Advanced Databases – LAB. 4
DB Design, Trigger, Procedures

EXAM RESULTS DATABASE

EXAM RESULTS					
Student Name	Mary	Student Surname	Smith	Student ID	D1231231
EXAM Code	EXAM Name	CA Weight	EXAM Weight	CA Marks	EXAM Marks
1234	Programming	40%	40%	67	45
4567	Databases	50%	50%	55	88
5987	Web Apps	30%	70%	77	66
4506	Algorithms	30%	70%	47	46
2345	Maths	20%	80%	56	75

DESIGN EXERCISE

The above sample form contains the results of the exams for a single student. Our database contains all the results for all the students. Student ID is unique, and Exam Code is a unique identifier for an exam. A student can only do an exam once.

Provide a normalized set of tables to store the data and implement them in ORACLE. Store the exam and CA weights as a float (for instance, 30% is 0.3). Note that CA Weight + Exam Weight is always equal to 100%. CA and Exam weight are between 0 and 1, exam and CA results are between 0 and 100.

TRIGGERS

Write a trigger that checks if, every time a new course result is inserted (or updated), CA weight and EXAM weight sums up to 1. If they do not, an exception is raised. Insert some data and check if the trigger is working.

Create a table Boards containing:

StudentID	Exam_Code	Result	Decision

Decision is a string that can take the following values 'RE', 'RCA', 'RAC', 'A','B','C','D'. Exam_Code and StudentID are FK to the table you have created in your design exercise.

PROCEDURES

Write a procedure to create a report that fills the table Boards. The field Final results is the weighted average of CA and EXAM, i.e. Final Results = ca marks * ca weight + exam marks * exam weight. See next page for a quick review on procedures.

Every time the report is generated, the table is first deleted and then populated. The decision is set according to the following rule:

RE – the student failed the exam (<40 marks) only.
RCA – the student failed the CA (<40 marks) only.
RAC – Repeat all components (the student failed both CA and EXAM)
If both EXAM and CA were above 40, then:
A – weighted average >= 70
B – weighted average in [60,70]
C – weighted average in [50,60]
D – weighted average in [40,50]

Very short review on Oracle procedures

First of all, turn the SQL developer output on if you want to see the output on the screen:

SET serveroutput ON; (don't forget it!)

How a procedure looks like:

```
procedure test2
IS
l_last_name employees.last_name%TYPE;
l_department_name departments.department_name%TYPE;
BEGIN
      SELECT last_name, department_name INTO l_last_name, l_department
      FROM
      employees e, departments d WHERE
      e.department_id=d.department_id AND   e.employee_id=138;

      DBMS_OUTPUT.put_line ( l_last_name || ' in ' || l_department);
END;
```

Notes:

The procedure print the name and the department of a specific employee

This line:

```
      l_last_name employees.last_name%TYPE;
```

declare a variable l_last_name of the same type of the filed last_name in the table employee

A command like this:

```
      l_employee employees%ROWTYPE;
```

declare a variable l_employee of the same type as a full row of the table employess. Each particular field can be accessed using the syntax l_employee.last_name , l_employee.employee_id ….

How a procedure is called:

In order to execute a function/procedure in Oracle use the `execute` command:

```
execute test2
execute  procedurename(paramenter list);
```

How to loop through the fields of a table.

A handy way is to use a cursor (explicit or implicit) to loop over one or more field of a table. You can use a cursor explicitly or implicitly declared.

Suppose to have this simple table:

```
create table patients(
      p_id number(2) primary key,
      p_name varchar(20));
```

Loop using cursors:

Explicit cursor (declared).

First declare the variable, than open the cursor, then use it:

```
CREATE OR REPLACE PROCEDURE Showpatients
IS
  CURSOR c1 IS
    SELECT p_id, p_name FROM patients;
BEGIN
 FOR item IN c1
      LOOP
            DBMS_OUTPUT.PUT_LINE ('id = ' ||  item.p_id || ', name
= ' || item.p_name);
      END LOOP;
END;
/
```

Implicit cursor example (no need to declare explicitly a cursor variable):

```
CREATE OR REPLACE PROCEDURE Showpatients IS
BEGIN
      FOR item IN ( SELECT p_id, p_name FROM patients)
      LOOP
            DBMS_OUTPUT.PUT_LINE ('id = ' ||  item.p_id || ', name
= ' || item.p_name);
      END LOOP;
END;
/
```