XML 문서에 대한 효율적인 검색기법

윤정혜*, 이미희**, 우용태*
*창원대학교 컴퓨터공학과
**창신대학 정보통신과
e-mail:jhyun@ce.changwon.ac.kr

An Efficient Querying Method for XML Documents

Jeong-Hye Yun*, Mee-Hee Lee**, Yong-Tae Woo*
*Dept of Computer Engineering, Changwon University
**Dept of Telecommunication, Changshin College

요 약

최근에 전자상거래, e-learning, e-book 등과 같은 다양한 분야에서 디지털 문서의 효율적인 관리를 위하여 XML문서를 이용하고 있다. 이에 따라 대량의 XML 문서들을 효율적으로 저장하고 관리하는 시스템의 필요성이 증가하고 있다. 이러한 시스템의 대부분은 XML 문서가 트리 구조로 이루어져 있기 때문에 DOM을 이용하고 있다. 그러나 DOM은 문서 전체의 문서 구조 정보를 메모리 트리 구조로 생성하는 과정에서 많은 시스템 자원을 필요로 한다. 본 논문에서는 이벤트-기반인 SAX를 이용하여 문서의 구조 정보를 내부 트리 구조로 만드는 대신 구문 분석 이벤트를 직접 응용프로그램에 전달하는 방법을 사용함으로써 DOM에서의 시스템 자원을 많이 사용하는 문제를 해결하였다.

1. 서론

인터넷의 대중화에 따라 웹 문서가 기하급수적으로 늘어나고 있지만 현재 인터넷상의 대부분 정보는 문서의 구조보다는 표현에 중점을 두고 있어 특정 응용분야의 구조를 표현하는 문서로는 기능이 부족하다. 이에 W3C(World Wide Web Consortium)에서는 문서의 구조정의 및 메타데이터를 제공하여 텍스트를 인코딩하는 마크업 언어인 XML(eXtensible Markup Language)을 차세대 웹 문서의 표준으로 제안하였다[1].

XML 문서는 문서의 구조적 특성이 적절한 태그에 의해서 분리되고 재현양식에 관한 정보는 XSLT (XML Stylesheet Language for Transformation) 등과 같은 별도의 스타일 문서에 의해 제공된다. 따라서 마치 데이터베이스에 질의를 가하는 것처럼 문서를 검색할 수 있으며 트랜잭션 단위의 인트라넷 데이터의 표현이 용이하다는 장점을 가지고 있기 때문에 웹 상에서의 다양한 전자 문서를 표시하기에 매우 적합하다. 또한 XML은 논리적 구조를 따르는 여러 DTD와 XML Schema를 사용할 수 있기 때문에 문서의 관리나 구조검색을 효율적으로 수행하는데 이용될 수 있다.

본 논문에서는 LOB형태로 저장된 XML문서에 SAX와 색인을 이용함으로써 아주 적은 메모리 용량으로 XML문서를 파싱하여 메모리의 오버헤드를 줄일 수 있도록 모델을 설계하였다. 그러나 SAX는 문서가 설계된 방법보다 실제 내용에 집중하기 때문에 부모/자식간의 구조 정보를 알기가 어렵다. 본 모델은 트리 구조에서 엘리먼트 속성으로 위치가 결정되는 EI(Element Index) 요소를 주어 메모리 사용량과 속도의 효율성을 높이고 문맥 정보를 자동으로 유지하도록 한다.

제안된 기법의 효율성을 검증하기 위해서 컴퓨터 관련 논문을 대상으로 데이터베이스, 통신, 멀티미디어, 인공지능, 소프트웨어 공학으로 도메인을 나누어 총 1,098개의 대표용어와 22,400개의 엘리먼트 노드를 이용하여 실험을 하였다. RDBMS와 DOM과의

XML문서 처리 시간을 비교 실험함으로써 제안된 기법에 대한 성능을 평가하였다.

2. 관련 연구

인덱스 기법 연구는 XML문서의 논리적 구조를 수용하는 구조 정보의 표현으로 XML문서를 효율적으로 검색하고 저장하므로 활발히 진행되고 있다. 구조화된 문서관리 시스템은 기존의 정보 검색에서의 문서 단위를 위주로 한 검색이외에 엘리먼트 단위의 검색과 문서를 구성하는 논리적인 구조에 의한 구조 검색, 그리고 엘리먼트 특성 값에 대한 질의 등을 지원하고 있다.

문서 정보를 하나의 레코드에 반영하는 문서 단위의 구문 트리는 문서 변경 시 특정 레코드에 변경을 가할 수 있고 엘리먼트 단위의 구문 트리는 B+트리를 사용함으로써 빠른 구조 검색을 할 수 있다[2]. RDBMS에서의 동적 색인을 위한 연구로는 엘리먼트의 이름으로 구조적 정보를 찾는 방법[3]과 연산에 의해 XML문서를 동적으로 갱신하는 방법[4]이 있다. 전자의 경우 두 엘리먼트의 구조적 정보를 갖는 식별자를 통해 구조 정보를 빠르게 계산할 수 있고 후자의 경우는 XQuery(XML Query)를 이용하여 엘리먼트를 삽입, 삭제, 변경할 수 있다. 엘리먼트 타입을 이용한 구조 정보의 표현은 주로 OODBMS에서 많이 이용되고 있는데 주로 엘리먼트의 경로를 저장한 후 검색 시 XPath(XML Path Language)를 이용하여 검색 속도를 향상시키고 있다[5].

그러나 이러한 연구들은 대부분 DOM(Document Object Model)과 같은 트리-기반을 이용하여 문서의 구조 정보를 추출하는 것이 대부분이다. DOM을 이용한 방법들은 문서 전체의 문서 구조 정보를 메모리에 트리 구조를 생성하는 과정에서 많은 시스템 자원을 필요로 하고 패스의 깊이가 깊어짐에 따라 노드의 순회가 많아지므로 검색 효율이 떨어진다.

반면 아직 연구는 미숙한 단계이나 적은 메모리 사용만으로 문서에서 찾고자 하는 위치를 찾아서 사용할 수 있는 SAX에 관한 연구도 진행되고 있다[6]. 그러나 SAX에 관한 연구는 구조적인 정보를 표현하지 못하므로 부모/자식 간의 관계를 잘 나타내지 못하며 문서를 갱신할 때는 적절하게 이용되지 못하고 있다.

3. 제안 모델

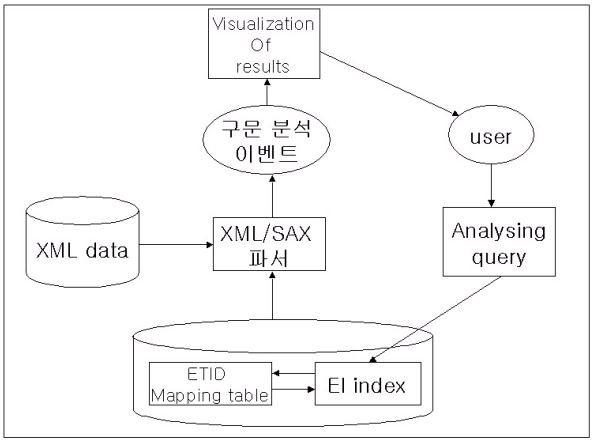
3.1 제안 API

본 논문에서는 SAX파서를 사용한다[7]. 파서의 역할은 XML문서를 순차적으로 읽어 내려가면서 각각의 요소들의 연관 관계를 데이터베이스에 입력하는 과정을 반복하게 된다. SAX파서를 사용하는 것은 DOM파서를 사용할 때 메모리 사용량과 속도의 효율성이 떨어지는 문제점을 해결하기 위함이다.

그러나 SAX는 이와 달리 DOM 생성작업이 단순하지 않다. 실제로도 DOM파서는 SAX파서를 이용하되, DOM 생성 부분을 추가하고 있는 경우가 대부분이다[8]. SAX는 단순하게 어떤 요소를 읽었는지 등의 정보를 줄 뿐, 현재 이 요소가 어떤 요소의 일부인가 등의 문맥 정보를 자동으로 유지해 주지 않는다. 따라서 SAX를 사용하는 데 있어 성패는 SAX에서 얼마나 구조 정보를 유지할 수 있느냐 하는 것이다. 이에 본 논문은 SAX에 EI요소를 줌으로써 이러한 문제점을 해결하였다.

3.2 제안 모델의 구성

본 연구에서는 XML문서의 구조 정보와 SAX를 이용하여 효율적으로 문서를 처리할 수 있는 방법을 제시하였다. 제안한 방법은 사용자가 질의한 질의문의 키워드를 기반으로 구조적인 정보를 수집하여 원하는 문서를 검색하는 것이다. 다음 (그림 1)은 본 논문에서 제안한 모델의 전체적인 구조도이다.

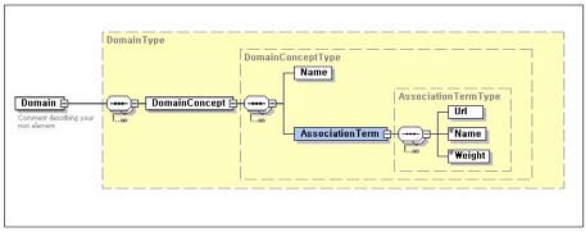


(그림 1)제안 모델 구성도

3.3 구조 정보 표현

본 논문에서는 데이터베이스, 통신, 멀티미디어, 인공지능, 소프트웨어 공학으로 도메인을 나누고 대표

용어와 연관 용어를 두어 대용량의 구조적 문서를 구성하였다. 다음 (그림 2)는 XML문서의 스키마 구조도이다. 스키마 구조도를 살펴보면 최대 4단계의 깊이와 6개의 엘리먼트 타입이 있다. 각 엘리먼트 타입은 고유한 ETID(Element Type Identification) 값을 가진다.



(그림 2)스키마 구조도

3.4 색인을 이용한 검색 알고리즘

본 논문에서 제안하는 색인 알고리즘을 통한 검색은 문서가 파싱되고 메모리 안에 노드 트리가 구성되는 DOM의 경우와는 달리, SAX에 EI요소를 이용함으로써 메모리 사용량과 속도의 효율성을 높일 수 있다. 또한 현재 엘리먼트 노드에 대한 부모 엘리먼트, 자식 엘리먼트, 형제 엘리먼트를 검색함으로써 SAX에서 표현하기 어려웠던 has-a 관계를 갖고 있는 하위 노드들의 구조 정보를 쉽게 표현할 수 있도록 한다.

부모 엘리먼트 노드는 해당 노드의 EI요소 중에서 PID값을 기준으로 하여 PID값과 동일한 PID값을 갖는 노드를 검색하면 된다. 자식 노드의 경우 해당 노드의 EI요소 중 NID를 기준으로 하여 NID값과 동일한 PID값을 갖는 노드를 검색한다. 제안 알고리즘을 살펴보면 (그림 3)과 같다.

```
(1) Analysing query of document keywords
(2) Let, Ci is a content element(s) //Content Search
(3) C_set = group of EI element nodes includes Ci
(4) KeyElement = search ElementType from mapping table of etid
(5) public void startElement(String namespaceURI,
                             String localName,
                             String qName,
                             Attributes atts)
                                      throws SAXException{
  for(int i=0 ; i ≤length(keyElement) ; i++){
    if(localName.equals(String keyElement)){
      for(int j=0; j < atts.getLength(); j++){
        if(atts.getLocalName(j).equals("NID")){
          nid = atts.getValue(j);
          if(C_set[i].nid=nid){
            for(int i=start; i ≤ end; i++){
              chars[i];}}
}   }   } }
```

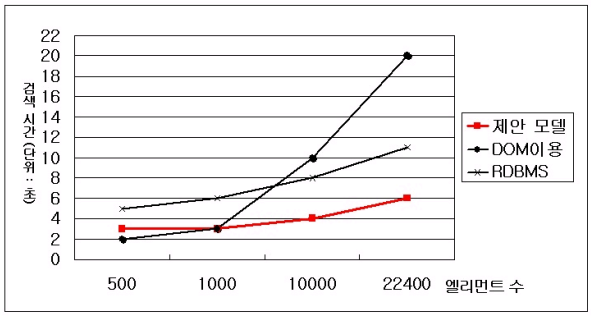
(그림 3)검색 알고리즘

4. 실험 및 성능평가

성능평가를 위한 실험은 오라클 9i를 기반으로 PL/SQL과 XSQL페이지, 그리고 JAVA를 이용하였다. 오라클의 XSQL페이지들은 데이터를 검색하거나 삽입하기 위한 SQL쿼리들이 포함된 XML로 구성된 데이터페이지이다. 사용한 툴은 JDeveloper3.1, XML SPY2004이다.

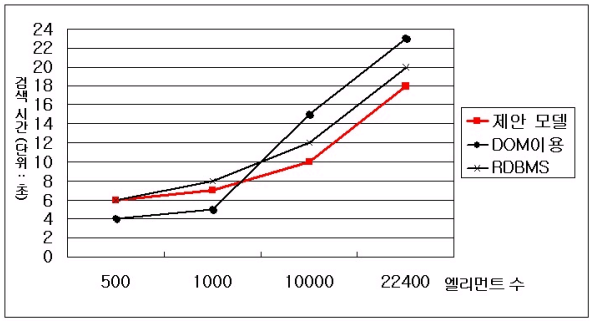
성능평가를 위한 실험 데이터는 데이터베이스, 통신, 멀티미디어, 인공지능, 소프트웨어 공학으로 도메인을 나누어 총 1,098개의 대표용어와 연관된 22,400개의 엘리먼트 노드를 이용하였다. 성능평가를 위한 방법으로는 제안모델과 RDBMS, 그리고 DOM을 대상으로 XML 문서처리시간을 측정하였다. 일반적인 SAX는 XML 문서 전체를 표현 할 수 없으므로 제외되었다.

또한 성능평가를 위한 실험 범위는 다음과 같다. RDBMS는 쿼리 검색 후 XSQL페이지와 XSLT를 사용하여 XML문서로 변환까지의 시간이고 DOM을 이용한 모델은 처음 검색 시 메모리에 트리 구조를 생성하는 과정부터 XSQL페이지와 Javascript를 이용하여 문서 바인딩까지의 시간이다. 제안 모델은 SAX를 이용하여 검색 후 자바의 GUI 인터페이스와 awt를 사용하여 표현까지의 시간이다.



(그림 4)단일 키워드를 기반한 XML 문서 처리시간 비교

(그림 4)는 하나의 키워드를 이용하여 XML 문서 처리 시간을 비교한 결과이다. 그림에서 DOM구조를 이용한 모델은 메모리에 트리 구조를 생성하는 과정에서 시간의 소모가 많았다. RDBMS는 쿼리 검색 속도는 빨랐으나 XSQL페이지와 XSLT를 사용하여 XML문서로 변환 시 시간이 소모되어 제안 모델에 비해 성능이 낮게 나타났다.



(그림 5)다중 키워드에 기반한 XML 문서 처리시간 비교

(그림 5)는 다중 키워드를 이용한 XML문서 처리시간을 나타낸 것이다. 실험 대상은 5개의 키워드를 이용하였다. DOM을 이용한 모델은 처음 키워드 검색 시 메모리에 트리 구조를 생성하지만 두 번째 키워드부터는 트리 구조를 생성 할 필요가 없으므로 XML문서 처리속도가 향상되었다. RDBMS는 쿼리 후 XML문서 변환 시 시간의 소모가 많으므로 다중 키워드 검색 시에는 쿼리 속도에만 변화가 있었다.
또한 엘리먼트의 수가 적을 때는 DOM을 이용하여 XML문서를 표현하는 것이 더 유리하지만 대량의 XML문서일 경우는 제안 모델이 적합함을 알 수 있었다. 그러나 DOM을 이용한 것과 제안 모델의 문서 처리 시간의 차이를 비교해 볼 때 단일 키워드 검색보다 다중 키워드 검색 시 시간 차이가 더 적게 났음을 알 수 있다.

5. 결론

본 논문에서는 SAX와 EI요소를 이용하여 문서의 구조적인 정보를 유지하고 이벤트 연속으로 문서를 접근하는 기법을 사용함으로써 DOM구조에서 대용량의 XML문서 검색 시 제기되는 메모리 사용량의 문제점을 개선하였다. 제안 기법의 효율성은 컴퓨터 관련 논문을 분야별로 도메인으로 나누어 각 도메인별로 대표용어의 엘리먼트 노드를 이용하여 검증하였다.
실험 결과, 엘리먼트 노드 수가 적을 경우에는 검색 시간과 메모리 사용량의 차이가 거의 없었다. 따라서 엘리먼트 노드 수가 적을 경우에는 문서의 수정이 용이하고 프로그래밍 작업이 필요없이 별도의 XSLT를 이용할 수 있는 DOM을 이용한 모델이 효과적이다. 그러나 엘리먼트 수가 많을 경우에는 DOM을 이용한 모델은 메모리 사용량이 많아지고 검색 시간이 길어지므로 제안 모델을 이용하는 것이 더 효과적이다.
오늘날 XML 문서의 사용은 XML이 데이터를 표현하는 표준적인 언어라는 측면에서 중요성이 증대함으로 인해 급증하고 있다. 제안 모델은 대용량의 XML 문서를 검색함에 있어 XML 문서 처리 속도의 효율성이 떨어지는 문제점을 해결하였다.
앞으로의 연구 과제는 본 논문에서 제안한 EI요소를 XML Schema에 의해 자동 분석하여 생성하는 방법이다. 또한, 제안 모델은 SAX를 이용하기 때문에 문서를 읽고 쓰는 속도는 빠르지만 XML데이터를 읽으면서 각각의 태그마다 파서가 이벤트를 생성하므로 별도의 프로그래밍 작업이 필요하다. 따라서 전문적인 지식이 없는 사용자도 쉽게 사용할 수 있는 방법의 개발도 요구되어야 한다.

참고문헌

[1] T. Bray, J. Paoli, and C. M. Sperberg-McQueen, "Extensible Markup Language(XML) 1.0," *W3C Working Draft, 1998*
[2] 한성근, 송정한, 장재우, 김현기, 강현규, “동적 환경에 적합한 SGML인덱스 관리자의 설계 및 구현,” *한국정보처리학회,* vol.27, no.2, pp.18-20, 2000.
[3] C. Kanne, G. Moerkotte, "Efficient storage of XML data," *Proc. Int'l Conference of Data Engineering,* pp.85-94, 2000.
[4] I. Tatarinov, "Updating XML," *Proc. ACM SIGMOD Int'l Conference on Management of Data,* pp.413-423, 2001.
[5] 김성완, 정헌석, 이재호, 임해철, “XML문서에서의 엘리먼트 타입을 이용한 구조적 검색 기법의 설계,” *한국정보과학회,* vol.30, no.1, pp.584-586, 2003.
[6] O. Becker, "Transforming XML on the Fly," *Proc. XML Europe,* 2003.
[7] D. Megginson, "SAX 2.0 Changes", *SAX Project,* 2002.
[8] B. Chang, J. Kesselman, and R. Rahman, "Document Object Model (DOM) Level 3," *W3C Working Draft,* 2003.