데이터베이스 기반의 XML 응용을 위한, UML을 이용한 통합 설계 방법론*

A Unified Design Methodology using UML for XML Application based on Database

방승윤(Sung-Yoon Bang)**, 주경수(Kyung-Soo Joo)***

초 록

B2B 전자상거래와 같이 XML을 이용한 정보교환이 확산되고 있으며 이에 따라 상호 교환되는 정보에 대하여 체계적이며 안정적인 저장관리가 요구되고 있다. 이를 위해 XML 응용과 데이터베이스간의 연계를 위한 다양한 연구가 이루어지고 있다. 특히 계층적 구조를 갖는 XML 파일을 다양한 형태의 데이터베이스에 저장하기 위한 데이터 모델링 방안이 요구된다.

본 논문에서는 UML을 이용한, 다양한 형태의 데이터베이스 기반의 XML 응용을 위한 통합 설계 방법론을 제안한다. 이를 위하여 먼저 UML을 이용하여 W3C XML schema를 설계하기 위한 XML 모델링 방안을 제시하고, 아울러 교환되는 XML 파일을 저장·관리하기 위하여 객체-관계 데이터베이스 스키마와 객체지향 데이터베이스 스키마 그리고 관계형 데이터베이스 스키마 설계를 위한 데이터 모델링 방법을 제안한다.

ABSTRACT

Nowadays an information exchange on XML such as B2B electronic commerce is spreading. Therefore the systematic and stable management mechanism for storing the exchanged information is needed. For this goal there are many research activities for connection between XML application and database. Accordingly, A unified modeling methodology need to store the XML file on the variety database.

In this paper, we propose a unified design methodology for XML applications based on variety databases using UML. To this goal, first we introduce a XML modeling methodology to design W3C XML schema using UML and second we propose data modeling methodology for object-relational database schema, object oriented database an schema and relational database schema to store efficiently XML data in databases.

키워드 : XML 모델링, 데이터 모델링, UML, XML Modeling, Data Modeling.

---

* 본 연구는 정보통신부의 ITRC 사업에 의해 수행된 것임

** 순천향대학교 전산학과 박사과정(sybang@hanseo.ac.kr)

*** 순천향대학교 정보기술공학부 교수(gsoojoo@sch.ac.kr)

■ 논문 접수일 : 2002년 4월 8일

■ 게재 확정일 : 2002년 6월 17일

1 서 론

XML은 구조화된 정보를 포함하고 있는 문서들을 위한 마크업 언어이다. 구조화된 정보는 구체적인 내용과 그 내용이 수행해야할 역할을 포함하고 있다(XML). XML Schema에 대한 연구는 지금도 계속 W3C에서 연구 진행중인 XML Schema가 표준 XML Schema의 스펙이 될 가능성이 가장 높은 실정이다. W3C XML Schema는 XML DTD보다 다양한 데이터타입을 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 (Migrating, 방승윤,주경수 2001).

한편 XML 어플리케이션과 데이터베이스 시스템 사이의 원활한 연계를 위해서, 그 동안 XML DTD를 관계형 데이터베이스 스키마로 변환하는 방법에 대해서는 많은 연구가 진행되었다. 그러나 아직까지 표준화가 진행중인 W3C XML Schema를 UML를 이용한 클래스 다이어그램에 의해서 모델링하고 그 모델링에 의해 교환되는 데이터를 저장하기 위한 통합 모델링에 대한 연구는 미비한 실정이다. 본 논문에서는 2장에서 관련 연구, 3장에서 UML를 이용한 통합 모델링, 4장에서는 W3C XML Schema의 관계성, 5장에서는 XML 모델링, 6장에서는 데이터 모델링 그리고 마지막 7장에서는 결론을 기술한다.

2 관련 연구

XML이 단순한 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.

XML Schema에 대한 연구는 W3C와 XML에 관심 있는 관련자들에 의해서 연구되었고, W3C Recommendation으로 채택되었다(W3C 2001). UML를 이용한 XML 모델링에 대한 연구는 UML를 XML Schema로 변환할 때 UML 확장 메커니즘(stereotype and tagged values)을 사용하여 XML 구조를 표현하였다 (UML). 또한 UML 클래스 다이어그램을 XML DTD로 자동 변환하고 XML DTD를 XML Schema로 자동 변환하는 도구(tool)의 개발은 모델자(modeller)가 XML Schema 구문에 익숙하지 못해도 XML 모델링을 신속하게 하고 개별적으로 서로 다른 언어 또는 환경에서 모델 어휘들을 재 사용할 수 있게 하였다 (XML Modeling, Modeling XML 2001).

그 동안 많은 논문들이 복잡한 XML

파일을 관계형 데이터베이스에 저장하는 데 초점을 맞추어 데이터베이스와 XML 문서의 연결에 대하여 발표되었다(방승윤 2001; 이정수 2001). 발표된 논문들의 내용을 분류하면 주어진 XML 파일을 관계형 데이터베이스에 어떻게 저장할 것인가 하는 제안들을 하고 있으며, 주어진 XML 파일이나 DTD로부터 정보의 의미에 대하여 데이터베이스 제약사항들을 다루는 방법을 제안한다(Florescu, D. 1999; Kanne 2000; Klettke 2000).

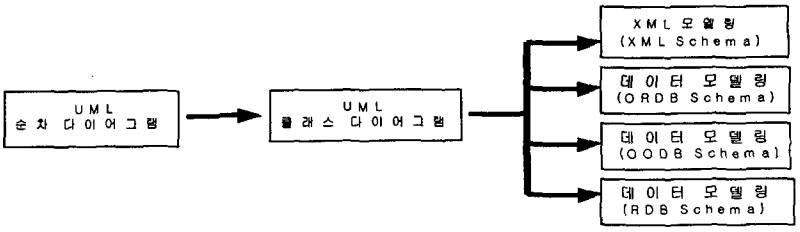
XML 문서를 객체-관계 시스템으로 구조화된 데이터 타입 형태로 저장하는 방법이 제안되고 있다(Cheng 2000, Kettke 2000). 다른 한편으로는 객체-관계 데이터베이스로부터 XML로 변환하는 내용을 다루고 있으며, 그 내용은 어떻게 XML 파일을 객체-관계 데이터베이스에 저장하여 빠르게 질의할 것인가를 다루고 있다(Carey 2000, Shanmugasundaram 2000). 또한 XML 파일을 객체지향 데이터베이스로 저장하기 위한 모델링 방안에 대한 연구는 객체모델을 토대로 하여 XML DTD를 객체지향 데이터베이스 스키마로 변환하기 위한 방법론이 제안되었다(최문영 2001).

위와 같이 부분적으로 제안된 연구는 많으며 UML 클래스를 이용하여 XML 모델링과 ORDB 모델링 또는 OODB모델링을 개별적으로 연구한 논문(방승윤 2002)도 있지만 본 논문에서 제안한 UML 클래스를 이용하여 XML 모델링과 데이터 모델링을 통합적으로 설계하는 방법론을 제안한 연구는 미비한 상태이다. 특히 UML 클래스를 이용한 XML 모델링에서 가이드라인(Guideline)을 준수하고 XML 특성을 참작하여 XML Schema를 설계한 논문은 전무한 실정이며 또한 데이터 모델링에서도 각각의 데이터베이스 스키마(ORDB, OODB, RDB)설계에 따른 가이드라인을 준수하고 데이터베이스들의 특성을 참작하여 데이터베이스 스키마를 설계한 논문 역시 전무한 실정이다.

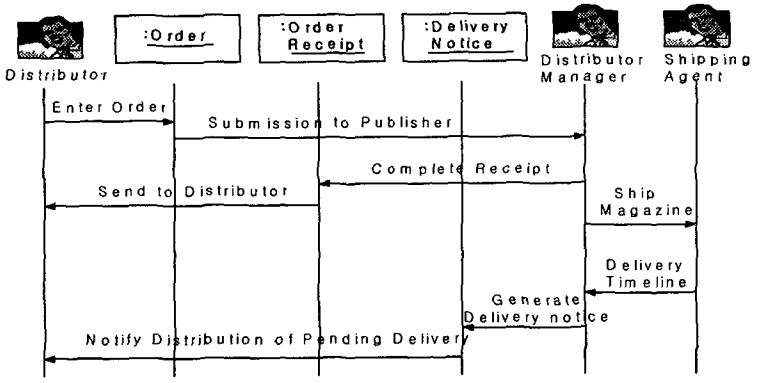
본 논문의 목적은 UML 클래스를 이용하여 XML 모델링과 데이터 모델링(ORDB, OODB, RDB)을 동시에 설계할 수 있는 통합적인 설계 방법론에 대한 제안과 설계 과정에서 참조 할 수 있는 가이드라인을 제시한 것에 큰 의미가 있다고 할 수 있다. 또한 XML를 통하여 교환되는 XML 응용 데이터를 효율적으로 저장·관리하기 위해서 선택한 데이터베이스에 대한 데이터 모델링과 XML 모델링을 동시에 설계하는 통합 방법론은 XML 어플리케이션을 체계적이고 효율적으로 설계할 수 있으며 프로그램 개발을 통합적으로 추진할 수 있는 장점을 가지고 있다.

3 UML를 이용한 모델링

UML은 소프트웨어 시스템의 구조물을 명세화, 구조화, 시각화, 문서화뿐만 아니라 비-소프트웨어 시스템 그리고 비즈



〈그림 1〉 UML을 이용한 XML 모델링과 데이터 모델링



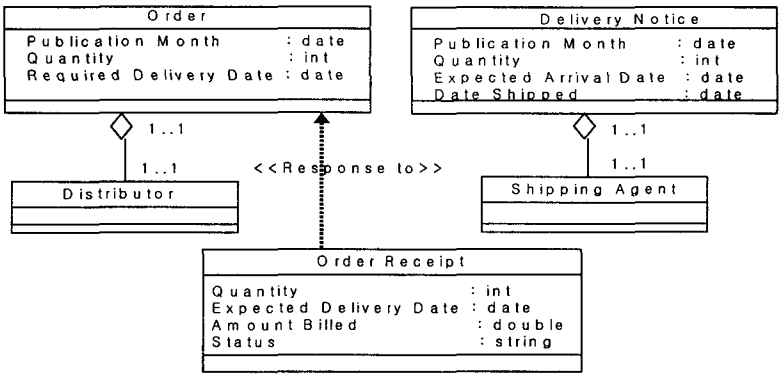
〈그림 2〉 메시지 객체와 순차 다이어그램

니스 모델링을 위한 하나의 언어이다 (임춘봉 1999). 또한 XML 프로그램이나 XML를 이용한 B2B 시스템 구축과 같은 XML 프로젝트에서는 객체 지향적 설계 언어인 UML을 이용하여 개발의 효율을 높일 수 있다.

순차 다이어그램은 응용 프로그램에서 사건의 흐름을 분명하게 하는 역할을 하며 글로 풀어서 쓴 유스 케이스(Use case)를 메시지로 표현한 그림으로 변환할 수 있다. 또한 그 그림은 항상 더 간결하게 이해되고, 특히 우리가 클래스 다이어그램을 만들 때 작업을 쉽게 해준다. 그러므로 UML을 이용해 유스케이스에 의한 순차 다이어그램을 도출하여 클래스 다이어그램을 만든 후 그 클래스 다이어그램에 의해서 〈그림 1〉과 같이 XML 모델링과 데이터 모델링으로 변환할 수 있다.

〈그림 2〉의 순차 다이어그램은 객체들 간의 메시지 과정을 설명한다. 이 다이어그램은 메시지 객체들과 진행과정에 관계된 actors를 포함하고 있다.

〈그림 2〉에서 객체들과 actors 이외에 수직의 축은 시간을 표현한다. 화살표는 행위 또는 객체와 actors간에 발생하는 사건을 표현한다. 이 순차 다이어그램은 2개의 새로운 actors('Distribution Manager', 'Shipping Agent')를 소개한다. 이것은



〈그림 3〉 클래스 다이어그램

〈표 1〉 객체 정의

클래스	기 술
Order	구체적인 달에 배달해야할 잡지의 수량을 Distributor가 Distribution Manager에게 송신한다. 또한 배달 요구 날짜를 포함.
Order Receipt	Distribution Manager는 Distributor에 의거한 Order에 대한 대답을 송신한다. 주문된 수량, 그 주문에 대한 계산된 총액, 예상 선적일, 주문상태를 기술한다.
Delivery Notice	Distribution Manager는 출판물이 Shipping Agent에 배달된 후에 Distributor에게 송신한다. 선적된 날짜, 예상 배달날짜, 수량, 출판 달을 기술한다

시스템에서 actors를 설명하는 객체를 의미한다. 〈그림 3〉의 클래스 다이어그램은 3개의 메시지 그리고 그들의 연관된 속성과 서브-객체를 예를 들어 설명한다 (Duckett Jon 2001).

〈그림 3〉은 'Order' 그리고 'Order Receipt'간에 관계성에서 연관 관계를 갖는다 왜냐하면 'Order Receipt'는 'Order' 객체에 의존하고 있기 때문이다. 이것은 반 직관적일 수 있다. 그 이유는 진행흐름의 방향이 'Order'가 발생하면 'Order Receipt'가 존재하기 때문이다. 그렇지만 〈그림 3〉은 진행과정을 설명하지 않고 오직 진행과정에 참여하는 정보항목 그리고 객체들간에 종속성과 관계성을 설명한다. 진행의 설명을 자세히 하기 위해서는 객체 정의가 필요하고 주문처리에 대한 객체들의 정의는 〈표 1〉과 같다(Duckett Jon 2001).

각각의 객체('Order', 'Order Receipt', 'Delivery Notice')에 대한 속성, 사상수, 관계성은 〈표 2〉와 같다.

〈표 2〉 객체들의 속성(서브-객체), 사상수, 관계성

클래스	속성/서브-객체	사상수	관계	기 술
Order	Quantity	1 . . 1	복합	요구한 잡지의 수량을 기술해야함
	Distributor	1 . . 1	집합	Distributor를 기술해야함
	Required Delivery Date	0 . . 1	복합	배달날짜를 기술할 수 있다
	Publication Month	1 . . 1	복합	출판 달을 기술해야함
Order Receipt	Quantity	1 . . 1	복합	주문수량을 승인해야함
	Status	1 . . 1	복합	주문 상태를 기술해야함
	Expected Delivery Date	0 . . 1	복합	예상 배달날짜를 기술할 수 있다
	Amount Billed	1 . . 1	복합	주문에 대한 계산금액을 기술
Delivery Notice	Quantity	1 . . 1	복합	배달된 수량을 기술해야함
	Date Shipped	1 . . 1	복합	선적날짜를 기술해야함
	Shipping Agent	1 . . 1	집합	선적한 중개인을 기술해야함
	Expected Arrival Date	1 . . 1	복합	예상된 도착날짜를 기술해야함
	Publication Month	1 . . 1	복합	출판 달을 기술해야함

4 W3C XML Schema

W3C XML Schema는 아직도 표준화가 진행 중인 상태이지만, 큰 틀은 바뀌지 않을 것이다. Schema의 표기법은 제안한 업체마다 조금씩 다르긴 하지만 데이터 형식 및 반복 횟수 지정 등, 그 자체가 XML 구조로 되어 있다는 것은 공통점이다(김채미 2001).

관계성 표현은 정보모델에서 객체들에 관련하여 4가지 유형(복합, 집합, 상속, 연관)의 관계성을 말한다. 이 관계성들은 각각 XML Schema에서 표현될 수 있다(Duckett Jon 2001).

① 복합관계

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

② 집합관계

복합관계와 마찬가지로 동일한 기법을 사용하며, element를 포함하는 복합관계의 내부에 독립적으로 존재하는 객체를 정의한다.

③ 상속관계

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

④ 연관관계

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

5 XML 모델링

XML 어플리케이션에서 기본이 되는 XML Schema 설계 과정에서 최적의 Schema가 되도록 하기 위해서 물리적인 모형이나 도해를 만드는 것을 말한다.

5.1 XML Schema를 위한 XML 모델링 방법

3절에서 제시한 UML 클래스로부터 W3C XML Schema를 도출하기 위하여 아래의 방법들을 사용한다(Duckett Jon 2001).

① empty elements 또는 empty attributes를 인정 안함. 그 대신 그들이 어떤 값을 갖는지 또는 값이 존재하지 않는지 둘 중의 하나를 분명히 함. 만일 element가 필수적이면 minOccurs="1", attribute가 필수적이면 use="required"로 설정함. 또한 그 값이 미심쩍을 때는 empty 문자열을 인정함. 그때 element인 경우 minOccurs="0", attribute인 경우 use="optional"로 설정함.

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

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

④ element 또는 attribute값의 유일함을 나타내기 위해 unique element를 사용함.

⑤ document에서 2개의 location을 연관시키기 위해 key/keyref element를 사용함.

⑥ 만일 Schema가 클 경우 그룹참조 방법을 사용하여 별개의 파일로 sections를 이동시킴. 이때 추상적 원리를 균일하게 적용하고 별개의 파일로 이동한 sections의 Schema 파일은 전역적 element를 갖지

않음. simple type정의, complex type정의, 그룹정의를 모두 포함하고 있는 파일을 가질 수 있고, 그들을 관련된 block으로 분할 가능함.

⑦ namespace를 어디서 변경해도 instance document에서는 어떤 element라도 사용 가능함. 그리고 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 표시 가능함. 그것은 element 또는 attribute 선언에서 default attribute사용으로 지정 가능함.

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

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

5.2 XML 모델링의 예

〈그림 3〉에서 'Order' 객체와 'Distributor' 객체는 관계성에서 집합관계이며 이 의미는 'Order' 객체가 'Distributor'를 갖고있어야 하며 따라서 'Distributor'쪽이 다중성 값이 1..1이라는 것을 보여주고 있다 반면 'Order' 객체의 다중성 값이 1..1인 것은 주문여부에 따라서 'Distributor'가 존재할 수 있음을 의미한다. 이를 XML 모델링을 하기 위해서는 XML Schema를 위한 XML 모델링 방법 ①과 ④를 적용하여 XML Schema로 모델링 하면 〈그림 4〉와 같다.

```
〈element name="Order"〉
 〈complexType〉
  〈sequence〉
   〈element name="Publication Month" type="date"
            minOccurs="1" maxOccurs="1"/〉
   〈element name="Quantity type="int" minOccurs="1"
            maxOccurs="1"/〉
   〈element name="Required Delivery Date" type="date"
            minOccurs="1" maxOccurs="1"/〉
   〈element REF="Distributor"
  〈/sequence〉
 〈/complexType〉
〈/element〉

〈element name="Distributor"〉
 〈complexType〉
  〈attribute name="Distributor" type="ID" use="required"/〉
 〈/complexType〉
〈/element〉
```

〈그림 4〉 Order객체 XML Schema 모델링

〈그림 4〉에서 'Order' 객체는 3개의 자식 객체와 1개의 참조객체를 가지고 있다. 또한 'Order' 객체에서 참조하는 'Distributor' 객체는 자식 객체를 갖고있지 않으면서 'Order Receipt' 객체와 연관관계로 인하여 type=ID로 설정하고 'Order' 객체에서 반드시 기술해야 한다는 〈표 1〉에 의해서

use=required로 설정하기 위해서 attribute로 정의하였다.

〈그림 3〉에서 'Delivery Notice' 객체와 'Shipping Agent' 객체는 관계성에서 집합관계이며 이 의미는 'Delivery Notice' 객체가 'Shipping Agent' 객체를 갖고 있어야하며 'Delivery Notice' 객체의 존재여부에 따라 'Shipping Agent' 객체가 존재할 수 있다. 이를 XML 모델링을 하기 위해서는 XML Schema를 위한 XML 모델링 방법 ①과 ④를 적용하여 XML Schema로 모델링 하면 〈그림 5〉와 같다.

```
〈element name="Delivery Notice〉
 〈complexType〉
  〈sequence〉
   〈element name="Publication Month type="data"
            minOccurs="1" maxOccurs="1"/〉
   〈element name="Quantity type="int"
            minOccurs="1" maxOccurs="1"/〉
   〈element name="Expected Arrival Date" type="date"
            minOccurs="1" maxOccurs="1"/〉
   〈element name="Date Shipped" type="date"
            minOccurs="1" maxOccurs="1"/〉
   〈element REF="Shipping Agent"
  〈/sequence〉
 〈/complexType〉
〈/element〉

〈element name="Shipping Agent"〉
 〈complexType〉
  〈attribute name="Shipping Agent"
            type="string" use="required"/〉
 〈/complexType〉
〈/element〉
```

〈그림 5〉 Delivery Notice객체의 XML Schema 모델링

〈그림 5〉에서 'Delivery Notice' 객체는 4개의 자식객체와 1개의 참조객체를 가지고 있다. 또한 'Delivery Notice' 객체에서 참조하는 'Shipping Agent' 객체는 자식객체를 갖고있지 않으면서 'Delivery Notice' 객체에서 반드시 기술해야 한다는 〈표 1〉에 의해서 use=required로 설정하기 위해서 attribute로 정의하였다.

〈그림 3〉에서 'Order Receipt' 객체와 'Order' 객체는 관계성의 종류에서 연관관계를 나타내며, 객체들간의 연관관계에서 역할이름은 '〈〈Response to〉〉'이다. 'Order Receipt' 객체는 4개의 자식객체를 포함하고 있으며 이중에서 'Expected Delivery Date' 객체는 'Order' 객체의 'Distributor' 속성과의 관계성을 고려해서 type=IDREF로 설정하였다. 이를 XML 모델링을 하기 위해서 XML Schema를 위한 XML 모델링 방법 ①과 ④ 그리고 관계성을 적용하여 XML Schema로 모델링하면 〈그림 6〉과 같다.

```
〈element name="Order Receipt"〉
 〈complexType〉
  〈sequence〉
   〈element name="Quantity" type="int"/〉
   〈element name="Expected Delivery
                Date" type="IDREF"/〉
   〈element name="Amounted Billed" type="double"/〉
   〈element name="Status" type="string"/〉
  〈/sequence〉
 〈/complexType〉
〈/element〉
```

〈그림 6〉 Order Receipt객체의 XML Schema 모델링

〈그림 3〉에서 'Order' 객체와 'Order Receipt' 객체간의 연관관계의 관계성 표현은 XML 모델링 방법 ⑤에 의해서 〈그림 7〉과 같다.

```
〈element name="Order Receipts"〉
  〈keyref refer="OrderID" name="dummy"
    〈selector xpath="Order Receipt"/〉
    〈filed xpath="Expected Delivery Date"/〉
  〈/keyref〉
〈/element〉

〈element name="Orders"
  〈key name="OrderID"〉
    〈selector xpath="Order"/〉
    〈filed xpath="Distributor"/〉
  〈/key〉
〈/element〉
```

〈그림 7〉 Order Receipt Schema의 관계성 표현

6 데이터 모델링

데이터베이스 스키마 변환 과정에서 정보 구조로부터 논리적 개념을 이용하여 어떤 논리적(데이터) 구조로 표현하는 것이 필요한데, 데이터 모델링이란 이 변환 과정을 말한다(이석호 1999).

6.1 객체-관계 데이터베이스 스키마 모델링

〈그림 3〉의 클래스 다이어그램을 객체-관계 데이터베이스 스키마로 변환한다. 〈표 1〉과 〈표 2〉에서 언급한 내용으로 객체(Object), 관계성(Relationships), 멀티플리시티(Multiplicity), 방향성(Direction)이 만들어지며, 이들은 ORDB Schema 객체, 즉 객체 타입, 객체 테이블, 참조 대 포함, 컬렉션으로 변환된다.

6.2 객체-관계 데이터베이스 변환 방법

〈그림 3〉의 클래스 다이어그램을 객체-관계 데이터베이스로 변환하는 방법은 다음과 같다(이상태 2001).

① 객체 클래스는 구조화된 객체 타입을 생성하고 테이블과 연결하여 타입을 1개 또는 여러 개의 테이블에 객체의 원소가 됨.

② 객체 타입은 사용자 정의 타입이거나, 속성을 가진 객체 타입임.

③ 클래스에서의 객체 속성은 객체 타입에서의 속성임.

④ 타입이 테이블이나 객체 또는 특정 타입으로 변환때 클래스의 객체 속성 타입은 객체 타입의 속성 타입임.

⑤ 객체 속성이 NULL제약조건을 가지면 NOT NULL 제약조건도 가짐.

⑥ 객체 속성이 초기 값을 가지며 컬럼에 DEFAULT값을 추가함.

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

⑧ 부 클래스를 위해 기본키 제약조건과 외래키 제약조건에 따라 각각의 부모

클래스의 키를 추가함.

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

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 집합을 위해 기본키를 생성하고, 기본키를 위한 컬럼을 추가하며, 테이블 안에서 그 집합을 저장하기 위한 객체 타입을 사용함. 또한 단일 객체를 위한 객체 속성 또는 컬렉션을 통하여 배열이나 중첩 테이블을 사용함.

⑭ 너무 많이 이동하게 되는 이진 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 제약조건을 생성함.

⑰ 객체 클래스의 전달 작용을 위한 객체 타입에 메소드를 생성함.

6.3 객체-관계 데이터베이스 스키마 변환 예

(1) 〈그림 3〉과 〈표 1〉및 〈표 2〉에 의한 'Distributor' 객체는 ORDB 변환 방법 ③번의 성질에 따라 'Distributor_t' 객체타입 속성을 저장하는 'Distributor' 테이블로 〈그림 8〉과 같이 변환된다.

```
SQL> CREATE TYPE Distributor_t AS OBJECT
(
  Distributor       string,
);
SQL> CREATE TABLE Distributor
       OF Distributor_t;
```

〈그림 8〉 Distributor 테이블

(2) 〈그림 3〉과 〈표 1〉및 〈표 2〉에 의한 'Shipping Agent' 객체는 ORDB 변환 방법 ③의 성질에 따라 'Shipping Agent_t' 객체타입 속성을 저장하는 'Shipping Agent' 테이블로 〈그림 9〉와 같이 변환된다.

```
SQL> CREATE TYPE Shipping Agent_t AS
       OBJECT
(
   Shipping Agent    string,
);
SQL> CREATE TABLE Shipping Agent
       OF Shipping Agent_t
```

〈그림 9〉 Shipping Agent 테이블

(3) 〈그림 3〉과 〈표 1〉및 〈표 2〉에 의한

'Delivery Notice' 객체는 ORDB 변환 방법 ③과 ④의 성질에따라 'Delivery Notice_t' 객체타입 속성을 저장하며 속성 'Shipping Agent'의 객체타입은 〈그림 9〉에서 정의한 'Shipping Agent_t'를 적용하고 〈그림 10〉과 같이 변환한다.

```
SQL> CREATE TYPE Delivery Notice_t AS
     OBJECT
(
  Publication Month       date,
  Quantity                int,
  Expected Arrival Date   date,
  Date Shipped            date,
  Shipping Agent          Shipping Agent_t,
  );
SQL > CREATE TABLE Delivery Notice OF
      Delivery Notice_t
```

〈그림 10〉 Delivery Notice테이블

(4) 〈그림 3〉과 〈표 1〉및 〈표 2〉에 의한 'Order' 객체는 ORDB 변환 방법 ③과 ④의 성질에 따라 'Order_t' 객체타입 속성을 저장하며 'Distributor'의 객체타입은 〈그림 8〉에서 정의한 'Distributor_t'를 적용하고 〈그림 11〉과 같이 변환한다.

```
SQL> CREATE TYPE Order_t AS OBJECT
(
  Publication Month       date,
  Quantity                int,
  Required Delivery Date  date,
  Distributor             Distributor_t,
);
SQL> CREATE TABLE Order OF Order_t;
```

〈그림 11〉 Order 테이블

(5) 〈그림 3〉과 〈표 1〉및 〈표 2〉에 의한 'Order Receipt' 객체는 ORDB 변환 방법 ③과 ④및 ⑫의 성질에 따라 'Order Receipt_t' 객체타입 속성을 저장하며 'Order' 객체와 'Order Receipt' 객체는 연관관계로서 'Order' 정보속성(refOrder)에 대해서 SCOPE FOR문을 이용하여 참조하는 'Order' 객체 테이블 명을 정의하여 〈그림 12〉와 같이 변환한다.

```
SQL> CREATE TYPE Order Receipt_t AS
     OBJECT
(
   Quantity                int,
   Amount Billed           double,
   Expected Delivery Date  date,
   Status                  string,
   refOrder                REF Order_t
);
SQL> CREATE  TABLE  Order Receipt OF
                    Order Receipt_t(
                    SCOPE FOR (refOrder)
                    IS Order);
```

〈그림 12〉 Order Receipt 테이블

6.4 객체지향 데이터베이스 스키마 모델링

〈표 1〉과 〈표 2〉에서 언급한 내용으로 객체(Object), 관계성(Relationships), 멀티플리시티(Multiplicity), 방향성(Direction)이 만들어지며, 〈그림 3〉의 클래스 다이어그램을 객체지향 데이터베이스 스키마로 변환한다.

6.5 객체지향 데이터베이스 변환 방법

〈그림 3〉의 클래스 다이어그램을 객체지향 데이터베이스 스키마로 .변환하는 방법은 다음과 같다(최문영 2001).

① "영속(persistent)" 클래스는 영속 객체지향 데이터베이스 클래스임.

② 인터페이스가 "영속" 클래스 객체지향 데이터베이스 인터페이스로 실행되기 위한 인터페이스를 지원하는 언어(JAVA, ODL)나 또는 추상적인 베이스 클래스를 위해 오직 순수한 가상멤버 그리고 no data 멤버의 그 언어는 인터페이스를 갖지 않음.

③ type 분류자는 enum 또는 typedef로 표시함.

④ 클래스의 attribute는 적당한 type 변환과 제한으로 객체지향 데이터베이스 클래스의 attribute임.

⑤ 만약 nullable 태그가 나타나면 nullable 상수형을(nullable_short) 사용하고 바인딩을 지원하는 널값을(ODL) 사용한다. 만약 그렇지 않으면, 그것을(C^{++}, JAVA) 무시함.

⑥ 만약 attribute가 initializer를 가지면, 생성자에 초기화 코드를 추가함. 생성자 메소드의 어느 한쪽의 부분으로서 또는 C^{++} 멤버에 초기화는 목록에 나열함.

⑦ 서브 클래스는 클래스 선언에 슈퍼클래스 사양을 포함.

⑧ 연관 클래스는 연관 클래스의 attribute와 클래스를 생성함.

⑨ 명백한 객체 일치 {oid} 또는 후보 키 {alternate oid}가 만약 바인딩을 지원하면 키의 선언을 명시함. 만약 지원하지 않으면, 객체의 집합에 억제를 점검하는 적당한 메소드를 공급함.

⑩ 각 명백한 억제에 대한 적당한 클래스에 메소드를 추가하고 안전하게 함. 시스템은 메소드를 시스템이 그 억제가 만족할 만한 것을 요구할 때마다 호출함.

⑪ 적당한 객체 또는 연관된 Multiplicity로부터 유래한 컬렉션 타입에 대해서는 연관 클래스를 가지고 있지 않은 각 이진 연관을 위해 관계성을 생성함. 명백한 화살표들이 연관되어 있지 않다면 역 관계성들을 사용함.

⑫ 다른 클래스의 각 역할 링크를 위한 연관 클래스에 관계성을 생성함.

⑬ 코드를 생성하거나 또는 특별한 객체지향 데이터베이스의 요구처럼 어떠한 복합 집단화 연관을 위해 삭제를 전파하기 위하여 객체지향 데이터베이스 특징들을 사용함.

⑭ 3진으로 이루어진 연결을 위해 연관 클래스를 생성함. 그리고 연관 클래스에서 Multiplicity를 주었던 적당한 자료 type의 연합한 클래스까지 생성함.

6.6 객체지향 데이터베이스 스키마 변환 예

(1) 〈그림 3〉과 〈표 1〉 및 〈표 2〉에 의한 'Distributor' 객체는 'Order' 객체와 집합관계이며 이를 OODB 변환 방법 ④에 의해서 〈그림 13〉의 'Distributor' 테이블로 변환된다.

```
Class Distributor(extent Distributors) {
 attribute String  Distributor;
 relationship      Order;
};
```

〈그림 13〉 Distributor 테이블

(2) 〈그림 3〉과 〈표 1〉 및 〈표 2〉에 의한 'Shipping Agent' 객체는 'Delivery Notice' 객체와 집합관계이며 이를 OODB 변환방법 ④에 의해서 〈그림 14〉의 'Shipping Agent' 테이블로 변환된다.

```
Class Shipping Agent(extent Shipping Agents) {
 attribute  String  Shipping Agent;
 relationship       Delivery Notice;
};
```

〈그림 14〉 Shipping Agent 테이블

(3) 〈그림 3〉과 〈표 1〉 및 〈표 2〉에 의한 'Delivery Notice' 객체와 'Shipping Agent' 객체는 하나의 서브 객체 관계이다. OODB 변환방법 ④와 ⑦에 의해서 〈그림 15〉의 'Delivery Notice' 테이블로 변환한다.

```
Class Delivery Notice(extent Delivery Notices) {
 attribute  Date  Publication Month;
 attribute  Int   Quantity;
 attribute  Date  Expected Arrival Date;
 attribute  Date  Date Shipped;
 relationship     Shipping Agent;
};
```

〈그림 15〉 Delivery Notice 테이블

(4) 〈그림 3〉과 〈표 1〉 및 〈표 2〉에 의한 'Order' 객체와 'Distributor' 객체는 하나의 서브 객체 관계이다. OODB 변환방법 ④와 ⑦에 의해서 〈그림 16〉의 'Order' 테이블로 변환한다.

```
Class Order (extent Orders) {
 relationship      Distributor;
 attribute  Date   Publication Month;
 attribute  Int    Quantity;
 attribute  Date   Required Delivery Date;
 relationship      Order Receipt;
};
```

〈그림 16〉 Order 테이블

(5) 〈그림 3〉과 〈표 1〉 및 〈표 2〉에 의한 'Order Receipt' 객체는 'Order' 객체와 연관관계이다. OODB 변환방법 ⑧과 ⑪에 의해서 〈그림 17〉의 'Order Receipt' 테이블로 변환한다.

```
Class Order Receipt(extent Order Receipts) {
 attribute  Int      Quantity;
 attribute  Date     Expected Delivery Date;
 attribute  Double   Amount Billed;
 attribute  String   Status;
 relationship        Order;
};
```

〈그림 17〉 Order Receipt 테이블

6.7 관계 데이터베이스 스키마 모델링

〈그림 3〉의 클래스 다이어그램을 관계형 데이터베이스 스키마로 변환한다. 〈표 1〉과 〈표 2〉에서 언급한 내용으로 객체

(Object), 관계성(Relationships), 멀티플리시티(Multiplicity), 방향성(Direction)이 만들어진다(이정수 2001).

6.8 관계 데이터베이스 변환 방법

〈그림 3〉의 클래스 다이어그램을 관계 데이터베이스 스키마로 변환하는 방법은 다음과 같다(이상태 외 2001).

① 클래스는 테이블임.
② 클래스의 속성(attribute)은 테이블의 열(column)임.
③ 클래스의 속성 타입은 테이블의 열 타입임.
④ 속성에 {nullable} 태그가 있으면 테이블 속성에 NULL 또는 NOT NULL을 추가함.
⑤ 속성이 초기치 값을 가지면, 열에 DEFAULT 문을 추가함.
⑥ 루트나 독립적인 클래스와 같이 일반화가 없는 클래스를 위해서는 integer 기본키를 생성하고, {oid}를 위해서는 기본키 제약조건에 {oid} 태그 열을 추가함.
⑦ 자식클래스(Subclasses)들은, 각 부모 클래스의 키를 기본키와 외부키 제약조건에 추가함.
⑧ 연관 클래스들은, 각 역할-실행 테이블에 대한 기본키를 기본키와 외부키 제약조건에 추가함.
⑨ 만일 {alternate oid = 〈n〉} 태그면, UNIQUE 제약조건에 대한 열을 추가함.
⑩ 각 명시된 제약에 대해 CHECK를 추가함.
⑪ 0....1, 1....1 규칙의 연관 관계에서 참조하는 테이블에 외부키를 생성함.
⑫ 집합 테이블(CASCADE와 같이)의 외부키를 갖는 복합집합을 위해서 기본키를 생성함 ; 기본키를 위해 추가적인 열을 추가함.
⑬ 이원 연관 클래스를 적당한 "N" 쪽 테이블로 이동함으로써 최적화함.
⑭ 연관 클래스가 아닌 3원 연관은 N : N에 대한 테이블을 생성함.
⑮ N : N, 3원 연관에서 역할-실행 테이블의 키로부터 기본키와 외부키 제약조건을 생성함.

6.9 관계 데이터베이스 스키마 변환 예

(1) 〈그림 3〉과 〈표 1〉 및 〈표 2〉에 의한 'Distributor' 객체는 관계형 데이터베이스 변환방법 ①과 ②및 ⑥번의 성질에 따라 'DistributorID' 객체타입 속성을 저장하는 'Distributor' 테이블로 〈그림 18〉과 같이 변환된다.

```
SQL> CREATE   TABLE  Distributor(
  DistributorID    INTEGER PRIMARY KEY
  )
```

〈그림 18〉 Distributor 테이블

(2) 〈그림 3〉과 〈표 1〉 및 〈표 2〉에 의한 'ShippingAgent' 객체는 관계형 데이터베이스 변환방법 ①과 ②및 ⑥번의 성질에 따라 'ShippingAgentID' 객체

타입 속성을 저장하는 'ShippingAgent' 테이블로 〈그림 19〉와 같이 변환된다.

```
SQL〉 CREATE   TABLE  ShippingAgent(
  ShippingAgentID  INTEGER PRIMARY KEY
  )
```

〈그림 19〉 ShippingAgent 테이블

(3) 〈그림 3〉과 〈표 1〉 및 〈표 2〉에 의한 'DeliveryNotice' 객체는 관계형 데이터베이스 변환방법 ①과 ②및 ⑥번의 성질에 따라 'DeliveryNotice' 객체타입 속성을 저장하며 속성 'ShippingAgent'는 변환방법 ⑪에 의해서 〈그림 19〉에서 정의한 'ShippingAgent'를 적용하고 〈그림 20〉과 같이 변환한다.

```
SQL〉 CREATE TABLE DeliveryNotice (
    DeliveryNoticeID  INTEGER  PRIMARY KEY
    Publication Month       date
    Quantity                int
    Expected Arrival Date   date
    Date Shipped            date
    Shipping AgentID INTEGER REFERENCE
                    Shipping Agent
CONSTRAINT DeliveryNotice_PK PRIMARY
      KEY(DeliveryNoticeID, ShippingAgentID)
)
```

〈그림 20〉 DeliveryNotice 테이블

(4) 〈그림 3〉과 〈표 1〉 및 〈표 2〉에 의한 'Order' 객체는 관계형 데이터베이스 변환방법 ①과 ②및 ⑥의 성질에 따라 'Order' 객체타입 속성을 저장하며 속성 'DistributorID'는 변환방법 ⑪에 의해서 〈그림 18〉에서 정의한 'Distributor'를 적용한다 또한 'OrderReceipt' 객체와의 연관관계를 변환방법 ⑧에 의해서 〈그림 21〉과 같이 변환한다.

```
SQL〉 CREATE TABLE Order (
      OrderID      INTEGER PRIMARY KEY
      Publication Month   date
      Quantity            int
      Required Delivery   date
      DistributorID INTEGER REFERENCE
                          Distributor
      ExpectedDeliveryDateID  INTEGER
                          REFERENCE
                          OrderReceipt
CONSTRAINT Order_PK PRIMARY
KEY(OrderID, DistributorID, Expected
DeliveryDateID)
)
```

〈그림 21〉 Order 테이블

(5) 〈그림 3〉과 〈표 1〉 및 〈표 2〉에 의한 'OrderReceipt' 객체는 관계형 데이터베이스 변환방법 ①과 ②및 ⑥의 성질에 따라 'OrderReceipt' 객체타입 속성을 저장하며 'Order' 객체와의 연관관계를 변환방법 ⑧과 ⑪에 의해서 〈그림 22〉와 같이 변환한다.

```
SQL〉 CREATE TABLE OrderReceipt (
    ExpectedDeliveryDateID INTEGER PRIMARY
                          KEY
    Quantity              int
    Amounted Billed       double
    Status                string
    OrderID   INTEGER REFERENCE Order
CONSTRAINT    ExpectedDeliveryDateID_PK
PRIMARY KEY (ExpectedDeliveryDateID, OrderID)
  )
```

〈그림 22〉 OrderReceipt 테이블

7 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 있다. 특히 계층적 구조를 갖는 XML 파일을 다양한 형태의 데이터베이스에 저장하기 위한 데이터 모델링 방안이 요구된다.

본 논문에서는 계층구조를 갖는 XML 파일을 기존에 개발된 다양한 형태의 데이터베이스로 저장이 가능하도록 하는 통합 모델링 방법론을 제안하였다. 이를 위하여 우선적으로 객체지향 설계언어인 UML를 이용해서 클래스 다이어그램을 도출한 후, 클래스 다이어그램에 의해서 W3C에서 연구중인 W3C XML Schema 설계를 위한 XML 모델링을 소개하고 이들의 모델링으로 교환되는 XML 데이터를 효율적으로 저장하기 위하여 객체-관계 데이터베이스 스키마와 객체지향 데이터베이스 스키마 그리고 관계형 데이터베이스 스키마로 변환하는 통합 설계 방법론을 제안한다. 본 연구에서 제안한 통합 설계 방법론은 XML 어플리케이션을 체계적이고 효율적으로 설계할 수 있으며 프로그램 개발을 통합적으로 추진할 수 있는 장점을 가지고 있다

참 고 문 헌

김채미, 최학열, 김심석 공저. 2001. 『전문가와 함께가는 XML Camp』. 마이트 Press.

방승윤, 주경수. 2001. XML 스키마를 관계형 스키마로의 변환에 관한 연구. 『한국정보처리지식 및 데이터 공학연구회 학술발표논문』: 293-299.

방승윤, 주경수. 2001. 전자상거래를 위한 UML기반의 XML 스키마 모델링에 관한 연구. 『순천향대학교 산업기술연구소 논문집』, 7(1) : 19-29.

방승윤, 주경수. 2002. UML Class 모델을 이용한 XML 응용 설계를 방법론. 『한국전자거래(CALS/EC)학회지』, 7(1): 153-166.

방승윤, 최문영, 주경수. 2002. 객체지향 데이터베이스 기반의 XML 응용을 위한, UML을 이용한 통합 설계 방법론. 『JOURNAL OF INFORMATION TECHNOLOGY APPLICATIONS & MANAGEMENT』, 9(1): 85-96.

이상태, 이정수, 주경수. 2001. 객체모델을 기반으로 한 XML DTD의 RDB 스키마로의 변환 방법. 『대한전자공학회 하계종합학술대회 논문집 Ⅲ』: 113-116.

이상태, 주경수. 2001. 객체모델을 이용한

XML DTD의 ORDB 스키마로의 변환. 『정보기술과 데이터베이스 저널』, 8(1): 111-112.

이석호. 1999. 『데이터베이스론』. 서울: 정익사.

이정수, 방승윤, 주경수. 2001. XML DTD의 관계형 데이터베이스 스키마의 자동 변환을 위한 컴포넌트 모델링. 『인터넷정보학회논문지』, 2(5): 81-91.

임춘봉, 신인철, 심재철 공저. 1999. 『UML 사용자 지침서』, 도서출판 인터비전.

최문영, 방승윤, 주경수. 2001. 객체모델을 이용한 XML DTD의 OODBMS 스키마로의 변환. 『한국정보처리학회 지식 및 데이터공학 연구회 제8회 학술발표대회 논문집』: 304-304.

Carey, M., Florescu, D., Ives, Z., Lu, Y., Shanmugasundaram, J., Shekita, E.,Subramanian, S. 2000. "XPERANTO: Publishing Object Relational Data as XML." In Suciu, D., Vossen(eds.), G., *Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000.* Dallas, Texas, USA, May 18-19, 2000, 105-110.

Cheng, J., Xu, J. 2000. "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.

Duckett Jon, Ozu Nik, Williams Kevin, Mohr Stephen, Cagle Jurt, Griffin Oliver, Norton Francis, Stokes-Rees Ian, and Tennison Jeni. 2001. *Professional XML Schemas.* Wrox Pr Inc.

Florescu, D., Kossmann, D. 1999. "Storing and Querying XML Data using an RDBMS." *Data Engineering,* 22(3): 27-34.

Kanne, C.-C., Moerkotte, G. 2000. "Efficient Storage of XML Data." *In Sixteenth International Conference on Data Engineering(ICDE'00),* 28 February-3 March 2000, San Diego, IEEE Computer Society Press, Los Alamitos, CA.

Klettke, M., Meyer, H. 2000. "XML and Object-Relational Database Systems-Enhancing Structural Mappings Based on Statistics." *In Suciu, D., Vossen(eds.), G., Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000,* Dallas, Texas, USA, May 18-19, 2000, 63-68.

Migrating from XML DTD to XML-Schema using UML,

〈http://www.rational.com/products/whitepapers/412.jsp〉.

Modeling XML vocabularies with UML, 〈http://www.xml.com/pub/a/2001/09/19/uml.html〉. 2001. 09 .19.

Shanmugasundaram, J., Shekita, E. J., Bar, R., Carey, M. J., Lindsay, B. G., Pirahesh, H., Reinwald, B. 2000. "Efficiency Publishing Relational Data as XML Documents." *In Abbadi, A. E., Brodie, M. L., Chakravarthy, S., Dayal, U., Kamel, N., Schlageter, G., Whang(eds.), K. Y., VLDB 2000-Proceedings of the 26th International Conference on Very Large Data Bases*, September 10-14, 2000, Cairo, Egypt, Morgan Kaufmann Publishers 2000.

UML for XML Schema Mapping Specification. [cited 1999.12.08]. 〈http://www.Rational.com/media/uml/resources/media/uml_xmlschema33.doc〉.

W3C Recommendation.[cited 2001.5.2]. 〈http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/〉.

What is XML ?. 〈http://www.xml.com/pub/a/98/10/guide1.html#AEN58〉.

XML Modeling. 〈http://www.xmlmodeling.com〉.