XML 데이타베이스 지원을 위한 통합 환경
(Unified Framework for XML Database Support)

박 상 원 † 민 경 섭 † 김 형 주 ††
(Sangwon Park) (Kyung-Sub Min) (Hyoung-Joo Kim)

요 약 웹에서 정보 전달의 수단으로 등장한 XML은 많은 응용 분야에서 사용될 것이다. 많은 양의 XML 문서를 효율적으로 다루기 위하여 데이타베이스의 지원은 필수적이다. 데이타베이스를 이용하여 XML 데이타를 처리할 때 데이타베이스의 종류 뿐만 아니라 그 인터페이스도 중요한 문제이다. 본 논문에서는 관계형 데이타베이스, 객체지향형 데이타베이스 및 랩퍼를 이용하여 XML 데이타를 저장, 질의하며 그 인터페이스로 XML-뷰, ODMG C++ 바인딩, OQL, DOM을 사용하였다. 또한 각각의 시스템의 구현을 통하여 각 방법들의 장단점을 논하고, 효율적인 XML 문서 처리에 대한 방법을 제시한다.

Abstract XML will be used in lots of areas in the Web environment as a method of information exchange. We have to use databases to manipulate lots of XML documents efficiently. When we use database to manipulate XML, not only type of database but also its interface is important. We develop a system using relational database, object-oriented database and wrapper to store XML data, of which interfaces are XML-View, ODMG C++ binding, OQL and DOM. We discuss pros and cons of each method by the implementation of the system, and propose an efficient manipulation method of XML documents.

1. 서 론

XML은 웹 상에서의 데이타 교환을 위해 제안된 표준 언어이다. 웹에서의 자유로운 데이타 교환을 위해, XML은 문서 자체에 문서의 구조를 기술하고 있으며, 사용자가 문서의 구조를 원하는 대로 정의할 수 있게 한다. 이러한 구조적 유동성은 모든 형식의 데이타가 XML 형태로 기술될 수 있도록 해주며, 웹에서 운용되는 모든 데이타가 통합, 저장, 처리될 수 있는 기반을 제공한다.

한편, 최근들어 기존의 레거시(legacy) 시스템들로부터 서로 다른 데이타 형식의 데이타를 통합, 가공하여 사용자에게 새로운 정보를 제공하고자 하는 연구가 활발히 진행되고 있다. 이러한 형태의 시스템을 중계자(mediator)라 한다. 중계자는 이질적인 형식의 데이타를 통합하여 가공하기 위해, 정보 소스에 독립적인 데이타 언어를 제공해야 하며, 이를 이용하여 데이타에 대한 접근이 가능해야 한다. 또한, 각 시스템으로부터 얻어온 정보를 지역적으로 저장하기 위한 방법도 제공해야 한다. 이에, 중계자들은 나름대로의 저장 시스템과 데이타 표현언어를 정의하고 있다. 하지만 이러한 중계자들은 또 다른 레거시 시스템이 될 수 있으며, 여러 시스템 간의 자유로운 데이타 교환과 통합을 통한 정보 생성에 어려움을 줄 수 있다.

XML은 여러 시스템 간의 정보 교환의 표준을 제공함으로써 이러한 문제를 해결할 수 있다. 앞으로 많은 데이타가 XML로 표현될 것이고 이런 데이타를 효율적으로 관리하기 위해서 데이타베이스의 사용은 필수적이다. XML을 처리하기 위한 데이타베이스는 크게 두가지 방법으로 구현할 수 있다. 첫번째 방법으로 Lore[1]와 같이 새로운 데이타 모델을 지원하는 데이타베이스 시스템을 하부 저장장치로부터 질의어 처리기까지 모두 만드는 것이다. 이와 같은 방법의 장점은 새로운 모델을 부조화(impedance mismatch)없이 잘 표현할 수 있고 질의할 수 있다는 점이다. 그러나 [2]에서 말한 바와 같

· 이 논문은 2000년도 두뇌한국21사업에 의하여 지원되었음.
† 비 회 원 : 서울대학교 컴퓨터공학부
swpark@oopsla.snu.ac.kr
ksmin@oopsla.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@oopsla.snu.ac.kr
논문접수 : 2000년 4월 14일
심사완료 : 2000년 9월 21일

이 시스템의 안정성 및 구현 완료 가능성이 떨어진다는 것이 문제이다. 새로운 시스템을 구현하면 성능 및 안정성에 관한 확신이 있을 때까지 실제 문제에 적용하는 것이 어렵다. XML 데이타 모델은 트리 형태로서 릴레이션을 모델로 하는 관계형 데이타베이스와는 데이타 모델이 상이하다. 그러므로 기존의 관계형 데이타베이스 시스템에서 많이 연구되어 왔던 질의어 최적화, 동시성 제어 등의 기능을 잘 활용할 수 없는 단점이 있다.

두번째 방법은 [3, 4, 5]와 같이 기존의 데이타베이스 시스템을 이용하여 XML 데이타를 저장하고 질의를 수행하는 것이다. 이것은 XML 문서를 저장하기 위한 적절한 테이블이나 클래스를 생성한 다음 XML에 대한 질의를 데이타베이스에 대한 질의어로 변환하여 원하는 결과를 얻는 방법이다. 이렇게 함으로써 기존의 기술로 새로운 데이타 형태인 XML을 지원할 수 있게 되어 새로운 시스템을 구현하지 않고 랩퍼(wrapper)를 이용하여 원하는 결과를 생성할 수 있는 장점이 있다. 기존의 데이타베이스로는 관계형 데이타베이스와 객체지향형 데이타베이스 및 객체관계형 데이타베이스가 있다. 또한 저장 방법으로는 크게 두가지 방법이 있는데 첫째로는 XML 데이타를 하나의 큰 객체로 저장하는 방법이고, 두번째는 XML 데이타를 적절한 크기의 객체로 나누어 저장하는 방법이다. 적절한 크기의 객체로 나누어 저장하는 방법에도 XML 문서의 의미를 보존하여 저장하는 방법[5]과 구조적인 정보를 이용하여 저장하는 두가지 방법[4]이 있다. 예를 들어 전자는 person이라는 태그가 XML 문서 내에 있을 때 person의 의미를 살려 Person 테이블이나 클래스에 person 엘리먼트 객체를 저장하는 방법이다. 이것은 엘리먼트의 의미적인 정보를 유지할 수 있으므로 데이타베이스에 저장된 XML 문서에 의미있는 질의를 쉽게 작성할 수 있는 장점이 있다. 후자는 DOM[6] 객체의 형태로 저장하는 방법으로 엘리먼트를 저장하는 클래스에 person 태그를 저장하는 방법이다. 이것은 XML 문서의 구조적인 정보를 유지하기 때문에 XML 응용 프로그램에서 적용할 때 편리한 장점이 있다.

우리는 웹 상의 데이타 교환과 저장에 대한 표준인 XML을 이용한 중계자 시스템인 XWEET[1](XML DBMS for Web Environment)[7]를 제안하였다. 이것은 XML 데이타를 처리하기 위한 미들웨어 시스템으로서 기존의 레거시 시스템으로부터의 XML 형식의 데이타 접근과 중계자 시스템 수준에서의 XML 데이타 저장과 접근 방법에 대한 연구를 데이타베이스 처리 관점에서 수행한 것이다. 본 논문에서는 XWEET 시스템 중 특히 데이타베이스에 XML 데이타를 저장하는 PDM(Persistent Data Manager)을 중심으로 설명한다. XWEET/PDM은 관계형 데이타베이스[2]와 객체지향형 데이타베이스[3]에 위에서 말한 세 가지 방법을 이용하여 저장하는 모듈로 구성되어 있다. 관계형 데이타베이스에서는 LOB이나 파일에 XML 데이타를 하나의 객체로 저장한 후 XML-뷰를 이용하여 질의를 처리하는 방법을 이용하였다. 객체지향형 데이타베이스에서는 XML 객체를 DTD를 이용하여 분석한 후 엘리먼트의 의미를 보존하는 적절한 이름의 클래스의 인스턴스로 저장하여 ODMG C++ 바인딩[4]이나 OQL로 데이타를 접근하는 방법을 이용하였다. 또한 관계형 데이타베이스[5]에 XML 문서의 구조적인 정보인 DOM 형태로 저장한 후 JDBC를 지원하는 랩퍼를 이용하여 접근하는 방법을 이용하였다. 본 논문에서는 이와 같이 다양한 데이타베이스에 여러 방법을 이용하여 XML 데이타를 저장하여 그 방법들에 대한 비교를 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 XWEET 아키텍쳐에 대하여 살펴보고 3장에서는 XML 저장 방법에 관하여 본 논문에서 사용한 세가지 방법에 대하여 살펴본다. 4장에서는 관련연구를 살펴보고 5장에서 결론 및 향후 연구 방향에 대하여 살펴본다.

2. XWEET

이 장에서는 XWEET 시스템의 세부적인 시스템 구조에 대해 설명한다. 최근 들어 기존에 존재하는 많은 서로 다른 형태의 데이타를 통합, 저장, 접근하고자 하는 노력들이 급속히 늘어나고 있다. 또한 웹을 중심으로 한 많은 응용들이 등장하고 있다. XWEET 시스템은 여러 정보 소스에 존재하는 이질적인 구조의 데이타를 단일한 구조(XML)로 표현, 관리, 접근할 수 있는 미들웨어 시스템이다.

그림 1은 전체적인 XWEET 구조를 보여주고 있다. DATA Source는 XWEET에서 사용되는 여러 XML 문서를 받아 XDM(XWEET Data Model)로 변환하는

1) [swi:t]라 읽는다.

2) 본 논문에서 사용한 관계형 데이타베이스로는 SRP[8]를 이용하였다.

3) 본 연구에서 사용한 객체지향형 데이타베이스로는 ODMG를 지원하는 SOP[8]를 이용하였다.

4) SOP에서는 C++, CLOS[9], Java 바인딩을 지원하므로 세가지 언어로 응용 프로그램을 작성할 수 있다.

5) JDBC와 Linux용 Oracle을 이용하였다.

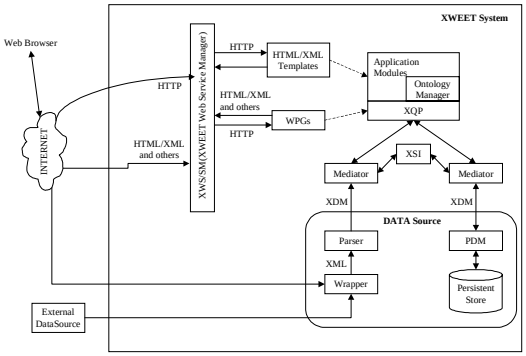

그림 1 XWEET 전체 구조

작업을 수행한다. XDM은 XWEET에서 사용하는 데이타 모델로서 DOM[6]을 간략화한 것이다.

DATA Source는 크게 랩퍼와 PDM(Persistent Data Manager)으로 구성되어 있다. 랩퍼는 웹 상의 여러 종류의 HTML 파일, e-mail, news, XML 파일과 같은 여러 외부 데이타 소스를 읽어 관심있는 정보만을 추출하여 XML 문서로 변환하는 모듈이다. PDM은 XWEET에서 직접 관리하는 XML 데이타들을 다루기 위한 모듈로, XML 데이타를 데이타베이스에 효율적으로 저장, 조작하는 기능을 제공한다.

XSI(XWEET Semantic Integrator)는 사용자로부터 받은 정보를 바탕으로 여러 데이타 소스를 통해 받은 의미적으로는 동일하지만 다른 방식으로 인코딩된 XML 문서를 동일한 뷰(view)로 볼 수 있도록 하는 기능을 제공하는 모듈이다. Mediator는 XSI와 협력하여 서로 다른 문법(syntax), 의미(semantic)를 사용하는 XDM 모델을 질의가 가능한 동일한 형태로 변환하는 일을 한다.

XQP(XWEET Query Processor)는 데이타 소스로부터 받은 여러 XML 데이타에 대한 선언적 질의를 처리하는 모듈이다. Application Module은 XWEET에서 제공하는 여러 서비스를 바탕으로 응용 프로그램 특징적인 작업을 처리하는 모듈이다. 이것은 XQP를 이용하여 작성할 수도 있고 Ontology Manager를 이용할 수도 있다. Ontology Manager는 스키마 통합이나 특정 응용 도메인에서 사용되는 용어들에 대한 통합 및 재정리를 담당하는 모듈이다.

실제 사용자가 응용 프로그램을 수행하는 것은 웹을 통해서이다. 사용자는 HTTP 프로토콜을 이용하여 XWSM(XWEET Web Service Manger)을 통해 서비스를 받는다. 이때 이 서비스들은 WPG(Web Page Generator)[10]나 HTML/XML Template을 통하여 처리된다. WPG는 클라이언트에서 전달되는 HTTP 요청을 바탕으로 웹 페이지를 생성하는 모듈로서 응용 프로그램은 WPG를 이용하는 CGI, 서블릿(servlet) 등으로 구현될 수 있다. HTML/XML Template은 WPG의 한 종류로서 웹 페이지를 보다 손쉽게 만들 수 있게 하는 모듈이다.

본 논문은 XML 데이타를 데이타베이스에 저장하는 여러 방법에 관한 것으로 XWEET 시스템에서 PDM 부분에 해당한다. PDM은 사용하는 데이타베이스의 종류, 저장 방법, 인터페이스에 따라 여러 방법으로 구현할 수 있다. 본 논문에서는 다양한 방법을 이용하여 XML 데이타를 데이타베이스에 저장하였다. 다음 장에서 PDM의 구현에 관하여 자세히 설명한다.

3. XWEET에서의 XML 지원

XML 데이타를 데이타베이스에 저장하여 처리하는 방법은 여러 가지로 분류할 수 있다. 첫번째로 파일, 관계형 데이타베이스, 객체지향형 데이타베이스 등 저장 시스템에 따른 분류 방법이 있다. 어떤 저장 시스템을 사용하느냐에 따라 데이타에 대한 접근 속도나 접근 용이성 및 모델링 방법이 달라진다.

두번째 분류 방법으로 저장 방법에 따른 분류가 있을 수 있다. XML 문서를 파싱 단계를 거치지 않고 전체를 하나의 객체로 저장하는 방법과 XML 문서를 파싱하여 적절한 형태로 가공한 후 저장하는 방법이 있다. 하나의 객체로 저장하면 실제 XML 데이타를 사용하는 시점에 문서를 적절한 내부 형태로 가공해야 하지만, 먼저 적절한 형태로 가공된 상태로 저장할 때는 파싱 시간이 줄어든다. 질의어 처리기에서 문서 전체를 접근할 경우에는 하나의 객체로 저장한 경우도 좋으나 문서의 일부분만 접근할 경우에는 여러 개의 객체로 저장된 경우가 유리하다. 하나의 객체로 저장하면 갱신 연산을 할 때 동시 접근성(concurrency)이 떨어지는 단점이 있다. 문서를 파싱하여 저장할 때도 두가지 방법이 있다. 하나는 문서에 나타난 엘리먼트의 의미를 살려서 저장하는 방법과 문서의 구조적인 형태를 그대로 저장하는 방법이 있다.

세번째 분류 방법으로 접근 방법에 따른 분류를 들 수 있다. 실제 XML 데이타를 접근하는 방법이 SQL, OQL과 같은 질의어일 수 있고, 혹은 랩퍼를 통하여 특정 인터페이스로 제공될 수 있다. 그 외의 방법으로 실제 데이타는 XML이 아니나 XML 데이타와 동일한 형태로 가공하여 사용자는 마치 XML 문서와 같이 사용

할 수 있게 하는 방법이 있다. 이번 장에서는 XWEET 시스템에서 구현한 여러 방법에 대하여 살펴보고 각각의 장단점을 비교한다.

3.1 PDM/RDB

관계형 데이타베이스에 저장하는 방법으로 XWEET에서는 크게 두가지 방법을 사용하였다. 먼저 XML 문서를 파싱하지 않고 하나의 객체로 파일이나 BLOB에 저장하는 방법과, XML 문서를 파싱하여 그 데이타 모델의 의미를 유지하는 여러 개의 객체로 나누어 저장하는 방법을 사용하였다. 본 절에서는 하나의 객체로 저장하는 방법에 대한 구현이고 3.3절은 여러 개의 객체로 나누어서 저장한 방법에 대하여 설명한다.

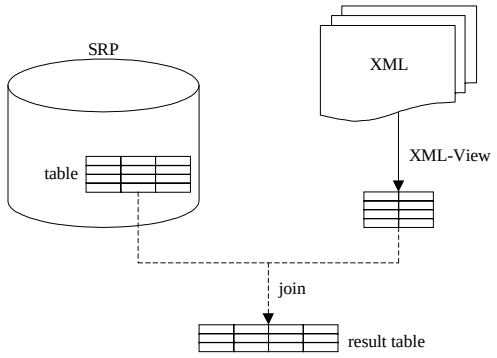

그림 2 XML-뷰를 사용한 질의의 예

PDM/RDB에서는 그림 2에서 보는 바와 같이 XML 데이타를 파싱하지 않고 파일 단위로 LOB이나 파일에 저장하거나 소켓을 통하여 파싱하지 않은 XML 문서를 가져온다. 이렇게 가져온 XML 파일은 XML-뷰를 통하여 테이블 형태로 보여진다. 그림 3은 XML 문서의 예로서 사람들에 대한 리스트를 나타낸 것이다. 이와 같은 XML 문서에서 테이블을 추출하는 방법은 예제 3.1에 나타난 것과 같이 확장된 SQL로 정의된다. 이것은 XQL을 이용하여 원시 XML 문서를 일차원적인 테이블의 형태로 추출하는 뷰 테이블을 정의한 것이다.

예제 3.1 *XML-뷰의 정의*

```
CREATE XMLVIEW person
(id CHAR(20), email(CHAR(30)) AS
('select p.personnel.person.id,
  p.*.email
 from "FILE:/home/user1/person.xml" p;
 where p.*.family = "Smith" ');
```

XML-뷰는 사용자가 XML 데이타를 위한 새로운 접근 방법을 익히지 않고도 기존의 데이타를 다루는 방식을 그대로 사용하여 XML 데이타를 다룰 수 있는 장점이 있다. 관계형 데이타베이스의 테이블 기반의 스키마는 표현에 여러 가지 제한이 있어 XML 데이타를 표현하는 것이 쉬운 작업이 아니다. XML-뷰는 XML 데이타 전체를 관계형 데이타베이스의 스키마에 맞도록 변환하는 부담을 덜기 위하여 XML의 일부분만을 관계형 테이블로 구성하여 사용자가 다룰 수 있도록 하는 방법을 사용하였다.

```
<?xml encoding="US-ASCII"?>
<!ELEMENT personnel (person)+>
<!ELEMENT person (name,email*, url*,link?)>
<!ATTLIST person id ID #REQUIRED>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT name (#PCDATA|family|given)*>
<!ELEMENT email (#PCDATA)>
<!ELEMENT url EMPTY>
<!ATTLIST url href CDATA #REQUIRED>
<!ELEMENT link EMPTY>
<ATTLIST link
      manager IDREF #IMPLIED
      subordinates IDREFS #IMPLIED>

<pesonnel>
<person id="John.Denver">
  <name>
      <family>Denver</family>
      <given>John</given>
  </name>
  <email>chief@foo.com</email>
  <link subordinates
      ="Tom.smith Mary.Lee"/>
</person>
<person id="Tom.Smith">
  <name>
      <family>Smith</family>
      <given>Tom</given>
   </name>
   <email>tom@fool.com</email>
   <link manager="John.Denver"/>
</person>
...
<person id="Mary.Lee">
  <name>
      <family>Lee</family>
      <given>Mary<given>
  </name>
  <email>mary@foo.com</email>
  <link manager="John.Denver"/>
</person>
```

그림 3 예제 XML 파일

XML-뷰는 SQL에서 제공하는 뷰의 개념을 확장시켜 만든 것으로 정의와 사용 방법이 뷰를 다룰 때와 비슷

한 방식이 되도록 하였다. SQL에서 뷰는 SELECT 질의의 결과를 보통의 테이블처럼 다룰 수 있게 해준다. 뷰는 실제 저장된 데이타를 가지고 있는 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 질의문이 필요하다는 것이다. 예제 3.1에서 AS 이후에 들어가는 질의가 XML 질의문이다. XML 질의 언어는 Lorel[11]의 기본 개념과 문법을 참고로 하였다. 예제 3.1에서 FILE을 HTTP로 변환하면 파일이 아니라 HTTP 프로토콜을 이용하여 네트워크에서 XML을 추출한다.

예제 3.1을 수행하면 데이타베이스의 카탈로그에 XML-뷰가 등록되며, SQL에서 person 테이블을 접근하면 파일에서 XML을 추출하여 테이블의 형태로 가공한다. 예제 3.1의 person 테이블은 관계형 데이타베이스 내의 일반적인 테이블이 사용될 수 있는 곳에 모두 사용할 수 있다. 이를 사용한 예는 예제 3.2에 잘 나타나 있다. 질의어를 수행하면 그림 2에서 보는 바와 같이 person 테이블은 XML 파일에서 뷰로 만들어진 것이며 employee 테이블은 데이타베이스 내의 테이블이라는 것을 알 수 있다. 두 테이블의 죠인 연산을 통하여 결과를 얻는 것과 같이 XML-뷰는 일반적인 SQL 문장이 들어가는 곳에 모두 사용될 수 있다.

예제 3.2 *XML-뷰를 사용한 질의의 예*

```
select p.id, p.email, e.phone_number
from person p, employee e
where p.id = e.id;
```

XML-뷰의 가장 큰 특징은 가져온 XML 데이타를 SQL의 테이블 함수를 이용하여 적절한 형태의 테이블로 변환하는 것이다. 이렇게 함으로써 XML 문서를 일반 테이블과 구별없이 접근하게 되며 데이타베이스 내부 데이타와 외부 데이타인 XML 데이타간의 연산을 SQL을 이용하여 접근할 수 있다. SQL 테이블 함수는 외부 프로세스로 동작하게 되며 SQL 엔진과는 소켓으로 통신한다. 이것은 그림 2에서 나타난 바와 같이 여러 데이타 소스로부터 XML 데이타를 가져와서 이를 테이블 형태로 가공한 다음 SQL 질의어 처리기에서 일반 테이블과 동일한 형태로 연산을 하게 되는 것이다.

이렇게 하면 SQL 질의어 처리기를 수정하지 않고도 XML 데이타에 대한 연산이 가능하다는 장점이 있다. 또한 기존 관계형 데이타베이스에 구축된 많은 데이타들을 XML과 연관지어 연산할 수 있다는 것이 큰 장점이다. 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. 그러므로 일반 사용자가 새로운 언어를 배우지 않고도 XML 데이타에 대한 질의를 할 수 있는 장점이 있다.

하지만 외부 프로세스를 이용하여 테이블 함수를 구현함으로서 통신비용이 따르며 외부 프로세스에서 XML 문서를 파싱하여야 하기 때문에 수행 시간이 많이 걸리는 단점이 있다. 이것은 테이블 함수의 결과로 생긴 중간 테이블을 저장한 다음에는 다시 임시 테이블을 생성하지 않는 방법을 이용하여 테이블 생성 시간을 줄일 수 있다. 또한 뷰가 가지는 단점인 갱신 연산을 할 수 없는 단점이 있다. 이것은 XML 문서 전체에 대한 뷰를 제공하는 것이 아니라 일부분에 대한 뷰를 제공하기 때문에 일어나는 현상이기도 하다.

3.2 PDM/ODB

3.1절에서 XML-뷰를 이용하여 XML 데이타를 처리하는 것을 살펴보았다. XML 문서를 저장하는 방법에는 PDM/RDB에서 사용한 방법과 다르게 XML 데이타를 하나의 객체로 저장하지 않고 문서의 구조와 의미를 보존하여 저장하는 방법이 있다[3, 4]. 이때 테이블의 개수가 많아지면 하나의 XML 질의어를 처리하는데 많은 죠인 연산이 필요하게 되어 성능이 저하된다[12]. 죠인의 갯수가 많아지는 경우는 질의어 최적화 시에 해결될 수 있는 경우가 많지만 [4]의 인라인(inline) 방법과 같이 엘리먼트를 하나의 테이블에 넣었을 경우 트리 탐색이 들어간 XML 질의어를 SQL로 변환하면 십여개 이상의 죠인 연산이 나타나는 것을 알 수 있다. 이를 수행하면 상당한 시간이 걸린다. 또한 [13]에서 XML을 처리하는데 있어 객체에 대한 갱신 연산이 가능한 클라이언트 쪽의 캐쉬가 필요하게 된다고 언급하고 있다.

관계형 데이타베이스의 데이타 모델이 릴레이션이기 때문에 XML 데이타 모델을 잘 표현하기에 어려운 점이 많다. 객체지향형 데이타베이스는 관계형 데이타 모델보다 더욱 풍부한 모델을 제공하기 때문에 트리 형태의 XML 데이타를 잘 표현할 수 있다. 객체지향형 데이타베이스는 클라이언트 쪽의 캐쉬를 유지하고 있고 XML의 링크가 내포 죠인(implicit join)으로 해결될 수 있기 때문에 이와 같은 관점에서 보면 XML을 저장하는 적합한 데이타베이스 시스템이라 할 수 있다.

그러므로 본 논문에서는 객체지향형 데이타베이스를

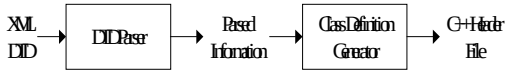

그림 4 DTD-to-Schema 시스템 구성도

이용하여 PDM/ODB를 만들었다. PDM/RDB에서 XML 문서는 하나의 객체로 저장되고 처리되는데 비하여 PDM/ODB에서는 그림 4와 같이 XML 문서의 DTD 정보를 이용하여 XML 문서를 의미있는 클래스의 집합으로 구성한 다음 각각의 엘리먼트들을 적절한 객체 타입에 저장하는 방법을 이용하였다. PDM/RDB에서는 XML 문서의 의미적인 정보를 사용자가 XML-뷰를 정의하면서 작성하게 되지만 PDM/ODB에서 의미적인 정보는 DTD 분석을 통하여 기계적으로 만들어진다. 즉 DTD 분석을 통하여 클래스 정보와 로딩 프로그램이 자동적으로 만들어진다. 그림 5는 그림 3의 DTD를 PDM/ODB를 이용하여 클래스를 생성하여 만든 ODMG C++ 헤더 파일이다.

이 과정에서 고려해야 할 점은 크게 순서(ordering)와 선택(selection) 문제이다. 이것은 반구조적 성격을 갖는 XML을 구조적 성격을 갖는 객체지향 데이타 모델로 변환했을 때 잃어버리는 정보를 어떻게 유지할 것인가 하는 문제이다. XML의 DTD는 엘리먼트(element)라는 논리적 요소들로 구성되어 있다. 엘리먼트의 이름은 XML 문서에서 태그로 사용된다. 그리고 엘리먼트의 구조는 다른 엘리먼트와 #PCDATA, EMPTY 등과 같은 기본 형들, 그리고 연결자(connector)들로 구성된다. 이때 XML이 반구조적 성격을 갖는 이유는 이러한 연결자들 때문이다. 연결자에는 순서를 나타내는 연결자(,), 선택을 나타내는 연결자(|), 옵션을 나타내는 연결자(?), 그리고 반복을 나타내는 연결자(+, *) 등이 있다. 이러한 연결자들 중 반복을 나타내는 연결자는 객체지향형 데이타베이스에서 List를 지원하기 때문에 스키마 사상시 고려할 필요가 없지만, 순서를 나타내는 연결자를 통해 XML 데이타 객체들에게 순서를 부여하는 점은 고려되어야 한다.

그림 5와 같이 클래스로의 변환시 DTD의 선택 연결자에 나타나는 엘리먼트는 일단 클래스의 애트리뷰트에 나타나게 하였다. 이렇게 되면 실제 데이타가 없이 NULL 값을 가지는 애트리뷰트가 많아지게 되는 단점이 있지만 정보 손실은 막을 수 있다. 순서 정보는 XMLObject 클래스의 애트리뷰트인 Ref<List<Ref<RefAny> > > _elm_order를 이용하여 해결하였다. 이 애트리뷰트는 XML 문서에 나타나는 객체 순대로

```
class Email : public XMLObject {
public:
  Ref<d_String> text;
  Email(Ref<d_String>&);
};

class Name : public XMLObject {
public:
  Ref<List<Ref<d_String> > > text;
  Ref<List<Ref<Family> > > family;
  Ref<List<Ref<Given> > > given;
  Name();
};
class Given : public XMLObject {
public:
  Ref<d_String> text;
  Given(Ref<d_String>&);
};
class Family : public XMLObject {
public:
  Ref<d_String> text;
  Family(Ref<d_String>&);
};
class Url : public XMLObject {
public: Ref<d_String> href;
  Url(Ref<d_String>&);
};
class Link : public XMLObject {
public:
  Ref<d_String> manager;
  Ref<List<Ref<d_String> > >
  subordinates;
  Link(Ref<d_String>&);
};
class Personnel : public XMLObject {
public:
  Ref<List<Ref<Person> > > person;
};
class Person : public XMLObject {
public:
  Ref<List<Ref<Email> > > email;
  Ref<List<Ref<Url> > > url;
  Ref<Name> name;
  Ref<Link> link;
  Ref<d_String> id;
  Person(Ref<Name>&, Ref<Link>&,
Ref<d_String>&);
```

그림 5 DTD로부터 생성된 클래스

OID를 리스트로 가지고 있는 애트리뷰트이다. 이 애트리뷰트를 이용하여 클래스의 애트리뷰트를 XML 데이타의 순서와 동일하게 만든다. XML DTD를 이용하여 생성되는 모든 클래스들은 이 XMLObject 클래스로부터 상속받도록 하여 자료 손실을 막는다.

이와 같이 생성된 C++ 헤더 파일을 이용하여 객체지향형 데이타베이스에 스키마를 생성하고 XML을 각 클래스의 인스턴스로 저장하게 된다. PDM/RDB에서 사용한 방법과는 다르게 엘리먼트의 의미를 유지하면서 객

체를 저장하기 때문에 응용 프로그램의 작성 없이도 즉시 사용자가 질의어를 통하여 XML 데이타로부터 원하는 결과를 얻을 수 있는 장점이 있다. 저장된 객체는 예제 3.3에서와 같은 ODMG OQL[14]이나 C++ 바인딩을 이용하여 XML 객체를 다룰 수 있게 된다. 또한 SOP Java 바인딩 및 CLOS 바인딩[9]을 통해서 각각의 객체를 접근할 수 있게 되어 객체지향형 데이타베이스 개발자들은 XML 데이타를 일반 객체와 동일한 관점에서 다룰 수 있는 장점이 있다.

예제 3.3 *OQL을 이용한 질의*

```
select p.email.text
from p in Person;
```

XML 문서는 DTD 없이도 존재할 수 있는데 이럴 경우 데이타로부터 DTD 정보를 추출하여 사용하면 된다. [15]에서는 DTD 정보가 없는 XML 문서에서 효율적으로 DTD를 추출하는 방법에 대하여 설명하고 있다. 이런 방법을 이용하면 DTD가 없는 XML 문서에서도 쉽게 스키마 정보를 추출할 수 있다.

3.3 PDM/Wrapper

레거시 시스템을 이용하는 일반적인 방법은 랩퍼를 이용하는 것이다[16,17]. 이것은 기존의 시스템 위에 공통된 인터페이스를 지원하는 랩퍼를 미들웨어로 두어 응용 프로그램과 레거시 시스템을 접근하는 것이다. XWEET 시스템에서도 그림 1에서 보는 바와 같이 중계자가 각각의 하부 시스템을 접근할 때 랩퍼를 통하여 원하는 데이타를 얻는다. XML 질의를 처리하는 질의어 처리기가 중계자에서 오는 데이타를 접근하기 위해서는 중계자에서 일정한 인터페이스를 제공해야 한다. 이때 중계자는 각각의 데이타 소스로부터 데이타를 추출하는 랩퍼를 통하여 실제 데이타를 접근한다. 이때 중계자가 랩퍼를 이용하는 방법은 중계자에서 사용되는 질의어를 각각의 랩퍼가 이해할 수 있는 언어로 바꾸어 수행하는 것이다.

기존의 관계형 데이타베이스를 이용하여 XML 데이타를 처리하는 시스템에서 XML 질의어 처리기를 구현하는 방법은 크게 두가지로 나누어 볼 수 있다. 첫째 관계형 데이타베이스의 SQL 질의어 처리기에 XML 질의를 처리할 수 있는 시스템을 구현하는 것이다. 이것은 근본적으로 데이타베이스에서 XML을 다루는 문제를 해결하는 방법이다. 하지만 데이타베이스 엔진을 수정해야 하는 문제가 있다. 이것은 [2]에서 언급한 것과 안정성과 같은 문제가 따르게 된다. 그러므로 다른 해결책으로는 기존의 SQL 질의어 처리기 위에 XML질의를 처리할 수 있는 랩퍼를 두는 것이다. 즉 XML 데이타에 대한 질의를 랩퍼에서 적절히 변환하여 관계형 데이타베이스를 접근하게 하는 것이다. 이때 랩퍼를 구현하는 관점을 여러가지로 볼 수 있다. XML 문서를 저장하는 방법은 [3, 4, 5] 등에서 많이 언급하고 있지만 XML 질의어 처리기가 어떻게 중계자에게 접근하여 데이타를 가져오는 가에 대한 연구는 많지 않다. 기존의 방법은 XML 질의를 SQL로 변환하여 관계형 데이타베이스에서 수행하여 데이타를 가져오는 것이다. 이때 XML 질의는 죠인이 많은 복잡한 SQL 질의가 된다. 또한 죠인이 많은 관계로 임시 테이블의 크기가 커져서 수행시간이 많이 걸리는 단점이 있다.

XML의 저장 장치로 객체지향형 데이타베이스가 적합하다고 주장하는 것은 클라이언트의 캐쉬와 빠른 죠인 연산 및 모델링의 적합성 때문이다[12]. 또한 객체지향형 데이타베이스에서 객체 캐쉬를 이용하기 때문에 XML 데이타와 같은 것을 빠르게 처리할 수 있다고 주장하고 있다. 그러나 객체지향형 데이타베이스에서 객체 서버가 페이지 서버보다 나은 성능을 보이지 않는다[18]. 그러므로 객체지향형 데이타베이스의 객체 캐쉬나 관계형 데이타베이스의 버퍼는 그 기능상 큰 차이가 있다고 보기는 어렵다.

객체지향형 데이타베이스에서 죠인 연산이 줄어드는 이유는 OID를 통하여 객체를 직접 가져오기 때문이다. 이때 OID는 두가지 방법으로 구현할 수 있다[19]. 첫째는 물리적 OID로서 디스크의 주소를 OID로 사용하는 것이다. 두번째 방법은 논리적 OID를 이용하는 것으로 이것은 논리적인 ID에 객체를 매핑하는 것이다. 논리적 OID는 OID로 객체의 위치를 빨리 알기위하여 인덱스를 이용한다. 그러므로 B 트리를 이용한 기본키와 논리적 OID의 성능차이는 그리 크지 않게 된다. 물론 SQL 문장을 컴파일하는 비용이 있으나 대부분의 데이타베이스에서 SQL 문장을 미리 컴파일하여 사용할 수 있기 때문에 그 비용 차이 또한 크지 않다.

본 논문에서는 관계형 데이타베이스를 접근하는 SQL을 단순히 접근 경로(access method)로 이용하였다. 이것은 XML 질의를 하나의 SQL로 변환하지 않고, OID로 객체를 가져오는 단순한 SQL 문장을 여러번 수행하도록 한 것이다. 이렇게 함으로서 XML 질의어 처리기는 질의어 플랜을 수행할 때 랩퍼에서 제공하는 일정한 인터페이스를 이용하여 관계형 데이타베이스에 저장된 객체를 가져오는 것이다. XML 질의는 대부분 OID를 이용한 포인터 질의들로 바뀌지며 OID는 이미 인덱스가 있어 빠르게 접근할 수 있게 하였다. 또한 객체를 가져오는 질의어들은 미리 컴파일하여 실행시 질의어 파

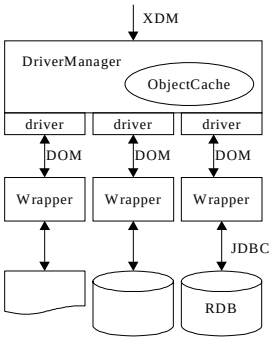

그림 6 PDM/Wrapper의 아키텍처

싱 시간이 없도록 하였다. 또한 죠인 연산도 미리 컴파일된 SQL 문장을 통하여 인덱스된 기본키를 통하여 가져온다.

XML 질의어 처리기는 현재 매우 연구가 활발한 분야이다. 새로운 인덱스 기법[1, 20, 21]이 많이 등장하고 있으며 질의어 최적화 기법[22, 23, 24]도 많이 연구되고 있다. 관계형 데이타베이스에서 XML 질의어 처리를 위한 기존의 방법은 단순히 XML 질의어를 SQL로 변환만 하므로 새로운 질의어 최적화 기법을 XML 질의어 처리기에 반영하려면 관계형 데이타베이스 엔진을 수정하지 않고서 이런 기술 발전 사항들을 즉시 반영하기 어렵다. 그러므로 그림 6에서 보는 바와 같이 랩퍼와 데이타베이스간에는 단순한 SQL을 이용하여 객체를 가져오는 기능만 수행하고 상위 인터페이스로는 DOM을 제공하여 XML 질의어 처리기가 이용하도록 하였다. 드라이버(driver)들은 DOM 인터페이스를 제공하는 여러 랩퍼를 붙이는 기능을 수행한다. Object Cache는 한 트랜잭션 동안 사용되는 객체를 캐싱하여 다시 접근할 때 빠르게 접근할 수 있게 해준다. 이 시스템은 Java를 이용하였고 데이타베이스는 JDBC를 이용하여 접근하였다.

XML 데이타를 저장하는 테이블은 Document, Element, Attribute 세 개로 구성되어 있다. 모든 엘리먼트는 Element 테이블에 저장되고 각 엘리먼트의 애트리뷰트는 Attribute에 저장된다. 이것은 3.2절에서 언급한 방법과는 상이하다. 3.2절에서는 XML 데이타를 DTD를 이용하여 분석한 후 클래스를 생성하므로 XML 데이타의 태그 정보가 살아있는데 비해 PDM/Wrapper에서는 XML 문서의 의미적인 정보를 저장하지 않고 문서를 DOM 형태로 저장한다. 이와 같이 저장하였을 때 PDM/ODB와 같은 질의어를 만들기가 쉽지 않다. 하지만 XML 문서를 데이타베이스에 저장하기 위한 적절한 스키마는 기계적으로 생성되기 때문에 일반적으로 사용자가 직접 질의어를 생성하기보다는 랩퍼나 다른 응용 프로그램에서 기계적으로 질의를 생성하여 사용하는 경향이 있다. 그러므로 사람이 의미있는 질의를 생성하기가 어렵다는 사실이 큰 단점이 되지 않는다. PDM/Wrapper는 사용자가 질의를 생성하는 것 보다는 상위 레벨에서 사용하는 중계자 등에서 사용할 수 있는 인터페이스를 제공하는 것이다.

3.4 요약

본 연구에서 XML을 저장하기 위하여 사용한 저장방법 및 저장 시스템, 인터페이스는 표 1에 잘 나타나 있다. 저장 시스템으로 관계형 데이타베이스, 객체지향형 데이타베이스를 사용하였고, 저장 방법으로 파일, LOB, 객체 혹은 테이블을 이용하였다. 사용자가 접근하는 인터페이스로는 질의어를 확장하는 방법(SQL 테이블 함수)과 기존의 데이타베이스 접근 방법(OQL, C++ 바인딩)을 이용할 수 있게 하였다. 또한 랩퍼를 만들때 기존의 방법에서는 질의어 변환을 통하여 접근하였지만, 본 연구에서는 랩퍼가 단순한 접근 경로만 지원하고 실제 질의어 플랜의 수행은 중계자에서 하도록 하였다.

PDM/RDB에서는 뷰의 형태로 XML 데이타를 지원하므로 기존의 SQL 사용자는 새로운 언어를 배우는 부담없이 XML 데이타를 다룰 수 있는 장점이 있다. 하지만 저장 장치로 파일이나 LOB을 이용하였기 때문에 갱신 연산 시에 부담이 많은 단점이 있다. 이러한 문제는 PDM/Wrapper에서와 같은 방법을 이용하여 저장하면 엘리먼트가 저장 단위이기 때문에 갱신 작업의 비용을 줄일 수 있다.

PDM/ODB에서는 DTD 정보를 이용하여 클래스를 생성하고 사용자는 OQL이나 ODMG C++/Java 바인딩을 이용하여 응용 프로그램을 작성하게 된다. 이것은 기

표 1 각 시스템의 비교

시스템	PDM/RDB	PDM/ODB	PDM/Wrapper
데이타베이스	RDB	OODB	RDB
저장 형태	파일/LOB	객체	테이블
접근 방법	SQL (XML-뷰)	OQL/C++ 바인딩	JDBC/DOM
갱신 연산	고비용	저비용	저비용
DTD	불필요	필요	불필요
부분 접근 연산	느림	빠름	빠름
전체 접근 연산	조금 빠름	빠름	느림

존의 사용자 인터페이스를 이용하여 XML 데이타를 처리하므로 사용자가 새로운 언어를 배울 필요가 없는 장점이 있다. 이때 순서 정보와 같이 XML 데이타 모델과 객체지향 데이타 모델이 일치하지 않는 경우가 있다. 이것은 XML을 처리하는 모든 클래스의 가장 상위 클래스로 XMLObject 클래스로부터 계승받도록 하여 해결하였다. 즉 XMLObject의 메쏘드를 이용하여 순서 정보와 같은 XML에만 해당하는 정보를 얻을 수 있도록 하였다.

PDM/Wrapper에서는 기존의 랩퍼를 만드는 방법에 대한 개선을 제시하였다. 기존의 랩퍼는 디자인할 때 상위 언어를 랩퍼가 인식할 수 있는 하위 언어로 변환하여 질의를 수행하는 방법을 이용하였다. 이때 XML 질의는 변환시에 많은 죠인 연산을 가진 질의어로 변환되며 이는 필요없는 임시 테이블을 크게 만드는 경향이 있다. 이로 인하여 질의어 수행 시간이 길어지는 단점이 있다. PDM/Wrapper에서는 이를 극복하기 위하여 SQL 질의어를 단순히 접근 경로로만 사용하였다. 이는 객체지향형 데이타베이스에서 OID를 통한 객체 접근과 유사한 형태이다. 이 방법은 실제 질의를 처리하는 중계자의 성능 개선을 직접 반영할 수 있는 장점이 있다. 기존의 방법은 단순히 질의를 변환만 하므로 XML 데이타 모델의 특징을 반영하여 질의를 최적화하는 기법을 데이타베이스에서 지원하지 않으면 처리할 수 없는 단점이 있다. 현재 많은 XML 질의어 최적화 기법이 제안되고 있는데 이를 지원하기 위해서는 중계자 단위에서 이를 지원하는 것이 최선의 방법이다. 그러므로 중계자에서 생성한 플랜을 처리할 수 있는 랩퍼는 하위 데이타베이스를 단순히 접근 경로로 이용하는 것이 필요하게 된다.

4. 관련 연구

XML이나 중계자와 관련된 연구들은 현재 활발히 진행 중이다. XML과 관련된 연구로는 먼저, XML을 위한 전용 시스템인 Lore[1]를 들 수 있다. Lore 는 XML 의 특성을 반영하는 여러 형식의 접근 방법과 저장 구조와 데이타 모델을 지원한다. 하지만, Lore처럼 시스템을 새로 개발하는 것은 많은 비용을 요구하므로, 기존의 데이타베이스 시스템을 이용하여 XML 문서를 관리하고자 하는 연구가 진행되었다. 이 중, STORED [3]와 [25]은 관계형 데이타베이스 시스템에 XML 문서를 저장하고, 검색할 수 있도록 지원하기 위한 방법을 제공한 연구로, 비슷한 점이 많으나, 다음과 같은 점에서 차이가 있다. 먼저, 전자는 임의의 반 구조적인 데이타 모델을 지원하여 스키마에 대응되지 못하는 부분은 독립적인 장소에 저장하도록 하며, 후자는 조작하고자 하는 XML 문서가 관계형 스키마에 완전히 대응된다고 가정한다. 또한, 전자는 XML 문서의 스키마 매핑을 위해 데이타 마이닝 기법을 사용하였으며, 후자는 인라이닝(inlining)을 통해 스키마를 생성하는 방법을 사용했다.

이외에도 [26]에서 파일이나 BLOB에 SGML 문서를 저장하여 관리하는 기법에 관한 연구가 있었다. 이것은 간단하고 저장 공간도 적으며 적절하게 클러스터링되는 효과가 있다. 하지만 갱신 연산이 어렵고 질의어 처리기가 이를 처리하기 위하여 더욱 많은 연산을 추가해야 하는 단점이 있다.

MIX[27]는 중계자 시스템에서 사용할 데이타 모델로 XML을 사용한다. 그리고, 기존의 데이타베이스 시스템을 XML 형태로 바라볼 수 있도록 하기 위한 뷰를 정의할 수 있다. 다른 연구들로 [28]은 위의 연구가 XML에 대한 질의를 수행할 때 성능이 떨어지는 단점을 보완하기 위해 스트링 매치를 통해 질의 수행 속도를 높이고자 하였다. XViews[29]는 관계형 테이블을 XML 문서 형태로 바라볼 수 있도록 하기 위한 방법을 제안하였다.

랩퍼 구조를 이용한 것으로 Garlic[30], TSIMMIS [31], YAT[32], WebOQL[33], W3QL[34] 등이 있다. TSIMMIS를 제외한 시스템들은 웹에서 생성된 데이타인 HTML에 대하여 질의를 수행하거나 데이타를 변환하여 정보를 얻는 시스템에 관한 것들이다. 이런 방법들은 대부분 질의 변환 방법을 이용하여 원하는 결과를 얻는다.

5. 결론 및 향후 연구 방향

본 연구에서는 XML을 데이타베이스에서 지원하기 위한 여러 방법들에 대하여 살펴보았다. 관계형 데이타베이스에서는 LOB에 저장하고 XML-뷰를 이용하여 XML 문서를 테이블의 관점으로 질의할 수 있게 하였다. 객체지향형 데이타베이스에서는 DTD 정보를 이용하여 클래스를 생성하고 OQL이나 ODMG C++ 바인딩과 같은 기존의 방법을 이용하여 데이타베이스를 이용하도록 하였다. 또한 XML 데이타를 관계형 데이타베이스로 접근하는 랩퍼를 디자인하는 방법으로 랩퍼가 관계형 데이타베이스를 단순히 접근 경로로 이용하는 인터페이스를 제공하는 방법을 제안하였다. 이는 모두 XWEET 시스템의 PDM을 통하여 구현되었다. 각각의 방법은 장단점을 가지고 있으며 지원하고자 하는 응용

에 따라 적절한 방법을 선택하여 사용할 수 있다. 이러한 방법들은 모두 지역 데이타베이스에 XML을 저장하고 질의하는 방법들에 관한 것이다. 현재 연구하고 있는 것은 인터넷을 통하여 얻어진 HTML이나 그외 다른 형태의 데이타들을 XML 모델로 바라보고 XWEET 시스템 내에서 처리하는 방법에 관한 연구를 수행중이다. 이러한 연구를 바탕으로 인터넷 정보 시스템에서 여러 정보를 XML 형태로 관리, 전달하는 시스템을 구현하고자 한다.

참 고 문 헌

[1] Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web*. Morgan Kaufmann, 2000.

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

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

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

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

[6] W3C. Document Object Model (DOM). http://www.w3.org/DOM/, 2 2000.

[7] 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,

[8] 안정호, 김형주. SRP 에서 SOP까지. *한국정보과학회 Review 지*, 4 1994.

[9] 정태선, 조은선, 김형주. Sopclos: 객체지향 데이타베이스 관리 시스템을 위한 CLOS 인터페이스. *정보과학회 논문지(B)*, 24(9), 1997.

[10] Kang-Woo Lee and Hyoung-Joo Kim. Support of a Web Transaction Processing System for Preserving Consistency. *2nd Web Technology Workshop (AREAU 99)*, October 1999.

[11] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel Query Language for Semistructured Data. *International Journal on Digital Library*, 1(1), 4 1997.

[12] John P. Desmond. XML Combined with Object Databases Has Potential to Drive Ecommerce Applications. *Component Strategies*, February 1999.

[13] P. Bernstein, Michael Brodie, Stefano Ceri, David DeWitt, Mike Franklin, Hector Garcia-Molina, Jim Gray, Jerry Held, Joe Hellerstein, H. V. Jagadish, Michael Lesk, Dave Maier, Jeff Naughton, Hamid Pirahesh, Mike Stonebraker, and Jeff Ullman. The Asilomar Report on Database Research. *SIGMOD Record*, 27(4), 1998.

[14] R.G.G. Cattell and Douglas K. Barry, editors. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann Publisher, Inc., 1997.

[15] Minos Garofalakis, Aristides Gionis, Rajeev Rastogi, S. Seshadri, and Kyuseok Shim. XTRACT: A System for Extracting Document Type Descriptors from XMLDocuments. *SIGMOD*, 2000.

[16] Mary Tork Roth and Peter Schwarz. Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources. *VLDB*, 1997.

[17] Gio Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3), 1992.

[18] David J. DeWitt, Philippe Futtersack, David Maier, and Fernando Velez. A Study of Three Alternative Workstation-Server Architectures for Object Oriented Database Systems. *VLDB*, 1990.

[19] Won Kim. *Introduction to Object-Oriented Databases*. The MIT Press, 1990.

[20] Jason McHugh, Jennifer Widom, Serge Abiteboul, Qingshan Luo, and Anand Rajaraman. Indexing Semistructured Data. *Technical Report*, January 1998.

[21] Tova Milo and Dan Suciu. Index Structures for Path Expressions. *ICDT*, 1999.

[22] Peter Buneman, Susan Davidson, Gerd Hillebrand, and Dan Suciu. A Query Language and Optimization Techniques for Unstructured Data. *SIGMOD*, 1996.

[23] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From Structured Documents to Novel Query Facilities. *SIGMOD*, 1994.

[24] Jason McHugh and Jennifer Widom. Query Optimization for XML. *VLDB*, 1999.

[25] 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.

[26] Serge Abiteboul, Sophie Cluet, and Tova Milo. Querying and Updating the File. *VLDB*, 1993.

[27] UCSD Database Lab component of MIX project. MIX Project. http://www.npaci.edu/DICE/MIX.

[28] Takeyuki Shimura, Masatoshi Yoshikawa, and Shunsuke Uemura. Storage and Retrieval of XML

Documents Using Object-Relational Databases. *DEXA*, 1999.
[29] Chaiyanya Baru. XViews : XML Views of relational schemes. *International Workshop on Internet Data Management*, 1999.
[30] Michael J. Carey, Laura M. Haas, Peter M. Schwarz, Manish Arya, William F. Cody, Ronald Fagin, Myron Flickner, Allen W. Luniewski, Wayne Niblack, Dragutin Petkovic, John Thomas, John H. Williams, and Edward L. Wimmers. Towards Heterogeneous Multimedia Information Systems: The Garlic Approach. *RIDE-DOM*, 1995.
[31] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and Jennifer Widom. Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. *In Proceedings of the AAAI Symposium on Information Gathering*, pages 61-64, 3 1995.
[32] Sophie Cluet, Claude Delobel, Jerome Simeon, and Katarzyna Smaga. Your Mediators Need Data Conversion! *SIGMOD*, 1998.
[33] G. Arocena and A. Mendelzon. WebOQL: Restructuring Documents, Databases and Webs. *ICDE*, 1998.
[34] David Konopnicki and Oded Shmueli. Information Gathering in the World-Wide Web: The W3QL Query Language and the W3QS System. *TODS*, 23(4), 12 1998.

박 상 원
1994년 2월 서울대학교 컴퓨터공학과 학사. 1997년 2월 서울대학교 컴퓨터공학과 석사. 1997년 ~ 현재 서울대학교 컴퓨터공학과 박사과정. 1999년 9월 ~ 2000년 8월 컴퓨터공학과 조교. 관심분야는 데이타베이스, XML, Semistructured data, Web

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

김 형 주
정보과학회논문지 : 컴퓨팅의 실제
제 6 권 제 1 호 참조