웹에서 정규경로 표현식을 포함한 XML 질의의 캐쉬를 이용한 처리*

박정기° 강현철

중앙대학교 컴퓨터공학부

jkpark@dblab.cse.cau.ac.kr hckang@cau.ac.kr

Cache-Answerability of XML Queries in Regular Path Expressions on the Web

Jungkee Park° Hyunchul Kang

School of Computer Science and Engineering

Chung-Ang University

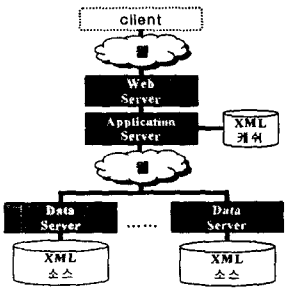
요 약

웹의 확산과 더불어 웹 페이지 검색의 성능 즉, 빠른 응답시간과 확장성(scalability)은 각 웹 사이트의 절대적 평가 기준이 되었다. 웹 응용은 일반적으로 불특정 다수를 대상으로 하기 때문에 확장성 또한 주요 성능의 척도가 된다. 이와 같은 웹 사이트 성능을 담보하기 위한 대표적 요소기술이 웹 캐싱이다. 본 논문은 웹 상에서 XML 데이터베이스 기반의 웹 응용(XML database-backed web application)을 위한 응용서버의 XML 캐쉬를 이용하여 주어진 XML 질의를 변환, 처리하는 기법과 구현에 관한 것으로 XPath의 경로표현식 중 가장 중요한 세가지 기능인 조건을 명시하는 필터 연산자, 부모-자식 관계를 나타내는 경로 연산자(/), 그리고 조상-후손 관계를 나타내는 경로 연산자(//)를 연구 범위로 하였다. [2]에서는 조상-후손 관계를 나타내는 경로 연산자(//)가 없는 경우에 경로표현식으로 주어진 XML 질의를 캐쉬를 이용하여 변환, 처리하는 알고리즘을 제시하였는데 본 논문에서는 [2]의 알고리즘을 확장하여 경로 연산자(//)가 지원되도록 하였다. 조상-후손 경로 연산자(//)로는 정규경로 표현식(regular path expression)을 나타낼 수 있는데 이는 스키마가 불확실한 반구조적 데이터인 XML 데이터에 대한 질의 표현에 유용하다. 제시된 알고리즘에서는 DTD를 이용하여 경로 정보를 얻어 처리함으로써 주어진 질의를 캐쉬와 하부 XML 소스에 대한 질의로 변환하였다. 이 알고리즘을 바탕으로 관계 DBMS를 이용하여 구현된 시스템으로 실제 웹 상에서 성능 실험을 수행하였다. 성능 실험 결과 정규 경로 표현식을 포함하는 XML 질의에 대해서도 웹에서 캐쉬를 이용한 처리가 효율적임을 확인하였다.

1. 서론

웹의 확산과 더불어 웹 페이지 검색의 성능 즉, 빠른 응답시간과 확장성(scalability)은 각 웹 사이트의 절대적 평가 기준이 되었다. Zona Research의 연구에 따르면 웹 페이지 검색 응답시간이 7초 이하일 때 사용자의 이탈율은 2%에 불과하지만 12초 이상이 되면 무려 70%를 넘어간다고 한다 [1]. 웹 응용은 일반적으로 불특정 다수를 대상으로 하기 때문에 확장성 또한 주요 성능의 척도가 된다. 이와 같은 웹 사이트 성능을 담보하기 위한 대표적 요소 기술이 웹 캐싱이다.

XML이 웹 상에서 데이터 교환의 표준으로 부각되기 시작한 이래로 웹에서 XML 문서가 차지하는 비중이 점점 커지고 있으며 가까운 미래에는 대부분의 웹 문서가 XML로 기술될 것으로 예상되고 있다. 본 논문은 (그림 1)과 같이 웹 상에서 XML 데이터베이스 기반의 웹 응용(XML database-backed web application)을 위한 응용서버의 XML 캐쉬를 이용하여 주어진 XML 질의를 변환, 처리하는 기법과 구현에 관한 것으로 XPath의 경로표현식 중 가장 중요한 세가지 기능인 조건을 명시하는 필터 연산자, 부모-자식 관계를 나타내는 경로 연산자(/), 그리고 조상-후손 관계를 나타내는 경로 연산자(//)를 연구 범위로 한다. 조상-후손 경로 연산자(//)로는 정규경로 표현식(regular path expression)을 나타낼 수 있는데 이는 스키마가 불확실한 반구조적 데이터인 XML 데이터에 대한 질의 표현에 유용하다. XML 캐쉬를 규정하는 XPath 경로표현식 및 질의의 XPath 경로표현식에 정규경로 표현식이 포함되어 있을 경우와 그렇지 않은 경우의 조합 네가지 모두에 대해 캐쉬를 이용한 XPath 질의의 변환이 가능하도록 본 논문에서는 조상-후손 경로 연산자(//)가 없는 경우의 기존 알고리즘[2]을 확장한 것을 제시한다.



(그림 1) XML 데이타베이스 기반 웹 응용을 지원하는 다계층 구조

본 논문의 구성은 다음과 같다. 2절에서는 확장된 질의 변환 알고리즘에 대해 기술하고, 3절에서는 새로운 알고리즘으로 구현된 시스템을 이용하여 실제 웹 상에서 수행한 실험 결과를 기술한다. 4절에서는 결론을 맺고 향후 연구 과제를 제시한다.

2. 정규경로 표현식을 포함하는 XML 질의의 캐쉬를 이용한 변환 알고리즘

* 본 논문은 정보통신부의 정보통신기초기술연구지원사업(정보통신 연구진흥원)으로 수행한 연구결과입니다 (과제번호: 03-기초-0100).

본 논문에서는 XML 캐쉬로 시맨틱 캐싱을 고려하고 XPath로 정의된 캐쉬는 실체뷰로 관리된다. XML 질의 처리에 실체뷰를 이용하는 유형은 질의 결과를 실체뷰에서만 가져오는 경우(MV: Materialized View Only)와 일부 결과를 실체뷰에서 가져오고 실체뷰에 없는 나머지 부분을 하부 XML소스에서 가져오는 경우(BD+MV: Both Base Data and Materialized View), 그리고 실체뷰에 결과가 없어 하부에서만 가져오는 경우(BD: Base Data Only)로 나눌 수 있다.
XML 질의 처리에 실체뷰를 이용하려면 (1) 실체뷰를 이용할 수 있는지, 있으면 어떤 형태로 이용하게 되는지를 판별하는 유형 결정과 (2) 판별된 유형에 따라 실체뷰와 하부 XML 소스에 대한 질의 변환을 수행해야 한다.

2.1 실체뷰를 이용한 XML 질의처리 유형결정 알고리즘

'/'만 포함되어 있는 질의는 path가 한개만 존재 하지만 '//'가 포함되어 있는 질의는 해당 path가 여러개 존재한다. '//'가 포함되어 있을 수 있는 질의 q를 역시 '//'가 포함되어 있을 수 있는 실체뷰 v를 이용하여 처리하는 유형을 알아내기 위해서는 다음 작업이 필요하다. DTD를 분석하여 q와 v가 가지고 있는 ① path들의 list와 ② 하부 path를 포함하는 집합과 ③ 조건을 나타내는 필터연산의 집합으로 결정한다.
예를 들어 '/a//d'의 경우, path들의 list는 '/a/b/d', '/a/c/d' 등으로 //의 부분에 나타날 수 있는 모든 경로를 열거하여 구한다. 하부 path를 포함하는 집합은 목적 엘리먼트인 d 하부의 path들까지 포함한 것인데 예를 들어 '/a/b'의 경우 그 집합은 {'/a/b', /a/b/d', '/a/b/d/e'}와 같이 d 이하 부분의 경로를 모두 열거하여 구한다. 조건 필터연산의 집합은 예를 들어 '/a/b[c="x" and d/e="y"]'의 경우 {'/a/b/c="x"', '/a/b/d/e="y"'}가 된다.

```
Check_Containment2(q,v)
{/*
 q : XML 경로표현식 질의
 v : XML 실체뷰 경로표현식 정의
*/
 q.pathset2=q.pathset
 if((q.pathset.remove(v.pathset))
 {
  if(q.pathset.size=0)
  {
   if(q.pathlist.size>=v.pathlist.size AND v.con⊆q.con) return(MV);
   if(q.pathset=v.pathset AND con⊃q.con)  return(BD+MV2);
   if(q.pathset≠v.pathset AND q.pathlist.size>=v.pathlist.size
     AND v.con⊃q.con)  return(BD+MV3);
  }
  else
  {
   if(q.pathlist.size>v.pathlist.size)  return(BD+MV4);
   else if(v.pathset.remove(q.pathset2))
   {
    if(v.pathset.size=0)  return(BD+MV1);
   }
  }
 }
 return(BD);
}
```

(그림 2) 확장된 Check_Containment 알고리즘

<그림 2>는 [2]의 '/'만 지원할 때 질의 처리 유형을 결정하는 알고리즘을 확장하여 '//'도 지원할 수 있는 알고리즘 Check_Containment2(q, v)를 기술한 것이다. 알고리즘에서 질의 q의 하부 path를 포함하는 집합을 q.pathset이라 표현하고 q의 path들의 list를 q.pathlist라 표현하였으며 질의 q의 조건을 나타내는 필터연산의 집합을 q.con으로 표현하였다.
q.pathset.remove(v.pathset)는 q.pathset에 대한 v.pathset의 차집합을 연산하는 것으로 만약 없어지는 부분이 있다면, 즉 둘 사이에 교집합이 존재한다면 true를 반환하고 q.pathset에는 차집합의 결과가 남게 된다. 교집합이 공집합이 되면 v는 질의 q의 결과에 기여하지 않음을 의미한다. q.pahtset, v.pathset, q.pathlist의 size(갯수크기)와 v.pathlist의 size로 실체뷰에서 질의 결과에 해당되는 부분과 그렇지 않는 부분이 구분 가능하며 이로서 처리 유형을 결정할 수 있다.

2.2 실체뷰를 이용한 XML 질의 변환 알고리즘

Check_Containment2()를 사용하여 질의 q의 실체뷰 v를 이용한처리 유형을 결정한 다음 (그림 3)의 Rewrite_path2()함수를 사용하여 실체뷰와 하부 XML 소스에 각각 제기할 질의 q'과 q''를 생성한다. 기존 [2]의 알고리즘 Rewrite_path()는 역시 '/'만이 지원되었다.(그림 3)의 알고리즘은 '//'도 지원한다.

```
Rewrite_Path2(q,v, type)
{
/*
q : XML 경로표현식 질의
v : XML 실체뷰 경로표현식 정의
type : Check_containment()에 의해 결정된 처리 유형
*/
 switch(type)
 {
  case MV:
  q' = "/mv"+v.elem+d_Path(q, v.path).path+Filter(and(q.con));
  q''= null;
  break;

  case BD+MV1:
  q' = "/mv"+v.elem+d_Path(q, v.path).path+Filter(and(q.con-v.con));
  q'' = q+backslash(d_Path(v, q.path).path)+Filter(and(q.con));
  break;

  case BD+MV2:
  q' = "/mv"+v.elem;
  q'' = q+Filter(and(q.con∪{not(and(v.con-q.con))} ));
  break;

  case BD+MV3:
  q' = "/mv"+v.elem+d_Path(q, v.path).path+Filter(and(q.con));
  q'' = v.path+Filter(not(and(v.con)))+d_Path(q, v)
  break;

  case BD+MV4:
  q' = "/mv"+v.elem+d_Path(q, v.path).path+Filter(and(q.con-v.con));
  q'' = q.c_prefix+backslash(m_Path(v.c_prefix+v.D_suffix.path, q.c_prefix,
        q.D_suffix.path))+Filter(and(v.con))+"/"+q.D_suffix;
  break;

  case BD:
  q' = null;
  q''= q;
  break;
 }
}
```

(그림 3) 확장된 Rewirte_Path 알고리즘

d_Path(q,v.path)는 질의 q의 pathlist에서 v.path 아래에 존재하는 부분만을 반환한다. 예를 들어 '/a//d'라는 질의 q에 대해 q의 pathlist에 '/a/b/d', '/a/f/d'가 포함될 때 v.path가 '/a/b'라면 d_Path(q,v.path)는 '/d'가 된다. backslash()함수는 BD+MV1과 BD+MV4에서 사용하는 것으로서 함수에 들어오는 path를 제외 연산자"\" [2]로 구분짓는 함수이다. 예를 들어 질의 q가 '/a/b'이고 실체뷰 v가 '/a/b/c' 이면 BD+MV1유형으로 q'= 'mv/c', q''='/a/b\c'가 된다 (mv에 대해서는 [2]를 참조). q''의 의미는 '/a/b'에 있는 내용을 검색하는 데 있어 '/a/b/c'내용은 제외한다는 것이다. 유형 BD+MV4는 제외 연산자가 중간에 사용되는 것으로 질의 q가 '/a//d'이고 실체뷰 v가 '/a/b/d'라면 q'= 'mv/d', q''='/a\b/d'로 된다. q''의 의미는 'a//d'를 검색하는 데 있어 '/a/b' 아래에 있는 '/a/b/d'는 제외한다는 것이다.
q''를 표현하기 위하여 c_prefix, D_suffix, m_Path()라는 표기가 사용되었다. c_prefix는 pathlist들의 공통적인 앞부분만을 반환하는 것이다. 예를 들어 path가 '/a//d'이고 pathlist에 '/a/b/d', '/a/f/d'가 있다면 c_prefix는 공통적인 앞부분인 '/a'가 된다. D_suffix는 '//' 이후부분을 나타내는 것으로 앞의 예에서는 '/d'가 된다. m_Path()는 그 파라미터 값으로 a.path, b.c_prefix, b.D_suffix가 들어온다. 이것은 a의 path의 앞부분에서 b.c_prefix인 부분과 뒷부분에서 b.D_suffix 부분을 제외한 중간부분을 반환한다. 이 함수를 사용하여 제외되는 중간부분을 구할 수 있다.

3. 성능 평가

본 절에서는 관계 DBMS 시스템을 기반으로 구현된 XML 저장 시스템을 대상으로 웹 상에서 본 논문의 질의 변환 알고리즘으로 실체뷰를 이용한 XML 질의 처리의 성능을 실제 실험을 통해 평가한 기초 결과를 기술한

다. 본 실험은, XML 데이터베이스 기반 웹 응용을 지원하는 웹 서버, 응용 서버, 데이타 서버의 다계층 구조를 대상으로, XML 소스는 웹 상의 원격 사이트에 존재하고 XML 실체뷰는 [4-5] 등의 연구에서와 같이 응용 서버에 캐쉬되는 환경을 대상으로 하였다.

3.1 개요

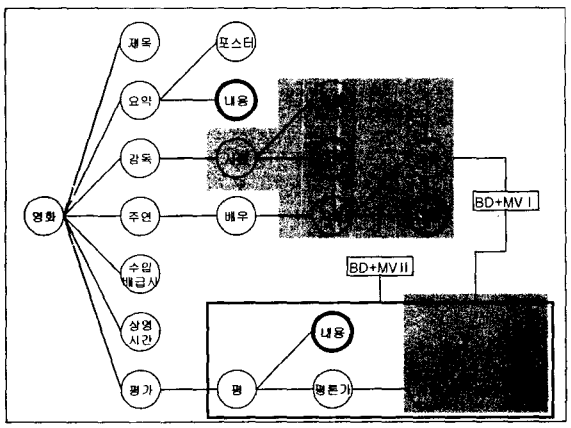
본 실험의 성능 척도는 웹 응용에서 XML 질의 처리를 수행할 때의 응답 시간이다. 응용 서버에 XML 질의가 제기된 시점부터 질의 결과 (관계 DBMS로부터 검색된 결과 셋을 태깅 처리한 XML 문서)가 응용 서버에 확보되기까지 걸린 시간으로, MV, BD+MV, 그리고 BD 유형의 XML 질의가 주어졌을 때, 이들 간에 처리 시간을 비교하였다. 실험 환경 및 데이터는 <표 1>과 같다.

<표 1> 실험 환경 및 데이타

항목		내용
데이터 서버	CPU	PentiumIV 1.6GHz
	메모리	512MB
	OS	Windows 2000 Server
	DBMS	Oracle 9i
응용 서버		데이터 서버와 동일
실험 데이터		확장된 "영화" XML 문서 문서당 평균 엘리먼트 개수 45개 문서당 평균 크기 1.5KB
웹의 네트워크 환경		2.8Mbps

<표 2> 실험에 사용된 조건

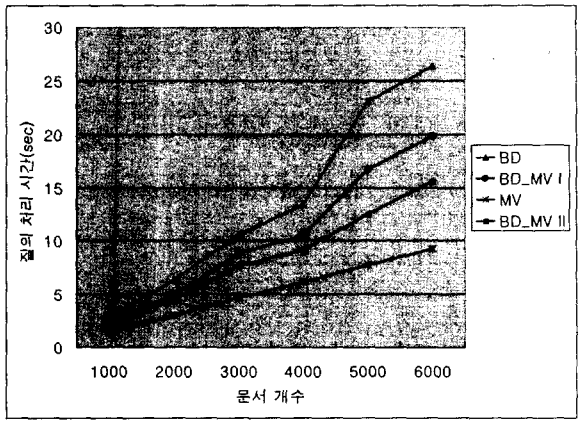
구분	경로 표현식	
질의	/영화//내용	크 기: 문서의 45.79% 노드 수: 문서의 26.67%
MV	/영화//내용	질의 결과와 100% 일치
BD+MV I	/영화//사람	크 기: 문서의 29.04% 노드 수: 문서의 46.67% 실체뷰의 결과 포함비율 • 크 기: 결과의 22.53% • 노드 수: 결과의 58.33% 결과의 실체뷰에서 포함율 • 크 기: 실체뷰의 35.53% • 노드 수: 실체뷰의 33.33%
BD+MV II	/영화/평가/평	크 기: 문서의 43.37% 노드 수: 문서의 53.33% 실체뷰의 결과 포함비율 • 크 기: 결과의 22.53% • 노드 수: 결과의 58.33% 결과의 실체뷰에서 포함율 • 크 기: 실체뷰의 35.53% • 노드 수: 실체뷰의 33.33%



(그림 4) 영화 DTD 및 실험의 질의/실체뷰 검색 범위

<표 2>는 실험에 사용하는 질의와 실체뷰들이 질의 결과에 기여하는 비율을 나태낸 것이다. 실험을 위해 "영화" XML문서를 사용하였다. (그림 4)는 실험에 사용된 "영화" 문서의 DTD를 Tree구조로 나타내고 질의와 실체뷰들이 포함하는 범위를 나태난 것이다.

3.2 실험 결과



(그림 5) XML 소스 문서 개수의 변화에 따른 성능 비교

(그림 5)는 <표 2>의 조건으로 문서의 개수를 1,000개부터 6,000개까지 1,000개씩 증가시키면서 질의를 처리한 실험 결과이다. BD+MV가 BD보다 우수하게 나타났으며 실체뷰로부터의 검색 비율이 증가함에 따라 BD+MV의 성능이 계속 향상되는 것으로 나타났다. 이는 BD+MV의 경우 XML 소스에 대한 접근과 실체뷰에 대한 접근이 웹 상의 서로 다른 사이트 상에서 병렬로 처리되어 검색비용이 적고, BD에 비해 통신 비용도 적기 때문이다. MV의 경우 검색과 통신비용의 이득으로 속도가 제일 빠르게 나타났다.

4. 결론

본 논문은 웹에서 정규경로 표현식을 포함하는 XML 질의를 관련 XML 캐쉬를 이용하여 처리하는 기법에 관한 것이다. 정규 경로 표현식을 구성하는 조상-후손 경로연산자(//)가 지원되지 않는 기존의 기법[2]을 확장한 Check_Containment2와 Rewrite_Path2 알고리즘을 제시하였다. 이들 새 알고리즘들의 핵심 이슈는 DTD를 분석하여 // 연산자가 포괄하는 경로에 대한 정보를 알아내어 Check_Containment2에서 그 path들의 list와 목적 엘리먼트의 하부 경로들의 집합을 구하여 캐쉬를 이용한 질의 처리 유형을 결정짓는 것과 Rewrite_Path2에서 정규경로 표현식을 포함하는 질의의 변환이다. 구현된 시스템으로 실제 웹 상에서 데이터베이스 기반 웹 응용을 위한 웹 서버, 응용 서버, 데이터 서버의 다계층 구조 하에서 응용 서버에 캐쉬를 둘 때 원격 XML 소스에 대한 XML 질의 처리의 성능을 평가하였다. 실험 결과 실체뷰를 사용하지 않는 BD 유형의 처리에 비해, 질의의 답을 실체뷰로부터 모두 얻는 MV 유형이나 질의의 부분 답을 실체뷰와 하부 XML 소스로부터 각각 얻어 통합하는 BD+MV 유형의 성능이 정규경로 표현식이 포함되는 경우에도 모두 우수하게 나왔다. 향후 연구 과제로 더욱 다양한 데이터 및 질의 셋에 대한 실험과 본 기법의 웹 상에서의 확장성(scalablilty) 평가 등이 수행되고 있다.

참고문헌

[1] Zona Research, http://www.zonaresearch.com/
[2] C. Moon et al "Processing XML Path Expressions Using XML Materialized Views," Lecture Notes in Computer Science, Vol 2712, pp. 19-37, 2003.
[3] A. Berglund et al., "XML Path Language (XPath) 2.0," http://www.w3.org/TR/xpath20/, Nov. 2002.
[4] V. Hristidis and M. Petropoulos, "Semantic Caching of XML Databases," Proc. Workshop on the Web and Databases, 2002.
[5] L. Chen and E. Rundensteiner, "ACE-XQ: A CachE-aware XQuery Answering System," Proc. Workshop on the Web and Databases, 2002.