RDBMS를 이용한 DTD 엘리먼트 타입 기반의 문서 색인 기법

박관순° 김택곤 김우생
광운대학교 컴퓨터과학과
kwansoon@hanwha.co.krO, tgkim@cs.kw.ac.kr, kwsrain@cs.kw.ac.kr

A XML Indexing Technique based on DTD's Element Types in RDBMS

Kwansoon Park° Tackgon Kim. Woosaeng Kim
Dept. of Computer Science, Kwangwoon Univ.

요 약

최근 XML 문서가 인터넷 기반의 애플리케이션 간의 자료 저장 및 교환을 위한 표준으로써 부상함에 따라 XML 문서의 저장 및 관리에 대한 연구가 활발히 이루어지고 있다. 하지만 XML 문서의 검색에 관련된 많은 연구들이 모든 XML 엘리먼트 경로에 대한 색인화로 인해 인덱스의 크기가 커지고 이에 비례하여 검색 성능이 떨어지는 문제를 보이고 있다. 본 논문에서는 이를 개선하기 위해 엘리먼트 타입을 기반으로 전통적인 역색인 방법을 XML 문서에 맞게 확장하고, RDBMS에 기반하여 계층구조를 갖는 XML 문서들의 자료를 구조적 넘버링(Numbering) 방법의 인덱스로 설계 하였다. 인덱스 테이블들은 엘리먼트 타입의 정보를 담고 있는 엘리먼트 타입 테이블, XML문서의 경로를 가지고 있는 경로 테이블, 역색인으로 구성된 Term테이블, Term 경로를 나타내는 Term경로 테이블을 생성한다. 이전의 XML 인덱싱 기법에 관련된 연구들에서 보이는 XML 문서상의 모든 경로에 대한 표현을 간소화 시키고, 이를 통해 보다 좋은 검색 성능을 보이고자 하였다.

1. 서 론

XML(eXtensible Markup Langua ge)[1]이 웹상에서 문서 교환의 표준으로 지정되고 난 후, 상당수의 회사나 단체들이 그들의 문서 포맷으로 XML을 채택하기 시작했다. 결국, 이러한 추세로 방대한 XML 문서가 생성되며, 이로 인해 XML 문서 데이터베이스로부터 정보를 효율적으로 추출해내기 위한 연구가 이루어져왔다. XML 문서에 대한 인덱싱 및 검색 기법들은 구조적 특징을 고려한 내용 검색이 많은 부분을 차지하고 있다.

XML 문서들로부터 정보를 추출하기 위한 수단으로 많은 질의어가 제안되었는데, 이들 질의어들을 이용하여 XML 문서들로부터 정보를 추출시 포함질의(containment query)[2][3]가 XML 정보검색 시스템 상에서의 질의의 핵심이 될 수 있다. 따라서 포함질의를 어떤 방법으로 지원하느냐 하는 것이 중요한 문제가 된다. 이것을 해결하기 위한 한 방법으로 기존의 정보검색분야에서 널리 쓰이는 역색인(Inverted Index)[4][5]기법을 고려할 수 있다.

이전의 많은 연구들이 모든 XML 경로를 색인화시켜 사용하는 방법을 제안하였는데, 이 경우 색인의 크기가 커지고 검색 시 성능이 감소하는 요인이 발생하게 된다. 본 논문에서는 검색 성능을 높이고 색인의 크기를 줄이기 위해 DTD 문서나 XML Scheme 정보를 통하여 엘리먼트 타입을 추출하고, 이를 기반으로 한 색인 방법을 제안하였다. 이를 위해 전통적인 역색인 방법을 XML 문서에 맞게 확장하고, RDBMS에 기반한 인덱스를 설계하여, 이전의 XML 인덱싱 기법에 관련된 연구들에서 보이는 XML 문서상의 모든 경로에 대한 표현을 간소화시키고, 이를 통해 보다 좋은 검색 성능을 보이고자 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 XML 인덱싱에 관련된 연구들에 대해 다루고, 3장에서는 포함질의를 처리하기 위해 본 논문에서 제안한 색인 모델에 대해 설명한다. 4장에서는 포함질의를 처리하기 위한 방법을 설명하고, 5장에서는 성능평가를 보이고 마지막으로 6장에서는 결론을 내린다.

2. 관련연구

XML 문서를 검색하는 방법을 위해 그 동안 많은 색인 기법에 대한 연구가 있어왔다.

포함질의를 처리하기 위해서 [2]에서는 XML 문서의 텍스트와 엘리먼트 각각에 대해서 키워드가 나타나는 위치를 XML문서의 시작부터 몇 번째 단어인지를 계산하여 (begin,end)의 숫자를 부여하고 이 숫자를 포함한 역 인덱스를 생성한다. 이 숫자는 자식-부모 관계를 파악하기 위한 모든 포함 질의에 그 값을 비교하는데 사용된다. 단어에 대한 색인을 처리하기 위한 T-INDEX와 엘리먼트 정보에 대한 색인을 구성하는 E-INDEX로 2개의 색인을 이용한 2-TABLE 방법은 두 개의 리스트로 확장된 역색인을 RDBMS를 이용해서 두 개의 테이블 즉, Element 테이블(term, document, begin:end,level)과 Text 테이블(term, document, position, level)에 저장하고 질의를 처리하는 방법이다. 이 방법은 색인의 크기가 커지고, 이에 따라 검색 성능이 감소하는 단점을 가지고 있다. 관계형 데이터베이스를 활용한 XML Query 검색에 관한 관련 연구[6]에서는 각 엘리먼트마다 서로 다른 테이블을 형성하는 방식을 사용함으로써 같은 단어에 대해서 루트 엘리먼트에서부터 실제 그 단어가 존재하는 엘리먼트까지 매 엘리먼트마다 역 인덱스(Inverted Index)를 생성하게 된다. 또한 이진 테이블 방식을 사용한 자체 조인으로 엘리먼트 사이의 종속 관계를 처리하기 때문에 복잡한 경로의 쿼리에서 상당한 부담이 따른다. [6]은 역색인을 확장해서 RDBMS에 저장했지만 데이터베이스의 테이블수가 많아지는 단점이 존재하게 된다.

3. 구조적 넘버링 색인 모델

3.1 구조적 넘버링 알고리즘

XML 문서에서 경로를 표현할 때, 수많은 중복되는 경로가 발생하게 되며, 이를 색인에 모두 포함시킨다면 많은 공간이 요

구되며, 자연적으로 검색 시 성능의 저하를 발생시키게 된다. 하지만, 이런 중복되는 경로들을 하나의 의미를 지닌 값으로 표현하면 이러한 문제를 해결할 수 있게 되므로, 본 논문에서는 엘리먼트 타입에 대한 노드의 유일한 ID값과 XML 문서의 경로에 대한 유일한 ID값에 대하여 구조 정보를 포함하는 특정 값을 부여하여 인덱싱의 효율성을 높이고자 한다. 그림1의 (a)에서와 같이 XML 문서가 있고, 이에 따른 DTD 문서를 그림1의 (b)와 같이 표현할 수 있다. 본 논문에서는 엘리먼트 타입에 대한 ID값과 XML 문서의 엘리먼트 경로들에 대한 ID값을 부여하기 위해 여러 가지 정보 값들을 더하여 새로운 값을 만들어 부여하게 된다.

이 ID값들은 그림2의 (a)의 트리 구조상에서 엘리먼트의 순서(count)값과 자신의 노드를 포함한 하위 노드의 개수를 표현하는 범위 (scope)값 그리고 레벨(level)값을 갖는다. 이렇게 구해진 값은 아래의 식을 사용하여 ID를 표현 한다.

ID = 순서 + 범위 + 레벨

이 ID값들은 그림2의 (b)에서 볼 수 있듯이 해당 위치를 기준으로 앞의 노드의 값보다는 항상 크며, 뒤의 노드들 중에서 값이 작은 것들은 자식 노드이거나 자손 노드이며, 뒤의 노드들 중에서 큰 값을 가진 노드들 은 다른 서브트리의 것임을 표현할 수 있는 계층적 구조적인 특징을 가지게 된다.

```
<STARS>
  <STAR>
    <NAME>Carrie Fisher</NAME>
    <ADDRESS>
        <STREET>Maple</STREET>
        <CITY>Hollywood</CITY>
    </ADDRESS>
    <MOVIES>
      <MOVIE>
        <TITLE>Star Wars</TITLE>
        <YEAR>1977</YEAR>
      </MOVIE>
    </MOVIES>
  </STAR>
</STARS>
```

(a) 예제 XML 문서

```
<!DOCTYPE Stars [
   <!ELEMENT STARS (STAR*)>
   <!ELEMENT STAR (NAME, ADDRESS+, MOVIES)>
   <!ELEMENT NAME (#PCDATA)>
   <!ELEMENT ADDRESS (STREET,CITY)>
   <!ELEMENT STREET (#PCDATA)>
   <!ELEMENT CITY (#PCDATA)>
   <!ELEMENT MOVIES (MOVIE*)>
   <!ELEMENT MOVIE (TITLE, YEAR)>
   <!ELEMENT TITLE (#PCDATA)>
   <!ELEMENT YEAR (#PCDATA)>
]>
```

(b) 예제 DTD 문서

그림 1. 예제 XML 문서와 DTD 문서

3.2. 색인 테이블

본 논문에서는 넘버링 방법을 이용하고, 세분화시켜 ElementType, PathIndex, Term, TermIndex로 4개의 테이블로 구성하여 처리한다.

3.2.1 ElementType 테이블

각 엘리먼트의 이름이 ElementType 테이블의 etName 필드의 값이 되고, Path 는 각각의 엘리먼트들의 계층구조를 Path로 표현함으로서 자료를 검색할 때 더욱더 효율적으로 할 수 있으며 docID는 XML문서의 고유한 문서번호이고 etID는 순서, 범위, 레벨의 합이 된다. etID값을 다른 3개의 필드의 합으로 표현하는 이유는 etID 값만을 가지고도 그 다음에 오는 엘리먼트의 위치가 자식인지, 자손인지 다른 서브트리인지를 파악할 수 있게 하여 검색 결과에서 효율적인 처리를 하기 위함이다.

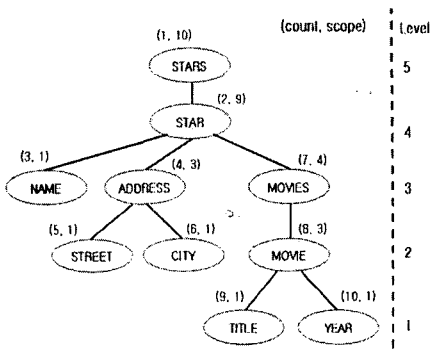
ElementType (etName,Path ,docID ,etID, level)

3.2.2 PathIndex 테이블

ElementType 테이블의 etID값을 이용한 PathIndex를 구성한다. ElementType Index에서는 DTD나 scheme에 대한 내용을 표현하므로 데이터 항목의 수도 적고, 디스크 상에서 차지하는 공간도 적다. 그리고 PathIndex 테이블에서 처리되는 데이터들은 정수값만을 저장하므로 이전의 연구들에서 사용하던 모든 경로의 데이터들을 저장할 필요가 없어 좋은 성능을 얻을 수 있게 되며, 그 구성은 다음과 같다.

PathIndex (elementID, docID, etID, count, scope, level)

PathIndex의 elementID 역시 ElementType 테이블의 etID와 같은 형태로 값을 구하게 되는데, 이것 역시 elementID값과 count 값을 통하여 엘리먼트들간의 구조적인 관계를 표현할 수 있도록 하기 위함이다.



(a) DTD 문서에 대한 트리 구조

Element Type	Path	Doc ID	Et ID	Count	scope	level
STARS	/STARS	1	16	1	10	5
STAR	/STARS/STAR	1	15	2	9	4
NAME	/STARS/STAR/NAME	1	7	3	1	3
ADDRESS	/STARS/STAR/ADDRESS	1	10	4	3	3
STREET	/STARS/STAR/ADDRESS /STREET	1	8	5	1	2
CITY	/STARS/STAR/ADDRESS /CITY	1	9	6	1	2
MOVIES	/STARS/STAR/MOVIES	1	14	7	4	3
MOVIE	/STARS/STAR/MOVIES /MOVIE	1	13	8	3	2
TITLE	/STARS/STAR/MOVIES /MOVIE/TITLE	1	11	9	1	1
YEAR	/STARS/STAR/MOVIES /MOVIE/YEAR	1	12	10	1	1

(b) DTD 문서에 기반한 엘리먼트 타입의 ID 값들

그림 2. 넘버링 알고리즘

3.2.3 Term 테이블, TermIndex 테이블

본 논문에서는 XML 검색을 처리하기 위해 텍스트 기반 검색에서 일반적으로 쓰이는 역색인[15]기법을 적용한다. 역색인은 단어와 출현 빈도의 쌍으로 이루어지는데, 단어마다 출현빈도를 직접 표현하게 되면 디스크 상에 차지하는 공간이 증가하게 되므로, 이를 감소시키기 위하여 Term 테이블과 TermIndex 테이블로 구분하

였고 그 구성은 다음과 같다.

Term (term, termID)
TermIndex (termID, docID, elementID, count)

4. 질의처리 방법

4.1 포함 질의

본 논문에서는 XML 문서의 내용을 검색하기 위한 방법으로 엘리먼트들(elements), 애트리뷰트들(attributes), 그리고 그것들의 내용을 이루는 텍스트 단어들간의 포함관계에 기반을 둔 포함질의를 이용한다.

4.1.1 직접 포함 질의

엘리먼트, 애트리뷰트, 텍스트를 포함한 노드들 간의 포함관계가 직접 포함관계 (부모-자식 관계)로 이루어 진 질의

질의 1 : */STARS/STAR/ADDRESS/CITY/'Hollywood'*

```
SELECT T1.docID
FROM ElementType ET,PathIndex P,Term T,TermIndex T1
WHERE ET.Path ='/STARS/STAR/ADDRESS/CITY'
   AND T.term = 'Hollywood' AND ET.etID = P.etID
   AND T1.elementID=P.elementID AND T1.termID =T.termID
```

4.1.2 간접 포함 질의

엘리먼트, 애트리뷰트, 텍스트를 포함하는 노드들 간의 포함관계가 간접 포함관계 (조상-자손 관계)로 이루어진 질의

질의 2 : */STARS//NAME//'Carrie'*

4.1.3 완전 포함 질의

노드간의 포함관계가 완전 포함 관계로만 이루어진 질의

질의 3 : *//STREET='oak'*

4.1.3 근접 질의

노드 사이의 거리에 대한 근접 정도에 따른 결과를 목적으로 하는 질의

질의 4 : *Distance('oak', 'Hollywood') <= 5*

4.2 포함질의를 확장한 계층처리 질의

본 논문에서는 구조적 넘버링 색인방법을 통하여 자손 노드에서 상위의 조상 노드를 찾아 가거나 조상 노드에서 하위의 자손 노드들을 찾아가는 질의를 처리 한다.

질의 5 : *//'Empire'*

```
SELECT *
FROM ( SELECT A.COUNT,A.LEVEL,A.ELEMENT_ID
       FROM   PathIndex A,Term B,TermIndex  C
       WHERE  B.TERM  = 'Empire'
       AND    B.TERM_ID  = C.TERM_ID
       AND C.ELEMENT_PATH_ID = A.ELEMENT_PATH_ID ) T1,
PathIndex T2, ElementType T3
WHERE  T1.LEVEL < T2.LEVEL  AND T1.COUNT > T2.COUNT
AND    T1.ELEMENT_ID < T2.ELEMENT_ID
AND    T2.ELEMENT_TYPE_ID = T3.ELEMENT_TYPE_ID
```

5. 성능평가

본 논문에서는 제안한 방법의 성능 평가를 위해 앞 절에 있는 4개의 질의를 이용한 수행시간을 XML 용량별로 비교하고 이를 그림 3에 보이고 있다. 비교대상은 본 논문에서 제안한 방법과 유사한 2테이블 방법과 비교를 했으며 그림3의 결과와 같이 본 논문에서 제안한 방법이 2-테이블 방법에 비해 좋음을 알 수 있다. 2-Table방식이 조인 수는 적지만 상대적으로 큰 테이블 간의 조인으로 오버헤드가 발생한다. 2-Table방식은 경로 길이가 길어지는 경우에는 조인수가 늘어나고 경로길이가 작아지는 경우에는 조인수가 작아지지만 본 논문에서 제안한 방식은 경로길이가 길어지던 짧아지던 경로정보를 가지고 있는 테이블이 존재하기 때문에 항상 조인이 일정함을 알 수 있다. 또한 본 논문에서 제안한 방법은 질의5에서 처럼 포함질의를 확장하여 계층구조처리 쿼리도 가능하다.

XML size	Query	2-TABLE 방법	제안한 방법(4-TABLE)
1MB	질의 1	70msc	50msc
	질의 2	75msc	53msc
	질의 3	88msc	60msc
	질의 4	100msc	67msc
3MB	질의 1	50msc	40msc
	질의 2	57msc	44msc
	질의 3	66msc	50msc
	질의 4	78msc	54msc
5MB	질의 1	220msc	139msc
	질의 2	224msc	140msc
	질의 3	300msc	170msc
	질의 4	410msc	216msc

그림 3. 질의 수행 시간 비교

6. 결론

본 논문에서는 XML 문서에 대한 포함질의를 효과적으로 처리하기 위해 RDBMS를 이용해서 XML문서의 DTD 정보를 저장하고 엘리먼트의 텍스트 정보를 역색인으로 저장하여 구조적 넘버링 알고리즘을 통하여 포함질의를 처리하는 방법을 제안 하였다. XML문서에 대한 역색인 정보뿐만 아니라 XML문서 자체도 RDBMS에 같이 저장될 경우 일반적으로 기업이 보유하고 있는 데이터의 양은 매우 크므로 저장 공간을 많이 필요로 할 것이다. 이를 위해 역색인과 XML문서를 RDBMS 상에서 따로 두지 않고 하나로 통합해서 관리하는 방법에 대하여 계속연구 하도록 하겠다.

[참고문헌]

[1] T. Bray, et al, "Extensible Markup Language (XML) 1.0 (SecondEdition), http://www.w3.org/TR/2000/REC-xml- 20001006

[2] C. Cheng, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman, "On supporting containment queries in relational database management system", ACM SIGMOD, Pages.425-436, 2001

[3] Chi-young Seo, Sang-won lee, "An Efficient Inverted Index Technique for xml Documents Using RDBMS, " Information and Software Technology , Vol.45, Pages.11-22, 2003

[4] Hector Garcia-Molina, Jeffrey D. Ul lman, Jennifer Wiom,"Database Syst ems : The Complete Book", Prentice Hall

[5] Silberschatz, Korth, Sudarshan, "Database System Concepts 4th Edition", Mc Graw Hill

[6] D.Florescu,D.Kossman,I,Mandlescu, "Integrating keyword search into XML query processing."Proceedings of the 9th International World Wide Web Conference, 2000