

## 객체-관계형 데이터베이스 응용을 위한 XML Schema 설계방법론 개발

최문영<sup>†</sup>, 주경수<sup>‡‡</sup>

### 요 약

B2B 전자상거래와 같이 XML을 이용한 정보 교환이 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 있어, 각 구조화된 정보를 객체-관계형 데이터베이스로 저장하기 위한 모델링 방안이 요구된다. 양질의 어플리케이션 시스템을 구축하기 위해서는 우선적으로 모델링이 중요하다. 1997년에 OMG는 표준 모델링 언어로 UML을 채택하였고, 이에 따라 UML은 보다 널리 사용될 것이다. 그러므로 효율적인 XML 어플리케이션을 개발하는데 UML에 기반을 둔 설계 방법론이 필요하다고 할 수 있다. 본 논문에서는 UML을 이용하여 객체-관계형 데이터베이스기반의 XML 응용을 위한 설계 방법론을 제안한다. UML을 도출해 내기 위한 체계적인 3단계 방법론을 제안하고, 만들어진 UML을 이용하여 W3C XML Schema를 설계하기 위한 XML 모델링 방안을 제시한다. 아울러 교환되는 XML 데이터를 효율적으로 저장하기 위하여 객체-관계형 데이터베이스 스키마 설계를 위한 데이터 모델링 방법을 제안한다.

### Developing an XML Schema Design Methodology for Object-Relational Database Applications

Choi Mun-Young<sup>†</sup>, Joo Kyung-Soo<sup>‡‡</sup>

### ABSTRACT

Nowadays the information exchange based on XML such as B2B electronic commerce is spreading. Therefore a systematic and stable management mechanism for storing the exchanged information is needed. For this goal there are many research activities for centering on relational databases the connection between XML application and databases. But when XML data which has hierarchical structure is stored as relational databases which are expressed as relational information, a set of 2-dimensional table, there is a limitation essentially. Accordingly the modeling methodology for storing such structured information in the form of object-relational databases is needed. In order to build good quality application systems, modeling is an important first step. In 1997, the OMG adopted the UML as its standard modeling language. Since industry has warmly embraced UML, its popularity should become more important in the future. So a design methodology based on UML is needed to develop efficient XML applications. In this paper, we propose a unified design methodology for XML applications based on object-relational database using UML. To reach these goals, first we propose a systematic three step methodology to extract UML, second we introduce a XML modeling methodology to design W3C XML schema using UML and third we propose a data modeling methodology for object-relational database schema to store exchanging XML data efficiently.

**Key words:** UML(통합모델링언어), Object-Relation Database(객체-관계형 데이터베이스), XML, XMI

※ 교신저자(Corresponding Author) : 최문영, 주소 : 충청남도 아산시 신창면(336-745), 전화 : 041)630-5125, FAX : 041)631-4405, E-mail : griffin@hyejeon.ac.kr

접수일 : 2003년 9월 22일, 완료일 : 2004년 1월 27일

<sup>†</sup> 준희원, 순천향대학교 대학원 전산학과 박사과정

<sup>‡‡</sup> 순천향대학교 정보기술공학부 교수  
(E-mail : gsoojoo@sch.ac.kr)

## 1. 서 론

XML은 최근 웹 상에서 데이터 교환의 표준 형식으로 주목을 받고 있으며 사용자가 문서의 구조를 정의 할 수 있기 때문에 다양한 형태의 데이터가 XML 문서로 표현될 수 있다. 이러한 이점은 XML이 기종 컴퓨터 사이에서 데이터 교환의 매체로 사용될 수 있다는 것을 의미한다. XML은 XML DTD나 XML Schema를 통하여 문서 자체에 문서의 구조를 기술하고 있다. 즉, 문서의 구조를 사용자가 원하는 대로 정의 할 수 있으며, 이러한 구조적 유동성은 다양한 형태의 데이터를 XML로 표현할 수 있게 해준다. 웹에서 운용되는 데이터가 동일한 형태로 저장, 처리 될 수 있음을 의미한다.

이러한 다양한 형태의 XML 문서를 효율적으로 저장·관리하고, 이를 문서로부터 유용한 정보를 추출하기 위해서는 데이터베이스 시스템의 활용이 필수적이며, 현재 이와 관련된 연구분야 중 XML 데이터 형식을 데이터베이스 스키마로 변환하고자 하는 연구가 활발히 진행되고 있다. 이러한 연구 결과들은 각각의 특징과 단점을 가지고 있으며, 데이터베이스 시스템에서도 이러한 방법들을 수용하여 선택적으로 사용할 수 있도록 하는 기능을 제공하고 있다.

기존의 연구들은 관계형 데이터베이스 기반으로 변환하는 연구들이 많고, 이러한 변환들의 단점은 웹 상에서의 멀티미디어 데이터들을 저장하기에는 어려움이 있다는 것이다. 그리고 현재 데이터베이스 업체는 관계형 데이터베이스에서 객체-관계형 데이터베이스로 변화해 가는 추세이다.

그러므로, 본 논문에서는 객체지향 설계언어인 UML을 통한 변환방법을 제안한다. UML을 객체지향적인 XML과 객체-관계형 데이터베이스로 변환하는 것은 데이터의 손실을 줄이고 저장·관리를 용이하게 한다. 본 논문은 그림 1처럼 3단계의 모델링 방법을 제안하고 있다. 모델링 방법은 개념적 모델링, 논리적 모델링, 물리적 모델링으로 이루어져 있고, 마지막 단계인 물리적 모델링에서는 논리적 모델링에서 제안한 UML 클래스 다이어그램을 이용하여 XML 모델링과 데이터 모델링을 제안한다. 먼저 2장에서는 관련 연구들을 살펴보고, 3장에서는 실세계에서 사용되는 정보들을 추출하여 체계적으로 정리하고, 4장에서는 정리된 자료를 UML 클래스 다이어그램으로 작성한다. 그리고 5장에서는 작성된 UML

클래스 다이어그램을 사용하여 XML Schema와 객체-관계형 데이터베이스로 변환을 기술하고 마지막으로 6장에서 결론을 맺는다.

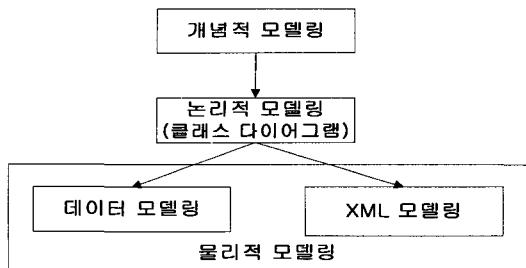


그림 1. 3단계 모델링

## 2. 관련 연구

XML은 구조화된 정보를 포함하고 있는 문서들을 위한 마크업 언어이다. 구조화된 정보는 구체적인 내용과 그 내용이 수행해야 할 역할을 포함하고 있다 [1]. XML Schema에 대한 연구는 W3C Recommendation으로 채택된 W3C XML Schema가 표준 XML Schema의 스페이 될 가능성이 가장 높은 설정이다. W3C XML Schema는 XML DTD보다 다양한 데이터 타입을 정의할 수 있고 강력한 표현력을 이용하여 다양한 어플리케이션으로 사용하기에 편리한 장점을 가지고 있다. 또한 attribute, element, 사용자 정의 데이터 타입에 대해서 상속하는 메커니즘을 갖고 있다. 그러므로 XML 관련자들의 많은 관심을 끌고 있는 것은 사실이다[2,3].

한편 XML 어플리케이션과 데이터베이스 시스템 사이의 원활한 연계를 위해서, 그동안 XML DTD를 관계형 데이터베이스 스키마로 변환하는 방법에 대해서는 많은 연구가 진행되었다. 그러나 UML 클래스를 이용하여 W3C XML Schema를 모델링하고 그 모델링으로 교환되는 데이터를 RDB로 저장하는 통합 설계 모델링에 대한 연구는 미비한 실정이다. XML이 컨텐츠에서 데이터베이스로까지 그 적용 분야가 확장되면서 XML로 표현된 정보들을 어떻게 효율적으로 저장하고 관리할 것인지에 대한 기술도 XML과 함께 발전되어 가고 있다. 가장 큰 이슈 중의 하나는 기존의 DBMS로도 XML 파일을 효율적으로 관리할 수 있는가 이다.

UML을 이용한 XML 모델링에 대한 연구는 UML

를 XML Schema로 변환할 때 UML 확장 메커니즘을 사용하여 XML 구조를 표현하였다[4,5]. 또한 UML 클래스 다이어그램을 XML DTD로 자동 변환하고 XML DTD를 XML Schema로 자동 변환하는 도구(tool)의 개발은 모델을 설계하는 자가 XML Schema 구문에 익숙하지 못해도 XML 모델링을 신속하게 하고 개별적으로 서로 다른 언어 또는 환경에서 부분적 모델 어휘들을 재 사용할 수 있게 하였다[6].

UML을 기반으로 하여 XML 형식으로 자동 변환하는 부분은 XMI(XML Metadata Interchange)라는 이름으로 표준화를 진행 중이며 UML의 각종 다이어그램들이 XMI에서 정한 규정에 따라 결국은 XML로 표현되게 된다[7,8].

많은 논문들이 복잡한 XML 문서를 관계형 데이터베이스에 저장하는데 초점을 맞추어 데이터베이스와 XML 문서의 연결에 대하여 발표되었다. 대표적인 예로 Ronald Bourret의 방법을 살펴보면 XML Schema를 UML 클래스 다이어그램을 통하여 관계형 데이터베이스로 변환하는 알고리즘을 제안하였다[9]. 다른 예를 보면 XML DTD에서 직접 관계형 데이터베이스로 변환하는 연구가 발표되었다.

현재 이러한 연구들은 관계형 데이터베이스보다는 객체지향 데이터와 기존에 존재하는 데이터를 함께 처리 할 수 있는 객체-관계형 데이터베이스로 변해가고 있고, XML에서는 DTD 보다 데이터의 형을 다양하게 지원하는 XML Schema로 바뀌고 있는 실정이다.

그러므로, 본 논문에서는 객체-관계형 데이터베이스와 XML Schema를 설계하는 개념적, 논리적, 물리적 모델링인 일괄된 방법론을 제안했다. 본 논문에서의 방법론은 데이터들의 효율적이고 일괄적인 저장·관리를 할 수 있고, 나아가 다른 응용 영역에 쉽게 적용시킬 수 있다.

### 3. 개념적 모델링

개념적 모델링은 어떠한 DBMS와도 독립적으로 기술되는 단계이다. 많은 개념적 모델링 언어들은 도메인의 본질에 초점을 맞추는데 노력한다.

개념적 모델링에서의 그래프는 개념과 도메인의 관계를 나타내는 것으로 사용된다. 개념적 모델링을 하는 이유 중에 하나는 정확한 도메인 정보를 추출하여 데이터베이스를 개발하는데 사용하기 위해 있다.

개념과 서로간의 관계성은 그래프를 세분화하여 도메인의 근본적인 성질을 충분히 나타낸다.

그림 2는 개념과 도메인에 대하여 간단한 개념적 그래프이다.

개념과 관계성은 그래프의 노드로 표현되고 개념과 관계성의 특징은 에지로 표현한다. 예를 들면, 이름, 크기, 색, 형, 버전, 상태, 객체의 구조는 에지로 표현되고 객체, 개념, 관계는 그래프의 노드로 모두 표현된다.

개념적 그래프는 개념과 도메인의 관계성이 그래프로 표현되는 것이다. 도메인을 그래프로 표현하는 방법은 다음과 같다.

- ① 그래프는 노드, 에지, 그리고 카디널리티(cardinality:계량수)의 집합으로 이루어져 있다.
- ② 두 개의 개념과 개념의 특성 그리고 도메인의 이진 관계는 에지로 연결한다.
- ③ 카디널리티 한 쌍은 완전한 정수 에지로 연결된다. 카디널리티는 정수대신에 조건 “Many (다수)”로 표현한다. 다수 카디널리티는 0 또는 더 많은 출현을 언급하는 것이다. 그리고 카디널리티 “one-or-more”는 “+”에 의해서 나타내어진다. 만약 카디널리티가 없으면 “1”로 가정한다.

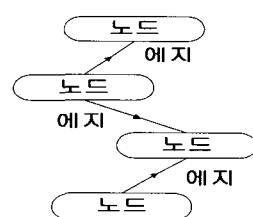


그림 2. 간단한 개념적 그래프

#### 3.1 개념적 그래프 모델링

그림 3처럼 7단계 처리를 통한 개념적 그래프를 모델링하는 알고리즘을 제안한다. 이 처리에서 하나의 이점은 데이터베이스 시스템에 대한 지식이 없이도 실행될 수 있다는 것이다.

예를 들어, 회사(Company)에 대한 도메인을 만들고 그래프를 그려보면 다음과 같다.

1. 도메인 개념과 복잡한 관계성의 리스트
  - 회사 (Company)

- ① 도메인 개념과 복잡한 관계성을 리스트로 만든다. 개념은 실사회의 객체, 관계, 또는 사건이다.
- ② 도메인에 관계성을 기술한 문장을 생성하는 것으로 도메인 개념을 연결한다. 그 문장은 “사원은 이름을 가지고 있다”, “문장은 글꼴을 가지고 있다”, 또는 “관리자는 사원을 관리한다”와 같은 도메인 개념의 특성을 기술하거나 관련 단어 또는 어구와 두 개의 도메인 개념을 연결한다. 관련 어구는 두 개의 도메인 개념에 대한 설명이다. 만약 관계성이 두 개념보다 더 많이 요구되면, 단계 ①의 리스트에 관계성을 추가할 수 있다.
- ③ 리스트로부터 도메인의 중요한 개념을 선택한다.
- ④ 개념 스키마에서 중요한 개념을 노드로 표현한다.
- ⑤ 그래프에서 에지로 간단한 문장을 표현한다. 도메인 개념의 특성을 그리는 2가지 방법이 있다. 가장 완전한 방법은 타원형에 새로운 개념을 표현하는 것으로써 특성을 그리는 것이다. 그리고 지시된 에지의 특성에 도메인 개념을 연결하고 특성에 대한 이름으로 에지에 표현한다. 다음 방법으로 특성은 도메인이 개념적 모델링을 위해 명백한 특성을 표현하는 타원형을 생략하는 것이다. 두 개의 개념 화살표사이에 에지를 그릴 수 있다. 예를 들면, 에지에 “관리자 관리 사원”은 “관리자”에서 “사원”까지의 “관리”를 표현한다. 그 개념이 단계 ①의 리스트에 모두를 표현하기 위하여 에지를 그릴 때 그래프에 추가적인 개념을 추가하는 것이 필요하다.
- ⑥ 에지에 카디널리티 상태를 추가하고 도메인 그래프에 카디널리티 상태를 추가한다. 논리적 설계에서 카디널리티 상태를 추가 할 수도 있다. 에지와 개념사이의 연결된 관계에서 일어날 수 있는 반복 횟수에 대하여 숫자로 표현한다. 카디널리티 “다수”는 “\*”로 표현한다. 그리고 카디널리티 “1”은 생략할 수 있다. 만약 카디널리티 “zero-or-one”를 필요로 하면, 기호 “0”으로 나타낸다.
- ⑦ 도메인을 조사하여 스키마를 수정한다. 이 처리는 데이터베이스의 구현에 대한 근거로 개념 스키마를 설명하기 위한 예비 스키마가 될 것이다.

그림 3. 개념적 그래프 모델링 알고리즘

- 부서 (Department)
  - 부서 이름 (Department name)
  - 사무실 (Office)
  - 사무실 주소 (Office address)
  - 사원 (Employee)
  - 사원 이름 (Employee name)
  - 사원 ID (EmployeeID)
  - 사원 직위 (Employee title)
  - 사원 연락정보 (Contact Information)
  - 사원 연락정보 주소 (Contact Information address)
  - 개인 기록 (Personal Record)
  - 주민번호 (JuminID)
2. 도메인에 관계성을 기술한 문장을 생성하는 것으로 도메인 개념을 표현한다. 문장은 제한적인 문장 형식으로 도메인을 표현하기 때문에 문장 표현이 잘 못될 수도 있다.
- 회사는 부서를 가지고 있다.
  - 회사는 사무실을 가지고 있다.
  - 회사는 ID를 가지고 있다.
  - 부서는 이름을 가지고 있다.
  - 사무실은 주소를 가지고 있다.
  - 사무실은 ID를 가지고 있다.
  - 부서는 부서원이 있다.
  - 부서는 ID를 가지고 있다.
  - 사원은 부서와 부서원을 관리하는 관리자가 있다.
- 사원은 이름을 가지고 있다.
  - 사원은 ID를 가지고 있다.
  - 사원은 직위를 가지고 있다.
  - 사원 연락정보는 주소를 가지고 있다.
  - 개인 기록은 주민번호를 가지고 있다.
3. 리스트로부터 도메인에 중요한 개념을 선택한다.
- 회사 (Company)
  - 부서 (Department)
  - 사무실 (Office)
  - 사원 (Employee)
  - 연락정보 (Contact Information)
  - 개인기록 (Personal Record)
4. 그림 4처럼 개념 스키마의 노드로써 추가 되는 개념들을 그린다.
5. 그림 5처럼 그래프의 에지로 문장을 표현한다. 각각의 문장은 하나의 에지가 된다.
6. 그림 6처럼 에지에 카디널리티 상태를 추가한다.
- 
- ```

graph TD
    Company --- Department
    Company --- Office
    Company --- Employee
    Company --- ContactInformation
    Company --- PersonalRecord
    Department --- Employee
    Office --- Employee
    ContactInformation --- Employee
    PersonalRecord --- Employee
  
```

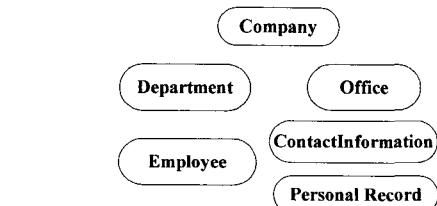


그림 4. 회사(Company)에 대한 도메인 리스트에서 중요한 개념을 선택한 개념적 그래프

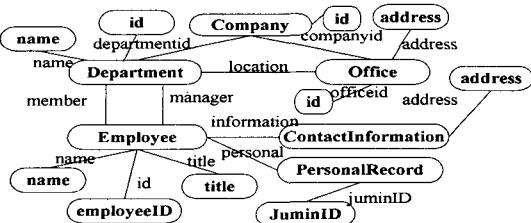


그림 5. 회사 개념적 그래프

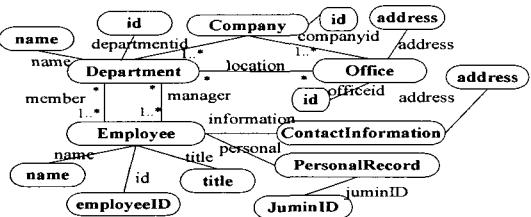


그림 6. 카디널리티 상태를 가진 회사 개념적 그래프

7. 도메인을 조사하여 스키마를 수정한다. 추가적인 정보는 각각의 도메인에 추가할 수 있다.

#### 4. 논리적 모델링

논리적 모델링 단계는 데이터베이스 관리 시스템의 데이터 모델을 위해 개념 스키마를 매핑하는 과정이다. 데이터베이스 설계자가 개념적 모델링에서의 도메인을 파악하고 나면, 데이터베이스 관리 시스템의 데이터 모델은 유일한 모델링 언어로 사용될 것이다. 논리적 데이터 모델은 개념적 모델링보다 기능성과 이상에 더 많이 제한적이다.

논리 스키마 형식의 작성은 개념적 그래프로부터 객체 모델로 표현할 수 있다.

##### 4.1 객체 모델

개념적 그래프에서 객체 모델까지 변환하기 위한 알고리즘은 그림 7에 있다.

개념적 그래프를 객체 모델로 변환하기 위하여 중간 다이어그램을 생성하는 것은 클래스 다이어그램을 만드는데 매우 도움이 될 수 있다. 카디널리티를 추가한 그래프는 도메인 객체 다이어그램과 집합 트리를 만드는데 두 가지의 유용한 다이어그램이다. 도메인 객체 다이어그램의 중심적인 개념은 클래스 다이어그램에서의 클래스로써 그려진다.

도메인 객체 다이어그램은 단순한 다이어그램으

- ① 카디널리티의 특성에 의하여 개념적인 모델을 수정한다. 만약 특성을 알고 있으면 숫자의 범위 또는 세트를 사용하고 0 또는 다수의 상태를 구별한다.(“\*” 다수를 표현). 특성들은 “0”, “0..1”, “?”, “opt”, 또는 dashed-edge 선으로 표시될 수 있다. 만약 이런 일이 발생하면 특성의 여러 가지 구성들을 설명하는 선을 다이어그램에 추가한다.
- ② 특성을 가지고 있는 각각의 개념은 클래스다이어그램에서 클래스가 된다.
- ③ 카디널리티가 1보다 더 많거나 또는 추가적인 도메인 정보에 대한 연관을 생성하는 것을 제안한다면 개념에 대한 각각의 특성이 연관된다. 만약 그렇지 않으면, 카디널리티가 “1” 또는 “opt”이면 특성은 속성이 된다.
- ④ 특성을 가지고 있지 않은 개념은 속성 또는 연관된다. 정리 안된 개념은 새로운 클래스, 내장 클래스, 또는 부분형이 될 수도 있다. 예를 들면, “Name”은 “String” 또는 문자열의 부분형으로 될 수 있다.
- ⑤ 다양한 구성의 개념에 대하여 일반화는 유용하다.

그림 7. 논리적 모델링의 객체 모델링 알고리즘

로 중요한 개념들을 표현한 개념적 그래프이다. 도메인 객체 다이어그램에서는 하나의 입력과 출력을 가지고 있는 개념들을 없앤 것이다.

집합 트리는 도메인 객체 다이어그램으로부터 생성된다. 하나의 도메인 객체가 존재를 위하여 다른 하나에 의존할 때, 그 관계성은 집합 다이어그램에서 변환된다. 집합은 도메인 객체와 다른 객체사이의 관계성이다. 객체의 설명은 다른 객체에 바탕을 둔다. 집합 트리는 개념적 그래프에서의 속성 역할을 하는 도메인 객체를 기술한다. 그러나 도메인 객체는 자신의 속성을 가지고 있기 때문에 각각 객체로서 모델링 되어야 한다. 예를 들면, 부서에 관리자와 사원이 있으면 집합으로 변환시킨다. 만약 도메인 문장에 관리자와 사원이 없이 부서만 가지고 있는 것은 가능하다. 연관에서 집합을 구별하는 방법은 도메인 개념의 정의를 기록하는 것이다. 개념을 정의하는 것이 필요한 조건은 집합 관계성에 있다. 그리고 다른 관계성은 연관이다.

객체 모델을 만드는 것은 개념적 그래프, 카디널리티 정보와 그림 8과 같은 도메인 객체 다이어그램, 그리고 그림 9와 같은 집합 트리를 필요로 한다. 객체 모델은 UML(Unified Modeling Language) 같은 객체를 위한 모델링 언어로 표현한다. 회사 UML 클래스 다이어그램은 그림 10에 주어진다.

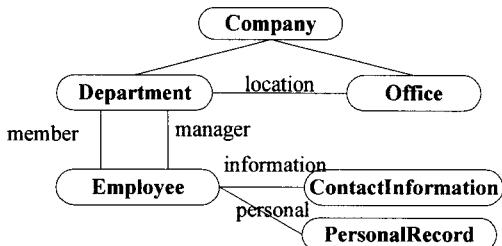


그림 8. 회사 도메인 객체 다이어그램

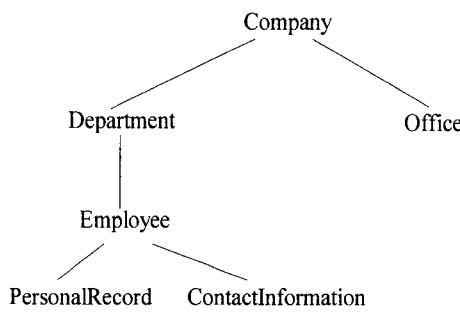


그림 9. 회사 집합 트리

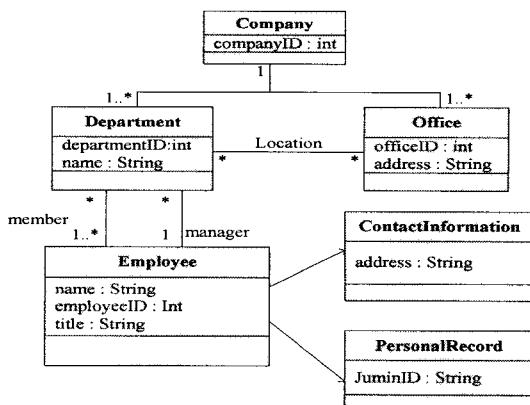


그림 10. 회사 UML 클래스 다이어그램

UML은 객체 관련 표준화기구인 OMG에서 객체 모델링 기술 OMT(Object Modeling Technique), OOSE(Object-Oriented Software Engineering) 방법론 등을 연합하여 만든 통합 모델링 언어로 객체 지향적 분석·설계 방법론의 표준으로 지정되고 있다. 본 논문에서는 UML의 클래스 다이어그램을 사용한다. UML 클래스 다이어그램은 클래스명, 속성, 그리고 동작으로 구성된다. UML 클래스 다이어그램도 클래스의 특징, 연관에 대한 카디널리티 상태와 일반화 사이의 관계성을 나타낸다. 연관은 단순한 화

살표로 표시된다. 그리고 일반화는 열린 삼각 화살표로 표시된다.

현재 객체 구조 내에 개념적 그래프를 사용하는 것은 가능하다. 개념적 모델링 처리의 부분으로 객체의 이름이 붙은 직사각형 박스를 사용한다.

## 5. 물리적 모델링

### 5.1 XML 모델링

UML 클래스를 사용하여 XML Schema로 변환하기 위한 규칙을 제안한다. 그리고 UML 클래스를 이용한 XML 모델링 과정의 다이어그램은 그림 11과 같다.

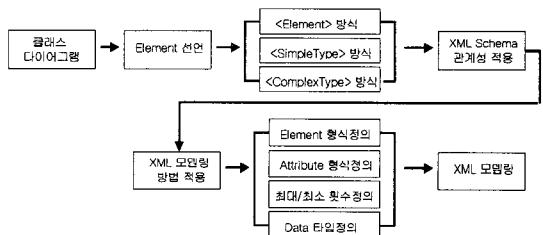


그림 11. XML 모델링 과정

XML Schema의 관계성 표현은 정보 모델에서 객체들에 관련하여 4가지 유형으로는 복합, 집합, 상속, 연관의 관계성을 말한다.

#### (1) 복합 관계

복합 관계는 가장 이해하기 쉬운 관계성이다. 이것은 하나의 용기(Container)안에 한 객체의 단순 중첩을 말한다. XML에서 이것은 또 하나의 element 자식과 마찬가지로 element 또는 attribute를 정의하는 것으로 변환한다.

#### (2) 집합 관계

복합 관계와 마찬가지로 동일한 기법을 사용하며, element를 포함하는 복합 관계의 내부에 독립적으로 존재하는 객체를 정의한다.

#### (3) 상속 관계

상속 관계는 오직 형식(Types)에 적용한다. XML Schema에서 상속 구조는 매우 간단하다고 할 수 있다. XML Schema에 2개의 기본적인 형식이 있는데 이것은 simple type과 complex type이다. 그렇지만 hybrid complex type은 simple type으로부터 상속 한다.

#### (4) 연관 관계

연관관계는 2개의 객체 또는 속성들이 서로 상호 간에 관련된 곳에서, 이들 2개의 항목들을 함께 묶도록 XML instance document안에 링크를 생성하는 것이 가능하다. 이것은 집합 관계에서 보여준 동일한 수법을 사용하며 관련된 항목들에 대해서는 key/keyref를 pair로 생성해서 처리한다.

##### 5.1.1 UML 기반의 XML 모델링을 위한 변환규칙

① empty elements 또는 empty attributes를 인정하지 않는다. 그 대신 그들이 어떤 값을 갖는지 또는 값이 존재하지 않는지 둘 중의 하나를 분명히 한다. 만일 element가 필수적이면 minOccurs="1", attribute가 필수적이면 use="required"로 설정한다. 또한 그 값이 미심쩍을 때는 empty 문자열을 인정한다. 그 때 element인 경우 minOccurs="0", attribute인 경우 use="optional"로 설정한다.

② whitespace는 instance document의 크기를 크게 할 수 있다. 그 경우 파일 크기가 제한적임을 분명히 하기 위해 XML instance document를 수신하는 어플리케이션이 필요하다. 또한 일반적으로 프로세싱 어플리케이션은 데이터구조의 크기에 관하여 약간의 제한을 가진다. 그렇기 때문에 개별적인 필드들은 길이를 제한한다. 특히 keys일 경우 대부분의 데이터베이스들은 필드들의 일정한 형식에 입각하여 크기를 제한한다.

③ 문자 집합과 필드크기를 제한할 경우 base types(string, decimal, integer)를 이용하여 속성 값의 데이터 형식을 지정한다.

④ element 또는 attribute값의 유일함을 나타내기 위해 unique element를 사용한다.

⑤ document에서 2개의 location 연관시키기 위해 key/keyref element를 사용한다.

⑥ 만일 Schema가 클 경우 그룹 참조 방법을 사용하여 별개의 파일로 section을 이동시킨다. 이때 추상적 원리를 균일하게 적용하고 별개의 파일로 이동한 section의 Schema 파일은 전역적 element를 갖지 않는다. simple type정의, complex type정의, 그룹정의를 모두 포함하고 있는 파일을 가질 수 있고, 그들을 관련된 block으로 분할 가능하다.

⑦ namespace를 어디서 변경해도 instance document에서는 어떤 element 라도 사용 가능하다. 그

리고 그 location에서 허락한 자식 elements의 number와 namespace를 일일이 지정함. 새로운 namespace에서 element의 이름을 분명히 하기 위해 element ref=". ." element를 상술한다.

⑧ 만일 element 또는 attribute가 특정한 조건으로 존재하지 않으면, 어플리케이션은 기본값(default value)이 존재함을 표시한다. 그것은 element 또는 attribute선언에서 default attribute사용으로 지정 가능하다.

⑨ instance document에서 유도형(derived type)이 치환되는 것을 방지하기 위해 Schema element에 blockDefault="#all" attribute를 포함시킨다.

⑩ 현재 Schema에서 선언된 것으로부터 어떤 새로운 형식이 유도되는 것을 방지하기 위해 Schema element에 finalDefault="#all" attribute를 포함시킨다. 이것은 blockDefault보다 엄격한 제한 조건이다.

요소 매핑은 XML Schema의 더욱 장황한 특징을 설명한다 그리고 또한 그것은 DTD와 비슷하지 않으며 XML Schema는 합법적인 XML로 표현된다. 표 1과 같이 요소 매핑을 표현하는데 많은 방법들이 있다. 요소 정의 안에 복합 타입을 정의하는 최상의 공통을 선택하였다. 이 두 가지를 개별적으로 분리하는 것은 불가능하다. 그리고 외부적 복합 타입 정의를 참조하는 요소 정의를 갖는다.

중첩된 “complexType”은 요소들에 대해서 내용 모델을 정의한다. 그리고 그 내용 모델은 요소 참조 같이 표현된다.

또 다른 요소 정의에서 요소들을 정의하는 것은 XML Schema에서 가능하므로 다음과 같다:

카디널리티 표현에 있어서 매우 다른 구성을 주목하는데 “minOccurs”와 “maxOccurs”속성에서 쌍방 어느 쪽이든 디폴트값은 “1”이다.

##### 5.1.2 XML 모델링의 예

그림 10에서 “Company” 객체와 “Department”, “Office” 객체는 관계성에서 집합 관계이며 이 의미는 “Company” 객체가 “Department”, “Office”를 갖고 있어야 하며 따라서 “Department”, “Office”쪽이 다중성 값이 1..\*이라는 것을 보여주고 있다. 반면 “Company” 객체의 다중성 값이 1..1인 것은 “Company”의 존재에 따라서 “Department”, “Office”가 존재할 수 있음을 의미한다. 이를 XML 모델링을 하기 위해서는 그림 11과 UML 기반의 XML 모델링을 위

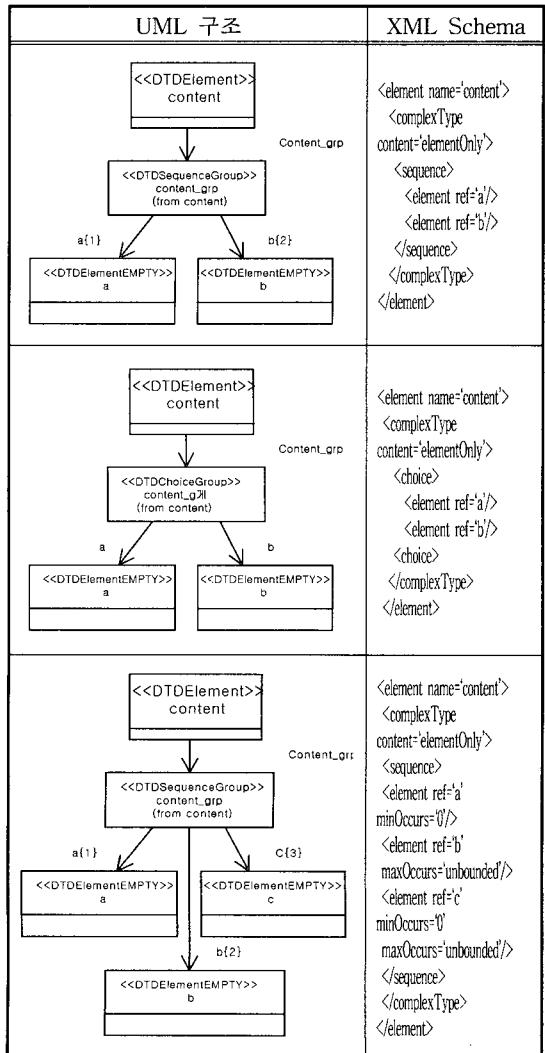
표 1. UML과 XML Schema(1)

| UML 구조                                                                                              | XML Schema                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;&lt;DTDElementPCDATA&gt;&gt;</code><br>pcdata                                             | <pre>&lt;element name='s3'&gt; &lt;complexType content='textOnly'&gt; &lt;/complexType&gt; &lt;/element&gt;</pre>                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>&lt;&lt;DTDElementANY&gt;&gt;</code><br>any                                                   | <pre>&lt;element name='any'&gt; &lt;complexType&gt; &lt;any minOccurs='0' maxOccurs='unbounded' processContents='skip' /&gt; &lt;anyAttribute processContents='skip' /&gt; &lt;/complexType&gt; &lt;/element&gt;</pre>                                                                                                                                                                                                                                                                                                          |
| <code>&lt;&lt;DTDElementEMPTY&gt;&gt;</code><br>empty                                               | <pre>&lt;element name='empty'&gt; &lt;complexType content='empty'&gt; &lt;/complexType&gt; &lt;/element&gt;</pre>                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>&lt;&lt;DTDElementPCDATA&gt;&gt;</code><br>sub2                                               | <pre>&lt;element name='empty'&gt; &lt;complexType content='textOnly'&gt; &lt;attribute name='notrequired' type='string' /&gt; &lt;attribute name='avalue' type='string' use='fixed' val='fixedval' /&gt; &lt;attribute name='anenum' simpleType base='NMTOKEN'&gt; &lt;enumeration value='val1' /&gt; &lt;enumeration value='val2' /&gt; &lt;enumeration value='val3' /&gt; &lt;/simpleType&gt; &lt;/attribute&gt; &lt;attribute name='required' type='string' use='required' /&gt; &lt;/complexType&gt; &lt;/element&gt;</pre> |
| Notrequired : CDATA<br>Avalue : CDATA=fixedval<br>Anenum : (val1   val2   val3)<br>Required : CDATA |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|                                                                                                     | <pre>&lt;element name = 'content' &lt;elementType content = 'elementOnly'&gt; &lt;sequence&gt; &lt;element name='a'&gt; &lt;complexType content = 'elementOnly'&gt; . . . &lt;/complexType&gt; &lt;/element&gt; &lt;/sequence&gt; &lt;/complexType&gt; &lt;/element&gt;</pre>                                                                                                                                                                                                                                                   |

한 변환규칙 ①과 ④를 적용하고 표 1과 표 2를 적용하여 XML Schema로 모델링 하면 그림 12와 같다.

그림 10에서 “Employee” 객체와 “ContactInformation”, “PersonalRecord” 객체는 관계성에서 집합 관계이며 이 의미는 “Employee” 객체가

표 2. UML과 XML Schema(2)



“ContactInformation”, “PersonalRecord” 객체를 갖고 있어야 하며 “Employee” 객체의 존재 여부에 따라 “ContactInformation”, “PersonalRecord” 객체가 존재할 수 있다. 이를 XML 모델링을 하기 위해서는 그림 11과 UML 기반의 XML 모델링을 위한 변환규칙 ①과 ④를 적용하고 표 1과 표 2를 적용하여 XML Schema로 모델링 하면 그림 13과 같다.

그림 13에서 “Employee” 객체는 3개의 자식 객체와 2개의 참조 객체를 가지고 있다. 또한 “Employee” 객체에서 참조하는 “ContactInformation”, “PersonalRecord” 객체는 자식 객체를 갖고 있지 않으면서 “Employee” 객체에서 반드시 기술해야 하므

```

<element name="Company" type="Company">
  <complexType name="Company">
    <sequence>
      <element name="CompanyID" type="Int"/>
      <element name="department" minOccurs="1"/>
      <element name="Office" minOccurs="1"/>
    </complexType>
    <element REF="Department"/>
    <element REF="Office"/>
  </sequence>
</complexType>
</element>
<element name="Department" type="Department">
  <complexType name="Department">
    <element name="DepartmentID" type="Int" use="required"/>
    <element name="name" type="String" use="required"/>
  </complexType>
</element>
<element name="Office" type="Office">
  <complexType name="Office">
    <element name="OfficeID" type="Int" use="required"/>
    <element name="Address" type="String" use="required"/>
  </complexType>
</element>

```

그림 12. Company 객체의 XML Schema 모델링

```

<element name="Employee" type="Employee">
  <complexType name="Employee">
    <sequence>
      <element name="name" type="String" minOccurs="1"/>
      <element name="employeeID" type="int" minOccurs="1"/>
      <element name="title" type="String" minOccurs="1"/>
    </complexType>
    <element REF="ContactInformation"/>
    <element REF="PersonalRecord"/>
  </sequence>
</complexType>
</element>
<element name="ContactInformation">
  <complexType>
    <element name="address" type="string" use="required"/>
  </complexType>
</element>
<element name="PersonalRecord">
  <complexType>
    <element name="JuminID" type="string" use="required"/>
  </complexType>
</element>

```

그림 13. Employee 객체의 XML Schema 모델링

로 use="required"로 설정하기 위해서 attribute로 정의하였다.

## 5.2 데이터 모델링

이 절에서는 UML 클래스 다이어그램을 사용하여

객체-관계형 데이터베이스 스키마로 변환하기 위한 규칙을 제안한다.

### 5.2.1 UML 클래스에서 객체-관계형 데이터베이스 스키마로의 변환규칙

그림 10의 UML 클래스를 이용한 데이터 모델링 과정의 다이어그램은 그림 14와 같다.

그림 14에서 객체-관계형 데이터베이스 스키마 도출을 위한 데이터 모델링의 변환 방법은 다음의 방법을 사용한다.

- ① 객체 클래스는 구조화된 객체 타입을 생성함. 테이블과 연결하여 타입을 1개 또는 여러 개의 테이블에 객체의 원소가 되도록 함.
- ② 객체 타입은 사용자 정의 타입이거나, 속성을 가진 객체 타입임.
- ③ 클래스에서의 객체 속성은 객체 타입에서의 속성임.
- ④ 타입이 테이블이나 객체 또는 특정 타입으로 변환될 때 클래스의 객체 속성 타입은 객체 타입의 속성 타입임.

⑤ 객체 속성이 NULL 제약 조건을 가지면 NOT NULL 제약 조건도 가짐.

⑥ 객체 속성이 초기 값을 가지면 컬럼에 DEFAULT 값을 추가함.

⑦ 독립적인 클래스나 핵심적인 성질을 가진 클래스들을 위해 기본키로 정수를 생성하고 {oid}를 위해 태그된 컬럼에 기본키 제약 조건에 따라 {oid}를 추가함.

⑧ 부 클래스를 위해 기본키 제약 조건과 외래키 제약 조건에 따라 각각의 부모 클래스의 키를 추가함.

⑨ 클래스들의 결합을 위해 객체 타입을 생성하고 기본키 제약 조건과 외래키 제약 조건에 따른 역할 테이블로부터 기본키를 추가함.

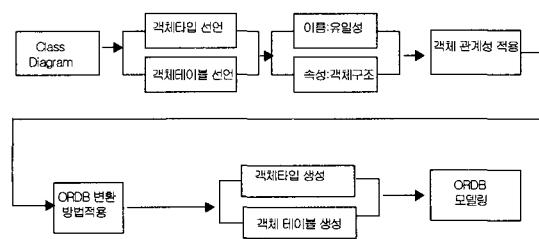


그림 14. 데이터 모델링 과정

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

⑪ 각각의 확실한 제약조건을 위해 CHECK를 실시함.

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

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

⑭ 너무 많이 이동하게 되는 이전 결합 클래스들을 최적화 함.

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

⑯ N : N 결합에서 역할 테이블의 키로부터 기본키, 외래키의 제약 조건을 생성함.

⑰ 객체 클래스의 전달 작용을 위한 객체 타입에 메소드를 생성함

### 5.2.2 데이터 모델링의 예

그림 10의 “Company” 객체는 그림 14와 ORDB 변환 방법 ③, ④번의 성질에 따라 “Company\_t” 객체

```
SQL> CREATE TYPE Company_t AS OBJECT (
  CompanyID Int,
  Department Department_t,
  Office Office_t );
SQL> CREATE TABLE Company OF Company_t;
```

그림 15. Company 테이블

```
SQL> CREATE TYPE Employee_t AS OBJECT (
  EmployeeID Int,
  Name string,
  title string,
  Department Department_t,
  ContactInformation ContactInformation_t,
  PersonalRecord PersonalRecord_t );
SQL> CREATE TABLE Employee OF Employee_t
```

그림 16. Employee 테이블

체 타입 속성을 저장하는 “Company” 테이블로 그림 15와 같이 변환된다.

그림 10의 “Employee” 객체는 그림 14와 ORDB 변환 방법 ③, ④의 성질에 따라 “Employee\_t” 객체 타입 속성을 저장하는 “Employee” 테이블로 그림 16과 같이 변환되고 “ContactInformation\_t”, “PersonalRecord\_t” 객체 타입을 포함한다.

## 6. 결 론

현재 UML은 객체지향 시스템 개발을 위한 표준적인 방법론으로 OMG의 승인을 받고 있으며 UMF (UML eXchange Format)의 등장으로 UML 클래스 다이어그램을 확장하여 많은 분야에 적용할 수 있게 되었다.

한편, XML은 이 기종 시스템간의 문서 교환을 위해 탄생하였으며, SGML의 단점을 극복하고 확장성과 문서의 구조적인 표현이 불가능한 HTML의 단점을 보안한 새로운 인터넷의 표준으로 자리잡아 가고 있다. 더욱이 XML은 어떠한 형태의 문서라도 다양하게 모델링이 가능하며, 차세대 하이퍼텍스트 기능 및 문서의 내용과 스타일 정보를 분리하여 사용하는 특성을 가지고 있다. 또한 문서의 재사용성이 뛰어나기 때문에 다방면에서 적용될 수 있는 언어이다.

이렇듯 XML 문서에 대한 중요성이 가중되는 시점에서, 본 논문에서는 UML 클래스 다이어그램을 XML Schema로 변환하고 객체-관계형 데이터베이스로 변환하는 것을 설계하였다. 즉, XML DTD에서 표현할 수 없는 데이터를 XML Schema를 통하여 표현하고, 관계형 데이터베이스에서 관리하기 힘든 데이터들을 객체-관계형 데이터베이스를 이용하여 관리하게 되면 현재 웹 상에서 사용되고 있는 데이터들을 쉽게 저장하고 관리할 수 있을 것이다.

향후 연구 과제로는 UMF를 통하여 확장된 수많은 UML 클래스 다이어그램을 XML Schema의 형태로 변환하거나, 변환된 Schema를 객체지향 데이터베이스에 저장하기 위한 일반적인 사상 알고리즘을 개발하는 것이다. 그리고 현재 UML 클래스 다이어그램을 이용하여 XML Schema를 자동 생성하는 프로그램을 개발중이다.

## 참 고 문 헌

[ 1 ] What is XML?, <http://www.xml.com/pub/a/98>

- /10/guide1.html#AEN58.
- [ 2 ] Migrating from XML DTD to XML-Schema using UML, <http://www.rational.com/products/whitepapers/412.jsp>.
- [ 3 ] Cheng, J., Xu, J., XML and DB2, In Sixteenth International Conference on Data Engineering(ICDE'00), 28 February–3 March 2000, San Diego, IEEE Computer Society Press, Los Alamitos, CA, 2000, 569–576.
- [ 4 ] W3C Recommendation, <http://www.w3.org/TR/2001/REC-xmlsc-hema-0-20010502/>, 05/02/2001.
- [ 5 ] UML for XML Schema Mapping Specification, [http://www.Rational.com/media/uml\\_xml-schema33.doc](http://www.Rational.com/media/uml/resources/media/uml_xml-schema33.doc), 12/08/99.
- [ 6 ] Modeling XML vocabularies with UML, <http://www.xml.com/p-pub/a/2001/09/19/uml.html>, 09/19/2001.
- [ 7 ] Bray, T., J. Paoli, and C.M. Sperberg-McQueen, eds. Extensible Markup Language (XML) 1.0. REC-xml19980210. W3C Recommendation 1998. <http://www.w3.org/TR/REC-xml>.
- [ 8 ] Iyengar, S. and S.A. Brodsky, eds. XML Metadata Interchange (XMI). Proposal to the OMG Object Analysis & Design Task Force

RFP 3: Stream-based Model Inter-change Format (SMIF) 1998, Object Management Group. <http://www.omg.org>.

- [ 9 ] Mapping W3C Schemas to Object Schemas to Relational Schmeas, <http://www.rpbourret.com/xml/SchemaMap.htm>.



### 최 문 영

1998년 청운대학교 전자계산학과 졸업(학사)  
2000년 청운대학교 정보산업대학원 전산전자정보학과 졸업(석사)  
2000년~2004년 8월 순천향대학교 전산과 졸업(박사)

관심분야 : Database Systems, Object-oriented, XML



### 주 경 수

1980년 고려대학교 이과대학 수학과 졸업(학사)  
1982년 고려대학교 일반대학원 전산학과 졸업(석사)  
1986년 고려대학교 일반대학원 전산학과 졸업(박사)  
1998년 University of North Carolina  
1999년 Visiting Professor  
1986년~현재 순천향대학교 정보기술공학부 교수

관심분야 : Database Systems, System Integration, Object-oriented Systems