JOE Game CD DataBase

Database Lab Project implementation Report

Prof. Jeonyoung Lee

TEAM : JOE

Computer Science & Engineering
9725002 Joon-Myung Kang
9725018 Taeho Oh
9725029 Hyunho Jung

email : joe@ohhara.postech.ac.kr

1999년 6월 11일

목 차

요 약

CS421 File and DataBase 시간에 우리는 이미 몇가지 상용 RDBMS(Relational DataBase Management System)에 대하여 배웠다. 그리고 그것들을 직접 실습 시간에 다루어 보고 그 기능들을 익혔다. 이제 이것들을 바탕으로 해서 저희 조에서는 Game CD를 관리하기 위한 DataBase를 구축하였다. 다음에서는 이것을 구축하기 위한 여러 가지 내용들과 저희 조원들이 한 성과들, 한 일들, 일정들을 소개해 두었습니다. 그리고 끝으로는 앞으로 이 프로그램을 어떠한 방향으로 더 발전시킬 것인가에 대한 내용을 같이 언급해 두었습니다.

제 1 절 Introduction

수업 시간에 배운 다양한 이론들을 직접 응용하여 하나의 프로그램을 만들어 봄으로써 그 관련 이론들을 좀 더 깊이 아는 것을 실습의 목적이 있다. DataBase라는 큰 이론을 배우면서 이것을 응용할 수 있는 분야는 정말로 무궁무진하다. 우리 조는 이런 이론들을 바탕으로 하여 요즈음 계속 그 양이 증가하고 있는 Game CD를 domain으로 잡고, 이것을 효율적으로 관리할 수 있는 프로그램을 개발했습니다. Java, Servlet을 사용하여 웹에서 쉽게 접할 수 있도록 interface를 구축하였고, 직접 Oracle 을 리눅스에 설치하여서 DBMS로 사용하였다.

1.1 절 Domain

Game CD

1.2 절 Domain Information

The computer hardware is improved very rapidly because the computer game market increases. And computer game is not made by not only the computer programmer but also computer graphic designer, music composer , scenario writer and so on. The games made one day are very various. Therefore the game users have difficulties in selecting the game which the users want to play. In the past, computer games were very simple and doesn't need special information. However, the computer games are very complicated and need the information about the scenario, sound, demo, and so on.

1.3 절 Objective of JOE DB

우리 조의 최종적인 목적은 우리 프로그램이 Game CD 공급자와 사용자 사이에서 효울적으로 거래가 이루어지도록 데이타를 관리하도록 하는 것이다. 이것을 위해서 Game, Company, Supplier, Price, Genre별 search가 가능하게 하였고, 각 Game들마다의 Quality, Requirement등도 볼 수 있다. 이렇게 하여서 사용자는 각 Game들을 사게 된다. 물론 이러한 사용자에 대한 DB도 따로 구축하여서 관리를 하고 있다. 그래서 우리 프로그램은 사용자 공급자 모두에게 유용하게 쓰일 수 있는 완벽한 프로그램이다.

제 2 절 MileStone

그림 1을 보시면 각 조원들이 한 일들과 일정들을 보실 수 있습니다. 일들은 대체적으로 잘 이루어 졌고, 조원들의 단합도 잘 되어서 성공적으로 프로젝트를 마칠

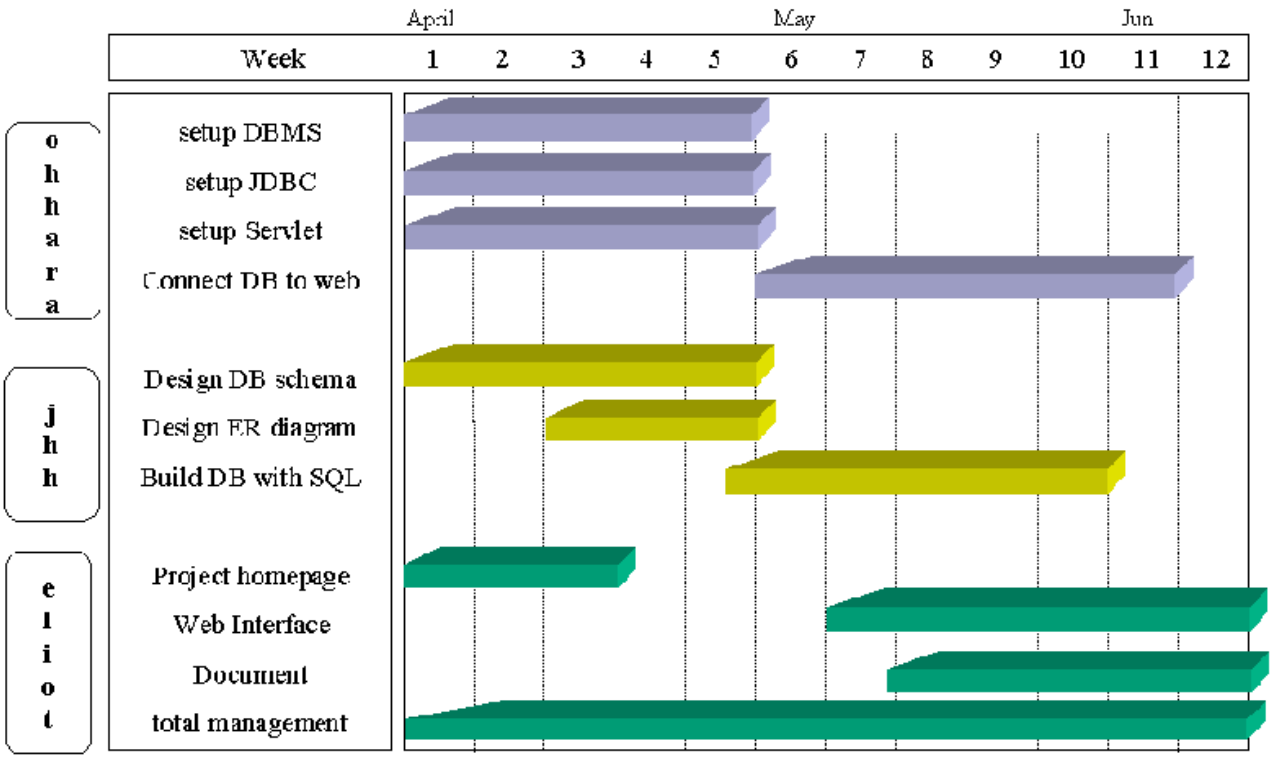

그림 1: MileStone

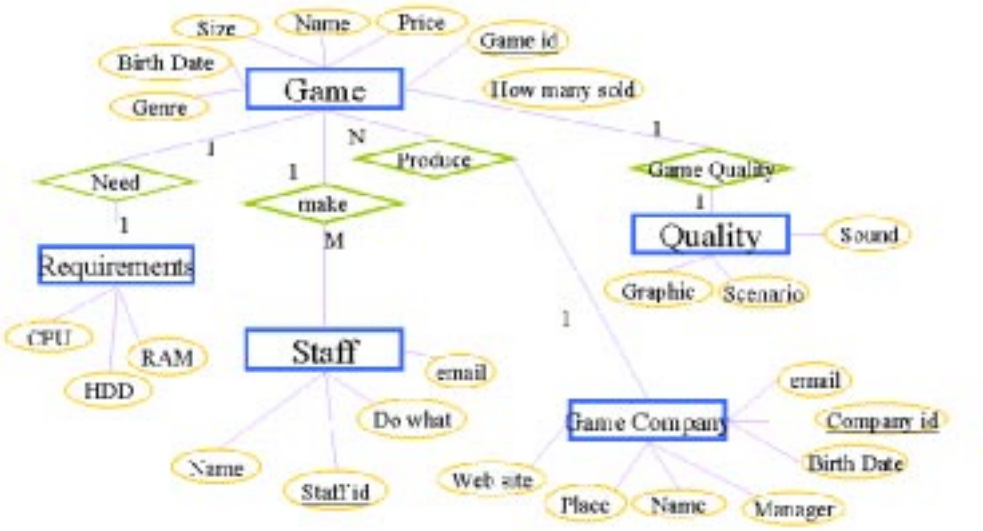

그림 2: ER Diagram

수 있었습니다.

제 3 절 ER Diagram

그림 2, 그림 3를 참조하십시오.

제 4 절 User Requirement

User can have difficulties in finding a game that the user wants to play. The early game market system can't support the user requirements completely. The user may want to find the game that the user likes even if others don't like. It's very difficult to find for the user himself(or herself). When there is a database that stores many data of game, game company and so on, the user can find the game more easily. However, there must be a pretty interface to query the database. The user may want to access the game database with World Wide Web(WWW), which is the most famous internet service. With DB, User can find the most famous game in these days. And when user asks the databse what is the company that made the most famous game. In addition, the database also stores the evaluation of the game. Therefore, if the user is only interested in the scenario quality of role playing game, the user can find such a game that user want by asking the database.

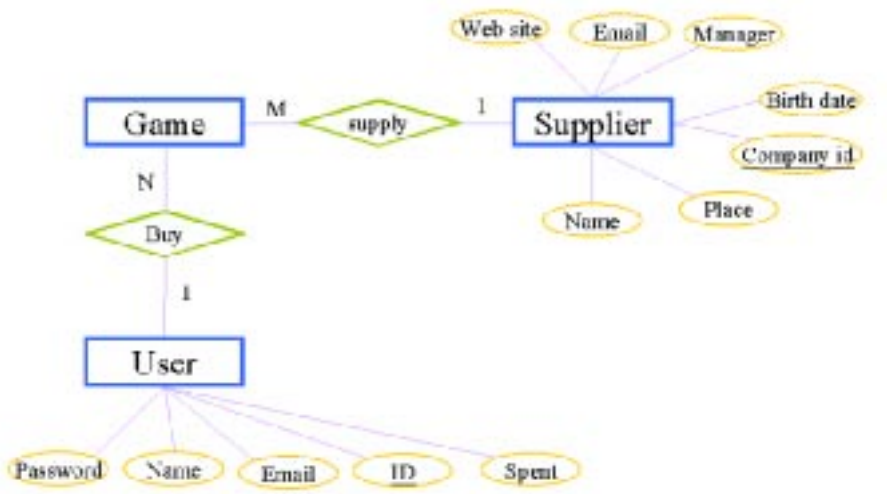

그림 3: ER Diagram2

User

Register ID	X
Login (ID, Password Check	O
Search games with keyword(game name, genre and so on	O
Search games with option(case sensitive, case insensitive)	O
View game list, quality and requirement	O
View staff list, company list, supplier list and user list	O
View game list user bought	O
Buy games	O
View and update user's own profile	O
Discount game if the user buy many games	O
Get an email if the new game is inserted	O

Admin

Can do what the user can do	O
View, insert, delete and update the company	O
View, insert, delete and update the supplier	O
View, insert, delete and update the user	O
View, insert, delete and update the game	O
View, insert, delete and update the staff	O
View, insert, delete and update the quality	O
View, insert, delete and update the requirement	O

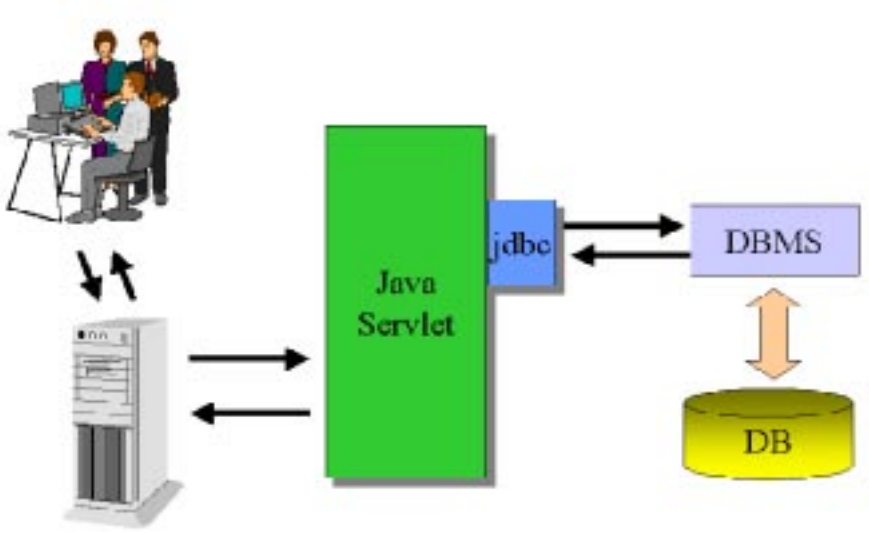

그림 4: System Architecture

(O:implemented, X:not implemented)

제 5 절 System Architecture

See the fig 4. The basic system architecture is composed with database, jdbc, servlet, jserv web server and web browser. The query of user goes like this.

web browser R web server R jserv R servlet R jdbc R database

The query of web browser is sent to web server with POST or GET method like CGI. The web server passes the query string to servlet with apache JServ module. The servlet sends the query of the web browser to database with jdbc. And the result of the query goes like this.

database R jdbc R servlet R jserv R web server R web browser

The result data of the query is sent to servlet with jdbc. After that, the servlet send the data to web server with jserv. Finally, the web browser shows the result of the query.

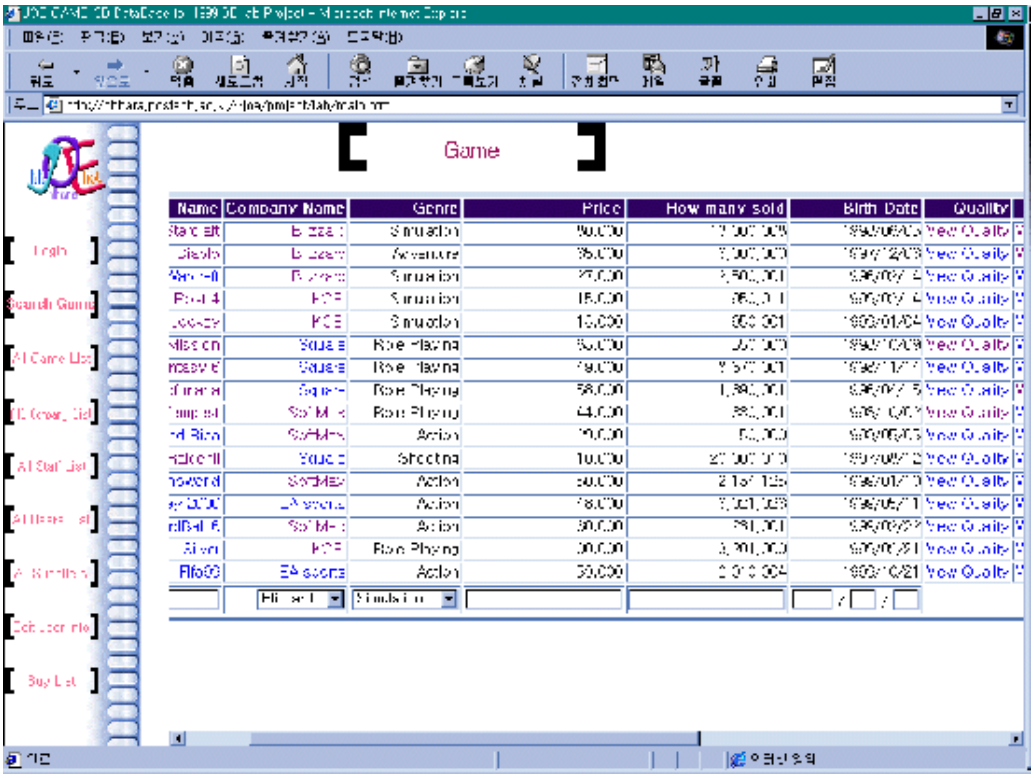

그림 5: Interface of JOE DB

제 6 절 Target DataBase

Oracle8 Server, Release 8.0.5 for linux

제 7 절 Connectivity

Oracle JDBC Drivers release 8.0.5 for linux
Java Development Kit (JDK) 1.2 for linux
Java Servlet Development Kit (JSDK) 2.0 for java

제 8 절 Interface

World Wide Web(WWW). See the fig 5

제 9 절 DB Schema

```
CREATE SEQUENCE joe_game_id
INCREMENT BY 1
START WITH 1
MAXVALUE 999999999999
```

```
NOCACHE
NOCYCLE
;

CREATE SEQUENCE joe_staff_id
INCREMENT BY 1
START WITH 1
MAXVALUE 999999999999
NOCACHE
NOCYCLE
;

CREATE SEQUENCE joe_company_id
INCREMENT BY 1
START WITH 1
MAXVALUE 999999999999
NOCACHE
NOCYCLE
;

CREATE TABLE joe_company
(
companyid NUMBER(12),
name CHAR(50) NOT NULL,
place CHAR(255),
website CHAR(255),
manager CHAR(50),
email CHAR(255),
birthdate DATE,
PRIMARY KEY (companyid)
)
;

CREATE TABLE joe_supplier
(
supplyid NUMBER(12),
name CHAR(50) NOT NULL,
place CHAR(255),
website CHAR(255),
manager CHAR(50),
email CHAR(255),
birthdate DATE,
PRIMARY KEY (supplyid)
)
;

CREATE TABLE joe_game
```

```
(
gameid NUMBER(12),
companyid NUMBER(12),
supplyid NUMBER(12),
name CHAR(50) NOT NULL,
genre CHAR(50),
price NUMBER(20),
birthdate DATE,
howmanysold NUMBER(20),
PRIMARY KEY (gameid),
FOREIGN KEY (companyid) REFERENCES joe_company ON DELETE CASCADE,
FOREIGN KEY (supplyid) REFERENCES joe_supplier ON DELETE CASCADE
)
;

CREATE TABLE joe_staff
(
staffid NUMBER(12),
gameid NUMBER(12),
name CHAR(50) NOT NULL,
email CHAR(255),
dowhat CHAR(50) NOT NULL,
PRIMARY KEY (staffid),
FOREIGN KEY (gameid) REFERENCES joe_game ON DELETE CASCADE
)
;

CREATE TABLE joe_requirement
(
gameid NUMBER(12),
cpu CHAR(50),
hdd NUMBER(20),
ram NUMBER(20),
PRIMARY KEY (gameid),
FOREIGN KEY (gameid) REFERENCES joe_game ON DELETE CASCADE
)
;

CREATE TABLE joe_quality
(
gameid NUMBER(12),
graphic NUMBER(1),
sound NUMBER(1),
scenario NUMBER(1),
PRIMARY KEY (gameid),
FOREIGN KEY (gameid) REFERENCES joe_game ON DELETE CASCADE
)
```

```
;

CREATE TABLE joe_user
(
password CHAR(10),
name CHAR(50),
email CHAR(255),
id CHAR(10),
spent NUMBER(20),
PRIMARY KEY (id)
)
;

CREATE TABLE joe_buy
(
id CHAR(10),
gameid NUMBER(12),
FOREIGN KEY (id) REFERENCES joe_user ON DELETE CASCADE,
FOREIGN KEY (gameid) REFERENCES joe_game ON DELETE CASCADE
)
;
```

Joe_game_id	The sequence number which decide the new game id
Joe_staff_id	The sequence number which decide the new staff id
Joe_company_id	The sequence number which decide the new company id
Joe_company	The company table
Joe_supplier	The supplier table
Joe_game	The game table which needs company and supplier table
Joe_staff	The staff table which needs game table
Joe_requirement	The requirement table which needs game table
Joe_quality	The quality table which needs game table
Joe_user	The user table
Joe_buy	All games that the users bought

제 10 절 Authentication

User authentication is very simple. When the user input ID and password, the Login.class servlet check the ID and password in the $joe_usertable. If the correct ID and password, the Login.cla$

제 11 절 Buy the game

The user can buy the game. If the user spent more than 100000 won to buy games, he(or she) can buy the game at 10% discount. More than 300000 won, 20% discount and more than 500000 won, 30% discount. When the user buy a game, the following is calculated and update the database.

```
joe_user.spent = joe_user.spent + joe_game.price * ratio
( ratio is determined by discount )
joe_game.howmanysold = joe_game.howmanysold + 1
```

제 12 절 The discription of the servlet source

BuyGame.java	Update DB when the user buy a game
CookieUser.java	Get the current user in the cookie
DBQuery.java	Connect to DB and update and query DB
DeleteCompany.java	Delete company
DeleteGame.java	Delete game
DeleteStaff.java	Delete staff
DeleteSupplier.java	Delete supplier
DeleteUser.java	Delete user
ExecuteUpdateCompany.java	Update company information
ExecuteUpdateGame.java	Update game information
ExecuteUpdateStaff.java	Update staff information
ExecuteUpdateSupplier.java	Update supplier information
ExecuteUpdateUser.java	Update user information
InsertCompany.java	Insert new company
InsertGame.java	Insert new game(need company and supplier)
InsertStaff.java	Insert new staff(need game)
InsertSupplier.java	Insert new supplier
InsertUser.java	Insert new user
JoeEnv.java	Store the DB environment variables(ex. the admin account)
ListBuy.java	List the game which the user bought
ListCompany.java	List company
ListGame.java	List game
ListQuality.java	List quality
ListRequirement.java	List requirement
ListStaff.java	List staff
ListSupplier.java	List supplier
ListUser.java	List user
Login.java	Set the user id cookie if the user input the correct ID and passwd
UpdateCompany.java	Dispaly update company form
UpdateGame.java	Dispaly update game form
UpdateQuality.java	Dispaly update quality form
UpdateRequirement.java	Dispaly update requirement form
UpdateStaff.java	Dispaly update staff form
UpdateSupplier.java	Dispaly update supplier form
UpdateUserinfo.java	Dispaly update current user form

제 13 절 Conclusion

13.1 절 앞으로 발전 방향

이 프로그램의 발전 방향은 무수하다. 우선 지금은 간단하게 각각의 데이타를 넣고 관리하는 것이지만, 이것이 여러개의 DB와 연동된다면, 전세계의 Game CD를 관리하는 DB를 만들 수 있고, 더 나아가서는 각종 쇼핑몰들에 기반을 둘 수도 있다. 특히 사용자들이 친숙한 인터페이스로 만들어져서 사용하기가 아주 편리하기 때문에 더욱 발전 방향성이 높다.
그리고 현재 생각하고 더 추가하고 싶었는데, 추가하지 못한 것들은 AI(인공지능) 기법 중 Knowlege Based DB를 구축하여 각 사용자들의 성향을 분석한 뒤에 그것들을 토대로 Reasoning을 하여서 각 사용자에게 적당한 게임을 추천하는 부분도 추가했으면, 더욱 완벽한 프로그램이 될 것이라 믿는다.
그리고 이번 프로그램을 하면서 수많은 데이타를 관리하는데 역시 DB가 많이 유용하다는 것을 느낄 수 있었다. 특히 요즈음은 많은 정보를 받아 들이지만, 그것들을 효율적으로 정리하지 못해서 고민이었는데, 이번에 DB를 배우고 나서 그런 것들에 대한 DB를 직접 만들어서 관리해 보기도 하였다.

13.2 절 프로그램 하면서 느꼈던 점들

이번 프로젝트를 수행하면서 느꼈던 점들과 그리고 저희가 사용한 각 프로그램들을 왜 사용했는지 어떻게 사용했는지 등에 대해서 다음에서 설명을 할 것입니다.

13.2.1 java servlet을 사용한 이유

java servlet은 CGI와 같이 한번 실행될때마다 fork가 되지 않고 multithreaded하게 실행되기 때문에 CGI에 비해 훨씬 빠르게 수행된다. 일부 사람들은 java는 속도가 느리다고 흔히 말한다. 그것은 사실이다. 아주 느리다. 하지만 Database를 Web과 연동시킬 때 가장 큰 문제가 한번 DB를 access를 할 일이 생길때마다 login부터 시작해서 처음부터 다시 작업을 해야 된다는 것이다. 물론 따로 server같은 것을 만들어서 그런 문제를 어느정도 줄일 수는 있지만 그것은 미봉책이다. 하지만 java servlet은 그런 문제를 해결했다. Access를 한번 하면 한동안 그 thread가 지워지지 않고 살아있다. 그리고 다시 누군가가 access를 하면 새로 thread를 만들거나 process를 fork하지 않고 그 thread를 사용한다. 물론 일정시간동안 아무도 access를 하지 않으면 해당 thread는 사라진다. 그래서 처음에 접속해 사용할 때는 상당히 느린 것을 느낄 수 있다. 많은 사람들이 접속해서 사용하게 되면 Database에 접속하면서 속도가 느려지는 문제를 완전히 해결할 수 있다.

또한 java를 한번이라도 써본 사람은 java servlet을 아주 쉽게 만들 수 있다. 문법도 아주 쉽다. CGI를 사용하기 위해서는 text 처리를 위해 자신이 하나하나 처리를 하거나 아니면 라이브러리를 가져다가 사용해야되는 불편함이 있는데 비해 java servlet은 그런 문제가 없다.

13.2.2 어려웠던 점

java servlet은 debugging하기가 가장 힘들었다. java servlet프로그래밍을 많이 해보지 않아서 어려웠던 것으로 생각된다. 보통 java program은 실행시키면서 중간중간의 상태를 standard output으로 내보내면서 debugging을 했는데 이것은 그럴

수가 없었다. 그래서 html comment형식(<!– comment –>)으로 중간중간의 상태를 comment로 print하도록 해서 debugging을 했다. 하지만 중간에 예기치않은 exception을 만나면 (ex. Null pointer exception) 아무런 소리 없이 프로그램이 죽어버려서 어디서 뭐가 잘못되었는지 찾기가 쉽지 않았다.

java servlet을 쓰는 사람을 주변에서 잘 보지를 못해서 환경설정하는 것이 어려웠다. apache web server에 servlet을 지원해 주는 module을 설치하고 sun에서 제공하는 java servlet development kit을 설치해서 설정해 servlet을 쓸 수 있는 환경을 만들었다. 그리고 아직 module이 불안정해서 가끔 메모리를 무한대로 차지해서 시스템이 사용불능상태에 빠지기도 했다.

oracle을 linux에 설치하는 것이 쉽지 않았다. 주변 사람들에게 알아보니 oracle을 설치시도하다가 전부 포기한 사람밖에 없었다. 그래서 oracle을 설치하는 방법을 알아내는데 시간이 생각보다 시간이 많이 걸렸다.

13.2.3 개선점

사용자 인터페이스가 약간 불편하다. 어느 정보를 얻기 위해서는 두세단계를 거쳐야 되는 경우도 가끔 있는데 이런 것을 개선해야 된다.

현재 사용자 인증방식이 보안성을 전혀 고려하지 않고 구현되었다. Database안에는 password가 plain text로 저장되어 있다. 그리고 사용자가 password를 입력할 때는 encryption을 전혀 하지 않고 주고받도록 되어 있다. 보안성을 강화시키기 위해서는 SSL을 사용하고 Database안에는 password를 encrypt해서 저장을 해야 된다.

사용자가 어떤 게임을 살 때 현재 그 사용자가 돈을 어느정도 가지고 있는지에 대한 검사가 전혀 없다. 즉, 돈이 전혀 없어도 물건은 무한대로 계속 살 수 있다. 사용자가 돈을 얼마나 가지고 있는지에 대한 검사하는 방법이 필요하다.

공급자가 게임을 얼마나 공급했고, 얼마에 공급했는지에 대한 정보가 저장되지 않는다. 이것도 구현해야 한다.

처음 오는 사용자가 ID를 추가시키고 싶으면 관리자에게 편지를 보내야 한다. 사용자가 직접 자신의 ID를 추가시킬 수 있는 기능을 넣어야 한다.

사용자가 어떤 종류의 게임을 많이 샀을 때 그 사용자의 성향을 분석하는 기능이 없다. 이것을 구현하기 위해서는 AI기법이나 통계적인 기법이 사용되야 할 것으로 보인다.