Pre-Order List를 이용한 XML 문서의 효과적인 색인방법

김영° 박상호[1] 박선[1] 이주홍[1] 홍준식[2]
인하대학교 컴퓨터정보공학과[0,1] 홍익대학교 전자.전기.컴퓨터공학부[2]
(youngjin°, parksangho, sunpark)@datamining.inha.ac.kr juhong@inha.ac.kr jnskhong@dreamwiz.com

An Efficient Indexing Method For XML Documents Using Pre-Order List

Young Kim° Sang-Ho Park[1] Sun Park[1] Ju-Hong Lee[1] Jun-Sik Hong[2]
School of computer science and engineering, Inha University[0,1]
Dept. of Electronic, Electrical & Computer Engineering, Hongik University[2]

요 약

최근 XML은 인터넷상의 데이터의 표현 및 교환의 표준으로 인식되면서 XML에 대한 많은 연구가 이루어지고 있다. 특히 XML문서의 정보량이 방대해짐에 따라 빠른 검색의 필요로 많은 인덱싱 기법들이 제안되었다. 최근의 연구 중, 패스를 기반으로 하는 인덱싱 기법들은 중간노드와 최하위노드의 검색, 조상-후손관계의 조인연산 등에서 성능이 떨어지는 경향이 있다. 이를 보완하기 위해 연구된 Numbering-Scheme 기반의 인덱싱 기법들은 대부분의 검색에서 우수한 성능을 보인다. 그러나 하위 노드가 늘어나는 경우엔 검색 오버헤드가 커질 수 있으며, 대량의 XML 문서나 구조가 다른 XML 문서가 추가되면 인덱스와 데이터 값의 재조정이 필요하게 된다. 이러한 문제를 해결하기 위하여 본 논문은 Numbering-Scheme을 기반으로 각 노드별 노드범위(Node-Range)와 Pre-Order List를 추가하여 검색성능을 높이고, 데이터의 삽입, 삭제에 효과적인 인덱싱 기법을 제안한다.

1. 서 론

최근 XML[1]은 인터넷상의 데이터의 표현 및 교환의 표준으로 인식되면서 정차 연구의 중요성이 증가하고 있다. 인터넷상의 주요한 정보들을 XML로 구현하여 저장, 검색, 교환하려는 요구들이 증가하고 있으며 의학, 국방, 법률, 전자상거래[2], 메신저등과 같이 인터넷상에서 일어나는 거의 모든 분야에서 XML이 사용되고 있다. 특히 전문성을 가진 XML문서들은 그 크기가 방대해지고, 사용자의 질의 또한 복잡해졌다. 따라서 수많은 XML 문서들을 저장하고, 복잡한 XML질의[3,4]를 빠르게 처리하기 위한 많은 인덱싱 기법들이 제안되었다. 최근에 제안된 인덱싱 기법들 중 많은 수가 루트노드에서 리프노드까지의 심플패스(Simple Path)를 인코딩하여 처리하거나, 각 노드에 <preorder, postorder>를 부여하여 검색하는 Numbering-Scheme[5]을 기반으로 한다. 그러나, 전자는 사용자의 질의에 따라 검색 성능에 차이를 보인다. 예를 들어 [그림 1]의 네가지 질의를 살펴보자.

```
예1) /Book/Authors/Author/Name
예2) /Book//title
예3) /Book/Authors//Name
예4) //Title
```

[그림 1] XPATH 의 예

패스를 인코딩하여 처리하는 심플패스기반의 인덱싱기법들은 예1)의 경우 뛰어난 성능을 보인다. 그러나, 예2)와 예3)의 경우엔 ' //' 이하의 모든 노드를 검색해야 하며, 예4)의 경우엔 최악의 경우 모든 노드를 검색해야 한다. 또한 ' //' 이후의 노드가 중복되어 있다면 더욱 성능이 떨어진다. 예1)에서도 리프까지의 패스가 생략되었다면 생략된 이후의 모든 노드를 검색하게 된다. Numbering-Scheme 기반 인덱싱 기법들의 경우에는 각 노드의 <preorder, postorder>를 분석하여 검색하므로 예1), 예2), 예3), 예4) 모두 좋은 성능을 보인다. 그러나 하위노드의 수가 많아지게 되면 인덱스(B+ Tree)의 성능이 떨어지게 된다. 또한 대량의 데이터가 삽입되거나, 구조가 다른 XML 문서가 삽입되어 패스의 이웃노드(Sibling-Node)나 하위노드가 추가된다면 최악의 경우 인덱스뿐만 아니라 데이터테이블의 <preorder, postorder>도 재조정 될 수 있다.

이러한 문제를 해결하기 위하여 본 논문에서는 Numbering-Scheme에 노드범위(Node-Range)와 중복노드(Duplicate Node)를 제안하여 확장하였고, Pre-Order List 란 별도의 인덱스를 관리한다. 제안된 방법은 다음과 같은 장점을 가진다. 첫째, XPATH 질의를 빠르게 분석한다. 둘째, Pre-Order List를 이용하여 실제 검색할 데이터테이블의 범위를 줄인다. 셋째, 데이터테이블을 최하위노드별로 유지하여 삽입, 삭제를 용이하게 한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 인덱싱 기법들을 소개하며, 각각의 장단점을 기술한다. 3장에서는 본 논문의 인덱싱 기법을 소개한다. 4장에서는 본 논문에서 제안한 인덱스 기법을 이용한 시스템의 구현과정을 보이며, XISS와 비교 실험한 결과를 보인다. 5장에서는 결론 및 향후 연구과제에 대해 기술한다.

2. 관 련 연 구

대용량의 XML문서에서 사용자의 질의를 빠르게 검색하기 위한 많은 인덱싱 기법들이 제안되어 왔다. 대표적인 인덱싱 기법에는 Lorel' s Index[6], Inverted Index를 이용한 인덱싱[7], XISS[8], Index Fabric[9], Stack Tree[10], BLAS[11]등이 있다. 최근의 인덱싱 기법인 Index Fabric은 XML 트리의 모든 노드와 데이터를 Patricia Trie[12]로 인코딩하여 패스검색시의 연산을 줄였으며, 레이어를 사용하여 트리의 바란스(Balance)를 맞췄다. 이 방법은 심플패스의 연산시엔 아주 뛰어난 성능을 보이나, 중간노드, 최하위노드의 검색, 조상-후손관계의 조인 연산시에 효율이 떨어진다. 이를 보완하기 위하여 Refined-Path의 개념을 두어 자주 사용되는 패스를 인덱싱에 추가하는 방법을 사용하였으나, DBA가 정의해야하므로 다양한 질의를 처리하기엔 부적합하며 많을 경우엔 인코딩의 효과가 떨어지게 된다.

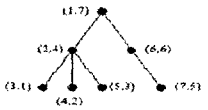
Numbering-Scheme기반의 XISS는 데이터의 삽입시에 유연한

확장을 위하여 노드별 <preorder, postorder>대신에, Order 와 후손들의 범위인 Size 로 레이블링(Labeling)하였다. XPATH 질의시 질의에 해당하는 Order 와 Size 를 계산하여 RDBMS(B+Tree)에 구현된 XML 데이터를 검색한다. 심플패스, 조상-후손관계의 조인연산, 중간노드의 검색 등 대부분의 연산에 좋은 성능이 보장되나 패스에 해당하는 태그, 속성 등의 노드가 너무 많거나 중복노드가 많을 경우엔 검색성능이 떨어질 수 있다. 또한 구조가 다른 XML 문서가 삽입되어 이웃노드나 하위노드가 추가될 경우 최악의 경우엔 인덱스뿐만 아니라 데이터의 Order 와 Size 도 재조정 될 수 있다. BLAS는 복잡한 질의처리의 효율을 높이기 위하여 D-Labeling과 P-Labeling을 사용하였다. D-Label은 Numbering-Scheme과 같은 방법으로 모든 노드를 <preorder, postorder>로 관리하며, P-Label은 최하위노드(Suffix)를 효율적으로 처리하기 위하여 비율(Proportion)로 나누어 관리한다. 사용자의 복잡한 질의가 주어졌을 경우 질의를 분해하고, D-Label과 P-Label을 이용하여 조상-후손관계와 최하위노드를 처리하여 검색의 범위를 줄인다. 그러나, 구현이 매우 복잡하고 위의 XISS와 같은 문제가 발생할 수 있다.

3. 인 덱 싱 기 법

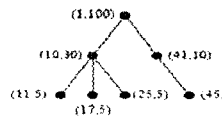
3.1. Numbering Scheme

XPATH 질의는 조상-후손관계의 조인연산(' // ')이 빈번하게 발생하게 된다. 이러한 연산을 처리하기 위해 기반이 되는 방법 중 하나가 Dietz 의 Numbering Scheme[5]이다.



[그림 2] Dietz 의 Numbering Scheme의 예

XML의 모든 노드는 $<d_1,d_2,d_3>$의 레이블을 가지며, 만약 노드 m, n이 있을 경우에 m이 n의 후손(Descendant)이면 $n.d_1 < m.d_1$ & $n.d_2 > m.d_2$를 만족하고, m이 n의 조상(Ancestor)이면 $n.d_1 > m.d_1$ & $n.d_2 < m.d_2$를 만족하며, m이 n의 자녀(Child)이면 $n.d_3+1 = m.d_3$를 만족한다. 만약 n과 m이 조상-후손관계가 아니라면 $n.d_2 < m.d_1$ OR $n.d_1 > m.d_2$의 중첩방지(Non-Overlap)조건을 만족하게 된다. 즉 $<d_1,d_2,d_3>$는 <preorder, postorder, level>을 나타낸다. XISS에서는 데이터의 삽입을 유연하게 하기 위하여 [그림 2]의 방법을 확장하였다.



[그림 3] Numbering Scheme의 확장의 예

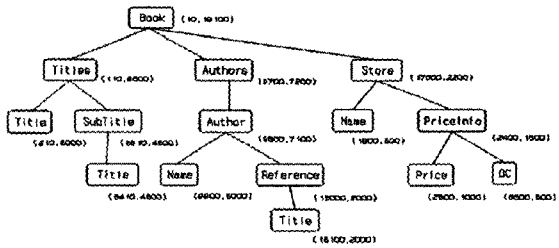
[그림 3]과 같이 preorder 는 order 로 postorder 는 Size로 변경하였고, Size 에는 하위노드에 포함될 데이터의 개수 + 확장가능한 범위가 들어가게 된다. 노드 m과 n이 있고 m이 n의 Parent라면 다음의 조건을 만족하게 된다.

Order(m) < Order(n) AND
Order(n) + Size(n) <= Order(m)+Size(m)

위의 방법으로 XML 문서를 검색해보면 사용자의 질의를 분석하여 위의 조건을 만족하는 조상-후손관계의 노드를 검색하고 거기서 나온 Order 와 Size 로 실제 데이터테이블을 검색하게 된다.

3.2. 인덱싱 제안

본 논문의 인덱싱 방법은 기존의 방법에서 질의의 결과로 나온 preorder 와 postorder 를 데이터인덱스(B+Tree)에서 비교하지 말고, XPATH 질의를 분석하여 나온 범위의 데이터를 바로 결과로써 사용하자는 것이다.

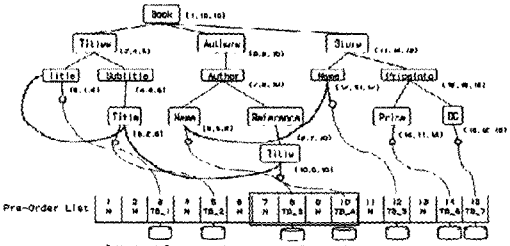


[그림 4] XISS의 Numbering Scheme을 사용한 예

[그림 4]와 같이 구성된 XML 데이터가 있을 때 XPATH 질의 " /Book//Author" 이 주어지게 되면 질의를 분석하여 최종적으론 Order 9800 보단 크면서 9800 + Size(7100) 보다 같거나 작은 값을 가진 XML 데이타를 데이타테이블에서 검색하게 된다. 즉

SELECT Doc_id, Values FROM Xiss_ValueTable
WHERE Order > 9800 AND (Order +Size <= 16900)

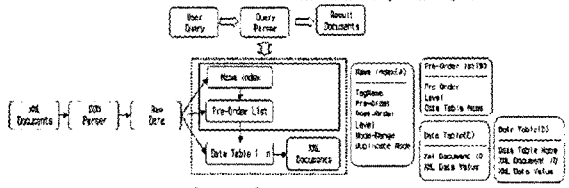
가 된다. 본 논문에서는 이러한 검색을 하지 않는 방법을 제안한며, XML 데이터의 삽입시 최하위노드별로 데이터테이블을 유일하게 생성한다. 그리고 노드범위(Node-Range)와 Pre-Order List를 사용하여 XPATH 질의에 해당하는 데이터테이블을 검색한다.



[그림 5] Pre-Order List를 사용한 예

본 논문에서는 " /Book//Author" 의 질의의 범위로 <7,10>을 얻는다. 그리고 Pre-Order List의 7부터 10사이에 위치한 데이터테이블을 오픈함으로써 검색이 완료된다.

본 논문에서 제안한 인덱싱 구조는 다음의 [그림 6]과 같다.



[그림 6] 인덱스의 구성

[그림 6]의 Name Index(A)는 노드별 preorder, postorder, level, Node-Range, Duplicate-Node 등의 정보로 인덱싱 된다. XPATH 질의시 질의를 분석하여 preorder 와 Node-Range를 얻은 후에 Pre-Order List의 해당범위로 이동하여 결과를 얻는다. Node-Range의 계산방법은 다음과 같다.

```
Node : n, Pre-Order List 의 범위 : x
[Node-Range의 계산]
if n.pos = ROOT then
   n.noderange = n.postorder
else
   if n.sibling is true then
      n.noderange = sibling.preorder - 1
   else
      n.noderange = n's parent.noderange
   end if
end if
[Node-Range를 이용한 Pre-Order List 의 데이타테이블의 범위계산]
n.preorder <= x and x <= n.noderange and x.data_flg = 'Y'
```

[그림 7] Node-Range 계산 방법

Duplicate-Node는 조상-후손(' //')연산시에 나올 수 있는 중복된 노드들을 연결하는 포인터로 중복노드가 존재할 시에는 빠르게 이동하며 조상의 <preorder, postorder>를 비교하여 질의의 범위를 계산할 수 있다. [그림 6]의 Pre-Order List (B)는 노드의 preorder 순으로 데이터의 유무와 데이터테이블의 이름을 보관한다. [그림 6]의 Data Table (C)는 XML 문서의 유일한 ID와 데이터 값이 보관된다. 이와 같이 질의의 결과범위에 해당되는 데이터테이블을 Pre-Order List에서 바로 검색하여 오픈하는 구조로 되어 있으며,사용자의 값에 대한 조건연산, 예를 들어 /Book//Author[.=' 김영']이라는 질의가 주어졌을 경우엔 오픈된 데이터테이블의 XML Data Value를 비교하여 처리한다. 그러나 질의의 결과범위가 넓거나, 중복노드가 존재하여 많은 데이터테이블을 오픈 할 경우 RDBMS에 많은 오버헤드를 초래할 수 있으므로 (D)와 같이 단일 데이타테이블에 데이터테이블 이름(Data Table Name)을 Cluster Index로 지정하여 처리하는 방법도 제안한다. 패스의 최하위 노드가 많을 경우엔 성능이 향상될 것이라 기대한다.

예) [그림 5]에서의 XPATH 질의의 예
/Book/Titles/Title : 루트부터 Title까지 검색한 후 범위<3,3>을 얻고, Pre-Order List 3의 데이터테이블을 오픈한다.
/Book/Titles : 루트부터 Titles까지 검색한 후 범위 <2,5>를 얻고, Pre-Order List 2~5까지의 데이터테이블을 오픈한다.
/Book//Author : 루트부터 Author까지 검색한 후 범위<7,10>을 얻고, Pre-Order List 7~10까지의 데이터테이블을 오픈한다.
/Book//Title[.=' 경제학개론'] : 루트부터 Title까지 preorder 순으로 검색후 중복노드(Duplicate-Node) 포인터로 이동하면서 Book의 <preorder, postorder>안에 포함되는 Title의 범위 <3,3>, <5,5>,<10,10>을 얻은 후 Pre-Order List 의 3,5,10번의 데이터테이블의 XML Data Value와 비교하여 결과를 리턴한다.

3.3. 데이터의 삽입, 삭제

본 논문에서 제안한 방법은 데이터테이블을 Pre-Order List를 통해 별도로 관리하므로, 기존의 방법에서와 같이 데이터테이블의 preorder 와 postorder 를 재조정할 필요가 없다.

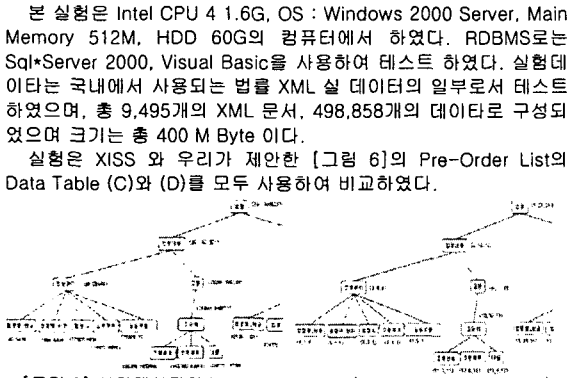
삽입방법은 XML 문서 구조에 따라 다르게 적용된다. XML 문서의 구조가 동일하면 단순히 Pre-Order List에 연결되는 데이터테이블에 항목을 추가한다. 이 경우 삽입시 preorder 와 postorder 의 계산이 필요가 없다. XML 문서의 구조가 다르면, Name Index와 Pre-Order List를 조정한다. 그러나 추가된 데이터테이블의 이름이 Pre-Order List에 유일하게 생성되므로 별도의 오버헤드가 없다.

데이터의 삭제시에는 Pre-Order List를 통해 데이터테이블로 이동한 후 항목을 삭제한다. 만약 데이터테이블이 Empty 가 되면 Pre-Order List의 Data_Flg 와 데이터테이블 이름을 지운 후 테이블을 삭제하고 Name Index 와 Pre-Order List를 재조정 한다.

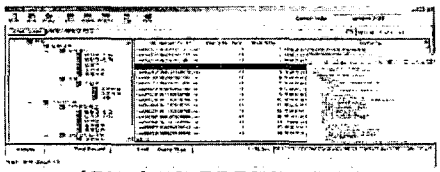
4. 성 능 실 험

본 실험은 Intel CPU 4 1.6G, OS : Windows 2000 Server, Main Memory 512M, HDD 60G의 컴퓨터에서 하였다. RDBMS로는 Sql*Server 2000, Visual Basic을 사용하여 테스트 하였다. 실험데이타는 국내에서 사용되는 법률 XML 실 데이터의 일부로서 테스트 하였으며, 총 9,495개의 XML 문서, 498,858개의 데이타로 구성되었으며 크기는 총 400 M Byte 이다.

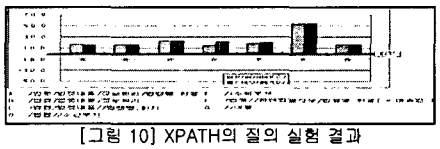
실험은 XISS 와 우리가 제안한 [그림 6]의 Pre-Order List의 Data Table (C)와 (D)를 모두 사용하여 비교하였다.



[그림 8] 실험데이타의 XML 노드의 구성 (XISS vs Pre-Order List)



[그림 9] 실험 프로그램의 실행화면



[그림 10] XPATH의 질의 실험 결과

실험결과 XISS 보다 조금 나은 성능평가가 되었지만, 패스의 최하위노드가 많다면 성능이 좋다. Pre-Order List의 Data Table (C)와 (D)의 격차는 거의 유사하였으나 질의에 여러 종류의 조인연산이 들어간 경우엔 단일 데이터테이블에서 검색하는 Data Table(D)의 성능이 높다.

5. 결 론 및 향 후 연 구 과 제

본 논문에서는 XML 문서의 효과적인 인덱싱 기법을 제안하였으며, 기존의 인덱싱 기법에서의 검색, 삽입시의 오버헤드를 최소화하는데 중점을 두었다. 본 논문에서 제안된 인덱싱 기법은 Numbering-Scheme을 기반으로 하지만, 노드범위(Node-Range)와 Pre-Order List를 추가하여 XPATH 질의시 데이터의 preorder 와 postorder 를 비교하지 않고 빠르게 검색한다. 또한 대용량 XML 문서의 추가 삽입, 구조가 다른 XML 문서의 삽입, 삭제 시에도 효과적으로 처리한다. 그러나, 다양한 패스의 복잡한 조인연산이 들어간 경우 많은 데이터테이블의 조인연산이 필요하므로 성능이 떨어질 수 있다. 향후 이와 같은 복잡한 XPATH 질의에도 효과적으로 검색할 수 있는 방법을 지속적으로 연구할 것이다.

6. 참 조 문 헌

[1] David Hunter외, Beginning XML, Wrox Press, 2000.
[2] Hiroshi Ishikawa, Manabu Ohta, " An Active Web-based Distributed Database System For E-Commerce" Proceedings of the International Workshop on web Dynamics, with the 8th ICDT' 01, London, UK, 3 January 2001.
[3] Jonathan Robie,Joe Lapp,David Schach," XML Query Language (XQL) http://www.w3.org/TandS/QL/QL98/pp/xql.html
[4] A.Deutsch,M.Fernandez,D.Florescu,A.Levy,D.Suciu. " XML-QL: A Query Language For XML" ,http://www.w3.org/TR/1998/Note- Xml-QL-19980819/
[5] Paul F.Dietz. Maintaining order in a linked list. In Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, pages 122-127, San Francisco, California, May 1982.
[6] S.Abiteboul et al. The Lorel Query Language For Semistructured Data, Int, J. on Digital Libraries 1(1): 68-88. 1997.
[7] 인경섭, 김형주, " 상이한 구조의 XML 문서들에서 경로질의 처리를 위한 RDBMS기반 역 인덱스 기법" ,정보과학회논문지 30권 제4호 420~428, 2003
[8] Quanzhong Li,Bonki Moon," Indexing and Querying XML Data For Regular path expression" ,Proceedings of the 27th VLDB Conference Page 361-370, 2001
[9] Brian F.Cooper, Neal Sample, Michael J.Franklin, Gisli R.Hjaltason, Moshe shadmon," A Fast Index For Semistructured Data" , Proceedings of the 27th VLDB Conference, Roma, Italy,2001.
[10] Shu-Yao Chien, Zografoula Vagena, Donghui Zhang, Vassillis J.Tsotras, Carlo Zaniolo, " Efficient Structural Joins On Indexed XML Documents" , Proceedings of the 28th VLDB Conference, 2002
[11] Yi Chen,Susan B.Davidson,Yifeng Zheng," BLAS : An Efficient XPath Processing System" , SIGMOD 2004 June 13-18, 2004
[12] Donald Knuth. The Art of Computer Programming, Vol.III, Sorting and Searching, Third Edition. Addison Wesley, Reading, MA,1998.