Journal of Korea Multimedia Society Vol. 9, No. 1, January 2006(pp. 101-108)

# 디자인 패턴을 적용한 네트워크 게임의 DB 관리 APIs 설계

김종수[†], 김태석[††], 권오준[†††]


## 요 약

현재 인터넷에서 서비스되고 있는 네트워크 게임을 개발하는 경우, 개발 인원과 시간이 많이 투입되는 프로젝트이기 때문에, 게임 제작 시 기존에 작성되어 있는 코드를 재사용이 가능하도록 설계하는 것은 중요한 일이다. 게임에 사용되는 데이터베이스는 많은 클라이언트들이 접근하는데, 자원의 효율적인 관리를 위해서, 데이터베이스의 접근횟수를 최소화하고 데이터를 효율적으로 처리할 수 있는 API(application program interface) 설계가 필수적이다. 이러한 문제를 해결하기 위해 관련 소프트웨어 모듈의 객체지향적인 설계가 필요하다. 본 논문에서는 데이터베이스 자원을 다루는데 필요한 효율적인 API 구현을 위해 GoF(gang of four)의 디자인 패턴을 제안한다. 몇 개의 게임 GUI(graphical user interface) 분석을 통해 설계된 데이터베이스는 일반적인 데이터베이스 설계 단계에서 설계의 검토 및 수정을 최소화하여 최적화된 스키마 집합을 빠른 시간에 생성할 수 있다는 장점이 있고, 이것을 기초로 데이터베이스 서버 측 API 설계에 GoF의 디자인 패턴을 적용함으로써 게임 서버와 데이터베이스의 호출 횟수가 실질적으로 감소하고, 개발된 API를 손쉽게 유지보수 할 수 있으며, 새로운 API의 추가가 쉽다는 장점이 있었다.


# The APIs Design for the Database Management of the Network Game Using Design Patterns

Jong-Soo Kim[†], Tai-Suk Kim[††], Oh-Jun Kwon[†††]


## ABSTRACT

Developing a network game that is serviced on the internet requires a lot of work, time and manpower. Therefore, it is important to design a game in such a way so that existing design codes could be used. The database for the multi-player network game is accessed from many clients. To manage the resource effectively, it is required to design the APIs to be minimized the database access and to be dealt with the related data efficiently. For this, it is needed to apply the object-oriented design for the related software modules. In the paper, we propose the design patterns of GoF to implement the APIs thatis needed to deal with the database resource. The database design through the analysis of some game's GUI has the advantage to create the optimized schema set more quickly, because it minimize the review step and the modification step of the database design generally. In addition, we apply to the design patterns of GoF for the APIs design of the server-side database. These reduce the times of the program call between the game server and the database server. These also make easily the maintenance for the already developed APIs, and it makes easily the addition of new APIs.


**Key words:** Design Patterns(디자인 패턴), Database Design(데이터베이스 설계), Network Game(네트워크 게임)

※ 교신저자(Corresponding Author) : 김태석, 주소 : 부산광역시 부산진구 엄광로 995(가야동 산24번지) (614-714), 전화 : 051)890-1707, FAX : 051)890-1724
E-mail : tskim@deu.ac.kr
접수일 : 2005년 7월 21일, 완료일 : 2005년 8월 26일

[†] 정회원, 동의대학교 정보통신연구소
(E-mail : seatree@deu.ac.kr)
[††] 종신회원, 동의대학교 공과대학 소프트웨어공학과 교수
[†††] 종신회원, 동의대학교 공과대학 소프트웨어공학과 교수
(E-mail : ojkwon@deu.ac.kr)

## 1. 서 론

MMORPG(massive multi-player online role playing game)와 같은 게임에서는 사용자의 흥미를 유발시키는 클라이언트 측 구현이 중요한 요소이지만, 게임 운영자 측면에서 볼 때는 클라이언트에게 안전하고 지속적인 서비스를 해주는 게임 서버의 제작이 더 중요한 요소라고 볼 수 있다.

네트워크 게임의 구조는 분산된 작업을 가능하게 하고, 기존에 개발된 소프트웨어의 재사용성을 높이기 위해 다계층(multi-tier) 구조로 제작되는데, 데이터베이스 서버의 경우는 새로운 계층으로 분리하는 것이 거의 일반적이다. 다계층으로 설계되는 애플리케이션의 한 종류인 온라인 게임 분야 중에서도 MMORPG와 같은 게임 서버의 구조는 그 성격이 폐쇄적이므로, 자료의 공유가 어렵다.

본 논문에서는 네트워크 게임의 GUI 분석을 통해서 일반적으로 사용될 수 있는 데이터베이스 설계 예를 보인다. 그리고 이것을 바탕으로 클라이언트와 게임서버가 주고받아야 하는 패킷들 중에서 데이터베이스로 저장되어야 하는 데이터를 관리하는데 사용될 수 있는 효율적인 API의 설계에 GoF 디자인 패턴을 적용하였다.

게임의 GUI 분석을 통한 데이터베이스 설계는 설계의 검토 및 수정 작업을 최소화하여 최적화된 스키마 집합을 빠른 시간에 생성할 수 있다는 장점이 있었고, 데이터베이스 서버 측 API 설계에 적용된 GoF의 싱글톤(Singleton), 퍼사드(Facade), 커맨드(Command) 패턴은 개발된 API를 손쉽게 유지보수 할 수 있으며, 새로운 API의 추가가 쉽다는 장점이 있었다.

## 2. 관련 연구

게임 서버를 구성하기 위해서는 하드웨어적으로 구성되는 네트워크 구조, 클라이언트에서 발생한 중요한 데이터를 저장하는 데이터베이스 서버와 관련된 기술 그리고 관련 API 구현을 위한 UML(unified modeling language) 설계 기술이 필요하다. 본 장에서는 게임 서버와 관련된 기술 중 서버의 구성 방식, 데이터베이스 관련 기술 그리고 디자인 패턴과 관련된 기술에 대해서 알아본다.

### 2.1 게임 서버의 네트워크 구성

3D 게임 엔진은 크게 클라이언트 부분과 서버 부분으로 나눌 수 있다[1]. 게임 서버의 설계와 관련하여 현재 많이 이용되고 있는 네트워크 구성 기술은 P2P(peer to peer), C/S(client/server), 분산 서버 구조가 있으며, 이것을 조합한 하이브리드(hybrid)형 서버 구조가 있다.

P2P 방식은 실시간 전략 시뮬레이션 게임이나 액션 게임의 경우에 많이 응용되고 있는 방법으로 게임 플레이어의 컴퓨터가 직접적으로 네트워크 상에서 상대편을 찾고, 상대편이 발견된 다음에는 다른 컴퓨터의 개입 없이 두 사람의 컴퓨터 간에 서로 메시지를 주고받는 방식을 말한다.

C/S 방식은 여러 대의 클라이언트가 서버와 자료를 주고받는 방식을 말한다[2]. C/S 방식의 서버 구성에서 모든 클라이언트는 공유해야 할 정보를 서버에 보내며 서버는 이러한 입력을 기반으로 게임의 모든 진행 결과를 전체 클라이언트에게 보낸다. C/S 방식으로 네트워크를 구성했을 경우, 고려해야 할 사항은 서버에 접속한 클라이언트들로부터 들어온 패킷을 얼마나 효율적으로 처리할 수 있는가하는 것이다[3,4].

분산 서버 구조는 대규모 네트워크 게임을 위해서 적합한 구조인데, 그림 1과 같은 구성을 가진다. 분산 서버 구조에서 게임 클라이언트는 여러 개로 묶어진 서버군 중 한 개의 서버 군에 접속하여 게임을 진행하게 되는데, 이렇게 함으로써 서버의 작업량을 여러 서버로 나누는 부하 분산의 효과가 있다.

네트워크 게임 개발의 현재 추세는 PC 게임과 게임 전용기 시장의 침체와 유무선 인터넷 인프라의 발전과 이를 연계한 문화 컨텐츠의 다양화로 인하여, 윈도우, 리눅스, PocketPC 등 다양한 운영체제에서

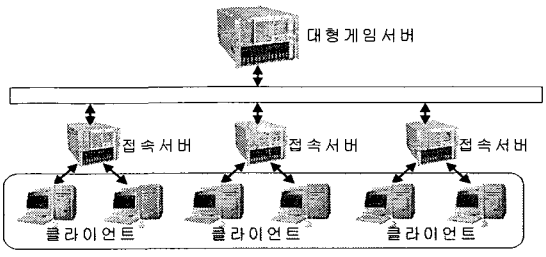


그림 1. 분산 서버 형식의 게임 서버 구성

한국어, 일본어, 영어 등의 다국어를 지원하는 크로스플랫폼 게임 콘텐츠 개발로 변하고 있다. 온라인 게임과 모바일 게임이 발달한 국내에서는 온라인게임과 모바일 게임간의 연동 기능을 가진 크로스플랫폼 게임 개발이 많이 이루어지고 있다[5].

### 2.2 데이터베이스 서버에 대한 검토

네트워크 게임 서버의 데이터베이스 구성에 있어서 데이터베이스 서버의 분리는 다른 서버의 일부 작업들은 데이터베이스 시스템과 같이 수행하고, 데이터베이스와 관련되지 않은 작업들은 기타 게임 서버가 담당하게 함으로써 작업을 분담시킬 수 있다.

데이터베이스 서버 시스템은 크게 트랜잭션서버와 데이터서버로 분류할 수 있다[6]. 트랜잭션서버 시스템은 클라이언트가 동작을 수행하기 위한 요청을 서버에 보내고, 그 요청에 대한 응답으로 시스템이 동작을 수행하고 결과를 클라이언트에게 다시 보내는 SQL이나 별도의 API를 사용하는 인터페이스를 제공한다. 데이터서버 시스템은 클라이언트가 파일이나 페이지와 같은 단위로 데이터를 읽거나 갱신을 요청함으로써 서버와 접속하는 시스템이다. 일반적으로 트랜잭션서버와 데이터서버 중에서 트랜잭션서버 시스템이 널리 사용된다.

### 2.3 GoF의 디자인 패턴(Design Patterns)

소프트웨어의 객체지향 설계에서 우선적으로 관심을 두어야 하는 것은 문제 영역에서 어떻게 클래스들을 추출할 것인가? 하는 것인데, 책임을 제대로 할당하는 것이 설계에서 매우 중요하다. 객체의 설계와 관련된 기본원리 중의 하나가 책임 할당의 일반적인 원리에 관한 패턴인 GRASP(general responsibility assignment software patterns)이다.

많은 객체들이 필요한 게임 소프트웨어에서 추출된 객체들은 GRASP의 정보 전문가(Information Expert)와 같은 패턴에 의해, 각각의 객체가 데이터베이스 관리와 필요한 메소드를 구현할 수도 있다. 그러나 이러한 설계의 단점은 특정 객체의 결합도를 높임으로써, 소프트웨어 재사용의 중요한 요소인 낮은 결합도(Low Coupling)를 지원하지 못한다. 이러한 문제를 해결하기 위해 소프트웨어 개발자가 다른 사람의 전문성을 쉽게 이용할 수 있도록 해주는 GoF의 디자인 패턴 사용은 매우 효율적이다.[7,8].

### 2.4 네트워크 게임을 위한 일반적인 데이터베이스 설계

관계형 데이터베이스 설계의 목표는 불필요한 데이터의 중복을 없애고, 정보 검색을 쉽게 할 수 있는 관계 스키마 집합을 생성하는 것이다. 본 논문에서는 일반적인 시스템 설계 과정과 다르게 이미 네트워크 게임으로 성공적으로 운영되고 있는 몇 개의 게임을 바탕으로 게임에서 사용하는 몇 가지 GUI를 이용하여 데이터베이스 설계를 분석하고, 일반적인 네트워크 게임에 적용할 수 있는 데이터베이스 설계 예를 보인다.

일반적인 네트워크 게임에서 클라이언트가 조작하는 캐릭터는 게임 내에서 다양한 아이템을 가질 수 있는데, 게임 중에 캐릭터가 습득한 아이템과 현재 캐릭터가 착용하고 있는 아이템에 대한 정보를 보여주는 GUI가 있고, 장비, 소비, 설치, 기타, 펫(Pet)등의 정보를 보여주는 GUI가 있다. 이러한 몇 개의 GUI를 바탕으로 하여 분석된 데이터베이스의 설계는 그림 2와 같다.

그림 2의 ERD에서 전체 기술에 대한 정보는 skill 엔티티에서 관리하고, 캐릭터가 사용할 수 있는 기술에 대한 정보는 c_skill 엔티티가 담당한다.

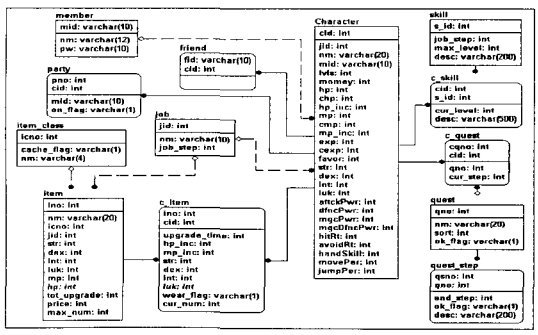


그림 2. 네트워크 게임을 위한 DB 설계

## 3. APIs 구현을 위한 GoF의 디자인 패턴 적용

게임 서버의 구현에 많은 소프트웨어적인 기술이 필요하다. 게임 소프트웨어 설계에 UML을 사용하는 것은 효과적이다. 특히 데이터베이스 서버와 같이 별도의 계층으로 존재하는 애플리케이션의 인터페

이스에 퍼사드(Facade)와 같은 디자인 패턴은 아주 유용하다. 본 절에서는 데이터베이스 설계의 예를 기초로 네트워크 게임 설계에서 일관된 인터페이스를 제공하기 위한 디자인 패턴의 적용을 검토해 보고, 이것을 사용한 효율적인 데이터베이스 서버를 구축하기 위한 방안을 제시한다.

### 3.1 데이터베이스 시스템을 사용하기 위한 퍼사드 (Facade) 패턴 적용

관계형 데이터베이스와 같이 하위 시스템을 구성하는 다수의 객체들을 효율적으로 사용하기 위해 공통된 인터페이스를 제공할 수 있는 패턴이 퍼사드(Facade) 패턴이다. 네트워크 게임 서버 구성에서 클라이언트로부터 들어온 요청에 효율적으로 응답하기 위해서 서버를 기능별로 분산하는 것은 아주 좋은 방법이다. 일반적으로 서버군은 채팅서버, NPC서버, 퀘스트 서버, 데이터베이스 서버 등으로 분리할 수 있다. 클라이언트가 획득한 정보를 데이터베이스로 저장하기 위해서 게임 서버를 구성하고 있는 여러 개의 객체들로부터 데이터베이스의 자원에 접하려고 하는 시도가 자주 일어난다. 그림 3은 데이터베이스 자원에 접근할 수 있는 여러 서버 애플리케이션을 보여준다.

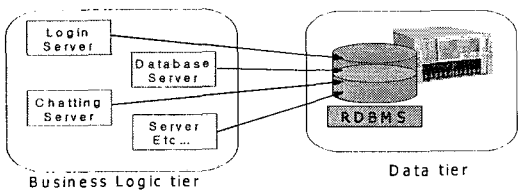


그림 3. Database 자원에 접근하는 서버 애플리케이션들

데이터베이스 서버의 구현에 그림 4와 같이 퍼사드를 적용하는 이유는 설계 단계에서 하위 시스템간의 연결성과 종속성을 최소화하려는 목적 때문인데, 퍼사드 패턴을 이용하여 객체를 구현함으로써, 복잡한 인터페이스를 단순화된 하나의 인터페이스로 구성할 수 있다.

트랜잭션-서버는 버퍼 풀(buffer pool), 질의 계획 캐쉬(query plan cache), 로그 버퍼(log buffer)와 같은 객체를 가지며, 이러한 객체에 특정한 요청이 들어오면 이것을 처리하기위한 서버 프로세서, 록 관리자 프로세서, 데이터베이스 쓰기 프로세서 등이 일련의 작업을 수행한다. 일반적으로 데이터베이스 시스템이 가진 자원에 접근하기 위해서 마이크로소프트사가 제공하는 ODBC(open database connectivity)나 자바의 JDBC(java database connectivity) 또는 기타 API등을 이용한다.

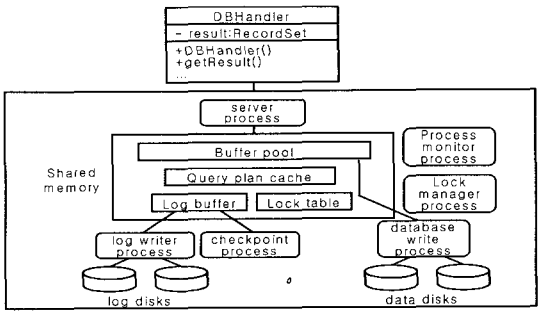


그림 4. DBMS를 활용하기 위한 Facade 패턴 적용

대부분의 애플리케이션 개발자들은 데이터베이스 내부의 구체적인 정보 없이 질의를 사용하여 데이터를 가져와서 가공하기만 하면 되고, 실제 데이터베이스 시스템의 내부가 어떻게 동작하는지 알 필요까지는 없다. 이러한 인터페이스 역할을 하는 클래스를 퍼사드(Facade) 객체로 정의할 수 있다. 데이터베이스 클래스는 사용자에게 데이터베이스를 사용하는데 필요한 가장 필수적인 인터페이스만 제공하면 되고, 내부적으로 데이터베이스는 버퍼 풀, 쿼리 계획 캐쉬 등과 같이 동작하면 된다.

게임 개발자가 복잡한 데이터베이스 시스템의 내부 구현까지 이해해 가면서 데이터베이스 시스템을 사용하기란 불가능하다. 이러한 문제를 해결하기 위해서 퍼사드 패턴을 사용하는 것이 데이터베이스 시스템과의 결합도를 줄일 수 있다. 즉 데이터베이스 시스템의 모든 인터페이스가 공개될 경우 빈번한 메소드 호출이 있을 수 있지만, 개발자에게는 통합된 단순한 인터페이스를 제공하고 나머지 부분은 내부적으로 처리함으로써 게임 서버와 데이터베이스 서버 사이의 호출 횟수가 실질적으로 감소하게 되는 효과가 생긴다.

### 3.2 퍼사드를 이용한 설계에서 싱글톤(Singleton) 패턴 적용

퍼사드 패턴을 적용한 앞의 설계에서, 데이터베이

스와 관련된 객체가 1개인 경우, 퍼사드 객체를 싱글톤으로 만들 것인가에 대해서 고려해 보아야 한다. DB 관리 클래스의 싱글톤 패턴 적용은 응집도를 높일 수 있다. 또한 퍼사드로 설계된 객체의 생성을 자신의 클래스가 담당하게 하는 것이 유지 보수 측면에서도 여러모로 편리하다.

### 3.3 서버 측 비즈니스 로직 구현을 위한 커맨드(Command) 패턴 적용

클라이언트가 보내오는 요청을 분석하고 처리하는데 파라미터를 이용하는 것이 일반적이지만, 커맨드 패턴을 사용하면 명령과 요청 자체를 객체안에 캡슐화하여 포함시킬 수 있다는 장점이 있으므로, 서버 측 비즈니스 로직 구현에 커맨드 패턴을 적용할 수 있다. 서버에서 처리해야하는 명령들을 클래스로 추상화하고, 오퍼레이션을 호출하는 객체와 오퍼레이션 수행 방법을 구현하는 객체를 분리하여, 이것을 사용하는 다른 객체들은 명령의 내용을 알지 않고도 실행할 수 있다. 그리고 명령어를 조합해서 다른 명령어를 쉽게 만들 수 있고, 새로운 명령어를 쉽게 추가할 수 있다는 장점이 있다.

### 3.4 효율적인 DB 접속을 위한 커넥션 풀(Connection Pool)의 사용

게임의 서버 측 애플리케이션 구현에서 비용 측면을 고려하면, 리눅스와 mysql을 사용할 수도 있다. 이 경우에는 마이크로소프트의 COM+ 라이브러리가 가지고 있는 커넥션 풀과 관련된 API를 사용할 수 없다는 단점이 있다. gcc 컴파일러를 이용한 리눅스용 게임 서버 개발에서 데이터베이스 관리를 위해 mysql이 제공하는 기본적인 C 라이브러리를 이용할 수 있다. 커넥션 풀은 풀 속에 데이터베이스와 연결된 커넥션들을 미리 생성해 놓고 커넥션이 필요할 경우, 미리 생성되어 있는 커넥션을 사용함으로써 커넥션을 생성하고 연결하는데 드는 시간을 줄여준다는 장점을 가지고 있다. 그리고 커넥션을 계속해서 재사용하기 때문에 생성되는 커넥션 수가 많지 않아서, 하드웨어 자원을 효율적으로 사용할 수 있다는 장점이 있다. 그러므로 리눅스 환경에서 게임서버를 구현하는 경우, 효율적인 자원 관리를 위해 gcc로 구현된 커넥션 풀의 구현이 필요하다고 할 수 있다. 커넥션 풀을 사용함으로써, DB는 많은 요청에 안정적으로 서비스할 수 있다. 그림 5와 같이 데이터베이스 연결과 관련하여 커넥션들의 풀을 만들어 관리할 수 있다.

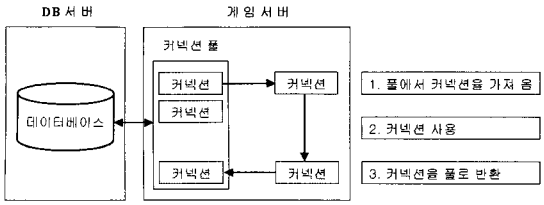


그림 5. 커넥션 풀의 개념도

게임 서버로 구현되는 많은 클래스들이 필요한 커넥션을 얻어 오는 코드의 중복을 막기 위해 커넥션 풀 API 자체가 싱글톤(Singleton)으로 구현되는 것이 검토되어야 한다.

## 4. 디자인 패턴을 적용한 APIs와 시스템 구현

생성된 스키마를 사용하기 위한 RDBMS (relational database management system)로 여러 가지가 있을 수 있지만, 손쉽게 구할 수 있고 가격대 성능비가 우수한 mysql을 사용하였다. 네트워크 게임에서 통신을 위해 사용되는 DirextPlay를 이용해 간단하게 정보를 주고받을 수 있는 클라이언트와 서버는 C++로 구현하였다.

그림 6에서 클라이언트와 서버는 서로의 통신을 위해서 2500 포트를 사용하고, 게임 서버 내에서 데이터베이스를 관리하기 위한 API는 3306포트로 mysql과 통신한다.

### 4.1 데이터베이스 서버 측 구현

GUI를 바탕으로 생성된 스키마 집합은 mysql에 생성되었다. 그림 7은 그림 2에서 제시한 네트워크 게임을 위한 데이터베이스 설계로부터 mysql에 만들어진 테이블 리스트를 보여준다.

게임에 대한 정보를 저장하기 위한 game 데이터베이스를 생성하여 게임에 필요한 전체 테이블을 생성하였다.

테이블 생성 시, 유일한 레코드를 임을 보장하는 기본 키(primary key)는 테이블과 같이 생성되었다.

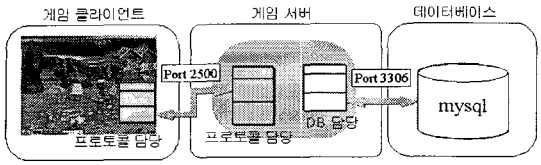


그림 6. DB관리 API를 테스트하기 위한 시스템 구성

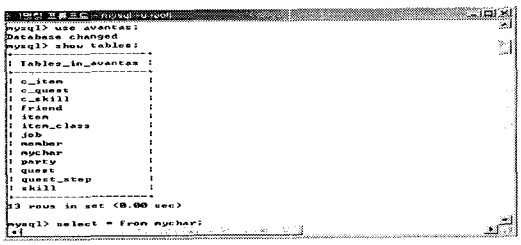

그림 7. mysql에 생성된 스키마 집합

## 4.2 클라이언트/서버 통신을 위한 애플케이션 구현

게임에 필요한 프로토콜은 DirectPlay를 이용하여 간단한 모듈로 설계되었다. 게임 클라이언트는 그림 8, 게임 서버는 그림 9와 같이 구현되었다.

서버 측과 클라이언트 측 애플리케이션 구현에 커맨드 패턴이 적용되었고, 퍼사드 패턴은 서버 측에서 데이터베이스와 관련된 작업을 하기 위해 적용되었다. 새로 추가된 User 클래스와 이것의 정보를 가져오고 추가하는 UserGetCommand, UserSetCommand는 Command 패턴이 적용되기 전에는 다음과 같은 코딩을 가진다.

```
void parseCommand(){
    int command;
    if(command != -99) {
        switch(command) {
            case 9 : login();
                     break;
            case 8 : getCharacter();
                     break;
            case 6 : getUser(); //새로 추가된 코드
                     break;
            default: greeting();
        }
    }
}
```

커맨드(Command) 패턴을 적용하지 않을 경우, 클라이언트와 서버가 처리해야 할 작업이 있을 때,

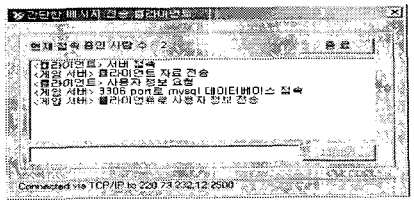

그림 8. 간단한 메시지 전송 클라이언트

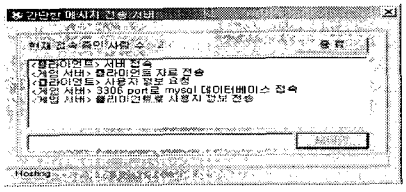

그림 9. 간단한 메시지 전송 서버

그것을 처리하기 위한 한개의 메소드가 정의될 수 있다. 메소드 parseCommand()에서 로긴, 캐릭터 조회, 채팅, 데이터베이스 접속, 자료 조회와 같은 작업을 위해 get과 set으로 정의된 메소드(getLogin(), getCharacter(), getUser(), setUser() 등등)를 클라이언트와 서버가 보내오는 IO(input output) 스트림을 가공하는데 사용한다. 그러나 이러한 방법은 사용자(User)와 같은 다른 객체가 추가되어 통신을 해야 하는 경우에 parseCommand() 메소드를 수정해야하고, 새로운 클래스와 그것을 다루는 새로운 메소드를 추가되어야 한다는 단점이 생긴다. Command 패턴의 사용으로 이러한 단점을 보완할 수 있다.

그림 10은 서버 측 구현에 적용된 퍼사드와 커맨드 패턴을 보여 준다.

다이어그램에서 애플리케이션 구현에 명령과 요청을 캡슐화하기 위해서 Command 추상클래스를 상속받는 LoginCommand, LoadCharacterCommand, SaveCharacterCommand, UserGetCommand, UserSetCommand 클래스를 추가하였다. 각각의 클래스는 부모 클래스인 Command로부터 상속받은 가상 excute() 메소드를 구현하고 있다. 클라이언트와 서버와의 통신에 커맨드 패턴을 적용함으로써, 서버의 자료를 요청하는 객체와 요청 정보에 따른 수행을 분리하여 별도의 객체로 구현할 수 있다는 장점이 있었다. 또한 메시지를 주고받는데 사용되는 새로운 명령의 추가가 쉽다는 장점이 있었다.

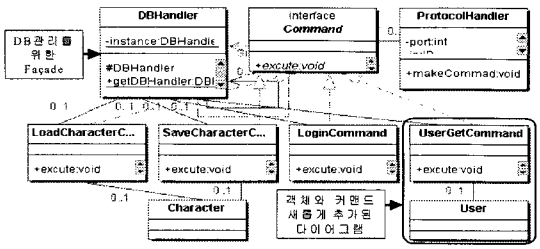


그림 10. Facade와 Command 패턴이 추가된 다이어그램

커맨드(Command) 패턴을 이용하는 코드는 일반적으로 다음과 같이 구현된다.

```
void parseCommand(Command *p) {
    Command *command;
    switch(p->type){
        case LoginCommand:          /* 로긴 */
            command = new LoginCommand();
            break;
        case LoadCharacterCommand:  /* 캐릭터 로딩 */
            command = new LoadCharacterCommand();
            break;
        case SaveCharacterCommand:  /* 캐릭터 저장 */
            command = new SaveCharacterCommand();
            break;
        case UserGetCommand:
            /* 사용자 리스트 조회(새로 추가된 커맨드) */
            command = new UserGetCommand();
            break;
        case UserSetCommand:
            /* 사용자 리스트 갱신(새로 추가된 커맨드) */
            command = new UserSetCommand();
            break;
    }
    Result result = command->excute(param)
}
```

게임 서버의 구현에서 데이터베이스를 연결하기 위한 DBHandler 클래스에 퍼사드 패턴이 적용될 수 있다. 퍼사드를 적용함으로써, 게임 구현과 관련된 API들이 데이터베이스 자원에 접근하는데 있어서 일관된 인터페이스를 사용할 수 있다는 장점이 있다. 퍼사드 객체인 DBHandler는 그림 11과 같이 싱글톤(Singleton)으로 구현될 수 있다.

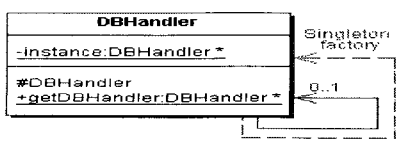


그림 11. Facade에 적용된 Singleton 패턴

싱글톤 패턴이 적용된 DBHandler 클래스는 다음과 같이 구현된다. 코드에서 객체의 인스턴스는 private 영역, DBHandler() 생성자는 protected 영역, 객체를 얻기 위한 getDBHandler() 메소드는 public 영역에 생성된 것을 볼 수 있다.

```
class DBHandler {
protected:
    DBHandler();  //보호 영역에 설정된 생성자
public:
    //인스턴스를 얻기 위한 메소드
    static DBHandler * getDBHandler();
    //... 생략
private:
    //개별 영역에 선언된 DBHandler클래스의 객체
    static DBHandler * instance;
    //...생략
};
```

객체의 인스턴스가 private으로 선언되고 정적영역에 생성되므로 DBHandler 객체는 유일한 객체가 된다. private 영역에 static으로 선언되었기 때문에 인스턴스에 접근하기 위해서 public 영역에 선언된 static 메소드인 getDBHandler()가 필요하다.

DBHandler의 인스턴스가 필요한 객체들은 static 메소드인 getDBHandler()를 호출함으로써, DBHandler 클래스의 유일한 인스턴스에 접근할 수 있다.

## 5. 결 론

본 논문에서 DirectX API를 기반으로 일반인들에게 크게 인기를 얻고 있는 네트워크 온라인 게임의 GUI를 바탕으로 서버 측 구현에 필요한 데이터베이스 설계의 한 예를 보였고, 이것과 관련된 API의 구현에 GoF가 제안한 디자인 패턴이 사용되었다.

게임의 GUI 분석을 바탕으로 한 데이터베이스 설계는 개발 단계에서 반복되어지는 관계형 데이터베이스의 설계를 최소화 시켜준다는 장점이 있었고, 데이터베이스 설계 툴인 ER-Win을 사용하여 만들어진 다이어그램은 게임에 필요한 테이블 스키마를 쉽게 생성할 수 있었다.

효율적인 소프트웨어의 설계를 위해 사용된 구조 패턴의 영역에서 퍼사드(Facade) 패턴은 데이터베이스를 사용하는데 하나의 통합된 인터페이스를 제공하여 하위 시스템과의 종속성을 줄임으로써, 데이

터베이스를 사용하는 서버 측 애플리케이션과 데이터베이스 시스템사이에 실질적인 호출횟수를 감소시킬 수 있다는 장점이 있었다. 퍼사드 객체를 만드는데 있어서 적용될 수 있는 싱글톤(Singleton)은 인스턴스가 유일함을 보장함으로써, 메모리를 효율적으로 사용할 수 있었다.

*DirectPlay*를 사용하여 구현된 간단한 클라이언트와 서버의 통신을 테스트하기 위한 애플리케이션 구현에 커맨트 패턴이 적용되었다. 행위패턴 영역의 커맨드(Command) 패턴은 상속 기법을 이용하여 오퍼레이션 수행과 구현객체를 분리하는 방법을 제공하여, 분산작업이 가능하였다.

다수의 사용자가 데이터베이스에 동시에 접속하는데 있어서 부하를 감소시키기 위해 커넥션 풀 API의 구현이 검토되었다. 커넥션 풀 API의 구현은 메모리를 절약시켜주고 액세스 시간을 감소시켜 준다는 효과가 있었고, 커넥션 풀 API의 구현에 싱글톤이 효율적으로 적용되었다.

## 참 고 문 헌


[1] 남재욱, 온라인 게임 서버 프로그래밍, 한빛미디어, 서울, 2004.

[2] 김종수, "웹을 기반으로 한 JAVA 네트워크 게임 시스템의 설계와 구현," 부산외국어대학교 석사학위논문, pp. 1-11, 2003.

[3] 김종수, 이종민, 김태석, "생성 패턴을 사용한 네트워크 기반 게임 API 설계," 한국멀티미디어학회 추계학술발표대회논문집, 제6권, 2호(하), pp. 669-674, 2003.

[4] Soon-Kak Kwon, Jong-Soo Kim, and Tai-Suk Kim, "An Implementation Avatar Chatting System for Network-Based Multi-User Environment," *EALPIIT2003, Proceedings of the 3rd*, pp. 119-123, 2003.

[5] 한국게임산업개발원 편집부 편자, *2003년 게임백서*, 한국게임산업개발원, 서울, 2003

[6] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan, *데이터베이스 시스템 4판*, 한국맥그로힐(주), 서울, 2003.

[7] Craig Larman, *UML과 패턴의 적용*, 홍릉과학출판사, 서울, 2003.

[8] Erich Gamma, Richard Helm, Ralph Johnson, and Hohm Vissides, *GoF의 디자인 패턴*, (주)피어슨 에듀케이션 코리아. 서울, 2003.



**김 종 수**

1992년 2월 부경대학교 냉동공학과(학사)
1998년 12월 On Line Technology 대표
2003년 2월 부산외국어대학교 컴퓨터공학과 석사
2003년 3월~현재 동의대학교 소프트웨어공학과 박사과정
2003년 4월~2005년 8월 동의대학교 영상 미디어센터 PM연구원
2005년 9월~현재 동의대학교 정보통신연구소 PM연구원
관심분야 : 네트워크 게임 제작과 설계, Web 프로그래밍



**김 태 석**

1981년 경북대학교 전자공학과 졸업(공학사)
1989년 일본 KEIO대학 이공학부 계산기과학전공(공학 석사)
1993년 일본 KEIO대학 이공학부 계산기과학전공(공학 박사)
1993년 일본 국제전신전화연구소(KDD)기술고문
1993년 일본 KEIO대학 이공학부 객원연구원
1994년~현재 동의대학교 소프트웨어공학과 교수
관심분야 : 정보시스템, 기계번역, 인터넷비즈니스



**권 오 준**

1986년 2월 경북대학교 전자공학과 졸업(공학사)
1992년 2월 충남대학교 대학원 전산학과(이학석사)
1998년 2월 포항공과대학교 대학원 전자계산학과(공학박사)
1986년 1월 ~ 2000년 2월 한국전자통신연구원 선임연구원
2000년 3월~현재 동의대학교 컴퓨터 · 소프트웨어공학부 조교수
관심 분야 : 정보 보호, 컴퓨터 통신, 신경망 응용, 패턴 인식