객체관계 데이터베이스를 이용한 효율적인 XML 문서 저장 스키마 설계

최윤진[O] 이언배
한국방송통신대학교 정보과학과
yj93071@hanmail.net[O] lub@mail.knou.ac.kr

Design of An Efficient XML Documents Storage schema Using ORDB

Yunjin Choi[0] Eunbae Lee
Dept. of Computer Science, Korea National Open University

요 약

본 논문에서는 객체관계 데이터베이스를 이용하여 XML 문서를 저장하기 위한 효율적인 XML 문서 저장 스키마를 제안한다. 제안하는 저장 스키마는 XML 문서를 구성하는 모든 구성요소를 모델링하여 데이터베이스 테이블을 구성하였으므로 스키마 생성과정에서 일부 구성요소를 다루지 않음으로 인해 발생할 수 있는 정보 손실 가능성을 제거하였다. 또한 이는 DTD 독립적인 저장 스키마이므로 모든 XML 문서에 대해 공통 스키마로 이용될 수 있어 효율적이며 저장 및 검색의 투명성을 높일 수 있는 장점이 있다. 제안한 스키마는 상용 객체관계 데이터베이스인 오라클 9i에 적용하여 테이블을 구성하고 저장 및 검색과정을 실험하였으며 실험결과 XML 문서의 모든 구성요소에 대해 정보 손실 없이 다양한 형태의 검색이 가능함을 확인하였다.

1. 서 론

XML(eXtensible Markup Language)이 웹상에서의 데이터 교환 표준으로 W3C(World Web Consortium)에서 제안된 이후 XML을 이용해서 교환되는 데이터의 양이 증가하면서 XML 문서의 저장과 검색에 대한 많은 연구가 진행되고 있다. 이러한 추세에 따라 SQL서버 2000이나 오라클과 같은 상용 DBMS를 판매하는 벤더들은 자사의 DBMS를 확장하여 XML 기능을 추가하고 있으며, Tamino와 같은 XML 전용 데이터베이스 제품도 개발되어 이용되고 있다. 그렇지만 이러한 방식들은 특정 DBMS에 종속적이므로 원활한 XML 응용을 위해서는 DBMS 독립적인 저장 방법이 필요하다.

XML문서 저장과 관련해서 주의하여야 할 것은 XML문서를 저장할 때 문서내의 데이터만 유지 관리하는 것은 구조화된 정보를 갖고 있는 XML문서를 저장하는 방식으로 적합하지 않다는 점이다. 따라서 XML 문서를 저장할 때에는 문서에서 표현하는 모든 정보를 손실 없이 저장하여야 하며 특히 XML 문서의 특징인 구조화 정보를 유지하는 것이 필요하다. 이를 위해 XML 문서를 저장할 때에는 데이터들간의 의미적, 구조적 관계를 설정하는 저장 스키마를 생성하여 데이터를 구조화된 정보로 저장 할 수 있어야 한다. XML 문서 저장 스키마 생성에 관한 기존 연구에서는 스키마 생성 시 주요 구성요소만 취급하여, 생성된 스키마를 이용해서 실제 문서를 저장할 때 정보 손실 가능성이 존재한다. 이러한 정보 손실 가능성을 방지하기 위해서는 저장 시 XML 문서 구조에 따라 저장 스키마를 일부 수정하여 처리할 필요가 있는데, 이 경우 공통된 스키마가 아닌 각기 다른 수정된 형태의 스키마가 생성 되므로 공통된 저장 및 검색 방법을 사용할 수 없고 저장할 문서 구조에 맞게 수정해서 사용하여야 하므로 효율적이지 못하다. 따라서 저장 및 검색 효율을 높이기 위해서는 XML 문서 저장 시 정보 손실을 방지할 수 있는 저장 스키마의 생성이 필요하다.

본 논문에서는 객체관계 데이터베이스를 이용해서 특정 DBMS에 독립적으로 XML 문서를 저장하고 검색하기 위한 공통 저장 스키마를 제안한다. 제안하는 스키마는 DTD(Document Type Definition)에 독립적이며 XML 문서의 모든 구성요소를 다루고 있으므로 하나의 저장 스키마를 이용해서 XML 문서 저장 시 정보 손실 없는 공통된 저장방법 및 검색 방법을 제공할 수 있다. 본 서론에 이어 2장에서 XML 문서 저장에 대한 관련 연구를 소개하고, 3장에서는 본 논문에서 제안하는 XML문서 저장 스키마를 소개하며, 상용 데이터베이스인 오라클 9i에 적용한 사례를 보여준다. 4장에서는 본 논문의 결론과 향후 연구과제에 대해 언급한다.

2. 관련연구

객체관계 데이터베이스를 이용하여 XML 문서를 저장하기 위한 기존 연구는 DTD를 이용한 저장 스키마 생성 방법과 DTD 독립적인 저장 스키마 생성 방법이 있다[1]. DTD를 이용한 XML 문서 저장 스키마 생성 방법은 DTD 스키마를 객체-관계 형식으로 매핑하는 규칙을 만들고 그 규칙에 따라 특정 XML 문서의 구조를 나타내는 DTD 스키마를 데이터베이스 테이블로 매핑하는 저장 스키마를 생성하게 된다[2]. 따라서 특정 DTD를 따르는 XML 문서에 최적화한 저장 스키마를 생성할 수 있어 공간을 절약하고 조인 오퍼레이션을 줄여준다는 장점이 있다. 그렇지만 이 방법은 DTD마다 각기 다른 저장 스키마를 생성해야 하므로 XML 문서 구조가 다양해지면 저장 스키마 수도 따라서 증가하게 되며, DTD의 수정 이 어렵고 DTD가 없는 XML 문서는 저장 스키마를 생성할 수 없는 문제가 있다.

DTD 독립적인 저장 스키마 생성 방법은 DTD 유무에 관계 없이 XML 문서만을 이용해서 생성하는 방법[3]과 DTD 독립적인 저장 스키마를 생성하되 DTD가 존재하는 문서의 경우 DTD정보를 별도의 테이블에 저장하여 DTD 정보를 유지하는 방법이 연구되고 있다[4]. DTD 독립적인 저장 스키마 생성 방법은 특정 XML 문서 구조에 독립적이므로 모든 XML 문서를 고정된 테이블에 저장할 수 있는 공통 스키마를 생성할 수 있는 장점이 있다. 그렇지만 기존 연구에서는 저장 스키마 생성 시 XML 문서의 전체 구성요소를 모두 다루지 않고 주요 구성 요소인 ELEMENT

ATTRIBUTE, TEXT 위주로 저장 스키마를 생성하고 있어 생성된 저장 스키마를 실제 XML 문서에 적용 시 문서의 정보를 일부 손실하게 될 가능성이 존재하게 되는 문제가 있다.

3. 효율적인 XML 문서 저장 스키마

3.1 XML 문서 구성요소 간의 관계 정의

본 논문에서는 XML 문서를 객체 모델링 기술을 이용해서 모델링하여 정보 손실 없는 저장 스키마를 생성 한다. XML 문서를 모델링 하기 위해서는 XML 문서 구조에 접근하여야 한다. 아직 XML 문서를 기술하는 완벽한 모델은 개발되지 않았지만 XML 문서를 위한 몇몇 구조들은 W3C스펙에 의해 개발되었다. DOM(Document Object Model)은 그 중 하나이다[5]. DOM을 이용해서 접근하는 노드 중 DOCUMENT 노드는 XML Document로서 XML 문서 자체를 나타내며 DOCUMENT TYPE은 DTD를 나타낸다. NOTATION노드는 DTD 내부에서만 표시되므로 본 논문에서는 별도로 다루지 않는다. 각 노드 간의 관계 및 특징은 그림 1과 같이 정의 한다.

노드 간의 관계	관계 정의 및 특징
XML Document와 DTD	0또는 1의 DTD를 갖고, DTD는 0개 이상의 XML Document를 참조한다
XML Document와 PI, Comment	PI와 Comment는 XML Document에 0개 이상 나타날 수 있다. PI는 이름-값 쌍으로 구성된다
XML Document와 Element	Element는 최상위 엘리먼트인 Root Element를 갖고 모든 엘리먼트는 Root Element안에 포함되는 트리와 같은 계층구조를 갖는다. XML Document 는 1개의 Root Element를 갖는다
Element와 Element	Root Element는 0개 이상의 자식 Element를 갖는다 부모 Element는 0개 이상의 자식 엘리먼트를 갖고 자식 Element는 1개의 부모 엘리먼트를 갖는다 Element는 0개이상의 형제 엘리먼트를 갖는다. 형제 엘리먼트에는 순서가 있다.
Element와 Text	Element는 0또는 1개의 Text를 갖는다. Text의 형태는 일반 문자 데이터 또는 Cdata-Section이거나 일반 문자 데이터와 Cdata-Section이 혼합된 형태로 구성될 수 있다. Cdata-Section은 마크업을 포함한 텍스트를 표시하기 위해 사용한다
Element와 Attribute	Element는 0개 이상의 Attribute를 갖는다. Attribute는 이름-값 쌍으로 구성된다
Entity	파싱된 Entity는 대체 텍스트로서 well-formed 내용을 갖는 다. 파싱된 엔티티는 DTD에서 Entity 선언으로 이름과 대체 텍 스트를 정의하고 Entity Reference를 통해 대체 텍스트를 XML 문서에 포함시킨다. 파싱되지 않은 엔티티는 XML 데이터일 수도 있고 아닐 수도 있다. 파싱되지 않은 Entity는 DTD에서 Entity로 표시되며 Attribute를 갖는다. XML 문서 내에서는 속성 값으로만 사용될 수 있다

그림1 DOM으로 접근하는 노드간의 관계 및 특징

3.2 XML 문서 모델링

그림2는 XML 문서 구성요소 간의 관계를 반영하여 작성된 객체도이다. 그림에서는 DOM을 통해 접근한 각 노드를 클래스로 표시하였다. 클래스간의 관계를 포함 관계로 표시하였고 숫자로 정량적 관계를 나타내었다. Attribute와 Entity 간의 참조 관계 그리고 형제 Element 간의 순서는 연관관계로 표시하였다.
모델링 과정에서 엔티티 중 일반 파싱된 엔티티인 Entity Reference는 대체 텍스트로 변환해서 XML 문서에 삽입하여 처리한다. Cdata-Section은 Cdata-Section 안에 있는 문자를 문자데이터로 취급하여 Text로 처리한다. 즉 Cdata-Section을 표시하는 " <!CDATA[" 와 "]]>" 사이에 있는 문자를 Text 클래스에서 처리한다. XML Document 클래스는 자신을 식별하는 고유한 이름이 있으며 Element 클래스, DTD 클래스, PI 클래스 그리고 Comment클래스를 포함하고 있다. DTD 클래스는 자신을 식별하는 고유한 이름과 DTD 내용으로 구성되며 0개 이상의 XML Document에서 참조한다. PI 클래스는 이름과 값을 갖고 있으며 자신이 포함된 1개의 XML Document에서 참조한다.

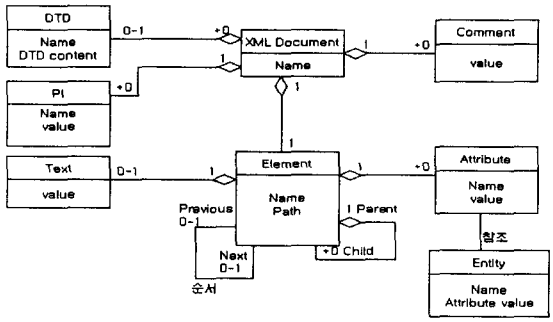

그림 2 XML 문서를 표현한 객체도

Comment는 이름 없이 값만 갖고 있으며 자신이 포함된 1개의 XML Document에서 참조한다. Element 클래스는 Root Element를 갖고 있으며 Root Element는 자신이 포함된 1개의 XML Document에서 참조한다. Element클래스는 다른 클래스를 포함할 수 있다. Element 클래스에 포함되는 클래스는 Text 클래스, Attribute 클래스, 그리고 부모 Element는 자식 Element인 Element 클래스를 포함한다. Text클래스는 이름이 없고 문자데이터 값을 갖고 있으며 자신이 포함된 1개의 Element에서 참조한다. Attribute 클래스는 이름과 값을 갖고 있으며 자신이 포함된 1개의 Element에서 참조한다. Element 클래스는 자신을 식별하는 이름과 Path 속성을 갖는다. Path속성은 구조 검색 시 검색 효율을 높이기 위해 Root Element로부터의 경로정보를 저장하는 속성이다[3]. 일반 파싱 되지 않은 데이터인 Entity 클래스는 이름과 속성값으로 구성되며 Attribute 클래스의 값으로 참조할 수 있다.

3.3 XML 문서 저장 스키마

그림 3은 그림 2 에서 제시된 객체도를 이용해서 생성한 데이터베이스 저장 스키마이다. 객체도에서 클래스로 표시된 객체는 데이터베이스 테이블이 되고 클래스이름이 테이블 이름이 된다. 포함관계에 있는 클래스들은 SQL3에서 제안한 ref 함수를 이용해서 참조한다. 참조되는 클래스 중 수량이 0 이상인 클래스는 중첩 테이블로 구성해서 참조한다. 그림에서 속성이름에 * 가 붙은 속성은 중첩 테이블이다.

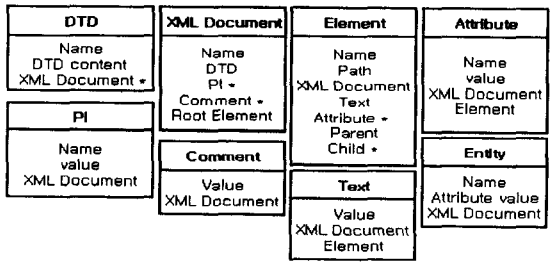

그림 3 XML 문서 저장 스키마

DTD 테이블의 DTD content는 DTD가 XML 문서 외부에 존재할 수도 있고 XML 문서 내부에 존재할 수도 있기 때문에 BLOB 등과 같은 대용량 객체 형태가 아닌 문자열 형태로 저장한다. XML Document 테이블은 이름 속성을 갖고 포함관계에 있는 다른 4개의 테이블을 참조하는 속성을 갖게 된다. 엘리먼트 테이블은 이름 속성과 Path 속성을 갖고, XML Document를 참조하는 속성을 가지며 부모 엘리먼트와 자식 엘리먼트를 포함하여 포함관계에 있는 다른 4개의 테이블을 참조하는 속성을 갖게 된다. 데이터 중심 문서(data-centric documents)에서 데이터들 간의 순서는 중요한 의미를 갖지 않으므로 엘리먼트의 순서관계는 무시한다[1]. 검색 효율을 높이기 위해 모든 테이블에 자신

이 포함된 XML Document를 참조하는 속성을 둔다. 엘리먼트와 포함관계에 있는 테이블들은 엘리먼트를 참조하는 속성을 갖는다. Entity 테이블은 일반 파싱되지 않은 엔티티를 관리한다. XML 문서 내에서는 파싱되지 않은 엔티티의 이름이 Attribute에서 값으로 표현된다. 파싱되지 않은 엔티티와 관계된 엘리먼트는 애트리뷰트 테이블을 통해 참조할 수 있다.

3.4 제안한 저장 스키마를 이용한 테이블 생성

지금까지 설명한 저장 스키마를 상용 객체관계 데이터베이스인 오라클 9i에 적용해서 실제 테이블을 생성 하였다. 생성과정은 먼저 스키마 구성요소에 해당하는 객체타입을 생성하고 객체간 관계에서 수량 0 이상으로 참조되는 객체는 중첩테이블에 해당하는 객체타입을 생성한 다음 생성된 객체타입을 이용해서 객체 테이블을 생성한다. 그림 4는 생성된 객체를 이용해서 실제 테이블을 생성하는 SQL문장들을 보여준다. 중첩테이블은 별도의 저장공간을 이용해서 저장한다.

```
CREATE TABLE tbl_XMLDocument OF typ_XMLDocument
    NESTED TABLE xm_PI STORE AS nt_xml_PI,
    NESTED TABLE xm_Comment STORE AS nt_xml_Comment;
CREATE TABLE tbl_DTD OF typ_DTD
    NESTED TABLE xml STORE AS nt_dtd_xml;
CREATE TABLE tbl_PI OF typ_PI;
CREATE TABLE tbl_Comment OF typ_Comment;
CREATE TABLE tbl_Element OF typ_Element
    NESTED TABLE attribute STORE AS nt_element_attr,
    NESTED TABLE child_element STORE AS nt_element_child;
CREATE TABLE tbl_Text OF typ_Text;
CREATE TABLE tbl_Attribute OF typ_Attribute;
CREATE TABLE tbl_Entity OF typ_Entity;
```

그림 4 테이블 생성 SQL문장

3.5 XML문서 저장 및 검색

제안한 스키마를 적용해서 생성된 테이블에 실제 XML 문서를 저장하고 검색하는 실험을 하였다. 그림5와 그림 6은 예제 DTD와 예제 xml문서이다. DTD에는 파싱된 일반 엔티티와 파싱되지 않은 일반 엔티티가 포함되어 있다. 그림 7은 저장된 예제문서에 대한 질의형태 및 질의 결과 예시 이다.

```
<!ENTITY mass "mass market paperback">
<!ENTITY hard "hardcover">
<!NOTATION DOC SYSTEM "Microsoft Word document">
<!ENTITY rev_leaves SYSTEM "Review of Leaves of Grass.doc" NDATA DOC>
<!ELEMENT INVENTORY (BOOK)*>
<!ELEMENT BOOK (TITLE, AUTHOR, BINDING, PAGES, PRICE)>
<!ATTLIST BOOK InStock (yes|no) #REQUIRED Reviews ENTITIES #IMPLIED
<!ELEMENT TITLE (#PCDATA|SUBTITLE)*>
<!ELEMENT SUBTITLE (#PCDATA)>
<!ELEMENT AUTHOR (#PCDATA)>
<!ATTLIST AUTHOR Born CDATA #IMPLIED>
<!ELEMENT BINDING (#PCDATA)>
<!ELEMENT PAGES (#PCDATA)>
<!ELEMENT PRICE (#PCDATA)>
```

그림 5 예제문서 sample1.dtd

```
<?xml version="1.0" encoding="EUC-KR"?>
<!-- file name:sample1.xml-->
<!-- write 2004-01-26-->
<!DOCTYPE INVENTORY SYSTEM "sample1.DTD">
<INVENTORY>
  <BOOK InStock="yes">
    <TITLE>The Adventures of Huckleberry Finn</TITLE>
    <AUTHOR Born="1835">Mark Twain</AUTHOR>
    <BINDING> &mass;</BINDING>
    <PAGES>298</PAGES>
    <PRICE> $5.49</PRICE>
  </BOOK>
  <BOOK InStock="no" Reviews="rev_leaves">
    <TITLE>Leaves of Grass</TITLE>
    <AUTHOR Born="1819">Walt Whitman</AUTHOR>
    <BINDING> &hard;</BINDING>
    <PAGES>462</PAGES>
    <PRICE> $7.75</PRICE>
  </BOOK>
</INVENTORY>
```

그림 6 예제문서 sample1.xml

예1. 문서 내용에 "Adventures"가 들어가는 문서 검색

```
select r.xml.xm_name from tbl_Text r where te_value like '%Adventures%';
```

질의결과 : sample1.xml

예2. 부모 엘리먼트와 자식 엘리먼트 구조가 "BOOK/PRICE"인 엘리먼트를 갖는 문서검색

```
select distinct r.xml.xm_name from tbl_Element r
where r.el_path like '%BOOK/PRICE%';
```

검색결과 : sample1.xml

예3. 'sample1.xml' 문서에서 속성값으로 사용되는 엔티티 이름 및 내용 검색

```
select a.at_value, r.en_attr_value from tbl_entity r join tbl_attribute a
on r.en_name=a.at_value where r.xml.xm_name='sample1.xml';
```

질의결과 : rev_leaves,Review of Leaves of Grass.doc

예4. 재고가 있는 책 제목 검색

```
select temp.column_value.text.te_value from table
(select r.child_element from tbl_element r where ref(r)=
(select element from tbl_attribute where at_name='InStock' and
at_value='yes')) temp
where temp.column_value.el_name='TITLE';
```

질의 결과 : *The Adventures of Huckleberry Finn*

그림 7 저장된 데이터에 대한 질의 예

질의 결과는 다양한 형태로 검색이 가능함을 보이고 있다. 이처럼 제안하는 스키마는 XML 문서 내에서 표현되는 모든 노드를 모델링하여 생성하였으므로 정보 손실 없이 XML 문서의 저장이 이루어질 수 있고, DTD에 독립적 이므로 모든 XML문서를 저장할 수 있는 공통 스키마로 사용할 수 있어 효율적이며 저장 및 검색의 투명성을 높일 수 있는 이점이 있다.

4. 결론 및 향후 연구과제

XML 문서 저장 시 저장 및 검색 효율을 높이기 위해서는 정보 손실을 방지하고 공통으로 이용할 수 있는 저장 스키마의 생성이 필요하다. 본 논문에서 제안한 저장 스키마는 DOM을 이용해 XML 문서 구조에 접근한 다음 XML 문서의 각 구성요소를 객체로 표현하였고, 객체 간의 관계와 특성을 정의해서 모델링 하는 과정을 통해 생성하였다. 모델링은 XML 문서 저장 과정에서 정보 손실이 없도록 하기 위해 XML 문서 구성요소 전체를 대상으로 하였다. 제안한 스키마는 상용 객체관계 데이터베이스인 오라클 9i를 이용하여 객체 테이블을 생성하고 XML 문서의 내용 저장과 검색하는 과정을 통해 정보 손실 없이 다양한 형태의 검색이 가능함을 확인하였다. 제안한 스키마는 XML 문서 구조에 독립적이면서 정보손실 없이 XML 문서를 저장할 수 있는 공통 스키마로 사용할 수 있으므로 효율적이며 저장 및 검색의 투명성을 높일 수 있는 이점이 있다.

향후 연구과제는 XML질의를 본 논문에서 제안한 스키마 구조에 맞게 처리할 수 있는 질의처리기를 구현할 계획이다.

참고문헌

[1] 박성진 " XML 데이터베이스" , 한국인터넷정보학회지, 제2권 제3호, pp38-46, 2001.

[2] Sangho Ha and Kyoungrea Kim, " Mapping XML Documents to the Object-Relational Form," Proceedings of the 2001 IEEE International Symposium on Industrial Electronics, Volume 3, pp.1757-1761, 2001

[3] Takeyuki Shimura, Masatoshi Yohikawa and Shunsuke Uemura, " Storage and Retrival of XML Documents Using Object-Relational Databases," Proc. of DEXA99, pp.206-217, 1999

[4] 김훈, " 객체관계 데이터베이스를 이용한 XML 문서 저장 모델 설계" , 서울시립대학교 이학석사 학위논문, 2001.

[5] Jun Wen, Rui Zhang, Xianliang Lu, " The Design of Efficient XML Document Model," Proceedings of the 2002 International Conference on Machine Learning and Cybernetics, Volume 2, pp.1102-1106, 2002