웹 트랜잭션 처리 시스템의 구현
(An Implementation of a Web Transaction Processing System)

이 강 우 † 김 형 주 ††
(Kangwoo Lee) (Hyungjoo Kim)

요 약 웹은 자금까지의 문자 위주의 응용에서 벗어나, 그림, 동영상과 같은 멀티미디어 자료를 브라우저를 통해 접근할 수 있도록 하여, 점차 많은 응용분야에서 사용되고 있다. 최근에 들어서는 데이타베이스와 웹의 연동을 통해, 보다 다양하고, 양질의 웹 서비스를 지원하고자 하는 노력으로, 많은 연구자들이 웹 통로에 대한 많은 연구로 좋은 결과를 내고 있다. 그러나 웹과 데이타베이스의 연동에 있어 필요한 웹 트랜잭션 처리시 발생하는 문제에 있어서는 아직까지 만족할 만한 연구결과가 제시되지 않고 있다.

본 논문에서는 웹 통로에서 웹 트랜잭션 처리를 위한 시스템인 WebTP를 제안하고, 이를 구현하였다. WebTP는 워크, 워크-전역변수 등을 도입하여, 웹 응용 단위에서의 저장점 설치와 부분 철회와 트랜잭션 상태 관리 기법을 제공하여, 시스템의 가용성과 안정성을 높이며, 구조적인 웹 응용작성이 가능토록 하여, 복잡한 웹 응용 개발을 돕는다.

Abstract Enabling users to access multi-media data like images and motion pictures, in addition to the traditional text-based data, Web becomes a popular platform for various applications. Recently, many researchers have paid a large amount of works on integrating Web and databases to improve the quality of Web services, and have produced many noticeable results on this area However, no enough research results have been presented on processing Web transactions that are essential in integrating Web and databases.

In this paper, we have designed and implemented WebTP, which is a Web transaction processing system for Web gateways to databases. WebTP, by introducing work and work-global-variables, provides a more robust state management, supports application-level savepoints and partial rollbacks. It also nhances productity by providing a modular way to develop Web applications

1. 연구 동기 및 배경

인터넷은 기존의 여러 종류의 네트워크를 서로 연결하여 구성된 하나의 거대한 인터네트워크(inter-net-work)이다. 인터넷은 표준 프로토콜인 TCP/IP을 바탕으로 사용자들로 하여금 여러 서로 다른 네트워크상의 많은 정보와 자원을 공유하였다. 제공되는 주요 서비스로는 전자우편(e-mail), TELNET, FTP 등이 있다. 최근에 들어서는 WWW(World Wide Web: 앞으로 웹으로 칭한다)의 등장으로, 보다 다양하고 양질의 서비스 제공이 가능하게 되어 현재 많은 수의 사용자들이 인터넷을 통하여 유용한 정보를 얻게 되었다.

웹은 일반적인 문자 자료뿐만 아니라, 그림, 동영상과 같은 멀티미디어 자료를 비교적 단순한 데이타 전송 프로토콜과 쉽게 구할 수 있는 브라우저를 이용하여 검색/접근할 수 있도록 하여, 보다 다양한 웹 응용들이 사용자들에게 여러 양질의 서비스를 제공하는 것이 가능하게 되었다. 최근 들어 효율적인 웹과 데이타베이스와의 연동 기술에 관한 심도 있는 연구[2,4,5,8,10,12]가 활발하게 진행되어 많은 수의 데이타베이스 연동 웹 응용들이 등장하게 되었다. 이러한 예로, 가상 쇼핑 몰, 사이버 서점, 사이버 예약 서비스 시스템 등을 들 수 있다. 그러나 데이타베이스 연동 웹 응용은 그 기능이나 성능면에 있어 아직까지 만족할 만한 성과를 보여주지 못하

† 비 회 원 서울대학교 컴퓨터공학과
kwlee@oopsla.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
hjk@oopsla.snu.ac.kr
논문접수 : 1998년 6월 22일
심사완료 : 1998년 9월 17일

고 있는데, 그것은 데이타베이스 연동 웹 응용을 위한 트랜잭션[6,7,9] 처리에 있어 많은 문제가 있기 때문이다. 트랜잭션은 지속적인 데이타를 다수의 사용자가 접근하는 응용에 있어 매우 중요한 개념이지만, 웹에서 사용하는 상태 비보존 프로토콜인 HTTP와 동적으로 HTML 문서를 생성하는 경우 주로 사용하는 CGI의 한계로 인하여 웹 응용에서는 극히 제한적으로 사용되고 있다[3]. 이를 해결하기 위해서는 데이타베이스 연동 웹 응용을 위한 트랜잭션 처리(앞으로는 웹 트랜잭션이라 칭한다.)에 관한 보다 심도 있는 연구가 필요하다

본 논문에서는 웹 통로를 위한 보다 효율적이고 다양한 기능의 웹 트랜잭션 처리를 지원하는 시스템인 WebTP을 제안한다. WebTP는 페이지 함수와 워크 개념을 도입하여 복잡한 웹 응용을 구조적으로 개발할 수 있도록 하고, 웹 응용 수준의 저장점(savepoint) 설치와 이를 이용한 부분 철회(partial rollback)[11] 기능을 지원한다. 또한, 워크 스크립트를 바탕으로 엄격한 상태관리 기법을 제공한다.

본 논문의 구성은 다음과 같다. 2절에서는 기존의 웹과 데이타베이스 연동에 관한 연구와 기존의 웹 트랜잭션 처리 기법에 대해 언급한다. 3절에서는 WebTP가 기존의 웹 트랜잭션 처리 방법의 문제점을 해결하기 위한 몇 가지 주요 기술에 대해 언급을 하고, WebTP를 기존의 웹 트랜잭션 처리 시스템과 비교 평가한다. 4절에서는 이러한 기술의 실제 구현에 관해 설명을 한다. 마지막으로 5절에서는 본 논문의 결과를 설명하고, 향후 연구 계획에 대해 언급한다.

2. 관련 연구

2.1 웹 통로의 분류

웹에서 주로 웹 통로를 이용하여 데이타베이스를 접근하는 방법을 사용한다. 웹 통로는 웹이 어느 위치에서 데이타베이스를 접근하는가에 따라 여러 구조로 분류되고, 이에 따라 성능이 달라지게 된다[10]. 본 논문에서 사용하는 분류방법은[10]에 의거한 것으로, 웹 통로는 서버쪽 확장과 클라이언트쪽 확장으로 구분된다. 서버쪽 확장은 CGI 이용 방식, 서버 API 이용방식, 그리고 전용 서버 방식으로 나누고, 클라이언트쪽 확장은 외부 뷰어를 이용하는 방식과 브라우저 확장 방식으로 나누게 된다. CGI 이용방식은 CGI 실행 화일 방식과 CGI 응용서버 방식으로 나누게 된다. [2,10]의 결론에 의하면, CGI 응용서버 방식을 사용하는 웹 통로가 웹 표준기술인 CGI를 이용함과 동시에 DBMS의 최적화 기능을 활용할 수 있어, 시스템의 성능을 높일 수 있다고 주장하였다.

2.2 초기 웹 트랜잭션 처리 방법

초기의 웹 트랜잭션 처리는 CGI 실행 화일 방식의 웹 통로에서 사용된 트랜잭션 처리 방법이다. 매 사용자 요구마다 생성되어 수행되는 CGI 실행 화일에 데이타베이스 인터페이스(API 내지는 ESQL 등)를 이용하여, CGI 실행 화일이 수행하는 동안 하나의 트랜잭션을 처리하는 방법이다. 이러한 방법은 기존의 Web 서버/클라이언트, HTTP 등의 기술을 수정 없이 사용할 수 있어, 매우 쉽게 웹 트랜잭션을 구현할 수 있는 장점을 갖고 있다.

그러나 이 방법은 실제로 대규모의 웹 응용으로 사용하기에는 다음과 같은 치명적인 단점을 갖는다. 첫째, CGI 실행 화일 종료시 데이타베이스 연결을 끊어야 하는 제약으로, 트랜잭션의 수행 기간이 한번의 CGI 실행을 넘을 수 없어, 실제 응용에 자주 발생하는 비교적 복잡한 트랜잭션을 구현할 수 없다는 단점을 갖는다. 둘째, 매 사용자 요구마다 CGI 실행 프로세스가 생성되어, 데이타베이스 연결을 수행하기 때문에, 다수의 사용자가 동시에 사용하는 경우에는 과도한 데이타베이스 연결/단절로 인한 오버헤드와 많은 수의 CGI 프로세스의 생성으로 인한 thrashing이 발생으로 시스템의 급격한 성능 저하로 대규모 트랜잭션 처리에 매우 치명적인 문제라 할 수 있다. 셋째, 하나의 웹 응용을 위한 여러 CGI 실행 화일들을 논리적으로 그룹화 하는 개념이 없고 그들 간의 정보 공유가 힘들어 복잡한 웹 응용 개발이 힘들게 된다.

2.3 UniWeb 2.0: UniSQL을 위한 웹 통로

UniWeb 2.0은 객체관계형 데이타베이스 시스템인 UniSQL/X 용 웹 통로[2,4]로 범용 스크립트 언어인 Tcl[13]을 이용하여 데이타베이스를 접근할 수 있도록 되어있다. UniWeb 2.0은 CGI 응용서버 방식을 통해, 웹 표준기술인 CGI를 이용함과 동시에 DBMS의 최적화 기능을 활용할 수 있도록 하여, CGI 실행 화일 구조의 통로의 성능 및 기능상의 문제점을 극복하였다. 즉, 하나의 UniTcl 응용서버로 하여금 한 사용자로부터의 모든 요구를 전담하도록 하여, CGI 실행 화일 수행 때마다 데이타베이스의 접속 및 단절로 인해 유발되는 부하를 줄이고, 여러 대화로 구성된 트랜잭션 처리를 가능하게 하며, 웹 응용의 문맥(context)이 UniTcl 서버에 유지되어, 스크립트 사이의 정보 공유가 Tcl 변수를 통해 가능하다는 장점을 갖는다.

그러나 웹 응용을 구성하는 스크립트들의 구조적으로 관리하는 기능이 없고, 또한 스크립트간 자료의 공유가

암시적이고, 응용서버 수준의 저장점 설치와 부분철회를 지원하지 않아, 여러 스크립트로 구성된 복잡한 웹 응용의 개발에 어려움이 있다. 또한, HTTP의 상태 비보존 프로토콜을 극복하기 위한 명확한 상태 관리 기법이 제공되지 않아, 비정상적인 상태로의 전이를 방지하는 해결책이 제시되지 못하고 있다.

2.4 DB2 WWW Connection

객체 관계형 DBMS인 IBM DB2에서는 웹과의 연동을 위해 DB2 WWW Connection[12]을 구현하였다. 데이타베이스 연동 웹 응용을 작성하는 경우, 개발자가 CGI 및 DBMS API에 대한 지식을 필요로 하는 문제와 작성된 코드에 대한 이해가 어려운 단점[12]을 해결하기 위해, DB2 WWW Connection에서는 상호-언어 변수치환(cross-language variable substitution)의 방식을 제공하여, 웹 응용 개발자는 HTML과 SQL만을 사용하여 응용 프로그램을 작성할 수 있도록 하였다 또한, HTML과 SQL 부분이 각기 나뉘어 있기 때문에 쉽게 코드를 이해할 수 있다는 장점을 갖는다.

그러나 상호-언어 변수치환 방식으로 동적으로 생성할 수 있는 HTML 문서에 한계로 다양한 HTML 문서를 생성할 수 없고, DB2 WWW Connection을 위한 트랜잭션 처리 기법에 대해서는 구체적으로 언급되어 있지 않고 있다.

2.5 Oracle Application Server

Oracle Application Server(OAS)는 Oracle사가 차기 데이타베이스 연동 웹 응용을 위해 개발한 제품으로, Oracle DBMS와의 밀접한 결속을 바탕으로 안정성과 확장성 향상을 꾀하였다[8]. OAS는 카트리지(cartridge) 개념을 이용하여, PL/SQL, Java, C, Perl, COBOL 등과 같은 다양한 언어로 웹 응용을 작성할 수 있도록 하였다. 또한, 웹 요구 중계자(Web Request Broker' WRB)를 도입하여 시스템의 부하균형(load balancing)을 실현하였으며 여러 사이트에 걸쳐 수행되는 응용 서버에 사용자의 요구를 수행시킬 수 있도록 하여 분산 처리 기능을 제공한다.

OAS에서도 한 사용자의 요구를 처리하는 실행모듈 사이에 공유되는 자료의 명시적인 방법이 제공되지 않아, 웹 응용 수준의 저장점 설치와 부분 철회에 관한 언급이 없으며, 한 응용에 속하는 실행 모듈을 그룹화 하는 기능이나 모듈들에 대한 정보 부족으로, 구조적인 웹 응용 개발이 어렵고 상태 관리에 많은 어려움이 있다.

2.6 SWeS

SWeS[5]는 SNU RDBMS Platform(SRP)[1]을 위한 웹 통로이다. SWeS는 CGI 응용서버 방식을 채택하여, 기존의 CGI 실행 화일 방식의 성능 저하 문제를 해결하였다. SWeS는, 기존의 CGI 응용서버 방식과는 달리, 하나의 응용서버(여기서는 SWeS서버)가 오직 한 사용자에 할당되는 것이 아니라, 여러 사용자의 요구를 처리할 수 있도록 하여 응용서버의 수가 사용자에 비해 상대적으로 적은 경우에도 원활히 작동할 수 있도록 하고, 응용서버에 고장이 발생하는 경우에는 해당 사용자의 요구를 다른 서버로 하여금 처리하도록 하여, 시스템의 가용성을 높였다. 또한, 질의에 대한 결과로 나오는 행의 개수를 지정해 주어, SWeS 서버가 한 사용자를 위해 오랫동안 사용중인 상태로 되는 경우를 막는 부하조절방식으로, 시스템의 확장성과 동시성을 높여 대규모의 사용자 수를 지원하는 웹 환경에 적합한 구조를 제시하였다. SWeS는 DB2 WWW Connection과 유사하게 HTML과 SQL만을 이용하여 웹 응용을 개발할 수 있도록 하여, DB2 WWW Connection의 장점을 함께 갖도록 하였다.

그러나 SWeS에서도 웹 트랜잭션에 대한 구체적인 언급은 없고, 여러 대화로 구성된 웹 응용을 작성하는데 한계가 있어, 제한된 상황의 경우에서만 여러 대화로 구성된 웹 응용을 하나의 트랜잭션으로 처리가 가능하고, 대화간의 자료의 공유와 전달이 극히 제한적이다.

3. WebTP의 개요

3.1 페이지 함수 및 워크

WebTP 에서의 트랜잭션은 하나 이상의 페이지 함수의 수행으로 구성된다. 페이지 함수는 웹 브라우저부터의 요구에 해당하는 하나의 HTML 문서를 동적으로 생성하는 함수로, 임의의 프로그래밍 언어로 구현될 수 있다[1]. 일반적으로 페이지 함수는 데이타베이스를 접근하여 사용자의 요구에 맞는 HTML 문서를 동적으로 생성한다.

그림 1은 SRP의 Tcl 언어 바인딩인 SRPTcl을 이용하여 작성된 페이지 함수의 예를 보여준다. 본 예는 사용자로부터 고객이름, 고객주민등록번호, 예약할 비행기좌석의 수를 입력받아, 다음 페이지 함수인 "logorder"를 호출할 수 있는 HTML 문서를 생성하는 페이지 함수이다 페이지 함수에서는 HTTP 에서 사용되는 질의 전달방식인 GET, POST 방식과 명령어 인자 전달방식을 통하여 전달되는 정보를 사용할 수 있다. 그림 1에서는 명령어 인자 전달방식으로 전해진 질의를 접근하여,

1) 현재의 WebTP 구현에서는 Tcl 언어로 작성된 페이지 함수만을 지원한다

```
proc fillorder { argv, argc } {
    global wtglobal

    set wtglobal(FLIGHTNO) [lindex $argv 0]
    set maxseats [lindex $argv 1]

    WebTP_puts "Content-type: text/html\n\n
            <html><body>
            <form method=get action=WebTP/logorder>
                    Name:  <input type=text name=cname><br>
                    ID: <input type=text name=cid><br>
                    # of seats: <input type=text name=nseats value=
                      $maxseats><br>
                    <input type=submit> <input type=reset>
            </form></body></html>"
}
```

그림 1 SRPTcl을 이용하여 작성된 페이지 함수의 예

사용자가 이전의 페이지 함수에 정한 비행기편과 해당 비행기의 최대 가용 좌석 수를 얻는 것을 볼 수 있다. 본 예에서는 나타나지는 않았지만, GET과 POST 방식으로 전달된 질의 값은 페이지 함수 호출 이전에 파싱(parsing)되어, Tcl 배열인 wtquery를 통하여 페이지 함수에서 사용될 수 있게 된다.

```
DATABASE kwlee kwleedb
GLOBAL FLIGHTNO
GLOBAL CUSTNAME
GLOBAL CUSTID
GLOBAL NSEATS = 0
BEGIN selectflight
COMMIT commit_rsvflight
ABORT abort_rsvflight
PAGE selectflight:fillorder
PAGE fillorder:logorder
PAGE logorder
```

그림 2 비행기 좌석 예약을 위한 워크의 스크립트

일반적으로 하나의 데이타베이스 연동 웹 응용은 하나 이상의 HTML 문서를 통한 사용자 상호작용으로 이루어지게 된다. WebTP에서는 이를 위해 워크(work)를 제공한다. 워크는 하나 이상의 페이지 함수의 순차적인 수행으로 이루어진 논리적 작업의 단위로, 웹 응용의 단위가 되며 트랜잭션 수행의 단위가 된다. 즉, 초기의 웹 트랜잭션과는 달리 트랜잭션이 하나 이상의 CGI 실행에 걸쳐 수행될 수 있어, 복잡한 응용 프로그램을 하나의 트랜잭션 영역 안에서 수행하는 것이 가능하게 된다. 하나의 워크에 속한 여러 페이지 함수들은 서로 밀접하게 관계를 갖는 경우가 흔하여, 그들 사이의 자료 교환 및 공유가 빈번하게 된다. WebTP 에서는 이를 위해 워크-전역변수(work global variable)를 도입하였다. 워크-전역변수는 하나의 워크에 속한 페이지 함수들이 공유하는 자료로, 이를 통해 페이지 함수사이의 정보 공유가 일어나게 되어, 복잡한 WebTP 응용을 쉽게 작성할 수 있도록 한다.

그림 2은 비행기 좌석 예약을 위한 워크의 스크립트를 보여준다. 'DATABASE' 구문은 워크 수행 중에 접근하게 될 데이타베이스의 이름과 접근하는 사용자의 이름을 기술한다. 'GLOBAL'로 시작하는 구문은 워크에서 사용될 워크-전역변수를 정의하는 구문으로, 본 예에서는 비행기편(FLIGHTNO), 고객이름(CUSTNAME), 고객의 주민등록번호(CUSTID), 요구한 예약 좌석 수(NSEATS)를 정의하고 있다. 이러한 워크-전역변수 사용은 웹 응용 작성시, HTML 문서간의 자료 전달을 위해, 쿠키를 사용하거나, HTML의 FORM 문에서의 hidden value를 사용하는 번거로움을 없앨 수 있다 워크를 구성하는 페이지 함수들과 그들 간의 수행순서 관계는 'PAGE' 구문을 통해 기술되며, 형식은 'PAGE 페이지-함수-이름:다음-페이지-함수-이름'을 따른다. 'BEGIN'는 처음으로 수행될 페이지 함수를 지정하는 구문이다. 그러므로, 본 예에서의 워크는 selectflight→fillorder→logorder 순서의 페이지 함수가 수행되는 것을 알 수 있다. 'COMMIT'과 'ABORT' 구문은 워크를 수행하는 트랜잭션이 완료되거나 철회된 후, 불리는 페이지 함수를 기술한다. 이 페이지 함수들은 데이타베이스와의 연결을 끊은 후 수행되기 때문에, 이 함수 수행 중에는 데이타베이스를 접근할 수 없다. 워크 스크립트에서 지정한 페이지 함수가 없는 경우에는, WebTP는 트랜잭션의 완료/철회 후, 시스템 default 페이지 함수를 호출한다.

3.2 WebTP의 기본구조

WebTP는 그림 3와 같이, Stub, FlowMgr와 Agent로 구성되어 있다. Stub은 웹 브라우저(사용자)의 요구에 의해 웹 서버에 의해 생성되어 수행되는 CGI 실행 화일이다. Stub은 사용자로부터 워크 수행에 필요한 여러 가지 정보(질의인자, 쿠키 및 수행할 페이지 함수 이름)를 받아 FlowMgr에게 전달하는 작업과, FlowMgr로부터 전달되는 결과 HTML 문서를 웹 브라우저에게 전달하는 단순한 역할을 한다. Stub은 매 사용자 요구마다 생성되고 한번의 요구가 완료되면 소멸되어, 여러 웹 브라우저로부터 다수의 요구가 발생되는 경우 서버에 많은 부하가 유발될 수 있다. 그러나 Stub은 UniWeb 2.0에서의 디스패처와 유사하게 사용자의 요구를 단순하게 FlowMgr에게 전달하는 작업만을 수행하기 때문에 실행 화일이 매우 작고, 데이타베이스 접속과

같은 복잡하고 많은 시간을 소모하는 작업을 하지 않기 때문에 stub의 생성 및 소멸로 인한 부하는 그리 크지 않게 된다[2,4][2].

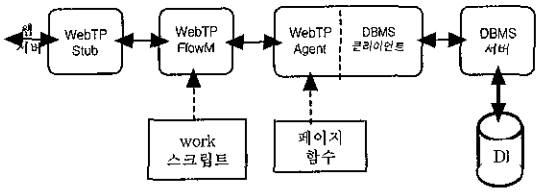


그림 3 WebTP의 실행 구조

Agent는 사용자가 요구하는 페이지 함수를 실제로 수행시켜 결과 HTML 페이지를 생성하는 모듈로, 필요에 따라 데이타베이스에 접속하여 필요한 자료를 검색/갱신하기도 한다. Agent는 하나의 워크에 배타적으로 바인딩되어 수행되며 워크의 수행이 끝날 때까지 다른 사용자의 워크를 위해 사용될 수 없게 된다. Agent는 사용자가 요구한 페이지 함수의 스크립트를 동적으로 적재하여 수행한다.

FlowMgr는 WebTP의 가장 핵심적인 작업을 수행하는 모듈로, WebTP에서의 워크 상태의 관리, 워크 수행 제어, 워크-전역변수 관리의 역할을 맡는다. FlowMgr는 워크 수행시 생성되어 배타적으로 할당되며, 수행되는 워크의 수행기간 동안 지속되게 된다.

3.3 워크 수행 제어

WebTP에서는 워크 스크립트에서 기술된 순서의 페이지 함수를 모두 수행시킨 상태에서만 해당 워크를 완료시킬 수 있다. 워크가 완료되면 그때까지 워크의 수행으로 갱신된 데이타베이스는 완료되고 워크의 수행이 종료된다. 워크의 철회는 워크 수행 중 임의의 시점에서 수행될 수 있다. 워크가 철회되면, 그때까지 워크의 수행으로 갱신된 데이타베이스는 완전 철회(total roll-back)되고, 워크의 수행이 종료되게 된다.

WebTP에서는 응용 수준의 저장점 설치와 부분 철회를 지원한다. 일반적으로 데이타베이스 응용 수준에서의 저장점 설치 작업과 이를 이용한 부분 철회는 실현하기 어렵다[9]. 데이타베이스의 상태는 DBMS가 제공하는 저장점 설치 기능과 이를 통한 부분철회를 통해 실현 가능하지만, 일반 데이타베이스 응용의 수행 상태를 저장하고, 이것을 다시 복원시키는 작업은 많은 문제점을 유발한다. 그러므로 대부분의 시스템에서는 극히 제한적인 응용의 경우(예를 들면, 대화형 SQL 처리 쉘)로 제한하거나, 저장점 설치와 부분 철회시 응용 프로그램의 수행 상태를 저장하거나 복원하는 작업을 전적으로 응용 프로그램에게 맡기는 방법을 사용하고 있다. 그러나 WebTP에서는 페이지 함수들 사이의 워크의 문맥이 워크-전역변수에 의해 결정되기 때문에, 정확한 워크-전역변수 값의 변화를 추적하면 워크의 수행상태의 기록 및 복원이 가능하여 워크(즉, 웹 응용) 수준의 저장점 설치와 부분 철회가 가능하게 된다.

저장점 설치시 기록되는 정보는, 워크 수행으로 변화된 데이타베이스의 상태와, 변화된 워크-전역변수 값들이다. WebTP는 Agent로 하여금 데이타베이스의 상태를 DBMS의 저장점 설치 기능을 통해 기록되게 하고, 워크-전역변수 값들은 FlowMgr에 의해 따로 기록하게 된다. 지정된 저장점으로의 부분 철회는 Agent 로 하여금 DBMS에서 제공하는 부분 철회기능을 이용하여 데이타베이스 상태를 전이시키고, FlowMgr로 하여금 기록된 워크-전역변수를 복원시키는 방법으로 수행된다.

3.4 상태 관리

상태 비보존(stateless) 프로토콜인 HTTP를 사용하는 웹 트랜잭션 처리의 경우, 사용자와의 여러 대화로 구성된 웹 응용의 상태를 관리하는 일은 어려운 문제이다. [3,4]에서는 웹에서의 상태 관리에 있어 해결해야 할 여러 이슈로 상태 유지, 상태 해제, 상태 보호, 비정상 상태 전이 방지를 들었다. 이 중 비정상 상태 전이 방지를 제외한 이슈에 대한 해결책은 어느 정도 제시되어 있으나, 비정상 상태 전이 방지를 위한 뚜렷한 방법이 제시되고 않고 있다. 특히, 웹 환경에서는 웹 브라우저의 탐색 방법에 따라 이전에 수행한 페이지 함수를 다시 수행하거나, 또한 워크의 중간 페이지 함수부터 수행하는 경우가 흔하게 발생하여 이에 대한 해결책이 필요하다.

WebTP에서는 이러한 문제를 해결하기 위해, 워크 스크립트에서 기술된 순서에 어긋나는 페이지 함수 수행을 막아, 사용자가 후진 탐색 및 기타 무작위의 페이지 함수의 수행으로 워크의 상태가 비정상적으로 전이되는 것을 막게 된다[3].

예를 들어 그림 2와 같이 정의된 워크 수행 중에 페

2) Stub의 잦은 생성과 소멸로 유발되는 오버헤드를 보다 완화시키기 위해, 웹 서버 API를 사용하는 방법도 고려하고 있으나, 구현은 되지 않았다.

3) 현재 WebTP에서는 이를 위해, 한 워크 내에 두개 이상의 같은 이름의 페이지 함수를 호출할 수 없다 만일, 같은 작업을 수행하는 페이지 함수가 필요한 경우에는, 각각 다른 이름의 부여하여 사용하게 된다. 우리는 이러한 현재 이러한 제약점을 극복하는 방법에 대해 연구를 수행하고 있다.

이지 함수 fillorder까지 수행된 상태에서, 웹 브라우저에서의 후진 탐색으로 페이지 함수 selectflight 수행을 다시 요구한 경우를 가정하자. FlowMgr에서는 다음으로 수행할 페이지 함수를 logorder를 기대하기 때문에 다른 페이지 함수 수행 요청을 에러로 처리하여 select-flight의 중복된 수행으로 워크의 상태가 비정상적으로 전이되는 것을 막게된다. WebTP에서는 워크의 부분 철회가 명시적인 방법(4.4절 참조)으로 행하여지기 때문에 [4]에서 언급된 후진 탐색과 트랜잭션 부분 철회 사이의 모호성이 유발되지 않는다.

위와 같은 사용자의 무작위의 페이지 함수의 수행으로 유발되는 비정상 상태 전이 이외에도 웹 환경에서의 여러 가지 고장으로 인해 웹 트랜잭션의 상태가 비정상적으로 전이될 수 있다. 이러한 문제에 대한 해결책은 주로 시간제한(timeout)방법을 근간으로 하여 고장이 발견되면 해당 워크를 철회시키는 방법을 사용한다. 지면 관계상 좀더 자세한 해결책은 다음 논문에서 언급하기로 한다.

3.5 WebTP의 비교 평가

WebTP는 기존의 여러 웹 통로에서의 웹 트랜잭션 처리 방법과 비교할 때, 다음과 같은 장점을 갖는다. 첫째, WebTP는 초기 웹 트랜잭션 처리 시스템과는 달리, 상태 비보존 HTTP의 한계를 극복하여 여러 대화로 이루어진 워크를 하나의 트랜잭션 하에서 수행될 수 있어 복잡한 웹 응용을 쉽게 개발할 수 있도록 한다.

둘째, WebTP에서는 Agent가 데이타베이스에 접속되면 웹 응용이 종료될 때까지 유지되기 때문에 DBMS 최적화 기술을 사용할 수 있어, CGI 실행 화일 방식을 사용하는 시스템에서 CGI 실행 화일의 잦은 생성/소멸, 데이타베이스 접속/단절로 인해 시스템의 성능저하를 방지할 수 있다.

셋째, 논리적으로 서로 연관된 여러 페이지 함수들을 묶어 하나의 논리적인 작업의 단위인 워크를 도입하여, 웹 응용을 구조적으로 개발할 수 있도록 하였다. 즉, 기존에는 여러 CGI 실행 화일사이의 관계가 명시적으로 기술되지 않고 그들이 생성하는 HTML 문서의 링크로부터 유추되는 암시적인 관계만 존재하였으나, WebTP에서는 워크를 이용하여 서로 연관된 페이지 함수를 하나로 묶을 수 있고, 그들간의 수행순서도 기술할 수 있다.

넷째, 하나의 웹 응용(즉, 워크)을 이루는 페이지 함수간의 자료 공유가 워크-전역변수를 통해 명시적으로 나타나기 때문에, 페이지 함수간의 상호작용을 명확히 이해할 수 있어 유지보수가 용이하게 된다.

다섯째, WebTP에서는 앞서 설명한 바와 같이 기존의 웹 트랜잭션 구현에 있어서의 문제 중 하나인 상태 관리에 대한 해결책을 제시한다. WebTP는 워크 스크립트로부터 수행해야 할 페이지 함수들과 그들의 실행 순서를 파악하기 때문에 응용서버의 상태가 비정상적으로 전이되는 일을 막을 수 있다.

현재 구조의 WebTP는 UniWeb 2.0과 같은 CGI 응용서버 방식과는 달리, FlowMgr의 추가로 인해 생성된 HTML 문서가 웹 브라우저로 전달되기까지 한단계를 더 거치기 때문에, 사용자 요구 처리 시간이 다소 길어지는 단점을 갖는다. 물론 WebTP에서도 FlowMgr와 Agent의 기능을 합쳐 하나로 만드는 구조를 생각할 수 있으나, 추후 WebTP의 분산 트랜잭션 및 워크플로우(workflow)의 지원을 위한 확장을 위해 현재의 구조를 갖도록 하였다. 분산 트랜잭션을 처리하게 되는 경우, FlowMgr가 2단계 완료규약의 조정자의 역할을 맡게 된다.

4. WebTP의 구현

WebTP는 서울대학교 객체지향 연구실에 개발된 RDBMS인 SRP[1]와 연동하여 구현하였다. 현재는 SRP에서 제공하는 SRPTcl을 이용하여 작성된 페이지 함수만을 지원하고 있다. SRPTcl은 범용 스크립트 언어인 Tcl을 이용하여 SRP 데이타베이스를 접근할 수 있도록 확장한 인터프리터 언어로, 동적으로 프로그램을 적재하여 실행하는 것이 용이하다. 본 절에서는 앞에서 언급한 WebTP기능을 위한 구현에 관련된 내용을 설명한다.

4.1 워크 상태 관리

표 1은 현재 구현된 WebTP에서 수행되는 워크의 상태를 구성하는 변수들이다. wt_mgrid는 워크 수행을 위해 할당된 FlowMgr의 식별자로, FlowMgr가 Stub과의 통신을 위해 개방한 TCP 포트 번호이다. 이 값은 쿠키에 기록되어 웹 클라이언트와 FlowMgr사이의 가상적인 연결을 유지하게 된다. 기타 나머지 상태 변수들은 FlowMgr 프로세스의 내부에서 기록/관리된다.

wt_user는 워크를 수행하는 사용자의 이름으로, WebTP에서 워크를 처음 수행시키려 할 때 로그인 과정에서 얻어지고, 이를 통해 권한 인증/부여 작업을 수행한다 wt_work는 현재 수행 중인 워크의 이름을 기록하는 변수이고, wt_pagefunc는 최근에 종료된 페이지 함수의 이름을 기록하는 상태 변수이다. WebTP에서는 워크 스크립트에 기술된 수행순서 상으로 wt_pagefunc 다음으로 위치하는 페이지 함수의 수행만을 허락하여 워크가 비정상적인 상태로 전이되는 것을 막는다.

표 1 WebTP에 사용된 워크 상태 변수들

상태 변수 이름	내 용	비 고
wt_user	FlowMgr의 식별자	쿠키에 기록
wt_user	WebTP 사용자 이름	FlowMgr에 기록
wt_work	수행 중인 워크 이름	FlowMgr에 기록
wt_pagefunc	마지막으로 종료된 페이지 함수 이름	FlowMgr에 기록
wt_svptlist	저장점 이름 리스트	FlowMgr에 기록

위의 상태 변수들과 워크-전역변수들은 페이지 함수 수행 중에 사용되기 때문에, 이들은 페이지 함수가 수행될 때마다 Agent로 전달되어 페이지 함수 수행시 각각 wtglobal과 wtcontext라는 이름의 Tcl 배열로 사용될 수 있도록 한다. 워크-전역변수는 페이지 함수의 수행으로 값이 변경될 수 있어, 페이지 함수 종료 후 Agent에서 FlowMgr에게 생성된 HTML 페이지를 전송할 때, 변경된 워크-전역변수의 값들도 함께 전송하여 FlowMgr로 하여금 변경된 워크-전역변수를 반영하도록 한다. Tcl 배열 wtcontext는 읽기 전용으로 구현[4]되어, 페이지 함수 수행시 참조만 하도록 한다.

4.2 워크 수행을 위한 FlowMgr의 생성 및 Agent의 할당

그림 4는 워크가 시작되는 경우, 수행에 필요한 FlowMgr와 Agent가 할당되는 단계를 보여준다. WebTP에서 새 워크를 시작시키기 위해서는 Stub의 실행 화일의 경로를 포함한 URL을 이용하고 그 형식은 "<Stub 경로>/<워크이름>" 이다. 예를 들어

http://www.snu.ac.kr/8080/WebTP/reserveflight[5]

는 reserveflight라는 워크를 시작시키는 URL이다.

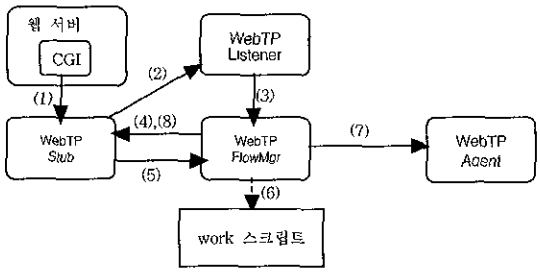


그림 4 워크 수행을 위한 FlowMgr와 Agent 할당

Stub은 웹 서버로부터 PATH_INFO을 통해 시작할 워크의 이름을 알아낸다(1). Stub은 listener에게 워크 수행에 필요한 FlowMgr의 생성과 Agent의 할당을 요구한다(2).

Stub으로부터 워크 시작 요청을 받은 listener는 자신에게 등록된 Agent들 중에서 유휴 상태인 Agent를 검색하여 이러한 Agent가 있는 경우에는 FlowMgr를 생성하고, 이 FlowMgr에게 검색된 Agent를 배당한다(3). 만일, 유휴 상태인 Agent가 없는 경우에는 해당 Stub에게 더 이상 사용할 수 있는 Agent가 없다는 사실을 알리게 된다.

생성된 FlowMgr는 앞으로 Stub과의 통신에 사용할 새로운 TCP 소켓의 포트번호를 OS로부터 할당 받아 wt_mgrid 상태 변수에 기록하고 해당 Stub에게 전달한다(4). Stub은 이 포트를 이용하여 FlowMgr와 TCP 연결을 생성하여 워크 이름 등의 정보를 전달한다(5). FlowMgr는 전달받은 워크 이름으로 해당 워크의 스크립트 화일을 읽어들여, 워크 수행에 필요한 워크-전역변수들에 대한 초기화와 페이지 함수의 수행 순서를 내부 자료구조에 기록하는 등 워크 수행에 필요한 여러 준비 작업을 수행하게 된다(6). 또한, FlowMgr는 listener로부터 배당 받은 Agent와의 TCP 연결을 생성하고 워크의 첫번째 페이지 함수의 수행을 요청하여(7), 생성된 HTML 화일을 Agent로부터 전달받아 Stub에게 전달해 준다(8). 이때, FlowMgr는 wt_mgrid를 쿠키로 등록하여, 웹 브라우저와의 연결을 설정한다.

4.3 페이지 함수 실행

페이지 함수 수행을 위한 URL의 형식은 "<Stub 경로}>/<페이지함수이름>/?<인자리스트>" 이다. 예를 들어,

http://www snu.ac.kr/8080/WebTP/fillorder/
?flightno=1234

는 두번째 페이지 함수 flightno를 수행시키는 URL으로, 인자 flightno의 값으로 1234를 전달하는 것을 알 수 있다. Stub은 웹 서버로부터 CGI를 통해 전달받은 위 형태의 URL를 분석하여, 수행시킬 페이지 함수 이름과 인자 리스트들을 받아 FlowMgr에게 전달한다.

그림 5과 그림 6는 FlowMgr와 Agent에서 일반적인 페이지 함수가 처리되는 과정을 의사 코드로 보여준다. FlowMgr는 Stub으로부터 페이지 함수 이름과 인자리스트 들을 받아 먼저 요구한 페이지 함수가 적합한지를 조사한 후, Agent에게 페이지 함수 이름, 인자 리스트, 워크-전역변수, 그리고 기타 상태 정보를 전달하고 해당

4) Tcl의 'trace variable' 명령어를 이용하여 읽기 전용 변수로 구현하였다.

5) 본 구현에서는 사용자의 편의를 위해 Stub 실행 화일의 이름을 WebTP로 하였다.

페이지 함수의 수행을 요청한다. 페이지 함수의 수행 결과 Agent로부터 반환되는 HTML 페이지와 변경된 워크-전역변수 리스트를 받아, HTML 페이지는 Stub에게 전달하고 변경된 워크-전역변수 값은 해당 전역변수에 반영시킨다.

```
FlowMgr로부터 페이지함수 이름(pfunc), wtquery, wtcontext,
wtglobal을 수신.
if pfunc에 해당하는 페이지 함수 존재하지 않으면 then
    pfunc에 해당하는 Tcl 페이지 함수를 적재.
end if
wtquery를 인자로 하여 해당 Tcl 페이지 함수 pfunc를 호출
페이지 함수 수행으로 생성된 HTML 페이지와 변경된
wtglobal을 FlowMgr로 전송
Tcl 배열 wtquery, wtcontext, 그리고 wtglobal을 제거
```

그림 5 Agent에서의 페이지 함수 수행

```
Stub을부터 페이지 함수 이름(pfunc)과 인자 리스트(wtquery)를
받는다
if next(wt_pagefunc) <> pfunc then
    웹 서버에게 부적절한 페이지 함수 수행을 알리는 HTML
    페이지 전송
end if
워크-전역변수를 이용하여 Tcl 변수 wtglobal을 구성
워크의 상태 변수들을 Tcl 변수 wtcontext를 구성
해당 Agent에게 pfunc, wtquery, wtcontext, wtglobal을 전달
Agent로부터 페이지 함수 수행으로 생성된 HTML 페이지와 변
경된 wtglobal을 수신함
wt_pagefunc ← pfunc
변경된 wtglobal을 워크-전역변수에 반영
웹 서버에게 생성된 HTML 페이지 전달
```

그림 6 FlowMgr에서의 페이지 함수 수행 요구 처리

Agent는 요구된 페이지 함수를 수행시켜 HTML 문서를 생성한다 페이지 함수 수행 중 FlowMgr로부터 전달받은 워크-전역변수 및 기타 상태 정보들을 이용하기도 하고, 데이타베이스를 접근하여, 워크-전역변수들의 값과 데이타베이스 내의 데이타가 변경시키게 된다. 생성된 HTML 문서와 변경된 워크-전역변수 값들은 FlowMgr에 반환한다.

4.4 저장점 설치 과정 및 부분 철회 과정

WebTP에서의 저장점 설치 과정과 부분 철회는 사용자의 요구로 행해진다. 사용자의 요구는 "<Stub 경로>wt_trans/save"와 "<Stub 경로>/wt_trans/rollback/?wt_savept=<복귀할-저장점이름>" 형식의 URL로 표현된다. 예를 들면,

http://www.snu.ac.kr/8080/WebTP/wt_trans/save
http://www.snu.ac.kr/8080/WebTP/wt_trans/rollback/?wt_savept=fillorder

는 현재 수행 중인 워크에 저장점을 설치하고, 저장점 fillorder까지의 부분 철회를 요구하는 URL이다.

사용자가 저장점 설치를 요구하게 되면, FlowMgr는 현재의 워크-전역변수 리스트를 다른 메모리 영역에 복사하고, Agent에게 DBMS의 저장점 설치 작업을 요청한다. 복사된 영역의 주소와 Agent에서의 DBMS 저장점 설치로 생성된 저장점 식별자[6]를 쌍으로 하여 상태변수 wt_svptlist에 추가한다. 추가된 쌍의 이름은 가장 최근에 수행된 페이지 함수의 이름인 wt_pagefunc로 부여하고, 이것을 WebTP 저장점 이름으로 정한다.

부분 철회를 요청 받은 FlowMgr는 저장점 이름을 이용하여 wt_svptlist에서 워크-전역변수 리스트가 저장된 영역과 DBMS 저장점 식별자를 알아낸다. DBMS 저장점 식별자는 Agent에게 전달하여 데이타베이스의 상태를 지정된 저장점까지 부분 철회를 시키고, 복사된 영역에서 저장된 워크-전역변수들의 값을 현재의 워크-전역변수에 overwrite하여 워크를 이전의 상태로 복원시키는 방법으로 부분 철회 과정을 마치게 된다. 이때, 철회된 저장점 이후에 생성된 모든 저장점들은 wt_svptlist에서 제거된다.

```
Stub으로부터 워크 완료/철회 요구를 받음
if 완료인 경우 then
    if 워크를 구성한 모든 페이지 함수를 수행하지 않으면
      then
        오류 HTML 페이지를 구성하여 웹 서버에게 전달
    end if
    Agent에게 트랜잭션 완료를 요청하고, 해당 HTML 페이지
    를 수신
else
    Agent에게 트랜잭션 완전철회를 요청하고, 해당 HTML 페
    이지를 수신
end if
wt_mgrid를 쿠키에서 제거함
수신된 HTML 페이지를 웹 서버에게 전달
FlowMgr의 수행을 종료시킴
```

그림 7 FlowMgr에서의 워크 종료 과정

4.5 워크 종료 과정

워크를 완료 또는 완전 철회시키기 위해서 각각 "<Stub 경로>/wt_trans/commit"과 "<Stub 경로>/wt_trans/abort" 형태의 URL을 사용한다. 예를 들어,

http://www.snu.ac.kr/8080/WebTP/wt_trans/commit

6) 일반적으로 DBMS에 저장점이 설치되면, LSN과 같은 저장점의 식별자가 생성된다.

http://www.snu.ac.kr/8080/WebTP/wt_trans/abort 는 각각 워크를 완료, 완전 철회시키는 URL이다.

그림 7과 그림 8는 워크가 완료 또는 철회되는 경우 FlowMgr와 Agent에서 수행되는 작업을 보여주는 의사 코드이다. FlowMgr는 쿠키에 기록된 wt_mgrid를 제거하여, 웹 클라이언트와 맺어진 접속을 단절하고, 자신의 수행을 종료시킨다. 이때, FlowMgr가 자신의 메모리 영역에서 관리하던 모든 상태 변수들을 제거되게 된다.

워크가 완료되거나 완전 철회되면 해당 웹 트랜잭션의 수행이 끝나게 되므로, Agent는 접속된 데이타베이스의 연결을 끊고, 유휴상태로 전이되어 다른 워크 수행 시 사용될 수 있게 된다.

```
FlowMgr로부터 완료/철회 요구를 수신
if 요구가 완료이면 then
    트랜잭션을 완료시킴
else
    트랜잭션을 철회시킴
end if
if 지정된 완료/철회 페이지 함수가 존재하면 then
    지정된 페이지 함수를 수행하여 HTML 페이지 생성
else
    default 페이지 함수를 수행시켜 HTML 페이지 생성
endif
생성된 HTML 페이지를 FlowMgr에게 전달
데이타베이스와의 접속을 단절시킴
자신의 상태를 유휴상태로 함
```

그림 8 Agent에서의 워크 종료 과정

5. 결론 및 향후 연구 계획

본 연구에서는 기존의 웹 응용에서 웹 트랜잭션 처리 시 야기되는 여러 문제를 해결하는 웹 트랜잭션 처리 시스템인 WebTP를 구현하였다. WebTP는 워크 개념을 도입하여 복잡한 웹 응용을 구조적으로 개발할 수 있도록 하고, 여러 대화로 구성된 웹 응용을 하나의 트랜잭션으로 처리될 수 있도록 하였다. 워크를 구성하는 페이지 함수의 실행순서를 명시하도록 하여, WebTP로 하여금 명시된 순서에 어긋나는 페이지 함수의 실행을 막아 시스템이 비정상적인 상태로 전이되는 상황을 막을 수 있다. 또한 워크-전역변수를 도입하여, WebTP로 하여금 웹 응용의 문맥을 직접 관리할 수 있도록 하여 웹 응용 단위에서의 저장점 설치와 이를 이용한 부분철회를 지원하게 된다. 또한, WebTP는 CGI 응용서버 방식을 사용하여, 초기의 CGI 실행 화일 방식을 사용하는 시스템에서의 성능문제를 해결함과 동시에 웹 표준인 CGI의 여러 장점을 유지할 수 있다. 실행 중 필요한 페이지 함수를 동적으로 적재하는 기능으로, 시스템의 운영 중에도 중단없이 새로운 페이지 함수 및 워크의 추가가 가능하여 시스템의 가용성을 높이는 장점을 갖는다.

현재 WebTP의 저장점 설치 기능을 확장하여 지속적 저장점(persistent savepoint)[9]을 지원하는 방안과 이를 바탕으로 roll-forward 고장회복 기법을 추가하는 연구를 수행할 계획이다. 이를 통해, 시스템에 고장이 발생하는 경우 단순히 워크를 철회시키지 않고 최근의 지속적 저장점부터 계속 수행토록 하여, WebTP의 가용성을 높이는 연구를 진행 중이다.

감사의 글

본 연구에 많은 도움을 준 서울대학교 컴퓨터공학과 객체지향 연구실 SRP, SOP 연구팀원들에게 감사의 말을 전합니다.

참 고 문 헌

[1] 이강우, 안정호, 김형주, "확장용이 클라이언트-서버 RDBMS의 설계및 구현", 한국정보과학회 SIGDB, 겨울 1994, http://wwwoopsla.snu.ac.kr/publication/

[2] 김평철, 민영훈, "웰드와이드웹용 데이타베이스 통로의 성능 평가", 데이타베이스 연구회지, 제13권, 제2호, pp.20-39, 1997, http://grigg.chungnam.ac.kr/tech-reports/.

[3] 김준, "웹에서의 데이타베이스 트랜잭션", Technical Memo(TM-96-U-10), 충남 대학교 정보통신 공학과 데이타베이스 연구실, July 1996. http://grigg.chungnam.ac.kr/tech-reports/.

[4] 김평철, "UniWeb 2.0-웹을 이용한 클라이언트-서버 데이타베이스 응용 개발 환경", 데이타베이스 저널, 제3권, 제2호, pp119-132, 1996

[5] 최일환, 이상철, 김형주, "SRP RDBMS를 위한 Web 통로", 정보과학회 논문지(C) 게제예정. http://wwwoopsla.snu.ac.kr/publication/

[6] Philip A. Berstein, Vassos Hadzilacos, and Nathan Goodman, "*Concurrency Control and in Database Systems,*" *Addison-Wesley Publishing Company*, 1987

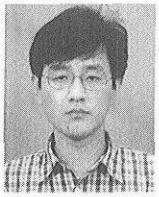
[7] Philip A. Berstein and Eric Newcomer, "*Principles of Transaction Processing,*" Data Management Systems, Morgan Kaufmann Publishers Inc., 1997

[8] Oracle Corporation, "*Oracle Application Server 4.0,*" White Paper(Draft), February, 1998

[9] Jim Gray and Andreas Reuter, "*Transaction Processing: Concepts and Techniques,*" Morgan Kaufman Publishers, Inc., 1993

[10] Pyung-Chul Kim, "*A Taxonomy on the Architecture of Database Gateways for Web,*" Proc. of the 13th Intl. Conf. on Advanced Science and Techno-

logy(ICAST97)}, pp. 226-232, April 1997. http://grigg.chungnam.ac.kr/tech-reports/.
[11] C. Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, and Peter Schwarz, *"ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging,"* ACM Transactions on Database Systems, 17(1):94-162, March 1992.
[12] T. Nguyen and V. Srinivasan, *"Accessing Relational Databases from the World Wide Web,"* Proc. of the ACM SIGMOD Conf. on Management of Data, June 1996
[13] Brent B. Welch, editor, "Practical Programming in Tcl and Tk," Prentice Hall PTR, 2nd edition, 1997



이 강 우
1991년 2월 서울대학교 계산통계학과(이학사). 1993년 2월 서울대학교 계산통계학과 전산과학전공(이학석사). 1993년 3월부터 현재까지 서울대학교 전산과학과 박사과정. 관심분야는 트랜잭션 처리 시스템, 질의어 처리 시스템, 객체관계형 데이타베이스 시스템

김 형 주
제 5 권 제 1 호(C) 참조