SQL/XML의 출판 함수를 이용한 관계 데이터의 XML 뷰 정의 및 처리

이 상 욱* · 김 진** · 강 현 철***

Defining and Processing XML View of Relational Data with Publication Functions of SQL/XML

Sangwook Lee* · Jin Kim** · Hyunchul Kang***

Abstract

Since XML emerged as a standard for data exchange on the web, it has been widely used for applications like e-Commerce, CRM, and BI. However, it is common that most of business data is stored in relational database systems, and it is expected that business data management would still be centered around the relational database systems. As such, the technique of viewing relational data as XML and processing XML queries against it is required. To meet such a need, in the SQL/XML standard, the functions to publish relational data as XML are provided. In this paper, we propose the techniques of providing an XML view of relational data defined by an SQL/XML statement in DTD(Document Type Definition), and of processing XPath queries against the XML view by translating them into SQL/XML statements, and describe the validation of such techniques through implementation and tests.

Keywords : XML Publication, SQL/XML, XML view, DTD, XML Database, Business Data

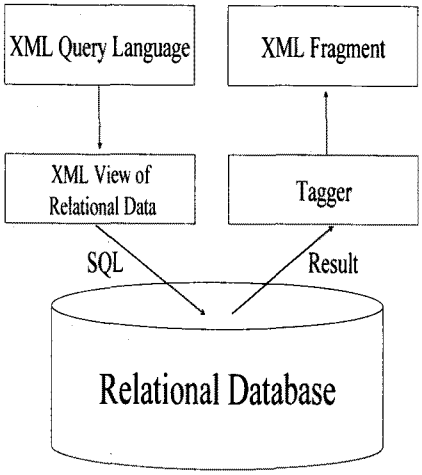
논문접수일 : 2009년 10월 23일 논문게재확정일 : 2009년 12월 22일
※ 이 논문은 2008년도 중앙대학교 학술연구비 지원에 의한 것임.
* 중앙대학교 컴퓨터공학부 대학원, e-mail : swlee@dblab.cse.cau.ac.kr
** 중앙대학교 컴퓨터공학부 대학원, e-mail : jinkim@dblab.cse.cau.ac.kr
*** 교신저자, 중앙대학교 컴퓨터공학부 교수, e-mail : hckang@cau.ac.kr

1. 서 론

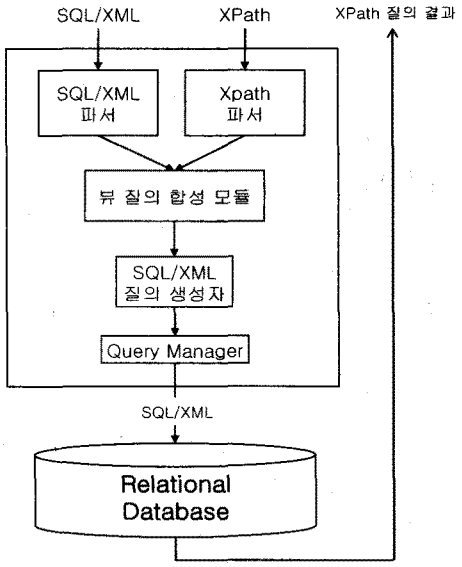
XML이 웹에서 데이터 교환의 표준으로 부각된 이래 전자거래, CRM(customer relationship management), BI(business intelligence) 등의 응용 분야에서 XML 데이터의 효율적인 처리에 관한 연구가 활발히 이루어지고 있다. 하지만 현재까지 대부분의 비즈니스 원시 데이터는 관계 데이터베이스 시스템에 저장되어 있으며, 앞으로도 관계 데이터베이스 시스템은 전사적 비즈니스 정보 시스템 구축의 근간이 될 것임에는 변함이 없을 것으로 예측된다. 그러나 웹 기반의 e-비지니스 관련 기술들이 표준 데이터 포맷으로 XML을 사용하는 등 XML의 활용 범위는 계속 확대되고 있는 추세이므로, XML을 기반으로 하는 비즈니스 응용에서 활용할 수 있도록 관계 데이터베이스에 저장된 비즈니스 데이터를 XML 문서로 보고 이에 대해 질의 처리를 수행할 수 있는 기법은 비지니스 정보 및 의사결정 지원 시스템의 구현에 있어 필수적인 요소이다.

이러한 관계 데이터베이스와 XML의 연동에 대한 요구는 2000년대 초반부터 주목을 받아 관계 데이터를 XML로 출판하는 기법을 개발하기 위해 XPERANTO[Carey외, 2000a; Carey외, 2000b; Shanmugasundaram외, 2000; Shanmugasundaram외, 2001a], SilkRoute[Fernandez외, 2000; Fernandez외, 2001a; Fernandez외, 2001b] 등의 프로젝트가 수행되었고, SQL/XML 표준[ISO/IEC; Melton and Buxton, 2006]에서는 관계 데이터베이스 시스템에서 XML을 사용할 수 있도록 지원하는 기능들이 제공되고 있다. SQL/XML은 관계 데이터베이스에 XML 데이터를 저장하거나, XML 질의어인 XPath나 XQuery를 통해 XML 데이터에 대해 질의 처리하거나, 기존에 저장되어 있는 관계 데이터를 XML의 형식으로 출판하는 것 등을 가능하게 한다. 이와 같은 SQL/XML 표준의 등장에 따라 관계 데이터베이스 시스템에서 XML 질의 처리는 XML 형(type)으로 저장된 데이터에 대해서 뿐만 아니라, 기존의 순수 관계 데이터를 XML 데이터로 출판된 것으로 보고 즉, 관계 데이터에 대한 XML 뷰를 제공하고, 이에 대해 XML 질의를 처리하는 것이 가능하다. 본 논문은 SQL/XML의 출판 함수들을 사용하여 순수 관계 데이터에 대해 정의된 XML 뷰를 XML 스키마를 명시하기 위해 널리 사용되고 있는 DTD(Document Type Definition)로 제공하고 이에 대해 XPath로 질의를 제기했을 때 이를 처리하는 기술에 관한 것이다.

관계 데이터를 XML로 보고 질의 처리하는 기존의 대표적인 기법들은 SQL을 매개로 하여 관계 데이터에 대한 XML 뷰를 생성하여 사용자에게 제공하고, 사용자는 이 XML 뷰에 대해 XML 질의어로 질의하는 것이다[Carey외, 2000a; Carey외, 2000b; Shanmugasundaram외, 2000; Shanmugasundaram외, 2001a; Fernandez외, 2000; Fernandez외, 2001a; Fernandez외, 2001b]. <그림 1>은 이와 같은 과정을 나타낸 것이다. 관계 데이터의 XML 뷰에 대해 XML 질의어를 통해 질의하게 되면, 이 질의는 SQL로 변환되어 하부의 관계 데이터베이스 시스템에서 처리된다. SQL 질의의 결과는 관계 튜플들로 구성된 결과 집합(result set)이므로 이는 적절한 XML 태깅(XML tagging)을 거쳐서 XML 조각(XML fragment)으로 변환되고 변환된 조각은 XML 질의의 결과로서 반환된다. 이러한 기존 기술의 문제점은, 복잡한 뷰의 정의 시 사용자 정의 함수를 사용해야하는 절차를 거쳐야 하고, 뷰 질의의 결과인 관계 데이터를 XML로 변환하는 작업을 DBMS 상층의 응용 또는 미들웨어가 담당해야 하는 부담 및 비효율성이 존재한다는 것



〈그림 1〉 기존의 관계 데이터에 대한 SQL 기반 XML 질의 처리



〈그림 2〉 관계 데이터에 대한 SQL/XML 기반의 XML 질의 처리

〈표 1〉 기존 기법과 본 논문의 기법 비교

항목	기존 기법	본 논문의 기법
SQL 표준	SQL92, SQL99	SQL/XML(SQL2003 이후 사양)
뷰 정의 복잡도	사용자 정의 함수 등을 이용	SQL/XML의 내장 출판함수 이용
뷰 질의 변환	SQL 문	SQL/XML 문
RDBMS의 뷰 질의 결과	관계 결과 집합	XML
XML 태깅 작업	필요하며 RDBMS 상층의 응용 또는 미들웨어가 담당	불필요

등이다. 전찬훈[2007]의 연구에 의하면, 관계 데이터를 대상으로 XML 질의를 처리하는 데 있어 결과의 XML 변환을 위한 태깅 작업이 질의 처리 성능에 미치는 영향은 아주 큰 것으로 나타났다.

본 논문에서 제안하는 기법에서는 이러한 전통적인 방법과 달리 SQL/XML 표준에서 명시한 XMLELEMENT(), XMLATTRIBUTES(), XMLAGG(), XMLFOREST(), XMLCONCAT() 등의 XML 출판 함수들을 이용하여 관계 데이터에 대한 XML 뷰를 정의하고, 이에 대한 XPath 질의는 SQL이 아닌 SQL/XML로 변환되어 하부의 관계 DBMS에서 처리된다(<그림 2> 참조). 따라서 XML 뷰 정의가 용이해져서 복잡도가 높은 뷰의 생성도 편리하게 수행할 수 있다. 또한 SQL/XML 문은 그 결과를 XML로 반환하므로 기존 SQL 기반 시스템의 경우 수행해야 했던 관계 결과 집합에 대한 XML 태깅 작업도 필요 없다. <표 1>은 기존의 기법과 본 논문에서 제시하는 기법을 비교한 것이다.

오늘날의 비즈니스 정보 및 의사 결정 지원 응용들이 웹 환경에서 다양한 데이타 소스 및 응용들과 연동하면서 표준 데이터 포맷으로 XML을 사용하고 있다. 예를 들어, 마이크로소프트사의 SQL Server Analysis Services와 같은 BI 분석 툴은 대량의 관계 데이터를 처리하고 XML for Analysis와 같은 API를 제공한다. LogiXML 사의 Logi Report와 같은 웹 기반의 BI 보고 툴은 XML을 기반으로 하고 있다. 또한 많은 CRM 솔루션들은 기능적인 유연성과 확장성을 위해 XML API를 제공하며, XML 웹 서비스 기술과 연동하여 CRM 시스템으로/으로부터 데이터를 직접 저장하거나 추출하는 기능을 제공한다. 이에 반해 방대한 비즈니스 원시 데이터의 저장 및 관리는 대부분 표준 관계 데이터베이스 시스템을 근간으로 하고 있다. 본 논문이 제시하는 기법은 이러한 괴리 및 제약을 보다 효율적으로 극복하기 위해 필요하며 따라서 비즈니스 실무에서 중요성을 갖는다.

본 논문에서 제시하는 기법을 실현하기 위해서는 기술적인 문제로서 아래 두 가지를 해결해야 한다.

- 일반적으로 SQL/XML의 출판 함수들을 사용하여 관계 데이터에 대한 XML 뷰를 정의한 경우(보통 여러 함수들이 중첩적으로 사용됨), 그 SQL/XML 문의 복잡성으로 인해 사용자가 XML 뷰를 이해하는데 난점이 있을 수 있다. 따라서 SQL/XML 문으로 정의된 XML 뷰를, XML 스키마를 규정하는 DTD로 자동 변환하여 제시하는 것이 바람직하다. 예를 들어, Melton and Buxton[2006]의 movie 관계 데이터베이스 스키마를 고려하자. <그림 3>은 이 관계 데이터베이스의 movies 테이블, 그것에 대한 XML 뷰를 정의하는 SQL/XML 문, 그리고 그 결과로 반환되는 XML 데이타를 나타낸 것이다. SQL/XML 문에서는 XMLELEMENT(), XMLAGG(), XMLFOREST()의 세 함수가 중첩적으로 사용되고 있다. 일반적으로 SQL/XML 문은 이 예의 경우보다 더 복잡할 수 있는데 이와 같이 정의된 XML 뷰의 모습을 해당 SQL/XML 구문만 보고 직관적으로 명확하게 파악하는 것은 용이하지 않으므로, XML 데이터의 계층적인 구조 등을 보다 쉽게 제시할 수 있는 DTD 등으로 자동 변환하는 것이 필요하다. <그림 4>는 <그림 3>의 SQL/XML 문으로부터 도출한 DTD를 나타낸 것으로 XML 뷰를 쉽게 이해하게 해준다.
- SQL/XML 문으로 정의된 XML 뷰에 대해 제기된 XPath 질의를 처리하기 위해서는 뷰를 정의한 SQL/XML 문과 XPath 식을 합성하여 질의의 결과를 반환하는 새로운 SQL/XML 문을 생성해야 한다. 이 과정에서 하부의 관계 데이타베이스 스키마 정보도 참조되어야 한다. 이 과정은 SQL/XML 표준에서 명시한 XML 출판 함수들의 기능 및 특성을 바탕으로 기존의 관계 데이타베이스 및 SQL 기반의 XML 출판 연구들에서 제시된 개념 및 기법의 확장을 요한다. <그림 4>는 <그림 3>의 DTD (즉, <그림 2>의 SQL/XML 문으로 정의한 XML 뷰)에 대해 XPath 식 /movie_information/movie_details[title = 'titanic']/yearReleased 이 제기되었을 때 이를 처리하기 위해 XML 뷰와 합성되어 생성된 SQL/XML 문을 나타낸 것이다.

본 논문에서는 이 두 가지 문제를 해결하기 위한 기법을 제시하고, 구현 및 테스트를 통한

ID	Title	yearReleased	Director	runningTime
42	An American Werewolf in London	1981	John Landis	98
43	Animal House	1978	John Landis	109

```
SELECT
   XMLELEMENT(NAME "movie-information",
      XMLAGG(
         XMLELEMENT(NAME "movie-details",
            XMLFOREST(title,yearReleased,runningTime))))
FROM movies
```

```
<movie-information>
   <movie-details>
      <title>
         An American Werewolf in London
      </title>
      <yearReleased>
         1981
      </yearReleased>
      <runningTime>
         98
      </runningTime>
   </movie-details>
   <movie-details>
      ...
   </movie-details>
</movie-information>
```

〈그림 3〉 관계 데이터베이스 테이블과 XML 뷰 정의

```
<!ELEMENT movie-information (movie-details*)>
<!ELEMENT movie-details (title, yearReleased, runningTime)>
<!ELEMENT title(#PCDATA)>
<!ELEMENT yearReleased(#PCDATA)>
<!ELEMENT runningTime(#PCDATA)>
```

〈그림 4〉 〈그림 3〉의 XML 뷰를 정의하는 DTD

```
SELECT XMLELEMENT(NAME "yearReleased", yearReleased)
FROM   movies
WHERE  title = "titanic"
```

〈그림 5〉 XPath 식을 변환한 SQL/XML 문

기능 검증 결과를 기술한다. 본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구를 살펴본다. 제 3장에서는 관계 데이터의 XML 뷰를 정의한 SQL/XML 문으로부터 DTD를 생성하는 기법 및 XML 뷰에 대한 XPath 질의 처리 기법에 대해 기술한다. 제 4장에서는 구현 및 테스트 결과를 기술하고, 마지막으로 제 5장에서는 결론을 맺고 향후 연구 내용을 기술한다.

2. 관련 연구

관계 데이타베이스 시스템은 비즈니스 응용 분야에서 그동안 널리 확산되어 사용되어 왔기 때문에 XML이 웹에서 데이터 교환의 표준으로 등장한 이후에도 가장 대표적인 XML 저장소로 간주되고 있다. 관계 데이터 모델은 수평(flat) 구조를 기반으로 하지만 XML은 반구조적(semistructured), 계층 및 중첩적인 특성을 가지므로 이런 차이를 고려하여 XML 데이터를 분할하고 관계 데이터베이스에 저장 및 질의 처리하는 연구가 먼저 큰 주목을 받았다[Schmidt, 2000; Shanmugasundaram외, 2001b; Yoshikawa외 3인, 2001]. 또한 관계 데이터의 XML 출판 및 XML로 출판된 관계 데이터에 대해 XML 질의

를 제기하여 처리하는 방법도 중요한 기술로 부각되었는데 가장 대표적인 프로젝트로는 XPERANTO[Carey외, 2000a; Carey외, 2000b; Shanmugasundaram외, 2000; Shanmugasundaram외, 2001a]와 SilkRoute[Fernandez외, 2000; Fernandez외, 2001a; Fernandez외, 2001b]가 있다. XPERANTO 프로젝트에서는 순수 관계 데이터 뿐만 아니라 객체-관계 데이터도 고려하였다[Carey외, 2000a; Carey외, 2000b]. 한편 관계 데이터베이스의 질의어인 SQL 표준화에서도 SQL과 XML 기능을 연동시키는 소위 SQL*A* 사양 제정을 위해 SQL : 2003, SQL : 2005, SQL : 2007 등 표준 사양들이 계속 발표되고 있다[ISO/IEC; Melton and Buxton, 2006].

[Shanmugasundaram외, 2001a]에서는 관계 데이터에 대한 XML 뷰를 정의하기 위해 기본(default) XML 뷰를 먼저 생성한다. 기본 XML 뷰를 만드는 이유는 관계 데이터에 대해 XQuery 등의 XML 질의 언어를 통해 직접 XML 뷰를 생성할 수 없기 때문이다. 사용자는 기본 XML 뷰에 XQuery를 통해 XML 뷰를 정의하게 된다. 생성된 XML 뷰에 대해 사용자는 XQuery를 통해 질의를 제기할 수 있다. 사용자 질의 구문은 뷰 정의 구문과 합성되어 SQL 질의로 변환된다. 변환된 SQL 질의는 관계 데이터베이스에서 처리되고 SQL 질의의 결과로 반환된 튜플들은 XML 태거(XML tagger)를 통해 XML 데이타로 변환되어 최종 결과로 반환된다. [Fernandez외, 2000]에서는 RXL을 사용하여 XML 뷰를 정의한다. RXL은 관계 데이터를 XML로 변환하는 언어이다. RXL을 통해 뷰가 정의되면, 사용자는 XML 질의 언어인 XML- QL로 질의를 제기한다. RXL을 사용한 뷰 정의 구문과 XML-QL 질의문은 합성되어 새로운 RXL 구문이 생성된다. 생성된 RXL 구문은 SQL 문으로 변환되어 관계 데이터베이스에서 처리되고, [Shanmugasundaram외, 2001a]에서와 유사하게 SQL 질의의 결과로 반환된 튜플들은 XML로 변환되어 결과로 출력된다.

이들 두 프로젝트에서 공통적으로 확립한 기술은 관계 데이터베이스에 대해 XML 뷰를 정의하기 위하여 소스 관계 데이터가 XML 형식으로 우선 변환되어야 한다는 점과, XML 뷰에 대한 XML 질의를 뷰 정의 구문과 합성하여 SQL로 변환한 후 관계 DBMS에서 처리하고, 그 결과 튜플들을 XML로 태깅한다는 점이다. 이러한 방식은 관계 데이터와 XML 간의 변환 작업 부담을 응용에서 소화해야 하는 근본적인 문제를 갖는다. 본 논문에서는 관계 데이터에 대한 XML 뷰 정의에 있어 소스 관계 데이터를 다른 형식으로 변환하지 않고 SQL/XML 표준의 출판 함수들을 이용하며, 관계 데이타에 대한 XML 질의 처리에서도 뷰 질의 합성 결과로 SQL/XML 문을 생성해 내므로 XML 태깅 부담을 응용에서 피하도록 하는 기법을 다룬다.

3. 관계 데이터의 XML 뷰 정의 및 처리

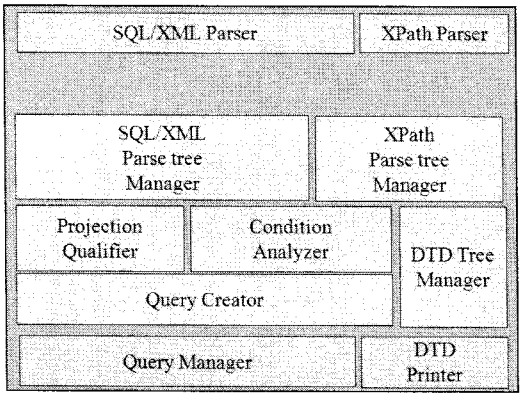
<그림 6>은 관계 데이타의 XML 뷰를 SQL/XML의 출판 함수들을 이용하여 정의하고, 이를 DTD로 사용자에게 제공하여, 이에 대한 XPath 질의를 처리하는 기능을 제공하기 위한 시스템 모듈 구성을 나타낸 것이다. 이들 모듈들은 크게 SQL/XML 문으로 정의된 XML 뷰를 DTD로 변환하기 위한 모듈들과 그 DTD를 바탕으로 제기된 XPath 질의를 새로운 SQL/XML문으로 변환하기 위한 모듈들로 나눌 수 있다. 본 절에서는 이들의 기능을 기술한다.

3.1 SQL/XML 문으로부터 DTD 생성

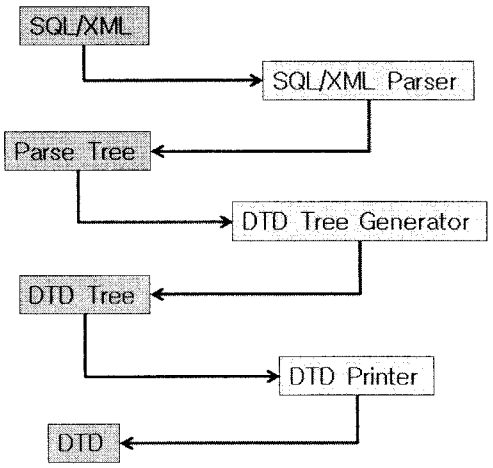
SQL/XML 출판함수들을 이용하여 정의된 관

계 데이터의 XML 뷰를 DTD로 제공하는 것은 일련의 변환 과정을 통해 가능하다. <그림 7>은 이 과정의 개요를 나타낸 것이다. 먼저 XML 뷰를 정의하는 SQL/XML 문은 SQL/XML 파서에 의해 파스 트리로 변환된다. 이는 DTD 트리 생성기에 의해 DTD 트리로 변환되고, 이는 DTD 출력기에 의해 DTD로 변환된다.

SQL/XML 문의 파스 트리는 SQL/XML 문의 SELECT 절에 존재하는 XML 출판 함수를 표현하는 노드를 루트로 하는 서브 트리들로 구성된다. XML 출판 함수들은 보통 중첩되어 나타나므로 각 함수에 해당하는 서브 트리들이 계층적으로 연결되어 파스 트리를 구성하게 된다. 또한 각 출판 함수 노드는 해당 함수의 인자 구성 방법에 의해 관계 데이터베이스의 테이블 컬럼에 상응되는 노드 등의 자식 노드를 갖는다. 예를 들어, <그림 8>의 상단 두 상자는 각각 <그림 3>의 SQL/XML 문과 이를 파싱하여 얻은 파스 트리를 나타낸 것이다. 파스 트리를 DTD로 변환하는 방법은 여러 가지가 존재할 수 있으나 SQL/XML 문으로 정의된 XML 뷰에 대한 질의 처리 과정을 효율적으로 지원할 수 있는 자료구조를 기반으로 하는 것이 바람직하다. 본 논문에서는 이를 위해 XSTM(XML Structure Tree Model)이라는 자료구조를 고안하였다.



<그림 6> 시스템 모듈 구성



<그림 7> SQL/XML문의 DTD 변환

(1) XSTM

관계 데이터베이스에서는 수평 구조의 테이블에 데이터를 저장하지만 이와는 달리 XML은 구조 정보를 나타내는 태그를 중첩하여 데이터를 표현한다. 관계 데이터에 대해 XML 뷰를 생성하기 위해 SQL/XML 출판 함수를 사용할 경우, 이는 XML 뷰가 생성되기 위해 필요한 구조 정보를 SQL/XML 출판 함수가 포함하고 있다는 것을 의미한다. 따라서 SQL/XML 문을 파싱하여 얻은 파스 트리의 각 출판 함수 노드는 자신의 자식 노드들을 해석할 수 있으므로 이로부터 XSTM을 생성한다. 즉, SQL/XML 문으로 정의된 관계 데이터의 XML 뷰에 던져진 XPath 질의를 처리하기 위해서는 XML 뷰를 생성한 SQL/XML 문과 XPath 식을 합성하여 새로운 단일 SQL/XML 문을 생성하는데, XSTM은 이러한 뷰 합성을 위한 전 단계에서 XML 뷰 정의에 사용된 SQL/XML 문의 파스 트리로부터 생성되는 것으로 XML 뷰의 구조 정보 및 질의 합성에 필요한 정보를 나타내는 자료구조이다. XSTM은 SQL/XML 문으로부터 생성되는 XML 뷰의 구조를 트리 형태로 나타내어 XPath 질의 처리에 용이하도록 설계되었다.

XSTM의 각 노드는 <표 2>와 같은 필드로 구성된다. '이름' 필드에는 노드 타입에 따라 적절한 이름이 기록된다. '노드 타입' 필드에는 노드의 타입이 기록된다. 노드 타입은 엘리먼트 타입(Elem), RDB 컬럼 타입(Col), SQL/XML 출판 함수 타입(SX)으로 나누어진다. 각각의 노드 타입과 그 의미는 <표 3>에 나타내었다.

⑵ DTD 변환

<그림 8>은 뷰 생성 SQL/XML 구문과 그에 따른 파스 트리 및 이로부터 생성되는 XSTM의 예를 나타낸 것이다. 그림 8에서 볼 수 있는 것과 같이 XSTM 트리의 모양은 SQL/XML 문으로 생성된 XML 뷰의 내용인 XML 문서가 준수하는 DTD를 트리 형태로 나타낸 것과 유사하며(<그림 4>참조) 이에 뷰 합성에 필요한 부가적인 정보를 추가한 것이다. DTD 역시 계층적인 구조를 지니므로 트리로 표현될 수 있다. 따라서 XSTM으로부터 DTD를 표현하는 DTD 트리를 구성하고 이를 DTD 출력기가 최종 DTD로 변환한다. 제 1절에서 예로 든 <그림 4>의 DTD는 <그림 3> (또는 <그림 5>)의 SQL/XML 문이 <그림 8>의 파스 트리 및 XSTM을 거쳐 DTD로 변환된 예이다.

3.2 XML 뷰에 대한 XPath 질의 처리

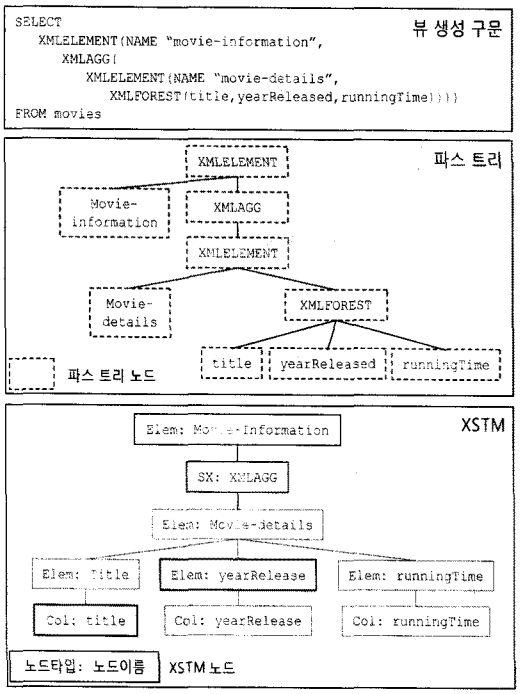
전절에서 기술한 것과 같이 관계 데이터의 XML 뷰를 정의하는 SQL/XML 문은 내부 처리를 위한 자료구조인 XSTM으로 변환되며 실제로 관계 데이터베이스에 제기되는 것은 아니다. 이렇게 정의된 뷰의 구조를 사용자는 DTD로 볼 수 있으며, 이를 토대로 XPath 질의를 제기할 수 있다. XPath 질의는 XSTM과 합성되어 사용자가 정의한 뷰에 대한 XPath 질의 결과를 관계 데이터베이스로부터 구하는 단일 SQL/XML 문이 생성된다. 이렇게 생성된 SQL/XML 문의 결과가, 정의된 뷰에 대해 사용자가 제기한 XPath 질의의 결과이다. <그림 2>는 이와 같은 관계 데이타베이스에 대한 SQL/XML 기반의 XML 질의 처리 과정을 나타낸 것으로, <그림 1>의 전통적인 SQL 기반의 XML 질의 처리와 대비된다. 이러한 과정의 핵심 기능을 담당하는 SQL/XML 질의 생성자와 뷰 질의 합

〈표 2〉 XSTM 노드 자료구조

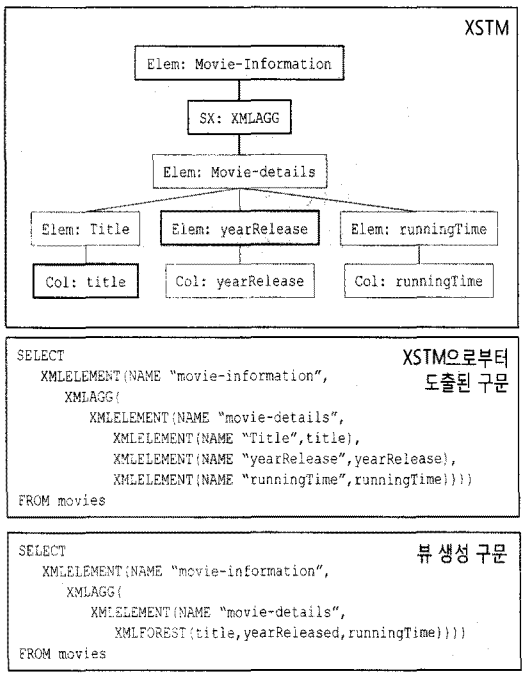
필드	내용
이름	노드 타입에 따른 이름(엘리먼트 이름, 컬럼 이름, XML 출판 함수 이름)
부모 노드	부모 노드 포인터
자식 노드	자식 노드 포인터들의 배열
노드 타입	노드 의 타입

〈표 3〉 XSTM 노드 타입

노드 타입	의미
엘리먼트(Elem)	XML 엘리먼트를 나타낸다.
RDB 컬럼(Col)	RDB 컬럼과의 매핑 정보를 나타낸다.
SQL/XML 출판 함수(SX)	SQL/XML 출판 함수 정보를 나타낸다.



〈그림 8〉 XSTM 생성 과정



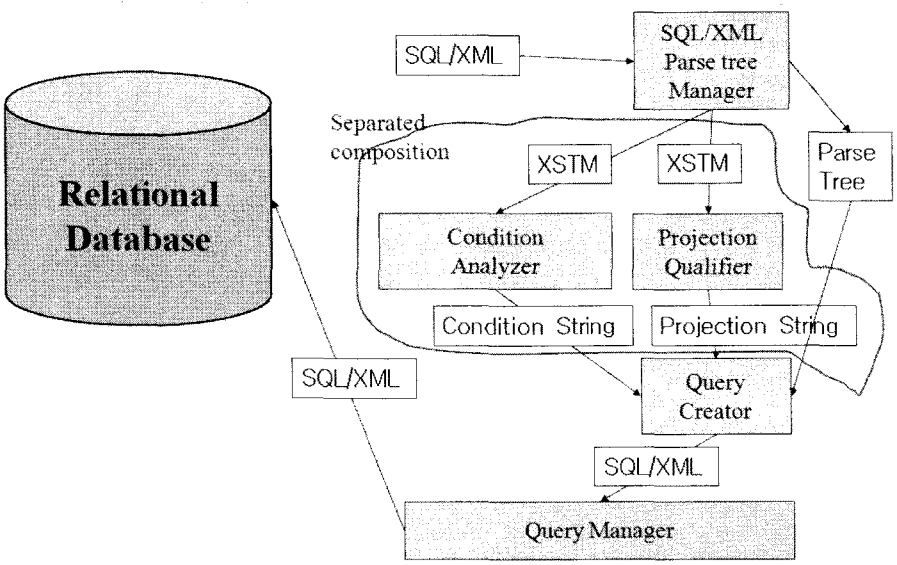
〈그림 9〉 XSTM과 SQL/XML의 관계

성 모듈을 설명하면 다음과 같다.

(1) SQL/XML 질의 생성자

뷰 질의 합성을 거쳐 최종 수행할 SQL/XML 질의문을 생성하는 것은 뷰 합성 과정에서 XSTM으로부터 추출한 정보로부터 가능해야 한다. 뷰 질의 합성 과정을 설명하기에 앞서 우선 이점부터 살펴보면 다음과 같다. XSTM은, XML 뷰를 생성하기 위한 SQL/XML 문이 출판하려는 XML 데이타의 구조 정보를 모두 기술하도록 설계되었다. 따라서 XSTM으로부터 해당 XSTM을 생성하는 데 사용된 SQL/XML 문이 출판하고자 했던 내용과 동일한 XML 조각을 출판하도록 하는 SQL/XML 문을 작성하는 것이 가능하다. <그림 9>는 뷰 생성 구문과 이것을 입력으로 하여 뷰 질의 합성 모듈이 생성한 XSTM, 그리고 이 XSTM을 입력으로 SQL/XML 질의 생성자가 작성한 SQL/XML 구문의 예를 나타낸 것이다. 이 예에서처럼 XML 뷰를 생성하기 위한 SQL/XML 문과 XSTM으로부터 SQL/XML 질의 생성자가 생성한 SQL/XML 문은 일반적으로 서로 다를 수 있지만 이들 두 SQL/XML 문으로부터 생성되는 XML 문서는 동일하다.

XSTM 자료구조는 트리 형식이다. SQL/XML 질의 생성자는 XSTM Elem 노드를 입력으로 해당 노드를 루트로 하는 XSTM 트리에 대한 SQL/XML 문을 작성한다. 이때 선택된 XSTM 트리는 전체 XSTM 트리일 수도 있고 서브 트리일 수도 있다. 전자의 경우, 위에서 설명한 것처럼 XML 뷰를 정의하기 위한 SQL/XML 문과 동일한 결과를 얻는 SQL/XML 문을 생성하게 되고, 후자의 경우에는 XML 뷰의 특정 부분의 구조를 기술하는 서브 트리로부터 생성된 SQL/XML이므로 전체 XML 뷰의 특정 부분만을 검색하는 SQL/XML 문을 생성하게 된다. 뷰 질의 합성 과정은 이러한 원리를 바탕으로



〈그림 10〉 프리디킷 부분과 경로 부분의 분리 처리

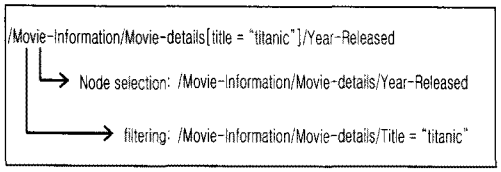
수행된다.

(2) SQL/XML 문과 XPath 질의의 합성

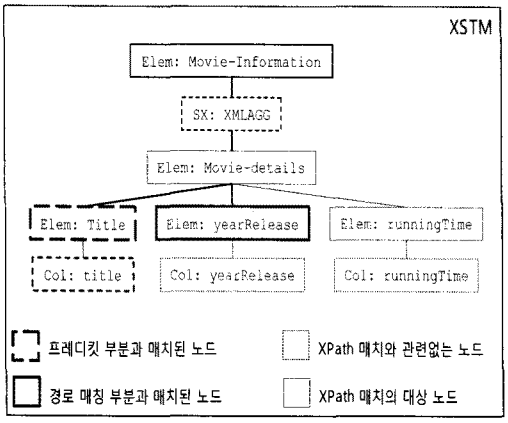
SQL/XML로 정의한 XML 뷰를 실체화하면 엘리먼트 내의 텍스트 데이터는 관계 데이터베이스로부터 추출되고 구조 정보는 SQL/XML 문으로부터 추출된다. 그러므로 XPath 질의를 프리디킷 부분과 프리디킷을 제외한 나머지 부분으로 나누었을 때, 프리디킷 부분은 관계 데이터베이스의 데이터를 검사하여 처리할 수 있고, 프리디킷을 제외한 부분은 SQL/XML 문을 검사하여 처리할 수 있다. 본 논문에서 제안하는 XML 뷰 합성 즉, XML 뷰를 정의한 SQL/XML 문과 XPath 질의를 합성하여 XPath 질의의 결과를 도출하는 SQL/XML 문을 생성하는 과정은 이러한 점을 이용하여 XPath 식을 구성하는 프리디킷 부분과 그 외의 경로 부분으로 나누어서 처리된다. <그림 10>은 이러한 과정의 개요를 나타낸 것이다.

XPath 질의의 경로는 XML 뷰 중에서 어떤 부분을 출판할 것인가를 결정하는 요소이다. 즉, XML 뷰를 정의하는 데 쓰인 SQL/XML 문의 결과 중에서 XPath 질의의 경로와 매칭되는 부분만을 검색하도록 해야 한다. 이를 위한 처리를 경로 매칭이라 부르고, 이는 두 단계에 걸쳐서 진행된다. 첫 번째 단계는 XPath 질의를 XSTM 트리에 대해 매치하는 단계이다. 매치 단계에서는 XSTM 트리의 Elem 노드만을 다룬다. XSTM 트리의 Elem 노드 이름을 엘리먼트 이름으로 취급하여 XPath 질의의 경로에 매치되는 노드를 선택할 수 있다. 이 단계가 바로 XML 뷰 중에서 실제로 검색될 부분을 선택하는 부분이다. 두 번째 단계는 선택된 XSTM Elem 노드를 루트로 한 XSTM 서브 트리를 입력으로 하여 SQL/XML 문을 생성하는 것이다. 선택된 XSTM 노드는 뷰의 어떤 부분이 검색되도록 할 것인가를 결정하는 것이므로 결정된 부분을 검색하도록 XML 출판 함수를 재작성한다. 이 두 단계를 거쳐 결과 SQL/XML 문의 SELECT 절이 완성된다.

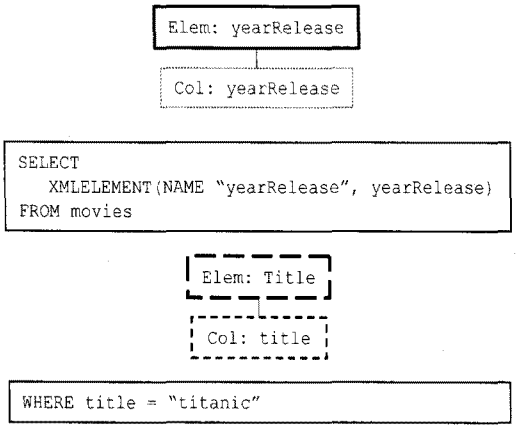
XPath 질의의 프리디킷 부분을 XML 뷰와 합성하는 것은 XPath의 필터 연산을 XML 뷰



〈그림 11(a)〉 XPath 질의의 경로 매칭 부분과 프리디킷 매칭 부분



〈그림 11(b)〉 XSTM에 대한 XPath 매칭



〈그림 11(c)〉 SQL/XML 질의 생성

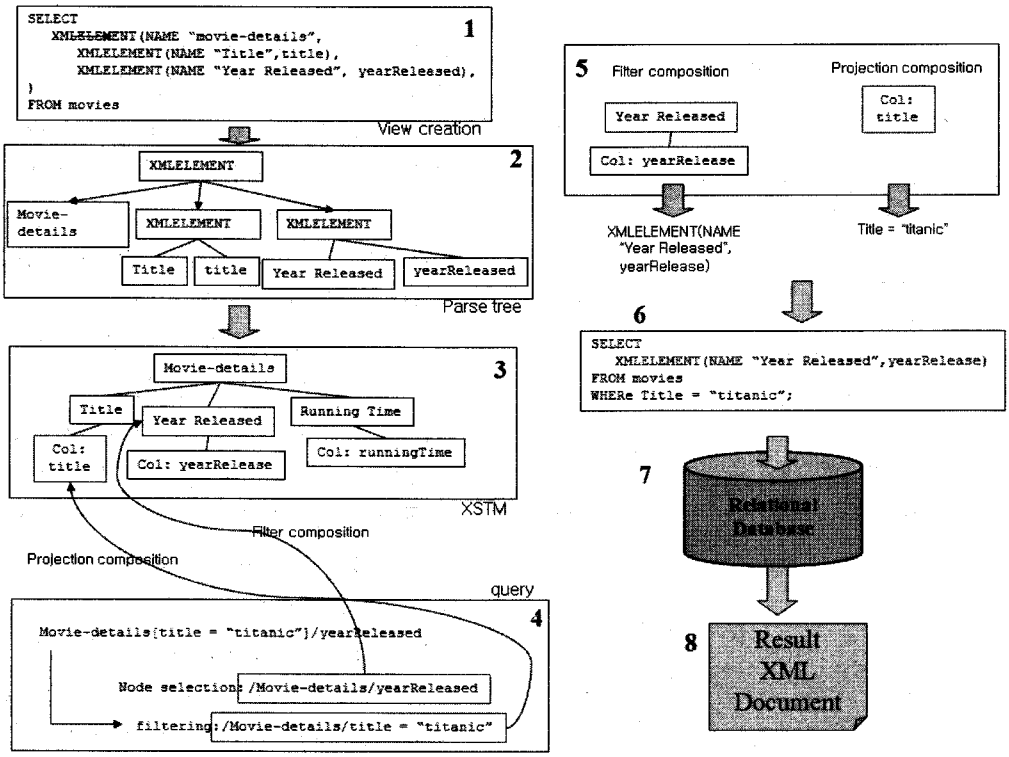
를 정의한 SQL/XML 문의 관련 부분과 합성하는 것으로 프리디킷 매칭이라 부른다. 이는 관계 데이터베이스 내의 데이터 중 프리디킷 조건을 만족하는 데이터만 검색하도록 처리하는 것이다. 이를 위해 먼저 프리디킷의 전체 경로를 XSTM 트리에 대해 매치한다. 매치된 XSTM Elem 노드의 자식노드인 Col 노드를 가져와서 프리디킷으로 검사하려는 엘리먼트의 내용이 관계 데이터베이스의 어떤 컬럼과 매핑되는지 조사한다. 이 컬럼은 프리디킷에 명시된 조건을 적용해야 하는 것이므로 이를 적용하여 SQL/XML 문의 where 절을 작성한다.

<그림 11>은 XML 뷰 질의 합성 전체 과정의 예를 나타낸 것이다. <그림 11(a)>는 XPath 질의를 경로 매칭 부분과 프리디킷 매칭 부분으로 나누는 것을 나타낸 것이고, <그림 11(b)>는 경로 매칭과 프리디킷 매칭에서 실제로 매치된 XSTM 노드들을 나타낸 것이다. <그림 11(c)>는 매칭된 노드들을 이용하여 SQL/XML 문을 생성한 것을 나타낸 것이다. 이렇게 경로 매칭 부분과 프리디킷 매칭 부분으로 나누어서 합성된 SQL/XML 구문 절(clause)들이 조합되어 SQL/XML 문이 생성된다. 이와 같이 생성된 최종 SQL/XML 문을 관계 데이터베이스에 제기하면 관계 데이터베이스는 XML 뷰에 대해 제기된 XPath 질의의 결과를 XML로 반환하게 된다.

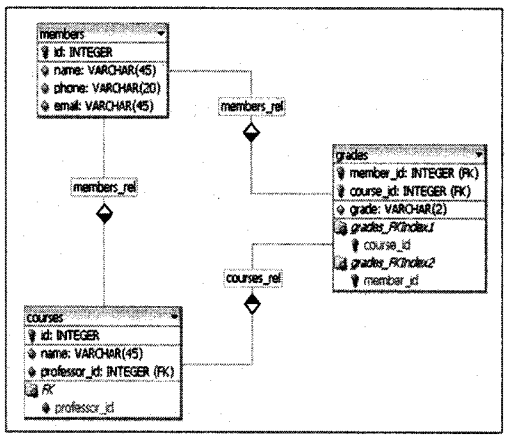
<그림 12>는 SQL/XML 문으로 XML 뷰를 정의한 것이 파스 트리 및 XSTM으로 변환되고, 이 뷰에 대한 XPath 질의가 뷰와 합성되어 SQL/XML 문을 생성하여 질의 결과를 XML로 얻는 전체 과정 흐름의 예를 나타낸 것이다. <그림 12>에서 과정 1은 SQL/XML 문을 통한 관계 데이타에 대한 XML 뷰의 생성, 2와 3은 그로부터 생성된 내부 자료 구조로 파스 트리 및 XSTM, 4는 뷰에 대해 제기된 XPath 질의에서 상기 자료 구조의 요소와 매치되어 합성될 부분의 추출, 5는 경로 및 프리디킷 매칭, 6은 최종 생성된 새로운 SQL/XML 문을 나타낸 것이다.

4. 구현 및 테스트

본 절에서는 본 논문에서 제시한 SQL/XML



〈그림 12〉 뷰 질의 합성 전체 과정



〈그림 13〉 구현의 테스트에 사용된 관계 데이터베이스 스키마

기반 처리의 검증을 위해 전절에서 기술한 내용의 설계 과정에서 수행된 예비 구현 및 구현을 통한 테스트 결과를 기술한다. 본 구현은 Windows XP SP2 상에서 Java로 수행되었으며 J2SE 5.0 update 6과 Eclipse SDK 3.1을 이용하였다. XML-enabled 관계 DBMS로는 Oracle 10g Express Edition을 이용하였다. XML 질의어로는 XPath 1.0[XML Path Language]을 사용하였다. <그림 13>은 본 구현의 테스트에 사용된 관계 데이터베이스의 스키마를 나타낸 것이다.

4.1 SQL/XML 문으로부터 DTD 생성

SQL/XML로 정의한 XML 뷰를 나타내는 DTD를 생성하는 구현에서는, SQL/XML 표준에 명시된 5개의 SQL/XML 출판 함수 중 XMLCONCAT을 제외한 나머지 4개 즉, XMLELEMENT, XMLATTRIBUTES, XMLAGG, XMLFOREST를 모두 구현하였다. 또한 한 함수 내

```
SELECT XMLELEMENT(NAME "members", XMLAGG(
 XMLELEMENT(NAME "member", XMLATTRIBUTES(members.id as "id"),
 XMLFOREST(members.name as "name", phone as "phone", email as "email"))))
 FROM members, grades
 where members.id = grades.member_id
```

Q1 : 각 member 별로 이름, 전화번호, email 주소를 열거. 단, member의 ID 값은 Attribute로 삽입.

```
Problems Javadoc Declaration Debug
<terminated> DTDGenerator [Java Application] C:\Program Files\Java\jre1.5.0_06\bin\javaw.exe (2006. 10.
SELECT XMLELEMENT(NAME "members", XMLAGG(
 XMLELEMENT(NAME "member", XMLATTRIBUTES(members.id as "id"),
 XMLFOREST(members.name as "name", phone as "phone", email as "email"))))
 FROM members, grades
 where members.id = grades.member_id


<!ELEMENT members (member+)>
<!ELEMENT member (name,phone?,email?)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST member id CDATA #REQUIRED >
```

Q1에 의한 XML 뷰를 나타내는 DTD 생성결과

```
- <members>
  - <member id="8">
      <name>name8</name>
      <phone>123-4567</phone>
      <email>h.com</email>
    </member>
  - <member id="9">
      <name>name9</name>
      <phone>123-4567</phone>
      <email>i.com</email>
    </member>
... 생략...

  - <member id="4">
      <name>name4</name>
      <phone>123-4567</phone>
      <email>d.com</email>
    </member>
  - <member id="11">
      <name>name11</name>
      <email>j.com</email>
    </member>
</members>
```

Q1 의 결과 XML 문서

〈그림 14〉 SQL/XML 문으로부터 DTD의 생성 테스트(Q1)

에 다른 함수가 중첩되는 것도 지원하였으며, SQL/XML 문의 from절에 복수개의 테이블이 열거되는 것도 지원하였다.

<그림 14>는 질의 Q 1 : 각 member의 이름, 전화번호, email 주소를 member 별로 열거해주

```
select XMLELEMENT(name "grades",
 XMLAGG(xmlelement(name "grade", XMLATTRIBUTES(courses.name as "cn",
 members.name as "sn"), grades.grade)))
 from members, courses, grades
 where members.id = grades.member_id and courses.id = grades.course_id
```

Q2 : 학생들의 과목별 성적을 열거. 단, 과목명과 학생 이름은 Attribute로 삽입.

```
Problems Javadoc Declaration Debug
<terminated> DTDGenerator [Java Application] C:\Program Files\Java\jre1.5.0_06\bin\javaw.exe (2006. 10. 27 오후 2:41:52)
select XMLELEMENT(name "grades",
XMLAGG(xmlelement(name "grade", XMLATTRIBUTES(courses.name as "cn", members.name as "sn"), grades.grade)))
from members, courses, grades
where members.id = grades.member_id and courses.id = grades.course_id


<!ELEMENT grades (grade*)>
<!ELEMENT grade (#PCDATA)>
<!ATTLIST grade cn CDATA #REQUIRED sn CDATA #REQUIRED >
```

Q2에 의한 XML 뷰를 나타내는 DTD 생성결과

```
- <grades>
    <grade cn="Network" sn="name8">F</grade>
    <grade cn="Database" sn="name8">F</grade>
    <grade cn="OS" sn="name8">F</grade>
    <grade cn="OS" sn="name5">C+</grade>
    <grade cn="Network" sn="name5">B</grade>
    <grade cn="Database" sn="name5">B+</grade>
    <grade cn="OS" sn="name6">D+</grade>
    <grade cn="Network" sn="name6">A</grade>
    <grade cn="Database" sn="name6">B</grade>
    <grade cn="OS" sn="name7">D</grade>
    <grade cn="Network" sn="name7">C</grade>
    <grade cn="Database" sn="name7">C</grade>
    <grade cn="Database" sn="name4">A+</grade>
    <grade cn="OS" sn="name4">B+</grade>
    <grade cn="Network" sn="name4">A</grade>
  </grades>
```

Q2 의 결과 XML 문서

〈그림 15〉 SQL/XML 문으로부터 DTD의 생성 테스트(Q2)

고 member ID는 attribute로 삽입시켜주는 XML 출판으로 (a) XML 뷰를 정의하는 SQL/XML 문, (b) Q1의 SQL/XML 문 실행 결과 XML 문서, (c) Q1의 SQL/XML 문으로부터 생성된 DTD를 나타낸 것이다. 그림 15는 질의 Q2 : 학생들의 과목별 성적을 열거하는 데 과목명과 학생 이름을 attribute로 삽입시켜주는 XML 출판으로 (a) XML 뷰를 정의하는 SQL/XML 문, (b) Q2의 SQL/XML 문 실행 결과 XML 문서, (c) Q2의 SQL/XML 문으로부터 생성된 DTD를 나타낸 것이다. 이들 이외에도 여러 질의를

〈표 4〉 테스트에 사용된 XPath 질의

질의ID	XPath 질의
Q1	/grades
Q2	/grades/member
Q3	/grades/member[name = '아무개85']
Q4	/grades/member[name = '아무개85'] /email
Q5	/grades/member[name= '아무개47']/course[name = 'MATH']
Q6	//email
Q7	//member[name = '아무개85']
Q8	//member[name = '아무개85']/phone
Q9	//member[name = '아무개77']//grade

```
SELECT XMLELEMENT (NAME grades,
XMLAGG(XMLELEMENT(NAME member,
 XMLFOREST(members.name as name, members.email as email, members.phone as phone),
 XMLELEMENT(NAME course, XMLATTRIBUTES(courses.id as cid),
 XMLELEMENT(NAME name, courses.name), XMLELEMENT(NAME grade, grades.grade)
))))
FROM courses , members , grades
WHERE grades.member_id = members.id and grades.course_id = courses.id
```

〈그림 16〉 테스트에 사용된 XML 뷰 정의를 위한 SQL/XML 문

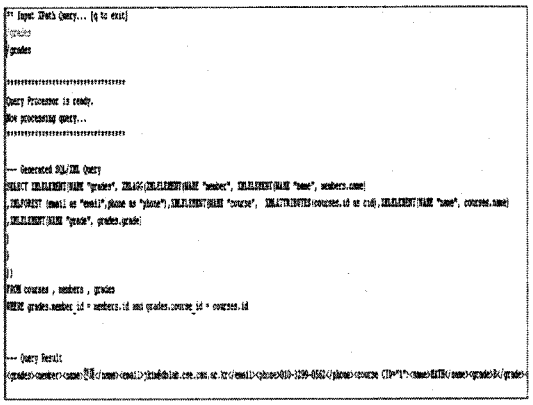
```
<!ELEMENT grades (member+)>
<!ELEMENT member (name,email?,phone?,course)>
<!ELEMENT course (name,grade)>
<!ELEMENT grade (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ATTLIST course cid CDATA #REQUIRED >
```

〈그림 17〉 그림 16의 XML 뷰를 나타내는 DTD

테스트한 결과 모두 정확한 결과를 얻었으며 특히, 생성된 DTD는 SCHOLARLY TECHNOLOGY GROUP의 XML Validation Form을 이용하여 검증하였고 모두 PASS 판정이 나왔다[XML Validation Form].

4.2 SQL/XML 문과 XPath 질의 합성 및 수행

SQL/XML로 정의한 XML 뷰가 DTD로 표현

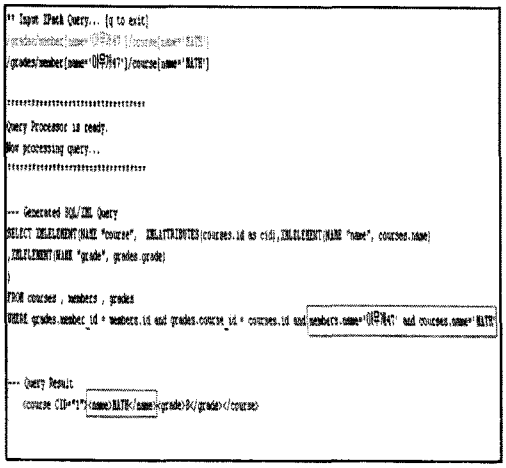


〈그림 18〉 Q1의 수행 결과 화면(단순 질의)

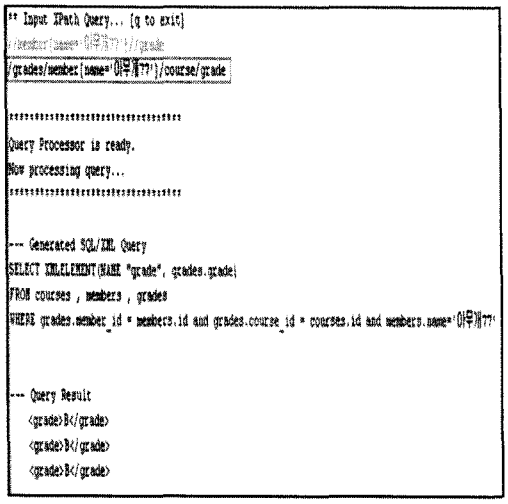
```
<?xml version="1.0" encoding="EUC-KR" ?>
- <grades>
  - <member>
      <name>천재</name>
      <email>jkim@dblab.cse.cau.ac.kr</email>
      <phone>010-3299-0562</phone>
    - <course CID="1">
        <name>MATH</name>
        <grade>B</grade>
      </course>
    </member>
  - <member>
      <name>아무개0</name>
      <email>abc0@test.com</email>
      <phone>010-123-0000</phone>
    - <course CID="1">
        <name>MATH</name>
        <grade>C</grade>
      </course>
    </member>
  - <member>
      <name>아무개1</name>
      <email>abc1@test.com</email>
      <phone>010-123-0001</phone>
    - <course CID="1">
        <name>MATH</name>
        <grade>A</grade>
      </course>
    </member>
  - <member>
      <name>아무개2</name>
```

〈그림 19〉 Q1 결과의 IE6.0 뷰

된 것을 보고 제기된 XPath 질의에 대해 XML 뷰를 정의한 SQL/XML 문과 XPath 문을 합성하여 질의 결과를 구하는 SQL/XML 문을 생성하는 기능의 구현에서는, XPath의 자식(/) 및 후손(//) 축 그리고 프리디킷을 지원하도록 구현하였다. <표 4>는 구현 결과의 테스트에 사용한 XPath 질의 집합을 나타낸 것이다. 이들 질의는 <그림 13>의 관계 데이타베이스에 대해 <그림 16>의 SQL/XML 문으로 정의한 XML 뷰 즉, <그림 17>의 DTD로 변환되어 표현된 XML 뷰에 대해 제기된 것들이다. Q1~Q5는 후손 축



〈그림 20〉 Q5의 수행 결과 화면(프리디킷 포함 질의)



〈그림 21〉 Q9의 수행 결과 화면(후손 축 포함 질의)

이 없는 XPath 식들로서 프리디킷은 하나도 없는것, 하나 또는 복수개 존재하는 질의들이고, Q6~Q9는 후손 축이 포함된 질의 들이다.

<그림 18>은 단순 질의인 Q1에 대한 수행 결과 화면이다. 화면 상단에는 입력된 XPath 식이 출력되고, 중앙부에는 그 XPath 식이 XML 뷰를 정의한 <그림 16>의 SQL/XML 문과 합성되어 생성된 SQL/XML 문이 출력되어 있다. 하단에는 XPath 식의 결과가 XML로 반환된 내용을 출력하고 있다. <그림 19>는 <그림 18> 하단의 XPath 질의 결과를 IE 6.0 XML 브라우저로 뷰(view)한 것으로 XML 엘리먼트들의 들여쓰기로 질의 결과의 정확성을 용이하게 확인하기 위해 사용되었다. <그림 20>과 <그림 21>은 각각 프리디킷 포함 질의인 Q5와 후손 축 포함 질의인 Q9의 실행 결과 화면이다. 화면의 포맷은 Q1의 경우와 동일하다. 단, <그림 21>에서는 Q9이 후손 축을 포함하므로 후손 축을 포함하는 XPath 식이 자식 축만을 포함하는 식으로 변환된 후(즉, 화면 상단의 첫 번째 XPath 식은 후손 축을 포함하는 사용자가 제기한 원래의 XPath 식이고, 두 번째 XPath 식은 그것이 자식 축만 포함하는 것으로 변환된 것이다) SQL/XML 문을 생성한 내용을 보여주고 있다. 이들 Q1, Q5, Q9뿐 아니라 나머지 모든 질의들의 합성 및 수행 결과는 모두 정확한 것으로 확인되었다.

5. 결 론

본 논문에서는 SQL/XML을 이용하여 관계 데이터에 대한 XML 뷰를 정의하고, 정의된 XML 뷰에 대해 XPath 질의를 처리하는 기법을 제시하였다. 이 두 가지 기능을 지원하기 위한 핵심 자료구조로 XSTM을 고안하였다. XSTM은 SQL/XML 구문을 파싱하여 파스 트리를 생성한 후 이로부터 생성된다. 이는 DTD를 생성하거나 뷰 질의 합성을 효율적으로 지원하기 위한 자료구조다. SQL/XML 기반의 관계 데이터에 대한 XML 뷰 정의 및 XML 질의 처리 체계는 구현 및 다양한 테스트를 통하여 검증하였다.

향후 연구 과제로 SQL/XML 표준을 지원하는 XML-enabled 관계 DBMS와 JDBC를 통해 연동하는 미들웨어로 관계 데이터의 XML 출판 툴의 연구 개발을 수행 중이다. 이 툴은 하부 관계 데이터베이스 상에서 아래 네 가지 기능을

제공한다.

- SQL/XML 문을 통해 관계 데이타의 XML 뷰를 정의하고 이를 DTD로 자동 변환하여 제공하는 기능
- XML 뷰에 대한 XPath 질의를 SQL/XML 문으로 자동 변환하여 처리하는 기능
- 관계 데이터베이스 스키마를 디스플레이하고 그로부터 출판할 XML 데이터의 DTD를 마우스 클릭/드래그 방식으로 정의하는 GUI
- GUI 상에서 정의된 DTD를 준수하는 XML 문서를 하부의 관계 데이타베이스로부터 출판하기 위한 SQL/XML 문을 자동 생성하는 기능

첫 두 기능은 본 논문에서 다루었고, 나머지 두 기능은 개발 중에 있다. 상기 기능들이 통합된 툴을 제공함으로써 비즈니스 데이터를 저장하고 있는 관계 데이터베이스 상에서 데이터베이스 설계자, 응용 설계자, 서비스 제공자 등에게 편리하고 효율적인 XML 출판 환경을 제공할 수 있다. 또한 상기의 요소 기술들은 관계 데이터를 바탕으로 XML 데이터를 처리하는 시스템 요소와 연동하는 전자거래, CRM, BI 툴 등의 보다 진보된 비즈니스 정보 및 의사 결정 지원 시스템의 기능 개발에 활용될 수 있다.

참 고 문 헌

[1] 전찬훈, "RDBMS를 이용한 XML 데이터의 혼합형 저장 기법", 석사학위논문, 중앙대학교 대학원, 2007.

[2] M. Carey et al., "XPERANTO : Publishing Object-Relational Data as XML", *Proc. Workshop on the Web and Databases*, May 2000a.

[3] M. Carey et al., "XPERANTO : Middleware for Publishing Object-Relational Data as XML Documents", *Proc. Int'l Conf. On VLDB*, 2000b, pp. 646-648.

[4] M. Fernandez et al., "SilkRoute : Trading between Relations and XML", *Proc. WWW Conf.*, 2000, pp. 723-746.

[5] M. Fernandez et al., "Efficient Evaluation of XML Middle-ware Queries", *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, 2001a, pp. 103-114.

[6] M. Fernandez et al., "Publishing Relational Data in XML : the SilkRoute Approach", *IEEE Data Eng. Bulletin*, Vol. 24, No. 2, June 2001b, pp. 12-19.

[7] J. Melton and S. Buxton, Chapter 15. SQL/XML in "*Querying XML : XQuery, XPath, and SQL/XML in Context*", Morgan Kaufmann, 2006.

[8] A. Schmidt et al., "Efficient Relational Storage and Retrieval of XML Documents", *Proc. Workshop on Web and Databases*, 2000, pp. 47-52.

[9] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. DeWitt, and J. Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities", *Proc. Int'l Conf. on VLDB*, 1999.

[10] J. Shanmugasundaram et al., "Efficiently Publishing Relational Data as XML Documents", *Proc. Int'l Conf. on VLDB*, 2000, pp. 65-76.

[11] J. Shanmugasundaram et al., "Querying XML Views of Relational Data", *Proc. Int'l*

Conf. on VLDB, 2001a, pp. 261-270.

[12] J. Shanmugasundaram et al., "A General Technique for Querying XML Documents Using a Relational Database System", *ACM SIGMOD Record*, Vol. 30, No. 3, Sep. 2001b, pp. 20-26.

[13] M. Yoshikawa, T. Amagasa, T. Shimura, S. Uemura, "XRel : A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases", *ACM Trans. on Internet Technology*, Vol. 1, No. 1, Aug. 2001, pp. 110-141.

[14] ISO/IEC FCD 9075-14, Information Technology-Database languages-SQL-part 14 : XML-Related Specifications(SQL/XML), http://www.ansi.org.

[15] XML Path Language(XPath), Version 1.0, W3C Recommendation, http://www.w3.org/TR/1999/REC-xpath-19991116.

[16] XML Validation Form, SCHOLARLY TECHNOLOGY GROUP. http://www.stg.brown.edu/service/xmlvalid/.

저자소개

이 상 욱

중앙대학교 컴퓨터공학과 졸업(학사), 중앙대학교 대학원 컴퓨터공학과 졸업(석사), 관심분야 XML 데이터베이스, XML 스트림 데이터 처리 등

김 진

중앙대학교 컴퓨터공학과 졸업(학사), 중앙대학교 대학원 컴퓨터공학과 졸업(석사), 관심분야 XML 스트림 데이터 처리, 웹 데이터베이스 등

강 현 철

서울대학교 컴퓨터공학과 졸업(공학사), 1985년 U. of Maryland at College Park, Computer Science(M.S.). 1987년 U. of Maryland at College Park, Computer Science(Ph.D.). 1988년~현재 중앙대학교 컴퓨터공학부 교수, 관심분야 XML 및 웹 데이터베이스, Stream 데이터 관리, 센서네트워크 데이터베이스 등

◆ 이 논문은 2009년 10월 23일 접수하여 1차 수정을 거쳐 2009년 12월 22일 게재확정되었습니다.