객체모델을 기반으로 한 XML DTD의 RDB 스키마로의 변환 방법

이 상 태, *이 정 수, *주 경 수
신성대학 컴퓨터응용계열,
*순천향대학교 컴퓨터학부
전화 : 041-350-1180 / 핸드폰 : 016-417-0079

Mapping from XML DTD to RDB Schema based on Object Model

Lee Sang-Tae, *Lee Jung-Soo, *Joo Kyung-Soo
Dept. of Computer Science Shinsung College
*Dept. of Computer Science, SoonChunHyang University
E-mail : stlee@shinsung.ac.kr

Abstract

XML (eXtensible Markup Language) is a flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere. A document type definition (DTD) is a specific definition of the rules of the Standard Generalized Markup Language. A relational database management system (RDBMS) is a program that lets you create, update, and administer a relational database. An RDBMS takes Structured Query Language (SQL) statements entered by a user or contained in an application program and creates, updates, or provides access to the database.
This paper has been studied a method of mappings from XML DTD to RDB schemas based on object model.

1. 서론

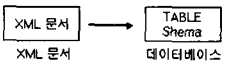
XML은 웹에서 데이터 교환을 위한 표준 마크업언어이다. 마크업이란 HTML 문서에서 사용되던 "<"와 ">"로 둘러싸인 태그들을 의미한다. XML에서도 태그는 시작 태그와 끝 태그로 구성되어 있는데 HTML과 다른 점은 HTML의 태그들은 HTML 문서가 어떻게 화면에 보여질 것인가를 나타내는 반면에, XML의 태그들은 문서의 구조나 데이터의 의미를 나타내기 위해서 사용된다. 따라서 문서를 작성할 때 문서의 구조 혹은 데이터의 의미에 따라 사용자가 필요한 태그들을 자유롭게 정의할 수 있다. DTD는 XML 문서 내에서 사용 가능한 요소를 관리하고, 사용할 수 있는 각 요소형의 내용과 특성을 지정하는 규칙들을 담고 있다. 본 논문에서는 이 DTD에 정의한 요소와 특성들을 객체화하여 RDB 스키마로 변환하는 기법들이 다루어 진다. 본 논문의 제2절에서는 매핑하는 기법들을 설명하고, 제3절에서는 XML DTD에서 객체로 매핑하는 방법을 다루며, 제4절에서는 객체를 RDB 스키마로 추출하는 방법을 다룬다. 마지막으로 결론을 기술한다.

본 연구는 정보통신부의 ITRC 사업에 의해 수행된 것임

2. Mapping 방법

2.1 테이블 기반 매핑

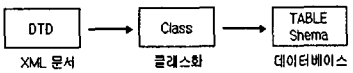
이 기법은 XML 문서를 테이블로 직접 변환하는 기법으로 단순한 XML 문서일 때 사용된다. XML 문서에서 루트 요소가 테이블이 되고 XML 문서에 있는 루트 요소 내에 있으며, 중복된 요소 내에 있는 각각의 요소들이 테이블의 컬럼이 되고 중복된 요소 내에 있는 각각의 요소에 속하는 내용이 테이블의 레코드로 처리된다. 이런 기법은 단순한 XML 문서에 대하여 하나의 테이블로만 변환이 되므로 이해하기 쉽고 데이터 변환 코드가 간단하게 이루어지므로 XML 문서와 RDB 사이에 데이터 변환이 쉽게 이루어진다. 그러나 복잡한 XML 문서들에서 이 기법이 사용될 수 없는 단점이 있다.[그림 1]



[그림 1] 테이블 기반 매핑

2.2 객체 기반 매핑

테이블 기반 매핑 기법은 복잡한 XML 문서를 변환하기가 불가능하므로 이 단점을 개선시키기 위해 중간에 객체를 이용하여 테이블 스키마를 추출하는 기법이다. [그림 2]



[그림 2] 객체 기반 매핑

3. XML DTD의 객체 기반 매핑

3.1 요소

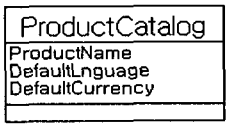
HTML은 태그가 바로 스타일 시트로 사용된다. 시작 태그는 성질을 열고, 끝 태그는 다시 닫는다. XML에서는 시작 태그와 끝 태그를 컨테이너로 사용한다. 시작 태그의 내용과 끝 태그가 함께 하나의 요소를 이룬다. 요소는 하나의 루트 요소만 가져야 하며, 다른 태그는 모두 요소 안에 확실하게 중첩되어야 한다. 이것은 한 요소가 다른 요소들을 포함할 경우, 그 요소들은 한 요소 안에 들어가야 한다는 뜻이다. 요소 타입은 두 개의 타입으로 분류하는데 PCDATA만 가진 요소의 타입을 단순 요소 타입이라고 하며, PCDATA을 제외한 모든 요소의 타입을 복합 요소 타입이라고 한다. 다시 말해 복합 요소 타입은 요소의 자식 요소, 혼합 요소, 요소의 특성들이다.

단순 요소 타입을 객체로 변환할 때는 클래스의 속성으로 변환된다. 아래에 단순 요소 타입의 예를 보여준다.[그림 3]

```
<!ELEMENT ProductCatalog (ProductName,
          DefaultLanguage, DefaultCurrency)>
<!ELEMENT ProductName (#PCDATA)>
<!ELEMENT DefaultLanguage (#PCDATA)>
<!ELEMENT DefaultCurrency (#PCDATA)>
```

[그림 3] 단순 요소 타입

[그림 3]은 클래스 속성으로 변환하면 다음과 같다.[그림 4]



[그림 4] 단순 요소 타입을 객체로 매핑

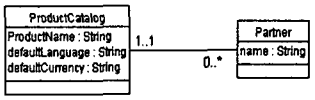
[그림 4]에서 보면 클래스 이름은 [그림 3]에 있는 요소가 포함하고 있는 부모 요소가 클래스 이름으로 지정되고, 데이터 형은 작성자가 요소의 의미에 따라 정수형, 문자형 등으로 지정된다. 복합 요소 타입은 단순 요소 타입을 제외한 모든 요소를 말하는데 다음과 같다.[그림 5]

[그림 5]에서는 루트 요소가 ProductCatalog이고 그에 자식 요소는 ProductName, DefaultLanguage, DefaultCurrency이다. 따라서 객체로 변환하면 다음과 같다.[그림 6]

[그림 6]은 객체로 변환된 클래스이다. 클래스 이름은 ProductCatalog이다. 여기서 데이터 형은 [그림 3]에서 요소의 의미가 문자형으로 사용되므로 객체로 변환할 때 String형으로 변환된다.

```
<!ELEMENT ProductCatalog (ProductName,
          DefaultLanguage, DefaultCurrency,
                    ClassInfo,Partner)>
<!ELEMENT ProductName (#PCDATA)>
<!ELEMENT DefaultLanguage (#PCDATA)>
<!ELEMENT DefaultCurrency (#PCDATA)>
<!ELEMENT Partner (Name)>
```

[그림 5] 복합 요소 타입



[그림 6] 복합 요소 타입의 객체로 변환

3.2 특성

모든 요소는 특성을 가질 수 있다. 특성은 이름/값 의 형태를 가진다. 특성은 요소에 포함되며, 특성에 부여된 값을 통해서 그 요소에 어떠한 특징을 제공하게 된다.

특성은 XML 파서가 애플리케이션에게 보내는 값들을 뜻하는데 요소의 내용과는 구별되는 값이므로 요소와 마찬가지고 특성도 단일 값을 가진 특성과 다중 값을 가진 특성으로 분류한다.

단일 값을 가진 특성은 문자열 특성(CDATA), 토큰 특성(ID, IDREF, NMTOKEN, ENTITY, NOTATION), 열거형 특성이며, 객체로 변환될 때 클래스의 속성으로 변환한다.[그림 7]

```
<!ELEMENT ProductDescription>
<!ATTLIST ProductDescription
   DescriptionPurpose CDATA #IMPLIED>
```

[그림 7] 단일 값을 가진 특성

[그림 7]에서 보면 클래스 이름이 ProductDescription이고 클래스의 속성은 DescriptionPurpose이다.[그림 8]

[그림 8] 단일 값을 가진 특성을 객체로 변환

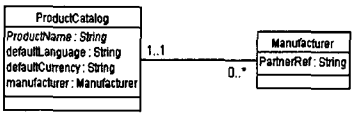
다중 값을 가진 특성은 특성에 IDREFS, NMTOKENS, ENTITIES으로 선언된 것이다. 이것들을 객체로 변환할 때 값이 하나 이상으로 들어가기 때문에 별도의 객체로 만들 수 있다. 다음 그림은 특성을 포함한 DTD를 보여준다.[그림 9]

```
<!ELEMENT ProductCatalog (ProductName,
          DefaultLanguage, DefaultCurrency,
                    Manufacturer)>
<!ELEMENT ProductName (#PCDATA)>
<!ELEMENT DefaultLanguage (#PCDATA)>
<!ELEMENT DefaultCurrency (#PCDATA)>
<!ELEMENT Manufacturer >
<!ATTLIST Manufacturer
          PartnerRef IDREFS #IMPLIED>
```

[그림 9] 복합 요소 타입의 객체로 변환

[그림 9]에서 보면 요소의 특성인 PartnerRef은 IDREFS로 선언되어 있다. IDREF는 단일 값을 가지고 있는 토큰 특성

ID와 의미는 같으나 ID는 요소에 대한 ID로 같은 문서에서 같은 ID 특성 값을 가진 두 개의 요소를 가질 수 없으며, IDREF는 ID를 가리키는 포인터이고 IDREFS는 하나 이상의 IDREF값으로 이루어진다. 따라서 객체로 만들 수 있다.



[그림 10] 다중 값을 가진 특성

[그림 10]에서는 두 개의 클래스로 이루어지며, 한 클래스는 부모 클래스이고 다른 하나는 자식 클래스로 구분된다. [그림 9]에서 부모 요소는 ProductCatalog이고 자식 요소는 Manufacturer이므로 객체로 변환하면 [그림 10]과 같다. 따라서 두 개의 클래스를 연결하려면 참조형으로 연결하면 된다.

3.3 복합 내용 모델 매핑

(1) 순차

DTD에 있는 요소와 요소에 속하는 자식 요소들을 순차적으로 객체로 변환하면, 자식 클래스와 부모 클래스간의 연결을 참조형으로 변환할 수 있다. 테이블 스키마로 변환할 때 그 테이블이 속하는 컬럼들은 반드시 NOT NULL 제약조건이 따른다.

(2) 선택

DTD 요소 안에 자식 요소들이 있는데 그 중에 하나만 선택할 경우에도 객체로 변환될 수 있고, 변환된 객체는 다시 RDB 스키마로 변환된다. RDB 스키마로 변환할 때 그 테이블에 속하는 컬럼들이 선택적이기 때문에 "NOT NULL" 제약이 없다.

(3) 반복

요소 안에 자식 요소가 중복된 경우 객체화로 변환할 때 요소 안에 중복된 같은 요소들이 클래스 속성으로 변환하는 경우 배열 형태로 변환된다. 이렇게 변환된 클래스 속성은 별도의 테이블로 변환된다. 하나 이상의 요소들이 존재한다는 의미로 클래스 속성의 데이터형을 String[]으로 변환한다. 즉 작성자가 알 수 없는 만큼 요소가 존재한다는 의미로 별도의 테이블로 변환한다.

(4) 서브그룹

<!ELEMENT A (B, (C | D))> 와 같은 경우를 말하며, 이 요소는 <!ELEMENT A (B, C))> 와 <!ELEMENT A (B, D))> 요소로 분류된다. 따라서 두 개의 객체로 변환할 수 있다.

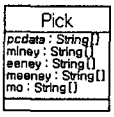
3.4 혼합 내용 모델 매핑

텍스트나 요소, 그 둘을 모두 포함할 수 있는 요소들을 혼합 내용 모델이라고 하며, XML 프로세서는 공백, 탭 등의 PCDATA와 요소 내용간의 구분이 어렵다. 왜냐하면 끝 태그와 다음의 시작 태그 사이에 공백이 있으면 불분명하게 된다. 혼합 내용 모델에서 선택이 가능한 그룹은 하나이고, #PCDATA로 시작하며, 그 뒤에는 혼합 내용의 연산자수를 나타내는 타입 순서로 되며, 각각은 한 번만 선언된다. #PCDATA만 유일한 옵션이며, "*"는 반드시 괄호를 닫은 바로 뒤에 와야 한다. 다음 DTD 예를 들어보자.[그림 11]

```
<!ELEMENT pick (#PCDATA | eeney | meeney
                        | miney | mo)* )>
<!ELEMENT eeney (#PCDATA)>
<!ELEMENT meeney(#PCDATA)>
<!ELEMENT miney(#PCDATA)>
<!ELEMENT mo(#PCDATA)>
```

[그림 11] 혼합 내용 모델 DTD

위에 있는 DTD는 혼합 내용 모델이다. pick 요소는 루트 요소이고 자식 요소들은 반복적으로 이루어진다. 따라서 루트 요소는 클래스 이름으로 변환하고 자식 요소는 클래스의 속성으로 변환이 된다. 그리고 반복의 의미가 있는 데이터 형은 배열로 선언이 된다. 이 DTD를 객체로 변환하면 다음과 같다.[그림 12]



[그림 12] 혼합 내용 모델 DTD를 객체화로 변환

DTD에서 "*" 연산자는 요소나 요소 그룹이 생략되거나 한 번 이상 나타날 수 있으므로 설계하는 작성자는 몇 개가 나타나는지 알 수 없기 때문에 String[] 형으로 선언한다.

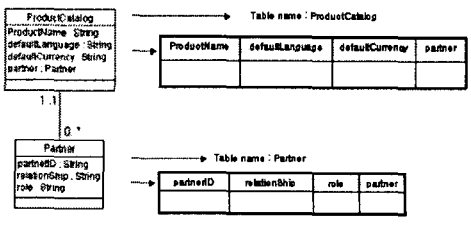
4. 객체의 RDB변환

요소들을 객체를 이용하여 RDB 스키마로 변환한다. 일반적으로 클래스 이름은 테이블 이름으로 변환이 되고, 클래스의 속성은 테이블 컬럼으로 변환할 수 있다. 다음 XML DTD를 RDB로 변환하는 예를 들어보자.[그림 13]

[그림 13]에서 보면 클래스가 두개가 추출되는데 ProductCatalog와 Partner이다. 따라서 클래스 이름이 테이블 이름으로 변환되므로 ProoductCatalog 테이블과 Partner 테이블로 변환한다.[그림 14]

```
<!ELEMENT ProductCatalog (ProductName,
    DefaultLanguage, DefaultCurrency, Partner)>
<!ELEMENT ProductName (#PCDATA)>
<!ELEMENT DefaultLanguage (#PCDATA)>
<!ELEMENT DefaultCurrency (#PCDATA)>
<!ELEMENT Partner >
<!ATTLIST Partner
          PartnerID ID #IMPLIED
        RelationShip (Seller | Manufacturer |
                      Imtermediary) #IMPLIED
          Role CDATA #IMPLIED>
```

[그림 13] DTD



[그림 14] 객체에서 RDB 스키마로 변환

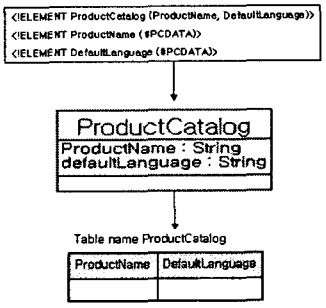
4.1 테이블 데이터 유형 및 제약조건

일반적으로 클래스 속성들을 보면 String형으로 선언되어 있다. 그렇지만 작성자가 요소를 보면 그 요소가 어떤 데이터 형이 맞는지는 정해주어야 한다. 클래스가 추출되었다면 그 클래스를 보고 테이블 스키마를 추출하여야 한다. 여기서 테이블 스키마를 추출할 때 각각에 대한 컬럼의 데이터형을 선언해주어야 하는데 클래스 속성 중에 String형으로 선언한 것을 테이블에서는 문자형으로 선언한다. 만일 클래스 속성이 int형으로 지정되어 있다면 테이블에서는 숫자형으로 선언 된다.

RDB 스키마를 추출할 때 컬럼의 제약조건 또한 중요하다. 즉 컬럼에 반드시 값이 입력되어야 할 때 NOT NULL이라고 지정하여야 하고 그렇지 못할 때는 NULL을 지정할 수 있다. 데이터 유형과 컬럼의 제약조건은 XML DTD에 있는 요소를 보고 정할 수 있는데 요소의 연산자를 보고 컬럼의 제약조건을 정할 수 있다. 요소 연산자가 없는 요소들이 테이블 컬럼에 NOT NULL이라고 사용자가 정할 수 있다.[그림 15] 여기서 ProductName, DefaultLanguage 요소는 내용 모델 연산자가 없으므로 테이블 컬럼, ProductName컬럼과 DefaultLanguage컬럼의 제약조건이 NOT NULL로 지정된다.

4.2 기본키/외래키 지정

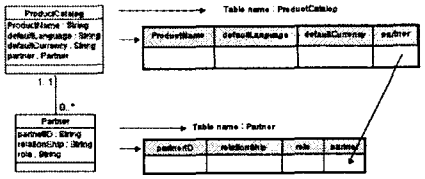
DTD에서 요소에 따른 클래스가 하나 이상 추출되었다. XML 문서에서는 계층적 구조이기 때문에 클래스 사이에 계층적으로 지정해 주어야 한다. 클래스에서는 참조형을 이용하여 계층적 구조로 만들어진다. 따라서 테이블간의 계층적 구조로 만들어야 하는데,



[그림 15] 엘리먼트들을 객체화하여 테이블 스키마로 변환하는 과정

테이블에서는 기본키/ 외래키를 이용하여 계층적인 테이블 관계를 만들어 준다. [그림 14]에서는 단순히 객체화된 데이터를 테이블로 직접 변환하여 독립적인 테이블로 만들었다. 따라서 [그림 14]에서 계층적으로 테이블 관계를 표현해 보면 다음 그림과 같다.[그림 16]

[그림 16]에서는 테이블 ProductCatalog에서 외래키로 Partner 컬럼을 지정한다. 테이블 ProductCatalog에 있는 Partner 컬럼은 테이블 Partner에 연결하려면 Partner컬럼을 추가시키면 된다. 따라서 테이블 ProductCatalog에서 외래키를 선언해 주므로 계층적 구조가 이루어진다. 기본키 선언은 테이블 ProductCatalog는 ProductName컬럼이 기본키로 선언되었고, 테이블 Partner는 Partner 컬럼과 PartnerID 컬럼이 합쳐서 기본키로 선언되었다.



[그림 16] 테이블마다 기본키/ 외래키 지정

5. 결론

XML은 웹에서 데이터 교환을 위한 표준 마크업언어이며, DTD는 XML 문서 내에서 사용 가능한 요소를 관리하고, 사용할 수 있는 각 요소형의 내용과 특성을 지정하는 규칙들을 담고 있다. 본 논문에서 XML DTD를 RDB 스키마로 변환하는 방법을 다루었으며, 테이블 기반 매핑과 객체 기반 매핑을 통하여 복잡한 구성의 객체도 RDB 스키마로 변환이 가능함을 보여주었다. 이로 인하여 XML 문서를 객체로 변환하고 변환된 객체를 RDB 스키마로 변환함으로서 웹 환경에서 데이터베이스를 효율적으로 구성할 수 있다.

참고문헌

[1] World Wide Wed Consortium. The XML Data Model. http://www.w3.org/XML/Datamodel.html, Jan 2000.

[2] Michel Goosens and Janne Saarela. A Practical Introduction to SGML. In TUGboat. volume 16(3),1995

[4] World Wide Web Consortium. XML Linking Language(XLink). TechnicalReportWD-xlink-2000021,W3C, February 2000.

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

[6] Namespaces in XML, Tim Bray, David Hollander, Andrew Layman. See http://www.w3.org/TR/REC-xml-names/

[7] 김명주. "XML and Java", 이한출판사

[8] Frank Boumphrey외 11인 저 / 류광 역, "Professionnal XML Applications", 정보문화사

[9] Alexander & Tom 저 유진희, 박성준 역, "Professional Java XML Programming", "정보문화사"

[10] 이상태, 주경수. "제약조건 유지를 위한 XML DTD의 관계 스키마로 변환 방법", 「한국인터넷정보학회」 제1권 2호, 2000, pp.189-196

[11] Sandra E. Eddy & John E. Schnyder, "Teach Yourself XML", IDG BOOKS

[12] Simon North 저 노정운 역, "초보자를 위한 XML 21일 완성", 인포북.