노드 범위와 Pre-Order List를 이용한 XML문서의 효율적 색인기법

An Efficient Index Scheme of XML Documents Using Node Range and Pre-Order List

김 영* 박 상 호** 이 주 홍***
Young Kim Sang-Ho Park Ju-Hong Lee

요 약

본 논문에서는 최근 방대해지는 XML문서의 효율적인 관리를 위해서 노드 범위와 Pre-Order List를 이용한 XML문서들의 인덱싱 기법을 제안한다. 기존의 제안된 인덱싱 기법들은 크게 패스(Path)와 넘버링(Numbering)을 기반으로 하고 있다. 그러나, 패스기반의 인덱싱 기법은 중간 노드와 최하위 노드의 검색과 조상-후손관계의 조인연산에 의해 효율이 떨어지는 단점을 가진다. 또한, 넘버링기반의 방법은 XML문서의 모든 노드에 번호를 부여하기 때문에 검색-오버헤드가 증가하는 문제를 가지며, 인덱스를 위해 많은 공간이 낭비된다.

따라서 본 논문에서는 이러한 문제점들을 해결하기 위해서 모든 XML문서에 노드범위 (Node Range)와 Pre-Order List를 이용한 인덱싱 기법을 제안한다. 이 방법은 유사한 구조의 XML문서가 많을수록 인덱스의 크기를 효과적으로 줄일 수 있으며, 검색 성능을 효율적으로 높일 수 있다. 또한 XML문서의 삽입, 삭제가 용이하다.

Abstract

In this paper, we propose indexing method to manage large amount of XML documents efficiently, using the range of node and Pre-Order List. The most of XML indexing methods are based on path or numbering method. However, the method of path-based indexing method shows disadvantages of performance degradation for join operations of ancestor-descendent relationships, and searching for middle and lower nodes. The method of numbers-scheme based indexing has to number all nodes of XML documents, since search overhead increased and the disk space for indexes was wasted.

Therefore, in this paper, we propose a novel indexing method using node ranges and Preorder-Lists to overcome these problems. The proposed method more efficiently stores similar structured XML documents. In addition, our method supports flexible insertion and deletion of XML documents.

☞ Keyword : 인덱싱(Indexing), 통합 패스(Integration Path), 노드 범위(Node Range), Pre-Order List

1. 서 론

최근 XML(Extensible Markup Language)은 인터넷상의 데이터 표현 및 정보 교환의 표준으로 인식되면서 점차 연구의 중요성이 증가하고 있다. 인터넷상의 중요한 정보들을 XML로 구현하여 저장, 검색, 교환하려는 요구가 증대되고 있으며, 특히 전문성을 가진 XML 문서들은 그 크기가 방대해지고, 사용자의 질의 또한 복잡해졌다. 따라서 많은 XML 문서들을 저장하고, 복잡한 XML질의[1,5,6,13]를 빠르게 처리하기 위한 여러 가지 인덱싱 기법들이 제안되었다. 인덱싱 기법들은 대체로 패스기반 인덱싱 기법[2,11,12,16]과 넘버링 기반 인덱싱 기법[9,10,14,17]으로 분류되며, 그 외에 기타 여러 인덱싱 기법[3,7,15]들이 있다. 패스기반 인덱싱 기법은 루트

* 정회원 : 인하대학교 컴퓨터 정보공학과
youngkim@sun-kwang.co.kr
** 정회원 : 인하대학교 컴퓨터 정보공학과 박사과정
parksangho@datamining.inha.ac.kr
***정회원 : 인하대학교 컴퓨터 정보공학과 부교수
juhong@inha.ac.kr
[2005/10/11 투고 - 2005/10/25 심사 - 2006/06/13 심사완료]

노드에서 리프노드까지의 심플패스를 인코딩[2,11,12,16]하여 처리하고, 넘버링기반 인덱싱 기법은 각 노드를 <preorder, postorder>로 인코딩하여 처리한다[9,10,14,17]. 그러나 전자는 심플패스의 질의에는 비교적 우수한 검색성능을 보이지만 조상-후손관계의 조인연산질의 혹은 하위노드의 질의 등에서 검색성능이 떨어진다. 후자는 각 노드에 부여된 <preorder, postorder> 혹은 <order, size> 등의 번호를 비교하여 검색하며 평균적으로 우수한 검색성능을 보인다. 또한 XML문서별로 노드에 번호를 부여하여 관리하므로 구조가 다른 XML문서를 쉽게 삽입할 수 있으며, 관계형 데이터베이스에 저장하는 것이 용이하다. 그러나 질의에 포함된 노드의 수가 많을수록 다중의 자체조인(Self-Join)연산으로 인해 검색-오버헤드가 커지게 된다. 또한 인덱스가 문서별로 관리되므로 저장 공간의 낭비를 초래한다.

본 논문에서는 위에서 언급한 문제점을 해결하기 위하여 넘버링기법에 기반 하면서 동시에 모든 문서의 패스를 통합하여 관리하고 노드범위와 Pre-Order List란 별도의 인덱스를 관리하는 새로운 방법을 제안한다. 제안된 방법은 다음과 같은 장점을 가진다. 첫째, XPath 질의를 빠르게 분석한다. 둘째, 패스의 인덱스 저장공간을 효율적으로 관리한다. 셋째, Pre-Order List를 이용하여 실제 검색할 데이터의 범위를 줄인다. 넷째, 데이터를 최하위노드별로 유지하여 삽입, 삭제를 용이하게 한다.

본 논문의 구성은 다음과 같다. 2장에서는 최근의 인덱싱 기법들을 간략히 소개하며, 각각의 장단점을 기술한다. 3장에서는 제안한 인덱싱 기법을 소개하고 질의처리의 예를 설명한다. 4장에서는 제안한 인덱스 시스템의 구현 과정을 보이며, 넘버링기법의 대표적인 인덱싱 기법 중 하나인 XISS[9,10]와 비교 실험한 결과를 보인다. 5장에서는 결론 및 향후 연구과제에 대해 기술한다.

2. 관련 연구

XML문서는 트리의 형태로 표현할 수 있으며, 수많은 중복된 노드를 포함한다. 따라서 XML문서 혹은 XML문서 내의 데이터를 검색할 때 중복된 노드를 포함한 모든 경로를 운행해야 한다는 문제점이 있다. 이 문제를 해결하기 위하여 Data Guide[11]란 색인기법이 제안되었다. 이 방법은 XML문서에 대응되는 모든 질의 형태를 심플패스로 요약하고 데이터의 저장 위치를 보관하여 중복된 경로를 통합하는 방법을 사용하였다. 이를 기반으로 하여 1-Index / 2-Index / T-Index[16]등이 제안되었다. Index Fabric[2]은 복잡한 XML문서의 패스연산을 줄이기 위하여 Patricia Trie[4]를 적용하였다. Patricia Trie는 문자열 삽입시에 중복된 문자를 제거하고 서로 다른 문자의 위치와 해당문자를 저장하는 압축 방식으로 구현된다. Index Fabric에서는 XML문서의 루트노드로부터 리프노드까지의 패스와 데이터를 Raw Data로 정의하고, Patricia Trie로 압축하여 패스연산의 범위를 줄이는 데 중점을 두었다. 그러므로 심플패스의 질의시에 루트에서 리프까지의 압축된 패스를 검색하므로 매우 좋은 검색성능을 보인다. 그러나 중간노드, 최하위노드의 검색, 조상-후손관계('//')의 조인 연산에는 효율이 떨어지게 된다. 이에 대한 보완책으로 자주 사용되는 패스를 인덱스에 추가하는 Refined-path방법을 사용했으나, DBA가 직접 정의해 야 하므로 다양한 질의처리에 부적합하며, 많을 경우엔 압축의 효과가 떨어진다.

XPath질의는 조상-후손관계의 조인연산('//')이 빈번하게 발생한다. 이러한 연산을 효율적 으로 처리하는 넘버링기법(Dietz's Numbering Scheme)에 기반한 여러 가지 인덱싱 기법 들이 개발되었다[9,10,14,17]. 넘버링기법은 XML 문서 내의 모든 노드에 <d1, d2, d3>의 번호를 부여한다. 만약 노드 m이 노드 n의 후손이라면 n.d1 < m.d1 and n.d2 > m.d2의 관계를 만족하고, m이 n의 조

상이라면 n.d1 > m.d1 and n.d2 < m.d2의 관계를 만족하며, m이 n의 자녀라면 n.d3 + 1 = m.d3의 관계를 만족한다. 즉 <d1, d2, d3>는 <preorder, postorder, level>의 속성을 갖는다. 이러한 속성을 이용하여 각 XML문서에 부여된 번호를 비교하여 사용자의 질의처리를 수행한다. 그러나 중간 노드나 하위노드의 삽입 시에 모든 노드의 번호를 재조정해 주어야 하는 문제점을 해결하기 위해서, XISS [9,10]는 모든 노드를 <preorder, postorder> 대신에 <order, size>로 확장하였고, size에는 하위노드에 포함될 데이터의 개수 + 확장가능 범위가 들어가도록 하였다. 즉, 노드 m과 n이 존재하고, n이 m의 후손이라면 order(m) < order(n) and order(n)+size(n) <= order(m) + size(m)의 관계를 만족하게 된다. 각 문서별로 <Order, Size>를 부여하므로 구조가 다른 문서의 삽입 시에도 쉽게 인덱싱 할 수 있는 장점을 가진다. 그러나 문서별로 각 노드별 <order, size>를 모두 관리하므로 인덱스의 공간이 비효율적으로 사용되며, 질의 내의 모든 노드의 수만큼의 자체조인 연산을 하므로 검색-오버헤드가 매우 크다. XML문서별로 인덱싱 되어 있기 때문에 XPath질의에 포함된 노드의 수만큼 문서번호별로 조인연산이 필요하며, 데이터 역시 각 문서별 노드에 부여된 번호를 비교하게 된다. 자료가 대용량이고 질의에 포함된 노드의 수가 많을 수록 검색성능은 떨어지게 되며, 데이터(또는 속성)값의 비교 연산이 포함된 경우 더욱 성능이 저하된다.

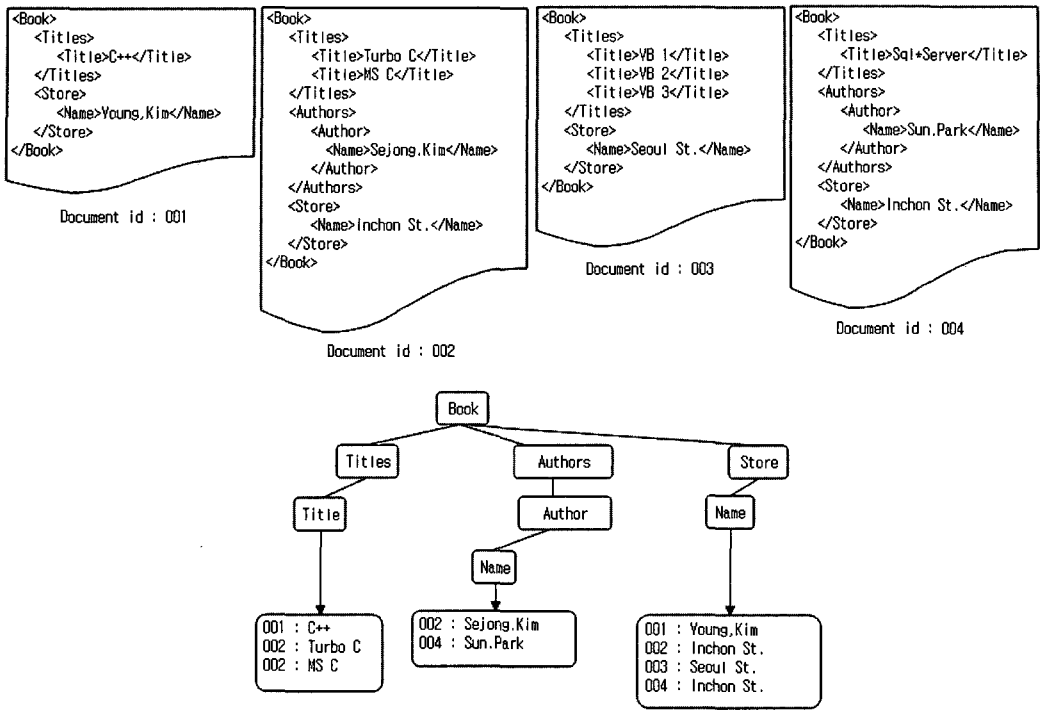
3. 통합패스의 노드 범위를 사용한 인덱싱 기법

본 논문에서 제안하는 방법은 문서의 패스를 통합 관리하므로 인덱스의 크기가 작으며 패스 연산의 검색-오버헤드도 매우 적다. 또한 데이터와 속성 인덱스는 번호를 부여하지 않으므로 번호에 대한 비교연산을 하지 않는다. 즉 조건 질의가 아닌 순수한 경로질의 에서는 데이터에 대한 검색을 하지 않으므로 검색성능이 우수하다.

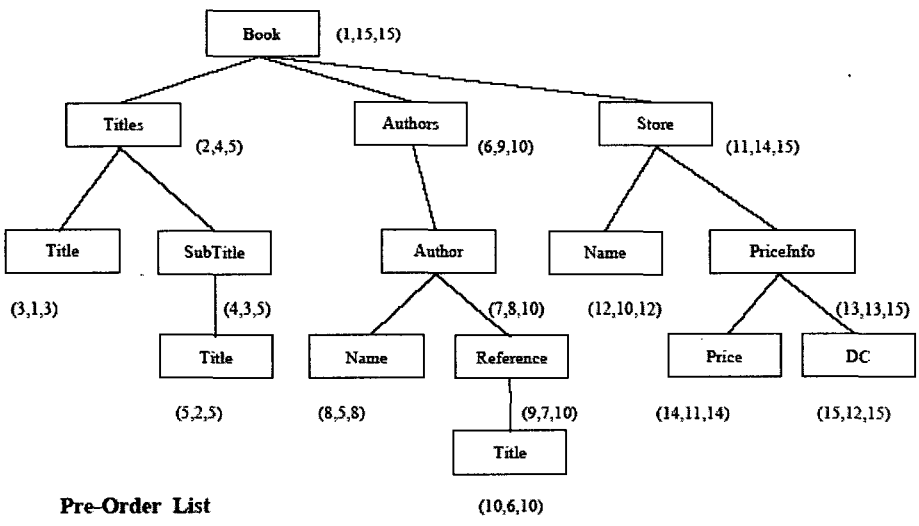
제안한 인덱스는 (그림1)의 (a)와 같이 여러 구조를 가지는 XML문서들을 동일한 노드를 경유하는 하나의 경로로 통합하여 구성하며, 심플패스의 최하위노드별로 문서ID별 XML데이터를 구성하는 별도의 인덱스를 관리한다. 즉 통합한 패스만 번호를 부여하여 관리하며 질의 시 패스검색결과는 데이터가 저장된 테이블의 범위만을 가져오는데 사용한다. 모든 XML문서의 삽입 시 통합된 XML패스의 각 노드는 (그림1)의 (b)와 같이 넘버링 기법을 사용하여 <preorder, postorder>를 부여하고 해당 데이터의 범위를 나타내는 노드범위(Node-Range)를 추가로 부여한다. 그리고 Pre-Order List는 통합 패스의 노드를 preorder 순의 배열로 구성하며, 각 항목에는 최하위노드별 데이터가 저장된 데이터테이블의 이름을 보관한다. XPath질의가 주어졌을 경우 통합패스를 이용하여 데이터의 범위(preorder, node-range)만을 구하고, 해당범위 내의 데이터테이블을 바로 개방하는 것이 제안방법의 주요한 아이디어 이다.

제안된 시스템은 (그림2)에 의해서 설명될 수 있다. 제안 시스템은 통합패스를 구성하여 XPath 질의를 분석하기 위한 Name Index(A), 통합패스의 모든 노드를 preorder 순으로 구성하여 데이터테이블의 이름을 저장하는 Pre-Order List(B), 데이터테이블 안에 포함된 XML문서의 ID를 보관하는 Doc_id_TB(C), 그리고 실제 데이터를 저장하는 Data Table(D)로 구성된다. XML문서의 삽입 시에 DOM Parser를 이용하여 모든 데이터를 심플패스와 데이터로 구성하여 임시 Raw Data에 보관한다. Name Index에는 기존의 저장된 정보와 비교하여 각 노드의 <preorder, postorder, node-range> 등의 정보를 저장한다. 즉, XML문서단위로 번호를 부여하지 않고, 기존의 패스와 비교하여 통합 패스로 관리하게 된다. 노드범위(Node-range)는 XPath 질의를 분석하여 나온 데이터의 범위를 이용하여 Pre-Order List로 빠르게 이동하기 위해 사용되며 계산 방법은 다음의 (그림3)과 같다.

(a)

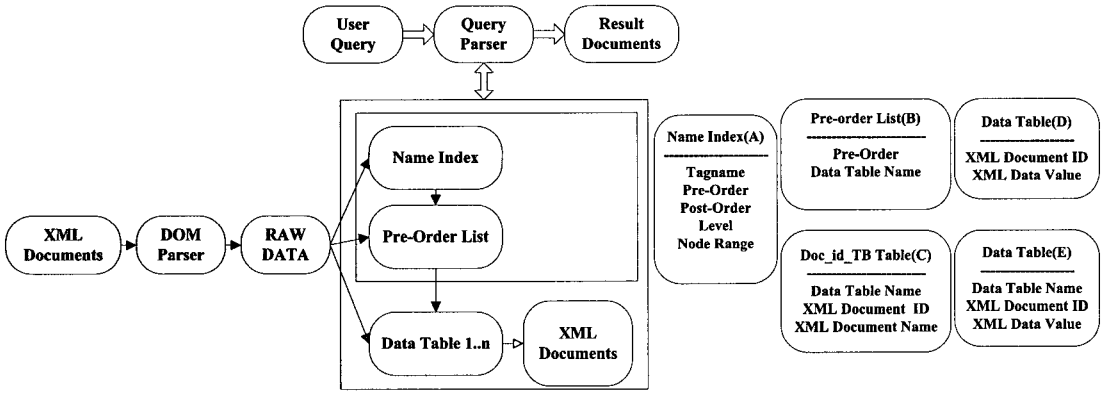


(b)



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
N	N	TB_1	N	TB_2	N	N	TB_3	N	TB_4	N	TB_5	N	TB_6	TB_7

〈그림 1〉 완성된 인덱스 구조



〈그림 2〉 인덱스 시스템의 구성

Pre-Order List(B)는 Name Index(A)의 preorder 순으로 저장되어 있는 하나의 동적배열 이다. 각 항목엔 Name Index의 preorder와 데이터테이블 이름이 보관되어 있으며, 크기가 매우 작다. XPath의 질의 처리시 Name Index(A)를 통하여 최하위의 preorder와 노드범위 를 얻고, 이를 이용하여 Pre-Order List(B)의 위치로 직접 이동한 후 데이터테이블의 이름을 얻어 바로 개방하는 방법을 수행하게 된다. 예를 들어 XPath 질의 "Book//Title [.='Visual C++']"를 처리한다고 할 때 기존의 XISS와 비교해 보면 다음 (그림4)와 같다.

(그림4)의 Pre-Order List의 수행순서 (1), (2), (3)을 보면, (1)에서 통합패스로 구성된 Name Index에서 XPath질의의 데이터 범위 <3,3>, <5,5>, <10,10>을 얻고, (2)에서 Pre-Order List의 해당위치로 직접 이동하여 데이터 테이블의 이름을 구한 후, (3)에서 해당 데이터 테이블내의 조건연산을 수행한다. 기존의 XISS와 같이 모든 문서내의 데이터 번호 비교를 하지 않으므로 빠른 검색 결과를 얻을 수 있다.

4. 실험 및 평가

본 절에서는 제안한 인덱싱 기법의 실험환경,

```
Node : n, Range of Pre-Order List : x
[ Calculate Node Range ]
if n.pos = ROOT then // 노드 n이 루트노드라면, 노드범위는 n의 postoder
   n.noderange = n.postorder
else
   // 노드 n의 오른쪽 형제노드가 있다면, n의 노드범위는 오른쪽 형제노드의 preorder - 1
   if n.rightsibling is true then
      n.noderange = rightsibling.preorder - 1
   // 그렇지 않다면, 노드 n의 노드범위는 부모노드의 노드범위
   else
      n.noderange = n.parent.noderange
   end if
[ Range calculation of Pre-Order List data table using Node Range ]

n.preorder <= x and x <= n.noderange and x.data_flg = 'Y'
```

〈그림 3〉 노드범위(Node-Range) 계산 알고리즘

```
XPath Query : /Book//Titles[.= 'Visual C++']
(Comparison of the Ancestor-descendant relationship and the data value)
(XISS)
SELECT i1.doc_id, i1.value
FROM   Element_ix e1, Element_ix e2, Value_ix i1
WHERE  e1.doc_id = e2.doc_id
AND    e2.doc_id = i1.doc_id
AND    e1.tagname = 'Book'
AND    e2.tagname = 'Titles'
AND    e1.order < e2.order  AND (e1.order + e1.size) >= e2.order
AND    e2.order < e3.order  AND (e2.order + e2.size) >= i1.order
AND    e1.level < e2.level
AND    i1.value = 'Visual C++'


(Pre-Order List)
  (1) SELECT  e2.preorder, e2.noderange
      FROM    NameIndex e1, NameIndex e2
      WHERE   e1.tagname = 'Book'
      AND     e2.tagname = 'Titles'
      AND     (e1.preorder < e2.preorder AND e1.postorder > e2.postorder)
      AND     e1.level < e2.level
  (2) Result of Pre-Order List : <3,3>, <5,5>, <10,10> => "TB_1", "TB_2", "TB_4"
  (3) SELECT * FROM TB_1 WHERE Values = 'Visual C++'
      UNION
      SELECT * FROM TB_2 WHERE Values = 'Visual C++'
      UNION
      SELECT * FROM TB_4 WHERE Values = 'Visual C++'
```

〈그림 4〉 XISS와 제안한 방법의 비교

실험 데이터 등에 대하여 간략히 설명하고 기존의 인덱싱 기법과 비교 실험하여 제안방법이 인덱스 공간을 줄이고, 검색성능향상에 기여하였음을 보이고자 한다. 비교실험 인덱스는 패스를 기반으로 방법의 경우 2장에서 설명한 바와 같이 중간노드, 조상-후손관계의 조인연산 등에서 효율이 매우 떨어지므로, 기존의 넘버링기법 기반의 대표적 인덱싱 기법 중 하나인 XISS와 비교·평가하였다. 또한 제안한 Pre-Order List를 동적배열이 아닌 디스크 기반으로도 구현하여 같이 비교 실험하였다. 결과적으로 기존의 방법보다 우수한 검색성능을 보였다.

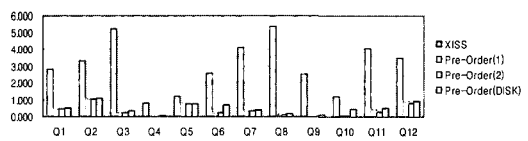
실험환경은 Windows 2000 서버를 OS로 사용하고, 펜티엄 IV 2.0G, 512MB RAM, 60GB HDD사양의 개인용 컴퓨터에 구현하였으며, 관계형 데이타베이스로는 MS-Sql*Server2000, 개발언어로는 Visual Basic6.0과 T-SQL을 사용하였다. 또한 실험 XML문서들의 내부 데이터와 구조를 추출하기 위한 Parser로는 Micro Soft 사의 MSXML (DOM) 라이브러리를 사용하였다.

실험데이터는 서로 다른 구조의 세 종류의 데이터로 구성하였다. 실험데이타(A) 는 실제 법률 사이트에서 운영되고 있는 법률 XML문서들로 구성되며, 각 각의 문서 안에 포함된 노드 및 데이터의 수는 적으나, 문서의 수는 매우 많은 특징을 가진다. 실험 데이터(B)는 XISS의 실험에서 사용된 세익스피어 희곡 XML데이터로 문서의 수는 적으나 각 문서들 내부에 포함된 노드와 데이터의 수가 매우 많으므로 문서별 크기가 매우 큰 특징을 가진다. 실험 데이터(C)는 임의의 Synthetic Dataset으로 많은 중복 노드와 데이터를 포함하고 레벨이 매우 깊은 특징을 가진 대용량의 XML문서들로 구성되어 있다. 상세내역은 다음의 (표1)과 같다.

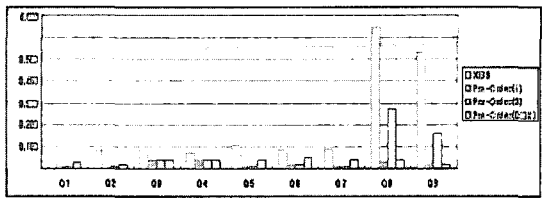
〈표 1〉 실험 데이터 상세 내역

구 분	실험데이타 (A)	실험데이타 (B)	실험데이타 (C)
문서의 수	9,495 개	38 개	200 개
전체 문서의 노드 및 데이터 수	725,538 건	150,059 건	2,741,100 건
전체 문서의 크기	287 MB	13 MB	251 MB
문서 당 평균 엘리먼트의 수	24 개	90 개	58 개
문서 당 평균데이터의 수	52 개	3,967 개	13,647 개
문서의 최대 깊이	5	6	20

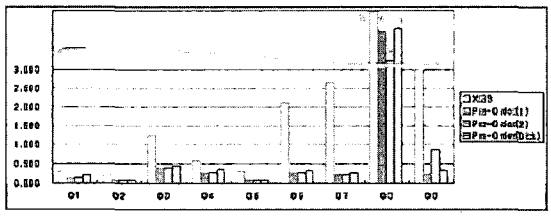
검색 실험에 사용한 질의
Q1:/법령/법령내용/정보관리/법령명_한글(심플패스)
Q2: /법령/관보정정문/관련법령정보(심플패스 생략)
Q3:/법령/관보정정문/관보 정문내용/정정일자(심플패스)
Q4:/법령//공동부령(조상-후손관계)
Q5: /법령//법령명_한글(조상-후손관계+중복노드)
Q6: /법령//조문//조문제목(다중 조상-후손 관계)
Q7:/법령//관련법령정보/법령명_한글(조상후손관계+심플패스)
Q8:/법령/법령내용/정보관리/법령명_한글[.='지방세법시행규칙']
Q9: /법령//공동부령[.='농림부']
Q10:/법령//관련법령정보/법령명_한글[.='공해방지법']
Q11: /법령/관보정정문/관련법령정보[.='무역거래법]
Q12:/법령//조문//조문제목[.'正富支侵']



검색 실험 질의
Q1: /PLAY/INDUCT/TITLE(심플패스)
Q2: /PLAY/PROLOGUE(심플패스생략)
Q3: /PLAY//ACT/SCENE(심플패스생략)
Q4: /PLAY//SPEECH(조상-후손관계)
Q5: /PLAY//INDUCT/SPEECH(조상-후손관계)
Q6: /PLAY//SCENE/SPEECH(심플패스+조상-후손관계)
Q7: /PLAY//SPEECH/TITLE(조상-후손 다중)
Q8: /PLAY//TITLE[.='ACT'](조상-후손관계+조건)
Q9:/PLAY//SCENE/TITLE[.='SCENE'](조상후손관계+조건)



검색 실험 질의
Q1: /A/D/E/C/B/E/E/E/F/E(심플패스생략)
Q2: /A/B/B/C/B(심플패스생략)
Q3: /A//B//C(조상-후손관계)
Q4: /A//D//C//B//E(조상-후손관계+심플패스)
Q5: /A//D//C//B//E//F(조상-후손관계+심플패스)
Q6: /A/D/E//B//E//F/F(심플패스+조상-후손관계)
Q7: /A/D/E//B/E/E//F/F(조상-후손 다중)
Q8:/A//F//E[.='addchannel'](조상후손관계+조건)
Q9:/A/D/E//B/E/F[.='kbEnglish'](조상후손관계+조건)



〈그림 5〉 실험 검색 질의 및 결과 그래프

(그림5)는 (표1)의 3가지 실험 데이터들을 이용한 실험 결과를 보여준다. 또한, 3가지 실험결과는 기존의 XISS 방법과 본 논문에서 제안한 방법을 (표2)와 (표3)를 각각 이용한 Pre-Order (1), Pre-Order(2), 그리고 Pre-Order List를 동적 배열이 아닌 디스크 기반으로 구현한 총 4가지의 인덱스 방법의 성능을 비교한 결과이다. 이때, 실험 검색 질의는 크게 심플패스 질의, 조상-후손관계의 조인연산, 조상-후손관계의 조건연산의 세가지 패턴으로 나누어 구성하였다.

〈표 2〉 실험 데이터의 테이블과 필드 설명(Pre-Order(1))

DataTable1~n(Pre-Order List에서 지정된 데이터 테이블 이름)	
Doc_id	XML 문서 고유 번호
Seq_value	XML 문서내의 데이터 값의 순서
Xml_value	XML 문서내의 데이터 값

〈표 3〉 실험 데이터의 테이블과 필드 설명(Pre-Order(2))

DataTable	
Doc_id	XML 문서 고유 번호
Seq_value	XML 문서내의 데이터 값의 순서
Xml_value	XML 문서내의 데이터 값
DataTableName	Pre-Order List에서 지정된 데이터 테이블 이름

(그림5)의 세가지 실험 결과에서 모든 질의에서 본 논문이 제안한 방법이 기존의 방법보다 검색 결과가 우수함을 볼 수 있다. 첫번째 실험의 경우, 통합패스 자체의 크기도 각 문서내의 노드별로 인덱스를 구성한 XISS보다 적으며, 데이터 역시 Q1부터 Q7까지는 넘버를 비교하지 않고, Pre-Order List안의 데이터 테이블을 개방함으로써 결과를 리턴하기 때문에 검색속도가 우수하다. 두번째 실험의 경우, 첫번째 실험 결과와 검색성능의 차이가 크지는 않았다. 그러나, 많은 데이터가 포함되어 있는 조건 질의 Q8의 경우는 제안된 방법이 우수한 검색 성능을 보였다. 세번째 실험의 경우, XISS는 질의에 포함된 노드의 수가 많을수록 많은 자체-조인연산이 일어나므로 제안된 방법보다 검색성능이 떨어짐을 보여준다. 또한 Q8과 Q9의 질의의 경우엔 데이터 인덱스의 넘버를 비교하며, 조건을 검색하므로 더욱 성능이 떨어짐을 볼 수 있다. 또한 본 논문에서 제안한 방법은 패스를 통합하여 관리하고, 각 데이터 테이블에 번호를 부여하지 않으므로 기존 방법보다 인덱스 공간을 효율적으로 사용할 수 있었다.

5. 결론 및 향후의 연구

패스를 기반으로 한 인덱싱 기법은 중간노드, 최하위노드의 검색, 조상-후손관계의 조인 연산 등에서 검색성능이 매우 떨어진다. 그에 반하여 넘버링기법 기반 인덱싱 기법은 모든 연산에서 평균적으로 좋은 성능을 보이며, 관계형 데이터베이스에 쉽게 구현이 가능하다. 그러나 각 문서별로 문서내의 모든 노드에 번호를 부여하고, 각각의 데이터와 속성에도 번호를 부여함으로써 검색-오버헤드가 커질 뿐만 아니라 인덱스의 공간 또한 비효율적으로 관리된다. 본 논문에서는 이러한 문제점을 해결하기 위한 방법을 제안하였다.

본 논문에서 제안한 방법은 다음과 같은 장점을 가진다. 첫째, 패스를 통합하여 인덱스의 공간을 줄였다. 둘째, 노드별로 preorder, postorder, node range를 부여하고 별도의 pre-order List를 추가하여 관리함으로써, 비교연산의 수를 줄이고 검색성능을 높였다.

또한, 본 논문에서 제안한 방법은 새로운 XML 문서의 삽입에도 효과적이다. 그리고 넘버링기법의 대표적인 인덱싱 기법 중 하나인 XISS와의 비교실험을 통해 검색성능의 우수함을 증명하였다.

그러나 다양한 패스의 복잡한 조인연산의 경우 많은 데이터테이블의 조인연산이 요구되므로 성능이 저하된다. 또한 서로 다른 구조의 XML 문서들이 많이 존재한다면, 그 만큼 최하위노드가 늘어나게 되므로 데이터테이블의 수도 늘어나게 된다. 향후에는 이와 같은 문제점들을 보완할 수 있는 방법을 지속적으로 연구할 것이다.

ACKNOWLEDGEMENT

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성.지원사업의 연구결과로 수행되었음.

참 고 문 헌

[1] A. Deutsch, M. Fernandez, D. Florescu, A. Levy and D. Suciu, "XML-QL : A Query Language For XML", http://www.w3.org/TR-/1998/Note-XML-QL-19980819/, 1998.

[2] B. Cooper, N. Sample, M. Franklin, G.

Hjaltason and M. shadmon, "A Fast Index For Semistructured Data", In Proc. of the VLDB, pp. 341-350, 2001.

[3] C. Chung, J. Min and K. Shim, "APEX: An Adaptive Path Index for XML Data", In Proc. of the ACM SIGMOD, pp. 121-132, 2002.

[4] D. Knuth, "The Art of Computer Programming, Vol.III, Sorting and Searching", Third Edition. Addison Wesley, Reading, MA, 1998.

[5] J. Clark and S. DeRose, "XML Path Language(XPath)", version 1.0 w3c recommenddation. Technical Report REC-xpath-1999 1116, World Wide Web Consortium, 1999.

[6] J. Robie, J. Lapp and D. Schach, "XML Query Language (XQL)", http://www.w3.org/TrandS/QL/QL98/pp/xql.htm, 1998.

[7] N. Bruno, N. Koudas and D. Srivastava, "Holistic Twig Joins: Optimal XML Pattern Matching", In Proc. of the ACM SIGMOD, pp. 310-321, 2002.

[8] P. Dietz, "Maintaining order in a linked list.", In Proc. of the Fourteenth Annual ACM Symposium on Theory of Computing, pp. 122-127, 1982.

[9] P. Harding, Q. Li and B. Moon, "XISS/R: XML Indexing and Storage System Using RDBMS ", In Proc. of the VLDB, pp. 1073-1076, 2003.

[10] Q. Li and B. Moon, "Indexing and Querying XML Data For Regular path expre-ssion" , In Proc. of VLDB, pp. 361-370, 2001.

[11] R. Goldman and J. Widom. "DataGuides: in semistructured databases", In Proc. of the VLDB, pp. 436-445, 1997.

[12] S. Abiteboul, D. Quass, J. McHugh, J. Widom and J. Wiener, "The Lorel Query Lan -guage For Semistructured Data", Int, J. on Digital Libraries 1(1): pp. 68-88, 1997.

[13] S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie and J. Simeon, "An XML Query Language(XQuery)", http://www.w3.org/TR/xquery/, 2004.

[14] S. Chien, Z. Vagena, D. Zhang, V. Tsotras and C. Zaniolo, "Efficient Structural Joins On Indexed XML Documents", In Proc. of the VLDB, pp. 263-274, 2002.

[15] S. Park and H. Kim, "A New Query Processing Technique for XML Based on Signature", In Proc. of the DASFAA, pp. 22-31, 2001.

[16] T. Milo and D. Suciu, "Index Structures for Path Expressions", In Proc. of the ICDT, pp. 277-295, 1997.

[17] Y. Chen, S. Davidson and Y. Zheng, "BLAS: An Efficient XPath Processing System", In Proc. of the ACM SIGMOD, pp. 47-58, 2004.

◐ 저 자 소 개 ◑

김 영 (Young Kim)
1994년 ~ 1998년 일진중공업(주) (구 이천전기(주)) 전산실 근무
1999년 선문대학교 자연과학대학 정보공학 학과 졸업(학사)
2005년 인하대학교 일반대학원 컴퓨터정보공학 학과 졸업(석사)
2005년 ~ 현재 (주)선광 컨테이너터미널 전산실 근무
관심분야 : 데이터베이스, 데이터마이닝, XML.
E-mail : youngkim@sun-kwang.co.kr

박 상 호 (Sang-Ho Park)
2002년 인하대학교 컴퓨터공학과 졸업(학사)
2004년 인하대학교 대학원 컴퓨터정보공학과 졸업(석사)
2004년~ 현재 인하대학교 대학원 컴퓨터공학과 박사과정
관심분야 : 데이터마이닝, 알고리즘, 패턴 인식
E-mail : parksangho@datamining.inha.ac.kr

이 주 홍 (Ju-Hong Lee)
1983년 서울대학교 컴퓨터공학과 졸업(학사)
1985년 서울대학교 대학원 컴퓨터공학과 졸업(석사)
2001년 한국 과학 기술원 컴퓨터 공학 졸업(박사)
2002~현재 인하대학교 컴퓨터공학부 부교수
관심분야 : 데이터마이닝, 데이터베이스, 정보검색, 신경망, 기계학습
E-mail : juhong@inha.ac.kr