웹과 객체-관계 데이터베이스 시스템 연동을 위한 Java 메소드 기법

윤현진[†] · 용환승[††]

요 약

본 논문에서는 웹 시스템을 기반으로 객체-관계 DBMS 내에 저장된 정보를 제공하기 위한 새로운 연동 방안인 *Java 메소드* 기법을 제안한다. 제안한 연동 방법은 객체-관계 DBMS의 핵심이라고 할 수 있는 객체 개념과 객체에 연관된 사용자 정의 함수(메소드)를 추가할 수 있도록 기능을 확장하여 기존의 C 언어나 SQL 언어뿐만 아니라, Java 언어를 사용하여 정의하도록 함으로써 데이타베이스의 객체를 검색할 때 연관된 Java 메소드를 함께 검색하여 클라이언트에서 검색된 객체와 함께 수행하도록 하는 시스템을 설계하고 구현하였다. 이를 위해 JDBC와 Java RMI를 이용한 3계층 클라이언트/서버 구조의 Java 메소드 운영 환경을 설계 및 구현하고, 이를 객체-관계 DBMS와 연동하여 구현함으로써 제안한 방법의 효용성을 검증하였다.

Java Method Technique for the Integration of Web and Object-Relational Database System

Hyun-Jin Yoon[†] and Hwan-Seung Yong[††]

ABSTRACT

In this paper, Java Method approach for the integration of Web and Object-Relational DBMS is proposed. For the purpose, we extend to use Java language in programming user-defined function which is one of important characteristic of Object-Relational DBMS with object concept. Therefore we can retrieve not only database objects but also their associated Java methods to run in clients. Finally, to demonstrate the feasibility of the proposed Java Method technique, we designed and implemented this Java method system based on three-tier client/server architectire using JDBC(Java Database Connectivity) and Java RMI(Remote Method Invocation).

1. 서 론

다양한 응용 분야에서 파생되는 복잡한 객체, 비구조화된 데이타 그리고 멀티미디어 데이타 등의 정보를 데이타베이스 시스템을 통해 관리하기 위해서는 새로운 데이타 타입의 정의가 가능해야 하고, 새로운 데이타 타입을 연산 대상으로 하는 사용자 정의 함수(메소드)와 연산자를 쉽게 작성할 수 있어야 한다[1]. 그러나 데이타베이스 내부적으로 정의된 자료형태만을 지원하는 관계 데이타베이스로는 이러한 기능을 제공하기 어렵다. 기존의 관계 데이타베이스 시스템이 갖는 이러한 제약을 해결하기 위하여 제안된 분야가 객체-관계 데이타베이스이다. 객체-관계 데이타베이스는 기본적으로 관계 데이타베이스의 SQL92 표준 모델에 기반하고 있으며, 여기에 객체 지향 개념을 지원하기 위해 확장되어진 형태이다[1]. 따라서 기존의 관계 데이타베이스 상에 구축된 데이타를 그대로 사용할 수 있고, 동시에 객체 기반의 멀티미디어 정보를 효율적으로 관리할 수 있다는 장점을 갖는다. 현재 이러한 이유로 객체-관계 DBMS의

본 연구는 한국 과학재단 1997년도 핵심전문연구비 지원(과제번호 : 971-0902-219-1)에 의해 수행되었음

[†] 한국과학기술원 정보시스템연구소
[††] 이화여자대학교 컴퓨터학과

활발한 상용화가 이루어지고 있으며, 이에 따라 객체-관계 DBMS에 저장된 정보를 웹 시스템을 통해 제공하기 위한 연동 방안의 필요성도 함께 증가하고 있다.

지금까지 이루어진 웹과 데이타베이스 시스템을 연동하기 위한 대부분의 연구는 데이타베이스 게이트웨이를 통한 관계 데이타베이스 응용 프로그램을 작성할 수 있는 환경 구축에 주로 연구가 진행되어 왔다[21]. 그리고 Java 언어와 데이타베이스의 연동에 대한 연구는 관계 데이타베이스와의 클라이언트/서버 접속을 위한 JDBC(Java Database Connectivity)가 제안되고[8] 이를 기반으로 한 자바 데이타베이스 인터페이스 분야의 연구가 주로 진행되었다[22].

객체-관계 DBMS에 대한 연구로 Illustra와 Uni SQL이 상용화되어 제공되는 데, Java 언어를 지원하지 못한다[1]. 최근에 연구 개발 중인 객체-관계 DBMS로 Cornell 대학에서 개발하고 있는 PREDATOR (PRedator Enhanced DAta Type Object-Relation DBMS)[20]가 있다. 이 시스템에서는 확장된 추상 데이타타입을 기반으로 객체-관계 기능을 제공하며, 웹과의 인터페이스에서 Java 언어를 사용하고 있다. 그러나 Java 언어가 사용자 인터페이스에서 사용될 뿐 데이타베이스의 메소드로 정의, 저장되어 검색되지는 않는다.

본 논문에서는 웹 환경 하에서 객체-관계 DBMS를 효율적으로 지원하기 위한 새로운 연동 방안으로 *Java 메소드* 기법을 제안한다. 제안한 연동 방법은 객체-관계 DBMS의 핵심이라 할 수 있는 객체 개념과 객체에 연관된 사용자 정의 함수인 메소드 기능을 Java 언어로 확장한 것이다. Java 언어의 사용은 서버에서 실행되던 메소드가 인터넷으로 클라이언트에 전송되어 하드웨어와 OS에 독립적으로 실행될 수 있다는 것이다. 객체-관계 DBMS의 객체에 대한 메소드를 Java로 정의하여 사용할 경우 메소드의 특성과 메소드가 처리하는 데이타의 특성을 고려하여 웹 클라이언트 측 또는 서버 측에서 선택적으로 실행할 수 있어 보다 유연하고 동적인 시스템을 설계할 수 있다는 장점을 갖는다. Java 메소드 기법을 지원하는 연동 구조는 웹 클라이언트와 서버의 효율성과 확장성을 효과적으로 지원하기 위해 JDBC와 Java RMI를 이용한 3계층 클라이언트/서버 구조 모델을 갖는다. 이 연동 구조는 웹 서버에 대해 독립적인 특성을 지니고 있으며, 동시에 데이타베이스 시스템의 종류와 지리적 위치에 대한 제약이 없는 확장성과 효율성을 제공한다. 본 논문에서 제안하는 Java 메소드 기법에 기반한 웹과 객체-관계 DBMS 연동 시스템을 이용하면 객체-관계 데이타베이스의 정보를 데이터베이스의 Java 메소드를 호출하여 실행하는 일련의 과정을 통해 손쉽게 제공할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 웹에서 객체-관계 DBMS 내의 정보를 제공하기 위한 Java 메소드 연동 기법을 제안하고, 이를 지원하는 연동 시스템의 구조 모델에 대해서 설명한다. 3장에서는 제안한 연동 시스템의 효용성을 검증하기 위해 구현한 프로토타입 시스템의 내용을 기술하고, 4장에서는 구현 된 시스템을 기반으로 정보를 제공하는 예를 보여 주도록 한다. 마지막으로 5장에서는 본 논문의 결과를 정리하고 앞으로의 연구 방향을 제시한다.

2. Java 메소드 기법 설계

본 장에서는 객체-관계 DBMS가 웹 시스템을 기반으로 정보를 제공할 경우 필요한 기능과 설계 시의 요구 사항들을 제시하고, 이를 위해 본 논문에서 제안한 Java 메소드 연동 기법과 동작 원리 및 이를 지원하는 시스템이 갖는 특징을 명시하도록 한다.

2.1 설계 시 고려 사항

사용의 편리함으로 웹 사용이 확대되면서 웹 시스템을 기반으로 다양하고 대량의 정보를 제공하기 위한 데이타베이스의 사용이 필수적으로 되었고, 이에 웹과 데이타베이스를 연동하는 많은 연구들이 있어 왔다[2][3][4][5]. 기존의 연구들을 살펴 보면 대부분이 게이트웨이를 통한 관계 데이타베이스의 검색 또는 Java와 데이타베이스를 연동 하기 위한 미들웨어의 구축 등 주로 관계 데이타베이스의 특성을 지원하는 수준에서 이루어져 왔다. 그러나 다양한 웹 응용에서 파생되는 복잡한 객체, 비 구조화된 데이타, 멀티미디어 데이타 등을 보다 효과적으로 관리하기 위해서 점차 데이타 추상화(Data Abstraction), 정보 은닉(Information Hiding), 캡슐화(Encapsulation), 상속(Inheritance) 등의 객체 지향적 접근법을 기반으로 새로운 데이타 타입의 정의가 가능하고, 이를 연산 대상으로 하는 연산자, 즉 메소드를 쉽게 작성할

수 있는 객체 지향 데이타베이스 시스템이나 객체-관계 DBMS를 사용하려는 시도가 점차 이루어지고 있다[6][7][19].

지금까지의 데이타베이스 연동 방법과 구조를 분석한 결과를 토대로 객체-관계 DBMS가 웹 시스템과 통합하기 위하여 본 논문에서 제안한 연동 방법과 구조 모델은 다음과 같다.

첫째, 웹과 객체-관계 DBMS를 위한 효율적인 연동 구조 모델이 필요하다. 기존의 CGI를 이용한 연동 구조는 데이타베이스와의 연결/해제를 매 데이타베이스 관련 작업마다 반복하여 여기서 파생되는 시스템의 성능 저하라는 문제를 가지고 있으며, Java를 미들웨어로 사용하는 2계층 클라이언트/서버 구조는 새로운 데이타베이스의 추가와 삭제 시의 비효율성과 유지, 보수의 어려움이라는 단점을 갖는다. 따라서 이러한 연동 구조를 웹과 객체-관계 DBMS의 연동 구조로 적용하는 것은 부적절하며, 이에 Java를 기반으로한 JDBC와 Java RMI를 이용한 3계층 클라이언트/서버 구조 모델을 적합하다고 판단된다.

둘째, 웹 시스템에서 객체-관계 DBMS의 객체 개념 및 메소드 기능을 제공하여 객체-관계 DBMS가 가지고 있는 객체 지향적 특성을 보장해야 한다. 객체-관계 DBMS의 메소드는 대부분 데이타베이스 시스템에서 동적 적재(Dynamic Loading)가 가능한 이진 화일로 제공이 되기 때문에, 특정 컴퓨터 구조에 제약을 받지 않는 웹 시스템 상에서 메소드를 실행시키는 것은 쉽지 않다. 특히 Java 애플릿에서 데이타베이스의 메소드를 실행시키는 것은 애플릿의 네트웍 보안상의 이유로 더욱 어렵다. 이를 위해 본 논문에서는 객체-관계 DBMS의 객체에 대한 메소드를 Java 언어로 정의한 후, Java 언어가 갖는 네트웍 지향적인 특성을 이용하여 이를 데이타베이스의 정보 요구 시 웹 시스템에서 실행하여 정보를 제공하는 Java 메소드 기법을 제안한다. 이 기법은 객체-관계 DBMS의 객체에 대한 메소드를 통한 정보 제공을 가능하게 하기 때문에, 객체-관계 DBMS가 갖는 객체 지향적 특성인 데이타 추상화, 정보 은닉, 캡슐화를 보장하게 된다.

다음으로 클라이언트와 서버의 효율적인 확장성을 제공해야 한다. Java 메소드 기법은 이를 위해 두 가지 실행 타입을 갖는 메소드를 제공한다. 즉 Java 메소드와 이 메소드가 처리하는 객체 정보의 특성을 고려하여 웹 클라이언트 혹은 서버에서 선택적으로 메소드 실행이 가능하며, 이에 클라이언트와 서버간의 적재 균형(Load balancing) 효과를 기대할 수 있을 뿐만 아니라, 메소드 실행을 분산시킴으로써 사용자가 요구하는 시점에 정보를 제공해 줄 수 있어 전체 자원을 보다 효율적으로 사용할 수 있다.

마지막으로 멀티미디어 정보의 처리 기능이 강화된 데이타베이스 검색 인터페이스가 필요하다. Java 언어로 구현된 인터페이스는 사용자가 인지하고 이해하기 쉽다는 장점과 함께 클라이언트 측의 처리 능력을 이용할 수 있고, API의 계속적인 확장으로 멀티미디어 정보에 대해 다양한 처리와 정보 제공이 가능하다. 이에 Java 애플릿을 데이타베이스의 인터페이스로 사용하도록 한다.

표 1은 앞서 제시한 설계 시의 요구 사항에 대해 본 논문에서 제안한 방법들을 정리한 것이다.

표 1. 웹과 객체-관계 DBMS 연동 시 요구 사항과 제안 방법

요구 사항	제안한 방법
효율적인 연동 구조 모델	JDBC와 Java RMI를 통신 프로토콜로 사용하는 3계층 클라이언트/서버 구조 모델
객체 개념 및 메소드 기능지원	Java 메소드 기법
시스템의 효율성과 확장성	웹 클라이언트와 RMI 서버에서 Java 메소드를 선택적으로 실행 Incremental Data Delivery 기법
향상된 인터페이스	그래픽 사용자 인터페이스 웹 클라이언트에서의 자료 처리 멀티미디어 데이타를 위한 Java API 활용

2.2 Java 메소드 기법의 종류

본 절에서는 2.1절에서 제시한 요구 사항을 만족시키는 웹과 객체-관계 DBMS 연동 기법을 기술하고, 이 기법이 제공하는 웹 클라이언트 실행 타입의 Java 메소드와 RMI 서버 실행 타입의 Java 메소드에 대해 설명하도록 한다.

가) 개요

Java 메소드 기법이란 객체-관계 DBMS의 객체에 대한 메소드를 Java 언어로 정의한 후, 사용자의 정보 요구 시 해당 메소드를 객체-관계 DBMS로부

터 검색하여 웹 시스템에서 실행하여 정보를 제공해 주는 연동 방법이다. Java 메소드 기법을 이용하여 정보를 제공할 경우 객체 개념과 메소드 기능을 웹 시스템을 통해 제공할 수 있기 때문에 객체-관계 DBMS가 갖는 객체 지향적 특징을 보장할 수 있고 동시에 Java 언어가 갖는 네트워 지향적인 특성을 이용하여 웹 클라이언트와 RMI 서버에서 선택적으로 실행할 수 있어 클라이언트와 서버간의 Load Balancing 효과를 기대할 수 있다. 또한 Incremental Data Delivery 기법 즉, 사용자 요구에 대해 부분적인 결과 제공과 사용자가 원하는 시점에 정보를 제공하여 시스템의 자원을 보다 효율적으로 사용할 수 있는 부가적인 장점도 얻을 수 있다.

객체에 대한 메소드의 타입은 메소드의 특성과 메소드가 처리하는 데이타의 특성, 메소드와 데이타의 크기 등을 고려하여 결정된다. 메소드의 크기가 작은 경우, 또 메소드가 처리하는 데이타가 이미지 데이타의 경우처럼 웹 클라이언트에서 여러 번 사용될 경우 웹 클라이언트 실행 타입을 이용하는 것이 보다 효율적인 반면, 제공하는 정보의 크기에 비해 상대적으로 메소드의 크기가 큰 경우 RMI 서버 실행 타입이 바람직하다.

나) 웹 클라이언트 실행 타입의 Java 메소드

Java 언어로 정의된 Java 메소드 중 웹 클라이언트 실행 타입의 Java 메소드는 사용자 정보 요구 시 객체-관계 DBMS로 부터 검색되어 RMI 서버를 거쳐 웹 브라우저로 전송되어 메소드를 호출한 Java 애플릿에 동적으로 링크되어 실행된다. 웹 클라이언트 실행 타입의 Java 메소드는 그림 1과 같은 구조 하에 동작한다.

그림 1의 구조를 기반으로 웹 클라이언트 실행 타입의 Java 메소드가 실행되는 절차는 다음과 같다.

① 검색 인터페이스를 제공하는 Java 애플릿에서 정보 요구
② RMI 서버에 사용자 요구가 전달되고, RMI 서버는 객체-관계 DBMS로 부터 해당하는 Java Method를 검색
③ RMI 서버는 검색된 Java 메소드를 웹 클라이언트로 전송
④ 웹 클라이언트의 Java 애플릿이 Java 메소드를 동적으로 링크하여 실행
⑤ Java 메소드 실행 중 처리할 데이타를 RMI 서버를 거쳐 데이타베이스로 부터 검색
⑥ RMI 서버는 데이타를 웹 클라이언트로 전송
⑦ Java 메소드는 전송된 데이타를 처리하면서 사용자에게 정보를 제공

이와 같은 절차로 실행되는 웹 클라이언트 실행 타입의 Java 메소드는 메소드의 크기가 작은 경우, 또 메소드가 처리하는 데이타가 이미지 데이타와 같이 웹 클라이언트에서 여러 번 사용되는 경우 보다 효율적으로 동작한다.

다) RMI 서버 실행 타입의 Java 메소드

Java 메소드 중 RMI 서버 실행 타입의 Java 메소드는 사용자의 정보 요구 시 객체-관계 DBMS로부터 검색되어 RMI 서버에서 실행되고, 처리된 결과만을 웹 브라우저로 전송하여 사용자에게 제공한다. 서버 실행 타입의 Java 메소드가 실행되는 구조는 그림 2와 같고, 이를 기반으로 RMI 서버 실행 타입의 Java 메소드가 실행되는 절차는 다음과 같다.

① 검색 인터페이스를 제공하는 Java 애플릿에서 정보 요구
② RMI 서버에 사용자 요구가 전달되고, RMI 서버

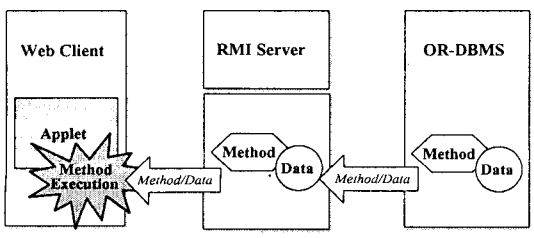


그림 1. 웹 클라이언트 실행 타입의 Java 메소드 실행 구조

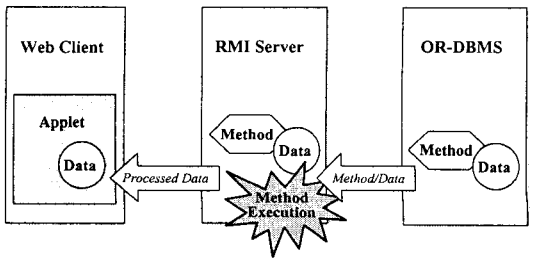


그림 2. RMI 서버 실행 타입의 Java 메소드 실행 구조

는 객체-관계 DBMS에서 해당하는 Java Method를 검색

③ RMI 서버는 검색된 Java 메소드를 실행

④ Java 메소드 실행 중 처리해야 할 데이타를 데이타베이스에서 검색

⑤ RMI 서버는 검색한 데이타를 처리하여 Java 메소드 실행을 마치고 처리된 데이타를 웹 브라우저로 전송

⑥ 웹 브라우저의 Java 애플릿은 전송 된 결과를 받아 사용자에게 제공

이와 같은 절차로 실행되는 RMI 서버 실행 타입의 Java 메소드는 제공하는 정보의 크기에 비해 상대적으로 메소드의 크기가 큰 경우, 또는 서버 쪽의 보다 강력한 컴퓨팅 파워를 필요로 하는 메소드에 적합한 방법이라고 할 수 있다.

웹을 통해 객체-관계 DBMS의 정보를 제공해 주기 위한 새로운 연동 방안인 Java 메소드 기법은 앞서 기술한 웹 클라이언트 실행 타입의 Java 메소드와 RMI 서버 실행 타입의 Java 메소드로 구분되고, 이들의 특징을 정리하면 표 2와 같다.

2.3 시스템 구조 모델

Java 메소드 기법을 지원하는 전체 시스템은 순수 Java 기술인 JDBC[8][9]와 RMI[10][11]를 이용한 3계층 클라이언트/서버 구조 모델을 사용하며 그림 3과 같다.

그림 3의 구조가 갖는 장점은 다음과 같다.

- 데이타베이스 접속/해제를 전담하는 전용 서버를 두어 데이타베이스와의 연결을 계속 유지하게 되어 데이타베이스 시스템이 제공하는 최적화 기능을 그대로 이용할 수 있다.
- 데이타베이스 접속을 위한 프로토콜과 드라이버를 전용 서버, 즉 Java 응용 프로그램이 갖게 되므로 Java를 이용한 2계층 구조에서의 문제를 해결하였다.
- 웹 클라이언트와 데이타베이스 접근을 위한 서버와의 통신 프로토콜로 Java RMI를 사용하여 개선된 통신 속도를 기대할 수 있다.
- 데이타베이스 시스템의 추가와 삭제가 용이하며, 데이타베이스 시스템의 종류와 지리적 위치에 대한 제약이 없다.
- 웹 시스템과 객체-관계 DBMS에 어떠한 변경도 필요없어 기존의 데이타베이스 시스템을 이용한 정보 시스템에 수정없이 그대로 적용될 수 있다.
- Java 메소드를 웹 클라이언트 혹은 서버에서 선택적으로 실행할 수 있어 전체 시스템의 자원을 보다 효율적으로 활용할 수 있다.
- 웹 클라이언트와 RMI 서버, 데이타베이스 서버의 프로세싱 파워를 최대로 활용할 수 있다.

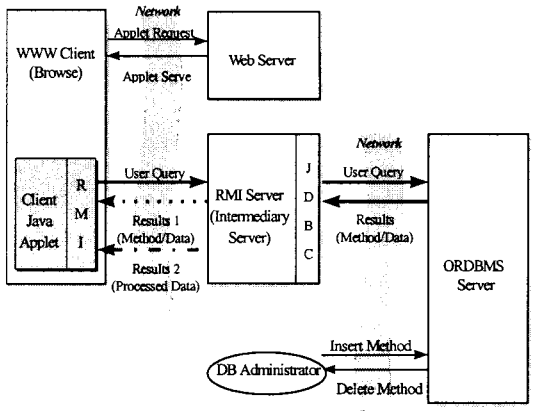


그림 3. 연동 구조 모델

표 2. Java 메소드 종류

종 류	웹 클라이언트 실행 타입	RMI 서버 실행 타입
실행위치	웹 클라이언트(웹 브라우저)	RMI 서버 (데이타베이스 접속을 위한 전용 서버)
실행방법	자바 애플릿에 동적으로 링크되어 실행	RMI 서버, 즉 자바 응용 프로그램에 동적으로 링크되어 실행
특 징	메소드의 크기가 작은 경우, 메소드가 처리하는 데이타가 웹클라이언트에서 여러번 사용되는 경우에 적합	메소드가 처리하는 정보의 크기에 비하여 메소드의 크기가 상대적으로 큰 경우, 서버 쪽의 강력한 컴퓨팅 파워를 이용하고자 할 경우에 적합

이와 같은 장점을 갖는 웹과 객체-관계 DBMS 연동을 위한 Java 메소드 기법을 지원하는 구조 모델의 구성은 물리적인 구성과 기능적인 구성으로 살펴볼 수 있다.

- 물리적 구성
  ① 1계층 - 검색과 결과에 대한 인터페이스를 제공하는 Java 애플릿을 위한 웹 클라이언트
  ② 2계층 - 웹 서버와 RMI 서버
  ③ 3계층 - 객체-관계 DBMS 서버
- 기능적 구성
  ① 객체-관계 DBMS에서의 Java 메소드 정의/관리 모듈
  ② 웹 클라이언트 실행 타입의 Java 메소드 실행 모듈
  ③ RMI 서버 실행 타입의 Java 메소드 실행 모듈

기능적 구성에서 Java 메소드 실행 모듈은 객체-관계 DBMS 종류와 무관하게 설계 가능한 모듈인 반면, 객체-관계 DBMS에서의 Java 메소드 정의/관리 모듈은 실제 사용하는 객체-관계 DBMS에 따라 다르게 설계된다. 가령, UniSQL, Illustra의 경우 객체에 대한 메소드의 정의는 해당 클래스에 종속적으로 정의하도록 하는 방법을 제공하고 있으며, PostgreSQL의 경우는 클래스에 독립적으로 메소드를 정의하도록 하고 있다. 따라서 Java 메소드를 정의하고 관리하는 모듈은 객체-관계 DBMS가 제공하는 메소드 정의 방법을 정확히 파악하여 설계해야 한다.

3. 시스템 구현

본 장에서는 논문에서 제안한 웹과 객체-관계 DBMS 연동을 위한 Java 메소드 기법과 이를 기반으로 하는 연동 시스템 구조의 효용성을 검증하기 위해서 구현한 프로토타입 시스템에 대해서 기술하도록 한다.

3.1 시스템 구성 및 구현 환경

Java 메소드 기법을 기반으로 하는 웹과 객체-관계 DBMS 연동 구조를 위한 프로토타입 시스템 구현을 위해 객체-관계 데이타베이스 시스템으로 PostgreSQL[12]을 사용하였고, 서버 구현을 위해 Solaris2.5상에서 JDK1.1.1[13]과 JDBC Driver인 JavaPostgres95[14]를 사용하였다. PostgreSQL은 Berkeley 대학의 Postgres95 DBMS에서 발전한 것으로 1986년 POSTGRES DBMS라는 이름의 관계 데이타베이스 시스템으로 구현되기 시작하여 현재는 새로운 데이타 타입, 연산자, 함수를 정의할 수 있는 기능을 통해 뛰어난 확장성을 제공하며, 클래스, 상속 등의 객체 지향적 개념을 지원하는 차세대 객체-관계 데이타베이스 시스템으로 인정받고 있다. JavaPostgres 95는 Postgres95와 PostgreSQL 데이타베이스 시스템을 위한 JDBC Driver로 JDBC 호출을 데이타베이스 시스템 전용의 네트웍 프로토콜로 바꿔주는 타입의 드라이버로 현재 버전 0.6까지 개발된 상태이다. 사용자로 부터 객체-관계 데이타베이스의 정보 검색을 담당하는 애플릿에 대한 요구를 받아 이를 전송해 주는 역할을 하는 웹 서버로 NCSA HTTPD 1.5[15]를 사용하였고, 웹 클라이언트로는 JDK 1.1.1 이상을 지원하는 웹 브라우저면 가능하다. 이를 위해 HotJava1.0[16]과 MS Internet Explore4.0[17] 이상 버전을 사용해야 하며, Netscape Navigator 4.0[18]을 사용할 경우 JDK 1.1.1을 지원하는 Patch 프로그램을 추가해야만 사용이 가능하다는 제약이 있다. 따라서 HotJava 1.0과 MS Internet Explore 4.0이 권장할 만한 브라우저라고 할 수 있다.

3.2 시스템 구조

시스템은 순수 자바 기술인 Java RMI와 JDBC를 통신 프로토콜로 사용하는 3계층 클라이언트/서버 구조를 갖는다. 시스템의 전체 구조는 물리적으로는 사용자에게 질의 인터페이스와 질의에 대한 결과를 제공하는 Java 애플릿을 위한 웹 클라이언트, 웹 서버와 RMI 서버, 그리고 데이타베이스 서버의 세 계층으로 구성되어 있다. 여기서 Java 애플릿이 갖는 보안상의 이유로 웹 서버와 RMI 서버는 같은 호스트에 있어야 하는 제약이 있다. 기능적인 모듈을 중심으로 구분한 시스템의 구조는 PostgreSQL에서의 Java 메소드 정의 및 관리 모듈, 웹 클라이언트에서의 Java 메소드 실행 모듈, RMI 서버에서의 Java 메소드 실행 모듈로 구성된다. 다음 절에서 시스템을 위해 구현한 모듈들을 설명하도록 한다.

3.3 PostgreSQL에서의 Java 메소드 정의

PostgreSQL을 비롯한 현재의 객체-관계 DBMS

는 Java 언어를 사용하기 훨씬 이전에 개발된 것들이기 때문에 Java 언어로 객체에 대한 메소드를 정의하는 환경을 제공하는 데이타베이스 시스템은 없다. 이러한 이유로 PostgreSQL의 멀티미디어 정보, 혹은 사용자 정의 객체에 접근하여 연산을 수행하고 정보를 제공하는 역할을 하는 객체에 대한 메소드를 Java로 정의하고 관리하기 위해서 다음의 Java 응용 프로그램과 카탈로그 테이블을 정의하였다.

가) Java 응용 프로그램 - InsertMethod.java/ DeleteMethod.java

PostgreSQL에서 객체에 대한 메소드를 Java언어로 정의하는 기능은 InsertMethod와 DeleteMethod라는 2개의 Java 응용 프로그램에 의해 제공된다. InsertMethod는 Java 메소드를 PostgreSQL에 정의하는 역할을, DeleteMethod는 PostgreSQL에서 Java 메소드를 삭제하는 역할을 담당하도록 설계하였다.

```
public class InsertMethod {

    public static void main(String[] args) {
      try {
        Driver pgd = (Driver) new postgres95.PGDriver();
        String db = "jdbc:postgres95://203.255.178.134:5432/Project" ;
        Connection conn = DriverManager.getConnection(db, "dblab", "elqldus9");

        Statement stat = conn.createStatement();
        ResultSet rs = stat.executeQuery("select max(id) from mapper");
        rs.next();
        int new_id = rs.getShort(1) + 1;

// Java 메소드에 대한 메타 데이타를 유지하는 카탈로그 테이블에 관련 정보 입력과
  Java 메소드를 구현한 Java클래스 화일 입력
        stat.executeUpdate("insert into mapper values("+new_id+", '"+args[0]+"')");
        stat.executeUpdate("insert into lotable values("+new_id+",
                    lo_import('/home/grad/dblab/hjyoon/Project/Demo/"+args[0]+".class'))" );
      } catch (Exception e) {
        System.out.println("Insert Error : " + e);
    } } }
```

그림 4. InsertMethod.java 프로그램 부분

```
import java.sql.*;

public class DeleteMethod {
    public static void main(String argv[]) {
      try {
        Driver pgd = (Driver) new postgres95.PGDriver();
        String db = "jdbc:postgres95://203.255.178.134:5432/Project" ;
        Connection conn = DriverManager.getConnection(db,"dblab","elqldus9");
        Statement stat = conn.createStatement() ;
        ResultSet rs = stat.executeQuery("select lotable.loid from mapper, lotable
                        where mapper.id = lotable.id and mapper.name = '"+argv[0]+"'");
       rs.next();
        int t_id = rs.getInt(1);

// 카탈로그 테이블에서 관련 레코드를 삭제하고, 해당 테이블과 인덱스 테이블을 삭제
        stat.executeUpdate("delete from lotable where loid ="+ t_id);
        stat.executeUpdate("delete from mapper where name = '"+argv[0]+"'");
        stat.executeUpdate("drop table Xinv"+t_id);
      } catch (Exception e) {
   ...
```

그림 5. DeleteMethod.java 프로그램 부분

그림 4와 그림 5는 각각의 구현 내용이다.

나) 카탈로그 테이블

PostgreSQL에서 Java 메소드에 대한 일종의 메타 데이타를 관리하기 위해서 mapper와 lotable이라는 2개의 카탈로그 테이블을 정의하였다. Mapper 테이블은 Java 메소드의 고유 번호와 이름을 관리하고, lotable은 Java 메소드의 고유 번호와 시스템에 의해서 할당되는 oid 값을 관리한다. InsertMethod에 의해 Java 메소드가 하나 정의되면 mapper와 notable에 하나의 레코드가 자동으로 생성되고, 할당된 oid값을 갖는 Xinvoid 테이블과 Xinxoid 라는 인덱스 테이블이 데이타베이스 시스템에 의해 자동으로 생성된다. 마찬가지로 DeleteMethod에 의해 Java 메소드가 삭제되면 해당하는 레코드가 mapper와 lotable에서 삭제되고, 삭제되는 oid 값을 갖는 Xinvoid 테이블과 Xinxoid 인덱스 테이블이 자동으로 삭제된다. PostgreSQL에서는 BLOB(Bimary Large Object) 데이타, 즉 Java 메소드의 실제 값은Xinvoid와 Xinxoid 두 개의 테이블에 저장되고 관리된다. 그림 6은 Java 메소드 관리를 위해 설계된 2개의 카탈로그 테이블의 스키마와 상관 관계를 보여 준다.

3.4 Java 메소드 기법 구현 모듈

InsertMethod/DeleteMethod 응용 프로그램을 이용하여 PostgreSQL에 정의한 Java 메소드를 웹 클라이언트 혹은 RMI 서버에서 호출하여 실행하는 방법은 간단한 모듈로 설계가 가능하다. 응용 프로그램 개발자는 웹 클라이언트 실행 타입의 Java 메소드와 서버 실행 타입의 Java 메소드를 메소드의 이름과 필요한 파라미터를 명시하여 다음과 같은 방법으로 호출하면 된다.

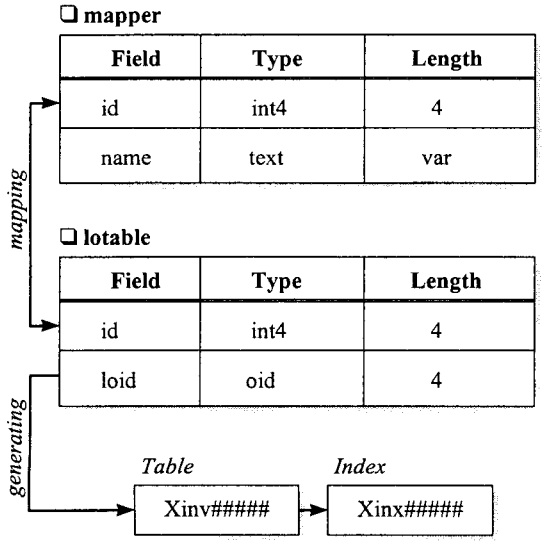


그림 6. mapper와 lotable의 스키마와 상관 관계

가) 웹 클라이언트 실행 타입의 Java 메소드

```
String ClassName = photoBox;
String MethodName = show;
Object[] Parameters = new Object[1];
Patameters[0] = 962COG08;
Object o = e.ClientSide(ClassName, MethodName, Parameters);
```

ClientSide 메소드는 명시한 photoBox 라는 Java 메소드를 PostgreSQL로부터 검색하여 웹 브라우저로 전송한 후 호출한 Java 애플릿에 동적으로 링크시켜 웹 클라이언트에서 실행한다. photoBox는 실행이 되면서 자신이 처리하는 데이타를 데이타베이스 서버로부터 검색하여 웹 클라이언트로 가지고 와서 처리한 후 사용자에게 처리된 정보를 제공한다. ClientSide 메소드의 처리 모듈은 그림 7과 같다.

나) RMI 서버 실행 타입의 Java 메소드

```
String ClassName = getData;
String MethodName = url;
Object[] Parameters = new Object[1];
Patameters[0] = 962COG08;
Object o = e.ServerSide(ClassName, MethodName, Parameters);
```

ServerSide 메소드는 명시한 getData 라는 Java 메소드를 PostgreSQL로 부터 검색하여 RMI 서버로 전송한 후 RMI 서버에서 실행한다. getData는 실행이 되면서 자신이 처리하는 데이타를 PostgreSQL 서버로 부터 검색하여 RMI 서버로 가지고 와서 처리한 후 처리된 정보만을 웹 클라이언트로 전송하여 사용자에게 제공한다. ServerSide 메소드의 처리 모듈은 그림 8과 같고 서버 실행 타입의 메소드인 getData는 웹 클라이언트 실행 타입의 Java 메소드에 비해 다소 복잡한 절차로 실행된다.

4. 정보 제공 예제 프로그램

다음은 Java 메소드 기법을 지원하는 시스템을 이

```
public Object clientSide(String className,
 String methodName, Object[] params ) {
   try {
     servlet.extClass(className);
        //Java 메소드를 PostgreSQL에서 검색
     Class c = Class.forName(className);
        //Java 메소드를 동적으로 링크
     Method[] m = c.getMethods();

     int count = 0;  boolean found = false;

     while ((!found) && (m[count] != null)) {
       if (methodName.equals(m[count].getName()))
         found = true;
       else count++;
     }

     if (found)
       //Java 메소드에 의해 처리된 정보를 반환
       return m[count].invoke(null, params);
     else
       return null;
 ...
```

그림 7. 웹 클라이언트 실행 타입의 Java 메소드 처리 모듈 : ClientSide

```
public Object serverSide(String className, String
  methodName, Object[] params) {
    try {

 //servlet은 RMI 서버 객체이고exeMethod는 원격 서
 버 객체의 메소드로 exeMethod는 인수로 받은 Java 메
 소드를 PostgreSQL로 부터 검색하여 RMI 서버에서 실
 행하고 결과만 반환
   return servlet.exeMethod(className,
    methodName, params);
 } catch (Exception e) {
    System.out.println("Executor.serverSide : " + e);
 return null;
 }
}
```

그림 8. RMI 서버 실행 타입의 Java 메소드 처리 모듈: ServerSide

용하여 학생에 대한 정보를 제공하는 응용 프로그램의 예를 보여 준다(그림 11에서부터 그림 13까지). 객체-관계 DBMS에 저장된 정보들은 함께 저장되어 관리되고 있는 두 가지 타입의 Java 메소드에 의해 웹 시스템에서 제공된다.

그림 9는 예제 프로그램에서 사용하는 학생에 대

```
CREATE TABLE person (
  id          varchar(15),
  name        varchar(25),
  birth       date,
  email       text,
  url         text,
  info        char16[],        // Object Array
  image       varchar(16) );

CREATE TABLE grad (
  major       text
) INHERITS (person);           // 상속 기능
```

그림 9. 학생 정보 테이블 정의

한 정보를 PostgreSQL이 제공하는 상속(Inheritance)과 Object Array 기능을 이용하여 정의한 것이다. grad 테이블은 person 테이블의 7개의 속성 모두와 자신의 속성 1개를 포함하는 스키마 구조를 갖는다. 다음의 그림 10은 예제 프로그램에서 사용하게 될 Java 메소드인 photoBox.class 와 getData를 PostgreSQL의 객체에 대한 메소드로 정의하는 방법을 보여 준다.

```
//웹 서버와  RMI 서버가 있는 호스트에서 실행
$ java InsertMethod photoBox
$ java InsertMethod getData
```

그림 10. Java 메소드 정의

그림 11은 사용자에게 데이타베이스 검색을 위한 질의를 받는 인터페이스를 위한 Java 애플릿이다. 사용자는 원하는 질의를 입력한 후 Search 버튼을 누른다. 그러면 사용자가 입력한 질의는 PostgreSQL 서버와의 접속을 담당한 RMI 서버로 전달되고, RMI 서버는 다시 PostgreSQL 서버로 질의를 전달하여 처리하도록 한다. 질의의 결과 즉, 데이타는 RMI 서버를 거쳐 애플릿으로 전송되어 사용자에게 제공된다. 결과 리스트 중 자세한 정보를 원하는 아이템을 선택하여 Show 버튼을 누르면 자세한 정보를 제공하는 Java 애플릿을 호출하는 요구가 웹 서버로 전달된다. 웹 서버는 요구를 받아 해당하는 Java 애플릿을 웹 브라우저로 전송한다. 결과 리스트에서 선택한 아이템의 ID 값은 Parameter.class 화일을 통해 호출한 Java 애플릿에게로 전달되어 파라미터로 사용된다.

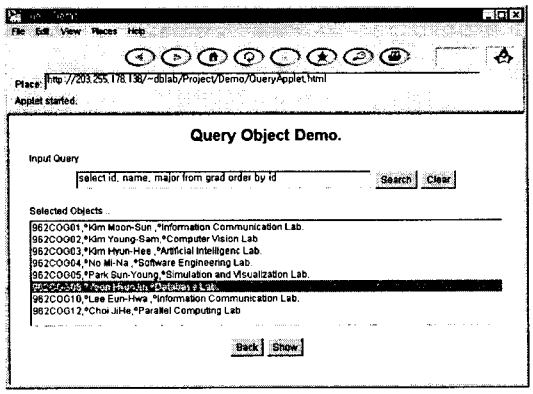
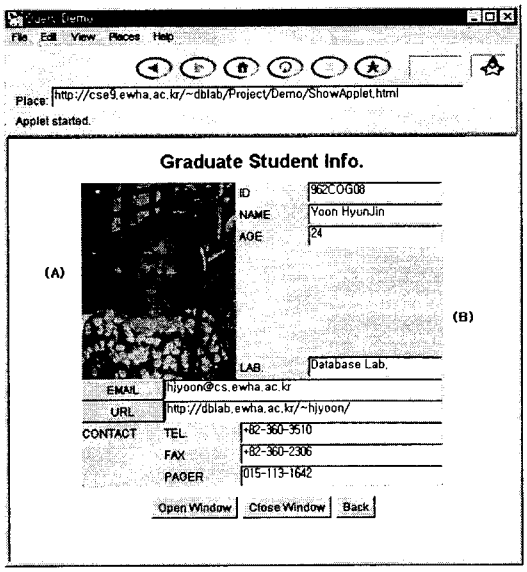


그림 11. 질의 인터페이스를 위한 애플릿

그림 12는 그림 11의 애플릿에서 선택한 아이템의 상세 정보를 제공하는 애플릿이다. 그림 12에서 (A) 부분의 이미지 데이타는 웹 클라이언트에서 실행되는 Java 메소드인 photoBox에 의해 제공된다. PostgreSQL로 부터 검색된 photoBox 메소드는 RMI 서버를 거쳐 웹 클라이언트로 다운로드되어 호출한 Java 애플릿에 동적으로 링크되어 실행된다. 실행 중에 필요한 이미지 데이타를 데이타베이스에서 검색하여 웹 브라우저로 가져오게 된다. 이때 photoBox와 이미지 데이타는 웹 서버가 있는 호스트의 화일 시스템에 임시로 저장되며, 임시로 저장된 이미지 데이타의 경우 하단의 메뉴에 의해 다양한 이미지 프로세싱(축소, 확대, 회전 등) 작업을 할 때 데이타베이스에서 검색하여 가져올 필요없이 재사용하게 된다.



(A) 웹 클라이언트에서 실행되는 Java 메소드에 의한 정보
(B) RMI 서버에서 실행되는 Java 메소드에 의해 처리된 정보

그림 12. 상세 정보 제공을 위한 애플릿

반면 (B)부분의 텍스트 데이타는 RMI 서버에서 실행되는 Java 메소드인 getData 메소드에 의해 제공된다. getData는 PostgreSQL에서 검색되어 RMI 서버에서 실행된다. 실행 중에 필요한 데이타를 데이타베이스 서버로부터 검색하여 처리하여 실행을 마치고, 처리가 된 데이타만을 웹 브라우저로 전송하여 정보를 제공하게 된다. 이와 같이 텍스트 정보의 경우 처리되어 웹 클라이언트로 전송되는 데이타의 양이 작기 때문에 서버에서 정보를 처리하는 메소드를 통해 제공하는 것이 효율적이다.

그림 12의 하단의 메뉴 중 Open Window 버튼을 누르면 학생의 이미지 데이타에 대해서 다양한 처리를 할 수 있는 팝업 윈도우가 제공된다. 그림 13은 이미지 프로세싱 윈도우에서 확대/축소, 명암 조절, 이동 등 다양한 메소드를 통해 이미지를 처리하는 예를 보여 주고 있다. 이 때 이미지 데이타는 데이타베이스로 부터 검색하는 과정없이 바로 웹 서버가 있는 호스트에서 다운 로드하여 이용하며, 이미지 처리를 위한 Java 메소드는 데이타베이스로 부터 검색되어 웹 클라이언트에서 실행된다.

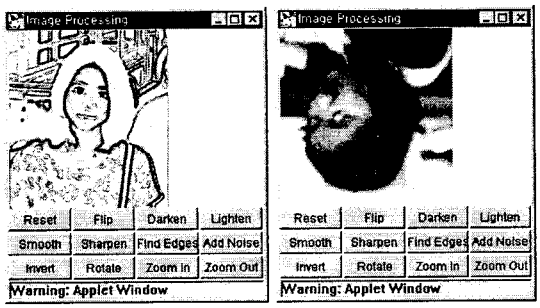


그림 13. 이미지 데이타에 대한 다양한 처리의 예

5. 결론 및 향후 연구 방향

본 논문에서는 웹 응용의 다양한 분야에서 파생되는 복잡한 객체, 비 구조화된 데이타, 비디오, 오디오, 이미지 등의 멀티미디어 데이타를 DBMS를 통해 효과적으로 관리하고 웹 시스템 상에서 이러한 정보들을 제공하기 위해 웹과 객체-관계 DBMS를 연동하

기 위한 방안으로 Java 메소드 기법을 제안하였다. Java 메소드 기법이란 객체-관계 데이타베이스의 멀티미디어 정보, 혹은 사용자 정의 객체에 접근하고 연산을 수행하기 위한 객체에 대한 메소드를 Java 언어를 이용하여 정의한 후, 사용자의 정보 요구 시 이 메소드를 웹 응용에서 실행하여 정보를 제공해 주는 웹과 객체-관계 DBMS를 위한 연동 방법이다. 본 논문에서 제안한 Java 메소드 기법을 이용하면 객체-관계 DBMS의 정보를 웹 시스템을 통해 손쉽게 제공할 수 있으며, 기존의 연구에 비해 다음과 같은 특징을 갖는다.

- 전체 시스템은 순수 Java 기술인 JDBC와 Java RMI를 통신 프로토콜로 사용하기 때문에 Java 언어가 갖는 가능한 모든 장점을 수용할 수 있다.
- 객체-관계 DBMS의 객체에 대한 메소드를 Java 언어로 작성하도록 하여 웹 시스템에서 메소드의 실행을 가능하게 하였다.
- Java 메소드는 메소드의 특징과 메소드가 처리하는 데이타의 성격을 고려하여 웹 클라이언트 실행 타입과 RMI 서버 실행 타입으로 구분되어 정의/실행되기 때문에 클라이언트와 서버에서의 분산된 메소드 실행이 가능하다.
- 사용자가 요구하는 시점에 메소드의 실행을 통해 정보를 제공하므로 사용자의 질의에 대해서 부분적인 결과 제공이 가능하여 전체 시스템의 자원을 보다 효율적으로 활용할 수 있도록 한다.
- Java 메소드 기법은 웹 서버와 독립적인 특성을 지니고 있으며, 동시에 일반적인 객체-관계 DBMS에서 제공되는 기능만을 이용한 간단한 구조를 지니고 있으므로, 다양한 객체-관계 DBMS 또는 객체 지향 데이타베이스 시스템에 쉽게 적용 및 확장될 수 있다.
- Java 언어가 갖는 탁월한 멀티미디어 정보 상연 기능을 최대로 활용하는 데이타베이스 응용 프로그램을 작성할 수 있어 보다 이해하기 쉽고 향상된 데이타베이스 검색 인터페이스를 제공한다.

본 논문에서 제안한 연동 기법과 시스템 구조의 효용성을 검증하기 위해 프로토타입 시스템을 구현하였고다. 앞으로 본 연구에 덧붙여 하나의 Java 메소드가 웹 클라이언트 실행 타입과 서버 실행 타입 모두를 지원하도록 하여 전체 시스템의 성능을 고려하여 동적으로 실행 타입을 결정하여 실행되도록 하는 방법을 추가할 예정이다. 또한 데이타베이스 엔진을 Java로 작성하여 질의 처리 자체를 분산시키는 방법에 대한 연구와 이들을 종합하는 연구가 필요하다.

참 고 문 헌

[1] M. Stonebraker, *Object-Relational DBMSs : The Next Great Wave*, Morgan Kaufmann Publishers, Inc., 1996.

[2] Tam Nguyen, V. Srinivasan, "Accessing Relational Databases from the World Wide Web, Database Technology Institute IBM Santa Teresa Laboratory," *Proc. ACM-SIGMOD conference on Management of Data, Motreal,* Canada, June 1996.

[3] J. Yang, G. Kaiser, "An Architecture for Integrating OODBs with WWW," *The Fifth International Conference on World Wide Web*, May, 1996, http://www5conf.inria.fr/fich_html/papers/P31/Overview.html.

[4] N. Duan. "Distributed Database Access in a Corporate Environment Using Java," *The Fifth International Conference on World Wide Web*, May 1996, http://www5conf.inria.fr/fich_html/papers/P23/Overview.html.

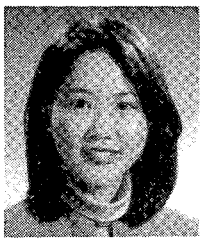
[5] D. Ingham, M, Little, S. Caughey, S. Shrivastava, "W3Objects : Bringin Object-Oriented Technology to the Web," *The First WWW Journal, World Wide Web Consortium*, 1996, http://www.w3.org/pub/WWW/Journal/1/ingham.141/paper/141.html.

[6] P. Seshadri, "Object-Relational Database on the WWW : Design and Implementation Issues," *Proc. 1997 8th International Conference on Management of Data*, Chennai(Madras), INDIA.

[7] P. Seshadri, M. Paskin, "PREDATOR : An OR-DBMS with Enhanced Data Types," *Proc. 1997 ACM-SIGMOD conference on Management of Data*, Tucson, Arizona, USA, May.

[8] G. Hamilton & R. Cattell, *JDBC^{TM} : A Java SQL API*, Sun Microsyst-ems, Jan 10, 1997.
[9] Rawn Shah, "Integrating Databases with Java via JDBC," Java World, May 1996.
[10] *Java RMI Specification*, Sun Microsystems, Mar 24, 1997.
[11] B. Bilow, *Client/Server Computing Using JDBC and RMI*, http://www.geocities.com/SiliconValley/Park/9841/rmi.html.
[12] Yu, J. Chen, *The Postgres95 User Manual*, U. C. Berkeley, 1995, http://www.postgresql.org/.
[13] Sun Microsystems, Java Development Toolkit, http://www.javasoft.com/products/jdk/1.1/index.html.
[14] P. Mount, *PostgreSQL JDBC Driver*, 1997, http://www.demon.co.uk/finder/postgres/index.html.
[15] NCSA HTTPd Development Team, NCSA HTTPd, http://hoohoo.ncsa.uiuc.edu/.
[16] Sun Microsystems, HotJavaTM Browser, http://www.javasoft.com/products/hotjava/.
[17] Microsoft, MS Internet Explorer, http://www.microsoft.com/ie/.
[18] Netscape Communications Corp., Netscape Communicator, http://cgi.netscape.com/cgi-bin/upgrade.cgi?cp=hp3noptun.
[19] 양철웅, "멀티미디어 데이타베이스 서비스," 96 삼성 휴먼테크 논문, 1996.
[20] "The PREDATOR Project," http://cs.cornell.edu/Info/Project/PREDATOR
[21] 김평철, "웹과 데이타베이스 연동," 한국정보과학회 데이타베이스 연구회 1997년 춘계 튜토리얼 웹과 데이타베이스 자료집, pp. 49-70, 1997. 5. 22.
[22] 조윤진, "Java와 데이타베이스," 한국정보과학회 데이타베이스 연구회 1997년 춘계 튜토리얼 웹과 데이타베이스 자료집, pp. 167-182, 1997. 5. 22.

[



윤 현 진

1996년 이화여자대학교 컴퓨터학과 학사
1998년 이화여자대학교 컴퓨터학과 공학석사
1998년 4월~현재 한국과학기술원 정보시스템연구소 연구원
관심분야 : 데이타베이스 시스템, 분산 데이타베이스 및 웹 기술

용 환 승

1983년 서울대학교 컴퓨터공학과 학사
1985년 서울대 대학원 컴퓨터공학과 공학석사
1985년~1989년 한국전자통신연구소 연구원
1994년 서울대 대학원 컴퓨터공학과 공학박사
1994년 서울대 컴퓨터신기술공동연구소 특별연구원
1995년~현재 이화여자대학교 컴퓨터학과 조교수
관심분야 : 객체-관계 데이타베이스 시스템, 멀티미디어 데이타베이스