객체-관계 데이터베이스에의 XML 문서의 저장 시스템*

김경래, 하상호
순천향대학교 정보기술공학부

Storing XML Documents on Object-Relational Databases

Kyungrea Kim and Sangho Ha
Dept. of Information Technology, SoonChunHyang University

요약

XML 은 웹 문서 기술을 위한 표준과 모든 유형의 정보에 대한 데이터 기술 언어로써 점차 그 사용 폭을 넓혀가고 있다. 최근에 관계형 혹은 객체 지향형 데이터베이스에 XML 문서를 저장하는 여러 연구가 진행되어 왔다. 그러나 관계 데이터베이스에의 저장은 XML 문서 구조와 관계 스키마와의 불일치에 따른 문제점을 안고 있으며, 객체지향 데이터베이스에의 저장은 관계 데이터베이스가 주를 이루는 상용 데이터베이스를 사용할 수 없다는 문제점을 안고 있다. 본 논문에서는 이러한 문제점을 해결하기 위한 방안으로, 최근에 XML 가용 인터넷 플랫폼을 지원하고 있는 Oracle 8i를 사용하여 객체-관계 데이터베이스 상에 XML 문서를 저장하는 방법을 제시하고, 그 저장 시스템을 구현한다.

1.서론

XML[1]이 웹 상의 데이터를 표현하고 교환하기 위한 새로운 표준으로 빠른 속도로 받아들여지고 있다. XML 데이터는 그 구조를 자체적으로 기술 가능하며, 따라서 XML 문서를 해석하고 그 문서를 조작하는 응용 프로그램을 작성하는 것이 가능하다. 이러한 프로그램은 내용에 기반 하여 문서를 필터링하고 필요에 따라 문서를 재구성하는 것을 포함한다.

최근에 XML 문서를 데이터베이스에 저장하는 방법에 대해서 많이 연구되어 왔다. 한 방법은 XML 데이터를 객체지향 데이터베이스에 저장하는 것이다. 이 방법은 주어진 XML 데이터로부터 클래스 계층을 도출한다. 여기서 문서의 각 데이터 요소는 한 개의 클래스가 된다. 이 방법은 데이터 요소의 순서와 포함 관계를 자연히 표현할 수 있고, 저장 시스템을 최적화시킬 수 있다는 장점을 제공한다. 그러나 클래스가 각 요소에 대해서 생성되므로 비효율성의 문제점이 제기된다. 다른 방법은 XML 문서를 관계 데이터베이스에 저장하는 것이다. 이 방법[2]은 XML 구조와 관계 테이블간의 불일치로 인하여 저장하는 방법이 자연스럽지 않으며, 검색 시에도 많은 join 연산 등 비효율성의 문제점이 제기된다. 그러나 관계데이터베이스가 주를 이루고 있는 상용 데이터베이스를 사용할 수 있다는 장점을 갖는다.

본 논문에서는 객체-관계 데이터베이스에 XML 문서를 저장하는 방법을 제안하고, 이를 위한 시스템을 구현한다. 객체-관계 데이터베이스에의 저장은 객체지향 데이터베이스와 관계 데이터베이스의 장점을 모두 취할 수 있다는 점에서 다른 두 방법에 비해서 우월하다고 볼 수 있다. 객체-관계 데이터베이스에서는 테이블의 한 필드의 값이 객체일 수 있다. 따라서 포함 관계를 갖는 데이터 요소를 객체로 정의함으로써 XML 문서를 효과적으로 객체-관계 데이터베이스에 사상할 수 있다. 논문에서는 도서 정보를 기술하는 XML 문서를 객체-관계형 데이터베이스에 저장하는 시스템을 구성한다. 먼저, XML 문서 구조를 제시하

* 본 연구는 정보통신부의 대학 S/W 연구센터 지원 사업에 의해 수행된 것임.

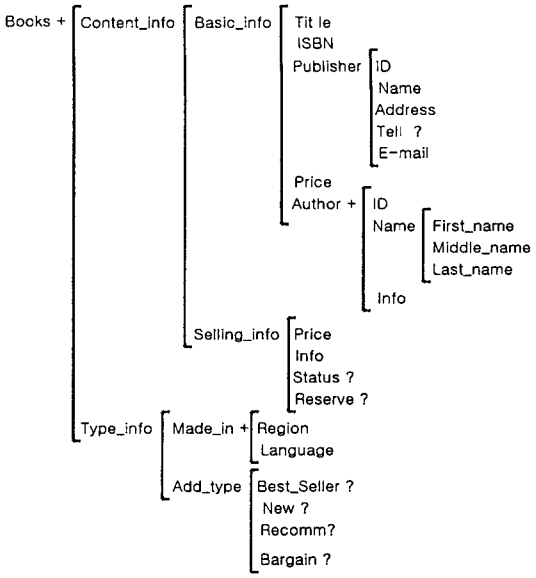
고, 이를 객체-관계 데이터베이스에 저장하는 방법을 제안하고, 마지막으로 시스템 구성을 통해 그 방법을 구현한다. 시스템은 Client, Middleware, DB 서버의 세 부분으로 구성되는데, Java Servlet[3]을 사용하여 Client와 Middleware간의 인터페이스를 구현하였으며, JDBC[4]를 사용하여 Middleware와 DB 서버간의 인테페이스를 구현하였다. Middleware는 클라이언트와 DB 서버간의 인터페이스 역할을 수행하면서 XML 데이터를 실제로 조작하는 부분을 담당한다. DB 서버로는 최근에 e-commerce를 위해서 XML 가용 인터넷 플랫폼을 지원하고 있는 Oracle 8i[5]를 사용한다. 또한, 여기서 개발된 전 시스템이 Java로 개발되어 특정 플랫폼에 독립적이다.

논문의 순서는 다음과 같다. 2장에서는 XML 문서를 객체-관계 스키마에 사상하는 원칙을 제시하고, 특정 XML 문서에 대해서 이를 적용한다. 3장에서는 객체-관계 데이터베이스에 XML 문서를 저장하는 시스템을 구현하고, 마지막으로 4장에서는 결론을 언급한다.

2. 객체 관계형 스키마 표현

데이터베이스 테이블은 XML문서에 맞춰 만들어진다. 관계형 데이터베이스 사용 시 테이블 생성을 위해선 각 테이블간의 관계를 정해줘야 했으며, 따라서 프로그램으로 XML에 맞는 데이터베이스 테이블을 생성해 주기엔 어려움이 많았다. 하지만 객체-관계 데이터베이스를 사용할 경우 이와 같은 문제가 쉽게 해결된다. 객체-관계 데이터베이스의 구조는 XML문서와 같이 트리-구조를 이룬다. XML문서는 Tag로 트리-구조를 표현한다[6]. 따라서 XML문서의 Tag name으로 객체들을 생성해 준다면 이는 XML문서의 트리-구조 그대로 객체-관계 데이터베이스에 프로그램으로 테이블과 객체를 만들어주는 것도 가능하다. 그러나 이와 같이 생성된 스키마는 문서구조에 따라 효율적으로 생성할 수가 없는 문제점이 발생한다. 데이터의 중복이나 DB의 효율성을 고려한다면, XML문서를 저장하는 데이터베이스 테이블을 직접 만들어 주는 것이 현재로선 더 바람직하다.

그림 1은 도서 정보를 트리-구조 형태로 표현하고 있다. 도서 정보는 내용 정보(content_info)와 유형 정보(type_info)로 구성되며, 내용 정보는 기본 정보(basic_info)와 판매 정보(selling_info)로 구성된다는 것을 알 수 있다.

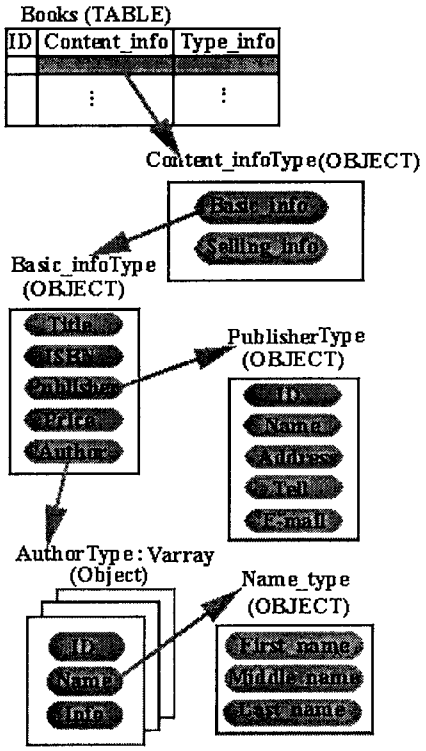


<그림 1: 트리-구조 형태의 도서 정보 표현>

다음은 그림 1과 같이 트리-구조 형태를 갖는 XML 문서를 Oracle 8i의 객체-관계 형태로 사상하기 위한 규칙을 제안한다.

(1) 트리-구조에서 최상위 노드는 테이블로 사상한다.
(2) 노드가 잎 노드일 경우에, 상위 노드는 객체 타입으로 선언하고 잎 노드는 그 객체의 필드가 된다.
(3) 노드가 여러 번 나타날 수 있는 '+'나 '*'로 표현될 경우에, 노드는 VARRAY 타입으로 표현된다.
(4) 각 객체들은 트리-구조상에서 하위의 개체들을 필드로 가질 수 있다.

위의 원칙에 따르면, Books 요소는 테이블로 표현된다. 이 테이블은 content_info와 type_info의 두 객체를 포함한다. content_info는 basic_info와 selling_info의 객체를 포함한다. basic_info는 title, ISBN, price의 3개의 스칼라 필드와 author와 publisher의 객체를 포함한다. author는 id, name, info의 3개의 스칼라 필드를 포함하는 객체의 배열로 구성된다. selling_info는 price와 status, reserve, info의 4개의 필드로 구성된다. type_info는 made_in과 add_type의 두 객체로 구성되며, made_in은 region, language의 두 필드로 구성된 객체 배열로 구성되며, add_type은 best_seller, new, recomm, bargain의 4개의 필드로 구성된다. 그림 2는 그림 1의 트리-구조 형태의 도서 정보를 객체-관계



<그림2: 객체-관계형으로 사상된 다이어그램>

형태로 사상한 결과를 다이어그램 형태로 표현한 것이다. 그림 3은 그림 1의 트리-구조를 Oracle 8i의 SQL 문으로 표현한 것이다. author, made_in과 같이 여러 번 발생 가능한 요소는 varray 타입으로 표현되는 것을 알 수 있다. varray 타입은 Oracle8i에서 제공되고 있으며 객체 배열을 표현하는데 용이하다.

```
CREATE TABLE books(
content_info content_infotype,
type_info type_infotype
)
CREATE TYPE content_infotype AS OBJECT(
basic_info basic_infotype,
selling_info selling_infotype
)
CREATE TYPE type_infotype AS OBJECT(
made_in      made_inarraytype,
Add_type     add_typetype
)
CREATE TYPE made_inarraytype AS
 VARRAY(10) OF made_intype;
CREATE TYPE made_intype AS OBJECT(
region   varchar2(20),
status   varchar2(20)
)
CREATE TYPE add_typetype AS OBJECT(
Best_seller      varchar2(20),
New              varchar2(20),
Recomm           varchar2(20),
Bargain          varchar2(20)
)
CREATE TYPE basic_infotype AS OBJECT(
Title      varchar2(20),
ISBN       varchar2(20),
Publisher  publishertype,
Price      varchar2(20),
Author     author_array
)
CREATE TYPE author_array AS VARRAY(10)
 OF author;
CREATE TYPE author AS OBJECT(
Id         varchar2(20),
Name       nametype,
Info       varchar2(20)
)
CREATE TYPE nametype AS OBJECT(
First_name       varchar2(20),
Middle_name      varchar2(20),
Last_name        varchar2(20)
)
CREATE TYPE publishertype AS OBJECT(
ID         varchar2(20),
Name       varchar2(20),
Address    varchar2(20),
Tel        varchar2(20),
Email      varchar2(20)
)
CREATE TYPE selling_info AS OBJECT(
Price      varchar2(20),
Info       varchar2(20),
Status     varchar2(20),
Reserve    varchar2(20)
)
```

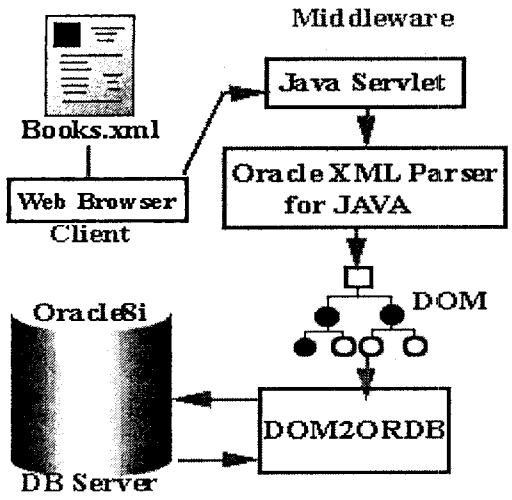
<그림3: 객체-관계형으로 사상하는 SQL 문>

3. 구현

XML문서의 저장 시스템은 Client, Middleware, DB 서버의 세 부분으로 구성되는데, Java Servlet을 사용하여 Client와 Middleware간의 인터페이스를 구현하였으며, JDBC를 사용하여 Middleware와 DB 서버간의 인테페이스를 구현하였다. Middleware는 클라이언트와 DB 서버간의 인터페이스 역할을 수행하면서 XML 데이터를 실제로 조작하는 부분을 담당한다. DB 서버로는 최근에 e-commerce를 위해서 XML 가용 인터넷 플랫폼을 지원하고 있는 Oracle 8i를 사용한다

XML문서는 Oracle8i의 XDK(Xml Developer's Kit)[7]에서 지원하는 XML parser for JAVA를 사용하여 파싱을 한다. Parser는 XML문서의 유효성을 검증하고 파싱된 문서를 DOM Tree로 만들어 준다. DOM Tree는 XML문서의 Element와 Attribute, Value 들을

Tree형식으로 구현한다[8].



<그림 4: XML 문서 저장 시스템 구조>

DOM Tree에서 XML문서의 Element와 Attribute의 이름, 그리고 해당 값들을 추출해 낼 수 있다. Element 이름과 Attribute의 이름들은 데이터 베이스의 Table이나 Object, Column으로 사상[9,10,11]되어 저장하는 Query를 생성하게 된다.

그림 4는 XML 문서가 데이터베이스에 저장되는 과정을 그린 것이다. Books.xml 문서가 웹브라우저에서 입력되면 Servlet에서 Books.xml을 받아들인다. Books.xml은 Parser에 의해 Parsing되고 그 결과물은 DOM으로 반환된다. Books.xml의 Element와 Attribute 그리고 각각의 Value들을 DOM Tree에서 추출하여 Element의 이름은 데이터베이스의 테이블 또는 Object의 이름과 사상을 한다. 사상이 성공하였을 경우 해당 Element의 이름을 이용해 Query를 생성하며, 이 Query는 JDBC를 통해 데이터베이스에 질의를 한다. 결국 질의 결과로 Element들의 값과 Attribute들의 값이 데이터베이스에 저장된다.

4.결론

본 논문에서는 객체-관계 데이터베이스에 XML 문서를 저장하는 방법을 제안하고, 이를 도서 정보를 기술하는 XML 문서에 적용하였다. 또한, XML 문서를 객체-관계 데이터베이스에 저장하는 시스템을 구현하였다. 시스템 구성요소 모두가 Java로 구현되었으며, 따라서 본 시스템은 특정 플랫폼에 독립적이다.

앞으로 객체-관계 데이터베이스에 저장된 XML 데이터를 검색하는 시스템을 설계/구현할 계획에 있다. 또한, 데이터베이스 상에 저장된 XML 데이터의 중복을 제거하는 것도 바람직한 연구 과제이다.

참고문헌

[1] "Extensible Markup Language (XML)", http://www.w3.org/XML/, 2000.

[2] 하상호, 이강석, 백인천, "Storing XML Documens using Oracle8i XDK", 멀티미디어춘계학회, 2000.

[3] James Goodwill, Reading, Developing Java™ Servlet, SAMS, 1999.

[4] Danny Ayers, Homes Bergsten, Michael Bogovich, Reading, Professional JAVA server Programming, WROX Press Ltd, 1999.

[5] 주종면, Reading, Oracle8i, 대림출판사, 2000.

[6]Jennifer Widom, "Data Management for XML", http://www-db.stanford.edu/~widom/xml-whitepaper.html, September 1999.

[7]Steve Muench, "Oracle XSQL Pages and the XSQL Servlet", http://technet.oracle.com/tech/xml/, 2000

[8] Frank Boumphrey, Reading, PROFESSIONAL XML APPLICATIOINS, Worx Press Ltd., 1999.

[9] Takeyuki Shimura, Masatoshi Yoshikawa, Shunsuke Uemura, "Storage and Retrieval of XML Documents using Object-Realtional Databases", DEXA, 1999.

[10] Jayavel Shanmugasundaram, Kristin Tufte, Gang Hi, "Relational Databases for Querying XML Documents: Limitaions and Opportunities", VLDB, 1999.

[11] Alin Deutsch, M.Fernandez, D. Suciu, "Storing Semistructured Data with STORED", Proceedins of the ACM SIGMOD Conference, May 1999.

[12] Oracle Corporation, "Oracle8i, The XML Enabled Data Management System", 1999.

[13] Ronald Bourret, "XML and Data Base", http://www.informatik.tu-darmstadt.de/DVSI/staff/bourret/xml/XMLAndDatabases.htm, 1999.