웹 트랜잭션 처리 시스템의 일관성 유지 지원

(Support of a Web Transaction Processing System for Preserving Consistency)

이 강 우 † 김 형 주 ††
(Kang-Woo Lee) (Hyoung-Joo Kim)

요 약 웹의 등장으로 전자 업무 처리는 급격한 성장을 보여왔다. 많은 업계에서는 업무 처리에 있어 핵심이 되는 트랜잭션을 웹에서 처리하기 위한 시스템 개발에 많은 노력을 기울여왔다. 그러나 대부분의 웹 트랜잭션 연구는 주로 CGI 방식의 트랜잭션 처리 성능의 한계를 극복하거나, 여러 HTTP 요청간의 상태 유지 방법 또는 시스템의 확장성 등에 대해 연구가 집중된 반면, 부적합한 웹 환경으로 유발되는 시스템의 일관성 손실에 대해서는 많은 관심을 받지 못했다.

본 논문에서는 웹 상에서 신뢰성 있는 트랜잭션을 처리하는 경우 발생되는 세 가지 문제점을 지적하고, 이를 막기 위해 웹 트랜잭션 처리 시스템인 WebTP에서 제공하는 해결책을 제안한다. 제안한 해결책은 쿠키 로깅, 웹 페이지 로깅, 그리고 일방적으로 철회된 트랜잭션의 상태를 유지하는 기법을 근간으로 한다. 또한 제안된 방법은 유사 구조의 타 웹 트랜잭션 처리 시스템에서도 적용 가능하다.

Abstract Electronic business processing has drastically grown since the advent of the World Wide Web. As transaction processing is at heart of business processing, many companies have been interested in transaction processing on the Web. However, little attention has been paid to supporting consistent transaction processing on the Web, while most of the work has been focused on performance enhancement, processing multiple HTTP requests in a single context, and scalability.

This paper identifies three problems in consistent transaction processing on the Web, and proposes corresponding solutions for each problem. The results are implemented in the Web transaction processing system WebTP. These solutions are based on cookie logging, Web page logging and keeping information for unilaterally aborted transactions. The solutions proposed in this paper also can be applied to other Web transaction processing system of similar architecture.

1. 연구 동기 및 배경

현재 웹은 잠재적으로 무제한적인 사용자에게 데이타의 접근을 제공하여 많은 기업들이 웹을 바탕으로 한 사업 수행(business processing)에 많은 관심을 갖게 되었다. 트랜잭션 처리는 사업 수행에 있어 핵심이 되기 때문에, 많은 기업들과 개발자들은 웹 환경에서 수행되는 트랜잭션 처리 시스템 개발에 많은 노력을 기울여왔다. 웹 트랜잭션 처리 시스템은 웹 환경에서 데이타베이스 등과 연동하여, 여러 중요한 업무를 처리하는 트랜잭션이 효과적으로 처리될 수 있는 환경을 제공하는 시스템이다.

그러나 웹은 트랜잭션 처리 목적으로 개발되지 않았기 때문에, 웹 환경에서의 트랜잭션 처리는 많은 문제점을 유발시킨다. 현재 TIP[8]와 같이 인터넷에서 트랜잭션을 효과적으로 처리할 수 있는 환경을 제공하기 위한 노력은 되고 있지만 아직 실용화 단계에는 이르지 못했다. 또 다른 노력으로 기존의 웹 환경의 변화를 최소화하면서 웹 트랜잭션을 처리하는 연구가 있다. 지금까지 대부분의 웹 트랜잭션 처리 시스템은 이러한 접근을 바

· 이 연구는 '96년도 통상산업부 공업기반과제(과제번호:961-28-3)의 위탁과제와 정보통신부 산학연 과제(과제명 미래로 D/B 통합 소프트웨어 시스템을 위한 SRP 확장) 지원에 의한 결과임.
† 비 회 원 : 서울대학교 전산과학과
kwlee@oopsla.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
hjk@oopsla snu ac kr
논문접수 : 1998년 11월 3일
심사완료 : 1999년 2월 1일

탕으로 개발된 시스템들이다. 그러나 이러한 접근 방법은 주로 기존의 CGI 방식의 트랜잭션 처리 성능의 한계를 극복하거나, 여러 HTTP 요구(HTTP request)로 이루어진 트랜잭션의 처리 방법 등에 대한 연구가 집중된 반면, 트랜잭션 처리에 부적합한 웹으로 유발되는 시스템의 일관성 상실에 대해서는 많은 관심을 두지 못하였다.

웹 트랜잭션 처리시 웹 브라우저의 독특한 인터페이스 또는 HTTP 의 특징을 충분히 고려하지 않으면, 웹 트랜잭션을 처리하는데 있어 다음과 같은 이유로 시스템의 일관성이 상실될 수 있다. 첫째, 웹 브라우저에서 유지되는 쿠키가 웹 트랜잭션 수행으로 변경된 상태에서 트랜잭션 완전 철회 또는 부분 철회되는 경우 브라우저의 쿠키 값의 변화가 복원되지 않아 일관성이 상실되는 경우를 들 수 있다. 둘째, 웹 브라우저와 웹 서버 사이의 메시지 손실 또는 메시지 전달 지연으로 시간 제한이 발생하여 브라우저와 서버 사이의 상태가 비정상적으로 전이되는 경우를 들 수 있다. 마지막으로 웹 브라우저에서의 이력(history) 정보를 이용한 탐색으로 이미 처리된 HTTP 요구를 다시 처리하여 일관성이 상실되는 경우를 생각할 수 있다. 일관성이 상실되는 구체적인 예는 3절에서 언급하기로 한다.

이러한 문제점들은 데이타베이스 트랜잭션 수행 그 자체보다는 웹 환경이 트랜잭션 처리에 적합하지 않아 발생되는 것들이다. 본 논문에서는 이러한 일관성 상실 문제를 해결하기 위해 웹 트랜잭션 처리 시스템인 WebTP에서 제공하는 해결 방법을 설명한다. WebTP에서는 쿠키 로깅, 웹 페이지 로깅, 그리고 일방적으로 철회된 웹 트랜잭션의 식별자를 기록하는 방법을 도입하여 위의 문제점을 해결한다. 본 논문에서 제안된 방법은 WebTP를 기반으로 한 방법이지만, 유사 구조의 대부분의 다른 웹 트랜잭션 시스템에서도 적용 가능하다.

이후의 논문 구성은 다음과 같다. 먼저 2절에서는 본 논문의 플랫폼에 해당하는 WebTP에 대해 간략히 설명하고, 3절에서는 언급한 일관성 상실 문제를 해결하기 위해 WebTP에서 제공하는 방법을 설명한다. 4절에서는 기존의 웹 트랜잭션 처리 시스템들을 개발된 접근 방법에 따라 나열하고, 각 접근 방법의 장단점을 설명한다. 또한 이러한 기존의 시스템들과 WebTP을 비교한다. 마지막으로 5절에서는 본 논문의 결과를 정리한다

2. WebTP의 개요

2.1 페이지 함수와 워크

WebTP에서의 트랜잭션은 하나 이상의 페이지 함수의 수행으로 구성된다. 페이지 함수(page function)는 웹 브라우저의 HTTP 요청으로 호출되어 하나의 웹 페이지를 동적으로 생성하는 함수로, C, C++, Tcl 등 다양한 프로그래밍 언어로 구현될 수 있다. 일반적으로 페이지 함수는 데이타베이스를 접근하여 브라우저의 요구에 맞는 웹 페이지를 생성한다. 그림 1은 RDBMS인 SRP[3]의 Tcl[16] 언어 바인딩인 SRPTcl로 작성된 페이지 함수 rsvflight를 보여준다. 이 페이지 함수는 고객 이름(custname)과 원하는 좌석 수(nseats)를 입력으로 받고, Tcl 전역 변수 wtglobal에 기록된 대상 비행기편(fltno)과 날짜(date) 정보를 이용하여 해당 비행기의 좌석을 예약하는 함수이다. rsvflight 함수는 예약 번호(rsvno)를 Tcl 전역 변수 wtcookie를 통해 쿠키에 등록하여, 해당 웹 브라우저가 추후에 사용될 수 있도록 한다.

```
proc rsvflight { custname nseats } {
    global wtglobal wtcookie
    webtp_puts "Content-type: text/html\n\n<html> <body>"
    set outrow [srp_execimd "select avseats from flight \
                                where flightno = '$wtglobal(fltno)',"]
    set maxseats [lindex $outrow 0]
    # check seats availability
    if { $nseats > $maxseats } {
        webtp_puts "Reservation failed: insufficient seats\n"
        webtp_puts "</body></html>"
        return
    }
    # decrease the number of available seats
    srp_execimd "update flight set avseats = avseats - $nseats \
                    where flightno = '$wtglobal(fltno)' \
                    and date = $wtglobal(date),"
    # insert this booking information into 'booking' table, and
    # generate a new reservation number
    srp_execimd "insert into booking values      \
                    ( , '$wtglobal(fltno)', '$custname',      \
                      $wtglobal(date), $nseats ),"
    # register the generated reservation number(rsvno) into cookie
    srp_execimd row "select rsvno from booking \
                            where recid = $srp_sqlca(rowid);"
    set $wtcookie(rsvno) [lindex $row 1]
    webtp_puts "Seats are successfully reserved.<br>\n</body> </html>"
}
```

그림 1 SRPTcl로 작성된 페이지 함수의 예

웹 브라우저는 "<WebTP-prefix>/<페이지-함수-이름>/<HRSN>/?<인자-리스트>}" 형태의 URL을 사용하여 서버에게 페이지 함수 수행을 요청한다. 예를 들어,

http://www.snu.ac.kr/webtp/rsvflight/4/

?*custname*=홍길동&*nseats*=2

는 페이지 함수 rsvflight를 수행시키는 URL으로, 인자 custname과 nseats의 값으로 각각 "홍길동"과 "2"를 전달하는 예이다. 위 URL에서 페이지 함수 이름과 인

자 리스트 사이에 있는 번호 "4"는 HTTP 요청 순차번호이다. HTTP 요청 순차번호에 대한 자세한 내용은 3.2절에서 다루도록 한다.

대부분의 경우 웹 트랜잭션은 하나 이상의 페이지 함수의 수행을 필요로 한다. WebTP에서는 이렇게 여러 페이지 함수의 수행으로 구성된 웹 트랜잭션을 처리하기 위해 워크(work)를 제공한다. 워크는 하나 이상의 페이지 함수의 순차적인 수행으로 이루어진 논리적 작업의 단위로, 응용(application)의 단위가 되며 트랜잭션 수행의 단위가 된다. 그림 2는 비행기 좌석 예약 응용을 위한 워크 reserveflight의 스크립트를 보여준다. "execute pagefunc" 구문은 워크를 구성하는 페이지 함수를 기술하고, 페이지 함수 수행에 필요한 인자를 기술한다. 주의할 점은 워크 스크립트에 기술된 페이지 함수의 인자 리스트와 실제 해당 페이지 함수의 인자 리스트는 동일하여야 한다. 페이지 함수들은 워크에 기술된 순서에 따라 수행된다. 워크 reserveflight는 페이지 함수 slctdate, slctflight, fillorder, 그리고 rsvflight의 순서로 수행되는 것을 알 수 있다. 웹 브라우저의 요청으로 워크가 시작되면 워크 식별자가 부여되고 이 식별자는 wt_workid라는 이름을 통해 쿠키에 기록된다. 페이지 함수 rsvflight 이외의 나머지 페이지 함수들은 부록에 기술되어 있다.

```
create work reserveflight {
    use global fltno, date;
    execute pagefunc slctdate(),
    execute pagefunc slctflight(from, to, date),
    execute pagefunc fillorder(avseats),
    execute pagefunc rsvflight(custname, nseats),
}
```

그림 2 비행기 좌석 예약을 위한 워크의 스크립트

자신을 구성하고 있는 모든 페이지 함수를 성공적으로 수행한 워크는 브라우저의 요구에 의해 완료(commit)될 수 있다. 워크가 완료되면 워크 수행으로 변경된 모든 데이타베이스 갱신은 완료되고, 워크의 수행은 종료하게 된다. 워크는 수행 중 시스템 내부적인 이유 또는 브라우저의 요구에 의해 워크 수행 중 임의의 시점에서 실패(abort)될 수 있다. 워크가 실패되면 워크의 수행으로 갱신된 데이타베이스는 모두 무효화되고 워크가 종료되게 된다.

한 워크에 속한 페이지 함수들의 결합으로 하나의 응용을 생성하므로, 워크에 속한 페이지 함수들은 서로 밀접한 관계를 갖고 있어 그들 사이의 빈번한 정보 공유가 발생하게 된다. WebTP에서는 워크 변수(work variables)를 도입하여 같은 워크에 속한 페이지 함수들 사이의 정보의 공유가 가능하도록 하여 복잡한 웹 응용을 쉽게 작성할 수 있도록 한다. 워크 변수는 해당 워크 스크립트의 "use global" 구문을 통해 정의된다. 그림 2의 워크는 fltno와 date를 워크 변수로 정의하고 있음을 알 수 있다. Tcl 언어로 구현된 페이지 함수의 경우, Tcl 배열 wtglobal을 통해 정의된 워크 변수를 접근할 수 있다(그림 1 참조).

2.2 WebTP 의 기본 구조

WebTP는 그림 3과 같이, Stub, FlowMgr 그리고 Agent로 구성되어 있다. Stub은 웹 브라우저의 요구에 의해 수행되는 CGI 실행 파일로, 브라우저의 HTTP 요청을 FlowMgr에게 전달하는 작업과, HTTP 요청의 처리 결과로 생성되는 웹 페이지를 FlowMgr로부터 받아 브라우저에게 전달하는 작업을 맡는다.

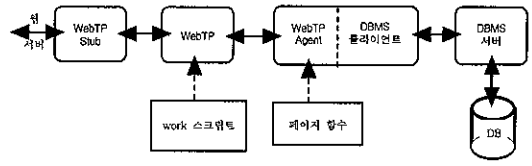

그림 3 WebTP의 실행 구조

Agent는 웹 브라우저가 전달한 HTTP 요청에 해당하는 페이지 함수를 실행시키는 모듈로, 페이지 함수에게 필요한 인자를 전달하고 워크 변수를 접근할 수 있도록 하는 초기화 작업을 맡는다. Agent는 SRP 클라이언트 라이브러리와 바인딩되어 페이지 함수 수행중 SRP에서 제공하는 API를 통해 데이타베이스를 접근할 수 있도록 한다. WebTP는 미리 지정된 개수의 Agent를 생성하여 가용 풀(available pool)을 통해 관리한다. WebTP는 워크가 수행되면 가용 풀에서 Agent를 꺼내어 워크에 할당하고, 워크가 종료되면 할당된 Agent를 다시 가용 풀에 삽입하여 재사용될 수 있도록 한다. 워크에 바인딩된 Agent는 워크의 수행이 끝날 때까지 워크의 페이지 함수를 실행하게 되며, 다른 워크를 위해 사용될 수 없게 된다. Agent는 실행시킬 페이지 함수가 메모리에 적재(load)되지 않은 경우에는 동적으로 메모리에 적재하여 수행시킨다.

FlowMgr는 WebTP의 가장 핵심적인 작업을 수행하는 모듈로, WebTP에서의 워크 수행 제어, 워크 상태 관리, 워크 변수 관리 등의 역할을 맡는다. FlowMgr는 워크가 시작되면 생성되어 배타적으로 바인딩되고, 워크

의 수행이 종료되면 함께 종료되게 된다. 기타 다른 WebTP에 관한 내용은 [2]를 참조하길 바란다.

3. 웹 트랜잭션을 위한 일관성 유지 기법

앞서 1절에서는 웹 트랜잭션 처리시 시스템의 일관성을 저해하는 3가지 경우를 언급하였다. 이 절에서는 일관성 상실 문제를 페이지 함수 rsvflight (그림 1)를 예로 들어 설명하고, 각 문제의 해결 방법을 제시한다.

3.1 웹 브라우저 상태 철회

최근의 웹 브라우저들은 상태 비보존 프로토콜인 HTTP의 한계를 극복하기 위해 쿠키(cookie)를 제공한다. 쿠키[11]는 웹 브라우저에 의해 유지되는 '<쿠키 이름, 쿠키 값>' 쌍들의 리스트로, 서버에서 브라우저로 전송하는 웹 페이지에 포함되어 브라우저에게 전달된다. 웹 브라우저로 전달된 쿠키는 계속 유지되어 다음 번 HTTP 요청시에 웹 서버로 전달되어 서버에서 사용된다. 쿠키의 도입으로 웹 브라우저는 어느 정도의 상태를 유지할 수 있어 비보존 프로토콜인 HTTP의 한계를 부분적으로 해결하게 된다. 그러나 쿠키의 도입은 웹 트랜잭션이 철회되는 경우 트랜잭션 수행으로 변경된 쿠키도 무효화시킬 필요성을 유발시킨다. 예를 들면 워크 reserveflight (그림 2)가 rsvflight 함수까지 수행하고 완료되기 이전의 상태라고 가정하자. 그러면 rsvflight 함수의 수행으로 생성된 예약 번호 rsvno가 쿠키로 등록되어 웹 브라우저에 저장되게 된다. 이 때 고객의 요구에 의해 수행중이던 워크를 철회시키면, 그때까지의 변경된 모든 데이타베이스 갱신이 DBMS에 의해 철회되어 테이블 flight와 booking의 갱신은 모두 복구되지만, 웹 브라우저에 저장된 쿠키 rsvno가 제거되지 않아 브라우저와 서버사이의 일관성이 상실된다. 그러나 웹 트랜잭션 철회에 따른 쿠키 값 복원의 필요성은 [7]을 제외한 대부분의 웹 트랜잭션 처리 시스템에서는 언급되지 않고 있다.

WebTP에서는 워크 수행 중에 변경되는 쿠키를 로깅하여 웹 트랜잭션이 부분 철회 또는 완전 철회하는 경우 로깅된 쿠키 값을 이용하여 웹 브라우저에서 유지되는 쿠키 값을 복원하는 방법을 사용한다. 웹 브라우저에 저장된 쿠키를 복구시키기 위해 웹 서버에서 트랜잭션 철회 후 브라우저로 전달되는 웹 페이지의 Set-Cookie 헤더에 로깅된 이전 쿠키 값을 기록하는 방법을 사용한다. 응용 개발자는 워크 스크립트에 'use cookie 문'을 추가하여, 워크 수행으로 변경되는 쿠키를 지정할 수 있도록 한다. FlowMgr는 use cookie 문에 등록된 쿠키의 변경만을 로깅하여, 워크 수행과 무관하게 변경되는 쿠키의 로깅을 막을 수 있다. 그림 4는 그림 2의 확장된 워크 스크립트를 보여준다. 그림 4는 워크 reserveflight에서 사용되는 쿠키 rsvno를 use cookie 문을 통해 등록하여 워크 수행중에 rsvno의 변경을 로깅하게 한다.

```
create work reserveflight {
        use global fltno, date;
        use cookie rsvno;
        execute pagefunc slctdate(),
        execute pagefunc slctflight(from, to, date);
        execute pagefunc fillorder(avseats);
        execute pagefunc rsvflight(custname, nseats);
}
```

그림 4 확장된 비행기 좌석 예약을 위한 워크의 스크립트

```
logrecord ← {}
filter ← 워크 스크립트의 "use cookie" 문에 기술된 쿠키 이름들의 집합
for HTTP 요구 메시지에 기록된 각 쿠키 c에 대해 do
        if c.name ∈ filter then
                logrecord ← logrecord ∪ {<c.name, c.value, Y>}
                filter ← filter - {c.name}
        end if
end for
for filter에 포함된 각 쿠키 이름 name에 대해 do
        logrecord ← logrecord ∪ {<name, undef, N>}
end for
```

그림 5 쿠키 로그 레코드 생성

그림 5는 브라우저로부터의 HTTP 요구 메시지에 포함된 쿠키가 FlowMgr에 의해 로깅되는 작업을 의사코드로 보여준다. FlowMgr는 워크가 처음 시작될 때(즉, 브라우저로부터의 워크에 속한 첫 번째 HTTP 요청이올 때)와 브라우저로부터의 저장점 설치[1] HTTP 요청이 도착할 때 전달되는 HTTP 요청에 포함된 쿠키를 로깅한다. 쿠키는 워크 수행 중에 생성/삭제 될 수도 있어, FlowMgr는 각 로깅될 쿠키에 대해 "<쿠키 이름, 해당 값, 존재 여부(Y/N)>" 쌍을 로깅하여 페이지 함수 수행으로 생성/삭제되는 쿠키의 변화도 추적할 수 있도록 한다. 쿠키 로깅 당시 use cookie 문에 언급된 쿠키 이름이 HTTP 요청에 포함되어 있지 않은 경우에는 "<쿠키이름, **undef**, N>" 쌍이 로그에 기록된다.

쿠키 복원은 웹 브라우저가 수행 중인 워크에 대한 전체 철회 또는 부분 철회를 요구하는 경우에 이루어진

1) WebTP에서의 저장점 설치 작업의 자세한 내용은 [2]를 참고하기 바란다.

다. 웹 브라우저가 워크의 전체 철회를 요구하는 경우에는 워크가 시작될 때 로깅된 쿠키 정보가 사용되고, 부분 철회를 요구하는 경우에는 부분 철회 요청에 첨부된 대상 저장점 설치시에 로깅된 정보가 사용된다. 그림 6은 선택된 쿠키 로그 레코드를 이용하여 철회 요청에 대한 결과 웹 페이지에 복원된 쿠키를 삽입하기 위한 Set-Cookie 헤더를 생성하는 과정을 보여준다.

```
logrecord ← 선택된 쿠키 로그 레코드
cookie ← 브라우저로부터 전달된 HTTP 요청에 포함된 쿠키들의 집합
for logrecord에 포함된 각 순서쌍 <name, value, tag>에 대해 do
    if (tag = Y) ∧ (name ∈ cookie) then
        쿠키 이름 name을 무효화하는 Set-Cookie 헤더에 삽입
    else
        <name, value> 쌍을 Set-Cookie 헤더에 삽입
    end if
end for
```

그림 6 쿠키 로그 레코드 이용한 쿠키 복원

워크는 웹 브라우저의 명시적인 요청에 의해서 철회되는 경우도 있지만, 다른 이유(예를 들어 교착상태 발생 등)로 서버에서 수행되는 워크가 일방적으로 철회되는 경우도 있다. 이 경우 현재 HTTP으로는 변경된 웹 브라우저의 쿠키를 복원시킬 방법이 없다. WebTP에서는 이를 해결하기 위해 워크가 웹 서버에서 일방적으로 철회되는 경우, 복원되어야 할 쿠키 값을 일정 기간 동안 해당 워크 식별자와 함께 웹 서버에서 유지하여 추후 해당 웹 브라우저로부터 HTTP 요청이 올 때, 결과 웹 페이지의 Set-Cookie 헤더에 복원된 쿠키를 삽입할 수 있도록 한다. 일방적으로 철회되는 워크에 대한 정보를 저장/관리하는 방법은 3.3절에서 다루기로 한다.

3.2 웹 페이지 로깅

웹 브라우저는 이력을 이용한 후위 탐색 기능을 통해 이미 접근한 웹 페이지를 다시 접근할 수 있다. 이 때 기존에 접근한 웹 페이지가 브라우저의 캐쉬에 저장된 경우에는 별 문제가 없지만, 캐쉬에 없거나 사용자가 명시적으로 "Reload" 등의 명령을 통해 이미 수행된 페이지 함수를 다시 수행시키는 경우에는 시스템의 일관성이 손상될 수 있다[1, 2]. 예를 들어 비행기 예약 워크 reserveflight가 rsvflight까지 수행된 상태에서 사용자의 고의 또는 실수로 인해 웹 브라우저의 'Reload' 버튼을 누른 경우를 가정하자. 이 때는 rsvflight 함수의 수행을 요구하는 URL이 웹 서버로 다시 전달되어 rsvflight 함수가 다시 수행되게 된다. 이 경우 데이타베이스 테이블 flight와 booking이 추가로 갱신되어 같은 예약 정보가 두 번 기록되는 일관성 손상을 초래한다.

기존 WebTP에서는 이러한 문제의 발생을 막기 위해 이미 수행된 페이지 함수의 재 수행 요청을 오류로 처리하여 워크의 상태를 보호하는 방법을 사용하였다[2]. 그러나 이 방법은 웹 브라우저의 이력 정보를 이용한 후위 탐색 기능을 사용할 수 없고, 한 워크 내에 같은 이름의 페이지 함수를 두 번 이상 수행시킬 수 없는 단점을 갖는다. 그러나 경우에 따라 웹 브라우저에서 이미 요청한 페이지 함수 수행을 다시 요청할 필요가 발생되기도 한다. 페이지 함수 수행으로 생성된 결과 웹 페이지가 네트워크 문제 등으로 브라우저로 전달되지 못하는 경우가 그 예이다. 웹 브라우저는 HTTP 요청이 서버에게 전달되는 도중 발생된 문제인지, 혹은 해당 요청이 이미 서버에서 처리되었으나 그 결과가 전달되는 과정에서 발생된 문제인지를 알 수 없기 때문에 HTTP 요청의 처리 여부를 판단할 수 없게 된다. 만일 이 때 웹 트랜잭션 처리 시스템에 이미 처리된 HTTP 요청에 대한 재 요청을 시스템의 일관성을 깨지 않으면서 다시 처리할 수 있다면, 브라우저에서는 이러한 상황에서 요청을 다시 서버에게 전달하는 방법으로 해결할 수 있게 된다. 그렇지 않으면, 위 상황에 대한 유일한 해결책은 수행중이던 워크를 철회시키고 다시 수행시키는 방법 밖에는 없게 된다. 그러나 기존의 트랜잭션 처리 시스템이 동작하는 환경과는 달리 인터넷으로 연결된 웹 환경에서는 브라우저의 요청 메시지 혹은 그 결과 메시지가 자주 제때에 전달되지 않아 시간 제한이 발생하는 경우가 흔하기 때문에, 두 번째 방법은 잦은 워크 철회가 유발될 수 있다. 그러므로 웹 트랜잭션 처리 시스템은 시스템의 일관된 상태를 해치지 않으면서 이미 처리된 페이지 함수 호출을 다시 처리할 수 있는 방법이 제공되어야 한다.

개선된 WebTP에서는 이를 해결하기 위해 웹 페이지 로깅(Web Page Logging)을 도입하였다. 웹 페이지 로깅은 페이지 함수 수행으로 생성된 웹 페이지를 웹 브라우저로 전송하기 전에 FlowMgr에 의해 로깅되는 것이다. FlowMgr는 웹 브라우저로부터 이미 처리된 페이지 함수 수행 요청이 오는 경우, 해당 페이지 함수를 다시 수행시키지 않고 기존에 생성된 웹 페이지를 로그에서 읽어 웹 브라우저에게 전송하여 중복된 페이지 함수 수행으로 시스템의 일관성이 상실되는 현상을 방지한다. 로깅된 웹 페이지는 해당 워크가 수행되는 동안만 필요하기 때문에 웹 페이지의 로깅으로 유발되는 공간 소모 부하는 그리 크지 않게 된다. 이 때 주의할 점은 웹 페이지 로깅시 대상 페이지의 Set-Cookie 헤더 부분을 제

외시키고 로깅하는 것이다. 이것은 이전에 처리된 HTTP 요청의 결과 웹 페이지를 브라우저로 다시 전송하는 웹 페이지에 Set-Cookie 헤더를 포함하게 되면, 브라우저에서 유지되고 있는 최근의 쿠키 값이 이전의 상태로 되돌아가는 것을 막기 위함이다.

정확한 웹 페이지 로깅을 위해서는 브라우저로부터의 페이지 함수 호출 요청이 이전에 처리된 것을 다시 요청한 것인지 혹은 새로운 요청인지를 구분할 수 있어야 한다. 기존의 WebTP에서는 브라우저로부터 전달된 페이지 함수 호출 HTTP 요청에 단지 대상 페이지 함수의 이름과 호출에 필요한 인자들만 전달되기 때문에, 단지 페이지 함수의 이름만을 이용하여 요청이 기존에 처리된 것인지에 대한 판정을 하였다. 즉, 요청한 페이지 함수 이름이 이미 처리된 페이지 함수 이름과 같다면 무조건 해당 요청을 이미 처리된 HTTP 요청이라 간주하였다. 이 때문에 기존의 WebTP에서는 한 워크는 같은 이름의 페이지 함수를 두 번 이상 호출할 수 없었다. 개선된 WebTP에서는 이러한 단점을 극복하기 위해 웹 브라우저에서 전송하는 각 HTTP 요청 URL에 "*HTTP 요청 순차번호(HTTP Request Sequence Number: HRSN)*"를 도입한다. HRSN은 워크 수행중 브라우저에서 서버로의 페이지 함수 호출 요청에 부여되는 순차 증가되는 번호로, 이를 통해 페이지 함수의 기처리 여부를 확인한다. 즉, WebTP는 이미 처리된 페이지 함수의 HRSN을 기억하여, 브라우저로부터 전달된 URL에 포함된 HRSN과 비교하여 URL에 포함된 HRSN이 기억된 HRSN보다 작거나 같은 경우에는 이미 처리된 페이지 함수의 재처리 요청으로 간주한다. 예를 들어, 페이지 함수 rsvflight를 호출하는 URL의 경우에는 HRSN이 4번임을 알 수 있다. 만일 사용자가 rsvflight의 수행을 마친 상태에서 동일한 HTTP 요청(즉, 동일한 URL)을 전달하는 경우, URL에 포함된 HRSN(4번)을 통해 이미 처리된 것의 재요청이라는 사실을 알 수 있게 된다.

3.3 시간 제한 처리

웹에서 사용하는 비연결 기반(connectionless) HTTP는 워크 수행중의 웹 브라우저 혹은 웹 서버의 고장으로 일방적으로 종료되는 경우 이를 상대방이 바로 확인할 수 없는 단점을 갖는다. 특히, 브라우저 또는 서버가 상대방으로부터의 메시지를 기다릴 때 상대방이 일방적으로 종료된 경우에는 이를 알 수 없기 때문에 계속 상대방의 메시지를 기다리는 문제가 발생한다. 이러한 문제를 해결하기 위해, WebTP를 포함한 많은 웹 트랜잭션 처리 시스템에서는 시간 제한(timeout) 방법을 사용한다. 시간 제한 방법이란 상대방의 메시지를 기다릴 때 지정된 시간 내에 상대방의 메시지가 없는 경우 상대방에 고장이 발생하였다고 가정하는 방법이다. WebTP에서는 서버에서 시간 제한이 발생하면 수행중이던 워크를 철회시키는 방법을 사용하고, 웹 브라우저에서 시간 제한이 발생되면 웹 브라우저의 의지에 따라 해당 워크를 철회시키는 방법 또는 동일 요청을 다시 서버로 전달하는 방법을 사용한다(3.2절 참조). WebTP는, 시간 제한 방법을 사용하는 경우, 특히 고려해야 하는 다음 두 가지 상황에 대한 처리방법을 제공한다.

첫째로, 워크가 웹 브라우저로부터 완료 요청을 받아 완료되고, 결과 웹 페이지를 전송하고, 종료되었으나 전송한 웹 페이지가 브라우저로 전달되기 전에 브라우저에서 시간 제한이 발생하는 경우이다. 이 때는 앞서 클라이언트에서 시간 제한이 발생되는 경우 취할 수 있는 두 가지 방법 즉, 워크를 철회시키는 방법과 동일 요청을 다시 전달하는 방법 모두 사용하기 어렵게 된다. 서버에서는 이미 워크가 완료되었기 때문에 워크를 철회시킬 수 없고, 워크가 이미 종료되었기 때문에 웹 브라우저에서 HTTP 요청을 다시 보내더라도 이를 처리할 FlowMgr가 없어 문제가 발생된다. WebTP에서는 이 문제를 해결하기 위해 Stub으로 하여금 브라우저로부터 이미 종료된 워크에 대한 완료 요구 메시지가 도착하는 경우, 해당 워크는 완료되었다고 간주하고 그에 적절한 결과 웹 페이지를 생성하여 브라우저로 반환토록 한다. 이 때 웹 브라우저와 웹 서버와의 가상적인 접속을 끊기 위해 워크 식별자(wt_workid)를 쿠키에서 제거하는 작업도 Stub에 의해 수행된다.

둘째는 웹 서버에서 시간 제한이 발생하거나 기타 이유로 일방적으로 워크가 철회되는 경우이다. 브라우저는 워크의 철회 사실을 모르기 때문에 워크에 대한 HTTP 요청을 계속 전달하게 된다. 이 때는 첫 번째 경우와 같이 이미 종료된 워크에 대한 요청이 서버에 전달되기 때문에 서버는 이 두 가지 경우를 구분할 수 있어야 그에 적절한 작업을 취할 수 있게 된다. WebTP에서는 이를 구분하기 위해서 일방적으로 워크가 철회된 경우에는 철회된 워크 정보를 웹 서버에 일정 기간 동안 유지하도록 한다. 이 때 유지되는 철회된 워크 정보는 워크 식별자(wt_workid)와 워크 철회로 복원될 쿠키 값이다. 워크 식별자를 통해 HTTP 요청에 해당하는 워크가 이미 종료된 경우 Stub은 워크 식별자를 이용하여 기록된 워크 정보를 검색하여 워크의 기록이 남아 있을 때는 서버 쪽에서 해당 워크가 일방적으로 철회된 것으로 간주하고, 기록이 없는 경우에는 완료된 것으로 간주

한다. Stub은 철회된 것으로 간주된 HTTP 요청에 대해서는 해당 워크가 철회된 사실을 알리는 웹 페이지를 전송하고, 이 때 Set-Cookie 헤더에 복원될 쿠키 값을 삽입하여 브라우저의 상태를 철회시킨다. 서버에서 일방적으로 철회되는 워크는 그리 흔치 않기 때문에, 이들의 정보를 유지하는 것은 많은 부하를 유발하지 않게 된다. 그림 7은 여기서 설명한 시간 제한 상황을 처리하는 Stub의 작업을 의사코드로 보여준다. 그림에서 "R.Cookie.wt_ workid"는 웹 브라우저로 전달받은 HTTP 요청에 포함된 쿠키 중에서 워크 식별자를 의미한다.

```
웹 브라우저로부터 HTTP 요청 (R)을 수신
워크 식별자 R.Cookie.wt_workid에 해당하는 워크 존재 확인
if 존재하면 then
   HTTP 요청 R을 해당 FlowMgr에게 전송
   FlowMgr로부터 결과 웹 페이지 P를 수신
else
   R.Cookie.wt_workid를 이용하여 서버에 저장된 워크 식별자 검색
   if 발견되면 then
      "해당 워크가 일방적으로 철회되었음"을 알리는 웹 페이지 P를 생성
   else
      "해당 워크가 이미 완료되었음"을 알리는 웹 페이지 P를 생성
   end if
   웹 페이지 P의 Set-Cookie 헤더에서 wt_workid 쿠키를 무효화시킴
end if
해당 웹 브라우저에게 생성된 웹 페이지 P를 전송
```

그림 7 Stub의 시간 제한 처리 모듈

서버에 유지되는 정보는 해당 웹 브라우저에게 철회 사실이 전달된 것이 확인된 후에 제거될 수 있다. 그러나 철회 사실은 해당 웹 브라우저로부터의 HTTP 요청이 오는 경우에만 알릴 수 있고, 또한 비 연결 프로토콜에서는 알린 내용이 정확히 전달되는 것을 확인할 수 없기 때문에 유지되는 정보가 제거되는 시점을 정할 수 없어, 기록된 정보는 무한히 유지되어야 한다. 그러나 이는 저장 공간의 부족을 유발하기 때문에, WebTP에서는 유지되는 정보의 최대 보유기간(예를 들어 일주일 또는 한달)을 정하여 이 기간이 경과한 정보는 자동적으로 제거한다. 이 방법은 최대 보유기간이 지난 후에 웹 브라우저로부터 전달된 요구에 대해 부정확한 판정을 내리게 되는 문제점을 갖지만, 일반적으로 이러한 경우는 극히 희박하리라 예상된다.

4. 관련 연구

현재 많은 업계와 학계에서는 웹과 데이타베이스와 연동하여 업무를 처리하는 시스템 개발 또는 기반 기술 개발에 많은 노력을 기울이고 있다. 데이타베이스 연동 내지는 웹 트랜잭션 처리에 대한 연구는 다음과 같이 크게 세 가지 접근 방법으로 나눌 수 있다.

첫째는 WebTP와 같이 기존의 웹 환경을 크게 변경시키지 않으면 웹 트랜잭션을 처리하는 방법이다. 이 방법에서는 데이타베이스를 접근하는 방법으로 다양한 프로그래밍 언어를 지원하기도 하고, 자체적인 HTML 템플릿을 이용하도록 하기도 한다. 이러한 접근 방법을 따르는 시스템으로는 UniWeb 2.0[1], ORACLE사의 Application Server[12], Informix사의 Web Integration Option™[6], 그리고 IBM사의 Net.Data[13] 등을 들 수 있다.

이 방법은 기존의 웹 환경(특히 웹 브라우저)을 그대로 사용하기 때문에, 이미 전세계적으로 퍼져 있는 웹 브라우저(기존의 텍스트를 바탕으로 한 브라우저 포함)를 웹 트랜잭션 처리 시스템의 사용자 인터페이스로 사용할 수 있는 장점을 갖는다. 즉, 웹 브라우저를 사용하는 사용자는 모두 잠재적인 웹 트랜잭션의 사용자가 될 수 있어, 웹 트랜잭션 시스템 개발의 비용이 저렴하게 된다. 그러나 이 방법은 웹 트랜잭션 처리 시스템 개발 시 트랜잭션 처리에 부적절한 웹 환경으로 발생되는 많은 문제를 직접 해결해야 하는 단점을 갖는다. 그러나 기존 대부분의 연구는 대부분의 웹 트랜잭션 처리 성능 향상, 다중 HTTP 요청으로 구성된 응용을 위한 상태 관리 방법, 시스템의 확장성 그리고 사용의 편이성에 집중된 반면, 트랜잭션이 처리되는 웹 환경으로 인해 발생되는 일관성 상실에 대해서는 거의 언급되지 않았다.

둘째는 Java를 이용하여 웹 환경의 단점을 해결하는 접근 방식이 있다. 애플릿(Applet)을 이용하면 웹 브라우저에서 Java 프로그램을 수행시킬 수 있고, HTTP 외에 자체적인 프로토콜을 사용할 수 있어 연결 관리도 용이하여 웹에서 데이타베이스 접근하거나 트랜잭션 수행하는 경우 발생되는 여러 문제를 비교적 손쉽게 해결할 수 있는 장점을 갖고 있다. 이러한 방식을 따르는 시스템으로 Transarc사의 Internet DE-Light Client[15]를 들 수 있다. 그러나 Java를 사용하는 방법은 웹 브라우저의 비교적 많은 자원을 요구하기 때문에 웹 브라우저에 많은 부하가 들어 제한적인 사용자에게만 서비스를 제공[7]하게 되어 웹 환경 사용의 근본적인 목적을 달성할 수 없다. 또한 아직까지는 인터넷상에서의 Java 애플릿 수행은 과도한 네트워크 통신량으로 만족할만한 성능을 보이지 못한다. 특히 업무 처리와 같이 데이타 처리가 심한 응용의 경우에는 심각한 성능저하를 보인다[14].

마지막으로 현재의 웹 환경(특히 웹 브라우저)을 수정

하여 트랜잭션 처리에 적합하게 바꾸는 접근 방법을 들 수 있다. 이러한 노력의 예로 TIP(Transaction Internet Protocol)[8]를 들 수 있다. TIP는 현재 개발되고 트랜잭션 처리에 접합한 프로토콜로 아직까지 완전한 명세가 완성되지 못하였다. TIP를 HTTP 대신 사용하면 보다 완벽하게 인터넷상에서 트랜잭션을 처리할 수 있으리라 생각된다. 그러나 이 방법은 단순히 웹 서버뿐만 아니라 웹 브라우저도 변경해야 하기 때문에 현재 이미 전세계적으로 사용하고 있는 웹 브라우저를 새로운 브라우저로 대치하기를 요구하기 때문에 당분간은 현실적인 방법이라 보기가 어렵다.

트랜잭션 처리에 있어 일관성의 유지는 매우 중요한 분야이고, 일관성 유지에 대해서는 예전부터 심도 있는 연구가 진행되어 왔다[4, 5, 10]. 그러나 웹이라는 독특한 환경 때문에 웹 트랜잭션 처리시 일관성을 유지시키기 위해서는 추가의 작업을 필요로 한다. [1]은 중복된 HTTP 처리로 유발될 수 있는 웹 트랜잭션의 비정상적인 상태로의 전이를 지적하였으나, 이 문제를 해결할 수 있는 구체적인 방법을 제시하지는 못하였다. [7]은 웹 트랜잭션 철회시 쿠키의 미복구로 유발된 일관성 문제점에 대해서 언급하였고, 트랜잭션 게이트웨이와 브라우저 proxy를 통한 해결 방법을 제안하였다. 그러나 이 방법은 java를 사용하는 시스템만 적용 가능하다는 단점과 기타 나머지 일관성 상실 문제에 대한 언급이나 해결책을 제시하지 못하고 있다.

5. 결 론

웹은 잠재적으로 무제한적인 사용자에게 데이타의 접근을 제공할 수 있어 많은 업계와 학계에서 웹 환경에서 다양한 업무를 처리하기 위한 웹 트랜잭션 처리 시스템 개발에 많은 노력을 기울여 왔다. 웹은 트랜잭션 처리 목적으로 개발되지 않았기 때문에, 웹 환경에서 트랜잭션을 처리하는 것은 많은 문제점을 유발한다. 그러나 대부분의 웹 트랜잭션 연구는 주로 기존의 CGI 방식의 트랜잭션 처리 성능의 한계를 극복하거나, 여러 HTTP 요청으로 이루어진 트랜잭션의 처리 방법 또는 시스템의 확장성등에 대해 연구가 집중된 반면, 트랜잭션 처리에 부적합한 웹 환경으로 유발되는 시스템의 일관성 손실에 대해서는 많은 관심을 받지 못했다.

본 논문에서는 웹 환경으로 유발되는 일관성 상실되는 다음의 세 가지 경우에 대해 웹 트랜잭션 처리 시스템인 WebTP에서 제공하는 해결책을 제시하였다. 첫째로 웹 브라우저로부터의 HTTP 요구에 전달되는 쿠키를 로깅하여, 웹 트랜잭션이 완전 철회 또는 부분 철회되는 경우 로깅된 쿠키를 이용하여 웹 트랜잭션 수행 중에 변경된 쿠키를 무효화시켜 웹 브라우저의 일관성을 유지되도록 하였다.

둘째로 웹 트랜잭션 수행 중 생성된 웹 페이지를 로깅하여, 이미 처리된 HTTP 요청을 다시 전달하는 경우 로깅된 웹 페이지를 반환하여 시스템의 일관성을 유지시켰다. 마지막으로 웹 서버 또는 웹 브라우저에서의 시간 제한이 발생하는 경우, 종료된 워크의 기록 유지를 통해 이미 종료된 워크에 대한 웹 브라우저로의 HTTP 요청을 처리하는 방법을 제안하였다. 본 논문에서 제안된 방법은 WebTP외에 대부분의 다른 웹 트랜잭션 시스템에서도 적용 가능하리라 생각된다.

감사의 글

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

부 록

워크 reserveflight의 나머지 페이지 함수들

```
proc slctdate { } {
    webtp_puts "Content-type: text/html\n\n<html><body>"
    webtp_puts "<form method=get action=webtp/slctflight/2>
                <pre>
                Departure: <input name=from>
                Arrival:   <input name=to>
                Date:      <input name=date>
                <input type="submit"> <input type="reset">
                </form>"
    webtp_puts "</body></html>"
}

proc slctflight { from to date } {
    set cusid [srp_declcus
            "select fltno avseats from flight\
             where from=$from and to=$to and date=$date,"]
    srp_opencus $cusid
    webtp_puts "Content-type: text/html\n\n <html><body>\n
            <table border=2 width=100%><tr>"
    # print column name
    set colnameset [srp_describe -colname $cusid]
    foreach colname $colnameset {
        webtp_puts "<th align=center> $colname"
    }
    # print flights information
    while { [catch { set row [srp_fetchcus $cusid] }] == 0 } {
        webtp_puts "</tr><tr>"
        set fltno [lindex $row 1]
        set avseats [lindex $row 2]
        webtp_puts "<td align=left> \
                <a href=webtp/fillorder/3?$fltno+$avseats>$fltno</a>"
        foreach col [lrange $row 2 end] {
            webtp_puts "<td align=left> $col"
```

```
        }
    }
    webtp_puts "</tr></table>"
    srp_closecus $cusid
    webtp_puts "</body></html>"
}

proc fillorder { avseats } {
    webtp_puts "Content-type: text/html\r\n\n<html><body>"
    webtp_puts "<form method=get action=webtp/rsvflight/4>
                <pre>
                Customer Name: <input name=custname>
                Number of Seats: <input name=nseats value
                  =$avseats>
                <input type="submit"> <input type="reset">
                </form>"
    webtp_puts "</body></html>"
}
```

참 고 문 헌

[1] 김평철, "UniWeb 2.0-웹을 이용한 클라이언트-서버 데이타베이스 응용 개발 환경", 데이타베이스 저널, 제3권, 제2호, pp.119-132, 1996

[2] 이강우, 김형주, "웹 트랜잭션 처리 시스템의 구현", 한국정보과학회 논문지 (C), 계재예정, http://oopsla.snu.ac.kr/publication/

[3] 이강우, 안정호, 김형주, "확장용이 클라이언트-서버 RDBMS의 설계 및 구현", 한국정보과학회 SIGDB, 겨울 1994, http://oopsla.snu.ac.kr/publication/

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

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

[6] Informix, "Informix Web Integration Option for Informix Dynamic Server," White Paper, 1998.

[7] M.C. Little, S.K. Shrivastava, S.J. Caughey, and D.B. Ingham, "Constructing Reliable Web Applications using Atomic Actions," Proc. of the 6th Int'l World Wide Web Conference, pp.7-11, April 1997.

[8] J. Lyon, K. Evans, and J. Klein, "Transaction Internet Protocol Version 3.0 (Internet-Draft)," Transaction Internet Protocol Working Group, May 1998.

[9] Susan Malaika, "Resistance is Futile: The Web Will Assimilate Your Database," IEEE Database Engineering bulletin, 제21권, 제2호, pp.4-13, June 1998.

[10] 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, 제17호, 제1권, pp.94-162, March 1992.

[11] Netscape Communication Corporation, "Persistent Client State HTTP Cookies," http://home.netscape.com/newsref/std/cookie_spec.html

[12] Oracle Corporation, "*Oracle Application Server 4.0^{TM},*" White Paper(Draft), February, 1998

[13] C.-S. Peng, S.-K Chen, J.-Y. Chung, A. Roy-Chowdhury, and V. Srinivasan, "Accessing existing business data from the World Wide Web," IBM Systems Journal, 제37권, 제1호, pp.115-132, 1998.

[14] Jurgen Sellentin and Bernhard Mitschang, "Data Intensive Intra- & Internet Applications-Experiences Using Java and CORBA in the World Wide Web," Proc. of the Conf. on Data Engineering, pp.302-311, 1998.

[15] Transarc Corporation, "Transarc Corporation's World Wide Web Strategy Overview," White Paper

[16] Brent B. Welch, editor, "Practical Programming in Tcl and Tk," Prentice Hall PTR, 2nd edition, 1997.

이 강 우
제 26 권 제 5 호(B) 참조

김 형 주
제 26 권 제 1 호(B) 참조