Advance Database Lab
Oracle Project Part 3

Done By:
Ayah Abu Hamra  Islam Abu Salem
220064486  220060769

Submitted To:
Eng. Doa'a Abu Jabel
For your college database created in part 1&2:

- Create a Package college that will contain all your work.

Package specification:

```sql
CREATE OR REPLACE PACKAGE college IS
    PROCEDURE Teachers_insert (id number,
                               First_Name varchar2,
                               Last_Name varchar2,
                               Head_ID Number);
    PROCEDURE Courses_insert (course_ID Number,
                               Course_Name varchar2,
                               course_year_number,
                               Number_of_Hours Number,
                               Description varchar2,
                               Teacher_id number);
    PROCEDURE departments_heads_insert (Department varchar2,
                                         Head_ID NUMBER,
                                         Head_Actions_ID Number);
    PROCEDURE students_insert (ID number,
                               First_Name varchar2,
                               Last_Name varchar2,
                               course_ID_number,
                               course_year_number,
                               number_of_hours_number,
                               grade number);
    PROCEDURE Teacher_Courses_insert (Teacher_id number,
                                       First_Name varchar2,
                                       Last_Name varchar2,
                                       Course_code varchar2,
                                       Course_Name varchar2,
                                       Course_Description varchar2);
    PROCEDURE Student_Courses_insert (stu_id number,
                                       Teacher_id number,
                                       Head_number,
                                       Student_Name varchar2,
                                       Teacher_Name varchar2,
                                       Course_code varchar2,
                                       Course_Name varchar2,
                                       Course_Description varchar2,
                                       grade number);
    PROCEDURE Tea_Course_Students_insert (Teacher_id number,
                                           Course_code varchar2,
                                           course_name varchar2,
                                           stu_id number,
                                           Student_Name varchar2,
                                           grade number);
    PROCEDURE Degree_update (New_Degree Number, Student_ID Number);
    PROCEDURE Teacher_update (Teacher_id number, id number);
    PROCEDURE Head_id_update (new_ID NUMBER, dep VARCHAR2);
    PROCEDURE Delete_teacher (First_name varchar2,
                               Last_Name varchar2);
    PROCEDURE Print_Teacher_Name;
    PROCEDURE Print_Students (ID NUMBER, cYear NUMBER, Hours NUMBER);
END college;
```
PROCEDURE Print_Students(Name varchar2);
-------------------------------
FUNCTION Teacher_info(teacher_id number) RETURN VARCHAR2;
-------------------------------
FUNCTION Course_description(code VARCHAR2) RETURN VARCHAR2;
-------------------------------
en end college;

Package Body:-
CREATE OR REPLACE PACKAGE BODY college IS

PROCEDURE Teachers_insert (id_number, 
First_Name varchar2, 
Last_name varchar2, 
Head_ID number) IS 
BEGIN 
insert into Teachers Values (id, First_Name, Last_name, Head_ID); 
end; 

-----------------------------

PROCEDURE Courses_insert (course_ID number, 
Course_Name varchar2, 
course_year number, 
Number_of_Hours number, 
Description varchar2, 
Teacher_id number) IS 
BEGIN 
insert into Courses 
values 
(course_ID, 
Course_Name, 
course_year, 
Number_of_Hours, 
Description, 
Teacher_id); 
end; 

-----------------------------

PROCEDURE departments_Hod Insert (Department varchar2, 
Head_ID number, 
Head_Assist_ID number) IS 
BEGIN 

insert into departments_Hod 
values 
(Department, Head_ID, Head_Assist_ID); 
end; 

-----------------------------

PROCEDURE students_insert (ID number, 
First_Name varchar2, 
Last_Name varchar2, 
course_ID number, 
course_year number, 
Number_of_Hours number) IS 
BEGIN 
insert into courses 
values 
(ID, 
First_Name, 
Last_Name, 
course_ID, 
course_year, 
Number_of_Hours); 
end; 

-----------------------------

PROCEDURE Teacher_Courses_insert (Teacher_ID number, 
Course_Name varchar2, 
Head_Name varchar2, 
course_code varchar2, 
course_description varchar2) IS 

insert number; 
"These Teachers First_ Lastname;";
begin
  index2 := instr(Teacher_Name, ' ');  
  Tlname := substr(Teacher_Name, 1, index2);  
  Lname := substr(Teacher_Name, index2);
  
cid := substr(Course_code, 1, 1);  
cyear := substr(Course_code, 2, 1);  
chours := substr(Course_code, 3, 1);

insert into teachers 
values(Teacher_id,  
        Tlname,  
        Lname,  
        head1);

insert into courses  
values  
  (cid,  
   Course_Name,  
   cyear,  
   chours,  
   Course_description,  
   Teacher_id);

end;

-------------------------------------------------------------------

PROCEDURE Student_Courses_insert(stu_id number,  
Teacher_id number,  
head number,  
Student_Name varchar2,  
Teacher_Name varchar2,  
Course_code varchar2,  
Course_Name varchar2,  
Course_description varchar2,  
grade number) is

  index1 number;
  index2 number;
  name Student.First_Name type;
  lname Student.Last_Name type;
  Tlname Teachers.First_Name type;
  Lname Teachers.Last_Name type;
  cid courses.Course_ID type;
  cyear courses.Course_Year type;
  chours courses.Number_0f_Hours type;

begin
  index1 := instr(Student_Name, ' ');  
  name := substr(Student_Name, 1, index1);  
  lname := substr(Student_Name, index1);  
  index2 := instr(Teacher_Name, ' ');  
  Tlname := substr(Teacher_Name, 1, index2);  
  Lname := substr(Teacher_Name, index2);
  
cid := substr(Course_code, 1, 1);  
cyear := substr(Course_code, 2, 1);  
chours := substr(Course_code, 3, 1);

insert into teachers 
values(Teacher_id,
values(Teacher_id,
    Teacher,
    Teacher_id);

insert into courses
values
(1,4,
    Course_Name,
    cpyear,
    course,
    Course_Description,
    Teacher_id);

insert into students
values
(1,10,
    Name,
    id1,
    cpyear,
    course,
    grade);

and:

PROCEDURE test_Course_Students_Insert(teacher_id NUMBER,
course_code VARCHAR2,
course_name VARCHAR2,
student_num VARCHAR2,
student_name VARCHAR2,
grade NUMBER);
```sql
Teacher_id;

insert into Students
values
('stu_id',
'fname',
 lname,
'd',
'cyen',
'chours',
'grade');

-----------------------------------------------------------------
PROCEDURE Degree_updater(Year_Degree_Number, Student_ID Number) IS
BEGIN
Update Students set grade = KeyDegree where ID = Student_ID;
end;
-----------------------------------------------------------------
PROCEDURE Teacher_update('teach_id_number', id_number) is
BEGIN
Update Courses set teacher_id = teach_id where course_ID = id;
end;
-----------------------------------------------------------------
PROCEDURE head_ID_updater(ID NUMBER, dep VARCHAR2) is
BEGIN
   SELECT head_ID into head FROM departments_heads WHERE Department = dep;
   DBMS_OUTPUT.PUT_LINE(head);
   update departments_heads set head_ID = Key_ID where Department = dep;
end;
-----------------------------------------------------------------
PROCEDURE Delete_teacher('teach_first_name varchar2',
'teach_last_name varchar2') is
   teach_F TEACHERS.FIRST_NAME TYPE;
   teach_L TEACHERS.LAST_NAME TYPE;
BEGIN
   SELECT FIRST_NAME, LAST_NAME into teach_F, teach_L
   FROM TEACHERS
   WHERE FIRST_NAME = teach_first_name
   and LAST_NAME = teach_last_name;
   DELETE FROM TEACHERS
   WHERE FIRST_NAME = teach_first_name
   and LAST_NAME = teach_last_name;
   DBMS_OUTPUT.PUT_LINE(teach_F);
   DBMS_OUTPUT.PUT_LINE(teach_L);
end;
-----------------------------------------------------------------
PROCEDURE Print_Teacher_Name is
   teach_F TEACHERS.FIRST_NAME TYPE;
   teach_L TEACHERS.LAST_NAME TYPE;
   counter NUMBER;
BEGIN
   counter := 0;
   SELECT First_Name, Last_Name
   INTO teach_F, teach_L
   FROM Teachers
   WHERE Head_ID IN (SELECT Head_ID FROM departments_heads);
   while counter < SQL%ROWCOUNT loop
      counter := counter + 1;
      DBMS_OUTPUT.PUT_LINE(teach_F);
      DBMS_OUTPUT.PUT_LINE(teach_L);
   end loop;
end;
```
PROCEDURE Print_Students(ID NUMBER, Class NUMBER, Hour NUMBER) IS
  std_std_Students, FIRST_NAME_TYPE;
  std_std_Students, LAST_NAME_TYPE;
  std_s grd_Students, Grade_TYPE;
  counter number;
  Current students_cursor IS
  SELECT First_Name, Last_Name, GRADE
  FROM Students
  WHERE Course_ID = ID
  AND Course_Year = cYear
  AND Number_of_Hours = Hours;
BEGIN
  open students_cursor;
  counter := 0;
  while counter < students_cursor_rowcount loop
    fetch students_cursor into std_0, std_l, grd;
    DBMS_OUTPUT.PUT_LINE(std_0);
    DBMS_OUTPUT.PUT_LINE(std_l);
    DBMS_OUTPUT.PUT_LINE(std_grd);
    counter := counter + 1;
  end loop;
END;

PROCEDURE Print_Students(Name varchar2) IS
  std_name Teacher_Course_Students, Student_Name_Type;
  std_c Teacher_Course_Students, Course_Code_Type;
  std_grd Teacher_Course_Students, Grade_Type;
  counter number;
  Current students_cursor IS
  SELECT Student_Name, Course_Code, Grade
  FROM Teacher_Course_Students
  WHERE 'Teacher_Name' = Name;
BEGIN
  open students_cursor;
  counter := 0;
  while counter < students_cursor_rowcount loop
    fetch students_cursor into std_name, std_c, std_grd;
    DBMS_OUTPUT.PUT_LINE(std_name);
    DBMS_OUTPUT.PUT_LINE(std_c);
    DBMS_OUTPUT.PUT_LINE(std_grd);
    counter := counter + 1;
  end loop;
END;

FUNCTION Teacher_info(teach_id number) RETURN VARCHAR2 IS
first Teachers, FIRST_NAME_TYPE;
last Teachers, LAST_NAME_TYPE;
BEGIN
  SELECT FIRST_NAME, LAST_NAME
  INTO first, last
  FROM Teachers
  WHERE ID = teach_id;
  DBMS_OUTPUT.PUT_LINE(first || ' ' || last);
  RETURN first || ' ' || last;
END;

FUNCTION Course_description(code VARCHAR2) RETURN VARCHAR2 IS
des Teacher_Course_Description, Type;
begin
  id number(1);
  cyear number(2);
  hours number(2);
BEGIN
  id := substring(code, 1, 1);
  cyear := substring(code, 2, 1);
  hours := substring(code, 3, 1);
END;
HERE: 

```sql
BEGIN

-- course description query
SELECT Description
FROM Course
WHERE Course_id = 123
AND Course_year = 2020;

RETURN 1;

-----------------------------------------------
```

BEGIN

```sql
college.Teacher_Term(12, 'Fall', '2021', 1);
and college;
```
Procedures Test:

1. Teachers_insert

Before calling "Teacher_insert" procedure.

calling "Teacher_insert" procedure.

After calling "Teacher_insert" procedure.
Courses_insert

before calling "course_insert" procedure.

calling "course_insert" procedure.

After calling "course_insert" procedure.
3. \texttt{departments_heads_insert}

Before calling "department_heads_insert" procedure.

After calling "department_heads_insert" procedure.
4. students_insert

```
BEGIN
    college.students_insert(6,'Islam','Abu-Salem',12,3,3,95);
END;
```

```
id number(5) mandatory;
```
5. Teacher_Courses_insert2

```sql
BEGIN
college.Teacher_Courses_insert(501, 'Usama Abunabili', 423, 'Advance DB', 'Advance data base lab');
END;
```

```
<table>
<thead>
<tr>
<th>Teacher Name</th>
<th>Course Code</th>
<th>COURSE NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ali Wessm</td>
<td>423</td>
<td>Advance DB</td>
<td>Advance data base lab</td>
</tr>
<tr>
<td>Aysabined</td>
<td>423</td>
<td>Advance DB</td>
<td>Advance data base lab</td>
</tr>
<tr>
<td>Hani Cebab</td>
<td>423</td>
<td>Advance DB</td>
<td>Advance data base lab</td>
</tr>
<tr>
<td>Hame Aqada</td>
<td>425</td>
<td>Advance DB</td>
<td>Advance data base lab</td>
</tr>
<tr>
<td>Phats Abudam</td>
<td>423</td>
<td>Advance DB</td>
<td>Advance data base lab</td>
</tr>
</tbody>
</table>
```
6. Tea_Course_Students_insert2

```sql
BEGIN
  college.Tea_Course_Students_insert2('SO', 'Ruba Abusalem', '521', 'MAFE', 88, 'ABC www', 55);
END;
```

![SQL Output](image1)

```sql
select * from Teacher_Course_Students;
```

![SQL Table](image2)
7. Degree_update

```
BEGIN
college.Degree_update(60,2);
END;
```
8. Teacher_update
9. Head_id_update

![SQL query]

```sql
select * from departments_heads;
```

![Database output]

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>HEAD_ID</th>
<th>HEAD_ASSIST_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>programming</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

![SQL query]

```sql
begin
college.Head_id_update(2,'programming');
end;
```

![Database output]

![SQL query]

```sql
select * from departments_heads;
```

![Database output]

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>HEAD_ID</th>
<th>HEAD_ASSIST_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>programming</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

1:33 2 rows selected in 0.032 seconds
10. Delete_teacher
11. Print_Students

```
begin
    college.Print_Students('Islam Salem');
end;
```

Done in 0.015 seconds

12. Teacher_info

```
begin
    college.Print_Students(15, 5, 4);
end;
```

Done in 0.031 seconds

```
Declare
    result varchar2(30);begin
    result:=college.Teacher_Info(6);
end;
```

Done in 0.015 seconds
13. Course_description

```
DECLARE result VARCHAR2(50);
BEGIN
result := college.Course_description('313');
END;
```

```
The description of the course code 313 is os os
```
Triggers Part:

Logon and logoff triggers

Create logon_trigger that add "user_id, log_date and action('Logging on')" on log_trig_table

- First, we have to create log_trig_table

- logon_trigger

Create logoff_trigger that add "user_id, log_date and action('Logging off')" on log_trig_table.
- The Result of the two previous triggers

Foreign Keys Triggers

1- Head_Assistant_ID in department_heads table is a foreign key for ID in the teacher table.

To test the trigger we will update the assistant id to 10 which is not exists in the teachers table, so the trigger will be fired.
Here the teachers table after firing the trigger

![SQL Window - SELECT * FROM DEPARTMENTS](image1)

![SQL Window - SELECT * FROM TEACHERS](image2)

Here we will again update the assistant id to 4 which is in the teachers id so the trigger do nothing (NULL)

![SQL Window - UPDATE DEPARTMENTS...](image3)

No change on the teachers table

![SQL Window - SELECT * FROM TEACHERS](image4)
2- Trigger for Course_ID, Year_ID, Number_Of_Hours in Students table which are foreign keys for the same columns in Courses table.

Here the query that will fire the trigger. We update the Course id in the Students table to 22 which is not in the Courses table.

Here we can see that 22 was added to the courses table.
Here the Update in the Students table

The following query will make the trigger to do nothing (NULL)

The update will not make any problem because the 15 is already in the Courses table

Here the result of the update
3- Trigger for the Teacher_ID in the Courses Table which references to the ID in the teachers table.

Here the teachers table before firing the trigger

Here the query that will fire the trigger
Here the teachers table after the trigger, we see that ID = 7 was added to the table.

Here the update in the courses table.

Here the update will cause nothing in the trigger and the update will be done without any change in the teachers table.
Here the courses table after update

The departments_head table before the firing of the trigger

4- Triggers for the foreign key Head_ID in the departments_head table which references to the Head_ID in the teachers table
Now we will update the head_id to 9 which is not in the teachers table which cause firing the trigger that will add the teacher with id = 9 to the teachers table

The result of firing the trigger on the teachers table (insertion of teacher with id = 9)

Here the result of updating the departments_head table
If we update the head_id with a value that already in the teachers table the trigger will make nothing in the teachers (NULL action)

The update in the departments_head table

5- Trigger for the Head_id in the teachers table which is foreign key for the ID in the same table.

Here we will face a mutating problem which can be solved as the following

Here a brief explanation

Pragma Autonomous transaction:-
Whenever we commit all the transactions before the last commit in that session get saved. Suppose that you have a scenario you just want a particular transaction only to be save. In order to achieve that you make this as a separate transaction which can be committed but prior to this the transactions are not commit.

eg:

insert into tab1 () values ();
update tab2 set values.. some more stmts..
pragma autonomous transaction
insert into tab3 values();
commit;
here.. insert into tab1 is only saved but not tab1 and tab2.

Here the teachers table before the trigger

<table>
<thead>
<tr>
<th>ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>HEAD_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ahmed</td>
<td>Wael</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Ahmed</td>
<td>Salem</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Mohamed</td>
<td>Ali</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Faisal</td>
<td>Kamal</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Wael</td>
<td>Salem</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>Ali</td>
<td>Wael</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>First</td>
<td>Last</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>First</td>
<td>Last</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>First</td>
<td>Last</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>First</td>
<td>Last</td>
<td></td>
</tr>
</tbody>
</table>

Here the query that will fire the trigger because the id = 12 is not in the teachers table

UPDATE TEACHERS
SET HEAD_ID = 12
WHERE ID = 5
Here the table after firing the table and inserting the teacher with id = 12

Security triggers

1- Trigger on the Courses table in order not to update or delete or insert on the table after the end of the work times

```sql
CREATE OR REPLACE TRIGGER secure_trigger_courses
BEFORE INSERT OR UPDATE ON courses
BEGIN
  IF (TO_CHAR(SYSDATE, 'DY') IN ('Su', 'Su') OR
  (TO_CHAR(SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '10:00') THEN
    RAISE_APPLICATION_ERROR(-20000, 'لا يمكن التعديل');
  END IF;
END secure_trigger_courses;
```
Here the test of the trigger

[Image of SQL query]

2- Trigger on the Teachers table in order not to update or delete or insert on the table after the end of the work times

[Image of trigger code]

Here the result of the test of this trigger

[Image of SQL query with error]

[Image of error message]