Laborator 10:
PACHETE in PL/SQL

1. Definirea pachetelor

- Pachetul (*package*) permite încapsularea într-o unitate logică în baza de date a procedurilor, funcţiilor, cursoarelor, tipurilor, constantelor, variabilelor şi excepţiilor.
- Spre deosebire de subprograme, pachetele nu pot:
  - fi apelate
  - transmite parametri
  - fi încuibărite
- Un pachet are două parţi, fiecare fiind stocată separat în dicţionarul datelor.
  - Specificarea pachetului (*package specification*) – partea „vizibilă”, adică interfaţa cu aplicaţii sau cu alte unităţi program. Se declară tipuri, constante, variabile, excepţii, cursoare şi subprograme folositoare utilizatorului.
  - Corpul pachetului (*package body*) – partea „acunsă”, mascată de restul aplicaţiei, adică realizarea specificaţiei. Corpul defineşte cursoare şi subprograme, implementând specificaţia. Obiectele conţinute în corpul pachetului sunt fie private, fie publice.
- Un pachet are următoarea formă generală:

```
CREATE PACKAGE nume_pachet {IS | AS} -- specificaţia
/* interfaţa utilizator, care conţine: declaraţii de tipuri şi obiecte
        publice, specificaţii de subprograme */
END [nume_pachet];

CREATE PACKAGE BODY nume_pachet {IS | AS} -- corpul
      /* implementarea, care conţine: declaraţii de obiecte şi tipuri private,
         corpuri de subprograme specificate în partea de interfaţă */
[BEGIN]
      /* instrucţiuni de iniţializare, executate o singură dată când
            pachetul este invocat prima oară de către sesiunea
      utilizatorului */
END [nume_pachet];
```

2. Pachete predefinite

- *DBMS_OUTPUT* permite afişarea de informaţii. *DBMS_OUTPUT* lucrează cu un *buffer* (conţinut în *SGA*) în care poate fi scrisă sau regăsită informaţie. Procedurile pachetului sunt:
  PUT – depune (scrie) în buffer informaţie
  PUT_LINE – depune în buffer informaţia, împreună cu un marcaj - sfârşit de linie
  NEW_LINE – depune în buffer un marcaj - sfârşit de linie
  GET_LINE – regăseşte o singură linie de informaţie;
  GET_LINES – regăseşte mai multe linii de informaţie;
  ENABLE/DISABLE – activează/dezactivează procedurile pachetului.

- *DBMS_SQL* permite folosirea dinamică a comenzilor *SQL* în proceduri stocate sau în blocuri anonime şi analiza gramaticală a comenzilor *LDD*.
  - *OPEN_CURSOR* (deschide un nou cursor, adică se stabileşte o zonă de memorie în care este procesată comanda *SQL*);
  - *PARSE* (stabileşte validitatea comenzii *SQL*, adică se verifică sintaxa instrucţiunii şi se asociază cursorului deschis);
  - *BIND VARIABLE* (leaga valoarea data de variabila corespunzatoare din comanda *SQL* analizata)
  - *EXECUTE* (execută comanda *SQL* şi returnează numărul de linii procesate);
  - *FETCH_ROWS* (regăseşte o linie pentru un cursor specificat, iar pentru mai multe linii foloseşte un *LOOP*);
  - *CLOSE_CURSOR* (închide cursorul specificat).
- *DBMS_JOB* este utilizat pentru planificarea execuţiei programelor PL/SQL. Dintre subprogramele acestui pachet menţionăm:
  - *SUBMIT* – adaugă un nou *job* în coada de aşteptare a *job*-urilor;
  - *REMOVE* – şterge un *job* specificat din coada de aşteptare a *job*-urilor;
  - *RUN* – execută imediat un *job* specificat;
  - *NEXT_DATE* – modifică momentul următoarei execuţii a unui *job*;
  - *INTERVAL* – modifică intervalul între diferite execuţii ale unui *job*.
- *UTL_FILE* permite programului *PL/SQL* citirea din fişierele sistemului de operare, respectiv scrierea în aceste fişiere. El este utilizat pentru exploatarea fişierelor text. Scrierea şi regăsirea informaţiilor se face cu ajutorul unor proceduri asemănătoare celor din pachetul DBMS_OUTPUT.
  Procedura FCLOSE permite închiderea unui fişier.

Probleme rezolvate

1.
a) Creaţi specificaţia şi corpul unui pachet numit DEPT_PKG_PNU care conţine:
- procedurile ADD_DEPT, UPD_DEPT şi DEL_DEPT, corespunzătoare operaţiilor de adăugare, actualizare (a numelui) şi ştergere a unui departament din tabelul DEPT_PNU;
- funcţia GET_DEPT, care determină denumirea unui departament, pe baza codului acestuia.

b) Invocaţi procedurile şi funcţia din cadrul pachetului atât prin blocuri PL/SQL cât şi prin comenzi SQL.

Soluţie:
Specificaţia pachetului:

```
create or replace PACKAGE dept_pkg_pnu IS
        PROCEDURE add_dept (p_deptid dept.deptno%TYPE, p_deptname dept.dname%TYPE);
        PROCEDURE del_dept (p_deptid dept.deptno%TYPE);
        FUNCTION get_dept (p_deptid dept.deptno%TYPE) RETURN dept.dname%TYPE;
        PROCEDURE upd_dept (p_deptid dept.deptno%TYPE, p_deptname dept.dname%TYPE);
END dept_pkg_pnu;
```

Obs: Dacă apare un *warning* care anunţă că există erori de compilare, atunci este utilă comanda:
SHOW ERRORS
Corpul pachetului:

```
create or replace PACKAGE BODY dept_pkg_pnu IS
```

```
	PROCEDURE add_dept (p_deptid dept.deptno%TYPE, p_deptname
dept.dname%TYPE)
   IS
	BEGIN
		INSERT INTO dept_pnu(deptno, dname)
		VALUES (p_deptid, p_deptname);
		COMMIT;
	END add_dept;

	PROCEDURE del_dept (p_deptid dept.deptno%TYPE) IS
	BEGIN
		DELETE FROM dept_pnu
		WHERE deptno = p_deptid;
		IF SQL%NOTFOUND THEN
			RAISE_APPLICATION_ERROR(-20203, 'Nici un departament sters');
		END IF;
	END del_dept;

	FUNCTION get_dept (p_deptid dept.deptno%TYPE)
	RETURN dept.dname%TYPE IS
		v_nume dept.dname%TYPE;
	BEGIN
		SELECT dname
		INTO v_nume
		FROM dept_pnu
		WHERE deptno = p_deptid;
		RETURN v_nume;
	END get_dept;

	PROCEDURE upd_dept (p_deptid dept.deptno%TYPE, p_deptname
dept.dname%TYPE) IS
	BEGIN
		UPDATE dept_pnu
		SET dname = p_deptname
		WHERE deptno = p_deptid;
		IF SQL%NOTFOUND THEN
			RAISE_APPLICATION_ERROR(-20204, 'Nici un departament
actualizat');
		END IF;
	END upd_dept;
END dept_pkg_pnu;
```

Obs: Pentru invocarea procedurii:

```
BEGIN
	dept_pkg_pnu.add_dept(12, ‘IT’);
	upd_dept(12, ‘Information technology’);
END;
```

Pentru invocarea funcţiei:

```
BEGIN
	DBMS_OUTPUT.PUT_LINE(‘Departamentul cautat este: ‘ ||
dept_pkg_pnu.get_dept(12) );
END;
```

2. Creaţi specificaţia şi corpul unui pachet numit EMP_PKG_PNU care conţine:
- procedura publică ADD_EMP - adaugă o înregistrare în tabelul EMP_PNU; utilizează o secvenţă pentru generarea cheilor primare; vor fi prevăzute valori implicite pentru parametrii nespecificaţi;
- procedura publică GET_EMP - pe baza unui cod de angajat transmis ca parametru, întoarce în doi parametri de ieşire salariul şi job-ul corespunzător;
- funcţia privată VALID_JOB_ID - rezultatul acestei funcţii indică dacă job-ul unui angajat corespunde unei valori existente în tabelul JOBS. Funcţia va fi utilizată în cadrul procedurii ADD_EMP, făcând posibilă doar introducerea de înregistrări având coduri de job valide.
Trataţi eventualele excepţii.

Soluţie:

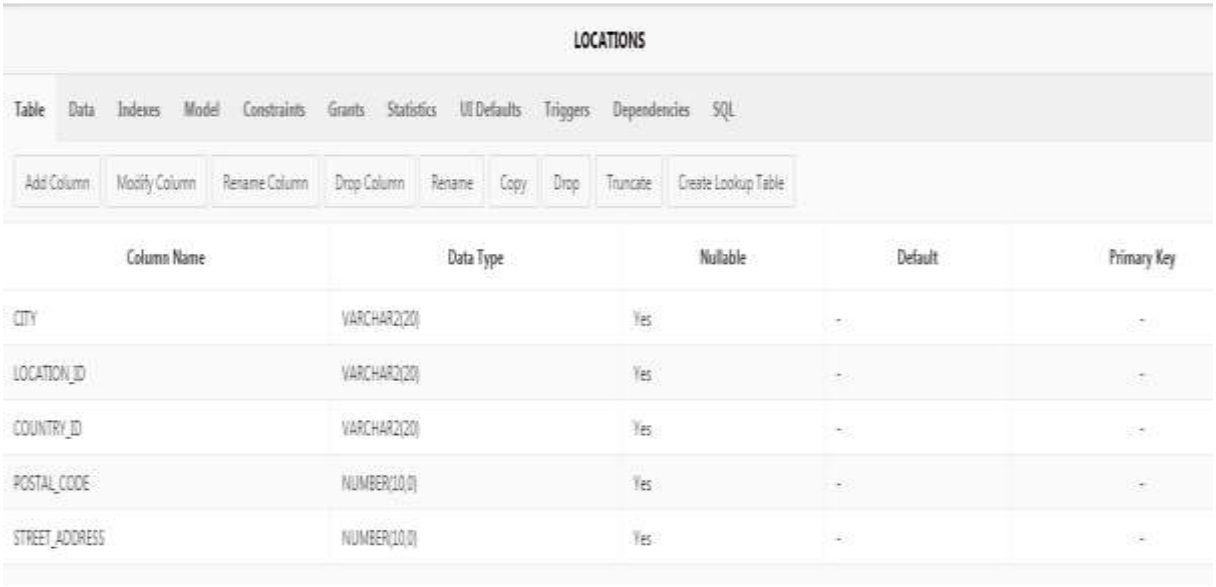
```
create or replace PACKAGE emp_pkg_pnu IS
        Procedure add_emp (
                p_name emp.ename%TYPE,
                p_hire_date emp.hiredate%TYPE DEFAULT SYSDATE,
                p_mgr emp.mgr%TYPE DEFAULT 145,
                p_sal emp.sal%TYPE DEFAULT 1000,
                p_job emp.job%TYPE DEFAULT 'SA_REP',
                p_comm emp.comm%TYPE DEFAULT 0,
                p_deptid emp.deptno%TYPE DEFAULT 30);
        Procedure get_emp (p_empid IN emp.empno%TYPE, p_sal OUT emp.sal%TYPE,
p_job OUT emp.job%TYPE);
END emp_pkg_pnu;
```

```
create or replace PACKAGE BODY emp_pkg_pnu IS

   PROCEDURE add_emp (
                p_name emp.ename%TYPE,  --implicit de tip IN
                p_hire_date emp.hiredate%TYPE DEFAULT SYSDATE,
                p_mgr emp.mgr%TYPE DEFAULT 145,
                p_sal emp.sal%TYPE DEFAULT 1000,
                p_job emp.job%TYPE DEFAULT 'SA_REP',
                p_comm emp.comm%TYPE DEFAULT 0,
                p_deptid emp.deptno%TYPE DEFAULT 30)
        IS
        BEGIN
              INSERT INTO emp_pnu(empno, ename, mgr, sal, job, comm, deptno, hiredate)
         VALUES(123, p_name, p_mgr, p_sal, p_job, p_comm, p_deptid, p_hire_date);
        END add_emp;

        PROCEDURE get_emp (p_empid IN emp.empno%TYPE,
                                p_sal OUT emp.sal%TYPE,
                p_job OUT emp.job%TYPE)
        IS
        BEGIN
                SELECT sal, job
                INTO p_sal, p_job
                FROM emp_pnu
                WHERE empno = p_empid;
        END get_emp;

END emp_pkg_pnu;
```

Exemplu de apel:

```
Declare
        v1 emp.sal%TYPE;
        v2 emp.job%TYPE;
begin
        emp_pkg_pnu.add_emp('Popescu', p_job => 'SA_MAN');
        emp_pkg_pnu.get_emp(123,v1,v2);
        DBMS_OUTPUT.PUT_LINE(v1||' '||v2);
End;
```

3. Să se creeze un pachet cu ajutorul căruia, utilizând un cursor şi un subprogram funcţie, să se obţină salariul maxim înregistrat pentru salariaţii care lucrează într-un anumit oraş şi lista salariaţilor care au salariul mai mare sau egal decât maximul salariilor din oraşul respectiv.



LOCATIONS

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL

Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table

Column Name	Data Type	Nullable	Default	Primary Key
CITY	VARCHAR2(20)	Yes	-	-
LOCATION_ID	VARCHAR2(20)	Yes	-	-
COUNTRY_ID	VARCHAR2(20)	Yes	-	-
POSTAL_CODE	NUMBER(10,0)	Yes	-	-
STREET_ADDRESS	NUMBER(10,0)	Yes	-	-

```
CREATE  OR REPLACE PACKAGE problema_3 AS
  CURSOR c_emp(nr NUMBER) RETURN emp%ROWTYPE;
  FUNCTION  sal_max  (p_oras  locations.city%TYPE) RETURN NUMBER;
END problema_3;

create or replace PACKAGE BODY problema_3 AS
  CURSOR c_emp(nr NUMBER) RETURN emp%ROWTYPE  IS
    SELECT * FROM emp WHERE sal >= nr;
  FUNCTION  sal_max (p_oras  locations.city%TYPE) RETURN NUMBER  IS
    maxim  NUMBER;
   BEGIN
       SELECT  MAX(sal)
       INTO    maxim
       FROM    emp e, dept d, locations l
       WHERE   e.deptno=d.deptno AND d.loc=l.location_id
             AND UPPER(city)=UPPER(p_oras);
       RETURN  maxim;
   END sal_max;
END problema_3;
```

```
DECLARE
  v_oras locations.city%TYPE:= 'Montreal';
  v_max NUMBER;
  v_emp emp%ROWTYPE;
BEGIN
   v_max:=problema_3.sal_max(v_oras);
   OPEN problema_3.c_emp(v_max);
   LOOP
      FETCH problema_3.c_emp INTO v_emp;
      EXIT WHEN problema_3.c_emp%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(v_emp.ename||' '||v_emp.sal);
   END LOOP;
   CLOSE problema_3.c_emp;
END;
```

Probleme propuse spre rezolvare

1. Să se creeze un pachet prin care să se calculeze, utilizând subprograme supraîncărcate, suma salariilor, salariul maxim, cel mediu şi cel minim, suma salariilor pe departamente, respectiv pe job-uri. Codul departamentului sau al job-ului vor fi furnizate ca parametri. Apelaţi funcţiile utilizând un bloc. Codurile de departament şi job vor fi initializate pe linia de declarare.

2. Să se creeze un pachet care conţine o funcţie ce returnează locaţia unui angajat al cărui cod este transmis ca parametru şi un cursor prin care se obţine lista angajaţilor care lucrează într-o locaţie al cărei cod este transmis ca parametru. Apelaţi funcţia şi cursorul prin intermediul unui bloc.

Bibliografie web:

https://www.w3resource.com/
https://www.bullraider.com/database/pl-sql/pl-sql-examples
https://www.oracletutorial.com/plsql-tutorial/plsql-package-specification/
https://www.oracletutorial.com/plsql-tutorial/plsql-package-body/
https://www.oracletutorial.com/plsql-tutorial/plsql-drop-package/