create or replace package project2 as

type student is ref cursor;

type course1 is ref cursor;

type prerequisite is ref cursor;

type classe is ref cursor;

type enrollment is ref cursor;

type log1 is ref cursor;

type get\_course is ref cursor;

type emp is ref Cursor;

overloaded number(4);

check1 number(4);

check2 number(4);

check3 number(4);

check4 number(4);

check5 number(4);

check6 number(4);

check7 number(4);

use\_class\_size number(4);

use\_class\_limit number(4);

use\_year number(4);

use\_semester varchar2(10);

use\_course\_no number(4);

type student\_details is ref cursor;

function show\_students return student;

function show\_courses return course1;

function show\_pre return prerequisite;

function show\_classes return classe;

function show\_enrollments return enrollment;

function show\_logs return log1;

function query\_7(s\_id students.sid%type,c\_classid classes.classid%type) return number;

function query\_6(c\_classid classes.classid%type) return get\_course;

function query\_4(s\_id students.sid%type) return student\_details;

end;

Project Body

create or replace PACKAGE BODY PROJECT2 AS

function show\_students

return student as

showstud student;

begin

open showstud for

select \* from students;

return showstud;

end;

function show\_courses

return course1 as

course course1;

begin

open course for

select \* from courses;

return course;

end;

function show\_pre

return prerequisite as

showpre prerequisite;

begin

open showpre for

select \* from prerequisites;

return showpre;

end;

function show\_classes

return classe as

showclass classe;

begin

open showclass for

select \* from classes;

return showclass;

end;

function show\_enrollments

return enrollment as

showenroll enrollment;

begin

open showenroll for

select \* from enrollments;

return showenroll;

end;

function show\_logs

return log1 as

showlog log1;

begin

open showlog for

select \* from logs;

return showlog;

end;

function query\_6(c\_classid in classes.classid%type)

return get\_course as

g\_course get\_course;

cursor emp is

select \* from CLASSES where c\_classid=classid;

emp1 emp%rowtype;

begin

open emp;

if(emp%notfound)

then RAISE\_APPLICATION\_ERROR(-20002, 'The classid is invalid');

else

select count(\*) into check2 from Students s,classes cl,enrollments e,courses co

where s.sid=e.sid

and e.classid = cl.classid

and cl.dept\_code = co.dept\_code

and cl.course\_no = co.course\_no

and cl.classid = c\_classid;

if check2=0

then RAISE\_APPLICATION\_ERROR(-20007, 'No student is enrolled in the class');

else

open g\_course for

select cl.classid,co.title,co.title,cl.semester,cl.year,s.sid,s.lastname

from Students s, classes cl,courses co,enrollments e

where s.sid=e.sid

and e.classid = cl.classid

and cl.dept\_code = co.dept\_code

and cl.course\_no = co.course\_no

and cl.classid = c\_classid;

return g\_course;

end if;

end if;

end;

function query\_4(s\_id in students.sid%type)

return student\_details as

s\_details student\_details;

cursor emp is

select count(\*) INTO check1 from Students where s\_id=sid;

emp1 emp%rowtype;

begin

open emp;

if (emp%notfound)

then RAISE\_APPLICATION\_ERROR(-20001, 'The sid is invalid');

return null;

else

select count(\*) into check2 from Students s,classes cl,enrollments e,courses co

where s.sid=s\_id

and e.classid=cl.classid

and s.sid=e.sid

and co.course\_no = cl.course\_no;

if check2=0

then RAISE\_APPLICATION\_ERROR(-20006,'The student has not taken any course');

return null;

else

open s\_details for

select s.sid,s.lastname,s.status,e.classid,cl.dept\_code||cl.course\_no,co.title,cl.year,cl.semester

from students s,classes cl,courses co,enrollments e

where s.sid=s\_id

and e.classid=cl.classid

and cl.course\_no=co.course\_no

and e.sid=s.sid ;

return s\_details;

END IF;

END IF;

end;

function query\_7(s\_id in students.sid%type, c\_classid classes.classid%type)

return number as overloaded number;

cursor emp is

select count(\*) into check1 from students where students.sid=s\_id;

emp1 emp%rowtype;

begin

open emp;

if(emp%notfound)

then RAISE\_APPLICATION\_ERROR(-20001, 'The sid is invalid');

else

select count(\*) into check2 from classes where c\_classid = classes.classid;

if(check2 = 0)

then RAISE\_APPLICATION\_ERROR(-20002, 'The classid is invalid');

else

select classes.class\_size,classes.limit into use\_class\_size,use\_class\_limit from classes

where c\_classid=classid;

if (use\_class\_size = use\_class\_limit)

then RAISE\_APPLICATION\_ERROR(-20009, 'class\_size>limit');

else

select count(\*) into check3 from enrollments where enrollments.sid=s\_id

and c\_classid=enrollments.classid;

if(check3>0)

then RAISE\_APPLICATION\_ERROR(-20010, 'The student is already in the class');

else

select year,semester into use\_year,use\_semester from Classes where c\_classid=classid;

select count(\*) into check4 from enrollments e,classes cl where e.classid=cl.classid

and e.sid=s\_id and cl.year=use\_year and cl.semester=use\_semester;

if(check4=3)

then RAISE\_APPLICATION\_ERROR(-20011, 'Students cannot be enrolled in more than three classes in the same semester');

else

select course\_no into use\_course\_no from classes where c\_classid=classid;

select count(\*) into check5 from prerequisites

START WITH course\_no=use\_course\_no connect by course\_no = PRIOR pre\_course\_no;

if (check5>0)

then select count(\*) into check6 from enrollments e, classes cl

where e.classid = cl.classid

and e.sid=s\_id

and cl.classid in(select classid from classes where (dept\_code,course\_no) in(

select pre\_dept\_code,pre\_course\_no from prerequisites START WITH course\_no=use\_course\_no

connect by course\_no=PRIOR pre\_course\_no)) and e.lgrade in ('F','I');

if(check6>0)

then RAISE\_APPLICATION\_ERROR(-20012, 'Prerequisite courses have not been completed');

end if;

end if;

select count(\*) into check7 from enrollments e,classes cl where e.classid=cl.classid

and e.sid=s\_id and cl.year=use\_year and cl.semester=use\_semester;

if(check7 = 2)

then overloaded := 1;

else

overloaded:=0;

end if;

insert into enrollments values(s\_id,c\_classid,null);

commit;

return overloaded;

end if;

end if;

end if;

end if;

end if;

end;

END PROJECT2;

Procedures

create or replace Procedure insertStudents(

s\_id in students.sid%type,

s\_fname in students.firstname%type,

s\_lname in students.lastname%type,

s\_status in students.status%type,

s\_gpa in students.gpa%type,

s\_email in students.email%type)

IS

Begin

Insert INTO students ("SID","FIRSTNAME","LASTNAME","STATUS","GPA","EMAIL")

values (s\_id,s\_fname,s\_lname,s\_status,s\_gpa,s\_email);

end;

create or replace PROCEDURE QUERY8(s\_id in students.sid%type,

c\_classid in classes.classid%type)

is

check1 number(1);

check2 number(1);

check3 number(1);

check4 number(1);

check5 number(1);

check6 number(1);

use\_dept\_code classes.dept\_code%type;

use\_course\_no classes.course\_no%type;

use\_year classes.year%type;

use\_semester classes.semester%type;

prerequisite\_course\_no prerequisites.course\_no%type;

prerequisite\_dept\_code PREREQUISITES.DEPT\_CODE%type;

BEGIN

select count(\*) into check2 from Classes

where c\_classid=classes.classid;

if(check2=0)

then RAISE\_APPLICATION\_ERROR(-20002, 'The classid is invalid');

else

select cl.dept\_code,cl.course\_no,cl.year,cl.semester

into use\_dept\_code,use\_course\_no,use\_year,use\_semester

from classes cl

where c\_classid=classid;

select count(\*) into check1 from students

where s\_id=students.sid;

if(check1=0)

then RAISE\_APPLICATION\_ERROR(-20001, 'The sid is invalid');

else

select count(\*) into check3 from classes where c\_classid=classes.classid and classes.CLASS\_SIZE<1;

if(check3>0)

then RAISE\_APPLICATION\_ERROR(-20003, 'The class has no students');

else

select count(\*) into check4 from enrollments where c\_classid=classid and s\_id=sid;

if(check4 =0)

then RAISE\_APPLICATION\_ERROR(-20004, 'This students is not enrolled in any classes');

else

select count(\*) into check5 from Classes where classes.semester=use\_semester and classes.year=use\_year

and classid in(select cl.classid from classes cl,enrollments e

where e.sid=s\_id and cl.classid=e.classid);

select count(\*) into check6 from enrollments where classid in(

select classid from classes where (dept\_code,course\_no) IN (select

dept\_code,course\_no from prerequisites

where prerequisite\_dept\_code=use\_dept\_code

and prerequisite\_course\_no=use\_course\_no)) and s\_id=sid;

if(check6 = 0)

then RAISE\_APPLICATION\_ERROR(-20005, 'The drop is not permitted because another class uses it as a prerequisite');

else

if(check6 = 1)

then delete from enrollments where s\_id=sid and c\_classid=classid;

commit;

RAISE\_APPLICATION\_ERROR(-20008,'The class now has no students');

NULL;

end if;

end if;

end if;

end if;

end if;

end if;

END QUERY8;

create or replace PROCEDURE QUERY9(s\_id in students.sid%type)

is

check1 number(1);

size1 number(4);

BEGIN

Select count(\*) into check1 from Students where sid=s\_id;

if(check1 = 0)

then RAISE\_APPLICATION\_ERROR(-20001, 'The sid is invalid');

else

Delete from Students where students.sid=s\_id;

commit;

end if;

END QUERY9;

create or replace PROCEDURE show\_courses

IS

row courses%rowtype;

BEGIN

Select \* into row from courses;

END show\_courses;

Sequences

CREATE SEQUENCE "PLASH"."QUERY1" MINVALUE 1000 MAXVALUE 9999 INCREMENT BY 1 START WITH 1053 NOCACHE NOORDER NOCYCLE ;

Trigger

create or replace TRIGGER QUERY7

AFTER INSERT ON ENROLLMENTS

for each row

declare

size1 number(4);

BEGIN

select class\_size into size1 from classes where classid =:new.classid;

update classes set class\_size = size1+1 where classid =:new.classid;

END;

create or replace TRIGGER QUERY9TRIGGER

AFTER DELETE ON STUDENTS

for each row

BEGIN

delete from enrollments where sid= :old.sid;

END;

create or replace TRIGGER LOG\_TRIGGER

before INSERT ON LOGS

for each row

BEGIN

select query1.nextval

into :new.logid

from dual;

END;

create or replace TRIGGER QUERY10

AFTER INSERT ON STUDENTS

for each row

BEGIN

insert into logs(who,time,table\_name,operation,key\_value)

values (user,sysdate,'Students','insert',:new.sid);

END;

create or replace TRIGGER TRIGGER1

after DELETE ON STUDENTS

for each row

BEGIN

insert into logs(who,time,table\_name,operation,key\_value)

values (user,sysdate,'Students','delete',:old.sid);

END;

CREATE OR REPLACE TRIGGER TRIGGER2

AFTER INSERT ON ENROLLMENTS

for each row

declare

BEGIN

insert into logs(who,time,table\_name,operation,key\_value)

values (user,sysdate,'Enrollments','insert',:new.sid);

END;

CREATE OR REPLACE TRIGGER TRIGGER3

BEFORE DELETE ON ENROLLMENTS

for each row

declare

BEGIN

insert into logs(who,time,table\_name,operation,key\_value)

values (user,sysdate,'Enrollments','insert',:old.sid);

END;

create or replace TRIGGER TRIGGER4

after DELETE ON ENROLLMENTS

for each row

declare

size1 number(4);

BEGIN

select class\_size into size1 from classes where classid = :old.classid;

update classes set class\_size=size1-1 where classid= :old.classid;

END;