XML 문서의 관계형 데이터베이스 구조로의 동적변환

김유신° 황부현
전남대학교
{loyfor@zgroup.co.kr°, bhhwang@chonnam.chonnam.ac.kr}

Dynamic Translation Of XML Document To Related DATABASE Structure

YuSin Kim° BuHyun Hwang
Dept. of Computer Science, Chonnam National University

요 약

XML 문서내의 정보를 데이터베이스에 저장하는 방법, 특히 관계형 데이터베이스에 저장하는 방법은 별도의 미들웨어를 사용하는 방법과 파싱을 통한 요소들의 매핑을 이용하는 것이 대표적이다. XML 문서 데이터를 데이터베이스에 저장할 때마다 관계형 데이터베이스에 XML 데이터를 파싱하여 그 요소를 각각의 적절한 테이블에 저장하는 방법은 언뜻 보기에는 가장 최적의 방법으로 보이지만 XML 문서 구조가 복잡해질수록 이 방법의 프로그램 로직은 복잡해지고 데이터 처리는 어려워진다. 그리고 계층이 깊은 복잡한 구조의 XML 문서일 경우 관계형 데이터베이스 테이블 구조로는 매핑이 불가능한 경우도 발생한다. 중첩된 구조의 복잡한 XML 데이터를 RDBMS에 저장할 경우 데이터 질의 시 여러 테이블에 걸친 복잡한 연산이 필요하고, XML 데이터의 입력, 수정, 삭제 시 모든 ROW에 걸어야 하는 LOCKING은 시스템의 성능을 떨어뜨릴 수 있다. 또한 XML 문서 스키마가 어떻게 바뀌는가에 따라서 새로 구성해야하는 복잡한 과정을 거칠 수도 있다는 것이다.

이 논문에서는 XML과 데이터베이스와의 공존이라는 측면에서 XML 문서의 관계형 데이터베이스 구조로의 동적 변환에 대하여 연구하고자 한다.

1. 서 론

XML은 뛰어난 문서 기술 언어이기는 하지만 관계형 데이터베이스에서 수행하는 대량의 트랜잭션 처리나 제어를 할 수 있는 구조는 아니다. 그렇기 때문에 새로운 표준인 XML과 이것을 구체화할 수 있는 구조를 갖춘 데이터베이스가 서로 밀접하게 관련되어 상호 보완하면서 발전해야하는 것이다.

이러한 XML이 산업과 각종 전자 문서의 표준이 된다 하더라도 결코 관계형 데이터베이스와의 공존을 피할 수는 없을 것이다. 이 논문은 XML 문서를 관계형 데이터베이스로 변환하는 과정에 있어 3단 테이블을 사용한 동적변환에 관하여 논하고자 한다. 특히, 모든 XML 문서를 3단 테이블로 변환하는 방법을 고안하였다.

2. 관련 연구

2.1 객체지향형 데이터베이스를 이용한 저장 시스템 설계

XML 문서는 크게 문서의 전체적인 구조를 나타내는 DTD(Document Type Definition)와 이러한 DTD를 따르는 문서 인스턴트로 구별할 수 있다. 이러한 특성을 이용하여 XML 문서를 저장하기 위한 스키마를 두 부분으로 분류하였다. 하나의 DTD를 따르는 문서들의 구조정보를 표현하기 위하여 DTD 객체와 DTDElement 객체를 이용하여 문서들의 구조정보를 트리형태로 저장하고, 각각의 문서 인스턴스를 저장하기 위하여 Document 객체와 Element, Attribute 객체를 이용하여 트리형태로 저장하게 된다. 설계된 XML 문서 저장 스키마는 그림 1과 같다.

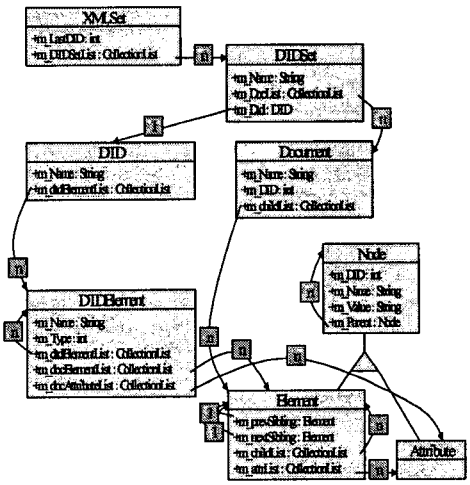

그림 1. XML 문서 저장 스키마

저장 스키마에 나타나는 객체들은 다음과 같은 특성을 갖는다

- XMLSet : 전체 XML 문서 집합을 관리한다.
- DTDSet : 하나의 DTD를 따르는 문서들을 관리한다.

- DTD : DTD를 따르는 모든 문서 인스턴스에서 나타나는 엘리먼트에 대한 구조 정보를 저장한다. DTD내에서 최상위에 존재하는 엘리먼트들에 대한 리스트를 갖고 있다.
- DTDElement : 문서 인스턴트 내에서 해당 위치에 존재하는 엘리먼트들에 대한 리스트와 해당 위치에 존재하는 에트리뷰트 리스트, 자손으로 존재하는 DTDElement 리스트를 갖는다. DTDElement로 이루어진 트리구조를 탐색함으로서 특정 구조를 갖는 객체를 검색할 수 있도록 지원한다.
- Document : 하나의 문서 인스턴스를 나타낸다. 문서 인스턴스의 최상위 엘리먼트에 대한 리스트와 문서번호를 갖는다.
- Node : 엘리먼트와 에트리뷰트의 상위 클래스. 엘리먼트 이름 또는 에트리뷰트 이름, 엘리먼트 내용 또는 에트리뷰트 값을 갖는다.
- Element : 왼쪽/오른쪽 형제, 자손 엘리먼트 리스트와 엘리먼트에 속한 에트리뷰트 리스트를 갖는다.

2.2 XML 문서 저장 방식

XML 문서를 저장하기 위하여 다음과 같은 과정을 거친다. 우선 저장하고자 하는 XML 문서를 파싱하여 DOM(Document Object Model) 형태로 결과를 얻어 온다[4]. 그리고 파싱 결과로 얻어진 DOM을 이용하여 데이터베이스에 Document 객체를 생성하고 Document.m_childList를 이용하여 문서에서 나타나는 Element와 Attribute 객체를 트리형태로 저장하게 된다. 또한 하나의 DTD를 따르는 문서들에 대한 구조 정보를 저장하기위해 저장하고자 하는 문서에 해당하는 DTD 객체를 XMLSet.m_DTDSetList에서 추출하여 DTD.m_dtdElementList를 이용하여 DTDElement들을 탐색하고 저장하고자 하는 Element와 Attribute의 구조정보를 나타낼 DTDElement를 추출하여 DTDElement.docElementList와 DTDElement.docAttributeList에 Element와 Attribute 객체를 추가하게 된다. 그림 2는 XML 문서의 저장 방식을 나타낸다.

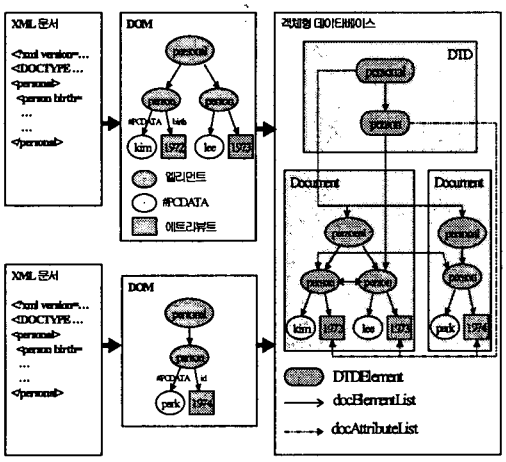

그림 2. XML 문서 저장 방식

그림 2와 같은 방식으로 저장된 XML 문서는 특정 위치의 엘리먼트를 쉽게 탐색할 수 있다. 예를 들어 "personal이라는 엘리먼트의 자손인 모든 person 엘리먼트중 이름이 kim인 엘리먼트를 찾아라"라는 질의를 생각해 보자. DTD 객체에서 DTDElement.m_Name이 personal인 DTDElement 객체를 찾고 DTDElement.m_dtdElementList에 존재하는 DTDElement 객체중에서 DTDElement.m_Name이 person인 DTDElement를 찾아서 DTDElement.docElementList에 대하여 콜렉션(collection)에 대한 질의를 통해 Element 객체 중에서 Element.m_Name이 kim인 Element 객체를 검색하여 얻어 오게 된다.

3. XML2DB 구현

XML 문서내의 데이터 검색효율을 올리고 싶다면 인덱스를 구축하여 XML텍스트의 위치 오프셋을 인덱스로 하면, 갱신시의 인덱스 재구축이 매우 어렵다. 갱신 시에 재구축 수고가 적은 것이 바람직하다. 또한 XML 문서를 취급하기 쉬운 형태로 변환하는 것이 좋다.

XML 트리구조를 기억할 수 있는 것이 좋다. 여기에는 xml2db의 구현방법을 작성하였다.

```
<customer#1 id="J-001">
   <name#2>  Jeffrey </name>
   <city#3> New York  </city>
   <order#4 oid="3">
          <item#5> Notebook </item>
        <date#6> 2002/02/11 </date>
        <num#7> 50 </num>
    </order>
    <order#8 oid="1">
        <item#9> Blank Label </item>
        <date#10> 2002/02/10 </date>
        <num#11> 100 </num>
          <status#12> delivered </status>
    </order>
</customer>
```

[리스트 1] Simple XML Document

우선 각각의 태그에 ID부여한다. 그런 다음 (Tag -> Children List) 라는 레코드를 만든다. customer#1 은 name#2, city#3, order#4, order#에 그리고 order#4는 item#5, date#6, num#7에 사상이 되며 나머지 태그들도 모두 관련 자식 리스트에 알맞게 사상시킬 수 있게 된다.

(Tag -> Children List)

customer#1 -> name#2, city#3, order#4, order#8
order#4 -> item#5, date#6, num#7
order#8 -> item#9, date#10, num#11, status#12
root#0 -> customer#1

최근 가장 많이 사용되는 XML문서의 데이터베이스 변환 방법은 DTD나 스키마 정보를 함께 사용함으로써 데이터 형과 구조 정보를 보존하며 또 간단한 문제점은 애플리케이션에서 처리하기도 한다. xml2db 루틴은 하나의 XML문서를 index table, attribute table, data table로 세분화하여 따로 저장함으로써 XML문서에 포함되어 있는 정보를 가능한 그대로 데이터

베이스로 표현하고자 하는 방법이다.

다음으로 (ID -> data) 형식의 레코드 셋을 만들어 낸다. 리스트 8을 예제로 레코드 셋을 만들자면 2 -> jeffrey, 3 -> New York ... 모든 ID들이 data 형식으로 사상될 수 있다.

(ID -> data)

```
2       -> Jeffrey
3       -> New York
5       -> Notebook
6       -> 2002/02/11
7       -> 50
9       -> Blank Label
10      -> 2002/02/10
11      -> 100
12      -> delivered
```

끝으로 (tag -> attributes) 의 레코드 셋을 만든다. 역시 리스트 8을 예제로 레코드 셋을 만들면 customer#1 -> id="J-001", order#4 -> oid="3"이 만들어지고 나머지 태그들도 레코드 셋으로 조합될 수 있다.

(tag -> attributes)

```
customer#1 -> id="j-001"
order#4    -> oid="3"
```

이에 근거하여 3가지의 테이블을 다음과 같이 생성할 수 있다. 인덱스 테이블(리스트 2), 속성 테이블(리스트 3), 데이터 테이블(리스트 4). 결국 하나의 XML 문서(리스트 1)를 받아들여 세 개의 관계형 데이터베이스 테이블을 동적으로 생성할 수 있는 것이다.

```
Index Table

customer#1   -> name#2, city#3, order#4, order#8
order#4      -> item#5, date#6, num#7
order#8      -> item#9, date#10, num#11, status#12
root#0       -> customer#1
```

[리스트 2] 결과 index table

```
Attribute Table

customer#1 -> id="J-001"
order#4    -> oid="3"
```

[리스트 3] 결과 attrubute table

```
Data Table

2       -> Jeffrey
3       -> New York

5       -> Notebook
6       -> 2002/02/11
7       -> 50
9       -> Blank Label
10      -> 2002/02/10
11      -> 100
12      -> delivered
```

[리스트 4] 결과 data table

결국 인덱스, 속성, 데이터 테이블로 구분되는 세 개의 테이블이 XML 문서를 데이터베이스로 전환하는데 소요되게 된다. 이 후 관계형 데이터베이스에서는 생성된 세 개의 테이블에 대하여 인덱스를 구성하고 XML 문서에 대한 검색 및 저장 조회를 제공하면 된다.

4. 결론 및 향후 연구방향

아직 XML에는 데이터의 일관성을 유지하는 로직 처리나 트랜잭션 처리, 또는 접근 속도를 빠르게 하기 위한 인덱스 기능과 같은 관계형 데이터베이스에는 포함되어 있는 기능들이 단순한 텍스트인 XML에는 포함되어 있지 않다. 최근에는 XML 형식을 직접 내부에 저장하는 객체 지향 방식의 데이터베이스(ODB)도 나와 있긴 하지만 이러한 형태가 일반화되려면 아직 더 시간이 들 것이다. 그러므로 아직까지는 데이터 보존은 관계형 데이터베이스를 이용하고 외부와의 인터페이스가 필요한 경우에만 XML로 변환하는 방법이 주로 사용되고 있으며, 역으로 XML 문서 형태로 전송된 데이터를 관계형 데이터베이스에 저장하는 일이 주요하겠다. 향후 두개이상의 사이트에서 동시에 XML 문서를 변경할 시 데이터베이스의 갱신에 관한 문제가 고려되어야 하겠으며, 거대 XML 문서에 대한 데이터베이스 변환의 속도 문제에 관하여 연구가 이루어져야 하겠다.

[참고문헌]

1. Alin Deutsch , University of Pennsylvania, USA, 1998 http://www.w3.org/TR/NOTE-xml-ql

2. *Massimo Marchiori, W3C Contact for XML Query* http://www.w3.org/XML/Query

3. Ronald Bourret, Mapping DTDs to Databases, http://www.rpbourret.com/xml/

4. The Lorel query language for semistructured data(*) http://link.springer.de/link/service/journals/00799/bibs/7001001/70010068.htm

5. 권 혁민, 정보과학회 논문지(B) 1996 직렬화 가능성 그래프 검사 기법을 이용한 장기 트랜잭션을 위한 동시성 제어 기법

6. Steve DeRose (Inso Corp. and Brown University), http://www.w3.org/TR/xpath

7. Ashok Malhotra (Microsoft), 2001, http://www.w3.org/TR/xqueryx