**CS532 – DATABASE SYSTEMS**

**Project: 2**

**Implement Student Registration System using PL/SQL and JDBC**

**SUBMITTED BY:**

|  |  |  |
| --- | --- | --- |
| NAME | EMAIL ID | SIGNATURE |
| VARUN SAXENA | [vsaxena1@binghamton.edu](mailto:vsaxena1@binghamton.edu) |  |
| KUNDAN SHRIVASTAV | [kshriva1@binghamton.edu](mailto:kshriva1@binghamton.edu) |  |
| SEAN GALLAGHER | [sgallag3@binghamton.edu](mailto:sgallag3@binghamton.edu) |  |
|  |  |  |

**CONTENTS:**

1. **Introduction – Project Description**
2. **Project Implementation**
3. **Project Overflow**
4. **PL/SQL code – Triggers**
5. **Stored Procedures**
6. **Java Methods**
7. **PL/SQL code**
8. **Procedure to show all table information**
9. **PL/SQL code – Procedure to show various tables details**
10. **PL/SQL code - Procedure to show the pre-requisites courses**
11. **PL/SQL code - Procedure to show the TAs details**
12. **PL/SQL code – Procedure to enroll a student**
13. **PL/SQL code – Procedure to drop a student from a class**
14. **PL/SQL code – Procedure to delete a student**
15. **Java Code**
16. **Introduction – Project Description:**

The project uses Oracle's PL/SQL and JDBC to create an application to for student registration system at a university using an interface.

Different data of students is manipulated as per the given requirements using procedures in PL/SQL.

The student data can be added and deleted from the tables.

Triggers and sequences are created to track the changes in tables.

1. **Project Implementation:**

The sequence starts generating **log\_seq** from 100 and increments by 1 when new log records are inserted into the logs table. It uses sequence to generate the output with starting point as 100 and incrementing the value by 1.

1. **Project Overflow created in this project:**

SYS\_REFCURSOR is used to return from the stored procedures and functions.

**PL/SQL code – Triggers**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **TRIGGER NAME** | **DESCRIPTION** |
|  | **ENROLLMENTS\_INSERT** | Insert or update the log table using triggers. The classes table is updated by incrementing the value by 1 when a new entry is entered in the enrollments table. |
|  | **ENROLLMENTS\_DELETE** | Update the log table by firing the trigger by decrementing the class size by 1 while deleting the student from enrollments. |
|  | **STUDENT\_DELETE** | Update the log table while deleting the student from the student table. If the student is enrolled in the enrollments table then the student should be deleted form enrollments table too. Similar case if the student is a TA. |

**Stored Procedures:**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Procedure** | **Description** |
|  | show\_student | Display student details and records |
|  | show\_courses | Display courses details and records |
|  | Show\_TAs | Display TA Table info. |
|  | show\_classes | Display classes information |
|  | show\_enrollments | Display student enrollments information |
|  | show\_prerequisites | Display required prerequisites for a required course |
|  | show\_logs | Display the table log |
|  | ta\_info | Parameter passed classid: Display the B#, Fist and last name of the TA |
|  | get\_prerequisites | Parameters passed are Dept\_code and course#. Get the direct or indirect prerequisites for a particular course. |
|  | enroll\_student | Parameters passed are B# and classid. Student is enrolled by adding an entry in the enrollment table. |
|  | drop\_student | Parameters passed are B# and classid. Entry in the enrollments table is deleted. |
|  | del\_student | Parameter passed is B#. Student is deleted from student table along with TA table if the student is a TA. |

1. **JAVA METHODS:**

|  |  |
| --- | --- |
| METHODS | DESCRIPTION |
| enrollStudentClass() | To Enroll a student for a class |
| dropStudentClass() | Drop a student from Enrollment table |
| deleteStudent() | Delete a student from student table |
| infoPrerequisites() | Display information for the prerequisites |
| TAInfo() | Display TA information regarding B#, First name and last name |
| showTableInfo() | Select the table to be shown or displayed |
|  |  |

1. **PL/SQL code:**

*/\*To display the output on the console/terminal\*/*

set serveroutput on;

*/\*Drop triggers\*/*

Drop trigger enrollments\_insert;

Drop trigger enrollments\_delete;

Drop trigger student\_delete;

--Q.1.

--Sequence

--Done by: Varun Saxena

*/\*Create sequence starting with 100\*/*

DROP SEQUENCE logs\_seq;

CREATE SEQUENCE logs\_seq increment by 1 START WITH 100;

--Package Started

Create or replace package student\_registration AS

*/\*Declaration of Procedures\*/*

--Q.2.

procedure show\_students(student\_cursor out sys\_refcursor);

procedure show\_courses(student\_cursor out sys\_refcursor);

procedure show\_TAs(student\_cursor out sys\_refcursor);

procedure show\_classes(student\_cursor out sys\_refcursor);

procedure show\_enrollments(student\_cursor out sys\_refcursor);

procedure show\_prerequisites(student\_cursor out sys\_refcursor);

procedure show\_logs(student\_cursor out sys\_refcursor);

--Q.3.

procedure ta\_info(v\_classid in Classes.classid%type,error\_message out varchar2,r\_cursor out sys\_refcursor);

--Q.4.

procedure get\_prerequisites(v\_dept\_code in courses.dept\_code%type,v\_course# in courses.course#%type,error\_message out varchar2,r\_cursor out sys\_refcursor);

--Q.5.

procedure enroll\_student(v\_B# in students.B#%type,v\_classid in classes.classid%type,error\_message out varchar2);

--Q.6.

procedure drop\_student(dropB# in Students.B#%type,dropClassid in Classes.Classid%type,error\_message out varchar2);

--Q.7.

procedure del\_student(delB# IN Students.B#%type,error\_message out varchar2);

END;

/

*/\*PL-SQL Code\*/*

create or replace package body student\_registration AS

--Q.2.

--Done by: Varun Saxena

*/\*students table \*/*

procedure show\_students(student\_cursor out sys\_refcursor) AS

BEGIN

open student\_cursor for

select \* from students;

END;

*/\* courses table \*/*

procedure show\_courses(student\_cursor out sys\_refcursor) AS

BEGIN

open student\_cursor for

SELECT \* FROM COURSES;

END;

*/\* TAs table \*/*

procedure show\_TAs(student\_cursor out sys\_refcursor)

AS

BEGIN

open student\_cursor for

select \* from TAs;

End;

*/\* classes table \*/*

procedure show\_classes(student\_cursor out

sys\_refcursor) as

BEGIN

open student\_cursor for

select \* from classes;

END;

*/\* enrollments table \*/*

procedure show\_enrollments(student\_cursor out sys\_refcursor) AS

BEGIN

open student\_cursor for

select \* from enrollments;

END;

*/\* prerequisites table \*/*

procedure show\_prerequisites(student\_cursor out sys\_refcursor) AS

BEGIN

open student\_cursor for

select \* from prerequisites;

END;

*/\* logs table \*/*

procedure show\_logs(student\_cursor out sys\_refcursor) AS

BEGIN

open student\_cursor for

select \* from logs;

END;

--Q.3.

--Done by: Varun Saxena

*/\*procedure in the package to list the B#, the first name and last name of the TA of the class. \*/*

procedure ta\_info (v\_classid in Classes.classid%type,error\_message out varchar2,r\_cursor out sys\_refcursor) is v\_data\_found\_classid Number;

v\_data\_found\_TA Number;

Begin

SELECT count(\*) into v\_data\_found\_classid from Classes WHERE classid = v\_classid;

select count(\*) into v\_data\_found\_TA FROM TAs,Classes WHERE TAs.B# = Classes.TA\_B# AND Classes.classid = v\_classid;

if (v\_data\_found\_classid = 0) THEN

error\_message := 'The classid is invalid';

else

if v\_data\_found\_TA = 0 then

error\_message := 'The class has no TA';

else

open r\_cursor for

select students.B#,students.first\_name,students.last\_name

FROM Students

JOIN TAs ON Students.B# = TAs.B#

JOIN Classes ON TAs.B# = Classes.TA\_B#

WHERE Classes.classid = v\_classid;

end if;

end if;

end;

--Q.4.

--Done by: Varun Saxena

*/\* procedure in the package that return all its prerequisites course \*/*

procedure get\_prerequisites(v\_dept\_code in courses.dept\_code%type,v\_course# in courses.course#%type,error\_message out varchar2,r\_cursor out sys\_refcursor) is

v\_found\_dept\_code\_course# Number;

cursor prereq\_cursor is

select pre\_dept\_code,pre\_course# from prerequisites

where dept\_code = v\_dept\_code

and course# = v\_course#;

prereq\_record prereq\_cursor%rowtype;

BEGIN

SELECT count(\*) into v\_found\_dept\_code\_course# FROM Courses WHERE

dept\_code = v\_dept\_code and course# = v\_course#;

if (v\_found\_dept\_code\_course# = 0) THEN

error\_message := v\_dept\_code||v\_course#||' is invalid';

else

insert into temp\_prerequisites select pre\_dept\_code,pre\_course# from

prerequisites where dept\_code = v\_dept\_code and course# =

v\_course#;

open prereq\_cursor;

loop

fetch prereq\_cursor into prereq\_record;

exit when prereq\_cursor%notfound; get\_prerequisites(prereq\_record.pre\_dept\_code,prereq\_record.pre\_course#,error\_message,r\_cursor);

end loop;

open r\_cursor for select \* from temp\_prerequisites;

close prereq\_cursor;

end if;

end;

--Q.5.

--Done by: Kundan Shrivastav

*/\*Procedure in the package to enroll a student in the class \*/*

procedure enroll\_student(v\_B# in students.B#%type,

v\_classid in classes.classid%type,error\_message out varchar2) is

v\_student\_B# Number;

v\_student\_classid Number;

v\_class\_sem Number;

v\_student\_in\_sem Number;

v\_capacity Number;

v\_student\_overloaded Number;

v\_count\_prereqs Number;

v\_count\_classid\_prereqs Number;

Begin

Select count(\*) into v\_student\_B# from Students where B# = v\_B#;

Select count(\*) into v\_student\_classid from Classes where classid = v\_classid;

if (v\_student\_classid > 0) then

select count(\*) into v\_class\_sem from classes where classid = v\_classid

and year = 2018 and semester = 'Fall';

select LIMIT-class\_size into v\_capacity from classes where classid =

v\_classid;

end if;

Select count(\*) into v\_student\_in\_sem from enrollments where B# = v\_B#

and classid = v\_classid;

Select count(\*) into v\_student\_overloaded from enrollments e,classes c

where e.B# = v\_B# and e.classid = c.classid and c.year = 2018 and c.semester = 'Fall';

Select count(\*) into v\_count\_prereqs from prerequisites where

(dept\_code,course#) in (Select dept\_code,course# from classes where classid = v\_classid);

Select count(classid) into v\_count\_classid\_prereqs from enrollments where lgrade <= 'C'

and B# = v\_B# and classid in (Select classid from classes where (dept\_code,course#) in

(Select pre\_dept\_code,pre\_course# from prerequisites where (dept\_code,course#) in (Select

dept\_code,course# from classes where classid = v\_classid)));

if (v\_student\_B# = 0) then

error\_message := 'The B# is invalid';

elsif (v\_student\_classid = 0) then

error\_message := 'The classid is invalid';

elsif (v\_class\_sem = 0) then

error\_message := 'Cannot enroll into a class from a previous semester';

elsif (v\_capacity = 0) then

error\_message := 'The class is already full';

elsif (v\_student\_in\_sem <> 0) then

error\_message := 'The student is already in the class';

elsif (v\_count\_prereqs <> v\_count\_classid\_prereqs) then

error\_message := 'Prerequisite not satisfied';

elsif (v\_student\_overloaded = 4) then

error\_message := 'The student will be overloaded with the new

enrollment';

INSERT INTO Enrollments(B#,classid) VALUES (v\_B#,v\_classid);

elsif (v\_student\_overloaded > 4) then

error\_message := 'Students cannot be enrolled in more than five classes

in the same semester';

else

INSERT into Enrollments(B#,classid) VALUES (v\_B#,v\_classid);

end if;

end;

--Q.6.

--Done by: Sean Gallagher

*/\* procedure in the package to drop a student from a class \*/*

procedure drop\_student(

dropB# in Students.B#%type,

dropClassid in Classes.Classid%type,error\_message out varchar2) IS

--Local declarations

count\_B# Students.B#%type;

count\_Classid Classes.Classid%type;

count\_Enrollment Enrollments.B#%type;

tempSemester Classes.Semester%type;

tempYear Classes.Year%type;

dCode Classes.DEPT\_CODE%type;

c# Classes.Course#%type;

countPre Number;

newSize Classes.Class\_size%type;

numClasses Number;

BEGIN

SELECT count(\*)

INTO count\_B# FROM Students WHERE B# = dropB#;

SELECT count(\*)

INTO count\_Classid FROM Classes WHERE Classid = dropClassid;

SELECT count(\*)

INTO count\_Enrollment FROM Enrollments WHERE B# = dropB# and Classid = dropClassid;

IF (count\_B# = 0) THEN

error\_message := 'The B# is invalid';

ELSIF (count\_Classid = 0) THEN

error\_message := 'The classid is invalid';

ELSIF (count\_Enrollment = 0) THEN

error\_message := 'The student is not enrolled in the class';

ELSE

SELECT SEMESTER, YEAR

INTO tempSemester, tempYear FROM CLASSES WHERE Classid = dropClassid;

IF tempSemester != 'Fall' or tempYear != 2018 THEN

error\_message := 'Only enrollment in the current semester can be dropped.';

RETURN;

END IF;

SELECT DEPT\_CODE, COURSE#

INTO dCode, c# FROM CLASSES WHERE Classid = dropClassid;

SELECT count(DEPT\_CODE) INTO countPre

FROM PREREQUISITES WHERE DEPT\_CODE in

(SELECT DEPT\_CODE FROM CLASSES WHERE Classid in

(SELECT Classid FROM ENROLLMENTS WHERE B# = dropB#)) and

COURSE# in (SELECT COURSE# FROM Classes WHERE Classid in

(SELECT Classid FROM Enrollments WHERE B# = dropB#))

and PRE\_DEPT\_CODE = dCode and PRE\_COURSE# = c#;

IF countPre != 0 THEN

error\_message := 'The drop is not permitted because another class the student registered uses it as a prerequisite.';

RETURN;

END IF;

DELETE FROM Enrollments WHERE B# = dropB# and Classid = dropClassid;

SELECT class\_size INTO newSize

FROM Classes WHERE Classid = dropClassid;

IF newSize = 0 THEN

error\_message := 'The class has no students';

END IF;

SELECT COUNT(Classid) into numClasses

FROM Enrollments WHERE B# = dropB#;

IF numClasses = 0 THEN

error\_message := 'This student is not enrolled in any classes';

END IF;

END IF;

END;

--Q.7.

--Done by: Sean Gallagher

*/\* procedure in the package to delete a student from the Students table \*/*

procedure del\_student(

delB# IN Students.B#%type,error\_message out varchar2) IS

--local

count\_B# Students.B#%type;

BEGIN

SELECT COUNT(\*) into count\_B# FROM Students Where B# = delB#;

IF (count\_B# = 0) THEN

error\_message := 'The B# is invalid';

ELSE

DELETE Students WHERE B# = delB#;

Commit;

END IF;

END;

END;

/

show errors;

--Q.8.

--Trigger

--Created by: Kundan Shrivastav

*/\* triggers to add tuples to the Logs table \*/*

*/\*Enrollment Insert\*/*

create or replace trigger enrollments\_insert

after insert on enrollments

for each row

Declare

user\_log varchar2(20);

operation\_log varchar2(20) default 'insert';

key\_value\_log varchar2(50);

B#\_log enrollments.B#%type;

classid\_log enrollments.classid%type;

table\_name\_log nvarchar2(20) default 'enrollments';

id\_log Number;

Begin

B#\_log := :new.B#;

classid\_log := :new.classid;

key\_value\_log := (B#\_log||','||classid\_log);

id\_log := logs\_seq.nextval;

select user into user\_log from dual;

Insert into logs

values(id\_log,user\_log,sysdate,table\_name\_log,operation\_log,key\_value\_log);

Update classes

set class\_size = class\_size+1

where classid = classid\_log;

End;

/

*/\*Enrollment Delete\*/*

create or replace trigger ENROLLMENTS\_DELETE

AFTER DELETE ON Enrollments

FOR EACH ROW

DECLARE

user\_log varchar2(20);

operation\_log varchar2(20) default 'delete';

key\_value\_log varchar2(50);

B#\_log enrollments.B#%type;

classid\_log enrollments.classid%type;

table\_name\_log nvarchar2(20) default 'enrollments';

id\_log Number;

BEGIN

B#\_log := :old.B#;

classid\_log := :old.classid;

key\_value\_log := (B#\_log||','||classid\_log);

id\_log := logs\_seq.nextval;

select user into user\_log from dual;

Insert into logs

values(id\_log,user\_log,sysdate,table\_name\_log,operation\_log,key\_value\_log);

Update classes

set class\_size = class\_size-1

where classid = classid\_log;

END;

/

*/\*Student Delete\*/*

create or replace trigger STUDENT\_DELETE

AFTER DELETE ON Students

FOR EACH ROW

DECLARE

user\_log varchar2(20);

operation\_log varchar2(20) default 'delete';

B#\_log enrollments.B#%type;

table\_name\_log nvarchar2(20) default 'students';

id\_log Number;

BEGIN

B#\_log := :old.B#;

id\_log := logs\_seq.nextval;

select user into user\_log from dual;

Insert into logs

values(id\_log,user\_log,sysdate,table\_name\_log,operation\_log,B#\_log);

Delete From Enrollments Where B# = B#\_log;

UPDATE Classes SET TA\_B# = NULL WHERE TA\_B# = B#\_log;

DELETE FROM TAs WHERE B# = B#\_log;

END;

/

show errors;

1. **JAVA CODE:**

|  |  |
| --- | --- |
|  |  |
|  |  | Import java.io.BufferedReader;  import java.io.InputStreamReader; |
|  |  | import java.sql.CallableStatement; |
|  |  | import java.sql.Connection; |
|  |  | import java.sql.ResultSet; |
|  |  | import java.sql.SQLException; |
|  |  | import java.sql.Statement; |
|  |  | import java.sql.Types; |
|  |  | import java.util.\*; |
|  |  | import oracle.jdbc.OracleCallableStatement; |
|  |  | import oracle.jdbc.OracleTypes; |
|  |  | import oracle.jdbc.pool.OracleDataSource; |
|  |  |  |
|  |  | class DeleteStudent{ |
|  |  | public static void deleteStudent(Connection conn) { |
|  |  | try |
|  |  | { |
|  |  | BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); |
|  |  | System.out.println("Student Bno: "); |
|  |  | String Bno = br.readLine(); |
|  |  |  |
|  |  | CallableStatement stmt = conn.prepareCall("BEGIN student\_registration.del\_student(?,?); END;"); |
|  |  | stmt.setString(1,Bno); |
|  |  | stmt.registerOutParameter(2, java.sql.Types.VARCHAR); |
|  |  | stmt.execute(); |
|  |  |  |
|  |  | String err\_msg = ((OracleCallableStatement)stmt).getString(2); |
|  |  |  |
|  |  | if(err\_msg == null){ |
|  |  | System.out.println("\nStudent deleted successfully."); |
|  |  | } |
|  |  | else{ |
|  |  | System.out.println(err\_msg); |
|  |  |  |
|  |  | } |
|  |  | stmt.close(); |
|  |  | } |
|  |  | catch (Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | class DropStudentClass{ |
|  |  | public static void dropStudentClass(Connection conn) { |
|  |  | try |
|  |  | { |
|  |  | BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); |
|  |  | System.out.println("Student B#: "); |
|  |  | String Bno = br.readLine(); |
|  |  | System.out.println("Enter Class ID: "); |
|  |  | String classid = br.readLine(); |
|  |  |  |
|  |  | CallableStatement stmt = conn.prepareCall("BEGIN student\_registration.drop\_student(?,?,?); END;"); |
|  |  | stmt.setString(1,Bno); |
|  |  | stmt.setString(2,classid); |
|  |  | stmt.registerOutParameter(3, java.sql.Types.VARCHAR); |
|  |  | stmt.execute(); |
|  |  |  |
|  |  | String err\_msg = ((OracleCallableStatement)stmt).getString(3); |
|  |  | if(err\_msg == null){ |
|  |  | System.out.println("\nStudent dropped from the course successfully."); |
|  |  | } |
|  |  | else{ |
|  |  | System.out.println(err\_msg); |
|  |  | } |
|  |  |  |
|  |  | stmt.close(); |
|  |  | } |
|  |  | catch (Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | class Enrollments{ |
|  |  | public static void enrollStudentClass(Connection conn) { |
|  |  | try |
|  |  | { |
|  |  | BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); |
|  |  | System.out.println("Student B#: "); |
|  |  | String Bno = br.readLine(); |
|  |  | System.out.println("Enter Class ID: "); |
|  |  | String classid = br.readLine(); |
|  |  |  |
|  |  | CallableStatement stmt = conn.prepareCall("BEGIN student\_registration.enroll\_student(?,?,?); END;"); |
|  |  | stmt.setString(1,Bno); |
|  |  | stmt.setString(2,classid); |
|  |  | stmt.registerOutParameter(3, java.sql.Types.VARCHAR); |
|  |  | stmt.execute(); |
|  |  |  |
|  |  | String err\_msg = ((OracleCallableStatement)stmt).getString(3); |
|  |  | if(err\_msg == null){ |
|  |  | System.out.println("\nStudent enrolled into course successfully."); |
|  |  | } |
|  |  | else{ |
|  |  | System.out.println(err\_msg); |
|  |  | } |
|  |  |  |
|  |  | stmt.close(); |
|  |  |  |
|  |  | } |
|  |  | catch (Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | class Prerequisites{ |
|  |  | public static void infoPrerequisites(Connection conn) { |
|  |  | try |
|  |  | { |
|  |  | BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); |
|  |  | System.out.println("Enter Dept Code: "); |
|  |  | String dept\_code = br.readLine(); |
|  |  | System.out.println("Enter Course No: "); |
|  |  | String course\_no = br.readLine(); |
|  |  |  |
|  |  | CallableStatement stmt = conn.prepareCall("begin student\_registration.get\_prerequisites(?,?,?,?); end;"); |
|  |  | stmt.setString(1, dept\_code); |
|  |  | stmt.setInt(2, Integer.parseInt(course\_no)); |
|  |  | stmt.registerOutParameter(3,java.sql.Types.VARCHAR); |
|  |  | stmt.registerOutParameter(4,OracleTypes.CURSOR); |
|  |  |  |
|  |  | stmt.execute(); |
|  |  |  |
|  |  | ResultSet rs = null; |
|  |  | try{ |
|  |  | rs = ((OracleCallableStatement)stmt).getCursor(4); |
|  |  | } |
|  |  | catch(Exception ex){ |
|  |  | String err\_msg = ((OracleCallableStatement)stmt).getString(3); |
|  |  | System.out.println(err\_msg); |
|  |  | } |
|  |  |  |
|  |  | if(rs != null){ |
|  |  | System.out.println("\n\nCOURSE"); |
|  |  | while (rs.next()) { |
|  |  | System.out.println(rs.getString(1) + rs.getInt(2)); |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | String TruncateTable = "Truncate table temp\_prerequisites"; |
|  |  | Statement stmt1 = conn.createStatement(); |
|  |  | stmt1.executeQuery(TruncateTable); |
|  |  |  |
|  |  | } |
|  |  | catch(Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | class TAInfo{ |
|  |  | public static void infoTA(Connection conn) |
|  |  | { |
|  |  | try |
|  |  | { |
|  |  | BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); |
|  |  | System.out.println("Enter classid: "); |
|  |  | String classid = br.readLine(); |
|  |  |  |
|  |  | CallableStatement stmt = conn.prepareCall("begin student\_registration.ta\_info(?,?,?); end;"); |
|  |  | stmt.setString(1,classid); |
|  |  | stmt.registerOutParameter(2,java.sql.Types.VARCHAR); |
|  |  | stmt.registerOutParameter(3,OracleTypes.CURSOR); |
|  |  | stmt.execute(); |
|  |  | ResultSet rs = null; |
|  |  | try{ |
|  |  | rs = ((OracleCallableStatement)stmt).getCursor(3); |
|  |  | } |
|  |  | catch(Exception ex){ |
|  |  | String err\_msg = ((OracleCallableStatement)stmt).getString(2); |
|  |  | System.out.println(err\_msg); |
|  |  | } |
|  |  | if(rs != null){ |
|  |  | while (rs.next()) { |
|  |  | System.out.println(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3)); |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | catch(Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | class ShowTable{ |
|  |  | public static void showTableInfo(int choice, Connection conn) |
|  |  | { |
|  |  | switch(choice) |
|  |  | { |
|  |  | case 1: |
|  |  | { |
|  |  | try |
|  |  | { |
|  |  | CallableStatement stmt = conn.prepareCall("BEGIN student\_registration.show\_students(?); END;"); |
|  |  | stmt.registerOutParameter(1, OracleTypes.CURSOR); |
|  |  | stmt.execute(); |
|  |  | ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1); |
|  |  |  |
|  |  | while (rs.next()) |
|  |  | { |
|  |  |  |
|  |  | System.out.format("%-4s %-15s %-15s %-10s %.2f %-20s %-15s %-6s\n",rs.getString(1),rs.getString(2),rs.getString(3),rs.getString(4),rs.getDouble(5),rs.getString(6),rs.getString(7).substring(0,11),rs.getString(8)); |
|  |  |  |
|  |  | } |
|  |  | rs.close(); |
|  |  | } |
|  |  | catch (Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 2: |
|  |  | { |
|  |  | try |
|  |  | { |
|  |  | CallableStatement stmt = conn.prepareCall("BEGIN student\_registration.show\_courses(?); END;"); |
|  |  | stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF CURSOR |
|  |  | stmt.execute(); |
|  |  | ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1); |
|  |  | while (rs.next()) |
|  |  | { |
|  |  | System.out.println(rs.getString(1)+"\t" |
|  |  | +rs.getInt(2)+"\t" |
|  |  | +rs.getString(3)); |
|  |  | } |
|  |  | rs.close(); |
|  |  | } |
|  |  |  |
|  |  | catch (Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 3: |
|  |  | { |
|  |  | try |
|  |  | { |
|  |  | CallableStatement stmt = conn.prepareCall("BEGIN student\_registration.show\_TAs(?); END;"); |
|  |  | stmt.registerOutParameter(1, OracleTypes.CURSOR); |
|  |  | stmt.execute(); |
|  |  | ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1); |
|  |  |  |
|  |  | while (rs.next()) |
|  |  | { |
|  |  | System.out.println(rs.getString(1) + "\t" |
|  |  | + rs.getString(2) + "\t" |
|  |  | + rs.getString(3)); |
|  |  | } |
|  |  | rs.close(); |
|  |  | } |
|  |  |  |
|  |  | catch (Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 4: |
|  |  | { |
|  |  | try |
|  |  | { |
|  |  | CallableStatement stmt = conn.prepareCall("BEGIN student\_registration.show\_classes(?); END;"); |
|  |  | stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF CURSOR |
|  |  | stmt.execute(); |
|  |  | ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1); |
|  |  | while (rs.next()) |
|  |  | { |
|  |  | System.out.println(rs.getString(1)+"\t" |
|  |  | + rs.getString(2)+"\t" |
|  |  | + rs.getInt(3)+"\t" |
|  |  | + rs.getInt(4)+"\t" |
|  |  | + rs.getInt(5)+"\t" |
|  |  | + rs.getString(6)+"\t" |
|  |  | + rs.getInt(7)+"\t" |
|  |  | + rs.getInt(8)+"\t" |
|  |  | + rs.getString(9)+"\t" |
|  |  | + rs.getString(10)); |
|  |  | } |
|  |  | rs.close(); |
|  |  | } |
|  |  | catch (Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 5: |
|  |  | { |
|  |  | try |
|  |  | { |
|  |  | CallableStatement stmt = conn.prepareCall("BEGIN student\_registration.show\_enrollments(?); END;"); |
|  |  | stmt.registerOutParameter(1, OracleTypes.CURSOR); |
|  |  | stmt.execute(); |
|  |  | ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1); |
|  |  | while (rs.next()) |
|  |  | { |
|  |  | System.out.println(rs.getString(1)+"\t" |
|  |  | + rs.getString(2)+"\t" |
|  |  | + rs.getString(3)); |
|  |  | } |
|  |  | rs.close(); |
|  |  | } |
|  |  | catch (Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 6: |
|  |  | { |
|  |  | try |
|  |  | { |
|  |  | CallableStatement stmt = conn.prepareCall("BEGIN student\_registration.show\_prerequisites(?); END;"); |
|  |  | stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF CURSOR |
|  |  | stmt.execute(); |
|  |  | ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1); |
|  |  | while (rs.next()) |
|  |  | { |
|  |  | System.out.println(rs.getString(1)+"\t" |
|  |  | + rs.getInt(2)+"\t" |
|  |  | + rs.getString(3)+"\t" |
|  |  | + rs.getInt(4)); |
|  |  | } |
|  |  | rs.close(); |
|  |  | } |
|  |  | catch (Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 7: |
|  |  | { |
|  |  | try |
|  |  | { |
|  |  | CallableStatement stmt = conn.prepareCall("BEGIN student\_registration.show\_logs(?); END;"); |
|  |  | stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF CURSOR |
|  |  | stmt.execute(); |
|  |  | ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1); |
|  |  | while (rs.next()) |
|  |  | { |
|  |  | System.out.println(rs.getInt(1)+"\t" |
|  |  | + rs.getString(2)+"\t" |
|  |  | + rs.getString(3)+"\t" |
|  |  | + rs.getString(4)+"\t" |
|  |  | + rs.getString(5)+"\t" |
|  |  | + rs.getString(6)); |
|  |  | } |
|  |  | rs.close(); |
|  |  | } |
|  |  | catch (Exception e) |
|  |  | { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | break; |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | public class mainInterface { |
|  |  | public static void main(String args[]) throws SQLException { |
|  |  | try { |
|  |  | OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource(); |
|  |  | ds.setURL("jdbc:oracle:thin:@castor.cc.binghamton.edu:1521:ACAD111"); |
|  |  | Connection conn = ds.getConnection("kshriva1","Ilovefcrit1"); |
|  |  |  |
|  |  | while(true) |
|  |  | { |
|  |  | System.out.println(); |
|  |  | System.out.println("\*\*\*\*\*Main Menu\*\*\*\*\*"); |
|  |  |  |
|  |  | System.out.println("1.View Table data"); |
|  |  | System.out.println("2.View TA Information"); |
|  |  | System.out.println("3.View Prerequisites Information"); |
|  |  | System.out.println("4.Enroll a Student in Class"); |
|  |  | System.out.println("5.Drop a Student from Class"); |
|  |  | System.out.println("6.Delete a Student"); |
|  |  | System.out.println("7.Exit"); |
|  |  | int n = 0; |
|  |  | Scanner sc = new Scanner(System.in); |
|  |  | System.out.println("Please select an option from the above : "); |
|  |  | n = sc.nextInt(); |
|  |  |  |
|  |  | switch(n) |
|  |  | { |
|  |  | case 1: |
|  |  | { |
|  |  | ShowTable showTable = new ShowTable(); |
|  |  |  |
|  |  | System.out.println(); |
|  |  | System.out.println("\*\*\*Select Table\*\*\*"); |
|  |  | System.out.println("1.Students\n" |
|  |  | + "2.Courses\n" |
|  |  | + "3.TAs\n" |
|  |  | + "4.Classes\n" |
|  |  | + "5.Enrollments\n" |
|  |  | + "6.Prerequisites\n" |
|  |  | + "7.Logs\n"); |
|  |  | int m = 0; |
|  |  | try { |
|  |  | BufferedReader inputReader = new BufferedReader(new InputStreamReader(System.in)); |
|  |  | do |
|  |  | { |
|  |  | System.out.println("Enter Choice From Above Options"); |
|  |  | m = Integer.parseInt(inputReader.readLine()); |
|  |  | } |
|  |  | while(m < 1 || m > 7); |
|  |  | } |
|  |  | catch (Exception e) { |
|  |  | e.printStackTrace(); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | showTable.showTableInfo(m,conn); |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 2: |
|  |  | { |
|  |  | TAInfo ta = new TAInfo(); |
|  |  | ta.infoTA(conn); |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 3: |
|  |  | { |
|  |  | Prerequisites prerequisite = new Prerequisites(); |
|  |  | prerequisite.infoPrerequisites(conn); |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 4: |
|  |  | { |
|  |  | Enrollments enroll = new Enrollments(); |
|  |  | enroll.enrollStudentClass(conn); |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 5: |
|  |  | { |
|  |  | DropStudentClass drop = new DropStudentClass(); |
|  |  | drop.dropStudentClass(conn); |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 6: |
|  |  | { |
|  |  | DeleteStudent delStudent = new DeleteStudent(); |
|  |  | delStudent.deleteStudent(conn); |
|  |  | break; |
|  |  | } |
|  |  |  |
|  |  | case 7: |
|  |  | { |
|  |  | System.exit(1);; |
|  |  | break; |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | catch (Exception e) { |
|  |  | System.out.println("Connection not Established. Try Again"); |
|  |  | System.exit(1); |
|  |  | } |
|  |  | } |
|  |  | } |