Exemplul 7.1

```
CREATE SEQUENCE s_id_pret START WITH 100;

CREATE OR REPLACE PACKAGE pack_ex1
IS
  CURSOR lista(
     pc_client     pret_preferential.id_client_j%type,
     pc_categorie  pret_preferential.id_categorie%type)
   IS
   SELECT * FROM pret_preferential
   WHERE  id_client_j = pc_client
   AND    id_categorie = pc_categorie
   ORDER BY data_in DESC;

  FUNCTION gaseste_client(
     pf_client    pret_preferential.id_client_j%type,
     pf_categorie pret_preferential.id_categorie%type)
   RETURN BOOLEAN;

  PROCEDURE afiseaza_client(
     pp_client    pret_preferential.id_client_j%type,
     pp_categorie pret_preferential.id_categorie%type);

  PROCEDURE adauga_pret_preferential(
     p_discount  pret_preferential.discount%type,
     p_data_in   DATE,
     p_data_sf   DATE,
     p_categorie pret_preferential.id_categorie%type,
     p_client    pret_preferential.id_client_j%type);
END pack_ex1;
/

CREATE OR REPLACE PACKAGE BODY pack_ex1
IS
   -- verifica daca un client are deja un pret
   -- preferential pentru acea categorie
  FUNCTION gaseste_client(
     pf_client    pret_preferential.id_client_j%type,
     pf_categorie pret_preferential.id_categorie%type)
   RETURN BOOLEAN
   IS
     rezultat NUMBER;
   BEGIN
     SELECT COUNT(*) INTO rezultat
     FROM   pret_preferential
     WHERE  id_client_j = pf_client
     AND    id_categorie = pf_categorie
     AND    SYSDATE BETWEEN data_in AND data_sf;
     IF rezultat>0 THEN RETURN TRUE;
                   ELSE RETURN FALSE;
     END IF;
  END gaseste_client;
```

```
   -- afiseaza preturile preferentiale avute pentru acea
   -- categorie
   PROCEDURE afiseaza_client(
     pp_client    pret_preferential.id_client_j%type,
     pp_categorie pret_preferential.id_categorie%type)
   IS
   BEGIN
    FOR i IN lista(pp_client, pp_categorie) LOOP
     IF lista%ROWCOUNT=1 THEN
      DBMS_OUTPUT.PUT_LINE('Pret preferential activ:');
      DBMS_OUTPUT.PUT_LINE('Discount de '||
           i.discount*100 ||'% valabil in perioada '||
           i.data_in||' - '||i.data_sf);
      DBMS_OUTPUT.PUT_LINE('Preturi pref. avute:');
     ELSE
      DBMS_OUTPUT.PUT_LINE('Discount de '||
           i.discount*100 ||'% valabil in perioada '||
           i.data_in||' - '||i.data_sf);
       END IF;
    END LOOP;
   END afiseaza_client;

   PROCEDURE adauga_pret_preferential(
     p_discount  pret_preferential.discount%type,
     p_data_in   DATE,
     p_data_sf   DATE,
     p_categorie pret_preferential.id_categorie%type,
     p_client    pret_preferential.id_client_j%type)
   IS
   BEGIN
    IF gaseste_client(p_client,p_categorie)
    THEN
       DBMS_OUTPUT.PUT_LINE('Clientul are deja pret
             preferential pentru aceasta categorie');

       afiseaza_client(p_client,p_categorie);
    ELSE
       INSERT INTO pret_preferential
       VALUES (s_id_pret.NEXTVAL,p_discount,p_data_in,
                     p_data_sf,p_categorie,p_client);
       DBMS_OUTPUT.PUT_LINE('Adaugare cu succes');
    END IF;
   END adauga_pret_preferential;
END pack_ex1;
/
```

Exemplul 7.2

```
--utilizarea pachetului
EXECUTE  pack_ex1.adauga_pret_preferential(0.1,SYSDATE,
                  ADD_MONTHS(SYSDATE,3),500,260);

BEGIN
  IF pack_ex1.gaseste_client(260,500) THEN
     FOR i IN pack_ex1.lista(260,500) LOOP
        DBMS_OUTPUT.PUT_LINE('Discount de '||
           i.discount*100 ||'% valabil in perioada '||
           i.data_in||' - '||i.data_sf);
     END LOOP;
  ELSE
     DBMS_OUTPUT.PUT_LINE('Clientul nu a avut pret
              preferential pentru aceasta categorie');
  END IF;
END;
/
```

Exemplul 7.3

```
CREATE OR REPLACE PACKAGE pack_ex3
IS
  PROCEDURE afiseaza_produs(p_id  NUMBER);
  PROCEDURE afiseaza_produs(p_den VARCHAR2);
END;
/

CREATE OR REPLACE PACKAGE BODY pack_ex3 IS
  PROCEDURE afiseaza_produs(p_id NUMBER) IS
    v_den  produse.denumire%TYPE;
    v_pret produse.pret_unitar%TYPE;
  BEGIN
    SELECT MAX(denumire), MIN(pret_unitar)
    INTO   v_den, v_pret
    FROM   produse WHERE id_produs = p_id;
    DBMS_OUTPUT.PUT_LINE('Produsul cautat este : '||
         v_den||' are pretul ' ||v_pret);
  END;

  PROCEDURE afiseaza_produs(p_den VARCHAR2) IS
  contor NUMBER := 0;
  BEGIN
    FOR i IN (SELECT denumire, pret_unitar
              FROM   produse
              WHERE  INSTR(denumire,p_den)<>0) LOOP
       contor := contor+1;
       DBMS_OUTPUT.PUT_LINE(i.denumire||' are pretul '||
                            i.pret_unitar);
    END LOOP;
    IF contor=0 THEN
      DBMS_OUTPUT.PUT_LINE('Nu exista produsul cautat');
```

```
    ELSE
      DBMS_OUTPUT.PUT_LINE('Au fost gasite '||contor||
                           ' produse');
    END IF;
  END;
END;
/

EXECUTE pack_ex3.afiseaza_produs(1000);

EXECUTE pack_ex3.afiseaza_produs('etichete');
```

Exemplul 7.4 – pachete fără corp

```
CREATE OR REPLACE PACKAGE constante IS
  cm_in_inch CONSTANT NUMBER := 0.39;
  inch_in_cm CONSTANT NUMBER := 2.54;
END;
/

DECLARE
  v_diagonala NUMBER := &p_diagonala;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Diagonala de '|| v_diagonala ||
      'inch este echivalenta cu '||
       ROUND(v_diagonala*constante.inch_in_cm) ||'cm');
END;
/
```

Exemplul 7.5 – pachete fără corp

```
CREATE OR REPLACE PACKAGE exceptii IS
   nu_a_gasit EXCEPTION;
   PRAGMA EXCEPTION_INIT (nu_a_gasit, 100);
END;
/

DECLARE
  v_id NUMBER(4) := &p_id;
  x VARCHAR2(100);
BEGIN
  SELECT denumire INTO x
  FROM   produse
  WHERE  id_produs = v_id;
EXCEPTION
 WHEN exceptii.nu_a_gasit THEN
  DBMS_OUTPUT.PUT_LINE('Nu exista produsul specificat');
  DBMS_OUTPUT.PUT_LINE(SQLCODE||' ---> ' ||SQLERRM);
END;
/
```

Exemplul 7.6 – persistența variabilelor

```
CREATE OR REPLACE PACKAGE variabila_globala
IS
  v VARCHAR2(100) DEFAULT 'V este modificat de ';
END;
/

-- 2 sesiuni - 2 utilizatori
--sesiune 1 – utilizator1
BEGIN
 variabila_globala.v := variabila_globala.v
           ||USER ||' in sesiunea 1';
 DBMS_OUTPUT.PUT_LINE(variabila_globala.v);
END;
/

BEGIN
 DBMS_OUTPUT.PUT_LINE(variabila_globala.v);
END;
/

--sesiune 2 - utilizator2
BEGIN
   curs_plsql.variabila_globala.v :=
   curs_plsql.variabila_globala.v ||USER
        ||' in sesiunea 2';
   DBMS_OUTPUT.PUT_LINE(
        curs_plsql.variabila_globala.v);
END;
/

BEGIN
 DBMS_OUTPUT.PUT_LINE(
       curs_plsql.variabila_globala.v);
END;
/

-- 2 sesiuni - acelasi utilizator
--sesiune 1 – utilizator1
--acelasi cod ca mai sus

--sesiune 2 - utilizator1
BEGIN
 variabila_globala.v := variabila_globala.v ||
           USER ||' in sesiunea 2';
 DBMS_OUTPUT.PUT_LINE(variabila_globala.v);
END;
/
```

Exemplul 7.9

```
BEGIN
  DBMS_OUTPUT.PUT('Astazi ');
  DBMS_OUTPUT.PUT('este ');
  DBMS_OUTPUT.PUT(SYSDATE);
  DBMS_OUTPUT.NEW_LINE;
END;
/

BEGIN
  DBMS_OUTPUT.PUT_LINE('Astazi este '|| SYSDATE);
  DBMS_LOCK.SLEEP(5);
END;
/
```

Exemplul 7.10

```
DECLARE
-- paramentri de tip OUT pentru procedura GET_LINE
   linie1 VARCHAR2(255);
   stare1 INTEGER;
   linie2 VARCHAR2(255);
   stare2 INTEGER;
   linie3 VARCHAR2(255);
   stare3 INTEGER;

v_id         NUMBER(4);
v_denumire   VARCHAR2(100);
v_pret       produse.pret_unitar%TYPE;

BEGIN
  SELECT id_produs,denumire, pret_unitar
  INTO   v_id, v_denumire, v_pret
  FROM   produse
  WHERE  id_produs = 1000;

-- se introduce o linie in buffer fara caracter
-- de terminare linie
   DBMS_OUTPUT.PUT(' 1 - '||v_id|| ' ');

-- se incearca extragerea liniei introduse
-- in buffer si starea acesteia
   DBMS_OUTPUT.GET_LINE(linie1,stare1);

-- se depune informatie pe aceeasi linie in buffer
   DBMS_OUTPUT.PUT(' 2 - '||v_denumire|| ' ');

-- se inchide linia depusa in buffer si se extrage
-- linia din buffer
   DBMS_OUTPUT.NEW_LINE;
   DBMS_OUTPUT.GET_LINE(linie2,stare2);

-- se introduc informatii pe aceeasi linie
-- si se afiseaza informatia
   DBMS_OUTPUT.PUT_LINE(' 3 - ID: ' ||v_id|| ' Pret: '
                        || v_pret);
   DBMS_OUTPUT.GET_LINE(linie3,stare3);
```

```
-- se afiseaza ceea ce s-a extras
   DBMS_OUTPUT.PUT_LINE('linie1 = '|| linie1||
                        '; stare1 = '||stare1);
   DBMS_OUTPUT.PUT_LINE('linie2 = '|| linie2||
                        '; stare2 = '||stare2);
   DBMS_OUTPUT.PUT_LINE('linie3 = '|| linie3||
                        '; stare3 = '||stare3);
END;
/
```

Exemplul 7.11

```
DECLARE
-- parametru de tip OUT pentru GET_LINES
-- colectie de siruri de caractere
   linii DBMS_OUTPUT.CHARARR;
-- paramentru de tip IN OUT pentru GET_LINES
   nr_linii INTEGER;

v_id         NUMBER(4);
v_denumire   VARCHAR2(100);
v_pret       produse.pret_unitar%TYPE;

BEGIN
  SELECT id_produs,denumire, pret_unitar
  INTO   v_id, v_denumire, v_pret
  FROM   produse
  WHERE  id_produs = 1000;

-- se mareste dimensiunea bufferului
   DBMS_OUTPUT.ENABLE(1000000);
   DBMS_OUTPUT.PUT(' 1 - ID: '||v_id|| ' ');
   DBMS_OUTPUT.PUT(' 2 - DEN: '||v_denumire|| ' ');
   DBMS_OUTPUT.NEW_LINE;
   DBMS_OUTPUT.PUT_LINE(' 3 - ID: ' ||v_id||
                ' PRET:  ' || v_pret);
   DBMS_OUTPUT.PUT_LINE(' 4 - ID: ' ||v_id||
     ' DEN: '|| v_denumire||' PRET: ' ||v_pret);
-- se afiseaza ceea ce s-a extras
   nr_linii := 4;
   DBMS_OUTPUT.GET_LINES(linii,nr_linii);
   DBMS_OUTPUT.PUT_LINE('In buffer sunt '||
                        nr_linii ||' linii');
FOR i IN 1..nr_linii LOOP
       DBMS_OUTPUT.put_line('Linia '||i ||': '||
                            linii(i));
   END LOOP;
END;
/
```

Exemplul 7.12

```
CREATE OR REPLACE PROCEDURE valoare_vanzari_per_zi
IS
  valoare NUMBER(10);
BEGIN
   SELECT NVL(SUM(cantitate*pret),0) INTO valoare
   FROM   facturi_produse fp, facturi f
   WHERE  f.id_factura = fp.id_factura
   AND    data=SYSDATE;
   DBMS_OUTPUT.PUT_LINE('Pana la aceasta ora '||
     's-au efectuat vanzari in valoare de '||
      valoare ||'lei');
END;
/

--varianta 1 - definire job

VARIABLE nr_job NUMBER

BEGIN
 DBMS_JOB.SUBMIT(
  -- întoarce numărul jobului,
  -- printr-o variabilă de legătură
  JOB => :nr_job,

  -- codul PL/SQL care trebuie executat
  WHAT => 'valoare_vanzari_per_zi;',

  -- data de start a execuţiei (dupa 3 secunde)
  NEXT_DATE => SYSDATE+3/86400,

  -- intervalul de timp la care se repetă
  -- execuţia = 3secunde
  INTERVAL => 'SYSDATE+3/86400');
END;
/

-- numarul jobului
PRINT nr_job;

-- informatii despre joburi
SELECT JOB, NEXT_DATE, WHAT
FROM   USER_JOBS;

-- lansarea jobului la momentul dorit
BEGIN
   -- presupunand ca jobul are codul 90 atunci:
   DBMS_JOB.RUN(job => 90);
END;
/
```

```
-- stergerea unui job
BEGIN
   DBMS_JOB.REMOVE(job=>90);
END;
/

SELECT JOB, NEXT_DATE, WHAT
FROM   USER_JOBS;


--varianta 2 - definire job

CREATE OR REPLACE PACKAGE pachet_job
IS
  nr_job NUMBER;
  FUNCTION obtine_job RETURN NUMBER;
END;
/

CREATE OR REPLACE PACKAGE body pachet_job
IS
  FUNCTION obtine_job RETURN NUMBER IS
  BEGIN
    RETURN nr_job;
  END;
END;
/

BEGIN
  DBMS_JOB.SUBMIT(
  -- întoarce numărul jobului,
  -- printr-o variabilă
  JOB =>  pachet_job.nr_job,

  -- codul PL/SQL care trebuie executat
  WHAT => 'valoare_vanzari_per_zi;',

-- data de start a execuţiei (dupa 3 secunde)
  NEXT_DATE => SYSDATE+3/86400,

  -- intervalul de timp la care se repetă
  -- execuţia = 3secunde
  INTERVAL => 'SYSDATE+3/86400');
END;
/

-- informatii despre joburi
SELECT JOB, NEXT_DATE, WHAT
FROM   USER_JOBS
WHERE  JOB = pachet_job.obtine_job;
```

```
-- lansarea jobului la momentul dorit
BEGIN
   DBMS_JOB.RUN(JOB => pachet_job.obtine_job);
END;
/
-- stergerea unui job
BEGIN
   DBMS_JOB.REMOVE(JOB=>pachet_job.obtine_job);
END;
/
SELECT JOB, NEXT_DATE, WHAT
FROM   USER_JOBS
WHERE  JOB = pachet_job.obtine_job;
```

Exemplul 7.14

```
CREATE OR REPLACE PROCEDURE scriu_fisier
 (director VARCHAR2,
  fisier   VARCHAR2)
IS
  v_file UTL_FILE.FILE_TYPE;
  CURSOR c IS
     SELECT RPAD(denumire,50) denumire,
            SUM(cantitate) cantitate_totala
     FROM   facturi_produse fp, produse p
     WHERE  fp.id_produs = p.id_produs
     GROUP BY denumire
     HAVING  SUM(cantitate) > 500
     ORDER BY SUM(cantitate) DESC;
v_lista c%ROWTYPE;

BEGIN
 v_file:=UTL_FILE.FOPEN(director, fisier, 'w');
 UTL_FILE.PUTF(v_file,
              'CANTITATI VANDUTE PER PRODUS
                        \nRAPORT GENERAT PE ');
 UTL_FILE.PUT(v_file, SYSDATE);
 UTL_FILE.NEW_LINE(v_file);
 UTL_FILE.NEW_LINE(v_file);
 UTL_FILE.PUTF(v_file,
     'Denumire produs           Cantitate');
 UTL_FILE.NEW_LINE(v_file);
 UTL_FILE.PUTF(v_file,
       '----------------- -----------------');

 OPEN c;
 LOOP
     FETCH c INTO v_lista;
     EXIT WHEN c%NOTFOUND;

    UTL_FILE.NEW_LINE(v_file);
    UTL_FILE.PUT(v_file, v_lista.denumire);
```

```
    UTL_FILE.PUT(v_file, '          ');
    UTL_FILE.PUT(v_file, v_lista.cantitate_totala);
 END LOOP;
 CLOSE c;
 UTL_FILE.FCLOSE(v_file);
END;
/
--creare director la nivelul sistemului de operare
--D:/OracleW7/Directory/curs_plsql

--setare valoare parametru de initializare Oracle
--utl_file_dir = D:/OracleW7/Directory/curs_plsql

--atribuire privilegiu CREATE ANY DIRECTORY
--utilizatorului
--conectare user sys
GRANT CREATE ANY DIRECTORY TO curs_plsql;

--conectare user curs_plsql
--definire director
CREATE DIRECTORY curs_plsql
AS 'D:/OracleW7/Directory/curs_plsql';

--apel procedura
EXECUTE scriu_fisier(
        'D:\OracleW7\Directory\curs_plsql',
        'ex7_13.txt');
```

Exemplul 7.15

```
DECLARE
  comanda     VARCHAR2(200);
  nume_cursor NUMBER;
BEGIN
  comanda := 'CREATE TABLE test (cod number(4))';
  nume_cursor := DBMS_SQL.OPEN_CURSOR;
  DBMS_SQL.PARSE(nume_cursor, comanda,
                 DBMS_SQL.NATIVE);
  DBMS_SQL.CLOSE_CURSOR(nume_cursor);
END;
/
```

Exemplul 7.16

```
CREATE OR REPLACE PROCEDURE sterge_linii
  (nume_tabel VARCHAR2, nr_linii OUT NUMBER) AS
  nume_cursor INTEGER;
BEGIN
  nume_cursor := DBMS_SQL.OPEN_CURSOR;
  DBMS_SQL.PARSE (nume_cursor,
                  'DELETE FROM ' ||nume_tabel,
                  DBMS_SQL.V7);
  nr_linii := DBMS_SQL.EXECUTE (nume_cursor);
  DBMS_SQL.CLOSE_CURSOR (nume_cursor);
END;
/
-- DBMS_SQL.V7 reprezintă modul (versiunea 7)
-- în care Oracle -- tratează comenzile SQL

VARIABLE linii_sterse NUMBER
EXECUTE sterge_linii('produse_test',:linii_sterse)
PRINT linii_sterse
```

Exemplul 7.17

```
DECLARE
   sir VARCHAR2(50);
   bloc VARCHAR2(500);
BEGIN
-- creare tabel
EXECUTE IMMEDIATE
'CREATE TABLE tabel (col VARCHAR2(15))';
--inserare in tabel
FOR i IN 1..10 LOOP
   sir := 'INSERT INTO tabel
           VALUES (''Contor ' || i || ''')';
   EXECUTE IMMEDIATE sir;
END LOOP;
-- tiparire continut tabel
bloc := 'BEGIN
           FOR i IN (SELECT * FROM tabel) LOOP
             DBMS_OUTPUT.PUT_LINE (i.col);
           END LOOP;
         END;';
EXECUTE IMMEDIATE bloc;
-- stergere tabel
EXECUTE IMMEDIATE 'DROP TABLE tabel';
END;
/
```

Exemplul 7.18

```
CREATE OR REPLACE PROCEDURE
   sterg_tabel(nume VARCHAR2) IS
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE '||nume||
         ' CASCADE CONSTRAINTS';
  -- EXECUTE IMMEDIATE
  -- 'DROP TABLE :n CASCADE CONSTRAINTS'
  -- USING nume;
END;
/
EXECUTE sterg_tabel('nume_tabel')

CREATE OR REPLACE PROCEDURE sterg_tabele
IS
BEGIN
  FOR I IN (SELECT TABLE_NAME
            FROM   USER_TABLES) LOOP
     sterg_tabel(i.table_name);
  END LOOP;
END;
/
```

Exemplul 7.19

```
DECLARE
    TYPE tip_imb IS TABLE OF
         produse.id_produs%TYPE;
    t           tip_imb;
    v_procent   NUMBER(3,2) := &p_procent;
    v_categorie VARCHAR2(50) := '&p_categorie';
    comanda     VARCHAR2(500);
BEGIN
comanda :=  'UPDATE produse
             SET pret_unitar =
                 pret_unitar*(1 + :p)
             WHERE id_categorie =
                 (SELECT id_categorie
                  FROM   categorii
                  WHERE  denumire = :c)
             RETURNING id_produs INTO :tablou';
EXECUTE IMMEDIATE comanda
USING v_procent, v_categorie
RETURNING BULK COLLECT INTO t;
FOR i IN 1..t.LAST LOOP
   DBMS_OUTPUT.PUT_LINE (t(i));
END LOOP;
END;
/
```

Exemplul 7.20

```
DECLARE
    TYPE tip_imb IS TABLE OF produse%ROWTYPE;
    t           tip_imb;
    v_categorie VARCHAR2(50) := '&p_categorie';
    comanda     VARCHAR2(500);
BEGIN
comanda :=  'SELECT *
             FROM   produse
             WHERE id_categorie =
                (SELECT id_categorie
                 FROM   categorii
                 WHERE  denumire = :c)';
EXECUTE IMMEDIATE comanda
BULK COLLECT INTO t USING v_categorie;
FOR i IN 1..t.LAST LOOP
   DBMS_OUTPUT.PUT_LINE (t(i).denumire);
END LOOP;
END;
/
```

Exemplul 7.21

```
CREATE OR REPLACE PROCEDURE
  colecteaza_statistici
  (p_obiect_tip  IN VARCHAR2,
   p_obiect_nume IN VARCHAR2)
IS
BEGIN
  DBMS_DDL.ANALYZE_OBJECT(p_obiect_tip, USER,
             UPPER(p_obiect_nume), 'COMPUTE');
END;
/

EXECUTE colecteaza_statistici ('TABLE', 'PRODUSE')

SELECT LAST_ANALYZED, NUM_ROWS, BLOCKS,
       EMPTY_BLOCKS, AVG_ROW_LEN
FROM   USER_TABLES
WHERE  TABLE_NAME = 'PRODUSE';
```