XML 응용 프로그램 프레임워크와 CASE 도구의 설계 및 구현

(Design and Implementation of XML Application Framework and CASE Tool)

박 철 만 † 박 상 원 † 김 형 주 ††
(Cheolman Park) (Sangwon Park) (Hyoung-Joo Kim)

요 약 XML의 등장으로 인터넷의 제2의 혁명을 가져올 것이라는 기대는 XML 스키마 정의 언어가 표준화됨으로써 현실적으로 실현 가능하게 되었다. XML 스키마를 설계하는 것은 DTD를 설계하는 것보다 더 복잡하며 더 많은 정보가 제공되게 된다. 이에 XML 스키마를 모델링하는 도구들이 등장하고 있으며, 이 도구는 일반적인 모델링 도구와는 다른 기능이 요구된다. 본 논문에서는 "설계 후 즉시 개발"이라는 XML 응용 프로그램 개발 방법론을 제안하고, 이를 XSD4j에서 구현하였다. XSD4j는 UML 기반의 XML 스키마 설계와 함께 XML 응용 프로그램 개발 기반을 제공해 준다. XSD4j를 사용하면 개발자는 쉽고 개념적인 수준의 스키마를 모델링할 수 있다. 또한, XML 문서를 위한 XML 저장매체 스키마의 생성 및 XML 문서와 응용 프로그램간의 표준적인 인터페이스인 DOM API를 얻을 수 있어 XML 응용 프로그램의 작성을 쉽게 할 수 있다.

Abstract XML Schema which enables the second revolution of Internet with XML is rapidly emerging as a standard for XML document structure modeling. XML Schema modeling has more complex structures and information than DTD modeling. In this reason, XML Schema modeling tool which has some more specialized functionalities than others is needed. In this paper, we have developed the XML application development method called "Design and develop" and implemented the prototype system named XSD4j which supports UML based XML Schema modeling and provides a XML application development platform. Using XSD4j, developers can get an easy and conceptual schema modeling method, XML storage schema for XML document, and persistent DOM API which is a standard interface between document and application program.

1. 서 론

1.1 연구 배경 및 필요성

최근 인터넷에서의 데이타 교환 및 문서 포맷의 표준으로 XML(eXtensible Markup Language)[1]이 각광을 받고 있다. XML 문서는 태그를 통해 의미 단위 별로 계층화되어 있는데, 이런 의미 단위 및 계층 구조는 DTD(Document Type Definition)를 통해 정의된다. 따라서 XML은 응용 프로그램이 DTD를 분석함으로써 취급하고자 하는 문서의 구조를 쉽게 파악하고 데이타를 효율적으로 추출할 수 있는 표준적인 기반을 제공할 수 있게 된다.

이러한 점에서 XML이 문서 표현의 표준으로 받아들여질 때 이것이 인터넷의 하나의 혁명을 이끌 것이라고 사람들은 기대를 하였으나, 현재 XML의 모습은 그 기대에 부응하고 있지는 못하다. 그 이유는 여러 가지가 있겠으나, 실제로 XML 응용 프로그램을 개발하기가 쉽지 않다는 점은 중요한 원인 중 하나이다. 이는 XML 응용 프로그램 개발 프로세스가 제대로 확립되지 못하였음을 말한다.

3장에서 보듯이 XML 응용 프로그램이 다루어야 하는 XML 문서는 잘 설계된 XML 스키마를 가지고, 영

· 본 논문은 두뇌한국21 사업에서 지원 받음.
† 비 회 원 : 서울대학교 컴퓨터공학부
cmpark@oopsla.snu.ac.kr
swpark@oopsla.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@oopsla.snu.ac.kr
논문접수 : 2001년 2월 21일
심사완료 : 2001년 9월 6일

구적으로 저장이 되어야 하며, 효율적으로 관리될 필요가 있다. 또한 같은 XML 문서가 다양하게 이용 될 수 있다. 현재의 XML 응용 프로그램의 개발 과정을 살펴보면 이런 XML 응용 프로그램의 요구사항을 만족시키기 위해 아래와 같이 4단계로 이루어지고 있다.

1. XML 스키마 설계도구를 이용하여 스키마를 설계
2. 설계된 스키마에 맞게 영구적으로 저장할 수 있는 기반을 조성
3. 영구적으로 저장된 XML 문서를 응용 프로그램이 사용할 수 있도록 중간 단계의 인터페이스를 생성
4. XML 응용 프로그램의 개발

현재까지 XML 응용 프로그램을 개발하기 위해서는 각 단계를 직접 개발자가 수행하여야 한다. 이는 개발자로 하여금 XML 응용 프로그램 개발이라는 애초의 목표에서 벗어나 기반을 조성하는 부가적인 작업을 수행하는 것으로써 개발 단계의 일관성을 많이 흩트리게 한다고 할 수 있다. 뿐만 아니라 현재까지의 연구동향도 각 단계별로 독립적으로 진행이 되고 있어 전체적인 응용 프로그램 개발이라는 측면에서의 연구는 많이 미흡한 실정이다. 이외에도 많은 문제점이 있으며 이는 3장의 서두에서 언급하도록 한다.

본 논문은 OOAD UML Modeler[2]를 이용하여 XML 스키마를 도입한 UML기반의 XML CASE 도구인 XSD4j(XML Schema Designer for Java)를 구현하였다. XSD4j에서는 XML 스키마를 도입하고, 개발 단계에서 개발자가 XML 스키마를 설계하고 바로 XML 응용 프로그램 개발에 착수할 수 있는 "설계 후 즉시 개발(Design and develop)"이라는 XML 응용 프로그램 개발 방법론을 제안하고 구현하였다. "설계 후 즉시 개발"은 전체적인 XML 응용 프로그램 개발 과정에 XML 응용 프로그램 개발 기반(XML Application Development Platform)을 제공함으로써 XML 스키마를 설계하고 설계된 스키마에 따라 응용 프로그램을 개발할 수 있는 일관성 있는 개발 프로세스를 제공할 수 있다.

XML 스키마의 설계에 있어서 기존의 트리 방식의 직접적이고 단순한 설계에서 벗어나, 표준적이고 논리적이고 개념적인 설계를 지원하기 위해 UML(Unified Modeling Language)[3]을 도입한다. 또한, 효과적인 XML 응용 프로그램 개발 기반을 구축하기 위해서는 설계단계에서 많은 자세한 정보가 제공되어야 하는데, 이 점에서 XML 스키마에 기반한 설계를 제공한다. 이를 위해서 UML을 XML 스키마 설계에 이용할 수 있도록 확장하였다.

XML 스키마를 이용하여 설계된 스키마는 다양한 데이타타입과 데이타 제약조건들을 포함하게 된다. 그러나 설계 당시에 정의된 이러한 정보는 개발 단계에서 큰 이득을 주지는 못한다. 왜냐하면 개발자는 설계된 스키마를 보고 이들을 개발 단계에서 반영해야 하기 때문이다. 이 문제점은 설계 단계의 결과물로 XML 문서와 응용 프로그램과의 중간 단계에서 데이타 변환 및 제약조건 체크 등을 수행하는 응용 프로그램 개발 기반을 제공함으로써 해결할 수 있다. 이 응용 프로그램 개발 기반은 개발자에게 표준적이고 사용하기 쉬운 인터페이스가 되어야 하는데, DOM(Document Object Model)[4]이나 SAX(Simple API for XML)[5]와 같은 XML을 위한 표준적인 인터페이스를 응용 프로그램 개발 기반으로 사용함으로써 설계 단계에서 정의된 내용을 쉽고 자연스럽게 개발 단계로 전달시킬 수 있을 것이다. 이 과정을 통하여 개발자에게 XML 스키마를 설계하고 바로 응용 프로그램 개발에 착수할 수 있는 "설계 후 즉시 개발"이라는 XML 개발 방법론을 제공할 수 있는 것이다.

본 논문은 2장에서는 UML과 기존의 XML 스키마 정의 언어와의 매핑 규약과, 현재 나와있는 여러 XML 스키마 에디터에 대해 살펴봄으로써, 유사 제품이 가지는 한계점과 그 한계의 극복 방안에 대해서 알아보기로 한다. 또한, XML 문서의 영구성 지원을 위한 XML 저장방법에 대한 현황도 살펴본다. 3장에서는 XML 응용 프로그램의 개발 단계와 각 단계별 고려사항을 알아보고, XML CASE 툴이 제공해야 할 기능들을 제시한다. 4장에서는 XSD4j의 설계 및 구현에 대해 살펴보고, XSD4j가 개발 단계에서 차지하는 역할과 그것의 장단점을 제시한다. 5장에서는 XSD4j를 이용하여 실제로 간단한 XML 응용프로그램을 작성하는 과정을 보고, 6장에서는 결론 및 향후 연구 계획에 대해 살펴보기로 한다.

2. 관련 연구

2.1 XML 스키마 정의 언어

이미 업계의 대표적인 위치에 있는 여러 업체들은 DTD의 한계성을 깨닫고, 나름대로의 XML 스키마 정의 언어를 만들어 사용하고 있었다. 이들의 특징은 다양한 데이타타입 및 사용자 정의 타입의 지원, 데이타 제약 조건의 지원, 그리고 응용 프로그램 작성에 필요한 여러 부가적인 정보를 표현할 수 있다는 점 등을 들 수 있다. XML 스키마 정의 언어의 예로는 DCD(Document Content Description), SOX(Schema for Object Oriented

XML), DDML(Document Definition Markup Language) 등을 들 수 있다. W3C (World Wide Web Consortium)에서는 이러한 XML 스키마 정의 언어 표준화의 필요성을 느끼고, 이에 대한 표준화 작업을 진행중에 있다. XML 스키마 안[6]의 내용을 살펴보면, 크게 문서의 구조를 정의하는 부분(XML Schema Part 1: Structures)과 데이타타입을 정의하는 부분(XML Schema Part 2: Datatypes)으로 나눌 수 있다. 즉, 향상된 데이타타입의 지원이 큰 비중을 차지하고 있음을 알 수 있다. 기존의 DTD가 문서의 구조만을 정의한데 반해, XML 스키마는 문서의 구조 외에도 실제로 문서의 각 요소가 어떤 데이타타입으로 정의되고, 그 요소가 가져야 하는 특별한 제약조건들도 반영할 수가 있다. 이 외에도 XML 스키마 자체가 XML로 정의되어 있다는 점, 상속과 같은 객체지향 개념의 도입, Namespace의 지원과 같은 특징들을 지니고 있다.

XML 스키마가 표준화되기 이전부터 XML EDI, B2B 등의 응용에서는 이를 도입하기 시작하였으며, 표준화가 된 지금에서는 많은 응용들이 XML 스키마를 지원하고 있다. 대표적으로 전자상거래의 세계적인 표준안인 ebXML(e-business XML)[7]에서 XML 스키마를 도입하고 있다.

2.2 UML과 XML 스키마의 매핑

UML은 표준적인 객체 모델링 언어이며, XML 스키마를 UML로 매핑하는 것은 표준적인 모델링 방법으로 XML 스키마 설계를 가능하게 한다는 의미이다. 현재 UML과 XML 스키마 정의 언어와의 매핑 규약은 Rational Software사와 CommerceOne사가 합작으로 제시한 'SOX를 위한 UML 매핑 규약'[8]이 있다. 이 규약은 UML의 확장 메카니즘(UML extension mechanism)[9][1]을 사용, XML 스키마에 존재하는 컴포넌트를 새로운 UML 클래스로 생성하고 있다. 이 방법의 장점은 모델링의 방법에서 UML이라는 표준을 따른다는 점이다. 따라서, 기존의 UML을 지원하는 CASE 도구에서 별다른 수정 없이 XML 스키마를 모델링할 수 있다는 장점을 지닌다. 그러나, 단지 규약일 뿐이며 구현에 대한 내용은 CASE 도구 개발업체에 전가시키고 있다.

2.3 XML 스키마 모델링 툴

현재 직접 XML 스키마를 지원하는 모델링 툴로 Extensibility사의 XML Authority[10], Icon Information-Systems사의 xmlspy[11]가 있다. 이 제품은 XML 스키마를 일반적으로 많이 사용하는 여러 XML 스키마 정의 언어로도 변환할 수 있는 기능을 가지고 있다. 실제로 모델링하는 방법에 있어서 두 제품은 XML 에디터 수준의 모델링 도구를 제공하고 있는데 이 때문에 XML 스키마 문서를 편집한다는 성격이 강하다고 할 수 있다.

2.4 XML 문서와 XML 저장매체

XML 문서를 저장하기 위한 저장매체에 대한 연구는 크게 세 분류로 나누어 볼 수 있다. 첫 번째는 관계형 데이타베이스 시스템을 이용하는 것이다. 이는 XML 문서를 검증된 데이타베이스 시스템인 RDBMS를 이용해서 저장함으로써, RDBMS를 위한 많은 기술들을 그대로 이용할 수 있다는 장점을 지니고 있다. 그러나 XML 문서의 구조와 RDBMS의 구조는 일대일 대응이 되지 않기 때문에 적절한 변환과정이 필요하게 되는데, 이 과정에서 많은 문제점들이 나타날 수 있다. 여기에 해당하는 연구내용으로는 STORED[12], 인라인 테크닉[13], Oracle8i[14], XML-View[15] 등에서 살펴 볼 수 있다.

두 번째 방법으로는 객체 데이타베이스 시스템을 이용하는 것이다. 객체 데이타베이스 시스템이란 객체관계형 데이타베이스 시스템이나 객체지향 데이타베이스 시스템을 말한다. ODBMS는 그 구조가 XML 문서와 일대일 대응이 될 수 있으므로, RDBMS와는 달리 복잡한 변환과정을 필요로 하지 않는다. 그러나 단점으로는 아직까지 ODBMS 관련 기술이 RDBMS의 기술에 비해 아직 미숙하다는 점이 있다. 여기에는 eXcelon[16], XML in SOP[17] 등의 예가 있다.

세 번째 방법으로는 XML 전용 저장매체를 이용하는 것이다. XML 문서를 직접 저장할 수 있으므로 앞의 두 방식과는 달리 변환과정을 필요로 하지 않는다. 또한, XML 문서에 대해서는 가장 자연스러워 보이는 방법이기도 하다. 그러나 XML 전용 저장매체에 대한 연구가 그렇게 오래 되지 않았으므로 성능이나 안정성 등에서 아직까지는 검증되지 않았다고 할 수 있다. 이 방법을 이용한 시스템으로는 Lore[18]가 있다.

3. XML 응용 프로그램 프레임워크

XML 응용 프로그램의 개발 과정에 앞서, 응용 프로그램이 다루게 되는 XML 문서의 일반적인 특징을 살펴보면 아래와 같은 세 가지로 분류할 수 있다.

---

1) UML의 확장 메카니즘이란 Stereotype과 tagged value를 의미한다. Stereotype이란, 기존에 존재하는 UML 클래스로부터 상속받아 새로운 클래스를 만들어 내는 메카니즘을 의미한다. tagged value는 이름과 값의 쌍으로 된 구조로, Stereotype의 속성 리스트(Property list)로 사용되기 위한 구조체이다.

1. XML 문서는 복잡한 구조를 지니고 있다.

기존의 문서들은 단순한 나열 구조를 지니고 있는데 반하여, XML 문서는 중첩된 구조와 요소간의 참조들이 많이 나타나게 된다. 이러한 특징으로 인해, XML 문서의 구조를 모델링할 때에는 개념적이고 논리적인 수준의 디자인 방식을 도입하여야 한다.

2. XML 문서는 영속성을 지니며, 그 규모가 방대하다.

XML 문서는 그 자체로 데이타의 성격을 지니기도 한다. 이는 그 문서가 일시적으로 사용되는 것이 아니라 영구적으로 보존되어야 한다는 것을 의미한다. 또, XML 문서는 중첩이 가능한 구조이기 때문에 작은 XML 문서들이 모여 하나의 큰 XML 문서가 형성되는 경향을 지니게 된다. 이로 인해, 하나의 XML 문서는 규모 면에서 하나의 데이타베이스에 해당할 정도로 방대해진다. 이 거대한 문서를 영구적으로 또 효율적으로 관리하기 위해서는 특별한 XML 저장매체에 대한 고려가 필요해 지게 된다.

3. XML 문서는 그 문서를 이용하는 다양한 응용 프로그램들이 존재한다.

XML 문서의 방대성만큼이나 이 문서를 이용하기 위한 다양한 응용 프로그램들이 존재하게 된다. 이들 응용 프로그램은 XML 문서의 일부집합일 수도 있고, 전체 문서에 대한 검색일 수도 있으며, 데이타 마이닝과 같은 작업일 수도 있다. 그 만큼 다양한 응용이 존재하는데, 매번 XML 저장매체의 하부 구조를 직접 접근하여 응용 프로그램을 개발하기에는 많은 문제점이 존재한다. 따라서, XML 저장매체에 독립적인 별도의 인터페이스가 필요하게 되며, DOM API는 표준적인 인터페이스로써 가장 적합하다고 볼 수 있다.

이 때문에 XML 응용 프로그램 개발 시에 XML 문서의 영속성과 효율적인 관리가 전제되어야 하며, 다양한 응용 프로그램들을 쉽게 개발할 수 있는 기반을 마련하는 것이 중요해진다. 일반적으로 XML 응용 프로그램의 개발은 그림 1과 같이 4단계로 이루어진다. XML 스키마 모델링 도구를 이용하여 XML 스키마를 설계한 다음, 이 XML 문서에 맞는 XML 저장 매체 스키마를 생성하고, 이 저장 매체 스키마와 응용 프로그램간의 인터페이스를 생성하는 과정을 통해 XML 응용 프로그램 프레임워크를 구축할 수 있다.

현재 이 일련의 과정이 어떻게 진행되는가를 살펴보면 다음과 같다.

먼저, XML 스키마의 설계에 있어서는 주로 트리 형태에 기반한 설계 기법이 사용되어지고 있는데, 이는 XML 문서의 구조가 일반적으로 트리 구조로 나타나기

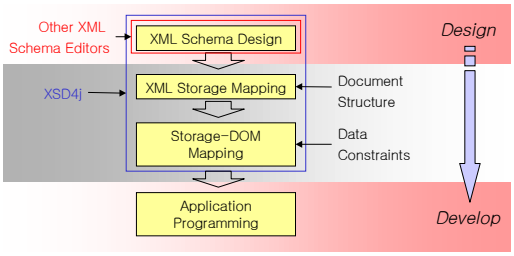

그림 1 XML 응용 프로그램의 개발 단계

때문이다. 이런 설계 방식은 XML의 구조에 맞춘 물리적이고 직접적인 설계라고 할 수 있다. 즉, XML 스키마 문서를 직접 편집하는 것과의 차이점이 크게 나지 않는 수준이다. 뿐만 아니라, XML 스키마로 가장 많이 사용되어 지는 DTD의 경우에도 그것이 표현하고 있는 정보는 XML 문서에서 사용되어지는 태그와 그것의 구조만 정의하기 때문에, 실제 응용에서 사용하기에는 DTD가 표현하고 있는 정보로는 충분하지 못하다. 이 문제점을 해결하기 위해 최근 XML 스키마 정의 언어(XML Schema Definition Language)[6]가 나타나게 되었다. XML 스키마는 XML 문서 구조의 정의를 위해 객체지향적 개념을 도입하고, 데이타타입 및 데이타 제약조건에 대해서도 지원을 하는 등 DTD에 비해 구체적이고 실용적인 정보를 표현하는데 초점을 맞추고 있다.

설계가 끝나면 개발자는 XML 문서를 저장하기 위한 저장매체를 선택하여야 하고, 만일 이 저장매체가 XML 스키마를 지원하지 않는다면 XML 스키마를 저장매체 스키마로 변환하는 작업을 하여야 한다. 현재 XML 전용의 저장매체도 나오고 있기는 하나 안정성이나 성능면에서 검증받지 못했으며 기존 시스템과의 통합에서도 문제가 발생한다. 따라서, 이미 그 안정성이나 성능이 입증된 관계형 데이타베이스시스템이나 객체관계형 데이타베이스시스템 등을 저장매체로 이용할 경우 이러한 문제점을 해결할 수 있다. XML 전용의 저장매체에 대한 시도는 2.4절에서 볼 수 있듯이 하부에 일반적인 데이타베이스 시스템을 두고 상위에 XML을 지원할 수 있는 인터페이스를 둔 방식과, 하부에 XML을 위한 전용의 저장매체를 두는 방식으로 나누어 볼 수 있다. 이런 시도는 상위의 인터페이스가 하위의 저장매체에 종속적이거나 저장매체 자체가 너무 비대해지는 문제가 발생될 수 있다.

이렇게 저장매체가 선택이 되고 스키마의 변환이 이루어졌다면, 저장매체에 저장된 문서를 XML 문서로 되돌려서 응용 프로그램이 사용할 수 있도록 저장매체와 응용 프로그램간의 인터페이스를 제작하는 것이 필수적

이다. 이 인터페이스에서는 저장매체에서 발생할 수 있는 에러 체크 및 XML 문서의 데이타 제약조건에 대한 고려가 필요하다. 이런 오류 체크는 일반적인 응용 프로그램의 개발에서 전체 개발 코드의 50% 정도를 차지할 정도로 그 비중이 크다. 따라서 이 단계가 XML 응용 프로그램 개발 단계에서 차지하는 비중이 상대적으로 많다고 볼 수 있다.

이 단계가 끝난 후 본격적인 XML 응용 프로그램 개발에 착수할 수 있다.

각 과정을 살펴보면, XML 저장매체 스키마 변환 과정과 저장매체와 응용 프로그램 간 인터페이스 개발은 XML 응용 프로그램 개발이라기보다는 XML 문서를 위한 기반 구축 과정이라고 볼 수 있다. 따라서, XML 응용 프로그램 프레임워크는 이 두 단계를 XML 응용 프로그램 개발 기반으로 보고 이를 시스템이 자동으로 생성함으로써 개발자로부터 숨기고, 개발자가 XML 스키마 설계 후 별도의 부가적인 작업 없이 개발에 들어갈 수 있는 두 단계의 프로세스를 구현한다. 또한 XML 스키마 설계를 위해 UML 기반의 설계 도구를 지원함으로써 "설계 후 즉시 개발"이라는 개발 방법론을 구현하였다.

아래에는 XML 응용 프로그램 프레임워크가 XML 개발의 각 단계별로 어떻게 지원이 되는지 살펴본다.

3.1 XML 스키마 설계

XML 스키마를 디자인하는 방법에는 여러 가지가 있을 수 있다. 기존의 방법들[10, 11]은 일반적인 XML 에디터를 확장하여 트리구조로 XML 스키마를 설계하도록 하였으며, "XML 스키마를 위한 UML 매핑 규약"[8]에서는 기존의 UML 기반의 CASE 툴에서 XML 스키마 정의 언어 중 하나인 SOX 스키마를 디자인 할 수 있는 기반을 제공해 준다. 본 논문에서는 UML을 이용하여 XML 스키마를 설계하는 방법을 설명한다.

스키마 모델링에 UML을 사용하게 되면 앞에서 언급했다시피 개념적인 모델링이 가능하고 또한 표준을 따르게 된다. UML과 XML 스키마와는 일대일 대응이 되지 않으며, 그렇기 때문에 XML 스키마와 UML의 매핑 과정이 필요하게 된다.

실제로 XML 스키마는 그 스키마를 사용하게 될 문서가 어떤 엘리먼트와 애트리뷰트를 가질 것인지, 그 값은 어떤 조건을 만족해야 하는지, 또 그것의 계층구조는 어떻게 될 것인지를 정의하는 것이다. 즉, XML 스키마의 가장 중요한 요소는 바로 엘리먼트와 애트리뷰트라고 할 수 있으며, 모델링 과정에서도 가장 중요한 것이 바로 이 두 요소이다. 나머지 요소들은 대부분 이 두 요소의 데이타타입이나 제약조건들을 정의하는 요소들이다. 따라서, UML의 다이어그램 상에 나타나는 정보는 그림 2와 같이 엘리먼트와 애트리뷰트가 중심이 되며, 나머지 요소들은 각 요소들의 속성으로 표현되어 진다. 이 다이어그램 상에서 나타나지 않는 정보는 그림 3에서 보듯이 각 엘리먼트의 속성으로 나타낼 수 있다.

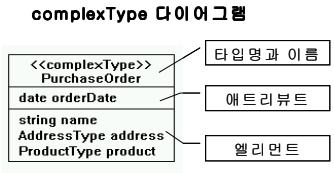

그림 2 XSD4j 모델러의 다이어그램

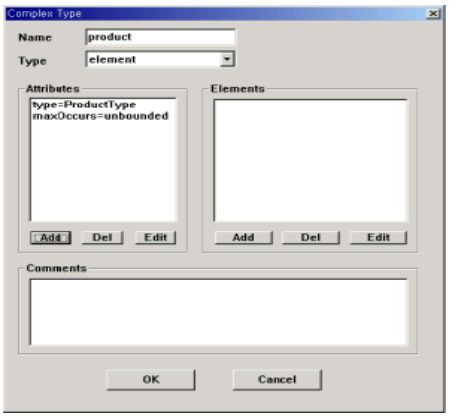

그림 3 product 엘리먼트의 속성

3.2 XML 스키마와 데이타베이스 스키마 매핑

XML 저장매체로는 여러 가지가 있을 수 있다. 일반적인 데이타베이스시스템일 수도 있고, XML 전용 스토리지 시스템일 수도 있으며, 파일시스템에 텍스트파일로 저장이 될 수도 있다. 어떤 저장매체가 좋은가에 대해서는 많은 논란이 존재한다. 그러나 본 논문에서는 어떤 시스템이 좋은지에 대해서는 언급하지 않기로 한다. 왜냐하면, XML 저장매체는 선택의 문제일 뿐, XML 응용 프로그램의 개발과는 큰 관련성이 없기 때문이다. 이는 XSD4j가 XML 저장매체와는 독립적이고, 표준적인 인터페이스인 DOM API를 생성하기 때문에 가능하다. XSD4j는 XML 저장매체로 관계형 데이타베이스시스템을 지원한다. ODBMS의 경우에는 XML 스키마에서 데이타베이스 스키마로의 매핑은 거의 일대일로 이루어질 수 있으며, XML 전용 스토리지 시스템의 경우는 별다른 매핑과정이 필요하지 않다. 파일시스템을 이용한

경우에도 별도의 XML 파서를 도입함으로써 마찬가지로 별다른 매핑과정이 필요하지 않게 된다.

XML 스키마와 관계형 데이타베이스 스키마의 매핑 규칙은 기본공유(Basic Shared) 테크닉, 공유 인라인(Shared Inlining) 테크닉, 하이브리드 인라인(Hybrid Inlining) 테크닉[13]과 유사한 방법을 이용한다. 이 테크닉은 원래 DTD로 정의된 XML 스키마를 관계형 데이타베이스 스키마로 변환하는 방식이다. XML 스키마는 DTD에 비해 데이타 타입이 향상되었으며 데이타 제약조건이 추가되었다. 이 장에서는 XML 스키마의 이 두 가지 특징이 데이타베이스 스키마 생성에 어떻게 적용될 수 있는지 살펴 볼 것이다.

데이타 제약조건은 XML 저장매체 스키마에 제약조건을 포함하는 방법과 DOM API에서 이를 포함하는 방법이 있을 수 있다. 전자의 경우 저장매체가 데이타 제약조건을 관리함으로써 DOM API가 간단해 지고 효율적으로 처리할 수 있는 장점이 있다. 그러나 저장매체마다 제약조건을 지원하는 방식이 다르기 때문에 제약조건을 일관적으로 관리하기가 힘들고, 개발자가 데이타 제약조건에 대한 처리를 하기가 힘들어진다는 단점이 있다. 후자는 저장매체에는 데이타 자체만을 저장하고 이에 대한 제약조건은 API에서 처리하는 방식이다. 이 방식은 저장매체의 제약조건을 지원하는 방식이 다르더라도 사용할 수 있으며, 인터페이스에 의해 제약조건이 관리되므로 일관적인 데이타 제약조건에 대한 인터페이스를 만들어 낼 수 있다. 반면에 저장매체에 제약조건과 관련된 정보가 없으므로 주어진 인터페이스가 아닌 다른 경로를 통한 데이타접근에는 제약조건을 제공하기가 힘들어지는 단점이 있다. 본 논문에서는 XML 응용 프로그램 프레임워크를 위해 일관적인 인터페이스를 생성할 수 있는 후자의 방법을 택하였다.

향상된 데이타타입은 [13]의 방법에 타입테이블이라는 개념을 도입하여 해결한다. 타입테이블이란 XML 스키마 정의에 사용된 모든 타입에 대해 이의 객체를 저장할 수 있는 각각의 테이블을 의미한다. 그림 4에서 보듯이 타입테이블을 추가함으로써 트리의 각 단말노드가 한 단계씩 확장되었다.

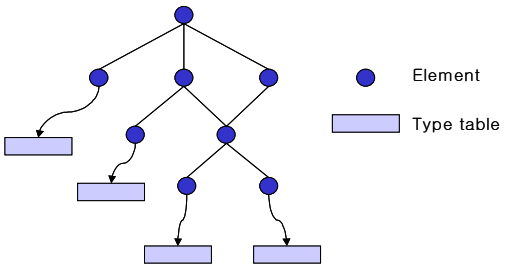

그림 4 타입테이블 개념을 도입한 XML 스키마 구조

3.2.1 기본 공유(Basic Shared) 테크닉

XML 스키마에서 실질적으로 문서에 나타나는 데이타를 정의하는 요소는 엘리먼트와 애트리뷰트가 있다. 기본적으로 엘리먼트는 RDB의 테이블로, 애트리뷰트는 테이블의 필드로 변환이 가능한데, 애트리뷰트의 경우 한 애트리뷰트의 타입은 기본타입(Primitive type)이므로 변환에 큰 문제는 없다. 따라서 엘리먼트에 대해서 RDB 스키마로 변환하는 과정을 간단히 살펴보도록 하자. 기본 공유 테크닉에서는 엘리먼트가 가질 수 있는 모든 타입에 대해서 새로운 테이블(타입 테이블)을 생성한다. 이 테이블에는 그 엘리먼트의 애트리뷰트 값만을 필드로 가지게 되며, 엘리먼트간의 구조는 별도의 엘리먼트 테이블을 두어 해결한다.

알고리즘은 우선 공통으로 사용하는 엘리먼트 테이블을 생성한 후, 최상위 엘리먼트에서부터 깊이 우선 순회를 통해 엘리먼트의 타입들을 순회하며 타입 테이블을 생성해 낸다.(표 1, 그림 5, 6)

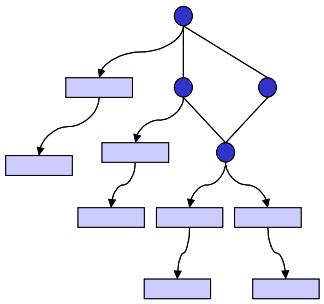

그림 5 깊이 우선 탐색으로 엘리먼트의 타입테이블을 생성

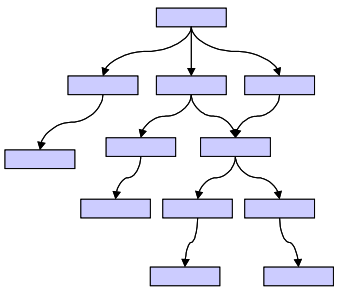

그림 6 모든 탐색을 마치면 모든 엘리먼트에 대해 타입 테이블이 생성됨

표 1 기본 공유 테크닉의 RDB 스키마 생성 알고리즘

```
BeginRDBSchemaGeneration() {
        GenerateElementTable();                 // Element를 위한 별도의 테이블을 생성
        ElementList = GetRootElements();
        for each element in ElementList
        do {
                MakeRDBSchema(element);
        }
}
MakeRDBSchema(element) {
        elementType = element.getType();                    // Element의 타입에 해당하는 컴포넌트
        ElementList = elementType.getElements();            // 타입의 엘리먼트에 대해서 작업을 수행
        for each element in ElementList
        do {
                MakeRDBSchema(element);
        }
        elementType.GenerateRDBSchema();       // Attribute들을 필드로 가지는 테이블을 생성
}
```

이 그림에서 상위 타입 테이블과 하위 타입 테이블을 연결한 선은 관계형 데이타베이스에서 외래키가 된다.

3.2.2 공유 인라인 테크닉 (Shared Inlining Technique)

기본공유 테크닉에 의해 생성된 결과는, XML의 구조를 다시 만들어 내기 위해서, 엘리먼트 테이블과 타입 테이블과의 조인 연산이 많이 발생하게 된다. 이러한 조인 연산은 엘리먼트 테이블의 내용을 타입 테이블에 옮김으로써 어느 정도 줄일 수는 있으나, 효과적인 수준은 아니다. 실제로 조인 연산은 시스템의 성능에 가장 큰 영향을 미치는 요소이므로, RDB 스키마를 생성하는데 있어 조인 연산을 줄이는 것은 중요하다.

공유 인라인 테크닉이란, 테이블간의 조인 연산을 줄이기 위해 도입하는 방식이다. 즉, 불필요하게 생성되는 타입 테이블의 수를 줄임으로써 조인 연산의 수를 줄일 수 있다는 것이다. 실제로 기본 공유 테크닉에서 엘리먼트의 타입마다 따로 테이블을 생성하는 이유는 엘리먼트가 나타날 수 있는 횟수가 일정하지 않기 때문이다. 그러나 엘리먼트가 나타날 수 있는 횟수가 제한되어 있다면 미리 조인된 결과를 테이블로 생성해 두는 것이 가능해 진다. 즉 유한개의 엘리먼트에 대해 기존의 방식은 각각의 엘리먼트가 하나의 행으로써 존재하였으나, 공유 인라인 테크닉은 유한개의 엘리먼트를 하나의 행에 여러 개의 필드로 나열함으로써, 마치 하나의 행처럼 표현되게 하는 것이다. 따라서 조인의 결과도 하나의 행으로 표현되므로, 미리 조인된 테이블로 나타낼 수 있는 것이다.

그림 7과 8에서 점선으로 표시된 타입 테이블은 이미 상위의 테이블에 포함되므로 실제로는 존재하지 않는 테이블이다. 실선으로 표시된 테이블은 나타나는 수가 무제한이므로 상위 테이블에 포함될 수 없다. 따라서 별도의 테이블을 생성하게 된다.

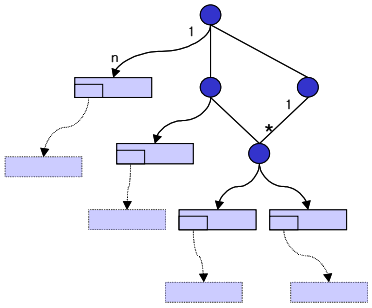

그림 7 상위의 타입 테이블 스키마에 하위 타입 테이블 스키마가 포함됨

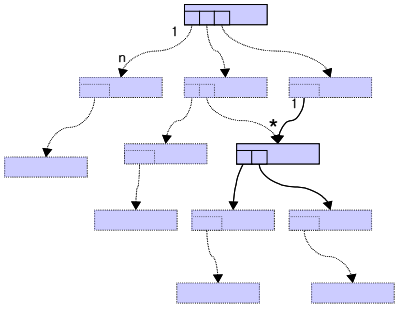

그림 8 1:unbounded의 관계일 경우 독립된 타입 테이블을 생성

3.3 저장매체와 응용프로그램간의 인터페이스

이 단계에서 API는 XML을 접근하기 위한 표준적인 방법이어야 하며, DOM(Document Object Model)[4]이나 SAX(Simple API for XML)[5]가 그 대상이 될 수 있다. 본 논문에서는 DOM API를 이용하여 저장매체와 응용 프로그램간의 인터페이스를 제공한다. DOM API를 이용하면 개발자로부터 XML 저장매체나 그 외의 XML과는 크게 관련이 없는 여러 가지 복잡한 작업들을 숨기고, XML 응용 프로그램 개발에만 전념할 수 있는 환경을 구축해 주는 표준적인 환경을 제공할 수 있다. DOM API를 통해 XML 응용 프로그램 개발자는 XML 문서가 응용 프로그램의 메모리 공간상에 로드되어 있다는 가정을 할 수 있으며, 따라서, XML 저장매체에 대한 최소한의 고려만으로 XML 문서의 영속성을 구현할 수 있다. 이에 대한 구체적인 내용은 4.4절에서 설명한다.

4. XSD4j의 설계 및 구현

4.1 구조

4.1.1 전체 구조

XSD4j가 하는 일을 크게 보면, XML 스키마를 모델링하는 것과, 모델링된 스키마를 이용하여 XML 응용 프로그램을 개발할 수 있는 환경을 구축해 주는 일로 나누어 볼 수 있다. 따라서, 그림 9와 같이 전체 구조를 크게 두 부분으로 나누어 볼 수 있는데, 그 하나는 스키마 모델링과 관련된 일을 하는 XSD4j 모델러(XSD4j Modeler)이고, 나머지 하나가 XML 스키마와 관련된 여러 문서(XML 스키마, XML 저장매체 스키마, DOM API 코드)들을 생성해 내는 XSD4j 문서 생성기(XSD4j Document Generator)이다.

XSD4j 모델러는 UML에 기반한 XML 스키마 모델링을 지원한다. 개발자가 설계한 XML 스키마는 XSD4j

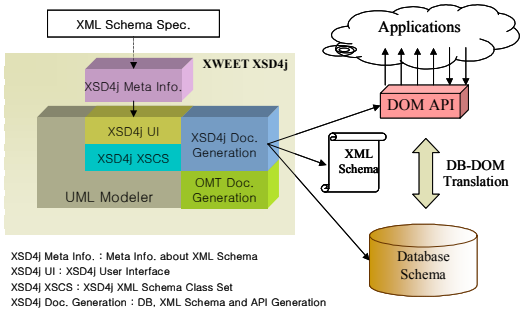

그림 9 XSD4j의 구조도

XSCS(XSD4j XML Schema Class Set)에 있는 XML 스키마 클래스의 리스트로 표현이 되며, 데이타 입력을 위한 인터페이스는 XSD4j UI(XSD4j User Interface)를 이용한다.

XSD4j XSCS는 XML 스키마에 정의된 여러 컴포넌트들을 UML의 스테레오 타입으로 변환한 클래스들의 집합을 말하며, XSD4j UI는 XSCS의 여러 클래스들에 필요한 데이타를 사용자로부터 입력받기 위한 인터페이스를 제공하게 된다. XSD4j UI는 XSCS의 각 클래스들의 애트리뷰트나 엘리먼트들을 입력받는다.

XSD4j UI가 사용하는 XSCS의 각 클래스에 대한 정보는 XSD4j 메타 정보 관리자(XSD4j Meta Information Manager)로부터 얻게 된다. XSD4j 메타 정보 관리자는 XML 스키마를 정의한 문서를 분석하여, XSD4j 모델러가 필요로 하는 여러 정보들을 제공한다.

XSD4j 문서생성기는 XSD4j 모델러에서 모델링된 결과에서부터 XML 스키마 문서, XML 저장매체 스키마, DOM API를 생성해 내는 역할을 수행한다.

4.1.2 동작 과정

XSD4j는 XML 스키마를 모델링한 후, 문서를 생성하는 두 단계로 이루어져있다. 사용자가 XML 스키마 다이어그램을 그리면, XSD4j UI는 다이어그램의 종류에 따라, 필요한 속성과 원소를 입력받는다. 입력받은 데이타를 이용하여 XSD4j XSCS에서 해당하는 클래스의 객체를 만든 후, UML Modeler의 객체리스트에 추가한다. XSCS의 클래스는 UML Modeler의 클래스를 스테레오타입으로 상속을 받은 것이므로, UML Modeler에서는 XSCS가 일반 UML 클래스로 처리가 된다.

이렇게 XML 스키마의 모델링이 끝나면, XSD4j 문서생성기는 XML 스키마 문서를 생성하게 된다. 문서생성기는 UML Modeler의 객체리스트를 순회하며, 객체의 타입에 맞게 XML 스키마 문서를 생성한다. XML 스키마와 XSCS의 클래스는 거의 일대일 대응이 되므로 간단히 XML 스키마가 생성될 수 있다.

XML 스키마를 생성한 후에는 XML 저장매체 스키마 생성과 DOM API 코드를 생성하게 된다. XSD4j는 XML 저장매체로 관계형 데이타베이스를 사용한다. XML 스키마와 관계형 데이타베이스 스키마는 일대일 대응이 되지 않으므로, 적당하게 변환해야 하는 작업이 필요하다. XSCS의 각 클래스는 관계형 데이타베이스 스키마 생성 룰을 포함하고 있으며, 이 룰에 의해 생성된 관계형 데이타베이스 스키마는 자료구조의 형태로 그 클래스에 저장되며, DOM API 생성 모듈에서 이 자료구조를 사용하게 된다.

DOM API 코드 생성은 자바 언어로 되어 있으며, 앞에서 언급한 XSCS의 관계형 데이타베이스 스키마 자료구조를 참고하여 코드가 생성이 된다.

이 단계까지 마치면, 사용자는 모델링한 XML 스키마에 맞는 자바 DOM API를 얻게 되며, 이 API는 응용 프로그램 작성의 기반이 된다.

4.2 XML 스키마와 UML 매핑

실제 예로써 어떤 쇼핑몰에서 사용할 수 있는 간단한 구매 정보 XML 문서를 보도록 하자. 이 쇼핑몰은 각 주문이 발생한 날짜에 대해 구매 정보를 관리하며, 한번의 구매에 대해서 구매자는 한명이어야 하며, 물품의 배송지는 구매자가 부재중일 때를 대비하여 두 곳을 지정할 수 있고, 한번의 구매를 통해 살 수 있는 제품의 종류는 무제한으로 한다는 정책을 가지고 있다고 하자.

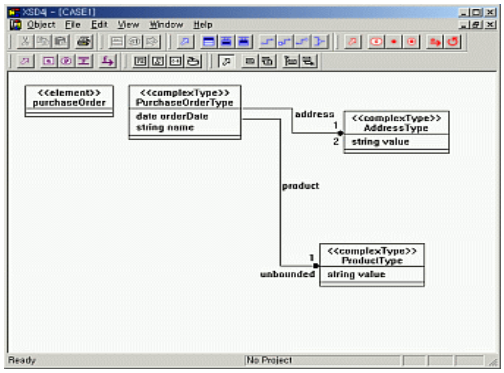

그림 10 PurchaserOrder 스키마

표 2 간단한 주문정보 관련 XML 스키마

```
<schema>
<element name="purchaseOrder" minOccurs="0"
maxOccurs="unbounded" type="PurchaseOrderType"/>
<complexType name="PurchaseOrderType">
      <attribute name="orderDate" type="date"/>
      <element name="name" type="string"/>
      <element name="address" ref="AddressType"
      maxOccurs="2"/>
      <element name="product" ref="ProductType"
      maxOccurs="unbounded"/>
</complexType>
<complexType name="AddressType">
      <attribute name="value" type="string"/>
</complexType>
<complexType name="ProductType">
      <attribute name="value" type="string"/>
</complexType>
</schema>
```

이 정책을 반영해서 설계한 XML 스키마는 그림 10과 같다. 즉, 한번의 구매에 대해 그 속성이 될 수 있는 구매 날짜를 애트리뷰트로 하고, 구매자의 이름과 배송지 주소, 구매 물품 등은 엘리먼트로 처리한다. 제품의 정보를 나타내는 ProductType이나 주소지에 관한 정보를 나타내는 AddressType등은 다른 요소에서 재활용할 수 있으므로 별도의 타입으로 지정하였으며, 구매정보타입인 PurchaseOrderType은 주소지와 제품 정보를 위해 각각 AddressType과 ProductType을 엘리먼트로 참조하게 된다. 이렇게 설계한 스키마를 XML 스키마로 생성하면 표 2와 같다.

4.3 XML 스키마와 관계형 데이타베이스 스키마

3.2.1절의 기본 공유 알고리즘에 따라 그림 10에서 설계한 XML 스키마를 RDB 스키마로 생성한 것이 표 3이다.

표 3 기본 공유 테크닉을 이용해서 생성한 RDB 스키마

```
CREATE TABLE XSE_ElementTable (
            _XSE_KEY_ int auto_increment primary key,
            parent int, child int, seq int default 0,
            TypeName varchar(255));
CREATE TABLE PurchaseOrderType (
            _XSE_KEY_ int auto_increment PRIMARY KEY,
            orderDate date );
CREATE TABLE AddressType (
            _XSE_KEY_ int auto_increment PRIMARY KEY,
            value varchar(255) );
CREATE TABLE ProductType (
            _XSE_KEY_ int auto_increment PRIMARY KEY,
            value varchar(255) );
CREATE TABLE string (
            _XSE_KEY_ int auto_increment primary key,
            data varchar(255));
```

엘리먼트의 부모/자식관계를 나타내는 테이블인 XSE_ElementTable에는 부모와 자식 노드의 키와, 자식의 키의 타입에 대한 정보, 엘리먼트의 순서에 대한 필드를 가지고 있다. 나머지 테이블은 엘리먼트의 타입에 대한 테이블이다.

3.2.2절의 공유 인라인 테크닉을 이용하여 그림 10의 XML 스키마를 RDB 스키마로 생성한 예는 표 4와 같다. 이 예를 보면, 생성된 테이블의 수가 기본 공유 알고리즘에 비해 적음을 알 수 있는데, 따라서 조인 연산이 현저히 줄어 들 것임을 예측할 수 있다. (별도의 조인 연산 없이도 address나 name 엘리먼트의 값을 얻어올 수 있다.)

표 4 인라인 공유 테크닉을 이용해서 생성한 RDB 스키마

```
CREATE TABLE PurchaseOrderType (
        _XSE_KEY_ int auto_increment PRIMARY KEY,
        parent int,
        seq int,
        orderDate date,
        name varchar(255),
        addressvalue1 varchar(255),
        addressvalue2 varchar(255) );
CREATE TABLE ProductType (
        _XSE_KEY_ int auto_icrement PRIMARY KEY,
        parent int,
        seq int,
        value varchar(255) );
```

4.4 XML 스키마와 자바 DOM API

4.4.1 자바 DOM API의 구조

XSD4j가 생성하는 자바 DOM API의 구조는 그림 11에서 보는바와 같이 개발자로부터 가능한 하부구조를 모두 숨기는 형태를 띄고 있다. 실제로 개발자는 DOM API 중에서도 *Document*, *Collection*, *User Defined XML Elements*가 인터페이스 역할을 수행하게 되며, 에러에 대한 예외처리(Exception Handling)는 *XSE Exception*에서 처리가 된다.

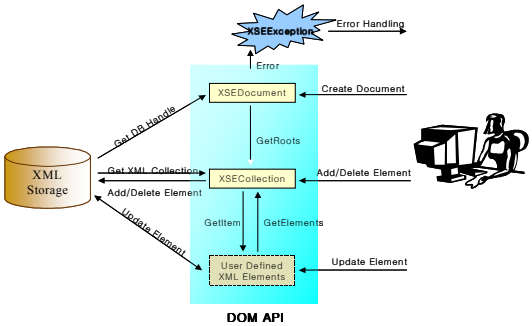

그림 11 자바 DOM API의 구조도

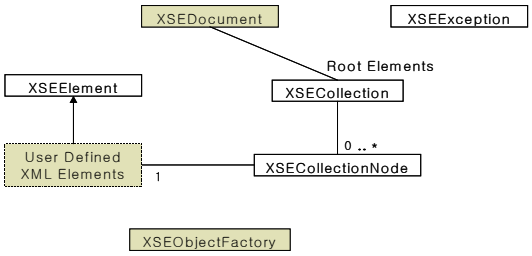

그림 12 XSD4j가 생성하는 자바 DOM API의 클래스 구조도

실제로 자바 DOM API는 그림 12와 같은 클래스 구조도를 갖는다. 그림에서 회색으로 표시된 클래스는 생성되는 XML 스키마에 종속적인 클래스이며, 그 외의 클래스는 모든 XML 스키마에 공통적으로 적용되는 클래스이다.

4.4.2 DOM 객체와 XML 저장매체와의 연동 전략

DOM 객체는 메모리 상에 존재하므로 객체의 생성/유지에 대한 부하가 작은 반면에, XML 저장매체는 영구적인 저장장치에 대한 작업을 필요로 하므로 그것에 대한 작업의 부하는 상당히 큰 편이다. 따라서, DOM 객체에 대한 작업(생성/수정/소멸)은 실제적인 XML 저장매체와는 독립적으로 이루어지게 해야 하며, 필요한 시점에서 작업 결과를 XML 저장매체에 반영하는 전략의 도입이 필수적이다.

XSD4j가 생성하는 자바 DOM API에서의 DOM 객체는 처음 생성될 때 XML 저장매체에서 데이타를 읽어오기 위한 기본적인 정보(키)만을 가지고 있는 불완전한 상태로 된다. 그 후 실제로 데이타가 필요로 하게 되는 시점에서 DOM 객체를 완성시키며, 객체에 대한 작업이 최종적으로 완료되는 시점에서 XML 저장매체에 그 결과를 반영시키는 전략을 취한다.

이를 위해 DOM 객체는 *DataReady*, *Dirty*, *Deleted*, *Temporary*의 4가지 상태 플래그를 가진다.

표 5 DOM 객체의 상태 플래그

상태 플래그	상 태
DataReady	DOM 객체가 완성 상태인지 아닌지를 나타냄
Dirty	DOM 객체의 내용이 수정이 됨
Deleted	현재 DOM 객체가 삭제됨
Temporary	현재 생성된 DOM 객체는 XML 저장매체에 존재하지 않는 임시 객체임

*DataReady*는 DOM 객체를 처음 얻어왔을 때는 거짓이다. 즉, DOM 객체가 불완전한 상태에 있음을 의미한다. 그 후, 객체에 대한 접근이 이루어질 때, *DataReady*가 거짓이면, XML 저장매체로부터 데이타를 읽어오고, *DataReady*는 참이 된다. 예외적으로 DOM 객체가 생성될 때 *DataReady*가 참이 되는 경우가 있는데, 이는 DOM 객체의 부모 객체의 데이타를 읽어올 때 현재 객체의 내용을 알 수 있는 경우에 해당한다. 예를 들어 이런 경우는 앞에서 언급한 공유 인라인 방식의 RDB 스키마 생성 테크닉을 사용할 경우 나타날 수 있다.

*Dirty*는 초기값은 거짓이다. 그리고 DOM 객체의 내용이 수정될 때 참이 된다. *Dirty*인 DOM 객체는 모든 작업이 완료된 후 XML 저장매체에 수정된 내용이 반영된다. 또한, 사용자가 명시적으로 DOM 객체를 XML

저장매체에 저장할 수도 있다.

*Deleted*는 초기값은 거짓이다. 그리고, DOM 객체를 사용자가 삭제할 때 참이 된다. DOM 객체는 Collection을 통해 집단적으로 삭제될 수도 있고, 객체 자신이 직접 자신을 삭제할 수도 있다. 삭제된 객체는 XML 저장매체에서 바로 삭제되지는 않으며, 모든 작업이 완료된 후 삭제된 결과가 XML 저장매체에 반영된다.

Temporary 상태는 새로운 엘리먼트를 추가할 때 사용된다. 즉, DOM 객체를 하나 생성하게 되면, 이 객체는 *Temporary* 상태에 있게 된다. 이 객체를 DOM 객체 Collection에 추가하면, *Temporary* 상태가 해제되며, 모든 작업이 완료된 후, 추가된 객체는 XML 저장매체에 추가된다.

4.4.3 자바 DOM API에서 XML 데이타 제약조건의 지원

XML 스키마로 정의된 문서는 앞 절에서 설명한 방식으로 XML 저장매체에 영구적으로 저장이 되며, 저장된 내용을 다시 XML의 구조로 읽어올 수 있다. 그러나 지금까지의 방법으로는 XML 문서의 구조만을 다룰 수 있을 뿐이다. 이 절에서는 XML 스키마에서 XML 문서의 유효성을 정의할 수 있는 XML 데이타 제약 조건을 어떻게 제공할 수 있는가에 대해서 살펴보도록 한다.

XML 스키마에서 데이타 제약 조건은 크게 두 가지로 나누어 볼 수 있다. 첫째는 타입에 대한 제약 조건이며, 둘째는 엘리먼트에 대한 제약 조건이다. 전자의 경우는 *simpleType*에 그 타입이 가질 수 있는 값의 범위에 대한 제약조건이 붙는 것이며, 후자는 *complexType*에서 엘리먼트가 나타날 수 있는 횟수 등을 제약하는 경우이다.

4.4.3.1 타입에 대한 제약 조건

타입에 대한 제약 조건은 크게 값의 크기에 대한 조건과, 문자열의 길이나 패턴에 대한 조건으로 나누어 볼 수 있다.

크기에 대한 조건은 일반적으로 숫자에 대해 적용이 되며, 부등호 연산자로 일대일 변환할 수 있다. 즉, *minExclusive*는 '>', *minInclusive*는 '>=', *maxExclusive*는 '<', *maxInclusive*는 '<='로 대응될 수 있다.

문자열에 대한 조건은 길이와 패턴이 있다. 길이는 문자열 길이 함수(자바에서는 *String.length()* 함수)를 통해 확인 가능하다. 패턴은 정규표현(Regular expression) 형태로 나타나지며, 따라서 별도의 정규표현 엔진을 제공하여 구현할 수 있다.

4.4.3.2 엘리먼트에 대한 제약 조건

엘리먼트에 대한 제약 조건은 엘리먼트가 몇 번 나타날 수 있는가를 나타내는 *maxOccurs*, *minOccurs*와 엘리먼트의 값의 특성에 대한 *fixed*, *nullable* 조건이 있다.

엘리먼트가 나타날 수 있는 최대값과 최소값을 지정하는 *maxOccurs*와 *minOccurs*는 엘리먼트의 Collection을 생성할 때와 새로운 엘리먼트를 Collection에 추가할 때 체크한다.

엘리먼트가 고정된 값을 가져야 함을 나타내는 *fixed* 속성과, 엘리먼트가 null값을 가질 수 있음을 나타내는 *nullable* 속성은 엘리먼트를 읽어올 때와 엘리먼트에 값을 지정할 때 체크한다.

5. XSD4j를 이용한 응용프로그램의 개발

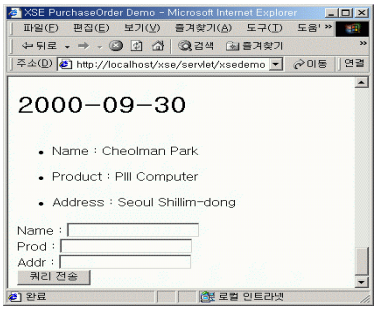

그림 13 간단한 구매정보관리 프로그램

앞에서 본 간단한 구매정보 관련 XML 스키마(그림 10)를 이용하여, 구매정보 목록을 출력하고 데이타를 추가하는 간단한 XML 응용 프로그램(그림 13)을 제작하는 과정을 살펴보도록 하자. 그림 10에서 설계한 스키마를 통해 표 3이나 표 4와 같은 관계형 데이타베이스를 위한 RDB 스키마와, 그림 11과 같은 자바 DOM API를 생성해 준다. 이를 통해 응용 프로그램 기반이 조성되었으며, 개발자는 이렇게 조성된 환경에서 간단히 DOM API를 사용함으로써 응용 프로그램을 개발 할 수 있다. 실제로 개발자가 작성한 코드는 표 6과 표 7에 나타나 있다.

표 6 XML 데이타를 읽어오는 모듈

```
try {
    // XSEDocument(DB Host, DB User, User Password, Database Name)
    m_Doc = new XSEDocument("localhost", "xse", "pwxsedemo", "xse");
    XSECollection elemCol = m_Doc.getRoots();
    int length = elemCol.getLength();
    for(int i = 0; i < length; ++i){
        XSEPurchaseOrderTypeElement elem =
            (XSEPurchaseOrderTypeElement)elemCol.getItem(i);
        out.println(display Purchase Order(elem));
    }
    m_Doc.preDestroy();
} catch(XSEException E) {
    out.println("<P>XSE Error1 : " + E.ErrorMsg() + "</P>\n");
}
```

표 7 XML 문서에 데이타를 추가하는 모듈

```
try {
      java.sql.Date date = new java.sql.Date(today.getYear
      (), today.getMonth(), today.getDay());
      XSECollection col = m_Doc.getRoots();
      XSEPurchaseOrderTypeElement elem = new
                  XSEPurchaseOrderTypeElement();
      elem.setorderDate(date);
      col.add(0, elem);
      col.UpdateAll();
      XSECollection products = elem.getproducts();
      XSEProductTypeElement prod = new
                  XSEProductTypeElement();
      prod.setvalue(product);
      products.add(0, prod);
      products.UpdateAll();
} catch(XSEException E){
      out.println("XSE Error3 : " + E.ErrorMsg());
}
```

6. 결론 및 향후 연구 계획

XML은 데이타를 표현하는 표준적인 언어라는 측면에서 그 중요성이 점점 증대하고 있다. 그에 따라 XML 문서를 이용한 응용프로그램의 개발이 늘어나고 있는 추세이다.

본 논문에서는 XML 스키마의 설계 단계에서 UML을 도입하여, 논리적이고 개념적인 수준의 스키마를 설계할 수 있는 환경을 제공할 수 있음을 보였다.

그리고, XML CASE 툴에 XML 스키마를 도입함으로써 XML 응용 프로그램의 개발에 CASE 툴이 어떤 면으로 기여할 수 있는지를 살펴보았다. 기존의 XML 응용 프로그램의 개발 단계를 보면, 먼저 스키마를 설계하고, 이 스키마를 위한 저장매체에 대한 스키마를 생성해야 하며, 저장매체와 응용프로그램과의 API를 구현, 비로소 XML 응용 프로그램 개발을 시작할 수 있었다. 그러나 본 논문에서는 XML 스키마를 도입하고, XML 스키마에서 XML 저장매체 스키마와 DOM API를 자동적으로 생성해 줌으로써, 개발자에게 “스키마를 디자인하고 XML 응용 프로그램을 개발”하는 두 단계의 개발 프로세스를 지원할 수 있음을 보였다.

본 논문의 본론에서는 실제로 XML 저장매체로 관계형 데이타베이스시스템을, DOM API를 위한 언어로 자바를 선택하여 실제로 시스템이 어떻게 구현되어 질 수 있는가를 살펴보았다. 또한, XML 스키마의 RDB 스키마 변환 과정에서는 [13]가 제안한 기본 공유 테크닉이나 공유 인라인 테크닉이 XML 스키마에 어떻게 도입될 수 있는가를 살펴보았으며, RDB를 위한 자바 DOM API 코드의 생성에서는, RDB 데이타로 변환된 XML 문서를 다시 복구하는 방법과, 사용자에게 제공되어지는 인터페이스를 제시하였다. 또한, XML 스키마의 데이타 제약조건을 DOM API에서 어떻게 지원할 수 있는지 살펴보았다.

향후 본 논문의 결과로 구현된 XSD4j에서, 현재보다 더 효율적인 RDB 스키마를 생성할 수 있는 방법을 제공하고, DOM API에서 XML 문서를 위한 효율적인 캐슁과 패칭 전략을 제공하도록 확장할 것이다. 또한, XSD4j를 이용하여 모델링된 XML 스키마의 스키마 진화에 대한 연구도 계속 할 것이다.

참 고 문 헌

[1] XML Specification, http://www.w3.org/XML/

[2] 조형주, 정진완, 김형주, “UML 클래스 다이어그램 기반의 효율적인 C++ 코드 생성기의 설계와 구현”, 정보과학회논문지 : 컴퓨팅의 실제, 제 6권 제 4호, 2000년 8월

[3] Grady Booch, James Rumbaugh, Ivar Jacobson, "The Complete UML Training Course," Addison-Wesley Pub. Co.

[4] Document Object Model, http://www.w3.org/DOM/

[5] SAX Specification, http://www.megginson.com/SAX/index.html

[6] XML Schema Specification, http://www.w3.org/XML/Schema.html

[7] ebXML Official homepage, http://www.ebxml.org

[8] Grady Booch, Magnus Christerson, Matthew Fuchs and Jari Koistinen, "UML for XML Schema Mapping Specification," http://www.rational.com

[9] "Foundation package: Extension mechanisms Overview," http://www.rational.com

[10] XML Authority, http://www.xmlauthority.com

[11] XML Spy, http://www.xmlspy.com

[12] Alin Deutsch, Mary f. Fernandez, Dan Suciu, "Storing Semistructured Data with STORED," SIGMOD 1999.

[13] Jayavel Shanmugasundaram, et. al., Relational Database for Querying XML Document: Limitations and Opportunities, Proceedings of the 25th VLDB Conference.

[14] Oracle 8i, http://www.oracle.com/ip/deploy/database/8i/index.html

[15] 이제민, 민경섭, 박상원, 김형주, "관계형 데이타베이스 시스템에서의 XML 데이타의 지원", 서울대학교 공학석사 논문, 2000년 2월.

[16] Excelon, http://www.exceloncorp.com

[17] 고봉수, 박상원, 민경섭, 김형주, “XML을 ODMG표준을 따르는 객체지향 데이타베이스에 사상하는 시스템의 설계 및 구현”, 서울대학교 공학석사 논문, 2000년

2월.
[18] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, "Lore: A Database Management System for Semistructured Data," SIGMOD Record, September 1997.

박 철 만
1998년 경북대학교 컴퓨터공학과(학사). 2001년 서울대학교 컴퓨터공학부(석사). 2001년 ~ 현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 데이타베이스, XML, 소프트웨어공학, 지식관리

박 상 원
정보과학회논문지 : 컴퓨팅의 실제 제 7 권 제 3 호 참조

김 형 주
정보과학회논문지 : 컴퓨팅의 실제 제 7 권 제 1 호 참조