XML을 이용한 이질적 데이터베이스간의 상호연동*

최정익, 하상호
순천향대학교 정보기술공학부

Interoperability Across Heterogeneous Databases using XML

Jungik Choi, Sangho Ha
Division of Information Technology Engineering, Soonchunhyang University
E-mail : timing@sch.ac.kr, hsh@sch.ac.kr

요 약

이질적 데이터베이스간의 상호연동을 위해서 데이터의 구조를 파악하고 유용한 데이터를 선택하는 일은 매우 어렵고 많은 작업시간이 요구된다. XML을 이용하여 이질적 데이터베이스간의 데이터 교환을 위한 기존 연구가 진행된바 있으나 DTD를 통한 사상이 지원되지 않는 등 미흡한 점이 있다. 본 논문에서는 데이터베이스간의 이질적 상호연동을 효과적으로 수행하는 시스템을 설계한다. 시스템은 원시와 목표 데이터베이스로부터 추출한 스키마 그래프간의 사용자에 의한 사상을 기반으로 원시 데이터베이스를 목표 데이터베이스로의 상호연동을 효과적으로 지원한다. 본 논문에서는 시스템에 의한 이질적 데이터베이스간의 상호연동을 예를 통해서 보여준다.

1. 서론

컴퓨터의 보급 및 발전으로 인하여 많은 양의 정보를 컴퓨터를 이용하여 저장, 관리하는 데이터베이스 시스템 또한 발전을 하고 있다. 데이터베이스 시스템의 발전에 따라 기존의 데이터베이스를 새로운 데이터베이스 시스템으로 교체하거나 이질적인 데이터베이스간의 통합시 데이터베이스의 데이터 중에서 유용한 데이터를 선택하여 새로운 데이터베이스에 저장하는 일은 데이터베이스의 구조파악이 용이하지 않고, 이질적인 데이터베이스간의 자료전달 방법이 통합되지 않는 등 많은 문제가 요구될 수 있다.

본 논문에서는 이러한 문제를 해결하기 위하여 데이터베이스간의 상호연동을 위한 시스템을 설계한다. 먼저, 데이터베이스는 관계형 데이터베이스로 가정하고, 데이터의 이동에는 XML[1]을 사용한다. XML은 사용자가 자신의 태그집합을 정의하고 사용할 수 있다. 또한 XML문서의 구조는 연속적인 중첩을 허용하여 객체 지향적 구조 혹은 데이터베이스 스키마의 구성을 위해 필요한 중첩을 허용하고있는 장점을 가지고 있다.

이 시스템은 데이터베이스의 구조를 파악하기 용이하게 테이블의 관계를 스키마 그래프 형태로 시각화하고, 사용자는 스키마 그래프를 통해서 두 데이터베이스간의 사상을 수행하고, 원시 데이터 베이스의 데이터는 XML문서로 표현되고 사용자의 사상정보를 토대로 하여 XML문서를 변형하는 XSL[2]문서를 자동으로 생성한다. 이 두 문서를 사용하여 원시 XML문서를 목표 데이터베이스의 구조에 맞는 XML문서로 변형하고, 변형된 XML문서는 목표 데이터베이스에 저장된다. 이 시스템은 사용자가 두 데이터베이스간의 사상만 수행하면 나머지 작업은 자동으로 이루어질 수 있도록 구성하였다.

관련된 연구로, K4M에서 개발된 eCross DBTalk[3]도 데이터이동간에 XML문서를 이용하지만 DTD를 통한 매핑이 지원되지 않는다. 따라서 DTD와 XML스키마를 고려한 XML 템플릿을 사용자가 만들어야 하는 부담이 있고, 테이블들간의 스키마를 표현

*본 연구는 한국소프트웨어 진흥원의 ITRC사업에 의해 수행된 것임.

하지 못하는 단점이 있다. 또한 XSL문서를 이용한 XML문서의 변환 없이 데이터를 이동하기 때문에 모든 데이터에 대한 XML문서를 작성해야하는 부담이 있다.

2. 시스템 설계

논문에서는 XML을 이용한 이질적 데이터베이스 간의 상호연동 시스템을 설계한다. 본 시스템의 구조는 그림 1과 같다. 데이터베이스에 질의를 하여 테이블의 스키마를 얻는다. 데이블의 스키마를 파악하여 그래프 형태인 스키마 그래프를 표현한다. 스키마 그래프에서 테이블은 노드로 관계는 에지로 표현된다. 또한 노드에는 테이블의 필드들이 나열되어 있다. 스키마 그래프로 표현된 데이터베이스의 스키마는 DTD 문서로 표현된다. 이 두 과정은 원시 데이터베이스와 목표 데이터베이스에 동일하게 적용된다.

다음은 스키마 그래프간의 사상으로부터 원시 데이터베이스를 목표 데이터베이스에 이동하는 과정을 간략히 설명한다. 두 데이터베이스로부터 스키마 그래프와 DTD를 구성하고 사용자로부터 사상을 입력받는다. 사상은 스키마 그래프 상에서 원시 스키마 그래프에 표현된 필드를 선택하고 그에 대응하는 목표 필드를 선택함으로 이루어진다. 사상이 완료되면 사상정보를 이용하여 XSL문서를 생성한다. 또한 원시 데이터베이스의 데이터를 XML문서로 구성한다. 이때 XML문서는 스키마 그래프로부터 생성된 원시 DTD의 구조를 가진다. 생성된 XML문서에 XSL문서를 적용하여 새로운 XML문서를 생성한다. 이때 생성되는 XML문서는 목표 데이터베이스로부터 생성된 목표 DTD의 구조를 따른다. 변형된 XML문서는 목표 데이터베이스에 요소별로 분리되어 저장된다.

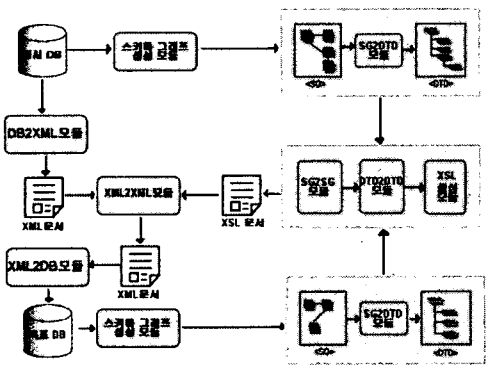

그림 1. 시스템의 전체 구조

2.1 스키마 그래프 생성모듈

데이터베이스의 스키마를 파악하여 스키마 그래프를 작성한다. 데이터베이스의 스키마는 기본키와 외래키로 파악이 가능하다. 모든 데이터베이스에는 이러한 스키마를 저장한 테이블이 존재한다. 이 테이블로부터 데이터베이스의 스키마를 파악하여 그래프로 표현한다. 오라클 데이터베이스의 경우 all_constraints 테이블에 제약조건에 대한 정보가 포함되어 있다. SQL질의를 통하여 all_constraints의 정보를 획득하여 스키마를 파악할 수 있다.

이 모듈은 먼저 데이터베이스에 접속하고 데이터베이스의 스키마를 저장한 테이블을 질의를 통하여 획득한다. 결과 테이블은 테이블의 사용자, 제약조건 이름, 제약조건의 종류, 테이블 이름, 참조 테이블, 참조 필드등의 데이터가 저장되어 있다. 획득한 테이블에서 모든 필드를 순차적으로 처리하여 테이블의 이름, 테이블이 가지는 필드들, 기본키, 외래키, 외래키가 참조하는 테이블, 외래키가 참조하는 필드를 저장한다. 모든 필드에 대한 처리가 완료되면 테이블의 정보가 들어있는 자료를 이용하여 스키마 그래프를 생성한다.

스키마 그래프에서 테이블은 노드로 표현하고, 테이블간의 스키마는 에지로 표현한다. 테이블이 가지는 필드는 노드에 목록으로 표현된다. 에지는 1-1, 1-n의 테이블간의 스키마를 표현한다. 이때 표현되는 스키마는 다음과 같다: 테이블 A, B가 동일한 기본키를 가지면, A, B는 1-1 스키마. 테이블 A, B에서 B가 A의 기본키를 외래키로 가지면 A, B는 1-n의 스키마를 가진다.

2.2 SG2DTD 모듈

스키마 그래프로부터 DTD를 구성한다. 이때 구성되는 DTD는 스키마 그래프에서 표현된 데이터베이스의 스키마의 구조를 표현하고 있다. 데이터베이스의 테이블은 DTD에서 한 요소(이 요소를 A라 한다.)로 표현된다. 또한 테이블은 여러 개의 레코드를 가질 수 있기 때문에 테이블을 표현한 요소 A는 '0-more'의 발생빈도수를 갖는다. 요소 A는 테이블이 가지는 필드를 자식 요소로 가질 수 있다. 이때 테이블의 필드 중에서 기본키로 정의되어 있는 필드는 요소 A의 속성으로 표현된다. 기본키를 표현한 속성의 이름은 필드명을 가지고 속성의 타입은 ID타입으로 한다. 또한 외래키를 가지는 필드는 요소 A의 속성으로 표현한다. 외래키를 표현한 속성의 이름은 필드명을 가지고

속성의 타입은 IDREF타입으로 하여 참조 테이블의 필드값을 표현한 속성을 참조할 수 있게 한다.

이렇게 생성된 DTD는 모든 테이블을 동일 수준에 위치하게 한다. 테이블들의 스키마는 요소의 ID, IDREF 속성으로 표현한다. DTD의 루트노드는 시스템에서 임의로 생성하여 DTD정의를 만족하게 한다.

2.3 사상 엔진

사상모듈은 SG2SG모듈, DTD2DTD모듈, XSL생성모듈로 이루어진다. 사상모듈은 SG2SG로부터 사용자의 사상을 지원하고 사상정보를 포함하는 해쉬테이블을 DTD2DTD모듈로 전하여 DTD간의 사상으로 변환한다. 또한 DTD간의 사상정보를 이용하여 XSL생성모듈에서 XSL문서를 생성한다.

SG2SG 모듈

SG2SG 모듈은 스키마 그래프 생성모듈에 의하여 생성된 두 스키마 그래프간의 사상을 지원한다. 사용자는 스키마 그래프를 이용하여 데이터베이스의 관계를 파악하고 두 데이터베이스에서 사상될 필드를 선택할 수 있다. 이때 사용자는 에지에서 1, ∞의 표현을 통하여 1-1, 1-n, n-n의 관계를 파악할 수 있다. 또한 사용자가 현재 사상중인 테이블은 다른 테이블들과 구별되게 표현하여 테이블의 구조나 다른 테이블들의 관계를 파악하기 용이하게 한다.

사상의 진행은 원시 스키마 그래프 상에서 한 노드의 필드를 선택하고 그에 대응하는 목표 스키마 그래프 상에서 한 노드의 필드를 선택함으로 진행된다. 사상이 이루어지면 노드와 필드가 쌍을 이루어 해쉬테이블에 저장되고 사상된 필드를 다시 사상할 수 없게 하여 중복 사상을 막는다. 또한 다대일 사상이 가능하게 하여 다양한 사상을 지원한다. 다대일 사상시 원시 스키마 그래프에서 테이블의 필드를 두 개 이상 선택하고 그에 대응하는 목표 스키마 그래프의 필드를 선택한다. 원시 스키마 그래프에서 선택된 필드들의 값은 하나로 합쳐져 목표 스키마 그래프의 필드로 저장된다. 사용자는 SG2SG 모듈을 통하여 비쥬얼한 환경에서 테이블의 관계를 파악하고 손쉽게 사상할 수 있다.

DTD2DTD 모듈

DTD2DTD 모듈은 SG2SG로부터 사상정보를 포함하는 해쉬테이블을 가져와 DTD간의 사상으로 변환한다. 이 모듈은 내부적으로 동작하는 모듈로 SG2SG상에서 사용자가 입력한 사상정보를 SG2DTD모듈에 의하여 생성된 원시DTD와 목표DTD간의 사상으로 변환한다.

XSL 생성 모듈

XSL 생성 모듈은 DTD간의 사상정보를 이용하여 XSL문서를 생성한다.

XSL문서는 사상정보를 포함하는 해쉬테이블과 목표 DTD를 사용하여 생성된다. 목표 DTD 트리에서 사상된 노드A와 A의 모든 부모를 포함하는 새로운 DTD 트리 T를 생성한다. 생성된 DTD 트리 T를 in-order 트리 방문 순서에 따라 모든 노드를 방문한다. 각 노드에 대하여 사상의 유형을 판단하고 그에 따른 적절한 XSL문을 작성함으로서 XSL문서가 생성된다. 사상의 유형은 요소-요소, 요소-속성, 속성-요소, 속성-속성으로 분류할 수 있다.

2.4 DB2XML 모듈

DB2XML 모듈은 SG2DTD 모듈이 원시 데이터베이스로부터 생성한 원시DTD의 구조를 가지는 원시 XML문서를 생성한다. 이 모듈은 데이터베이스에 접속하고 사용자의 사상에 참여한 필드의 데이터를 획득하여 원시 DTD의 구조를 가지는 XML문서를 생성하다. 이때 생성된 원시 XML문서에는 데이터베이스의 모든 데이터가 저장된다. 이 모듈은 IBM의 XML4Java[4]를 사용하여 원시 DTD로부터 유효한 XML문서를 생성한다.

2.5 XML2XML 모듈

이 모듈은 DB2XML모듈이 생성한 XML문서에 XSL생성모듈이 생성한 XSL문서를 적용하여 목표 XML문서로 변환한다. 이 모듈은 Xalan의 XSLT[5]엔진을 사용하여 XML문서를 변환한다.

2.6 XML2DB 모듈

XML2DB 모듈은 XML문서를 DB에 저장하는 모듈이다. XML 변환 모듈에 의하여 생성된 XML문서는 목표 DTD의 구조를 가진다. 목표 DTD는 목표 데이터베이스에 따른 구조이기 때문에 새로운 XML문서는 데이터베이스에 종속적이 된다. 이 모듈은 새로운 XML문서를 파싱하여 요소별로 분리하고 사상정보로부터 테이블과 필드의 정보를 파악하여 데이터베이스에 저장한다. 루트 요소의 자식 요소 A의 이름은 테이블의 이름과 일치한다. 테이블의 구조는 스키마 그

래프 생성모듈에 의하여 파악이 된 상태이므로 A 요소의 자식 요소가 가지는 텍스트 값을 JDBC를 이용하여 질의하여 A 요소와 같은 이름의 테이블에 저장하게 된다.

3. 예제

본 시스템을 이용하여 두 데이터베이스간의 데이터의 이동 예를 보인다. 원시 데이터베이스는 도서의 정보를 가지고 있는 데이터이고 목표 데이터베이스는 상품정보를 저장 할 수 있는 데이터베이스이다. 두 데이터베이스의 테이블구조는 표 1, 표 2와 같다.

Authors	BookAuthors	Books	Affiliation
AuthorID(PK) FirstName LastName Address	**ISBN(FK)** **AuthorID(FK)**	**ISBN(PK)** Title Pub_Date Price Abstract Content	**AuthorID(FK)** CompanyName Tel Fax Url

표 1. 원시 데이터베이스의 테이블구조

ProductCatalog	Participant	Affiliation
ProductID(PK) ProductName DateOfIssue Price Specification	**ProductID(FK)** **Id(FK)** Role FirstName LastName Address	**Id(PK)** Name Tel Fax

표 2. 목표 데이터베이스 테이블구조

스키마 그래프(SG) 생성

두 데이터베이스는 스키마 그래프 생성모듈에 의하여 데이터베이스의 스키마가 그래프형태로 구성되어 그림 2와 같이 사용자에게 보여진다.

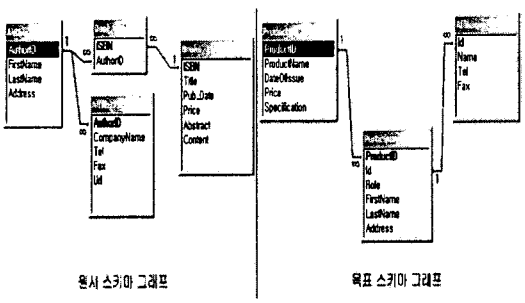

그림 2. 스키마 그래프 생성

원시 데이터베이스의 스키마는 그림 2의 원시 스키마 그래프로 표현되었다. Authors테이블은 BookAuthors, Affiliation테이블과 1-n의 관계를 갖고 Books테이블은 BookAuthors테이블과 1-n의 관계를 갖고 있음을 알 수 있다. 또한 목표 데이터베이스의 스키마 역시 그림 2의 목표 스키마 그래프로 표현되었다. ProductCatalog테이블은 Participant테이블과 1-n의 관계를 갖고 Affiliation테이블은 Participant테이블과 1-n의 관계를 갖고 있음을 알 수 있다.

SG로부터 DTD 생성

스키마 그래프가 생성된 후 SG2DTD모듈에 의하여 DTD가 생성된다. DTD의 구조는 그림 3과 같다.

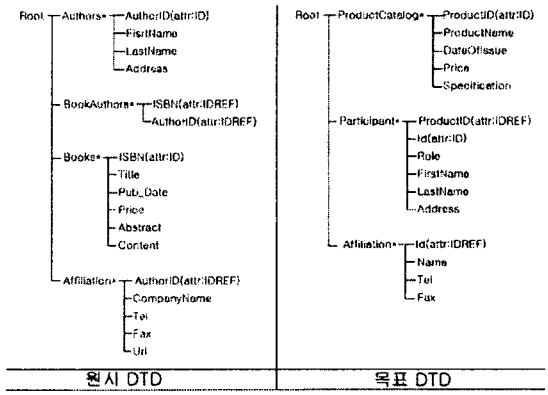

그림 3. DTD 구성

테이블의 이름은 DTD의 중간 요소로 표현되고 각각의 필드는 텍스트 값을 가질 수 있는 요소로 표현된다. 또한 테이블들의 관계는 기본키와 외래키를 DTD의 ID, IDREF속성으로 표현하였다.

원시 스키마 그래프의 Authors테이블은 AuthorID, FisrtName, LastName, Address를 필드로 가진다. 이 필드 중에서 기본키로 정의되어 있는 AuthorID필드는 DTD에서 Authors요소의 ID속성으로 정의된다. BookAuthors테이블은 ISBN, AuthorID를 필드로 가진다. 이 필드들은 모두 외래키로 정의되어 있기 때문에 BookAuthors요소의 IDREF속성으로 정의된다. Book테이블은 ISBN, Title, Pub_Date, Price, Abstract, Content를 필드로 가진다. 이 필드 중에서 기본키로 정의되어 있는 ISBN필드는 Book요소의 ID속성으로 정의된다. Affiliation테이블은 AuthorID, CompanyName, Tel, Fax, Url을 필드로 가진다. 이 필드 중에서 외래키로 정의되어 있는 AuthorID필드는 Affiliation요소의 IDREF속성으로 정의된다.

목표 스키마 그래프의 ProductCatalog테이블은 ProductID, ProductName, DateOfIssue, Price, Specification을 필드로 가진다. 이 필드 중에서 기본키로 정의되어 있는 ProductID필드는 ProductCatalog요소의 ID속성으로 정의된다. Participant테이블은

ProductId, Id, Role, FirstName, LastName, Address를 필드로 가진다. 이 필드 중에서 기본키로 정의되어 있는 Id필드는 ID속성으로 외래키로 정의되어 있는 ProductID필드는 Participant요소의 IDREF속성으로 정의된다. Affiliation테이블은 Id, Name, Tel, Fax를 필드로 가진다. 이 필드 중에서 외래키로 정의되어 있는 Id필드는 Affiliation요소의 IDREF속성으로 정의된다.

그림 3과 같이 구성된 DTD는 스키마 그래프에서 테이블의 관계를 표현할 수 있다. 기본키는 ID속성으로 외래키는 IDREF속성으로 정의된다. 또한 테이블을 표현한 요소는 '0-more'의 발생빈도를 가져 테이블의 데이터의 수만큼 반복될 수 있다.

스키마그래프간의 사상

스키마 그래프가 생성된 후 사용자는 스키마 그래프 상에서 이동 데이터의 사상이 가능하다. 사상의 순서는 원시 스키마 그래프 테이블의 필드를 선택하고 그에 대응하는 목표 스키마 그래프 테이블의 필드를 선택함으로 이루어진다. 사상되어진 필드는 다시 사상되는 중복사상을 피하기 위하여 사용자가 선택할 수 없게 한다. 또한 다대일 사상시에는 원시 스키마 그래프 테이블의 필드들을 선택하고 그에 대응하는 목표 스키마 그래프 테이블의 필드를 선택함으로 이루어진다. 일대일 사상시에는 필드가 갖고 있는 데이터만 이동하지만 다대일 사상시에는 데이터의 의미파악을 위하여 필드명을 데이터에 포함한다. 예제에서는 표 3과 같은 사상을 진행하였다고 가정한다.

원시 스키마 그래프	목표 스키마 그래프
Authors/AuthorID	Participant/Id
Authors/FisrtName	Participant/FirstName
Authors/LastName	Participant/LastName
Authors/Address	Participant/Address
Books/ISBN	ProductCatalog/ProductId
Books/Title	ProductCatalog/ProductName
Book/Pub_Date	ProductCatalog/DateOfIssue
Books/Price	ProductCatalog/Price
Books/Abstract	ProductCatalog/Specification
Books/Content	ProductCatalog/Specification
Affiliation/AuthorID	Affiliation/Id
Affiliation/CompanyName	Affiliation/Name
Affiliation/Tel	Affiliation/Tel
Affiliation/Fax	Affiliation/Fax

표 3. 사상 정보

표 3의 사상정보 중에서 Books/Abstract, Books/Content는 ProductCatalog/Specification으로 다대일 사상됨을 알 수 있다. 이때 Specification의 데이터는 데이터의 의미 파악의 용이하게 두 데이터의 앞에 필드명을 표시한다.

XSL문서의 생성

사상이 완료되면 XSL 생성모듈에 의하여 XSL문서가 생성된다. 이때 생성되는 XSL문서는 사용자의 사상정보를 이용하여 생성된다. XSL 생성 모듈에 의하여 생성된 XSL문서는 표 4와 같다.

```
<xsl:template match="/">
   <xsl:apply-templates select="Root"/>
</xsl:template>
<xsl:template match="Root">
   <Root>
   <xsl:apply-templates select="Books"/>
   <xsl:apply-templates select="BookAuthors"/>
   <xsl:apply-templates select="Affiliation"/></Root>
</xsl:template>
<xsl:template match="Books">
   <ProductCatalog>
   <xsl:attribute name="ProductID">
      <xsl:value-of select="@ISBN"/></xsl:attribute>
   <ProductName>
   <xsl:value-of select="Title"/></ProductName>
   <DateOfIssue>
   <xsl:value-of select="Pub_Date"/></DateOfIssue>
   <Price><xsl:value-of select="Price"/></Price>
   <Specification>
      Abstract : <xsl:value-of select="Abstract"/>
      Content : <xsl:value-of select="Content"/>
  </Specification>
</ProductCatalog>
</xsl:template>
<xsl:template match="BookAuthors">
   <Participant>
   <xsl:attribute name="ProductID">
      <xsl:value-of select="@ISBN"/></xsl:attribute>
   <xsl:attribute name="Id">
      <xsl:value-of select="@AuthorID"/></xsl:attribute>
   <Role/>
   <FirstName>
   <xsl:value-of select="../Authors[@AuthorID =
./@AuthorID]/FirstName"/>
   </FirstName>
   <LastName>
   <xsl:value-of select="../Authors[@AuthorID =
./@AuthorID]/LastName"/>
   </LastName>
   <Address>
   <xsl:value-of select="../Authors[@AuthorID =
./@AuthorID]/Address"/>
   </Address>
   </Participant>
</xsl:template>
<xsl:template match="Affiliation">
   <Affiliation>
   <xsl:attribute name="Id">
      <xsl:value-of select="@AuthorID"/></xsl:attribute>
   <Name><xsl:value-of select="CompanyName"/></Name>
   <Tel><xsl:value-of select="Tel"/></Tel>
   <Fax><xsl:value-of select="Fax"/></Fax>
   </Affiliation>
</xsl:template>
</xsl:stylesheet>
```

표 4. XSL 문서

원시 XML문서의 생성과 목표 XML문서로 변환

XML2XML 모듈은 생성된 XSL문서와 DB2XML 문서에 의하여 생성된 XML문서를 이용하여 새로운 XML문서를 생성한다.

DB2XML 모듈은 스키마 그래프로부터 생성된 DTD의 구조에 맞추어 데이터베이스의 데이터를 XML문서로 생성한다.

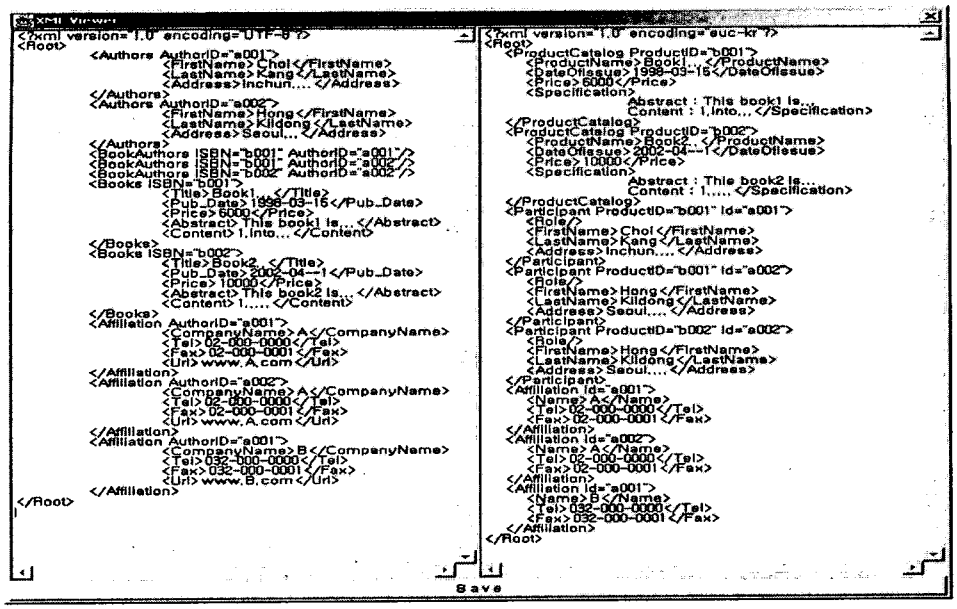

그림 4. XML문서의 생성

그림 4는 DB2XML모듈에 의하여 생성된 XML문서를 XML2XML 모듈에서 XSL문서를 적용하여 생성된 XML문서를 보이고 있다. 새로운 XML문서는 목표 DTD의 구조를 따름을 확인할 수 있다.

XML문서의 저장

새로운 XML문서는 XML2DB 모듈에 의하여 목표 데이터베이스에 저장된다. XML문서의 모든 요소의 값이 데이터베이스에 저장되기 때문에 DOM 파서를 사용하여 문서를 파싱하고 요소별로 분리하는 것이 적당하다.

4. 결론

본 논문에서는 데이터베이스간의 데이터전달을 손쉽게 수행할 수 있는 방법을 제시하고 핵심모듈의 구현 방안을 제시하였다. 이 시스템은 데이터 베이스의 구조를 그래프 화하여 사용자에게 테이블간의 스키마를 파악하는데 도움을 주고 테이블의 스키마를 표현하는 DTD를 정의하여 데이터를 XML문서로 표현하였다. 또한 사용자는 손쉬운 사상을 통하여 XSL문서를 생성한다. 사용자의 사상을 바탕으로 생성된 XSL문서와 XML문서를 사용하여 목표 데이터베이스의 구조를 갖는 XML문서로 변환한 후 XML문서의 요소를 파악하여 데이터베이스에 저장한다. 향후 연구로는 본 논문에서 제안한 모듈을 구현하고 실행을 통하여 그 효과를 입증한다.

[참고문헌]

[1]Neil Bradley, 'The XML companion', Second Edition, Reading, Addison-Wesley, 1999

[2]Khun Yee Fung, 'XSLT Working with XML and HTML', Addison-Wesley, 2001

[3]http://www.k4m.co.kr

[4]http://www.alphaworks.ibm.com/

[5]http://xml.apache.org/

[6]Hiroshi Maruyama, Kent Tamura, Naohiko Uramoto, 'XML and Java: Developing Web Applications', Addison-Wesley, 1999

[7]Benoit Marshal, 'Applied XML Solutions, Sams, 2000

[9]Simon St. Laurent, Ethan Cerami, 'Building XML Applications', McGraw-Hill Professional Publishing, 1999

[9]Michael H. Kay, 'XSLT Programmer's Reference 2nd Edition', Wrox Press Inc, 2001

[10]Elliotte Rusty Harold, Elliote Rusty Harold, 'XML Bible 2nd Edition', Hungry Minds, Inc, June 2001

[11]Simon St. Laurent, Ethan Cerami, 'Building XML Applications', McGraw-Hill Professional Publishing, 1999