공학박사 학위논문

Scheme을 이용한 XML 기반의 데이터 관리 웹 에이전트 시스템 설계 및 구현

Design and Implementation of XML based Data Management Web Agent Systems using Scheme

지도교수 임 재 홍

2005년 2월

한국해양대학교 대학원

전자통신공학과

유 선 영

공학박사 학위논문

Scheme을 이용한 XML 기반의 데이터 관리 웹 에이전트 시스템 설계 및 구현

Design and Implementation of XML based Data Management Web Agent Systems using Scheme

지도교수 임 재 홍

2005년 2월

한국해양대학교 대학원

전자통신공학과

유 선 영

목 차

Abbreviations

Abstract

제 1 장 서 론 ····· 1

제 2 장 XML ····· 4

2.1 등장 배경 ····· 4

2.2 XML 개요 ····· 6

2.3 XML의 문서 구조 정의 ····· 7

제 3 장 XML과 데이터베이스 ····· 17

3.1 데이터베이스 ····· 17

3.2 XML과 데이터베이스 연구 현황 ····· 18

3.3 XML 기반의 효율적 데이터 저장 관리 ····· 21

3.4 XML 접근 방식에 따른 DBMS 유형 ····· 22

3.5 데이터와 도큐먼트 ····· 31

3.6 XML 미들웨어 ····· 34

제 4 장 스킴 ····· 36

4.1 스킴 등장 배경 ····· 36

4.2 Dr.Scheme ····· 37

4.3 스킴의 기본 문법 ····· 42

제 5 장 DTD 생성 규칙 ····· 60
5.1 XML과 데이터베이스 매핑 ····· 60
5.2 데이터베이스 변환 DTD 생성 규칙 ····· 64

제 6 장 웹 에이전트 시스템 ····· 67
6.1 웹 에이전트 시스템 설계 ····· 67
6.2 웹 에이전트 시스템 구현 ····· 74
6.3 고찰 ····· 97

제 7 장 결 론 ····· 98

참 고 문 헌 ····· 100

표 차 례

<표 2-1> XML 구성 요소 ····· 6
<표 2-2> XML 스키마 데이터 타입 ····· 13
<표 6-1> TEST 테이블 데이터 ····· 80
<표 6-2> 데이터 검색을 위한 문제 ····· 83

그 림 차 례

<그림 4-1> 언어 선택 창 ····· 38
<그림 4-2> Dr.Scheme 언어 레벨 선택 창 ····· 38
<그림 4-3> Dr.Scheme 실행 화면 ····· 39
<그림 4-4> (+ 2 (* 3 4)) 실행 화면 ····· 40
<그림 4-5> 첫 번째 계산 과정 ····· 41
<그림 4-6> 두 번째 계산 과정 ····· 41
<그림 4-7> define 문법 예제 ····· 44
<그림 4-8> cond 문법 예제 ····· 45
<그림 4-9> if 문법 예제 ····· 46
<그림 4-10> 언어 선택 창 ····· 48
<그림 4-11> 언어 변경 화면 ····· 49
<그림 4-12> lambda 문법 예제 ····· 49
<그림 4-13> list 문법 예제 ····· 51
<그림 4-14> car, cdr 문법 예제 ····· 52
<그림 4-15> Teachpack 추가 화면 ····· 53
<그림 4-16> map 문법 예제 1 ····· 54
<그림 4-17> map 문법 예제 2 ····· 55
<그림 4-18> odd?, even? 사용 예 ····· 56
<그림 4-19> filter 문법 예제 ····· 57
<그림 4-20> fold-right/fold-left 문법 예제 ····· 59
<그림 5-1> 테이블-기반 매핑 ····· 61
<그림 5-2> 객체-관계형 매핑 ····· 62
<그림 5-3> 테이블-리스트 매핑 ····· 63
<그림 5-4> 단독 테이블인 경우 ····· 64

<그림 5-5> 다중 관계형 테이블인 경우 ····· 65
<그림 5-6> DTD 문서 ····· 66
<그림 6-1> 웹 에이전트 시스템 구성 ····· 69
<그림 6-2> 데이터베이스 관리 데이터 흐름 ····· 70
<그림 6-3> XML 문서를 데이터베이스에 저장하는 과정 ····· 71
<그림 6-4> 서로 다른 DBMS간의 데이터 복제 ····· 73
<그림 6-5> 데이터베이스 로그온 웹 인터페이스 ····· 75
<그림 6-6> 로그온 프로그램 소스 ····· 76
<그림 6-7> 자바와 JDBC 드라이버의 경로 설정 ····· 78
<그림 6-8> 오라클 데이터베이스의 테이블 목록 ····· 78
<그림 6-9> 데이터베이스 연결 후 결과 화면 ····· 79
<그림 6-10> TEST 테이블 컬럼 목록 ····· 80
<그림 6-11> test.scm 파일 소스 ····· 82
<그림 6-12> TEST 테이블 데이터 목록 ····· 83
<그림 6-13> 리스트를 이용한 풀이 ①, ② ····· 84
<그림 6-14> 리스트를 이용한 풀이 ③ ····· 86
<그림 6-15> 리스트를 이용한 풀이 ④ ····· 87
<그림 6-16> 리스트를 이용한 풀이 ⑤ ····· 89
<그림 6-17> 데이터 검색을 위한 화면 ····· 90
<그림 6-18> 데이터 검색 결과 화면 ····· 91
<그림 6-19> 테이블 생성 화면 ····· 92
<그림 6-20> 테이블 생성 프로그램 소스 ····· 93
<그림 6-21> 테이블 삭제 프로그램 소스 ····· 94
<그림 6-22> Test.xml 문서 ····· 96

Abbreviations

API	Application Program Interface
BLOB	Binary Large Object
CMS	Content Management System
CSS	Cascading Style Sheets
DBMS	DataBase Management System
DHTML	Dynamic HTML
DOM	Document Object Model
DTD	Document Type Definition
EBNF	Extended Backus-Naur Form
GML	General Markup Language
HTML	Hyper Text Markup Language
IEEE	Institute of Electrical and Electronics Engineers
IIS	Internet Information Server
ISO	International Standard Organization
JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JSP	Java Server Page
LISP	LISt Processing
ODBC	Open Database Connectivity
OLE	Object Linking and Embedding
OODBMS	Object-Oriented DataBase Management System
PDF	Portable Document Format
RDBMS	Relational DataBase Management System
RTF	Rich Text Format
SAX	Simple API for XML
SGML	Standard Generalized Markup Language
SQL	Structured Query Language
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XPath	XML Path Language
XQL	XML Query Language
XQuery	XML Query
XSU	XML SQL Utility

Abstract

XML(eXtensible Markup Language) has recently emerged as a new standard for data representation and exchange on the Internet. Web servers and applications encoding their data in XML can quickly make their information available in a simple and usable format, and such information providers can interaction easily. Information content is separated from information rendering, making it easy to provide multiple views of the same data. Most of us are familiar with web sites that contain vast databases of useful information, but whose query and search facilities are surprisingly primitive.

This thesis proposes web agent for information retrieval and information extraction from DBMSs(Database Management System) using table-list mapping method. Web agent accesses database using information, such as IP address, port number, database name and password. After web server is connected with database, it takes data using query sentence. Web server transforms data to XML DTD(Document Type Definition) by table-list mapping method. Table-list mapping method using Scheme language, it is easy to process vast data at once.

Web server makes XML document from source database data, XML document saves destination database. Using these methods, web agent exchange data between heterogeneous databases. It is easy to manage data on the web, exchange data between heterogeneous DBMSs using XML and Scheme language.

제 1 장 서 론

인터넷이 널리 보급됨에 따라 지식과 정보 교류의 기반이 웹(web)으로 옮겨지고, 일반인에게 익숙한 웹에서 정보를 추출하거나 가공하고, 정보를 공유하거나 교환하려는 요구가 증가하는 추세이다. 현재 웹 문서를 작성하는 하이퍼텍스트 생성 언어(HTML ; HyperText Markup Language)는 초기 출판을 목적으로 탄생된 생성 언어로 웹 브라우저(browser) 화면에서 보여줄 문서의 모양을 정의하는 표현 위주의 언어이다.

제한된 태그(tag)를 사용하는 HTML은 정보 축적과 추출하는 방법이 비효율적이고, 추출한 정보의 가공도 어렵다[1]. 이는 웹에서 정보를 관리하려는 사용자의 요구에 적합하지 않기 때문에 HTML의 단점을 보완할 수 있는 새로운 표준 즉, 새로운 형태의 언어가 필요하게 되었다. 월드 와이드 웹 컨소시엄(W3C ; World Wide Web Consortium)은 1996년 웹 문서의 표준으로 표준 범용 생성 언어(SGML ; Standard Generalized Markup Language)를 기반으로 하는 확장성 생성 언어(XML ; eXtensible Markup Language)를 제안하였다. 생성 언어란 문서의 본문 내용 이외에 첨가되는 부가적인 정보를 기술하는 언어이다.

XML과 HTML은 SGML의 부분집합이지만 HTML은 문서 표현을 중시하는 반면, XML은 문서 내용에 대한 구조와 의미를 기술한다[2]. XML은 문서 하나를 여러 형태로 보여줄 수 있으며, 컨텐츠(contents)를 기반으로 데이터를 필터링(filtering)하거나 응용 목적에 맞게 재구성이 가능하다. 또한 XML이 단순한 컨텐츠에서 데이터베이스로까지 그 적용 분야가 확장되면서 XML로 표현된 정보를 효율적으로 저장하고 관리하는 방법에 대한 기술도 연구되고 있다[3].

오늘날 인터넷이 널리 확산되면서 웹을 이용하여 정보를 검색하거나 인터넷을 통한 구매 활동이 활발하게 되고, 기업은 웹을 이용하여 데이터베이스를 관리하기 시작했다. 이에 따라 웹에서 데이터베이스를 관리하고 문서를 데이터베이스에 저장하거나 데이터베이스에 저장되어 있는 내용을 웹 문서로 표현하고, 서로 다른 데이터베이스에 저장되어 있는 내용을 웹에서 교환하기 위한 연구가 진행되고 있다. 기존의 데이터베이스 관리 시스템(DBMS ; DataBase Management System)에서 저장된 데이터를 웹에서 교환하기 위해서는 표현 위주의 HTML보다는 문서 구조에 대한 정보를 가지고 있는 XML이 더 적합하다.

그러나 XML은 구조상 기존 데이터베이스와 많은 차이점이 있어 관계형 데이터베이스나 객체지향형 데이터베이스에 바로 적용하기 어렵다. 데이터베이스와 XML간의 데이터를 교환하기 위해 데이터베이스 관리 시스템 구조를 XML로 변환하거나 데이터베이스 관련 구조를 XML 문서로 변환하는 기술이 필요하다. 데이터베이스 관련 구조를 데이터 타입으로 저장하기 위해서는 XML 문서와 데이터베이스 사이의 매핑(mapping)이 중요하며 XML 문서의 구조적 정보를 토대로 데이터베이스에 그 형태대로 저장하는 방식에 대한 연구가 진행되고 있다[4].

XML을 데이터 교환의 매개체 역할을 하면서 데이터베이스에 저장된 방대한 데이터를 XML 형식으로 변환시켜 기존의 정보를 보다 효율적으로 사용할 수 있는 기반을 마련하고, 이와는 반대로 데이터 교환을 통해 생성된 XML 데이터를 어떻게 데이터베이스에 저장하고 XML 문서에 대해 데이터베이스 기술을 어떻게 효율적으로 적용시킬 것인가에 대한 연구가 필요하다. 또한 데이터를 효율적으로 검색하고 처리하는 방법에 대한 연구도 필요하다.

따라서 본 논문은 데이터베이스에 저장된 데이터를 XML 문서로 변환하고 XML 문서를 데이터베이스에 저장하는 규칙을 제안하고, 이 규칙을 이용하여 웹에서 데이터베이스와 XML 사이의 데이터를 교환하고 관리하는 웹 에이전트 시스템을 구현하였다. 구현한 웹 에이전트 시스템은 데이터베이스에 저장된 데이터를 XML로 변환하거나 XML을 데이터베이스에 저장하고, 웹을 이용하여 초보자에게도 데이터베이스에 쉽게 접근하여 데이터베이스의 종류에 상관없이 데이터베이스를 관리할 수 있다. 또한 웹 에이전트 시스템은 데이터베이스에 저장된 데이터를 리스트로 표현하여 스킴(Scheme) 언어를 이용해 정보를 추출하거나 계산하고 검색하는 것을 이용하는 사용자들에게 익숙한 방법으로 구현하였다.

본 논문의 구성은 제 2 장에서 XML의 등장 배경과 개요에 대하여 설명하고, 제 3 장에서는 데이터베이스와 XML 접근 방식에 따른 데이터베이스 관리 시스템을 기술한다. 제 4 장에서는 데이터베이스로부터 가져온 정보를 검색하기 위해 이용한 프로그래밍 언어인 스킴을 소개하고, 제 5 장에서는 본 논문에서 제안한 테이블-리스트 매핑 방법과 XML 문서형 정의(DTD ; Document Type Definition)를 생성하는 규칙을 제안하며, 제 6 장에서는 본 논문에서 제안한 DTD 생성 규칙을 적용한 웹 에이전트 시스템의 특성과 데이터 처리 과정을 소개하고 구현한 웹 에이전트 시스템에 대하여 고찰한다. 끝으로 제 7 장에서는 결론 및 향후 연구 과제에 대해 논의한다.

제 2 장 XML

2.1 등장 배경

SGML은 문서 작성 언어로써 문서의 각 구성 요소가 논리적인 구조를 갖도록 문서의 계층적 구조를 설정할 수 있는 방법을 제시한다. SGML은 모든 문서 작성 소프트웨어가 인식할 수 있는 일반적인 마크업(markup)을 사용하고, 사용자는 자신이 필요한 문서의 종류 및 그 문서에서 사용할 태그 종류 및 태그의 사용 규칙을 정하여 사용한다. 마크업 정보를 별도로 분리하고 분리된 정보에 접근하기 위하여 사용하는 태그는 문서 작성 중에 문서의 구성 요소가 어떻게 보일 것인가에 대한 고려 없이 그 구성 요소가 문서 내에서 내용상 어떤 성격을 가질지만 고려한다.

SGML 문서는 일반화된 마크업을 이용함으로써 하드웨어 및 소프트웨어의 종류에 관계없이 사용할 수 있으므로 서로 다른 시스템을 가진 어느 사용자와도 문서를 교환할 수 있다. 하드웨어 및 소프트웨어의 종류에 관계없이 서로 다른 기종 간의 문서를 교환하거나 문서의 정보를 검색할 수 있는 SGML은 국제 표준화 기구(ISO ; International Standard Organization)에서 1986년에 제정된 문서 정보를 기술하기 위해 필요한 요소를 완벽하게 정의한 표준이다[5].

SGML은 문헌이나 문서의 표준 규격을 규정하는 것이 아니라 문서의 다양성을 정의할 수 있는 일종의 메타(meta) 언어이다. 소프트웨어 패키지(package)나 소프트웨어 제공업자, 하드웨어 기종과 관계없이 문서를 저장하고 정보를 교환하기 위한 개념에서 출발한 SGML은 지나치게 복잡하여 관련 소프트웨어의 개발이 쉽지 않다. SGML은 복잡하고 고가의 툴(tool)을 사용하며 사용을 위한 투자비

용이 높고, 전체를 지원하는 소프트웨어의 개발이 어렵기 때문에 웹에서 상용화되지 못했다.

1991년 팀 버너스-리(Tim Berners-Lee, 영국)는 자신의 기술 논문을 쉽게 작성하고 다른 기종을 사용하는 학계의 사람들이 원래의 문서 모양을 그대로 볼 수 있도록 HTML을 만들었다[6].

현재의 인터넷 혁명을 가능하게 한 HTML은 태그의 사용이 제한적이고, 새로운 응용 프로그램이나 웹 기술을 다루는데 부적절하다. 또한, 웹에서 정보를 표현하는 목적으로 설계된 HTML은 태그로 묶인 데이터에서 정보를 추출하기 어렵다[7]. 이러한 HTML 한계를 극복하기 위해 여러 가지 스크립트(script) 언어나 종속형 시트(CSS ; Cascading Style Sheets), 동적 HTML(DHTML ; Dynamic HTML) 등을 이용하여 한계를 극복하고자 하였으나, 결국 근본적인 문제점을 해결하지 못했다.

유리 루빈스키(Yuri Rubinsky, 레바논)는 HTML의 단점인 장애자용 브라우저나 점자 출력이 불가능하다는 점을 지적하면서 장애자를 위한 웹을 주장했다. 그의 사후 W3C SGML 워킹 그룹(working group) 회원이 모여 유리의 정신을 기리며 새로운 생성 언어에 대하여 논의하기 시작하였다.

1996년 4월 팀 브레이(Tim Bray, 미국)가 구조화된 문서를 위한 XML의 설계 목표를 발표했다. 팀 브레이는 XML의 목적을 “미래의 웹에서 SGML을 사용자 에이전트(agent)가 전달받고 처리하는 것이 가능하도록 만드는 것이다”라고 정의했다. 또한, XML은 SGML을 경량화해서 사용하기 쉽고 응용 프로그램을 구현하기 쉽게 구성되었다고 발표하였다. 1996년 11월 보스톤에서 열린 SGML 컨퍼런스(conference)에서 처음 XML 초안이 발표된 후, 1998년 2월 10일에는 W3C 권고에 따르는 XML 1.0 사양이 발표되고 오늘날까지 꾸준히 연구되고 있다[8].

2.2 XML 개요

XML은 1980년대 초반에 미국 국무성에서 군사적 목적으로 만들어진 범용 문서 생성 언어(GML ; General Markup Language)에서 시작되었다. 1986년 산업계 전반의 광범위한 문서화 작업을 위해 GML로부터 확장된 SGML은 ISO의 표준이 된 후, 20년간 널리 사용되었다. XML은 SGML과 완벽한 호환성을 유지하면서 인터넷에 더욱 적합하도록 발전시킨 SGML의 한 부분집합이라 할 수 있다.

웹에서 사용 가능한 XML은 언어에 대해 독립적이고 여러 응용 프로그램을 지원하며 유연하고 개방적인 표준 기반 형식으로 뛰어난 상호 운용성을 제공한다. XML은 SGML에서 파생되었기 때문에 SGML과 호환이 가능하고, 프로그래밍이 용이해서 문서를 처리하는 프로그램 작성이 쉽다. 뿐만 아니라, DTD가 보이지 않는 HTML과 달리, XML은 필요에 따라 DTD를 정의하거나 생략할 수 있다. 다양한 형태의 정보와 연결할 수 있고, 유니코드 기반(UTF-8 및 UTF-16을 지원)이므로 외국어를 표현하는데 용이하다. XML의 구성을 간단히 살펴보면 <표 2-1>과 같다.

<표 2-1> XML 구성 요소

<Table 2-1> XML Construction Element

항 목	표 준
내용(Contents)	XML
구조(Schema)	DTD, Schema
접근(Acess)	DOM, Parser
변환(View/Transaction)	XSL, XSLT
획득	XQL
링크(link)	Xlink, Xpointer

2.3 XML의 문서 구조 정의

XML은 사용자 정의 태그를 이용하여 문서를 작성할 수 있으나, 사용자 정의 태그를 이용하여 작성된 XML 문서는 같은 의미적 요소를 다르게 표현하거나 순서성이 없을 수 있다. 이러한 XML 문서 작성은 일관성 없는 관리가 이루어지고 이로 인한 문제를 해결하기 위해서는 문서의 구조를 정의할 수 있는 표현이 필요하다. XML 문서를 정의하는 방법으로는 DTD와 스키마(schema)가 있다.

2.3.1 DTD

문서의 형태를 정의하는 DTD는 XML 문서의 구조를 명시적으로 선언하는 역할을 하며, XML 문서가 잘 만들어진 유효한 문서인지 확인하기 위해 사용되는 문서이다. DTD를 사용하여 요소와 요소의 내용, 속성과 속성 내용, 요소의 순서나 반복성 등을 미리 정의해 놓고 XML 문서를 작성하게 하는 것이다. 이는 XML 문서를 작성할 경우 잘못된 문서를 작성하게 되는 실수를 줄일 수 있다.

데이터베이스 개발자들이 데이터베이스 구조에 따라 응용 프로그램을 작성하듯이 XML 개발자들은 DTD에 따라 응용 프로그램을 작성한다. 즉 DTD는 XML 문서 안에 결합되거나 외부에 독립적으로 존재할 수 있는 XML 문서이다. DTD에는 XML 문서가 어떻게 구조화되고 어떤 요소를 포함해야 하며 어떤 종류의 데이터가 포함되어야 하는지를 정하고 기본값이 무엇인지 등을 규정하는 규칙을 정의하고 있다.

그러나 XML 문서는 단 하나의 DTD만을 가져야 한다는 제한조건이 있다. DTD는 원래 SGML에 사용되기 위해 개발된 것으로

XML 문법과 다르다. DTD는 XML 문서에 대한 논리, 물리 구조를 정의하게 되며 그 내용은 다음과 같다.

- 내용 모델과 문서에서 허용되는 요소(element)형을 정의.
- 각 요소에 할당되어 있는 속성을 정의.
- 문서에 허용되는 엔티티(entity)를 정의.
- 외부 엔티티와 함께 사용되는 표기법을 정의.

DTD는 XML 문서에서 사용되는 태그들을 위한 일련의 규칙으로 문서 안에서 어떤 태그를 쓸 수 있고, 태그들이 문서 안에서 어떤 순서로 나타나야 하며, 다른 문서들 속에서 어떤 태그들이 나타날 수 있는지, 어떤 태그가 속성(attribute)을 갖는지 알려 준다[9].

XML 개발자들은 DTD에 따라 응용 프로그램을 작성하므로 DTD를 가지고 무한히 많은 XML 문서를 만들 수 있다.

XML은 언어 자체가 아니고 언어를 정의하는 체계이기 때문에 HTML이 가지고 있는 것과 같은 공통 DTD를 갖지 않는다. 대신 데이터 교환을 위해 XML을 사용하고자 하는 경우는 독자적인 DTD를 정의할 수 있다.

DTD 문서는 다음과 같은 문법에 의해 작성된다.

(1) 선언

DTD 문서를 선언하는 방법은 다음과 같다.

<!DOCTYPE *root_element source location [internal DTD]*>

- DOCTYPE은 DTD 문서 정의 키워드
- *root_element*는 XML 문서의 최상위 요소
- *source*는 “SYSTEM” 또는 “PUBLIC” 키워드를 사용하여 DTD 문서의 허용범위를 나타낸다.
- *location*은 외부 선언 DTD 파일의 위치
- *[internal DTD]*는 내부 DTD를 정의하는 부분으로 기본적인 형식은 다음과 같고 형식은 일반 DTD 작성 방법과 동일하다.

  <!keyword parameter1 parameter2 .....>

(2) 요소 선언

요소(element)를 선언하는 방법은 다음과 같다.

<!ELEMENT *element_name contents*>

- ELEMENT는 XML 문서에서 사용되는 요소를 선언하는 키워드
- *element_name*은 정의하려는 XML 문서의 요소 이름
- *contents*는 요소가 가질 수 있는 데이터 타입을 정의한다.
- *contents*는 일반 텍스트만을 포함하는 #PCDATA, 자식 요소를 포함하는 자식(children), 일반 텍스트와 자식 요소를 같이 포함하는 혼합, 요소에 내용이 없는 빈 요소를 선언하는 공백, 요소의 모든 데이터를 포함할 수 있는 any가 있다.

1) #PCDATA

요소에 문자 데이터를 갖는 요소는 다음과 같이 선언한다.

<!ELEMENT element_name (#PCDATA)>

2) 자식

최상위 요소의 하위 요소인 자식 요소는 다음과 같이 선언한다.

<!ELEMENT element_name (child1, child2 ...)>

3) 혼합

문자와 자식 요소를 혼합하는 요소는 다음과 같이 선언한다.

<!ELEMENT element_name (#PCDATA | child1 ...)>

- 문자 데이터 선언과 함께 정의할 경우 '|'를 사용하여 선언한다.
- 문자 데이터 선언은 반드시 먼저 선언한다.

4) 공백

데이터를 가지지 않는 공백(empty)은 다음과 같이 선언한다.

<!ELEMENT element_name EMPTY>

5) ANY 선언

요소에 대해 아무런 조건을 두지 않을 경우 다음과 같이 선언한다.

<!ELEMENT element_name ANY>

(3) 속성 선언

요소가 가지고 있는 속성(attribute)은 다음과 같이 선언한다.

<!ATTLIST *element_name attribute_name attribute_type*
attribute_default data_default>

- ATTLIST는 속성에 대한 선언을 시작하는 키워드
- *element_name*은 XML 문서에서 작성하는 요소 이름
- *attribute_name*는 *element_name*에 적용할 속성의 이름을 지정
- *attribute_type*은 열거형(Notation), 문자열(CDATA), 토큰
- *attribute_default*는 속성의 기본값
- 속성의 기본값은 속성 초기값, #FIXED, #IMPLIED, #REQUIRED가 있다.
  ① 속성 초기값은 XML 문서에서 속성을 사용하지 않을 경우에 파서(parser)가 DTD에서 선언한 속성 초기값을 사용한다.
  ② #FIXED는 속성이 하나의 고정된 값을 가진다. 즉, 속성은 초기값과 동일하게 정의하고 정의한 초기값을 XML 문서 내에서 사용해야 한다.
  ③ #IMPLIED는 반드시 속성을 요구하지 않는다. 즉, 요소의 속성을 선택적으로 사용하거나 사용하지 않을 수 있다.
  ④ #REQUIRED는 요소에 대한 속성이 반드시 존재하는 경우에 선언한다. XML 문서에서 요소를 사용할 때 반드시 속성값을 지정해 주어야 한다.

(4) 카디넬리티 규칙

- 요소가 반복되는 수나 형태를 나타내는 규칙
- 기호가 없으면 지정한 요소를 1번만 사용한다.
- 쉼표(,)는 선언한 요소의 순서대로 자식 요소를 사용한다.
- ?는 서브 요소는 없거나 한번 나타난다(0～1).
- *는 서브 요소가 없거나 여러 번 반복해서 나타난다(0～n).
- +는 서브 요소가 적어도 한 번 이상은 나타나야 한다(1～n).
- |는 나열된 요소 중에서 하나가 서브 요소가 된다.

2.3.2 스키마

XML 스키마는 DTD와 마찬가지로 문서 구조를 표현하기 위한 언어이다. DTD는 XML 문법이 아닌 확장 배커스-나우어 형(EBNF ; Extended Backus-Naur Form) 문법 형식을 따르기 때문에 DTD 작성 방법을 별도로 익혀야 하는 번거로움이 있고, 한번 만들면 읽기만 가능하고 수정이 어려워 확장은 물론, 전체를 사용하는 경우가 아니면 재사용하기가 쉽지 않다. 또한, DTD는 제한적인 데이터 타입만을 지원하기 때문에 문자열 이외에 데이터 타입이 아닌 숫자, 날짜 구별이나 값의 범위, 패턴 등을 체크할 수 없다. 이와 같이 제한된 데이터 타입만으로는 문서의 내용을 좀 더 정확하게 표현하기 어렵다[10]~[12].

W3C의 XML 스키마 워킹 그룹은 개발자나 컨텐츠 설계자에 이르기까지 다양한 사용자의 요구를 만족할 수 있는 스키마 표준을 만들기 위해 노력했다. 이러한 결과가 바로 XML 스키마로 1999년 2월에 발표하여 2001년 5월에 정식 권고안이 되었다.

XML 스키마 워킹 그룹은 XML 문법으로 작성할 수 있는 DTD보다 강력한 표현이 가능한 스키마 언어를 만드는 것이 목적이다. 이러한 요구를 반영한 XML 스키마의 목표는 문서 구조와 내용을 제한하는 방법을 제공하고 DTD에 비해 문서 클래스에 대해 더욱 강력한 또는 느슨한 제한을 허용한다. 여러 개의 네임스페이스에 속한 생성 언어로 구성한 문서에 대해서도 유효성을 검증할 수 있는 기능은 XML 스키마의 가장 큰 장점 중 하나이다.

스키마에 요소 내용과 속성 값의 데이터 타입을 명시하는 XML 스키마 응용 프로그램은 유효성 검증 컴포넌트(component)를 사용하여 문서의 유효성을 보장할 뿐만 아니라 정보가 특정 데이터 타입에 일치하는지도 확인할 수 있다. 따라서 W3C XML 스키마를 따

르는 프로세서를 사용하면 데이터가 허용된 범위인지, 또는 올바른 형인지 검증하는 코드를 작성하지 않아도 된다.

XML 1.0에서 이용할 수 있는 데이터 타입은 <표 2-2>와 같다.

<표 2-2> XML 스키마 데이터 타입

<Table 2-2> Data Type of XML Schema

데이터 타입	설 명
string	문자열 형을 나타냄
entity	XML ENTITY 형을 나타냄
entities	XML ENTITIES 형을 나타냄
enumeration	(속성의) 열거형을 나타냄
id	XML ID 형을 나타냄
idref	XML IDREF 형을 나타냄
idrefs	XML IDREFS 형을 나타냄
nmtoken	XML NMTOKEN 형을 나타냄
nmtokens	XML NMTOKENS 형을 나타냄
notation	NOTATION 형을 나타냄

(1) XML 구문 사용

XML 스키마와 DTD의 가장 큰 차이는 XML 스키마가 XML 문법으로 표현된 것이다. W3C XML 스키마 명세는 XML 문서를 제한하기 위해서 사용할 수 있는 XML 어휘를 기술한다. 이는 XML 문서를 생성하고 조작하기 위해서 사용하는 모든 도구를 XML 스키마로 처리할 때도 이용할 수 있다는 것을 의미한다. XML 스키마에도 표준 XML 파서, 문서 객체 모델(DOM ; Document Object Model), XML을 위한 단순 응용 프로그램 인터페이스(SAX ;

Simple API for XML), XML 문서 스타일(style)로 변환하는 변환 언어(XSLT ; eXtensible Stylesheet Language Transformations) 등을 사용할 수 있다. 또한 표준 XML 기술을 사용하여 스키마를 생성하거나 변환할 수 있음을 의미한다.

(2) 내용 모델에 대한 더욱 강력한 지원

DTD 내용 모델은 빈약해서 문서를 간단한 연속이나 선택 목록으로 제한할 뿐이다. DTD는 혼합된 내용 모델의 유효성을 검증하기 위해서는 사용할 수 없고, 요소의 출현 횟수를 단지 0, 1, 또는 많음으로만 나타낼 수 있다. 그리고 내용 모델을 재사용할 수 있도록 하는 이름 있는 요소나 속성 그룹도 없다.

XML 스키마는 DTD에 비해 좀 더 자세하고 융통성 있는 내용 모델을 지원한다. XML 스키마는 혼합된 내용에 대해서도 유효성 검증을 할 수 있다. 즉, 출현 횟수의 정확한 수를 지정하거나 요소의 그룹에 이름을 부여할 수 있다. 또한, 텍스트를 표현하거나 데이터의 다른 형태를 표현할 수 있는 기능이 추가되어 있다.

(3) 확장 가능

XML의 완전한 이름에 대해 생각할 때, 명세를 통해 발표한 기술 언어(description language)를 확장할 수 없다는 것은 DTD의 약점이다. 그러나 W3C XML 스키마 명세는 다른 스키마 안에 있는 스키마의 일부를 재사용하거나 다른 스키마에서 재사용할 수 있는 복합 구조를 정의할 수 있는 능력, 새로운 데이터 타입을 파생할 수 있는 능력, 문서에서 다중 스키마를 참조할 수 있는 능력을 통하여 약점을 극복하고 있다.

이는 표준 어휘를 매우 쉽게 공유할 수 있음을 의미하고 모든 사람이 자신의 스키마를 만들지 않고 필요에 따라서 기존의 스키마를 맞출 수 있다는 것을 의미한다.

(4) 자기 문서화

XML 스키마 명세에 정의된 주석 요소를 사용하여 요소, 속성, 또는 스키마의 특정 부분의 용도를 기술할 수 있다. 모든 프로그래밍과 마찬가지로 주석을 사용하는 습관은 코드를 관리하기 쉽게 만드는데 좋다. 또한 스키마를 공유하고자 할 때 주석을 사용하여 원래 작성자의 의도를 명확히 알려줄 수 있다.

2.3.3 DTD와 스키마 비교

XML DTD와 XML 스키마의 차이점은 다음과 같다.

첫째, XML 문서의 구조를 정의하는데 있어서 XML DTD는 XML과 다른 문법을 사용한다. XML DTD는 간단하긴 하지만 XML 형식이 아니라 SGML의 DTD에서 일부분을 활용하는 형식이다. 반면에 XML 스키마는 그 자체가 XML 문서이면서 특정 XML 문서의 구조를 정의하는 규칙을 표현한다. 그러므로 문서 구조를 정의하는 부분과 실제 사용하는 부분들이 XML로 표현되게 되므로 XML 스키마와 그에 맞게 작성되어진 XML 문서들은 서로 잘 연결이 된다.

둘째, XML DTD는 사용할 수 있는 데이터 타입이 제한적이다. 숫자와 알파벳 문자를 구분하여 지정할 수 없을 뿐만 아니라 “나는 0부터 1200까지의 정수형만 쓴다”라고 사용자가 원하는 형태의 데이터 타입을 정의하여 사용할 수가 없다. 이는 데이터 타입과 값에 대한 제한을 주고 유효성을 검증하고 싶은 사용자에게 필요한 기능을

제공하지 못한다. 이러한 이유로 인해 데이터 타입에 대하여 강력하면서도 유연하고 재사용이 가능한 XML 스키마가 필요하다. XML 스키마에서는 총 45개의 기본적인 데이터 타입을 지원하고, 사용자 정의 데이터 타입을 선언할 수 있다.

셋째, XML DTD는 반복 횟수를 지정하기보다 반복 여부에 대한 결정 기능만을 제공한다. 반면에, XML 스키마는 정확하게 몇 번까지 반복하는지, 몇 번은 최소한 반복해야 하는지에 관한 정보를 지원할 수 있으므로 더욱 엄격하게 문서의 구조를 정의할 수 있다.

제 3 장 XML과 데이터베이스

3.1 데이터베이스

데이터베이스는 사람이 관심을 가지고 있는 데이터를 모아둔 것이다. 일상생활에서 자주 접할 수 있는 데이터베이스의 예로는 은행의 고객 자료의 데이터베이스나 학교에서 학생 자료 데이터베이스, 도서관에서 도서 데이터베이스 등이 있다. 이처럼 하나의 데이터베이스에는 서로 관련된 데이터들이 컴퓨터가 처리할 수 있는 형태로 저장되어 있다.

데이터베이스에 저장된 데이터는 사용자의 물음에 대해 대답할 수 있어야만 의미를 가질 수 있다. 예를 들면 도서관 데이터베이스에는 현재 도서관에 있는 도서들에 대한 데이터 즉 작가, 출판년도, 도서명 등의 데이터들이 저장되어 있으며 사용자는 자신이 찾고자 하는 도서에 관한 정보를 데이터베이스로부터 추출해 낼 수 있다. 즉 데이터베이스는 데이터를 보관하고 사용자의 물음에 대답하는 시스템이라고 할 수 있다.

데이터베이스에 저장된 데이터는 끊임없이 변한다. 예를 들면 도서관 데이터베이스는 새로운 도서가 들어올 때마다 새로운 도서에 대한 데이터와 도서관에 있는 도서의 대출과 반납에 대한 정보를 저장해야 한다. 데이터베이스는 새로운 데이터를 저장하거나 기존의 데이터를 삭제, 변경시키는 작업을 저장된 데이터가 일관성을 유지할 수 있도록 수행해야 한다. 데이터베이스를 만들거나 관리하고 데이터로부터 사용자의 물음에 대한 대답을 추출하는 프로그램을 데이터베이스 관리 시스템이라 한다.

데이터베이스 관리 시스템의 첫 번째 목표는 데이터를 저장하고 저장된 데이터로부터 유용한 정보를 얻어 내기 위한 효율적이면서도 편리한 방법을 사용자에게 제공하는 것이다. 데이터베이스 관리 시스템이 데이터를 관리하기 위해서는 저장할 데이터의 구조를 정의해야 하고 정의된 구조에 따라 효율적으로 데이터를 저장해야 하고 저장된 데이터로부터 빠르게 정보를 추출할 수 있는 방법을 제공해야 한다.

실제로 데이터를 관리하는 데 있어서 여러 문제점이 있다. 예를 들면, 학사 관리 데이터베이스에서 학생의 출생 년도가 1800년인 데이터가 있거나 학생의 소속 학과가 실제로 존재하지 않는 학과로 되어 있는 경우, 학번이 같은 사람이 둘 이상 있는 것과 같이 잘못된 데이터가 데이터베이스에 저장될 경우 이로부터 얻은 정보는 아무런 의미가 없다. 이와 같이 잘못된 데이터가 데이터베이스에 저장되는 것을 막는 것 또한 데이터베이스가 해야 할 일들 중 하나이다.

대부분의 경우 데이터베이스는 동시에 여러 사람들이 사용하게 된다. 데이터베이스 하나가 동시에 여러 장소에서 사용될 경우, 한 명의 사용자가 사용할 때와는 다른 문제점을 발생한다. 데이터베이스를 효율적으로 사용하기 위해서는 이처럼 동시에 데이터베이스에 접근하는 것을 가능하도록 해야 하며, 이 때 발생할 수 있는 문제를 데이터베이스 관리 시스템이 해결해야 한다.

3.2 XML과 데이터베이스 연구 현황

인터넷의 확산과 다양한 정보의 등장으로 인해 효율적으로 정보를 관리하고 저장할 수 있는 정보 시스템의 요구가 증가하고 있다.

대부분의 상업적인 데이터가 관계형 데이터베이스 시스템에 저장되어 있는 현재 이러한 요구를 만족시키기 위해서는 기존의 데이터베이스 관리 시스템과 연동하여 다양한 플랫폼과 데이터 종류에 관계없이 저장과 관리가 용이한 시스템의 구축이 필요하다[13],[14]. W3C에서 제안한 XML 표준안이 발표된 이후 인터넷 기반의 다양한 분야에서 XML 기술을 활용하고자 하는 움직임이 활발하다.

XML은 문서에 대한 구조 정보를 제공하고, XML 태그는 데이터를 해석하는데 사용될 수 있기 때문에, 데이터로서 XML 역할에 대한 중요성이 인식되어 왔다. 이에 따라 구조 정보를 내포하고 있는 데이터로서의 XML 문서를 효과적으로 관리하는 구조와 질의어 설계 및 처리에 대한 연구가 많이 진행되고 있다.

3.2.1 XML 문서 관리 및 질의

XML 문서를 관리하고 질의하는 방법으로 로렐(Lorel) 언어나 스토어드(STORED) 프로시저와 같이 기존의 XML 스키마에 의존하지 않고 반구조적 데이터에 대한 저장과 질의 처리 구조를 개발하는 방향과 XML 스키마를 기반으로 저장 구조를 설계하는 방향이 있다[15],[16]. DTD는 서로 다른 기종 간에 문서를 교환하는 과정에서 중요한 역할을 하기 때문에 DTD 기반의 XML 문서 관리 시스템에 관한 연구가 가장 활발히 진행되고 있다.

3.2.2 XML과 관계형 데이터베이스

XML 문서를 저장하고, 질의 처리를 하기 위한 하부 저장소로서 파일 시스템, 기존의 관계형 데이터베이스 관리 시스템(RDBMS ;

Relational DataBase Management System), 객체지향형 데이터베이스 관리 시스템(OODBMS ; Object-Oriented DataBase Management System), 그리고 반구조적 데이터에 대한 고유의 시스템을 바탕으로 설계될 수 있다. 이중에서 관계형 데이터베이스에서 DTD 기반의 XML 문서 저장 구조를 설계하기 위해서는 DTD를 데이터베이스 테이블로 매핑시켜야 한다.

관계형 모델을 사용하여 XML 데이터를 저장하는 방법은 세 가지로 나눌 수 있다. 첫째, DTD와 같은 기존의 스키마에 의존하지 않고, 반구조적 데이터를 분석해서 관계형 데이터 모델로 변환하는 방법이 있다. 이 시스템은 반구조적 데이터가 주어지면 데이터 마이닝(mining) 기술을 이용하여 관계형 모델로 자동으로 변환되어 저장된다. 둘째, DTD가 주어졌을 때, DTD를 바탕으로 하여 XML 요소와 관계 테이블 사이의 매핑이 자동으로 이루어진다. 셋째, Oracle 8i처럼 사용자나 시스템 관리자가 XML 문서를 관계형 테이블로 매핑하는 방법을 제공해 주어야 하는 경우가 있다.

XML DTD를 관계형 테이블로 매핑시키는 방법은 다음과 같다.

첫째, DTD에서 요소를 하나의 테이블로 매핑하고 요소의 속성을 테이블의 필드(field)로 두는 방법이 있다. 요소와 테이블간의 직접적인 매핑은 문서의 과도한 단편화를 초래하는 단점을 가지기도 한다.

둘째, 요소의 많은 자손들을 한 개의 테이블 필드로 고정하여 단편화 문제를 해결하는 방법이 있다. 이는 DTD에 나와 있는 모든 요소에 대해서 테이블이 만들어지고, 중복된 데이터가 저장된다.

셋째, DTD 그래프에서 자식이 없는 요소만을 하나의 테이블로 매핑하고, 그 테이블의 필드는 요소의 자손들로 둠으로서 요소와 테이블이 1:1로 매핑되는 결점을 보완하는 방법이 있다.

이러한 방법들에 대한 효율성을 검증하기 위해서 많은 실험이 이루어지고 있다[17].

3.2.3 XML과 객체지향형 데이터베이스

객체지향형 데이터베이스에 저장된 데이터를 XML 문서로 변환하기 위해 객체지향형 데이터베이스 스키마를 DTD로 변환하는 부분과 실제 저장된 데이터를 XML 문서로 변환하는 두 부분으로 나누어 스키마와 데이터, 데이터 변환의 결과가 되는 DTD와 XML 문서를 정의하고, 이에 따라 변환 알고리즘에 대한 연구가 진행되고 있다[18]~[20]. 그러나 대부분의 상업적인 데이터베이스가 관계형 데이터베이스이고 객체지향형 데이터베이스가 가지고 있는 단점을 극복하지 못하고 있어서 XML과 관련된 연구가 활발하게 진행되지 않는다.

3.3 XML 기반의 효율적 데이터 저장 관리

XML 문서의 효율적인 저장, 관리 및 다양한 검색을 지원하는 XML 저장 관리 시스템은 기존의 일반 문서 관리 시스템과는 달리 다음과 같은 특성을 고려하여야 한다. 논리적인 구조와 다양한 멀티미디어를 포함하는 XML 문서는 구조 정보를 이용하여 문서를 효율적으로 관리하므로 데이터베이스는 XML 문서, DTD, 구조 정보 및 다양한 미디어를 저장하고 관리해야 한다. XML은 문서에 대한 내용 검색, 요소가 갖는 속성에 대한 검색, 혼합 검색 등을 수용할 수 있다. 현재 구현된 대부분의 XML 저장 관리 시스템은 DBMS 기반의 시스템을 구현하고 있다.

기존의 데이터베이스를 서로 다른 운영 시스템(operation system)에 영향을 받지 않고 XML을 매개로 하여 저장 및 관리가

가능한 시스템으로 구축하고자 하는 연구가 진행되고 있다[21],[22].

분산 환경에서 각자의 고유한 데이터베이스 스키마를 가지고 운영되는 기존의 정보 시스템들을 XML이라는 공통 데이터 모델을 사용하여 표현함으로써 저장 및 관리상에 효율성 증대를 제공하고자 한다. 이러한 연구는 기존 데이터베이스 관리 시스템의 스키마 구조를 XML로 변환하는 방법과 변환된 XML 파일에서 기존의 데이터베이스 시스템처럼 조작 및 정의 기능이 가능하도록 하는 XML DTD에 유효한 정의가 필요하다.

B2B(Business to Business) 시대에는 XML 문서의 양도 많아지며, 이것을 저장, 관리할 XML 저장 관리 기술이 필요하고, 데이터베이스 관련 구조를 XML 문서로 변환하는 기술이 필요하다. 데이터베이스 관련 구조를 데이터 타입으로 저장하기 위해서는 XML 문서와 데이터베이스의 데이터를 매핑하는 것은 중요하다. XML 문서의 구조적 정보를 토대로 데이터베이스에 그 형태대로 저장하는 방식에 대한 연구가 진행되고 있다.

3.4 XML 접근 방식에 따른 DBMS 유형

XML 데이터를 데이터베이스 관리 시스템에 저장하는 가장 일반적인 접근 방법은 복잡한 프로그램 기술과 파서를 이용하여 XML의 요소들을 테이블과 칼럼(column)의 형태로 매핑하는 것이다. XML 데이터를 데이터베이스에 저장할 때마다 데이터베이스 관리 시스템은 XML 데이터를 분해해서 분해된 컴포넌트들을 각각 적절한 테이블에 저장하고, 질의할 때는 다시 분해된 컴포넌트들을 결합하여 XML 문서로 만들어낸다.

이 과정을 통해 나타난 결과가 XML을 처리하는데 큰 문제가 없어 보이지만, XML을 분해·변환하는 과정에서 XML 문서의 원래 구조를 잃어버리는 등 높은 신뢰성과 성능에 대해서는 보장할 수 없게 된다.

구조적 스키마를 가진 XML의 변환을 위한 대표적인 접근 방법으로 관계형 데이터베이스 매핑형, 객체지향형, 원시(native) XML형 데이터베이스의 특성은 다음과 같다.

3.4.1 관계형 데이터베이스 매핑형

관계형 데이터베이스는 역사가 오래 되었고 많은 응용 프로그램이 개발되어 있어 기존의 많은 데이터가 관계형 데이터베이스 형태로 유지 관리되고 있다. 현재에는 대부분의 많은 관계형 데이터베이스 시스템들 역시 XML 문서를 지원하기 시작하였으며 저장 방법과 관련된 연구가 진행되고 있다[23]~[25]. 그러나 관계형 데이터베이스로는 XML 문서를 표현하는 부분에 여러 가지 한계점이 따른다.

첫째, 모든 구조를 테이블로 표현해야 하는 관계형 데이터베이스에서는 XML을 분해·변환하는 과정에서 XML 문서의 원래 구조를 잃어버린다는 것이다. XML 문서에서 중요한 의미를 지니는 요소의 순서라든가, 속성의 정보라든가, 부모-자식 노드(node)의 관계 등에 대한 정보를 테이블로서는 구현하기가 매우 까다롭기 때문에 단순하게 요소나 속성 값만 저장하게 된다. 따라서 저장된 데이터를 다시 읽어 왔을 때 원래 문서의 구조가 그대로 복원되지 못할 가능성이 크다.

둘째, 계층적인 구조의 XML 데이터를 여러 테이블과 칼럼에 나누어서 저장해야 하는 문제점인데, 먼저 XML 데이터에 대한 질의를 할 때에는 여러 테이블과 관련된 복잡한 조인(join)이 필요하다.

XML 데이터를 검색하기 위한 평범한 질의를 위해서도 관계형 데이터베이스는 여러 번의 조인 질의를 해야 한다.

셋째, XML 데이터를 입력·수정·삭제하기 위해서는 해당 XML 문서를 다른 사용자가 접근하지 못하도록 하기 위해 관계된 모든 레코드에 락(lock)을 걸어야 한다. 레코드에 걸린 락 때문에 관계형 데이터베이스에서는 사용자에 대한 접근 관리 기능과 XML의 문서 집합, XML 문서, 요소에 대한 접근 관리 기능이 필요하지만 접근 제어 단위가 작으면 작을수록 관리 비용이 많이 들며, 접근 제어의 단위가 클수록 실제의 트랜잭션 처리 시간이 길어지는 경향이 있다. 또한 다수의 사용자와 트랜잭션 환경에서 복잡한 XML 데이터를 다루어야 한다면 이런 조인과 락은 데이터베이스의 성능을 현저히 떨어뜨릴 수도 있다.

결과적으로 XML의 계층 구조를 관계형 데이터베이스의 테이블로 표현하고, XML의 각 요소의 값과 속성을 테이블의 필드로 표현하게 된다. XML 문서 구조는 인덱스(index)를 가지고 있으나 저장 단위가 테이블이기 때문에 실제로 데이터를 매핑시키기 위해서 조인 연산이 필요하게 된다. 실제 상용 데이터베이스 관리 시스템에서는 기존 관계형 데이터베이스 관리 시스템에서 이를 위한 미들웨어를 제공한다.

이와 같이 프로그램에서 일일이 이러한 변환 작업을 하는 것은 매우 많은 노력을 필요로 한다. XML 데이터 구조가 복잡해질수록 데이터베이스 스키마 설계나 프로그램 구조는 복잡해지고, 관계형 데이터베이스에서 데이터 처리는 어려워진다. XML 데이터의 구조가 바뀐다면 그것이 단순히 태그 하나를 추가하는 것이라 할지라도 관계형 데이터베이스에는 매우 많은 변경을 일으킬 수도 있다. 또한 XML 스키마가 바뀐 형태에 따라서 새롭게 정규화를 해야 하는 복잡한 과정을 거쳐야 한다.

XML을 저장하기 위한 또 다른 방법으로 테이블의 매핑 작업을 하지 않으려면 XML 데이터를 블랍(BLOB ; Binary Large Object)으로 저장하는 방법이 있다. 이렇게 하면 XML 문서는 요소 레벨이 아니라 문서 전체 단위로 검색·입력·수정·삭제가 이루어지게 된다. 그러나 XML을 BLOB으로 저장하게 되면 검색을 위한 인덱스를 효율적으로 만들 수 없어 XML 문서의 계층적 구조에 따른 검색이나 요소, 속성별로 조건을 주는 검색이 어렵다. 쉬운 방법으로 매핑 가능한 데이터는 테이블에 저장하고, 복잡한 구조의 데이터는 BLOB으로 저장할 수도 있겠지만 그렇게 하면 테이블과 BLOB으로 나누어 저장된 데이터를 다시 결합하는 일 또한 중간 과정이 많이 필요로 하게 된다.

3.4.2 객체지향형

XML 데이터의 계층 구조를 객체 지향의 클래스(class) 계층에 매핑시키는 타입으로 초기의 관계형 데이터베이스에서는 데이터의 자료 구조와 응용 프로그램의 자료 구조가 다르므로 중간에 상이한 두 자료 구조간의 데이터 변환을 응용 프로그램에서 해주지만 객체지향형 데이터베이스는 응용 프로그램과 데이터베이스의 자료 구조가 동일하므로 데이터 변환과 같은 부가적인 작업이 없다.

객체지향형 데이터베이스에 XML 문서를 저장할 경우 문서의 요소를 객체로 생각하여 비교적 쉽게 모델링할 수 있고, XML 문서와 DTD의 복잡한 구조 역시 객체지향형 데이터베이스에서는 자연스럽게 유지될 수 있다. 객체지향형 데이터베이스의 스키마와 스키마 정보는 데이터베이스 이름과 클래스들의 스키마 정보로 이루어진다. 그리고 각 클래스는 클래스 이름과 속성들의 정보로 구성되며 각 속성은 이름과 자신의 형태 정보를 가진다. 즉 XML의 데이터 모델

이 IS-A 관계나 HAS-A 관계를 이용한 객체지향의 데이터모델과 유사하므로 거의 모든 XML 데이터 구조를 저장할 수 있는 것이다.

또한 W3C에서 응용 프로그램에 활용하기 위한 XML 응용 프로그램 인터페이스(API ; Application Program Interface)의 표준으로서 권고하고 있는 DOM이 객체를 기반으로 하고 있어서 객체지향형 데이터베이스로의 구현은 더욱 간단하다. 그러나 객체지향 데이터베이스는 대량의 문서가 저장될 경우, 즉 객체가 많아질 경우 저장과 검색 효율이 크게 떨어진다. 대부분의 객체지향형 데이터베이스는 높은 관심과 주목을 받았지만 상업적으로는 크게 성공을 거두지는 못하고 있는 실정이다. 하지만 객체지향형 데이터베이스의 개념은 원시 XML 데이터베이스의 모태가 된다.

3.4.3 원시 XML형

원시 XML 데이터베이스는 XML에 대한 논리적 구조 형태로 저장 및 검색이 이루어지는 구조인데 XML 데이터를 입력받아 저장하고, 질의 결과를 XML로 출력하기에는 가장 적합하게 설계되어 있다. XML 구조에 따른 검색 지원, 부분 범위 추출 등 XML의 특성에 맞는 검색이 이루어질 수 있다.

XML 데이터베이스에는 기존 관계형 데이터베이스에 XML 기능을 추가시킨 XML-Enabled 데이터베이스들은 일반 데이터베이스에 XML 데이터를 테이블과 컬럼으로 매핑하여 저장하고, 질의 결과를 다시 XML로 바꾸어 돌려주는 컨버터(converter) 기능을 추가하여 XML을 지원하는 형태로 발전해 나가고 있다.

반면에, 원시 XML 데이터베이스는 XML 데이터를 있는 그대로 저장하기 때문에 컨버터 기능이 필요 없으므로 데이터베이스 관리 시스템 기능으로 XML을 다루기에 적합하다.

XML의 계층 구조를 계층형 데이터베이스 구조로 관리하며 요소의 데이터나 속성을 하나의 XML 데이터로 취급하므로 문서 관리에 적합하며 구조가 정의되어 있는 XML 데이터에 인덱스를 가지므로 보다 고속처리를 할 수 있다.

일반적으로 원시 XML 데이터베이스의 장점은 다음과 같다.

첫째, 데이터베이스 설계 및 프로그래밍 유지 보수가 용이하다. XML 데이터를 관계형 데이터베이스의 테이블에 저장하기 위해서는 정규화 과정을 거쳐 고유 키(primary-key)와 외래 키(foreign-key)로 복잡하게 연결된 여러 개의 테이블을 설계해야 한다.

부모/자식의 관계를 관계형 데이터베이스의 테이블에 맞도록 설계하기가 어렵다. 만약 계층적 구조의 관계를 관계형 데이터베이스에 매핑하기 위해서는 복잡한 구조의 테이블을 설계해야 한다. 중첩되는 구조의 XML 데이터의 경우 관계형 데이터베이스에 매핑이 어려운 경우도 있고, 또 XML의 구조가 바뀌면 데이터베이스 설계를 다시 해야 한다.

원시 XML 데이터베이스는 별도의 정규화 과정을 거치는 테이블 설계가 필요 없기 때문에 설계가 쉬우며 XML 문서의 구조가 바뀌어도 데이터베이스 스키마의 변경이 쉬워 유지 보수가 용이하다.

둘째, XML 데이터의 저장 및 질의 성능을 향상시킬 수 있는데, XML-Enabled 데이터베이스들은 계층적 구조의 XML 문서를 여러 개의 테이블과 칼럼에 걸쳐 쪼개어 저장하고, 질의할 때 다시 이를 XML로 재구성해야 하므로 XML의 구조가 복잡할수록 복잡한 조인 질의를 해야 하기 때문에 검색의 속도가 떨어진다.

XML 데이터의 각 요소, 속성 값을 테이블의 칼럼에 쪼개어 저장하기 때문에 XML 문서의 원래 구조를 잃어버리는 등의 단점이 있다. 반면 원시 XML 데이터베이스 관리 시스템은 XML 데이터를 있는 그대로 저장하기 때문에 변환 과정이 필요 없어 복잡한 구조의

XML 문서도 빠른 속도로 처리 가능하며, XML 문서의 구조를 잃지 않고 저장한다.

셋째, XML 구조에 맞는 검색이 이루어지는데 XML-Enabled 데이터베이스는 XML을 질의하기 위해서는 복잡한 질의 문장을 만들어야 한다. 원시 XML 데이터베이스는 Xpath(XML Path Language) 표준을 이용하여 질의하기 때문에 XML 구조에 맞는 검색을 간단한 질의 문장으로 수행할 수 있다.

넷째, XML을 관계형 데이터베이스에 저장할 때는 XML의 구조 정보인 각 요소나 속성 값들만 골라내어 저장하기 때문에 부모/자식 관계, 요소의 순서, 속성 등 XML 문서의 구조를 잃어버리게 된다. 원시 XML 데이터베이스는 XML의 구조를 깨지 않고 그대로 저장하므로 저장된 XML 문서를 질의했을 때 문서의 구조를 그대로 복원할 수 있다.

3.4.4 XML과 데이터베이스 상호 매핑

XML이 단순한 컨텐츠에서 데이터베이스로까지 그 적용 분야가 확장되면서 XML로 표현된 정보들을 어떻게 효율적으로 저장하고 관리할 것인지에 대한 기술도 XML과 함께 발전되어 가고 있다.

가장 큰 이슈 중의 하나가 기존의 데이터베이스 관리 시스템으로도 XML을 효율적으로 관리할 수 있는가이다. 여러 가지 측면에서 다양한 장단점들이 존재하기 때문에 이를 위해 시스템을 설계하는 것은 사실상 어려운 일이다.

XML을 가장 효율적으로 저장하고 검색하며 변경, 관리할 수 있는 데이터베이스 관리 시스템이 갖추어야 할 조건은 XML을 검색할 수 있는 표준인 XPath나 XML 질의를 인터페이스로 제공해야 하고

실제 내부적으로 인덱스 기술 및 객체 저장 관리를 효율적으로 지원해야 한다. 특히, XML 문서의 검색은 DOM에서 XPath를 기반으로 문서의 특정 위치에 접근하는 과정에서 조건을 설정하는 형식이므로 기존의 2차원 평면적인 테이블의 컬럼에 인덱스를 구성하는 관계형 데이터베이스와는 특성상 큰 차이점을 가지고 있다[26].

정보의 관리 측면에서 XML 적용을 확산시키기 위해서는 관계형 데이터베이스에서 표준적으로 지원하는 구조적 질의어(SQL ; Structured Query Language)와 같은 질의어에 대한 표준이 제공자마다 다르기 때문에 완전한 통합은 어렵다. 현재 방대한 양의 XML 문서를 저장 관리하거나 또는 다른 곳에 저장된 XML 문서를 가져와야 한다.

XML은 문서 교환에 있어서 현재 초기 단계이지만 앞으로 XML 문서의 양도 증가하므로 이를 저장 관리할 XML 저장 관리 기술이 요구되고 있다. 그러므로 대용량의 XML 문서를 저장, 검색, 관리하기 위해서 XML의 특성에 맞는 저장 관리 시스템 개발이 지속적으로 발전해야 한다.

가장 많이 사용되고 있는 관계형 데이터베이스에 저장하는 방식은 현재 관계형 데이터베이스 시장이 다른 데이터베이스 관리 시스템에 비해 넓고 그에 따라 사용자 수도 많기 때문에 확산이 빠르며 쉽게 상용화할 수 있다는 장점이 있다. 현재 Oracle, Microsoft SQL 등 기존의 상용 관계형 데이터베이스가 XML을 지원하기 위해 많은 노력을 기울이고 있다.

관계형 데이터베이스에서 XML 문서 구조에 대한 구조 정보를 저장하기 위해서는 많은 테이블이 필요하고 이에 따른 조인 연산으로 시스템의 성능이 저하된다는 문제점을 안고 있다[27].

XML은 XML 문서, DTD, 스키마 등과 같은 스키마, XQuery (XML Query), XPath, XQL(XML Query Language)과 같은 질의

언어, SAX, DOM 등과 같은 프로그래밍 인터페이스를 제공하고 있다. 그러나 효율적인 저장, 인덱스, 보안, 트랜잭션 및 데이터 무결성, 다중-사용자 연결, 여러 문서에 걸친 질의 등과 같은 실제 데이터베이스에서 나타나는 많은 장점들을 활용하기에는 XML로는 부족한 점이 많다. 따라서 데이터의 양이 적고, 사용자가 적고, 적당한 성능을 요구하는 환경에서는 일종의 데이터베이스로서 XML 도큐먼트(document)를 사용하는 것이 가능하지만, 많은 사용자, 엄격한 데이터 무결성 요구, 우수한 성능이 필요한 환경에서 사용하기에는 많은 어려움이 발생된다.

이를 위해 XML 저장 방법에 대한 연구, XML 인덱스 지원에 대한 연구, XML 질의 처리 모듈과 관계형 데이터베이스 질의 모듈과의 효율적 통합에 대한 연구가 많이 진행되고 최근 이슈화 되고 있다. 그 중 근본적으로 XML과 관계형 데이터베이스간의 데이터 저장을 위한 매핑에는 여러 가지 해결해야 할 점이 있다.

저장 방식을 객체지향형 데이터베이스 관리 시스템으로 접근하는 방법은 XML 구조를 객체의 기존 개념에 기반을 두고 있어 XML을 저장 관리하는데 적합하다고 할 수 있다. 이러한 것이 가능한 구체적인 이유로는, 객체지향형 데이터베이스 관리 시스템을 사용하여 스키마를 설계할 때 사용되는 클래스를 이용하여 각 요소들의 구조를 정의할 수 있으며, 요소 간의 관계를 객체 간의 연결 관계로 나타낼 수 있기 때문에 XML 문서를 모델링하는데 적합하다[28],[29].

XML을 객체지향형 데이터베이스 관리 시스템에 저장하는 기술은 현재 컨텐츠 관리 시스템과 결합하여 많이 발전되어 새로운 시스템인 XML 리포지터리 시스템(repository system)이라는 새로운 개념을 만들었다. 기존의 객체지향형 데이터베이스 관리 시스템 제공자들은 XML 기능을 특화하여 XML 전용 데이터베이스 관리 시스템으로 발전하고 있다.

3.5 데이터와 도큐먼트

다양한 분야에서 XML 문서가 급격히 증가하고 있고, 이를 영구적으로 데이터베이스에 저장해야 할 필요성도 절실하다. XML 문서는 계층 구조를 가지고 있는 반면, 전통적인 관계형 데이터베이스는 2차원 평면적인 구조를 가지고 있으므로 기존의 데이터베이스 구조를 XML 문서에 맞도록 확장할 경우가 많다. XML에서 데이터베이스에 저장되는 요소를 구분해 보면 데이터를 저장하기 위해 데이터베이스를 사용하는지 또는 도큐먼트를 저장하기 위해 데이터베이스를 사용하는지에 따라 나눌 수 있다.

XML 문서의 형태를 유지하면서 저장·관리하는 방식으로는 파일 시스템에 저장하거나 데이터베이스 시스템에 BLOB 형태로 저장하는 방식은 도큐먼트-중심의 도큐먼트를 저장할 때 유용하다고 할 수 있다. 이 방식을 통해서 문서를 저장하거나 가져올 때 빠르다는 장점은 있지만, 문서의 구조에 접근하기 위해서는 매번 분석하기 때문에 대용량 문서를 다루기에는 적합하지 않다.

3.5.1 데이터-중심 도큐먼트

데이터-중심 도큐먼트는 XML을 사용하여 데이터 전송을 하는 문서 중에서 컴퓨터 관점에서 사용하기 편리하게 설계된 XML 문서를 특별히 데이터-중심 도큐먼트라고 표현한다. 컴퓨터 관점에서 설계된 데이터라는 것은 사람이 특별히 구분하지 않아도 요소나 속성 내용을 추출함으로써 XML DTD를 이용하여 데이터의 구조 정보를 파악하고 필요한 데이터를 XML에서 DOM에 의해 생성된 노드 트리의 내용을 논리적이고 일관되게 표현한다.

데이터-중심 도큐먼트는 정형화된 구조를 가지고 최소의 독립적인 데이터 단위가 PCDATA 전용 요소 또는 속성 레벨로 선언되어 있으며, 관계형 데이터베이스에서 레코드 단위로 표현할 수 있는 형제 요소는 중첩되게 표현되므로 순서에 관계없이 저장될 수 있다.

데이터-중심 도큐먼트에서 사용되는 데이터는 데이터베이스에 저장되어 있는 데이터를 XML로 나타내거나, 또는 XML을 데이터베이스로 저장하기에 적합한 문서 구조를 가지고 있다. 데이터-중심 도큐먼트의 예로서 증권시스템, 통계시스템 등을 들 수 있다. 또한 데이터-중심 도큐먼트는 요소와 속성이 정형화된 구조로 중첩되게 표현된다. 특히 데이터-중심 도큐먼트는 컴퓨터 연산에서 활용도를 높일 수 있기 때문에 하나의 저장소로서 XML을 활용하는 것이다.

웹은 XML 문서 또는 데이터베이스에서 추출된 데이터-중심 도큐먼트를 XSL 스타일시트로 표현하고 있다. 이는 HTML 문서를 동적으로 생성할 수 있는 웹 사이트에서 데이터베이스에서 추출된 데이터 템플릿을 데이터-중심 도큐먼트와 하나 이상의 XSL 스타일시트로 표현하여 웹 브라우저를 통해 볼 수 있다.

XML에서 분리된 데이터와 구조의 특성을 활용한 추출된 XML 데이터의 재사용은 문서를 여러 형태로 보여줄 수 있어 데이터-중심 도큐먼트 형태에 적합한 XML 데이터 추출 및 포맷팅으로 다양한 응용에 사용되고 있다.

3.5.2 도큐먼트-중심 도큐먼트

도큐먼트-중심 도큐먼트는 사람의 입장에서 사용하기 편리하게 설계된 도큐먼트이다. 예를 들면 책, 이메일, 광고, 그리고 대부분의 정형화되지 않은 도큐먼트이다.

도큐먼트-중심 도큐먼트들은 비정형적인 구조, 독립적이고 적은 규모 단위의 데이터가 아닌 여러 가지 혼합된 컨텐츠를 가지며, 데이터 규모도 상대적으로 많은 요소 레벨 또는 전체 도큐먼트 자체를 표현한 것으로 이때 데이터-중심 도큐먼트와는 달리 형제 요소는 중첩되었을 때 의미가 바뀔 수 있으므로 순서에 영향을 받는다.

도큐먼트-중심 도큐먼트는 일반적으로 XML 또는 나중에 XML로 변환될 RTF(Rich Text Format), PDF(Portable Document Format) 또는 SGML 등과 같은 다른 형식으로 작성될 수 있는데, 데이터-중심 도큐먼트와는 달리 도큐먼트-중심 도큐먼트는 일반적으로 데이터베이스로부터 생성되지 않는다. 이것은 XML 데이터 모델은 비정형화된 컨텐츠 및 정형화된 데이터를 모두 포괄하기 때문에 도큐먼트-중심 및 데이터-중심 도큐먼트에 액세스하고 SQL을 통해 질의할 수 있는 XML 리포지터리를 제공해야 한다.

3.5.3 데이터, 도큐먼트, 데이터베이스

실제로 데이터-중심 도큐먼트와 도큐먼트-중심 도큐먼트를 명확하게 구분하기란 쉽지 않은데, XML에서 선택적으로 DOM을 추출할 경우 추출된 데이터가 데이터-중심 도큐먼트가 될 수 있고 도큐먼트-중심 도큐먼트가 될 수 있기 때문이다. 도큐먼트의 형태에 있어서 정형화된 구조를 가지는 도큐먼트와 비정형화 구조를 가진 도큐먼트의 차이를 명확하게 구분하기란 어렵지만 이러한 점에도 불구하고 도큐먼트에 대해 데이터-중심적인지 또는 도큐먼트-중심적인지 특징짓는 것은 사용할 데이터베이스의 종류가 무엇인지 결정하는데 도움이 된다.

일반적인 규칙으로 데이터는 관계형, 객체지향형, 계층적 데이터베이스 등과 같은 전통적인 데이터베이스에 저장되는데 이는 미들웨어 또는 데이터베이스 자체적으로 구축된 기능에 의해 수행된다.

도큐먼트는 데이터베이스나 컨텐츠 관리 시스템(CMS ; Content Management System)에 저장될 수 있다. 특히 비정형화된 데이터는 원시 XML 데이터베이스 내에 저장될 수 있고, 도큐먼트는 몇몇 XML 관련 기능이 필요할 때 전통적인 데이터베이스에 저장된다.

일반적으로 XML을 지원하는 데이터베이스 관리 시스템은 XML 문서와 데이터 전송을 위한 확장 모듈을 가지고 있으며 데이터-중심 도큐먼트를 저장할 때에는 테이블에 저장하게 하고 도큐먼트-중심 도큐먼트를 저장할 때에는 한 필드에 저장되도록 하고 있다.

3.6 XML 미들웨어

XML 미들웨어는 XML 도큐먼트와 데이터베이스 간에 데이터-중심 도큐먼트를 위한 데이터 전송 소프트웨어를 총칭하는 것으로 여러 구조를 가진 다양한 언어로 구현되지만 대부분은 관계형 데이터베이스에서 사용하기 적합한 ODBC(Open Database Connectivity), JDBC(Java Database Connectivity), OLE(Object Linking and Embedding) 데이터베이스를 사용하여 구성된다. 대표적인 미들웨어로는 ASP2XML, XML-DBMS, DB2XML이 있다[30]~[33].

첫째, ASP2XML 컴포넌트는 데이터베이스와 XML 사이에 변환을 위한 쉬운 인터페이스를 제공하는데, 주요 메소드로는 데이터로부터 XML 스트림을 추출하는 getXML과 생성된 XML을 수정하는 putXML 메소드로 구성된다. ASP2XML 컴포넌트는 ODBC 또는

OLE 데이터베이스를 사용하며 XML로 변환할 때 성능 향상을 위해서 인덱스 키 값을 사용하여 키 값 중심으로 요소를 구성하게 된다. 또한 변환할 때 ASP2XML에서 태그 부여 규칙을 정해서 원시 데이터와 변환된 XML 문서 사이에 차이를 줄인다.

둘째, XML-DBMS는 XML 도큐먼트와 관계형 데이터베이스 간의 변환을 위한 미들웨어로서 객체-관계형 매핑을 사용한 자바 패키지로, 사용자가 작성하는 응용 프로그램에서 호출하여 사용할 수 있다. 또한 XML 문서 구조와 객체-관계형 매핑을 기술하기 위해서 XML 기반 매핑 언어를 사용하며, DTD나 데이터베이스 스키마로부터 자동으로 매핑하거나 사용자가 직접 정의할 수 있다. 이 언어는 XML 기반의 언어로서 XML 문서의 태그와 테이블의 필드 관계를 기술하게 된다.

셋째, DB2XML은 관계형 데이터베이스로부터 데이터를 XML 문서로 변환하는 자바 클래스로서 데이터베이스에서 검색한 질의 결과에 해당되는 XML 또는 HTML 문서로 변환해주며 메타 데이터 기술, XSLT 스타일시트 변환 기능이 있다. DB2XML 클래스는 자바 응용 프로그램이나 서블릿(servlet)에서 사용하고 옵션 선택을 통해 XML의 태그 이름을 지정한다. 또한 도큐먼트의 메타 데이터를 파일이나 스트림 또는 DOM 객체로 생성하여 XSL 프로세서에 제공한다.

DB2XML은 기존 시스템에 저장된 관계형 데이터를 XML로 자동 변환시켜 관계형 데이터베이스에 저장된 데이터에 대해서 XML 문서와 DTD를 제공하는데 관계형 모델의 각 구성 요소에 대한 매핑 규칙을 제시하여 그 규칙에 의한 데이터 변환을 제시한다.

제 4 장 스킴

4.1 스킴 등장 배경

LISP(LISt Processing) 언어는 1950년대 후반 순환 방정식으로 알려진 논리식의 정형적 추론을 위한 수단으로 발명되었다. 이 언어는 존 매카시(John McCarthy, 미국)가 개발하였으며, “기호 식의 순환적 함수와 머신에 의한 계산”이라는 그의 논문에 기초하고 있다. 수학적 형식주의에서 출발하였음에도 불구하고 LISP은 실용적인 프로그래밍 언어이다[34].

LISP 인터프리터는 LISP 언어로 표현된 계산 과정을 실행하는 머신이다. 최초 LISP 인터프리터는 매카시가 MIT(Massachusetts Institute of Technology)의 전자공학 실험실의 인공지능 그룹과 계산 센터에 소속된 동료들과 학생들의 도움을 받아 개발하였다. LISP 1 사용 방법은 1960년에 처음 알려졌으며, LISP 1.5 사용 방법은 1962년에 출판되었다.

LISP은 대수식의 미분이나 적분을 구하는 것과 같은 프로그래밍 문제에 활용될 수 있도록 기호 처리 기능을 강조하여 설계된 것이다. 아톰(atom)이나 리스트(list)와 같은 LISP의 기호 처리용 객체들은 당시의 다른 모든 언어들과 LISP을 크게 구별하는 특징이다. LISP은 여러 사람들의 협력으로 제품화된 것이 아니며, 사용자의 요구가 있거나 효과적으로 구현할 수 있는 기능 위주로 점차 확장되는 실험적이고도 비공식적인 방법으로 발전했다. 이러한 과정이 오랜 기간 계속된 결과, LISP 사용자는 언어 기능을 확정하여 널리 공식적으로 홍보하는 것을 꺼리게 되었다.

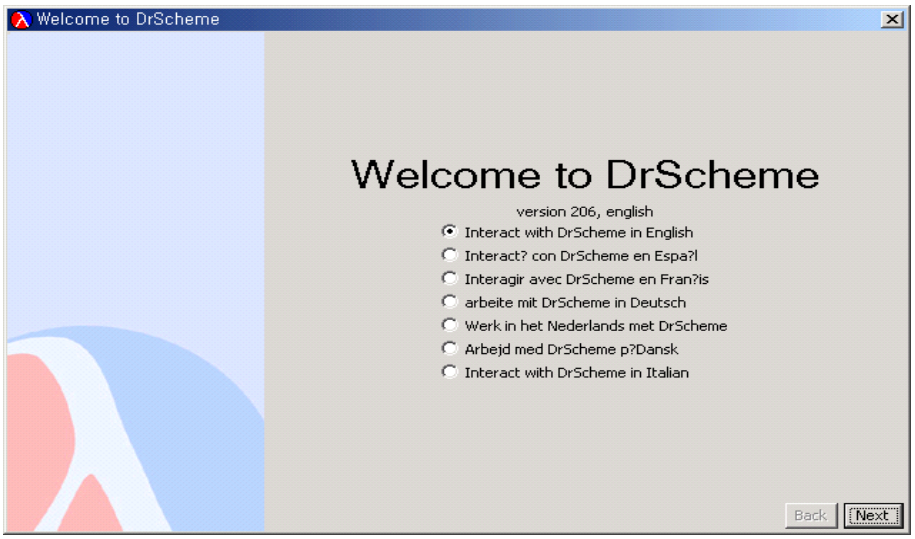
기본 개념이 유연하고 우아하며, 실험 정신으로 발전해 온 결과 LISP은 오늘날까지 널리 사용되는 두 번째로 오래된 언어임에도 불구하고 가장 최신의 프로그램 설계 기법도 수용할 정도로 꾸준히 적응, 발전하고 있다. 현재 LISP은 여러 방언들이 있으며, 비록 근본적인 기능 대부분을 공유하기는 하지만 각 방언들은 서로 크게 다르다.

LISP의 방언 중 하나인 스킴(Scheme)은 MIT 인공지능 실험실의 가이 루이스 스틸(Guy Lewis Steele, 미국)과 제럴드 제이 서스만(Gerald Jay Sussman, 미국)에 의해 1975년에 발명되고, MIT에서 학생들의 교육에 사용되도록 뒤에 다시 구현된 것이다. 1990년에 미국 전기 전자 학회(IEEE ; Institute of Electrical and Electronics Engineers)에서 스킴은 표준(IEEE 1990)으로 확정되었다.

4.2 Dr.Scheme

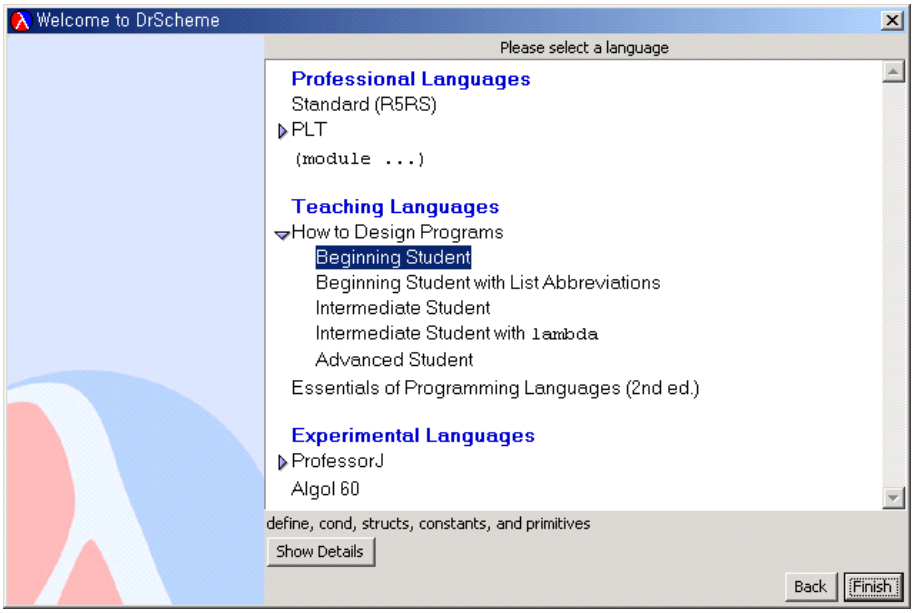
Dr.Scheme은 스킴 언어를 실습할 수 있는 툴이다. 실행 파일은 웹 사이트 http://download.plt-scheme.org/drscheme/에서 다운 받아 설치할 수 있다[35]. 파일을 설치한 뒤 실행시킬 경우 처음 나오는 화면은 <그림 4-1>과 같고, Dr.Scheme에서 사용할 언어는 한글을 지원하지 않기 때문에 언어 선택 창에서 영어를 선택한다.

언어를 선택하고 Next 버튼을 누르면 Dr.Scheme은 언어 레벨을 선택할 수 있는 창이 나타나고, Dr.Scheme의 기본 언어 레벨은 초보자를 위한 Beginning Student 레벨이다. 언어 레벨을 선택할 수 있는 화면은 <그림 4-2>와 같다.



<그림 4-1> 언어 선택 창

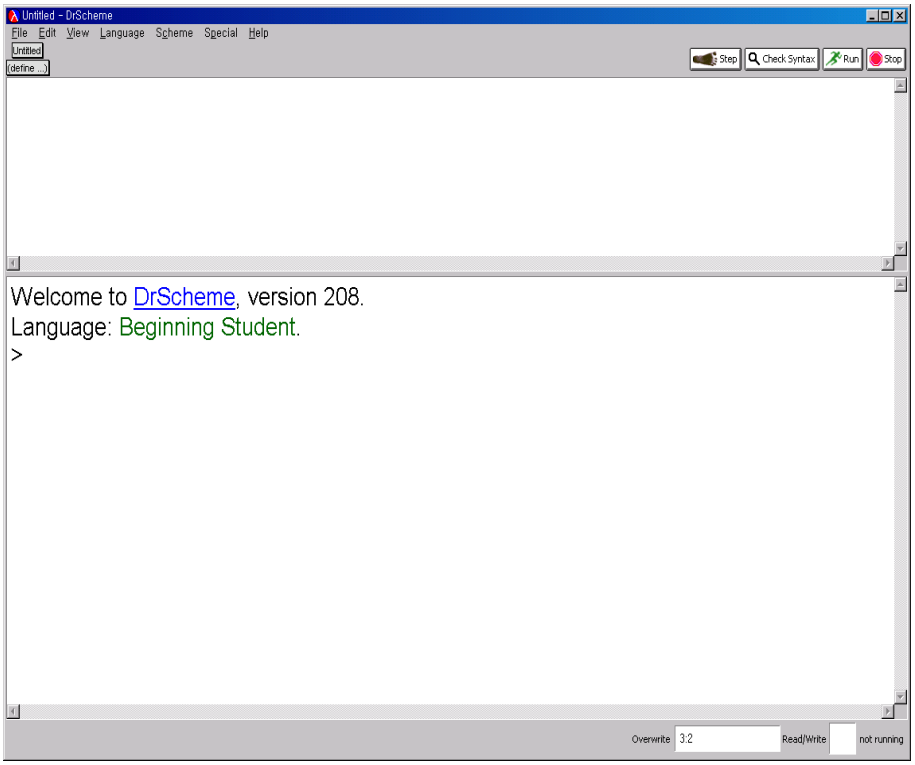
<Fig. 4-1> Window of Language Select



<그림 4-2> Dr.Scheme 언어 레벨 선택 창

<Fig. 4-2> Window of Language Level on Dr.Scheme

언어 레벨을 Beginning Student로 선택하고 Finish 버튼을 누르면 나타나는 Dr.Scheme의 실행 화면에서 F5 또는 Run 버튼을 누르면 <그림 4-3>과 같다.



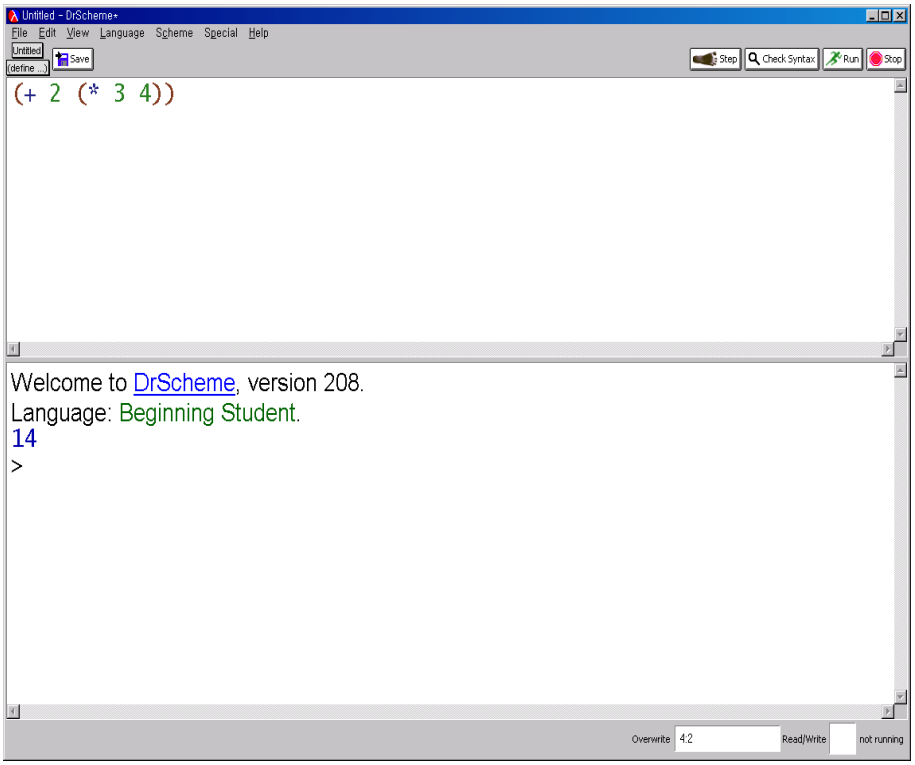
<그림 4-3> Dr.Scheme 실행 화면

<Fig. 4-3> Execute Window of Dr.Scheme

Dr.Scheme의 실행 화면을 보면 Step, Check Syntax, Run, Stop 버튼이 오른쪽 상단에 있고 창이 두 개로 나누어져 있다. 위의 창을 정의창이라 하고 아래쪽에 있는 창을 실행창이라 한다. 정의창은 스킴 언어를 이용하여 프로시저를 정의하는 창이고, 실행창은 정의창에서 정의된 프로시저를 이용하여 계산한 결과를 확인할 수 있는 창이다.

(1) Step 버튼

주어진 식의 계산 과정을 단계별로 보여 준다. 예를 들면 (2 + (3 * 4))을 계산하는 과정을 단계별로 보여준다. 스킴은 식을 표현할 때 전위법을 사용하므로 위의 식을 (+ 2 (* 3 4))로 바꾸어 정의창에 식을 입력하는 화면은 <그림 4-4>와 같다.

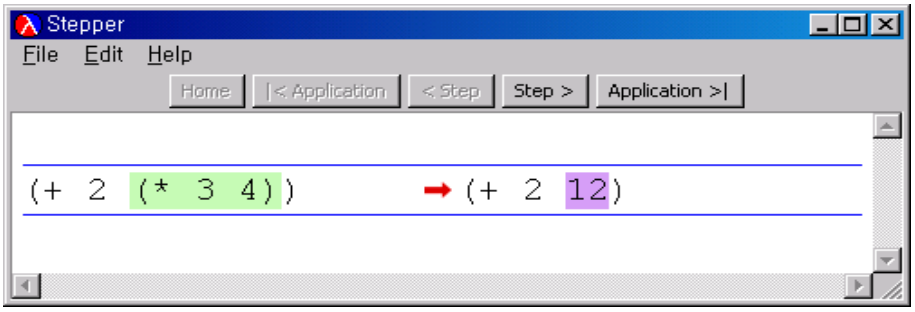


<그림 4-4> (+ 2 (* 3 4)) 실행 화면

<Fig. 4-4> Execute Window of Expression (+ 2 (* 3 4))

Step 버튼을 누르면 (* 3 4)를 먼저 계산하는 과정을 보여주는 Stepper 창이 나타난다. 나타난 창에서 Next 버튼을 누르면 (* 3 4)를 계산한 결과와 2를 더하는 계산 결과가 14라는 것을 보여준다.

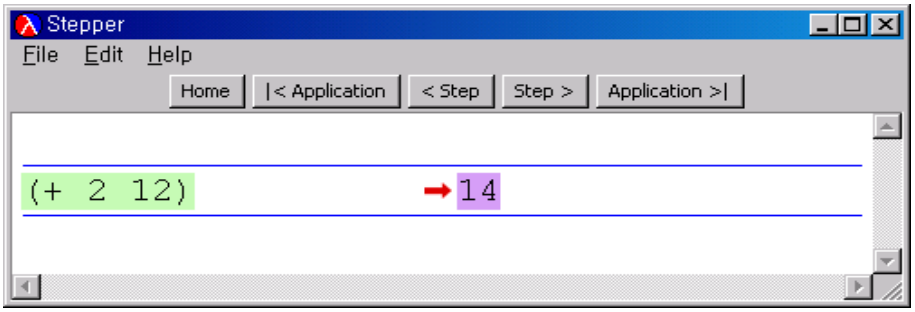
<그림 4-4>에서 Step 버튼을 누르면 <그림 4-5>와 같은 주어진 식에서 제일 먼저 계산되는 부분을 보여준다.



<그림 4-5> 첫 번째 계산 과정

<Fig. 4-5> The Fist Step of Expression

<그림 4-5>에서 Step 버튼을 누르면 두 번째 계산 과정을 보여주는 Stepper 화면이 나타나며 <그림 4-6>과 같다. <그림 4-6>은 주어진 식의 결과를 보여주므로 계산 과정의 마지막 단계이다.



<그림 4-6> 두 번째 계산 과정

<Fig. 4-6> The Second Step of Expression

(2) Check Syntax 버튼

정의창에 입력된 식이 스킴의 문법에 맞는지 검사하는 버튼이다.

Check Syntax 버튼을 누르면 입력된 식이 Dr.Scheme의 환경 설정에 따라 색깔이 바뀌게 된다. 또한, 마우스를 임의의 값에 갖다 놓으면 같은 값을 가리키는 것을 화살표로 표시해 준다.

(3) Run 버튼

Run은 실행 버튼으로 정의창에 입력한 식을 해석한다. 정의창에 입력된 식을 해석해야만 실행창을 통해 계산한 결과를 묻고 답할 수 있다.

(4) Stop 버튼

Stop 버튼은 Run 버튼을 눌렀을 때 무한 반복을 하거나 사용자가 도중에 실행을 멈추고 싶을 때 누르는 버튼이다.

4.3 스킴의 기본 문법

4.3.1 프로시저와 기본 문법

스킴은 기본식과 복합식 그리고 추상화 수단을 이용하여 사용자가 원하는 기능을 하는 프로그램을 작성한다[36].

기본식은 단순한 개체를 표현하기 위한 것으로 수와 수를 계산하기 위해 사용되는 +, -, *, /와 같은 원시 프로시저가 기본식의 범위에 속한다. 복합식은 수를 나타내는 식과 원시 프로시저를 결합하여 만든 복잡한 식을 의미한다. 예를 들면 (+ 2 (* 3 4))와 같은 형태이다. 복합식을 만들 때 스킴은 전위법을 사용하여 식을 표현한다.

(1) define

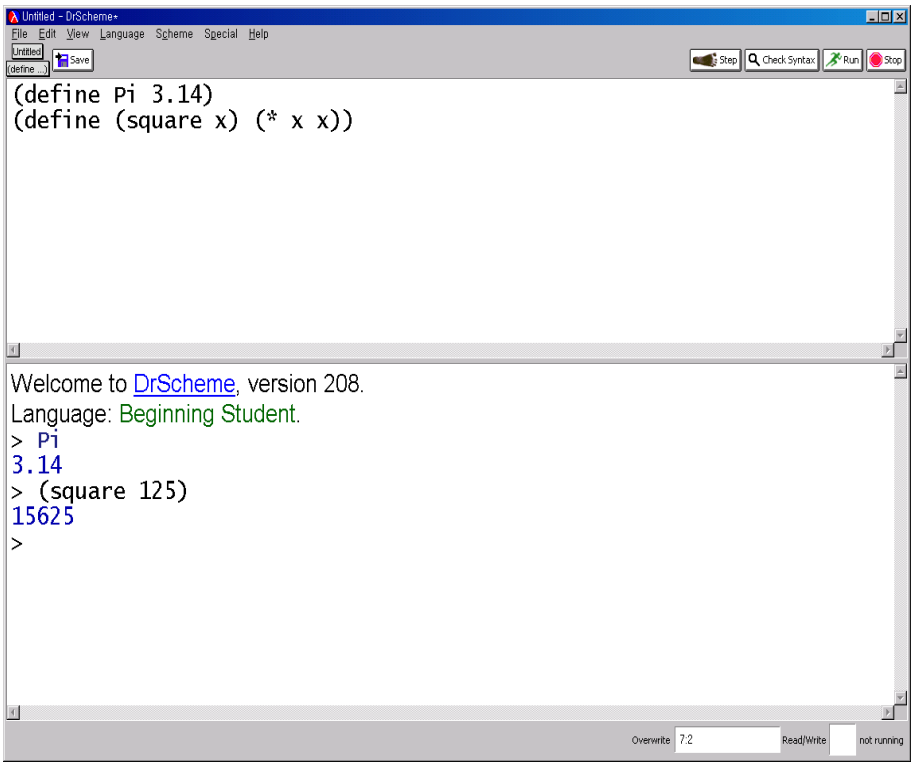
스킴에서 제공하는 추상화 수단은 “define” 이라는 예약어를 이용하고 “define”을 이용하여 작성된 것을 스킴에서는 프로시저라 한다. 프로시저를 작성하는 기본 형식은 다음과 같다.

① (define <이름> <바디>)
② (define (<이름> <인자이름1> ..... <인자이름n>) <바디>)

“define”의 기본 형식 중에서 ①은 값에 이름을 지은 것으로 예를 들면 (define Pi 3.14)라고 정의할 경우 사용자는 실제 값인 3.14 대신 Pi라는 이름을 이용하게 된다. <바디>에는 이름을 붙이고 싶은 수나 문자가 올 수 있다.

②는 복합식에 이름을 지은 것으로 예를 들면 (define (square x) (* x x))에서 square는 프로시저의 이름이 되고, square 프로시저의 인자 x 대신 값을 받아서 <바디>에 해당하는 (* x x)를 계산한다. 계산하는 방법을 <바디>에 표현하고 ①과 달리 프로시저 이름과 인자를 괄호()로 묶어서 하나로 취급하도록 되어 있다.

인자는 <바디>에서 계산되는 실제 값이 입력된다. 만약, 사용자가 125의 제곱을 알고 싶을 경우 Dr.Scheme의 실행창에 (square 125)라고 입력하면 (* 125 125)를 계산하여 15625라는 값을 화면에 보여준다. ①과 ②의 예를 Dr.Scheme에서 실행한 화면은 <그림 4-7>과 같다.



<그림 4-7> define 문법 예제

<Fig. 4-7> Example of define Syntax

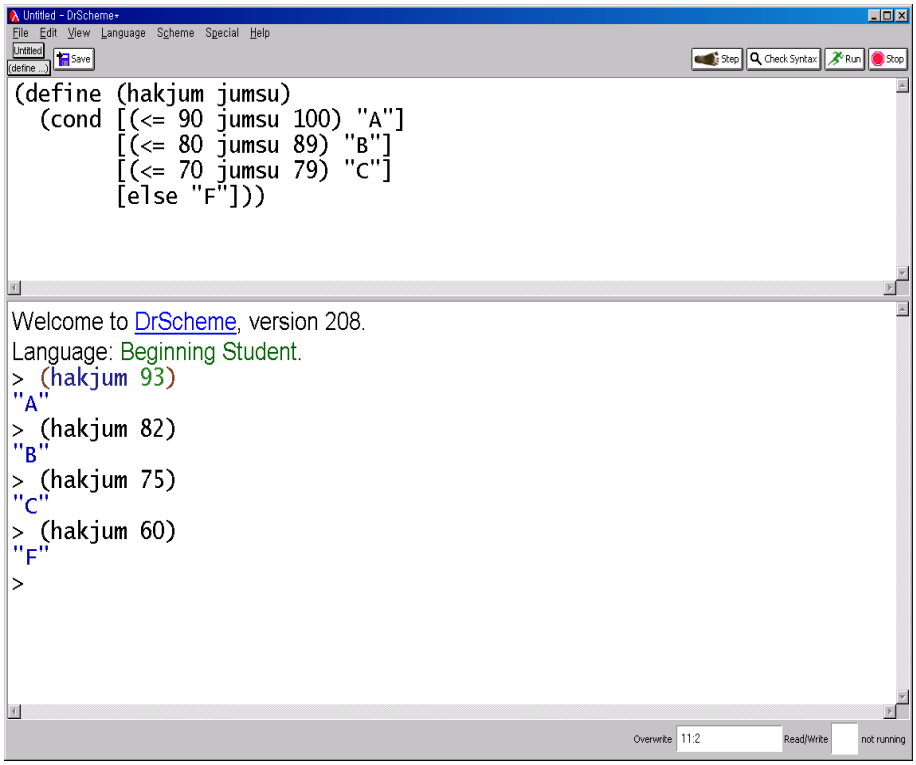
(2) cond 구문

학생의 학점은 성적에 따라 A, B, C, D, F가 된다. 성적에 따라 계산 결과인 학점이 나오도록 하는 스킴 표현은 cond 구문이고 기본 형식은 다음과 같다.

```
(cond
    [<질문1> <답1>]
        .....
    [<질문n> <답n>])
```

cond 구문에서 사용되는 <질문>에 대한 결과는 true 혹은 false 중 하나이다. cond 구문에서 사용된 여러 개의 질문 중에서 true가 되는 경우에 해당하는 <답>을 결과로 보여주고 나머지는 무시한다. cond 구문에서 마지막 <질문n> 대신 나머지 모든 경우를 포함하는 “else”라는 예약어를 이용할 수도 있다.

예를 들면, 학생의 학점은 성적이 90 이상 100 이하이면 “A”, 80 이상 89 이하이면 “B”, 70 이상 79 이하이면 “C”, 나머지는 “F”라 할 때 학생의 점수를 입력받아서 학점을 보여주는 “hakjum” 프로시저를 작성하여 실행한 결과는 <그림 4-8>과 같다.



<그림 4-8> cond 문법 예제

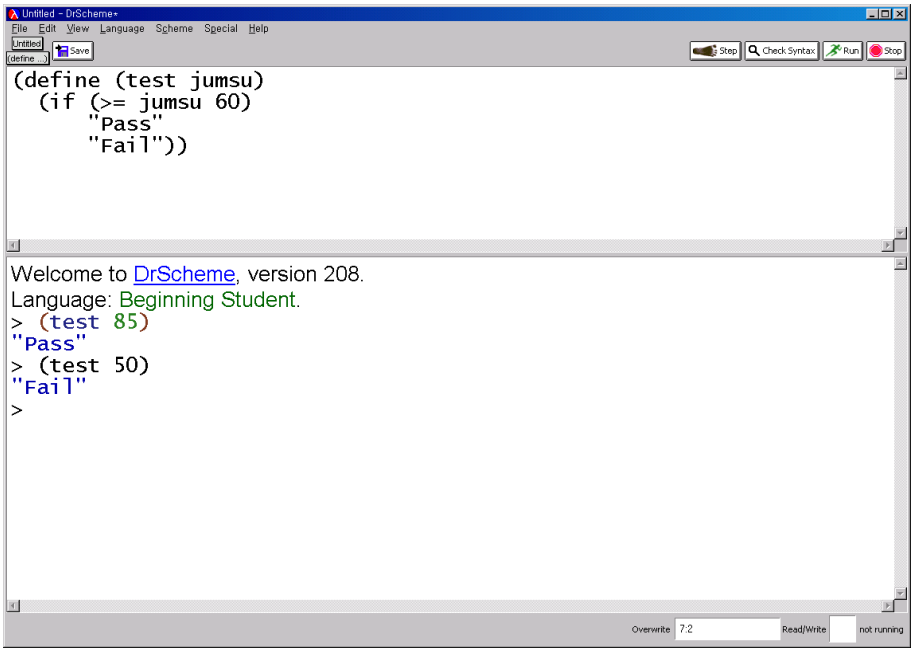
<Fig. 4-8> Example of cond Syntax

(3) if 구문

if 구문의 기본 형식은 다음과 같다.

(if <질문> <참일 때 답> <거짓일 때 답>)

if 구문은 cond 구문 중에서 <질문>의 결과에 따라 답이 두 개로 나오는 경우를 표현한다. if 구문에서 <질문>의 결과가 참일 때는 <참일 때 답>, <질문>의 결과가 거짓일 때는 <거짓일 때 답>이 결과가 된다. 예를 들면, 학생의 점수가 60 이상이면 “Pass”이고, 점수가 60 미만인 경우 “Fail”이라고 화면에 보여주는 “test” 프로시저를 작성하여 실행한 결과는 <그림 4-9>와 같다.



<그림 4-9> if 문법 예제

<Fig. 4-9> Example of if Syntax

4.3.2 lambda

“lambda”는 이름 없는 프로시저를 정의한다. “lambda”의 기본 형식은 다음과 같다.

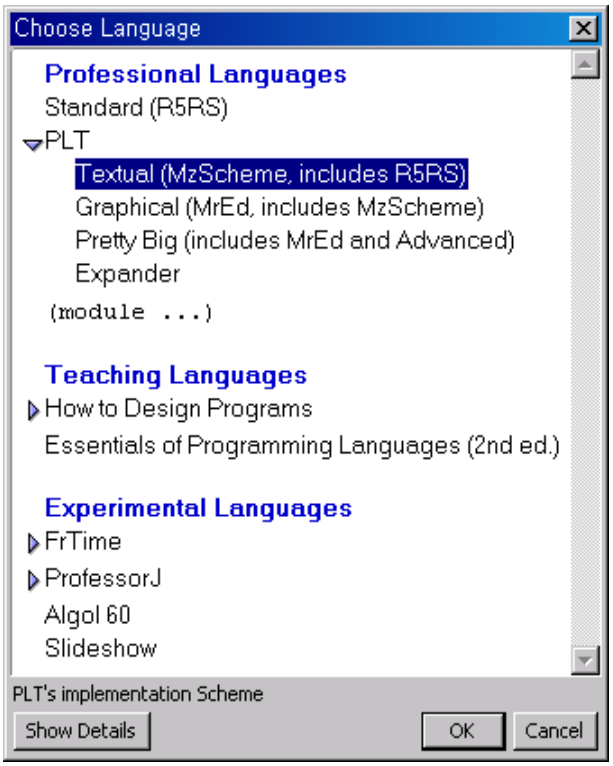
```
(lambda (<인자1> <인자 2>..... <인자 n>) <바디>)
```

“lambda” 표현은 추상화 수단으로 제공되는 “define”을 사용할 때 지어야 하는 이름을 정하는 번거로움을 없애기 위한 수단이다. 그러나, 이름이 없기 때문에 프로시저를 사용할 때마다 “lambda”로 표현된 부분을 전부 불러야 하는 번거로움이 있다. square, hakjum, test 프로시저를 “lambda”로 표현하면 다음과 같다.

```
① (lambda (x) (* x x))
② (lambda (jumsu)
        (cond [(<= 90 jumsu 100) “A”]
              [(<= 80 jumsu 89) “B”]
              [(<= 70 jumsu 79) “C”]
              [else “F”]))
③ (lambda (jumsu)
        (if (>= jumsu 60)
            “Pass”
            “Fail”))
```

①은 “square”, ②는 “hakjum”, ③은 “test” 프로시저를 “lambda”로 바꾼 것이다. “lambda”를 Dr.Scheme에서 실행하기 위해서는 언어 레벨을 먼저 바꾸어야 한다.

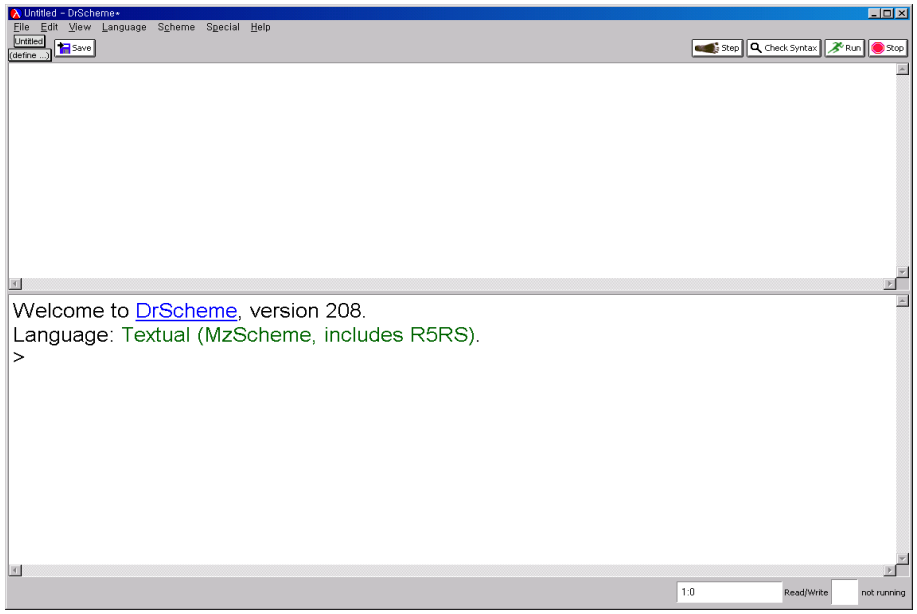
언어 레벨은 Dr.Scheme의 “Language-Choose Language”를 선택하면 언어를 고를 수 있는 <그림 4-10>과 같은 Choose Language 화면이 나타난다. 언어 레벨을 선택하는 Choose Language 창에서 “lambda”가 정의된 언어 레벨인 “PLT-Textual(MzScheme, includes R5RS)”를 선택하고 “OK” 버튼을 누르면 Choose Language 창이 사라진다.



<그림 4-10> 언어 선택 창

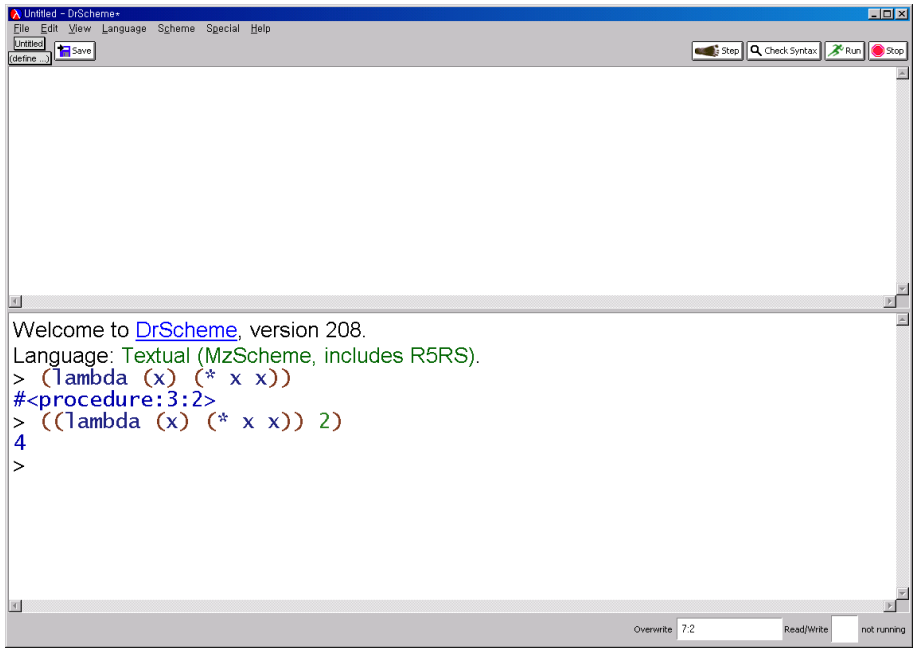
<Fig. 4-10> Window of Choose Language

Dr.Scheme에서 언어 레벨이 바뀐 화면은 <그림 4-11>과 같고, “square” 프로시저를 “lambda” 로 바꾸어 실행하는 화면은 <그림 4-12>와 같다.



<그림 4-11> 언어 변경 화면

<Fig. 4-11> Window of Choose Language Result



<그림 4-12> lambda 문법 예제

<Fig. 4-12> Example of lambda Syntax

4.3.3 리스트와 리스트 연산

여러 값을 묶어서 하나로 취급하는 리스트와 리스트 원소를 계산하는 map, filter, fold-right, fold-left 프로시저를 설명한다.

(1) 리스트

“list”는 여러 개의 값을 묶어서 하나로 처리하는 명령어이다.

“list”는 수나 문자를 여러 개를 괄호로 묶어 하나로 표현하는 프로시저로써 기본 형식은 다음과 같다.

```
(list <값1> <값2> <값3>..... <값(n-1)> <값n>)
```

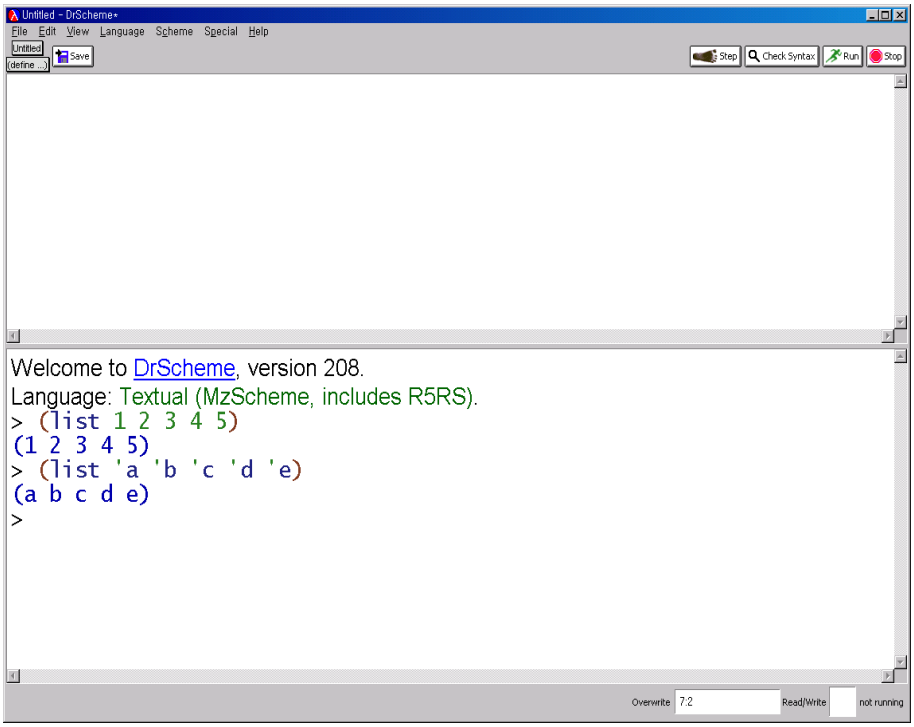
“list”는 두 개의 값을 하나로 묶는 쌍을 여러번 겹쳐서 표현한다. 쌍은 “cons”를 이용하여 표현하고 (cons <값1> <값2>)와 같이 표현한다. 쌍으로 묶을 수 있는 값은 수, 문자, 리스트, 쌍이다. “cons”는 왼쪽에 있는 값을 가져오는 “car”와 오른쪽에 있는 값을 가져오는 “cdr”가 있다. 리스트를 쌍으로 표현하면 다음과 같다.

```
(list <값1> <값2>..... <값n>)
= (cons <값1>
        (cons <값2>
                .....(cons <값n-1>
                            (cons <값n> nil))).....)
```

리스트를 쌍으로 표현한 것을 보면 리스트가 “nil”로 끝난다.

“nil”은 리스트 원소가 한 개도 없는 공 리스트 “()”를 의미한다. 쌍에서 값을 골라내는 “car”, “cdr”를 조합하면 리스트 원소를 골라낼 수 있다.

Dr.Scheme에서 수를 원소로 갖는 리스트와 문자를 원소로 갖는 리스트를 만드는 예를 실행한 화면은 <그림 4-13>과 같다.



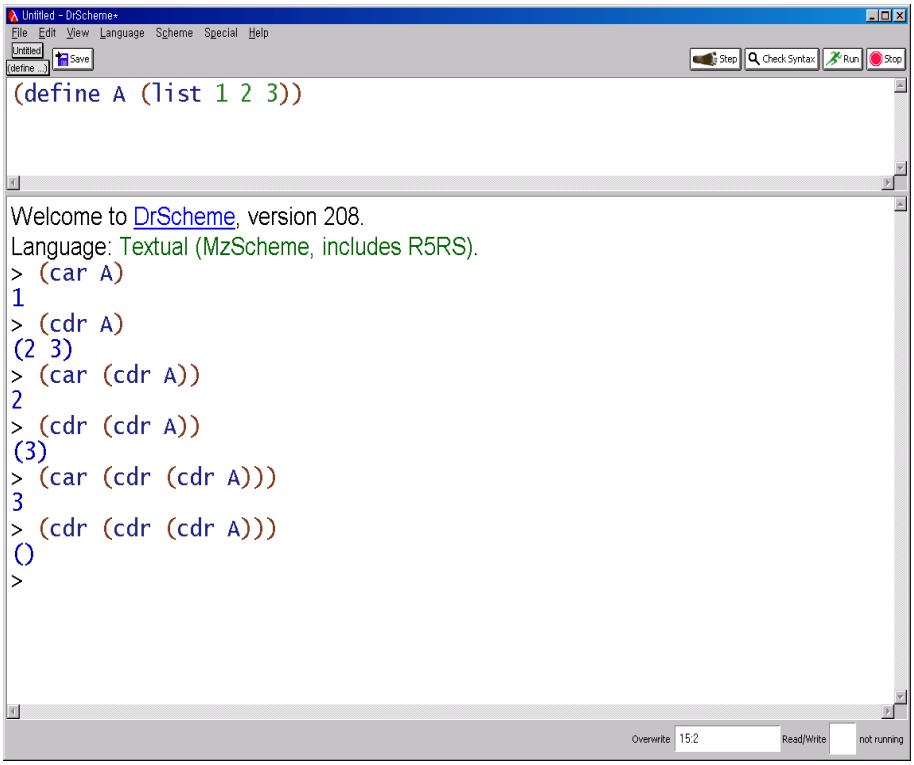
<그림 4-13> list 문법 예제

<Fig. 4-13> Example of list Syntax

Dr.Scheme에서 리스트 (1 2 3)에서 “car”, “cdr”를 조합해서 리스트 원소를 하나씩 골라내는 화면은 <그림 4-14>와 같다. <그림 4-14>와 같이 리스트의 처음 원소부터 마지막 원소까지 “car”, “cdr”를 조합하여 찾아낼 수 있다. 리스트의 처음 원소는 “car”로 골라낼 수 있고, “cdr”를 리스트 원소 개수만큼 반복하면 리스트의 끝

인 “nil”을 가리키는 “()”이다.

리스트에서 “nil”은 보이지 않기 때문에 “nil” 앞에 있는 값이 리스트의 마지막 원소가 된다. 결국, 리스트에서 “cdr”를 따라 가면 리스트의 모든 원소를 찾을 수 있다.



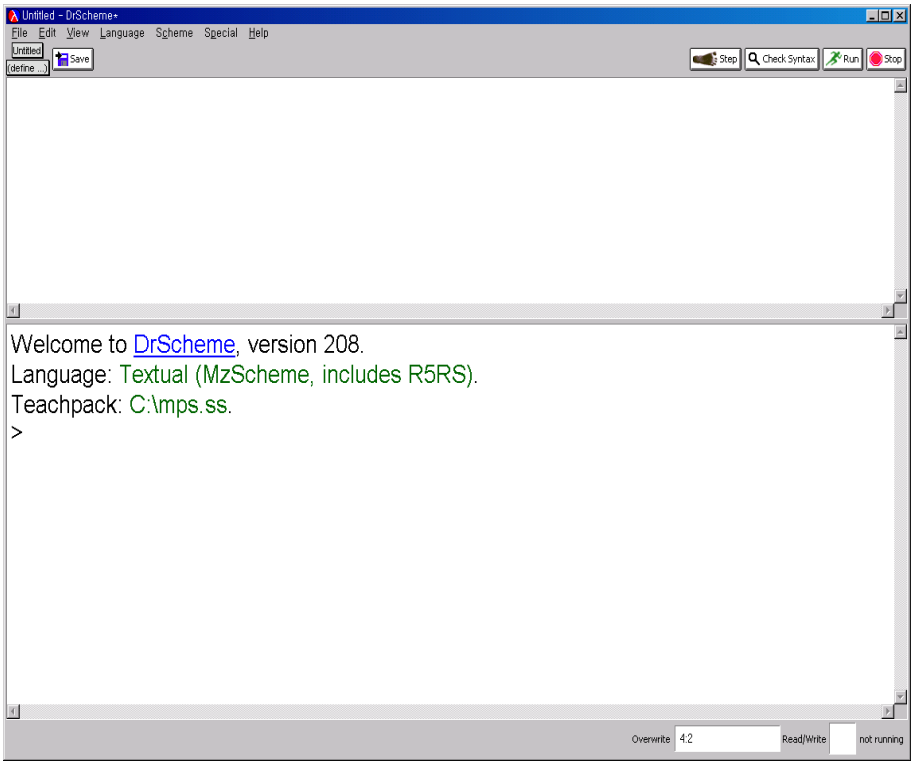
<그림 4-14> car, cdr 문법 예제

<Fig. 4-14> Example of car, cdr Syntax

(2) map

Dr.Scheme에서 “map” 프로시저를 사용하기 위해서는 “mps.ss”라는 Teachpack 파일을 추가해야 한다. 먼저, “mps.ss” 파일을 다운받을 수 있는 “http://mps.tit.ac.kr/html/03_lecterdata/lecturedata.htm”에 접속하여 Teachpack 파일을 저장한다. Dr.Scheme의 메뉴 중에

서 “Language-Add Teachpack...”을 선택하여 저장한 파일을 추가하고 Run 버튼을 누르면 <그림 4-15>와 같이 실행창에서 Teachpack 파일이 추가된 것을 확인할 수 있다.



<그림 4-15> Teachpack 추가 화면

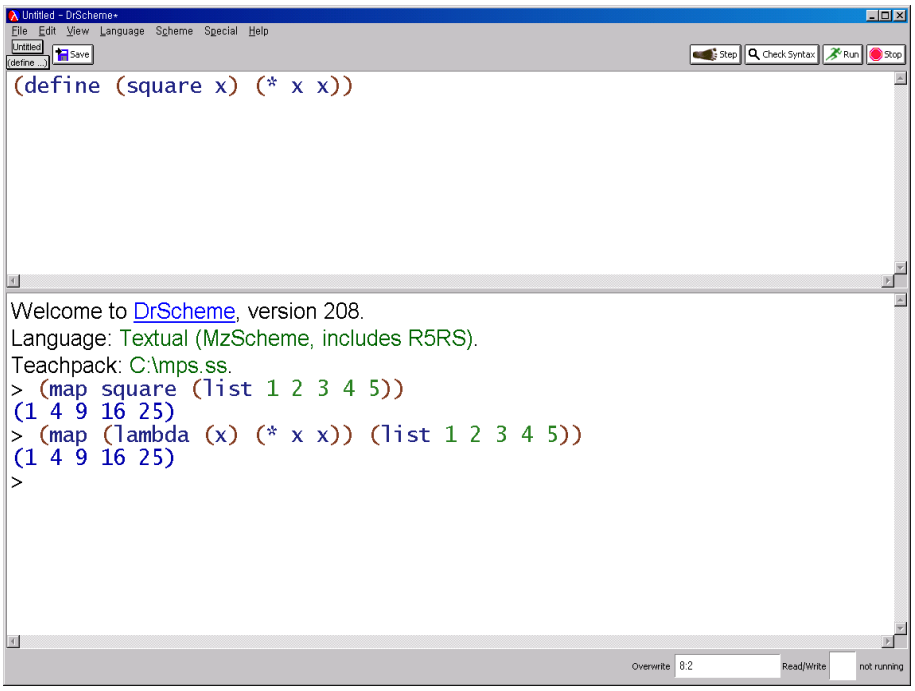
<Fig. 4-15> Window of Add Teachpack

“map”은 리스트 원소를 같은 계산 방법으로 계산한 결과를 리스트로 표현하는 프로시저이다. “map”의 기본 형식은 다음과 같다.

(map <프로시저> <리스트1> <리스트2> ..... <리스트n>)

기본 형식에서 <프로시저>는 리스트 원소를 계산하는 방법을 나타내는 프로시저의 이름이 온다. 이름없이 “lambda”를 이용하여 프로시저를 직접 작성해도 된다. “map”은 리스트 여러 개를 동시에 계산할 수 있다. “map”에서 리스트가 하나일 때는 리스트 원소를 하나씩 차례대로 <프로시저>의 인자로 주어서 계산하고, 리스트가 여러 개일 때는 각각의 리스트에서 같은 위치에 있는 값들을 차례대로 <프로시저>의 인자로 주어서 계산한다.

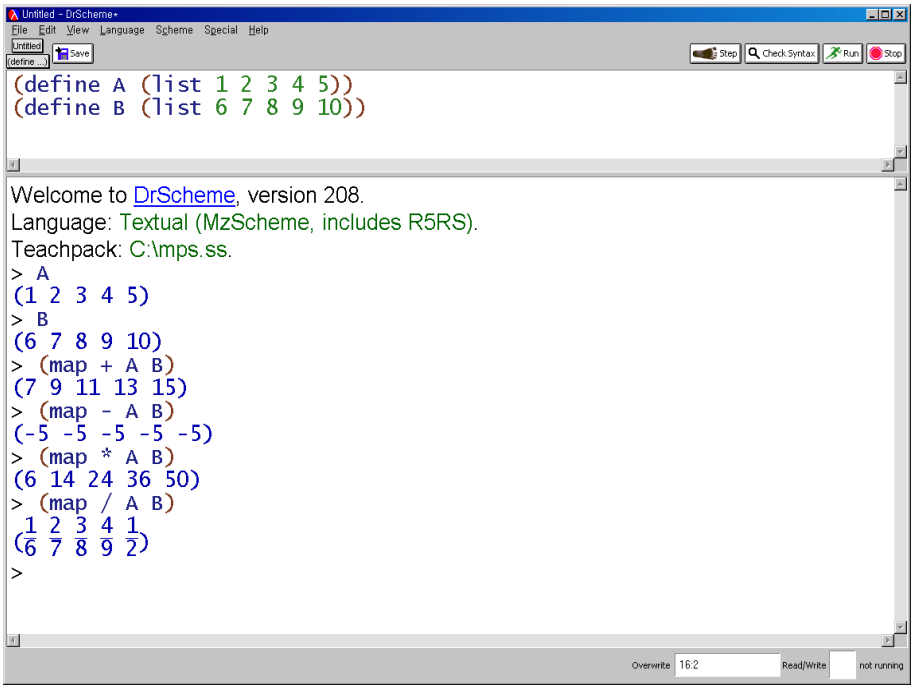
예를 들면, 리스트 (1 2 3 4 5)를 제곱할 경우 수의 제곱을 계산하는 “square”라는 프로시저를 만들거나 “lambda”를 이용하여 “map” 식을 완성할 수 있다. 리스트 원소를 제곱하는 “map” 식을 Dr.Scheme에서 작성하여 실행하는 화면은 <그림 4-16>과 같다.



<그림 4-16> map 문법 예제 1

<Fig. 4-16> Example 1 of map Syntax

"map"을 이용하여 리스트 (1 2 3 4 5)와 리스트 (6 7 8 9 10)을 "A", "B"라는 이름을 붙여 정의한 뒤에 두 리스트 A와 B를 더하기/빼기/곱하기/나누기와 같은 사칙연산을 하는 "map" 식을 작성하여 실행하는 화면은 <그림 4-17>과 같다.



<그림 4-17> map 문법 예제 2

<Fig. 4-17> Example 2 of map Syntax

(3) filter

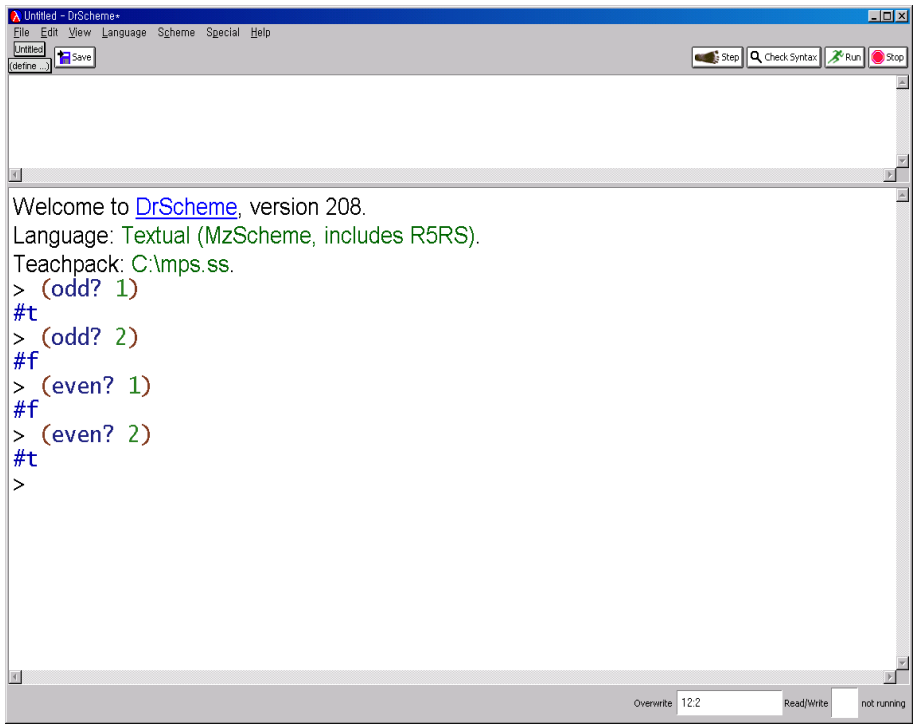
"filter"는 리스트 원소 중에서 조건을 만족하는 원소만 골라 내는 프로시저로써 기본 형식은 다음과 같다.

(filter <조건> <리스트>)

“filter”는 계산 결과가 리스트 형태이고, “map”과 달리 리스트 하나일 때만 계산할 수 있다. “filter”는 리스트 각각의 원소가 <조건>을 만족할 경우에는 “#t”이 되고 만족하지 않을 경우에는 “#f”가 되는 형태의 프로시저가 <조건>이 된다. “#t”는 “true”, “#f”는 “false”를 의미한다.

“filter”는 리스트 각각의 원소에 <조건>을 실행하여 그 결과가 “#t”인 것만 리스트로 묶는다. <조건>에 “lambda”를 이용하여 계산 결과가 “#t”, “#f”가 되는 프로시저를 정의하여 표현할 수도 있다.

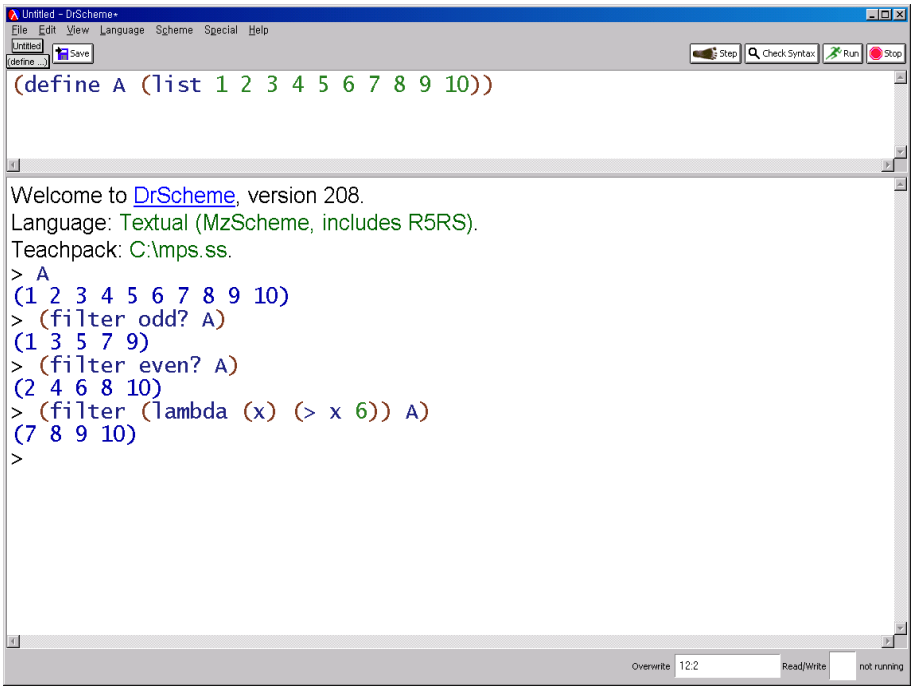
스킴에 내장된 “odd?”와 “even?” 프로시저는 수가 홀수 혹은 짝수인지 확인하는 프로시저이다. “odd?”와 “even?” 프로시저를 <수>에 적용하여 계산한 결과는 <그림 4-18>과 같다.



<그림 4-18> odd?, even? 사용 예

<Fig. 4-18> Example of odd? even?

“odd?”와 “even?” 프로시저를 “filter”에 적용하여 리스트 (1 2 3 4 5 6 7 8 9 10)에서 홀수와 짝수만 골라내는 것과 리스트 원소 중에서 6보다 큰 수만 골라내는 “filter” 식을 작성하여 실행한 화면은 <그림 4-19>와 같다. “filter”는 <조건>을 만족시키는 리스트 원소만 골라 내는 프로시저이므로 <조건>을 프로시저로 작성하면 사용자가 원하는 리스트 원소만 골라낼 수 있다.



<그림 4-19> filter 문법 예제

<Fig. 4-19> Example of filter Syntax

(4) fold-right/fold-left

“fold-right”와 “fold-left”는 리스트 원소를 계산하는 프로시저로써 기본 형식은 다음과 같다.

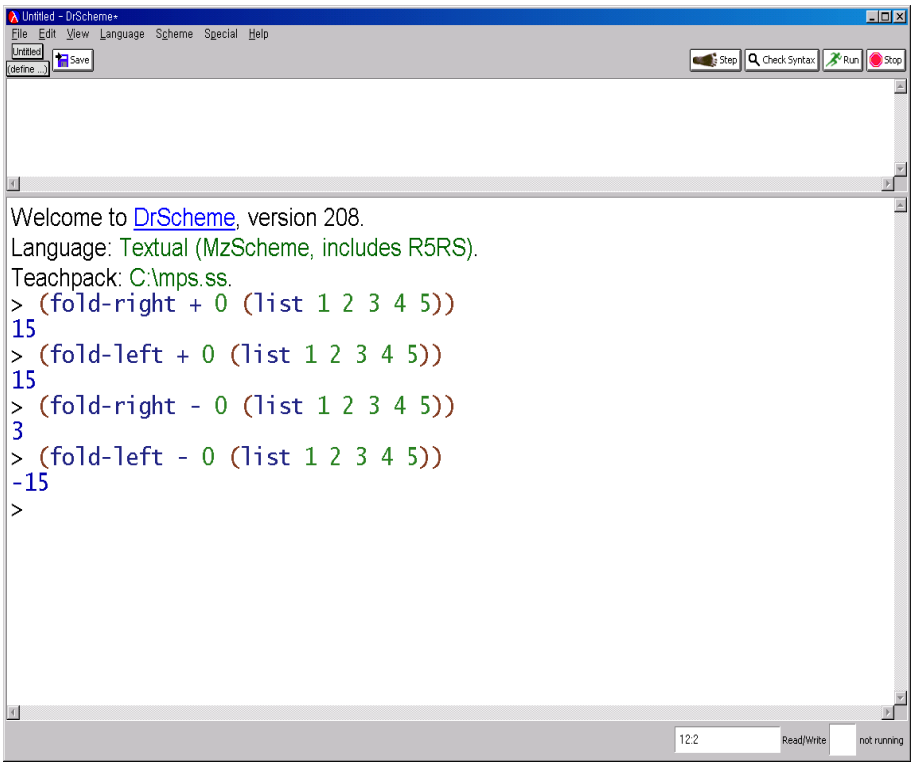
```
(fold-right <프로시저> <초기값> <리스트>)
(fold-left <프로시저> <초기값> <리스트>)
```

<프로시저>는 리스트 원소를 계산하는 방법이고, <초기값>은 리스트 원소와 제일 먼저 계산되는 값으로 리스트 원소가 없는 경우에는 초기값이 결과가 된다. “fold-right”와 “fold-left”는 사용 방법은 같지만 계산하는 방향이 반대이다.

“fold-right”는 리스트 원소 중 가장 오른쪽에 있는 n번째 원소와 <초기값>을 먼저 계산하고, 계산한 결과를 (n-1)번째 원소와 계산하는 과정을 반복한다. 그러나, “fold-left”는 리스트 원소 중 가장 왼쪽에 있는 0번째 원소와 <초기값>을 먼저 계산하고, 계산한 결과를 첫 번째 원소와 계산한다. 이와 같은 과정을 반복하여 리스트 원소 전체를 계산한다. “fold-right”와 “fold-left”을 이용하여 리스트 (1 2 3 4 5)의 원소를 더하는 계산 과정은 다음과 같다.

```
(fold-right + 0 (list 1 2 3 4 5))
⇒ (+ 1 (+ 2 (+ 3 (+ 4 (+ 5 0)))))
(fold-left + 0 (list 1 2 3 4 5))
⇒ (+ (+ (+ (+ (+ 0 1) 2) 3) 4) 5)
```

“fold-right”와 “fold-left”에서 사용되는 <프로시저>가 +, *와 같이 교환법칙이 성립하는 경우에는 계산 결과가 같지만 <프로시저>가 -, /와 같이 교환법칙이 성립하지 않을 경우에는 계산 결과가 다르다. 리스트 (1 2 3 4 5)를 “fold-right”와 “fold-left”를 이용하여 교환법칙이 성립하는 덧셈을 한 경우와 교환법칙이 성립하지 않는 뺄셈을 계산하는 화면은 <그림 4-20>과 같다.



<그림 4-20> fold-right/fold-left 문법 예제

<Fig. 4-20> Example of fold-right/fold-left

리스트를 처리하는 map, filter, fold-right, fold-left를 이용하면 엑셀에서 처리할 수 있는 대량 데이터 처리와 데이터베이스에 저장된 테이블을 리스트로 취급하여 데이터를 관리할 수 있다.

본 논문에서 구현한 웹 에이전트 시스템은 이러한 리스트의 특성을 살려서 데이터베이스에 저장된 데이터를 관리하는 기능과 함께 데이터를 처리해서 사용자가 원하는 자료를 만드는 엑셀의 기능까지 추가한 것이다.

제 5 장 DTD 생성 규칙

XML 문서와 데이터베이스 간의 데이터 교환을 위해 XML 문서는 데이터베이스로, 데이터베이스는 XML 문서로 매핑해야 한다.

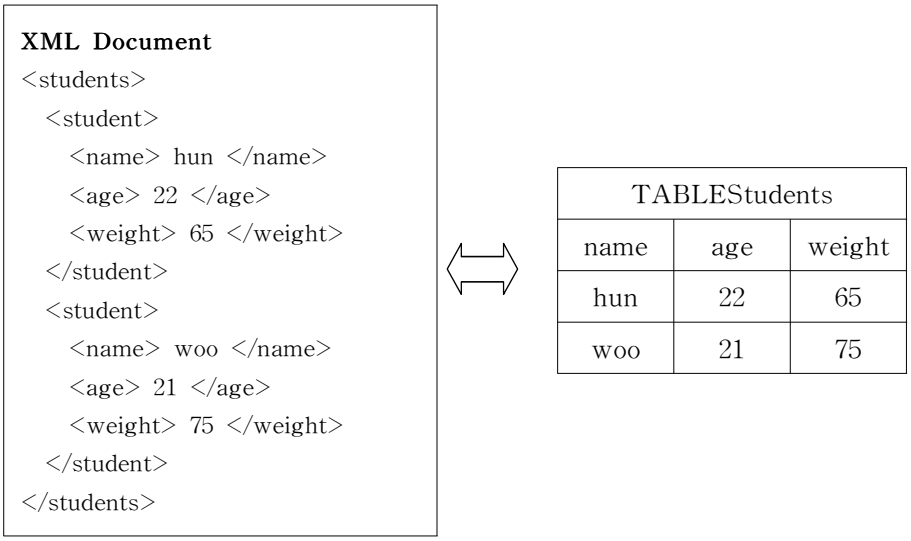
5.1 XML과 데이터베이스 매핑

XML 문서를 데이터베이스로 매핑하기 위해서 기존에는 테이블-기반 매핑과 객체-관계형 매핑 방법이 사용되고 있다[37]~[42]. 데이터베이스에는 데이터가 있는 테이블이 저장되고 테이블은 컬럼에 해당하는 정보와 필드로 구성되어 있다. 테이블은 컬럼을 여러 개 나열한 것과 같은 구조이므로 스킴에서 사용되는 리스트와 같은 구조이다. 따라서 본 논문에서는 테이블의 구조를 리스트로 해석하여 XML 문서로 매핑하는 테이블-리스트 매핑 방법을 제안한다.

5.1.1 테이블-기반 매핑과 객체-관계형 매핑 방법

테이블-기반 매핑은 XML 문서와 관계형 데이터베이스 간의 데이터를 전송하는 미들웨어 제품에서 주로 사용되는 방식으로 XML 문서를 하나의 테이블 또는 테이블 셋(set)으로 모델링한다[43]~[48]. 즉, XML 문서의 구조는 <그림 5-1>과 같이 <database 이름> 요소와 데이터베이스 내에 여러 개의 테이블이 존재하는 경우 추가적인 <table 이름> 요소로 표현한다.

테이블 기반 매핑은 컬럼 데이터를 자식 요소나 속성으로 저장할지 혹은 요소, 속성의 명명규칙으로 지정할지를 선택할 수도 있고 소프트웨어에서 자동으로 지정하게 할 수도 있다.



<그림 5-1> 테이블-기반 매핑

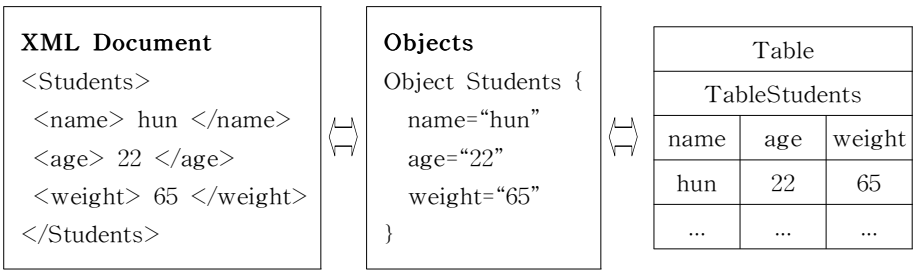
<Fig. 5-1> Table-Based Mapping

테이블-기반 매핑을 사용하는 소프트웨어에서는 추가적으로 문서의 시작 위치에 또는 각 테이블 또는 컬럼 요소의 속성에 테이블과 컬럼 메타데이터를 포함하여 저장한다.

매핑은 두 개의 관계형 데이터베이스 간의 데이터를 전송할 때와 같이 관계형 데이터를 직렬화시킬 때는 유용하지만, 기존의 포맷과 일치하지 않는 XML 문서에서는 사용할 수 없다는 단점이 있다.

객체-관계형 매핑은 XML을 지원하는 모든 관계형 데이터베이스 제품에서 사용되는 방식으로서 애트리뷰트, 요소 컨텐cm, 복잡한 요소 타입 등을 클래스로 모델링하고 PCDATA 요소 타입, 속성 등을 갖는 요소는 스칼라로 모델링된다. 객체-관계형 매핑은 전통적인 객체-관계형 매핑 기술 또는 객체 뷰(view)를 사용하여 관계형 데이터베이스로 매핑할 때 클래스는 테이블로 매핑되고, 스칼라는 컬럼으로 매핑되며 객체-값은 고유 키와 외래 키에 쌍으로 매핑된다.

Students 테이블을 객체-관계형 매핑으로 표현한 <그림 5-2>와 같이 Students 문서를 클래스로 구성된 객체 트리로 모델링될 수 있다. 문서로부터 DOM 트리를 생성한다면 요소, 애트리뷰트, 텍스트 등과 같은 객체들로 구성된다. 모델 내에 있는 객체는 모델링 내에 클래스들을 생성하게 하고, 생성된 클래스를 응용 프로그램 내에서 객체로 선언하여 사용한다. XML 문서와 객체 또는 객체와 데이터베이스 간에 데이터의 전송이 이루어지고, 다른 소프트웨어에서는 객체를 단지 매핑을 시각화하는데 도움을 주기 위한 툴로서 사용하고 데이터는 XML 문서와 데이터베이스 간에 직접 전송한다.



<그림 5-2> 객체-관계형 매핑

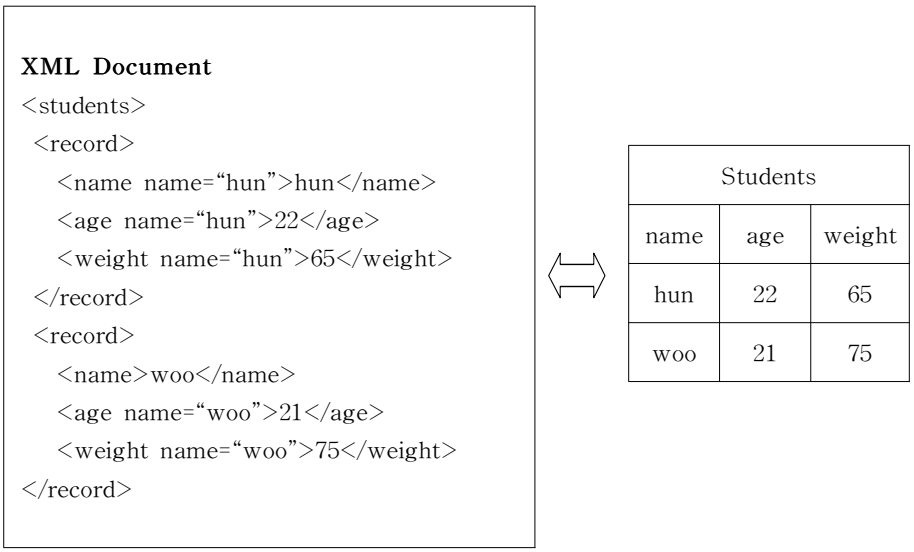
<Fig. 5-2> Object-Relational Mapping

최근에는 XML 문서를 객체로 바인딩(binding)하는 방식으로 많이 사용되고 있고, JAXB(Java Architecture for XML Binding)가 나온 후에 XML 데이터 바인딩 방식도 많이 사용되고 있다. 데이터 바인딩(data binding)이란 XML 데이터와 객체간의 매핑 및 프로그램 구현을 쉽게 해주기 위한 일련의 기술을 말한다. 데이터 바인딩 기술을 이용하면 클래스 파일을 자동으로 생성하고, XML과 클래스 간의 매핑을 자동으로 처리하여 데이터 접근이 용이하다. 현재 많은 XML 데이터 바인딩을 위한 상용 소프트웨어가 있으며 이를 이용하여 객체와 데이터베이스 간에 데이터를 전송한다.

5.1.2 테이블-리스트 매핑

본 논문에서 제안한 테이블-리스트 매핑은 데이터베이스에 저장된 테이블의 내용을 확인하거나 수정, 삭제하는 기능과 몇 개의 테이블을 조합해서 하나의 새로운 테이블을 만드는 기능, 테이블에 저장된 데이터를 이용하여 새로운 데이터를 만드는 기능이 추가되었다. 또한, 테이블의 내용을 수정할 경우 저장할 때 테이블 전체 내용이 저장되는 기존의 방법과 달리 수정한 부분만 저장할 수 있다.

예를 들면, 데이터베이스에 “students”라는 이름의 테이블에는 학생의 이름, 나이, 몸무게에 대한 데이터가 저장되어 있다. 이를 테이블-리스트 매핑으로 표현하면 <그림 5-3>과 같다.



<그림 5-3> 테이블-리스트 매핑

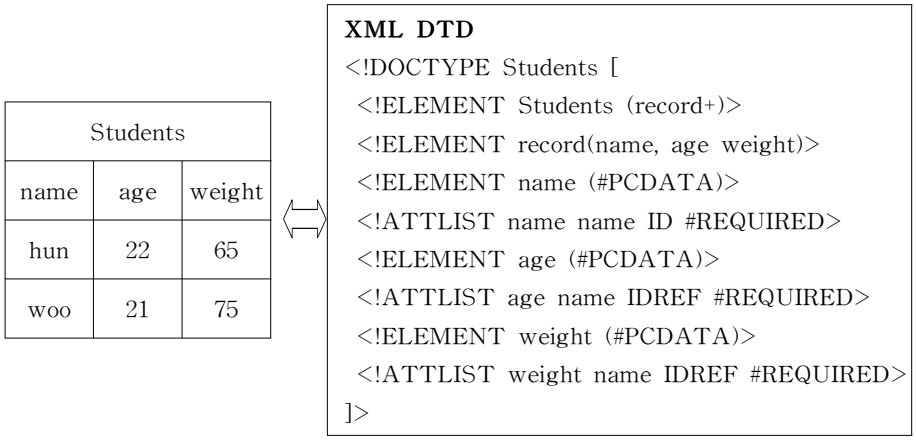
<Fig. 5-3> Table-list Mapping

5.2 데이터베이스 변환 DTD 생성 규칙

데이터베이스에 저장된 데이터를 XML 문서로 모델링하기 위해서는 데이터베이스에 저장되어 있는 테이블의 구조를 파악해야 하며, 데이터베이스에 저장된 데이터를 변환하는 범위를 정해야 한다. 그러나 본 논문에서 제안한 매핑 규칙은 웹에서 동작하기 때문에 사용자가 변환하는 범위를 목적에 맞게 선택할 수 있다.

(1) 단독 테이블인 경우

<그림 5-4>와 같이 테이블에 저장된 데이터가 다른 테이블의 내용을 참조하지 않을 경우에는 테이블 이름인 Students라는 루트(root) 요소를 생성한다. 루트 요소는 테이블에서 데이터 컬럼을 의미하는 “record”라는 자식 요소를 가지고, record 요소는 자식 요소로 name, age, weight를 요소로 가지고, name, age, weight 요소는 각각 PCDATA 타입의 요소로 선언한다.



Students		
name	age	weight
hun	22	65
woo	21	75

XML DTD

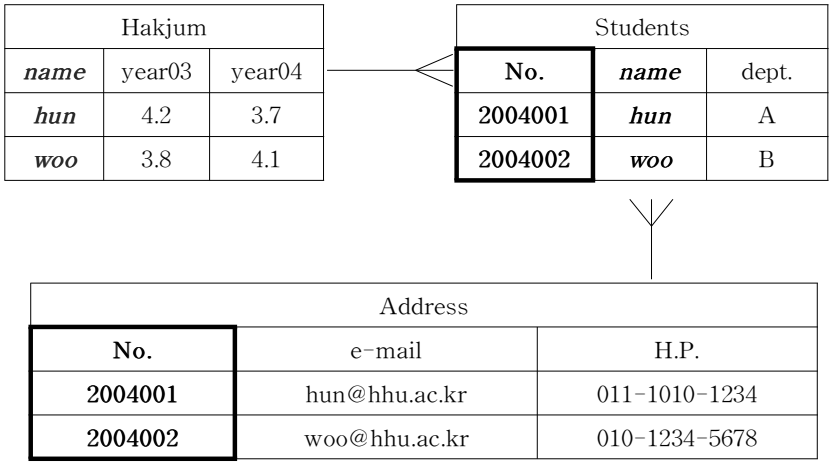
```
<!DOCTYPE Students [
 <!ELEMENT Students (record+)>
 <!ELEMENT record(name, age weight)>
 <!ELEMENT name (#PCDATA)>
 <!ATTLIST name name ID #REQUIRED>
 <!ELEMENT age (#PCDATA)>
 <!ATTLIST age name IDREF #REQUIRED>
 <!ELEMENT weight (#PCDATA)>
 <!ATTLIST weight name IDREF #REQUIRED>
]>
```

<그림 5-4> 단독 테이블인 경우

<Fig. 5-4> Case of Single Table

(2) 다중 관계형 테이블인 경우

테이블에 저장된 데이터가 다른 테이블의 내용을 참조할 경우에 내용을 참조하는 모든 테이블의 내용을 XML DTD로 생성할 경우 테이블의 성격에 따라 요소의 수가 많아져 구조가 복잡해지고 이로 인해 사용자가 원하는 작업을 수행하는 시간보다 데이터를 읽어오는 시간이 더 걸리게 된다. 따라서 본 논문에서는 사용자가 선택한 범위에 속하는 테이블만 관계를 분석하여 XML DTD를 만드는 규칙을 제안한다. 다중 관계를 가지는 <그림 5-5>와 같은 테이블 구조에서 만들어지는 XML DTD를 이용해서 매핑 과정을 설명한다.



Hakjum

name	year03	year04
hun	4.2	3.7
woo	3.8	4.1

Students

No.	***name***	dept.
2004001	***hun***	A
2004002	***woo***	B

Address

No.	e-mail	H.P.
2004001	hun@hhu.ac.kr	011-1010-1234
2004002	woo@hhu.ac.kr	010-1234-5678

<그림 5-5> 다중 관계형 테이블인 경우

<Fig. 5-5> Case of Multiple Relation Table

<그림 5-5>에서 Hakjum 테이블의 name 컬럼은 Students 테이블의 name과 연결되어 있고, Address 테이블의 No.은 Students 테이블의 No.와 연결되어 있다. <그림 5-5>와 같은 구조에서 테이블 여러 개가 서로 연결된 복잡한 경우에는 컬럼명이 같고 그 내용이

같은 경우를 찾아서 이를 애트리뷰트로 정의한다. <그림 5-5>에서 최상위에 있는 테이블이 Students이므로 최상위 요소를 <Students>로 정한다.

Hakjum 테이블의 경우 2003, 2004 컬럼은 애트리뷰트 속성으로 name은 ID 속성 타입으로 선언하고 속성 값 중 하나를 가져야 하는 IDREF를 이용하고 속성의 기본값은 꼭 사용해야 하고 유일한 값이어야 하는 #REQUIRED를 사용한다. Address 테이블의 경우에는 e-mail과 H.P 컬럼이 이에 속한다. <그림 5-5>와 같은 관계를 가지는 테이블의 내용에 맞는 DTD 문서를 작성한 결과는 <그림 5-6>과 같다.

XML DTD

```
<!DOCTYPE Students [
  <!ELEMENT Students (record+)>
  <!ELEMENT record(No., name, dept.)>
  <!ELEMENT No. (#PCDATA)>
  <!ATTLIST No. No. ID #REQUIRED>
  <!ELEMENT name (#PCDATA)>
  <!ATTLIST name name ID #REQUIRED>
  <!ELEMENT dept. (#PCDATA)>
  <!ELEMENT Hakjum (year03, year04)>
  <!ELEMENT year03 (#PCDATA)>
  <!ATTLIST year03 name IDREFS #REQUIRED>
  <!ELEMENT year04 (#PCDATA)>
  <!ATTLIST year04 name IDREFS #REQUIRED>
  <!ELEMENT Address (e-mail, H.P.)>
  <!ELEMENT e-mail (#PCDATA)>
  <!ATTLIST e-mail No. IDREFS #REQUIRED>
  <!ELEMENT H.P. (#PCDATA)>
  <!ATTLIST H.P. No. IDREFS #REQUIRED>
]>
```

<그림 5-6> DTD 문서

<Fig. 5-6> DTD Document

제 6 장 웹 에이전트 시스템

6.1 웹 에이전트 시스템 설계

6.1.1 웹 에이전트 시스템

제 5 장에서 언급한 내용을 토대로 구현한 웹 에이전트 시스템은 사용자가 웹 브라우저를 통해 데이터베이스에 접근하여 데이터를 관리하거나 XML 문서로 저장한다. 또한, XML 문서를 데이터베이스에 저장하거나 서로 다른 데이터베이스간의 데이터를 복제하여 전송할 수 있다.

웹 에이전트 시스템 환경은 윈도우 2000 서버이고, 데이터베이스는 오라클과 MS SQL 서버 7.0을 사용한다. 웹 서버는 인터넷 정보 서버(IIS ; Internet Information Server) 5.0을 사용하여 웹 서버를 구동하고, 동적인 웹 페이지를 생성하는 JSP(Java Server Page) 엔진으로 레신(resin)을 이용하고 JDK(Java Development Kit)를 설치하여 JDBC 드라이버와 연동이 되도록 설정하였다.

웹 에이전트에서 데이터베이스에 접근하기 위해 JDBC 드라이버를 설정하고, XML 문서를 볼 수 있는 인터넷 익스플로러(explorer) 4.0 이상의 브라우저를 사용한다. 웹 서버는 자바와 연동이 되는 JSP를 이용하여 다른 환경으로 큰 변경 작업없이 이식이 가능하고, 프로세스를 생성하는 것보다 50배 이상 빠르다. JSP는 컴파일 과정에서부터 서블릿을 불러들이는 과정까지를 JSP 컨테이너가 처리해 주기 때문에 많은 부하를 줄일 수 있다. 따라서, 본 논문에서 웹 서버는 서블릿이 아닌 JSP를 이용한다.

6.1.2 웹 에이전트 시스템 구성

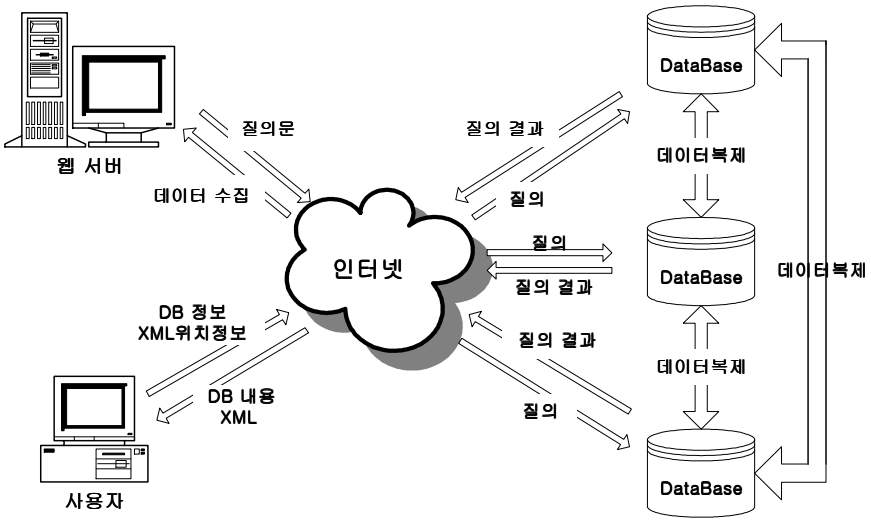
본 논문에서 데이터베이스에 저장된 테이블의 데이터와 XML 문서를 매핑하는 방법을 제안하고, 제안한 매핑 방법을 이용하여 웹에서 데이터베이스에 저장된 데이터를 관리하는 웹 에이전트 시스템을 구현한다. 웹 에이전트 시스템은 기존의 데이터베이스에서 이루어지던 테이블에 저장된 데이터의 저장 및 삭제, 수정이 가능할 뿐만 아니라 테이블에서 데이터를 선택하여 처리한 결과를 새로운 테이블로 만들 수 있다.

본 논문에서 구현한 웹 에이전트 시스템은 웹 서버가 데이터베이스에 접근하기 위해 JDBC와 자바를 이용하여 데이터를 불러오면 스킴 언어를 이용하여 테이블의 내용을 분석하여 DTD를 작성하고 테이블의 컬럼에 저장되어 있는 데이터를 자신의 컬럼 이름을 이름으로 하는 리스트를 만든다. 웹 브라우저를 통해 사용자가 원하는 데이터 처리는 실제 리스트의 내용을 처리하는 것이다. 이는 엑셀 프로그램에서 데이터를 관리하는 것과 유사하게 다양한 데이터 처리가 가능하다는 의미이다. 또한, 테이블의 구조가 웹 페이지에서 테이블 형태로 보여주기 때문에 사용자는 훨씬 더 간편하게 데이터베이스를 관리할 수 있다.

웹 에이전트의 전체 시스템 구성은 <그림 6-1>과 같다.

<그림 6-1>과 같이 구성된 웹 에이전트는 인터넷을 통하여 사용자와 웹 서버간, 웹 서버와 데이터베이스간의 정보를 교환하고, 사용자는 웹 브라우저를 통하여 결과를 확인한다.

웹 에이전트 시스템은 웹에서 데이터베이스를 관리하고 처리하는 과정, 데이터베이스를 XML 문서로 만드는 과정, XML 문서를 데이터베이스에 저장하는 과정, 끝으로 데이터베이스간의 데이터 복제과정으로 나누어진다.

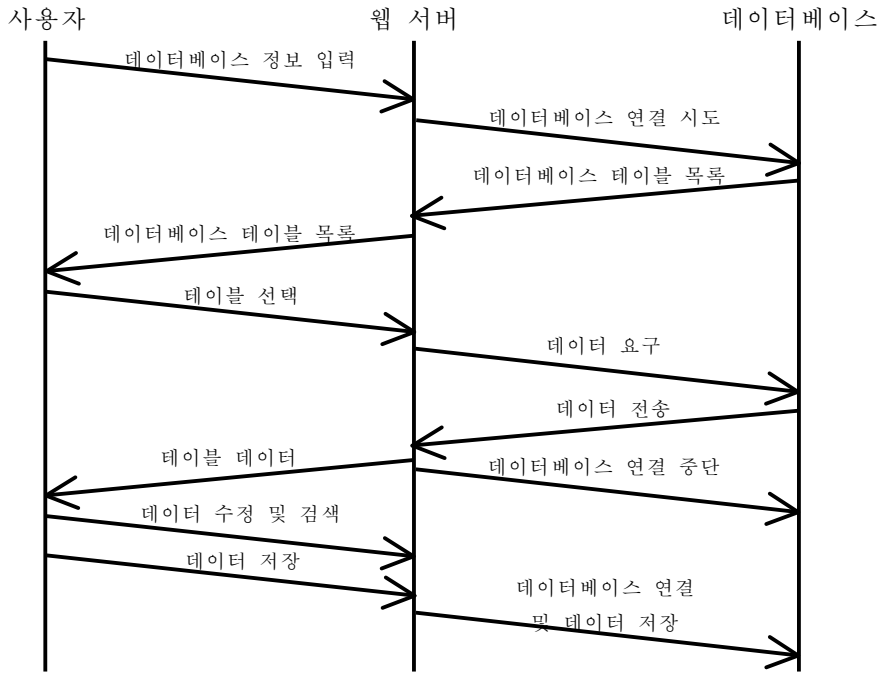


<그림 6-1> 웹 에이전트 시스템 구성

<Fig. 6-1> Configuration of Web Agent System

웹 에이전트에서 데이터베이스에 저장된 데이터를 관리하는 과정은 <그림 6-2>와 같다.

웹에서 데이터베이스를 관리하는 경우, 사용자는 웹 브라우저를 통해 데이터베이스에 대한 정보를 입력하여 웹 서버로 전송한다. 웹 서버는 사용자가 원하는 데이터베이스와 연결하여 데이터를 가져오고, 데이터베이스에서 가지고 온 데이터를 웹 브라우저 화면에서 사용자가 원하는 처리과정을 입력하면 웹 서버는 웹 서버에 저장되어 있는 리스트를 처리하여 브라우저 화면에 보여주고 사용자가 최종 확인한 경우에만 데이터베이스에 저장하도록 하여 데이터베이스의 부하를 줄여준다. 웹 서버가 데이터베이스와 연결하여 데이터를 가져올 때 웹 서버는 데이터에 해당하는 XML 문서를 자동으로 생성하여 “테이블이름.xml” 파일로 저장한다.

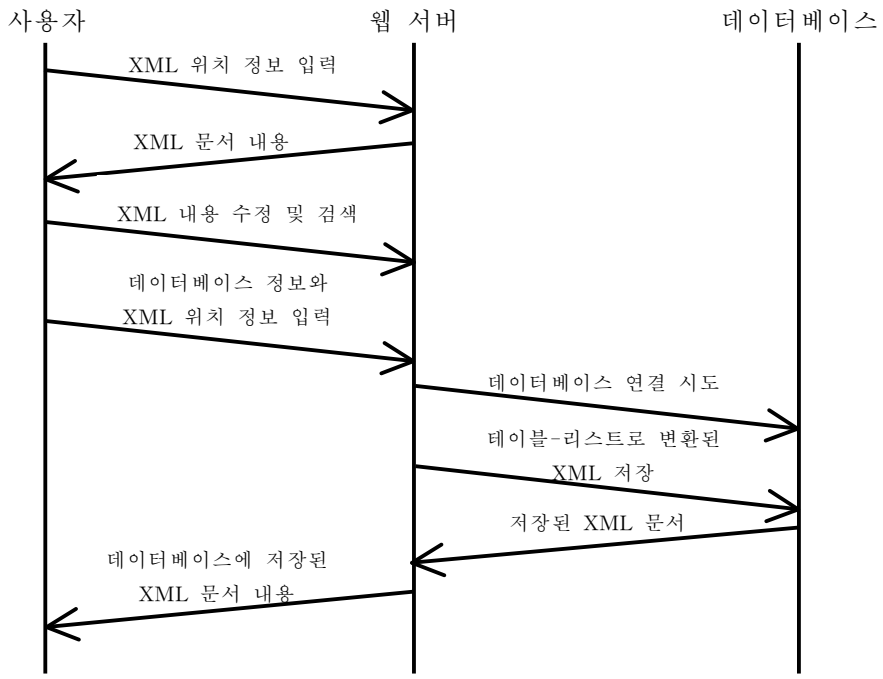


<그림 6-2> 데이터베이스 관리 데이터 흐름

<Fig. 6-2> Data Flow for Database Management

웹 브라우저 화면에서 사용자가 연결하기 원하는 데이터베이스를 선택하여 정보를 입력하면 데이터베이스는 사용자가 입력한 아이디와 암호를 확인하여 웹 서버와 데이터베이스를 연결한다. 웹 서버와 데이터베이스가 연결되면 웹 서버는 스킴을 이용하여 데이터베이스의 데이터를 리스트로 만들고 데이터베이스와의 연결을 중단한다. 사용자가 원하는 데이터베이스의 정보와 똑같은 데이터가 단순히 리스트의 형태로 바뀌게 되고, 사용자가 관리를 위한 추가 정보를 사용자가 입력하면 웹 서버가 이에 해당하는 리스트 연산을 통해 데이터를 수정한다.

웹 에이전트에서 데이터베이스에 XML 문서를 저장하는 과정은 <그림 6-3>과 같다.



<그림 6-3> XML 문서를 데이터베이스에 저장하는 과정

<Fig. 6-3> Save Process from XML Document to DataBase

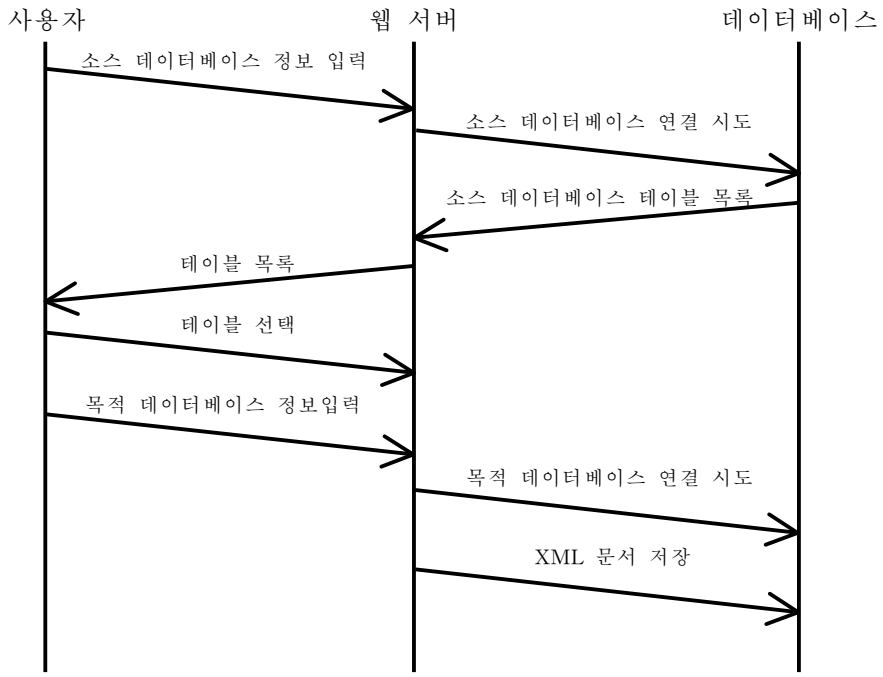
XML 문서를 데이터베이스에 저장할 경우, 사용자가 웹 브라우저 화면에 XML 문서의 위치에 대한 정보를 입력하면 웹 서버는 XML 문서에 접근하여 XML 문서를 파싱(parsing)하여 문서의 내용을 웹 브라우저를 통해 사용자가 보기 편한 테이블로 보여준다.

XML 문서를 데이터베이스에 저장할 수 있는 구조가 정해져 있는 테이블-기반 매핑을 사용하지 않고 본 논문에서 제안한 테이블-리스트 매핑을 사용하므로 저장하고자 하는 XML 문서를 데이터베이스에 단독 테이블로 저장할 수 있는 DTD 구조에 맞게 XML을 변환하는 과정은 웹 서버가 수행한다. 따라서 어떠한 구조의 XML 문서라도 데이터베이스에 저장할 수 있고, 또한 테이블 형태를 가지고 있는 HTML 문서도 XML 문서로 바꾸어 데이터베이스에 저장할 수 있게 된다.

사용자는 항상 웹 브라우저가 보여주는 화면의 내용을 추가, 삭제 및 수정 작업을 한 뒤, 사용자가 데이터베이스로 저장할지 아니면 XML 문서로 저장할지를 결정하면 저장장소에 대한 추가 정보를 입력하여 최종 저장 작업을 한다. 최종 저장을 데이터베이스에 할 경우에는 저장하고자 하는 데이터베이스에 연결하기 위한 정보와 저장할 테이블의 이름을 입력해야 하고, XML 문서로 저장할 경우에는 저장할 위치를 입력하면 된다.

데이터베이스로부터 데이터를 웹 서버로 가져오면 웹 서버는 데이터베이스에 저장되어 있는 데이터를 스킴 언어에서 사용하는 리스트 형태로 바꾼다. 웹 서버에서 사용자가 원하는 데이터를 검색하고 추출하거나 가공하는 것은 데이터는 데이터베이스에 저장된 데이터가 아니라 스킴의 리스트로 바꾼 것이다. 데이터를 검색하거나 추출하기 위해 어려운 데이터베이스 질의문이 아니라 웹 브라우저를 사용하는 이용자에게 익숙한 화면을 웹 서버가 웹 페이지로 제공하여 쉽게 데이터베이스의 데이터를 검색하거나 추출할 수 있다. 스킴의 리스트 표현은 여러 데이터를 묶어 하나로 취급하는 형태로써 리스트의 원소를 한꺼번에 수정하거나 조건을 만족하는 것만 골라내는 계산하는 것이 쉬울 뿐만 아니라 리스트 연산에 사용되는 기본 프로시저를 이용하여 복잡한 연산도 수행할 수 있다.

<그림 6-4>와 같이 서로 다른 데이터베이스의 데이터를 복제할 경우, 사용자는 복제하고자 하는 데이터를 가진 소스 데이터베이스에 대한 정보를 입력한다. 입력된 정보를 이용하여 웹 서버는 소스 데이터베이스와 연결한다. 소스 데이터베이스와 연결이 이루어지면, 사용자가 복제하고자 하는 테이블을 선택한다. 웹 서버는 선택된 테이블의 내용을 소스 데이터베이스로부터 전달받고, 그 내용을 XML 문서로 저장한다. XML 문서의 이름은 “테이블이름.xml”이고, 사용자가 지정한 위치에 저장된다.



<그림 6-4> 서로 다른 DBMS간의 데이터 복제

<Fig. 6-4> Data replication between heterogeneous DBMSs

사용자가 복제한 데이터를 저장하고자 하는 목적지 데이터베이스에 대한 정보를 입력하여 웹 서버로 전송하면, 웹 서버는 입력된 정보를 이용하여 데이터베이스와 연결한다. 웹 서버는 소스 데이터베이스에서 선택한 테이블과 같은 이름과 구조의 테이블을 생성하는 질의문을 완성하여 데이터베이스에게 전송하면 데이터베이스는 테이블을 생성한다.

사용자는 데이터베이스에 대한 간단한 정보를 입력하면 웹 서버를 통해 데이터베이스의 데이터를 가져와서 엑셀과 같은 데이터 처리 작업을 손쉽게 할 수 있고 또한 데이터베이스에서 데이터를 가져와야 하는 처음과 저장해야 하는 순간에만 데이터베이스에 직접 연결되므로 데이터베이스의 부하를 줄여준다. 또한 기존의 데이터베

이스를 관리하기 위해서 질의 언어나 데이터베이스에 관련된 전문적인 지식이 요구되었으나 웹 브라우저를 통해 데이터가 관리되므로 사용자는 데이터베이스를 관리하는 것이 아니라 웹 브라우저 화면에 나타나는 자료를 관리하는 것과 같이 생각되어 인터넷을 사용할 수 있는 초보자도 데이터베이스 관리가 가능하다.

본 논문에서 구현한 웹 에이전트는 인터넷에 연결되어 있는 컴퓨터에 사용자가 웹 서버에 접속하게 되면, 언제 어디서나 데이터베이스에 저장된 데이터를 손쉽게 관리할 수 있고 사용자가 원하는 정보를 추출하거나 조합하는 과정이 훨씬 간략화된 것이다.

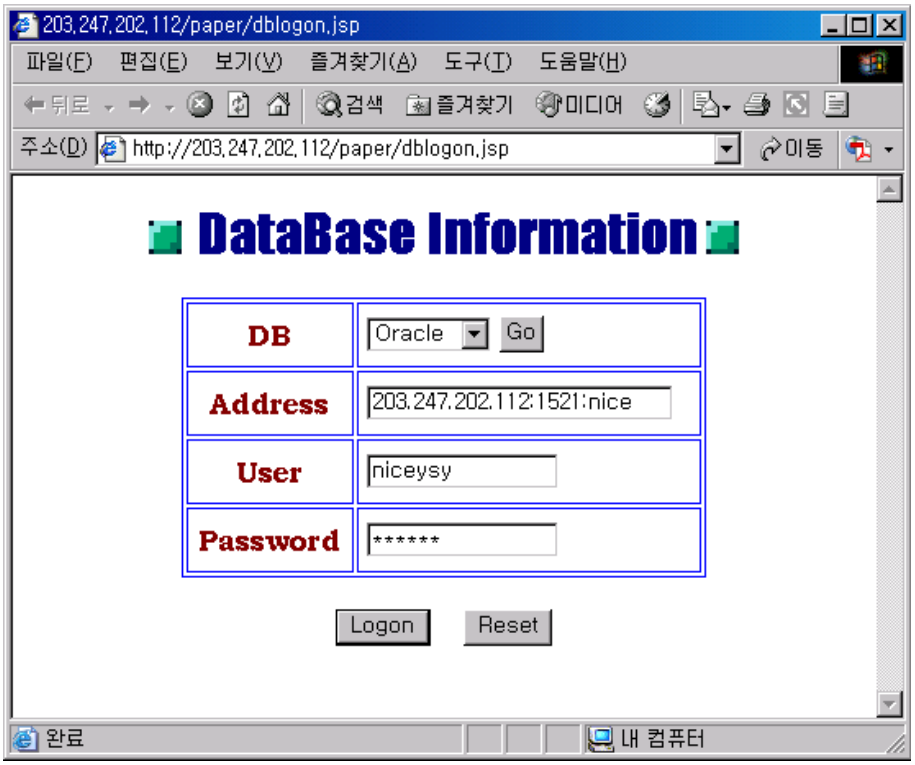
6.2 웹 에이전트 시스템 구현

앞에서 언급한 내용을 토대로 구현한 웹 에이전트 시스템은 사용자가 웹 서버를 통해 데이터베이스에 연결하고, 웹 브라우저 화면을 통해 데이터를 관리하거나 XML 문서로 변환하여 저장하고, 서로 다른 데이터베이스간의 데이터를 복제할 수 있도록 구현하였다.

6.2.1 데이터베이스와 연결

본 논문에서 구현한 웹 에이전트 시스템은 데이터베이스와 연결하기 위해서 원시 프로토콜(native protocol) 순수 자바 드라이버를 사용한다. 데이터베이스의 종류에 따라 드라이버의 이름이 다르므로 데이터베이스에 맞는 드라이버를 선택하기 위해 사용자가 데이터베이스를 선택할 수 있어야 한다. 웹 에이전트 시스템은 드라이버가 제공되는 모든 데이터베이스와 연결할 수 있다.

구현된 웹 에이전트 시스템은 사용자가 웹 서버에 접속하여 데이터베이스의 종류를 선택하면 해당 드라이버의 이름이 저장된 웹 페이지로 이동하여 데이터베이스의 종류에 상관없이 데이터를 관리할 수 있도록 하였다. 예를 들면, 데이터베이스의 종류를 오라클 데이터베이스로 설정하고, 연결하고자 하는 데이터베이스의 주소와 데이터베이스와 연결하는 사용자의 아이디(ID ; IDentification)와 암호(password)를 입력하는 화면은 <그림 6-5>와 같다.



<그림 6-5> 데이터베이스 로그온 웹 인터페이스

<Fig. 6-5> Web Interface of Database Logon

사용하는 아이디와 암호는 연결하고자 하는 데이터베이스에 대한 접근권한을 가지고 있어야 한다. <그림 6-5>에서 보면 연결하고자

하는 데이터베이스의 주소는 203.247.202.112이고, 연결 포트번호는 1521, 데이터베이스 이름은 nice이다.

<그림 6-5>와 같이 데이터베이스에 대한 정보를 사용자가 입력하여 웹 서버로 전송하면, 웹 서버는 입력된 정보를 이용하여 데이터베이스에 연결할 수 있는 질의문을 완성한다. 이러한 과정의 프로그램 소스는 <그림 6-6>과 같다.

```
String url = "jdbc:oracle:thin:@";
String user = null;
String pw = null;
String chk = "chk";
String list = "select * from tab";
url2 = request.getParameter("txtadd");
user= request.getParameter("txtid");
pw = request.getParameter("txtpw");
Connection conn=null;
Statement stmt=null;
ResultSet rs=null;

try{
    Class.forName("oracle.jdbc.driver.OracleDriver");
    conn=java.sql.DriverManager.getConnection(url1+url2,
            user, pw);
    stmt=conn.createStatement();
    rs=stmt.executeQuery(list);
}catch(java.sql.SQLException e){
     out.println(e);
}
```

<그림 6-6> 로그온 프로그램 소스

<Fig. 6-6> Logon Program Source

<그림 6-6>에서 나오는 “url” 변수는 접속하고자 하는 데이터베이스 종류에 따라 주소영역에 붙은 접두어이다. 예를 들면 사용자가 <그림 6-5>에서 오라클 데이터베이스를 선택하면 주소 영역에 붙는 접두어는 “jdbc:oracle:thin:@”가 된다. 또한, “list” 변수는 데이터베이스에 저장되어 있는 테이블의 목록을 가져오기 위한 질의문을 나타낸다.

<그림 6-5>에서 사용자가 로그온 버튼을 선택하면 텍스트 박스에서 입력된 정보를 이용하여 웹 서버는 <그림 6-6>의 try 문을 수행하여 웹 서버와 데이터베이스를 연결하고, 데이터베이스에 저장된 테이블 목록을 가져오는 질의문인 변수 “list”를 수행한 결과를 웹 서버로 전송한다.

드라이버의 이름은 데이터베이스의 종류에 따라 다르다. 오라클 데이터베이스인 경우는 “oracle.jdbc.driver.OracleDriver”이고, SQL 서버를 사용할 경우는 “com.inet.tds.TdsDriver”이다. <그림 6-6>에서 보면 try 구문 안에 들어있는 Class.forName 부분을 이용하여 원하는 드라이버를 로딩할 수 있도록 하였다.

데이터베이스 회사별로 제공하고 있는 드라이버를 사용하기 위해서는 각 데이터베이스 회사로부터 파일을 다운받아 JDK가 설치된 폴더(folder)에 저장하고 배치 파일을 수정해야 한다.

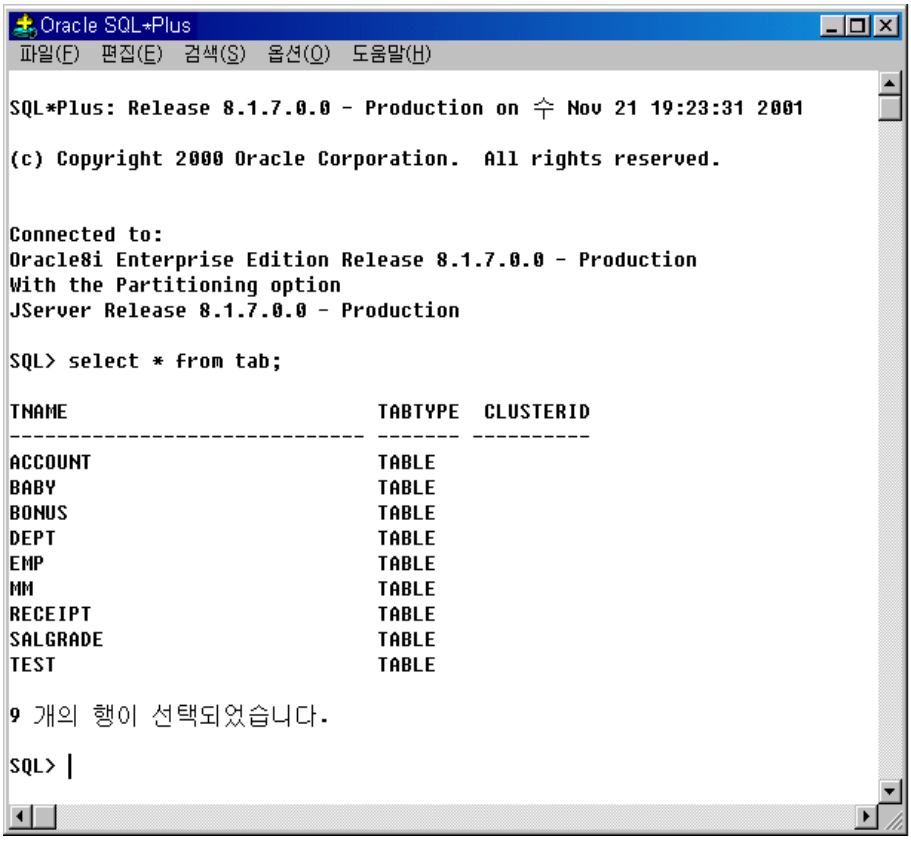
논문에서 구현한 웹 에이전트 시스템에서는 경로를 설정하기 위해서 배치 파일을 <그림 6-7>와 같이 설정하고, 드라이버를 사용하기 위해 “c:\jdk1.3\jre\lib\ext”에 저장한다.

실제 오라클 데이터베이스와 연결하여 테이블 목록을 알아본 결과는 <그림 6-8>와 같고, 웹 서버가 보여주는 화면은 <그림 6-9>과 같이 테이블 목록의 내용이 같다는 것을 확인할 수 있다.

```
set path=c:\jdk1.3\bin;
set classpath
=c:\orant\jdbc\lib\classes12.zip;c:\jdk1.3\jre\lib\ext\Sprinta2000.jar;
 c:\jdk1.3\jre\lib\ext\classes12.zip;c:\jdk1.3\jre\lib\ext\parser.jar;
 c:\jdk1.3\jre\lib\ext\jaxp.jar;c:\jdk1.3\jre\lib\ext\xerces.jar;
 c:\jdk1.3\jre\lib\ext\xercesSamples.jar;
```

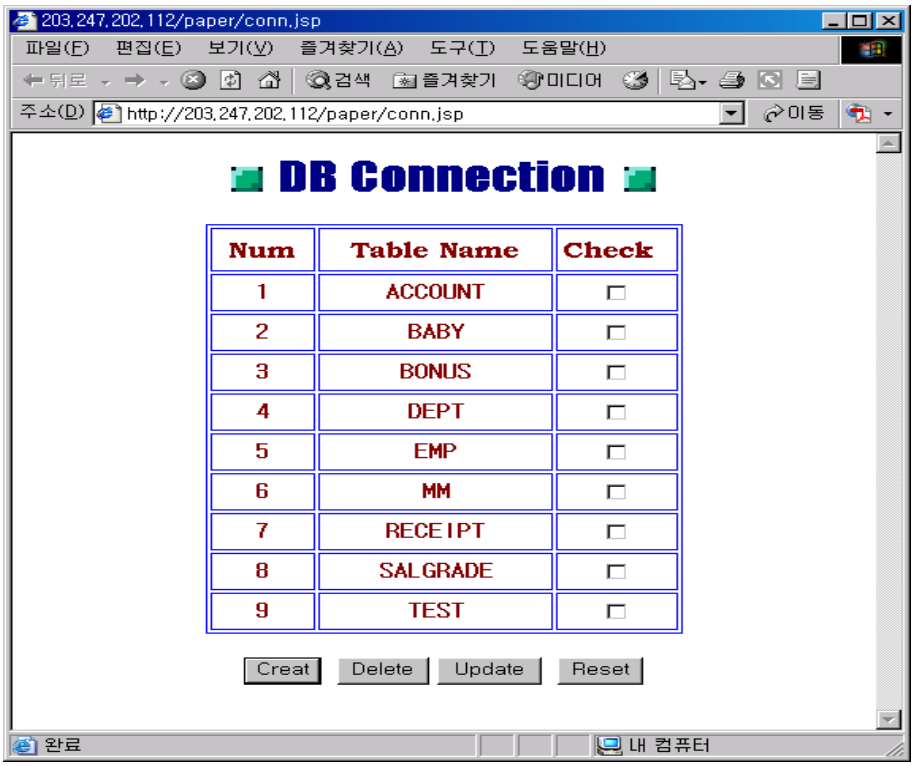
<그림 6-7> 자바와 JDBC 드라이버의 경로 설정

<Fig. 6-7> Path Configuration of Java and JDBC Driver



<그림 6-8> 오라클 데이터베이스의 테이블 목록

<Fig. 6-8> Table List of Oracle Database



<그림 6-9> 데이터베이스 연결 후 결과 화면

<Fig. 6-9> Result Window after Database Connection

6.2.2 데이터 처리

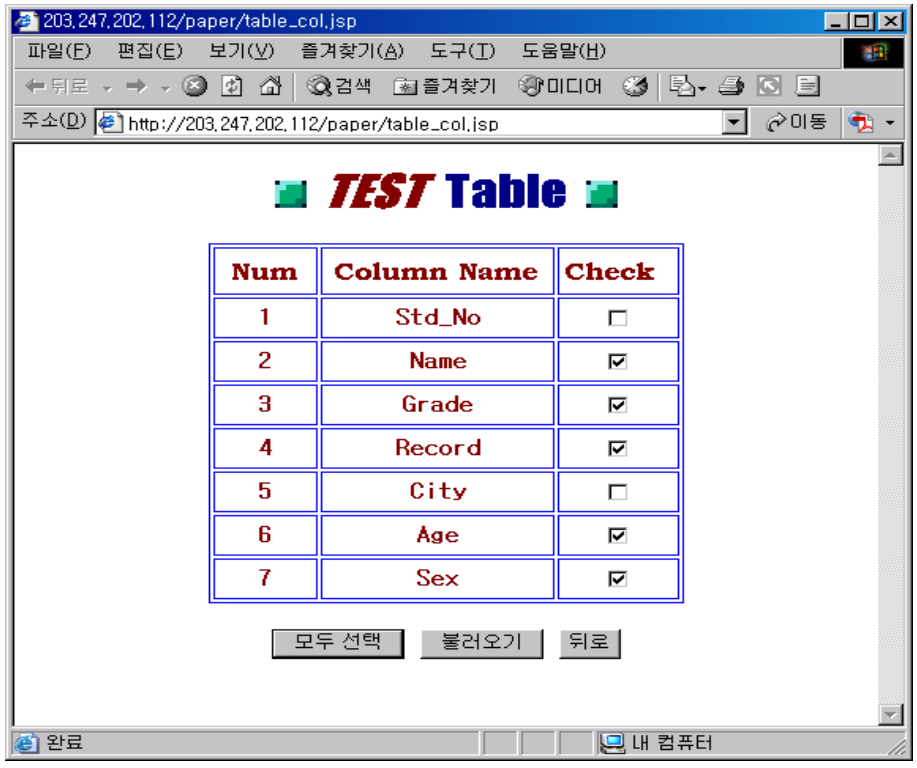
예를 들면, <그림 6-9>에서 TEST 테이블을 체크하여 Update 버튼을 선택하면 웹 서버는 데이터베이스에 연결된 상태에서 TEST 테이블의 컬럼명을 보여주고 사용자가 보고자 하는 컬럼을 체크하여 선택적으로 테이블의 내용을 가져올 수 있다.

TEST 테이블에 저장되어 있는 내용은 <표 6-1>과 같고 <그림 6-9>에서 TEST 테이블을 선택하여 Update 버튼을 눌렀을 때 웹 브라우저의 화면은 <그림 6-10>과 같다.

<표 6-1> TEST 테이블 데이터

<Table 6-1> TEST Table Data

Std_No	Name	Grade	Record	City	Age	Sex
2000111	young	3	92	Busan	26	Female
2001002	jong	2	75	Incheon	30	Male
1999012	hun	4	94	Busan	29	Male
2002010	woo	2	88	Seoul	22	Female
2002008	nam	1	60	Busan	22	Male
2003026	soo	2	54	Incheon	21	Female
2004012	nimda	1	76	Busan	25	Male
1998075	moyh	4	82	Seoul	26	Female
2001039	min	2	74	Incheon	28	Female
2004024	kids	1	65	Busan	20	Male



<그림 6-10> TEST 테이블 컬럼 목록

<Fig. 6-10> Column List of TEST Table

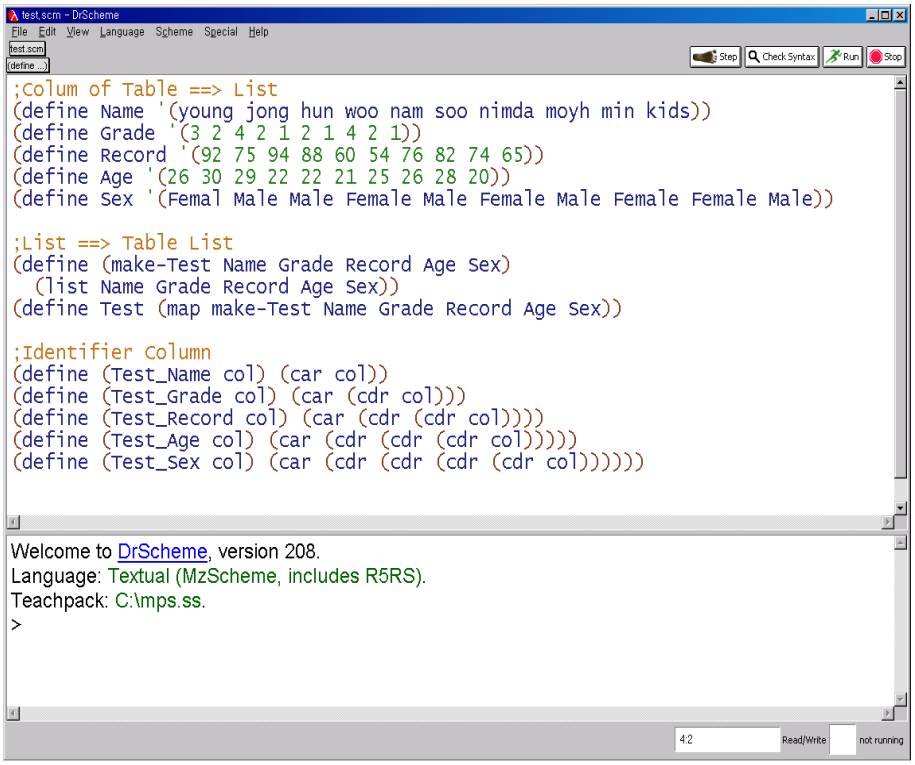
<그림 6-10>과 같이 TEST 테이블의 컬럼 이름과 사용자가 컬럼을 선택할 수 있는 체크 박스가 함께 보인다. <표 6-1>과 <그림 6-10>을 비교해 보면 실제 테이블에 저장되어 있는 컬럼 이름과 수가 같다는 것을 알 수 있다.

예를 들면, 사용자가 TEST 테이블의 컬럼 중에서 Std_No, City를 제외한 나머지를 체크하여 “불러오기” 버튼을 누르면 웹 서버는 데이터베이스에게 선택된 컬럼의 내용만 불러오는 질의문을 전송하여 데이터베이스로부터 선택된 컬럼의 데이터만 불러온다. 데이터베이스로부터 가져온 데이터를 웹 서버는 스킴의 리스트로 저장하고 데이터베이스와의 연결을 중단하고 웹 서버는 리스트를 이용하여 사용자와 데이터를 처리하게 된다. 그러나 웹 브라우저 화면을 보게 되는 사용자는 그 차이를 알 수 없다.

사용자가 선택한 컬럼을 스킴의 리스트 형태로 웹 서버가 “테이블이름.scm”으로 저장한 파일은 <그림 6-11>과 같고 웹 서버가 웹 브라우저로 보여주는 화면은 <그림 6-12>와 같다.

<그림 6-12>에서 수정 버튼을 누르면 웹 브라우저 화면에서 테이블의 데이터가 데이터를 입력할 수 있는 입력상자 형태로 바뀐 화면이 나타나서 테이블의 내용을 수정할 수 있다. DB 저장과 XML 저장 버튼은 웹 브라우저 화면에 보이는 테이블 전체의 내용을 저장하기 위해 필요한 추가 정보를 입력하는 화면이 나타난다.

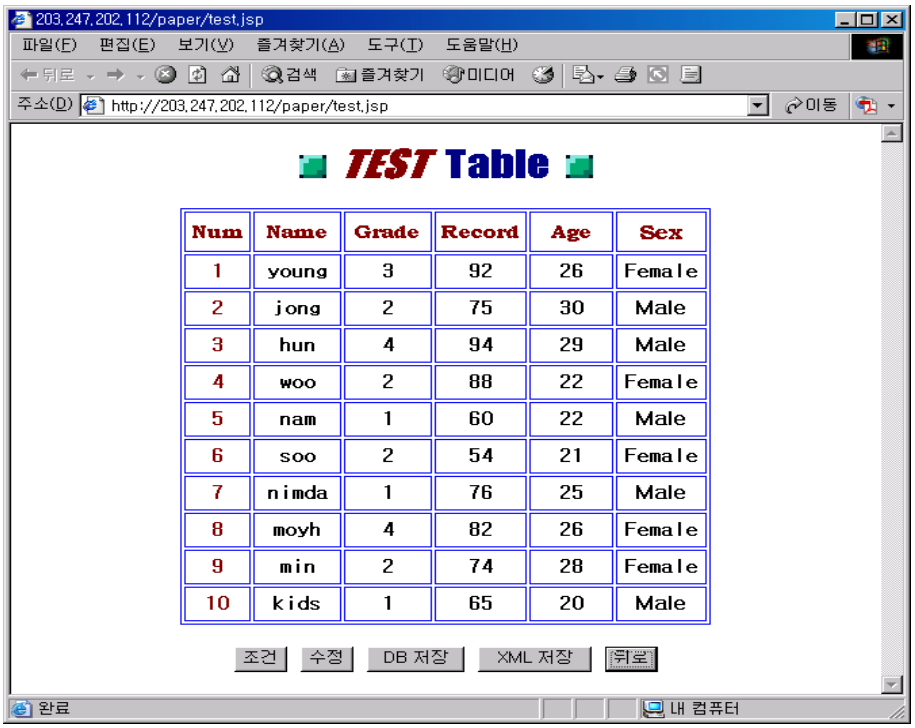
데이터베이스에 저장할 경우에는 저장하고자 데이터베이스에 연결하기 위한 정보와 테이블의 이름을 입력해야 하고, XML 문서로 저장할 경우에는 저장할 위치와 저장할 파일의 이름을 입력하는 화면이 나타난다.



<그림 6-11> test.scm 파일 소스

<Fig. 6-11> File test.scm Source

조건 버튼은 테이블의 내용을 수정하지 않고 데이터 내용을 분석하여 원하는 정보를 찾거나 계산하기 위해 추가 정보를 입력할 수 있는 화면이 나타난다. 웹 서버가 만든 “test.scm” 파일의 내용은 사용자가 선택한 테이블의 내용에 따라 “Name”과 같은 리스트 이름과 리스트 원소가 바뀔 뿐 파일의 구조는 비슷하다. Dr.Scheme에서 “test.scm” 파일을 이용하여 계산할 수 있는 다양한 문제는 <표 6-2>와 같고, Dr.Scheme에서 실행한 결과는 리스트 형태나 값으로 보여준다.



Num	Name	Grade	Record	Age	Sex
1	young	3	92	26	Female
2	jong	2	75	30	Male
3	hun	4	94	29	Male
4	woo	2	88	22	Female
5	nam	1	60	22	Male
6	soo	2	54	21	Female
7	nimda	1	76	25	Male
8	moyh	4	82	26	Female
9	min	2	74	28	Female
10	kids	1	65	20	Male

<그림 6-12> TEST 테이블 데이터 목록

<Fig. 6-12> List of TEST Table Record

<표 6-2> 데이터 검색을 위한 문제

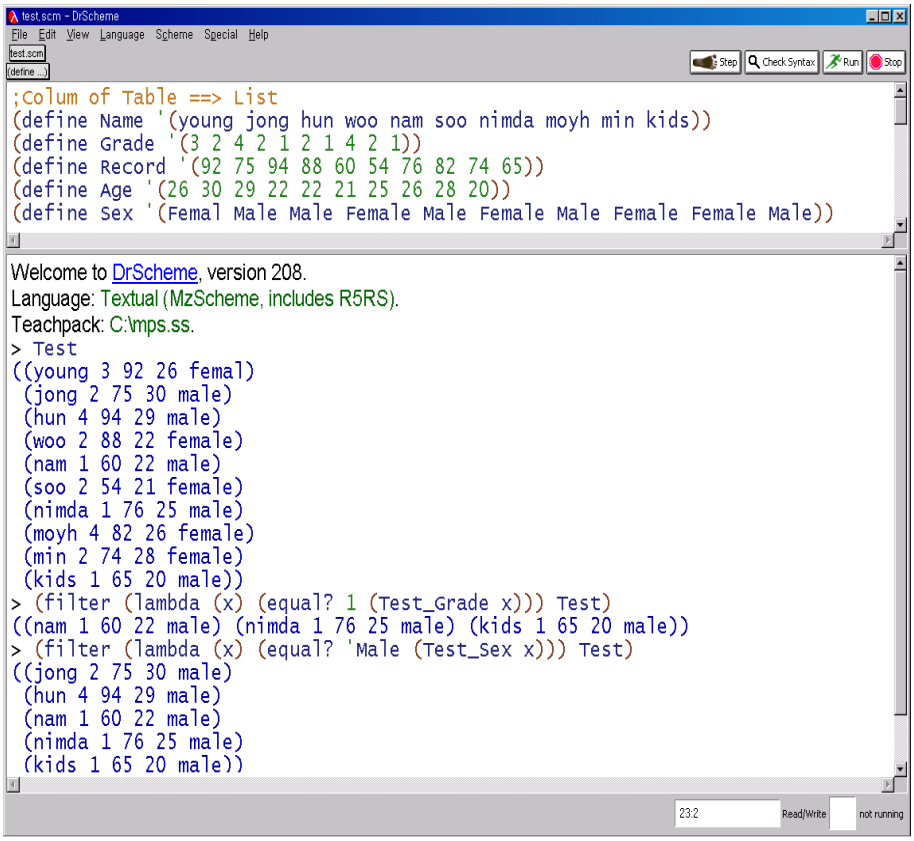
<Table 6-2> Example for Data Search

① 1학년 학생에 대한 정보 보기
② 남학생에 대한 정보 보기
③ 전체 학생의 성적을 5로 나누기
④ 전체 학생의 평균 점수 구하기
⑤ 남학생 중 1학년 학생의 평균 점수 구하기

본 논문에서 구현한 웹 에이전트 시스템에는 Dr.Scheme의 계산 결과를 웹 브라우저 화면에 보여주도록 만들어주는 부분이 추가되어 있어 사용자는 번거로움 없이 데이터를 처리할 수 있다.

<표 6-1>과 같은 데이터가 데이터베이스에 테이블 형태로 저장되어 있는 테이블에서 <표 6-2>와 같은 문제를 푸는 것은 대량의 데이터에서 사용자가 원하는 정보를 골라내거나 추출하여 가공하는 것과 같다. 구현한 웹 에이전트 시스템에서 정보를 골라내거나 추출하여 가공하는 기능은 데이터베이스에 저장된 데이터를 스킴 언어의 리스트로 표현하여, 리스트에서 정보를 골라내거나 추출하여 가공할 수 있는 map, filter, fold-right, fold-left 등을 이용한다.

웹 서버가 만든 TEST 테이블의 내용과 같은 Test 리스트와 ①과 ②의 조건을 만족하는 filter 식과 결과는 <그림 6-13>과 같다.



<그림 6-13> 리스트를 이용한 풀이 ①, ②

<Fig. 6-13> Solution ①, ② using List

<표 6-2>에서 ①과 ②는 조건을 만족하는 학생에 대한 정보만 보고자 하는 경우이다. 특정 조건을 만족하는 정보만 추출하고자 할 경우에 해당하는 문제이다.

①의 조건은 학년이 “1”과 같은 경우이고 ②는 성별이 “Male”과 같은 경우이다. 스킴에서 조건을 만족하는 경우의 정보만 골라 낼 경우 filter를 사용하여 식을 완성한다. <그림 6-13>과 같이 ①은 비교 대상이 숫자이기 때문에 “=”를 사용한다.

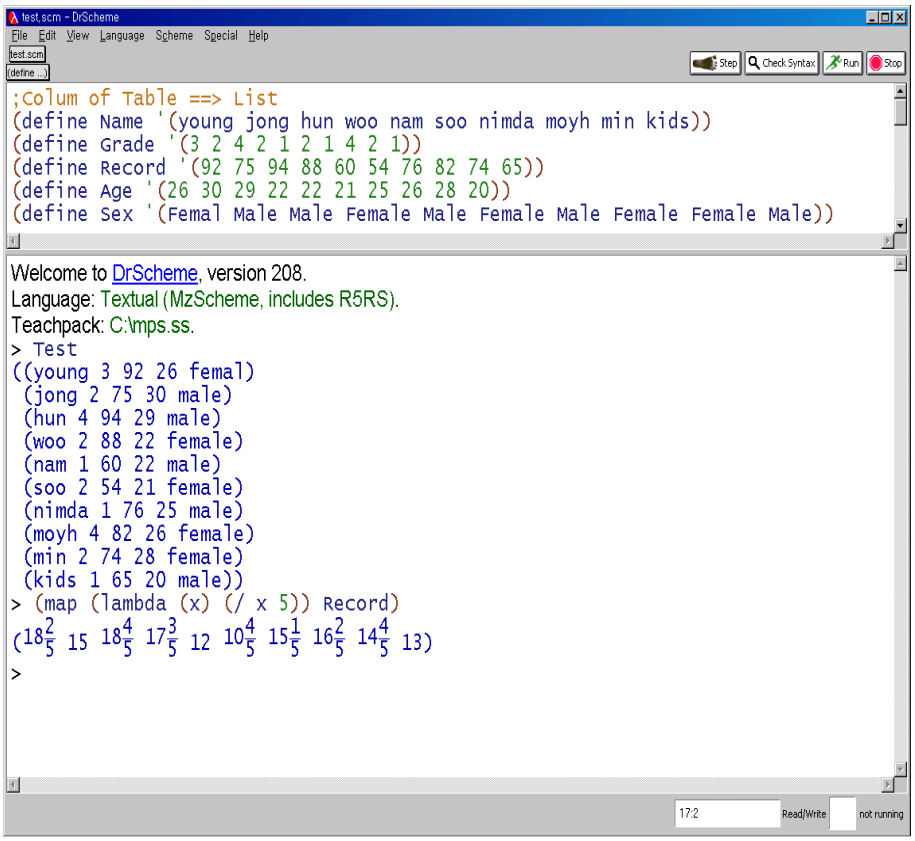
②는 비교 대상이 문자이기 때문에 “eq?”를 사용해야 한다. 그러나 스킴에서 비교 대상이 숫자이든 문자든 리스트든 구별하지 않고 비교해 주는 “equal?”이 있으므로 본 논문에서는 비교 대상에 따라 연산자 모양이 바뀌는 번거로움을 없애기 위해 “equal?”로 고정하여 사용한다.

③은 Record 리스트 원소를 5로 나누는 똑같은 계산을 반복하는 경우이다. 이와 유사한 형태의 경우는 년도가 바뀔 때마다 학년이 1만큼 증가하거나 나이가 한 살 많아지는 것이다. 실제 데이터베이스의 내용을 똑같은 계산 방법을 적용하여 한꺼번에 처리하는 것과 같은 형태이다.

리스트 원소에 똑같은 계산 방법을 적용하는 map을 이용하여 완성한 식은 <그림 6-14>와 같다.

④는 리스트 원소의 평균을 구하는 문제로 가장 높은 점수나 가장 낮은 점수를 구하거나 Grade 리스트 원소를 내림차순, 오름차순으로 정렬하는 문제를 푸는 것과 같은 방법으로 풀 수 있다.

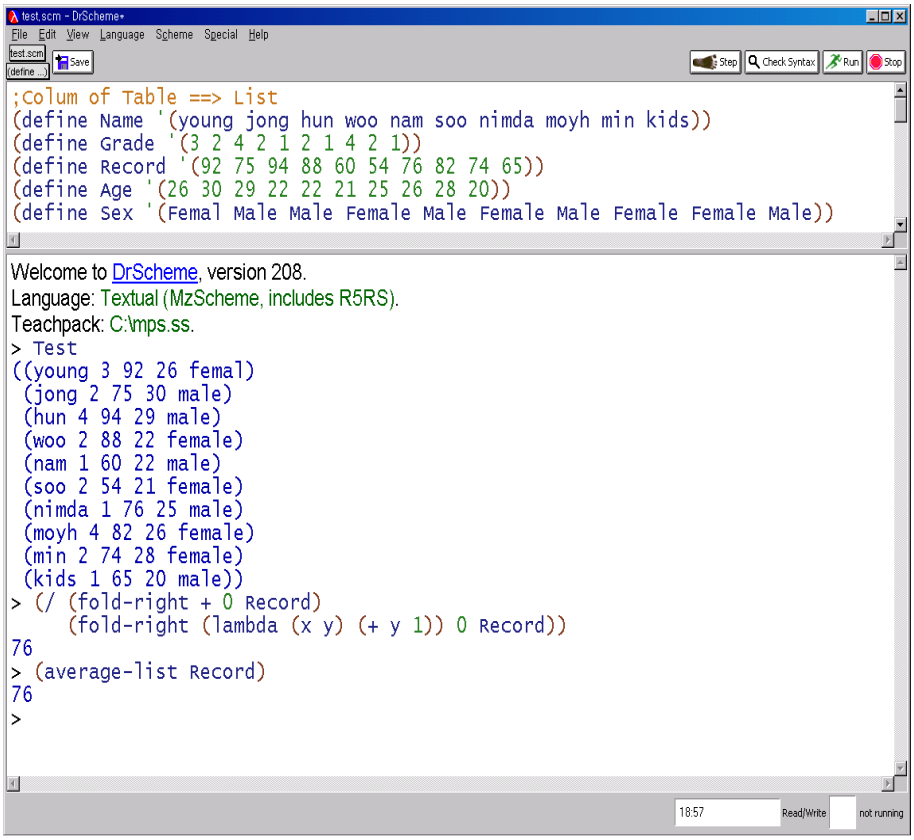
리스트 원소끼리 계산하는 fold-right나 fold-left를 이용하여 식을 완성하여 실행한 화면은 <그림 6-15>와 같다.



<그림 6-14> 리스트를 이용한 풀이 ③

<Fig. 6-14> Solution ③ using List

본 논문에서 구현한 웹 에이전트 시스템에서는 사용자가 자주 계산하는 형태인 평균, 제곱, 세제곱, 최대값, 최소값 등과 같은 계산 방법을 엑셀에서 함수를 이용하여 계산하는 것과 비슷하게 웹 브라우저 화면에 선택 상자에서 골라서 쓸 수 있게 구현하였다. 자주 사용되는 계산 방법을 정의한 프로시저를 Teachpack 파일인 “mps.ss” 파일에 추가하여 웹 브라우저를 통해 사용자가 선택한 계산 방법에 해당하는 프로시저가 짝을 이루도록 하였다.



<그림 6-15> 리스트를 이용한 풀이 ④

<Fig. 6-15> Solution ④ using List

예를 들면, 그림과 같이 average-list 프로시저는 리스트의 평균을 구하는 프로시저로 사용자가 웹 브라우저에서 계산 방법을 평균으로 선택하면 웹 서버는 average-list 프로시저를 호출하여 계산하도록 설정되어 있다. 시간이 지남에 따라 사용자가 요구하는 계산 방법이 다양해지더라도 계산 방법을 정의한 프로시저만 작성하여 Teachpack 파일에 추가하면 되므로 본 논문에서 구현한 웹 에이전트는 확장성이 좋다고 할 수 있다.

끝으로, 마지막 ⑤의 경우는 복잡한 계산 방법으로 여러 개의 조건을 모두 만족시키는 것을 골라서 계산하는 형태이다. 이러한 계산

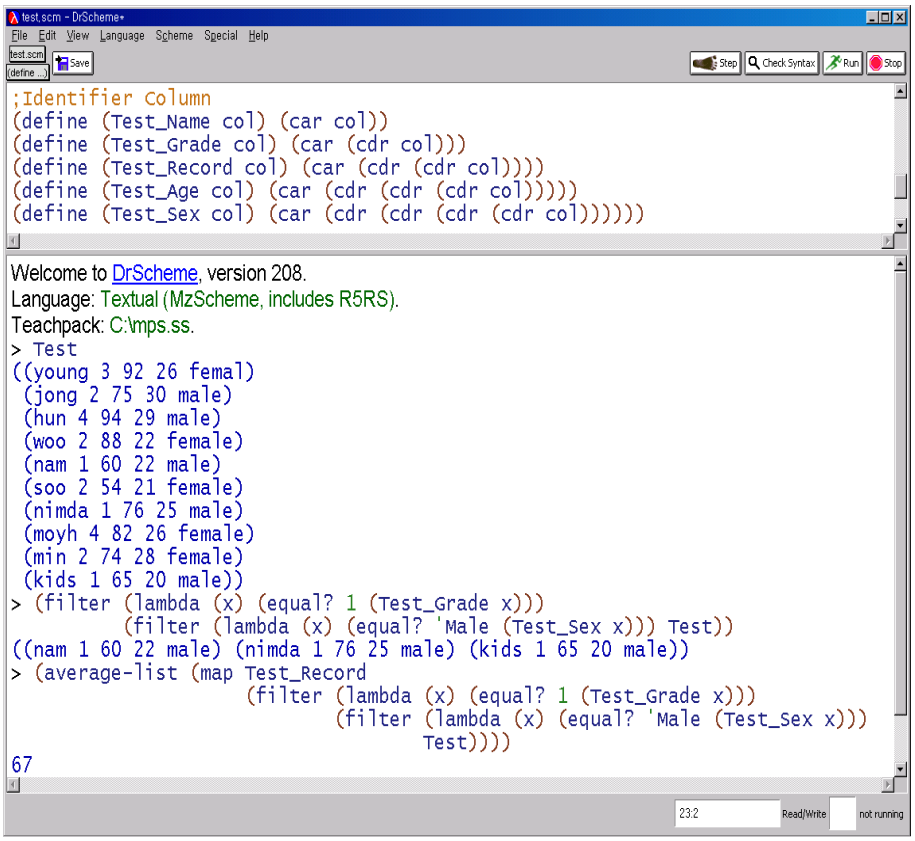
방법을 데이터베이스에 질의하는 질의문을 통해 답을 구하기는 어려운 질문 중 하나이다. ⑤는 전체 학생 중에서 성별이 남자이면서 1학년인 학생만 골라내어 성적을 평균하는 문제이다. 두 조건을 동시에 만족시키는 학생을 구하기 위해 식을 세운다면 조건이 많아질수록 식을 세우기가 쉽지 않다.

본 논문에서는 동시에 조건을 만족하는 것이 아니라 문제를 하나씩 쪼개어 조건 하나를 만족하는 결과를 구하고, 구한 결과에서 그 다음 조건을 따져서 골라내는 형태로 문제를 풀이하고자 한다. 이렇게 하면 조건이 아무리 많아도 계산 과정은 조건 하나를 만족하는 식을 여러 번 겹쳐 쓰는 형태가 되므로 식이 복잡해 보여도 계산 과정은 같은 형태의 계산을 여러 번하는 것이 되어 계산 과정은 단순해진다.

⑤는 전체 학생 중에서 첫째, 성별이 남자인 학생만 골라내고 둘째, 이 중에서 1학년 학생을 골라내고 끝으로 이들의 성적을 평균하면 되는 것이다. 첫째와 둘째 식은 filter를 이용하고, 성적 평균을 구하기 위해서는 fold-right를 이용하면 된다. 이를 실행한 화면은 <그림 6-16>과 같다.

<그림 6-16>에서 완성된 식이 복잡해 보이지만 average-list, map, filter, equal?, lambda와 같은 표현식은 고정되어 있고 1, Male, Test_Record, Test_Grade, Test_Sex만 사용자의 선택에 따라 바뀌는 부분이다.

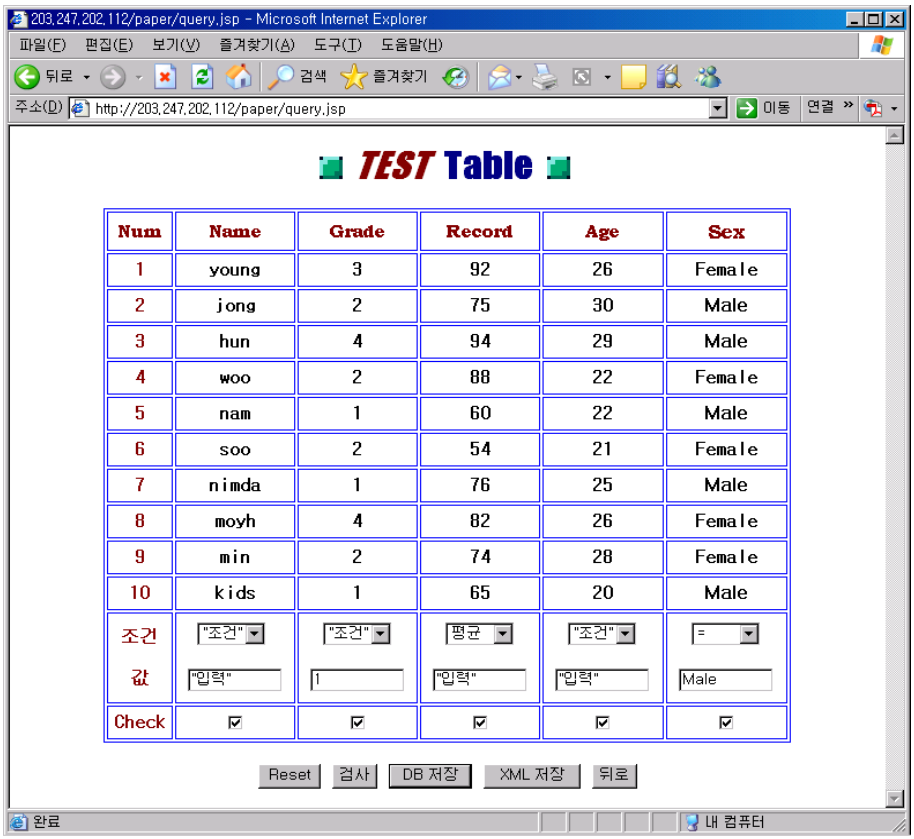
지금까지 스킴을 이용하여 데이터를 처리하는 Dr.Scheme의 실행 결과와 식을 살펴보았다. 이는 웹 서버가 실행하는 부분이고 사용자는 단지 웹 브라우저 화면을 통해서만 데이터를 처리하므로 사용자에게 익숙한 화면을 제공하는 것이 중요하다. 앞에서 살펴본 것과 같은 다양한 계산을 웹 에이전트 시스템에서는 <그림 6-17>과 같이 하나의 웹 브라우저 화면으로 처리하였다.



<그림 6-16> 리스트를 이용한 풀이 ⑤

<Fig. 6-16> Solution ⑤ using List

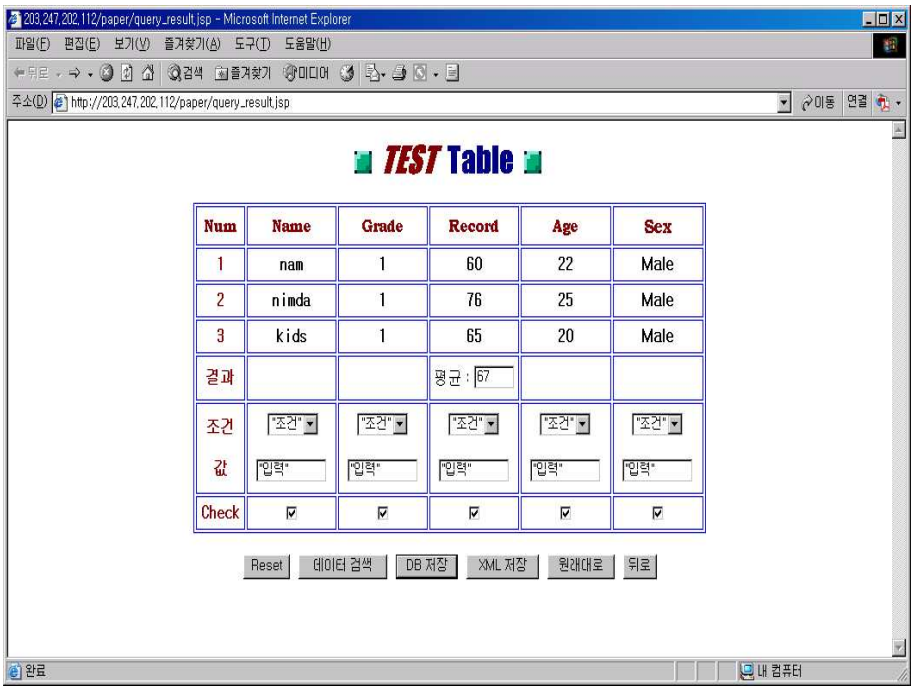
<그림 6-17>과 같이 조건과 값을 사용자가 선택하고 입력할 수 있고 사용자가 결과로 보고 싶은 컬럼도 선택할 수 있다. 앞에서 예를 든 ①은 Grade 컬럼의 조건이 "="이고 값이 1이 되고 Check는 컬럼 전부를 선택하는 것과 같고, ②는 Sex 컬럼의 조건이 "="이고 값이 "Male"이 되고 컬럼 전체를 선택한 것이고, ③의 경우는 Record 컬럼의 조건이 "/"이 되고 값이 5가 되고 Record 컬럼만 선택하여 검사한다.



<그림 6-17> 데이터 검색을 위한 화면

<Fig. 6-17> Window for Data Searching

④의 경우에는 Record의 조건이 “평균”이 되고 값은 입력하지 않고 Default 값인 “입력”을 그대로 두고 Record 컬럼만 선택하여 결과를 확인하고, ⑤는 Grade 컬럼의 조건은 “=”, 값은 1로 하고 Sex 컬럼의 조건은 “=”, 값은 Male로 하고 Record 컬럼의 조건은 평균으로 하고 값을 입력하지 않고 모든 컬럼을 선택해서 Run 버튼을 누르면 결과를 웹 브라우저 화면은 <그림 6-18>과 같다.



<그림 6-18> 데이터 검색 결과 화면

<Fig. 6-18> Window for Data Searching Result

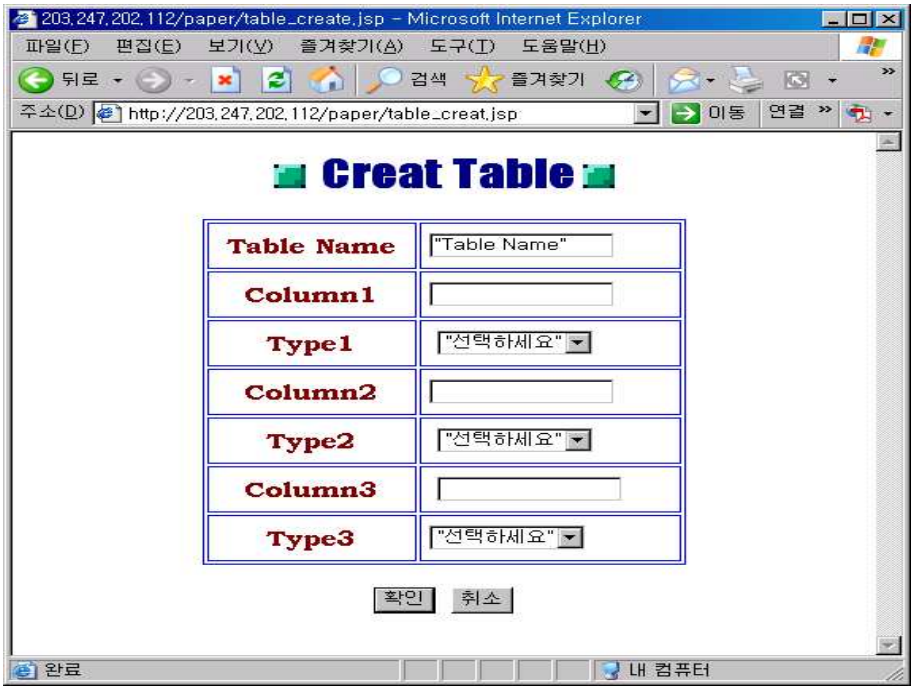
⑤와 같이 조건이 여러 개인 경우 우선순위를 결정하는 규칙은, 각 컬럼의 조건을 만족하는 개개의 식을 실행한 결과가 리스트 형태인 경우에는 어느 조건을 먼저 실행하든지 상관없으나 실행한 결과의 값이 리스트의 형태로 나온 경우에는 반드시 결과가 리스트로 나오는 식을 먼저 실행하는 것이다.

6.2.3 데이터베이스 관리

본 논문에서 구현한 웹 에이전트 시스템은 사용자가 테이블의 생성과 삭제 및 수정, 특정 테이블을 다른 데이터베이스로 저장하는 기능도 수행할 수 있다.

<그림 6-9>에서 사용자가 Create, Delete 버튼을 선택한 경우 사용자는 추가 정보를 입력해야 하고 웹 서버는 입력된 정보를 이용하여 새로운 질의문을 완성하여 데이터베이스로 전송한다.

Create 버튼은 테이블을 생성하는 버튼으로서 생성하고자 하는 테이블 이름, 컬럼 이름과 같은 추가 정보가 필요하다. 그 중에서 제일 먼저 필요한 정보가 컬럼 수이므로 Create 버튼을 누르면 컬럼 수를 묻는 메시지 상자가 나타나 사용자로 하여금 컬럼 수를 입력하게 한다. 웹 서버는 입력된 컬럼 수를 이용하여 <그림 6-19>과 같이 해당 컬럼 수에 해당하는 컬럼 이름을 적을 수 있는 텍스트 상자와 컬럼에 입력되는 데이터 타입을 선택할 수 있는 선택 상자가 화면에 나타난다. <그림 6-19>는 사용자가 컬럼 수를 3으로 하였을 경우이다.



<그림 6-19> 테이블 생성 화면

<Fig. 6-19> Table create window

사용자는 생성하고자 하는 테이블의 이름과 컬럼 이름 및 데이터 타입을 입력하고 웹 서버는 입력된 정보를 이용하여 새로운 테이블을 생성한다. 웹 서버는 추가 정보를 웹 브라우저를 통해 입력받아 테이블을 생성하는 질의문을 완성한다. 완성된 질의문을 웹 서버는 데이터베이스에게 전송하여 테이블을 생성하고, 그 결과를 웹 서버에게 되돌려준다. 웹 서버는 추가된 테이블 이름을 포함한 테이블 목록 화면을 웹 브라우저를 통해 사용자에게 보여준다.

테이블을 생성하기 위한 질의문을 만드는 과정의 프로그램 소스는 <그림 6-20>과 같다.

```
    for(j=1; j<= n; j++){
        String cj = c + j;
        String vj = v + j;
        col = request.getParameter(cj);
        val = request.getParameter(vj);
        total[j] = col + " " + val;
    }
    for(j=1; j<= n; j++){
         sum += total[j];
           if( j < n){
               sum = sum + "," + " ";}
           else{
               sum = sum;
           }
    }
String name = null;
name = request.getParameter("tbnm");
String createqy = "create table "+name+"(" + sum + ")";
```

<그림 6-20> 테이블 생성 프로그램 소스

<Fig. 6-20> Table Create Program Source

프로그램 소스에서 for 문은 <그림 6-19>에서 사용자가 입력한 테이블 이름을 “name”이라는 변수에 저장하고, 컬럼 이름과 데이터 타입을 조합하여 “sum”이라는 변수에 저장한다. 테이블을 생성하기 위해 “createqy”라는 변수에 “name”과 “sum”을 조합하여 질의문을 완성하는 과정을 나타내고 있다.

테이블을 삭제하는 프로그램 소스는 <그림 6-21>과 같다.

```
int checklen = 0;
checklen = checks.length;
for(int i=0;i<checklen;i++){
    num = checks[i];
    int j = Integer.parseInt(num);
    table = nm[j-1];
}

Connection conn2=null;
Statement stmt2=null;
String del =“drop table ” + table;
try{
    Class.forName(“oracle.jdbc.driver.OracleDriver”);
    conn2=java.sql.DriverManager.getConnection(url1+url2,user, pw);
    stmt2=conn2.createStatement();
    stmt2.executeUpdate(del);
}
catch(java.sql.SQLException e){
    out.println(e);
}
     conn2.close();
```

<그림 6-21> 테이블 삭제 프로그램 소스

<Fig. 6-21> Table Delete Program Source

테이블을 삭제하는 Delete 버튼은 추가 정보를 입력하지 않고 사용자가 선택한 테이블을 데이터베이스에서 삭제하는 기능을 수행한다. 테이블을 삭제하기 위한 질의문을 완성하여 테이블을 삭제하고, 웹 서버는 테이블 목록에서 삭제된 테이블의 이름이 빠진 테이블 목록을 사용자에게 보여준다.

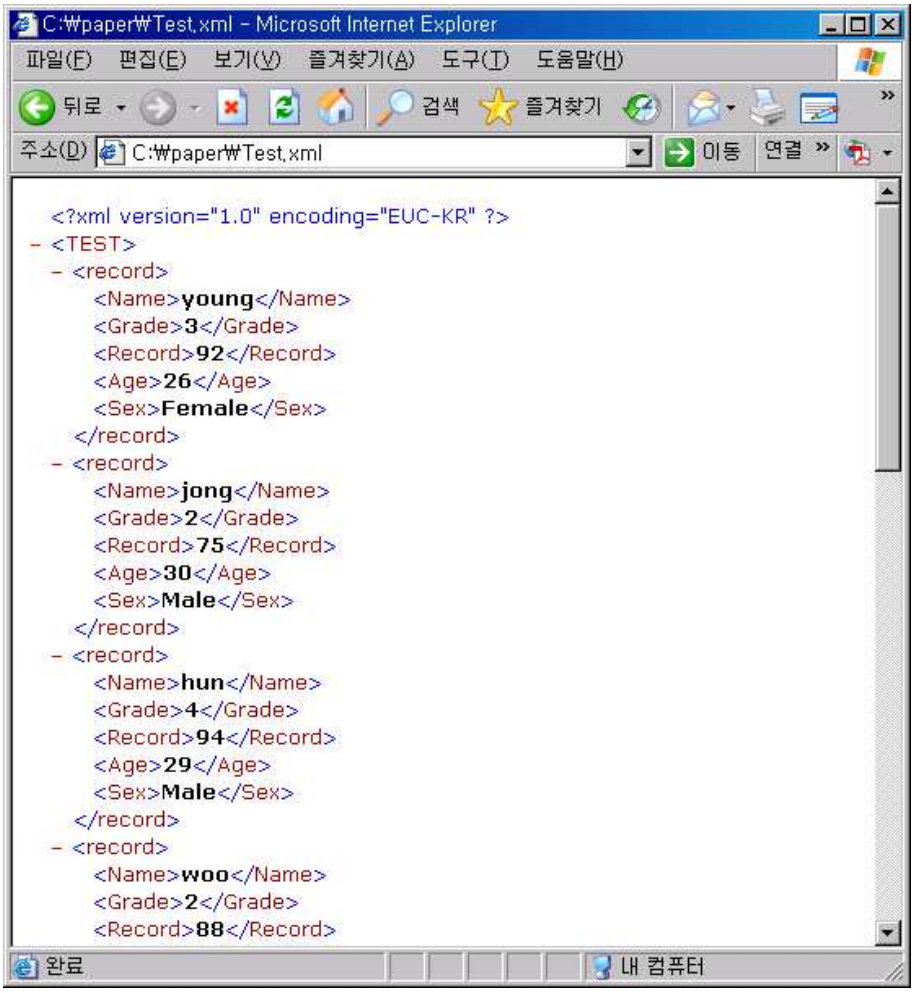
6.2.4 XML 문서를 저장할 경우

XML 문서를 사용자가 데이터베이스에 저장하는 과정을 살펴보면 다음과 같다. 사용자는 저장해야 하는 XML 문서에 대한 위치 정보를 입력하고 웹 서버는 입력된 위치에 접근하여 데이터베이스에 저장된 데이터를 스킴의 리스트 형태로 저장하고, 저장된 리스트 내용을 XML 문서로 만든다. 웹 서버는 XML 문서에 접근하여 DTD를 생성한 뒤 XML 문서의 내용을 데이터베이스의 테이블 내용을 보여주는 화면과 같이 테이블 형태의 웹 브라우저 화면으로 보여준다.

웹 브라우저 화면에서 XML로 저장 버튼을 선택하게 되면 웹 브라우저 화면에 보이는 형태와 같은 XML 문서를 만들어 사용자가 원하는 폴더에 저장할 수 있다. 예를 들면, 앞에서 사용한 TEST 테이블을 XML 문서로 저장하는 경우 <그림 6-22>와 같다.

테이블의 이름인 TEST가 XML 문서의 루트 태그가 되고, 테이블은 여러 행이 모여 만들어지므로 모든 테이블은 루트 태그 밑에 record 태그를 여러 개 가질 수 있다. record 태그는 Name, Grade, Record, Age, Sex를 자식 태그로 가진다. 데이터베이스의 테이블에 따라 루트 태그 이름과 record 태그의 자식 태그 이름이 바뀌지만 구조는 동일하다. TEST 테이블은 단독 테이블이므로 속성값은 정의되지 않고 단순한 요소만 가진 구조로 되어 있다.

데이터베이스의 데이터 관리와 유사하게 XML 문서의 내용을 수정 버튼을 이용하여 추가와 삭제, 수정할 수 있다. 또한 수정 과정을 거치지 않고 XML 문서의 내용을 원하는 데이터베이스에 저장할 수도 있다. 이는 웹에 존재하는 XML 형식의 웹 페이지를 데이터베이스에 저장하여 관리함으로써 정보의 재사용과 정보 공유를 더욱 원활하게 한다.



<그림 6-22> Test.xml 문서

<Fig. 6-22> Test.xml Document

6.3 고찰

현재 XML과 연동하여 데이터베이스를 관리할 수 있는 상용 제품은 많이 있다. 그 중에서 펜타 시스템의 타미노(Tamino)와 오라클 제품은 본 논문에서 구현한 웹 에이전트 시스템과 유사한 기능을 제공하지만 웹 에이전트 시스템은 기존의 제품이 갖는 한계를 극복하였다[49]~[51].

XML 문서를 XML 포맷으로 저장하고 관리하는 타미노는 클라이언트와 서버간의 연결을 위하여 별도의 소프트웨어와 라이센스를 구매해야 하고 데이터베이스 데이터를 교환할 수 있는 오라클 제품은 오라클에서 제작한 XSU(XML SQL Utility)를 가진 오라클 데이터베이스와 다른 데이터베이스간의 정보 교환만 가능하다. 타미노는 데이터 관리를 XML 데이터를 관리하는 별도의 소프트웨어에서 사용하는 명령어와 인터페이스에 대한 교육이 필요하고, 오라클은 데이터베이스 명령어를 이용하여 데이터를 관리하기 때문에 일반 사용자가 데이터베이스를 관리하는 경우에는 어렵다.

본 논문에서 구현한 웹 에이전트 시스템은 별도의 소프트웨어가 없이 인터넷에 연결되어 있는 웹 브라우저를 사용한다는 점이 타미노와 다르다. 오라클 데이터베이스에 저장된 데이터만을 다른 데이터베이스로 전달할 수는 있는 오라클 제품과는 달리 웹 서버가 가지고 있는 JDBC의 드라이버에 의해 연결 가능한 데이터베이스가 결정되므로 모든 데이터베이스에 대한 JDBC가 개발된다면 웹 서버는 모든 데이터베이스간의 정보 교환이 가능하게 된다. 또한, 스킴을 이용하여 데이터베이스를 관리하고 사용자가 원하는 다양한 검색 조건을 데이터베이스 질의문이 아닌 웹 브라우저 화면에서 조건을 입력하거나 선택하여 다양한 데이터 처리를 쉽게 할 수 있다.

제 7 장 결 론

오늘날 지식과 정보 교류의 기반이 웹으로 옮겨지면서 웹을 이용하여 정보를 공유하거나 교환하려는 사용자의 요구 조건이 다양하고 복잡해지고 있다. 웹에서 데이터베이스를 관리하는 기업이 늘어나면서 웹 문서를 데이터베이스에 저장하거나 데이터베이스에 저장되어 있는 내용을 웹 문서로 표현하고, 서로 다른 데이터베이스에 저장되어 있는 내용을 웹을 통해 데이터 교환이 빈번하게 이루어지고 있다. 기업이 사용하고 있는 데이터베이스 관리 시스템에서 저장된 데이터를 웹에서 교환하기 위해서는 문서 구조에 대한 정보를 가지고 있는 XML이 더 적합하다.

본 논문은 데이터베이스에 저장된 데이터를 XML 문서로 변환시키기 위한 DTD 생성 규칙과 데이터베이스에 저장된 데이터를 스킴을 이용하여 데이터를 검색하고 추출하기 위해 테이블-리스트 매핑 방법을 제안하였다. 제안한 매핑 방법을 이용하여 웹에서 데이터베이스와 XML 사이의 데이터 교환이나 서로 다른 데이터베이스간의 데이터 교환과 데이터베이스에 저장된 데이터를 웹에서 관리할 수 있는 웹 에이전트 시스템을 설계하고 구현하였다.

웹 에이전트 시스템은 데이터베이스 시스템에 저장된 데이터를 XML 문서로 변환하거나 XML 문서를 데이터베이스에 저장하고, 데이터베이스로부터 가져온 데이터를 스킴의 리스트로 바꾼 내용을 XML로 만들고 이를 웹 브라우저 화면에서 테이블로 보여 준다.

웹 브라우저에서 데이터를 테이블 형태로 보여주면 일반 사용자들은 데이터베이스의 데이터 구조를 쉽게 이해할 수 있고, 데이터 수정과 삭제와 같은 기능을 웹 브라우저 화면에서 선택된 부분을 지우거나 수정하는 것으로 대신하게 되므로 일반 사용자들은 어려

운 데이터베이스 질의문이 아닌 웹 브라우저를 이용하여 데이터를 간단하게 관리할 수 있다.

본 논문은 데이터베이스의 데이터를 사용자가 원하는 목적에 따라 조인과 같은 복잡한 연산을 할 때 데이터베이스의 성능이 저하되지만 웹 서버로 가지고 온 데이터를 대상으로 데이터베이스에서 이루어지는 복잡한 조인 연산을 하더라도 성능이나 속도와 상관없이 데이터를 검색하거나 추출할 수 있는 기반을 마련하고자 하였다.

웹 에이전트 시스템은 복잡하고 다양한 데이터 처리를 사용자에게 익숙한 웹 브라우저 화면과 기본적인 검색 조건을 이용하여 복잡한 데이터 처리가 가능하도록 스킴의 리스트를 이용하였다.

구현된 웹 에이전트 시스템은 다양한 데이터 처리와 기본적으로 데이터베이스에서 수행하는 수정, 삭제, 생성과 같은 작업을 웹 브라우저를 통해 웹 서버에서 작업하게 함으로써 실제 데이터베이스에 계속적으로 연결하지 않으므로 데이터베이스의 부하를 줄이는 효과가 있다.

웹 에이전트 시스템에서 사용자는 웹 브라우저를 통해 데이터베이스의 데이터와 XML 문서의 내용을 확인하고, 데이터베이스를 관리하기 위해 단순히 원하는 기능에 해당하는 버튼을 선택하고 필요한 추가 정보를 입력하고 데이터 처리를 하기 위한 검색 조건만 적절하게 입력하면 원하는 작업을 수행할 수 있다. 또한, 웹을 이용하여 데이터베이스를 관리하므로 사용자가 웹 서버에 접속되어 있다면 시간과 장소에 구애받지 않고 데이터베이스를 관리할 수 있다.

향후 연구 방향으로는 스킴의 기능을 확대시켜 데이터베이스를 만들고, 웹에 존재하는 HTML 문서와 XML 문서로부터 정보를 추출하거나 검색할 수 있는 기능이 추가된 웹 에이전트 시스템을 구현할 계획이다.

참 고 문 헌

[1] A Beginner's Guide to HTML, http://archive.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimerAll.html

[2] W3C Extensible Markup Language(XML) 1.0, http://www.w3.org/TR/2000/REC-xml-20001006

[3] 이강찬, 손홍, 박기식, "XML 표준화 동향", 정보과학회지 제 19 권, 제 1 호, pp. 6-14, 2001.

[4] 유선영, XML 기반의 異機種 DBMS간 데이터 復除 웹 에이전트 設計 및 具現, 한국해양대학교 대학원, 석사학위논문, 2002.

[5] Martin B, "An Introduction to the Extensible Markup Language", http://www.personal.u-net.com/~sgml/xmlintro.htm

[6] W3C HTML, http://www.w3.org/MarkUp/#historical

[7] W3C XML, 1.1, http://www.w3.org/TR/xml11/

[8] 홍성용, XML 원리와 응용, 한빛미디어, 2003.

[9] W3C DOM(Document Object Model) Level 1 specification, http://www.w3.org/TR/REC-DOM-Level-1

[10] W3C Schema Part 0 Primer, http://www.w3.org/TR/xmlschema-0/

[11] W3C Schema Part 1 Structure, http://www.w3.org/TR/xmlschema-1/

[12] W3C Schema Part 2 Datatypes, http://www.w3.org/TR/xmlschema-2/

[13] 이경하, 정명희, 홍의석, XML 데이터베이스 구축, 성안당, 2002.

[14] 박은경, 정채영, 김현주, 배종민, "XML DTD로부터 관계형 테이블로의 매핑구조 설계", 정보과학회 춘계학술대회 제 28 권, 제 1 호, pp. 133-135, 2001.

[15] 윤정희, 박창원, 정진완, “객체 식별자를 이용한 객체지향 데이터베이스의 XML 문서로의 변환”, 정보과학회 논문지 : 데이터베이스, 제 28 권, 제 2 호, pp. 131-139, 2001.

[16] Frakes, Baeza-yates, *Information Retrieval : Data Structure and Algorithms*, Prentice-Hall, 1992.

[17] 강형일, 최영길, 이종설, 유재수, 조기형, “RDBMS와 IRS를 이용한 XML 저장관리 시스템 설계 및 구현”, 정보과학회 논문지 : 컴퓨팅의 실제, 제 7 권, 제 1 호, pp. 1-10, 2001.

[18] 장우혁, 김홍식, “XML기반의 효율적인 데이터 저장관리를 위한 DB2XML 변환 Wrapper의 설계”, 정보과학회 춘계학술대회, 제 28 권, 제 1 호, pp. 106-108, 2001.

[19] 박철현외 정재헌, 심대익, 이상구, “구조화된 문서에 대한 DBMS와 IRS의 성능 비교”, 한국 데이터베이스 학술대회 논문집 15권 1호, pp. 218-225, 1999.

[20] 손정한, 이희주, 장재우, 심부성, 주종철, “SGML 정보 검색을 위한 인덱스 관리자의 설계 및 구현”, 정보과학회 논문지 : 컴퓨팅의 실제, 제 5 권, 제 2 호, pp. 135-146, 1999.

[21] 유재수외 8명, 전자도서관 표준문서 관리를 위한 XML 저장관리기 기술 개발, 한국지식웨어 최종보고서, 1999.

[22] 이호섭, 홍충선, “분산 환경에서의 CORBA와 XML의 연동구조”, 정보과학회 춘계학술대회논문집, 제 28 권, 제 1 호, pp. 424-426, 2001.

[23] 장희철, 관계형 스키마에서 XML DTD로의 변환에 관한 연구, 연세대학교 공학대학원, 석사 학위 논문, 2003.

[24] Mark Graves, Designing XML Database, PHPTR, 2002.

[25] Christine Parent and Stefano Spaccapietra, “Issues and Approaches of Database Integration”, Communications of the ACM, 제 41 권, 제 5 호, pp. 166-178, 1998.

[26] 고승규, 조승기, 최윤철, “SGML/XML 검색시스템의 설계 및 구현", 한국멀티미디어학회 추계학술대회, 제 3 권, 제 2 호, pp. 99-102, 2000.

[27] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, J. F. Naughton, “Relational Databases for Querying XML Documents: Limitations and Opportunities”, Proc. of 25th Int'l Conf. on VLDB, Edinburgh, Scotland, UK, pp. 302-314, 1999.

[28] XML and Databases, http://www.rpbourret.com/xml/XMLAndDatabases.htm

[29] Ronald Bourret, http://www.rpbourret.com/xml/XMLDatabaseProds.htm

[30] XML Database Products : Middleware, http://www.rpbourret.com/xml/ProdsMiddleware.htm

[31] ASP2XML, http://www.stonebroom.com/axdocs.htm

[32] XML-DBMS, http://www.rpbourret.com/xmldbms/index.htm

[33] DB2XML, http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html

[34] Harold Abelson, Gerald Jay Sussman, Structure and Interpretation of Computer Programs, The MIT Press,

[35] HTDP, http://www.htdp.org

[36] MPS, http://mps.tit.ac.kr

[37] Mapping DTDs to Databases, http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html

[38] 연제원, 조정수, 이강찬, 이규철, “XML 문서 구조 검색을 위한 저장시스템 설계”, 정보과학회 춘계학술대회논문집, 제 26 권, 제 1 호, pp. 185-187, 1999.

[39] Litwin W., Mark L., Roussopoulos N., “Inter operability of Multiple Autonomous Databases”, ACM Computing Survey, 제 22 권, 제 3 호, pp. 267-293, 1990.

[40] Soon M. Chung and Pyeong S. Mah, “Schema Integration for Multidatabases Using the Uni- fied Relational and Object-Oriented Model”, ACM Conference on Computer Science, pp. 208-215, 1995.

[41] 유선영, 임재홍, “XML 기반의 이기종 DBMS간 데이터 복제 및 웹 에이전트 설계 및 구현”, 한국항해항만학회, 제 26 권, 제 4호, pp. 427-433, 2002.

[42] J.A. Larson, S.B. Navathe, and R. Elmasri. “A Theory of Attribute Equivalence in Database with Application to Schema Integration”, IEEE Transactions on Software Engineering, 제 15 권, 제 4 호, pp. 449-463, 1989.

[43] 유선영, 신송아, 임재홍, “인터넷에서 XML/EDI를 이용한 사이버 쇼핑몰 구현”, JCCI 춘계학술대회, pp. 409-412, 2001.

[44] Stefano Spaccapietra, Chirstine Parent, Yann Dupont, “Model Independent Assertions for Integration of Heterogeneous Schemas”, Very Large Data Bases Journal, 제 1 권, 제 1 호, pp. 81-126, 1992.

[45] A. P. Sheth and J. A. Larson. “Federated Databases Systems and Managing Distributed, Heterogeneous, and Autonomous Databases”, ACM Computing Surveys, 제 22 권, 제 3 호, pp. 183-236, 1990.

[46] 조재만, 신송아, 유선영, 임재홍, “WAP 서비스를 이용한 ETC 인증 및 과금 결제 시스템”, 정보처리학회 춘계학술대회, 제 10 권, 제 1호, pp. 767-770, 2003.

[47] 유선영, 이춘근, 임재홍, “이기종 DBMS간 데이터 교환과 복제를 위한 XML 웹 에이전트 설계 및 구현”, 한국멀티미디어학회, 제 7 권, 제 7 호, pp. 967-975, 2004.

[48] Martin Endig, Thomas Herstel, Eike Schallehn, Zoltan Sera, “An Approach for Semi-Automatic Derivation of XSLT Information Based on DTD Descriptions”, Proceeding of the Fifth East-European Conference on Advances in Databases and Information Systems, pp. 25-28, 2001.

[49] 오라클 iAS와 DB에서의 XML 프로그래밍, http://xmlgo.net/document/db/iAS_oracle_xml.htm

[50] Transaction Architecture for Management of INternet Object, http://www.ebm.co.kr/product/tamino/intro_tamino.htm

[51] Panta, http://www.tamino.co.kr/information/sagtamintro.htm

감사의 글

논문을 끝낸 오늘까지 저에게 많은 격려와 가르침을 주신 고마운 분들에게 이 지면을 빌려 감사의 마음을 전하고 싶습니다.

저에게 힘이 되어 주신 아버지, 어머님께 감사드립니다. 제가 태어날 때부터 이 논문이 나올 때까지 부모님의 사랑이 없었더라면 아마도 지금의 제가 있지 못했을 것입니다. 그리고 제가 사랑하는 오빠에게도 감사합니다. 갓 결혼한 새댁이 공부한다고 했을 때 아낌없이 격려해 주시고 며느리로써 부족한 저를 사랑으로 감싸주시고 물심양면으로 도와주신 시부모님께 진심으로 감사드립니다.

제가 대학원에 들어와서 졸업하기까지 많은 것을 가르쳐주시고 이끌어주신 임재홍 교수님께 깊이 감사드립니다. 교수님께서는 저에게 학문뿐만 아니라 인성을 키우는 데에도 많은 가르침을 주셨습니다. 그리고 바쁘신 중에도 제 논문을 심사해주신 김기문 교수님과 양규식 교수님, 이상배 교수님, 박연식 교수님, 대학원 시절 동안 저를 가르쳐 주신 교수님들께도 감사드립니다.

아내의 역할에 충실하지 못한 저를 사랑으로 이해하고 지켜보면서 언제나 든든한 후원자의 모습을 보여준 남편에게 고맙다고 전하고 싶습니다. 또한 저에게 너무나 많은 힘이 되어 준 영훈이와 영우 두 아들에게도 고맙다고 말하고 싶습니다.

일일이 다 소개할 순 없지만 연구실에서 생활하면서 같이 기쁨과 슬픔을 함께한 분들께도 감사드립니다.

마지막으로 저를 사랑해주시고 도와주신 많은 분들에게 항상 건강과 행복이 가득하시길 기원하며 감사의 글을 전합니다.