관계형 데이타베이스 시스템에서 XML-뷰를 통한 XML 데이타의 지원

(XML Data support in RDBMS through XML-View)

이 제 민 † 민 경 섭 † 박 상 원 † 김 형 주 ††
(Je-Min Lee) (Kyung-Sub Min) (Sangwon Park) (Hyong-Joo Kim)

요 약 XML이 웹 문서 표준으로 자리잡아감에 따라, 최근 XML 문서를 기존의 관계형 데이타베이스 시스템을 이용해 저장, 검색하고자 하는 연구가 활발히 진행되고 있다. 이에 본 논문에서는, 기존의 관계형 데이타베이스 시스템에서 제공하는 뷰와 테이블 함수의 개념을 XML로 확장하여, XML 문서를 관계형 테이블과 같은 방식으로 사용할 수 있게 하는 방법을 제안하였다. 제시한 방법에서 XML 문서는 관계형 데이타베이스 시스템에 독립적으로 존재하고 사용자는 XML 문서에서 필요한 부분에 대해 XML-뷰 정의문을 이용하여 데이타베이스 스키마에 등록한다. 등록된 XML-뷰는 SQL을 통해 사용될 수 있으며, 이때 XML 문서를 테이블과 같은 형식으로 접근, 사용하게 해주기 위해 내부적으로 XML-테이블 함수가 사용된다. 제안한 방식은 기존의 관계형 데이타베이스 시스템이 XML에 대한 뷰 정의 구문과 테이블 함수를 제공하게 하여 XML 문서에 대한 접근이 쉽게 이루어지도록 해 준다. 또한 XML 문서에 대한 스키마 변환이 요구되더라도 XML-뷰 정의의 수정으로 이를 해결할 수 있다.

Abstract Recently, XML is emerging as a web document standard and researches on storing and querying XML documents using existing RDBMS are actively in progress. In this paper, we propose a method that enables users to view and use XML documents like a relational table by extending concept of RDBMS's SQL view and table function. In our approach, XML documents are existed independently of RDBMS and users can register relational schema of necessary part of XML document by using XML-View definition statement. Registered XML-View can be queried by SQL, and XML-table function is internally used so that XML-View can be accessed and used like a relational table. The proposed method enables users to access XML documents by providing the XML-View definition syntax and the table-function for existing RDBMS. Also, when schema modification for XML is needed, user can handle by modification of the XML-View definition.

1. 서 론

XML[1]은 자기 기술적(self-describing) 특징을 가진 마크업(markup) 언어이다. 사용자는 자신이 정의한 태그(tag)를 통해 모든 형태의 데이타를 XML 문서 형태로 표현할 수 있다. 이는 웹이나 일반 응용에서의 데이타 교환과 저장을 XML로 기술된 문서로 표현할 수 있음을 말하며, 그 결과로 데이타에 대한 구문적 일관성을 보장해 준다. 이러한 XML의 장점은 앞으로 상당한 양의 데이타가 XML로 기술될 것이다. 이에 XML 문서를 효율적으로 저장, 검색하기 위한 방법에 대한 연구가 활발히 진행되고 있다. 한편으로 이러한 연구들은 XML 문서가 새로운 데이타 형식임을 고려하여 새로운 시스템을 구현하고자 하는 연구가 진행되었다. Stanford 대학에서는 XML과 같은 반구조적(semistructured) 형태의 데이타를 저장, 검색할 수 있도록 하기 위한 데이타베이스로 Lore[2]와 그 질의어로 Lorel[3]을 제안, 구현하였다. 이는 XML 문서를 가장 효과적으로 표현, 처리할 수 있다는 장점이 있으나, 시스템의 검증에 많은 시

• 본 연구는 BK21에서 지원받았음.
† 비 회 원 : 서울대학교 컴퓨터공학부
jmlee@xinics.com
ksmin@oopsla.snu.ac.kr
swpark@oopsla.sun.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@oopsla.snu.ac.kr
논문접수 : 1999년 12월 13일
심사완료 2001년 4월 30일

간이 필요하며, 구현을 위해 상당한 비용을 필요로 한다[4].

다른 한편으로 새로운 시스템을 제안하기보다는 관계형/객체지향 데이타베이스 시스템과 같이 현재 유효성을 인정받고 있는 검증된 시스템을 이용하고자 하는 연구가 진행되고 있다[5,6,7,8]. 이는 기존의 시스템을 이용함으로써 새로운 시스템을 만드는 비용을 줄이고 검증된 기능을 사용할 수 있는 장점이 있다. 그러나 XML 형식의 데이타를 관계형이나 객체형 모델로 매핑하는 데의 어려움이 있으며 저장/검색에서 XML의 특징을 이용할 수 없어 성능이 저하될 수 있다. 이러한 문제를 해결하고자 하는 연구들은 공통적으로 새롭게 저장, 관리되어야 하는 XML 문서를 위한 방법만을 제안하고 있다. 하지만 기존의 데이타베이스에 저장되어 있는 데이타와 통합하여 전체 데이타의 일부분으로서 XML 문서를 바라보고자 하는 경우에 다음과 같은 문제들이 나타날 수 있다.

첫째, 새로운 XML 문서를 데이타베이스로 표현, 저장하기 위해 사용자는 전처리기를 통해 XML 문서의 매핑 스키마를 결정한다. DTD는 XML 문서의 구조 정보를 표현하는 방법이며 XML 문서의 스키마라 할 수 있다. 하지만 XML 문서는 DTD 정보를 가질 수도 있고 그렇지 않을 수도 있기 때문에 각각의 XML 문서의 DTD는 서로 다를 수 있다. XML을 관계형 데이타베이스에 저장하기 위하여 스키마를 생성할 때 기존의 연구에서는 DTD 정보를 이용하여 스키마 정보를 생성한다. 그러므로 DTD 정보가 변하는 경우에는 기존의 스키마와 달라서 스키마 간의 충돌이 발생할 수 있다. 둘째, DTD 정보를 이용하여 스키마를 생성하였을 경우 XML 문서의 형식이 변하거나 기존의 XML 문서를 새로운 문서로 대체해야 하는 경우 관계형 데이타베이스의 스키마를 변경하고 기존의 데이타를 변경된 스키마에 맞게 이관하는 등 많은 작업을 수행해야 한다. 셋째, 데이타베이스 사용자들은 XML 문서 전체가 아니라, 자신이 필요한 부분만을 가공하여 보고자 할 수 있다. 하지만 기존의 방법은 모든 XML 문서를 데이타베이스에 저장한 후 질의를 통하여 데이타를 가져온다. 이때 전체 XML 문서 중 질의에 참여하는 부분이 적을 경우 질의 처리 시간이나 저장 공간의 낭비를 초래한다. 또한 DTD를 이용하여 스키마를 생성하면 스키마가 복잡하게 만들어지게 되므로 스키마의 의미를 파악하여 의미있는 질의를 만드는 것이 어렵게 되므로 스키마 자체가 가지는 의미적(semantic)인 정보를 상실하게 되는 경향이 있다.

본 연구에서는 위의 문제점들을 해결하고, 새롭게 추가되는 XML 문서에 대하여 사용자가 관심 있는 부분만을 접근하여 질의할 수 있도록 하기 위한 효율적인 방법을 제안하고 구현하였다. 제안된 방법은 관계형 데이타베이스[1]에서 제공하는 뷰(view)와 테이블 함수(table function)[12]의 개념을 확장하여 구현하였으며, 사용자들로 하여금 SQL을 통해 XML 데이타에 대해 질의할 수 있도록 하였다. 이를 위해 XML 문서의 일부분을 사용자가 테이블 형태로 바라볼 수 있도록 하기 위한 XML-뷰를 정의하였으며 XML 문서에 대한 접근을 처리해주기 위한 XML 질의 처리기를 구현하였다. 이 질의 처리기는 후에 XML 표준을 따르는 질의 처리기로 대체가능 하도록 하기 위하여 데이타베이스 시스템 외부에서 동작하도록 구현하였다.

이와 같은 방법은 XML 문서가 파일이거나 네트워크 상에 있는 임의의 문서일 경우라도 테이블로 바라볼 수 있는 XML-뷰만 제공하면 SQL로 질의를 할 수 있는 장점이 있다. 또한 실제 관계형 데이타베이스에 저장되어 있는 데이타와 XML 문서간의 조인 연산과 같은 연관된 연산을 수행할 수 있기 때문에 기존의 관계형 데이타베이스에 저장되어 있는 데이타를 이용한 질의가 가능하게 되는 장점이 있다.

2장은 본 연구에서 정의한 XML 질의어에 대한 기존 연구와 제안된 시스템 구조와 유사한 다른 시스템에 대해 간략히 소개한다. 3장은 본 연구에서 제안한 XML 처리 방법에 대한 개략적인 소개와 예를 보이고, 4장은 제안된 XML 처리 방법에 대한 설계와 구현된 시스템의 구조, 동작 방식을 설명한다. 마지막으로 5장은 결론과 향후 연구에 대해 기술한다.

2. 관련 연구

구조화되어 있지 않은 데이타에 관계형 데이타베이스에서 사용하는 테이블 형태의 스키마를 부여하는 방법과 관련된 연구로는 Araneus[13,14]가 있다. 이 연구는 ADM(Araneus Data Model)이라는 데이타모델을 이용해 웹 문서를 모델링 했으며, ADM으로 표현된 스키마에서 ULIXES[13,14]라는 질의어를 이용하여 테이블 형식의 뷰를 생성할 수 있도록 하였다. 이 뷰는 Araneus의 ULIXES를 통해서 접근 가능하며 데이타베이스로의 매핑을 위해 추가의 작업이 필요하다. 하지만 본 연구는 XML 문서에 대한 뷰를 정의하기 위해 관계형 질의어

1) 본 연구에서 사용한 관계형 데이타베이스로는 SRP[9]를 이용하였다. 본 논문에서 제안한 방법은 XWEET[10]의 PDM[11] 구축에 이용되었다.

인 SQL 을 확장하였으며, 이를 지원하기 위한 방법을 제공하였다.

관계형 데이타베이스에서 XML 문서의 처리에 대해 언급한 논문으로는 [5,6,7,15] 등이 있다. [6]에서는 에지(edge) 방법을 이용하여 그래프를 표현할 수 있는 스키마를 생성한 후 이 스키마에 맞게 데이타를 저장하는 방법에 대하여 기술하고 있다. 이 방법의 장점은 임의의 XML 문서가 들어오더라도 잘 저장할 수 있으며 구현이 간단하다는 점이다. 하지만 질의를 처리하게 되면 조인 연산이 많아지는 단점이 있으며 XML의 DTD 정보를 이용하지 않는다는 점이다. 이와는 달리 [7]은 DTD 정보를 이용하여 관계형 데이타베이스의 스키마를 생성한다. 이때 테이블의 개수를 줄이는 방법을 제안하고 있으며 이는 질의 수행시에 조인 연산의 수를 감소시킨다. STORED[5]는 관계형 데이타베이스에 반구조적인 데이타를 저장하는 방법에 대하여 기술하고 있다. 또한 정의된 방법을 벗어나는 반구조적인 데이타는 오버플로우에 저장되게 한다. [15]는 [6]과 유사한 방법으로 저장하지만 정규 경로식(regular path expression)을 잘 지원하기 위하여 각각의 경로마다 숫자를 부여하여 저장하였다. 이들 방법의 공통점은 XML 문서를 적당한 형태로 가공하여 특정 스키마에 저장한다는 점이다. 그러므로 형태가 변화되는 데이타를 잘 저장하기 위한 것은 빠른 질의 수행이 어렵고 그렇지 않은 경우는 형태 변화를 반영하기 어렵다. 또한 각각의 데이타는 모두 데이타베이스에 저장되므로 실제 사용자가 필요로 하는 일부분의 데이타만 다루기가 힘들다.

위의 방법들은 본 논문의 방식과 달리 관계형 데이타베이스 시스템 사용자의 관점에서가 아니라 XML 데이타의 저장과 검색을 위한 기능을 처리하는 용도만을 고려하였다. XML 문서를 분해하여 관계형 데이타베이스 스키마에 맞는 형식으로 저장하면 질의를 기존의 질의 처리기를 그대로 사용하여 처리할 수 있는 장점이 있다. 사용자는 XML 질의문을 이용하여 XML 문서에 접근하지만 실제로 질의어는 내부적으로 관계형 데이타베이스의 질의문으로 변환하여 처리가 되는 것이다. 이러한 방식들의 단점은 XML이 덜 엄격한 구조를 가지고 있기 때문에 관계형 데이타베이스 스키마로의 변환이 어려운 경우가 생길 수 있다. 관계형 데이타에 저장한 후 질의를 SQL로 할 때 생기는 장점으로 사용자의 의미있는 질의를 잘 처리할 수 있는 점이라고 하지만 실제로는 XML 질의를 기계적으로 SQL로 변환하므로 그 질의의 의미 파악이 큰 의미가 되지 않는다. 또한 간단한 XQL 문장도 조인 연산이 많은 복잡한 SQL로 변하게 되므로 SQL 자체만으로 질의를 한다는 것도 큰 의미가 없는 경우가 많다.

본 논문에서는 테이블 형태의 결과를 생성해 내기 위해서 XML 데이타를 위한 질의어를 사용한다. XML 또는 반구조적 데이타를 위한 질의어는 몇 가지가 제시되었다. Lorel[3]은 OQL을 확장하여 만든 질의어로서 일반적인 반구조적 데이타를 질의 대상으로 하고 있다. XML-QL[16]은 패턴 매칭을 통한 질의와 질의 결과를 XML로 만드는 것 등이 특징이라고 할 수 있다. 이 언어들의 특징은 XML이나 반구조적 데이타의 특성을 고려하여 기존의 데이타베이스의 질의어에는 없는 정규 경로식을 지원하는 점이다. 본 논문에서 제작한 질의어는 Lorel과 비슷한 형태를 가지고 있으며 전체 시스템의 동작을 위한 최소한의 기능을 제공한다.

3. XML 데이타 처리 방법

XML 데이타를 사용자에게 보여주는 방식은 크게 두 가지로 나눌 수 있다. XML을 그대로 사용자에게 보여주는 방식과, XML을 특정 데이타베이스 스키마에 적용시켜 데이타베이스의 데이타로 사용자에게 보여 주는 방식이 있다. XML 데이타를 데이타베이스 데이타로 보여주는 경우 사용자는 XML 데이타를 위한 새로운 접근 방법을 익히지 않고도 기존의 데이타를 다루는 방식을 그대로 사용하여 XML 데이타를 다룰 수 있는 장점이 있다. 그러나 XML의 불규칙하고 엄격하지 않은 구조 때문에 특정 데이타베이스의 스키마에 맞도록 XML 데이타를 바꾸는 작업은 어려울 수가 있다. 특히 관계형 데이타베이스의 테이블 기반의 스키마는 표현에 여러 가지 제한을 가지고 있어서 XML 데이타를 표현하는 것이 쉬운 작업이 아니다. 본 논문에서는 앞에서 말한 장점을 가지면서 XML 데이타 전체를 관계형 데이타베이스의 스키마에 맞도록 변환하는 부담을 덜기 위하여 XML의 일부분만을 관계형 테이블로 구성하여 사용자가 다룰 수 있도록 하는 XML-뷰 기반의 방법을 제시하였다.

XML-뷰는 SQL에서 제공하는 뷰(view)의 개념을 확장시켜 만든 것으로 정의와 사용방법이 뷰를 다룰 때와 비슷한 방식이 되도록 하였다. SQL에서 뷰는 SELECT 질의의 결과를 보통의 테이블처럼 다룰 수 있게 해준다. 뷰는 실제 저장된 데이타를 가지는 것이 아니고 뷰를 사용하는 시점에서 SELECT 질의를 수행하여 결과를 얻어내 사용자에게 보여 주는 방식으로 동작을 한다. XML-뷰도 이와 같이 데이타베이스 내에 데이타를 저장하고 있는 것이 아니라 필요한 순간에 데이타를 생성한다. 다른 점은 XML-뷰를 정의할 때는 XML 질의

문이 필요하다는 것이다. XML 데이타의 검색을 위해서 새로운 질의어인 XML 질의어를 사용하는 이유는 XML의 특성상 엄격한 스키마를 필요로 하는 기존 데이타베이스의 질의어를 사용하기가 적합하지 않기 때문이다.

XML-뷰 정의의 방법도 뷰를 정의하는 문법과 유사하여 자연스럽게 느낄 수 있는 방법을 택하였다. SRP에서 제공하는 SQL 문법에 XML-뷰 정의를 위한 하나의 문법을 추가했는데, 이에 대해서는 3.1절에 자세한 설명이 있다.

정의된 XML-뷰는 보통의 테이블, 뷰과 똑같이 사용할 수 있다. 본 논문에서는 XML 데이타에 대한 읽기만을 생각했기 때문에 SELECT 질의문에서만 사용가능하며 수정, 삭제 등을 위한 질의문에서의 사용은 허용하지 않는다.

3.1 XML-뷰의 정의

XML 문서를 접근하기 위해 사용자는 그 문서에 대하여 먼저 XML-뷰를 정의해야 한다. XML-뷰 정의를 위하여 SRP의 SQL문법에 아래와 같은 문법을 추가하였다.

```
<?xml version="1.0"?>
<!DOCTYPE personnel SYSTEM "personal.dtd">
 <personnel>
   <person id="Big.Boss" >
     <name>
        <family>Boss</family>
        <given>Big</given>
     </name>
     <email>chief@foo.com</email>
     <link subordinates="one.worker two.worker
           three.worker four.worker five.worker"/>
   </person>
   <person id="one.worker">
     <name>
       <family>Worker</family>
       <given>One</given>
     </name>
     <email>one@foo.com</email>
     <link manager="Big.Boss"/>
   </person>
. . . . . .
<person id="five.worker">
     <name>
       <family>Worker</family>
       <given>Five</given>
     </name>
     <email>five@foo.com</email>
     <link manager="Big.Boss"/>
   </person>
 </personnel>
```

(a) XML 데이타의 예

```
CREATE XMLVIEW xview_1 ( id CHAR(20), email(CHAR
        (30) ) AS
(' select p.personnel.person@id , p.personnel.person.email
   from "FILE:/home/user1/personal.xml" p; ' );
```

(b) XML-뷰의 정의

그림 1 XML 데이타와 XML-뷰의 정의

CREATE XMLVIEW XML-뷰이름(테이블 스키마 정의) AS ('XML 질의문장');

그림 1에는 XML 데이타의 예와 이 XML 데이타에 대한 XML-뷰 정의의 예가 있다. 'CREATE XML VIEW' 이라는 키워드 다음에 SQL에서 테이블을 선언하는 것처럼 컬럼들의 이름과 그 타입을 기술한다. 뷰의 경우에는 결과로 나오는 값의 타입이 이미 결정되어 있지만 XML-뷰의 경우에는 XML 질의의 결과로 추출되는 값에 타입 개념이 없기 때문에 사용자가 그 타입을 결정해 주어야 한다. 그리고 AS 뒤에는 XML 질의 처리기가 수행할 XML 질의 문장을 기술한다. 테이블 스키마 정의에 나타난 컬럼의 수와 XML 질의어의 결과의 컬럼의 수는 일치해야 한다. XML 질의의 결과로 나온 데이타는 스키마 정의에서 기술된 타입으로 변환이 된다.

그림 1-(a)는 파일시스템에 저장되어 있는 XML 문서의 예로서, 개인 신상 정보를 포함하고 있으며, 그림 1-(b)는 이 XML 문서에서 개인의 id와 email만을 추출하도록 정의된 XML-뷰 정의문이다.

3.2 XML-뷰의 접근

사용자는 관계형 데이타베이스 시스템에서 정의된 XML-뷰를 이용하여 XML 데이타를 접근하게 된다. 정의된 XML-뷰는 SQL에서 테이블이나 뷰를 사용할 때와 같기 때문에 사용자에게 자연스럽다. 그림 2에는 XML-뷰와 관계형 데이타베이스의 데이타를 조인하여 사용하는 모습을 보이는 SQL 질의문의 예가 있다.

그림 2의 질의의 예에서 table1은 관계형 데이타베이스에 저장되어 있는 데이타이고, xview1은 그림 1-(b)에서 정의된 XML-뷰 이다. XML-뷰의 데이타는 XML 질의의 결과로 얻어지지만 형태는 완전히 관계형 데이타베이스의 데이타처럼 사용자에게 보여지므로 XML 질의를 몰라서 XML-뷰를 정의하지 못하는 사용자라도 이미 정의된 XML-뷰를 사용하는 것은 아무런 지장이 없다.

3.3 XML-뷰 정의를 위한 질의어

XML 데이타에서 원하는 데이타를 추출하여 XML-뷰를 만들기 위해서, 본 논문의 구현 시스템에서는 XML을 위한 간단한 질의어를 정의하고 구현하였다. 본 논문에서 사용한 XML 질의어는 Lorel[3]의 기본 개념과 문법을 참고로 하였으며 현재는 간단한 기능들을 지원하고 있다. Lorel이 XML만을 위한 질의어가 아니라 일반적인 반구조적 데이타를 대상으로 하는 질의어이기 때문에 본 논문의 XML 질의어도 반구조적 데이타를 질의 대상으로 할 수가 있다.

3.3.1 질의어의 구문 형식

본 논문에서 제시된 방법으로는 XML 데이타를 읽기

전용으로만 사용할 수 있기 때문에 XML 질의어도 SELECT 문장만을 지원하며, 문법은 정의 3.1과 같다.

정의 3.1 (XML-뷰 정의를 위한 XML 질의어의 문법)

SELECT *path_expression_list*
FROM *'xml-file' filename-alias*
WHERE *path_expression operator 'value'*;

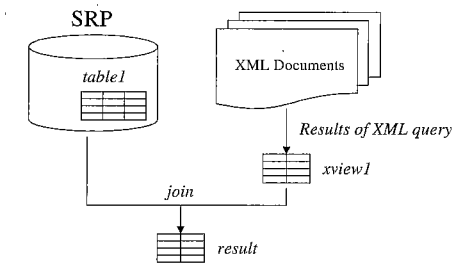

그림 2 XML-뷰를 사용한 질의의 예

SELECT 절에는 추출할 경로 이름들이 나열된다. 각 추출된 값의 타입은 XML-뷰의 정의문에서 기술한 테이블 스키마의 컬럼 타입에 따라 결정된다. FROM 절에는 XML 파일의 이름과 경로이름에서 사용할 파일이름의 별명(alias)을 기술하게 된다. XML 파일의 이름은 따옴표 사이에 기술해야 되며 현재 FILE: 과 DB:로 시작하는 형태를 지원하는데 FILE:로 시작하는 것은 SRP 서버가 있는 파일 시스템에서의 경로를 나타내며, DB:로 시작하는 것은 SRP의 저장 장치에 LOB의 형태로 저장되어 있는 파일을 나타낸다. 현재는 FROM절에서 하나의 XML 파일 이름만을 기술할 수 있다.

WHERE 절에서는 하나의 조건식을 기술할 수 있는데 하나의 경로 이름과 비교 연산자, 그리고 비교할 값으로 구성된다. 비교 연산자로는 '=', '<>', '<', '>', '>=', '<=', 'grep' 사용할 수 있다. 'grep'조건은 문자열이 전체 문자열의 일부로 포함될 때 참이 된다.

SELECT 절과 WHERE 절에서 데이타를 가리키는 방식으로 경로 이름이 사용된다. 경로 이름은 점(.)으로 연결된 경로를 통해 데이타를 나타내는 것으로 일반적인 반구조적 데이타를 가리킬 때 필수적인 요소로 인정되어 있다. 제시된 질의어에서의 경로 이름은 와일드 카드(*) 문자와 선택적인 경로를 포함할 수 있도록 하였다.

예제 3.1 (XML 질의어) XML-뷰를 정의할 때 사용하는 XML 질의어의 예이다.

SELECT *lib.book.title , lib.book.(date).year*
FROM *'DB:library.xml' lib*
WHERE *lib.book.* = 'John Smith';*

예제 3.1은 XML-뷰 정의문에 포함되는 XML 질의문의 예이다. 예제 3.1의 XML 질의 문장에서 SELECT 절에서는 lib.book.(date).year, WHERE 절에서는 lib.book.* 이라는 경로 이름이 쓰였다. lib.book.(date).year는 lib.book.year와 lib.book.date.year의 두 가지 경로이름을 나타내며, lib.book.*는 lib.book으로 시작하는 모든 경로 이름을 나타낸다. 이들을 이용하여 데이타의 구조를 완전히 알지 못하는 경우나 데이타의 구조가 약간 불규칙한 경우에도 질의를 할 수가 있다. 또한 경로식에서 애트리뷰트와 엘리먼트의 구분을 두지 않고 사용하였다. 하지만 * 문자나 정규 경로식은 사용자가 의도하지 않았던 경로 이름까지 포함할 수 있는 위험이 있으므로 주의해야 하며 이는 온전히 사용자의 책임이다.

본 논문은 XML-뷰 방법을 이용하여 XML 문서를 관계형 데이타베이스의 테이블과 같은 방법으로 접근하는 것에 중점을 두어 구현되었기 때문에 XML 문장을 만드는 다양한 방법에 관해서는 생략한다.

3.3.2 XML-뷰 정의문의 형식

사용자는 XML 질의문을 XML-뷰 정의문에 포함시켜 사용한다. 예제 3.2는 XML-뷰 정의에 대한 예이다. 여기서 XML-뷰 정의는 논문 목록에서 1999년에 발행된 technical report의 id와 저자(author)를 추출하여 컬럼이 두 개인 테이블의 형태로 만드는 모습을 보여 주고 있다.

예제 3.2 (XML-뷰 정의) XML 질의어를 이용하여 XML-뷰를 정의한 예로서 AS 이하 절에 들어가는 질의어는 XML 질의어이다

```
CREATE XMLVIEW xview_2 (id CHAR (40),
                author CHAR (30))
AS ('select bib.*.techreport@id,
          bib.*.techreport .author
from"DB:bibliography.xml" bib
where bib.*.techreport.year = "1999";');
```

XML 질의문 안에서 나타난 이중 따옴표는 SQL에서 하나의 따옴표를 나타내기 위해 사용된 것이다. XML 질의의 결과는 XML-뷰 이름 뒤에 선언된 테이블 스키마에 정의된 타입으로 변환이 되는데, 현재는 SRP에서 제공하는 데이타 타입들 중에서 문자열을 위한 CHAR(n) 타입과 정수 타입을 나타내는 INTEGER만을 지원하고 있다. 질의어 중의 @는 애트리뷰트를 가리킨다. 즉 techreport의 애트리뷰트 중 id를 지칭하는 것이다.

3.3.3 XML 질의의 결과

XML 질의의 결과는 정의된 테이블 형식으로 변환이 된다. 하지만 데이타들이 본래 관계형 데이타베이스의 데이타들이 아니기 때문에 발생할 수 있는 몇 가지 문제가 있다. 예제 3.3의 XML-뷰 정의에 나타난 XML 질의문은 도서관의 책 목록에서 제목에 XML이 포함된 책의 제목과 저자, 발행 연도를 추출하는 모습을 보이고 있다. 이 질의 문장에서 발생할 수 있는 문제 중 하나는 해당 결과의 원자성에 관한 문제이다. 예를 들어 XML 문서에서 책의 제목이 lib.book.title라는 경로 이름으로 접근 가능한 경우만 존재하는 것이 아니라, 주제목과 부제목으로 나누어져 각각 lib.book.title.main, lib.book.title.sub로 접근 가능한 경우가 있을 수 있다. 그러나 현재의 XML 질의 처리기는 사용자가 지정한 경로 이름이 하위 항목으로 나누어지는 경우에, 그 데이타는 처리할 수 없다.

예제 3.3 (XML-뷰) XML 질의어를 이용하여 XML-뷰를 정의한 예이다.

CREATE XMLVIEW *xview_3*
(title CHAR (50) , author CHAR (30), year INTEGER)
AS
('select lib.book.title, lib.book.author, lib.book.pub -year
from "DB:library.xml" lib
where lib.book.title grep "XML";');

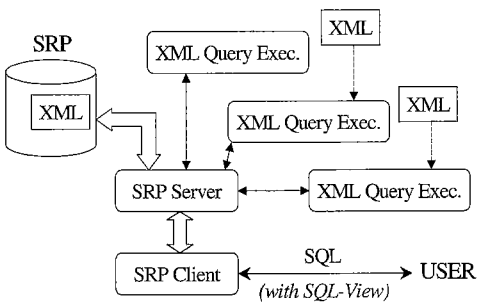

그림 3 시스템의 전체 구조

다른 문제로 집합을 값으로 가지는 결과를 생각할 수 있다. 위의 XML 질의에서 하나의 책에 대하여 제목은 하나가 나오지만 저자는 여러 명이 나올 수 있다. 그런데, SRP에서는 애트리뷰트가 여러 개의 값, 즉 집합이 되는 것을 지원하지 않기 때문에 이 경우에는 하나의 튜플로 만들 수가 없다. 현재는 이런 경우에 애트리뷰트의 중복을 이용하여 여러 개의 튜플로 만드는 방식으로 해결하고 있다. 예를 들어 제목과 발행 연도 하나와 저자 세 명이 결과로 나왔다면 이는 제목과 발행 연도가 세 번 중복된 세 개의 튜플로 만들어진다.

4. 시스템 설계 및 구현

4.1 전체 구조

그림 3은 전체 시스템의 구조를 나타내고 있다. 기존의 SRP 시스템에 XML 질의처리기가 추가된 구조이며, 사용자가 데이타베이스 시스템에 접근하는 방식이나 SRP 서버와 SRP 클라이언트간에 통신 방식, 그리고 SRP의 저장 장치를 이용하는 방식에는 변화가 없다. XML 질의처리기는 사용자가 XML-뷰를 접근하는 순간에만 실행이 되고, SRP 서버 외부에 위치하여 메시지 큐를 통하여 SRP 서버와 질의문과 그에 따른 결과 데이타를 주고받는다. 하나의 XML 질의 처리기는 하나의 XML 질의만을 수행하기 때문에 동시에 여러 개의 XML 질의 처리기가 실행될 수 있다.

XML 질의 처리기가 데이타베이스 시스템의 외부에서 동작하는 구조는 전체적인 시스템의 관리 측면에서 장점을 가지게 된다. XML 질의 처리기의 변경이나 대체가 데이타베이스 시스템에 영향을 주지 않는 다는 것이다. SRP 서버와 XML 질의 처리기 사이의 데이타를 주고받는 방식만 일관되게 유지하면 데이타베이스 시스템이 수행중이라도 수행 중단 없이 XML 질의 처리기의 변경이나 새로운 질의 처리기로의 대체 등의 작업을 해낼 수가 있다.

사용자는 SRP 클라이언트의 SQL 환경을 이용하여 SRP 서버가 위치한 파일 시스템에 있는 XML 파일과 SRP 저장 장치에 저장되어 있는 XML 파일에 대해 XML-뷰를 정의할 수 있다. 현재 SRP 서버는 UNIX 환경에서 동작하며, SRP에서 XML 파일은 LOB의 형태로 저장된다.

4.2 SRP 시스템으로의 구현

제안한 방식을 SRP 시스템에서 구현하기 위해 SRP 클라이언트와 SRP 서버에 몇 가지 부분을 추가, 변경하였다. SRP 클라이언트에서는 XML-뷰를 정의하기 위한 SQL 문법의 확장과 XML-뷰 정의와 접근 연산을 위한 물리적 연산자를 질의 수행 계획에 포함하는 작업 등이 이루어졌고, SRP 서버에서는 XML-뷰를 정의, 접근하기 위한 물리적 연산자를 실제로 처리하는 부분과 XML 질의 처리기와 데이타를 주고받는 부분을 추가하는 작업등이 이루어졌다.

4.2.1 SRP의 질의 처리와 수행 계획

SRP 시스템에서는 사용자의 SQL 질의문을 수행 계획으로 만들어 처리하게 된다. 사용자의 질의문이 입력되면 SRP 클라이언트에서는 수행 계획을 만들어 SRP 서버로 전달한다. SRP 서버에서는 전달받은 수행 계획을 처리하게 되는데 수행 계획은 각 단계의 작업을 담당하는 여러 개의 물리적 연산자의 트리로 구성되어 있다. 물리적 연산자는 수행 계획을 이루는 단위가 되며 SRP에서는 SPROP(SRP Physical Relational OPerator)이라고 부른다. XML-뷰를 다루기 위해 물리적 연산자의 추가가 필요하여, XML-뷰의 정의와 접근을 위한 두 개의 물리적 연산자를 추가하였다.

모든 SPROP은 open(), next(), close() 등의 공통된 인터페이스를 가진다. 이 중 open()은 수행에 필요한 준비를 담당하고, next()는 여러 개의 튜플을 처리할 때 하나의 튜플씩 접근하는 작업을 담당하며, close()는 수행을 마치기 위한 작업을 담당한다. 여러 개의 튜플 접근이 필요 없는 SPROP에서는 open()에서 모든 작업을 처리한다.

4.2.2 XML-뷰 정의의 구현

XML-뷰를 정의할 때 SRP 데이타베이스 시스템의 동작은 그림 4에 잘 나타나 있다. 사용자는 앞에서 보여주었던 예와 같이 CREATE XMLVIEW로 시작하는 XML-뷰 정의 문장을 이용하여 XML-뷰를 만들게 된다. XML-뷰를 정의하는 문장을 받았을 경우, SRP 클라이언트에서 최종적으로 하는 일은 XML-뷰 생성을 담당하기 위해 추가한 물리적 연산자인 SPROPCreate XMLVIEW가 포함된 수행 계획을 만들어 SRP 서버로 전달하는 것이다.

SRP 서버에서는 전달받은 수행 계획을 처리하는 과정에서, SPROPCreateXMLVIEW을 처리할때, SRP 카탈로그 저장소에다 사용자가 정의해 놓은 테이블의 스키마정보, 컬럼의 정보 등과 함께 XML 질의문장을 저장한다. 이는 뷰를 생성할 때와 비슷한 작업인데, 뷰를 생성할 때에도 뷰를 정의한 질의 문장을 카탈로그에 기록해 놓는다.

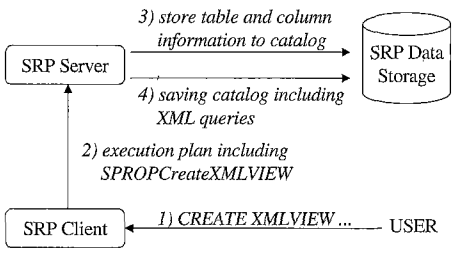

그림 4 XML-뷰 정의 동작 과정

이 과정에서 사용되는 XML-뷰 정의를 위한 물리적 연산자에 대해 살펴보기로 하겠다. XML-뷰 정의 문장을 보면 테이블을 선언할 때의 문법과 같이 각 컬럼들의 타입을 기술하는 것을 볼 수 있다. 그리고 사용하는 질의어는 다르지만 뷰를 정의할 때처럼 결과를 추출할 질의문을 입력하도록 하고 있다. 따라서 XML-뷰를 위한 SPROP은 테이블과 뷰를 생성할 때 사용되는 SPROP인 SPROPCreateTBL, SPROPCreateView의 기법을 혼합하여 사용하기로 하였다. 두 SPROP의 open() 메쏘드가 공통적으로 하는 일은 카탈로그에 사용자가 정의한 테이블과 컬럼의 정보를 기록하는 것이다. 두 SPROP의 차이점은, 테이블을 생성할 때만 실제 데이타가 저장될 장소가 만들어지고, 뷰를 생성할 때는 실제 저장 공간은 만들지 않고 뷰를 정의한 질의문을 그대로 카탈로그에 저장하는 작업을 한다는 것이다. XML-뷰의 경우는 테이블 스키마 선언은 테이블 정의 때와 같고, 실제 데이타는 뷰와 같이 SRP 내에 저장 공간을 가지지 않기 때문에 각 작업에서 테이블 정의와 뷰 정의에서의 기법을 적절히 사용하였다.

4.2.3 XML-뷰 접근의 구현

XML-뷰를 접근하는 경우는 SELECT 질의의 FROM 절에 XML-뷰가 포함되는 경우이다. 클라이언트에서 SELECT 질의문을 파싱하여 수행 계획을 만드는 과정에서 FROM 절에 기술된 테이블이나 뷰, 또는 XML-뷰의 이름들을 보게 되는데 클라이언트에서는 이 이름만 보고는 이것이 테이블인지, 뷰인지, XML-뷰인지 판단할 수가 없다. 수행 계획을 만들기 위해서는 이것을 알아야 하는데 이를 위하여 SRP 클라이언트는 SRP 서버를 통해 카탈로그에 있는 정보를 받아온 후에 판단한다.

여기서 FROM 절에 나타난 이름이 XML-뷰인 경우에는 XML-뷰 접근을 담당하는 물리적 연산자인 SPROPXMLVIEWAccess를 포함하는 수행 계획을 만들어 SRP 서버에 전달한다. SRP 서버에서는 전달받은 수행 계획을 처리하는 과정에서 SPROPXMLVIEW Access를 처리할 때, XML 질의 처리기를 수행시키고 SRP 저장 장치에서 XML-뷰의 카탈로그 정보에서 정의된 XML 질의문을 꺼낸 다음 이 질의문을 XML 질의 처리기로 전달한다.

XML 질의 처리기가 XML 질의문을 전달받으면 XML 질의문에 대한 수행이 시작된다. XML 질의 처리기는 FROM 절에 기술된 XML 파일을 가져오고, 그 후에는 SELECT 절에 있는 경로 이름들과 XML 파일의 이름, 그리고 WHERE 절에 있는 조건을 추출한다.

가져온 XML 문서를 DOM(Document Object Model) [17]으로 바꾸고 이 DOM을 단순한 방법으로 순회하면서 조건에 맞는 데이타들을 추출하는 것이다. 이와 같은 처리 방법은 그림 5에 잘 나타나 있다. 본 논문에서는 XML 문서를 DOM으로 변환, 순회하는 과정에서 IBM에서 제공하는 XML for C++[18]의 XML 파서와 DOM API 라이브러리를 이용하였다.

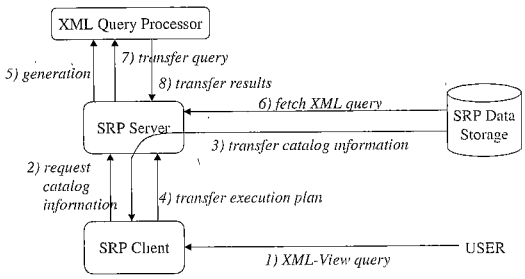

그림 5 XML-뷰 접근 동작 과정

현재의 XML 질의 처리기는 질의 최적화를 하지 않는데, 인덱스 기법이나 XML의 스키마 정보를 분석하여 사용하지 않는 것이다. 스키마 정보를 이용하지 않으므로 질의 처리 과정에서 최적화 작업등을 할 수가 없어서 때문에 현재의 XML 질의 처리기는 성능 면에서 좋지 않다.

XML 질의 처리기가 XML 질의 수행을 마치면 결과를 튜플 단위로 메시지 큐에 보내게 된다. 보낸 결과는 SRP 서버에서 필요한 시간에 하나씩 전달이 되어 SRP 데이타로 변환이 된 후 사용된다. XML-뷰의 정의에서와 마찬가지로, 정의된 XML-뷰의 접근을 담당하는 물리적 연산자가 필요하여 SPROPXMLVIEWAccess라는 SPROP을 만들었다. SRP에서 저장된 테이블에서 튜플을 가져오는 작업을 담당하는 물리적 연산자는 SPROPTblAccess이다. 이 SPROP이 실제 저장장치 속에서 테이블에 해당하는 파일에서 튜플들을 읽어 오는 역할을 한다. 뷰를 접근하는 경우는 이와는 다른데, 뷰는 실제 물리적으로 저장된 파일을 가지지 않는다. 그 대신 SRP 클라이언트에서 질의 문장을 파싱하여 수행 계획을 만드는 중에 뷰를 접근하는 부분이 있으면 서버를 통해 카탈로그에 있는 뷰를 정의한 질의문을 가져오고, 가져온 질의문을 파싱하여 질의 계획을 만든다. 즉, 테이블을 접근하는 경우와는 달리 뷰는 하나의 물리적 연산자 대신에 SELECT 질의 문장을 파싱하여 만들어낸 수행 계획 전체가 담당을 하는 것이다.

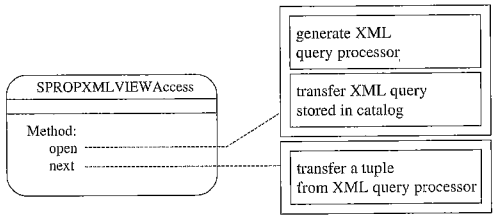

그림 6 XML-뷰 접근을 위한 SPROP

XML-뷰도 뷰와 마찬가지로 SRP 내에 실제 저장된 데이타를 가지지 않는다. 하지만 뷰와는 달리 테이블 연산을 담당하는 수행 계획으로 대체해서 처리하는 것이 아니라, 그림 6과 같이 XML-뷰 연산을 담당하는 물리적 연산자인 SPROPXMLVIEWAccess가 사용된다.

SPROPXMLVIEWAccess의 open() 메쏘드가 불리면 하나의 XML 질의 처리기를 수행시키고 카탈로그에서 XML-뷰의 정의문, 즉 XML 질의문을 꺼내서 XML 질의 처리기에 전달한다. 실제 데이타 요구는 next() 메쏘드에서 이루어진다. next() 메쏘드가 불리면 SRP 서버는 XML 질의처리기와 연결된 메시지 큐를 통하여 하나의 튜플에 해당하는 데이타를 받아서 이것을 SRP 내부의 튜플 형식으로 고친 후에 상위 레벨의 물리적 연산자로 전달한다. SQL 문의 WHERE 절에 조건이 있는 경우에는 전달받은 데이타가 조건에 맞는지를 확인하고, 조건에 맞지 않을 때는 조건에 맞는 튜플을 찾을 때까지 next()를 호출하게 된다. 데이타 전달 과정 이외의 작업은 테이블에서 데이타를 읽어 올 때와 같기 때문에 테이블 접근 때의 기법을 그대로 활용할 수가 있다.

4.2.4 성능

본 논문에서 제안한 테이블 함수는 XML 질의를 처리하는 외부 프로세스로 동작하게 된다. 이때 외부 프로세스와 계속적인 통신이 이루어지게 되며 이것은 매번 next 호출이 발생할 때마다 계산이 이루어지게 되므로 성능을 저하시킬 수 있다. SRP에서는 [12]와 같이 사용자 정의 함수를 외부 프로세스로 동작시킬 때 발생할 수 있는 성능 저하를 막기 위하여 함수 호출에 대한 결과를 캐싱하는 방법을 사용하고 있다. 본 논문의 테이블 함수는 사용자 정의 함수와 같이 동작하며 XML 질의의 결과가 변하지 않기 때문에 [12]에서 제안한 캐싱 방법이 그대로 적용된다. 그러므로 이와 같은 방법을 이용하면 성능 저하를 줄일 수 있다.

5. 결론 및 향후 연구 계획

본 연구에서는 XML 문서를 관계형 테이블 형식으로

바라볼 수 있도록 하기 위한 방법을 제안하고 구현하였다. 이를 위해, 우리는 기존의 뷰 개념을 확장하여 XML-뷰라는 새로운 형식을 제안했으며, 이 뷰의 정의를 수행하기 위한 XML 질의 처리기를 구현하였다. 이러한 작업의 결과로서, 데이타베이스 사용자에게 XML 데이타에 대한 테이블 형식의 일관된 접근 방법을 제공하며, 사용자의 요구가 바뀌더라도 XML-뷰의 정의를 바꿈으로써 적은 비용으로 이를 처리할 수 있도록 하였다. 앞으로, XML-뷰의 질의 능력 향상을 위한 XML 질의 처리 시스템에 대한 연구와 데이타베이스의 다른 여러 기능들의 적용에 대한 연구가 필요할 것으로 생각한다.

참 고 문 헌

[1] T. Bray, J. Paoli, and C. M. Sperberg-McQueen, *Extensible Markup Language (XML) 1.0.*, W3C Recommendation, February 1998.

[2] Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, and Jennifer Widom, Lore: A Database Management System for Semistructured Data, *SIGMOD Record*, 26(3), 9 1997.

[3] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener, The Lorel Query Language for Semistructured Data, *International Journal on Digital Libraries*, 1(1):68-88, April 1997.

[4] Michael Stonebraker and Paul Brown, *Object-Relational DBMSs Tracking The Next Great Wave*, Morgan Kaufmann, 2 edition, 1999.

[5] Alin Deutsch, Mary Fernandez, and Dan Suciu, Storing Semistructured Data with STORED, *SIGMOD*, 1999.

[6] Daniela Florescu and Donald Kossmann, Storing and Querying XML Data using an RDBMS, *Data Engineering Bulletin*, 22(3), September 1999.

[7] Jayavel Shanmugasundaram, Kristin Tufte, Chun Zhang, Gang He, David J. DeWitt, and Jeffrey F. Naughton, Relational Databases for Querying XML Documents: Limitations and Opportunities, *VLDB*, 1999.

[8] Jayavel Shanmugasundaram, Eugene Shekita, Rimon Barr, Michael Carey, Bruce Lindsay, Hamid Pirahesh, and Berthold Reinwald, Efficiently Publishing Relational Data as XML Documents, *VLDB*, 2000.

[9] 안정호, 김형주, SRP 에서 SOP까지, *한국정보과학회 Review지*, 4 1994.

[10] Jae-Mok Jeong, Sangwon Park, Tae-Sun Chung, and Hyoung-Joo Kim, XWEET: XML DBMS for Web Environment, *The First Workshop on Computer Science and Engineering 2000*, Seoul, Korea, pages 16-17, June 2000.

[11] 박상원, 민경섭, 김형주, XML 데이타베이스 지원을 위한 통합 환경, *정보과학회 논문지(CP)*, 6(6), 2000.

[12] 고정미, 정재목, 김형주, 관계형 데이타베이스 시스템에서의 사용자 정의 함수 지원, *한국정보과학회 논문지(C)*, 5(3), June 1999.

[13] Paolo Atzeni, Giansalvatore Mecca, and Paolo Merialdo, To Weave the Web, *VLDB*, 1997.

[14] G. Mecca, P. Merialdo, and P. Atzeni, Araneus in the Era of XML, *Bulletin of the Technical Committee on Data Engineering*, 22(3), 1999.

[15] Takeyuki Shimura, Masatoshi Yoshikawa, and Shunsuke Uemura, Storage and Retrieval of XML Documents Using Object-Relational Databases, *DEXA*, 1999.

[16] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu, A Query Language for XML, *In Proceedings of Eighth International World Wide Web Conference*, 1999.

[17] V. Apparao et al, *Document Object Model (DOM) Level 1 Specification Version 1.0*, W3C Recommendation, October 1998.

[18] IBM Corporation, *XML for C++*, http://www.alphaworks.ibm.com/tech/xml4c, 1999.

이 제 민

1998년 서울대학교 컴퓨터공학과(학사). 2000년 서울대학교 컴퓨터공학과(석사). 현재 한국정보공학(주) 연구소 관심분야는 데이타베이스, XML

민 경 섭

1995년 2월 항공대학교 컴퓨터공학과 학사. 1997년 2월 서울대학교 컴퓨터공학과 석사. 1997년 ~ 현재 서울대학교 인지과학전공 박사과정. 관심분야는 데이타베이스, XML, Semistructured data, Web

박 상 원

1994년 서울대학교 컴퓨터공학과(학사). 1997년 서울대학교 컴퓨터공학과(석사). 1997년 ~ 현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 데이타베이스, XML, Semisturctured data, Web.

김 형 주

정보과학회논문지 : 데이타베이스 제 28 권 제 1 호 참조