XML DTD의 관계형 데이터베이스 스키마로의 자동변환을 위한 컴포넌트 모델링

Modeling Components for mapping from XML DTD to RDB schema

이 정 수* 방 승 윤** 주 경 수***
Jung-Soo Lee Sung-Yoon Bang Kyung-Soo Joo

요 약

XML은 웹에서 구조화된 정보나 반-구조화된 정보를 교환하고 저장하는데 있어서 하나의 표준 마크업 언어로 채택되고 있는 추세이다. 웹을 통한 정보교환에 있어서, XML 메시지의 소스 데이터는 Legacy 데이터베이스에 저장되어 있기 때문에 XML 응용과 데이터베이스 시스템과의 원활한 연계가 요구되어진다. 이를 위하여 Oracle8i와 9i 및 Informix 그리고 SQL2000과 같은 DBMS들은 XML을 취급하기 위하여 자신의 DBMS들을 확장하고 있다. 그러나 이러한 방법은 플랫폼-종속적이며 DBMS-종속적이다. 따라서 원활한 XML 응용과 데이터베이스 시스템과의 연계를 위해서는 플랫폼-독립적이며 DBMS-독립적인 연계방안이 요구된다.

이에 따라 본 논문에서는 XML 응용과 기존의 데이터베이스 시스템과의 원활한 연계를 위하여, XML DTD를 토대로 관계형 데이터베이스 스키마를 설계하기 위한 미들웨어 컴포넌트를 설계하였다.

Abstract

XML is a standard markup language for exchange and storage of formed or well-formed information in World Wide Web. Because the source data of XML message for exchange of information in World Wide Web is stored in legacy database systems, it is necessary the easier connection between XML application and database systems. In Oracle8i, 9i, Informix and SQL2000, DBMS vendors extend their DBMSs for XML. This approach for the connection between XML application and database system is Platform-dependent and DBMS-dependent. Therefore it is necessary the platform- and DBMS-independent approach for the connection between XML application and database system.

In this paper, we modeled a set of middleware components for relational database design based on XML DTD is modeled. Those components are modeled on object-based algorithm for connection between XML application and database system.

1. 서 론

HTML 문서는 고정된 태그로 되어 있다. 그렇기 때문에 만일 사용자가 임의로 태그를 정의하여 사용한다면 웹 브라우저는 잘못된 태그로 인식할 것이다. 반면에 XML은 사용자가 태그를 정의하여 사용할 수 있는 마크업 언어이다. 따라서 XML은 사용자가 정의한 태그로 인해 인터넷상에서 구조화된 문서로 표현될 수 있으므로 단순한 마크업 언어의 범주를 넘어서 사용자의 다양한 요구를 수용할 수 있는 언어로 발전하였다. 특히 전자 상거래 분야에서 XML을 이용하여 이루어지는 문서 교환이 많아지고 있는 실정이다.

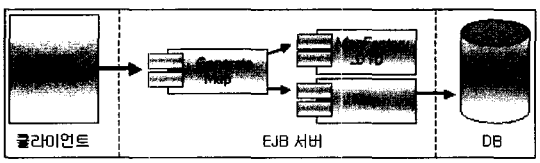
또한 웹에서 정보교환을 위한 XML 메시지의 소스 데이터는 Legacy 데이터베이스에 저장되어 있기 때문에, 이에 따라 XML 응용과 데이터베이스 시스템과의 원활한 연계가 요구되어진다. 따라서 현재 XML과 데이터베이스는 핵심 기술로 성장하고 있으며, eXcelon과 tamino 같은 XML 전용

* 준 회 원 : 순천향대학교 전산학과 석사과정
jungsoo6@hitel.net
** 정 회 원 : 순천향대학교 전산학과 박사과정
sybang@hanseo.ac.kr
*** 종신회원 : 순천향대학교 정보기술공학부 교수
gsoojoo@asan.sch.ac.kr

DBMS는 XML 문서를 자동적으로 저장하고 검색을 할 수 있도록 되어 있다. 하지만 기존의 자료를 활용하여야 하는 상황에서 XML 문서를 기존의 데이터베이스에 저장하고 검색할 수 있도록 해야 하는 필요성이 요구되고 있다. 이러한 필요성에도 불구하고 기존에 사용하고 있는 RDBMS에서는 저장 및 검색이 불가능하다 이것은 XML 구조와 관계형 데이터베이스 구조가 서로 다르기 때문이다 이러한 문제를 해결하기 위하여 XML 구조를 관계형 데이터베이스 구조로 변환해 주는 미들웨어가 필요하다고 하겠다[1,4,5].

따라서 XML DTD를 관계형 데이터베이스 스키마로 변환하기 위한 미들웨어 컴포넌트를 설계하였고, XML 문서를 RDBMS에 저장하고 검색할 수 있도록 한다.

본 논문의 제2절에서는 관련 연구로써 CBD (Component Based Development)와 EJB (Enterprise JavaBeans)에 대하여 서술하고, 제3절에서는 XML DTD를 관계형 데이터베이스 스키마로 변환하는 알고리즘을 기술하며, 제4절에서는 XML DTD를 관계형 데이터베이스 스키마로 변환하는 모델링을 기술하고, 마지막으로 결론을 기술한다.

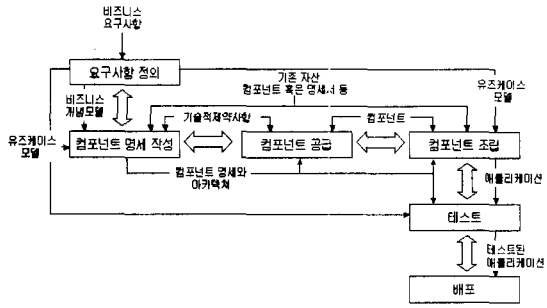


(그림 1) 시스템 구조

2. 관련 연구

2.1 CBD

CBD 기술은 소프트웨어 위기 극복으로부터 시작되었으며 소프트웨어 공학 기법으로써, 보다 높은 생산성과 고품질의 소프트웨어 개발을 목표로 한다. 즉, 생산성을 향상시키기 위해 컴포넌트를 만들고 재 사용할 수 있도록 하여 마치 부품을 조립하듯이 소프트웨어를 쉽게 개발할 수 있게 해준다.



(그림 2) 전체 개발 프로세스의 워크플로우

또한 컴포넌트가 제대로 동작하기 위해서 개발자는 컴포넌트의 행위를 이해할 필요가 있으며, 이런 컴포넌트가 어떤 서비스를 제공해야 하고, 이러한 서비스가 어떻게 동작하게 될지를 묘사해야 한다. 따라서 컴포넌트는 그의 행위와 관련한 명확한 명세서를 요구하게 되며 이 명세서는 구현과 독립적이기 때문에 다양한 환경과 다른 시스템에서도 적용할 수 있다.

2.1.1 요구 사항

요구분석에서 시스템의 요구사항을 이해하고 기술하기 위해 비즈니스 타입 모델과 유즈케이스를 기술한다. 비즈니스 타입 모델은 비즈니스 프로세스 기술에 명확히 정의해야 할 필요가 있는 업무 용어를 산출하는 것이고, 유즈케이스는 시스템의 동적인 면을 모델링하기 위한 것으로 비즈니스 타입 모델에 대한 책임을 단계적으로 할당해 주는 초기작업이다[6].

2.1.2 분석 단계

분석과정에서 컴포넌트 아키텍쳐 설계와 인터페이스 상호작용에 대한 모델링을 통하여 인터페이스들을 도출한다. 컴포넌트 개발에 있어서 중요한 작업은 바로 적절한 인터페이스를 정의하는 것이다[9]. 즉, 인터페이스는 개발자들 간의 약속과

같다. 그래서 약속이 지켜지지 않는다면 컴포넌트를 재사용하는 것은 불가능하다. 따라서 컴포넌트 개발의 핵심은 충분히 재사용이 가능하고 변화에 영향을 적게 받는 인터페이스를 도출하는 것이다.

2.1.3 설계 단계

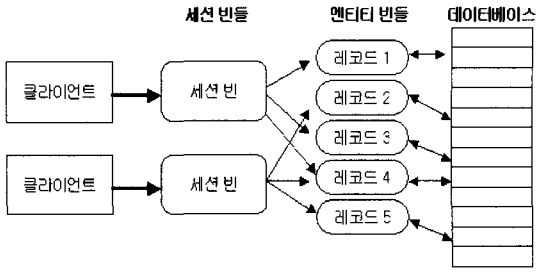
설계과정에서 인터페이스를 정의하기 위하여 다이어그램을 작성한다. 컴포넌트 아키텍처와 상호작용이 끝나면, 인터페이스 다이어그램을 작성하는 것은 단지 양식을 바꾸어 옮겨 적은 것과 같다. 하지만, 실제로 인터페이스 다이어그램을 작성하다 보면, 정의된 아키텍처와 상호작용 모델링에서 미비한 점들이 발견되므로 컴포넌트가 지켜야 하는 구조적 불변식을 기술할 필요가 있다. 불변식이란 컴포넌트의 상태가 항시 만족시켜야 할 조건을 기술하는 것을 말한다[9].

2.2 EJB

J2EE(The Java 2 Platform, Enterprise Edition)는 다계층 엔터프라이즈 애플리케이션을 개발하기 위한 표준을 말한다. 즉 J2EE는 표준화되고 모듈화 된 컴포넌트상에 다 계층 엔터프라이즈 애플리케이션을 기본으로 하고, 해당 컴포넌트에 완벽한 서비스를 제공하며, 복잡한 프로그래밍 없이 자동으로 애플리케이션을 처리함으로써, 엔터프라이즈 애플리케이션을 단순하게 만든다[8,9].

J2EE는 단순성, 이동성, 확장성, 통합성 등의 특징을 지원하는 엔터프라이즈 솔루션을 위한 플랫폼이다. 또한 J2EE는 JSP, Servlet, Java Bean, EJB 등의 집합이라고 말하며, 핵심적인 기술로는 EJB라고 말할 수 있다.

EJB는 컴포넌트 기반 분산 객체 기술로서 엔터프라이즈급 애플리케이션 개발에 있어 추상 데이터와 비즈니스 로직에 대한 부분을 담당하는 매우 중요한 핵심 기술을 가지고 있다. 또한 개발자를 도와주는 EJB 컨테이너는 자동으로 엔터프라이즈



(그림 3) 세션 빈과 엔티티 빈의 관계

빈의 생명주기 관리, 상태 정보관리, 보안, 트랜잭션 처리, 영속성 처리 등을 포함한 수많은 내재된 서비스를 제공해 주기 때문에 개발자는 비즈니스 로직만 담당하면 된다[9].

엔터프라이즈 빈(Enterprise Bean)이란 클라이언트가 호출하여 사용할 수 있는 EJB 컴포넌트를 의미한다. 이런 엔터프라이즈 빈은 엔티티 빈(Entity Bean)과 세션 빈(Session Bean)으로 구분되어 개발할 수 있다. 그림 3은 세션 빈과 엔티티 빈의 관계를 보여준다.

3. XML DTD를 관계형 데이터베이스 스키마로 변환하는 알고리즘

XML 문서의 구조를 정의한 DTD를 관계형 데이터베이스 스키마로 변환하기 위하여 그림 4에서 보여주는 것과 같이 알고리즘이 필요하다. 이는 XML 문서를 저장하거나 검색할 때 변환해 주는 미들웨어를 통하여 진행된다.

본 논문에서 이용할 수 있는 변환 알고리즘은 객체를 이용한 객체 기반 알고리즘이다. 이 알고리즘은 복잡한 XML 문서를 데이터베이스에 저장하려고 할 때 사용하는 알고리즘이며, DTD를 객체로 변환하고, 변환한 객체를 관계형 데이터베이스 스키마로 변환한다[5].

(그림 4) 객체 기반 알고리즘

3.1 XML DTD를 객체로 변환

3.1.1 요소(Element)

요소는 XML 문서의 기본이 되는 논리적 단위이므로 XML 문서의 모든 내용물은 반드시 요소 안에 포함되어야 하고, HTML과 마찬가지로 시작 태그(begin-tag)부터 그에 상응하는 끝 태그(end-tag)까지의 영역으로 되어 있다[7].

영역을 가지고 있는 요소들은 객체로 변환할 때 클래스 이름으로 변환이 되고, 영역을 가지고 있지 않은 요소들은 클래스 속성으로 변환된다. 즉, PCDATA형으로 선언된 요소들이 클래스 속성으로 변환된다. 또한 서브 요소는 별도의 클래스 이름과 그 서브 요소 영역에 속한 요소들은 클래스 속성으로 변환한다. 요소와 서브 요소의 관계는 클래스의 참조형으로 지정된다[5].

3.1.2 특성(Attribute)

특성은 요소와 관련된 이름과 값의 쌍으로 이루어지고, 특성값은 요소의 시작 태그에서 결정될 수 있으며 디폴트값은 DTD로부터 상속받을 수 있다.

특성은 요소와 마찬가지로 단일 값을 가진 특성과 다중 값을 가진 특성으로 분류한다. 단일 값을 가진 특성은 문자열(CDATA), 토큰 특성(ID, IDREF, NMTOKEN, ENTITY, NOTATION), 열거형 특성으로 선언되며, 객체로 변환될 때 클래스 속성으로 변환된다. 다중 값을 가진 특성들은 IDREFS, NMTOKENS, ENTITIES으로 선언되어 있어, 특성 값이 다중으로 존재하므로 객체로 만들 때 다중 값을 가지고 있는 요소는 클래스 이름으로 변환하고, 다중 값을 가진 특성들은 클래스 속성으로 변환된다[5].

3.1.3 복합 내용 모델

(1) 순차

DTD에 있는 요소와 요소에 속하는 자식 요소들이 순차적으로 변환되면, 자식 클래스와 부모 클래스간의 연결은 참조형으로 변환될 수 있다.

(2) 선택

요소 안에 자식 요소들이 있는데, 그 중에 하나만 선택할 경우에도 객체로 변환된다.

(3) 반복

요소 안에 중복된 같은 요소들이 클래스 속성으로 변환하는 경우 배열 형태로 변환된다.

(4) 서브그룹

<!ELEMENT A(B, (C | D))>와 같은 경우에 다시 <!ELEMENT A(B, C)>와 <!ELEMENT A(B, D)>으로 분류될 수 있다. 따라서 <!ELEMENT A(B, C)>의 경우 클래스 이름이 A 요소이고 클래스 속성은 B 요소와 C 요소이다.

3.1.4 혼합 내용 모델

텍스트와 요소가 모두 포함된 경우 혼합 내용 모델이라고 한다. XML 프로세서는 공백, 탭 등의 PCDATA와 요소 내용간의 구분이 어려우며, 끝 태그와 다음의 시작 태그 사이에 공백이 있으면 불분명하게 된다[7]. 혼합 내용 모델은 선택이 가능한 그룹이 하나이고, #PCDATA로 시작하며, 그 뒤에는 혼합 내용의 연산자수를 나타내는 타입 순이어야 하며 각각은 한번만 선언되어야 한다. #PCDATA만 유일한 옵션이며, '*'는 반드시 괄호를 닫은 바로 뒤에 와야 한다[5].

3.2 객체를 관계형 데이터베이스로 변환

요소들을 객체로 변환하고, 변환된 객체를 관계형 데이터베이스 스키마로 변환한다. 일반적으로 클래스 이름은 테이블 이름으로 변환이 되고 클래스의 속성은 테이블 컬럼으로 변환할 수 있다[5].

3.2.1 데이터 유형

일반적으로 객체 지향 언어에서 변수의 데이터 유형은 정수형, 실수형, 문자형, 논리형 등과 같은

데이터 유형들이 있으며, 관계형 데이터베이스에서 데이터 유형은 데이터를 저장할 컬럼을 그 데이터 유형에 맞게 설정해 주어야 한다. 이렇게 객체 지향 언어와 관계형 데이터베이스간의 데이터 유형이 불일치하게 이루어지고 있다. 따라서 본 논문에서는 객체 지향 언어와 관계형 데이터베이스간에 데이터 유형은 사용자가 직접 설정을 해 주어야 한다. 예를 들면, 객체 지향 언어에서 데이터 유형이 문자열이면 관계형 데이터베이스에서는 데이터 유형이 CHAR형이나 VARCHAR형으로 설정해 주어야 한다[5].

3.2.2 제약 조건

관계형 데이터베이스 스키마로 변환할 때 제약조건을 설정해 주어야 하는데, 컬럼에 값이 반드시 입력되어야 할 때 NOT NULL이라고 지정하여야 하고 그렇지 못할 때는 NULL로 지정하여야 한다. 제약조건을 지정할 때는 DTD에 선언된 요소 연산자를 보고 그 요소가 아무런 연산자(+,*)가 없으면 NOT NULL로 지정해 주고 연산자 기호가 있으면 NULL로 지정해 주면 된다[5].

3.2.3 기본키/외래키 지정

DTD에서 요소에 속하는 서브 요소가 존재할 경우 부모 클래스와 하나 이상의 자식 클래스가 추출된다. 이렇게 추출된 부모 클래스와 자식 클래스간의 관계는 참조형을 이용하여 부모 클래스와 자식 클래스를 연결할 수 있다. 한 부모 클래스와 자식 클래스간의 관계를 관계형 데이터베이스에서는 기본키와 외래키를 이용하여 관계를 맺는다[5].

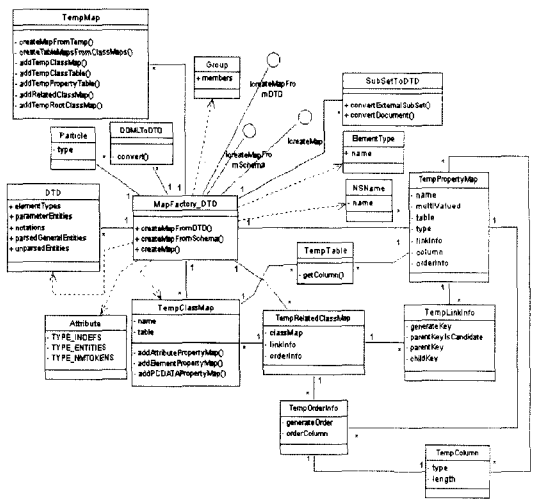
4. XML DTD를 관계형 데이터베이스 스키마로 변환 모델링

XML DTD를 관계형 데이터베이스 스키마로 자동 변환하는 컴포넌트들을 모델링하여, 실행 가능한 환경에서 실행한다.

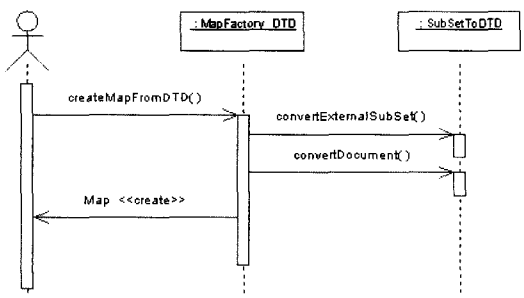
4.1 MapFactory_DTD 컴포넌트

MapFactory_DTD는 DTD를 입력받아 Map 객체를 생성시키는 컴포넌트이다. 이 컴포넌트는 DTD 뿐만 아니라 XML 문서를 입력받아 Map 객체를 생성할 수도 있다.

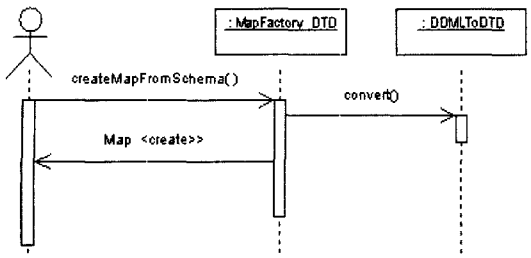
그림 5에서 MapFactory_DTD 컴포넌트 협력 다이어그램은 내부적으로 컴포넌트를 구현하기 위해 필요로 하는 클래스들을 관계를 보여주고 있다. 여기서 실선은 연관(association)관계이고 점선 화살표는 의존(dependency)관계이다. 인터페이스는 원형과 실선으로 되어있어 그림 5에서 제시된 인터페이스는 모두 3개로 이루어졌다.



(그림 5) MapFactory_DTD 컴포넌트 협력 다이어그램



(그림 6) createMapFromDTD() 인터페이스 순차 다이어그램

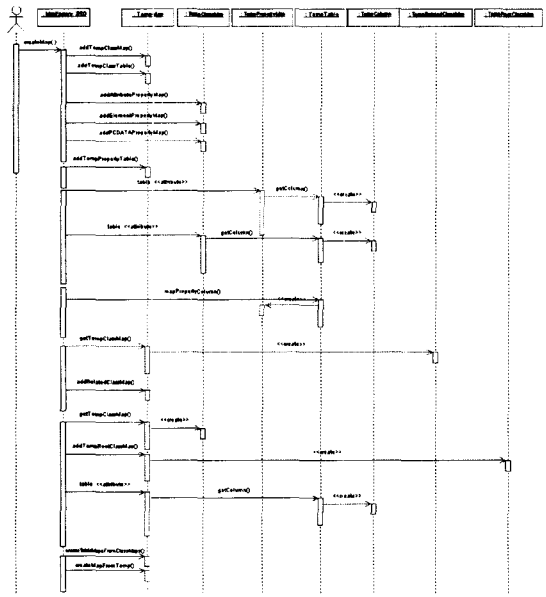


(그림 7) createMapFromSchema() 인터페이스 순차 다이어그램

그림 6에 정의한 createMapFromDTD() 인터페이스는 DTD이나 XML 문서를 입력받아 Map 형식으로 변환하여 그림 8에 정의된 createMap() 인터페이스로 리턴된다.

그림 7에서 정의한 createMapFromSchema() 인터페이스는 DDML문서를 DTD문서로 변환하여 그림 8에 정의된 createMap() 인터페이스로 리턴된다.

그림 8은 createMap() 인터페이스 순차 다이어그램을 보여주며, 가장 중요한 createMap() 인터페이스는 그림 6과 그림 7에서 변환한 DTD를 Map 객체로 생성시켜 주는 인터페이스이다.

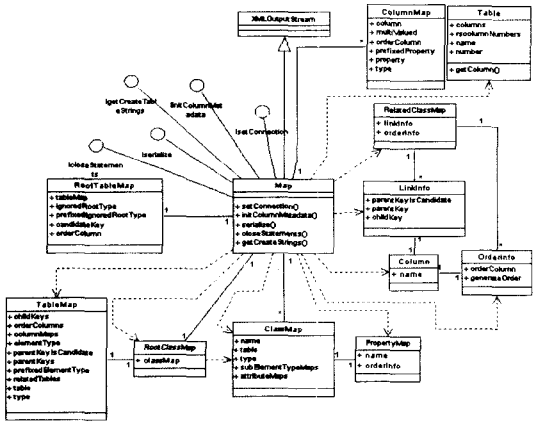


(그림 8) createMap() 인터페이스 순차 다이어그램

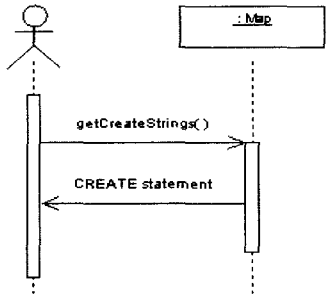
4.2 Map 컴포넌트

XML 문서와 관계형 데이터베이스 테이블 사이에 존재하는 관계를 기술하기 위해 XML 형태인 map 문서가 필요하다. 이런 map 문서는 XML 문서의 요소, 특성 등의 정의와 그들에 대응하는 테이블 컬럼들을 정의한 문서이다. 이런 map 문서는 MapFactory_DTD에서 생성된 Map 객체를 입력받아서 Map 컴포넌트에 의해 자동으로 map 문서와 관계형 데이터베이스 스키마를 기술한다.

그림 9이 보여주는 Map 컴포넌트 협력 다이어그램은 Map 객체를 map 문서와 관계형 데이터베이스 스키마로 표현할 때 사용되는 컴포넌트이다. 이런 Map 컴포넌트에서는 5개의 인터페이스가 추출된다.



(그림 9) Map 컴포넌트 협력 다이어그램



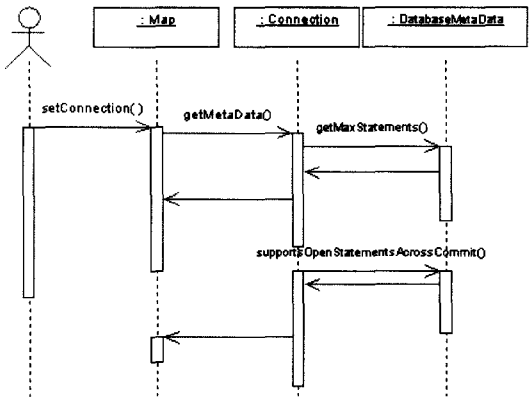
(그림 10) getCreateStrings() 인터페이스 순차 다이어그램

그림 10은 SQL CREATE문을 출력하는 인터페이스로서 MapFactory_DTD 컴포넌트에서 정의한 대로 관계형 데이터베이스에 필요한 부분을 생성하여 Map 컴포넌트에서 정의된 getCreateStrings()를 이용한다.

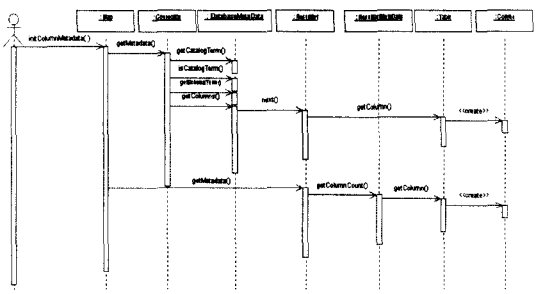
그림 11에서 정의한 getConnection() 인터페이스는 데이터베이스에 대한 메타데이터 정보를 얻는 인터페이스이다.

그림 12에 정의된 initColumnMetadata() 인터페이스는 현재 접속하기 위한 컬럼 메타데이터를 초기화하기 위한 인터페이스이다. 컬럼 메타데이터는 데이터를 변환할 때나 SQL 문을 생성할 때 양쪽으로 사용된다.

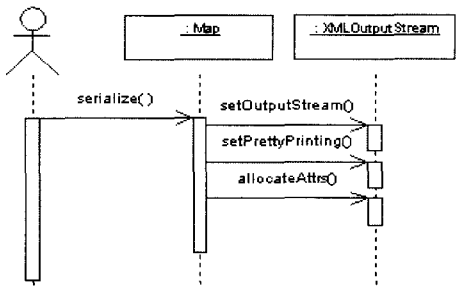
그림 13에서 정의한 serialize() 인터페이스는 MapFactory_DTD 컴포넌트에서 Map 객체가 생성하여, map 문서를 기술하기 위한 인터페이스이다.



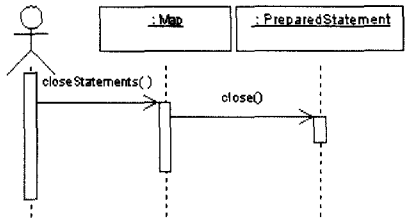
(그림 11) getConnection() 인터페이스 순차 다이어그램



(그림 12) initColumnMetadata() 인터페이스 순차 다이어그램



(그림 13) serialize() 인터페이스 순차 다이어그램

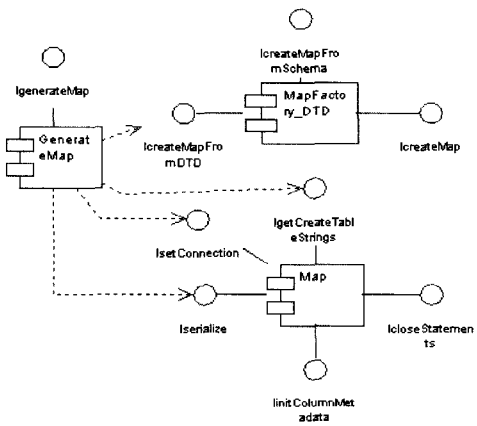


(그림 14) closeStatements() 인터페이스 순차 다이어그램

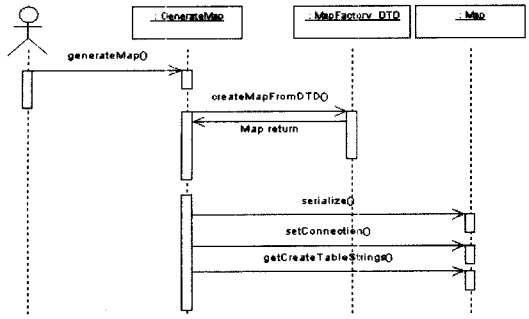
그림 14에서 정의한 closeStatements() 인터페이스는 데이터베이스를 닫기 위한 인터페이스이다.

4.3 GenerateMap 컴포넌트

GenerateMap은 MapFactory_DTD 컴포넌트와 Map 컴포넌트를 이용하여 Map 문서와 SQL 스키마를 추출한 컴포넌트이고, EJB에서 Session 빈으로 구현한다.



(그림 15) 컴포넌트 다이어그램



(그림 16) 컴포넌트 순차 다이어그램

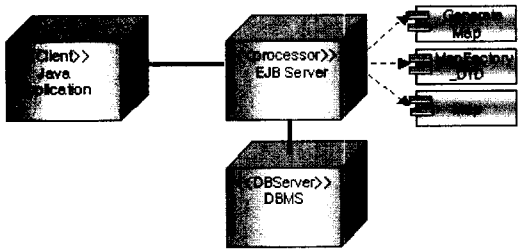
그림 15에 있는 GenerateMap 컴포넌트는 두 개의 컴포넌트(Map, MapFactory_DTD)를 사용하여 Map 문서와 관계형 데이터베이스 스키마를 생성해주는 컴포넌트이다. 이 컴포넌트의 인터페이스인 IgenerateMap은 DTD 문서를 입력받아 Map 문서와 관계형 데이터베이스 스키마를 생성시키는 역할을 한다.

그림 16에서 정의한 인터페이스는 클라이언트에서 DTD를 입력받아 3개의 컴포넌트간에 상호작용을 한다.

4.4 시스템 구조

본 논문에 제시된 컴포넌트들에 따른 시스템구조는 그림 17과 같다.

클라이언트에서 DTD를 입력하며 GenerateMap이 입력받아 MapFactory_DTD 컴포넌트와 Map 컴포넌트를 사용하여 Map 문서와 관계형 데이터베이스 스키마가 출력된다. 그림 17에서 3개의 컴포넌트는 EJB Session Bean으로 구현한다.



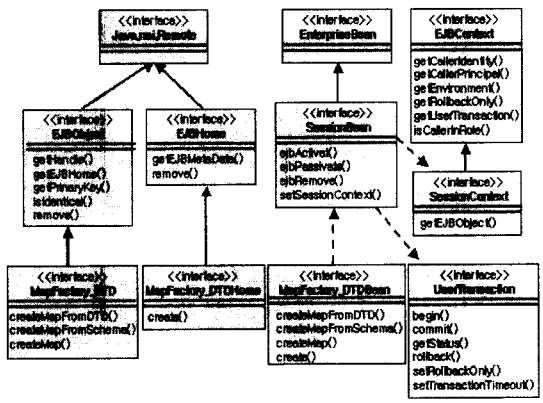
(그림 17) 시스템 구조

4.5 빈 클래스 다이어그램

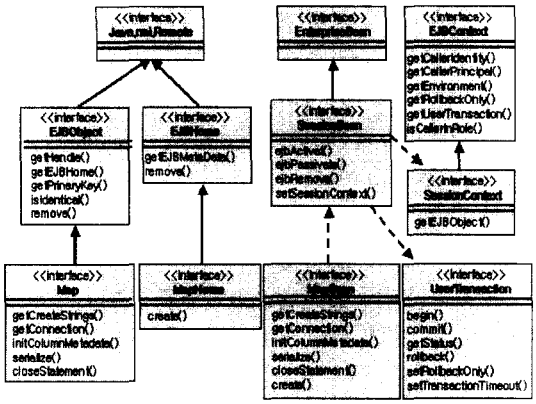
먼저 MapFactory_DTDBean는 Map 객체를 생성하는 컴포넌트이고, 이를 바탕으로 생성된 Map 객체를 MapBean에서 입력받아 map 문서형식으로 기술하는 컴포넌트이다. GenerateMapBean은 두 개의 빈을 조립한 컴포넌트로 클라이언트에서 하나의 인터페이스를 통하여 DTD를 입력받아 MapFactory_DTDBean으로 리턴 해 준다.

그림 18은 MapFactory_DTD 컴포넌트를 EJB 세션 빈으로 구현하기 위한 클래스 다이어그램이다. 이 다이어그램에서 실선 화살표는 클래스의 "상속(extends)"을 의미하고 점선 화살표는 인터페이스의 "구현(Implement)"을 의미한다. EJB 에서는 EJB 컨테이너가 트랜잭션, 보안, 통신 등을 자동으로 관리해 주기 때문에 MapFactory_DTDBean에서는 실질적으로 비즈니스 로직만 정의할 수 있다. 여기서 4개의 인터페이스를 정의했는데, create() 인터페이스는 Home 인터페이스로 정의되어 있고 나머지 3개의 인터페이스는 비즈니스 부분이므로 Remote 인터페이스로 정의한다.

그림 19는 Map 컴포넌트를 EJB 세션 빈으로 구현하기 위한 클래스 다이어그램이다. Home 인터페이스는 빈들을 컨테이너 안에서 생성하고 소멸시키고, 그리고 찾는데 필요한 메소드를 가진 인터페이스이고, 구현한 것은 EJBHome이다. Remote



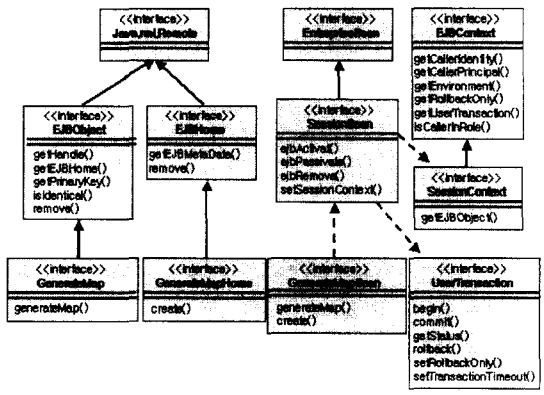
(그림 18) MapFactory_DTD 빈 클래스 다이어그램



(그림 19) Map 빈 클래스 다이어그램

인터페이스는 빈 클래스들이 제공하는 비즈니스 메소드들을 정의하고, 이것들은 Bean 클래스가 직접 구현하지 않고 클라이언트 호출을 Bean 객체에 중개하는 EJBObject 클래스가 구현한다.

그림 20은 GenerateMap 컴포넌트를 EJB 세션 빈으로 구현하기 위한 클래스 다이어그램이다. GenerateMapBean 인터페이스를 구현한 것을 SessionBean이다. 여기서 SessionBean에 여러 개의 메소드들이 있는데, ejbActivate()는 컨테이너가 SessionBean에게 이제 곧 활성화 될 것이라고 알려주기 위해 사용한다. ejbPassivate()는 ejbActivate() 반대의 의미로 곧 비활성화될 것이라고 알려주기 위해 사용한다. 여기서 활성화와 비활성화는 메모리에서 복구와 퇴출의 의미한다. ejbRemote()는 컨테이너



(그림 20) Generate 빈 클래스 다이어그램

가 SessionBean에게 이제 곧 컨테이너가 종료되니 Bean이 가지고 있는 모든 자원들을 내놓으라고 알려주기 위해 사용한다. setSessionContext()는 나중에 참조하기 위해 Bean의 세션 문맥에 대한 참조를 전달한다. 세션 문맥은 컨테이너가 유지하고 있는 인스턴스 문맥에 대한 접근을 제공한다[9].

5. 결 론

XML은 현재 웹에서 구조화된 정보나 반-구조화된 정보를 교환하기 위한 표준 마크업 언어로 채택되어 가고 있다. 한편 웹에서 정보교환을 위한 XML 메시지의 소스 데이터는 Legacy 데이터베이스에 저장되어 있으며, 이에 따라 XML 응용과 데이터베이스 시스템과의 원활한 연계가 요구되고 있다. 이에 따라 Oracle8i와 9i 및 Informix 그리고 SQL2000 등과 같은 DBMS 공급자들은 XML을 다루기 위하여 자신의 DBMS을 확장하고 있다. 이렇게 확장된 DBMS들은 XML 문서에 대한 저장 및 검색이 가능하지만, 기존의 데이터베이스 즉, Legacy 데이터베이스에 대하여 XML 문서의 저장 및 검색이 불가능하다. 이러한 단점을 개선하기 위하여 미들웨어를 두어 XML 응용과 데이터베이스 사이에 보다 원활한 데이터 전송을 가능토록 하였다[1].

본 논문에서는 XML DTD를 관계형 데이터베이스 스키마로 변환함으로써 XML 응용과 관계형 데이터베이스 사이에 원활한 데이터 전송을 가능하게 하였다. 이를 위해 재사용할 수 있는 미들웨어 컴포넌트들을 설계하였다. 본 논문에서 제시한 관계형 데이터베이스로의 변환을 위한 미들웨어 컴포넌트들은 객체-관계형 데이터베이스 또는 객체지향 데이터베이스에 쉽게 적용될 수 있을 것이다.

Acknowledgement

본 연구는 정보통신부의 ITRC 사업에 의해 수행된 것임

참 고 문 헌

[1] Joo Kyung-Soo, "A Design of Middleware Components for the Connection between XML and RDB", 2001 IEEE International Symposium on Industrial Electronics Proceedings, Pusan, Korea, June 2001.

[2] 이상태, 주경수, "제약조건 유지를 위한 XML DTD의 관계 스키마로 변환 방법", 한국인터넷정보학회, 제1권 제2호, pp. 189-196, 2000.

[3] 이상태; 주경수, "UML 객체모델의 ORDB 객체 매핑", 한국인터넷정보학회, 제2권 제1호, pp. 187-191, 2001.

[4] 이상태, 주경수, "객체모델을 이용한 XML DTD의 ORDB 스키마로의 변환", 한국데이터베이스학회, 춘계Conference, pp. 303-310, 2001.

[5] 이상태, 이정수, 주경수, "객체모델을 기반으로 한 XML DTD의 RDB 스키마로의 변환 방법", 대한전자공학회, 하계종합논문대회, 제24권 제1호, pp. 113-116.

[6] 김경주, 조남규, UML Components 컴포넌트 기반 소프트웨어 명세를 위한 실용적인 프로세스, 인터비젼, 2001, pp. 75-77.

[7] 김명주, XML and Java, 이한출판사, 2000, pp. 16-24.

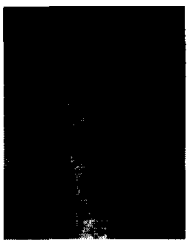
[8] 박지훈, 자바 개발자를 위한 EJB 최신 입문서 Enterprise JavaBeans, 대청미디어, pp. 20-28.

[9] Henri Jubin; Jurgen Friedrichs and the Jalapeno Team 저, 신인철 역, Enterprise JavaBeans 예제로 배우기, 인터비젼, 2000, pp. 32.

◐ 저 자 소 개 ◑

이 정 수
2000년 청운대학교 컴퓨터과학 학과 졸업(학사)
2000년~현재 : 순천향대학교 일반대학원 전산학과 재학중(석사과정)
관심분야 : Database Systems, EJB, CBD
E-mail : jungsoo6@hitel.net

방 승 윤
1996년 한국방송통신대학교 전산학과 졸업(학사)
1998년 호서대학교 산업경영대학원 전산학과 졸업(석사)
2000년 순천향대학교 일반대학원 전산학과 재학중(박사과정)
관심분야 : XML, XML Schema, UML.
E-mail : sybang@hanseo.ac.kr



주 경 수
1980년 고려대학교 이과대학 수학과 졸업(학사)
1985년 고려대학교 일반대학원 전산학과 졸업(석사)
1993년 고려대학교 일반대학원 전산학과 졸업(박사)
1986년~현재 순천향대학교 정보기술공학부 교수
관심분야 : Database Systems, System Integration, Object-oriented Systems.
E-mail : gsoojoo@asan.sch.ac.kr