XML 문서에 포함된 구조 정보의 표현과 검색

조 윤 기[†] · 조 정 길[††] · 이 병 렬[††] · 구 연 설[†††]

요 약

웹의 발전으로 인터넷 상의 정보 양이 증가하면서 XML을 이용하여 이들 정보를 효율적으로 저장하고 검색하기 위한 많은 연구들이 진행되고 있다. 이 논문에서는 XML 문서의 효율적인 관리와 구조 검색을 위해 구조 정보 표현과 검색 매카니즘을 제안한다. 기존의 방법은 특정 엘리먼트의 부모, 자식, 형제 엘리먼트에 대한 다양한 구조 검색을 효율적으로 지원하지 못한다. 이 논문에서는 XML 문서의 구조 정보를 표현하기 위해 엘리먼트에 대한 부모 노드와 현재 노드의 계층 정보, 형제 노드 및 동일한 형제 노드의 순서 정보를 갖는 고정된 크기의 LETID를 제안한다. 또한, 구조 정보를 검색하기 위해 내용 색인, 구조 색인 그리고 애트리뷰트 색인 모델과 구조 정보 검색 알고리즘을 제안한다. 제안한 방법을 이용하여 XML 문서의 구조 정보를 효율적으로 표현할 수 있을 뿐 아니라 간단한 연산으로 특정 엘리먼트에 직접적인 접근과 다양한 질의 처리가 가능하다.

Representing and Retrieving the Structured Information of XML Documents

Youn-Ki Cho[†] · Jeong-gil Cho[††] · Byung-Ryul Lee[††] · Yeon-Seol Koo[†††]

ABSTRACT

As growing the number of Webs, the total amount of accessible information has been greater than ever. To storage and retrieve the vast information on the Webs effectively, many researchers have been made utilizing XML (extensible Markup Language). In this paper, we propose an effective method of representation and retrieval mechanism for the structured retrieval of the XML documents : (1) the fixed sized LETID (Leveled Element Type ID) that contains the information of elements such as parent node, sibling nodes, and identical sibling nodes, and the hierarchical information of current node, and (2) content index, structure index, attribute index model, and the information retrieval algorithm for the structured information retrieval. With our methods, we can effectively represent the structured information of XML documents, and can directly access the specific elements by simple operations to process various queries.

키워드 : XML, 구조 검색(structured retrieval), 색인 모델(index model), 검색 알고리즘(search algorithm)

1. 서 론

WWW(World Wid Web)의 발전으로 인터넷 사용과 더불어 정보의 양이 급증하면서 인터넷 상의 정보를 효율적으로 사용하기 위한 연구들이 진행되고 있다. 이를 위해 인터넷 상의 정보를 구조화된 문서로 표현함으로써 보다 효율적으로 검색 관리하기 위한 표준이 필요하게 되었다. 구조화된 문서를 표현하기 위한 표준으로 1996년 W3C(World Wide Web Consortium)에서 제안한 XML(eXtensible Markup Language)은 고정된 태그만을 사용하여 정보를 표현하는 HTML(HyperText Markup Language)의 한계를 극복하고, SGML(Standard Generalized Markup Language)의 복잡함을 단순화함으로써 현재 웹 문서뿐만 아니라 전자상거래, 전자도서관 및 EDI(Electronic Data Interchange)를 포함한 다양한 분야에서 사용되고 있다[1-3].

XML 문서는 기존의 문서와 달리 하나의 문서에 내용 정보와 구조 정보를 포함한다. 따라서, 기존의 문서에서 제공하던 키워드를 기반으로 한 내용 정보에 대한 검색뿐만 아니라 문서의 논리적인 구조 정보에 대한 검색이 필요하다. 이를 위해 XML 문서의 구조 정보 표현 방법에 대한 연구가 진행 중에 있다. 그러나 일부 방법들은 조상, 자손, 형제 관계의 엘리먼트에 접근하기 위해 복잡한 연산을 수행하거나 형제관계 노드를 알 수 없다[1, 4]. 또한 구조 트리의 깊이가 깊어질수록 엘리먼트 정보를 표현하기 위해 무한대의 저장 공간이 필요하다.

따라서 이 논문에서는 XML 문서의 구조 정보를 효율적으로 검색하기 위해 XML 문서의 구조 정보 표현 방법과

† 준 회 원 : 충북대학교 대학원 전자계산학과
†† 정 회 원 : 충북대학교 대학원 전자계산학과
††† 정 회 원 : 충북대학교 컴퓨터과학과 교수
논문접수 : 2001년 4월 16일, 심사완료 : 2001년 6월 20일

색인 구조 및 구조 정보 검색 알고리즘을 제안한다.

2. 관련 연구

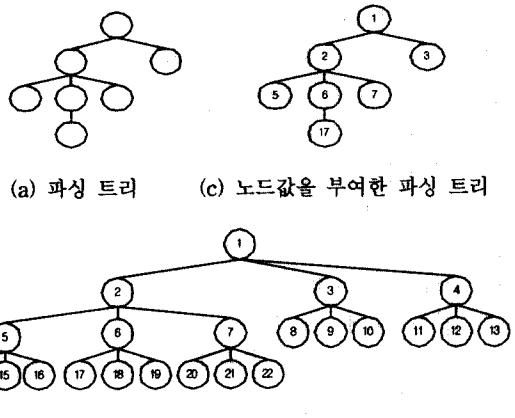
XML로 표현된 구조화된 문서를 검색하기 위해서는 키워드에 의한 문서 단위의 내용 검색뿐만 아니라 엘리먼트를 기본 단위로 하는 구조 검색 및 애트리뷰트 검색이 지원되어야 한다[5, 8]. 이를 위해 구조 정보를 효율적으로 표현하기 위한 연구가 선행되어야 한다. 현재 구조 정보를 표현하기 위한 모델로는 SCL(Simple Concordance List) 모델, K-ary 완전 트리 모델, 그리고 ETID(Element Type ID) 모델 등이 있다.

SCL 모델은 (그림 1)과 같이 구조 문서의 계층적 관계보다는 포함 관계를 이용한 표현 방법으로서 SC-list라는 데이터 타입을 통해 중첩된 정보를 허용하므로 리스트의 리스트와 같은 순환 구조를 다룰 수 있다는 장점이 있다. 이 모델은 텍스트와 마크업에 대해 색인 넘버를 부여한 후, 불용어를 제외한 텍스트 어휘들을 텍스트 인덱스에 색인 넘버로 저장하고, 마크업은 시작 태그와 종료 태그의 쌍으로 마크업 인덱스에 저장한다. 그러나 SCL 구조는 트리의 깊이를 표현할 수 없으므로 조상이나 형제 엘리먼트를 검색할 수 없다는 단점이 있다[4, 7].

```
 100.1          100.2    101 102 103 104    104.1
<doc n=1975><dtitle> Tarzan of the Apes </dtitle>
 104.2          105        105.1
<div type=toc> Contents … </div>
  105.2                  106  107 108   109
<div type=chapter id=C7> The Light of Knowlege …
   109.1   110 111  112     113 114 115  116  117 118  119
   <p> … Let all respect Tarzan of the Apes and Kala, his
            120       120.1    120.2
          mother … </p> … </div>
```

(그림 1) SCL 모델

K-ary 완전 트리 모델은 문서에 대한 트리로부터 이들 노드 중 가장 큰 차수 K를 구하여 K-ary 완전 트리로 재구성한 후, 여기에 문서 트리를 매핑하여 각 노드에 모든 번호를 부여한다. 이 모델은 문서 구조 사이의 계층 관계를 간단한 공식을 통해 쉽게 구할 수 있다는 장점이 있는 반면, 매핑 과정에서 Null 노드가 많아질 수 있고 노드의 깊이가 깊어질수록 노드 변화가 커진다는 단점이 있다[6]. (그림 2)는 K-ary 완전 트리 모델의 예이다.

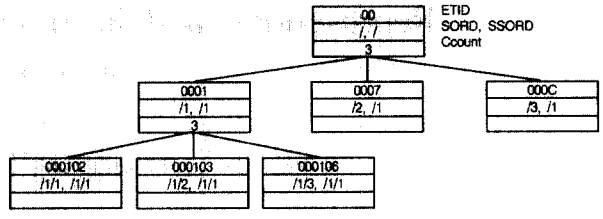


(a) 파싱 트리 (c) 노드값을 부여한 파싱 트리

(b) K-ary Tree(K=3)

(그림 2) K-ary(K=3) 트리 모델

ETID 모델은 (그림 3)과 같이 엘리먼트들 간의 계층 정보와 동일 부모 엘리먼트를 갖는 자식 엘리먼트들의 순서 정보, 그리고 동일한 부모 엘리먼트를 갖는 자식들 중 동일한 타입의 엘리먼트들에 대한 순서 정보를 통해 구조 문서를 표현한다. 이 방법은 기준 엘리먼트로부터 특정 엘리먼트에 대한 계층 정보와 순서 정보를 간단한 문자열 조작만으로 쉽게 구할 수 있다는 장점이 있는데 반해 트리의 깊이가 깊어질수록 각 노드를 표현하기 위한 공간이 무한대로 늘어난다는 단점이 있다[9, 10].



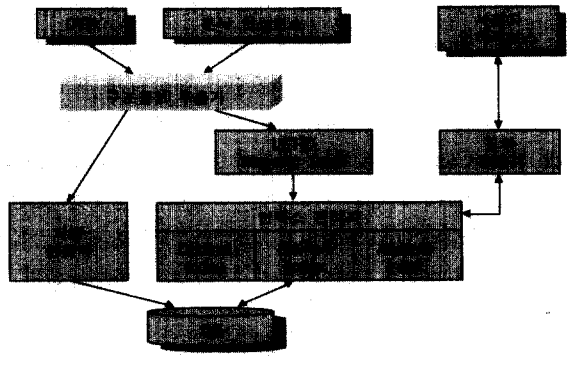
(그림 3) ETID 모델

따라서 본 논문에서는 고정된 크기의 노드 정보를 통해 엘리먼트 간의 계층 정보뿐만 아니라, 형제 노드의 순서를 표현함으로써 임의의 엘리먼트로부터 간단한 검색 과정을 통해 특정 엘리먼트에 쉽게 접근할 수 있도록 하고자 한다.

3. 시스템 구성 및 구조 정보 표현

3.1 구조 정보 검색 시스템

XML 문서의 구조 정보를 검색하기 위해 먼저 구조 정보 추출기를 통해 DTD로부터 엘리먼트 타입을 추출하고 엘리먼트 이름과 엘리먼트 ID를 연결시켜 주는 LETID(Leveled Element Type ID) Mapping Table을 구성한다[11]. 인덱스 관리자는 LETID 정보를 통해 키워드, 엘리먼트, 그리고 애트



(그림 4) XML 문서 검색 시스템 구성도

리뷰트에 대한 색인 테이블을 구성하고, 저장 관리자를 통해 XML 문서 인스턴스를 저장한다. 구조 정보를 검색하기 위해서는 사용자 인터페이스를 통해 질의를 하게되고 질의 처리기는 질의 내용을 분석하여 인덱스 관리자에게 분석 정보를 전달한다. 인덱스 관리자는 질의에 해당하는 색인 테이블을 통해 데이터베이스에 저장된 문서 인스턴스로부터 해당 정보를 검색하여 질의 처리기에게 전달한다. 질의 처리기는 검색된 정보를 다시 사용자 인터페이스를 통해 보여주게 된다. (그림 4)는 이와 같은 검색 시스템의 구성을 나타낸 것이다.

3.2 구조 정보 표현

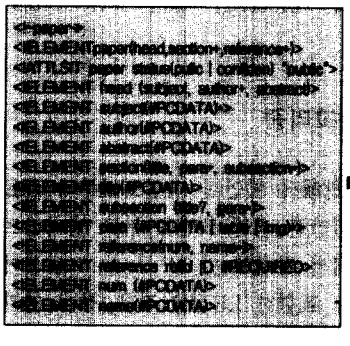
XML 문서는 내용 정보 뿐만 아니라 구조 정보를 함께 표현한다. 따라서 XML 문서의 부모 자식 관계뿐만 아니라 해당 엘리먼트에 대한 형제 관계, 그리고 동일한 타입의 엘리먼트에 대한 순서 정보를 표현하기 위해 LETID를 이용한 구조 정보 표현을 제안한다. 이 방법을 통해 사용자는 간단한 연산으로 질의에 대한 결과를 쉽게 검색할 수 있으며 고정된 크기로 LETID를 표현하기 때문에 문서에 대한 계층 구조의 깊이에 상관없이 엘리먼트 정보를 표현하기 위해 추가적인 공간이 필요 없다.

3.2.1 LETID

LETID는 DTD에서 정의된 각 엘리먼트에 대한 고유값을 나타낸다. 엘리먼트에 고유 ID를 부여하는 방식으로 DTD의 논리적 구조를 분석할 때 부모, 형제 노드를 직접적으로 찾을 수 있고 ID 값에 깊이 정보가 포함되어 있기 때문에 고유 번호만으로 깊이를 알수 있다.

LETID는 모든 엘리먼트의 노드를 고정된 8바이트로 표현하는데 각 바이트는 ASCII code 순서를 따르는 '0'→'9'→'A'→'Z'→'a'→'z' 순으로 구성된다. (그림 5)는 DTD에서 정의된 각각의 엘리먼트에 대해 LETID를 부여하고 이를 LETID 매핑 테이블로 표현한 것이다.

Paper DTD



LETID mapping Table

Element Type	LETID
paper	00000100
head	01000211
section	01000221
reference	01000231
subject	02110311
author	02110321
abstract	02110331
title	02210311
para	02210321
subsection	02210331
num	02310311
name	02310321

(그림 5) LETID 부여 방법

3.2.2 구조 정보 표현 방법

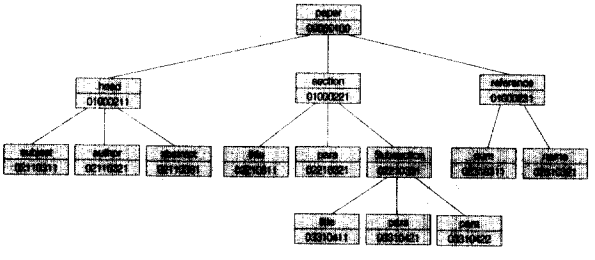
XML 문서의 구조 정보 표현 방법은 엘리먼트를 기반으로 표현하므로 특정 엘리먼트를 구별하고 엘리먼트의 위치 정보를 알 수 있도록 계층 정보로 표현한다. LETID의 각 자리가 의미하는 표현 정보는 <표 1>과 같다.

<표 1> LETID 표현 정보

자리수	표 현 정 보
1,2	부모 노드의 계층 정보
3,4	3 : 부모노드의 형제 노드에 대한 순서 정보 4 : 동일한 형제 노드에 대한 순서 정보
5,6	현재 노드의 계층 정보
7,8	7 : 현재 노드의 형제 노드에 대한 순서 정보 8 : 동일한 형제 노드에 대한 순서 정보

각각 2바이트로 엘리먼트의 부모, 자식간의 계층 정보와 형제간의 순서 정보를 나타낸다. 상위 4바이트(1234)는 부모 노드의 계층 정보와 부모 노드의 형제 노드에 대한 순서 정보를 유지한다. 이중 하위 2바이트(34)는 형제 엘리먼트들의 순서 정보와 동일한 타입의 형제 엘리먼트들 간의 순서 정보를 나타낸다. 마찬가지로 뒤의 4바이트(5678)는 현재 노드의 계층 정보와 현재 노드의 형제 노드에 대한 순서 정보를 표현한다. 각 엘리먼트들에 유일한 LETID 값을 부여하기 때문에 구조적 검색을 지원할 수 있고 반복적인 엘리먼트의 경우 서로 다른 LETID값을 부여하므로 다양한 구조 검색이 가능하다.

```
<?XML version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE paper SYSTEM "paper.dtd">
<paper status = "public">
  <head>
      <subject>A Dynamic Signature File Declustering
              Method</subject>
      <author>K.H.K</author>
      <abstract>For processing signature file</abstract><
  </head>
  <section>
      <title>1.Introduction</title>
      <para>Information</para>
      <subsection>
          <title>2.1 partitioning methods</title>
          <para>This section is</para>
          <para>This means. </para>
      </subsection>
    <reference refid = ref1 >
        <num>[1]</num>
        <name>Dan</name>
    </reference>
</paper>
```



(그림 6) XML 문서의 구조 정보 표현 트리 구조

(그림 6)은 위의 XML 문서의 구조 정보를 LETID를 이용하여 트리로 표현한 예이다. 위 트리는 엘리먼트들 간의

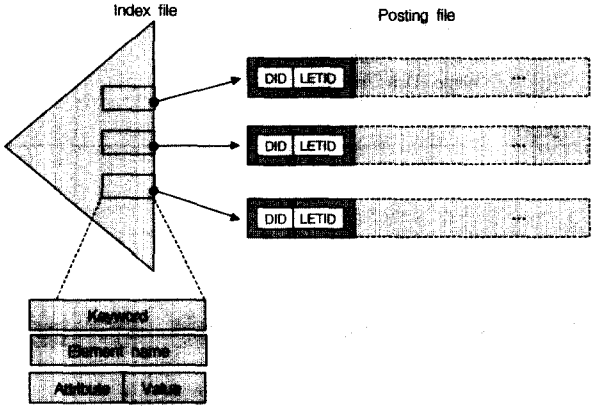
계층 정보를 나타내 주는 구조로서 LETID값은 문서의 논리적인 구조상의 각각의 엘리먼트에 부여되는 고유의 값이다.

계층 정보표현 방법을 보면, 예를 들어 (그림 6)의 구조 정보 표현에서 루트 엘리먼트인 paper의 LETID는 "00000100"이고, head의 LETID는 "01000211"이다. (그림 5)에서 LETID를 표현한 방법을 보면 paper의 뒤 4바이트(5678)가 head에 상속되어 head의 앞 4바이트(1234)가 "0100"이 된다. 이렇게 각각의 엘리먼트들은 부모의 정보를 상속받아 XML 문서의 접근이 필요 없이 LETID값만을 가지고 엘리먼트들 간의 계층 정보를 검색한다. 또한 XML 문서에서는 발생 지시자에 의한 반복적인 엘리먼트 사용이 가능하기 때문에 동일한 엘리먼트들이 반복적으로 나타난다. 따라서 이들의 표현 방법은 8바이트 중에"34", "78"번째 바이트에 형제 노드에 대한 순서 정보를 표현하여 구별한다.

(그림 6)에서 subsection의 자식노드로 para가 두 번 반복된다. para의 값은 "03310421", "03310422"로 각각 표현된다. 상위 4자리 "0331"은 "subsection"의 하위 4자리(5678) 값이 상속된 것이다. 이때 "04"은 현재 노드의 계층 정보를 표현해주고, "21"과 "22"는 현재 노드의 형제 노드에 대한 순서 정보를 나타내는데 8번째 바이트를 "1", "2"로 표현하여 각 형제 노드의 순서 정보를 표현한다. 결과적으로 LETID값은 모든 엘리먼트에 대해서 서로 다른 값을 갖고, LETID만으로 특정 엘리먼트를 바로 검색할 수 있다.

4. XML 색인 구성 및 질의 처리

XML 문서의 특성을 고려하여 다양한 질의를 효율적으로 처리할 수 있는 색인 구조를 설계한다. 색인은 구조 정보 추출기로부터 추출한 구조 정보를 이용하여 내용 색인, 구조 색인, 애트리뷰트 색인의 3가지로 구성되며 기존의 역파일 방식을 사용한다. 색인 구조는 (그림 7)과 같다.



(그림 7) 색인 구조

4.1 내용 색인

내용 색인은 내용 위주의 검색을 수행할 수 있도록 도와주는 색인으로 각 키워드에 대한 인덱스 파일을 구성하고 포스팅 파일은 문서 인스턴스와 매핑 할 수 있도록 DID와 LETID로 구성한다.

4.2 구조 색인

구조 색인은 구조 단위가 되는 엘리먼트가 중심이 되며 구조 색인의 포스팅 파일은 개개의 엘리먼트와 매핑할 수 있는 정보인 DID, LETID로 구성된다. 또한 엘리먼트 간의 계층관계 검색, 형제관계 검색을 지원하며, LETID의 뒤 4자리를 이용하여 구조적 검색을 수행한다.

4.3 애트리뷰트 색인

애트리뷰트 색인은 애트리뷰트 이름과 값으로 인덱스가 구성되고, 해당 애트리뷰트가 포함되어 있는 엘리먼트와 매핑할 수 있도록 DID, LETID로 구성된 포스팅 파일이 존재한다.

4.4 질의 처리

XML 문서에 대한 검색은 내용 검색, 구조 검색, 애트리뷰트 검색으로 나눌 수가 있다. 내용 검색은 해당 Keyword를 갖는 문서의 DID와 LETID를 이용하여 저장 관리자를 통해 문서의 내용을 가져와서 사용자 인터페이스로 넘겨준다. 구조 검색은 기준 엘리먼트를 식별할 수 있는 DID와 LETID를 구하고, 기준 엘리먼트와의 관계에 의한 연산과 이를 통한 목적 엘리먼트를 찾아 엘리먼트의 내용을 전달받아 넘겨준다. 애트리뷰트 검색은 애트리뷰트 명과 값을 이용하여 애트리뷰트 인덱스를 통해 해당 애트리뷰트가 정의되어 있는 엘리먼트를 식별할 수 있는 DID, LETID, Attribute Value를 구한다.

한 예로 (그림 6)에 있는 title의 2단계 상위의 부모 노드를 검색 할 때 title의 LETID값은 "03310411"로 level이 3인 것을 알 수 있다. 검색시 앞 네자리 "0331"을 뒤 자리로 갖는 노드들을 검색하면 subsection을 찾을 수 있다. 찾아진 subsection의 LETID값은 "02210331"이며 다시 "0221"을 뒤 자리로 갖는 노드를 찾으면 "01000221" 값을 갖는 section을 찾을 수 있다.

5. 구조 정보 검색 알고리즘

구조 정보를 검색하기 위해서 제안된 색인으로부터 DID와 LETID 값을 구한다. 제안한 색인 모델은 부모 엘리먼트의 정보를 자식 엘리먼트에서 가지고 있으므로 구조 검색이 용이하며, 엘리먼트 정보를 표현한 LETID 값은 형제 엘리먼트들 간의 순서 정보까지 포함하고 있으므로 임의의 엘리먼트로부터 특정 엘리먼트를 검색하기 용이하다. 따라서 구조 정보 검색 알고리즘에서는 LETID값의 각 자리에 해당되는 정보를 이용하여 해당 엘리먼트를 구할 수 있다. 다음은 제안한 색인 구조를 이용한 구조 검색 알고리즘이다.

5.1 부모 검색

사용자 인터페이스로부터 전달받은 해당 엘리먼트 정보를 LETID 매핑 테이블로부터 구한다. 이때, LETID 정보와 N 단계 상위 부모를 나타내는 레벨 정보를 키 값으로 하여 색인 테이블로부터 해당 부모 노드를 검색한다. 즉, 전달받은 LETID 값의 상위 4자리 값을 하위 4자리 값으로 갖는 엘리먼트를 검색한다. 이때, N 단계 상위 부모를 검색하기 위해서는 전달받은 레벨 정보만큼 위의 단계를 반복하여 해당 엘리먼트를 구할 수 있다. (그림 8)은 특정 엘리먼트의 부모 엘리먼트를 검색하는 알고리즘이다.

```
search_parent_element(letid, level) {
    curren_first_letid = letid의 앞 네자리 값 ;
    if (level == 0)          {return letid}
    else {
        parent_etid = get_element(element_table[i]) ;
        parent_level = parent_letid의 앞 두자리 ;
        while(level < parent_level) {
            parent_last_letid = parent_letid의 뒤 4자리 값 ;
            parent_level = parent_letid의 앞 두자리 ;
            if (current_first_letid == parent_last_letid) {
                level-- ;
                search_parent_element(parent_letid, level) ;
            }
        parent_letid = get_element(element_table[i]) ;
        }
    }
}
```

(그림 8) 부모 엘리먼트 검색 알고리즘

5.2 자식 검색

질의 분석기를 통해 사용자 인터페이스로부터 전달받은 엘리먼트 정보와 자식 엘리먼트의 순서 정보를 키 값으로 하여 색인 구조로부터 해당 엘리먼트를 검색한다. 이때, 자식 엘리먼트의 순서 정보는 LETID의 7번째 자리 값으로 알 수 있다. 즉, 전달받은 엘리먼트의 LETID 값으로부터 하위 4자리 값을 상위 4자리 값으로 갖는 엘리먼트를 구한 다음, 이 엘리먼트의 LETID 정보로부터 7번째 자리 값과 전달받은 순서 정보를 비교하여 자식 엘리먼트를 검색한다. (그림 9)는 자식 엘리먼트를 검색하는 알고리즘이다.

```
search_child_element(letid, order) {
        curren_last_letid = letid의 뒤 네자리 값;
        child_letid = get_element(element_table[i]);
                  //해당 엘리먼트의 자식 엘리먼트들을 가져온다.
        child_first_letid = child_letid의 앞 4자리 값;
        while(letid의 수) {
            if(current_last_letid == child_first_letid &&
               order == child_order_letid) return child_letid;
            else {
                child_letid = get_element(element_table[i]);
                child_first_letid = child_letid의 앞 4자리 값;
            }
        }
}
```

(그림 9) 자식 엘리먼트 검색 알고리즘

5.3 평가

XML 문서 검색을 위해 제안한 색인 모델의 비교 평가를 위해 기존 K-ary 완전 트리구조 방법과 ETID에 의한 색인 모델을 비교 대상으로 <표 2>와 같이 비교하였다. 비교 항목으로는 구조정보 표현방법, 디스크 검색 회수, 그리고 각 엘리먼트 정보 크기로 하였다. 검색 성능 평가를 위해 디스크 접근 회수를 측정할 경우, K-ary 완전 트리 구조 방법은 K값을 구하기 위해 각 색인 파일마다 구조 색인에 접근한다. 또한 문서 파싱을 통해구한 형제나 자식 엘리먼트가 실제 문서에 존재하는 노드인지 가상 노드인지를 판별하기 위해 검색 결과 수만큼 다시 구조 색인에 접근하여야 한다. 그러나 제안한 색인 모델의 경우 LETID로 부모, 자식, 형제 노드의 정보를 모두 표현하므로 가상 노드 여부를 판별하기 위한 접근 시간이 줄어든다. (그림 10)은 부모와 자식에 대한 구조 검색시, 엘리먼트 수의 증가에 따른 디스크 접근 회수의 변화를 측정한 결과이다.

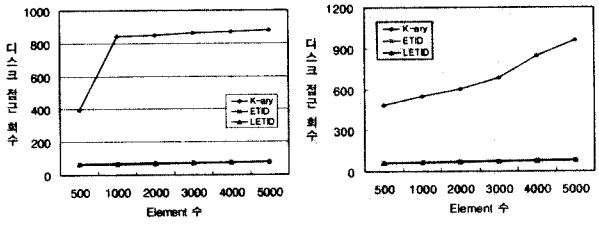
또한 노드 정보 저장 공간을 비교하면 ETID에 의한 색

〈표 2〉 각 방법의 비교

	K-ary	ETID	LETID
구조 정보 표현 방법	K-ary 완전 트리 방식으로 기존 트리를 완전 트리로 변환하여 노드 값을 구함	ETID, SORD, SSORD를 이용하여 엘리먼트 정보 표현	고정 크기의 LETID 만으로 엘리먼트 정보 표현
디스크 검색 회수	$I_n+R_n*P_n+S_n*I_n$ (*부모검색*) $I_n+R_n*P_n+S_n*I_n+R_n$ (*부모검색*)	$2I_n+R_n*P_n$ (*부모/자식 검색*)	$I_n+R_n*P_n$ (*부모/자식 검색*)
엘리먼트 정보 크기	2바이트(정수형으로 노드 순서 표현)	ETID(depth * 2바이트) SORD, SSORD(depth * 1바이트)	LETID(8바이트 고정 크기)
장 점	간단한 수식을 통해 부모/자식 관계의 엘리먼트를 구함	부모/자식/동일한 타입의 형제 엘리먼트의 순서 정보를 구할 수 있음	부모/자식/동일한 타입의 형제 엘리먼트의 순서 정보를 구할 수 있음
단 점	노드의 깊이에 따른 노드변화가 크고, 사용하지 않는 노드가 많아지므로 데이터 양이 커짐	노드의 깊이가 깊어질수록 엘리먼트 정보를 표현하는 ETID의 크기가 계속적으로 증가함	2단계 이상의 부모 노드 검색 시 재귀함수에 의한 추가 연산 필요.

I_n : 각 색인 파일 접근 회수 R_n : 검색 결과 P_n : 포스팅 엔트리 크기 S_n : 구조 색인 접근 회수 *SORD(Sibling ORDer)* *SSORD(Same Sibling ORDer)*

인 모델의 경우 노드의 깊이가 깊어질수록 계속적인 추가 공간이 필요한 반면 제안한 모델의 경우 고정된 공간을 사용하므로 추가적인 공간이 필요 없다.



(a) 부모 검색 (b) 자식 검색

(그림 10) 구조 검색

6. 결 론

이 논문에서는 XML 문서의 구조 정보를 검색하기 위해 구조 정보를 표현하는 방법과 구조 검색을 지원하기 위해 색인 구조 및 검색 알고리즘을 제안하였다. 구조 정보 표현 방법은 DTD에 나타나는 각 엘리먼트들에 대해 LETID를 부여하여 고유값을 주었다. LETID는 고정된 크기로 구성이 되며 각 자리에 부모의 계층 정보와 형제 노드의 순서 정보를 알 수 있도록 번호를 부여하였다. 제안한 색인 구조는 내용 검색을 지원하는 내용 색인, 구조 검색을 지원하는 구조 색인, 애트리뷰트 검색을 지원하는 애트리뷰트 색인으로 구성된다. 또한 제안한 색인 구조를 기반으로 한 구조 검색 알고리즘은 바로 위의 부모 엘리먼트 뿐만 아니라 N 단계 상위의 부모 엘리먼트와 N 번째 자식 엘리먼트를 구할 수 있다. 이와 같은 구조정보 표현과 색인을 이용하여 특정 엘리먼트에 직접적인 접근이 가능하고, 구조화 된 문서를 효율적으로 관리할 수 있으며, 다양한 질의처리가 가능하다. 따라서 보다 효율적이고 빠른 검색을 지원할 수 있다. 향후 연구과제로는 제안한 방법을 기반으로 한 구조 문서 검색 시스템의 구현과 문서의 갱신 시 색인 모델을 적용하기 위한 연구가 필요하다.

참 고 문 헌

[1] Brian Lowe, Justin Zobel, Ron Sacks-Davis "A Formal Model for Databases of Structured Text," Proceedings of the Fourth International Conference on Database Systems for Advanced Applications (DASFAA '95), pp.449-456, 1995.

[2] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Extensible Markup Language(XML)1.0, REC-xml-1998.

[3] W3C, "Extensible Markup Language(XML) 1.0," http : // www.w3c.org/TR/1998/REC-xml-1998210.html, 1998.

[4] Toung Dao "An Indexing Model for Structured Documents to Support Queries on Content, Structure and Attributes," Proceedings of ADL'98, pp,88-97, 1998.

[5] V. Christophides. et al, "From Structured Documents to Novel Query Facilities," ACM SIGMOD, pp.313-324, Minesota, USA, 1994.

[6] Sung-Geun Han, Jeong-Han Son, Jae-Woo Chang Zong-Cheol Zhoo, "Design and Implementation of a Structured Information Retrieval System for SGML Documents," IEEE, pp.81-88, 1999.

[7] T. Dao, R. Sacks-Davis and J. A. Thom "An indexing scheme for structured documents and its implementation," In Proceedings of the 5th International Conference on Database Systems for Advanced Applications, pp.125-134, Melbourne, Australia, April. 1997.

[8] 이종설 외 7, "구조 정보 검색을 위한 XML 저장관리시스템 설계 및 구현".

[9] 박종관, 강형일, 손충범, 유재수 "XML 문서에 대한 효율적인 구조 기반 검색을 위한 색인 모델," 2000 추계학술발표논문집, 한국정보과학회, pp,18-20, 2000.

[10] 박종관, " XML 문서에 대한 효율적인 구조 기반 검색을 위한 색인 모델", 충북대학교 석사학위논문, 2001.

[10] 고혜경, 조윤기, 조정길, 이병렬, 구연설, "효율적인 구조 정보 검색을 위한 색인 모델", 2001 춘계학술발표논문집(A), 한국정보과학회, pp,649-651, 2001.

조 윤 기

e-mail : ykcho@selab.chungbuk.ac.kr

1994년 충북대학교 컴퓨터과학과 졸업(학사)

1996년 충북대학교 대학원 전자계산학과 졸업(이학석사)

1998년~현재 충북대학교 대학원 전자계산학과 박사과정 수료

관심분야 : 저장관리시스템, 정보검색, 소프트웨어 테스팅

조 정 길

e-mail : cho0530@chollian.net

1987년 숭실대학교 정보과학대학 졸업(학사)

1993년 숭실대학교 정보과학대학원 졸업 (이학석사)

2000년~현재 충북대학교 대학원 전자계산학과 박사과정

관심분야 : 객체지향 자료 모델링, XML 문서관리, 질의처리, 정보검색

이 병 렬

e-mail : inanet@splinux.co.kr

1988년 서울 시립대학교 경상대학 졸업(학사)

1994년 숭실대학교 정보과학대학원 졸업 (이학석사)

2000년~현재 충북대학교 대학원 전자계산학과 박사과정

관심분야 : XML, Linux 시스템 프로그래밍

구 연 설

e-mail : yskoo@cbucc.chungbuk.ac.kr

1964년 청주대학교 졸업(학사)

1981년 동국대학교 대학원 졸업(이학석사)

1988년 광운대학교 대학원 졸업(이학박사)

1979년~현재 충북대학교 컴퓨터과학과 교수

관심분야 : 객체지향 모델링, 테스팅, 정보검색 등