XWEET: 웹 환경을 위한 통합 데이타베이스 시스템*

XWEET Team

요 약

XML은 Web 에서 운영되는 모든 데이타 표현에 대한 표준이다. 여러 웹 소스들로부터 이질적인 데이타를 XML 데이타로 통합할 수 있다. 많은 연구자들이 XML처리에 대한 연구를 해 왔다.

이러한 연구의 집합체로서, 우리는 XML 데이타의 효율적인 저장, 추출, 질의, 웹 환경에서의 응용을 위한 기반 시스템인 XWEET[1]를 제안, 구현하였다. XWEET는 XML 데이타의 표현을 위해 XDM이라는 데이타 모델을 사용하였으며, XML 데이타의 저장을 위해 PDM을, 이질적인 정보 소스로부터의 데이타 통합을 위해 Wrapper와 XWS를, 서로 다른 XML 문서 형식의 통합된 표현을 위해 XSI를 제공한다. 이렇게 통합, 저장된 데이타는 XML/QL 형식의 질의를 통해 접근되며, XQP는 제공되는 질의를 처리한다. 또한, 사용자들은 웹 환경에서 수행되는 응용을 작성할 수 있는 기반을 제공받는다. WPG와 HTML/XML Template는 웹 응용의 수행 결과를 정의된 HTML이나 XML로 생성하도록 해 주며, WebTP는 웹기반 워크플로우를 위한 기반 기능을 제공한다.

주제어 : XWEET, Web, HTML, WWW, XML

Abstract

XML is a standard representation format for data operated in web environment. We can integrate the heterogeneous data from several web sources into XML data. Many researchers have been working on XML processing.

As a conglomerate of those works, we proposed and implemented the XWEET system. In this system, we store, extract and query XML data. XWEET has several interesting modules: XDM(XWEET data model) for representing XML data, PDM(persistent data manager), wrapper, XSI(XWEET semantic integrator), and XQP(XWEET query processor). Also, it supports an environment for generating web applications; WPG(a html/xml result generation module) and WebTP(a web-based workflow system).

Keyword: XWEET, Web, HTML, WWW, XML

1 개요

웹의 등장으로 인하여 기존의 문서 표현방법이나 저장 방법이 변화하고 있다. 기존의 문서들은 인쇄화된 형태로 그 검색이나 이용에 일정한 한계를 가지고 있었으나 웹은 정보의 시간, 공간적인 제약을 없애고 언제 어디서나 원하는 정보를 쉽고 빠르게 찾을 수 있도록 해준다. 웹은 사용자의 관점에서 원하는 데이타를 적당한 질의어로 검색하여 얻는 하나의 데이타베이스로 볼 수 있다[31]. 웹을 데이타베이스로 보고 질의하는 것은 WebSQL[30], WebOQL[10] 등에서 그 예를 찾아 볼 수 있다. 이러한 시스템은 기본적으로 문서의 형태로 HTML을 가정하고 있다.

*이 논문은 2000년도 두뇌한국21사업에 의하여 지원되었음.

[1] **X**ML DBMS for **WE**b **E**nvironmen**T**, [swi:t]라 읽는다.

HTML은 웹에서 문서를 표현하는 태그 기반의 언어이다. 하지만 HTML은 문서를 브라우징하기 위한 언어이기 때문에 정보전달의 수단으로 사용하기가 어렵다. 앞으로 많은 응용들이 웹에서 구현, 동작하게 되면 각 서버들간의 정보전달이 필요하게 된다. 이를 지원하기 위한 언어로 XML이 등장하게 되었다. 앞으로 다양한 응용들이 웹에서 동작할 것이며 웹서버간의 협동을 통한 처리가 증가할 것으로 예상된다. 이때 서버간의 데이타 전송의 표준으로 XML이 널리 사용될 것이다.

정보들이 여러 형태로 여러 저장매체에 저장되어 있으므로 이러한 다양한 정보 소스들에서 유용한 정보를 얻거나, 협동을 통한 질의 처리를 하기 위한 시스템으로 이질(heterogeneous) 데이타베이스 시스템이 등장하였다. 이것은 데이타베이스를 통합적으로 보기 위한 시도이며 여러 데이타 모델의 데이타베이스 시스템을 하나의 모델로 통합하고 그 질의를 처리하는 시스템을 말한다. 최근 들어 기존의 레거시(legacy) 시스템으로부터 서로 다른 데이타 형식의 데이타를 통합, 가공하여 사용자에게 새로운 정보를 제공하고자 하는 연구가 활발히 진행되고 있다[41]. 이러한 형태의 시스템을 중개자(mediator)라 한다. 중개자는 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 로컬하게 저장하기 위한 방법도 제공해야 한다. 이에 중개자들은 나름대로의 저장 시스템과 데이타 표현언어를 정의하고 있다. 하지만, 이러한 중개자들은 또 다른 래거시 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.

본 논문에서는 정보전달의 수단으로 사용되는 XML을 잘 처리하기 위한 시스템으로 XWEET를 제안하였다. XML 데이타는 외부 데이타 소스로부터 전달되어질 수도 있고, 내부 XML 데이타 혹은 내부에서 가공되어진 XML 데이타등 여러 형태일 수 있다. XWEET는 웹상에서 여러 형태의 XML 데이타를 XML 질의어 처리기를 통하여 처리하는 시스템이다.

XWEET 시스템의 특징적인 모듈로 PDM, 래퍼, 중개자(mediator), XQP(XWEET Query Processor), WPG등이 있다. PDM은 내부적으로 사용되는 XML 데이타를 관리하는 모듈이다. 내부적으로 발생하는 데이타는 크게 두가지 방법으로 저장할 수 있는데 첫째로는 Lore[28]와 같이 새로운 데이타 모델인 XML을 지원하는 새로운 데이타베이스를 만들어 저장하는 방법이고, 둘째는 기존의 데이타베이스 시스템을 이용하여 저장하는 것이다. 새로운 데이타베이스를 만들어 저장하는 방법은 시스템의 검증이 어렵기 때문에 실제 문제에 적용하기에 어려움이 있다[37]. 이러한 문제로 인하여 관계형 데이타베이스에 저장하는 방법을 택하고 있다. [15, 35]등에서 관계형 데이타베이스에 저장하는 방법을 제안하고 있는데 이러한 방법들은 실제 데이타를 저장하기 전에 DTD나 데이타의 분석으

로 통하여 그 XML의 스키마 정보를 구한 다음 이를 관계형 데이타베이스의 스키마로 매핑하여 저장하는 방법을 택하고 있다. 이러한 방법은 데이타를 저장하기 전에 미리 스키마 정보를 얻을 수 있어야 하는데 일반적으로 반구조적인 데이타는 그 스키마 정보를 미리 얻기는 어려운 경우가 많다. 그러므로 임의의 XML 문서를 잘 저장하기 위하여 XWEET/PDM에서는 [17]에서 제안한 방법을 택하고 있다. 이때 XQL을 SQL로 변환하여 질의를 처리해야 하는데 기존의 방법은 하나의 XQL을 한두개의 SQL 문장으로 변환하여 질의를 처리한는데 이때 너무 많은 죠인이 발생한다. 또한 XML 질의의 처리를 전적으로 SQL 처리기에 맡기기 때문에 새로운 인덱스나 질의어 최적화 기법을 적용하기가 어렵다. 그래서 PDM에서는 SQL을 단지 OID를 이용하여 객체를 가져오는 기본적인 접근방법으로 이용하는 방법을 택하고 있다.

XWEET에서는 XWS를 통하여 다양한 HTML 소스로부터 얻은 데이타를 XML로 변환하여 처리하는 방법을 제공하고 있다. 이것은 Perl로 구성되어 있는 모듈로서 객체 인터페이스를 통한 스크립트 언어로 설계되어 있다. XWS는 인터프리터 방식으로 동작하며 여러 XWEET 시스템이 동일한 설정화일을 공유하여 사용 가능하며, 에이전트 방식으로도 동작이 가능하다. 고수준의 처리를 원하는 사용자는 객체 인터페이스를 이용하고, 세밀한 제어를 원하는 사용자는 Perl의 다양한 문자 처리 함수를 이용하여 래퍼를 설계 가능하게 한다.

XSI(XWEET Semantic Integrator)는 중개자(mediator)와 협력하여 다양한 DTD 입력을 동일한 형태의 DTD로 변환하는 작업을 한다. 이것은 동의의 사전을 참조하여 XML 태그를 변환하여 처리하는 형태로 구성되어 있다.

웹 응용에서 일반적인 응용 프로그램의 재 사용성은 매우 낮다. 그 이유는 웹 응용은 대부분 HTML을 기반으로 하고 있기 때문에 응용 프로그램들이 로직과 사용자 인터페이스가 하나의 프로그램에 혼재하기 때문이다. 그러므로 로직이 변하거나 사용자 인터페이스가 변할 때 기존의 응용을 그대로 사용하기에 어려움이 많다. XWEET에서는 이와 같은 문제를 해결하기 위하여 워크, 워크-전역변수를 도입하여 로직과 사용자 인터페이스를 분리하였다. 이를 통하여 응용 프로그램의 재사용성을 높였다. 또한 웹 응용에서 트랜잭션의 처리는 매우 중요하다. XWEET 시스템에서는 기존의 웹 응용에서 웹 트랜잭션 처리시 야기되는 여러 문제를 해결하는 웹 트랜잭션 처리 시스템인 WebTP(Web Transaction Processing System)[27]를 구현하였다. WebTP는 워크 개념을 도입하여 복잡한 웹 응용을 구조적으로 개발할 수 있도록 하고, 여러 대화로 구성된 웹 응용을 하나의 트랜잭션으로 처리될 수 있도록 하였다. 워크를 구성하는 페이지 함수의 실행순서를 명시하도록 하여, WebTP로 하여금 명시된 순서에 어긋나는 페이지 함수의 실행을 막아 시스템이 비정상적인 상태로 전이되는 상황을 막을 수 있다. 또한 WebTP로 하여금 웹 응용의 문맥을 직접 관리할 수 있도록 하

여 웹 응용 단위에서의 저장점 설치와 이를 이용한 부분철회를 지원하게 된다. 또한 WebTP는 CGI 응용 서버 방식을 사용하여 초기의 CGI 실행 화일 방식을 사용하는 시스템에서의 성능문제를 해결함과 동시에 웹 표준인 CGI의 여러 장점을 유지할 수 있도록 하였다.

2 관련 연구

웹과 반구조화(semistructured) 데이타에 대한 연구가 현재 활발히 진행되고 있다[7]. 반구조화 데이타란 데이타베이스의 스키마와 같이 데이타가 항상 일정한 형태를 가지는 것이 아니라 그 구조를 미리 예측할 수 없는 데이타를 말한다. TSIMMIS[18]에서 제안한 OEM 모델이 반구조적 데이타의 원형이라 할 수 있다. OEM은 DAG 형태의 그래프로 표현되는 데이타로서 데이타 내부에 구조적인 정보를 가지고 있는 모델이다. 이와 같은 데이타는 기존의 관계형 모델이나 객체지향 모델과는 많은 차이가 있다. 이런 새로운 데이타 모델을 지원하는 데이타베이스 시스템으로 Lore[28]가 있다. Lore에서는 OEM 모델을 지원하는 저장 장치를 제공하며, Lorel[8]을 이용하여 반구조적 데이타에 대한 질의를 수행한다. Lorel에서는 경로식(path expression) 및 정규식(regular expression)을 지원한다. 또한 data guide[20]를 통하여 반구조적 데이타로부터 스키마 정보를 추출하여 사용자가 질의를 할 수 있는 단서를 제공한다. OEM 모델과 XML 데이타 모델은 유사하여 Lore에서 OEM 모델을 확장하여 XML을 지원하도록 하였다[19]. OEM 모델에서는 순서에 대한 정보가 없으나 XML은 각 엘리먼트들 간의 순서가 정해져 있기 때문에 OEM 모델에서 순서에 대한 정보를 저장할 수 있는 리스트를 추가하여 XML을 지원하도록 하였다. 그외에 반구조적인 데이타에 대한 질의어를 처리하는 것으로 UnQL[12]이 있다. XML에 대한 질의어로는 XML-QL[16], XSL[42], MIX[11] 등이 제안되고 있다.

TSIMMIS는 래퍼와 중개자(mediator)를 두고 있으며 하부 데이타 소스에 맞는 질의어 변환을 통하여 각 데이타 소스에서 원하는 결과를 가져온다. 이것은 래퍼를 이용하는 시스템의 공통적인 디자인 형태로서 레거시 시스템을 기반으로하는 이질 분산 데이타베이스 시스템에서 일반적으로 따르는 아키텍처다.

웹 페이지를 데이타베이스로 보고 질의를 통하여 이를 재구성하여 사용자에게 보여주는 시스템으로 WebSQL, WebOQL이 있다. 이것은 HTML의 태그 및 링크 정보를 이용하여 문서의 의미를 재구성하고 질의를 통하여 의미있는 정보를 추출하는 형태로 구성되어 있다. 또한 확장성에 중점을 둔 W3QL[23]이 있다. 이들과 유사한 것으로 ARANEUS[29]가 있는데 이것은 웹 데이타를 자신의 고유 데이타 모델인 ADM으로 보고 PENELOPE라는 질의어 처리기를 통하여 새로운 데이타를 생성한

다. 이와는 다르게 데이타 변환을 통하여 웹페이지를 생성하는 시스템으로 Yat[36]가 있다. 이것은 질의 언어를 통하여 데이타를 변환하여 웹 페이지를 생성할 수 있도록 한다. 또한 웹 데이타를 데이타 로그 문법에 기초하여 표현하는 WebLog[26]가 있다.

웹상에서 정보를 추출하거나 통합하는 일은 최근 들어 활발히 연구되고 있다. 웹 페이지에서 특정한 구조적 정보(튜플들의 집합, 또는 객체들의 집합)를 추출하여 다른 형태의 데이타모델로 변환하는 작업을 포함하고 있다. 웹 페이지에서 정보를 추출하는 일은 래퍼라는 프로그램이 담당한다.

웹 래퍼 시스템을 구축하는 데 어려운 점은 HTML 페이지들이 프로그램에 의해 정보를 추출하는 목적보다는 사람에게 정보를 보이기 위한 목적으로 설계되었다는 점이다. 그러므로 정보가 자연언어로 기술된 문장 속이나 그래픽으로 표현된 그림내에 숨겨져 있는 경우가 많다. 더우기 정보의 소스가 되는 HTML 페이지가 자주 바뀌는 경우 래퍼도 변경하여야 하기 때문에 래퍼를 관리하는게 힘들어지게 된다. 래퍼의 빠른 생성을 돕기 위해 여러 시스템들이 제안되었다. 래퍼 생성 도구는 데이타가 웹 페이지에 어떻게 포함되어 있고 어떤 식으로 정보를 추출해 내는지 기술하는 문법(Grammar)기반의 방법[14, 21, 22, 33]과 자동으로 귀납적인 학습 추론을 통해서 정보를 추출하는 방법[9, 25]으로 크게 분류할 수 있다. 후자는 시스템에 정보가 저장되어 있는 HTML 페이지에 특정 라벨을 붙여서 입력하거나 대화식으로 정보를 입력하면 시스템 엔진이 특정한 룰의 문법을 생성해 내는 방법이다.

이와 같은 시스템들은 모두 반구조적인 데이타에 대한 질의어를 수행하거나 데이타 모델 및 질의어 정의 등에 대한 것과 래퍼 시스템에 관한 것들이다. 하지만 이런 반구조적인 데이타와 질의어 및 래퍼 시스템을 이용하여 웹 상에서 응용 프로그램을 작성할 수 있도록 도와주는 전체적인 시스템에 대한 것에 대해서는 언급하지 않고 있다. XWEET 시스템은 XML 데이타에 대한 질의어 수행 및 트랜잭션, 웹 페이지 생성 등에 관하여 총체적인 해결책을 제시하는 시스템이다.

3 XWEET 데이타모델

XDM(XWEET Data Model)은 경량화된 DOM[39]으로 XWEET 내부 데이타 모델이다. Data source 사용자는 실제 데이타의 위치나 포맷과는 무관하게 동일한 인터페이스를 통하여 데이타를 접근할 수 있게 된다. XWEET는 각 모듈간의 인터페이스를 XML로 통일하여 사용한다. 하지만 이런 XML 데이타를 각 모듈에서 처리하려면 데이타를 파싱하여야 한다. 또한 XML 데이타의 구조가 간단하지 않기 때문에 각 모듈에서 담당해야 할 부분들이 상당히 많게 되는 문제가 있다. 이런 문제를 극복하기 위하여 XWEET에서는 XML 데이타를 전달하기 위한 최소한의 인터페이스를 제안하고 이를 각 모듈에서 구성함으로서 래퍼나 중개자에서의 데이타 전달을 쉽게할 수 있도록 하였다. 이렇

```
abstract public class Node {
    abstract public String getName();
    abstract public String getValue();
    abstract public int getType();
    abstract public int getRank();

    abstract public Node getParentNode();
    abstract public Node getChildNode(int rank);
    abstract public boolean hasChildNodes();
    abstract public Attribute[] getAttributes();
};
```

그림 1: XDM(XWEET Data Model)

게 함으로써 하위 모듈인 래퍼나 중개자는 XDM의 인터페이스만을 지원하면 쉽게 XWEET의 한 모듈로 장착이 가능한 것이다.

XML 문서는 트리 형태의 그래프로 표현할 수 있다. XDM에서는 그림 1와 같이 이들 데이타를 검색하는 방법으로 부모 노드에서 자식 노드로 트리를 순회할 수 있는 인터페이스를 제공한다. XML 문서를 내부 자료구조로 표현하기위한 표준으로 DOM이 제안되어 있다. DOM은 트리를 순회할 수 있는 여러 인터페이스를 제공하며 XML 문서 자체의 수정 및 각 노드 타입에 대한 정보를 얻는 인터페이스 등도 제공하고 있다. XDM은 이러한 DOM에서 트리를 탐색할 수 있는 최소한의 인터페이스만을 취한 것으로 DOM의 일부분이라 할 수 있다. 그러므로 XDM 모델을 지원하는 응용은 DOM을 지원하는 응용으로 쉽게 전이시킬 수 있다.

XWEET에는 DOM을 표준 인터페이스로 정하여 하위 모듈에서 이를 지원하도록 할 수 있지만 그렇게 하면 하위 모듈을 구현하기 위한 코드의 크기가 너무 커지는 단점이 있다. 또한 하위 모듈에서 다룰 데이타는 XML뿐만 아니라 일반적인 텍스트, 웹 페이지 등도 있다. 이를 XML 형태의 자료구조로 바꾸게 되는데, DOM 인터페이스를 모두 구현하여 래퍼를 구성하면 그 의미가 변하거나 존재하지 않을 가능성이 많다. 그러므로 XWEET에서는 DOM 인터페이스의 부분 집합인 XDM을 제안하여 하위 모듈에서 이를 구현하도록 하고 있다.

4 XWEET 구조

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 중심으로한 많은 응용들이 등장하고 있다. XWEET 시스템은 여러 정보 소스에 존재하는

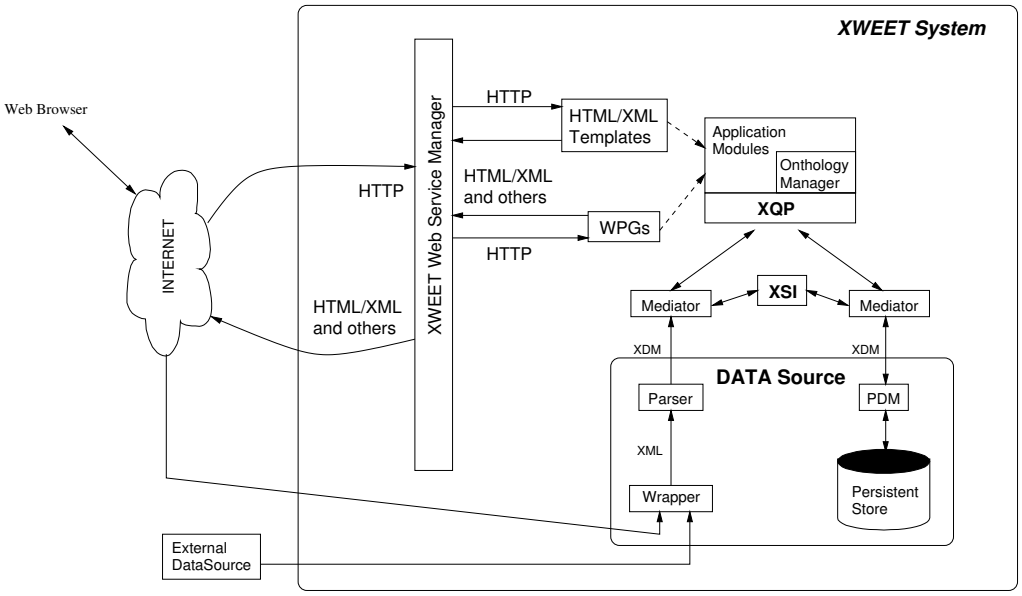

그림 2: XWEET 전체 구조

이질적인 구조의 데이타를 단일한 구조(XML)로 표현, 관리, 접근할 수 있는 미들웨어 시스템이다.

그림 2은 전체적인 XWEET 구조를 보여주고 있다. *Wrapper*는 여러 외부 데이타 소스(eg. Web 상의 여러 종류의 HTML 화일, e-mail, news, XML 화일 등)를 읽어 관심있는 정보만을 추출하여 XML 문서로 변환하는 모듈이다. *Data source*는 XWEET 에서 사용되는 여러 XML 문서 (또는 동등한 DOM 구조)를 받아 XDM(*X*WEET *D*ata *M*odel)로 변환하는 작업을 수행한다. XDM은 3 장에 자세히 설명되어 있다. 데이타 소스로는 여러 외부 데이타 소스로부터 XML 문서를 추출하는 래퍼, XWEET 자체적인 XML 데이타 저장소 PDM, 그리고 일반 화일 및 HTTP 를 통해 얻는 XML 문서가 있다. *PDM*(*P*ersistent *D*ata *M*anager)는 XWEET에서 직접 관리하는 XML 데이타들을 다루기 위한 모듈로 XML 데이타를 RDBMS에 효율적으로 저장/조작하는 기능을 제공한다. 또한 신속한 XML 데이타 검색을 위한 인덱스도 제공한다. XML 질의어를 하나의 SQL문장으로 변환하여 수행하면 많은 수의 죠인이 발생하고 새로운 인덱스가 만들어졌을 때 XQP의 성능향상을 제대로 반영하기가 어렵다. 그래서 OID로 객체를 인출하는 인터페이스를 제공한다. 많은 수의 미리 컴파일된 SQL 문장을 수행처리하는 식으로 동작한다. *XSI*(*X*WEET *S*emantic *I*ntegrator)는 사용자로부터 받은 정보를 바탕으로, 여러 데이터 소스를 통해 받은 XML의 의미적으로 동일하지만 다른 방식으로 인코딩된 XML 문서를 동일한 뷰(view)로 볼 수 있도록 하는 기능을 제공하는 모듈이다. 필요에 따라 입력 XML 문서의 내용을 변경하기도 한다. *Mediator*는 XSI와 협력하여 서로 다른 문법과 시

```
<!ELEMENT item (authorlist,title)>
<!ATTLIST item id CDATA #IMPLIED>
<!ELEMENT authorlist (author)*>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
```

그림 3: Wrapper에서 생성한 XML 화일

```
<!ELEMENT person (name, email*, title, organization)*>
<!ATTLIST person sex CDATA #IMPLIED>
<!ELEMENT name (#PCDATA|first_name|last_name)*>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT title (#PCDATA)>

<!ELEMENT organization (name, address, tel, zip)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
```

그림 4: PDM에 저장된 예제 XML 화일

맨틱을 사용하는 XDM 모델을 질의가 가능한 동일한 형태로 변환하는 일을 한다. *XQP*(*X*WEET *Q*uery *P*rocessor)는 데이타 소스로 부터 받은 여러 XML 데이타에 대한 선언적 질의를 처리하는 모듈이다. *Ontology Manager*는 스키마 통합이나 특정 응용 도메인에서 사용되는 용어들에 대한 통합 및 재정리를 담당한다. *Application Modules*는 XWEET에서 제공하는 여러 서비스를 바탕으로 응용 프로그램 고유한 작업을 처리하는 모듈들이다. *WPG*(*W*eb *P*age *G*enerator)는 웹 클라이언트에서 전달되는 HTTP 요청을 바탕으로 웹 페이지를 생성하는 모듈로, CGI, Servlet 등으로 구현될 수 있다. *HTML/XML Templates*는 WPG의 한 종류로 HTML/XML template을 통해 웹 페이지 생성을 보다 손쉽게 하는 모듈이다. *XWEET Web Service Manager*는 특정 응용에 속한 WPG들 사이의 데이타 및 제어 흐름을 관리하는 모듈이다.

아래에서 각각의 모듈에 대해서 자세히 설명하겠다.

```
Document(id, name, url)
Node(id, name, type, docid, parid, val, attr, groupping)
Attribute(id, name, elmid, val, type)
```

그림 5: PDM의 스키마

4.1 PDM

PDM은 XWEET 시스템 내부에서 관리하는 XML 데이타를 저장하기 위한 장치이다. 이것은 Java/JDBC로 구현되어 있으며 DOM 인터페이스를 지원한다. DOM은 XDM을 포함하는 인터페이스이기 때문에 XDM을 이용하여 PDM을 접근하는데는 문제가 없다. XML 데이타를 데이타베이스에 저장하는 방법은 크게 두가지 방법이 있다. 첫번째는 XML 문서의 각 엘리먼트의 의미를 살려서 저장하는 방법이다. 예를 들면 그림 4에서 문서의 엘리먼트인 `person`을 이용하여 `person` 클래스나 테이블을 생성하여 각 객체를 저장하는 방법이 있다. 두번째 방법은 XML 문서를 구조적으로 저장하는 것이다. 즉, 엘리먼트, 도큐먼트, 애트리뷰트를 저장하는 테이블 혹은 클래스를 생성하여 저장하는 것이다[17].

첫번째 방법의 장점은 SQL이나 OQL로 사용자가 직접 질의어를 이용하여 데이타를 가져올 수 있다는 점과 관련있는 데이타끼리 클러스터링 될 가능성이 높다는 것이다. 하지만 XML은 DTD가 없이 존재할 수 있어 문서의 구조적인 정보를 얻기가 힘든 경우도 있으며, DTD가 없음으로 인하여 반구조적인 데이타의 특징을 가지고 있다. 이러한 경우에 엘리먼트의 종류가 바뀔 경우 오버플로우 노드외에는 특별한 대안이 없는 단점이 있다. 또한 래퍼를 디자인 하는 경우 XML DTD가 달라지면 래퍼 또한 그에 따라 변해야 하기 때문에 래퍼를 자동적으로 생성하기가 어렵게 된다. 두번째 방법은 구조적인 방법으로 저장하는 형태이기 때문에 XML 문서의 의미적인 정보를 잃어버리는 단점이 있다. 즉 그림 3의 XML 문서를 보면 그 문서는 `person` 엘리먼트의 집합으로 구성되어 있다. 하지만 구조적으로 저장하면 `person` 엘리먼트가 `name` 엘리먼트와 다르다는 의미를 표현하는 것이 단지 하위 노드라는 점 밖에 없게 된다. 하지만 이 방법은 반구조적인 데이타를 저장하는데 아무런 문제가 없다. 또한 래퍼를 디자인 할 때도 항상 동일한 인터페이스로 접근할 수 있게 되는 장점이 있다. PDM에서는 두번째 방법인 구조적인 방법을 이용하여 XML 문서를 저장한다. 그림 5에 PDM의 스키마가 나타나있다.

PDM은 실제로 JDBC를 이용하여 XDM을 지원하는 래퍼로 구현되어 있다. 일반적으로 관계형 데이타베이스에 XML 데이타를 저장하였을때 XQL을 하나 혹은 수개의 SQL 문장으로 변환하여 데이타를 가져온다. 구조적인 방법으로 저장하였을 때 이러한 방법의 문제점은 죠인의 수가 너무

```
where <book> <title> $T </title> <price> $p </price>
      </book> in "www.a.b.c./bib.xml"
construct <result> <booktitle> $T </booktitle>
                   <bookprice> $P </bookprice>
          </result>
```

그림 6: XML-QL 예

많아지는 것이다. XML 데이타의 각 노드를 탐색하는 것이 모두 죠인으로 표현되기 때문에 하나의 XQL 질의어에 수십개의 죠인이 나타나는 경우가 빈번하게 발생한다. 이러한 문제를 해결하기 위하여 PDM에서는 단순히 OID를 이용하여 데이타를 가져오는 인터페이스만을 제공한다. 즉

```
select * from Node where id = OID
```

와 같은 질의어를 여러 번 수행하여 질의를 처리한다. PDM에서 수행하는 SQL 문장은 모두 미리 컴파일되어 있기 때문에 파싱과 같은 수행외 비용이 거의 들지 않는다. 또한 id는 인덱싱이 되어 있으므로 논리적 OID를 사용하는 객체지향형 데이타베이스와 그 성능의 차이가 크지 않게 된다. 하지만 이런 방법도 성능이 문제가 될 수 있으므로 몇개의 노드를 하나의 객체에 저장한다거나 애트리뷰트를 노드에 직접 기술할 수 있게 하는 등의 방법을 사용하여 성능을 향상시켰다.

PDM을 구현하기 위하여 객체지향형 데이타베이스인 ODMG[13]를 지원하는 SOP[3]에 저장하는 방법도 시도하였다. 이때 앞에서 언급한 바와 같이 의미를 보존하는 방법으로 저장하였다. 즉 XML 문서에서 스키마 정보를 추출하여 클래스를 만들고 Java, OQL, ODMG C++ 바인딩을 이용하여 래퍼를 구현하도록 하였다. 하지만 PDM에서 다루어야 하는 데이타는 반구조적인 특징을 가지고 있으므로 매번 스키마를 생성해야하는 방법이 적절치 않았다. 그래서 PDM은 JDBC를 이용하는 방법 만으로 구현하였다.

4.2 래퍼

PDM에 저장되어 있지 않은 다른 XML 관련 소스는 래퍼를 통하여 XWEET 시스템으로 읽혀들여진다. 래퍼는 논리적으로 하위 레벨의 객체를 공통된 XDM으로 변환하는 작업을 한다.

그림 6은 XML-QL의 예를 보여주고 있다. 여기서는 `www.a.b.c` 호스트로부터 `bib.xml`화일을 읽어서 `title`과 `price` 태그를 추출하는 예이다. XWEET 시스템에서는 대상이 되는 XML화일이 PDM을 통해 입력될 수도 있고, 다른 형식의 데이타가 래퍼를 통해서 변환될 수 있기 때문에 XML-QL의 `in` 절을 확장하였다.

```
DBLP-title    exec:/home/xweet/bin/xws.pl?script=dblp
XML           http://oopsla.snu.ac.kr/xweet/data.xml
```

그림 7: XWEET 시스템의 Data source 설정화일 예

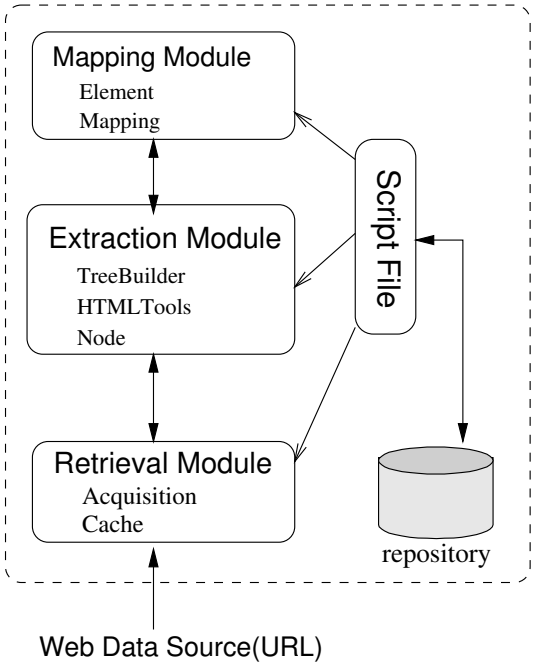

그림 8: XWS 시스템 구조

같은 구조의 XML 화일을 생성해내는 프로그램이나 데이타 소스는 XWEET의 입력 소스 설정 화일에 그림 7와 같이(이름, 프로그램 또는 데이타 소스의 URL) 쌍으로 등록을 한다. XML-QL에서는 `in` 항목에 "DBLP-title:title=xml"과 같은 형식으로 입력이 된다. DBLP라는 이름은 `http://www.informatik.uni-trier.de/~ley/dbbin/title` 페이지로부터 실시간으로 변환된 XML 화일을 나타내는 이름이다. XWS(XWEET Web–wrapper system)[6]에서 웹으로부터 읽어온 HTML 화일을 스크립트 설정화일을 바탕으로 XML화일로 변환하는데 이 때 동일한 스크립트 화일을 사용하는 각 변환모듈이 독립된 이름으로 명명된다. E-mail이나 news 등으로부터 래퍼를 통해서 변환되는 데이타 소스도 각각 수행되는 래퍼의 경로및 인자값을 설정화일에 등록하게 된다.

XWEET 시스템에서 PDM과 함께 주된 데이타 입력 소스로 쓰이는 HTML의 변환은 XWS 시스템에 의해 이루어진다. XWS는 문자열 처리에 뛰어난 능력을 지닌 Perl[40]을 이용하여 제작된 라이브러리와 프로그램으로 구성된 웹–래퍼 시스템이다. 그림 8는 XWS 시스템의 구조를 보여주

```
$html = getpage('http://www.informatik.uni-trier.de/~ley/dbbin/title?title=xml');

$h = new XWS::Node $html;
$r =  $h->elem_w('table',1)->elem_w('tr')->elem_w('td',2);

@string = $r->to_flat_string;
$result = convert_nl(\@string);

$xml = new XWS::Mapping ".thesis*.item(.id^ .authorlist*.author .title)\n", $result;
$xml->print_dtd();
$xml->print_xml();
```

그림 9: XWS 스크립트 언어 예

고 있다. XWS 시스템은 추출모듈(retrieval module), 가공모듈(extraction module), 그리고 대응모듈(mapping module)로 구성되어 있다. 사용자는 XWS에서 사용되는 선언적인 스크립트 언어를 이용하여 추출할 HTML 영역및 데이타를 기술하고 이 값을 대응모듈에 기술한 DTD 생성 언어를 이용하여 XML화일로 변환하게 된다. XWS에서 사용하는 스크립트 언어에 대한 예제가 그림 9에 나와 있다. DBLP에서 `XML` 타이틀을 가지는 논문들을 검색하여 XML화일로 변환하는 래퍼 모듈에서 사용되는 XWS 스크립트 예이다.

이렇게 바뀐 XML문서(그림 10)는 XML 파서를 통해 XDM 형태로 질의어 처리기에 전달되게 된다. XWS에서 객체지향적으로 디자인된 고수준의 연산자를 이용하여 원하는 데이타를 추출하도록 하였다. 그리고 좀 더 복잡한 처리가 필요한 래퍼는 Perl의 뛰어난 문자열 처리함수나 다양한 확장모듈을 이용하여 처리하도록 되어있다. 또한 XWS에서 쓰이는 설정화일 자체가 Perl로 작성되어 별도의 컴파일 과정없이 수행가능하고, 필요한 설정화일은 미리 지정된 저장소에서 실시간으로 다운로드 받아서 수행가능하기 때문에 여러 XWEET 시스템에서 공유하여 사용이 가능하다.

4.3 중개자와 XSI

서로다른 두개의 정보 소스로부터 얻어지는 객체가 같은 것인지를 결정하는 것은 힘든 문제 중의 하나이다. XQP에서 일반적인 질의어를 처리하는 경우는 사용자가 XML 데이타의 스키마 정보인 DTD에 대하여 알고 있다는 가정하에 처리한다. 그러나 정보를 표현하고 있더라도 각각의 XML 데이타가 서로 다른 태그를 사용할 수 있다. 예를 들어 사람의 이름을 표현하는 방법으로 단순히 `name`을 사용하는 경우와 `first name`, `last name`과 같이 사용하는 경우가 있을 수 있다. 단순히 태그의 이름만을 보면 두 경우의 태그는 서로 다른 것을 나타내지만 의미적으로 보면 같은 의미의 태

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE XWS_DOC [
  <!ELEMENT thesis (item)*>
  <!ELEMENT item (authorlist,title)>
  <!ATTLIST item id CDATA #IMPLIED>
  <!ELEMENT authorlist (author)*>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT title (#PCDATA)>
]>
<XWS_DOC>
 <thesis>
  <item id="0">
   <authorlist>
    <author>Takeyuki Shimura</author>
    <author>Masatoshi Yoshikawa</author>
    <author>Shunsuke Uemura</author>
   </authorlist>
   <title>Storage and Retrieval of XML Documents Using Object-Relational Databases</title>
  </item>
  ...
  <item id="99">
   <authorlist>
    <author>Bert Bos</author>
   </authorlist>
   <title>XML: From Bytes to Characters</title>
  </item>
 </thesis>
</XWS_DOC>
```

그림 10: XML로 변환된 HTML 페이지

그라는 것을 알 수 있다. 질의어 처리기가 질의를 처리하기 전에 XSI를 이용하면 사용자의 질의를 onthology에 맞게 질의어를 변환한 다음 각각의 중개자로 질의를 처리하게 한다.

각 데이타 소스의 입력값은 중개자를 거쳐서 좀 더 정제된 형태로 변환된다. 중개자는 특정한 타입의 정보를 처리하기 위한 정보를 포함하고 있다. 그리고 전체적인 관점에서 조정해야 하는 데이타는 XSI(XWEET Semantic integrator)와의 협조를 통해서 변환이 된다. 이 과정에서 DTD의 변환이나 포맷의 변환 작업, 불필요한 엘리멘트의 삭제등이 일어난다. 예를 들어 DBLP의 논문 검색 결과로 얻어진 (`author-list`, `title`, `book`, `pages`, `month`, `year`)등의 정보 중 `month`, `year`의 정보가 XQP에 의해 필요하지 않는 경우 이를 제외한 XDM을 넘겨주게 된다. 또한 데이타 소스의 출처에 대한 정보가 필요한 경우 데이타 변환작업을 하는 래퍼에 정보를 요구하여 해당되는 XML화일에 부가적인 태그정보를 추가하여 변환하는 일을 한다. 이를 통하여 이 정보가 `e-mail`에서 변환된 정보인지 `news`에서 읽혀진 정보인지 구별하게 된다.

```
where <person> <name> $P </name>
                <email> $E </email>
      </person> in "PDM:person.xml"
      <authorlist> <author> $P </author>
      </authorlist> in "DBLP-title=database"
construct <result>
                <email> $E </email>
          </result>
```

그림 11: XQP에서 사용되는 질의문 예

4.4 XQP

XQP(XWEET Query Processor)는 데이타 소스로부터 받은 여러 XML 데이타에 대한 선언적 질의를 처리하는 모듈이다.

XML 데이타는 그래프 기반의 비정형 데이타 모델[32, 34]에 매핑될 수 있다. 즉, O를 무한한 객체 식별자(object identity)의 집합, C를 O와 교집합을 가지지 않는 무한한 상수의 집합이라 가정하면 비정형 데이타 모델에서는 데이타 그래프 DB를 DB=(V,E)로 정의할 수 있다. 여기서 $V \subseteq O$는 노드의 집합, $E \subseteq V \times C \times V$는 방향성 있는 에지(directed edge)의 집합이다. XML 데이타는 XML의 요소(element)를 노드 V에, 태그를 에지 E에 매핑하면 비정형 데이타 모델에 매핑될 수 있다.

질의는 데이타 그래프의 패스에 대한 정규 패스 표현(regular path expression)을 기반으로 한다. 정규 패스 표현은 다음과 같이 정의된다.

Definition 1 *정규 패스 표현은* $R = \epsilon|a|\_|(R_1.R_2)|(R_1|R_2)|R^*$ *으로 표현된다. 여기서* R, R_1, R_2*는 정규 패스 표현,* $a \in C$*는 특정 상수,* ϵ*은 빈 문자열(empty string)을 나타낸다.*

그러므로 XQP에서 질의 처리는 XML 데이타가 매핑된 데이타 그래프 DB에 대하여 정규 패스 표현을 만족하는 그래프 상의 노드를 찾는 작업이 된다. 이때 가장 효율적으로 질의 처리를 수행하는 것이 XQP의 디자인 목표이다. 특히 XML은 비정형 데이타 모델에서는 없는 DTD(Documnet Type Definition)가 있어서 XML 데이타에 대한 스키마 정보를 제공하기 때문에 이 정보를 이용하면 효율적인 질의 처리를 할 수 있다. 예를 들면, 그림 3과 4의 XML 파일에 대하여 그림 11의 질의를 처리한다고 하자.

주어진 질의는 PDM의 person.xml 파일과 DBLP의 `title`페이지에서 `database`를 입력했을때의 결과 데이타를 XML화한 데이타에 대하여 그 이름이 같은 경우 그의 `email`을 출력하는 질의의다. 이때 XML 데이타에 대한 DTD는 다음과 같은 정보를 제공한다.

```
Q:  where <person> <name> Takeyuki Shimura  </name>
                  <email> $E </email>
                  <organization> <address> Seoul </address> </organization>
          </person> in "PDM:person.xml"
    construct <result> <name> Takeyuki Shimura </name>
                      <email> $E </email>
              </result>

V:  where <person> <name> $N </name>
                  <email> $E </email>
                  <organization> <address> Seoul </address> </organization>
          </person> in "PDM:person.xml"
    construct <result> <name> $N </name>
                      <email> $E </email>
              </result>
```

그림 12: 질의와 뷰의 정의

1. Person은 email을 가질 수도 있고 가지지 않을 수도 있다.

2. Person의 이름 항목은 직접 이름 항목이 올 수도 있고, 성과 이름 항목으로 나누어질 수도 있다.

이러한 정보를 이용하면 주어진 질의의 탐색 범위는 Person 중에서 그 이름이 성과 이름 항목으로 나누어지지 않는 사람중에서 email을 가지는 사람으로 제한될 수 있다. 즉, XQP에서는 DTD로부터 각 요소에 대하여 그 요소가 가지는 레이블에 따라 요소를 구분하여 질의 처리기에게 그 정보를 제공하는 기법인 NodeInfo와 각 노드가 그 하위 노드 전체에 대한 주요 레이블 정보를 제공하는 MergeNodeInfo 기법[5]을 제공한다.

또한, XQP에서는 형상화된 뷰를 이용한 질의 최적화 기법을 제안한다. 즉 사용자의 질의가 이전의 질의 결과를 가지고 있는 뷰를 포함하고 있다면 이전 질의 결과를 이용하여 효율적인 질의 처리를 할 수 있을 것이다.

그림 12와 같이 형상화 된 뷰 V와 질의 Q가 주어졌다고 하자.

질의 Q는 PDM에서 올라오는 person.xml 파일에서 그 사람이 속하는 조직의 주소가 서울이고 이름이 Takeyuki Shimura인 사람의 e-mail을 추출하기 위한 질의이다. 반면에 주어진 뷰는 마찬가지로 어떤 사람의 조직의 주소가 서울인 사람에 대하여 그 이름과 e-mail을 가지고 있는 뷰이다. 이 때 질의 Q는 의미상 뷰 V를 이용하여 다시 쓰여질 수 있고, 따라서 질의 최적화에 이용될 수 있다.

XQP에서는 이와같이 형상화된 뷰를 질의 최적화에 이용하는 2*-index-join[4] 기법을 제공한다.

2*-index-join 기법은 질의를 형상화된 뷰를 포함하는 질의로 바꾸는 질의 변환(query rewriting) 알고리즘과 2-index[38] 기법을 확장하여 질의 패스에 * 연산이 있을때 효율적인 처리 기법을 제공한다.

4.5 WPG와 HTML/XML templates

통상적으로 인터넷 상에서 개발되고 있는 많은 미들 티어(middle tier) 응용들은 응용의 처리 결과를 웹으로 공개하기 위하여, 웹 페이지 생성기(Web Page Generator: WPG)를 이용한다.

현재 가장 많이 사용되고 있는 웹 페이지 생성기는 CGI(Common Gateway Interface)이며, 어떤 언어로든 개발 가능하고 개발이 손쉽다는 장점에 힘입어 가장 널리 사용되고 있다. 현재 CGI를 위해 사용되는 언어 중 가장 널리 쓰이는 것에는 Perl, C++, C등을 들 수 있다. 또한 브라우저 측의 요구당 해당 요구를 처리하기 위해 웹 서버측에서 프로세스를 하나씩 생성되는 구조 때문에 서버측의 부담이 증가하는 것을 경감하기 위해 서블릿(Servlet), Active Server Page, PHP등의 기술이 개발되어 공개되었다. 그러나 이런 기술들은 웹을 기반으로 한 일반적인 응용의 개발에 촛점이 맞춰져 있기 때문에, 웹을 바탕으로 한 사업 수행(business processing)과, 사업 수행에 핵심이 되는 웹에서의 트랜잭션 처리 기술에 대해서는 해결방안을 제시하지 못하고 있다.

또한, 웹 환경에서 CGI와 같은 프로그램을 작성할 경우 객체지향 패러다임이나 컴포넌트 프로그래밍 등 기존의 프로그래밍 개발 방법론을 적용하기 힘들기 때문에, 개발된 시스템의 재사용 가능성이 떨어지고 아울러 새로운 시스템을 만드는 경우 기존의 프로그래밍 성과물들을 이용하기 힘들다는 문제가 있었다[24].

이러한 문제들을 해결하기 위해 XWEET 시스템에서는 웹 트랜잭션 서버 시스템인 WebTP을 설계하고 구현하였다[1,2]. 이 시스템은 쿠키 로깅, 웹 페이지 로깅(web page logging), 그리고 HRSN(HTTP Request Sequence Number)라는 세가지 방법에 의거하여 웹 트랜잭션의 일관성을 보존한다. 그림 13에 WebTP의 실행구조가 나와 있다. WebTP는 워크 개념을 도입하여 복잡한 웹 응용을 구조적으로 개발할 수 있도록 하고, 여러 대화로 구성된 웹 응용을 하나의 트랜잭션으로 처리될 수 있도록 하였다. 그리고 각 페이지 함수의 실행순서를 명시하도록 하여 명시된 순서에 어긋나는 페이지 함수의 실행을 막아 시스템이 비정상적인 상태로 전이되는 상황을 막을 수 있다. 또한 WebTP가 웹 응용의 문맥을 직접 관리할 수 있도록 하여 웹 응용 단위에서의 저장점 설치와 이를 이용한 부분철회를 지원하게 된다.

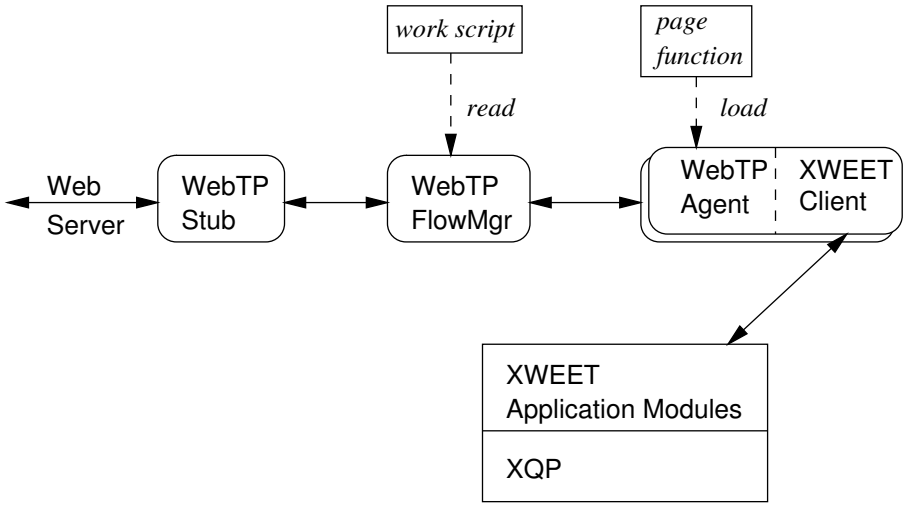

그림 13: WebTP의 실행구조

5 결론

이 논문에서는 여러 정보 소스에 존재하는 이질적인 구조의 데이타를 단일한 구조로 표현, 관리, 접근할 수 있는 미들웨어 시스템인 XWEET 시스템에 대해 설명하였다. XWEET 시스템은 서로 다른형태의 데이타를 통합, 저장, 접근하여 사용자에게 손쉽게 데이타에 대해 접근할 수 있는 방법을 제공한다. XWEET 시스템은 XML 데이타의 효율적인 저장, 추출, 질의를 위하여 설계되었다. XML 데이타의 저장을 위해 PDM을, 이질적인 정보 소스로부터의 데이타 통합을 위해 Wrapper 와 XWS를, 서로 다른 XML 문서 형식의 통합된 표현을 위해 XSI를 제공한다. 이렇게 통합, 저장된 데이타는 XML/QL 형식의 질의를 통해 접근되며, XQP는 제공되는 질의를 처리한다. 또한, 사용자들은 웹 환경에서 수행되는 응용을 작성할 수 있는 기반을 제공받는다. WPG와 HTML/XML Template 는 웹 응용의 수행 결과를 정의된 HTML이나 XML로 생성하도록 해 주며, WebTP 는 웹기반 워크플로우를 위한 기반 기능을 제공한다.

향후 연구계획으로는 XWEET 시스템에서 사용되고 있는 XQP의 인덱스 기법을 보강하여 멀티패스 인덱스에서 정규식을 처리가능하도록 확장하고, 전체적인 XWEET 시스템을 관리하고 조절할 수 있는 GUI 툴을 제공할 예정이다. 데이타 소스로 사용되는 XWS 시스템에서 가변적으로 변하는 웹 페이지로부터 정보를 추출하기 위해서 휴리스틱 모듈을 추가하려한다.

참고 문헌

[1] 이 강우, 김 형주. 웹 트랜잭션 처리 시스템의 구현. *한국정보과학회 논문지(C)*, 5(1), 2 1999.

[2] 이 강우, 김 형주. 일관성 유지를 위한 웹 트랜잭션 처리 시스템의 지원. *한국정보과학회 논문지(B)*, 2000.

[3] 안정호, 김형주. SRP에서 SOP까지. *한국정보과학회 Review지*, April 1994.

[4] 정태선, 김형주. 형상화된 뷰를 이용한 향상된 XML 질의 처리 기법. Technical report, Seoul National University, Feb 2000. http://oopsla.snu.ac.kr/xweet/material.ps.gz.

[5] 정태선, 김형주. DTD를 이용한 XML 데이터에 대한 질의 최적화 기법. Technical report, Seoul National University, Feb 2000. http://oopsla.snu.ac.kr/xweet/dtd.ps.gz.

[6] 정재목. XWS: 웹 정보의 추출및 통합. Technical report, Seoul National University, Feb 2000. http://oopsla.snu.ac.kr/xweet/xws.ps.gz.

[7] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web From Relations to Semistructured Data and XML.* Morgan Kaufmann Publisher, 2000.

[8] 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.

[9] B. Adelberg. Nodose - a tool for semi-automatically extracting semi-structured data from text documents. In L. M. Haas and A. Tiwary, editors, *SIGMOD 1998*, pages 283–294. ACM Press, 1998.

[10] G. Arocena and A. Mendelzon. WebOQL: Restructuring documents, databases and webs. In *Proc. of 14th. Intl. Conf. on Data Engineering (ICDE 98)*, 1998.

[11] C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, and P. Velikhov. XML-based Information Mediation with MIX. *SIGMOD Systems Demonstration*, 1999.

[12] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A Query Language and Optimization Techniques for Unstructured Data. *SIGMOD*, 1996.

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

[14] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, 1994.

[15] A. Deutsch, M. Fernadez, and D. Suciu. Storing semistructured data with stored. In *SIGMOD*, 1999.

[16] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. *XML-QL: A Query Language for XML*, 1998.

[17] D. Florescu and D. Kossman. A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database. *INRIA Technical Report No.3680*, May 1999.

[18] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, , and J. Widom. Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. *In Proceedings of the AAAI Symposium on Information Gathering*, pages 61–64, 3 1995.

[19] R. Goldman, J. McHugh, , and J. Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. *WebDB*, June 1999.

[20] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Dataguides: Enabling Query Formulation and Optimization in Semistructured Databases. *Proceedings of the Conference on Very Large Data Bases*, 1998.

[21] J. Hammer, H. Garcia-Molina, S. Nestorov, R. Yerneni, M. Breunig, and V. Vassalos. Template-based wrappers in the TSIMMIS system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 26,2 of *SIGMOD Record*, pages 532–535, New York, May 13–15 1997. ACM Press.

[22] G. Huck, P. Fankhauser, K. Aberer, and E. J. Neuhold. Jedi: Extracting and synthesizing information from the web. In *CoopIS 1998*, pages 32–43, 1998.

[23] D. Konopnicki and O. Shmueli. W3QS: A Query System for the World Wide Web. *VLDB*, 1995.

[24] A. Kristensen. Developing HTML based Web Applications. In *First International Workshop on Web Engineering*, april 1998.

[25] N. Kushmerick, R. Doorenbos, and D. Weld. Wrapper induction for information extraction. In *International Joint Conference on Artificial Intelligence*, 15, 1997.

[26] L. Lakshmanan, F. Sadri, and I. Subramanian. A Declarative Language for Querying and Restructuring the Web. *RIDE*, 1996.

[27] K.-W. Lee and H.-J. Kim. Support of a Web Transaction Processing System for Preserving Consistency. *2nd Web Technology Workshop (AREAU 99)*, Oct. 1999.

[28] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A Database Management System for Semistructured Data. *SIGMOD Record*, 26(3), 9 1997.

[29] G. Mecca, P. Atzeni, A. Masci, P. Merialdo, and G. Sindoni. The Araneus Web-Based Management System. *SIGMOD*, 1998.

[30] A. O. Mendelzon, G. A. Mihaila, and T. Milo. Querying the world wide web. *Journal on Digital Libraries*, 1:54–67, 1997.

[31] A. Nori. Database technology for the internet. In *ICDE*, 1999.

[32] Peter Buneman. Semistructured data. In *Proceedings of ACM Symposium on Principles of Database Systems*, 1997.

[33] A. Sahuguet and F. Azavant. Building light-weight wrappers for legacy web data-sources using w4f. In *VLDB*, 1999.

[34] Serge Abiteboul. Querying semi-structured data. In *Proceedings of the International Conference on Database Theory*, 1997.

[35] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, and J. F. Naughton. Relational databases for querying xml documents: Limitations and opportunities. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, editors, *VLDB '99*, pages 302–314. Morgan Kaufmann, 1999.

[36] J. Simeon and S. Cluet. Using YAT to Build a Web Server. *WebDB*, 1998.

[37] M. Stonebraker and P. Brown. *Object-Relational DBMSs Tracking The Next Great Wave*. Morgan Kaufmann, 2 edition, 1999.

[38] Tova Milo and Dan Suciu. Index structures for path expressions. In *Proceedings of the International Conference on Database Theory*, 1999.

[39] W3C. *Document Object Level (DOM) Level 1 Specification*, oct 1998. http://www.w3.org/TR/.

[40] L. Wall, R. L. Schwartz, T. Christiansen, and S. Potter. *Programming Perl*. Nutshell Handbook. O'Reilly & Associates, 2nd edition, 1996.

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

[42] World Wide Web Consortium (W3C). Extensible Stylesheet Language(XSL), 1998. http://www.w3.org/Style/XSL.