CORBA를 이용한 Web기반 전자상거래 시스템의 설계 및 구현

°전 찬 용*, 이 제 현*, 송 화 선**, 정 영 준*
* 강원대학교 전자계산학과, ** 강원대학교 전기공학과

Design and Implementation of a Web-based Electronic Commerce System using CORBA

°Chan-Yong Jeon*, Je-Hyun Lee*, Hwa-Sun Song**, Young-Jun Chung*
*Dept. of Computer Science, Kangwon National University
**Dept. of Electrical Engineering, Kangwon National University

요 약

웹 전자상거래 이용이 증가함에 따라, 기존의 2단계 방식의 클라이언트-서버 구조를 갖는 시스템은 네트워크 병목현상이나, 시스템 성능저하, 시스템 확장문제 및 서버의 시스템 부하등의 문제가 유발되었다.[1] 이와 같은 문제는 전자상거래와 같은 대규모의 클라이언트를 요구하는 환경에서 가장 큰 취약점으로 지적 되고 있다. 또한 네트워크상의 가상기업, 가상시장 등 각 기업의 전자상거래 구축을 위한 다양한 플랫폼이 개발됨에 따라 이들의 독자적인 플랫폼들간의 지불, 보안등의 상호 호환성을 해결하기 위해 CORBA IIOP(Internet Inter ORB Protocol)를 사용하는 추세에 있다.[2]

이에 본 논문에서는 웹기반 전자상거래를 CORBA를 이용하여 3계층 방식의 데이터베이스 접근방법으로 구현하였다. 이는 웹을 통해 고객이 주문한 상품은 CORBA 에이전트의 서버 구현객체 호출을 통해 데이터베이스에 저장되며, 전자우편을 통하여 관리자에게 통보하여 주는 기능을 가지고 있다.

1. 서 론

분산시스템이 대두되기 전의 시스템은 중앙의 컴퓨터에 접속하여 모든 작업을 수행하는 호스트 중심의 중앙 집중식 방식이였다. 이 방식은 중앙의 시스템에 모든 작업이 집중되어 여러 문제를 초래하였다. 특히 최근에는 웹에서의 전자상거래 시스템이 방대해 짐에따라 시스템의 성능 향상이 크게 요구되고 있으며, 이를 해결하기 위해 이종 컴퓨터들 간에 프로그램을 분산시켜 부하를 줄이고 시스템의 성능 저하 및 네트워크 병목 현상을 해결하고 있는 추세이다. [3][4]

따라서, 기존의 웹 전자상거래 시스템에서는 2단계의 클라이언트-서버 구조[5]를 사용하여 왔다. 그러나 이 모델은 클라이언트-서버간 대규모의 데이터 이동이 불가피하다는 단점이 있다. 또한 클라이언트측 사용자 수에따라 서버의 세션을 설정하기 때문에 사용자가 많아질수록 서버에 걸리는 부하가 증가하는 문제점이 있다. 이를 해결하기 위해 다단계 클라이언트-서버 시스템이 대두 되었다.

이러한 다단계 클라이언트-서버 시스템에서는 분산환경에서 여러 이질적인 시스템을 구성하기 위해 클라이언트와 서버사이의 상호 작용을 위해 미들웨어를 제공하며, 적절한 미들웨어를 사용하여 하나의 시스템처럼 구성한 후 서비스 프로그램을 작성하여 여러 DBMS(DataBase Management System)와 부하를 줄이며 연결할 수 있게 해준다.

이러한 데이터베이스 미들웨어는 복수개의 데이터베이스 서버들을 일관되게 이용하기 위한 환경을 제공해 주며, 서로 다른 공급업자에 의해 제공된 이질적인 데이터베이스 서버들을 호출하는 데에도 사용된다

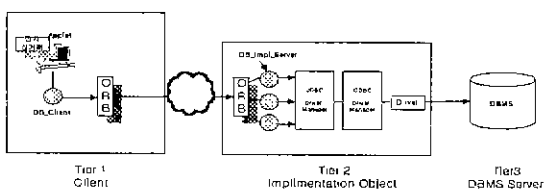
이에 본 논문에서는 전자상거래 시스템에서 중요시 되어지는 데이터베이스 이용에 있어 이질적인 데이터베이스의 호출과 시스템 부하를 줄이기 위하여 클라이언트와 서버사이에 CORBA(Common Object Request Broker Architecture) 미들웨어를 두어 이러한 문제를 해결하고자 한다.

2. Web 전자상거래 시스템의 설계

2.1 3단계의 전자상거래 시스템 구조설계

웹을 통한 전자상거래 시스템은 많은 수의 클라이언트 접속이 이루어 지므로 서버가 받게되는 부하 및 빠른 처리시간의 요구, 다량의 데이터 전송 등의 문제를 해결하여야 한다. 이에 본 논문에서는 CORBA를 이용하여 3단계의 구조(그림 1)로 시스템을 설계하여 보았다.

3계층에 해당하는 DBMS에 접근하기 위해 ODBC(Open DataBase System)에 연결하여 질의를 처리하였으며, 이때 구현객체에서 ODBC 연결에는 Java에서 데이터베이스 연결시 필요한 JDBC(Java DataBase Connectivity)를 사용 하였다[6][7]



(그림 1) 3단계 방식의 전자상거래 구조

2.2 데이터베이스 스키마 설계

클라이언트에서는 ORB(Object Request Broker)통신을 통해 구현객체의 데이터베이스 저장 객체에 의해 DBMS에 저장되어진다. 이때 고객이 주문할 상품에 관한 정보 테이블과 주문한 고객에 대한 신상정보 및 주문한 상품이 저장될 테이블이 필요하며, 주문서 작성을 위해 이러한 테이블에 대해 조인, 관계 설정의 질의 테이블이 필요하다.

1) 상품정보 테이블

상품정보와 상품 식별코드(PDCode) 필드로 구성된다.

2) 고객정보 테이블

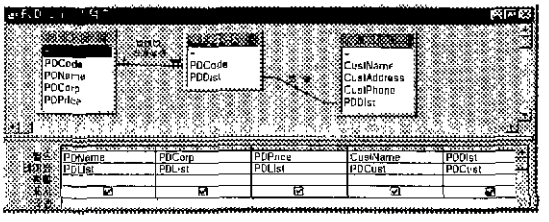
고객정보와 주문한 상품에 대한 key필드(PDDist)를 가진다.

3) 주문 상품 테이블

PDCode 필드는 상품정보 테이블과 일대다 관계를 가지며, 주문한 고객을 알기위한 key필드(PDDist)를 가진다

4) 결과출력 Query Table

신택 질의를 통해 상품과 고객에 대한 주문서 테이블을 작성한다.



(그림 2) 각 테이블간의 선택질의

2.3 CORBA IDL 정의 [8][9][10]

DB(DataBase)로부터 상품 정보를 Client로 가져오고, 주문한 정보에 대한 DB 저장을 위해 계층2 부분의 구현객체에 대해 IDL로 정의하고(그림 3) Java 원시코드로 변환한다.

```
module DataBase {
    interface DB     {
        void dbCon();          //DBMS에 연결
        void dbClose();        //DBMS연결 종료
        //DB로 부터 상품정보를 가져옴
        void getData(out string pCode,
                     out string pName,
                     out string pCorp,
                     out long pPrice);

        //고객이 주문한 상품을 서버의 DB로 저장.
        void setData(in string cpu,
                     in string hdd,
                     in string mainBoard,
                     in string ram,
                     in string graphicCard,
                     in string monitor,
                     in string comCase,
                     in string custName,
                     in string cusAddress,
                     in string cusPhone,
                     in string PDDist);
    };
};
```

(그림 3) DB로의 데이터 Read/Write를 위한 IDL정의

3. 구 현

3.1 구현 환경

CORBA의 핵심이라고 할 수 있는 IDL은 자바 매핑방법을 이용하여 컴파일 하였으며, VisiBroker3.2를 사용하였다. 구현객체와 클라이언트는 자바언어를 이용하여 구현하였으며 JDK 1.1.6을 이용하였다. DBMS는 마이크로소프트사의 액세스를 사용하였으며, 자바 구현객체에서 ODBC에 연결하기 위해 JDBC를 사용하였다. 구현객체와 DBMS는 Pentium-150 윈도우 환경에서 실행시켰다.

3.2 구현객체 프로그램

먼저 데이터베이스에 연결하기 위해 JDBC를 사용하여 ODBC로 연결하였다.

일단 데이터베이스에 연결된후 클라이언트측의 상품정보 출력을 위해 getData 메소드(그림 4)가 사용되어기며 SQL문[11]에 의해 서버 DB의 PDList 테이블로부터 데이터를 가져온다 (그림 4)는 계층2의 구현객체 부분에 대한 원시코드이다.

```
public void getData(
    org.omg.CORBA.StringHolder pCode,
    org.omg.CORBA.StringHolder pName,
    org.omg.CORBA.StringHolder pCorp,
    org.omg.CORBA.IntHolder pPrice
  ) {
    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery("SELECT * FROMPDList");
        rs.next();
        pCode.value = rs.getString("PDCode");
        pName.value = rs.getString("PDName");
        pCorp.value = rs.getString("PDCorp");
        pPrice.value = rs.getInt("PDPrice");
      }catch(Exception e) {System.out.println(e);}
  }
```

(그림 4) 데이터베이스로 부터 상품정보를 가져오기 위한 객체

getData에 의해 웹으로 전송된 상품 중에서 고객이 주문한 상품 데이터는 (그림 5)의 setData 메소드에 의해 서버의 DB로 저장되며, SQL문의 INSERT명령에 의해 PDRequest 테이블에 삽입한다. 또한 고객의 정보는 PDCust 테이블에 저장시킨다.

```
public void setData(
  java.lang.String cpu, java.lang.String hdd, java.lang.String mainBoard,
  java.lang.String cusPhone, java.lang.String PDDist  ) {
  try{
      //데이터베이스에 주문한 정보를 삽입
      PreparedStatement cpustate = con.prepareStatement("INSERT
                                   INTO  PDRequest VALUES(?,?)");
        cpustate.setString(1,cpu);
        cpustate.setString(2,PDDist);
        cpustate.execute();
      PreparedStatement custState = con.prepareStatement("INSERT
                                   INTO PDCust VALUES(?,?,?,?)");
        custState.setString(1,custName);
        custState.setString(2,cusAddress);
        custState.setString(3,cusPhone);
        custState.setString(4,PDDist);
        custState.execute();
        } catch(SQLException e){System.out.println(e);}
}
```

(그림 5) 데이터베이스에 데이터 저장을 위한 객체

4. 실험 및 결과

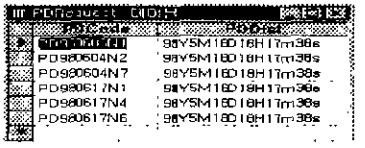
4.1 실험환경

구현한 전자상거래 시스템을 실험해 보기위해 우선 웹서버 데몬을 실행시키고 visiBroker에서 제공해 주는 에이전트를 실행시켰다 다음으로 클라이언트가 요청한 HTML과 애플릿은 http를 기반으로 서버에 접속하여 HTML문서와 자바 바이트 코드를 가져가 실행되어진다. 클라이언트 측에서 작업요청을 하게되면 IIOP를 기반으로 port 15000으로 들어오게 되며, 서버 측에서는 gatekeeper를 실행시켜 port 15000으로 들어오는 데이터를 선택하여 에이전트로 보내주는 역할을 수행한다 에이전트에서는 구현객체와 통신하여 해당객체를 실행시켜 데이터를 저장하거나 결과를 돌려준다.

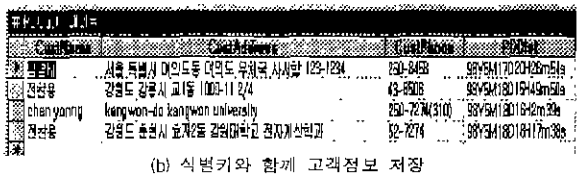
4.2 결 과

1) 전송된 데이터

클라이언트에서 고객이 주문한 상품정보와 고객정보는 각각 PDRequest 테이블(a)과 PDCust 테이블(b)에 저장되어진다. 이 두 테이블은 PDDist라는 key필드(전송한 시간)로서 조인연산을 하게 된다 또한 PDRequest 테이블에 전송된 PDCode로써 상품정보 테이블과 조인을 통하여 주문상품의 상세내역을 보여준다



(a) 주문상품의 코드와 식별키 저장



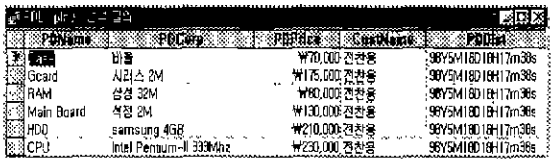
(b) 식별키와 함께 고객정보 저장

(그림 6) 웹을 통해 저장된 데이터

2) 최종적인 결과출력

고객의 상품주문서는 식별코드와 상품 구분 Key에 의한 조건질의를 통해 작성되어지며 최종적인 상품 주문서가 되는 것이다

다음 (그림 7)은 Query에 의해 작성되어진 주문서 테이블이다. 또한 사용자 고객이 상품 주문시 고객 정보는 전자우편으로도 전송되어진다.



(그림 7) 데이터베이스에 기록된 주문서

5. 결 론

본 논문에서는 전자상거래에 있어 가장 기본적인 기능인 데이터베이스 접근에 관한 기능을 구현하였으며, 효율적인 분산처리를 지원하기 위해 객체지향 프로그래밍 구조인 CORBA, Java등을 이용하여 3계층 시스템 구조로 설계하였다. CORBA를 채택함에 따라 작업요청 실행시간이 기존의 2계층 방식에 비해 상당히 짧아졌고, 오브젝트웹의 부하분산기능을 이용하여 피크타임시 서버가 받는 부하를 줄이기 위해 여러대의 시스템으로 분산 운영할수도 있게 되었다. 또한 오브젝트웹의 애플리케이션은 오브젝트 또는 컴포넌트로 모듈화되는데, 이는 CORBA가 정의한 IDL에 의하여 다양한 프로그래밍 언어로 구현해도 상관이 없다는 장점을 가지고 있다. 그러나 ORB통신시 요구되어지는 클라이언트측 통신 객체들을 로딩하는데 많은 시간이 소요된다는 단점도 있다.

전자상거래 시스템을 구축하기 위해서는 본 논문에서 구현한 사항 이외에도 필수 요소시스템들이 구현되어 거래 인프라가 형성되어야 한다. 이에 필요한 시스템을 보면 인증기관, 전자지갑을 장착한 고객시스템, 지불시스템, 상점시스템, 보안시스템 등이 구성되어야 한다.[7]

끝으로, 웹을 통한 전자상거래는 앞으로 초고속 통신망의 구축과 더불어 미래 정보화 사회에서 필연적으로 추진해야할 과제이며, 더 나은 시스템 아키텍쳐의 개발과 정부의 주도적인 협조아래 정보통신 분야의 선두주자로 나서기 위해 부단한 노력이 수반되어야 할 것으로 사료된다

참 고 문 헌

[1] 박재현, "Core CORBA, 영한 출판사", 1998.1.

[2] 김춘길 외, "전자상거래", 정보과학회지 pp 5-18 제16권 제 5호, 1998.5.

[3] H. Onozawa, 분산 오브젝트 지향기술 CORBA, 홍릉과학 출판사, 1997.8.

[4] "네트웍 컴퓨팅 컴포넌트," http://nicstech.co.kr/nicshome/news/nc9711.html

[5] 김평철, "A Taxonomy on Database Gateways for WWW", 1996.3

[6] 김태현, "Java Programing", 크라운 출판사, 1997.3.

[7] Prashant Sridharan, "Advanced JAVA networking," PTR, 1997

[8] Robert Orfali & Dan Harkey, Client/Server "Programing with JAVA and CORBA," 2nd Ed, 1998.

[9] "CORBA Programmer's Guide," Visigenic, February 27, 1998.

[10] "Orbix Web CORBA Programing," Orbix, October 1997.

[11] 황규영,홍의경,음두헌,박영철,김진호 편역, "데이터베이스 시스템", 생능 출판사, 1997 8.