XML 문서를 위한 구조정보 추출기의 설계 및 구현

민영수*, 강승헌*, 강형일*, 유재수*, 이하욱**, 최한석***
*충북대학교 정보통신공학과
**(주) 한국지식웨어
***목포대학교 정보공학부

Design and Implementation of a Structure Information Extractor for XML Documents

Young Su Min*, Seung Heon Kang*, Hyung Il Kang*, Jae Soo Yoo*,
Ha Wook Lee**, Han Suk Choi***
*Dept. of Computer & Communication Eng. Chungbuk National University
**Korea Knowledge Ware inc.
***Division of Information Eng. Mokpo National University

요 약

XML 문서의 구조검색을 위한 기존 구조정보 표현방법들은 특정 엘리먼트의 조상, 자손, 형제에 대한 구조검색을 효율적으로 지원하지 못한다. 본 논문에서는 XML 문서의 효율적인 관리와 구조검색을 위해 DTD(Document Type Definition)의 논리적 구조를 따르는 XML 문서의 구조정보 표현을 고안하고 구조정보 추출기를 설계하고 구현한다. 이를 통하여 특정 엘리먼트에 직접적인 접근이 가능하도록 하고, 다양한 구조적 질의를 효과적으로 처리할 수 있도록 한다.

1. 서론

XML은 HTML보다 사용자의 다양한 요구를 충분히 수용할 수 있고 SGML보다 사용하기 쉽다는 장점으로 인해서 웹 문서뿐만 아니라 전자도서관, EDI(Electronic Data Interchange), CALS(Commerce At the Light Speed) 등을 포함한 다양한 분야에서 XML을 활용하고자 폭 넓은 연구를 하고 있으며 최근 구체화되는 응용이 부쩍 많아지고 있다[7].

XML은 HTML이 하나의 고정된 DTD를 사용하는 것과는 달리 논리적 구조를 나타내는 여러 DTD를 사용할 수 있다는 점에서 SGML과 같다. DTD의 논리적 구조를 따르는 XML 문서의 구조정보는 데이터베이스에 저장된 XML 문서의 효율적인 관리나 구조 검색을 수행하는데 이용된다[4].

이러한 논리적 구조를 수용하는 기존의 구조정보 표현 연구들은 XML 문서를 가지고 구조정보를 표현하려고 했다. 일부 XML 문서 위주의 구조정보 표현 방법은 문서마다 특정 엘리먼트에 부여된 ID가 다를 수 있다[2,8]. 또한, 특정 엘리먼트에 대한 직접적인 접근이 불가능하고 조상, 자손, 형제의 관계에 있는 엘리먼트를 접근하기 위해 복잡한 연산을 수행해야 했다[2,3,5,6,9,10]. 이에 본 논문에서는 특정 엘리먼트에 대한 직접적인 접근이 가능하고 엘리먼트 간의 관계를 구하기 위해 복잡한 연산이 필요 없도록 DTD의 논리적 구조정보를 사용해서 XML 문서를 효율적으로 관리할 수 있는 구조정보 표현을 고안하고, XML 문서에서 이러한 구조정보 표현을 가지는 구조정보를 추출해내는 구조정보 추출기를 구현하였다.

본 논문의 구성은 2장에서 기존의 XML 문서에 대한 구조정보 표현의 문제점들을 지적하고 3장에서 효율적인 구조정보 표현의 제안과 구조정보 추출기를 설계한다. 4장에서는 구현한 구조정보 추출기에 대해 설명한다. 마지막으로 결론 및 향후 연구 방향을 기술한다.

2. 관련 연구

구조정보를 표현하기 위한 ID 할당 방법은 기존에 많은 방법이 제시되었다. 일반적으로 자신의 노드 ID는 부모의 노드 ID에 자신의 순서정보(부모의 몇 번째 자식)를 붙여서 노드 ID를 생성하는 것이다. 엘리먼트 타입 ID에 의해 계층정보를 나타내고 엘리먼트 순차정보로 순서를 나타낸다. 이러한 구조정보 표현은 특정 엘리먼트의 조상, 자손, 형제에 대한 구조검색을 효과적으로 수행하기 어렵다[3,6,10]. 그 이유는 검색하고자하는 엘리먼트 ID를 만든 후, 그 ID를 갖는 모든 엘리먼트를 찾고 순서정보를 비교해야 한다. 또한 경우에 따라서는 부모 엘리먼트의 순서정보를 참조해야 한다. 이것은 불필요한 검색과 복잡한 비교연산을 요구한다[1,5]. 조상검색의 경우 특정 조상 엘리먼트의 ID를 갖는 모든 엘리먼트를 구한 후 순서정보를 비교하여 찾기 때문에 불필요한 검색과 추가적인 비교연산이 요구된다. 엘리먼트의 순차정보를 기반으로 하는 경로 엘리먼트 ID를 이용하여 구조검색을 지원하는 경우, 출현순서에 의해 ID를 부여하기 때문에 다른 문서들에서뿐만 아니라 한 문서 내에서도 동일한 타입의 엘리먼트에 다른 ID를 갖게 되므로 특정 엘리먼트를 검색하는데 어려움이 있다[2,8].

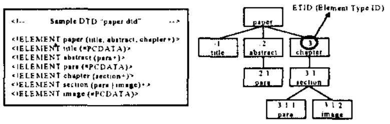
3. 구조정보 표현

본 논문에서 제안하는 구조정보는 부모와 자식 엘리먼트간의 계층정보, 형제 엘리먼트의 순서정보, 그리고 같은 엘리먼트 형에 대한 순서정보로 표현된다. 이들은 모두 부모 엘리먼트의 정보를 유지한다. 그러므로 기존 엘리먼트로부터 특정 엘리먼트에 대한 계층정보와 순서정보를 간단한 문자열 조작만으로 쉽게 구할 수 있다. 이렇게 구한 순서정보는 각 엘리먼트에 유일하게 할당된 값이기 때문에 직접 특정 엘리먼트를 접근할 수 있다. 또한 형제 엘리먼트와 같은 엘리먼트 형에 대한 두 종류의 순서정보를 이용함으로써 "세 번째 자식 엘리먼트를 찾아라", "Chapter의 자식 엘리먼트 중 세 번째 section을 찾아라"와 같은 보다 다양한 구조검색 질의를 처리할 수 있다.

3.1 계층정보

DTD의 논리적 구조를 따르는 XML 문서의 구조정보는 엘리먼트를 기반으로 한다. 이러한 XML 문서상의 엘리먼트를 유일하게 구별하면서 엘리먼트간의 계층정보를 표현하기 위해서 일반적으로 ID를 부여한다. 본 논문에서는 특정 엘리먼트를 구별하면서 엘리먼트간의 계층정보를 표현할 수 있는 ETID(Element Type ID)를 부여한다. ETID는 DTD의 논리적 구조를 분석한 후 각 엘리먼트 타입에 부여되는 유일한 값이다. 이 ETID를 부여하는 방법은 UNIX 파일시스템에서 디렉토리를 표현하는 방법을 사용한다. 즉, root 엘리먼트는 '/'로 표현되며 root 엘리먼트에 포함되는 엘리먼트는 '/1'나 '/2'와 같이

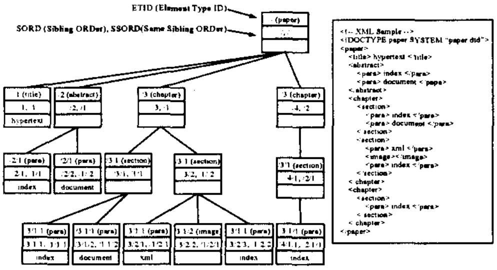
표현된다. 엘리먼트 이름과 ETID와의 사상을 위해 매핑 테이블이 존재하게 된다. 이 매핑 테이블을 참조하면 엘리먼트간의 조상, 형제, 자식이 어떤 형의 엘리먼트인지 쉽게 알 수 있다. (그림 1)에서 **image** 엘리먼트의 부모 엘리먼트는 **image** 엘리먼트의 ETID '/3/1/2'에서 부모 엘리먼트의 ETID '/3/1'을 구하고 매핑 테이블에서 ETID가 '/3/1'인 엘리먼트를 찾으면 **section**이라는 것을 알 수 있다.



(그림 1) ETID 할당의 예

3.2 순서정보

XML 문서에서는 DTD에 나타난 발생지시자에 의한 반복적인 엘리먼트의 사용이 가능하다. 본 논문에서는 다양한 구조검색을 지원하기 위해서 두 종류의 순서정보를 제안한다. 형제 엘리먼트의 발생순서를 나타내는 SORD(Sibling ORDer)와 동일한 타입의 형제 엘리먼트간의 순서정보를 나타내는 SSORD(Same Sibling ORDer)가 그것이다. SORD는 XML 문서에서 유일하게 구분되는 값으로 특정 엘리먼트를 검색하는데 유용하게 이용된다. 이들의 표현방법은 ETID 표현방법과 동일하여 부모 엘리먼트의 순서정보를 유지한다. 이러한 순서정보는 "A 엘리먼트의 3번째 자식 엘리먼트가 B인 문서를 찾아라", "A 엘리먼트의 3번째 B 엘리먼트에 X라는 단어가 있는 문서를 찾아라"와 같은 질의 처리에 효과적이다. 따라서 특정 엘리먼트는 ETID, SORD, SSORD를 이용하면 쉽게 검색할 수 있다. (그림 2)는 XML 문서의 구조정보를 ETID, SORD, SSORD를 사용해서 트리 형태로 표현한 예이다. 예로 SORD가 '/3'인 **chapter**의 두 번째 자식 엘리먼트인 **section**은 **para**라는 자식 엘리먼트를 두 개 갖는다. 두 **para**의 ETID는 '/3/1/1'로 동일하지만 SORD는 각각 '/3/2/1'과 '/3/2/3'이고 SSORD는 각각 '/1/2/1'과 '/1/2/2'이다.

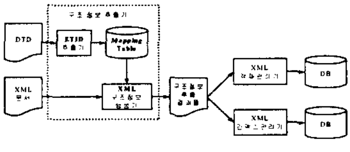


(그림 2) XML 문서의 구조정보 표현 예

4. XML 구조정보 추출기 설계 및 구현

4.1 시스템 구성도

XML 문서의 구조정보는 ETID 추출기와 XML 구조정보 생성기로 이루어지는 XML 구조정보 추출 모듈에 의해서 얻어진다. (그림 3)은 구조정보 추출기의 시스템 구성도를 나타낸다. ETID 추출기는 DTD로부터 엘리먼트 형(Type)를 추출하고 엘리먼트 이름과 ETID를 연결시켜주는 매핑 테이블을 구성한다. 그리고 XML 구조정보 생성기는 매핑 테이블을 참조하여 XML 문서를 분석하고 문서의 구조정보를 추출한다. 이렇게 추출된 문서의 구조정보는 XML 객체 관리기와 XML 인덱스 관리기에 의해 사용되어질 수 있다.



(그림 3) 구조정보 추출기 시스템 구성도

4.2 ETID 추출기

DTD가 XML 문서와 동일 파일에 존재하는 경우와 없는 경우는 DTD를 공유할 수 없으며 같은 형태의 문서 작성에 부적합하다. 독립된 파일로 존재하는 DTD를 공유해서 같은 형태의 여러 XML 문서를 작성하는 것이 가장 일반적이며 효과적일 것이다. 따라서, 본 논문의 구현은 독립된 파일로 존재하는 DTD를 대상으로 한다. 구조정보 추출기가 XML 문서 내에 있는 각각의 엘리먼트에 올바른 ETID를 줄 수 있도록 ETID 추출기는 DTD에 선언되어 있는 엘리먼트를 가지고 계층적인 구조를 나타내기 위한 트리를 생성하고 트리를 전위 순회하면서 엘리먼트 이름과 ETID로 구성된 매핑 테이블을 생성한다.

```
<!--                Sample DTD                -->
<!ELEMENT paper (head, section+, reference+)>
<!ELEMENT head (title, author+, abstract)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (name, department)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT department (#PCDATA)>
<!ELEMENT abstract (#PCDATA)>
<!ELEMENT section (num, title, para, section*)>
<!ELEMENT num (#PCDATA)>
<!ELEMENT para (#PCDATA | img)*>
<!ELEMENT img EMPTY>
<!ELEMENT reference (#PCDATA)>
```

(그림 4) 독립된 파일로 존재하는 DTD의 예

4.2.1 엘리먼트 트리 구성

각각의 엘리먼트 이름에 ETID를 부여하는 ETID 추출기에서는 엘리먼트 트리를 구성하기 위해 다음의 방법을 사용했다.

1) 실 엘리먼트 노드

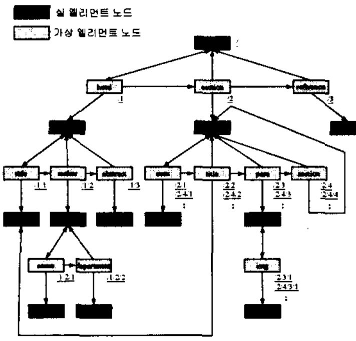
DTD의 ELEMENT 키워드 바로 옆에 있는 엘리먼트를 나타내는 노드이다. 예를 들어 <!ELEMENT paper (head, section+, reference+)>에서 paper 엘리먼트를 나타내는 노드이다. 가상 엘리먼트 노드들을 자식으로 가지며, 루트가 되는 실 엘리먼트 노드를 제외하고는 하나 이상의 가상 엘리먼트 노드가 참조한다.

2) 가상 엘리먼트 노드

DTD의 괄호 안에 나타나는 엘리먼트를 표현하는 노드이다. 예를 들어 <!ELEMENT paper (head, section+, reference+)>에서 head, section, reference 엘리먼트를 나타내는 노드이다. 실 엘리먼트 노드의 자식 노드들로 부모를 나타내는 실 엘리먼트 노드, 형제를 나타내는 가상 엘리먼트 노드, 그리고 자신을 나타내는 실 엘리먼트 노드를 가리키고 있는 요소를 포함한다.

4.2.2 ETID 부여

구성된 엘리먼트 트리를 전위 순회하면서 루트가 되는 실 엘리먼트 노드와 루트의 하위에 있는 가상 엘리먼트 노드들만 ETID를 부여한다. 이것은 엘리먼트 이름과 ETID로 구성되는 매핑 테이블을 생성하며 구조정보 생성기에서 사용한다. (그림 5)는 (그림 4)의 예제 DTD를 엘리먼트 트리로 구성하고 구성된 엘리먼트 트리의 루트 노드와 가상 엘리먼트 노드에 ETID를 부여한 상황을 보여준다.



(그림 5) 엘리먼트 트리 생성과 ETID 부여의 예

4.2.3 순환 처리

엘리먼트 트리에 따라 ETID를 부여하면서 가장 문제가 되는 것은 순환의 발생이다. 순환을 포함하고 있는 엘리먼트 트리의 경우 순환을 따라서 무한히 ETID를 부여한다. 이러한 순환 문제를 처리하기 위해서 ETID를 부여할 때 순환의 횟수에 제한을 줄 수 있어야 한다. 순환의 횟수에 제한을 줄 수 있는 방법으로 루트로부터 하나의 가지를 이루는 엘리먼트 노드들의 이름을 저장하면서 동일한 이름의 엘리먼트 노드가 몇 번이나 나오는지 검사해서 (제한 횟수 + 2)를 넘어가면 해당 엘리먼트 노드의 자식 엘리먼트 노드가 없는 것처럼 취급해서 순회를 진행하면 된다. (그림 5)에서 **section** 노드는 자기 자신

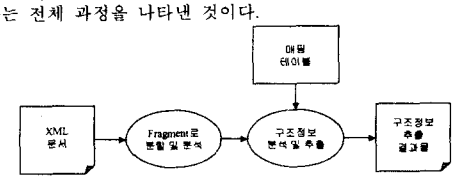
인 **section**을 자식으로 갖는다. **section** 노드에 ETID를 부여하다 보면 ETID의 부여가 끝나지 않는 순환을 포함한다. 만일 순환 횟수를 2로 주었을 경우, 루트로부터 한 가지인 paper -> section -> section 부분을 보면 ETID가 부여되는 노드는 **section**이 4번 나오는 papaer -> section -> section -> section -> section이다. 각각의 가지에 대해서 이런 방법을 사용해서 순환 문제를 해결했다.

4.3 구조정보 생성기

실제 XML 문서에서는 ETID로 표현되는 계층정보뿐만 아니라 각 엘리먼트가 발생하는 순서도 중요한 구조정보이다. 이러한 순서정보는 이미 설명한 바와 같이 SORD와 SSORD로 표현된다. 구조정보 생성기는 XML 문서를 분석하여 엘리먼트 단위로 분할하고 각 엘리먼트에 계층정보인 ETID와 순서정보인 SORD, SSORD를 부여한다. 이때 ETID는 ETID 추출기에서 생성한 매핑 테이블을 참조하여 부여하고 SORD와 SSORD는 엘리먼트들 사이의 관계를 분석하여 할당한다.

4.3.1 Fragment 분할 및 분석

구조정보 추출과정은 크게 두 모듈로 나뉜다. 우선 XML 문서를 스캔하여 Fragment라는 단위로 나누고 그것의 Type을 분석한다. 이때 Fragment Type은 PCDATA, COMMENT, CDATA, START TAG, END TAG, EMPTY TAG의 6가지이다. 다음으로 이 Fragment Type과 Fragment간의 관계를 고려하여 엘리먼트를 구성하고 ETID, SORD, SSORD를 부여한다. 다음 (그림 6)은 구조정보 추출하는 전체 과정을 나타낸 것이다.

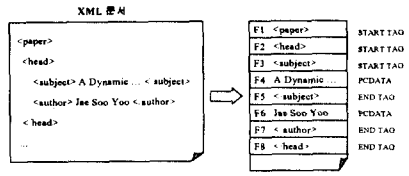


(그림 6) 구조정보 추출 과정

Fragment Type은 분할된 Fragment의 종류를 나타내며 엘리먼트를 구성하는 과정에서 참조한다. 다음은 Fragment Type에 대한 설명이다.

1) PCDATA : 엘리먼트내의 문서 내용
2) COMMENT : 주석으로 구조정보 추출할 때는 무시됨
3) CDATA : CDATA 섹션으로 "<![CDATA["로 시작하고 "]]>"로 끝남
4) START TAG : 시작 태그
5) END TAG : 끝 태그
6) EMPTY TAG : 아무런 내용도 갖지 않는 태그

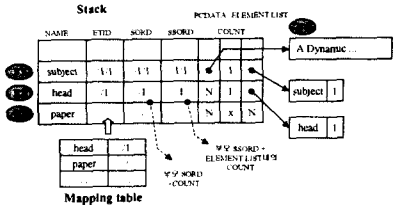
다음 (그림 7)은 XML 문서를 분석하여 Fragment로 분할하고 Fragment Type을 부여한 예이다.



(그림 7) Fragment Type 부여의 예

4.3.2 구조정보 분석 및 추출

본 논문에서는 구조정보를 추출하는 과정 즉, 엘리먼트에 ETID, SORD, SSORD를 부여하는 과정에 스택을 이용한다. Fragment가 START TAG일 경우 스택에 push하고, 매핑 테이블을 검색하여 ETID를 부여한다. 그리고 자신의 순서정보를 카운트하고 부모 엘리먼트의 SORD와 SSORD에 자신의 순서정보를 덧붙여 SORD와 SSORD를 생성한다. PCDATA일 경우 스택 top의 엘리먼트에 연결하여 이 PCDATA가 속하는 엘리먼트를 지정한다. END TAG일 경우 스택의 top 엘리먼트를 pop하여 한 엘리먼트의 구조정보를 완성한다. EMPTY TAG일 경우 push와 pop과정을 연속적으로 수행한다. CDATA일 경우 PCDATA의 경우와 동일하게 처리하고 COMMENT일 경우 무시한다. (그림 8)은 (그림 7)의 XML 문서에 대한 구조정보를 추출하는 과정을 나타낸 것이다.



(그림 8) 스택을 사용한 구조정보 추출 과정

4.3.3 구조정보 추출 결과물

구조정보 추출기에 의해 추출되는 정보는 다음 (그림 9)와 같은 필드를 갖는 텍스트 파일이다. ETID, SORD, SSORD는 이미 설명한 바와 같고 Tag name은 엘리먼트의 이름이다. Start offset와 End offset은 각각 XML 문서 화일에서 엘리먼트의 시작위치와 끝위치를 나타낸다. Content는 현재 엘리먼트에 속하는 내용이며 Content length는 이 내용의 길이를 나타낸다.

(그림 9) 구조정보 추출기에 의해 추출되는 정보

5. 결론

본 논문에서 XML 문서의 효과적인 관리와 검색을 위해 DTD 분석 후 각 엘리먼트 형에 할당되는 유일한 값인 ETID, XML 문서에서 형제 엘리먼트의 순서정보인 SORD, 그리고 같은 엘리먼트 형에 대한 순서정보인 SSORD를 생성하는 구조정보 표현 방법을 고안하고 DTD와 XML 문서를 통해 구조정보를 얻을 수 있는 구조정보 추출기를 설계하고 구현하였다. 이와 같은 구조정보 표현은 특정 엘리먼트에 대한 직접적인 접근이 가능하며, 다양한 구조적 질의를 효과적으로 처리할 수 있다. DTD에 선언되어 있는 각각의 엘리먼트에 올바른 ETID를 부여하기 위해서 실 엘리먼트 노드와 가상 엘리먼트 노드로 구성되는 엘리먼트 트리를 만들고 전위 순회 방법을 사용해 구현했다. XML 문서의 구조적 정보를 추출하기 위해 Fragment라는 단위로 나누고 스택을 사용하여 각 Fragment type에 따라 처리를 달리했다. 이러한 구조정보 표현의 설계와 구현으로 구조화된 문서관리 시스템 개발을 위한 기반을 마련하였다. 향후 연구 과제로, 보다 복잡한 문법의 SGML 문서의 구조정보 추출기를 개발하는 것과 순환의 횟수에 제한을 두는 것이 비효율적일 수도 있기 때문에 ETID 추출기와 구조정보 생성기의 결합을 통해서 이러한 문제를 해결하는 것이다.

참고 문헌

[1] 김용훈, "다양한 구조 검색을 지원하는 XML 문서 검색기의 설계 및 구현", 충남대학교 석사학위논문, 1998
[2] 손정한, 이희주, 장재우, 심부성, 주종철, "구조화된 문서를 위한 정보검색시스템의 설계 및 구현", '98 동계 데이터베이스 학술대회 논문집 제 14권 1호, pp. 102-106, 1998
[3] 연제원, 조정수, 이강찬, 이규철, "XML 문서 구조검색을 위한 저장 시스템 설계", 한국정보과학회 학술 발표 논문집(B), 제 26권 1호, pp. 3-5 1999
[4] 유재수외 8인, "전자도서관 표준문서관리를 위한 XML 저장 관리기 기술 개발", 케이오텍 최종보고서, 1999
[5] 이용석, 손기락, "XML 문서 저장 시스템의 설계 및 구현", 한국정보과학회 학술 발표 논문집(I), 25권 2호, pp. 347-349, 1998
[6] 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
[7] Extensible Markup Language(XML) 1.0, "http://www.w3.org/TR/1998/REC-xml-19980210"
[8] Lee, Y.K., Yoo, S.J., Yoon, K., and Berra, P.B., "Index Structures for Structured Documents", Proc. Digital Library 96, pp. 91-99, 1996
[9] R. Sacks-Davis, T. Arnold-Moore, and J. Zobel, "Database systems for structured documents", Proc. The International Symposium on Advanced Database Technologies and Their Integration (ADTI '94), Nara, Japan, pp. 277-283, 1994
[10] Tuong Dao, Ron Sacks-Davis, James A. Thom, "An Indexing Scheme for Structured Documents and its Implementation", Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA '97), pp. 125-134, 1997