XQL질의 처리시스템을 위한 인덱스 알고리즘 설계 및 구현

장복선[0]*, 손기락*
한국외국어대학교 컴퓨터 및 정보통신공학과
boksun77@orgio.net , ksohn@hufs.ac.kr

Design and Implementation of Index Algorithm for XQL Query Processing System

Boksun Jang[0], Kirack Sohn
*Dept. of Computer & Information Telecommunication Engineering,
Hankuk University of Foreign Studies

요 약

효율적인 문서 교환을 위해 의미 있는 태그를 사용하는 XML문서가 인터넷상에서 널리 사용되고 있다. 이 XML문서를 저장하고 검색하기 위해 여러 분야에서 시스템이 개발되었지만 특별히 각광을 받는 시스템은 없었다. 본 연구에서는 관계형 데이터베이스에 XML문서를 트리 형태로 저장하고, 저장된 데이터베이스의 정보를 검색하기 위해 XQL질의처리 시스템을 개발하였다. 또한, 본 논문에서는 XQL질의 처리에 있어 사용자가 보다 체계적이고 편리하게 정보를 검색할 수 있도록 하기 위한 인덱스 알고리즘의 설계 및 구현하였다.

1. 서 론

나날이 인터넷이 발전함에 따라 네트워크 상에서의 정보의 교류가 많아지고 있다. 1989년 팀 버너스 리(Tim Berners-Lee, 웹의 창시자)는 CERN(유럽 입자 물리 연구소)에 제출한 그의 첫 제안서에서 다음과 같이 웹의 비전을 설명했다.

"…세계적으로 연결된 정보 시스템, 즉 화려한 그래픽과 복잡한 다른 기능들보다 보편성과 이식성이 더 중요한 곳"

이런 의미에서 XML(eXtensible Markup Language)는 인터넷상에서의 효율적인 문서 교환을 위한 표준으로서 중요한 위치를 차지하게 되었다. 전자 상거래 환경에서 XML이 갖는 의미는 실로 크다고 할 수 있다. 기존의 HTML은 구조적이지 못하고, 구문검사를 할 수 없을 뿐더러 구조적인 내용 제공을 할 수 없어서 문서 교환에서는 부족한 점이 많았다. XML이 문서 교환 표준으로 많은 관심을 받는 이유는 HTML Tag로서는 표현할 수 없는 문서의 구조를 자유롭게 기술할 수 있을 뿐만 아니라 문서에 들어있는 정보를 식별할 수 있다. 이렇듯 구조적이고 의미있는 문서를 표현할 수 있는 XML문서는 앞으로 효율적인 문서 관리나 검색 시스템으로 개발할 수 있는 잠재력이 있다.

구조화되고 확장성 있게 내용정보를 제공할 수 있는 XML 문서로부터 필요한 정보를 추출해 내기 위해, XML구조에 맞게 많은 질의 언어들이 개발되었다. XML정보 추출을 위해 개발된 질의언어는 다음과 같다. : Lore[2], XML-QL[3], XML-GL[4], Quilt[5], XQL[6], XQuery[7].

XQL은 정규 경로 표현에 의해서 임의의 경로로 이동이 쉽고, 질의 표현이 간단하고 쉬워서 누구나 쉽게 이해하고 사용할 수 있다는 장점을 가지고 있다.

본 논문에서는 XQL을 사용하여 질의를 하는 관계형 데이터 베이스를 이용한 XQL질의 처리 시스템에서 정보의 효율적인 검색을 위한 "인덱스 알고리즘"을 제안하고 설계 및 구현에 대해 기술한다.

2. 관련연구

관계형 데이터베이스에 XML 문서를 저장하고, 저장된 문서에 경로표현을 제공하는 질의언어로 질의를 하게되면 만족하는 경로를 표현하고 수행하기 위해서는 문서 트리의 깊이만큼을 조인해야 하는 단점을 갖고 있다. 이 방법은 트리의 깊이에 비례하여 성능이 떨어진다는 것을 나타내는 것이다. 트리를 이용하는 방법으로서 과거에 제안된 [9]은 정규경로 표현을 위해서 성능이 그렇게 좋지 않았다. 왜냐하면 경로의 깊이가 깊어지거나 복잡한 질의에 있어서는 트리 운행에 많은 시간이 소요된다. [10]는 *DFS-Numbering* 방식을 이용하도록 저장되고, 검색하는 방법으로서 기본적인 질의만 지원되었다. [10]은 [8,9]의 단점인 트리 검색성능을 개선하였다.

XQL 질의 처리 시스템에서는 XQL에서 제안한 질의 범주에 해당하는 질의 연산을 제공하고 있다. 본 논문에서는 이 시스템에서 효율적인 "인덱스" 연산을 하기 위한 알고리즘을 제안하고 시스템에서 자동으로 생성될 수 있도록 설계 및 구현하여 효과적인 XML문서 검색에 기여한다.

3. 질의 처리 시스템의 구성

XQL 질의 처리 시스템은 크게 문서관리부분과 XQL 처리부분으로 나누어볼 수 있다. 문서관리부는 XML형태로 구성되어 있는 문서를 파싱후에 인덱싱(*DFS-Numbering*)[8] 하여 데이터베이스에 저장하는 부분이고, XQL 처리부분은 사용자가 입력한 XQL형태의 질의를 [12]을 이용하여 XQL BNF에 맞게 파싱한 후 알맞게 처리된 SQL형태로 변환하여 저장된 XML문서를 검색하는 부분이다. 본 알고리즘은 XQL을 처리하는 과정에서 "인덱스 메소드"를 지원하기 위해 고안되었다.

3.1 저장 테이블의 기본 구조

XML 형태로 되어 있는 문서는 각각의 요소별로 파싱해서 미

리 구성된 (그림 1)과 같은 RDB 테이블에 저장한다.

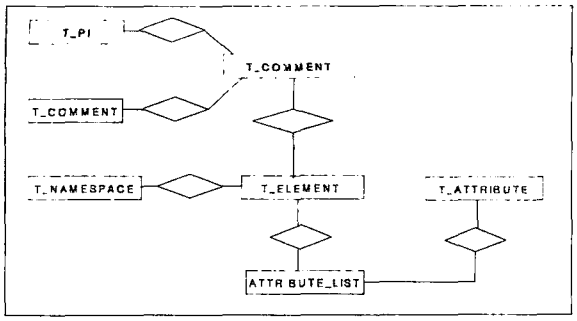

그림 1 XQL스키마를 위한 E-R모델링

본 논문에서 제안되는 알고리즘은 구성된 관계형 데이터베이스들 중 T_Element 테이블에 저장된 데이터를 이용하여 XML트리 구조에 맞는 SQL문을 구성한다. T_Element 테이블의 구조는 (표 1)과 같다.

Column 이름	기능
ID	Parsing 된 Element들의 고유 번호
DOC_ID	Element가 속한 문서 번호
TSTART	문서의 트리구조에서 DES_Numbering에 의해 부여받은 Element의 시작 Tag 위치 정보
TEND	문서의 트리구조에서 DES_Numbering에 의해 부여받은 Element의 종료 Tag 위치 정보
DEPTH	Root에서부터의 Element의 레벨
NAME	Element 이름
CONTENT	Element가 가진 정보(용량이 2Kbytes이하)
CONTENT_LONG	Element가 가진 정보(용량이 2Kbytes이상)
PREFIX_ID	Namespace 테이블과 연결되어 prefix name과 Url을 제공한다.
PARENT	트리 구조에서 부모 노드 ID

표 1 T_Element 테이블의 구조

3.2 인덱스(Index)의 기본 성질

인덱스는 자료의 순번을 이용하여 얻고자 하는 정보를 손쉽게 찾을 수 있도록 하는 방법이다. XQL의 정의에 따라 본 논문에서는 인덱스의 시작을 0으로 하였다. (그림 2)은 XML 문서 데이터의 구조이다. 하나의 *Book* 요소(Element) 아래 여러 개의 *Author, Article, Paper* 요소가 있고, *Paper* 요소는 여러 개의 *Article* 요소로 다시 구성되고, 각 *Article* 요소는 여러 개의 *Author* 요소를 갖는다. *Author* 요소는 하위에 *First name, Last name* 등의 요소를 갖는다.

질의가 *Article/Author[3]*일 경우, 이 질의의 의미는 문서에 포함된 각각의 *Article*에 대해 그 하위 트리를 검색하여 *Author*를 찾고 *Author*를 순번을 매기고, 그 순번이 3인 즉, 네번 째 *Author*의 정보를 찾아서 반환하라는 의미이다.

여기에서 주목해야 할 점은 *Article*의 위치는 문서의 어디이건 상관이 없다는 점이고, 그 하위에 *Author*에 대한 정보가 있다면 이 질의가 만족된다는 것이다.

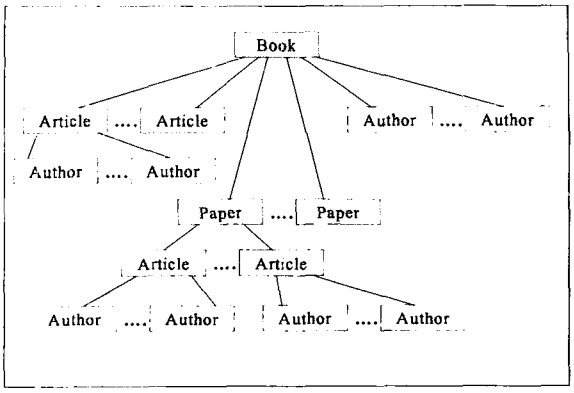

그림 2 XML 문서 구조

위의 질의의 결과로 *Author*의 하위 트리에 있는 *First Name*과 *Last Name*이 모두 결과 값으로 반환된다.

3.3 질의 처리와 인덱스 알고리즘

XQL질의를 위한 파싱(Parsing) 처리는 JavaCC(Java Compiler Compiler)[13]에 의해 처리하였으며, XQL질의 각각의 정의에 따라 알맞은 형태의 SQL로 자동 변환하여 이미 구성된 관계형 데이터베이스에 적용하여 정보를 획득하도록 구현하였다. 인덱스는 경로정보의 어디에나 사용할 수 있고, 본 논문에서 실험한 인덱스를 이용한 XQL질의는 다음과 같은 유형으로 분류할 수 있다.

- // Author[ ]
- Article / Author[ ]
- Book / * / Author[ ]
- Book // Author[ ]
- Article[ ] / Author[ ]
- Book[ ] / * / Author[ ]
- Book[ ] // Author[ ]
- (Book/Article/Author) [ ]

우선 "절대경로"에 대해 알아보자. "절대경로"에서는 요소(Element)의 위치에 대한 제약이 없으므로 요소 이름만으로 순번을 계산하여 결과를 반환할 수 있다.

다음으로 "단순경로"를 생각할 수 있다. "단순경로"의 경우 두 요소이름과 이 두 요소의 관계가 "*부모-자식*"관계임을 명시한 다음 그 부모의 트리 내에서 같은 경로를 갖고 요소이름이 같은 자료를 검색하여 그 순번을 계산하여 주어진 값에 맞는 결과 값을 반환한다.

경로에 "Wild Character"가 들어있는 경우도 생각해야 한다. 이 경우는 "Wild Character"에 대해 파싱 과정부터 따로 설정된 값으로 경로에 대한 정의와 순번에 관한 정보를 표현할 수 있다. 얻고자 하는 요소이름과 그 요소의 하위 트리에 속한다는 것을 시작태그(Start Tag)와 끝 태그(End Tag)의 조건으로 명시하였다. 그리고 연결관계(*Level* 차이)가 "/*/"로 표시되어 있으므로 "부모-자식" 관계가 아니라, 트리의 깊이 수준(Depth Level)에 관하여 "*y.depth=x.depth-2*"로 추가 정보를 조건으로 세공하여 "Wild Character"에 대한 처리를 하였다. 인덱스 순번은 같은 조건에 속하고 시작태그(Start Tag)의 숫자

가 작은 요소(Element)의 개수를 이용하여 결과 값을 얻었다.

위와 비슷한 방법으로 "//"도 처리 할 수 있다. (그림 3)에서 보여지는 XQL 질의 예제 *Book//Author[2]*는 *Book*의 하위 트리에 속한 *Authors*중 세 번째 *Author*의 하위 트리를 반환하라는 의미이다. 하지만 그 하위 트리에 속한 노드이면서 *Author*이라는 이름을 가지기만 하면 그 조건이 충족되므로 그 경로나 깊이 수준(Depth Level)의 정보가 제공되지 않고 오직 트리 내부에 속했는지 그 요소이름이 무엇인지에만 초점을 맞추었다. 그렇기에 순번을 계산하기 위해 굵은 글씨로 표현된 *count*의 조건 절에서 역시 하위 트리에 대한 조건만이 명시되어있다.

```
-- book//author[2]
select id,doc_id,tstart,tend,depth,name,content
from t_element a
where exists
 (select x.* from t_element x, t_element y
  where x.name='author' and a.doc_id=x.doc_id
   and a.tstart>=x.tstart and a.tend<=x.tend
   and y.name='book' and x.tstart>=y.tstart
   and  x.tend<=y.tend
   and 2=(select count(id) from t_element z
          where z.name='author'
          and z.doc_id=x.doc_id and z.tstart<x.tstart
          and z.tstart>=y.tstart and z.tend<=y.tend)
);
```

그림 3 Book//Author[2]에 대해 자동 생성된 SQL

다음으로 생각해 볼 수 있는 경우는 경로에 대해 모두 인덱스가 붙은 경우이다. (그림 4)의 XQL 질의 예제 *Article[4]/Author[2]*은 각각의 *Article*에 대해 순번을 매기고 그 순번이 4인 즉 다섯 번째 *Article*을 찾고 그 하위 트리에서 *Author*이라는 이름을 가진 요소들 중에 세 번째 *Author*의 하위 트리를 반환하라는 의미이다. 이 질의에서는 인덱스가 두 번 사용되었으므로 각각의 경로에 맞게 인덱스 알고리즘이 두 번 사용되어야 한다. (그림 4)에서는 이 조건에 맞게 생성된 SQL문을 보여준다. (그림 4)에서 굵은 글씨로 된 첫 번째 *Author*과 *Article*은 둘 사이의 관계가 부모-자식 관계임을 정의하고, 그 부모 *Article*이 다섯 번째 요소라는 조건과 그 Article의 하위 트리에 속한 요소중 이름이 *Author*인 자료를 찾아서 세 번째 *Author*의 조건을 추가한다. 여기에서도 앞에서 했던 방법으로 목표 요소보다 시작태그의 숫자가 작은 요소의 개수를 세서 그 순번을 찾는다.

```
-- article[4]/author[2]
select id,doc_id,tstart,tend,depth,name,content
from t_element a
where exists
 (select * from t_element x, t_element y
  where x.name='author' and a.doc_id=x.doc_id
   and a.tstart>=x.tstart and a.tend<=x.tend
   and y.name='article' and x.parent=y.id
   and 4=(select count(id) from t_element w
          where w.name='article'
          and y.doc_id=w.doc_id  and w.tstart<y.tstart)
   and 2=(select count(id) from t_element z
          where z.name='author'
          and y.doc_id=z.doc_id and y.tstart<=z.tstart
          and z.tend<=y.tend and z.tstart<x.tstart)
);
```

그림 4 Article[4]/Author[2]에 대해 자동 생성된 SQL

4. 결론 및 향후 연구

본 논문에서는 XML문서의 요소, 속성의 특성 및 부모-자식 요소 관계를 유지하는 "*DFS-Numbering*" 방식을 사용하여 관계형 데이터 베이스를 기반으로 구축된 XML문서 데이터베이스에서 XQL질의어를 처리에 있어 인덱스를 효율적으로 활용하기 위한 알고리즘을 제안하고, XQL 질의 처리 시스템에서 이 알고리즘을 구현하였다.

향후 연구 계획은 데이터 베이스 서버 내부에서의 질의 처리 과정을 연구하여 기존의 색인 시스템보다 기능적으로 향상 된 색인 시스템으로 개발하여 상업적으로 활용 가능한 시스템을 구현하는 것이다.

5. 참고문헌

[1] Tim bray, Jean Paoli, C.M.Sperberg-McQueen, and EveMaler. Extensible Markup Language(XML)1.0 second edition W3C recommendation. Technical Report REC-xml-20001006, World Wide Web Consortium, October 2000.

[2] S. Abiteboul, D. Quas, J. McHugh, J.Widom, J.Widom, and J.L. Wiener. The Lore query language for semistructured data. International Journal on Digital Libraries, 1(1):68-88, April 1997.

[3] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu, A query language for XML. In Proceedings of the 8th International World Wide Web Conference, pages 77-91, Toronto, Canada, Mary 1999.

[4] Stefano Ceri, Sara Comai, Ernesto DAmiani, Piero Fraternali, Stefano Paraboschi, and Letizia Tanca. XML-GL: A graphical language for querying and restructuring XML documents. In Proceedings of the 8th International World Wide Web Conference, pages 93-109, Toronto, Canada, May 1999.

[5] Don Chamberlin, Jonathan Robie, and Daniela Florescu. Quilt: An XML query language for heterogeneous data sources. In International Workshop on the Web and Databases(WebDB'2000), Dalla, TX, May 2000.

[6] Jonathan Robie, Joe Lapp, David Schach, XML Query Language (XQL), http://www.w3.org/TandS/QL/QL98/pp/xql.html#

[7] Don Chamberlin, Daniela Florescu, Jonathan Robie, Jrme Simon, and Mugur Stefanescu. XQuery : A Query Language for XML W3C working draft. Technical Report WD-xquery-20010215, World Wide Web Consortium, February 2001.

[8] 이용석, "XML문서 저장시스템의 설계 및 구현", 한국외국어대학교 석사학위 논문,1999, 2

[9] Jason McHugh and Jennifer Widom. Query optimization for XML. In Proceedings of the 25th VLDB Conference, pages 315-326, Edinburgh, Scotland, September 1999.

[10] Quanzhong Li, Bongki Mon : Indexing and querying XML data for regular path expressions. VLDB 2001.

[11] Web Gain, http://www.metamate.com/products/java_cc/

[12] David Megginson, http://www.megginson.com/ SAX/