한국지능시스템학회 논문지 2008, Vol. 18, No. 3, pp. 306-310

관계형 테이블을 이용한 W3C XQuery 변경 기능의 지원

W3C XQuery Update facility on SQL hosts

홍동권

Dong-Kweon Hong

계명대학교 컴퓨터공학과

요 약

XML 질의어의 표준으로 인정받고 있는 XQuery의 검색 기능의 확장으로 새로운 XML의 삽입, 삭제 기능에 대한 표준화가 진행되고 있다. XML 데이터베이스가 단순한 문서 관리의 기능에서 벗어나 기존 데이터베이스의 장점인 OLTP 기능까지 지원하려는 노력을 하고 있다. 본 논문은 XQuery 검색 기능을 관계형 환경에서 지원하기 위한 선행 연구의 결과에 XQuery 변경 기능을 추가하기 위한 연구의 결과로 1) XML을 저장하기 위한 테이블 구조, 2) 계층 구조를 저장하기 위한 번호 부여 방식, 3) 효율적인 검색 기능을 지원하기 위한 경로 사용의 장.단점, 4) XQuery 변경 구문의 SQL 변환 과정을 제시한다.

키워드 : XQuery 변경 기능, XML 저장. 관계형 데이터베이스. 계층 구조

Abstract

XQuery is a new recommendation for XML query. As an efforts for extending XQuery capabilities XML insertion and deletion are being studied and its standardization are going on. Initially XML databases are developed simply for XML document management. Now their functions are extending to OLTP. In this paper we are adding updating functions to XQuery processing system that is developed only for XQuery retrievals. We suggest the structure of tables, numbering schemes for hierarchical structures, and the methods for SQL translations for XQuery updates. .

Key Words : XQuery update, storage of XML, Relational database. hierarchical structure

1. 서 론

XQuery가 질의어로서 완전한 기능을 가지기 위해서는 XML에 대한 검색 기능 외에 동적 환경에 대한 요구 사항으로 인하여 입력, 삭제, 변경 기능이 필요하다. XML 관련 산업체에서의 계속적인 요구와 표준화 작업에 의해 XQuery update 기능에 대한 스펙이 정의되고 있는 중이다 [1, 2, 3, 4]. 기존의 관계형 모델이 OLTP (On-Line Transaction Processing)에 적합하게 개발되어 빈번한 데이터의 입출력에 좋은 성능을 보이는 것과는 달리 XML 모델은 계층적 구조와 데이터에 내재하는 순서 정보로 인하여 XML의 부분적인 변경, 삭제, 새로운 내용의 추가에 많은 어려움을 보이고 있다. 이는 데이터의 빈번한 입력, 변경, 삭제가 발생하는 환경에서는 관계형 모델이 더 적합하다는 사실을 입증하는 부분이다. 하지만 XML 모델이 동적인 환경에서 비효율적이지만 경우에 따라서는 변경 기능을 제공해야 하는 경우가 종종 발생한다 [3].

XML 데이터의 일부분을 변경하기 위한 변경 구문 및 기능에 대한 표준화 노력은 2000년 초반부터 꾸준히 있어 왔다. XML변경 기능을 위한 XML 질의의 선구자적인 표준화 노력은 기본적인 제안(working draft)[2]를 바탕으로 XML 문서 변경 기능을 고려하였으며, 가장 최근에는 XQuery 1.0에 XQuery의 확장으로 update 기능을 포함하려고 준비 중에 있으며 다음의 3가지 문서를 발표했다[1].

1. XQuery Update Facility Requirements (2005 working draft)
2. XQuery Update Facility (2007년 8월 working draft)
3. XQuery Update Facility Use Cases (2007 8월 working draft)

XQuery 변경 기능 표준화의 현재의 상태는 W3C의 Last Call Working Draft (2007년 8월)이므로 곧 표준안이 확정될 것으로 예측된다.

본 논문은 XQuery 검색 기능을 관계형 환경에서 지원하기 위한 연구의 결과에 XQuery의 변경 기능을 추가하기 위한 방법을 연구한다. XQuery 변경 기능을 위하여 관계형 테이블에 추가하여야 하는 정보, 계층 정보를 지원하기 위한 번호 부여 방식, 삽입 및 삭제 연산 시 변경하여야 할 정보, XQuery 변경 구문의 SQL 변환 방법을 제시한다. 본 논문의 구성은 다음과 같다. 먼저 2장에서는 본 논문에서 사용하는 XQuery를 지원하는 환경에 대해서 설명하고, 3장에서는 W3C의 XQuery 변경 기능과 형식을 알아보고 각각의 기능을 SQL 형식으로 변환하는 과정을 알아본다. 4장에서는 XQuery 변경 기능을 지원하기 위한 오버헤드와 문제점을 알아보고, 5장에서는 결론과 향후 연구계획을 설명한다.

2. XQuery 검색 지원 환경

본 논문의 XQuery 변경 기능의 추가에 사용하는

접수일자 : 2007년 10월 16일
완료일자 : 2008년 2월 4일

XQuery 처리 환경은 [그림 1]과 같이 관계형 DBMS의 상위 계층으로 만들어져있다[5].

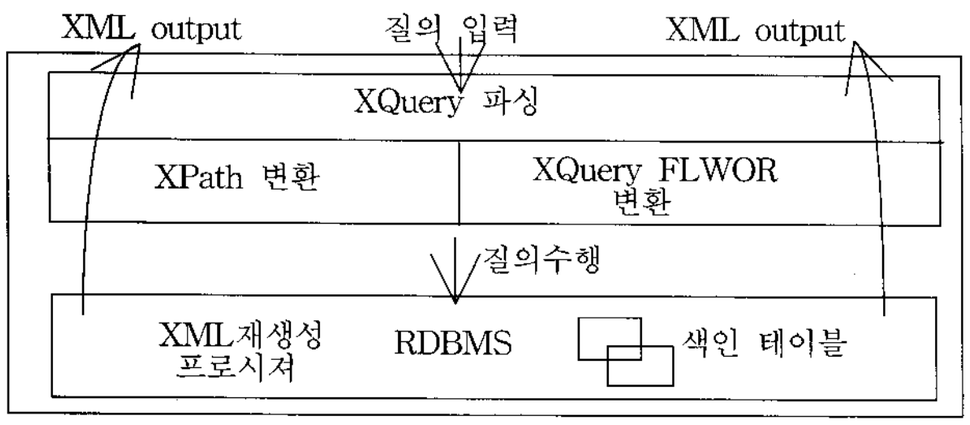

그림 1. XQuery 검색 시스템의 전체 구조도
Fig. 1. System structure of XQuery

사용자가 XQuery 검색 질의를 입력할 경우 먼저 이를 파싱하고 구문을 분석하여 내부적으로 트리 형태로 표현한다. 그리고 질의 구문이 XPath 혹은 XQuery FLWOR 식인지를 결정한 다음 XQuery를 SQL로 변환하여 관계형 DBMS에 처리를 요청한다. 마지막으로 처리된 SQL의 결과는 다시 XML 형태로 재생성 되어 사용자에게 보내진다.

XQuery 검색 질의를 지원하기 위한 관계형 데이터베이스의 색인 테이블은 [그림 2]와 같다. 앞으로 확장할 동적 환경의 특성에 따라 XQuery의 전문 검색은 지원하지 않는 것으로 가정하며, 전문 검색이 필요할 경우 최소한의 변경으로 쉽게 적용 가능하다. 우선 여러 개의 XML 문서들을 쓰임새에 따라 분류하여 하나의 컬렉션 단위로 관리한다. 이러한 각각의 컬렉션에는 서로 다른 색인 테이블 집합이 생성되며 모든 질의는 사용자가 선택한 현재 컬렉션(current collection)에 대해 이루어진다.

```
Collections (collection_id, Cname)
Cname_location(doc_id, path_id, path, depth, ath_cnt)
Cname_element(doc_id, e_id, name, sibord, path_id,
                    p_id, value, numbering)
Cname_attribute(doc_id, a_id, e_id, a_name, a_value)
```

그림 2. XQuery 지원을 위한 관계형 테이블
Fig. 2. Relational tables for XQuery

Collection 테이블은 전체 시스템에서 1개가 존재하며 사용자가 생성한 컬렉션의 이름과 식별자(id)가 생성된다. 여기서 Cname은 사용자가 생성한 컬렉션의 이름을 뜻하며, 각 컬렉션에는 XML 문서의 위치 정보를 나타내는 Cname_location, 엘리먼트들의 정보를 나타내는 Cname_element, 애트리뷰트의 정보를 나타내는 Cname_attribute 테이블로 구성된다. 이때 Cname_element 테이블의 numbering 컬럼에는 XQuery 검색 질의를 효과적으로 지원하기 위하여 dewey order 번호 부여 방식의 순서 값이 저장된다[4]

3. XML 질의의 변경 기능

XML 질의에서 변경 기능의 형식에 대하여 심각하게 고려한 내용은 XML:DB[2]가 있다. XML:DB의 기본적인 제안에 있는 기능 중에서 핵심 기능은 XQuery와 거의 같다. XML:DB의 내용을 바탕으로 하여 작성된 W3C의 XML 질의의 변경 기능에 대한 표준안은 XQuery의 형식을 따르지만 그 기능은 [표 1]과 유사하다. XQuery의 변경 기능도 새로운 노드의 추가, 기존 노드의 삭제, 노드의 속성 변경, 새로운 노드의 복제 등으로 크게 분류할 수 있다. W3C XQuery 변경 기능은 [표 1]과 비슷한 변경 기능을 지원하지만 그 형식은 매우 다르다. XQuery의 변경 기능의 형식과 특징을 XQuery 변경 기능 사용 예 (use cases) 문서에 있는 변경 기능 중에서 Q1에서 Q9까지의 예제를 살펴보면 다음과 같은 형식을 가진다 [1].

```
insert nodes
  <user_tuple>
    <userid>U07</userid>
    <name>Annabel Lee</name>
  </user_tuple>
into doc("users.xml")/users
```

그림 3. Q1: users에 새로운 노드를 첨가
Fig. 3. Q1: insert new node to users

```
let $uid := doc("users.xml")/users/user_tuple[name="Annabel
Lee"]/userid
return
  insert nodes
    <bid_tuple>
      <userid>{data($uid)}</userid>
      <itemno>1001</itemno>
      <bid>60</bid>
      <bid_date>1999-02-01</bid_date>
    </bid_tuple>
  into doc("bids.xml")/bids
```

그림 4. Q3: 특정 위치에 문장을 삽입
Fig. 4. Q3: insert para to a specific position

```
let $user := doc("users.xml")/users/user_tuple[name=
                                    "Annabel Lee"]
return
if ($user/rating)
then replace value of node $user/rating with "B"
    else insert node <rating>B</rating> into   $user
```

그림 5. Q4: XML 내용의 변경
Fig. 5. Q4: modification of XML

```
insert nodes
 <comment>This is a bargain !</comment>
as     last     into                          doc("i-
tems.xml")/items/item_tuple[itemno=1002]
```

그림 6. Q7: XML 노드를 마지막 자식으로 입력
Fig. 6. Q7: Insertion of XML node as last child

4. 관계형에서 변경 기능의 지원

지금까지 대부분의 관계형 데이터베이스를 활용한 XQuery 지원 시스템들의 연구에서는 효과적인 검색을 위한 인덱스의 구성과 XQuery 검색 기능을 SQL로 변환하는데 중점을 두었다 [6, 7, 8]. 이들 연구 중에서 XML 문서의 변경을 처리하는 방법을 제안한 것을 보면 XML 문서 내의 일부분이 변경될 경우 이미 생성되어진 인덱스를 재생성 하거나 [3], 또는 각 노드의 위치 정보에 일정한 여백을 비워둠으로써 추후 발생할 삽입, 변경에 대해서 인덱스를 재생성하는 문제를 해결하려고 했다. 그러나 XML 문서에서 변경이 자주 발생하면 예비로 마련해 둔 번호 여백이 전부 소모되기 때문에 결국은 인덱스를 재 생성해야 하는 문제가 발생한다. 따라서 이러한 기존의 기법들은 XML 문서의 변경을 완전히 처리하는 기법이라고는 볼 수 없다. 전역 번호 부여 방식 (global numbering scheme)[9]는 XML 문서의 각각의 구성 원소에 정적 방식으로 적절한 번호를 부여하여 그 번호를 이용하여 구성 원소들 사이의 계층 관계를 쉽게 파악하게 하는 대표적이 방식이다. 하지만 XML 변경이 발생하여 새로운 구성 성분이 XML 문서의 중간에 삽입됨에 따라 직렬화된 (serialized) 문서의 관점에서는 삽입 위치 다음에 존재하는 모든 내용들의 위치는 전부 이동 된다. 정적인 번호 부여 방식을 이용하여 위치가 변화된 구성 원소에 새로운 번호를 부여하기도 어려울 뿐 아니라 적절한 번호를 모두 소모하여 XML 문서의 모든 구성 원소에 새로 번호를 부여해야 하는 일도 발생할 수 있다 [9, 10]. Dewey order 번호 부여 방식 [9]은 XML 문서의 원소에 계층적인 번호를 부여하여 원소 사이의 순서 정보를 표현하는 방식이다. 하지만 이 방식도 새로운 원소의 삽입, 삭제가 발생할 경우 번호의 조절이 필요하다.

다른 방법은 동적 번호 부여 방식을 사용하는 것이다. XML 문서의 일부분이 새로 삽입되거나 원소의 삭제가 발생하는 동적 XML 문서에서 번호 부여 방식에 대한 연구에서 가장 대표적인 연구는 ORDPATH 방식이다 [11]. 이 방식은 마이크로소프트 SQL 서버에서 채택한 방식으로 XML 문서에 새로운 입력이 발생해도 기존의 번호를 수정할 필요한 없는 아이디어를 사용하고 있다. 초기에 각 노드에 번호를 부여할 때 모든 번호에 전부 홀수를 사용하여 dewey 번호 부여 방식으로 번호를 부여한다. 새로운 삽입이 발생하여 2개의 노드 사이에 새로운 노드가 삽입될 경우 새로운 삽입되는 노드의 번호에 짝수를 사용한 끼어넣기 (careting) 방식을 사용한다. 이 방식은 기존의 노드에 새로운 번호를 다시 부여할 필요 없이 새로운 노드를 삽입할 수 있는 장점이 있다.

본 논문에서는 [그림 2]의 테이블 구조의 numbering 컬럼을 dewey 방식을 사용하였다. XQuery 검색 질의 SQL 변환 연구에서 전혀 문제가 없었으나 삽입, 삭제가 발생할 경우 새로운 번호의 부여에 의해 XML 내부의 원소 일부분이 번호를 새로 부여 받아야 하는 문제점이 발생한다. 본 논문에서는 삽입, 삭제의 문제점을 해결하기 위하여 동적 번호 부여 방식인 ORDPATH 개념을 적용한 새로운 번호 방식 ordno로 변경하여 Cname_element 테이블을 다음의 [그림 7]로 바꾼다.

이렇게 새로운 ordno 부여 방식을 사용할 경우 XML 문서의 초기 입력 시 문서의 각 노드는 홀수 번호만을 사용한 dewey 번호를 부여 받게 된다. 이 때 ordno는 기존의 dewey 번호와 차이점이 없으므로 기존의 XQuery 검색 기능의 SQL 변환 방식은 전혀 영향을 받지 않는다 [5].

Cname_element(doc_id, e_id, name, sibord, path_id, p_id, value, ordno)

그림 7. ORDPATH 기법을 적용한 ordno의 사용
Fig. 7. Use of ORDPATH ordno

4.1 새로운 노드의 추가

새로운 노드를 삽입하는 경우 삽입된 노드의 ordno 번호 생성은 [그림 8]과 같다.

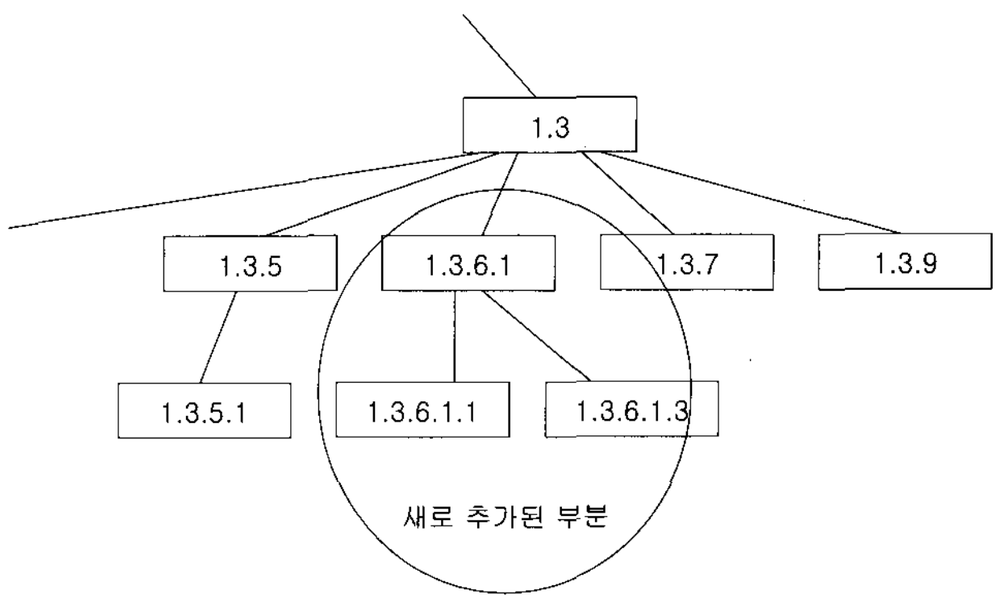

그림 8. 끼워넣기 기법을 이용한 ordno 번호의 생성
Fig. 8. Creation of ordno using careting

4.1.1 새로운 번호의 부여

새로운 노드의 삽입 위치에 따라 노드의 적절한 ordno가 새로 부여된다. 따라서 새로운 노드의 삽입 시 먼저 삽입 위치의 양쪽 형제 노드를 찾아서 그 번호를 확인해야 한다. 적절한 ordno 번호의 여유가 있는 경우 (예를 들어서 1.3.5와 1.3.9 사이에는 ordno 1.3.7이 적절한 번호이다. 하지만 [그림 8]과 같이 1.3.5와 1.3.7 사이에 같은 레벨을 가진 적절한 ordno 번호의 여유가 없다. 따라서 이 같은 경우에는 끼워넣기 (careting) 기법을 사용하여 1.3.6.1의 ordno 번호를 만들어 내며 그것의 자식 노드는 각각 1.3.6.1.1과 1.3.6.1.3의 ordno 번호를 가지게 된다. 이 방식을 사용할 경우 새로 삽입되는 노드 외는 XML 트리에 존재하는 다른 어떤 노드도 ordno 번호를 새로 부여 받을 필요가 없다.

Cname_element와 Cname_attribute 테이블에 분석 과정에서 생성된 엘리먼트 정보와 애트리뷰트 정보를 입력한다. 같은 부모를 가진 형제 노드들 사이의 순서 정보 (document order)를 확인하기 위하여 문서 전체의 순서 정보를 가지고 있는 ordno만을 사용하는 방법과, 형제 정보를 가지고 있는 새로운 번호 sibord를 사용하는 방법이 있다. 만약 ordno만을 이용하여 형제들의 순서 정보를 같이 표현한다면 [그림 8]과 같은 끼워넣기가 필요하며, 만약 sibord를 추가적으로 사용하는 경우 ordno는 부모, 자식의 정보만 나타내면 되며, 각 형제 노드들의 sibord의 값을 변경하기 위한 절차가 필요하다.

```
// 추가될 엘리먼트 까지의 경로가
// 이미 존재하는지 검사
SELECT PATH_CNT INTO L_PATH_CNT
FROM Cname_location
WHERE PATH  = IN_PATH;
if (L_PATH_CNT) // 이미 존재하는 경우
{
    UPDATE Cname_location
    SET PATH_CNT = PATH_CNT +1
    WHERE PATH = IN_PATH;
}
else //새로운 경로의 추가
{
  INSERT INTO Cname_lcoation
  VALUES (doc_id, path_id, IN_PATH, depth, 1);
}
```

그림 9. 경로를 추가하기 위한 절차

Fig. 9. Procedure to add path

4.1.2 ordno 번호 부여의 오버헤드

[그림 8]의 새로운 노드 삽입 시 부모, 양쪽 형제의 ordno 번호에 따라 새로운 노드의 ordno 번호가 결정된다. 부모 노드의 검색은 검색 조건에 따라 결정되므로 직접적인 억세스가 가능하지만 삽입 위치 양쪽에 있는 형제를 찾는 과정은 복잡한 과정이 필요하다. [그림 8]과 같이 2번째 자식으로 입력하기 위해서는 1번째와 2번째 형제를 구하여야 한다. 하지만 ordno는 형제들의 순서를 정하는 크기 정보만 가지고 있으며 그 번호가 연속적이지 않기 때문에 모든 형제들을 데이터베이스에서 전부 메모리로 읽어온 후 정렬의 과정을 거친다. [그림 8]에서는 부모가 (1.3)이므로 삽입되는 노드의 모든 형제들은 (1.3.odd) 형식 또는 (1.3.even.odd), (1.3.even.even.odd) 등과 같은 형식을 가진다. ordno의 레벨이 실제 XML 트리의 레벨과 다르고 또는 짝수, 홀수에 레벨의 깊이가 달라지므로 모든 형제들을 구하는 과정이 복잡하다.

4.1.3 경로 정보의 변경

동적인 노드 번호 ordno의 사용은 문서 내부의 계층 정보와 노드의 순서를 위해서 사용된다. 하지만 동적 번호가 모든 계층 정보를 다 표현하는 것은 아니다. [그림 10]의 XQuery는 입력할 위치를 찾지 못하고 오류를 발생시키는 예제이다. 이 예제 문장은 먼저 입력할 위치 조건을 찾아야 하는데 질의 처리 시스템은 [그림 10]의 XQuery에서 제시된 'users/user_tuple[name='Annabel Lee' 와 '/users/user_tuple/userid' 경로가 'bids.xml' 문서에 존재하는지를 먼저 검사하는 것이 효율적이다. 이 경우 ordno 번호만으로는 질의에 필요한 경로가 존재하는지를 판단할 수 없다. 따라서 경로 정보의 사용은 고유한 번호인 ordno와는 다른 방식으로 효과적인 질의 처리 시스템에 사용된다. XML 문서에는 같은 모양을 가진 경로가 많이 존재할 수 있다. 따라서 경로 정보는 노드의 고유한 번호가 아니므로 XML 문서에 존재하는지 여부를 나타내면 된다.

```
let $uid :=
 doc("users.xml")/users/user_tuple[name="Annabel
   Lee"]/userid
return
 insert nodes
   <bid_tuple>
     <userid>{data($uid)}</userid>
     <itemno>1010</itemno>
     <bid>60</bid>
     <bid_date>2006-04-23</bid_date>
   </bid_tuple>
 into doc("bids.xml")/bids
```

그림 10. Q8: 존재하지 않는 곳에 입력 시도

Fig 10. Q8 Insertion of a node to wrong path

4.2 노드의 삭제 및 변경

XQuery의 삭제 형식은 [그림 11]과 같은 형식을 가진다.

```
let $user := doc("users.xml")/users/user_tuple[name="Dee Linquent"]
return
  delete nodes $user
```

그림 11. Q6 XML 노드의 삭제

Fig 11. Q6 Deletion of XML node

4.2.1 ordno의 삭제

삭제 구문은 해당 노드만 삭제 하는 것이 아니라 [그림 12]와 같이 해당 노드의 모든 서브 트리도 같이 삭제한다.

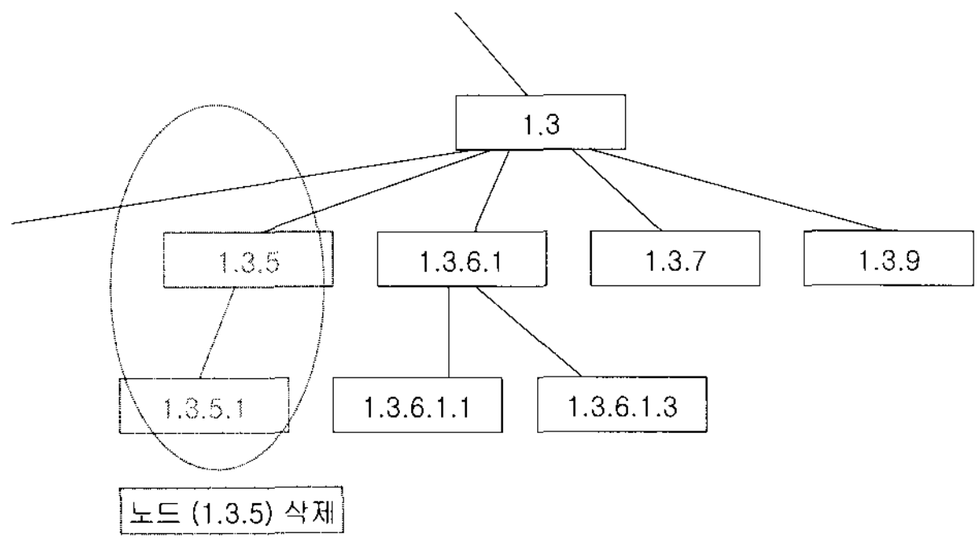

그림 12. 노드 (1.3.5)와 서브 트리의 삭제

Fig. 12. Deletion of node (1.3.5) and its subtree

따라서 삭제하려는 XML 부분의 루트 노드의 ordno를 찾은 후 그 노드의 모든 직계 자손을 ordno를 이용하여 구한다. 예를 들어 삭제하려고 하는 노드가 1.3.5의 ordno를 가지고 있다면 그 자손들은 1.3.5.*의 형식을 가진다. ordno 번호를 이용하여 찾은 모든 서브 트리의 엘리먼트도 Cname_element 테이블에서 같이 삭제된다. ordno가 형제 정보도 가지고 있다면 노드 (1.3.5)의 다른 형제들의 ordno를 전혀 변경할 필요가 없다. 하지만 형제 정보를 sobord를 사용하여 유지할 경우 각 형제들의 sibord 값을 변경하여야 한다.

4.2.2 경로의 삭제

노드와 그 노드의 서브 트리가 삭제되면 삭제되는 노드와 서브 트리가 형성하고 있던 경로 정보도 Cname_location에서 삭제되어야 한다. 서브 트리의 모든 경로를 찾기 위하여 삭제 되는 노드의 직계 자손을 모두 방문하는 것은 오버헤드가 매우 크다. 따라서 ordno를 이용하여 구한 모든 직계 자손들의 ordno 번호를 이용하여 삭제될 엘리먼트를 구하고 Cname_element 테이블의 path_id 컬럼 정보를 이용하여 각 엘리먼트가 연결된 경로를 구하여 삭제한다.

5. 결론 및 향후 연구 계획

XML 모델은 계층적이며 노드의 순서가 의미가 있는 모델이다. 반면에 관계형 데이터 모델은 레코드의 순서가 의미가 없는 집합 이론을 바탕으로 만들어진 모델이다. 대용량의 XML 저장 기법으로 관계형 데이터베이스를 사용하는 경우 XML 문서의 각 요소들이 가지고 있는 계층적인 정보를 관계형 테이블에 효과적으로 표현하는 것은 XML 질의의 성능에 큰 영향을 미친다. 특히 XML 데이터가 동적인 환경에서 XML 데이터의 부분적인 삽입, 삭제가 발생할 경우 계층 정보를 효율적으로 유지하기 위한 많은 방법들이 연구되고 있다. XML 질의의 삽입, 삭제 기능의 확장은 XML 데이터베이스가 단순히 XML 문서의 저장소 기능에서 벗어나 전통적인 데이터베이스 기능을 필요로 한다는 것을 의미한다. 본 논문에서는 XML 데이터의 관계형 테이블 저장 방법에서 표준화 작업이 한창인 XQuery의 변경 기능을 효과적으로 지원하는 방법을 제시하였다. 제안한 방법은 1) XML 문서에 존재하는 경로를 문자열로 표현하여 삽입, 삭제 위치를 쉽게 찾을 수 있으며, 2) 문서의 순서 정보를 나타내는 번호 부여 방식이 새로운 번호의 부여와 삭제를 쉽게 받아들이고 있다. 따라서 관계형 테이블을 이용한 XQuery 변경 기능도 동적 번호 부여 방식과 경로 정보의 활용으로 효과적으로 지원 가능하다는 것을 알 수 있다. XQuery 변경 기능은 트랜잭션이 가지는 ACID 속성도 지원하여야 한다. 관계형 데이터베이스의 트랜잭션 지원 기능을 활용할 경우 XQuery의 트랜잭션도 쉽게 지원할 수 있을 것으로 예측한다.

참 고 문 헌

[1] www.w3c.org

[2] XML:DB Andreas Laux and Lars Martin. XUpdate Working Draft-2000-0914. http://www.xmldb.org

[3] I. Tatarinov, Z. Ives, A. Halevy, D. Weld, "Updating XML" in Proceedings of ACM SIGMOD May Santa Barbara, CA 2001.

[4] Dare Obasanjo, "A proposal for an XML Data Definition and Manipulation Language" in *Lecture Notes in Computer Science* #2590 Springer-Verlag, 2003.

[5] 홍동권, 정민경 "XSTAR: XML 질의의 SQL 변환 알고리즘" *퍼지 및 지능시스템 학회 논문지* Vol 17, No. 3, 2007.

[6] J. Shanmugasumdaram et al, "Relational Databases for Querying XML document: Limitations and Opportunities" in Proceedings of the 25th VLDB Conference, 1999.

[7] M Yoshikawa et al, "XRel: A Path-Based Approach to storage and retrieval of XML document using relational databases" in ACM Transactions on Internet Technology, August 2001.

[8] Igor Tatarinov, Stratis D. Viglas, Kevin Bayer, J. Shanmugasundaram, Eugene Shekita and C. Zhang, "Storing and Querying Ordered XML Using a relational database system" in Proceeding of ACM SIGMOD June 2002.

[9] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman, "On supporting Containment Queries in Relational Database Management Systems" in Proceedings of ACM SIGMOD, May Santa Barbara, CA 2001.

[10] D. Dehaan, D. Toman. M. Consens, and M. Tamer Ozsu, "A Comprehensive XQuery to SQL Translation using Dynamic Interval Encoding" in Proceedings of ACM SIGMOD, San Diego CA, 2003.

[11] P. O'Neil, E. O'Neil, S. Pal, I. Cseri, G. Schaller, "ORDPATHs: Insert-friendly XML node labels" in Proceeding of ACM SIGMOD, Paris, France 2004.

저 자 소 개

홍동권(Dong-Kweon Hong)

2008년 18권 2호 참조