Project 5:

Administrator Manual

Ryan D. Ernst

Prof. Dawn Robinson

CMSC 461-01, Database Management Systems

May 18, 2017

Introduction:

The project assigned to Professor Robinson’s section of Database Management Systems required the creation of a functioning application that interfaces with an existing database. The exact requirements for this application were listed as being able to perform a selection statement, insertion statements, and delete statements. Also specified was the ability for a user to enter his or her password and user identification prior to connecting to the database. The purposes of this document are to show that these requirements were met, explain the installation and development set-up process, how to perform routine operational tasks, and provide references used during development.

Installation:

This installation procedure assumes that the database has already been created. In addition, the file ojdbc14.jar must have been downloaded from the course Blackboard page. This is because this program is to be run within the confines of an IDE such as Eclipse. Although the exact IDE configuration process varies from person to person, it is absolutely paramount that the ojdbc14.jar file be added to the IDE’s build path/class path. The exact steps for installing this program are simply as follows.

1. Locate and launch the Start Database function of Oracle to initialize the database.
2. Download the file CMSC461\_Project5.java to the Desktop.
3. Import ojdbc14.jar into your IDE build path environment.
   1. Follow the procedure as stated by your IDE of choice.
4. Compile and then Run the CMSC461\_Project5.java file

Routine Tasks:

The routine tasks one can perform in this system involves performing the required selection, insertion, and deletion operations. Some additional tasks the user can perform in this program involve logging into the system and deciding whether to commit or rollback the changes made during the insertion and deletion operations. Each of these tasks will be discussed in the subsequent sections

1. Logging on

The log in procedure operates in a similar manner to a typical log in applet one would expect on a major application. The specific code used to handle this applet is shown below.

Scanner scan = new Scanner(System.in);  
String userID = "";  
String password = "";  
   
System.out.println("Welcome. Please enter your USERNAME and PASSWORD.");  
System.out.print("USERNAME: ");  
userID = scan.next();  
System.out.print("PASSWORD: ");  
password = scan.next();  
JDBC(userID, password, scan);

Fig. 1. – Log in code

This code operates by simply requesting the user to enter his or her username and password. These are then to be scanned into the program as String literals and passed into the method named JDBC. It is then within the JDBC method that the actual connection to the database is attempted.

Class.forName("oracle.jdbc.driver.OracleDriver");  
Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",userID,password);

Statement stmt = conn.createStatement();  
System.out.println("Connection Established!");  
conn.setAutoCommit(false);

Fig. 2 - Connection

All that is required from the user during this procedure is that they first enter their username and password when required. An example of this is shown in Figure 3 below.

Welcome. Please enter your USERNAME and PASSWORD.

USERNAME: <INSERT USERNAME HERE>

PASSWORD: <INSERT PASSWORD HERE>

Connection Established!

Fig. 3. – Log in Example

1. Selection

Following the login procedure, the program will automatically perform the selection statement. It, specifically, will display each instructor in the database teaching a Computer Science course during a spring semester. The query used is hardcoded into the system and is automatically displayed. The exact query is shown below in addition to its output.

select distinct INSTRUCTOR.NAME, TEACHES.COURSE\_ID, TEACHES.SEC\_ID, TEACHES.SEMESTER, TEACHES.YEAR, SECTION.BUILDING, SECTION.ROOM\_NUMBER

from INSTRUCTOR join TEACHES on TEACHES.ID = INSTRUCTOR.ID join SECTION

on TEACHES.COURSE\_ID = SECTION.COURSE\_ID

where INSTRUCTOR.DEPT\_NAME = 'Comp. Sci.' and TEACHES.SEMESTER = 'Spring'

order by INSTRUCTOR.NAME, TEACHES.COURSE\_ID, TEACHES.SEC\_ID, TEACHES.YEAR

Fig. 4 – Selection Query

Query 1: Selection  
Name Course\_Name Section\_ID Semester Year Building Room#  
-------------------------------------------------------------------------------------  
Brandt CS-190 1 Spring 2009 Taylor 3128  
Brandt CS-190 2 Spring 2009 Taylor 3128  
Brandt CS-319 2 Spring 2010 Taylor 3128  
Brandt CS-319 2 Spring 2010 Watson 100  
Katz CS-101 1 Spring 2010 Packard 101  
Katz CS-319 1 Spring 2010 Watson 100  
Katz CS-319 1 Spring 2010 Taylor 3128  
Srinivasan CS-315 1 Spring 2010 Watson 120

Fig. 5 – Selection Output

1. Insertion

It is once the selection statement finish printing that the insertion statement automatically executes and the results are displayed. In particular, the following queries are executed.

INSERT INTO STUDENT(ID, NAME, DEPT\_NAME, TOT\_CRED)

VALUES('99999','Flacco', 'Biology', 80)

INSERT INTO STUDENT(ID, NAME, DEPT\_NAME, TOT\_CRED)

VALUES('99998','Ripken', 'History', 120)

INSERT INTO STUDENT(ID, NAME, DEPT\_NAME, TOT\_CRED)

VALUES('45679','Smith', 'Finance', 5)

INSERT INTO STUDENT(ID, NAME, DEPT\_NAME, TOT\_CRED)

VALUES('00022','Freese', 'History', 100)

Fig. 6 – Insertion Statements

These then produce the following results.

Query 2: Insertion  
ÏInserting Flacco, Ripken, Smith, and Freese  
Ï  
ÏID Name Major Credits  
Ï-----------------------------------------  
Ï00022 Freese History 100  
Ï00128 Zhang Comp. Sci. 102  
Ï12345 Shankar Comp. Sci. 32  
Ï19991 Brandt History 80  
Ï23121 Chavez Finance 110  
Ï44553 Peltier Physics 56  
Ï45678 Levy Physics 46  
Ï45679 Smith Finance 5  
Ï54321 Williams Comp. Sci. 54  
Ï55739 Sanchez Music 38  
Ï70557 Snow Physics 0  
Ï76543 Brown Comp. Sci. 58  
Ï76653 Aoi Elec. Eng. 60  
Ï98765 Bourikas Elec. Eng. 98  
Ï98988 Tanaka Biology 120  
Ï99998 Ripken History 120

Fig. 7 – Insertion Output

It is after this output that the user will be prompted to either commit or rollback the changes made by the insertion statement. Selecting “r” would cause a rollback to happen and selecting “c” would cause a commit. It is case insensitive so a capital version of these characters would also be considered acceptable input. All other inputs result in an “unable to process” message being displayed and automatically causing a rollback. An example of these prompts is displayed below.

ÏDo you wish to commit this changes?  
ÏPress "c" to commit or "r" to rollback.  
 Choice: r  
ÏRolling back.

Fig. 8 – Commit/Rollback Prompt Example

1. Deletion

The program will automatically perform the deletion statement after processing the user’s previous request. This is similar to the insertion statement in that the query is hardcoded into the program, displays the results, and prompts the user for committing or rolling back confirmation. The key difference here is that the deletion queries shown below are executed.

DELETE FROM INSTRUCTOR WHERE NAME = 'Mozart'

DELETE FROM INSTRUCTOR WHERE NAME = 'Crick'

Fig. 9 – Deletion Queries

Query 3: Deletion  
ÏDeleting Mozart and Crick  
Ï  
ÏID Name Department Salary  
Ï---------------------------------------------------------------------------  
Ï10101 Srinivasan Comp. Sci. 65000  
Ï12121 Wu Finance 90000  
Ï22222 Einstein Physics 95000  
Ï32343 El Said History 60000  
Ï33456 Gold Physics 87000  
Ï45565 Katz Comp. Sci. 75000  
Ï58583 Califieri History 62000  
Ï76543 Singh Finance 80000  
Ï83821 Brandt Comp. Sci. 92000  
Ï98345 Kim Elec. Eng. 80000  
Fig. 10 – Deletion Output

ÏDo you wish to commit this changes?  
ÏPress "c" to commit or "r" to rollback.  
 Choice: c  
ÏCommitting.

Fig. 12 – Commit/Rollback Prompt Example

1. Shut Down

The program will automatically close after finalizing the commit or rollback from the deletion statement. However, the user can reinitialize the program and continue to run it so long as the database remains active. It is by click Oracle’s Stop Database function that the entire system is shut down. When the program completes its shut down procedure, the message in Figure 13 is displayed on screen.

System.out.println("\nProgram Over. Shutting down.");  
System.out.println("Thank you and goodbye.");

Fig. 13 – Shutting down Message

Appendix A – References:

<http://www.ntu.edu.sg/home/ehchua/programming/java/JDBC_Basic.html>

<https://www.tutorialspoint.com/servlets/servlets-database-access.htm>

Appendix B – Source Code:

import java.util.Scanner;  
import java.sql.\*;  
  
public class CMSC461\_Project5{  
 public static void main(String []args){  
 Scanner scan = new Scanner(System.in);  
 String userID = "";  
 String password = "";  
   
 System.out.println("Welcome. Please enter your USERNAME and PASSWORD.");  
 System.out.print("USERNAME: ");  
 userID = scan.next();  
 System.out.print("PASSWORD: ");  
 password = scan.next();  
 JDBC(userID, password, scan);  
   
 System.out.println("\nProgram Over. Shutting down.");  
 System.out.println("Thank you and goodbye.");  
 }  
   
 public static void JDBC(String userID, String password, Scanner scan){  
 try{  
 Class.forName("oracle.jdbc.driver.OracleDriver");  
 Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",userID,password);  
 Statement stmt = conn.createStatement();  
 System.out.println("Connection Established!");  
 conn.setAutoCommit(false);  
   
 System.out.println("\nQuery 1: Selection");  
 try{  
 System.out.println("Name\tCourse\_Name\tSection\_ID\tSemester\t" +   
 "Year\tBuilding\tRoom#");  
 System.out.println("--------------------------------------------------------" +   
 "-----------------------------");  
 String sql1 = "SELECT distinct INSTRUCTOR.NAME, TEACHES.COURSE\_ID, TEACHES.SEC\_ID," +   
 " TEACHES.SEMESTER, TEACHES.YEAR, SECTION.BUILDING, SECTION.ROOM\_NUMBER" +  
 " from INSTRUCTOR join TEACHES on TEACHES.ID = INSTRUCTOR.ID join SECTION" +   
 " on TEACHES.COURSE\_ID = SECTION.COURSE\_ID" +  
 " where INSTRUCTOR.DEPT\_NAME = 'Comp. Sci.' and" +   
 " TEACHES.SEMESTER = 'Spring'" +   
 " order by INSTRUCTOR.NAME, TEACHES.COURSE\_ID, TEACHES.SEC\_ID, TEACHES.YEAR";  
 ResultSet rs = stmt.executeQuery(sql1);  
   
 while (rs.next()){  
 String output = rs.getString(1) + "\t" +  
 rs.getString(2) + " " +  
 rs.getString(3) + " " +  
 rs.getString(4) + " " +  
 rs.getString(5) + "\t" +  
 rs.getString(6) + " " +   
 rs.getString(7);  
 System.out.println(output);  
 }  
 }  
 catch(SQLException se){  
 System.out.println("Exception: " + se);  
 conn.rollback();  
 }  
   
 System.out.println("\nQuery 2: Insertion");  
 try{  
 System.out.println("Inserting Flacco, Ripken, Smith, and Freese");  
   
 String sqlInsert1 = "INSERT INTO STUDENT(ID, NAME, DEPT\_NAME, TOT\_CRED) " +  
 "VALUES('99999','Flacco', 'Biology', 80)";  
 stmt.executeUpdate(sqlInsert1);  
   
 String sqlInsert2 = "INSERT INTO STUDENT(ID, NAME, DEPT\_NAME, TOT\_CRED) " +  
 "VALUES('99998','Ripken', 'History', 120)";  
 stmt.executeUpdate(sqlInsert2);  
   
 String sqlInsert3 = "INSERT INTO STUDENT(ID, NAME, DEPT\_NAME, TOT\_CRED) " +  
 "VALUES('45679','Smith', 'Finance', 5)";  
 stmt.executeUpdate(sqlInsert3);  
   
 String sqlInsert4 = "INSERT INTO STUDENT(ID, NAME, DEPT\_NAME, TOT\_CRED) " +  
 "VALUES('00022','Freese', 'History', 100)";  
 stmt.executeUpdate(sqlInsert4);  
   
   
 String sql2 = "SELECT distinct \* " +  
 " from STUDENT" +   
 " order by STUDENT.ID";  
 ResultSet rs2 = stmt.executeQuery(sql2);  
   
 System.out.println("\nID\tName\t Major\t\tCredits");  
 System.out.println("-----------------------------------------");  
 while (rs2.next()){  
 String output = rs2.getString(1) + "\t" +  
 rs2.getString(2) + "\t " +  
 rs2.getString(3) + "\t " +   
 rs2.getString(4);  
 System.out.println(output);  
 }  
   
 System.out.println("\nDo you wish to commit this changes?\nPress \"c\" to commit or \"r\" to rollback.");  
 System.out.print("Choice: ");  
 String choice1 = scan.next();  
   
 switch(choice1){  
 case "c":  
 case "C":  
 System.out.println("Committing.");  
 conn.commit();  
 break;  
 case "r":  
 case "R":  
 System.out.println("Rolling back.");  
 conn.rollback();  
 break;  
 default:  
 System.out.println("Unable to process. Incorrect Input.");  
 break;  
 }//end switch  
 }  
 catch(SQLException se){  
 System.out.println("Exception: " + se);  
 conn.rollback();  
 }  
   
 System.out.println("\nQuery 3: Deletion");  
 try{  
 System.out.println("Deleting Mozart and Crick");  
   
 String sqlDelete1 = "DELETE FROM INSTRUCTOR " +  
 "WHERE NAME = 'Mozart'";  
 stmt.executeUpdate(sqlDelete1);  
   
 String sqlDelete2 = "DELETE FROM INSTRUCTOR " +  
 "WHERE NAME = 'Crick'";  
 stmt.executeUpdate(sqlDelete2);  
   
 String sql3 = "SELECT distinct \* " +  
 " from INSTRUCTOR" +   
 " order by INSTRUCTOR.ID";  
 ResultSet rs3 = stmt.executeQuery(sql3);  
   
 System.out.println("\nID\tName\t\tDepartment\tSalary");  
 System.out.println("----------------------------------------------" +   
 "-----------------------------");  
 while (rs3.next()){  
 String output = rs3.getString(1) + "\t" +   
 rs3.getString(2) + "\t\t" +  
 rs3.getString(3) + "\t" +  
 rs3.getString(4);  
 System.out.println(output);  
 }  
   
 System.out.println("\nDo you wish to commit this changes?\nPress \"c\" to commit or \"r\" to rollback.");  
 System.out.print("Choice: ");  
 String choice2 = scan.next();  
   
 switch(choice2){  
 case "c":  
 case "C":  
 System.out.println("Committing.");  
 conn.commit();  
 break;  
 case "r":  
 case "R":  
 System.out.println("Rolling back.");  
 conn.rollback();  
 break;  
 default:  
 System.out.println("Unable to process. Incorrect Input.");  
 break;  
 }//end switch  
 }  
 catch(SQLException se){  
 System.out.println("Exception: " + se);  
 conn.rollback();  
 }  
   
 }  
 catch(Exception sqle){  
 System.out.println("Exception: " + sqle);  
 }  
 }  
}