제21회 한국정보처리학회 춘계학술발표대회 논문집 제11권 제1호 (2004. 5)

XML 문서에서 상대경로를 포함한 질의 처리를 위한 색인 기법

정현숙 O 이민수
이화여자대학교 컴퓨터학과
{toylovesme O, mlee}@ewha.ac.kr

Indexing method to process XML query containing relative paths

Hyunsuk Jung O Minsoo Lee
Dept. of Computer Science & Engineering, Ewha Womans University

요 약

웹의 출현으로 XML 데이터에 대한 관심은 더욱 커지고 있다. XPath 와 XQuery 같은 XML 질의 언어는 비정규적인 데이터를 탐색하기 위해 경로에 라벨을 붙여 사용한다. 이러한 XML 데이터에 대한 질의를 효율적으로 처리하기 위해서는 효율적인 색인 기법이 필요하다. 그 동안 제안되어 왔던 기존의 색인은 일반적으로 XML 데이터 안에 루트 원소로부터 모든 경로의 라벨을 기록한다. 그런 경로 색인들은 자손을 찾는 "//"와 같은 상대 경로를 포함한 질의 경우 지나친 탐색으로 질의 수행의 성능을 저하시키게 된다. 이를 극복하기 위해 효율적인 색인 기법을 제안하고자 한다.

1. 서 론

XML(eXtensible Markup Language)은 정보 표현의 유연성 때문에 전자상거래와 지능적인 웹 탐색을 포함한 차세대 웹 어플리케이션을 필요로 하는 웹에서 문서를 교환하고 질의하기 위한 사실상의 표준으로 급속히 성장하고 있다. XML 데이터는 반구조적 데이터(semistructured data)[1]의 한 예이다. XML 문서는 계층적으로 중첩된 원소의 집합을 포함하며 원소와 함께 저장된 태그들은 데이터의 의미를 나타낸다. 따라서 반구조적인 XML 데이터는 계층적 구조를 형성하며 의미있는 설명적 태그들로 이루어진다.

또한 여러 XML 질의 언어가 최근 제안되었다. XPath[2]와 XQuery[3]와 같은 XML 질의 언어는 비정규적 구조의 XML 데이터를 탐색하기 위해 경로 표현을 사용한다. 그러므로, 비정규적 구조 그래프의 탐색은 XML 질의를 처리하는데 필수적인 요소의 하나이다. 특히, 상대 경로를 포함한 질의를 수행하는 경우 XML 을 처리하기 위해서 XML 데이터 그래프 전체를 탐색하는 것은 매우 비효율적이다. 그러나 구조적 요약이나 경로 색인은 단지 관련된 XML 의 일부분만을 찾도록 함으로써 질의 수행 시간을 빠르게 할 수 있다. 따라서 상대 경로를 포함한 질의 처리의 수행을 향상시키기 위한 제안 기법을 소개하고자 한다.

2 장에서 관련연구에 관해서 소개하고, 3 장에서는 Index Fabric 을 소개하고 그에 노드 넘버링 기법(node numbering scheme)을 추가한 색인기법을 제안하고, 4 장에서 결론을 내리도록 하겠다.

2. 관련연구

그 동안 경로 라벨 표현을 지원 하는 다양한 경로 색인이 개발되어 왔다. Goldman 과 Widom[4]은 strong DataGuide 라 불리는 경로 색인을 개발했다. strong DataGuide 는 간단한 라벨 경로에 국한되며, 여러 정규 표현식을 가진 복잡한 경로의 질의에는 유용하지 않다.

Milo 와 Suciu[5]는 1/2/T-index 를 개발했다. 그들의 접근은 역행 시뮬레이션 (backward simulation)과 그래프 검증으로 시작된 역행 양방향 시뮬레이션 (backward bisimulation)을 기반으로 한다. 1-index 는 트리 구조의 데이터일 때 strong DataGuide 와 같다.

Cooper et al[6]은 개념적으로 strong DataGuide 와 유사한 Index Fabric 을 발표했다. Index Fabric 은 각각의 XML 엘리먼트에 대한 경로 라벨을 문자열로 표시한다. 그런 다음 표시된 경로와 데이터 값을 문자열에 대해 효율적인 색인인 Patricia trie 에 삽입하게 된다. Index Fabric 은 데이터 값을 가지지 않는 XML 원소의 정보는 단말 노드에 저장하지 않으므로 원소 사이에 부모-자식 관계를 잃게 된다. 그러므로 Index Fabric 은 상대 경로를 포함한 질의를 처리하는 데는 효율적이지 않다.

XML 데이터 사용자들은 데이터의 구조를 고려하지 않고 의도한 결과를 내기 위한 부분적 경로 표현을 계획적으로 만들기 때문에 XML 데이터에 대한 많은 질의들은 부분적인 경로 표현을 가진다. Strong DataGuide, 1-index, Index Fabric 은 데이터 그래프의 루트로부터 시작하는 경로만을 기록하기 때문에 질의 처리기는 부분적인 질의를 색인 구조의 과도한 탐색에 의해서 간단한 경로 표현을 가진 질의로 다시 쓰게 된다. 이것은 수행 성능을 저하시키는 결과를 초래한다. 따라서 부분적인 표현을 가진 질의에 효율적인 색인 기법을 개발하는 것이 필요하겠다.

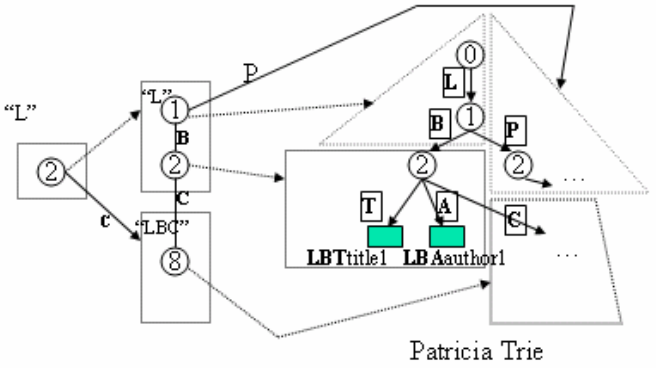
3. 색인 방법 제안

자주 변하지 않는 문서, 즉 검색 위주의 시스템에서 질의가 이뤄지는 경우를 바탕으로 색인 방법을 제안하도록 한다. Index Fabric 의 Patricia Trie 구조에 노드 넘버링 정보를 담고 있는 테이블을 추가한다. 부모-자식 뿐만 아니라 조상-후손의 관계를 찾기 위해 기존의 노드 넘버링 기법(node numbering scheme)을 Index Fabric 에 적용하는 것이다. 따라서 단순 경로를 가진 질의뿐만이 아니라 상대 경로를 가진 질의도 처리할 수 있도록 한다.

3.1 Index Fabric

Index Fabric 은 트리 구조이며 개념적으로 strong DataGuide 와 유사하다. 계층적 구조를 가지고 있고, 많은 탐색키를 색인하기 위해서 Patricia trie 를 사용한다.

루트에서 시작해서 단말노드에 이르는 단순 경로를 검색할 수 있도록 하기 위해 데이터 값을 가지는 원소의 단순 경로는 특정한 문자 구문으로 표시되며, 구문과 데이터 값의 조합으로 키를 유지한다.



[그림 1] Index Fabric 의 구조

그림 1 은 Index Fabric 의 구조를 나타낸다. 수직 계층으로는 데이터 값을 가지는 원소들의 정보만을 유지한다. 또한 수평 계층을 두어서 트리를 유사한 크기를 가진 블록으로 쪼개어서 탐색 시간을 줄인다. 쪼개진 블록에 관한 색인을 수평 계층의 왼쪽에 유지하게 하는 것이 특징이다.

본 논문에서는 Index Fabric 의 엘리먼트를 나타내는 designator 가 색인 구조에 한 번씩만 나타난다는 가정하에 제안한 기법을 다루도록 하겠다. 따라서 designator 가 각 노드를 구분하는 식별자 역할을 하게 된다.

3.2 노드 넘버링 기법을 적용한 Index Fabric

Index Fabric 은 부모-자식 관계를 유지하지 않기 때문에 루트에서 시작하는 데이터 값을 가지고 있는 경로들만을 처리하는데 적합하다.

여기서는 그러한 점을 극복하고자 Index Fabric 에 각 경로마다 노드 넘버링 정보를 가진 테이블을 추가함으로써 부모-자식 또는 조상-후손의 관계를 파악하여 어떤 노드라도 빨리 접근하여 상대 경로를 가진 질의("//A//B")도 처리할 수 있도록 한다.

노드 넘버링 기법의 장점은 트리 구조에서 어떤 노드라도 빠르게 조상-후손 관계를 찾을 수 있으며, 트리를 전부 검색하지 않고 조인 연산이 수행될 수 있다는 것이다. 그림 2 는 노드 넘버링으로 조상-후손 관계를 찾는 알고리즘이다.

```
Search(relative_path p, index I){
R=root if I;
   while p[i] is not NULL do{
   if p[i]="//" then{
                  /*후손 노드일때*/
    if (p[i+1]= descendent node of R, p[i-1])then
          /*node numbering*/
          Search(path p[i+2]…[n], index I)
     else return NULL;
    }
   else p[i]= "/" then{
                  /*자식 노드일때*/
    if(p[i+1]= child node of R, p[i-1])   then
          /*node numbering*/
          Search(path p[i+2]…[n], index I)
     else return NULL;
     }
   } /*while*/
   if p[i+1] contains a pointer to data then
      return the address of the data; /*data 가 있을 때*/
   else if sub-element of p[i+1] then /*data 가 없을 때*/
    return the address of the sub-element;
```

[그림 2] 노드 넘버링을 적용한 색인 알고리즘

Search()는 질의가 상대경로를 포함할 때에만 적용하도록 하며 색인을 검색하여 노드를 반환하는 함수이다. 먼저, 경로 p 와 인덱스 I 를 입력값으로 하고, 경로에 "//"와 "/"가 있는지를 구분하여, "//"이면 다음 엘리먼트가 조상-후손(ancestor-descendent) 관계인지를 "/"이면 부모-자식(parent-child) 관계인지를 노드 넘버링 기법을 적용하여 확인한다. 질의 경로의 맨 마지막까지 그러한 관계가 참임을 조사해본다. 참인 경우 마지막 엘리먼트가 데이터를 가지고 있다면, 데이터의 주소를 반환한다. 데이터가 없을 경우 종속된 엘리먼트가 있는지를 확인하고 있다면 그 주소를 반환하고 없다면 NULL 을 반환하게 된다. 다음은 노드 넘버링 기법으로 인덱스 노드의 관계를 확인하는 방법을 설명한 것이다.

1) "//A//B"의 경우 A 와 B 의 조상-후보 관계를 만족하려면 A.docID==B.docID, A.start<=B.start, A.end>=B.end 의 조건이 성립되어야 한다.

2) "//A/B"의 경우 A 와 B 의 부모-자식 관계를 만족하려면 위의 1)번 조건에 A.level==B.level-1 의 조건이 추가되어야 한다.

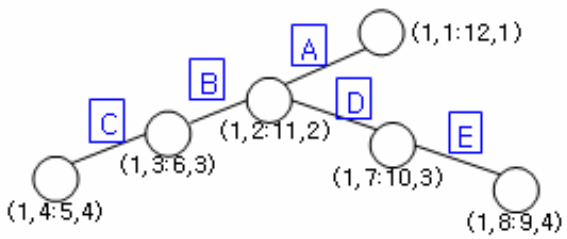
각각 A 와 B 의 두 튜플만 검사하면 되므로 루트에서 시작하는 모든 노드를 검색하는 것보다 검색시간이 훨씬 줄어든다. <그림 5 를 참고할 것.>

XML 문서가 실제 인덱스 구조에 적용되는 예를 보면 다음과 같다.

```
<book>
   <title> xyz
      <chapter> 12 </chapter>
   </title>
   <history>
      <year> 2004 </year>
   </history>
</book>
```

Designator dictionary	
A	Book
B	Title
C	Chapter
D	History
E	Year

[그림 3] XML 문서와 designator dictionary



[그림 4] 노드 넘버링을 적용한 색인 구조

그림 3 은 원래의 XML 문서와 그에 해당하는 엘리먼트들에 고유한 designator 를 할당한 테이블을 나타낸 그림이다. Designator 는 Index Fabric 에서 사용하는 것으로써 각 엘리먼트마다 할당된 고유한 문자로써 경로를 스트링으로 쓸 수 있도록 하는 역할을 한다. 예를 들어 book 아래 history 아래 year 아래

2004 의 값을 가지는 중첩된 경로를 표현할 때 '**ADE**2004'로 표현한다. 모든 경로를 이와 같은 방식으로 표현할 수 있다. 이것은 여러 개의 단순 경로를 스트링으로써 단순하게 표현할 수 있다는 장점을 가지고 있다.

그림 4 의 트리 구조는 각 경로의 원소마다 designator 를 할당하는 Index Fabric 의 인덱스에 추가로 각 경로마다 노드 넘버링 기법을 적용한 구조의 그림이다. 모든 엘리먼트들은 고유의 designator 를 얻게 되고, 그림 4 와 같이 인덱스 구조에는 각 designator 가 한번씩만 나타나게 된다. 따라서 각 designator 를 식별자로 해서 그림 5 와 같은 테이블을 생성할 수 있다.

Designator	Start position	End position	Level	Document id
ROOT	1	12	1	1
A	2	11	2	1
B	3	6	3	1
C	4	5	4	1
D	7	10	3	1
E	8	9	4	1

[그림 5] 그림 4 의 테이블 구조

그림 5 의 테이블은 각 엘리먼트마다 노드 넘버링 정보를 적용한 것이다. 예를 들어, 상대 경로를 포함하는 "//A//E" 와 같은 질의의 경우, A 를 원소로 가지는 튜플과 B 를 원소로 가지는 튜플을 찾아서 그것의 start_position 과 end_position 의 대소관계를 비교해봄으로써 B 가 A 의 후손인가를 확인해보게 된다. A 의 start_position 값:2 < E 의 start_position 값:8, A 의 end_position 값:11 > E 의 end_position 값:9, A 의 level 값:2 < E 의 level 값:4 의 세가지 조건을 모두 만족하므로 E 는 A 의 후손임을 알 수 있다.

다시 말해서 본 논문에서 제안하고 색인 방법은 기존의 노드 넘버링 기법이 XML 문서 자체에 적용됐던 것과는 다르게 Index Fabric 트리에 적용함으로써 색인 구조의 조상-후손 관계를 찾을 수 있다는 것이다. 따라서 상대 경로를 포함하고 있는 질의를 검색하는데 유용한 색인 방법이다.

4. 결 론

기존의 색인 기법인 DataGuide, 1-index, Index Fabric 이 루트에서 시작하는 경로만 기록하는 것과는 다르게 본 논문에서는 상대적인 경로도 지원할 수 있는 색인 기법을 다음과 같이 제안하였다.

Patricia trie 의 구조에 색인을 저장하는 Index Fabric 의 기법에 추가로 노드 넘버링 기법을 적용한다. 그럼으로써 색인 노드가 조상-후손 관계인지 부모-자식 관계인지를 확인하여 상대 경로를 가진 부분적 질의도 빠르게 검색할 수 있도록 하는 장점을 가지게 된다.

상대 경로를 포함하고 있는 "//A//B"와 같은 질의의 경우, A 와 B 두 개의 튜플만 찾아서 관련 노드 넘버링 기법을 적용하면 훨씬 빠르게 처리할 수 있다는 것을 알 수 있다.

앞으로 본 논문에서 제안한 기법을 기본으로 하여 구현 작업을 하고 성능 평가를 할 예정이다.

5. 참고문헌

[1] S. Abiteboul. Querying semi-structured data. In Proceedings of the 6th International Conference on Database Theory, pages 1-18, January 1997.

[2] J. Clark and S.DeRose. XML path language (XPath) version 1.0. W3C Recommendation, http://www.w3.org/TR/xpath, November 1999.

[3] S. Boag, D. Chamberlin, M.Fernandez, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu. XQuery 1.0: An XML query language. Working Draft, http://www.w3.org/TR/2001/WD-xquery-20011220, 20 December 2001.

[4] R. Goldman and J. Widom. DataGuides: Enable query formulation and optimization in semistructured databases. In Proceedings of 23rd International Conference on Very Large Data Bases, pages 436-445, August 1997.

[5] T. Milo and D. Suciu. Index structures for path expression. In Proceedings of 7th International Conference on Database Theory, pages 277-295, January 1999.

[6] B.Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon. A fast index for semistructured data. In Proceedings of 27th international Conference on Very Large DataBases, January 2001.