Databases (INF.01014UF) – Report

Hojun, Cho (11739592)

Dayeong, Lee (11739321)

Table of Content

1. Motivation

Our application is for managing the airplane tickets of clients. When someone plans to travel, he has to buy airplane tickets and remember the information such as the airline, the departure time and so on. If he wants to check his airplane ticket information, he should find the boarding pass, of which format is pdf. Then, it takes lots of time to find a boarding pass and open it. Therefore, our app helps clients to manage the airplane tickets by showing all tickets what he bought. That is, clients can manage their airplane tickets by adding, modifying and deleting their airplane tickets through our app.

2. Database schema

ADomain :

Name	Type	Description
CID	Int	The client ID
LID	Int	The airline ID
TDTime	Datetime	The departure time
TATime	Datetime	The arrival time
PDID	Int	The departure airport ID number
PAID	Int	The arrival airport ID nubmer
TMileage	Int	The Mileage of the airticket
TSeat	Int	The seat
PID	Int	The airport ID number
Pname	String	The airport name
PCity	String	The airport location city
LID	Int	The airline ID number
LName	String	The airline name
LAlliance	String	The air Alliance of airline
LNation	String	The nationality of airline
CName	String	The name of client
CAge	Int	The age of client
CGender	Boolean	The gender of client
CNation	String	The nationality of client

*** The domain of PID, PDID and PAID are same. ***

- **Relation : airticket** (<u>CID</u>, <u>LID</u>, <u>TDTime</u>, TATime, TMileage, TSeat, PDID, PAID)
- **Relation : airline** (<u>LID</u>, LName, LAlliance, LNation)
- **Relation : airport** (<u>PID</u>, PName, PCity)
- **Relation : client** (<u>CID</u>, CName, CAge, CGender, CNation)

3. Functional Dependencies

- **Relation : airticket** (<u>CID</u>, <u>LID</u>, <u>TDTime</u>, TATime, TMileage, TSeat, PDID, PAID)
  (CID, LID, TDTIme) → TATime
  (CID, LID, TDTIme) → TMileage
  (CID, LID, TDTIme) → TSeat
  (CID, LID, TDTIme) → TDID
  (CID, LID, TDTIme) → TAID

- **Relation : airline** (<u>LID</u>, LName, LAlliance, LNation)
  LID → LName
  LID → LAlliance
  LID → LNation

- **Relation : airport** (<u>PID</u>, PName, PCity)
  PID → PName
  PID → PCity

- **Relation : client** (<u>CID</u>, CName, CAge, CGender, CNation)
  CID → CName
  CID → CAge
  CID → CGender
  CID → CNation

AirTicketDB is in the 3rd normal form:

⇨ To satisfy the 3rd normal form, non-primary keys should be dependent only on the primary key set, not on the subset of the primary key. That is, there should be no dependency between non-primary keys. Also, it should be in the the 2nd normal form.

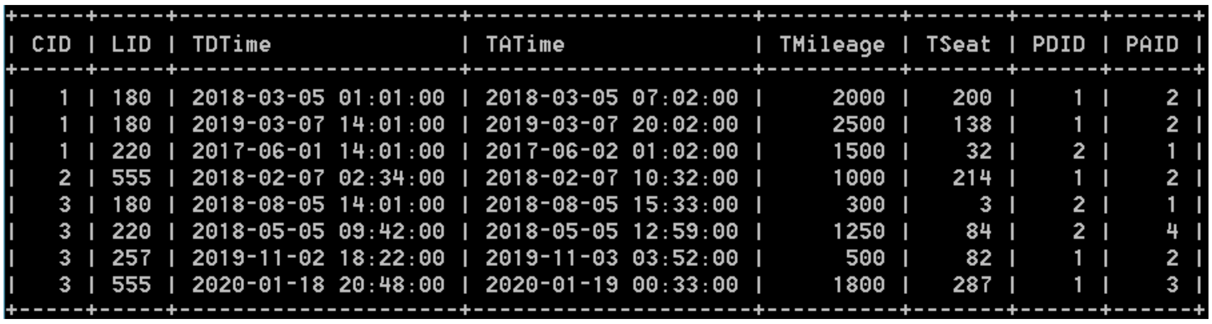
⇨ In the case of <u>client</u>, non-primary keys are CName, CAge, CGender, CNation. They are dependent only on CName. Also, there is no dependency between them.

⇨ In the case of airticket, airline and airport, all of them satisfies the condition of 3rd normal form like the case of client.

⇨ Therefore, Our DB, AirTicketDB, satisfies the condition of the 3rd normal form.

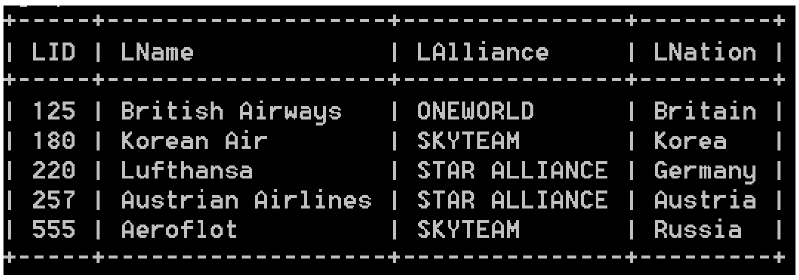
4. Current state of the database

<Airticket>



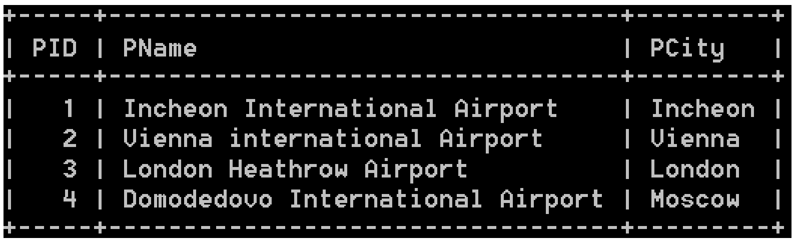
```
+-----+-----+---------------------+---------------------+----------+-------+------+------+
| CID | LID | TDTime              | TATime              | TMileage | TSeat | PDID | PAID |
+-----+-----+---------------------+---------------------+----------+-------+------+------+
|   1 | 180 | 2018-03-05 01:01:00 | 2018-03-05 07:02:00 |     2000 |   200 |    1 |    2 |
|   1 | 180 | 2019-03-07 14:01:00 | 2019-03-07 20:02:00 |     2500 |   138 |    1 |    2 |
|   1 | 220 | 2017-06-01 14:01:00 | 2017-06-02 01:02:00 |     1500 |    32 |    2 |    1 |
|   2 | 555 | 2018-02-07 02:34:00 | 2018-02-07 10:32:00 |     1000 |   214 |    1 |    2 |
|   3 | 180 | 2018-08-05 14:01:00 | 2018-08-05 15:33:00 |      300 |     3 |    2 |    1 |
|   3 | 220 | 2018-05-05 09:42:00 | 2018-05-05 12:59:00 |     1250 |    84 |    2 |    4 |
|   3 | 257 | 2019-11-02 18:22:00 | 2019-11-03 03:52:00 |      500 |    82 |    1 |    2 |
|   3 | 555 | 2020-01-18 20:48:00 | 2020-01-19 00:33:00 |     1800 |   287 |    1 |    3 |
+-----+-----+---------------------+---------------------+----------+-------+------+------+
```

<Airline>



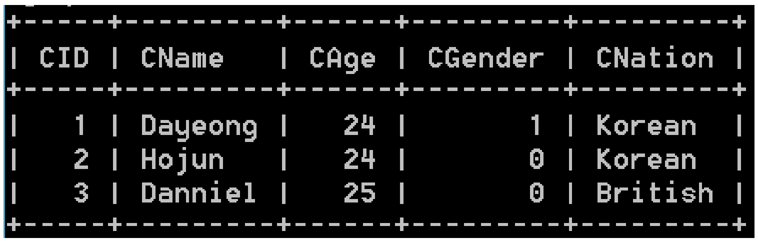
```
+-----+-------------------+---------------+---------+
| LID | LName             | LAlliance     | LNation |
+-----+-------------------+---------------+---------+
| 125 | British Airways   | ONEWORLD      | Britain |
| 180 | Korean Air        | SKYTEAM       | Korea   |
| 220 | Lufthansa         | STAR ALLIANCE | Germany |
| 257 | Austrian Airlines | STAR ALLIANCE | Austria |
| 555 | Aeroflot          | SKYTEAM       | Russia  |
+-----+-------------------+---------------+---------+
```

<Airport>



```
+-----+-----------------------------------+---------+
| PID | PName                             | PCity   |
+-----+-----------------------------------+---------+
|   1 | Incheon International Airport     | Incheon |
|   2 | Vienna international Airport      | Vienna  |
|   3 | London Heathrow Airport           | London  |
|   4 | Domodedovo International Airport  | Moscow  |
+-----+-----------------------------------+---------+
```

<Client>



```
+-----+---------+------+---------+---------+
| CID | CName   | CAge | CGender | CNation |
+-----+---------+------+---------+---------+
|   1 | Dayeong |   24 |       1 | Korean  |
|   2 | Hojun   |   24 |       0 | Korean  |
|   3 | Danniel |   25 |       0 | British |
+-----+---------+------+---------+---------+
```

5. Database queries in terms of the Relational Algebra

Get the the locations of airport, which the clients, whose name are 'Hojun', visit.

Select client Where CName = 'Hojun' Giving A;

Join A and airticket over CID Giving B;

Join B and airport over airport.PID = B.PDID Giving D;

Join B and airport over airport.PID = B.PAID Giving E;

D Union E Giving F;

Project F over PCity Giving Result;

Get the names of client, who have ever used both of airline 'Korean air' and 'Japan air'.

Select airline Where LName = 'korean air' Giving A;

Join A and airticket over LID Giving B;

Select airline Where LName = 'Japan air' Giving C;

Join C and airticket over LID Giving D;

B Intersect D Giving E;

Join E and client over CID Giving F;

Project F over CName Giving Result;

Get the names of client, who have ever arrived at the airport 'Vienna' and have not ever arrived at the airport 'Graz'

Select airport Where PName = 'Vienna' Giving A;

Join A and airticket Over A.PID = airticket.PAID Giving B;

Select airport Where PName = 'Graz' Givng C;

Join C and airticket Over C.PID = airticket.PAID Giving D;

B Minus D Giving E;

Join E and client over CID Giving F;

Project F Over CName Giving Result;

6. Database queries in terms of the Relational Calculus

Get the seat number of the ticket, of which the client is 'hojun', departure time is '2018-07-07' and airline is 'korean air'.

T -> ticket

C -> client

L -> airline

(T.TSeat) : (T.CID = C.CID & T.TDTime = '2018-07-07' & T.LID = L.LID & C.CName = 'hojun' & L.LName = 'korean air')

Get the lowest Mileage of the ticket, of which the owner is 'hojun' and the airline is 'korean air'

T1 -> airticket

T2 -> airticket

C -> client

L -> airline

(T1.TMileage) : ∀T2 ∃C ∃L (C.CName = 'hojun' & L.LName = 'korean air' & T1.CID = C.CID & T1.LID = L.LID & T2.CID = C.CID & T2.LID = L.LID & T1.TMileage <= T2.TMileage)

Get the highest seat number of the ticket, of which the departure airport is 'Vienna', the arrival airport is 'Graz' and the airline is 'LOT'.

T1 -> airticket

T2 -> airticket

P1 -> airport

P2 -> airport

L -> airline

(T1.TSeat) : ∀T2 ∃P1 ∃P2 ∃L (T1.PDID = P1.PID & T2.PDID = P1.PID & T1.PAID = P2.PID & T2.PAID = P2.PID & T1.LID = L.LID & T2.LID = L.LID & P1.PName = 'Vienna' & P2.PName = 'Graz' & L.LName = 'LOT' & T1.TSeat >= T2.TSeat)

7. Database queries in terms of the SQL

Get the client names, who has a ticket of which Korean air mileage is 4000.

```
Select CName from client where CID IN
     ( Select CID from airticket where LID IN
                ( Select LID from airline where LName = 'Korean Air')
     AND TMileage =4000);
```

Get the client names, whose 'SKYTEAM' Alliance mileage sum is over 5000;

```
Select CName from client where CID IN
     ( Select A.CID from
                ( Select * from airticket where LID IN
                          ( Select LID from airline where LAlliance = 'SKYTEAM')
                    ) A GROUP BY CID having sum(TMileage) >=5000
     );
```

Get the number of male and female, who arrive at the airport 'Incheon International Airport'.

```
Select CGender, count(CID) from client where CID IN
            ( Select CID from airticket where PDID IN
                        ( Select PID from airport
                                    where PName = 'Incheon International Airport')
            )GROUP BY CGender;
```

8. Database queries in terms of the SQL without nested SQL Blocks

Get the client names, whose Korean air mileage is 4000.

```
Select CName from airline, airticket, client where
    client.CID = airticket.CID AND airticket.LID = airline.LID
    AND airline.LName = 'Korean Air' AND airticket.TMileage = 4000;
```

Get the client names, whose ‘SKYTEAM’ Alliance mileage sum is over 5000;

```
Select CName from airline, airticket, client where
    airline.LAlliance = 'SKYTEAM' AND airticket.LID = airline.LID
    AND airticket.CID = client.CID
            GROUP BY client.CID having sum(TMileage) >= 5000;
```

Get the number of male and female, who arrive at the airport 'Incheon International Airport’.

```
Select CGender, count(DISTINCT client.CID)
    from airticket, airport, client where
            airticket.PDID = airport.PID AND airticket.CID = client.CID
            AND airport.PName = 'Incheon International Airport'
                    GROUP BY CGender;
```

9. Practical implementation of the database with SQL

```
CREATE DATABASE 11739592_airticketDB;

USE 11739592_airticketDB;


# create all relations

CREATE TABLE airticket(
            CID INT NOT NULL,
            LID INT NOT NULL,
            TDTime DATETIME NOT NULL,
            TATime DATETIME NOT NULL,
            TMileage INT NOT NULL,
            TSeat INT NOT NULL,
            PDID INT NOT NULL,
            PAID INT NOT NULL,

            primary key (CID,LID,TDTime),
            foreign key (CID) references client(CID) ON DELETE CASCADE,
            foreign key (LID) references airline(LID) ON DELETE RESTRICT,
            foreign key (PDID) references airport(PID) ON DELETE RESTRICT,
            foreign key (PAID) references airport(PID) ON DELETE RESTRICT
);


CREATE TABLE client(
            CID INT NOT NULL primary key,
            CName VARCHAR(30) NOT NULL,
            CAge INT NOT NULL,
            CGender BOOLEAN NOT NULL,
            CNation VARCHAR(30) NOT NULL
);


CREATE TABLE airport(
            PID INT NOT NULL primary key,
            PName VARCHAR(50) NOT NULL,
            PCity VARCHAR(20) NOT NULL
);


CREATE TABLE airline(
            LID INT NOT NULL primary key,
            LName VARCHAR(30) NOT NULL,
            LAlliance VARCHAR(30) NOT NULL,
            LNation VARCHAR(30) NOT NULL
);
```

```
# insert the content into the relations
INSERT INTO airticket(CID, LID, TDTime, TATime, TMileage, TSeat, PDID, PAID)
VALUES
        (1, 180, "2018-03-05 01:01:00", "2018-03-05 07:02:00", 2000, 200, 1, 2),
        (1, 180, "2019-03-07 14:01:00", "2019-03-07 20:02:00", 2500, 138, 1, 2),
        (1, 220, "2017-06-01 14:01:00", "2017-06-02 01:02:00", 1500, 32, 2, 1),
        (2, 555, "2018-02-07 02:34:00", "2018-02-07 10:32:00", 1000, 214, 1, 2),
        (3, 257, "2019-11-02 18:22:00", "2019-11-03 03:52:00", 500, 82, 1, 2),
        (3, 180, "2018-08-05 14:01:00", "2018-08-05 15:33:00", 300, 3, 2, 1),
        (3, 555, "2020-01-18 20:48:00", "2020-01-19 00:33:00", 1800, 287, 1, 3),
        (3, 220, "2018-05-05 09:42:00", "2018-05-05 12:59:00", 1250, 84, 2, 4);


INSERT INTO client(CID, CName, CAge, CGender, CNation)
    VALUES (1, "Dayeong", 24, TRUE, "Korean"),
    (2, "Hojun", 24, FALSE, "Korean"),
    (3, "Danniel", 25, FALSE, "British");


INSERT INTO airport(PID, PName, PCity)
    VALUES (1, "Incheon International Airport", "Incheon"),
    (2, "Vienna international Airport", "Vienna"),
    (3, "London Heathrow Airport", "London"),
    (4, "Domodedovo International Airport", "Moscow");
```

```
INSERT INTO airline(LID, LName, LAlliance, LNation)
        VALUES (180, "Korean Air", "SKYTEAM", "Korea"),
        (220, "Lufthansa", "STAR ALLIANCE", "Germany"),
        (555, "Aeroflot", "SKYTEAM", "Russia"),
        (257, "Austrian Airlines", "STAR ALLIANCE", "Austria"),
        (125, "British Airways", "ONEWORLD", "Britain");
# print the content of tables (just for testing if everything went right)
SELECT * FROM airticket;
SELECT * FROM airline;
SELECT * FROM airport;
SELECT * FROM client;
# now start with the queries (see queries from chapter 6 and 7)
```

10. Servelets

The following html page is index.html. The server shows this page when a client accesses the homepage by default path */airticketDB*. This page describes sign in and sign up views for clients.

/airticketDB/index. html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Sign in</title>
</head>
<body>
<form method="get" action="/airticketDB/TicketTable">
Customer ID: <input type="number" name="CID"/> <br>
<input type= "submit" value="Sign In"/>
</form>
<br>
or Sign up!
<form method="get" action="/airticketDB/Signup">
Name: <input type="text" name="CName"/> <br>
Age: <input type="number" name="CAge"/> <br>
Gender <br>
<input type="radio" name="CGender" value="male"/> Male<br>
<input type="radio" name="CGender" value="female"/> Female<br>
Nationality: <input type="text" name="CNation"/> <br>
<input type= "submit" value="Sign Up"/>
</form>
</body>
</html>
```

If a client tries to sign up, the server sends it to servlet */airticketDB/Signup*, or send it to servlet /airticketDB/TicketTable to show tickets with client id *CID*.

A. Inserting a new client into the database

The following servlet describes creating new user by parameters and inserting it into the database.

/airticketDB/Signup

```
package servlets;


import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
```

```java
import java.util.logging.Logger;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class Signup
 */
@WebServlet("/Signup")
public class Signup extends HttpServlet {
private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Signup() {
        super();
        // TODO Auto-generated constructor stub
    }

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
String name = request.getParameter("CName");
String age_text = request.getParameter("CAge");
String gender_text = request.getParameter("CGender");
String nation = request.getParameter("CNation");

int age;
boolean gender;

try {
if (name == null)
throw new IllegalArgumentException("Illegal Name");

try {
if (age_text == null)
throw new IllegalArgumentException("Illegal Age");

age = Integer.valueOf(age_text);
}
catch (NumberFormatException exc) {
throw new IllegalArgumentException("Illegal Age");
}

if (gender_text == null)
throw new IllegalArgumentException("Illegal Gender");
```

```
switch (gender_text) {
case "male":
gender = false;
break;

case "female":
gender = true;
break;

default:
throw new IllegalArgumentException("Illegal Gender");
}

if (nation == null)
throw new IllegalArgumentException("Illegal Nationality");
}
catch (IllegalArgumentException exc) {
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", exc.getMessage());
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
return;
}

Logger logger = Logger.getLogger(Signup.class.getName());

Connection conn = null;
Integer cid = null;

try {
Class.forName("com.mysql.cj.jdbc.Driver");

conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/11739592_airticketDB?se
rverTimezone=UTC&useSSL=false&allowPublicKeyRetrieval=true", "student",
"student");

Statement Stmt = conn.createStatement();
ResultSet rs = Stmt.executeQuery("SELECT MAX(CID) FROM client");

if (rs.next())
cid = rs.getInt(1);
if (cid != null)
cid++;

if (cid == null)
cid = 1;

PreparedStatement pStmt = conn.prepareStatement("INSERT INTO client(CID, CName,
CAge, CGender, CNation) VALUES (?,?,?,?,?)");
pStmt.setInt(1, cid);
pStmt.setString(2, name);
pStmt.setInt(3, age);
pStmt.setBoolean(4, gender);
pStmt.setString(5, nation);
pStmt.execute();

response.sendRedirect("/airticketDB/TicketTable?CID=" + Integer.toString(cid));
}
catch(ClassNotFoundException | SQLException exc) {
if (exc instanceof ClassNotFoundException)
logger.log(Level.SEVERE, "Can't find JDBC driver", exc);
else
logger.log(Level.SEVERE, "JDBC Error");

exc.printStackTrace();

RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Internal Server Error");
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
}
finally
{
try {
if (conn != null)
conn.close();
}
catch (SQLException exc) {
logger.log(Level.SEVERE, "Fail to close the connection");
exc.printStackTrace();
}
}
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
doGet(request, response);
}

}
```

The front part of the codes is for checking validity of parameters, and the back part is for inserting a new user into the database.

B. Displaying tickets of the client from the database

The following servlet is for displaying the ticket of the client by client id *CID*.

/airticketDB/TicketTable

```
package servlets;
```

```java
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import data.Ticket;

/**
 * Servlet implementation class TicketTable
 */
@WebServlet("/TicketTable")
public class TicketTable extends HttpServlet {
private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public TicketTable() {
        super();
        // TODO Auto-generated constructor stub
    }

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
Logger logger = Logger.getLogger(TicketTable.class.getName());

String cid_text = request.getParameter("CID");
Integer cid = null;

try {
if (cid_text == null)
throw new IllegalArgumentException("Illegal CID");
cid = Integer.valueOf(cid_text);
}
catch (IllegalArgumentException exc){
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Please Enter CID");
request.setAttribute("return_page", "/airticketDB/index.html");
```

```
view.forward(request, response);
return;
}

Connection conn = null;
try {
Class.forName("com.mysql.cj.jdbc.Driver");

conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/11739592_airticketDB?se
rverTimezone=UTC&useSSL=false&allowPublicKeyRetrieval=true", "student",
"student");

Statement Stmt = conn.createStatement();
ResultSet rs = Stmt.executeQuery("SELECT CName FROM client WHERE client.CID = "
+ cid.toString());
String name = null;

if (rs.next())
name = rs.getString("CName");

if (name == null)
throw new IllegalArgumentException("Illegal CID");

rs = Stmt.executeQuery("SELECT PD.PName, PA.PName, TDTime, TATime, TSeat, LName
"
+ "FROM client, airport PD, airport PA, airline, airticket "
+ "WHERE client.CID = " + cid.toString() + " "
+ "AND airticket.CID = client.CID "
+ "AND PD.PID = airticket.PDID "
+ "AND PA.PID = airticket.PAID "
+ "AND airline.LID = airticket.LID");

List<Ticket> tickets = new ArrayList<Ticket>();
while (rs.next()) {
Ticket ticket = new Ticket();
ticket.setAirportDName(rs.getString("PD.PName"));
ticket.setAirportAName(rs.getString("PA.PName"));
ticket.setTicketDTime(rs.getTimestamp("TDTime").toLocalDateTime().toString());
ticket.setTicketATime(rs.getTimestamp("TATime").toLocalDateTime().toString());
ticket.setTicketSeat(rs.getInt("TSeat"));
ticket.setAirlineName(rs.getString("LName"));

tickets.add(ticket);
}

RequestDispatcher view = request.getRequestDispatcher("/WEB-
INF/ticket_table.jsp");
request.setAttribute("CName", name);
request.setAttribute("tickets", tickets);

view.forward(request, response);
}
catch(IllegalArgumentException exc) {
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
```

```
request.setAttribute("msg", "No such CID");
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
}
catch(ClassNotFoundException | SQLException exc) {
if (exc instanceof ClassNotFoundException)
logger.log(Level.SEVERE, "Can't find JDBC driver", exc);
else
logger.log(Level.SEVERE, "JDBC Error");

exc.printStackTrace();

RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Internal Server Error");
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
}
finally
{
try {
if (conn != null)
conn.close();
}
catch (SQLException exc) {
logger.log(Level.SEVERE, "Fail to close the connection");
exc.printStackTrace();
}
}
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
doGet(request, response);
}

}
```

The front part of code is for checking validity of parameter, and the back part describes communication with the database. *Ticket* class is pre-defined data object class that have getter and setter methods to store information of a ticket. **The SELECT query in this servlet operates with 4 relations simultaneously to collects ticket information.**

Following *ticket_table.jsp* file describes the view part of the ticket table.

/airticketDB/WEB-INF/ticket_table.jsp

```
<%@ page language="java" import="java.util.List, data.Ticket"
contentType="text/html; charset=UTF-8"
```

```
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tickets</title>
</head>
<body>
Hello ${CName}! You are ${param.CID}. <br>
<a href="/airticketDB/add_ticket.jsp?CID=${param.CID}">Add Ticket</a>
<a href="/airticketDB/delete_ticket.jsp?CID=${param.CID}">Delete Ticket</a>
<a href="/airticketDB/update_user_info.jsp?CID=${param.CID}">Change User
Information</a>
<a href="/airticketDB/MileageTable?CID=${param.CID}">Check Your Mileages</a>
<a href="/airticketDB/MembershipWithdrawal?CID=${param.CID}">Membership
Withdrawal</a>
<a href="/airticketDB/NationalCarrier?CID=${param.CID}">Find national carriers
of your country</a>
<table border="1">
<thead>
<tr>
<th> Departure Airport </th>
<th> Departure Time </th>
<th> Arrival Airport </th>
<th> Arrival Time </th>
<th> Seat Number </th>
<th> Airline Name </th>
</tr>
</thead>
<tbody>
<%
@SuppressWarnings("unchecked")
List<Ticket> tickets = (List<Ticket>) request.getAttribute("tickets");
for (Ticket ticket : tickets) {
%>
<tr>
<td><%= ticket.getAirportDName() %></td>
<td><%= ticket.getTicketDTime() %></td>
<td><%= ticket.getAirportAName() %></td>
<td><%= ticket.getTicketATime() %></td>
<td><%= ticket.getTicketSeat() %></td>
<td><%= ticket.getAirlineName() %></td>
</tr>
<% } %>
</tbody>
</table>
</body>
</html>
```

C. Inserting a ticket into the database

Following *add_ticket.jsp* file describes the view for adding ticket.

/airticketDB/add_ticket.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Add Ticket</title>
</head>
<body>
<form method="get" action="/airticketDB/AddTicket">
<input type="hidden" name="CID" value="${param.CID}">
The departure airport location city: <input type="text" name="PDCity"/> <br>
The departure date: <input type="date" name="TDDate"/> <br>
The departure time: <input type="time" name="TDTime"/> <br>
The arrival airport location city: <input type="text" name="PACity"/> <br>
The arrival date: <input type="date" name="TADate"/> <br>
The arrival time: <input type="time" name="TATime"/> <br>
The airline name: <input type="text" name="LName"/> <br>
The seat number: <input type="number" name="TSeat"/> <br>
The Mileage of the airticket: <input type="number" name="TMileage"/> <br>
<input type="submit" value="Add Ticket"/>
</form>
</body>
</html>
```

The jsp file collects ticket information from a client and sends it to *AddTicket* servlet. *AddTicket* servlet gets this data by parameters and tries to add a ticket into the database.

/airticketDB/AddTicket

```
package servlets;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class AddTicket
 */
@WebServlet("/AddTicket")
public class AddTicket extends HttpServlet {
```

```
private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public AddTicket() {
        super();
        // TODO Auto-generated constructor stub
    }

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
Logger logger = Logger.getLogger(AddTicket.class.getName());

String cid_text = request.getParameter("CID");
Integer cid = null;

try {
if (cid_text == null)
throw new IllegalArgumentException ("Illegal CID");

try {
cid = Integer.valueOf(cid_text);
}
catch(NumberFormatException exc) {
throw new IllegalArgumentException ("Illegal CID");
}
}
catch (IllegalArgumentException exc) {
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", exc.getMessage());
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);

return;
}

Connection conn = null;
try {
Class.forName("com.mysql.cj.jdbc.Driver");

conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/11739592_airticketDB?se
rverTimezone=UTC&useSSL=false&allowPublicKeyRetrieval=true", "student",
"student");

String PDCity = request.getParameter("PDCity");
String PACity = request.getParameter("PACity");

if (PDCity == null)
throw new IllegalArgumentException("Please enter the departure city");
```

```
else if (PACity == null)
throw new IllegalArgumentException("Please enter the arrival city");

String TDDate_text = request.getParameter("TDDate");
String TDTime_text = request.getParameter("TDTime");
String TADate_text = request.getParameter("TADate");
String TATime_text = request.getParameter("TATime");

if (TDDate_text == null || TDTime_text == null || TADate_text == null ||
TATime_text == null)
throw new IllegalArgumentException("Please enter the dates and times");

Timestamp TDTime;
Timestamp TATime;
try {
SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd hh:mm");
TDTime = new Timestamp(formatter.parse(TDDate_text + " " + TDTime_text +
":00").getTime());
TATime = new Timestamp(formatter.parse(TADate_text + " " + TATime_text +
":00").getTime());
}
catch (ParseException exc){
throw new IllegalArgumentException("Illegal Date Format (yyyy.MM.dd hh:mm)");
}

String LName = request.getParameter("LName");
if (LName == null)
throw new IllegalArgumentException("Please enter the name of airline");

String TSeat_text = request.getParameter("TSeat");
int TSeat;

if (TSeat_text == null)
throw new IllegalArgumentException("Please enter the seat number");
try {
TSeat = Integer.valueOf(TSeat_text);
}
catch (NumberFormatException exc){
throw new IllegalArgumentException("Illegal seat number");
}

String TMileage_text = request.getParameter("TMileage");
int TMileage;

if (TMileage_text == null)
throw new IllegalArgumentException("Please enter the mileage of the ticket");
try {
TMileage = Integer.valueOf(TMileage_text);
}
catch (NumberFormatException exc) {
throw new IllegalArgumentException("Illegal mileage");
}

PreparedStatement pStmt = conn.prepareStatement("INSERT INTO airticket(CID, LID,
TDTime, TATime, TMileage, TSeat, PDID, PAID) "
+ "SELECT ?, airline.LID, ?, ?, ?, ?, PD.PID, PA.PID "
+ "FROM airline, airport PD, airport PA "
+ "WHERE airline.LName = ? "
+ "AND PD.PCity = ? "
+ "AND PA.PCity = ?");
pStmt.setInt(1, cid);
pStmt.setTimestamp(2, TDTime);
pStmt.setTimestamp(3, TATime);
pStmt.setInt(4, TMileage);
pStmt.setInt(5, TSeat);
pStmt.setString(6, LName);
pStmt.setString(7, PDCity);
pStmt.setString(8, PACity);

if (pStmt.executeUpdate() == 0)
throw new IllegalArgumentException("Illegal parameters. Please checks a airport
name or airline name.");

response.sendRedirect("/airticketDB/TicketTable?CID=" + cid.toString());
}
catch(IllegalArgumentException exc) {
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", exc.getMessage());
request.setAttribute("return_page", "/airticketDB/TicketTable?CID=" +
cid.toString());
view.forward(request, response);
}
catch(ClassNotFoundException | SQLException exc) {
if (exc instanceof ClassNotFoundException)
logger.log(Level.SEVERE, "Can't find JDBC driver", exc);
else
logger.log(Level.SEVERE, "JDBC Error");

exc.printStackTrace();

RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Internal Server Error");
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
}
finally
{
try {
if (conn != null)
conn.close();
}
catch (SQLException exc) {
logger.log(Level.SEVERE, "Fail to close the connection");
exc.printStackTrace();
}
}
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
doGet(request, response);
}

}
```

The INSERT query in the servlet gets foreign keys LID, PDID, and PAID by selecting LName, PDCity, and PACity simultaneously. Thus, it restricts referential integrity.

D. Deleting a ticket from the database

Following *delete_ticket.jsp* file describes the view for deleting ticket.

/airticketDB/delete_ticket.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Delete Ticket</title>
</head>
<body>
<form method="get" action="/airticketDB/DeleteTicket">
<input type="hidden" name="CID" value="${param.CID}">
The airline name: <input type="text" name="LName"/> <br>
The departure date: <input type="date" name="TDDate"/> <br>
The departure time: <input type="time" name="TDTime"/> <br>
<input type="submit" value="Delete Ticket"/>
</form>
</body>
</html>
```

The jsp file collects ticket information from a client and sends it to *DeleteTicket* servlet. *DeleteTicket* servlet gets this data by parameters and tries to delete the matched ticket from the database.

/airticketDB/DeleteTicket

```
package servlets;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class DeleteTicket
 */
@WebServlet("/DeleteTicket")
public class DeleteTicket extends HttpServlet {
private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public DeleteTicket() {
        super();
        // TODO Auto-generated constructor stub
    }

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
Logger logger = Logger.getLogger(DeleteTicket.class.getName());

String cid_text = request.getParameter("CID");
Integer cid = null;

try {
if (cid_text == null)
throw new IllegalArgumentException ("Illegal CID");

try {
cid = Integer.valueOf(cid_text);
}
catch(NumberFormatException exc) {
throw new IllegalArgumentException ("Illegal CID");
}
}
catch (IllegalArgumentException exc) {
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", exc.getMessage());
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);

return;
}

Connection conn = null;
try {
Class.forName("com.mysql.cj.jdbc.Driver");

conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/11739592_airticketDB?se
rverTimezone=UTC&useSSL=false&allowPublicKeyRetrieval=true", "student",
"student");

String LName = request.getParameter("LName");

if (LName == null)
throw new IllegalArgumentException("Please enter the airline name");

String TDDate_text = request.getParameter("TDDate");
String TDTime_text = request.getParameter("TDTime");

if (TDDate_text == null || TDTime_text == null)
throw new IllegalArgumentException("Please enter the dates and times");

Timestamp TDTime;
try {
SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd hh:mm");
TDTime = new Timestamp(formatter.parse(TDDate_text + " " + TDTime_text +
":00").getTime());
}
catch (ParseException exc){
throw new IllegalArgumentException("Illegal Date Format (yyyy.MM.dd hh:mm)");
}

PreparedStatement pStmt = conn.prepareStatement("DELETE FROM airticket "
+ "WHERE CID = ? and TDTime = ? "
+ "and LID IN (Select LID FROM airline WHERE LName = ?)");
pStmt.setInt(1, cid);
pStmt.setTimestamp(2, TDTime);
pStmt.setString(3, LName);

if (pStmt.executeUpdate() == 0)
throw new IllegalArgumentException("No such ticket. Please check the airline
name");

response.sendRedirect("/airticketDB/TicketTable?CID=" + cid.toString());
}
catch(IllegalArgumentException exc) {
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", exc.getMessage());
request.setAttribute("return_page", "/airticketDB/TicketTable?CID=" +
cid.toString());
view.forward(request, response);
```

```
}
catch(ClassNotFoundException | SQLException exc) {
if (exc instanceof ClassNotFoundException)
logger.log(Level.SEVERE, "Can't find JDBC driver", exc);
else
logger.log(Level.SEVERE, "JDBC Error");

exc.printStackTrace();

RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Internal Server Error");
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
}
finally
{
try {
if (conn != null)
conn.close();
}
catch (SQLException exc) {
logger.log(Level.SEVERE, "Fail to close the connection");
exc.printStackTrace();
}
}
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
doGet(request, response);
}

}
```

The DELETE query in the servlet gets foreign key LID by selecting LName simultaneously. Thus, it restricts referential integrity.

E. Displaying mileage points sum that is over 1000 points for each airline alliances.

The following servlet is for displaying mileage points sum for each airline alliances.

/airticketDB/MileageTable

```
package servlets;

import java.io.IOException;
```

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import data.Alliance;

/**
 * Servlet implementation class MileageTable
 */
@WebServlet("/MileageTable")
public class MileageTable extends HttpServlet {
private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public MileageTable() {
        super();
        // TODO Auto-generated constructor stub
    }

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
Logger logger = Logger.getLogger(MileageTable.class.getName());

String cid_text = request.getParameter("CID");
Integer cid = null;

try {
if (cid_text == null)
throw new IllegalArgumentException("Illegal CID");
cid = Integer.valueOf(cid_text);
}
catch (IllegalArgumentException exc){
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Illegal CID");
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
```

```
return;
}

Connection conn = null;
try {
Class.forName("com.mysql.cj.jdbc.Driver");

conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/11739592_airticketDB?se
rverTimezone=UTC&useSSL=false&allowPublicKeyRetrieval=true", "student",
"student");

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT LAlliance, SUM(TMileage) as Sum_Mileage
"
+ "FROM airticket, airline "
+ "WHERE CID = " + cid.toString() + " "
+ "AND airticket.LID = airline.LID "
+ "GROUP BY LAlliance "
+ "Having Sum_Mileage >= 1000");

List<Alliance> alliances = new ArrayList<Alliance>();
while (rs.next()) {
Alliance alliance = new Alliance();
alliance.setName(rs.getString("LAlliance"));
alliance.setMileage(rs.getInt("Sum_Mileage"));
alliances.add(alliance);
}

RequestDispatcher view = request.getRequestDispatcher("/WEB-
INF/mileage_table.jsp");
request.setAttribute("alliances", alliances);

view.forward(request, response);
}
catch(ClassNotFoundException | SQLException exc) {
if (exc instanceof ClassNotFoundException)
logger.log(Level.SEVERE, "Can't find JDBC driver", exc);
else
logger.log(Level.SEVERE, "JDBC Error");

exc.printStackTrace();

RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Internal Server Error");
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
}
finally
{
try {
if (conn != null)
conn.close();
}
```

```
catch (SQLException exc) {
logger.log(Level.SEVERE, "Fail to close the connection");
exc.printStackTrace();
}
}
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
doGet(request, response);
}

}
```

Alliance class is pre-defined data object class that have getter and setter methods to store information about an airline alliance name and its mileage points. **The SELECT query in this servlet illustrates Group by and Having clauses to get data for each alliances and operates with 2 relations simultaneously.** Following *mileage_table.jsp* file describes the view part of the mileage table.

/airticketDB/WEB-INF/mileage_table.jsp

```
<%@ page language="java" import="java.util.List,data.Alliance"
contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Mileage Table</title>
</head>
<body>
Your Mileages having 1000 points or more. <br>
<table border="1">
<thead>
<tr>
<th> Airline Alliance </th>
<th> Mileage Sum </th>
</tr>
</thead>
<tbody>
<%
@SuppressWarnings("unchecked")
List<Alliance> alliances = (List<Alliance>) request.getAttribute("alliances");
for (Alliance alliance : alliances) {
```

```
%>
<tr>
<td><%= alliance.getName() %></td>
<td><%= alliance.getMileage() %></td>
</tr>
<% } %>
</tbody>
</table>
</body>
</html>
```

F. Updating client information.

Following *update_user_info.jsp* describes the view for updating user information..

/airticketDB/update_user_info.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Update User Info</title>
</head>
<body>
<form method="get" action="/airticketDB/UpdateUserInfo">
<input type="hidden" name="CID" value="${param.CID}">
Name: <input type="text" name="CName"/> <br>
Age: <input type="number" name="CAge"/> <br>
Gender <br>
<input type="radio" name="CGender" value="male"/> Male<br>
<input type="radio" name="CGender" value="female"/> Female<br>
Nationality: <input type="text" name="CNation"/> <br>
<input type= "submit" value="Update"/>
</form>
</body>
</html>
```

The jsp file collects updated client information and sends it to *UpdateUserInfo* servlet. *UpdateUserInfo* servlet gets this data by parameters and tries to update the user infomation. **The query in this servlet illustrates updating.**

/airticketDB/UpdateUserInfo

```
package servlets;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class UpdateUserInfo
 */
@WebServlet("/UpdateUserInfo")
public class UpdateUserInfo extends HttpServlet {
private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public UpdateUserInfo() {
        super();
        // TODO Auto-generated constructor stub
    }

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
Logger logger = Logger.getLogger(UpdateUserInfo.class.getName());

Integer cid = null;

try {
String cid_text =request.getParameter("CID");

if (cid_text == null)
throw new IllegalArgumentException ("Illegal CID");

try {
cid = Integer.valueOf(cid_text);
}
catch(NumberFormatException exc) {
throw new IllegalArgumentException ("Illegal CID");
}
}
catch (IllegalArgumentException exc) {
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", exc.getMessage());
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
```

```
}

Connection conn = null;

try {
String name = request.getParameter("CName");
if (name == null)
throw new IllegalArgumentException("Illegal Name");

String age_text = request.getParameter("CAge");
int age;
try {
if (age_text == null)
throw new IllegalArgumentException("Illegal Age");

age = Integer.valueOf(age_text);
}
catch (NumberFormatException exc) {
throw new IllegalArgumentException("Illegal Age");
}

String gender_text = request.getParameter("CGender");
boolean gender;
if (gender_text == null)
throw new IllegalArgumentException("Illegal Gender");

switch (gender_text) {
case "male":
gender = true;
break;

case "female":
gender = false;
break;

default:
throw new IllegalArgumentException("Illegal Gender");
}

String nation = request.getParameter("CNation");
if (nation == null)
throw new IllegalArgumentException("Illegal Nationality");

Class.forName("com.mysql.cj.jdbc.Driver");

conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/11739592_airticketDB?se
rverTimezone=UTC&useSSL=false&allowPublicKeyRetrieval=true", "student",
"student");

PreparedStatement pStmt = conn.prepareStatement("UPDATE client "
+ "SET CName = ?, CAge = ?, CGender = ?, CNation = ? "
+ "WHERE CID = ?");
pStmt.setString(1,  name);
pStmt.setInt(2, age);
pStmt.setBoolean(3, gender);
pStmt.setString(4, nation);
pStmt.setInt(5, cid);

if(pStmt.executeUpdate() == 0)
throw new IllegalArgumentException("No such CID");

response.sendRedirect("/airticketDB/TicketTable?CID=" + Integer.toString(cid));
}
catch (IllegalArgumentException exc) {
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", exc.getMessage());
request.setAttribute("return_page", "/airticketDB/TicketTable?CID=" +
String.valueOf(cid));
view.forward(request, response);
return;
}
catch(ClassNotFoundException | SQLException exc) {
if (exc instanceof ClassNotFoundException)
logger.log(Level.SEVERE, "Can't find JDBC driver", exc);
else
logger.log(Level.SEVERE, "JDBC Error");

exc.printStackTrace();

RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Internal Server Error");
request.setAttribute("return_page", "/airticketDB/TicketTable?CID=" +
cid.toString());
view.forward(request, response);
}
finally
{
try {
if (conn != null)
conn.close();
}
catch (SQLException exc) {
logger.log(Level.SEVERE, "Fail to close the connection");
exc.printStackTrace();
}
}
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
doGet(request, response);
}

}
```

G. Membership withdrawal.

Following *MembershipWithdrawal* servlet describes withdrawal of membership. Because there is no private page for this servlet, there is no jsp file for view part. **The thickets related with the deleted client are deleted automatically by *ON DELETE CASCADE* clause, thus this illustrates referential integrity of the database.**

/airticketDB/MembershipWithdrawal

```
package servlets;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class MembershipWithdrawal
 */
@WebServlet("/MembershipWithdrawal")
public class MembershipWithdrawal extends HttpServlet {
private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public MembershipWithdrawal() {
        super();
        // TODO Auto-generated constructor stub
    }

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse 
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) 
throws ServletException, IOException {
Logger logger = Logger.getLogger(MembershipWithdrawal.class.getName());

String cid_text = request.getParameter("CID");
Integer cid = null;
```

```
try {
if (cid_text == null)
throw new IllegalArgumentException ("Illegal CID");

try {
cid = Integer.valueOf(cid_text);
}
catch(NumberFormatException exc) {
throw new IllegalArgumentException ("Illegal CID");
}
}
catch (IllegalArgumentException exc) {
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", exc.getMessage());
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);

return;
}

Connection conn = null;

try {
Class.forName("com.mysql.cj.jdbc.Driver");

conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/11739592_airticketDB?se
rverTimezone=UTC&useSSL=false&allowPublicKeyRetrieval=true", "student",
"student");

Statement stmt = conn.createStatement();
stmt.execute("DELETE FROM client WHERE CID = " + cid.toString());

response.sendRedirect("/airticketDB/index.html");
}
catch(IllegalArgumentException exc) {
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", exc.getMessage());
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
}
catch(ClassNotFoundException | SQLException exc) {
if (exc instanceof ClassNotFoundException)
logger.log(Level.SEVERE, "Can't find JDBC driver", exc);
else
logger.log(Level.SEVERE, "JDBC Error");

exc.printStackTrace();

RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Internal Server Error");
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
}
```

```
finally
{
try {
if (conn != null)
conn.close();
}
catch (SQLException exc) {
logger.log(Level.SEVERE, "Fail to close the connection");
exc.printStackTrace();
}
}
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
doGet(request, response);
}

}
```

H. Showing national carriers by client nationality.

The following servlet is for showing national carriers according to client nationality.

/airticketDB/NationalCarrier

```
package servlets;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import data.Airline;
import data.Alliance;
```

```java
/**
 * Servlet implementation class NationalCarrier
 */
@WebServlet("/NationalCarrier")
public class NationalCarrier extends HttpServlet {
private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public NationalCarrier() {
        super();
        // TODO Auto-generated constructor stub
    }

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
Logger logger = Logger.getLogger(MileageTable.class.getName());

String cid_text = request.getParameter("CID");
Integer cid = null;

try {
if (cid_text == null)
throw new IllegalArgumentException("Illegal CID");
cid = Integer.valueOf(cid_text);
}
catch (IllegalArgumentException exc){
RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Illegal CID");
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
return;
}

Connection conn = null;
try {
Class.forName("com.mysql.cj.jdbc.Driver");

conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/11739592_airticketDB?se
rverTimezone=UTC&useSSL=false&allowPublicKeyRetrieval=true", "student",
"student");

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT CNation, LName, LAlliance "
+ "FROM client, airline "
+ "WHERE CID = " + cid.toString() + " "
+ "AND CNation = LNation");
List<Airline> airlines = new ArrayList<Airline>();
while (rs.next()) {
Airline airline = new Airline();
airline.setName(rs.getString("LName"));
airline.setNation(rs.getString("CNation"));
airline.setAlliance(rs.getString("LAlliance"));
airlines.add(airline);
}

RequestDispatcher view = request.getRequestDispatcher("/WEB-
INF/national_carrier.jsp");
request.setAttribute("airlines", airlines);

view.forward(request, response);
}
catch(ClassNotFoundException | SQLException exc) {
if (exc instanceof ClassNotFoundException)
logger.log(Level.SEVERE, "Can't find JDBC driver", exc);
else
logger.log(Level.SEVERE, "JDBC Error");

exc.printStackTrace();

RequestDispatcher view = request.getRequestDispatcher("/WEB-INF/error.jsp");
request.setAttribute("msg", "Internal Server Error");
request.setAttribute("return_page", "/airticketDB/index.html");
view.forward(request, response);
}
finally
{
try {
if (conn != null)
conn.close();
}
catch (SQLException exc) {
logger.log(Level.SEVERE, "Fail to close the connection");
exc.printStackTrace();
}
}
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
doGet(request, response);
}

}
```

Airline class is pre-defined data object class that have getter and setter methods to store information about an airline name, nationality, and alliance that the airline joined. **The SELECT query in this servlet operates with 2 relations simultaneously.** Following *national_carrier.jsp* file describes the view part of the airline table.

/airticketDB/WEB-INF/national_carrier.jsp

```
<%@ page language="java" import="java.util.List, data.Airline"
contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<title>National Carriers</title>
</head>
<body>
National carriers for your country.
<table border="1">
<thead>
<tr>
<th> Nationality </th>
<th> Airline Name </th>
<th> Alliance </th>
</tr>
</thead>
<tbody>
<%
@SuppressWarnings("unchecked")
List<Airline> airlines = (List<Airline>) request.getAttribute("airlines");
for (Airline airline : airlines) {
%>
<tr>
<td><%= airline.getNation() %></td>
<td><%= airline.getName() %></td>
<td><%= airline.getAlliance() %></td>
</tr>
<% } %>
</tbody>
</table>
</body>
</html>
```