UML 객체 모델을 이용한, 객체-관계형 데이터베이스 기반의 XML 응용 설계 방법

임종선*, 주경수**
순천향대학교 전산학과
e-mail:*ronmer74@hanmail.net, **gsoojoo@sch.ac.kr

A Design Methodology for XML Applications based on ORDB using UML Object Models

Lim Jong-Seon, Joo Kyung-Soo
Dept. of Computer Science, College of Engineering Soonchunhyang University

요 약

현재 B2B 전자상거래와 같은 XML을 이용한 정보교환이 확산되고 있으며, 이에 따라 상호 교환되는 정보에 대하여 체계적이며 안정적인 저장관리가 요구되고 있다. 이를 위해 XML 응용과 데이터베이스 연계를 위한 다양한 연구가 관계형 데이터베이스를 중심으로 수행되었다. 그러나 계층구조를 갖는 XML 데이터를 2차원 테이블의 집합인 관계형 정보로 표현하는 관계형 데이터베이스로의 저장에는 본질적인 한계가 있으므로, 해결책으로 계층구조를 지원하는 ORDB로의 저장이 요망되고 있다. 따라서 계층구조를 갖는 XML 데이터를 ORDB로 저장하기 위한 모델링 방안이 요구된다. 양질의 어플리케이션 시스템을 구축하기 위해서는 우선적으로 모델링이 중요하다. 1997년에 OMG는 표준 모델링 언어로 UML를 채택하였고, 이에 따라 UML은 보다 널리 사용되고 있다. 그러므로 효율적인 XML 어플리케이션을 개발하는데 UML을 기반으로 한, 설계 방법론이 필요하다고 할 수 있다. 본 논문에서는 UML을 이용하여, ORDB 기반의 XML 응용을 위한 통합 설계 방법론을 제안한다. 이를 위하여 먼저 UML을 이용하여 XML DTD를 설계하기 위한 XML 모델링 방안을 제시하고, 아울러 교환되는 XML 데이터를 효율적으로 저장하기 위하여 ORDB 스키마 설계를 위한 데이터 모델링 방법을 제안한다.

1. 서 론

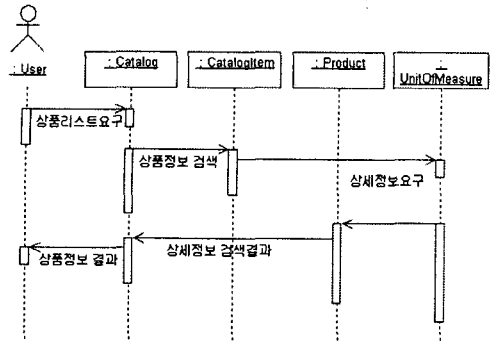
XML은 구조화된 정보를 포함하고 있는 문서들을 위한 마크업 언어이다. 구조화된 정보는 구체적인 내용과 그 내용이 수행해야할 역할을 포함하고 있다[1]. 한편 XML 어플리케이션과 데이터베이스 시스템 사이의 원활한 연계를 위해서, 그 동안 XML DTD를 관계형 데이터베이스 스키마로 변환하는 방법에 대해서는 많은 연구가 진행되었다. 본 논문에서는 UML을 기반으로 하여 XML DTD를 산출하고, 그 결과를 이용하여 ORDB 스키마로 저장하는 통합 모델링에 대한 연구를 하였다. 본 논문에서는 2장에서 UML를 이용한 통합 모델링, 3장에서는 XML 모델링, 4장에서는 데이터 모델링 그리고 마지막 5장에서는 결론을 기술한다.

2. UML을 이용한 모델링

UML은 시스템 개발 과정에서 객체지향 시스템의 결과물을 명세화하고, 시각화하고, 문서화하기 위하여 사용되는 모델링 언어로서, 다른 많은 방법론의 우수한 개념을 통합한 것이다. 또한 객체지향 방법론에 사용되는 표기법을 통합함으로써 객체지향 분석과 설계 분야에서 표준을 위한 기초를 제공하였다. UML을 이용해 유스케이스에 의한 순차 다이어그램을 도출하여 클래스 다이어그램을 만든 후 그 클래스 다이어그램에 의해서 그림 1과 같이 XML 모델링과 데이터 모델링으로 변환할 수 있다.

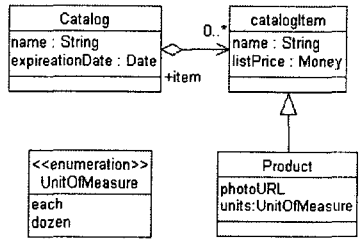
(그림 1) UML을 이용한 XML 모델링과 데이터 모델링

그림 2의 순차 다이어그램은 객체들간의 메시지 과정을 설명한다. 이 다이어그램은 메시지 객체들과 진행 과정에 관계된 액터들을 포함하고 있다[7].



(그림 2) 메시지 객체와 순차 다이어그램

그림 2에서 객체들과 액터들 이외에 수직의 축은 시간을 표현한다. 화살표는 행위 또는 객체와 액터들간에 발생하는 사건을 표현한다. 그림 3의 클래스 다이어그램은 4개의 클래스, 그리고 그들의 연관된 속성과 서브-객체를 예를 들어 설명한다.



(그림 3) 클래스 다이어그램

(표 1) 객체 정의

클래스	기술
Catalog	상품목록의 이름과 그 정보에 대한 만기일자를 가지고 있다. 사용자가 특정 카탈로그의 목록을 원하면, CatalogItem 객체를 통해서 정보를 가져온다.
Catalog Item	특정한 목록의 이름과 가격정보를 가지고 있다. 자식 클래스인 Product를 이용하여 각 상품에 대한 정보를 표현할 수 있다.
Product	CatalogItem에서 상속받은 속성과 자신이 가지고 있는 속성을 이용하여, 하나의 제품에 대한 여러 가지 정보를 표현한다.

그림 3은 Catalog 와 CatalogItem간에 관계성에서 연관 관계를 갖는다. 왜냐하면 CatalogItem은 Catalog 객체에 의존하고 있기 때문이다. 또한 CatalogItem은 Product의 부모 클래스가 된다. 이것은 Product 클래스에서 CatalogItem의 일부 속성을 상속받기 위해서이다. 속성의 상속은 하위 클래스에서 유용하게 사용되기 때문이다. 표 1은 각 클래스의 객체의 특성을 서술한 것이다. 또한 각각의 객체에 대한 속성, 사상수, 관계성은 표 2와 같다.

(표 2) 객체들의 속성(서브-객체), 사상수, 관계성

클래스	속성	사상수	관계	기술
Catalog	name	1..1	복합	상품의 목록을 기술
	expiration Date	1..1	복합	만기일자를 기술
Catalog Item	name	1..1	복합	세부 상품이름을 기술
	ListPrice	1..1	복합	상품의 가격을 기술해
Product	PhotoURL	1..1	복합	상품의 사진이 있는 URL을 기술
	Units	1..1	복합	단위 수량을 기술

3. UML 클래스로부터 XML DTD 변환

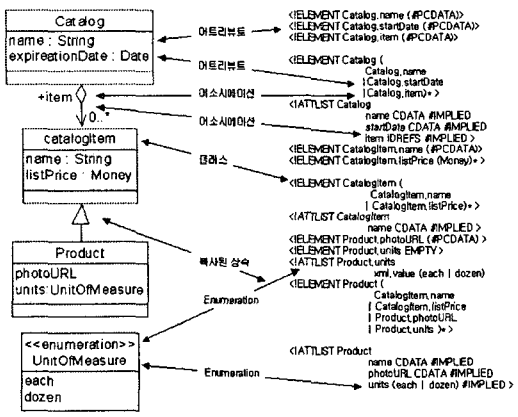
기존 명세서의 XMI(XML Metadata Interchange) 생성 규칙은 DTD를 관대하게 처리하거나 강제적으로 느슨하게 생성시킨다. 이 선택은 DTD의 표현에 제한을 두었기 때문에 부분적인 모델의 변환을 가능하게 만들었다. 반면에, UML에서 엄격한 DTD의 생성을 가능하게 한다. UML 스테레오 타입은 디자이너의 컨트롤을 지원하는데, 이것은 엄격한 DTD는 표준화된 느슨한 DTD의 서브셋이기 때문이다[2][7].

3.1 느슨한 DTD(Relaxed DTD)

느슨한 DTD는 XMI 1.1 표준에서 명시한 법칙과 동등하다. 느슨함을 확인할 수 있는 DTD의 일반적인 특징은 다음의 세 가지와 같다. 첫째, 모델의 부분을 교환하는데 자유로운 다양성을 허락한다. 둘째, 자유로운 목록 모델은 문서 생성의 제한이 없도록 강요한다. 셋째, UML을 XML의 구조로 매핑 할 때, 엘리먼트나 어트리뷰트를 자유롭게 선택한다. 그림 4는 느슨한 DTD 매핑을 나타낸 것이다[2][3].

3.2 엄격한 DTD(Strict DTD)

UML 모델에 의하여 다양성이 제한되며, 순차적으로 정렬된 목록 모델은 다양성을 요구한다. 또한 엘리먼트나 어트리뷰트가 아닌 XML 엘리먼트만 생성된다. 엄격한 DTD는 UML 모델의 어트리뷰트나 관계의 규칙에 대한 다양성을 명시한다. 엄격한 DTD의 규칙은 UML 스테레오 타입의 확장을 사용하지 않고, 모든 UML 클래스 모델로부터 DTD를 생성할 것이다[2][3].



(그림 4) UML을 XML의 느슨한 DTD로 매핑

4. 데이터 모델링

데이터베이스 스키마 변환 과정에서 정보 구조로부터 논리적 개념을 이용하여 어떤 논리적 구조로 표현하는 것이 필요한데, 데이터 모델링이란 이러한 변환과정을 말한다[8].

4.1 ORDB 변환 방법

그림 3의 클래스 다이어그램을 ORDB Schema로 변환하는 방법은 다음과 같다[4][5].

① 객체 클래스는 구조화된 객체 타입을 생성하며, 테이블과 연결하여 타입을 1개 또는 여러 개의 테이블에 객체의 원소가 되도록 한다.

② 객체 타입은 사용자 정의 타입이거나, 속성을 가진 객체 타입이다.

③ 클래스에서의 객체 속성은 객체 타입에서의 속성이다.

④ 타입이 테이블이나 객체 또는 특정 타입으로 변환 할 때, 클래스의 객체 속성 타입은 객체 타입의 속성 타입이다.

⑤ 객체 속성이 NULL제약조건을 가지면 NOT NULL 제약조건도 가진다.

⑥ 객체 속성이 초기 값을 가지며 컬럼에 DEFAULT 값을 더한다.

⑦ 독립적인 클래스나 함축적인 성질을 가진 클래스들을 위해 기본키로 정수를 생성하고 {oid}를 위해 태그된 컬럼에 기본키 제약조건에 따라 {oid}를 더한다.

⑧ 부 클래스를 위해 기본키 제약조건과 외래키 제약조건에 따라 각각의 부모 클래스의 키를 더한다.

⑨ 클래스들의 결합을 위해 객체 타입을 생성하고 기본키 제약조건과 외래키 제약조건에 따른 역할 테이블로부터 기본키를 추가한다.

⑩ 교환 oid가 <n>인 경우 UNIQUE 제약조건에 따른 컬럼을 추가한다.

⑪ 각각의 확실한 제약조건을 위해 CHECK를 실시한다.

⑫ 결합하는데 있어서 각각의 0..1, 1..1 역할을 위한 참조 테이블에서 외래키 컬럼을 생성한다. 교대로 단일 객체들이나 컬렉션을 위한 그 자신 안에서 속성으로 객체를 선언하기 위하여 객체 타입에 참조를 사용하거나 또는 다중관계 객체를 위한 참조의 배열로 객체를 선언하기 위하여 객체 타입에 참조를 사용한다.

⑬ 집합 테이블에 외래키를 가진 다중 집합을 위해 기본키를 생성하고, 기본키를 위한 컬럼을 추가하며, 테이블 안에서 그 집합을 저장하기 위한 객체 타입을 사용한다. 또한 단일 객체를 위한 객체 속성 또는 컬렉션을 통하여 배열이나 중첩 테이블을 사용한다.

⑭ 너무 많이 이동하게 되는 이진 결합 클래스들을 최적화 한다.

⑮ 외래키를 사용하여 결합이 안된 클래스와 함께 N : N 결합의 관계 테이블을 생성한다.

⑯ N : N 결합에서 역할 테이블의 키로부터 기본키, 외래키의 제약조건을 생성한다.

⑰ 객체 클래스의 전달 작용을 위한 객체 타입에 메소드를 생성한다.

4.2 ORDB 스키마 변환 예

그림 3과 표 1 및 표 2에 의한 UnitOfMeasure 객체는 ORDB 변환 방법 ③, ⑤번의 성질에 따라 UnitOfMeasure_t 객체타입 속성을 저장하는 UnitOf Measure 테이블로 그림 5와 같이 변환된다.

```
SQL> CREATE TYPE UnitOfMeasure_t AS OBJECT
(
  each    string,
  dozen   string
);
SQL> CREATE TABLE UnitOfMeasure OF
       UnitOfMeasure_t
(
  each    NOT NULL,
  dozen   NOT NULL
);
```

(그림 5) UnitOfMeasure 테이블

그림 3과 표 1 및 표 2에 의한 Product 객체는 ORDB 변환 방법 ③, ④, ⑤의 성질에 따라 Product

_t 객체타입 속성을 저장하는 Product 테이블로 그림 6과 같이 변환된다.

```
SQL> CREATE TYPE Product_t AS OBJECT
(
  photoURL          string,
  units             UnitOfMeasure_t
);
SQL> CREATE TABLE Product OF Product_t
(
  photoURL          NOT NULL,
  units             NOT NULL
);
```

(그림 6) Product 테이블

그림 3과 표 1 및 표 2에 의한 CatalogItem 객체는 ORDB 변환 방법 ③, ④, ⑤, ⑫의 성질에 따라 CatalogItem_t 객체타입 속성을 저장하며 속성 CatalogItem의 객체타입은 그림 7과 같이 테이블로 변환한다. 배열을 이용한 것은 0..* 관계에서 *는 무한정의 매핑이지만, 실제로 데이터베이스 테이블에 무한정으로 매핑을 할 수가 없다. 그래서, 사용자 정의에 따라 한정된 배열을 지정해 준다.

```
SQL> CREATE TYPE CatalogItem_t AS VARYING ARRAY
(40) OF CatalogItem
(
  name              string,
  listPrice         integer
  );
SQL > CREATE TABLE CatalogItem OF CatalogItem_t
(
  name              NOT NULL,
  listPrice         NOT NULL
);
```

(그림 7) CatalogItem 테이블

그림 3과 표 1 및 표 2에 의한 Catalog 객체는 ORDB 변환 방법 ③, ④, ⑤, ⑧의 성질에 따라 Catalog_t 객체타입 속성을 저장하며 CatalogItem의 객체타입은 그림 7에서 정의한 CatalogItem_t를 적용하고 그림 8과 같이 테이블로 변환한다.

```
SQL> CREATE TYPE Catalog_t AS OBJECT
(
  CatalogID         integer,
  CatalogItemID     integer,
  name              string,
  expireationDate   date
);
SQL> CREATE TABLE Order OF Order_t
(
  PRIMARY KEY       (CatalogID, name),
  CatalogID         REFERENCES Catalog,
  CatalogItem       REFERENCES CatalogItem,
  name              NOT NULL,
  expireationDate   NOT NULL
);
```

(그림 8) Catalog 테이블

5. 결 론

XML을 이용하여 상호 교환되는 정보를 체계적이고 안정적으로 저장·관리하기 위해서 XML 응용과 데이터베이스 연계를 위한 다양한 연구가 지금까지 관계형 데이터베이스를 중심으로 수행되었다. 그러나 XML 응용에서 DTD가 다양한 데이터타입을 정의할 수 없는 한계 때문에 데이터베이스에 매끄럽게 연계시키기 어려운 점이 있으며, 또한 계층구조를 갖는 XML 데이터를 2차원 테이블의 집합인 관계형 데이터베이스로의 저장에는 본질적인 한계가 있다.

본 논문에서는 계층구조를 갖는 XML 데이터를 계층구조를 지원하는 ORDB로 저장이 가능하도록 하는 통합 모델링 방법론을 제안하였다. 이를 위하여 우선적으로 객체지향 설계언어인 UML를 이용해서 클래스 다이어그램을 도출한 후, 클래스 다이어그램에 의해서 XML DTD 생성을 위한 XML 모델링을 제안하였다. 또한, 이들의 모델링으로 교환되는 XML 데이터의 효율적인 저장을 위하여 ORDB 스키마 설계를 위한 데이터 모델링 방법을 제안하였다.

참 고 문 헌

[1] http://www.xml.com/lpt/a/98/10/guidel.html.

[2] World Wide Web Consortium. Extensible Mark up Language(XML)1.0 W3C Recommendation, 10 Febury 1998. see http://www.w3.org/TR/REC-xml

[3] Elliotte Rusty Harold , XML in a Nutshell - A Desktop Quick Reference, O'reilly, 2001

[4] Robert J. Muller, "Database Design for Smarties", Morgan Kaufmann Publishers, 1999

[5] 이상태, 주경수, "객체모델을 이용한 XML DTD의 ORDB 스키마로의 변환", 정보기술과 데이터베이스 저널, 제8권 제11호, pp.111-112, 2001.

[6] 조완수, UML 객체지향 분석 설계, 홍릉과학출판사, 2000.

[7] 김채미, 최학열, 김심석 공저, 전문가와 함께 가는 XML Camp, 마이트 Press, 2001.

[8] 이석호, 데이터베이스론, 정익사, 1999.