SGBD-Anul 3
Laborator 6 PL/SQL

Pachete

I. Definirea pachetelor

- Pachetul (*package*) permite încapsularea într-o unitate logică în baza de date a procedurilor, funcțiilor, cursoarelor, tipurilor, constantelor, variabilelor şi excepțiilor.
- Spre deosebire de subprograme, pachetele nu pot:
  - fi apelate,
  - transmite parametri,
  - fi încuibărite.
- Un pachet are două parți, fiecare fiind stocată separat în dicționarul datelor.
  - Specificarea pachetului (*package specification*) – partea „vizibilă”, adică interfața cu aplicații sau cu alte unități program. Se declară tipuri, constante, variabile, excepții, cursoare şi subprograme folositoare utilizatorului.
  - Corpul pachetului (*package body*) – partea „acunsă”, mascată de restul aplicației, adică realizarea specificației. Corpul defineşte cursoare şi subprograme, implementând specificația. Obiectele conținute în corpul pachetului sunt fie private, fie publice.
- Un pachet are următoarea formă generală:

```
CREATE PACKAGE nume_pachet {IS | AS} -- specificația
        /* interfața utilizator, care conține: declarații de tipuri şi obiecte
           publice, specificații de subprograme */
END [nume_pachet];
/
CREATE PACKAGE BODY nume_pachet {IS | AS} -- corpul
        /* implementarea, care conține: declarații de obiecte şi tipuri private,
           corpuri de subprograme specificate în partea de interfață */
[BEGIN]
        /* instrucțiuni de inițializare, executate o singură dată când
          pachetul este invocat prima oară de către sesiunea utilizatorului */
END [nume_pachet];
/
```

II. Pachete predefinite

- *DBMS_OUTPUT* permite afişarea de informații. *DBMS_OUTPUT* lucrează cu un *buffer* (conținut în *SGA*) în care poate fi scrisă sau regăsită informație. Procedurile pachetului sunt:

  PUT – depune (scrie) în buffer informație
  PUT_LINE – depune în buffer informația, împreună cu un marcaj - sfârşit de linie
  NEW_LINE – depune în buffer un marcaj - sfârşit de linie
  GET_LINE – regăseşte o singură linie de informație;
  GET_LINES – regăseşte mai multe linii de informație;
  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 instrucțiunii 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 execuției programelor PL/SQL. Dintre subprogramele acestui pachet menționăm:
- *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 execuții a unui *job*;
- *INTERVAL* – modifică intervalul între diferite execuții 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 informațiilor se face cu ajutorul unor proceduri asemănătoare celor din pachetul DBMS_OUTPUT.
Procedura FCLOSE permite închiderea unui fişier.

Exerciții
I. [Pachete definite de utilizator]

1. a) Creați specificația şi corpul unui pachet numit DEPT_PKG care conține:
- procedurile ADD_DEPT, UPD_DEPT şi DEL_DEPT, corespunzătoare operațiilor de adăugare, actualizare (a numelui) şi ştergere a unui departament din tabelul DEPT;
- funcția GET_DEPT, care determină denumirea unui departament, pe baza codului acestuia.

2. Creați specificația şi corpul unui pachet numit EMP_PKG care conține:
- procedura publică ADD_EMP - adaugă o înregistrare în tabelul EMP; utilizează o secvență pentru generarea cheilor primare; vor fi prevăzute valori implicite pentru parametrii nespecificați;
- procedura publică GET_EMP - pe baza unui cod de angajat transmis ca parametru, întoarce în doi parametri de ieşire salariul şi job-ul corespunzător;
- funcția privată VALID_JOB_ID - rezultatul acestei funcții indică dacă job-ul unui angajat corespunde unei valori existente în tabelul JOBS. Funcția va fi utilizată în cadrul procedurii ADD_EMP, făcând posibilă doar introducerea de înregistrări având coduri de job valide.
Tratați eventualele excepții.

3. Modificați pachetul EMP_PKG anterior supraîncărcând procedura ADD_EMP. Noua procedură va avea 3 parametri, corespunzători numelui, prenumelui şi codului job-ului. Procedura va formata câmpul email astfel încât acesta să fie scris cu majuscule, prin concatenarea primei litere a prenumelui şi a primelor 7 litere ale numelui. Va fi apelată vechea procedură ADD_EMP pentru inserarea efectivă a unei înregistrări.

4. a) Creați două funcții supraîncărcate GET_EMP în pachetul EMP_PKG:
- o funcție GET_EMP va avea un parametru p_emp_id de tipul employees.employee_id%TYPE şi va regăsi linia corespunzătoare codului respectiv;
- cealaltă funcție GET_EMP va avea un parametru p_nume_familie de tipul employees.last_name%TYPE şi va regăsi linia corespunzătoare numelui respectiv;
- ambele funcții vor returna o valoare de tipul employees%ROWTYPE.

b) In pachet se va mai adăuga procedura PRINT_EMPLOYEE având un parametru de tipul EMPLOYEES%ROWTYPE, care afişează codul departamentului, codul angajatului, prenumele, numele, codul job-ului şi salariul, utilizând DBMS_OUTPUT.

c) Utilizați un bloc anonim pentru apelarea funcțiilor şi a procedurii anterioare.

5. Introduceți în pachet funcția valid_deptid. Modificați prima procedură add_emp astfel încât introducerea unui angajat nou să fie posibilă doar dacă departamentul este valid.

Presupunând că firma nu actualizează frecvent datele despre departamente, pachetul EMP_PKG poate fi îmbunătățit prin adăugarea procedurii publice INIT_DEPT care populează un tablou privat PL/SQL de coduri de departament valide. Creați această procedură.

6. a) Modificați funcția VALID_DEPTID pentru a utiliza acest tablou PL/SQL.

b) Testați procedura ADD_EMP adăugâng un salariat în departamentul 15. Ce se întâmplă ?

7. Să se creeze un pachet cu ajutorul căruia, utilizând un cursor şi un subprogram funcție, să se obțină salariul maxim înregistrat pentru salariații care lucrează într-un anumit oraş şi lista salariaților care au salariul mai mare sau egal decât maximul salariilor din oraşul respectiv.

8. Să se creeze un pachet *verif_pkg* ce include o procedură prin care se verifică dacă o combinație specificată de valori ale atributelor *job_id* şi *department_id* este o combinație care există în tabelul *EMPLOYEES*.

II. [Pachete standard]

[DBMS_OUTPUT]

9. Să se scrie un bloc anonim care reține în 3 variabile PL/SQL numele, salariul şi departamentul angajatului având codul 145. Să se afişeze aceste informații (implicit, se va introduce o linie în buffer-ul specific DBMS_OUTPUT). Să se regăsească această linie şi starea corespunzătoare (0, dacă există linii în buffer şi 1, altfel). Să se afişeze linia şi starea.

[DBMS_JOB]

10. a) Să se utilizeze pachetul *DBMS_JOB* pentru a plasa pentru execuție în coada de aşteptare a *job*-urilor, procedura *verifica* din pachetul *verif_pkg.* Prima execuție va avea loc peste 5 minute.

b) Aflați informații despre job-urile curente în vizualizarea USER_JOBS.

c) Identificați în coada de aşteptare job-ul pe care l-ați lansat şi executați-l.

d) Stergeți job-ul din coada de aşteptare.

[UTL_FILE]

11. Creați o procedură numită EMP_REPORT care generează un raport într-un fişier al sistemului de operare, utilizând pachetul UTL_FILE. Raportul va conține lista angajaților care au depăşit media salariilor din departamentul lor. Procedura va avea doi parametri: directorul de ieşire şi numele fişierului text în care va fi scris raportul. Tratați excepțiile care pot apărea la utilizarea pachetului UTL_FILE.

[SQL dinamic, DBMS_SQL]

12. Să se construiască o procedură care foloseşte *SQL* dinamic pentru a şterge liniile unui tabel specificat ca parametru. Subprogramul furnizează ca rezultat numărul liniilor şterse (*nr_lin*).

13. a) Creați un pachet numit TABLE_PKG care utilizează SQL nativ pentru crearea sau ştergerea unui tabel şi pentru adăugarea, modificarea sau ştergerea de linii din tabel.
Specificația pachetului va conține procedurile următoare:
- PROCEDURE make (table_name VARCHAR2, col_specs VARCHAR2)
- PROCEDURE add_row (table_name VARCHAR2, values VARCHAR2, cols VARCHAR2 := NULL)
- PROCEDURE upd_row(table_name VARCHAR2, set_values VARCHAR2, conditions VARCHAR2 := NULL)
- PROCEDURE del_row(table_name VARCHAR2, conditions VARCHAR2 := NULL)
- PROCEDURE remove(table_name VARCHAR2)

b) Executați procedura MAKE pentru a crea un tabel, astfel:
make(’contacte’, ‘cod NUMBER(4), nume VARCHAR2(35)’);

c) Listați structura tabelului contacte.

d) Adăugați înregistrări prin intermediul procedurii ADD_ROW.
Exemplu: add_row(‘contacte’, ‘1, ‘’Geoff Gallus’’, ‘cod’, ‘’nume’’);

e) Afişați conținutul tabelului contacte.

f) Executați procedura DEL_ROW pentru ştergerea contactului având codul 1.

g) Executați procedura UPD_ROW.
Exemplu: upd_row(‘contacte’, ‘nume = ‘’Nancy Greenberg’’’, ‘cod=2’);

h) Afişați conținutul tabelului, apoi ştergeți tabelul prin intermediul procedurii remove.