JOURNAL OF KOREA MULTIMEDIA SOCIETY, VOL. 9, NO. 6, JUNE 2006(pp. 742-749)

Design of Network-based Game Using the GoF Design Patterns

Jong-Soo Kim†, Tai-Suk Kim††

ABSTRACT

In the network-based game where it uses the Internet infrastructure, the implementation is possible with the various methods. Such Applications are developed in the multi-tired architecture. There are many cases for the server to be separated from the hardware or the software. In this case, a lot of applications make the distributed process possible and are made as the multi-tiered architecture to develop the reusability of the existing software module. Especially, it is mostly general to separate for the case of a database server to a new tier. One of the important points of multi-tiered server side applications is security and because of this, it is difficult to share the related data about the design skill. Using design patterns, it gives help in reusing the existing written-code for the design of the game that needs a lot of money and time. Design Patterns are related to the software reuse. For the development of more efficient games, if well-defined design patterns are provided to the developers, then it would make more easy advanced game API and make possible the framework for the game development based on the API. Through the analysis of the general network-based game currently servicing in the Internet, in this paper, we discuss how to implement a business logic tier using database system among the server side architectures. The main objective of this article is to show an efficient APIs(Application Programming Interfaces) design method which can be used to manage the data that must be saved to the database system among the packets that client/server have to be exchange.

Keywords: GoF Design Patterns, Database Design, ERD(Entity Relationship Diagram), Network Game, Facade Pattern, Command Pattern

1. INTRODUCTION

Currently, in the entire Korean game market, the network game holds the major portion. In relation to a network game, the hardware and software of the computer are also developed together. The recent network game production tendency shows that it supports various character, animation and sound, and puts the keynote to make the game players feel the reality[1,2]. Specially, in the case of developing multimedia-applied games like the network games serviced recently, many staff are added in its development. To design and implement the network game, many domestic companies apply the object-oriented paradigm.

The Internet is popular at the present, and many companies service various contents on the World Wide Web. The network game service is the best method to form a large-scale community. In the Internet, a lot of applications take the client/server architecture as a basis, it is important to develop basic APIs that can save the needed data for the each client information to be saved in the database. However, such APIs design technique that is related with the database system is difficult to share due to the intellectual property protection. Moreover, it is not easy for each other design and implementation method to be compared because of the variety.

※ Corresponding Author : Tai-Suk Kim, Address : (614-714) Dept. of Software Engineering, Dong-Eui Univ., 995 Eomgwangno, Busanjin-gu, Busan, Korea, TEL : +82-51-890-1707, FAX : +82-51-890-1724, E-mail : tskim@deu.ac.kr

Receipt date : Feb. 1, 2006, Approval date : Mar. 14, 2006

† Insitute of Telecommunications Information, Dong-Eui Univ. Korea. (E-mail: seatree@deu.ac.kr)

†† Dept. of Software Engineering, Dong-Eui Univ. Korea.

It uses design pattern of object-oriented structures by using UML(Unified Modeling Language) to take advantage which is an object-oriented language. One of the methods to measure the quality of object-oriented systems are that foretell how much developers paid attention to cooperation methods among objects.

While the software is designed and implemented, as we know, it is useful to reuse the existing code at the design and implementation of related database APIs considering the time, charge, and efficient aspect[3]. This paper suggests design patterns of GoF(Gang of Four) on server side APIs design techniques to deal with a database system.

The application is made for the client/server architecture to be separated into two applications ; the first one charges the request of the client and the second one, a server, manages the request to come in from the client[4,5]. It is an important technique to the architecture organization of the server in dealing with the client's various requests efficiently. In the network-based game architecture that is composed of the hardware, it is needed technique that is related with database server that can save the important data of the client and is also the technique of the related with UML for design of the game application. This paper shows the architecture method of the server. The database design techniques and the applying design patterns are related server side application implementation. We use the C++ compiler for the implementation and suggest Facade, Singleton and Command patterns for efficient game design.

2. MATERIAL AND METHODS

2.1 Network Synthesis

In the multi-user network application, the reusability is considered for additional development and the easy maintenance[6]. It is efficient for that only if the interface of the various server applications may not related each other. In the network synthesis of the applications, There are P2P(Peer to Peer) based system, client/server-based system, distributed server system, and Hybrid server system that is mixed with the others.

Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request[7]. The basic frame is the multi-tiered architecture of the client/server-based system. Fig. 1 shows an example of the multi-tired architecture.

The architecture has the following advantages over one-tiered and two-tiered architectures:

- Makes wider data more readable.
- Easier approach through the network.
- Data can be reset more easily.

Table 1. shows the summarization of logical software and physical hierarchies for each tier's. In almost all sever-side applications, client/server architecture is selected partially from the whole. In the server-side application, all clients send the information that has to be share. After that, the server sends the result of the process to the clients on the basis of that input.

In the integrated network for the game application, the server has to deal with the packets coming from the connected clients at the server efficiently.

The processing about the requirement of the clients often depends on the information that the server has.

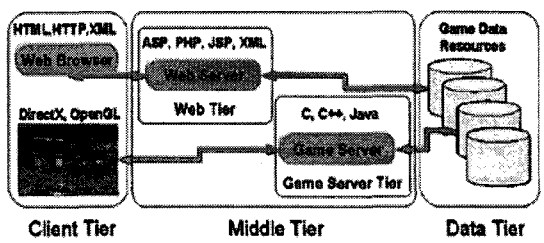


Fig. 1. An Example of the multi-tired architecture.

Table 1. Logical Software & Physical Hierarchy

Category	Logical software hierarchy		Physical hierarchy
1 tier	Presentation	Receive data, or processing (User Interface, Window, Screen)	Client
2 tier	Business logic	Business rule achievement	Application Server
3 tier	Data & Application	Data deposit and access	Database Server

It is needed the synchronization technique to solve the deadlock problem should be examined while the server side process the information that the client shares in the implementation of the server application[8].

The distributed server architecture is suitable for massive network applications. It has the architecture that is the same as Fig. 2. In the distributed server architecture, the client applications try to connect each of the server groups and then exchange each other. After that, the job load of server can be divided into several servers and it is efficient to reduce the load.

Among the server side architectures, the database system architecture you adopt is also important. Part of tasks of different servers accomplished with one database system and other work can share the task of the game server and make the client to charge. We can apply a distributed database or parallel database system to the data base architecture of a network server which is composed of several distributed servers, However, the distributed database system and the parallel database system stay at the development step yet, and it is also difficult to find a system worthy of referring to.

2.2 GoF(Gang of Four) Design Patterns

To help produce quality software is the goal of software engineering. Reusability is the basic issue in software engineering.

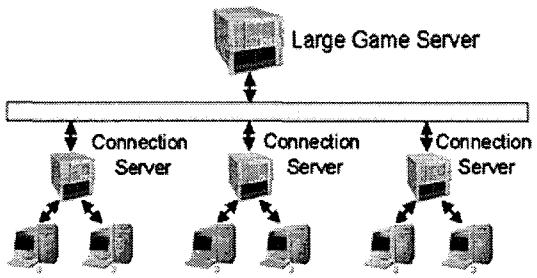


Fig. 2. The server side application architecture of the distributed server.

It is easy to find the already-written code for searching and other basic data structure manipulations. To make complex software, using design patterns is very efficient[9].

Using them makes the software developers use other people's excellent idea easily[10]. The developer can use a software design idea again using the design patterns. There are various patterns to design the object oriented software, but in this paper we show what GoF(Gang of Four) design patterns proposed.

GoF design patterns propose are classified as 3 pattern areas largely. They are Creational Patterns, Structural Patterns, and Behavioral Patterns. The Creational Patterns offer the inclusive method that determines the generation method of the object. The Structural Patterns uses a succession technique for forming bigger architecture. Finally, The Behavioral patterns offer organization, management and combination methods.

This paper shows one of the structural patterns, Facade pattern, to be applied to the implementation of APIs about a database processing of a network application. The intent of this pattern is to "Provide a unified interface with a set of interfaces into a subsystem. Façade defines a higher-level interface that makes the subsystem easier to use". We also suggest two creational patterns on the APIs design. One is Singleton Pattern. It is useful to "Ensure a class has only one instance and provides a global point of access to it". The other one is Command patterns. The command pattern is ex-

cellent for supporting undo. The idea is to have a base class that defines a method to "do" a command and another method to "undo" a command.

We need a lot of the software technique at the implementation of the application to deal with the date of the server. To use UML is quite useful for the software design[11-13]. Especially, in the application of the interface that exists into the class of the separate way with a database server, the Facade pattern is very useful. We examine the application of some design patterns based on the database system and present the plan to construct the efficient database server that uses those.

2.3 Example of the Game Database Design

Entity Relationship Diagrams(ERDs) illustrate the logical structure of databases[14]. The ERD of the database system analyzed and designed on the basis of the existing game is like Fig. 3.

In the diagram, the "Character" entity that manages the various characters that operated by the game users. And the employee entity has a "jid" attribute in the job entity as the foreign key to record the job of the Character entity. The "c_item" entity that saves the data concerning characters has the "item" entity to record the each game user's items. Fig. 4. shows that the automatic database schema generator in the ER-Win creates database-specific tables for the saving the game data.

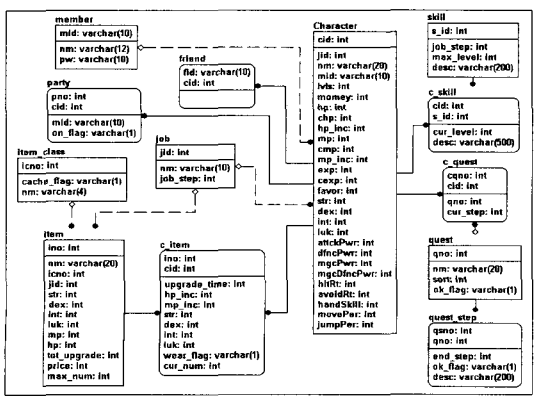


Fig. 3. Example of a ERD of a network game.

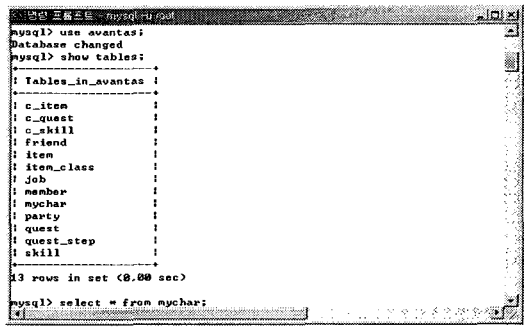


Fig. 4. Creation schema set in the MySQL.

The primary can be a normal attribute that is guaranteed to be unique. We defined a table's primary key in the CREATE TABLE statement.

2.4 Advanced Database Design

Additionally, the one of important thing that should be considered to design the database system is the interface with the legacy system. The database server of the game and the legacy system(Other MIS, Internet system etc...) should be the integrated system and it could be more efficient system once built on[15-18]. But this development method may have some defaults in cutting down expenses and saving time. In the light of the database design and the effective use of the data, the design through the careful consideration about the interface with other systems must be followed.

3. APPLY DESIGN PATTERNS TO IMPLEMENTATION OF DB SERVER

Applying UML(Unified Modeling Language) to the implementation of the complex software just like implementation of server side database APIs (Application Programming Interfaces) gives a lot of advantages[19-22]. In this chapter, we will figure out the design patterns that can be applied to the design of server side database APIs.

3.1 Apply to Façade Pattern

To record the data received from clients into the

database, many attempts have occurred to access the resource of the database from several objects that compose the server. The reason to apply the façade pattern like Fig. 5 into the implementation of database server is to minimize the dependency of the sub systems in each design steps. By applying the façade pattern to the objects, a unified interface can be made.

In general, to access the resource with database system, developers use the ODBC(Open Database Connectivity), JDBC of Java, or other APIs to access the database resource. The object related to the implementation of server side APIs that developers want to implement can directly access the particular object in SQL_Server of Microsoft. However, most of the database applications may receive the data and process it through the simple query without any detailed internal information of the database system. It is no need to know how the internal database system works.

The class that works as a unified interface role can be defined with the façade pattern. This pattern provides the most basic required interface to use the database for its users, and the internal database system can be operated like buffer pool, query plan, cash, etc...

It is impossible to use the system while understanding the application development and the design of the complex database system. It is too dependent to access the related object whenever each object of the database server application is needed. The application of the façade pattern to solve this problem can reduce the coupling with the database system.

That is, in case all the interface of the database system is open to the public, frequent method calls may happen but by providing the simple integrated interface and implementing the residues internally, developers can get some advantages to practically reduce the frequent calls between the application and the database system.

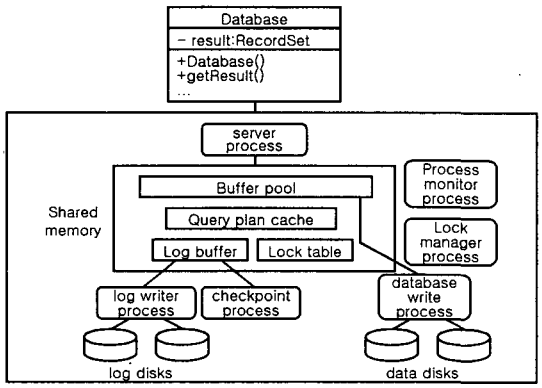


Fig. 5. Example of a Façade pattern.

3.2 Apply to Singleton Pattern in Facade Pattern

In the previous design that applies the façade pattern, if object related to the database system is one, developers should consider making the façade object as singleton. The singleton pattern application of DB manager class is able to increase cohesion. To manage the creation of the object designed with facade itself is convenient in many aspects including maintenance.

3.3 Apply to Command Pattern

The server analyzes the data and properly processes the requests sent by clients. A general method to analyze and process the client's request is to use a parameter. Generally, the code is written as follows.

```
void parseCommand() {
    int command;
    if(command != -99) {
        switch(command) {
            case 9 : login();
                     break;
            case 8 : loadCharacter();
                     break;
            case 7 : saveCharacter();
                     break;
        /* Process additional command */
            default: greeting();
        }
    }
}
```

A pattern that is used to process a job in accordance with the value of a particular variable is the command pattern. Consider the code above. We can notice the login process when the special client first logins the game server, the process to read the information related the user's game character from DB and the process to save the user's data to DB.

These methods are able to be implemented to inherit the abstract command class like login(), loadCharacterCommand(), saveCharacterCommand (). Command in Fig. 6 shows those classes that inherit the abstract command class.

Each class implements the virtual excute() method inherited from parent Command class. The code to use this command pattern is generally written as follows.

```
void parseCommand(Command *p) {
  Command *command;
  switch(p->type){
     /* Process general login*/
     case LoginCommand:
        command = new LoginCommand();
        break;
     /*load the character informations */
     case LoadCharacterCommand:
        command = new LoadCharacterCommand();
        break;
     /* save the character information */
     case SaveCharacterCommand:
        command = new SaveCharacterCommand();
        break;
  }
  Result result = command->excute(param)
}
```

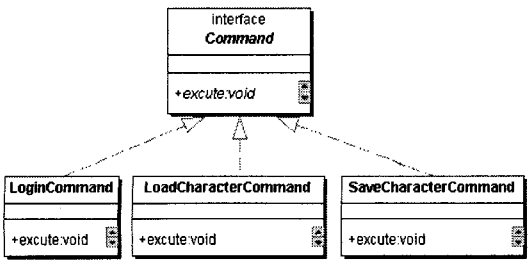


Fig. 6. Command pattern diagram to make various packets.

By using the command pattern in the business logic implementation of the server application, the object calling the operation and the object managing the operation implementation method can be separated. The advantage is that developers can combine the command and make anther command and insert the new command easily.

In case of communication with additional objects such as a User class in the figure 6, if it is not an adopted command pattern in design, there are some problems that it is necessary to modify parseCommand() method, or add to a new class related to methods that deal with the logic.

Figure 7. shows that how the command pattern is used to modify of a little to process the logic related to User object.

We recommend adding the new classes to processing user information and saving user data related to User object in the diagram. At this point, a developer has advantages that less effort is needed to be made to modify the code appending necessary class as compared with the implementation that modifies the existing parseCommand() method.

4. CONCLUSION

In this paper, we have designed the database of the game application built on to manage the work of the game effectively. We have also mentioned that the design patterns suggested by GoF can be

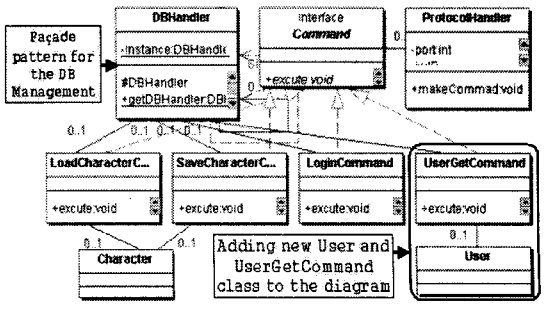


Fig. 7. Putting Facade & Command pattern together.

effectively used to the design of APIs related to the implementation of server side application. Various design patterns are applied for the game application to be effectively designed.

The façade pattern, a type of structural patterns, has the advantage of providing the consolidated interface to use database. The command pattern, a type of behavioral patterns, used to process the requests received from the clients, has the advantage that the distributed development is possible by processing the request data into the capsulated object. Furthermore, the singleton pattern applied to create the object that manages the database also showed the fact that the memory can be effectively used by guaranteeing the uniqueness of the instance.

The Application Wizard is automatically installed based on the existing Visual Studio 6.0 or Visual Studio .Net when the DirectX SDK 9.x is installed. It is used to make simple applications that provide automatically some classes related DirectX API and simple GUI.

But provided classes using DirectX API by the Application Wizard have some problems of many object variables and methods that are not needed to implement an ideal game system. The problems also happen when making DirectX components used in games. To avoid these problems in the analysis and design steps, various design patterns are studied to make game frameworks related to special games like Star Craft, Lineage, etc...

We show that such GoF design patterns are useful to the game application design. In the network game design, by using GoF design patterns, it makes easy to design related game API and to modify already written codes. The advantage is that developers can make a new function and remove the existing function and modify the function easily.

We suggest some design methods using the GoF design patterns here, They have some advantages that provide more advanced design and easily reuseable architecture than the old design methods. We have also studied and listed verified design patterns. When new game developers use the suggested design patterns more efficiently, They are able to easily maintain, do refactoring and document the already existing game system.

5. REFERENCES

[1] David H. Eberly, *3D Game Engine Designe,* Morgan Kaufmann, 2001.

[2] Mary Firestone, *Computer Game Developer (Weird Careers in Science)(Library Binding),* Chelsea House Publications, 2005.

[3] Erich Gamma, Richard Helm Ralph Johnson, Hohm Vissides, *Gof's Design Patterns,* Pearson education Korea, 2002.

[4] Charles Patzold, *Programming Windows Fifth Edition,* Microsoft Press, 1998.

[5] ©Microsoft, *Microsoft SQL Server 2000 Developer,* Microsoft, 2000.

[6] ©Sun Microsystems, *sun educational serviecs Advanced Java Programming SL-300,* SunSoft Press, 2000.

[7] Soon-Kak Kwon, Jong-Soo Kim, Tai-Suk Kim, "An Implementation Avatar Chatting System for Network-Based Multi-User Environment" EALPIIT2003 National University of Mongolia, Ulaanbaatar, Mongolia July 6-9, pp. 119-123, 2003.

[8] James Gosling, Frank Yellin, Java Team, *The Java Application Programming Interface Volume1,* Addison Wesley, 1996.

[9] Toyaki Tomura, Kiyoshi Uehiro, Satoshi Kanai, Susumu Uamamoto. "Object-Oriented Design Pattern Approach for Modeling and Simulating Open Distributed Control System" *Proceedings of the 2001 IEEE International Conference on Robotics & Automation Seoul, Korea,* May 21-26, 2001.

[10] John Lewis, William Loftus, *Java Software Solutions Foundations of Program Design,* Addison Wesley, 1998.

[11] Toyaki Tomura, Kiyoshi Uehiro, Satoshi Kanai, Susumu Uamamoto. "Object-Oriented

Design Pattern Approach for Modeling and Simulating Open Distributed Control System" *Proceedings of the 2001 IEEE International Conference on Robotics & Automation Seoul, Korea,* May 21-26, 2001.

[12] Dr. Raymond J. Toal, Robert G. Hayes. "ATS Software Design Patterns" IEEE, 2001.

[13] Craig Larman, *Applying UML and Patterns,* Prentice Hall PTR, 2001.

[14] Avi Silberschatz, Henry F. Korth. S. Sudarshan, *Database System Concepts Forth Edition,* McGraw-Hill, 2003.

[15] Jong-Soo Kim, Oh-Jun Kwon, Tai-Suk Kim, "APIs Design in the side of the Database Using Design Patterns for Online Game," *Proceeding of the Korea Multimedia Society Spring Conference, 2005,* Vol. 8, No.1, pp. 847-850, May 2005.

[16] Jong-Soo Kim, Tai-Suk Kim, "A Study on a Database Design for the Multi-User Online Game," *Proceeding of the Korea Information Processing Society Spring Conference, 2005,* Vol. 12, No. 1, pp. 361-364, May 2005.

[17] Jong-Soo Kim, Tai-Suk Kim, "The Study of the APIs Design in the Internet Application to Construct a Database Server," *IEEE-HealthCom2005 7th International Workshop,* pp. 335-338, June 2005.

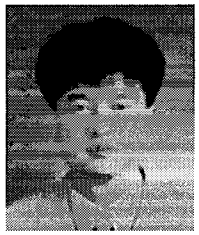
[18] Jong-Soo Kim, Tai-Suk Kim, Oh-Jun Kwon, "The APIs Design for the Database Management of the Network Game Using Design Patterns," *Journal of Korea Multimedia Society,* Vol. 9, No. 1, July 2006.

[19] Jong-Soo Kim, Tai-Suk Kim, "The Creational Patterns application to the Game Design using the DirectX," *Journal of Korea Multimedia Society,* Vol. 8, No. 4, pp. 536-543, Feb. 2004.

[20] Jong-Soo Kim, Tai-Suk Kim, "A Study on the Effective Class Composition for the Various Game Character Design," *Proceeding of the Korea Information Processing Society Fall Conference, 2005,* Vol. 9, No. 1, pp. 313-316, Nov. 2005.

[21] Jong-Soo Kim, Tai-Suk Kim, "a Study of the Template Method for Various Behavior of the Game Characters," *Proceeding of the Korea Multimedia Society Fall Conference, 2005,* Vol. 8, No. 2, pp. 69-72, Nov. 2005.

[22] M. McNaughton, M. Cutumisu D. Szafron and J. Schaeffer, J. Reford, D. Parker. "SciptEase: Generative Design Patterns for Computing Role-Playing Games" *Proceedings of the 19th International Conference on Automated Software Engineering(ASE'04),* 2004.



Jong-Soo Kim

He received his B.S. degree from Pukyong National University in 1992, his M.S. degree from the department of Computer Engineering, Busan University of Foreign Studies in 2003, and his Ph.D. degree from the department of Software Engineering, Dong-eui University in 2006. He has worked at the Institute of Telecommunications Information in Dong-eui University as a post-doc. His current research interests are network game design and web applications.

Tai-Suk Kim

received the B.S. degree in Electronic Engineering form Kyungpook National University, Korea, in 1981 and the M.S. and ph.D. degree in Computer Science from KEIO University, Japan, in 1989 and 1993, respectively. Since 1994, he has been a faculty member of the Dongeui University, where he is now Professor in department of Computer software engineering. His reserch field has been in information system, internet business, network game and NLP.