사용자 정의 XML 뷰 기반의 XML 스키마 관리 시스템

정 채 영[†] · 김 영 옥[††] · 이 미 영[†††] · 강 현 석[††††] · 배 종 민[†††††]

요 약

이질 데이터베이스 통합 방법론의 하나인 미디에이터-랩퍼 시스템은 각 데이터베이스의 내용이 물리적으로 이동하는 대신, 가상의 통합된 뷰를 유지한다. 미디에이터는 가상의 통합된 뷰를 유지하기 위하여, 지역 데이터베이스와 직접 통신하는 랩퍼의 스키마 관리기의 도움이 필요하다. 본 논문에서는 미디에이터에게 관계형 데이터베이스의 스키마를 XML 스키마로 제공하는 랩퍼의 스키마 관리기를 설계 구현한 결과를 제시한다. 설계된 XML 스키마 관리기가 사용자 정의 XML 뷰를 지원하도록 하기 위하여 사용자 정의 XML 뷰로부터 XML 스키마를 생성하기 위한 뷰 트리 모델을 제시하고, 그 변환 알고리즘을 제시한다. 제시된 모델은 이질의 정보원의 스키마 관리기에 대하여 일관되게 적용할 수 있다.

An XML Schema Manager based on the User-defined XML View

Chai-Young Jung[†] · Young-Ok Kim[††] · Mi-Young Lee[†††]
Hyun-Syug Kang[††††] · Jong-Min Bae[†††††]

ABSTRACT

A mediator-wrapper system, which is one of integration methods of heterogeneous databases, preserves a virtual integrated view rather than physical movement of the contents of each database. In order for a mediator to preserve a virtual integrated view, it requires the schema manager of the wrapper which communicates with local databases. This paper presents a schema manager of a wrapper that provides a mediator with XML schema which is generated from relational database schema. The XML schema manager supports a user-defined XML view. We present the view tree model which is used for materializing the use-defined XML view to generate XML schema, and the conversion algorithm based on the proposed view tree model. This model is can be uniformly applied to all schema managers of heterogeneous information sources.

키워드 : 랩퍼(Wrapper), 기본 XML 스키마(Base XML Schema), 응용 XML 스키마(Application XML Schema), 응용 XML 뷰(Application XML View), 뷰 트리(View Tree)

1. 서 론

분산된 데이터베이스의 통합에 대한 필요성은 전자상거래, 디지털 라이브러리, 포탈서비스, 의학-생물정보 분야 등에서 오랜 전부터 제기되어 왔다. 이는 사용자에게 얻고자 하는 정보가 있는 정보원에 대한 위치 결정에 대한 부담을 덜어주고, 다수의 데이터베이스에 대하여 사용자에게 하나의 데이터베이스 관점을 제공함으로서, 연관된 데이터를 다수의 정보원으로부터 검색, 가공하여 하나의 통합된 정보를 제공할 수 있도록 한다.

데이터베이스 통합 방법론 중의 하나인 미디에이터-랩퍼 시스템에서는 지역 데이터베이스에 있는 실제 데이터가 물리적으로 이동되는 것이 아니라, 각 지역 데이터베이스의 내용을 가상적으로(virtual) 통합된(integrated) 뷰를 미디에이터(mediator)가 유지한다. 그리고 사용자는 통합된 뷰를 통해서 질의한다. 이를 위해서는 각 지역 데이터베이스와 직접 대화하는 자료 저장소 랩퍼(wrapper)가 필요하다. 즉, 랩퍼는 각 지역 데이터베이스마다 하나씩 존재하여, 지역 데이터베이스에 대한 내용을 미디에이터에게 전달함으로서, 미디에이터가 통합된 뷰를 구성할 수 있도록 정보를 제공하고, 미디에이터로부터 받은 사용자 질의를 지역 데이터베이스에서 수행시킨 후 그 결과를 미디에이터에게 넘겨주는 역할을 한다. 미디에이터에게 지역 데이터베이스의 내용을 전달하기 위해서는 각 지역 데이터베이스의 스키마의 내용을 표현할 수 있는 하나의 중립적인 모델이 필요한데, 본 논문에서는 W3C 표준안이 XML Schema[9]를 이용한다. XML Schema 명세는 2001년 5월에 발표된 W3C의 권고안을 따른다.

본 논문에서는 먼저, 지역 데이터베이스의 스키마를 XML Schema로 표현하는 기본적인 방법과 고려사항에 대하여 논한다. 변환된 XML 스키마는 일반적인 데이터베이스를 XML 데이터베이스의 관점으로 변환한 것이라 할 수 있다. 따라서 관계형 데이터베이스에서 사용자 정의 뷰를 정의할 수 있는 것과 마찬가지로 XML 데이터베이스에 대해서도 사용자가 스키마를 보는 관점을 다시 정의하는 기능이 필요

※ 본 연구는 한국전자통신연구원의 "클러스터 기반 통합 멀티미디어 DBMS 개발" 사업의 일부로 수행된 결과임.
† 준 회 원 : 경상대학교 대학원 컴퓨터과학과
†† 정 회 원 : 경상대학교 대학원 컴퓨터과학과
††† 정 회 원 : 한국전자통신연구원
†††† 종신회원 : 경상대학교 컴퓨터과학과/컴퓨터 정보통신연구소 교수
††††† 종신회원 : 경상대학교 컴퓨터과학과/컴퓨터 정보통신연구소 교수, 교신저자
논문접수 : 2002년 7월 24일, 심사완료 : 2003년 1월 24일

하다. 즉, 사용자가 원하는 XML 뷰를 정의하는 기능이 필요하다. 뷰를 정의하기 위하여 새롭게 정의한 뷰 정의어를 이용하거나 기존의 XML 질의어를 이용하여 뷰를 정의할 수 있는데, 본 논문에서는 사용자가 정의하는 XML 뷰를 기술하기 위한 뷰 정의어로서 XML 질의어인 XQuery[10]를 이용한다. XQuery 명세는 2001년 6월에 발표된 W3C의 초안을 따른다.

사용자가 정의한 XML 뷰는 XQuery로 정의되지만 실제 미디에이터에서 필요한 것은 XQuery로 정의된 XML 뷰가 아니라 이로부터 생성되는 가상의 스키마, 즉 XML Schema로 표현된 XML 스키마가 필요하다. 따라서 랩퍼는 XQuery로 기술된 XML 뷰로부터 XML Schema로 표현된 XML 스키마를 생성하여 미디에이터에게 전달해야 하는데, 본 논문에서는 사용자 정의 XML 뷰를 뷰 트리로 변환하는 모델을 제시하고, 제시된 뷰 트리 기반의 XML 스키마 변환 알고리즘을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 살펴보고, 3장에서는 XML 스키마 관리기의 구조를 제시한다. 4장에서는 XML 뷰에 관하여 논하고, 5장에서는 뷰 트리 모델과 XML 스키마 변환 알고리즘을 제시한다. 6장에서는 구현 내용을 보이고 마지막으로 7장에서는 결론과 향후과제를 논한다.

2. 관련 연구

미디에이터가 이질의 정보원에 대한 통합된 뷰를 유지하기 위해서는, 하나의 공통된 모델을 사용해서 랩퍼와 대화해야 한다. TSIMMIS 프로젝트[8]의 경우 랩퍼와 미디에이터 사이에 OEM(Object Exchange Model) 이라고 불리는 데이터 모델을 이용한다. 랩퍼는 미디에이터에게 지역 정보원의 내용을 OEM으로 변환하고 미디에이터는 다른 형태의 사용자 뷰를 정의하기 위해 MSL(Mediator Specification Language)을 이용한다. INEEL 프로젝트[7]에서 랩퍼는 지역 정보원의 내용을 기술하기 위해 ODL(Object Definition Language)을 이용한다. ODL은 각 객체의 클래스를 정의하는 기능을 제공한다. 각각의 클래스에는 해당 객체의 애트리뷰트, 함수, 관계를 표현 할 수 있다. ODL로 정의된 랩퍼 명세는 Semantic Model로 변환되어 유지된다. MIX 프로젝트[5]에서 랩퍼는 지역 정보원의 내용을 DTD로 표현한다. MIX에서는 통합하고자 하는 지역 정보원의 스키마를 DTD로 표현하고 이를 원시 DTD라 정의하고 미디에이터에서는 XMAS(XML Matching And Structuring)를 이용하여 뷰를 정의한다. 정의한 뷰에 대한 스키마에 대한 DTD를 뷰 DTD라 한다. 원시 DTD는 정보원의 메타정보를 이용하여 자동으로 생성되는데 관계형 데이터베이스의 스키마를 원시 DTD로 변환시 관계형 데이터베이스의 릴레이션들의 참조 관계를 이용하여 다양한 방법으로 원시 DTD를 정의하는 방법도 연구되었다[6]. 반면에 본 논문에서는 미디에이터와 랩퍼 사이의 공통된 표현 모델로서 W3C에서 정의한 XML Schema를 이용한다.

한편, 관계형 데이터베이스에 대하여 XML 질의를 지원하는 시스템에서, 사용자 정의 XML 뷰를 지원하는 연구가 최근에 이루어지고 있다[1-4]. 그 중에서 XPERANTO[3]에서는 관계형 데이터베이스의 스키마 내용을 XML Schema로 표현하고 이를 바탕으로 XQuery를 이용하여 사용자 정의 XML 뷰 기능을 지원한다. 그리고 사용자 질의는 이 사용자 정의 XML 뷰상에서 처리된다. SilkRoute[1]에서는 뷰 정의어 RXL(Relational to XML Transformation Language)를 이용하여 관계형 데이터베이스에서의 XML 뷰를 정의하고 이를 기반으로 XML-QL로 표현된 사용자 질의를 처리한다. 본 논문에서는 위와 같은 관계형 데이터베이스에서의 사용자 정의 XML 뷰 개념을 랩퍼 스키마 관리기의 설계에 활용하며, 이때 필요한 제반 요소들을 제시한다.

3. XML 스키마 관리기의 구조

3.1 기본 용어 정의

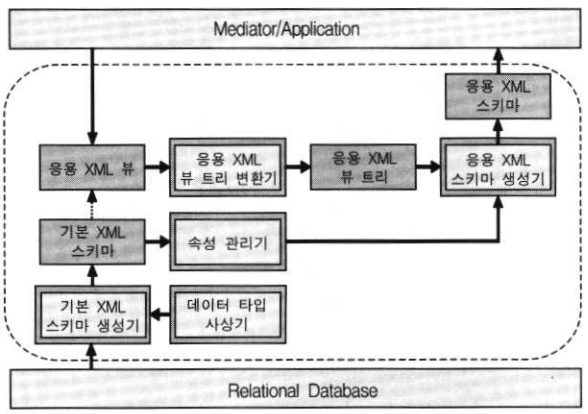
지역 데이터베이스 스키마를 XML 스키마로 표현한 결과를 기본 XML 스키마(base XML schema)라고 정의한다. 기본 XML 스키마는 데이터베이스의 전체 스키마 내용을 표현한다. 기본 XML 스키마로부터 사용자가 관심 있는 정보만을 추출하여 사용자 관점에서 생성된 XML 스키마를 응용 XML 스키마(application XML schema)라 정의한다.

사용자가 원하는 응용 XML 스키마를 생성하기 위해 직접 기본 XML 스키마를 이용하여 응용 XML 스키마를 정의하는 것은 비현실적이다. 이를 위하여 사용자가 기본 XML 스키마를 바탕으로 사용자 정의 XML 뷰를 기술하면 그 뷰로부터 응용 XML 스키마가 자동으로 생성되도록 한다. 이때 사용자가 정의한 XML 뷰를 응용 XML 뷰(application XML view)라 정의한다. 사용자가 응용 XML 뷰를 정의하면 이에 대응하는 응용 XML 스키마가 자동으로 생성된다.

3.2 XML 스키마 관리 시스템의 구조

랩퍼의 스키마 관리기는 미디에이터의 요구에 의해 XML 스키마를 생성, 수정, 삭제, 관리하는 기능을 가진다. 지역 데이터베이스로부터 생성된 XML 스키마는 XML 문서 형태로 관리되며 필요에 따라 수정, 삭제된다. (그림 1)은 랩퍼의 스키마 관리기에서 핵심이 되는 XML 스키마 생성 및 관리 시스템의 구조이다.

기본 XML 스키마 생성기는 관계형 데이터베이스의 시스템 카탈로그나 릴레이션의 메타 데이터를 이용하여 이질의 정보원 스키마로부터 기본 XML 스키마를 자동 생성한다. 이때 관계형 데이터베이스와 XML Schema의 데이터타입 사이에 발생하는 불일치 문제는 데이터 타입 사상기에서 해결한다. 속성 관리기는 기본 XML 스키마의 생성에



(그림 1) XML 스키마 생성 및 관리 시스템

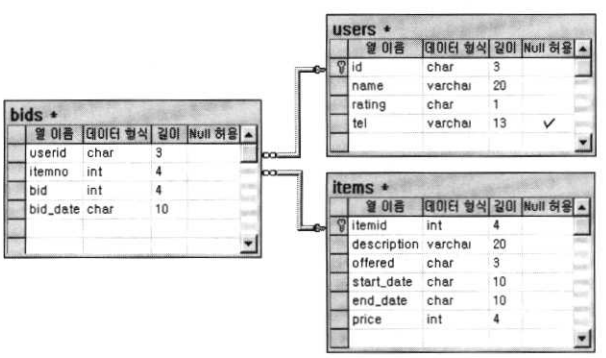
사용된 관계형 데이터베이스의 릴레이션과 애트리뷰트에 대한 이름과 데이터 타입 등의 정보를 관리하며 응용 XML 스키마를 생성할 때 이를 이용한다. 응용 XML 뷰 트리 변환기는 사용자가 기술한 응용 XML 뷰를 응용 XML 스키마로 변환하기 위해 트리 모델로 변환하는 역할을 한다. 응용 XML 스키마 생성기는 사용자가 정의한 응용 XML 뷰로부터 응용 XML 스키마를 생성한다.

4. XML 뷰

본 장에서는 지역 데이터베이스의 스키마를 XML 관점에서 XML 스키마로 표현하는 기본 방법과 사용자가 정의하는 사용자 정의 XML 뷰에 대해 논한다.

4.1 기본 XML 스키마의 생성

지역 데이터베이스로부터 기본 XML 스키마를 생성하는 방법은 XPERANTO[3]에서 제시된 방법을 그대로 활용한다. 여기서는 그 개념을 간단히 소개한다. 기본적으로 각 릴레이션 이름은 XML의 엘리먼트로 변환되고, 릴레이션의 각 속성들은 해당 엘리먼트의 하위 엘리먼트로 표현한다. 그리고 각 릴레이션들 사이의 관계는 무시하고 모두 동일한 레벨로 표현한다. 변환과정을 설명하기 위하여 (그림 2)와 같은 데이터베이스의 예를 생각하자.



(그림 2) 관계형 데이터베이스의 예

(그림 3)은 (그림 2)의 관계형 데이터베이스의 스키마를 기본 XML 스키마로 변환한 예의 일부로서 (그림 2)의 users 릴레이션의 스키마를 XML 스키마로 나타난 예이다. 릴레이션과 릴레이션의 속성들은 기본 XML 스키마에서 엘리먼트로 표현되는데 릴레이션 이름인 users가 엘리먼트로 표현되고 users 릴레이션이 가지는 id, name, rating, tel 애트리뷰트는 해당 엘리먼트의 하위 엘리먼트로 표현된다. 이때 릴레이션의 내용을 XML 문서로 표현할 경우 애트리뷰트에 해당하는 속성 엘리먼트들은 튜플의 수만큼 반복되므로 릴레이션 엘리먼트와 속성 엘리먼트 사이에 가상의 'tuple' 엘리먼트를 추가하여 릴레이션 엘리먼트와 속성 엘리먼트를 구별함과 동시에 속성 엘리먼트들이 반복적으로 나타날 수 있음을 표시한다. 또한 각 속성 엘리먼트들은 실제 데이터를 가지게 됨으로 릴레이션에서 해당 속성이 가지는 데이터 타입과 NULL 허용 여부와 같은 부가 정보를 추가한다.

```
...
  <xsd : element name = "users">
    <xsd : complexType>
      <xsd : sequence>
        <xsd : element name = "tuple" minOccurs="0"
                       maxOccurs = "unbounded">
          <xsd : complexType>
            <xsd : sequence>
              <xsd : element name = "id" ··· />
              <xsd : element name = "name" ··· />
              <xsd : element name = "rating" ··· />
              <xsd : element name = "tel" ··· />
            </xsd : sequence>
          </xsd : complexType>
        </xsd : element>
      </xsd : sequence>
    </xsd : complexType>
  </xsd : element>
...
```

(그림 3) 기본 XML 스키마의 일부

4.2 데이터 형 변환

XML 스키마로 변환시 릴레이션의 애트리뷰트에 해당하는 속성 엘리먼트들은 데이터 타입을 가진다. 각 속성 엘리먼트의 데이터 타입은 XML Schema에서 제공하는 데이터 타입으로 변환된다. 이때 관계형 데이터베이스와 XML Schema에서 제공하는 데이터 타입 사이에 차이가 존재함으로 이들간의 적절한 변환이 필요하다. XML Schema에서는 기본 데이터 타입 이외에 데이터형의 확장이나 제한 또는 변환을 통해 다양한 데이터 타입을 재정의 할 수 있는 방법을 제공한다.

<표 1>은 관계형 데이터베이스인 MS-SQL SERVER 2000에서 제공하는 데이터 타입과 XML Schema에서 제공되는 데이터 타입간의 사상을 나타내는 내용의 일부이다. 관계형 데이터베이스는 다양한 형태의 데이터 타입을 제공하는데 제시된 스키마 관리기는 그 중에서 숫자형이나 문자형 또는 시간을 표현하는 데이터 타입과 같은 텍스트형(textual) 데이터 타입에 대해서만 사상한다. 사용자 정의 타입의 경

우는 XML Schema의 anyType에 대응된다. 일반적으로 두 모델간의 형에 대하여 정확한 사상을 하기 어렵다. 예를 들어 이진(binary) 데이터 타입의 경우 실제 해당하는 데이터 타입에 대한 표현이 어렵고, 또한 실제 데이터베이스에 들어있는 내용을 추출하는 과정과 추출한 내용을 XML 문서로 표현하기 어렵다. 제시된 스키마 관리기는 의미 있는 형에 대해서만 선택적으로 변환하는 방식으로 구현되었다.

〈표 1〉 관계형 데이터베이스와 XML Schema상의 데이터 타입 사상 예

MS-SQL 2000	XML Schema
bit	<xsd : element> <xsd : simpleType> <xsd : restriction base = "xsd : unsignedByte"> <xsd : minInclusive value = "0"/> <xsd : maxInclusive value = "1"/> </xsd : restriction> </xsd : simpleType> </xsd : element>
decimal	<xsd : element type = "xsd : decimal"/>
float	<xsd : element type = "xsd : double"/>
int	<xsd : element type = "xsd : int"/>
smallint	<xsd : element type = "xsd : short"/>

4.3 속성 엘리먼트 관리

사용자가 정의하는 응용 XML 스키마는 기본 XML 스키마에 기반을 둠으로 응용 XML 스키마에 나타나게 될 엘리먼트는 기본 XML 스키마로부터 유도된다. 따라서 응용 XML 스키마 생성을 위해서 기본 XML 스키마에서 필요한 엘리먼트를 찾기 위해 계속적인 탐색이 필요하다. 이를 위해 생성에 관여한 각 속성 엘리먼트를 속성 관리기에서 관리하고 이를 새로운 응용 XML 스키마 생성시에 이용함으로써 불필요한 탐색을 줄인다. 이를 위해 속성 관리기의 자료 구조는 <표 2>와 같다.

〈표 2〉 속성 관리기의 자료 구조

필 드	내 용
속성에 대한 경로식	기본 XML 스키마상의 속성에 대한 경로식
기본 데이터 타입	속성 엘리먼트의 기본 데이터 타입
부가 데이터 Facets	데이터 타입의 확장 또는 제한
NULL 값 허용 여부	NULL 값 허용 여부

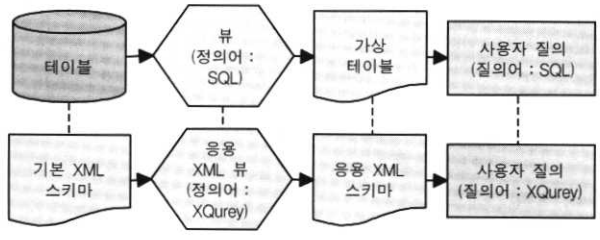
속성 관리기에서는 기본 XML 스키마상의 경로식으로 각 속성 엘리먼트를 구별한다. 그리고 속성 엘리먼트를 표현하기 위해 기본 데이터 타입과 데이터베이스와 XML Schema의 데이터 타입 사상을 위해 필요한 부가 정보를 가지며 해당 속성이 NULL 값을 허용하는지에 대한 정보를 가진다. 예를 들어 (그림 3)의 users 릴레이션의 애트리뷰트 중 id 애트리뷰트에 대하여 속성 관리기가 가지는 자료 구조는 <표 3>과 같다.

〈표 3〉 속성 관리기의 자료 구조 예

필 드	내 용
id에 대한 경로식	view("db")/users/tuple/id/text()
기본 데이터 타입	xsd : string
부가 데이터 Facets	<xsd : length value = "3" />
NULL 값 허용 여부	false

4.4 응용 XML 뷰의 필요성

응용 XML 스키마는 기본 XML 스키마를 바탕으로 사용자가 정의할 수 있는 가상 데이터베이스에 대한 스키마이다. 이는 관계형 데이터베이스에서 뷰를 정의하는 개념과 유사하며, 그 관계는 (그림 4)와 같다. 기본 XML 스키마는 관계형 데이터베이스의 시스템 카탈로그를 이용하여 자동으로 생성되는 반면, 응용 XML 스키마를 생성하기 위해서는 사용자가 XML 뷰를 정의해야 한다.



(그림 4) 관계형 데이터베이스에서의 뷰와 응용 XML 뷰와의 관계

4.5 응용 XML 뷰의 표현

(그림 3)에서 제시한 기본 XML 스키마에 대하여, 사용자가 (그림 5)와 같은 가상 XML 문서 인스턴스를 정의하고자 한다고 가정했을 때, 이에 대응하는 응용 XML 뷰는 (그림 6)과 같다.

```
<Auction>
  <Users>
    <Name ID = "U01"> Tom Jones </Name>
    <Bids>
      <ItemNumber> 1002 </ItemNumber>
      <Bid> 400 </Bid>
    </Bids>
    <Bids>
      <ItemNumber> 1004 </ItemNumber>
      <Bid> 40 </Bid>
    </Bids>
    <Rating> B </Rating>
  </Users>
  ...
  <Users>
    <Name ID = "U05"> Jack Sprat </Name>
    <Bids>
      <ItemNumber> 1003 </ItemNumber>
      <Bid> 20 </Bid>
    </Bids>
    <Bids>
      <ItemNumber> 1007 </ItemNumber>
      <Bid> 200 </Bid>
    </Bids>
```

```
        <Rating> B </Rating>
      </Users>
    </Auction>
```

(그림 5) 가상 XML 문서의 인스턴스 예

기본 XML 스키마로부터 (그림 5)와 같은 문서 인스턴스를 생성하기 위하여 뷰를 정의하는 방법은 다음과 같다. 우선 (그림 6)에서 첫 번째 줄 "FOR $x IN view("db")/users/tuple"에서 보는 바와 같이, (그림 3)에서 제시된 기본 XML 스키마에서의 'users/tuple' 해당하는 요소를 변수 '$x'로 지정한다. 그리고 Users와 그에 대응되는 속성을 생성한 후에, 각 속성의 값에는 기본 XML 스키마에 대응되는 속성값이 주어져야 한다. $x/id, $x/name, $x/rating 등은 기본 XML 스키마에서의 값을 말한다. 다음으로, 경매자의 경매 기록에 대한 XML 문서 구조를 만들기 위하여, 기본 XML 스키마의 users의 userid와 bids의 id가 일치하는 튜플을 구해야 한다. 이를 표현한 문장이 (그림 6)에서 "FOR $y IN view("db")/bids/tuple WHERE $y/userid = $x/id"이다. 여기서 생성되는 변수 '$y'는 앞에서 설명한 방법과 유사하게 표현된다.

```
<Auction>
  {
    FOR $x IN view("db")/users/tuple
    RETURN
    <Users>
     <Name ID ={$x/id/text( )}> $x/name/text() </Name>
       {
       FOR $y IN view("db")/bids/tuple
       WHERE $y/userid = $x/id
       RETURN
        <Bids>
          <ItemNumber> $y/itemno/text( ) </ItemNumber>
          <Bid> $y/bid/text( ) </Bid>
        </Bids>
       }
     <Rating> $x/rating/text( ) </Rating>
    </Users>
  }
</Auction>
```

(그림 6) 응용 XML 뷰 예

5. 뷰 트리 모델과 응용 XML 스키마 변환 알고리즘

미디에이터는 랩퍼로부터 XML 스키마를 전달받기 때문에 XQuery로 표현된 응용 XML 뷰를 XML 스키마, 즉 응용 XML 스키마로 변환해야 한다. 이를 위하여 뷰 트리 기반의 변환 모델을 제시한다.

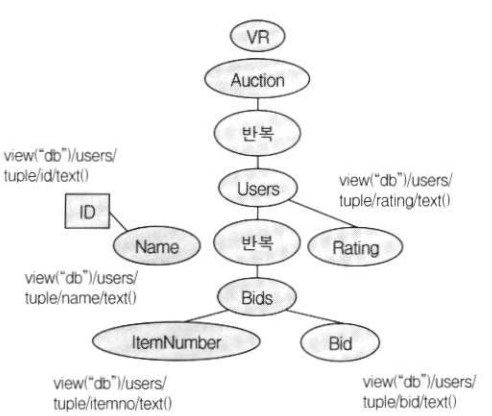
5.1 뷰 트리 모델

XQuery로 표현된 응용 XML 뷰를 트리 형태로 변환한 것을 응용 XML 뷰 트리라 정의한다. 뷰 트리의 노드는 가상 루트 노드, 반복 노드, 엘리먼트 노드로 구분된다. 뷰 트리를 구성하기 위해서 우선 하나의 루트 엘리먼트를 가지도록 가상의 루트 노드를 생성한다. 뷰 상의 각 엘리먼트는 뷰 트리의 엘리먼트 노드로 추가되며, 한 엘리먼트의 자식 엘리먼트는 대응되는 뷰 트리 노드의 자식 노드가 된다. 즉, XML 뷰를 엘리먼트의 계층 구조를 표현한 XML 문서로 간주하여, 그 계층 구조에 따라서 트리가 구성된다. 응용 XML 뷰에서 FLWR절을 만나면 이에 대응되는 반복 노드형을 트리에 추가한다. 트리의 각 노드는 <표 4>와 같은 자료 구조를 가진다.

〈표 4〉 응용 XML 뷰 트리의 노드 구조

속 성	내 용
이 름	엘리먼트 이름
타 입	노드 타입
부모 노드	현재 노드의 부모 노드 포인터
첫 번째 자식 노드	현재 노드의 첫 번째 자식 노드 포인터
다음 형제 노드	현재 노드의 다음 형제 노드 포인터
값	기본 XML 스키마상의 경로식
애트리뷰트	애트리뷰트 이름, 애트리뷰트 값

<표 4>에서 '이름' 필드는 응용 XML 뷰에 나타나는 엘리먼트의 이름을 의미한다. '타입' 필드는 트리를 구성하는 노드의 타입이 루트 노드인지 반복 노드인지 엘리먼트 노드인지를 나타낸다. '부모 노드' 필드, '첫 번째 자식 노드' 필드 그리고 '다음 형제 노드' 필드는 현재 엘리먼트를 기준으로 부모 엘리먼트, 첫 번째 자식 엘리먼트 그리고 다음 형제 엘리먼트를 의미하며 각각 해당 노드의 포인터 값을 가진다. '값' 필드는 엘리먼트 노드이면서 단말 노드일 경우 가지게 되며, 노드의 경로식이나 상수를 가지게 된다. 마지막으로 '애트리뷰트' 필드는 엘리먼트에 애트리뷰트가 존재할 경우 값을 가지며, 여기서 값은 애트리뷰트 이름과 애트리뷰트 값을 의미하며 리스트로 구성된다.



(그림 7) 응용 XML 뷰 트리 예

(그림 7)은 (그림 6)의 응용 XML 뷰를 응용 XML 뷰 트리로 변환한 예이다. 응용 XML 뷰 트리는 미디에이터 관

점의 가상 XML 인스턴스에 대한 구조 정보를 나타내며 트리의 각 노드들은 XML 스키마로 변환시 엘리먼트로 변환된다. (그림 6)의 응용 XML 뷰상의 엘리먼트들은 모두 트리의 노드로 변환되고 FLWR 절이 나타난 곳은 반복 노드가 추가됨을 알 수 있다. 뷰 트리상에는 응용 XML 뷰에 나타나는 조건절에 대한 정보가 포함되지 않았음을 알 수 있는데, 이는 응용 XML 뷰에 의해 생성될 응용 XML 스키마는 XML 인스턴스가 포함하는 내용이 아니라 인스턴스에 대한 스키마 정보이므로, 응용 XML 뷰에 나타나는 인스턴스가 포함하는 내용에 대한 조건은 문서의 구조 정보에 영향을 미치지 못하기 때문이다.

5.2 응용 XML 스키마 생성 알고리즘

응용 XML 스키마 생성기는 응용 XML 뷰 트리의 가상 루트를 시작으로 깊이 우선 탐색으로 트리를 순회하며 응용 XML 스키마를 생성한다. (그림 8)은 응용 XML 뷰 트리로부터 응용 XML 스키마를 생성하는 알고리즘이다.

```
createAppSchema(StringBuffer doc, XQueryNode node) {
  XQueryNode child = node.getfirstChild( )
  short type = node.getType( )

  if (child != NULL) {
    if (type == ELEMENT_NODE) {
      String name = node.getName( )
      doc.append("<element>")
      Call addElement2Schema(doc, name)
      doc.append("<complexType>")
      doc.append("<xsd : sequence>")
    }
    else if (type == REPETITION_NODE) {
      doc.append("<xsd : sequence>")
      Call addRepetition2Schema(doc)
    }

    Call createAppSchema(doc, child)

    if (type == ELEMENT_NODE) {
      doc.append("</xsd : sequence>")
      Call addAttr2Schema(doc, (XQueryElement)node)
      doc.append("</complexType>")
      doc.append("</element>")
    }
    else if (type == REPETITION_NODE)
      doc.append("</xsd : sequence>")
    else
      return
  }
  else
    if (type == ELEMENT_NODE)
      Call addElement2Schema(doc, (XQueryElement)node)
  }

  XQueryNode sibling = node.getSibling( )

  if (sibling != NULL)
    Call createAppSchema(doc, sibling)
}
```

(그림 8) 응용 XML 스키마 생성 알고리즘

응용 XML 뷰 트리는 가상 XML 인스턴스의 구조 정보를 가지고 있으므로 뷰 트리를 깊이 우선으로 순회하며 구조 정보를 바탕으로 응용 XML 스키마를 생성한다. 응용 XML 스키마를 생성하기 위해서는 구조정보 이외에 엘리먼트에 대한 데이터 타입과 NULL 허용 여부에 관한 부가 정보가 필요하며 이를 위해서 4.3절에서 언급한 속성 관리기를 이용한다. 뷰 트리상의 단말 노드나 노드가 가지는 애트리뷰트의 경우에 속성 관리기로부터 필요한 부가 정보를 얻게 된다.

```
if (path_expression != SIMPLE_PATH) {
  if (path_expression has a FUNCTION_EXPRESS) {
    date_type = compareFunction(path_expression)
  }
  else {
    if (path_expression has a FIXED_NUMBER)
      date_type = "String"
    else
      date_type = "Empty"
    }
  }
else {
  date_type = comparePath(path_expression)
}
```

(그림 9) 응용 XML 스키마 데이터 타입 결정 알고리즘

(그림 9)는 응용 XML 스키마 생성에 필요한 데이터 타입을 결정하는 알고리즘이다. 변환된 뷰 트리상의 노드가 가지는 값이 단순한 경로식일 경우, 속성 관리기에 저장되어 있는 기본 XML 스키마상의 경로식과 비교하여, 서로 일치하는 엘리먼트의 데이터 타입을 뷰 트리 노드의 데이터 타입으로 결정한다. 뷰 트리 노드상의 값과 속성 관리기의 경로식이 일치하더라도 XQuery에서 지원하는 집단 함수(avg, count 등)에 따라 응용 XML 스키마상에 나타나는 데이터 타입은 변경될 수 있으며, 뷰 트리 노드상의 값이 사용자가 새롭게 정의한 상수값일 경우 모두 String형으로 간주한다.

6. 구 현

본 장에서는 스키마 관리기의 구현 결과를 응용 프로그램 인터페이스를 중심으로 논한다.

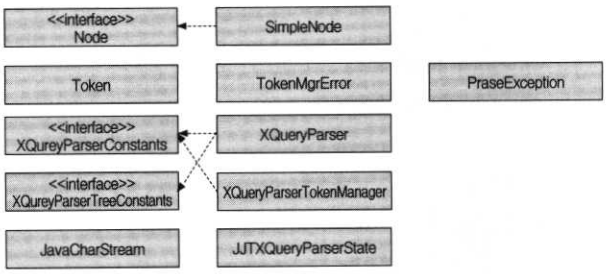
6.1 구현 환경

본 논문에서 제시하는 스키마 관리기는 자바로 구현되었다. 개발 도구는 JDK 1.3을 사용하였고 윈도우 2000 Server를 운영체제로 사용하였다. 구현의 예로 든 관계형 데이터베이스는 Microsoft SQL Server 2000을 이용하였고 해당 데이터베이스와의 연결을 위한 JDBC는 SQL Server 2000 Driver for JDBC(Version 2.1)를 이용하였다. XQuery 파서를 생성하기 위한 도구로써 JavaCC 2.1을 이용하였다.

6.2 뷰 트리 구현

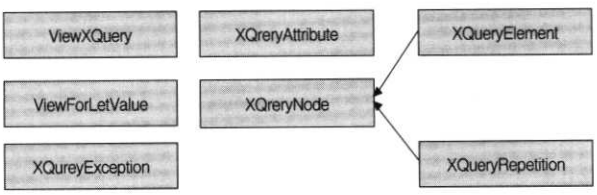
XQuery로 기술된 응용 XML 뷰를 뷰 트리로 변환하기

위해서는 XQuery 파서가 필요한데, 이를 위해 파서 생성기 JavaCC를 이용하여 XQuery 파서를 생성하였다. JavaCC는 XQuery를 위한 EBNF 표기법을 JavaCC 포맷에 맞게 변경하면 자동으로 파서를 생성한다. 생성된 파서는 XQuery를 입력받아 파스 트리를 생성한다.



(그림 10) JavaCC에 의해 생성된 클래스 다이어그램

(그림 10)은 JavaCC에 의해서 생성된 클래스 다이어그램이다. Node 인터페이스는 파스 트리를 구성하는 노드의 기능을 선언하며, SimpleNode 클래스는 그 기능을 구현한다. Token 클래스는 어휘 분석기에서 처리되는 토큰을 나타내며, TokenMgrErr 클래스는 토큰 관리자에서 발생하는 에러를 나타낸다. XQueryParserConstants 인터페이스에는 파서에서 사용되는 상수들이 정의되어 있으며, XQueryParserTreeConstants 인터페이스에는 파스 트리에서 사용되는 상수들이 정의되어 있다. XQueryParser 클래스는 파서 드라이버 기능을 수행하며, XQueryParserTokenManager는 어휘 분석기의 기능을 수행한다.



(그림 11) 뷰 트리 생성 클래스 다이어그램

(그림 11)은 파스 트리로부터 뷰 트리 생성기를 구현한 클래스 다이어그램이다. ViewXQuery 클래스는 XQuery를 파싱하여 생성된 파스 트리를 순회하며 실제 뷰 트리를 구현하는 역할을 한다. ViewForLetValue 클래스는 파스 트리를 순회할때 변수들에 대한 처리를 한다. XQueryNode 클래스는 각 노드형의 공통된 속성과 연산을 가지는 클래스이고, 이를 상속받은 XQueryElement 클래스와 XQueryRepetition 클래스는 엘리먼트형 노드와 반복형 노드의 고유의 속성과 연산을 정의한다. 뷰 트리 상에 나타나는 애트리뷰트는 XQueryAttribute 클래스에서 처리된다. 발생하는 예외사항들은 XQueryException 클래스에서 처리된다.

6.3 응용 프로그램 인터페이스

스키마 관리기는 미디에이터가 사용할 수 있는 API 형태로 구현되었다.

```
public interface SchemaManager {
    public String getBaseSchema( ) ;
    public void updateBaseSchema( ) ;
    public void registerView(String name, String view) ;
    public String[ ] getViewNames( ) ;
    public String getAppSchema(String name) ;
    public String getAppView(String name) ;
    public void renameView(String oldName, String newName) ;
    public void updateView(String name, String view) ;
    public void dropView(String name) ;
}
```

(그림 12) 스키마 관리기 API

(그림 12)는 랩퍼의 스키마 관리기 사용자 API이다. getBaseSchema는 생성된 기본 XML 스키마를 반환하는 기능을 제공한다. updateBaseSchema는 생성된 기본 XML 스키마를 수정하는 메소드이다. 기본 XML 스키마의 내용은 지역 데이터베이스의 스키마로부터 자동으로 생성되며, 지역 데이터베이스의 스키마가 변경되었을 때 사용한다. registerView는 새로운 응용 XML 뷰를 등록하는 메소드이다. 이때 등록하고자 하는 응용 XML 뷰와 뷰의 이름이 필요하다. 응용 XML 뷰를 등록하면 (그림 11)에서 제시된 클래스를 이용하여 뷰 트리를 구성한 후 응용 XML 스키마를 자동으로 생성하여 관리한다. 응용 XML 스키마의 이름은 응용 XML 뷰의 이름과 동일하게 관리된다. getViewNames는 사용자가 등록한 모든 응용 XML 뷰의 이름을 반환한다. getAppSchema는 사용자가 이미 등록한 응용 XML 스키마를 반환하는 기능을 제공한다. getAppView는 사용자가 이미 등록한 XML 뷰를 반환하는 기능을 제공한다. renameView는 등록된 응용 XML 뷰의 이름을 변경하는 기능을 제공한다. 이때 같은 이름으로 등록된 응용 XML 스키마의 이름도 같이 변경된다. updateView는 등록되어 있는 기본 XML 뷰를 수정하는 기능을 제공한다. 변경하고자 하는 응용 XML 뷰의 이름과 새로운 응용 XML 뷰가 필요하며 같은 이름의 응용 XML 스키마도 함께 변경된다. dropView는 등록된 응용 XML 뷰를 삭제하는 기능을 제공한다. 이 경우 같은 이름으로 등록된 응용 XML 스키마도 함께 삭제된다.

7. 결론 및 향후 과제

미디에이터 기반의 데이터베이스 통합 방법론에서, 스키마를 관리하는 자료 저장소 랩퍼는 핵심 요소 중의 하나이다. 본 논문에서는 XML을 기반으로 이질의 분산 데이터베이스 통합을 위한 랩퍼의 XML 스키마 관리기를 설계, 구현하였다. 특히 응용 XML 뷰에 대한 뷰 트리 모델을 제시하고 이를 기반으로 응용 XML 뷰를 응용 XML 스키마로 실체화시키는 알고리즘을 제시하였는데, 이 모델은 다양한 이질의 정보원에 대한 스키마 관리기와 질의 처리를 위하여 일관되게 적용될 수 있다. 앞으로 계속해서 응용 XML

뷰 기반의 질의 처리 모델에 대한 연구와 함께 기본 XML 뷰의 정의가 질의 처리기와 스키마 관리에 미치는 영향에 대한 연구가 더 진행되어야 한다.

참 고 문 헌

[1] M. Fernandez, W. Tan, and D. Suciu, "SilkRoute : Trading between Relations and XML," WWW9, pp.723-745, 2000.

[2] J. Shanmugasundaram, J. Kiernan, E. Shekita, C. Fan, and J. Funderburk, "Querying XML Views of Relational Data," VLDB Conference, pp.261-270, 2001.

[3] M. Carey, D. Florescu, Z. Ives, Y. Lu, J. Shanmugasundaram, E. Shekita, and S. Subramanian, "XPERANTO : Publishing Object-Relational Data as XML," Workshop on the Web and Databases (WebDB), pp.105-110, May, 2000.

[4] J. Shanmugasundaram, E. Shekita, J. Kiernan, R. Krishnamurthy, E. Viglas, J. Naughton, and I. Tatarinov, "A General Technique for Querying XML Documents using a Relational Database System," SIGMOD Record 30(3), pp.20-26, September, 2001.

[5] Chaitan Baru, Amarnath Gupta, Bertram Ludaescher, Richard Marciano, Yannis Papakonstantinou, and Pavel Velikhov, "XML-Based Information Mediation with MIX," SIGMOD Conference 1999, pp.597-599, 1999.

[6] Chaitanya K. Baru, "XViews : XML Views of Relational Schemas," DEXA Workshop 1999, pp.700-705, 1999.

[7] Panchapagesan, Hui, Wiederhold G., Erickson S., Dean L., and Hempstead A., "The INEEL Data Integration Mediation System," International ICSC Symposium on Advances in Intelligent Data Analysis (AIDA 99), June, 1999.

[8] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom, "The TSIMMIS Approach to Mediation : Data Models and Languages," JIIS 8(2), pp.117-132, 1997.

[9] World-Wide Web Consortium, "XML Schema Part 0 : Primer," http://www.w3.org/TR/xmlschema-0/.

[10] World-Wide Web Consortium, "XQuery 1.0 : An XML Query Language," http://www.w3.org/TR/xquery/.

정 채 영

e-mail : wcdi@rtp.gsnu.ac.kr

1997년 경상대학교 전자계산학과 학사

2001년 경상대학교 대학원 컴퓨터과학과 석사

2001년~현재 경상대학교 대학원 컴퓨터과학과 박사 과정

관심분야 : XML, WWW, 데이터베이스, 정보검색

김 영 옥

e-mail : yokim@kornet.net

1988년 울산대학교 전자계산학과 학사

1993년 동아대학교 산업대학원 컴퓨터공학과 석사

2002년 경상대학교 대학원 컴퓨터과학과 박사과정 수료

현재 진주제일병원 전산실장 재직

관심분야 : XML, 데이터베이스 통합, 데이터마이닝

이 미 영

e-mail : mylee@etri.re.kr

1981년 서울대학교 식품영양학과(이학사, 계산통계학 부전공)

1983년 서울대학교 대학원 계산통계학과 (이학석사)

1983년~1985년 한국전기통신연구소

1988년~현재 한국전자통신연구원 컴퓨터소프트웨어연구소 책임연구원

관심분야 : 데이터베이스 시스템, XML 문서관리, 정보통합, 바이오인포매틱스

강 현 석

e-mail : hskang@nongae.gsnu.ac.kr

1981년 동국대학교 전자계산학과 학사

1983년 서울대학교 계산통계학과 석사 (전산학)

1989년 서울대학교 계산통계학과 박사 (전산학)

1984년~1993년 전북대학교 전자계산학과 부교수

1993년~현재 경상대학교 컴퓨터과학과 교수

관심분야 : 객체지향 데이터베이스, 전자문서 관리, 멀티미디어

배 종 민

e-mail : jmbae@nongae.gsnu.ac.kr

1980년 서울대학교 수학교육과 학사

1983년 서울대학교 대학원 계산통계학과 석사

1995년 서울대학교 대학원 계산통계학과 박사

1982년~1984년 한국전자통신연구소 연구원

1997년~1998년 Virginia Tech. 객원연구원

1984년~현재 경상대학교 전임강사, 조교수, 부교수, 교수

관심분야 : XML 데이터베이스 통합, 정보검색, 디지털 라이브러리, 데이터마이닝