Sistemas de Bases de Dados/Database Systems
Lab Session 1 (PLSQL)

Goal

- Get acquainted with the configuration of the database server and clients in the labs, e review some basics in Oracle and PL-SQL:
- Use the Oracle database server via a SQL-Developer client
- Implement some procedures in PL-SQL
- Use cursors to access tables

Resources

During the class you should use an Oracle 18c server. You will also need:

- Scripts criarbd.sql and insdados.sql for creating a small example database
- PL/SQL User's Guide and Reference (parts regarding Procedures and Cursors)

Other useful links:

- www.oracletutorial.com/plsql-tutorial

Connecting to the database server

1. Run the SQL client to connect to the server (SQL Developer is recommended).

   You should have an account with username sbd<YourStudentNumber>. The password was communicated to you by email.

   To connect use the following hostname and SID:

   10.170.138.40 and orclE

2. Create the small example database using the provided scripts. In SQL Developer you can run them by executing the following commands in the SQL Worksheet *@path_to_script*/createdb.sql and *@path_to_script*/insdados.sql

3. Check within SQL Developer that the tables were created. You may query the dictionary of Oracle querying **user_tables** table, or use the DESC command.

4. Also check, by posing several queries, that some data is indeed there.

Creating some PL/SQL procedures

PL/SQL procedures are created as:

```
CREATE [OR REPLACE] PROCEDURE <name>

IS

BEGIN

        PLSQL code;

END;
```

For example:

```
CREATE OR REPLACE PROCEDURE helloworld

IS

BEGIN

      dbms_output.put_line('Hello World');

END;
```

The procedure is executed with exec helloworld;

Try it! (You may need to set serveroutput on)

Exercise 1:

For this exercise you will need to access 3 tables: alunos, cadeiras and inscricoes. Make sure that you first understand the model of the database, even if restricted to these tables, including the integrity constraints and the use of sequences.

Define a procedure enroll_student(student,subject) that enrolls the student in the subject. For this procedure assume that both the student and the subject already exist in the database.

Try your procedure by making some enrollments. E.g. try to enroll 'Joaquim Pires Lopes' in both 'Bases de Dados' and 'Sistemas de Bases de Dados'

Check with a SQL query that this student was indeed enrolled (in one of the courses; the other one fails - why?)

Exercise 2:

Implement a procedure new_student(name,gender,degree_name) that registers a new student. Allow that the degree_name could use wildcards like '%' and '_'.

The student number should be automatically assigned by an already existing sequence. (Tip: the current value of a sequence is returned, and then incremented when accessing <sequenceName>.nextval).

Try the procedure by adding a new student.

Cursors in PL/SQL

Before you exercise alone the use of cursors, let's start with a guided exercise to recall how cursors work. Try them in your database!

A simple cursor can be made to view all students:

```
CREATE OR REPLACE PROCEDURE test_cursor

IS

CURSOR cur_aluno IS SELECT * FROM alunos;

v_aluno alunos%ROWTYPE;

BEGIN

   OPEN cur_aluno;

   FETCH cur_aluno into v_aluno;

   dbms_output.put_line(v_aluno.num_aluno || ' ' || v_aluno.nome);

   CLOSE cur_aluno;

END;
```

This procedure only obtains the **first** student. But one can iterate, e.g. with:

```
        LOOP

           FETCH cur_aluno into v_aluno;

           EXIT WHEN cur_aluno%NOTFOUND;

           dbms_output.put_line(v_aluno.num_aluno || ' ' || v_aluno.nome);

        END LOOP;
```

or with:

```
        FOR v IN cur_aluno LOOP

           dbms_output.put_line(v.nome || ' ' || v.sexo);

        END LOOP;
```

Note that in the latter case you don’t need to open or close the cursor.

You can test with the following adapted versions of the test_cursor procedure

```
CREATE OR REPLACE PROCEDURE test_cursor_1

IS

CURSOR cur_aluno IS SELECT * FROM alunos;

v_aluno alunos%ROWTYPE;

BEGIN

   OPEN cur_aluno;

   LOOP
      FETCH cur_aluno into v_aluno;
      EXIT WHEN cur_aluno%NOTFOUND;
      dbms_output.put_line(v_aluno.num_aluno || ' ' || v_aluno.nome);
   END LOOP;

   CLOSE cur_aluno;

END;
```

Note that in the 2nd version the auxiliary variable is not needed. NEAT!

```
CREATE OR REPLACE PROCEDURE test_cursor_2

IS

CURSOR cur_aluno IS SELECT * FROM alunos;

BEGIN

   FOR v IN cur_aluno LOOP

      dbms_output.put_line(v.nome || ' ' || v.sexo);

   END LOOP;

END;
```

Guidelines on the use of cursors
(from https://blogs.oracle.com/connect/post/working-with-cursors)

Here are some guidelines to help you decide which technique to use:

- When you are fetching a single row, use SELECT-INTO or EXECUTE IMMEDIATE-INTO (if your query is dynamic). Do not use an explicit cursor or a cursor FOR loop.
- When you are fetching *all* the rows from a query, use a cursor FOR loop *unless* the body of the loop executes one or more DML statements (INSERT, UPDATE, DELETE, or MERGE). In such a case, you will want to switch to BULK COLLECT and FORALL.
- Use an explicit cursor when you need to fetch with BULK COLLECT, but *limit* the number of rows returned with each fetch.
- Use an explicit cursor when you are fetching multiple rows but might conditionally exit before all rows are fetched.
- Use a cursor variable when the query you are fetching from varies at runtime (but isn't necessarily dynamic) and especially when you need to pass a result back to a non-PL/SQL host environment.
- Use EXECUTE IMMEDIATE to query data *only* when you cannot fully construct the SELECT statement while you write your code.

Exercise 3:

Implement a procedure that lists all the students in the database as in the listing below:

> Ana Maria Fonseca está inscrita em Engenharia Electrotecnica
> Joana Ramalho Silva está inscrita em Matematica
> Joaquim Pires Lopes está inscrito em Engenharia Informatica
> Paula Antunes está inscrita em Engenharia do Ambiente

Mark the gender agreement in Portuguese!

Exercise 4:

Cursors may have parameters. For example:

```
CURSOR cur_aluno(sx alunos.sexo%type) IS
SELECT * FROM alunos WHERE sx = sexo;
```

When the cursor is open, one must define the parameter. For example:

```
open cur_aluno('M');
```

To test this (or even without using it, if you prefer) implement a procedure that given a number N, returns the student-number of the N students with more enrolments (extra/challenge: how would you restrict this to enrollment in different subjects? What about current enrolments?).

Exercise 5:

Improve the procedure for enrolling students in subjects enroll_student(student,subject) that enrolls all students whose name starts by student in all subjects whose name start with subject

- If the student does not exist, the procedure should warn the user.
- If the subject does not exist, the procedure should warn the user.
- If the student is already enrolled in the subject, the procedure should warn the user, and do nothing.

Write the code using the BULK COLLECT mechanisms in order to improve efficiency.

Tip: You may find useful to use Oracle's exception handler. For that, check the manual.