깊이탐색과 노드간 최단거리를 이용한 XML 인덱싱 알고리즘 설계 및 구현

김광남°, 윤희병*, 김화수**
°*국방대학교 전산정보학과, **아주대학교 정보통신대학원
intermet°@chol.com, hbyung*@kndu.ac.kr, ajhskim**@ajou.ac.kr

Design and Implementation of XML-based Indexing Algorithm Using Depth-First and Shortest Distance Between Nodes

KwangNam Kim°, Heebyung Yoon*, Hwa-Soo Kim**
°*National Defence University, Dept. of Computer & Information Science
**Ajou Univ., Graduate School of Information & Communication Technology

요 약

웹기반 하에서 구조적인 정보를 표현하기 위해서 XML이 다양하게 사용되고 있으나 XML 기반 문서는 다양한 Schema와 노드의 표현으로 구성되어 있어서 이를 효율적으로 인덱싱 하여 저장하는 것은 매우 어려운 일이다. 이를 해결하기 위하여 추상화, DTD, K-ary 완전트리 기법 등 다양한 연구가 이루어지고 있으나 응용에 많은 제한을 가지고 있다.

본 논문에서는 XML 기반의 웹문서를 효율적으로 인덱싱하고 사용자의 질의에 최적의 결과를 제공하기 위한 알고리즘을 설계 및 구현한다. 인덱싱 시스템 설계를 위해서 먼저 노드(부모, 형제)의 ID를 추출하는 알고리즘을 제안하며, 문서 및 노드 테이블 설계 결과를 제시한다. 그리고 C#을 이용한 파싱과 인덱스 알고리즘을 구현하기 위하여 깊이탐색과 관계 노드간 최단거리를 이용하며, 알고리즘 실행 결과와 이 결과로 자동 생성된 문서 및 노드 테이블의 파싱 결과를 또한 제시한다.

1. 서 론

웹상에는 HTML 문서가 다양하게 존재하고 있고 사용자에게 검색엔진을 통해서 많은 정보를 제공해주고 있다. 그러나 태그가 표현 위주로 되어 있고 또한 한정된 키워드로 인해 사용자 질의에 대해 정확하게 시스템이 인식하여 제공한다는 것은 매우 어려운 일이다. 이러한 문제를 해결하기 위해 의미를 제공하고 사용자 태그가 가능한 XML이 사용된다. XML은 특정 도메인별로 분류할 수 있으며, 문맥의 의미를 더욱더 정확하게 표현할 수 있다[1].

XML은 HTML보다 사용자의 다양한 요구를 충분히 수용할 수 있고 SGML보다 사용하기 쉽다는 장점으로 인해서, 웹문서뿐만 아니라 전자도서관, CSCW, EDI, CALS 등을 포함한 다양한 분야에서 XML을 활용하기 위하여 폭 넓은 연구를 하고 있으며, 수학 분야의 MathML, 채널 기술의 CDF, 메타데이타를 기술하기 위한 RDF, 이동통신에서의 HDML, WML 등 많은 영역으로 활용범위가 확대되고 있다[2]. 그러나 XML 문서의 관리나 구조검색을 효율적으로 수행하기 위해서는 추가적으로 인덱싱 기법이 필요하다. 그래서, 추상화[1], DTD[2], K-ary 완전트리 기법 등 다양하게 연구가 이루어지고 있지만, 응용에는 많은 제한을 가지고 있다. 또한 특정 노드에 대한 직접적인 접근이 불가능 하거나 관련 노드를 접근하기 위해서는 복잡한 연산을 수행해야 하는 문제점을 가지고 있다[3][4][5].

본 논문은 깊이탐색과 노드간 최단거리를 이용한 인덱싱 알고리즘을 설계 및 구현한다. 이를 위해, 2장에서는 깊이탐색과 깊이차이를 이용하여 노드(부모,형제)의 ID를 추출하는 알고리즘을 작성하며, 사용자 질의에 효과적으로 응답하기 위해서 노드의 특징들을 매핑하기 위한 테이블 구조를 설계한다. 3장에서는 제안된 알고리즘을 C#으로 구현하여 실행한 결과와 그 결과로 자동 생성된 문서 및 노드 테이블을 보여주며, 4장에서 결론을 맺는다.

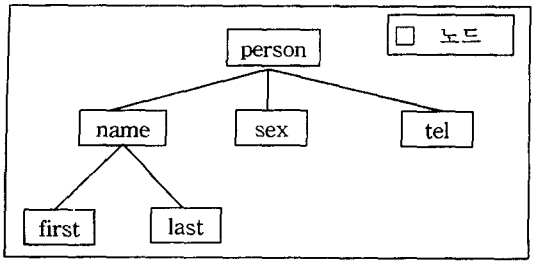
2. 인덱싱 시스템 설계

2.1. 노드의 ID 추출

XML 문서 위주의 구조정보 표현시 노드에 부여된 ID가 다를 수 있다[6]. 노드의 ID는 해당 노드를 유일하게 식별할 수 있게 한다. XML 문서에는 문서 내 또는 문서 사이에 동일 이름의 노드가 존재하고, 형제노드 사이뿐만 아니라 노드와 attribute간에도 동일한 이름이 존재한다. 따라서, 동일 이름을 갖는 노드들간의 관계는 해당 노드의 고유한 ID를 이용하여 추출한다.

2.1.1 부모 노드

부모노드의 추출은 다음의 두 가지 단계에 의해 수행된다. 첫 번째는 현재 노드에 대한 자신의 상위노드를 깊이탐색을 이용하여 식별하는 단계이며, 두 번째는 상위노드에 대하여 가장 짧은 연결선의 길이(distance)를 구하는 단계이다. 이를 위하여 [그림 1]과 같은 XML 문서의 트리구조를 고려한다.



[그림 1] XML 문서의 트리구조

[그림1]의 예에서, person의 깊이는 0이고, name, sex, tel의 깊이는 1, first, last는 2이다. 깊이탐색을 하면 바로 자신의 상위노드를 알 수가 있으나 동일한 깊이를 갖는 상위노드에 대해서는 최단거리를 이용하여 식별이 가능하다. 즉, last의 상위노드는 name, sex, tel이며, 최단거리를 갖는 상위노드는 name이라는 것을 알 수 있다. 따라서, 상위노드를 찾기 위한 식은 다음과 같이 부모노드 p로 정의할 수 있다.

$p = (depth(n) - 1) \wedge distance_{min}(p,n)$ ----------(1)

여기서 depth는 노드의 깊이를, n은 현재의 노드를, distance는 노드사이의 거리를 나타낸다. 식 (1)에서 부모노드 p는 현재 노드의 깊이보다 한 단계 위에 있어야 하며, 상위노드 사이에서 가장 짧은 거리에 위치하는 노드이다.

2.1.2 형제노드

형제노드를 식별하기 위해서는 노드의 깊이가 변화되는가를 확인하여 구할 수 있다. 만일 노드의 깊이 변화가 생기면 상위노드와 하위노드간의 관계이며, 노드의 깊이 변화가 없으면 형제노드로 간주하여 순차적으로 순서를 계산한다. [그림 2]는 형제노드를 추출하기 위한 알고리즘을 보여 준다.

```
Let Q(i) be Length of Queue Sibling nodes locate
Let Dep(j) be the depth of node
Let n be a name of node
Let i be the depth of node
Let j be the Length of Queue
procedure
begin
    if(j==0)                          ------------ ①
      add 1 into Q(0)
    else if(i==j)                     ------------②
    {
     extract depth(j)
     set 1 into Q(i)
     }
    else if(i>j)                      ------------③
    { extract Q(depth(j))
      Update Q(depth(j)) into Q(depth(j)+1)
    }
end
```

[그림 2] 형제노드를 추출하는 알고리즘

형제노드 추출 알고리즘을 [그림 1]의 트리구조로 설명 하면, person의 깊이는 0, name, sex, tel의 깊이는 1, first, last의 깊이는 2를 나타낸다. person은 root 노드이므로 형제노드가 없으며 순서는 1이 된다. 이것은 [그림 2]의 ①에 해당된다. 다른 노드들의 형제노드 순서를 식별하기 위해서는 깊이 변화상태를 확인한다. 만약 노드 탐색동안에 깊이 변화가 발생되면 해당 노드의 깊이를 저장하고 있는 Queue 값을 1로 설정한다. 여기에 해당하는 알고리즘은 [그림 2]의 ②이다. name, sex, tel의 경우에 깊이탐색 중에는 깊이의 변화가 생기지 않는다. 따라서 순서정보를 저장하는 Queue 값에 노드의 깊이 변화가 생기지 않는 한 1을 더하게 되며, 이에 해당하는 부분이 [그림 2]의 ③이 된다.

2.2 테이블 설계

추출된 부모노드와 형제노드의 ID를 통하여 노드 사이의 관계를 식별한 다음에는, 구조정보를 테이블에 저장한다. 그런 다음, 사용자 질의에 매핑되는 관련 노드들의 정보를 이용하여 확인할 수 있는 문서 및 노드 테이블을 설계하며, 이들 테이블은 [그림 3]과 [그림 4]에 나타나 있다.

DocId	DocName	DocPah

[그림 3] 문서 테이블

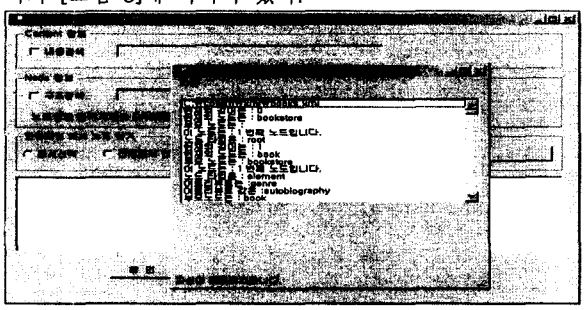
Eid	DocId	Name	Pid	depth	Sorder	type	content

[그림 4] 노드 테이블

문서 테이블에서 DocId는 문서의 ID이며, DocName은 문서의 이름이고, DocPath는 문서가 저장되어 있는 위치를 나타낸다. 노드 테이블은 노드의 정보를 나타낸 매핑 테이블로서, Eid는 노드에 대한 ID이고, DocId는 문서 ID를 나타내며, 이것은 문서 테이블의 DocId와 동일한 값을 갖는다. Name은 노드명, Pid는 부모노드를 식별하기 위한 ID로 상위노드 ID를 나타낸다. depth는 노드의 깊이를 나타내는 값이며, Sorder는 형제노드간의 순서를 나타낸다. type은 element, attribute, root 노드 등을 식별하기 위한 값이며, content는 노드가 가지고 있는 텍스트를 저장한다. 만약, type이 attribute를 나타내면 content는 attribute가 갖는 텍스트 값을 의미하고, type이 element를 나타내면 노드가 갖는 텍스트의 값을 나타낸다.

3. 구 현

제안한 인덱스 알고리즘과 문서 및 노드 테이블은 C# 을 이용하여 구현한다. 각 노드는 깊이탐색으로 식별하고, 노드의 이름과 타입, 노드간의 관계는 노드간 최단거리를 이용하여 식별한다. 또한, 형제노드간의 순서는 노드의 정보와 깊이, 그리고 깊이와 Queue의 거리를 비교하여 식별한다. 제안한 알고리즘을 구현한 후 실행된 결과가 [그림 5]에 나타나 있다.



[그림 5] 파싱 결과

실행이 완료되면 문서 및 노드 테이블이 자동으로 인덱싱되어 생성되며, 생성된 문서 테이블은 [그림 6]에, 노드 테이블은 [그림 7]에 그 결과가 나타나 있다.

DocId	DocName		
1	books.xml	C:₩cptkim₩xm	2004-02-27 오후
2	EID.xml	C:₩cptkim₩xm	2004-02-27 오후
3	KNOWLEDGE.xr	C:₩cptkim₩xm	2004-02-27 오후
4	mail.xml	C:₩cptkim₩xm	2004-02-27 오후
5	Noname1.xml	C:₩cptkim₩xm	2004-02-27 오후

[그림 6] 생성된 문서 테이블

Eid	DocId	ENname				EType	Content
1	1	bookstore	0	0	1	root	
2	1	book	1	1	1	element	
3	1	genre	2	1	1	attribute	autobiography
4	1	publicationdate	2	1	2	attribute	1981
5	1	ISBN	2	1	3	attribute	1-861003-11-0
6	1	title	2	2	1	element	Hybrid Architecti
7	1	author	2	2	2	element	
8	1	first-name	7	3	1	element	Herman
9	1	last-name	7	3	2	element	Melville
10	1	price	2	2	3	element	9.99
11	1	book	1	1	2	element	

[그림 7] 생성된 노드 테이블

[그림 7]에서 Eid가 9인 last-name 노드를 보면 Pid는 7이므로 부모노드는 author가 되며, 노드의 깊이는 3을 나타내고 Sorder를 통해서 author 노드의 2번째 자식 노드가 된다는 것을 확인할 수 있다. 또한 타입은 element이며 노드가 갖는 텍스트는 Melville임을 알 수 있다. 그리고 노드가 속한 문서는 DocId를 통해서 [그림 6]의 DocId와 일치하는 book.xml에 있음을 알 수 있다.

4. 결 론

본 논문에서는 XML 기반의 웹문서를 효율적으로 인덱싱하기 위하여 깊이탐색과 노드간 최단거리를 이용한 XML 인덱싱 알고리즘을 설계 및 구현하였다. 이를 위해 부모와 형제노드의 ID를 추출하는 알고리즘을 제안하였으며, 사용자의 질의에 효율적으로 응답할 수 있는 XML 구조정보를 이용한 최적의 문서 및 노드 테이블도 또한 제시하였다. 구현 결과 제안한 알고리즘은 DTD나 Schema와 같은 DocumentType을 이용하지 않아도 노드 관계를 쉽게 식별할 수 있었다. 또한, 제안된 알고리즘은 사용자가 내용, 구조, 혼합검색 질의시에도 효과적으로 응답할 수 있는 기능을 지원한다.

5. 참고문헌

[1] J.H. Chow et al., "Indexing Design for Structured Documents Based on Abstraction," 6th International Conference on Database Systems for Advanced Applications, pp.89-96, 1999.

[2] 강형일 외2., "XML 문서를 위한 효율적인 인덱싱 모델," Journal of the Research Institute for Computer and Information Communication, Vol. 8, No. 2, pp.89-97, 2000.

[3] Brian Lowe et al., "A Formal Model for Databases of Structured Text," 4th International Conference on Database Systems for Advanced Application(DASFAA), pp.449-456, 1995.

[4] Tuong Dao et al., "An Indexing Scheme for structured Documents and its Implementation," 5th International Conference on Database Systems for Advanced Applications(DASFAA), pp.125-134, 1997.

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

[6] Y.K. Lee et al., "Index Structures for Structured Documents," Proc. Digital Library 96, pp. 91-99, 1996