Java를 기반으로 한 웹 데이타베이스 응용을 위한 프레임워크

구 흥 서*

A Framework for Java-based Web Database Applications

Heung-Seo Koo*

Abstract

The World-Wide-Web have many advantages as a front-end of database systems. Hence in internet applications, such as E-Commerce systems, the requirements for Web-based database applications have been increasing. In this work we propose a framework for java-based Web database applications - JaWAF(Java-based Web Application Framework). JaWAF has the 3-tier architecture of Client/Server, and consists of database gateway, application server, and communication and message component. When this framework is applied to the database applications developments, it can provide advantages, such as ease understanding and fast implementations of the Web applications. Thus we can build 3-tier Web-based database application systems with high scalability and distributed processing capability. Database gateway in JaWAF works as a daemon process that connects to the database systems and waits for requests from clients, and supports state-oriented service between clients and database servers. Hence the performance of Web applications could be increasing.

※ 본 연구는 과학기술부 · 한국과학재단 지정 청주대학교 정보통신연구센터(RRC)의 지원에 의한 것입니다.
* 청주대학교 컴퓨터정보공학과

1. 서 론

1990년대 후반에 급속하게 성장한 WWW (World Wide Web, 이하 웹이라고 함)은 거의 모든 분야에서 적은 비용으로 정보를 전세계로 전파시킬 수 있도록 만들었다. 현재 웹은 수없이 많은 기업들의 컴퓨팅 환경을 변화시키고 있지만, 대부분의 기관들은 중요한 비즈니스 데이타와 관리 데이타를 모두 데이타베이스에 저장하므로, 극복해야 할 중요한 기술적 요소들 중 하나가 데이터베이스와의 연동이다. 웹 환경에서 가상금융 및 전자상거래 등과 같은 e-비즈니스 응용들을 효과적으로 구현하려면 이 문제에 대한 우수한 해결책이 개발되어야 할 것이다.

웹은 데이타베이스의 전위 시스템(front-end)으로서 많은 장점을 제공할 수 있다. 웹브라우저는 거의 모든 운영체제 플랫폼에서 작동하기 때문에 데이타베이스 시스템이 가지고 있는 다중 플랫폼 문제를 해결할 수 있다. 그러므로 웹 기반 데이타베이스 전위 시스템은 거의 모든 클라이언트로부터 데이타베이스 서버와 상호작동(interaction)이 가능하다. 더욱이 웹의 하이퍼미디어-기반 모델은 직관적이어서 대부분의 사용자에게 친숙하므로, 웹과 데이타베이스 연동 솔루션이 주어지면 웹은 데이터베이스 응용을 개발하기 위한 개방형 플랫폼으로 사용될 수 있다[1].

웹과 데이타베이스를 연동하는 방법으로 웹 서버와 외부 실행 프로그램간의 표준 인터페이스인 CGI(Common Gateway Interface)를 이용하는 방법이 지금까지 가장 많이 사용되고 있지만, 이 방식은 클라이언트가 요청할 때마다 서버쪽의 해당 CGI 프로세스가 매번 생성되기 때문에 사용자의 요청이 늘어나게 되면 서버에 많은 부하(load)가 발생하게 된다. 또한 클라이언트와 서버간의 통신시 HTTP를 사용하기 때문에 데이타베이스 시스템에서 기본적으로 요구되는 상호 관련된 연산간에 상태유지(stateness)가 이루어지지 않으므로 응용 개발자가 쿠키(cookie) 등을 이용한 추가적인 해결 방안을 모색해야 한다는 부담을 가지게 된다. 그리고 클라이언트로부터 요청이 들어올 때마다 매번 데이타베이스 연결·해제를 반복해야 하기 때문에 질의처리 비용이 상승하게 된다.

그러므로 최근 들어 웹과 데이타베이스의 효율적인 연동기술에 관한 심도 있는 연구들[1-10]이 국내외에서 진행되고 있으며, 이 연구들은 크게 두 가지 부분에 주안점을 두고 있는데, 하나는 효율적인 웹 데이타베이스 게이트웨이(gateway)의 구조와 다른 하나는 웹응용 프로그램을 쉽게 구현하는 방법이다[6]. 웹 데이터베이스 게이트웨이는 웹과 데이터베이스 시스템을 연동할 때 미들웨어(middleware)로 동작하는 프로그램이며, 웹을 통한 사용자 질의를 응용 프로그램으로 전달해 주는 역할을 수행한다. 데이터베이스 게이트웨이의 소스코드를 논리적으로 보면 미들웨어의 역할을 수행하는 게이트웨이의 제어논리(control logic) 부분과 응용 프로그램의 응용논리(application logic) 부분으로 나눌 수 있다.

이 연구들은 성능 향상을 위한 효율적인 데이터베이스 게이트웨이의 구조와, 확장 HTML [11]이나 확장 TCL(Tool Command Language) 또는 템플릿 파일(template file)을 이용하여 웹 응용을 쉽게 작성하는 방법과 같은 세부적인 관점에서만 논의하였을 뿐, 웹 환경에서 데이타베이스 응용을 개발하는 프로그래머의 관점에서 보다 쉽게 응용을 개발하는데 지침이 될 수 있는 웹 데이타베이스 응용 프레임워크(application framework)에 대한 연구는 미약하였다. 또한 이 연구들은 대부분 HTML을 기본으로 한 사용자 인터페이스를 제공하기 때문에 복잡한

응용에서 요구되는 대화식 웹 인터페이스를 구현하기 어렵고, 게이트웨이의 응용서버의 소스코드에 제어논리와 응용논리가 함께 섞여 있어 기술 숙련도가 낮은 응용 프로그래머가 이해하기 어려울 수 있다.

본 연구에서는 웹 환경에서 데이타베이스 응용을 개발할 때 기준이 될 수 있는 웹 데이타베이스 응용 프레임워크를 제시하였다. 이 프레임워크는 확장성(scalability)을 지원하는 3계층 클라이언트/서버 구조를 기반으로 하며, 각 계층에 속하는 객체들을 규정한다. 그리고 이 프레임워크는 클라이언트와 서버간에 상태유지(state-less)를 지원하며 클라이언트의 접근을 제어하여 투명하게 해당 응용서버로 연결시켜 주고 웹 응용의 성능을 향상시킬 수 있도록 대몬(deamon) 프로세스 형태로 응용서버를 운영해 주는 데이터베이스 게이트웨이와, 응용 프로그램 작성자가 쉽게 사용할 수 있는 통신 콤포넌트와 메시지 콤포넌트를 포함한다. 본 연구에서 제시한 웹 응용 프레임워크인 JaWAF(Java-based Web database Application Framework)는 Java를 기반으로 한 객체지향 클라이언트/서버 구조를 따르기 때문에 객체지향 소프트웨어 개발의 이점을 그대로 얻을 수 있다.

프로그래머는 웹응용 개발시 JaWAF를 적용하면 웹 환경에서 구동하는 데이타베이스 응용의 시스템 구조를 쉽게 이해할 수 있으며, 제공되는 통신 콤포넌트와 메시지 콤포넌트를 사용하여 쉽게 중간 계층과 데이터를 주고받을 수 있다. 또한 중간 계층에서 제공되는 데이터베이스 게이트웨이가 응용 프로그램을 응용서버 형태로 운영하기 때문에 웹응용의 성능을 높일 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 JaWAF 설계와 관련된 웹기반 응용 시스템, 웹 데이터베이스 연동시 고려사항, 3계층 클라이언트/서버 구조, 웹 게이트웨이 구조 등을 설명하고, 3장에서 JaWAF의 구조와 동작을 포함한 설계 요소를 설명한다. 그리고 4장에서 구현 요소와 JaWAF를 적용한 작은 응용의 예를 설명하고, 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

2.1 웹기반 응용 시스템

웹 환경은 복잡한 프로그램을 설치하지 않아도 일반적인 웹브라우저(Internet Explorer, Netscape 등)를 통해서 사용자가 원하는 특정 정보에 접근할 수 있는 장점을 가진다. 그래서 최근 이러한 웹 환경의 장점을 이용한 웹기반 응용 시스템들의 개발이 활발해지고 있다.

웹기반 응용이란 웹브라우저 환경에서 실행되는 컴퓨터 프로그램을 의미하며, 그러한 응용은 전세계 어디서라도 접근할 수 있는 완전한 플랫폼 독립성(cross-platform)을 지원한다. 응용 프로그램의 실행 환경으로써 웹브라우저가 가지는 장점은 설치가 용이하고 조작법이 직관적이고 매우 쉬우며, 거의 모든 운영체제에서 동일한 플랫폼을 제공한다는 점이다. 그러므로 웹응용 시스템의 사용자들은 원하는 정보를 쉽게 접근할 수 있다.

HTML을 기반으로 한 웹 시스템은 응용코드와 데이터를 중앙집중식 서버에서 관리하고, 클라이언트는 단지 사용자 인터페이스만을 제공하므로, 전통적인 클라이언트/서버 구조에서 나타나는 소프트웨어 배포, 버전 관리 등의 시스템 관리 문제를 해결할 수 있다. 그러나 복잡한 GUI를 요구하는 데이터베이스 응용에서는 HTML의 한계인 직관적이고 대화성(interaction)이 높은 사용자 인터페이스 설계 문제가 해결되어야 한다[12]. 그러므로 JaWAF에서는 사용자 인터페

이스 구현시 Java 애플릿을 사용하도록 규정하였다.

2.2 웹과 데이터베이스의 연동시 고려사항

웹과 데이타베이스의 효율적인 연동을 위해서는 다음 요소를 고려해야 한다[3].

- 조화로운 사용자 인터페이스(harmonised user interface) : 멀티미디어 데이터를 처리할 수 있어야 한다.
- 대화식 웹 모델(interactive web model) : 사용자 입력에 대해 즉각적인 응답(immediate reaction)을 보일 수 있어야 한다. 입력 오류 등에 대해 클라이언트쪽에서 처리함으로써 사용자에게 높은 대화성을 제공할 수 있고 서버쪽 작업부하를 감소시킬 수 있을 것이다.
- 일관성과 무결성 유지(consistency and integrity) : 데이터 조작은 데이터베이스의 일관성과 무결성을 보장하는 방식으로 수행되어야 한다. 이것은 웹브라우저를 질의 인터페이스로 사용하게 되면 HTTP의 비상태유지(stateless) 속성과 DBMS와 밀결합되지 않은 브라우저 때문에 문제가 발생할 수 있기 때문이다.
- 성능(performance)
- 확장성(scalability) : 사용자 요청이 증가함에 따라 시스템 확장이 용이해야 한다.
- 개방성(openness) : 새로운 데이터베이스나 브라우저 등과 같은 새로운 도구들이 시스템에 쉽게 통합될 수 있어야 한다.

2.3 3계층 클라이언트/서버 구조

90년대 중반에서 후반까지 수년동안 기업들은 2계층 구조를 갖는 제1세대 클라이언트/서버 시스템으로 수많은 정보 시스템을 구축해 왔다. 2계층 구조는 클라이언트 계층에 표현 논리와 응용논리를 배치하고, 서버 계층에 데이터베이스 서버를 배치한다. 그러나 이와 같이 표현논리와 응용논리를 모두 클라이언트에 배치함으로써 "Fat" 클라이언트가 가지는 시스템 관리상의 여러 문제점을 발생시켰다.

2계층 구조가 가지는 문제점들을 해결하기 위해 현재 수많은 개발 그룹들이 3계층 구조로 전환하고 있다. 3계층 구조는 클라이언트 계층에 표현논리를, 중간 계층에 응용논리를, 서버 계층에 데이타베이스 서버를 배치하여 표현논리와 응용논리의 분리, 응용논리의 중앙집중식 관리, 클라이언트/서버간에 균형적인 작업분담(load balance), 확장성(scalability) 제공 등의 장점을 가진다. 이 구조는 실행 성능의 향상 뿐 아니라 표현논리를 응용논리와 분리시킴으로써 사용자 인터페이스 설계자가 복잡한 응용논리의 구현에 대해 신경 쓰지 않아도 되므로, 보다 기능적이고 보다 효과적인 사용자 인터페이스의 설계에 집중할 수 있고, 그 결과로 작업의 생산성도 향상될 수 있다[13, 14].

2.4 Java기반의 웹응용 구조

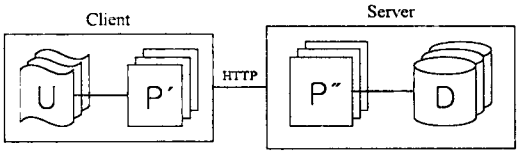
[15]에서는 전통적인 CGI 웹응용에서 나타나는 성능저하 문제를 해결하고, Java기반 웹응용을 개발하는데 적용할 수 있는 다음 3가지의 3계층 클라이언트/서버 구조를 제시하였다.

- Data Server 클라이언트/서버 구조
- Process Balanced 클라이언트/서버 구조
- Data/Process Balanced 클라이언트/서버 구조

이 중에서 Data Server 클라이언트/서버 구조는 클라이언트쪽에 응용논리가 포함되므로 클라이언트쪽의 작업부하 집중과 소프트웨어 배포 등의 "Fat" 클라이언트가 가지는 문제점이

나타날 수 있다. 소프트웨어 배포 문제 등을 해결하기 위해 Java 애플릿을 사용하면 애플릿의 크기가 커져서 적재(downloading)와 초기화에 많은 시간이 소요될 수 있다. Data/Process Balanced 클라이언트/서버 구조는 데이터 객체의 일부(D')가 클라이언트쪽에 위치하므로 다수의 클라이언트가 존재할 때 D'에 대한 일관성 유지가 어려울 수 있다.

본 연구에서 제시한 JaWAF는 "Thin" 클라이언트를 구현할 수 있는 Process Balanced 클라이언트/서버 구조를 기본 모형으로 한다. Process Balanced 클라이언트/서버 구조는 (그림 1)과 같으며, U는 사용자 인터페이스 계층(user interface layer)에 속하는 객체들을 나타내며, 관련된 사용자 인터페이스 데이터와 관련된 함수들을 포함한다. P는 프로세싱 계층(processing layer)에 속하는 객체들을 나타내며, U 객체들로부터 요청 받은 트랜잭션을 처리한다. 수행에 필요한 관련 데이터는 D 객체들로부터 전송 받는다. D는 데이터 계층(data layer)에 속하는 객체들을 나타내며, 데이터의 저장 및 관리 기능을 수행한다. 이 구조의 장점은 P 객체들이 클라이언트와 서버쪽에 분산되어 있어서, 균형적인 작업분담이 가능하고, 서버에서 실행되는 P"객체들을 모델링 할 수 있어서, 특정 도메인에서 요구되는 다양한 트랜잭션 모델들을 지원할 수 있다는 점이다.



(그림 1) Process Balanced 클라이언트/서버 구조

JaWAF에서는 "Fat" 클라이언트의 문제점을 해결하고 사용자와의 대화성을 높이기 위해서, P'에는 서버쪽 데이터와 독립적인 일부 정당성 검사(validation check) 논리만 포함시킨다.

2.5 웹 게이트웨이의 구조

데이타베이스 게이트웨이의 구조는 크게 서버쪽 확장 방식과 클라이언트쪽 확장 방식으로 분류할 수 있는데, 이 중에서 대표적인 것들은 서버쪽 확장 방식에 속하는 CGI 실행화일 방식, 확장 API 방식, CGI 응용서버 방식, 그리고 전용서버 방식이다[7]. 초기부터 현재까지 가장 많이 사용되고 있는 웹 데이터베이스 게이트웨이 구현방법은 CGI 실행화일 방식으로 Perl · TCL · C · C++ 등과 같은 다양한 프로그래밍 언어를 사용할 수 있으며, 응용 프로그램 개발이 비교적 단순하다. 또한 이 접근방식은 기술 발전이 빠르게 진행되는 인터넷 환경에서 새롭게 부상하는 표준에 종속적이지 않으며, 유지 보수에 매우 복잡한 지식을 요구하지 않는다는 장점을 가지고 있다. 그러나 클라이언트로부터 사용자 요청이 들어올 때마다 새로운 CGI 프로세스가 생성되고, 그 때마다 데이터베이스 연결 · 해제가 새롭게 이루어지기 때문에 사용자 요청이 증가하는 경우는 서버에 상당한 부하(load)를 발생시킨다. 이러한 문제 때문에 대규모 트랜잭션의 발생이 예측되는 경우에는 CGI 응용서버 방식이나 웹서버의 확장 API 방식을 사용하고 있다. 그러므로 이 절에서는 CGI 응용서버, 확장 API, 그리고 전용서버 방식을 사용하는 웹 게이트웨이 시스템들을 살펴본다.

성능면에서는 CGI 응용서버, 전용 API, 전용서버 방식이 모두 우수하며 [4-8]이 CGI 응용서버 방식을 사용하고, [9]가 전용 API 방식을, 그리고 [10]이 전용서버 방식을 사용한다. 그러나 전용 API나 전용서버 방식은 특정 웹서버에 종속적이기 때문에 특별한 의도가 없는 경우는 CGI 응용서버 방식이 이식성 측면에서 바람직하다.

응용 프로그램 작성 방법에서는 [4, 5]가 확장 TCL을 지원하고 [1, 2, 6, 7, 9]가 HTML 테그(tag)를 확장하여 데이터베이스 언어(SQL)를 포함하는 확장 HTML 접근방식을 사용하여 응용 프로그래밍의 편의성을 제공한다. 그러나 이 방식은 HTML을 기반으로 하기 때문에 다양한 응용 프로그램의 복잡한 논리를 지원하는데 여러 가지 제약이 발생할 것이다. [1, 9] 등에서는 범용 프로그래밍 언어로 작성된 응용 프로그램도 지원하여 이 문제를 해결하지만 응용 프로그램의 실행이 서버쪽에 집중되어 클라이언트/서버간에 작업부하의 분담이 어렵고, 기본적으로 HTML을 사용하기 때문에 [3]에서 지적한 높은 대화성이 요구되는 사용자 인터페이스를 지원하기 어렵다.

특히 사용자 플랫폼으로 사용되는 개인용 PC의 경우 현재 고성능(high performance) 워크스테이션임에도 불구하고 모든 처리가 서버쪽에 집중되는 것은 자원의 효율적 이용뿐 아니라 작업 집중에 따른 서버의 성능 저하가 빠르게 발생할 것이다. 또한, 표현 논리와 응용 논리가 하나의 프로그램 또는 스크립트 파일로 구현되므로 코드의 판독성이 떨어지는 문제점이 있다.

3. 설 계

3.1 JaWAF의 설계 정책

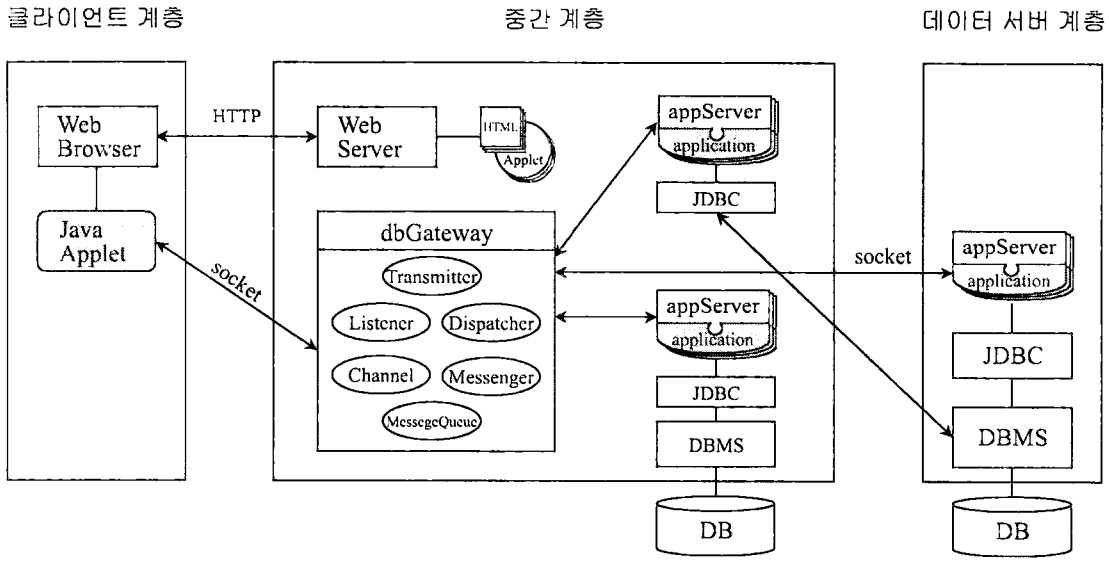
이 절에서는 JaWAF 설계시 고려한 설계 정책을 살펴본다. 이 정책은 [3]에서 제시한 웹과 데이터베이스 시스템의 연동시 고려사항들과 소프트웨어 공학 측면에서 고려사항들을 기반으로 하였다.

- 표준 기술을 사용하여 특정 시스템과의 독립성을 유지한다. 웹은 매우 빠르게 성장하는 기술분야이기 때문에 추후 발생할 수 있는 기술 변화에 따른 영향을 피한다.
- 3계층 클라이언트/서버 구조를 기반으로 하며, 클라이언트/서버간에 작업부하(workload) 를 분산시킨다. 그리고 표현논리 부분과 응용논리 부분을 분리하고, 중간 계층에서 데이 터베이스 게이트웨이의 제어논리 부분과 순수한 응용논리 부분을 가능한 분리시켜 응용 프로그램 작성자의 관점을 단순화시킨다.
- 사용자 인터페이스, 응용코드 등의 시스템 자원을 중앙집중식으로 관리할 수 있도록 한다. 이것은 소프트웨어 배포, 버전 관리 등 시스템 관리에 소요되는 막대한 유지보수 비용을 줄인다.
- 일반적으로 데이터베이스 연결 작업이 약 1~3초 정도 소요되므로[13], 응용서버와 데이터베이스간의 연결상태를 유지한다.
- 응용서버의 확장성(scalability)을 지원한다. 사용자 질의가 증가함에 따른 성능 저하를 막는다.
- 응용 개발 도구로서 Java를 사용하여 Java의 유용한 객체지향 장치들을 활용한다. Java에서는 다중 쓰레드, 소켓, 복합 객체, 추상 클래스 등 여러 유용한 객체지향 장치들을 제공하기 때문이다.

3.2 JaWAF의 전체 구조

이 절에서는 본 연구에서 제시한 3계층 구조의 Java기반 웹 데이터베이스 응용 프레임워크인 JaWAF(Java-based Web database Application Framework)의 구조를 설명한다.

JaWAF는 (그림 2)에 나타난 것처럼 클라이언트 계층, 중간 계층, 그리고 데이터 서버 계층으로 이루어졌으며, 각 계층에 속하는 객체와 기능은 다음과 같다.



(그림 2) JaWAF의 전체 구조

3.2.1 클라이언트 계층

클라이언트는 중간 계층과 데이터를 주고받기 위한 통신 객체와 메시지 객체를 기본적으로 포함한다. 그리고 표현논리를 나타내는 GUI 객체와 사용자와의 대화성(interaction)을 높이기 위해 서버쪽 데이터에 독립적인 데이터 유효성 검사의 일부를 포함하도록 규정하였다. GUI 객체는 HTML과 Java 애플릿으로 구현하도록 규정하였다. HTML만을 사용하면 대화성이 높은 사용자 인터페이스를 구현하기 어렵고, Java 애플리케이션(application)의 경우도 종래의 클라이언트/서버 구조에서 나타나는 소프트웨어 배포, 버전 관리 등의 시스템 관리 문제가 일부 발생할 수 있다. 특히 전자상거래 시스템에서는 불특정 다수 사용자가 접속하기 때문에 이러한 속성이 더욱 큰 문제를 야기할 것이다.

JaWAF에서 제공되는 통신 객체인 Channel은 소켓(socket)을 기반으로 하며 응용에 독립적으로 콤포넌트화 시켜서 사용자 인터페이스 설계자가 중간 계층과 통신하는 부분을 구현할 때 쉽게 재사용 할 수 있도록 하였다.

이 구조의 장점은 다음과 같다.

- 표현논리를 응용논리와 분리함으로써 표현논리 설계자는 응용논리의 구현에 대해 신경쓰지 않아도 되고, 시각적 Java 응용개발 도구들(Visual Age for Java, JBuilder 등)을 사용하여 보다 기능적이고 효과적인 사용자 인터페이스 설계에 집중할 수 있다.
- Java 애플릿을 사용하면 대화성이 높은 GUI 설계가 가능하고, 클라이언트/서버간의 작업부하 분산이 이루어지고, 시스템의 중앙집중식 관리를 통해 종래의 클라이언트/ 서버 구조가 가지는 소프트웨어 배포 등의 문제를 해결할 수 있다.

3.2.2 중간 계층

JaWAF에서 가장 핵심되는 부분은 중간 계층으로써, 사용자가 서버 자원(resource)에 관해 상세히 몰라도 클라이언트로부터의 정보 요청을 투명하게 처리해 준다. 데이터베이스 게이트웨이의 기능을 수행하는 중간 계층은 사용자의 정보 요청을 실제로 수행하는 응용서버인

appServer 객체(application server object)와 클라이언트의 요청을 적절한 appServer 객체로 전달하는 dbGateway 객체(database gateway object)를 포함한다. 또한 클라이언트와 중간 계층의 dbGateway 객체, 그리고 dbGateway 객체와 appServer 객체간에 메시지를 주고받기 위한 Channel 객체와 메시지 객체인 Messenger를 포함하며, 이들은 모두 콤포넌트화 되어 있다. dbGateway 객체에는 클라이언트의 요청을 제어하는 제어논리를 포함하며, appServer 객체에는 응용논리와 제어논리의 일부를 포함한다. appServer의 응용논리 부분은 정보 서비스 내용에 따라 응용 프로그래머가 작성하게 된다.

중간 계층은 응용논리를 포함하는 appServer 객체와 제어논리만을 포함하는 dbGateway 객체로 분리해서 응용 프로그램 작성자가 가능한 응용논리 부분에만 집중할 수 있도록 하였으며, 다른 외부 객체들과의 통신은 재사용 콤포넌트인 Channel 객체와 Messenger 객체를 통해서 이루어지므로, 응용 프로그래머 관점에서는 dbGateway 객체의 제어논리 부분이 투명하다.

3.2.3 데이터 서버 계층

데이터 서버 계층은 다양한 DBMS 객체를 포함할 수 있으며, 표준 인터페이스인 JDBC를 통해 접근하도록 규정하였다. 일반적으로 데이터베이스를 접근하는 방식은 확장 HTML, 템플릿 파일, 저장 프로시저, 독자적인 DBMS API, 그리고 표준 인터페이스(ODBC, JDBC) 등이 있으나[11, 16], 표준 인터페이스 접근방식의 장점은 이식성이 높고 분산처리 환경을 지원할 수 있으며, 클라이언트 계층으로부터의 요청이 증가함에 따라 데이터 서버 계층의 시스템 재구성이 매우 용이하다는 것이다.

3.3 dbGateway의 구조

dbGateway 객체는 클라이언트의 사용자 질의를 적절한 응용서버(appServer)로 연결해 주고, 그 실행결과를 다시 해당 클라이언트로 전달해 주는 중계자 역할을 수행한다. dbGateway 객체는 Listener, Dispatcher, Transmitter 객체로 구성되어 있으며, 각각은 쓰레드로 동작한다. 이 절에서는 이 객체들의 기능을 살펴보자.

3.3.1 리스너

리스너 객체(Listener)는 dbGateway가 초기 가동후 계속적으로 사용자 요청을 기다리다, 사용자 요청이 접수되면 그 데이터를 메시지큐(messageQueue)에 저장한다. 그런 다음 계속해서 다음 사용자 요청을 기다린다. messageQueue에 저장된 사용자 요청은 디스패쳐 객체가 주기적으로 읽어서 해당 서비스를 제공하는 appServer 객체로 전달하게 된다.

또한 리스너 객체는 포트관리와 클라이언트 관리를 수행하는데, 새로운 클라이언트가 접속 요청을 하면 클라이언트 번호(CID : Client ID)와 미사용 중인 포트번호를 할당한다. 클라이언트는 이후의 모든 작업 요청시 이 정보를 사용자 질의와 함께 Messenger 객체로 포장하여 리스너 객체로 전송한다.

3.3.2 디스패쳐

디스패쳐 객체(Dispatcher)는 주기적으로 messageQueue를 점검하며, 사용자 요청이 들어오면 Messenger 객체를 해석하여 해당 appServer 객체로 전달한다. 이 작업을 위해서 응용서버관리 테이블(appServer_table)과 포트관리 테이블(appServer_ports)에 각각 응용서버에 관한 정보와 포트번호를 관리한다. 이 정보는 시스템 관리자에 의해 특정 파일에 유지되며, 디스패쳐 객체가 기동되면 초기화 과정에서 읽

어들여서 메모리에 유지한다. appServer가 기동되면 자신의 "서비스 가능" 메시지를 디스패쳐 객체에 전달하게 되는데, 이때 디스패쳐 객체는 appServer_table에서 해당 응용서버 상태를 "on"으로 설정한다. 디스패쳐 객체는 사용자 질의가 접수되면 appServer_table을 참조하여 해당 응용서버의 상태 속성이 "off"로 설정되어 있으면 "서비스 불가능" 메시지를 리스너 객체로 전송한다.

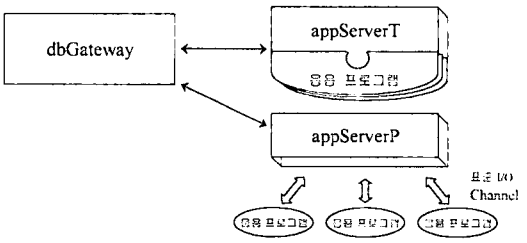
3.3.3 트랜스미터

트랜스미터 객체(Transmitter)는 리스너와 디스패쳐 객체에 비해 단순하게 설계되었으며, appServer 객체로부터 전달된 실행결과를 리스너 객체로 전송하여 해당 클라이언트에 전달되도록 한다.

3.4 응용서버

응용서버 객체(appServer)는 dbGateway 객체와는 별도의 프로세스로 동작하도록 설계되었으며 사용자 질의를 실제적으로 수행하는 응용 프로그램을 논리적으로 포함하는 객체이다. 기동을 하면 초기화 과정에서 자신의 "서비스 가능" 메시지를 dbGateway의 디스패쳐 객체에 전달하고 서비스 등록과정이 완료되면 디스패쳐 객체로부터 전달된 사용자 질의를 기다린다. appServer 객체는 정보 서비스 내용에 따라 다수가 존재할 수 있으며 각각 고유한 응용서버 ID(ASID : application server ID)를 가진다.

응용서버 객체는 (그림 3)에 나타난 것처럼 두 가지 형태로 설계하였다. 하나는 사용자 질의 요청이 빈번한 응용 프로그램과 한 쌍을 이루는 응용서버 객체 appServerT (appServerThread)인데, 이것은 응용서버 객체가 데몬 프로세스로 운영되면서 사용자 질의가 전달되면 응용 프로그램을 하나의 쓰레드로 생성하여 서비스를 수행한다. 다른 하나는 사용자 질의 요청이 빈번하지 않은 다수의 응용 프로그램과 논리적으로 한 쌍을 이루는 응용서버 객체 appServerP (appServerProcess)인데, 이것은 응용서버 객체가 데몬 프로세스로 운영되면서 사용자 질의가 전달되면 해당 응용 프로그램을 별도의 프로세스로 생성하여 서비스를 수행하고 사용자 접속이 해제되면 프로세스를 종료시킨다. appServerP는 제어논리 부분과 응용 프로그램이 완전히 분리되도록 설계되었기 때문에 응용 프로그램 작성자가 중간 계층을 신경 쓰지 않고 응용논리에만 집중할 수 있기 때문에 응용 프로그램 작성이 보다 단순해진다.



(그림 3) appServer의 구조

appServerT는 사용자 요청이 빈번히 발생하는 소수의 응용 프로그램들을 데몬 형태로 운영하기 때문에 사용자 질의에 대한 빠른 응답을 보장할 수 있는 장점이 있다. 그러나 appServerT 객체가 응용 프로그램 객체를 포함하여 하나의 프로세스로 동작하도록 설계되었고 다중 쓰레드를 지원하기 때문에 응용 프로그램 작성자의 관점이 조금 복잡해질 수 있다. 이 문제는 appServerT 객체에 다중 쓰레드를 포함하는 제어논리 부분을 잘 정의된 프레임 클래스(frame class)로 제공해서 응용 프로그램 작성자의 관점을 가능한 단순화시켰다. 응용 프로그래머는 이것을 상속받아서 해당 메소드를을 호출함으

로써 dbGateway 객체와 연동을 하게 된다.

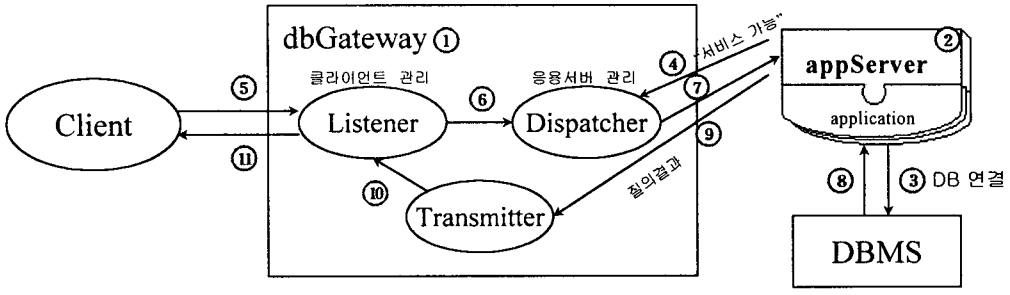
3.5 동 작

이 절에서는 JaWAF에서 중간 계층에 속하는 dbGateway 객체와 appServer 객체의 동작에 대해 살펴본다. 동작 과정은 (그림 4)에 나타나 있다. 기동 순서는 dbGateway 객체가 먼저 기동되며 초기화 과정이 완료된 다음에, 정보 서비스를 제공할 appServer 객체들이 하나씩 기동하게 된다. appServer는 적재가 완료되면 관계된 DBMS와 연결을 설정하고 "서비스 가능" 메시지를 dbGateway의 디스패쳐 객체로 전송한 다음, 디스패쳐로부터 전달될 사용자 요청을 기다리게 된다.

dbGateway는 트랜스미터, 디스패쳐, 그리고 리스너 객체 순으로 기동시킨다. 트랜스미터 객체는 기동돼서 초기화 과정을 마치면 appServer 객체로부터 전달될 질의결과를 기다린다. 디스패쳐 객체가 기동되면 appServer를 위한 포트관리 테이블(appServer_ports)과 응용서버관리 테이블(appServer_table)을 초기화한다. appServer_table은 시스템 관리자가 운영하는 지정된 파일로부터 정보를 읽어서 메모리에 복사본을 유지한다. 그런 다음, appServer로부터의 "서비스 가능" 메시지 또는 리스너 객체를 통해 전달되는 사용자 질의를 기다린다. 마지막으로 리스너 객체가 기동되는데 초기화 과정에서 클라이언트를 위한 포트관리 테이블(client_ports)과 클라이언트관리 테이블(client_table)을 초기화하고, 이 작업이 완료되면 클라이언트로부터의 사용자 질의를 기다린다.

이상과 같은 준비과정이 끝난 다음, 리스너 객체는 클라이언트로부터 접속 요청이 들어오면, 클라이언트 번호(CID)와 미사용 포트번호를 할당하고 사용자 질의를 기다린다. 이때 포트풀(port pool)에 가용 포트가 없으면 클라이언트의 접속 요청을 거절하는 메시지를 전송하게 된다. 클라이언트로부터 호스트이름, 클라이언트 번호, 포트번호 등의 정보가 사용자 질의와 함께 Messenger 객체로 포장되어 통신 객체(Channel)를 통해서 전송되면 이것을 디스패쳐 객체로 전달한다.

디스패쳐 객체는 Messenger 객체를 분석하고 appServer_table을 참조해서 해당 appServer의 서비스 가능 여부를 검사한다. appServer 서비스 상태 속성이 "off"이면 "서비스 불가" 메시지를 리스너 객체로 전송하고, "on"이면 해당 appServer 객체를 식별한 다음, Messenger 객체를 해당 appServer로 전달한다. appServer는 디스패쳐 객체로부터 전달된 Messenger를 해석하여 SQL문을 DBMS 객체로 전달하고, 다시 그 질의결과를 Messenger 객체에 정해진 형식



(그림 4) JaWAF의 동작 과정

으로 포장하여 트랜스미터 객체로 전달하고, 다음 사용자 질의를 기다린다. 트랜스미터 객체로 전송된 질의결과는 리스너 객체를 통해서 해당 클라이언트로 전달된다.

4. 구 현

JaWAF는 웹 상에서 3계층 클라이언트/서버 구조 기반으로 하여 다중 쓰레드, 소켓, 복합 객체, 추상 클래스 등 Java의 유용한 객체지향 장치들을 활용하여 구현하였다. 이 장에서는 JaWAF의 구현 내용과 JaWAF를 적용한 응용의 예를 살펴본다.

4.1 JaWAF의 구현

이 절에서는 JaWAF에서 가장 핵심이 되는 dbGateway, appServer, 그리고 Channel 객체의 구현 요소에 대해 살펴본다.

4.1.1 dbGateway

dbGateway는 리스너(Listener), 디스패쳐(Dispatcher), 트랜스미터(Transmitter) 객체를 포함하며 이들은 쓰레드(thread)로 구현하였다. 그리고 이들 간의 데이터 전송은 물론 외부 계층간에도 통신객체(Chammel)와 메시지 객체(Messenger)를 통해서 이루어진다.

Channel 객체는 내부에 소켓, 메시지큐 객체(messageQueue) 그리고 세마포어 객체(Semaphore)를 포함하며, 구현의 상세함을 Channel 객체로 포장했기 때문에 클라인언트 또는 응용서버쪽 응용 프로그램 작성자 관점에서는 사용법이 매우 단순하다. Channel 객체에 대해서는 뒷부분에서 설명한다.

클라이언트는 웹브라우저를 통해서 필요한 HTML 문서에 접근하면, Java 애플릿 객체가 웹브라우저에 적재되면서 시작된다. Java 애플릿 객체는 초기화 과정에서 미리 정해진 포트번호를 사용하여 Channel 객체를 생성한 다음 dbGateway의 리스너 객체에 접속을 요청하여 새로운 포트번호를 할당받는다. 클라이언트는 할당받은 포트번호를 사용하여 새로운Channel 객체를 생성한 다음, 이후의 모든 메시지 전송과 수신의 통로로 사용한다. 클라이언트로부터 전송되는 Messenger 객체는 메시지 헤더와 사용자 질의 부분으로 구성되며, 사용자 질의 부분은 SQL문 또는 HTML 폼(Form) 객체의 POST() 메소드에서 사용하는 "매개변수 + 값" 쌍 형태의 문자열로 표현된다. 포트관리 테이블(client_ports)과 클라이언트관리 테이블(client_table)을 현재는 Java의 해쉬 테이블로 구현하였다. 사용자 질의는 문자열 객체 형태로 전달되고, 질의결과는 결과 객체들의 벡터(vector) 형태로 포장돼서 전달되도록 구현하였다.

디스패쳐 객체는 주기적으로 메시지큐를 검사하고 FIFO(First Come First Out) 알고리즘을 사용하여 사용자 질의를 읽고, Messenger 객체 내의 헤더 정보를 해석하여 해당 appServer를 식별하고 존재 유무와 서비스 가능 여부 등을 검사한다. 이러한 매핑 정보는 응용서버 관리자(appServerManager)가 운영하며, 해당 appServer의 "서비스 가능" 속성이 "on"으로 설정되어 있으면 appServer의 호스트이름, 포트번호 등을 제공한다. 디스패쳐는 이 정보를 사용하여 Messenger 객체를 해당 appServer 객체로 전송한다. appServer는 dbGateway와 별도의 프로세스로 동작하며 dbGateway와 다른 플랫폼에 존재할 수도 있다. 이러한 구조는 다수의 응용 프로그램이 존재할 때 작업부하의 집중을 막고, 분산처리가 가능하다.

4.1.2 appServer

응용서버(appServer)는 dbGateway와 분리된

별도의 데몬 프로세스로 설계하였고 응용 프로그램 작성자의 편의성과 정보 서비스의 성능을 고려하여 두 가지 형태로 구현되었지만, 그 역할은 기본적으로 동일하다. appServer의 설계 원리는 응용논리를 중간 계층의 제어논리와 가능한 격리시키려 하였고, 기능은 dbGateway와 응용 프로그램 사이에서 매개체 역할을 수행한다. 논리적으로는 dbGateway와의 통신 및 메시지 전송 등을 appServer에 구현하여 dbGateway로부터 응용 프로그램을 격리시켰다.

appServer는 두 가지 형태(appServerT, appServerP)가 존재하는데 appServerT(appServerThread)는 응용 프로그램이 appServer 내에서 하나의 쓰레드로 동작하도록 하였고, appServerP(appServerProcess)는 사용자 질의가 들어오면 appServer에 의해 호출돼서 메모리에 적재되고 별도의 응용 프로세스로 동작을 하도록 하였다. 전자는 사용자 질의가 집중되는 소수의 응용 프로그램을 위한 것이고, 후자는 사용자 요청이 빈번하지 않은 다수의 응용 프로그램을 위한 응용서버 구조이며, appServer 자체는 두 가지 모두 데몬 프로세스로 구현되었다. 이것은 하나의 응용을 구성하는 수많은 응용 프로그램들이 모두 각각 하나의 데몬 프로세스로 동작하면 불필요한 작업부하를 생성할 수 있기 때문이다. appServerT와 appServerP 모두 dbGateway와 통신을 담당하여 질의결과를 Messenger 객체로 포장하여 전달하는 기능도 수행한다. appServerT는 응용 프로그램 쓰레드와 Channel 객체를 통해서 메시지를 주고받으며, appServerP는 Channel 객체 또는 표준 I/O 장치를 통해서 응용 프로그램과 사용자 질의와 질의결과를 주고받을 수 있다. 후자의 경우가 응용 프로그래머 관점에서는 중간 계층의 제어논리와 완전히 분리되기 때문에 프로그램 작성이 단순해지는 장점을 가진다. 다만 이 방식은 일부 플랫폼에서 버퍼크기에 제한을 받기 때문에 입·출력 데이터의 크기가 크지 않을 경우만 사용한다.

4.1.3 Channel

통신 객체(Channel)는 소켓을 기반으로 하며, 응용에 독립적이면서 사용이 용이하도록 고수준(high-level)으로 추상화시켰다(<표 1> 참조). Channel 객체 내부에는 원형큐를 기반으로 한 메시지 객체(messageQueue)를 포함하며 messageQueue 객체 내에는 동기화를 위한 세마포어 객체(Semaphore)를 구현하였다. 그러나 이러한 구현의 상세함은 Channel 객체로 포장되어 캡슐화되어 있기 때문에 클라이언트 또는 응용서버쪽 응용 프로그램 작성자 관점에서는 단순히 Channel 객체를 생성하고 인터페이스인 "send()" 또는 "receive()" 메소드를 호출하여 통신하기 때문에 투명하다. 그리고 멀티미디어 데이터를 비롯한 다양한 형태의 데이터를 범용 객체인 Messager로 포장하고 Java의 객체 직렬화(object serialization) 장치를 사용하여 전송하기 때문에 응용에 독립적이며, 콤포넌트화 되어있기 때문에 응용 프로그램 작성자도 쉽게 재사용할 수 있다. Channel 객체는 dbGateway와 appServer 뿐 아니라 응용 프로그램들도 dbGateway와 통신할 때 사용한다.

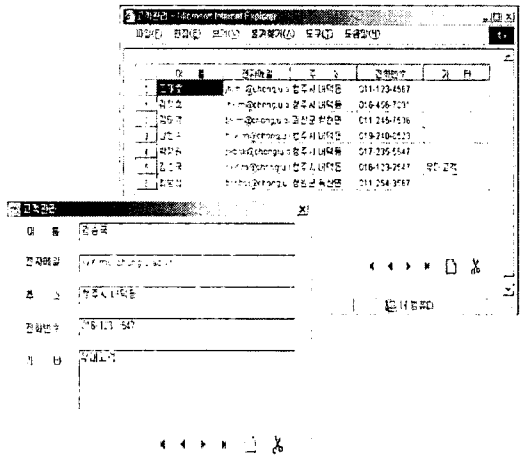
<표 1> 통신 객체(Channel)의 인터페이스

기 능	메소드 이름	형 식
전 송	send	(Messenger, hostname, port) (Messenger, port)
수 신	receive	( )

4.2 응용의 예

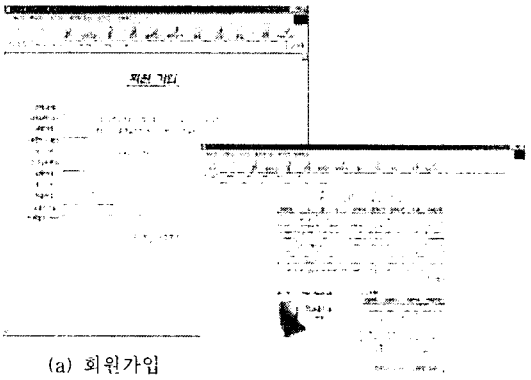
이 절에서는 웹응용 프레임워크인 JaWAF를 응용에 적용한 예를 간략히 살펴본다. 적용한 응

용은 전자상거래 시스템의 일부이며, 개발도구는 JDK 1.2.2, Inprise사의 JBuilder 3.0, Kawa 3.5를 사용하였다. 특히 시각적 개발환경을 지원하는 JBuilder를 사용하여 사용자 인터페이스를 설계하였고, (그림 5)에 나타난 네비게이션 버튼(navigation button)은 간단하지만 사용자에 대한 높은 대화성(interaction)을 지원한다. 이것이 HTML을 사용한 사용자 인터페이스와 다른 점은 사용자가 버튼을 클릭(click)하면 그것에 대한 처리를 서버쪽에서 수행하는 것이 아니고 클라이언트쪽에서 수행하기 때문에 대화성이 높고 서버쪽에 작업부하(workload)를 주지 않는다. 그리고 사용자 요청에 대한 빠른 응답성으로 인해 사용자의 만족도를 높일 수 있다.



(그림 5) 고객관리 서브시스템의

JaWAF의 또다른 특징 중 하나는 예상되는 사용자 요청 빈도수에 따라 응용 프로그램을 두가지 형태, 즉 쓰레드 또는 프로세스 형태로 운영할 수 있다는 점이다. (그림 6)은 경매 시스템의 예를 나타낸 것이다. 예상 트랜잭션의 발생 빈도수가 적은 (그림 6a)의 회원가입 프로그램은 응용서버 appServerP의 자식 프로세스로 운영되고, 예상 트랜잭션 빈도수가 높은 (그림 6b)의 입찰현황 조회 프로그램은 응용서버 appServerT 내의 쓰레드로 운영된다. 이와 같이 하나의 응용에 속하는 각 응용 프로그램들을 성격에 따라 응용서버 내의 쓰레드나 응용서버의 자식 프로세스로 운영함으로써 시스템에 대한 불필요한 부하 생성을 방지하여 응용 시스템의 전체적인 성능 향상을 기대할 수 있고, 사용자 요청이 높은 응용의 빠른 서비스를 보장할 수 있게 된다.



(a) 회원가입

(b) 입찰현황 조회

(그림 6) 경매입찰 서브시스템의 예

5. 결 론

웹은 데이터베이스 전위 시스템으로서 많은 장점을 제공할 수 있기 때문에 인터넷 응용 등에서 웹기반 데이터베이스 응용 프로그램 개발에 대한 요구가 급증하고 있다. 본 연구에서는 웹 환경에서 데이터베이스 응용을 개발할 때 적용할 수 있는 Java기반 웹 데이터베이스 응용 프레임워크인 JaWAF를 제시하였다. JaWAF는 3계층 클라이언트/서버 구조를 기반으로 하며, 각 계층에 포함되는 객체들을 규정하고 사용자 질의를 투명하게 응용서버로 연결해 주는 데이

터베이스 게이트웨이 객체(dbGateway), 그리고 응용 프로그램 작성자가 dbGateway와 데이터를 주고받을 때 쉽게 사용할 수 있는 고수준으로 추상화된 통신 콤포넌트(Channel)와 메시지 콤포넌트(Messenger)를 포함한다.

웹 환경에서 데이터베이스 응용을 개발할 때 JaWAF를 적용하면 웹응용의 시스템 구조를 쉽게 이해할 수 있고, 제공되는 데이터베이스 게이트웨이(dbGateway)와 통신 컴포넌트, 그리고 메시지 컴포넌트를 사용하여 확장성(scalability)이 높고 분산처리가 가능한 3계층 구조의 웹 데이터베이스 응용 시스템을 구현할 수 있다. JaWAF에서 제공되는 dbGateway 콤포넌트는 클라이언트와 응용서버간의 상태유지를 지원하며 웹응용의 성능을 향상시킬 수 있도록 데몬 형태로 응용서버를 운영해 주며, 사용자 관점에서는 사용자 질의를 투명하게 적절한 응용서버로 연결시켜 주기 때문에 정보 서비스 접근이 매우 쉽다. 또한 JaWAF는 분산처리가 가능한데 dbGateway 객체만 웹서버와 같은 플랫폼에 존재하면 appServer 객체는 다른 플랫폼에 존재할 수 있기 때문에 서버쪽 작업부하가 늘어남에 따른 성능 저하를 대처할 수 있다. 이것은 시스템 관리자가 해당 응용서버(appServer)를 원하는 플랫폼으로 이동하여 실행시키면 소스코드 수정 없이 실현되며, 프레임워크가 Java 응용을 기반으로 하기 때문에 플랫폼 변동에 따른 제약은 해결할 수 있다.

JaWAF는 청주대학교 정보통신연구센터에서 수행 중인 전자상거래 시스템과 인터넷 응용 개발을 위한 응용 프레임워크로 활용하기 위해 개발되었으며, 앞으로는 우선순위를 기반으로 한 효율적인 스케쥴링 알고리즘의 적용과 특정 응용서버에 대한 작업부하의 집중을 모니터링하여 분산시킬 수 있는 작업부하의 균형(load balancing) 기술을 적용할 계획이다.

참 고 문 헌

[1] Khoshafian, S., et al., The Jasmine Object Database Multimedia Applications for the Web, Morgan Kaufmann Publishers, 1999.

[2] Nguyen, T., V. Srinivasan, "Accessing Relational Database from the World Wide Web," Database Technology Institute IBM Santa Teresa Laboratory, ACM SIGMOD, 1996.

[3] Gerald Ehmayer, Gerti Kappel, "Connecting Databases to the Web : A Taxonomy of Gateways," DEXA97, 1997.

[4] Kim, W.-S., "BADA-III/Web : Integration of the Web and an OODBMS," Proceedings of the 32nd Hawaii International Conference on System Sciences, 1999.

[5] 김평철, "UniWeb - 웹을 이용한 클라이언트-서버 데이터베이스 응용 개발 환경", 데이터베이스 저널, 제3권 제2호, 1996.

[6] 최일환, 이상철, 김형주, "SRP RDBMS를 위한 Web 게이트웨이", 한국정보과학회 논문지(C), 제5권, 제1호, 1999.2.

[7] 김은경, 황병연, "WWW 데이터베이스 인터페이스를 위한 UCM(United CGI Management) 시스템의 설계", 한국정보처리학회 논문지, 제6권 제8호, 1999.8.

[8] 최원익, 김형주, 이석호, "웹 응용을 위한 자바 질의 스텁의 구현 및 성능 평가", 한국정보과학회 논문지(C), 제5권 제6호, 1999. 12.

[9] 이재길 외 3인, "오디세우스 객체지향 멀티미디어 데이터베이스 관리 시스템을 위한 웹 데이타베이스 게이트웨이의 설계 및 구현", 한국정보과학회 봄 학술발표논문집, 제27권 제1호, 2000.

[10] http : //www.oracle.com/ip/deploy/ias/ind

ex.html?content.html
[11] Rowe, J., Building Internet Database Servers with CGI, New Riders, 1996.
[12] Black, B., "OLTP on the Internet, Internet Systems", 1996.10.
[13] Marc A. Mnich, Multi Tier Architectures for Database Connectivity, JavaExchange.com, White Paper, Jan. 5, 1998.
[14] Tom Kim, "Looking for a 3-Tier App Builder", Java Developers Journal, Vol.3, No.1, Jan. 1998.
[15] 김수동, "Java기반 인터넷 어플리케이션 아키텍쳐 및 설계기법", 한국정보과학지, 제16권 제4호, 1998. 4.
[16] Ken North, "Building Web Databases : Tools and Techniques for Web Database Developers," Web Techniques, Sep. 1996.
[17] 구홍서, "웹과 데이터베이스 통합 메카니즘에 관한 연구", 청주대학교 산업과학연구, 제17권 제1호.
[18] 구홍서, "WWW과 데이터베이스 연동기술의 조사분석", 한국정보과학회지, 제18권, 제4호, April 2000.

저자소개

구 홍 서

인하대학교 전산학과를 졸업하고, 인하대학원에서 석사와 박사 학위를 취득하였으며, 현대전자산업(주)에서 MRP II 시스템 개발 업무를 담당하였다. 현재 청주대학교 이공 대학 컴퓨터정보공학과 조교수로 재직하고 있으며 주요관심분야는 데이터베이스, 컴포넌트 소프트웨어, 웹기반 정보 시스템 분야이다.