CS532 Project 2
Implement Student Registration System using PL/SQL and JDBC

Submitted by:

Vishvas Patel

vpatel19@binghamton.edu

Akshay Shah

ashah38@binghamon.edu

Kushal Shinde

kshinde1@binghamton.edu

Keys

1. **Project Description**
2. **Project Implementation**
3. **Project Overflow**
4. **PL/SQL code – Sequence**
5. **PL/SQL code – Declaration of Procedures**
6. **PL/SQL code – Trigger for insert into enrollments**
7. **PL/SQL code – Trigger for drop a student from enrollments**
8. **PL/SQL code – Trigger for delete a student**
9. **PL/SQL code – Procedure to show all table information**
10. **PL/SQL code – Procedure to show student course details**
11. **PL/SQL code - Procedure to show the pre-requisites courses**
12. **PL/SQL code - Procedure to show the course and students details**
13. **PL/SQL code – Procedure to enroll a student**
14. **PL/SQL code – Procedure to drop a student from a class**
15. **PL/SQL code – Procedure to delete a student**

Project Description:

This project uses Oracle's PL/SQL and JDBC to create an application to support student registration system at a university.

Using procedures in PL/SQL, the different data of students is manipulated as per different requirements.

The data is allowed to be added and deleted from the tables.

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

Project Implementation:

SYS_REFCURSOR is used in this project to allow records to be returned from stored procedures and functions.

In this project, **logid** in Logs table is generated using sequence. The sequence starts generating Logid from 1000 and increments by 1 when new log records are inserted into the logs table.

Project Overflow created in this project:

Procedures:

1. **show_students:** This procedure is created to display all student information from the students table.
2. **show_courses**: This procedure is created to display all course information from the courses table
3. **show_course_credit**: This procedure is created to display all course credit information from the course_credit table
4. **show_classes**: This procedure is used to display the information about the classes in the classes table.
5. **show_enrollments**: This procedure is used to display the information about enrollments from the enrollments table
6. **show_grades**: This procedure is used to display all information about grads from the grades table
7. **show_prerequisites**: This procedure is used to display all information about prerequisites from the prerequisites table
8. **show_logs**: This procedure is used to display all information about logs from the logs table
9. **get_prerequisites**: Dept_code and course# are passed as an input parameter and return all courses that need this course as a prerequisite. It also shows indirect prerequisites
10. **get_students_course_details**: For a given student with B# as a parameter, can list every class the student has taken or is taking as well as the letter grade and number grade the student received for the class.
11. **get_class_course_student**: For a given class with classid as input can list the classid and course title of the class as well as all the students showing B# and firstname who took or are taking the class.
12. **enroll_student**: B# and classid are passed as input. Once the given B# and classid satisfies all the checks made, student will be enrolled by adding an entry in the enrollment table.
13. **drop_student**: B# and classid are passed as input. Once all the checks are satisfied, entry in the enrollment table is deleted.
14. **delete_student**: B# is passed as input. An entry from students table is deleted.

Triggers:

1. **enrollments_trigger_insert**: Trigger to do entry in the log table and the classes table is updated by incrementing the class size by 1 when there is a new entry in enrollments table.

2. **enrollments_trigger_delete**: Trigger for entry is made in the log table and the classes table is updated by decrementing the class size by 1 when the entry in the enrollments table is deleted.

3. **delete_student_trigger**: Trigger for entry in the enrollments table corresponding to the B# which is deleted in students table is deleted when the entry in the students table is deleted. As the entry from the enrollments table is deleted enrollments_trigger_delete after trigger is executed/triggered.

PL/SQL code:

1. **Sequence:**

```
create sequence log_sequence start with 1000;
/
/*It creates a sequence starting with 1000.*/
```

2. **Declaration of Procedures:**

```
create or replace package displaytables as
procedure show_students(stud_curs out sys_refcursor);

procedure show_courses(stud_curs out sys_refcursor);

procedure show_course_credit(stud_curs out sys_refcursor);

procedure show_classes(stud_curs out sys_refcursor);

procedure show_enrollments(stud_curs out sys_refcursor);

procedure show_grades(stud_curs out sys_refcursor);

procedure show_prerequisites(stud_curs out sys_refcursor);

procedure show_logs(stud_curs out sys_refcursor);


procedure get_prerequisites(dept_code_pre in prerequisites.dept_code%type, course#_pre in
prerequisites.course#%type, r_cursor out sys_refcursor );

procedure get_students_course_details(std_b# in students.b#%type,error_msg out varchar2,r_cursor out
sys_refcursor);

procedure get_class_course_student(std_classid in classes.classid%type,msg out varchar2,stud_curs out
sys_refcursor);

procedure drop_student(std_b# in students.b#%type,std_classid in classes.classid%type, msg out varchar );

procedure enroll_student(std_b# students.b#%type,std_classid classes.classid%type,msg out varchar2);

procedure delete_student(std_b# in students.b#%type,msg out varchar);
end;
/
```

3. **Trigger for insert into enrollments**

```
/*
Trigger to do entry in the log table and the classes table is updated by incrementing the class size by 1
when there is a new entry in enrollments table.
*/

create or replace trigger enrollments_trigger_insert
after insert on enrollments
for each row
declare
log_user varchar2(12);
log_table_name varchar2(12) default 'enrollments';
log_operation varchar2(12) default 'insert';
log_key_value varchar2(14);
log_b# char(4);
log_classid char(5);
log_concat varchar2(30);
begin
log_b# := :new.b#;
log_classid := :new.classid;
log_concat := (log_b#||','||log_classid);
select user into log_user from dual;
log_key_value:=log_sequence.nextval;
 insert into logs values(log_key_value,log_user,sysdate,log_table_name,log_operation,log_concat);
 update classes set class_size=class_size+1 where classid=log_classid;
 end;
/
```

4. **Trigger for drop a student from enrollment**

```
/*
Trigger for entry is made in the log table and the classes table is updated by decrementing the class size by 1
when the entry in the enrollments table is deleted.
*/

create or replace trigger enrollments_trigger_delete
after delete on enrollments
for each row
declare
log_user varchar2(12);
log_table_name varchar2(12) default 'enrollments';
log_operation varchar2(12) default 'delete';
log_key_value varchar2(14);
log_b# char(4);
```

```
log_classid char(5);
log_concat varchar2(30);

begin
log_b# := :old.b#;
log_classid:= :old.classid;
log_concat := (log_b#||','||log_classid);

select user into log_user from dual;
log_key_value:=log_sequence.nextval;
 insert into logs values(log_key_value,log_user,sysdate,log_table_name,log_operation,log_concat);
 update classes set class_size=class_size-1 where classid=log_classid;
 end;
/
```

5. **Trigger for delete a student**

```
/*

Trigger for entry in the enrollments table corresponding to the B# which is deleted in students table is deleted when the entry in the students table is deleted.  As the entry from the enrollments table is deleted enrollments_trigger_delete is executed/triggered.

*/


create or replace trigger delete_student_trigger
after delete on students
for each row
declare
log_user varchar2(12);
log_table_name varchar2(12) default 'students';
log_operation varchar2(12) default 'delete';
log_key_value varchar2(14);
log_b# char(4);
begin
log_b# := :old.b#;
select user into log_user from dual;
log_key_value:=log_sequence.nextval;
  insert into logs values(log_key_value,log_user,sysdate,log_table_name,log_operation,log_b#);
  delete from enrollments where b#=log_b#;

  end;
/
```

6. Procedure to show all table information

```
/* lists the students table */
   procedure show_students(stud_curs out sys_refcursor) as
   begin
   open stud_curs for
   select * from students;
   end show_students;

   /*lists the courses table */
   procedure show_courses(stud_curs out sys_refcursor) as
   begin
   open stud_curs for
   select * from courses;
   end show_courses;

   /*lists the courses table */
   procedure show_course_credit(stud_curs out sys_refcursor) as
   begin
   open stud_curs for
   select * from course_credit;
   end show_course_credit;

   /*lists the classes table */
   procedure show_classes(stud_curs out sys_refcursor) as
   begin
   open stud_curs for
   select * from classes;
   end show_classes;

   /*lists the enrollments table */
   procedure show_enrollments(stud_curs out sys_refcursor) as
   begin
   open stud_curs for
   select b#,classid,nvl(lgrade,' ') from enrollments;
   end show_enrollments;

   /*list the grades table */
   procedure show_grades(stud_curs out sys_refcursor) as
   begin
   open stud_curs for
   select * from grades;
   end show_grades;

   /*list the pre-requisites table */
   procedure show_prerequisites(stud_curs out sys_refcursor) as
   begin
   open stud_curs for
   select * from prerequisites;
   end show_prerequisites;
```

```
  /*lists the logs table */
  procedure show_logs(stud_curs out sys_refcursor) as
  begin
  open stud_curs for
  select * from logs;
  end show_logs;
```

7. Procedure to show student course details

```
/*Procedure for showing student course details*/

procedure get_students_course_details(std_b# in students.b#%type,error_msg out varchar2,r_cursor out
sys_refcursor)
is
 student_count number;
 student_course_count number;
begin
 select count(*) into student_course_count from enrollments where b#=std_b#;
 select count(*) into student_count from students where b#= std_b#;
 if student_count = 0
  then error_msg := 'The B# is invalid';
 else
 if student_course_count = 0
then error_msg := 'The student has not taken any course.';
 else
 open r_cursor for
  select cl.dept_code, cl.course#, co.title, cl.sect#, cl.year, cl.semester, e.lgrade, g.ngrade
  from enrollments e inner join classes cl on cl.classid = e.classid left join grades g on g.lgrade = e.lgrade
  inner join courses co on co.dept_code||co.course# = cl.dept_code||cl.course# where e.b# = std_b#;

end if;
end if;
end get_students_course_details;
```

8. Procedure to show the pre-requisites courses

```
/* Procedure to show the pre-requisites courses for a given dept_code and course#*/

procedure get_prerequisites(dept_code_pre in prerequisites.dept_code%type, course#_pre in
prerequisites.course#%type, r_cursor out sys_refcursor )
is
  cursor pre_req_cursor is
  select pre_dept_code, pre_course# from prerequisites
  where dept_code = dept_code_pre
  and course#=course#_pre;

  pre_req_record pre_req_cursor%rowtype;

  begin
```

```
insert into prerequisites_temp select pre_dept_code,pre_course# from prerequisites where
dept_code=dept_code_pre and course#=course#_pre;
open pre_req_cursor;
loop
  fetch pre_req_cursor into pre_req_record;
  exit when pre_req_cursor%notfound;
  get_prerequisites(pre_req_record.pre_dept_code,pre_req_record.pre_course#,r_cursor);
end loop;
 open r_cursor for select * from prerequisites_temp;
 close pre_req_cursor;
end get_prerequisites;
```

9. Procedure to show the course and students details

```
/* Procedure to show the course and students details when a classid is provided  */
procedure get_class_course_student(std_classid in classes.classid%type,msg out varchar2,stud_curs out
sys_refcursor)
   is
   std_classid_count int;
   std_enroll_count int;
   begin
   select count(*) into std_classid_count from classes where classid=std_classid;
   select count(*) into std_enroll_count from enrollments where classid=std_classid;
   if std_classid_count>0
   then
      if std_enroll_count>0
      then
         open stud_curs for
      select cl.classid,cr.title,s.b#,s.firstname from students s,enrollments e,classes cl,courses cr
      where cl.classid=std_classid and cl.dept_code=cr.dept_code and cl.course#=cr.course# and s.b#=e.b# and
e.classid=cl.classid;
      else
      msg:='No student has enrolled in the class.';
      end if;
   else
      msg:='The classid is invalid';
      end if;
   end get_class_course_student;
```

10. Procedure to enroll a student

```
/* This procedure enrolls a student */

procedure enroll_student(std_b# students.b#%type,std_classid classes.classid%type,msg out varchar2)
is
std_b#_count number;
std_classid_count number;
std_stud_class_enrolled_count number;
std_classid_limit classes.limit%type;
std_classid_clsize classes.class_size%type;
std_year_classid classes.year%type;
std_sem_classid classes.semester%type;
std_over_count number;
std_prereq_count number;
std_enroll char(200);
temp_dept_code prerequisites.dept_code%type;
temp_course# prerequisites.course#%type;
temp_cursor_dept_code prerequisites.dept_code%type;
temp_cursor_course prerequisites.course#%type;
std_pre_temp_count int;
temp_cursor sys_refcursor;
begin
select count(*) into std_b#_count from students where b#=std_b#;
select count(*) into std_classid_count from classes where classid=std_classid;

if std_b#_count=0
then
msg:='The B# is invalid.';
else
if std_classid_count=0
then
msg:='The classid is invalid.';
else
select limit,class_size into std_classid_limit,std_classid_clsize from classes where classid=std_classid;
if (std_classid_clsize+1) > std_classid_limit
then
msg:='The class is full.';
else
select count(*) into std_stud_class_enrolled_count from enrollments where b#=std_b# and classid=std_classid;
if(std_stud_class_enrolled_count > 0)
then
msg:='The student is already in the class.';
else

select dept_code,course# into temp_dept_code,temp_course# from classes where classid=std_classid;

select count(*) into std_pre_temp_count from prerequisites where dept_code=temp_dept_code and
course#=temp_course#;

if std_pre_temp_count=0
```

```
then
select year,semester into std_year_classid,std_sem_classid from classes where classid=std_classid;
select count(*)into std_over_count from enrollments where b#=std_b# and classid in(select classid from classes
where year=std_year_classid and semester=std_sem_classid and classid<>std_classid);

 if std_over_count>=4
 then
 msg:='Students cannot be enrolled in more than four classes in the same semester.';
 else
 select year,semester into std_year_classid,std_sem_classid from classes where classid=std_classid;
 select count(*)into std_over_count from enrollments where b#=std_b# and classid in(select classid from classes
where year=std_year_classid and semester=std_sem_classid and classid<>std_classid);

if std_over_count=3
then
std_enroll:='You are overloaded';
/*inserting the b#,classid values into enrollments table*/
insert into enrollments(b#,classid) values(std_b#,std_classid);
msg:='Student with b#: '||std_b#||' has enrolled successfully'||'--'||std_enroll;
else
/*inserting the b#,classid values into enrollments table*/
insert into enrollments(b#,classid) values(std_b#,std_classid);
msg:='Student - '||std_b#||' has enrolled successfully';
end if;
end if;
else
select dept_code,course# into temp_dept_code,temp_course# from classes where classid=std_classid;
get_prerequisites(temp_dept_code,temp_course#,temp_cursor);

loop
fetch temp_cursor into temp_cursor_dept_code,temp_cursor_course;
exit when temp_cursor%notfound;
select count(*) into std_prereq_count from enrollments enr where enr.classid in (select classid from classes
where dept_code=temp_cursor_dept_code and course#=temp_cursor_course) and enr.lgrade is not null and
enr.lgrade!='E' and enr.lgrade!='F' and enr.lgrade!='D' and enr.lgrade!='C-' and enr.b#=std_b#;

end loop;

if std_prereq_count>0
then
select year,semester into std_year_classid,std_sem_classid from classes where classid=std_classid;
select count(*)into std_over_count from enrollments where b#=std_b# and classid in(select classid from classes
where year=std_year_classid and semester=std_sem_classid and classid<>std_classid);

if std_over_count>=4
then
msg:='Students cannot be enrolled in more than four classes in the same semester.';
else

select year,semester into std_year_classid,std_sem_classid from classes where classid=std_classid;
```

```
select count(*)into std_over_count from enrollments where b#=std_b# and classid in(select classid from classes
where year=std_year_classid and semester=std_sem_classid and classid<>std_classid);
if std_over_count=3
then
std_enroll:='You are overloaded';
insert into enrollments(b#,classid) values(std_b#,std_classid);
msg:='Student - '||std_b#||' has enrolled successfully'||'--'||std_enroll;
else
insert into enrollments(b#,classid) values(std_b#,std_classid);
msg:='Student - '||std_b#||' has enrolled successfully';
end if;
end if;
else
msg:='Prerequisite not satisfied.';
end if;
end if;
end if;
end if;
end if;
end if;
end enroll_student;
```

11. Procedure to drop a student from a class

```
/* Procedure to drop a student from class when B# and classid is given */

procedure drop_student(std_b# in students.b#%type,std_classid in classes.classid%type, msg out varchar )
is
std_count int;
std_class_cnt int;
std_class_enroll_cnt int;
std_dept_code classes.dept_code%type;
std_course# classes.course#%type;
std_prereq_cnt int;
std_class_size int;
std_class_size_cnt int;
msg1 varchar(50);
msg2 varchar(50);

begin
select count(*) into std_count from students where b# = std_b#;
if std_count=0
then
msg:='The B# is invalid';
else
select count(*) into std_class_cnt from classes where classid=std_classid;
if std_class_cnt=0
then
msg:='The classid is invalid';
else
select count(*) into std_class_enroll_cnt from enrollments where classid=std_classid and b#=std_b#;
if std_class_enroll_cnt=0
then
msg:= 'The student is not enrolled for any classes';
else
select dept_code,course# into std_dept_code,std_course# from classes where classid=std_classid;
select count(*) into std_prereq_cnt from classes cl,prerequisites p where classid in (select classid from
enrollments where classid<>std_classid and b#=std_b#) and cl.dept_code=p.dept_code and cl.course# =
p.course# and p.pre_dept_code=std_dept_code and p.pre_course#=std_course#;

if std_prereq_cnt>0
then
msg:='The drop is not permitted because another class uses it as a prerequisite';

else
delete from enrollments where b#=std_b# and classid=std_classid;
msg:='Student dropped successfully';

select class_size into std_class_size from classes where classid=std_classid;

select count(*) into std_class_size_cnt from enrollments where b#=std_b#;

if std_class_size=0
```

```
then
msg1:='The class now has no students.';
end if;

if std_class_size_cnt=0
then
msg2:='This student is not enrolled in any classes.';
end if;
msg:=msg||' '||msg1||' '||msg2;

end if;
end if;
end if;
end if;
end drop_student ;
```

12. Procedure to delete a student

```
/* procedure  to delete a student from the students table */
procedure delete_student(std_b# in students.b#%type,msg out varchar)
is
 std_count int;
begin
 select count(*) into std_count from students where b#=std_b#;
if std_count=0
then
 msg:='The B# is invalid.';
else
 delete from students where b#=std_b#;
 commit;
 msg:='student deleted sucessfully';
end if;
end delete_student;
```