디렉토리 인덱스 : 관계형 데이타베이스 시스템에서 XML 데이타의 효과적인 질의 처리를 위한 인덱스 구조

백성호[0] 이석호
서울대학교 전기컴퓨터공학부
major@db.snu.ac.kr[0], shlee@cse.snu.ac.kr

Directory Index : Effective Index Structure for Query Processing of XML Data stored in RDBMS

SeongHo Baek[0] SukHo Lee
School of Electric and Computer Engineering, Seoul National University, Korea

요 약

XML이 웹상에서 데이타 교환의 표준으로 채택되면서 XML 데이타를 관계형 데이타베이스를 이용하여 저장하고 처리하는 것이 많이 연구되고 있다. 본 연구에서는 관계형 데이타베이스에 저장되어 있는 XML 데이타의 효과적인 질의 처리에 사용할 수 있는 인덱스 구조로서 디렉토리 인덱스를 제안한다. 디렉토리 인덱스는 정규 경로식 처리에 있어서 비트맵을 이용하여 조인 연산을 크게 줄여 처리 시간이 빠르며 인덱스의 갱신에도 효과적으로 대처할 수 있다.

1. 서 론

XML[1]데이타에 대한 질의어로 XQuery[2]가 표준화 되었다. XQuery는 XML 문서의 데이타를 기술하는 형식으로 XPath[3]를 사용한다. 이에 따라 XML 데이타를 관계형 데이타베이스에 저장, 처리하는 최근의 연구에서는 XPath의 효율적인 처리 방법에 관한 방법이 제시되고 있다.

XPath는 엘리먼트들의 부모-자식, 조상-후손과 같은 구조적인 정보를 담고 있다. 엘리먼트 단위로 데이타베이스에 저장하는 방식[4,5]에서는 기본적으로 엘리먼트를 저장한 테이블들 사이의 조인 연산을 통해서 구조적 정보를 파악하게 된다. 이 경우에 질의에 나타난 XPath 경로식을 구성하는 엘리먼트 수가 많을수록 더 많은 조인 연산이 수행되어 질의 성능이 현저히 나빠진다. 최근에 조인 연산을 줄이기 위해 XML 문서 내의 엘리먼트들 사이의 포함 관계를 이용한 기법[6,7]이 소개되었다. 이 방법에서는 사용자가 명시한 XPath 경로식의 형태에 따라 조인 연산의 수가 결정되어 XPath 경로식의 길이가 길 경우에는 결국 많은 수의 조인 연산을 피할 수 없다.

이러한 XPath 처리의 문제점 때문에 XML 문서에 등장하는 각 엘리먼트들의 XPath 경로식에 기반한 경로 인덱스 구조가 고안되었다[8,9,10]. 경로 인덱스들은 XML 문서의 구조적 정보를 이용하는데 XPath 경로식이 정규 경로 표현일 경우에는 효율적이지 못하며 저장된 XML 데이타의 갱신에 드는 비용이 크다.

본 연구에서는 정규 경로 표현의 경우에도 효율적인 처리가 가능하고 저장된 XML 데이타의 갱신에 큰 비용이 들지 않는 인덱스 구조로서 디렉토리 인덱스를 제안한다. 논문의 구성은 2절에서 디렉토리 인덱스의 구조 및 질의 처리를 설명하고 3절에서 결론 및 향후 과제를 제시한다.

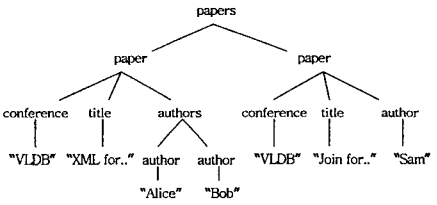

그림 1. 예제 XML 문서

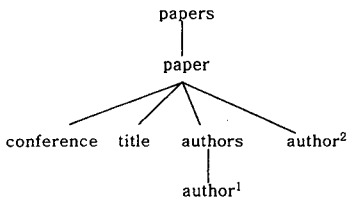

그림 2. 예제 문서의 구조

2. 본 론

2.1 트리 모델

디렉토리 인덱스는 XML문서의 구조적 정보를 추출하여 트리로 구성하는데 기본 트리 모델은 DataGuides[9], 1-Index[10]의 트리 모델과 유사하다. 트리의 노드는 XML문서내의 엘리먼트를 표현하고 노드 사이의 간선은 부모-자

식 관계를 의미한다. 그림 1의 XML 문서로부터 추출된 구조는 그림 2와 같다.

2.2 디렉토리 인덱스의 구조

디렉토리 인덱스는 디렉토리 트리, 디렉토리 검색 테이블 두 부분으로 구성 된다. 디렉토리 트리는 XML문서에서 추출된 구조적 정보를 나타낸 트리이다. 트리 노드는 엘리먼트를 표현하고 해당 엘리먼트가 저장된 물리적 위치에 대한 정보를 유지한다. 디렉토리 검색 테이블은 디렉토리 트리 내에서 특정 엘리먼트에 해당하는 노드들을 빠르게 찾기 위한 정보들을 유지한다.

2.2.1 디렉토리 트리

디렉토리 트리는 그림 2와 같은 문서 구조를 바탕으로 생성된다. 디렉토리 트리의 노드는 그림 3-a와 같은 구조이며 각 필드의 의미는 다음과 같다.

a. name – 노드의 이름
b. parent/child – 문서 구조상에서 부모/자식 노드를 가리키는 포인터
c. bitmap – 루트 노드로부터 이 노드까지의 경로에 대한 비트맵. 경로에 나타나는 모든 노드의 비트맵을 가지는 테이블이다.
d. pointers – 이 노드에 해당되는 실제 데이타들을 가리키는 포인터

그림 2의 'title'노드에 대한 디렉토리 트리의 노드는 그림 3-b와 같다.

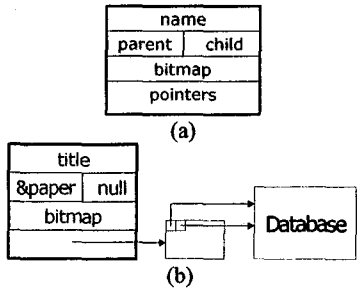

그림 3. 디렉토리 트리 노드

노드의 비트맵은 해당 노드가 XPath 경로식에 부합하는지 빠르게 검사하는데 사용된다. 특정 노드의 비트맵은 루트 노드로부터 해당 노드까지의 경로 상에 나타나는 각 노드들에 유일한 비트 문자열을 할당함으로써 구성된다. 예로 그림 2의 "title" 노드의 비트맵은 그림 4-a와 같다. "title" 노드의 루트노드까지의 경로는 "/papers/paper/title"인데 루트 노드부터 순서대로 유일한 비트 문자열을 할당한다. 비트맵에서 1이 나타나는 위치는 경로상에서 해당 노드가 나타나는 위치와 같으며 비트맵의 길이는 루트 노드까지 경로의 길이와 같다. 다른예로 그림 2에서 "author" 노드는 2개가 존재하는데 author[1] 에 대한 비트맵은 그림 4-b와 같다

/papers/paper/title	/papers/paper/authors/author
papers : 100	papers : 1000
paper : 010	paper : 0100
title : 001	authors : 0010
(a)	author : 0001
	(b)

그림 4. 노드의 비트맵

만약 XPath 경로식이 "/A/B/C/B"와 같이 노드가 반복되는 경우에는 해당 노드의 각 비트 맵 문자열을 OR 연산한것을 반복된 노드의 비트맵 문자열로 정한다. "/A/B/C/B"의 경우 "B"의 비트맵 문자열은 "0101"이 된다.

2.2.2 디렉토리 검색 테이블

디렉토리 트리 상에서 XPath 경로식 "/A//B//C"을 만족하는 노드 "C"를 찾는 경우, 노드 "A"를 찾은 후 후손으로 "B", "C"를 갖는지 검색하는 것보다 노드 "C"를 먼저 찾고 조상으로 "B", "A"를 갖는지 검색하는 것이 보다 작은 트리 탐색 공간을 갖는다. 이런 형태로 디렉토리 트리를 검색하기 위해서는 노드 "C"에 해당하는 디렉토리 트리 내의 노드들을 전체 트리의 탐색 없이 찾을 수 있어야 효율적이다.

디렉토리 검색 테이블은 디렉토리 트리 내의 각 노드들의 위치를 유지하는 테이블로서 구조는 그림 5와 같다.

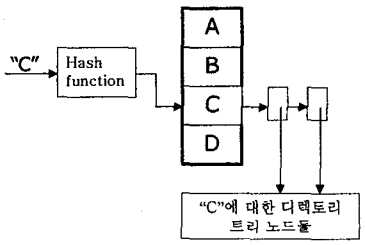

그림 5. 디렉토리 검색 테이블

디렉토리 트리 내에 동일한 노드가 N번 나타난다면 N개의 각각의 위치에 대한 포인터가 해당 노드의 이름과 연관된다.

2.3 질의 처리

질의 처리는 질의에 나타난 경로식의 마지막 엘리먼트를 이용하여 디렉토리 검색 테이블을 검색한 후 엘리먼트에 해당하는 디렉토리 트리의 노드들을 찾는 것으로 시작한다. 예로 질의에 나타난 경로가 "/A//B/C"라면, 먼저 "C"를 이용하여 디렉토리 검색 테이블에서 "C"에 해당하는 디렉토리 노드를 얻는다. 해당하는 디렉토리 트리 노드를 얻으면 각 노드에 대하여 노드의 비트맵을 이용, 해

당 노드가 경로식을 만족하는지 검사한다. 디렉토리 트리 노드가 갖고 있는 비트맵들은 다음과 같은 성질을 가진다.

a. 노드의 비트맵에는 노드로부터 루트 노드까지의 경로 상에 나타나는 모든 노드의 비트맵 문자열이 존재한다.

b. 노드의 루트 노드까지의 경로 상에 나타나는 모든 노드의 집합을 S_N ,노드 N 의 비트맵 문자열을 N_b 라 할 때,

$A \in S_N, B \in S_N, A \neq B$ 이면

$A_b \neq B_b$, $A_b \,\&\, B_b = 0$ 이다.

c. 노드의 경로 상에 나타나는 두 노드 A , B 에 대하여 $A // B$ 의 관계가 있으면 A, B 의 각 비트맵 문자열 A_b , B_b 와 비트맵 문자열의 길이 n 에 대하여 아래 1을 만족한다.

1. A_b 에서 1의 값을 가지는 비트 위치 중 가장 왼쪽의 비트 위치가 i 이라면, A_b, B_b 의 부분 문자열 $A_b[i,n], B_b[i,n]$ 에 대하여,

($A_b[i,n]$ 의 이진 값 $\geq B_b[i,n]$ 의 이진 값)

and ($B_b[i+1,n]$ 의 이진값 $\neq 0$)

A / B 의 경우에는 아래 2도 만족한다.

2. $A_b[i,n] \,\&\, (B_b[i,n] << 1) \neq 0$

위 3가지 성질을 이용하면 질의로 주어진 XPath 경로식에 디렉토리 테이블을 통해 얻은 디렉토리 트리 노드가 만족하는지 쉽게 검사할 수 있다. 만족되는 것으로 판명된 노드는 노드의 pointer 필드를 통해 데이타베이스에 저장된 값을 얻는다. 디렉토리 트리 노드에서 XPath 경로식을 평가하는 알고리즘은 부록으로 싣는다.

3. 결론 및 향후 과제

본 논문에서는 효과적이고 빠른 XPath 경로식 평가를 위한 인덱스 구조체로서 XML 문서의 구조에 기반한 디렉토리 인덱스를 제안하였다. 디렉토리 인덱스는 [9,10]의 인덱스와 달리 XPath 경로식 평가에 트리의 탐색이 필요하지 않다. 디렉토리 인덱스의 디렉토리 검색 테이블은 XPath 경로식을 만족하는 노드를 찾기 위해 조사해야 할 노드의 범위를 크게 줄여주며 디렉토리 트리 노드의 비트맵 연산은 XPath 경로식을 만족하는지 검사하기위한 불필요한 트리 탐색을 줄여준다. XML 문서의 크기에 비해 디렉토리 인덱스의 크기는 상대적으로 작으며 디렉토리 트리 노드의 비트맵은 해당 노드의 경로에만 기반하여 생성되므로 경로상에 존재 하지 않는 노드들의 갱신 연산으로 인한 추가작업이 필요치 않다. 향후 과제로 디렉토리 인덱스의 최적화된 구현 및 성능 평가가 필요하다.

-부록: XPath 경로식 검사 알고리즘

```
Query Path = a1N1a2N2...anNn
Axis = '/' or '//'
ai = Axis, Ni = Node, 0 < i ≤n
BM(Ni) = Ni의 비트맵
Check_Axis(BM(Node1), BM(Node2), Axis)
 : 비트맵의 성질 a,b,c를 이용하여 Node1과 Node2가
Axis관계를 만족하는지 검사하는 함수

function match(Path: Query Path)
{
  BM(Nn) = 0000....01;
  if( a1 == "/" ) BM(N1) = 100...0;
  bitmap = BM(Nn);
  for( i = n; i > 0; i-- )
  {
    if( Check_Axis(BM(Ni-1), bitmap, ai) == FALSE ) then
      return NOT_MATCH;
    if( ai == '/' ) then
      bitmap = BM(Ni-1) & (bitmap << 1);
    else bitmap = BM(Ni-1);
  }
  return MATCH_SUCCESS;
}
```

참고 문헌

[1] T. Bray, J. Paoli, C.M. Sperberg-McQueen and E. Maler. *Extensible Markup Language(XML) 1.0 second edition W3C recommendation, Technical Report REC-xml-20001006*, WWW Consortium, October 2000.

[2] 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.

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

[4] D. Florescu and D. Kossmann. *A performance evaluation of alternative mapping scheme for storing XML data in a relational database.* INRIA Technical Report 3684, 1999.

[5] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, J. Naughton. *Relational Databases for Querying XML Documents : Limitaion and Opportunities.* Proc. of the 1999 VLDB Conference, September 1999.

[6] C. Zhang, J. Naughton, D. Dewitt, Q. Luo, G. Lohman. *On Supporting Containment Queries in Relational Database Management Systems.* Proc. of the 2001 ACM SIGMOD Conference, May 2001.

[7] Q. Li and Bongki Moon. *Indexing and Querying XML Data for Regular Path Expressions.* Proc. of the 27th VLDB Conference, 361-370, September 2001.

[8] B. Cooper, N. Sample, M. Franklin, G. Hjaltason, and M. Shadmon.. *A Fast Index for Semistructured Data.* Proc. of the 27th VLDB Conference, January, 2001.

[9] R. Goldman and J. Widom. DataGuides: *Enabling Query Formulation and Optimization in Semistructured Databases.* Proc. of the 23rd VLDB Conferences, 436-445, August, 1997.

[10] T. Milo and D. Suciu. *Index Structures for Path Expressions.* Proc. of 7th ICDT, 277-295, January 1999.