분할 저장 시스템에 적합한 XPath 질의 처리기 설계*

고영기[0] 홍의경
서울시립대학교 컴퓨터통계학과 데이터베이스 연구실
{ykko[0], ekhong}@venus.uos.ac.kr

Design of XPath Query Processor in Decomposition Storage System

Young-Ki Ko[0] Eui Kyeoung Hong
Department of Computer Science and Statistics, University of Seoul

요 약

인터넷에서 XML은 고유의 확장성과 문서 관계성의 우수성을 활용하여 새로운 정보 공유 환경의 표준으로 자리잡고 있으며 XML문서 안의 정보 검색을 위해서 XPath 질의어가 널리 사용 중이다. 따라서, XML 문서를 데이터베이스에 효율적으로 저장하고 검색하는 연구들이 진행되고 있다.

본 연구는 관계형 데이터베이스(RDBMS)를 통하여 XML문서를 저장하고 검색할 수 있게 하기 위해 XPath 질의어에 적합하도록 하부 저장 스키마를 설계하였다. 그리고, XPath 질의를 SQL문으로 변환시켜 수행함으로써 XML 데이터에 대한 접근을 허용하였다. 더욱이 SQL문 수행 후의 결과를 효율적으로 DOM 형식의 XML 문서를 생성시킴으로써 문서의 재 조작을 가능하게 하였다.

1. 서 론

지식과 정보 교류의 기반이 웹으로 이동하면서 W3C에서는 기존의 HTML의 단점을 보완하고 SGML의 복잡성을 제거한 XML(eXtensible Markup Language)을 웹 문서의 표준으로 제정하였다. XML은 확장성과 문서 관계성 표현이 우수하여 새로운 정보 공유 환경의 표준 매체로 자리 잡아가고 있다. 또한 XML은 이 기종 시스템 간의 정보 교환을 용이하게 하는 특성을 가지고 있어 EC/EDI, 전자 도서관, 전자 상거래 등 다양한 응용 분야에서 사용되고 있다[4,5]. 이런 XML문서를 파일 시스템, 반 구조적인 저장 시스템, 데이터베이스 시스템 등에서 관리할 수 있는데, 현재 XML 문서를 데이터베이스 시스템에 저장하는 기법에 대한 연구가 활발히 진행 중이다. 또한 다양한 형태로 저장된 XML 문서에서 원하는 데이터를 추출하고 변환하며, 작은 XML 문서들을 통합하는 작업들이 필요하게 된다. 이런 문제들은 데이터베이스에서도 나타나고 있으며 이를 해결하기 위해서 일반적으로 질의어를 사용하게 된다. W3C에서는 XML 문서에 대한 질의어로서, 구조적 질의어인 XPath를 핵심으로 하는 XQuery를 새로운 질의어의 표준으로 발표하였다[1,3]. 따라서 XML 문서를 관계형 데이터베이스 시스템(RDBMS)에 효율적으로 저장하는 시스템을 개발하고, 저장된 XML 문서에서 XML사용자가 필요한 정보를 검색할 수 있도록 하기 위해 XQuery 질의어를 지원하는 연구가 필요하다.

본 논문에서는 서울시립대학교에서 설계한 XML 문서 저장 기법[5]을 XPath 질의 수행을 효율적으로 할 수 있도록 저장 스키마를 변경하였으며, XPath질의어를 통해 XML 데이타를 접근할 수 있도록 하기 위해서 XPath를 SQL문으로 변환하여 실행하는 XPath 질의 처리기를 설계하였다.

* 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음.

본 논문의 구성은 다음과 같다. 2절에서는 XML 문서 구조의 저장 기법 및 XPath에 대해 알아보고, 3절에서는 본 논문에서 사용하는 XML 문서 저장 시스템에 대하여 설명한다. 4절에서는 XPath를 RDBMS에서 수행 가능한 SQL문으로 변환하여 처리하는 XPath 질의 처리기 설계를 논의한다. 마지막으로 5절에서는 결론 및 향후 연구 방향을 기술한다.

2. 관련 연구

2.1. XML 문서 구조 저장 기법 분류

데이터베이스 기술 분야에서는 방대한 양의 데이터를 웹을 통해 공유하기 위하여 XML 문서를 효율적으로 DBMS에 저장하고 검색하는 기술에 대한 연구가 활발히 진행되고 있다[3,5].

그 중 관계형 데이터베이스에 저장하는 방식은 크게 가상 분할 저장 기법과 분할 저장 기법으로 나눌 수 있다. 가상 분할 기법은 XML 문서 전체 내용을 하나의 BLOB (Binary Large Object) 형으로 데이터베이스에 저장하고 XML 문서의 엘리먼트들을 추출하고, 그 엘리먼트가 실제 문서상에 위치 정보를 표현하기 위해 시작 오프셋과 종료 오프셋을 사용하여 표현한다. 가상 분할 기법은 검색 효율이 우수하지만 문서 일부분이 갱신되었을 때 오버헤드가 크기 때문에 일반적으로 검색 위주의 응용 개발 시스템에 주로 사용된다. 그리고, 분할 저장 기법은 XML 문서의 내용을 엘리먼트 단위로 나누어서 저장하고 검색 시 구조 정보를 참조하여 해당 요소 노드나 하위 엘리먼트들을 재구성하여 검색 결과를 반환하는 기법을 말하는데 사용하는 테이블의 개수에 따라 단일 테이블에 모든 정보를 저장하는 단일 에지 테이블 (single edge table) 방식과 여러 개의 테이블 집합을 이용하여 저장하는 관계형 테이블 집합 방식이 있다[2,3].

2.2. XPath

XPath는 XSLT 1.0과 함께 1999년 11월 W3C에서

XPath 1.0으로 권고(Recommendation)되었다. XPath는 XML문서의 일부분을 지시하기 위해서 사용된다. XPath는 URL 경로 표기법을 사용하여 XML문서의 계층적인 구조를 논리적으로 탐색하는데 이 구조는 요소 노드(element node), 속성 노드(attribute node), 값 노드(text node)를 사용하여 트리 구조로 형성된다. 또한 XPath는 각 노드의 문자열 값을 계산하는 방법을 정의하고 있다[1].

XPath는 노드의 경로를 나타내기 위해 위치 경로(Location Path)를 사용하는데, 경로 위치 표현방식은 절대 경로와 상대 경로 두 가지 방식이 있다. 절대 경로란 / 로 시작하여 문서의 루트 노드를 시점으로 경로를 지시하는 것을 말하고, 상대 경로는 현재 노드를 기점으로 해서 요구되는 상대적인 노드의 경로를 나타낸다. 이때 경로 위치를 기초부(basis)와 술어부(predicate)로 나누어 설명할 수 있는데, 기초부는 축(Axis)과 노드 검사 부분으로 구성된다. 여기서 축이란 문맥 노드(Context Node)와 경로 단계로 선택된 노드간의 관계를 명시한다. 한편, 술어부는 대괄호([,])로 묶인 표현식을 말하는데, 축에 관련된 노드집합을 새로운 노드집합으로 만들기 위한 필터링 작업을 한다.

표 1. XPath의 생략된 표기식

비생략된 문법	생략된 문법
child::	(없음)
attribute::	@
/descendant-or-self::node()	//
parent::node()	..
(문서 루트 노드)	/

본 연구에서는 XPath의 모든 기능을 만족하기보다는 XQuery에서 지원하는 생략된 표기식만을 사용한다. 비생략된 표기식과의 비교는 표 1를 통해 확인할 수 있다. 또한, 단일 술어부에 대한 지원을 보장하며, 술어부에 사용되는 비교 연산자로는 관계 연산자만을 사용한다.

3. XML 저장 시스템 구조

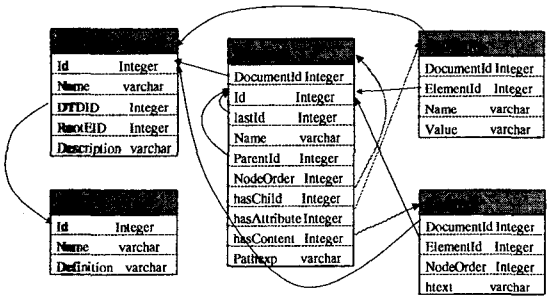


그림 1. XML 저장 스키마

본 논문은 관계형 테이블 집합 방식을 응용하여 개발한 서울시립대학교 저장 시스템 방식을 활용하였다. 논문의 저장 시스템은 XML 문서의 구조정보를 유지하기 위해 엘리먼트의 경로를 기록하는 PathExp 정보와 부모 엘리먼트 ID를 저장하고 있으며, 문서 구조 검색 시 범위 검색을 용이하게 하기 위해서 DFS Numbering 기법을 추가적으로 사용하였다. 본 시스템의 XML 문서 저장 스키마는 그림 1과 같이 구성하여 각 노드의 정보를 분할하여 저장할 수 있게 하였다[5].

본 시스템에서 XML 문서를 저장하기 위해 다음과 같은 과정을 거친다. 우선 XML 문서를 파싱해야 하는데 파싱 방법에는 노드 중심의 DOM 파서와 이벤트 중심의 SAX 파서가 있다. SAX 파서가 DOM 파서보다 문서를 파싱하는데 걸리는 시간이 더 적기 때문에 본 논문에서는 SAX2 파서를 이용하였다. 문서를 파싱해서 발생된 이벤트의 정보에 따라 저장 관리기를 통해 엘리먼트, 애트리뷰트, 컨텐츠 테이블에 각각의 정보를 저장한다.

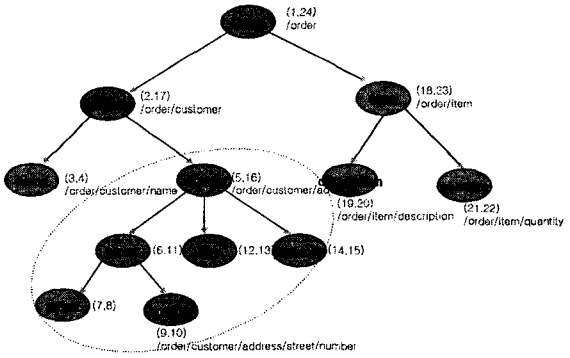


그림 2. DFS Numbering과 Pathexp 정보 표현 트리

SAX 이벤트의 발생 순서가 XML문서의 루트노드부터 시작해서 DFS방식으로 각 노드 이벤트를 발생시키기 때문에 각 엘리먼트에 대해 DFS Numbering 방식으로 쉽게 고유 ID를 부여할 수 있다. 또한 현 이벤트된 엘리먼트의 경로 위치를 기록하기 위해 PathExp 정보를 저장한다. 그림 2는 DFS Numbering과 PathExp 정보를 예제 XML문서에 적용하여 요소 노드 중심으로 표현한 트리이다.

만약 노드 타입이 엘리먼트이면 Element 테이블에 엘리먼트 이름, 부모 엘리먼트 ID, 경로, 현 엘리먼트의 종료 ID 등의 정보를 저장하고, 하위 요소 노드의 존재 유무, 속성 노드의 존재 유무 등 부가 정보 역시 저장하였다. 이러한 이벤트 발생 순서에 따라 각 자식 노드에 대해 순회하며 적용돼서 모든 엘리먼트 정보를 RDBMS에 저장한다.

XML 문서 검색은 XPath 질의 처리기를 통해서 수행되는데, XPath 질의 처리기는 XPath질의어를 SQL문으로 변환하여 처리하는 작업을 수행한다. 그리고, 질의 처리기에 의해 반환된 결과 집합은 DOM형식의 트리 구조로 재구성되어 XML문서 조각을 형성하게 된다. 이에 대한 세부적인 내용은 다음 절에서 알아본다.

4. XPath 질의 처리기 설계

XML은 XML문서에서 정보를 검색하기 위해 XPath 질의어를 주로 사용하고 있다. 따라서 RDBMS를 기반으로 한 XML문서 저장 시스템 역시 XPath 질의어에 대해

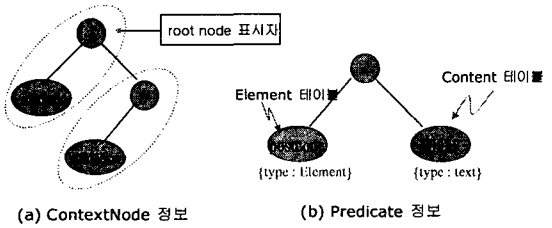
적절한 처리를 수행해야 한다.

본 절에서는 저장 시스템하에서 XPath 질의를 수행하기 위해 RDBMS의 표준 질의어인 SQL문으로 변환하여 처리하는 XPath 질의 처리기를 설계하였다.

XPath 질의어는 본질적으로 엘리먼트의 범위 검색을 위주로 수행하는 질의어이기 때문에 변환되어야 하는 SQL문을 반환되는 요소 노드의 범위 정보를 구할 수 있도록 생성하였다. 따라서 범위 정보 생성을 위해서 본 논문의 XPath 질의 처리기는 XPath 질의 파서 단계, 토큰 분류 단계, 그리고 SQL 질의문 생성 단계 등 3단계로 구성하였다.

본 논문의 예제 XML문서에서 address의 postcode값이 60325인 주문 사항의 주소를 확인하는 XPath질의문을 작성하면 "/order//address[postcode='60325']"가 되며 이 질의문을 각 단계별 처리 내용을 설명하는데 활용하였다.

먼저 XPath 파서 단계에서는 사용자의 질의어에 대한 적합성을 검증하고 축 타입, 노드 타입, 속성 타입, 문자 타입, 비교 연산자 타입 등의 토큰 정보를 발생시킨다. 즉, 본 예제의 경우 {"/","order","//","address","[","postcode","=","'60325'","]"}와 같은 토큰 정보가 발생된다.



(a) ContextNode 정보 (b) Predicate 정보

그림 3. 토큰 분류 단계

토큰 분류 단계에서는 XPath 파서 단계에서 발생된 토큰을 문맥 노드 정보를 저장하는 ContextNode와 술어부 정보를 저장하는 Predicate로 분류한다. 만약 토큰이 술어부의 시작을 알리는 "["이 나타나기 이전이라면 문맥 노드에 해당하기 때문에 ContextNode에 저장하며 "//"토큰이 있는지 검색한다. 술어부는 "["토큰 다음부터 "]" 토큰 이전까지 해당되며 이 부분의 토큰은 Predicate에 저장된다. Predicate에는 술어부에 대한 정보를 담으면서 비교 노드의 유형이 요소 노드인지 속성 노드인지를 확인하는 유형 판별 인자를 포함해서 저장한다. 위 예제의 XPath질의문 같은 경우, 토큰 분류기에 의해 ContextNode에는 "/order//address" 정보가 그림 3 (a)와 같이 저장되며, Predicate에는 "postcode ='60325'" 정보가 그림 3 (b)와 같이 저장이 된다.

Path 질의 변환 단계에서는 토큰 분류 단계에서 구성한 ContextNode 정보와 Predicate 정보를 활용하여 RDBMS에서 실행될 SQL문으로 변환 처리한다. ContextNode 정보는 요소 노드의 집합으로 표현되며 저장 스키마 중 엘리먼트 테이블과 관련되어 있으며, 위치 정보를 나타내는 PathExp 필드에 대해 연산을 하면 되고 범위 정보를 나타내는 엘리먼트의 시작 ID와 종료 ID를 검색하면 된다. 그리고, Predicate 정보는 좌측 비교 연산자의 유형이 요소 노드이면 엘리먼트 테이블과 컨텐츠 테이블에 관련되고 속성 노드이면 애트리뷰트 테이블이 질의 생성에 관련이 있다. 이 때 주의할 점은 문맥 노드 정보에 "//"라는 위치 지시자 토큰이 존재하면 PathExp필드와 비교할 때 ContextNode 정보를 "/order/%/address"로 변경하고 LIKE연산을 통해 질의를 생성해야 한다. 즉 위 예제 질의문의 경우 다음과 같은 SQL문을 생성할 수 있다.

```
SELECT e1.id startId, e1.lastid lastId
FROM helement e1, helement e2, content con1
WHERE e1.Pathexp LIKE'/order/%/address'
    and e2.Pathexp LIKE '/order/%/address/postcode'
    and e2.Id >= e1.Id and e2.lastId <= e1.lastId
    and con1.htext ='60325' and e2.id = con1.elementId
```

본 과정을 통해 만들어진 SQL 질의문을 RDBMS에서 수행한 후 반환되는 결과 값을 이용하여 반환되어야 하는 엘리먼트의 범위 ID를 얻게 된다. 이 범위 ID를 이용하여 범위 안에 있는 모든 엘리먼트, 애트리뷰트, 컨텐츠 등의 정보를 검색하고, 반환된 결과를 DOM형태의 XML 데이터 조각으로 만든다.

위와 같은 과정을 통해 XPath 질의문을 처리해서 요구되는 XML 데이터를 구성하였다.

5. 결론 및 향후 연구 방향

본 논문에서는 DTD 독립적으로 저장하는 분할 방식 저장 시스템에서 XPath 질의어를 통하여 XML 데이터를 검색할 수 있도록 하부 저장 스키마를 설계하였으며, XPath 질의어를 RDBMS의 SQL문으로 변환시켜서 수행함으로써 XML 데이터에 대한 검색 질의를 허용하였다.

XPath 질의어를 RDBMS의 표준 질의어인 SQL문으로 적절히 변경 실행할 수 있는 질의 처리기를 설계하였다. 또한 RDBMS의 결과값을 DOM형식의 XML문서로 생성시킴으로써 타 응용 프로그램에서 바로 사용할 수 있게 되었다. 향후에는 본 저장 시스템이 XML의 표준 질의어인 XQuery 언어를 지원할 수 있게 함으로써 더 폭넓은 범위의 질의 수행 능력을 갖출 수 있게 할 예정이다.

6. 참고 문헌

[1] J.Clark and S. DeRose, "XPath 1.0 : XML Path Language," W3C Recommendation, http://www.w3.org/tr/XPath, 1999.

[2] D.Florescu and D.Kossman, "Storing and Querying XML Data Using an RDBMS," In IEEE Data Engineering Bulleltin 22(3), pp.27-24, 1999.

[3] J. Shanmugasundaram, et al. "Relational Databases for Querying XML Documents : Limitations and Opportu -nities," Proc. of the 25th VLDB Conf., 1999.

[4] J. Shanmugasundaram, et al. "A General Technique for Querying XML Documents Using a Relational Database System," SIGMOD Rec, Vol 30., No 3. pp.20-26, 2001.

[5] 김훈, 홍의경, "객체 관계 데이터베이스를 이용한 XML문서 저장 모델 설계," 한국정보과학회 가을학술 대회 논문집 27(2), pp.225-227, 2000.