라서 증가하는 단점을 가진다. 본 논문에서 제안한 X-Binder의 통합 경로 테이블은 사용자 질의에 대해 검색해야하는 행의 수를 줄였기 때문에, 저장공간과 검색시간에서 효율을 가진다. 또한 Path 테이블에서는 XRel에서 사용한 것과 같이 '/' 대신 '#/'를 사용하였다. 세 테이블은 서로 다른 테이블에 대해 역인덱스의 역할을 수행한다.

```
<movie title="Old boy" year="2003" country="Korea">
    <director>
        <fullname>Chan-wook Park</fullname>
        <nationality>Korean</nationality>
    </director>
    <cast>
        <players>
            <player>
                <role>Dae-su Oh</role>
                <name>Min-sik Choi</name>
            </player>
            <player>
                <role>Mi-do</role>
                <name>Hye-jeong Kang</name>
            </player>
        </players>
    </cast>
    <genre>Action, Mystery, Thriller</genre>
    <comments>
        <comment>Old boy is definitely my favorite movie ... </comment>
        <comment>I am a big fan of Asian cinema and ... </comment>
    </comments>
</movie>
```

Document	
docID	docName
1	\dataset\imdb\01.xml
2	...
3	...
4	\dataset\reuter\27.xml

Path	
pathID	CombinedPathexp
1	#/movie#/director#(/fullname#/nationality#)
2	#/movie#/cast#/players#/player#(/role#/name#)
3	#/movie#(/genre#/@title#/@year#/@country#)
4	#/movie#/comments#(/comment#)

Text				
textID	docID	pathID	endPath	value
1	1	1	fullname	Chan-wook Park
2	1	1	nationality	Korean
3	1	2	role	Dae-su Oh
4	1	2	name	Min-sik Choi
5	1	2	role	Mi-do
6	1	2	name	Hye-jeong Kang
7	1	3	genre	Action, Mystery, Thriller
8	1	3	@title	Old boy
9	1	3	@year	2003
10	1	3	@country	Korea
11	1	1	comment	Old boy is definitely my favorite movie ...
12	1	2	comment	I am a big fan of Asian cinema and ...

그림 2. XML 문서와 X-Binder 테이블 저장 예

3.2 경로 통합 모듈

XML 문서의 유사한 경로를 통합하여 작고 빠른 인덱스를 구성하는 방법은 1-Index[11], A(k)-Index [4], 그리고 DataGuide[12]와 같은 그래프 기반 인덱스 기법에서 연구된 것들이 있다. 그래프 기반의 기법들은 말단 노드뿐만 아니라 경로 중간의 노드까지도 통합하므로, 관계형 DBMS로 관리하는 본 연구에 적용하는 것은 적합하지 않다. 이 절에서는 X-Binder의 경로 통합 기법을 소개한다. 제안하는 방법은 루트에서 시작하여 value가 존재하는 말단 노드까지의 전체 경로(full path)를 추출하고, 그 경로에서 말단 노드를 제외한 모든 조상 노드의 경로(조상 경로)가 같은 노드를 유사한 경로(similar path)라고 보아 이들을 통합하고, 관계형 데이터베이스의 테이블에 저장할 때, 한 개의 행에 함께 저장한다. 이때 부모가 같은 요소와 속성이 있으면, 속성이름 앞에 요소와 구분하기 위해 속성 구분자 '@'을 붙인 다음, 모두 형제 노드(sibling node)로 인식한다.

정의 5. XML 문서의 경로를 통합하는 것은 서로 형제 노드(sibling node)로 인식되는 형제 경로들(sibling paths : $p_1, p_2, p_3, \cdots, p_n$)을 한 개로 묶어서 저장하는 것이다.

PathCombine(*sibling_paths*) = {(*CombinedPathexp*) | *sibling_paths*는 복수의 형제 경로 입력이다. 그리고 *CombinedPathexp*는 입력된 경로들이 통합된 한 개의 경로 표현식으로 저장된 것을 의미한다.}

예를 들어, 그림 2의 예제 XML 문서에서 *PathCombine*(*sibling_paths*)를 수행하면 '/movie/director/fullname'과 '/movie/director/nationality'는 서로 같은 조상 경로를 가지며, 말단 노드만 다르기 때문에 서로 형제 노드이다. 그리고 '/movie/genre'와 속성 노드인 '/movie/@title', '/movie/@year', '/movie/@country'는 네 개의 경로가 모두 형제 노드이다. 다시 말하면, 서로 유사한 경로이다. 또한 '/movie/cast/player/role'과 '/movie/cast/player/name'도

형제 노드이다. 마지막으로 '/movie/comments/comment' 경로가 저장된다.

이 방법으로 형제 노드들을 데이터베이스의 경로 정보를 저장하는 테이블에 한 개의 행으로 통합하여 저장하면, 그림 2의 Path 테이블과 같이 4개의 경로 표현식이 생성된다. 이러한 결과는, XRel 시스템에서 같은 XML 문서를 저장할 때 14개의 경로 표현식이 추출되는 것과 비교될 수 있다. 그림 2에서 '|' 기호는 서로 통합된 말단 노드들을 구분하기 위한 것이다. 이 기법을 알고리즘으로 나타내면 그림 3과 같다. 알고리즘 설명에서 '/#' 기호처리에 대한 내용은 생략한다. 이 알고리즘에는 우선 fullPath, endPath, queryFullPath, queryPrePath의 네 가지 입력이 필요하다. 예를 들어, 저장하려는 경로가 '/aa/bb/cc'라면, 각각의 변수는 '/aa/bb/(cc)', 'cc', '/aa/bb/(%cc%)', '/aa/bb/(%'와 같다. 앞의 두 변수는 저장을 위한 입력 값이고, 뒤의 두 변수는 데이터베이스 검색 질의를 위한 입력 값이다.

그림 3의 알고리즘에서 다음과 같은 세 가지의 경우에 대해 처리가 가능한데, ① 경로 테이블에 queryFullPath와 같은 것이 존재하는 경우, ② queryFullPath가 존재하지 않지만, queryPrePath가 존재하는 경우, ③ queryFullPath와 queryPrePath가 모두 존재하지 않는 경우이다. ①의 경우에는 입력하려는 경로가 이미 존재하기 때문에, 역 인덱스 테이블을 구성하기 위해 해당 경로가 저장된 곳의 pathID를 반환한다. ②는 '/aa/bb/(dd|ee|ff)'와 같은 형태로 저장된 유사한 경로가 존재하는 경우이므로, '/aa/bb/(dd|ee|ff|cc)'의 형태로 경로 테이블을 수정하고 pathID를 반환한다. 마지막으로 ③의 경우에는 입력하려는 경로가 경로 테이블에 존재하지 않는 새로운

Path Combining Algorithm

```
Input : fullPath, endPath, queryFullPath, queryPrePath
Output : pathID

Method :
 1   resultQFP = getStoredPath(queryFullPath); //동일한 경로 검색
 2   if(resultQFP != null){
 3       return pathID;
 4   }else{
 5       resultQPP = getStoredPath(queryPrePath); // 유사한 경로 검색
 6       if(resultQPP != null){
 7          CPathexp = combinedPath(resultQPP,endPath);
 8          updatePath(CPathexp); // 새로운 경로 유사한 경로에 추가하여 업데이
 9          return pathID;
10       }else{
11          insertPath(fullPath); // 새로운 경로를 경로 테이블에 추가
12          return pathID;
13       }
14   }
```

그림 3. 경로 통합 알고리즘

경로이므로 '/aa/bb/(cc)'와 같은 형태로 저장한다. 여기서 통합된 경로가 없어도 마지막 경로에 괄호를 해 주는 이유는 마지막 경로를 나타내기 위해서이고, 그것은 '/aa/bb/cc/gg/(hh|ii)'와 같은 경로가 '/aa/bb/cc'나 그와 유사한 경로를 찾으려는 질의에 검색되는 것을 막기 위해서이다.

3.3 질의 변환 모듈

PathFinder(*p*)를 수행하기 위해 선형 경로 표현식을 이용하여 직접 관계형 데이터베이스 테이블을 검색할 수는 없다. 따라서 관계형 DBMS에 저장된 XML 문서의 정보를 검색하기 위해서는 앞의 2장에서 정의한 선형 경로 표현식 질의에 대해 표준 질의 언어(SQL)로 변환해주는 과정이 필요하다.

전체 매치 질의는 루트로부터 시작하여 말단 노드까지 연결되는 선형 경로 질의의 한 종류를 말한다. '/movie/cast/players/player/role'와 같은 질의 Q4가 입력된다고 가정한다. 질의가 입력되면 SQL 형태로 변형시키는데, 이 경우 Text 테이블의 구조를 이용하기 위하여 말단 노드 'role'과 그 조상 경로로 나누어 다음의 SQL 구문을 생성한다. SQL의 3번째 줄에서 Text 테이블의 endPath에 대해 'role'로 한정지어 주므로, role과 함께 통합되어 저장된 'name'에 관련된 정보는 검색 결과에서 제외된다.

```
SELECT      d1.docName, t1.value
FROM        Document d1, Path p1, Text t1
WHERE       t1.endPath='role'
AND         p1.CombinedPathexp LIKE '#/movie#/ cast#/players#/player#/(%'
AND         p1.pathID=t1.pathID
AND         t1.docID=d1.docID
```

부분 매치 질의는 조상-자손 관계를 의미하는 '//'를 한 개 이상 포함하고 있고, 그 시작도 루트가 아닐 수 있는 질의를 말한다. 이 형태의 질의를 SQL로 바꾸려면, JOIN연산과 LIKE에 몇 가지의 조건을 포함시키면 된다. 예를 들어, '/movie//cast'와 같은 질의 Q5가 있다고 가정한다. 주어진 질의에서 '//'의 의미는 movie를 루트로 가지는 XML 문서로부터, 'cast'를 그 하위 노드로 가지는 모든 경로를 선택한다는 것이다. 이것은 문자열 매치 검색 방법으로 쉽게 찾아낼 수 있다. 위 질의를 SQL 형식으로 변환해 주기 위해서 우선 X-Binder의 데이터베이스에 저장된 경로 형태에 맞게 '/'를 '#/'로, '//'를 '#%/'로 바꾸어야 한다. 이 질의는 다음과 같은 SQL 질의로 변환될 수 있다. 이 SQL 질의를 살펴보면, WHERE 절에 'movi

e'와 'cast'의 위치에 대해 조건을 제한하였다. SQL 5번째 줄의 조건은 'cast'가 말단 경로(end path)가 아닌 경우이고, 6번째 줄의 조건은 'cast'가 말단 노드인 경우의 검색 조건을 제시한다. 또한, 말단 노드일 경우에는 Text 테이블의 endPath가 cast인지 다시 확인한다. 만약 'movie'가 루트가 아닌 '//movie//cast' 형태의 질의가 입력되면, 5, 6번째 줄에 해당하는 LIKE 조건의 맨 앞에 '%'를 넣어 '%#/movie#%/ cast#%(%', '%#/movie#%(%/cast#%'의 형태로 변형시켜주면 된다.

```
SELECT      d1.docName, t1.value
FROM        Document d1, Path p1, Text t1
WHERE       p1.pathID=t1.pathID
AND         t1.docID=d1.docID
AND         (p1.CombinedPathexp LIKE '#/movie#%/ cast#%(%'
OR          (p1.CombinedPathexp LIKE '#/movie#% (%/cast#%'
AND         t1.endPath='cast'))
```

복합 질의는 선형 경로 질의와 내용 질의가 혼합된 형태의 질의를 의미한다. '/movie[/@year='2003']//player/name'과 같은 질의 Q6는 'movie'를 루트로 가지는 문서들 중, 'movie'의 하위 노드인 'year' 속성의 값이 2003이고, '/movie//player/name'의 경로를 가지는 문서를 선택하는 것이다. 이 질의는 2개의 부분으로 나누어 생각할 수 있는데, 첫 번째는 '/movie/@year='2003''이고, 두 번째는 '/movie//player/name'을 검색하는 것이다. 이것을 고려하여 X-Binder의 질의 생성 모듈은 다음과 같이 질의를 변환한다. 이 질의의 2번째 줄에서는 질의의 첫 번째 부분에 대한 테이블 t1을 생성하고, 4번째 줄에서는 두 번째 부분에 대한 테이블 p1을 생성한다.

```
SELECT      d1.docName
FROM        (SELECT * FROM Text
             WHERE endPath='@year' and value='2003'} t1,
            (SELECT * FROM Path
             WHERE CombinedPathexp LIKE '#/movie#%/ player#(%/name#%') p1,
             Document d1
WHERE       (t1.docID = d1.docID)
```

4. 성능평가

이 장에서는 본 연구에서 제안한 X-Binder 시스템에 대해 몇 가지 실험에 대한 성능 평가의 결과에 대해 소개한다. X-Binder는 MS-Windows XP Professional 운영체제가 설치된 환경에서 JAVA 1.4.2.09 버전으로 구현되었으며, 데이터베이스 관리 시스템으로 MS SQL Server 2000이 사용되었다. 또한 XML 문서를 파싱하기 위해 J-DOM 1.0 버전[13]을 사용하였다.

본 논문에서는 두 가지 factor에 대해 성능 평가를 수행하였다. 첫째는 관계형 DBMS 기반의 대표적인 XRel 시스템과 제안된 X-Binder의 저장공간 비교이다. 테이블 구조가 다르기 때문에, Path 테이블의 저장 구조를 같도록 하여 이에 대해서 실험을 수행하였다. 다음으로는 *PathFinder*(*p*)의 검색 속도에 대해 몇 개의 샘플 질의를 생성하여 XRel과 비교 실험을 수행하였다.

4.1 저장공간 비교

저장공간을 비교하기 위한 실험에 사용한 XML 문서 데이터는 인터넷 영화 데이터베이스(IMDB)[14]를 XML 파일 형태로 저장한 것과 로이터(REUTER) 뉴스 기사[15]를 XML 형태로 저장한 것, 그리고 문서 수집 프로그램인 ReGet[16]을 이용해 인터넷 상에서 무작위적으로 얻어낸 Random Dataset의 세 가지를 사용하였다. 각 데이터의 최대 깊이는 6, 9, 14였고, 문서의 개수는 100개, 250개, 3000개 이다. 우선 이 실험은 본 논문에서 제안한 X-Binder 시스템의 데이터 테이블에 저장된 경로의 수와 XRel에 저장된 경로의 수를 비교해 보았다.

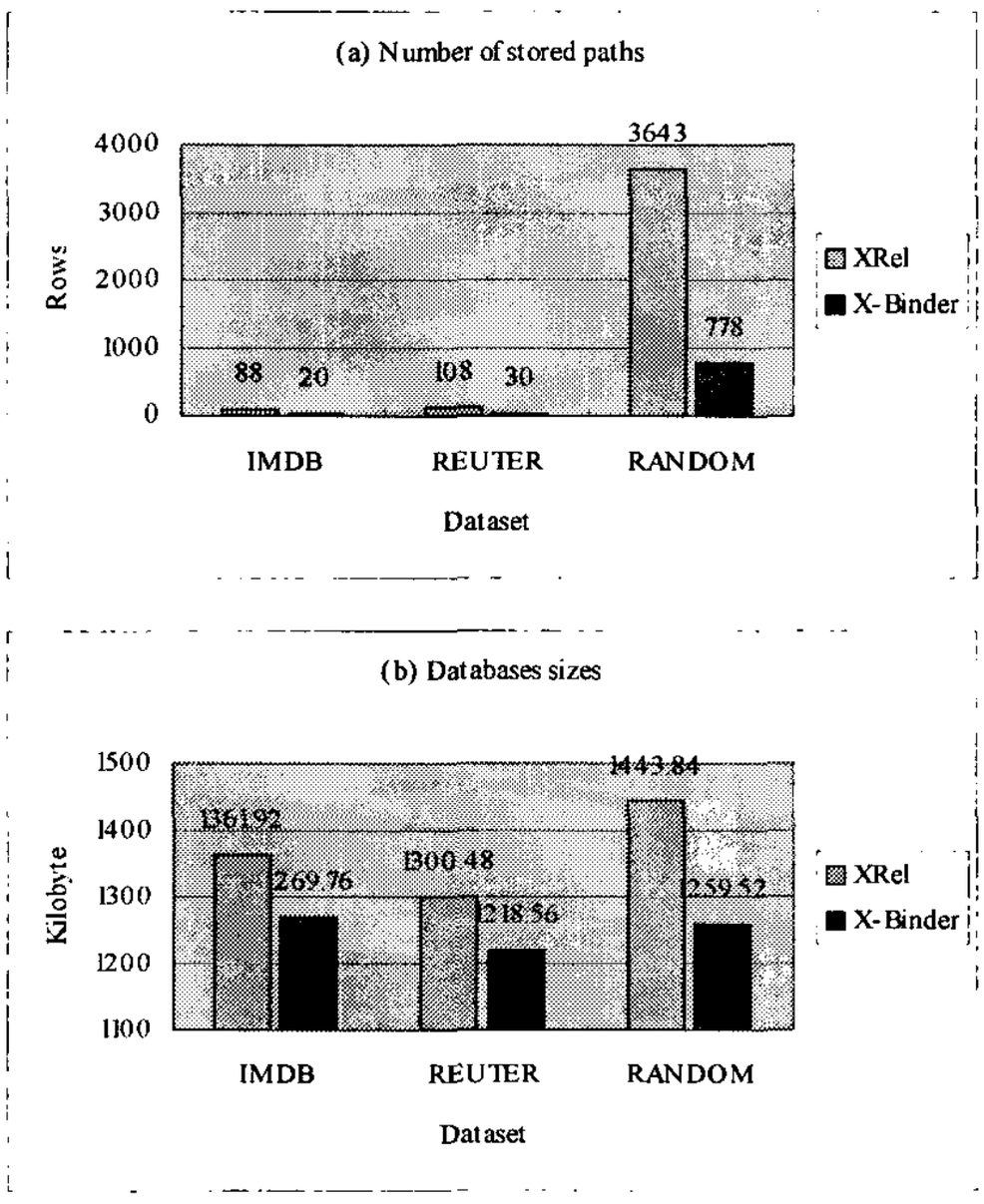

그림 4. 저장된 경로의 수와 저장공간 비교

그림 4(a)에서 실험 결과를 확인해보면, X-Binder에 저장된 경로의 수가 XRel보다 크게 줄어든 것을 알 수 있다. XRel은 XML 문서에 나타나는 모든 경로를 저장하지만, X-Binder는 같은 조상 경로를 가지는 것들끼리 통합하여 저장하기 때문에 이런 결과를 얻을 수 있다.

다음으로는 데이터베이스의 크기를 비교하였다. 그림 4(b)에서 나타낸 것처럼 데이터베이스의 크기도 줄어든 것을 확인할 수 있었다. 그래프에서 IMDB의 경로 수가 REUTER의 경로 수 보다 작은 데 비해 데이터베이스의 크기가 더 크게 측정된 이유는 요소나 속성의 이름이 REUTER보다 긴 단어의 형태이기 때문이다.

4.2 *PathFinder*(*p*) 검색 속도 비교

PathFinder(*p*)의 검색 속도를 비교하기 위해 전체 매치 질의, 부분 매치 질의, 그리고 복합 질의를 각각 2개씩 생성하였다. 각 질의는 다음과 같은 특징을 가지게 구성하였다. TQ1과 TQ2는 전체 매치 질의이고, 그 길이를 다르게 하였다. TQ3과 TQ4는 부분 매치 질의이고, 각각 '//'의 개수를 한 개와 두 개로 설정하였다. 마지막으로 TQ5와 TQ6은 전체 매치 질의와 부분 매치 질의에 내용 질의를 포함시킨 복합 질의이다.

TQ1 ::: /seller/name

TQ2 ::: /movie/cast/player/filmographies/ filmography/work

TQ3 ::: //topic/topicname

TQ4 ::: //news//topicname

TQ5 ::: members/member/id/name[first='Lee']

TQ6 ::: members/member//[first='Seo']

PathFinder(*p*)의 성능평가는 그림 5에 나타내었다. *PathFinder*(*p*)에 대한 성능평가 결과 제안한 시스템이 기존의 XRel보다 비교적 뛰어난 성능을 가지는 것을 알게 되었다. 특히 복합 질의를 수행하였을 때 검색 속도가 약 1/3에서 1/4로 줄어드는 것을 알 수 있다. 이는 제안한 시스템에서 경로를 저장한 테이블 행의 수가 줄었고, 이에 따라 SQL의 JOIN 연산을 수행할 때 생기는 테이블 행의 수가 급격히 줄어들기 때문이다.

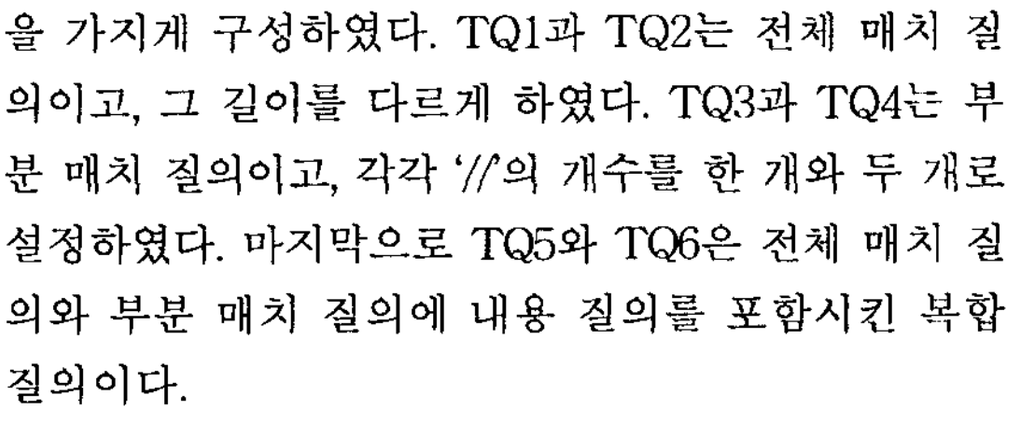

그림 5. *PathFinder*(*p*)의 검색 속도 비교

5. 결 론

관계형 DBMS를 기반으로 하는 다양한 XML 문서 저장 및 검색 시스템에 대한 연구가 증대되고 있다. 이러한 연구들 중 역인덱스를 이용한 시스템들이 대체적으로 빠른 검색 속도를 보장하며, 성능 평가에서 우수한 결과를 보이고 있다. 그러나 역인덱스는 데이터 테이블의 특정한 Field 값을 중심으로 인덱스를 새롭게 구성하기 때문에, 저장공간이 낭비되는 문제점을 가지고 있다.

이러한 역인덱스의 문제점을 해결하기 위해 경로통합 기법을 적용한 관계형 DBMS 기반의 XML 문서 검색 시스템 X-Binder를 제안하였다. 이 시스템의 경로통합모듈은 XML 문서를 저장할 때, 먼저 공통의 조상 경로를 가지는 형제 노드들을 한 개로 통합하여 저장한다. 이 방법으로 경로 테이블에 저장되는 행의 수를 크게 줄일 수 있고, 따라서 저장공간과 검색속도의 효율성을 가진다. 본 논문의 기여도는 다음과 같이 정리할 수 있다. 1) 그래프 인덱스나 비트맵 인덱스에서 시도되었던 XML의 경로통합을 관계형 DBMS에 적용이 가능하도록 경로통합 알고리즘을 제안하였다. 2) 경로통합 모듈을 가진 시스템을 구현하고 기존 시스템과의 성능평가를 수행하였다.

제안한 시스템의 성능 평가를 위해 관계형 DBMS 기반의 대표적인 XML문서 관리 시스템인 XRel과 저장공간 및 *PathFinder*(*p*)의 검색 속도에 대해 제안한 시스템과 비교하였다. 데이터베이스 검색 속도에 영향을 미치는 테이블에 저장된 데이터 행의 수 비교에서는 저장하는 문서의 종류에 따라 다르지만, XRel보다 약 1/4에서 1/5정도의 크기로 줄어드는 결과를 보여주었다. 데이터베이스의 크기를 비교한 실험에서는 많은 차이는 아니지만 일정한 양 만큼 줄어들게 되었다. *PathFinder*(*p*)의 검색 속도 비교에서도 XRel보다 항상 더 나은 성능을 보장하고, 특히 복합 질의에 대해서 뛰어난 성능을 보였다.

향후 연구로 기존에 연구되어진 역인덱스의 포스팅 리스트(posting list)를 이용하거나 본 논문에 제시된 질의변환모듈을 개선한다면 더 나은 성능을 기대할 수 있을 것이다.

참 고 문 헌

[1] W3C, "Extensible Markup Language(XML) Version 1.0 (Second Edition)," http://www.w3c.org/TR/REC-xml, Oct. 2000.

[2] S. Ceri, P. Fraternali, and S. Paraboschi, "XML: Current Developments and Future Challenges for the Database Community," In Proc. of the 7th Int'l Conf. on EDBT, pp. 3-17, Mar. 2000.

[3] C. Chung, J. Min, and K. Shim, "APEX: An Adaptive Path Index for XML Data," In Proc. of the Int'l Conf. on ACM SIGMOD, pp. 121-132, Jun. 2002.

[4] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes, "Exploiting Local Similarity for Indexing Paths in Graph-Structured Data," In Proc. of the 18th IEEE Int'l Conf. on Data Engineering, pp. 129-140, Feb. 2002.

[5] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura, "XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases," *ACM Transactions on Internet Technology*, Vol.1, No.1, pp. 110-141, Aug. 2001.

[6] H. Jiang, H. Lu, W. Wang, and J. X. Yu, "XParent: An Efficient RDBMS-Based XML Database System," In Proc. of the 18th Int'l Conf. on Data Engineering, pp.335-336, Feb. 2002.

[7] J. Clark and S. DeRose, "XML Path Language (XPath) Version 1.0," W3C Recommendation, Nov. 1999.

[8] D. Florescu, D. Kossmann, and I. Manolescu, "Integrating Keyword Search into XML Query Processing," In Proc. of the 9th Int'l WWW Conf. on Computer Networks, pp. 119-135, May. 2000.

[9] W3C, "XQuery 1.0: An XML Query Languag e," http://www.w3.org/TR/2005/WD-xquery -20050915/, Sep. 2005.

[10] D. Chamberlin, J. Robie, and D. Florescu, "Quilt: An XML Query Language for Heterogeneous Data Sources," In Proc. of the 3rd Int'l Workshop on ACM WebDB, pp. 53-62. 2000.

[11] T. Milo and D. Suciu, "Index Structure for Path Expressions," In Proc. of the 7th Int'l Conf. on Database Theory, Lecture Notes in Computer Science, Vol.1540, pp. 277-295, Jan. 1999.

[12] R. Goldman and J. Widom. "Approximate DataGuides," In Proc. of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats, pp.436-445, Jan. 1999.

[13] http://www.jdom.org

[14] http://us.imdb.com/top_250_films

[15] http://about.reuters.com/newsml

[16] http://deluxe.reget.com/en/

이 범 석

2004년 가톨릭대학교 분자생물학과(이학사), 국사학과(문학사)
2006년 가톨릭대학교 컴퓨터공학과(공학석사)
2006년~현재 가톨릭대학교 컴퓨터공학과 박사과정 재학
관심분야 : XML, 데이터베이스, 정보검색, 웹2.0

황 병 연

1986년 서울대학교 컴퓨터공학과(공학사)
1989년 한국과학기술원 전산학과(공학석사)
1994년 한국과학기술원 전산학과(공학박사)
1994년~현재 가톨릭대학교 컴퓨터정보공학부 교수
1999년 University of Minnesota Visiting Scholar
2007년 California State University, Sacramento Visiting Scholar
관심분야 : XML 데이터베이스, 데이터마이닝, 정보검색, 지리정보시스템