객체모델을 이용한, XML DTD의 RDB 스키마로의 변환

김경수[O] 주경수
천안외국어대학 컴퓨터정보과, 순천향대학교 전산학과
kkskim@ccfs.ac.kr[O], gsoojoo@asan.sch.ac.kr

Transformation XML DTD to RDB Chema using Object Model

Kyung Soo Kim[0] Joo Kyung-Soo
Dept. of Computer Information, Cheonan College of Foreign Studies
Dept. of Computer Science, College of Engineering Soonchunhyang University

요 약

XML이 단순한 컨텐츠에서 데이터베이스로까지 그 적용 분야가 확장되면서 XML로 표현된 정보들을 어떻게 효율적으로 저장하고 관리하는 것이다. 가장 큰 이슈 중의 하나는 기존의 관계형 데이터베이스에도 XML을 효율적으로 관리할 수 있는가 이다. 이를 위해 XML 응용과 관계형 데이터베이스 연계를 위한 다양한 연구가 이루어지고 있으나, 객체를 기본 개념에 기반을 두었으며 계층구조를 갖는 XML 데이터를 2차원 테이블의 집합인 관계형 데이터베이스에 저장하기 위해서는 많은 테이블이 필요하며, 이에 따른 죠인 연산으로 시스템 성능이 저하 될 수 있는 본질적인 한계가 있다. 따라서 XML 데이터를 데이터베이스에 저장하기 위해서는 계층적 구조를 2차원 정보로 변환하는 변환 방법을 만들어 각 구조화된 정보를 데이터베이스에 저장하고 다시 XML로 연동할 수 있는 것이다. 본 논문은 객체 모델을 토대로 XML DTD에 정의한 요소와 특성들을 객체화하여 관계형 데이터베이스 스키마로 변환하기 위한 연계 방법을 제안한다. 이를 위하여 먼저 XML DTD를 객체 모델로 변환시키기 위한 객체 변환 방안을 제시하고, 변환된 객체 모델을 관계형 데이터베이스 스키마로 변경시키기 위한 스키마 변환 방법을 제안했다.

1. 서론

XML이 단순한 컨텐츠에서 데이터베이스로까지 그 적용 분야가 확장되면서 XML로 표현된 정보들을 어떻게 효율적으로 저장하고 관리할 것인지에 대한 기술도 XML과 함께 발전되어가고 있다. XML로 B2B 포털 사이트를 구축하기 위해서는 방대한 양의 XML 문서를 저장 관리하든지, 다른 사이트에 저장된 XML 문서를 가져와야만 한다. 앞으로 도래하는 B2B 시대에는 XML 문서의 양도 많아질 것이며, 이것을 저장 관리할 XML 저장 관리 기술이 요구될 것이다. 가장 많이 사용되고 있는 관계형 데이터베이스에 저장하는 방식은 현재 관계형 데이터베이스 시장이 다른 데이터베이스에 비해 넓고 그에 따라 사용자 수도 많기 때문에 확산이 빠르며 쉽게 상용화할 수 있다는 장점이 있다. 따라서 XML 스키마에서 관계형 데이터베이스 모델링까지의 과정을 자동화시키고자 하는 많은 연구가 진행되고 있지만, XML은 계층적 구조를 가지고 있으므로 이를 2차원적인 테이블의 집합인 관계형 정보로 표현하기에는 한계가 있다. 따라서 계층적 구조를 가진 DTD를 관계형 데이터베이스에 저장하기 위한 모델링으로 변환하기 위한 작업이 필요하다[1].

따라서 본 연구는 XML DTD와 관계형 데이터베이스의 연동을 위하여 객체 모델을 기반으로 했으며, 이 객체 모델을 관계형 데이터베이스 스키마로 변환하기 위한 변환 방법을 제안했다. 아울러 변환 방법에 의해 관계형 데이터베이스 스키마 스크립트를 구현했다.

본 논문의 제 2절에서는 XML DTD에서 객체로 변환하는 방법을 다루고, 제 3절에서는 객체를 관계형 데이터베이스 스키마로 변환하는 방법을 제안하며, 마지막으로 결론을 기술한다.

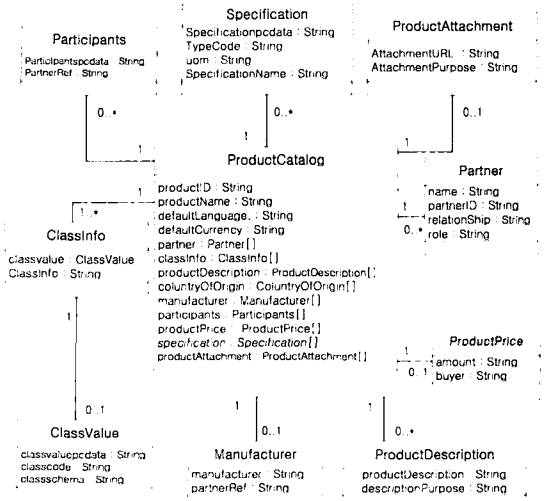
2. XML DTD의 객체 모델로의 변환

XML의 태그들은 문서의 구조나 데이터의 의미를 나타내기 위해서 사용된다. 따라서 문서를 작성할 때 문서의 구조 혹은 데이터의 의미에 따라 사용자가 필요한 태그들을 자유롭게 정의할 수 있으며, DTD는 XML 문서 내에서 사용 가능한 요소를 관리하고, 사용할 수 있는 각 요소형의 내용과 특성을 지정하는 규칙들을 담고 있다[3].

이러한 규칙에 따라 그림 1에 보여준 XML DTD는 그림 2와 같이 객체모델로 변환될 수 있다.

```
<?xml version="1.0" encoding="EUC-KR"?>
<!ELEMENT ProductCatalog (ProductName,
    DefaultLanguage, DefaultCurrency, Partner*,
    ClassInfo+, ProductDescription*,
ColuntryOfOrigin?, Manufacturer?, Participants*,
ProductPrice*, Specification*,
ProductAttachment?)>
<!ATTLIST ProductCatalog ProductID ID #REQUIRED >
<!ELEMENT ProductName (#PCDATA)>
<!ELEMENT DefaultLanguage (#PCDATA)>
<!ELEMENT DefaultCurrency (#PCDATA)>
<!ELEMENT Partner (#PCDATA)>
<!ATTLIST Partner
            PartnerID ID #IMPLIED
            RelationShip (Seller | Manufacturer |
              Imtermediary) #IMPLIED
            Role CDATA #IMPLIED >
<!ELEMENT Name (#PCDATA)>
<!ELEMENT ClassInfo (ClassValue, ClassInfo?)>
<!ELEMENT ClassValue (#PCDATA)>
<!ATTLIST ClassValue
            ClassCode CDATA #IMPLIED
            ClassSchema CDATA #IMPLIED>
<!ELEMENT ProductDescription (#PCDATA)>
<!ATTLIST ProductDescription
            DescriptionPurpose CDATA #IMPLIED >
<!ELEMENT CountryOfOrigin (#PCDATA)>
<!ELEMENT Manufacturer (#PCDATA)>
<!ATTLIST Manufacturer
            PartnerRef IDREF #IMPLIED >
<!ELEMENT Participants (#PCDATA)>
<!ATTLIST Participants
            PartnerRef IDREF #IMPLIED >
<!ELEMENT ProductPrice (Amount, Buyer?)>
<!ELEMENT Amount (#PCDATA)>
<!ELEMENT Buyer (#PCDATA)>
<!ELEMENT Specification (#PCDATA)>
<!ATTLIST Specification
          TypeCode (Size | Weight | Ingredient | Color |
                Shape | Packing | Performance |
                Content | Others) #IMPLIED
          UOM CDATA #IMPLIED
          SpecificationName CDATA #REQUIRED >
<!ELEMENT ProductAttachment (AttachmentURL,
AttachmentPurpose?)>
<!ELEMENT AttachmentURL (#PCDATA)>
<!ELEMENT AttachmentPurpose (#PCDATA)>
```

(그림 1) XML DTD



(그림 2) 객체모델

3. 객체로부터 관계형 데이터베이스 스키마로 변환

앞에서 제시한 XML DTD의 객체변환에 따라 변환된 객체들을 다음과 같은 변환 방법에 의해 데이터베이스 스키마로 변환시킨다[3][4].

3.1 변환 방법

다음은 XML DTD로부터 변환된 객체 클래스가 관계형 데이터베이스 스키마로 변환하는 방법이다.

① 클래스는 테이블이 된다
② 클래스의 속성은 테이블의 컬럼이 된다
③ 클래스의 속성 타입은 테이블의 컬럼 타입이 된다.
④ 속성에 {nullable} 태그가 있으면 테이블 속성에 NULL 또는 NOT NULL을 추가한다.
⑤ 속성이 초기치 값을 가지면, 컬럼에 DEFAULT 문을 추가한다.
⑥ 루트나 독립적인 클래스와 같이 일반화가 없는 클래스를 위해서는 integer 기본키를 생성하고, {oid}를 위해서는 기본키와 외부키를 {oid} 태그 컬럼을 추가한다.
⑦ 자식클래스들은, 각 부모클래스의 기본키와 외부키를 추가한다.
⑧ 연관 클래스들은, 각 역할-실행 테이블에서 기본키 제약에 대한 기본키를 추가한다.
⑨ 만일 {alternate oid = <n>} 태그면, UNIQUE 제약을 컬럼에 추가한다.
⑩ 각 명시된 제약에 대해 CHECK를 추가한다.
⑪ 0....1, 1....1 규칙의 연관.관계에서 참조하는 테이블에 외부키를 생성한다.
⑫ 집합 테이블(CASCADE와 같이)의 외부키와 같이 복합연관에 대한 기본키를 생성한다 ; 기본키를 위해 컬럼을 추가한다.
⑬ 적당한 곳에 "1 : 다" 부분 테이블로 이동함으로써 이원 연관 클래스을 최적화한다.
⑭ 연관 클래스가 아닌 3원 연관인 다 : 다에 대한 테이블을 생성한다.
⑮ 다 : 다 연관에서 역할-실행 테이블의 키로부터 기본키와 외부키를 생성한다.

그림 2의 객체 모델을 3.1에서 제시한 변환방법에 따라 관계형 데이터베이스 스키마로 변환하면 그림 3과 같다.

```
CREATE TABLE ProductCatalog (
   productID CHARACTER(8)
   productName CHARACTER(20)
   defaultLanguage CHARACTER(10)
   defaultCurrency CHARACTER(5)
   partner CHARACTER(20)
   producPrice CHARACTER(20)
   ProductDescription CHARACTER(20)
   classInfo CHARACTER(20)
   manufacturer CHARACTER(20)
   participants CHARACTER(20)
   specification CHARACTER(20)
   productAttachment CHARACTER(20)
CONSTRAINT ProductCatalog_pk PRIMARY KEY (productID)
)
CREATE TABLE Partner (
   name CHARACTER(8)
   partnerID CHARACTER(6)
   relationship CHARACTER(5)
   role CHARACTER(10)
   partner CHARACTER(20)
CONSTRAINT Partner_pk PRIMARY KEY(partner)
)
CREATE TABLE ClassInfo (
   classInfo CHARACTER(20)
   classValue CHARACTER(20)
CONSTRAINT ClassInfo_pk PRIMARY KEY(classinfo)
)
CREATE TABLE ClassValue (
   classValuePcdata CHARACTER(6)
   classCode CHARACTER(4)
   classScheme CHARACTER(6)
   classInfo CHARACTER(20)
)
CREATE TABLE ProductDescription (
   productDescriptionPcdata CHARACTER(10)
   descriptionPurpose CHARACTER(10)
ProductDescription CHARACTER(20)
CONSTRAINT ProductDescription_pk PRIMARY
KEY(productdescription)
)
CREATE TABLE Manufacturer (
   manufacturePcdata CHARACTER(8)
   productDescriptionPcdata CHARACTER(8)
   descriptionPurpose CHARACTER(10)
   manufacturer CHARACTER(20)
CONSTRAINT ProductDescription_pk PRIMARY
KEY(productdescription)
)
CREATE TABLE Participants (
   participantsPcdata CHARACTER(8)
   partnerRef CHARACTER(8)
   participants CHARACTER(20)
CONSTRAINT Participants_pk PRIMARY KEY(participants)
)
CREATE TABLE ProductPrice (
   amount CHARACTER(10)
   buyer CHARACTER(8)
   productprice CHARACTER(20)
CONSTRAINT ProductPrice_pk PRIMARY KEY(productPrice)
)
CREATE TABLE Specification (
   specificationPcdata CHARACTER(10)
   typeCode CHARACTER(8)
   uom  CHARACTER(6)
   specificationName CHARACTER(10)
   specification CHARACTER(20)
CONSTRAINT Specification_pk PRIMARY KEY(specification)
)
CREATE TABLE ProductAttachment (
   attachmentURL CHARACTER(16)
   productAttachment CHARACTER(20)
CONSTRAINT ProductAttachment_pk PRIMARY
KEY(productAttachment)
)
```

```
ALTER TABLE ProductCatalog ADD CONSTRAINT partner_pk
    FOREIGN KEY (partner) REFERENCES scott.Partner (partner)
;
ALTER TABLE ProductCatalog ADD CONSTRAINT classInfo_pk
    FOREIGN KEY (classInfo) REFERENCES scott.ClassInfo
    (classInfo) ;
ALTER TABLE ClassInfo ADD CONSTRAINT classValue_pk
    FOREIGN    KEY (classValue) REFERENCES scott.ClassValue
    (classValue) ;
ALTER TABLE ProductCatalog ADD CONSTRAINT
    productDescription_pk FOREIGN KEY (productDescription)
        REFERENCES scott.ProductDescription (productdescription)
;
ALTER TABLE ProductCatalog ADD CONSTRAINT
    manufacturer_pk FOREIGN KEY (manufacturer) REFERENCES
    scott.Manufacturer (manufacturer) ;
ALTER TABLE ProductCatalog ADD CONSTRAINT participants_pk
    FOREIGN KEY (participants) REFERENCES scott.Participants
    (Participants) ;
ALTER TABLE ProductCatalog ADD CONSTRAINT productPrice_pk
    FOREIGN KEY (productPrice) REFERENCES scott.ProductPrice
    (productPrice) ;
ALTER TABLE ProductCatalog ADD CONSTRAINT specification_pk
    FOREIGN KEY (specification) REFERENCES scott.Specification
    (specification) ;
ALTER TABLE ProductCatalog ADD CONSTRAINT
    productAttachment_pk FOREIGN KEY (productAttachment)
    REFERENCES scott.ProductAttachment (productAttachment) ;
```

(그림 3) 관계형 데이터베이스 스키마 구현

4. 결론

본 연구는 XML DTD와 관계형 데이터베이스의 연동을 위하여 객체 모델을 기반으로 했으며, 이 객체 모델을 관계형 데이터베이스 스키마로 변환하기 위한 변환 방법을 제안했다. 아울러 변환 방법에 의해 관계형 데이터베이스 스키마 스크립트를 구현했다.

향후 연구 방향으로는 XML DTD를 관계형 데이터베이스 스키마로 자동변환키 위한 미들웨어 컴포넌트들을 CBD(Component Based Design) 방법에 따라 추출하고 설계하여 이를 EJB(Enterprise JavaBeans)로 구현하고자 한다.

5. 참고문헌

[1] James Rumbaugh, Object-oriented modeling and design, Englewood Cliffs, N.J, Prentice Hall, 1991.

[2] Andrew Davidson, Matthew Fuchs, Mette Hedin, Mudia Jain, Jari Koistinen, Chris Lloyd, Murray Maloney, Kelly Schwarzhof, "Schema for Object-Oriented XML 2.0", July, 1999. See http://www.w3.org/TR/NOTE-SOX

[3] R.G.G.Cattell, Douglas K. Barry, The Object database Standard : ODMG 2.0, Morgan Kaufmann Publishers, Inc, 1997.

[4] Robert J. Muller, Database Design for Smarties: Smarties: Using UML for Data Modeling, Morgan Publishers, Inc, 1999.