CORBA를 이용한 주식매매 관리 시스템 설계 및 구현

Design and Implementation of a Stock Market Management System using CORBA

황 준* 김 영 신**
Jun Hwang Young-Sin Kim

요 약

전자상거래 시스템은 시스템의 확장, 유지, 보수문제 등의 문제가 개발에 어려움을 주고 있다. 이에 본 논문에서는 JAVA와 CORBA를 이용하여 3-Tier 주식매매 관리 시스템을 제안한다. 본 시스템은 CORBA의 이벤트 서비스를 이용하여 인터렉티브한 환경을 구현하였고, 확장성, 속도, 유연성, 보안성, 유지·보수, 효율성이 뛰어난 시스템을 위하여 CORBA와 JDBC 미들웨어를 이용하여 3-tier 구조 주식매매 관리 시스템을 구현하였다.

Abstract

It is difficult to develop Electronic Commerce System due to expansion, maintenance and repair of the system. In this paper, the author proposes 3-Tier structure Stock Market Management System using JAVA and CORBA. The event service of CORBA supports the interactive environment. For improvement of expansion, performance, security, maintenance, repair, and efficiency, the 3-Tier structure Stock Market Management System is implemented using CORBA and JDBC middle ware in this environment.

1. 서 론

인터넷 환경은 전 세계를 시간과 공간을 초월한 가상 공간을 형성하게 되었다. 이러한 가상 공간에서 전자상거래는 기업의 경영자와 소비자가 누리게 될 초현대적인 편리함과 효율성을 제공해 주게 될 것으로 내다보고 있다.

최근에는 국내에서도 전자 상거래를 개발하고 있거나 개발하고자 하는 기업들이 계속 증가하고 있다. 이러한 확산 과정에서 가장 문제가 되는 것들로는 개발되는 시스템을 기존의 시스템에 연동시키기 위한 표준 방법, WWW에서 제공하는 외부시스템 연동 방법상의 문제, 개발된 시스템의 유지·보수문제가 대두되고 있다.

또한, 전자 상거래에서는 가장 저렴한 비용으로 최신 정보의 공유 및 분배를 실시간으로 가능하게 함으로써 업무 능률이 극대화되어야 하나, 현재 인터넷 환경에서는 클라이언트가 새로운 요구를 제출할 때에만 동기화 되고 있다.

기존의 Web기반 전자 상거래 시스템의 보안문제와 현재 2-Tier 기반에서의 클라이언트, 서버, DB가 서로 종속적이어서 확장하는데도 한계점이 드러나고 있다.

따라서, 본 연구에서는 플랫폼에 독립적인 JAVA와 OMG의 분산객체 표준 기술인 CORBA(Common Object Request Broker Architecture)의 기술 공유를 이용하여 3-Tier 주식매매 관리 시스템을 설계, 구현함으로써 인터렉티브한 환경과 확장성, 속도, 유연성, 보안성, 유지·보수, 효율성이 뛰어난 인프라 구조를 제시하고자 한다.

즉, 자바가 제공하는 유연성(Flexibility)과 이식성(Portability), CORBA가 제공하는 분산객체 인프라를 결합함으로써 인터넷 환경하에서 WWW서버를 포함한 다양한 서버들과 자바의 이식성 있

* 종신회원 : 서울여자대학교 정보통신공학부 교수
hjun@swu.ac.kr
** 준 회 원 : 서울여자대학교 대학원 컴퓨터학과
amaryllis@empal.com

는 코드를 기반으로 한 클라이언트의 구축을 지향하여 Object Web 환경하의 주식매매관리 시스템을 설계, 구현하고자 한다.

2. 관련 연구

2.1 CORBA(Common Object Request Broker Architecture)

CORBA(Common Object Request Broker Architecture)는 OMG에서 정의한 분산 객체 컴퓨팅 환경의 표준으로서 이질적인 분산환경에서 어플리케이션을 구현하는데 필요한 프레임워크, 객체서비스, 분야별 객체 클래스 라이브러리가 정의되어 있다[1].

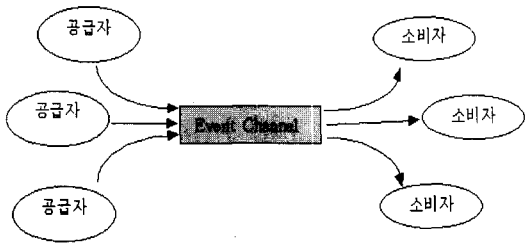
따라서, 이를 이용해 분산 객체 시스템을 개발할 경우 CORBA가 제공하지 않는 모듈들만 구현하면 되기 때문에 구현 작업의 부담이 상당 부분 덜어지게 된다.

CORBA의 구성요소 중에 하나인 ORB는 메소드 호출 요청과 그에 대한 결과를 전달하는 데 핵심적인 역할을 한다. 또한 ORB는 플랫폼에 독립적, 언어에 독립적이다. 따라서, 클라이언트/서버의 완전한 분리를 가능하게 하며, 시스템 통합과 프로그램간의 통합이 쉽게 되므로, 분산시스템의 확장성과 유연성이 크게 향상된다.

2.1.1 이벤트 서비스

OMG에서는 다중 응용 프로그램간의 비동기적인 통신을 지원할 수 있도록 CORBA 이벤트 서비스를 제공하고 있다. 그림 1과 같이 CORBA 이벤트 서비스의 일부인 이벤트 채널로 공급자와 소비자가 연결되어 서로 직접적으로 연결되어 있지 않다. 따라서, 공급자와 소비자는 메시지 전송을 위하여 서로의 정보가 필요 없으며, 새로운 공급자나 소비자 추가가 쉽다.

그림 1과 같이 각각의 공급자들은 메시지를 이벤트 채널에 전송한 후에 자신의 작업을 계속 수



(그림 1) 이벤트 채널을 통한 통신

행하며, 채널에 등록되어 있는 소비자들은 공급자가 전송한 메시지를 받게 된다.

따라서, 이 서비스는 비 동기적인 메시지나 이벤트들을 보내고 받는 결합도가 낮은 객체들을 대상으로 제공한다[2].

CORBA 이벤트 서비스는 이벤트를 전달하는 두 가지 방법을 규정하고 있다.

- Push 모델

공급자가 이벤트를 생성하면 생성된 이벤트를 소비자에게 전달하며, 소비자들은 이벤트가 공급자로부터 전달되기를 기다린다[3]. 능동적인 공급자는 이벤트 채널을 사용하여 이벤트 채널에 등록되어 있는 소비자들에게 데이터를 보낸다.

- Pull 모델

소비자는 능동적으로 공급자가 발생하는 이벤트를 요구하는 방법으로 공급자는 능동적인 소비자로부터 Pull 요구가 전달되기를 기다린다. 능동적인 소비자는 이벤트 채널을 통하여 수동적인 공급자로부터 데이터를 받는다[3].

2.2. ObjectWeb

HTTP/CGI는 동적인 사용자 인터페이스와 지속적인 상태 정보를 요구하는 정보 시스템을 구축할 때 많은 문제가 발생한다. 그러나 Java는 바이트 코드를 생성해내기 때문에 해당 바이트 코드를 해석하는 컴파일러가 있는 곳이면 어디서나

실행될 수 있는 구조를 가지고 있다.

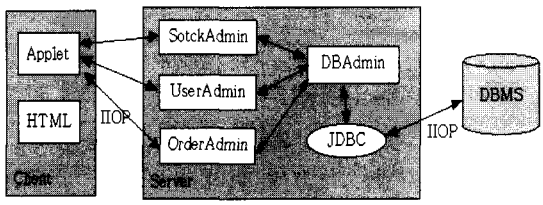
그러나, Java는 클라이언트/서버 시스템으로는 충분한 해결책을 가지고 있지 못하다. 따라서, CORBA의 결합으로 인하여 자바는 분산 객체 인프라로 확장할 수 있다. 또한 현재 Java만으로는 원격 객체의 메소드를 호출할 수 있는 방법은 없으나 CORBA는 자바 애플릿이 원거리에 있는 다른 프로그래밍 언어로 작성된 객체와 통신할 수 있게 해준다.

1995년 후반 OMG는 HTTP와 자바를 CORBA IIOP에 통합한 새로운 ObjectWeb 모델을 제안하였다. 이 모델은 클라이언트, 비즈니스 객체, 서버로 3-Tier을 이루고 있어 CORBA를 사용하여 클라이언트가 직접 서버의 메소드를 호출한다. 그러므로 HTTP/CGI와 비교할 때 클라이언트 서버의 오버헤드가 매우 적다. 또한, 서버들간의 확장 가능한 인프라를 제공할 수 있다.

ObjectWeb에서는 CORBA의 IDL언어를 자바 언어로 변환하고 기존의 WWW환경에서 클라이언트와 서버사이의 통신은 IIOP를 사용한다. 구축 환경은 기존의 WWW환경과 동일하며, 다른 점은 CORBA IIOP를 지원하는 웹 브라우저와 CORBA와 연동 가능한 웹서버에 있다.

3. 설계 및 구현

본 연구에서는 다이나믹한 인터렉티브 환경을 구현하기 위하여 CORBA에서 제공하는 이벤트 서비스에서 PUSH 기술을 이용하였고, 확장이 용이한 시스템 구현을 위하여 각 계층간에 종속성이 없는 N-Tier기반 하에서 시스템을 구축하였다. N-Tier에서 미들웨어로 CORBA이외에도 JDBC를 사용함으로써 시스템 의존적이지 않는 유연한 환경을 만들었다. 따라서, 본 연구에서는 Java와 CORBA의 장점을 접목시킨 ObjectWeb을 활용함으로써 인터렉티브한 환경을 제공하고 확장성, 유연성, 속도, 보안, 유지·보수 측면에서 진보된 주식매매 관리 시스템을 구현하였다.



(그림 2) 주식 마켓 시스템 Object 구성도

3.1 시스템 구조

본 연구에서 설계된 주식 마켓시스템은 3-Tier로 이루어진 분산구조를 이루고 있으며 전체적인 구조는 그림 2와 같다. Tier 1는 Client로서 Java Applet으로 구현하였고, Tier 2는 비즈니스 객체로서 자바로 구현된 Stock Admin, User Admin, Order Admin의 집합인 Server로 구성되며, Tier 3는 DB로써 JDBC 인터페이스와 관계 데이터베이스를 이용하였다.

서로 다른 N개의 클라이언트가 주식생성 및 삭제, 사용자 계정 생성 및 인증, 주식매매 주문 등과 같은 작업을 하고자 할 경우 각기 작업내용에 따라 Server의 StockAdmin, UserAdmin, OrderAdmin 객체중에 한 객체를 호출하며, 각각의 객체들이 DB와 관련된 작업이 필요할 경우 DBAdmin 객체를 호출하여 JDBC를 통해 정보를 생성 및 변경한다.

이렇게 서버 쪽에서 이벤트가 발생했을 경우 Server는 변경된 내용을 Event Channel을 통해서 실시간으로 클라이언트 인터페이스 화면에 push 해준다. 그림 3에서 보는바와 같이 Server, Client, DBMS는 2개의 Event Channel을 이용하여 연결되어 있다. 주식거래가 일어날 경우에 변경된 데이터를 OrderChannel을 통해서 Client에 전달하며, 주식 관련 관리작업을 하였을 경우에는 Stock-Channel을 통해서 변경된 데이터를 Client에 전달한다. 이때 Server는 플랫폼이나 OS에 상관없이 단일 또는 복수개의 서버가 가능함으로 서버에 부하가 많이 걸릴 경우 유연하게 확장 가능하다.

3.2 시스템 구현

(1) StockAdmin Object

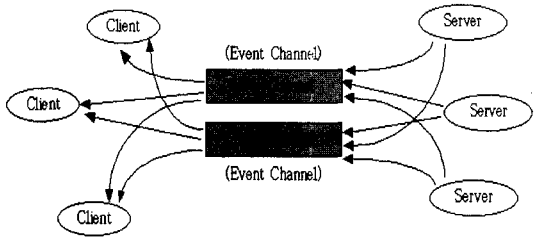
주식생성 및 주식정보의 변경 등과 같이 주식 관련 관리 작업을 하고자 할 경우 Client는 ORB를 통하여 Server의 StockAdmin Object를 호출한다. DB에 접근이 필요한 경우에는 Server의 DBAdmin Object를 호출하며, DB의 정보가 변경이나 추가되는 등의 이벤트 발생 시마다 StockAdmin Object에서 Channel을 통하여 변경된 정보가 Client로 전달되므로 Client에서는 실시간으로 변경된 정보를 받아 볼 수 있다. 이 경우 PUSH 기술을 사용하므로 서버 쪽에서 이벤트가 발생할 때에만 변경된 정보를 Client쪽으로 전달하게 된다.

따라서, 새로운 주식을 등록할 경우 Client에서는 주식 등록을 위해 StockAdmin 객체의 addStock() 메소드를 ORB를 통하여 호출하게 되고 addStock() 메소드는 DBAdmin Object에 addStock() 메소드를 호출하여 실제 DB에 새로운 주식을 등록한다. 그 후 Channel을 통해서 Client로 데이터가 전송되고 Client에서는 변경된 데이터를 사용자에게 보여준다.

주식 삭제도 위의 방법과 동일한 방법으로 deleteStock()를 호출하며, 또한, updateStock()를 호출함으로써 변경된 주식정보 즉, 현재가, 전일가 등을 Update한다.

(2) OrderAdmin Object

OrderAdmin Object는 Client에서 주식주문을 하고자 할 경우 호출되는 객체로, 실제로 주식매매가 이루어질 경우 DBAdmin 객체를 호출하여 DB에 데이터를 변경시킨다. 또한 OrderAdmin은 Channel을 통하여 실시간으로 거래결과를 update하여 Client 화면에 push해준다. 거래결과 뿐만 아니라 모든 사용자의 거래주문 내역을 실시간으로 볼 수 있도록 보고서를 작성하는 getFillReportList() 메소드도 포함되어 있다.

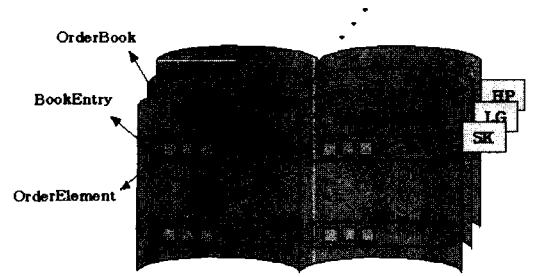


(그림 3) 이벤트 채널을 이용한 연결 구조

OrderAdmin Object는 OrderBook 인스턴스를 포함하고 있으며, OrderBook의 개념 및 구조는 그림 4에서 보이는 바와 같이 클래스 계층구조로 구현하였다.

OrderBook은 주식거래에 있어서 매도 및 매수를 구분하여 거래된 주식에 대한 정보를 제공하고 시장가 거래인지 주문가 거래인지를 구분하기 위한 클래스로서 BookSide의 인스턴스인 SellBookSider와 BuyBookSide로 이루어져 있다. 실질적인 주식 거래 관련기능을 하는 BookSide 클래스는 BookEntry 인스턴스들로 이루어져 있으며, 주식거래를 할 경우 일차적으로 주문가격 및 주문량을 비교하기 위한 정보를 제공하는 BookEntry 클래스는 OrderElement 인스턴스들로 구성되어 있다. OrderBook 클래스는 거래 정보를 전달하기 위해 getTradingInfo(), getLastSaqlePrice() 등의 메소드를 포함하고 있으며, 시장가 거래와 주문가 거래를 구분하여 거래하기 위해 marketOrderTrading(), limitOrderTrading() 메소드가 포함되어 있다.

BookSide 클래스는 거래된 적이 없는 주식이 거래되었을 경우, BookEntry를 추가하기 위한 add



(그림 4) OrderBook 클래스의 개념 및 구조

Order() 메소드와 시장가 거래를 위한 marketTrade() 메소드, 주문가 거래를 위한 limitTrade() 메소드 등이 포함되어 있으며, 실제로 DB에 존재하는 데이터를 변경시킨 후, Channel을 통해 변경된 데이터를 Client로 전달한다.

BookEntry 클래스에는 주문이 완료되었을 경우, OrderElement 인스턴스를 추가하는 addOrder() 메소드와 주문이 취소되었을 경우 주문을 삭제하는 deleteOrder() 메소드가 포함되어 있으며, 주문량을 알려주는 getTotalQuantity() 메소드, 주문가격을 알려주는 getPrice() 메소드가 포함되어 있다.

OrderElement 클래스는 quantity(주문량), orderId (주문번호), userId(사용자 ID)의 public 변수들로 이루어져 있다.

(3) UserAdmin Object

UserAdmin Object는 사용자가 계정을 추가하거나 사용자 인증을 할 때 ORB를 통해 Server에서 호출되는 객체이며, Server의 다른 Object들과 마찬가지로 DB에 접근해야하는 작업이 필요한 경우에 DBAdmin Object을 호출한다. User Admin Object의 메소드로는 User를 추가하는 addUser(), 사용자인증을 위한 verifyUser() 메소드가 있다.

(4) DBAdmin Object

DBAdmin은 StockAdmin, UserAdmin, Order-Admin이 데이터베이스에 접근할 필요가 있을 때 호출되는 객체로서 JDBC를 이용하여 데이터베이스에 접근한다.

이 객체는 다양한 형식의 데이터 크기를 처리하고, 데이터베이스로부터 추출된 결과값을 StockAdmin 이나 UserAdmin 또는 OrderAdmin 객체로 반환하며, 각각의 객체는 결과값을 각각의 Client들로 반환한다.

또한, 이 객체는 데이터베이스에 연결, 연결을 해지하는 메소드와 객체들의 요청에 의하여 데이터를 생성, 수정 및 삭제하는 메소드를 포함한다.

4. 결 론

가상 공간에서 전자 상거래는 기업의 경영자와 소비자가 누리게 될 초현대적인 편리함과 효율성을 제공해 주게 될 것이다. 따라서, 최근에 전자 상거래가 활발해지고 있으며, 전자 상거래 시스템을 개발하는 기업들이 계속 증가하고 있다.

그러나 개발되는 시스템을 기존의 시스템에 연동시키기 위한 표준 방법, WWW에서 제공하는 외부 시스템 연동 방법상의 문제, 개발된 시스템의 유지, 보수문제 등이 전자 상거래의 확산을 방해하고 있다.

본 연구에서는 기존의 전자상거래 방식에서 벗어난 ObjectWeb 환경하에서 주식매매 관리 시스템을 설계, 구현하였다.

객체간에 상호 연동 기능을 지원하는 ORB와 플랫폼에 독립적인 JAVA와 CORBA(Common Object Request Broker Architecture) 서비스 중의 하나인 이벤트 서비스의 PUSH기술을 이용하여 인터렉티브한 환경을 구현하였고, 미들웨어인 CORBA와 JDBC를 사용함으로써 3-Tier 기반의 시스템을 구축하여 확장성, 속도, 유연성, 보안성, 유지 · 보수, 효율성이 뛰어난 주식 매매 관리 시스템을 구현하였다.

본 시스템에서는 CORBA ORB 자체에서 제공되는 보안 서비스를 이용하였다. 그러나, CORBA에서는 응용프로그램 수준에서도 보안 서비스를 정의하여 사용할 수 있도록 지원해 주고 있다. 본 시스템에는 이 부분은 포함되어 있지 않으며, 사용자 인증 부분이 역시 포함되어 있지 않다. 따라서 향후, 본 시스템에 보안 서비스 부분과 사용자 인증 부분의 추가가 필요하다.

Acknowledgement

이 논문은 1999년 한국 과학기술 평가원에서 지원된 연구비에 의해서 연구되었음.

참 고 문 헌

[1] 김석원, "CORBA 구조 및 제품현황", 정보과학회지, 제17권, 제7호, p.5-6, 1999.

[2] 이재완, "분산 객체 통합을 위한 CORBA 이벤트 서비스 기법", 한국인터넷정보학회, Vol. 2, No. 1, pp.77-85, 2001.

[3] 왕창종, 이세훈, Inside CORBA3 프로그래밍, 대림, 1999.

[4] Robert Orfail, Dan Harkey, Jeri Edwards, Client/Server Survival Guide, 3rd ed, Wiley, 1998.

[5] Robert Orfail, Dan Harkey, Client/Server Programming with JAVA and CORBA, 2nd ed, Wiley, 1998.nh

[6] 김홍근, 최영철, "전자상거래 정보보호기술 현황 및 대응방안", 정보처리학회지, 제6권,제1호, pp.22-26, 1999.

[7] JAVA2 SDK, Standard Edition version1.3 Summary of New Features and Enhancements

[8] What's Coming in CORBA3, http://www. omg. org/news/pr98/component.html

[9] Object Soft, http://www.objectsoft.co.kr/

[10] Client/ServerArchitecture,http://www.hanji n.com/~rhh/learning/tier basic.html

[11] 분산 컴퓨팅/미들웨어, http://user.chollian. net/~ces90/info/middle ware/

[12] Internet & Electronic Commerce, http://www. korec.com

◐ 저 자 소 개 ◑

황 준

1985년 중앙대학교 전자계산학과 졸업(학사)
1987년 중앙대학교 대학원 전자계산학과 졸업(석사)
1991년 중앙대학교 대학원 전자계산학과 졸업(박사)
1992~현재 서울여자대학 정보통신공학부 교수
관심분야 : 분산처리시스템, 실시간 시스템
E-mail : hjun@swu.ac.kr

김 영 신

1999년 서울여자대학교 컴퓨터학과 졸업(학사)
1999~현재 서울여자대학교 대학원 컴퓨터학과 재학중(석사)
관심분야 : 실시간 시스템
E-mail : amaryllis@empal.com