# Definire pachet

|  |
| --- |
| CREATE OR REPLACE PACKAGE hw\_s9 AS      SUBTYPE id\_client\_t IS clienti.id\_client%TYPE;      SUBTYPE id\_factura\_t IS facturi.id\_factura%TYPE;      SUBTYPE id\_casa\_t IS NUMBER(38); -- tabelul se numeste 'case'...      SUBTYPE id\_categorie\_t IS categorii.id\_categorie%TYPE;      TYPE lista\_facturi IS TABLE OF id\_factura\_t;      -- Activitatea unui client la o anumită casă - facturile achitate și neachitate, si sumele.      TYPE activitate\_client IS RECORD (id\_client id\_client\_t,suma\_achitate NUMBER(10),suma\_neachitate NUMBER(10),achitate lista\_facturi,neachitate lista\_facturi);      -- Lista activităților tuturor clienților la o casă.      TYPE lista\_activitate IS TABLE OF activitate\_client;      TYPE lista\_activitate\_case IS TABLE OF lista\_activitate INDEX BY BINARY\_INTEGER;      FUNCTION get\_activitate\_case RETURN lista\_activitate\_case;      TYPE nr\_facturi\_client IS RECORD (id\_client id\_client\_t, facturi BINARY\_INTEGER);      CURSOR c\_nr\_facturi RETURN nr\_facturi\_client;      TYPE total\_factura IS RECORD (id\_factura id\_factura\_t, total BINARY\_INTEGER);      CURSOR c\_pret\_facturi(v\_id\_client id\_client\_t) RETURN total\_factura;      PROCEDURE print\_achizitii\_client\_facturi\_max;      input\_invalid\_pt\_sql\_dinamic EXCEPTION;      PRAGMA EXCEPTION\_INIT (input\_invalid\_pt\_sql\_dinamic, -20000);      /\* 1 = salariu, 2 = job, 3 = proiecte \*/      /\* 1 = ASC, 2 = DESC \*/      PROCEDURE print\_employees\_dynamic\_sorting(v\_alegere NUMBER, v\_directie NUMBER);      produs\_nu\_exista EXCEPTION;      PRAGMA EXCEPTION\_INIT (produs\_nu\_exista, -20001);      PROCEDURE proc\_ex1 (v\_id produse.id\_produs%TYPE, v\_procent NUMBER);      PROCEDURE forall\_in\_category\_proc\_ex1(id\_categorie\_ales id\_categorie\_t);  END hw\_s9;  CREATE OR REPLACE PACKAGE BODY hw\_s9 AS      CURSOR c\_nr\_facturi RETURN nr\_facturi\_client      IS          SELECT id\_client, COUNT(\*) facturi          FROM facturi          GROUP BY id\_client;      CURSOR c\_pret\_facturi(v\_id\_client id\_client\_t) RETURN total\_factura      IS          SELECT id\_factura, SUM(pret\_facturare) total          FROM facturi          JOIN facturi\_contin\_produse          USING (id\_factura)          WHERE id\_client = v\_id\_client          GROUP BY id\_factura;      FUNCTION get\_activitate\_case      RETURN lista\_activitate\_case      IS          TYPE info\_factura IS RECORD ( id\_factura id\_factura\_t,                                        id\_casa id\_casa\_t,                                        id\_client id\_client\_t,                                        status FACTURI.status%TYPE,                                        suma\_facturare FACTURI\_CONTIN\_PRODUSE.pret\_facturare%TYPE);          v\_factura info\_factura;          v\_casa\_curenta id\_casa\_t := -1;          v\_activitate activitate\_client := activitate\_client();          v\_retval lista\_activitate\_case;          PROCEDURE init\_activitate(casa IN id\_casa\_t, client IN id\_client\_t) IS          BEGIN              v\_casa\_curenta := casa;              v\_activitate := activitate\_client(client, 0, 0, lista\_facturi(), lista\_facturi());          END;          PROCEDURE save\_activitate IS          BEGIN              IF v\_casa\_curenta = -1              THEN                  RETURN;              END IF;              IF NOT v\_retval.exists(v\_casa\_curenta)              THEN                  v\_retval(v\_casa\_curenta) := lista\_activitate();              END IF;              v\_retval(v\_casa\_curenta).extend(1);              v\_retval(v\_casa\_curenta)(v\_retval(v\_casa\_curenta).last) := v\_activitate;              v\_activitate := activitate\_client();              v\_casa\_curenta := -1;          END;      BEGIN          FOR v\_factura IN (SELECT id\_factura, id\_casa, id\_client, status, SUM(pret\_facturare) suma\_facturare                            FROM FACTURI                            JOIN FACTURI\_CONTIN\_PRODUSE                            USING (id\_factura)                            GROUP BY (id\_factura, id\_casa, id\_client, status)                            ORDER BY id\_casa, id\_client, id\_factura)          LOOP              IF v\_casa\_curenta = -1 OR v\_activitate.ID\_CLIENT != v\_factura.ID\_CLIENT OR v\_casa\_curenta != v\_factura.ID\_CASA              THEN                  save\_activitate();                  init\_activitate(v\_factura.ID\_CASA, v\_factura.ID\_CLIENT);              END IF;              IF v\_factura.STATUS = 'achitat'              THEN                  v\_activitate.suma\_achitate := v\_activitate.suma\_achitate + v\_factura.suma\_facturare;                  v\_activitate.ACHITATE.EXTEND;                  v\_activitate.ACHITATE(v\_activitate.ACHITATE.LAST) := v\_factura.ID\_FACTURA;              ELSIF v\_factura.STATUS = 'neachitat'              THEN                  v\_activitate.suma\_neachitate := v\_activitate.suma\_neachitate + v\_factura.suma\_facturare;                  v\_activitate.NEACHITATE.EXTEND;                  v\_activitate.NEACHITATE(v\_activitate.NEACHITATE.LAST) := v\_factura.ID\_FACTURA;              ELSE                  DBMS\_OUTPUT.PUT\_LINE('WARNING: status factura "'||v\_factura.STATUS||'" necunoscut');              END IF;          END LOOP;          save\_activitate();          RETURN v\_retval;      END get\_activitate\_case;      PROCEDURE print\_achizitii\_client\_facturi\_max      IS          v\_client id\_client\_t := -1;          v\_max\_count BINARY\_INTEGER := 0;          v\_este\_ambiguu BOOLEAN := false;      BEGIN          FOR client IN c\_nr\_facturi          LOOP              DBMS\_OUTPUT.PUT\_LINE(client.id\_client||' '||client.facturi);              IF client.facturi > v\_max\_count              THEN                  v\_max\_count := client.facturi;                  v\_client := client.id\_client;                  v\_este\_ambiguu := FALSE;              ELSIF client.facturi = v\_max\_count              THEN                  v\_este\_ambiguu := TRUE;              END IF;          END LOOP;          IF v\_este\_ambiguu THEN              dbms\_output.PUT\_LINE('Warning: multiplii clienți cu nr. max de facturi. Rezultatul va fi imprevizibil.');          END IF;          dbms\_output.PUT\_LINE('Clientul '||v\_client||' are nr. maxim de facturi: '||v\_max\_count||'.');          FOR factura IN c\_pret\_facturi(v\_client)          LOOP              dbms\_output.PUT\_LINE('Factura '||factura.id\_factura||' => '||factura.total||' lei');          END LOOP;      END print\_achizitii\_client\_facturi\_max;      PROCEDURE print\_employees\_dynamic\_sorting(v\_alegere NUMBER, v\_directie NUMBER)      IS          v\_cursor SYS\_REFCURSOR;          v\_emp\_id EMPLOYEES.employee\_id%TYPE;          v\_emp\_name VARCHAR2(100);          v\_emp\_salary EMPLOYEES.salary%TYPE;          v\_emp\_job EMPLOYEES.job\_id%TYPE;          v\_nr\_proiecte BINARY\_INTEGER;          v\_order\_by VARCHAR(20);          v\_order\_asc\_desc VARCHAR(5);          input\_invalid\_pt\_sql\_dinamic EXCEPTION;      BEGIN          CASE v\_alegere              WHEN 1 THEN v\_order\_by := 'salary';              WHEN 2 THEN v\_order\_by := 'job\_id';              WHEN 3 THEN v\_order\_by := 'nr\_proiecte';              ELSE raise\_application\_error(-20000, 'v\_alegere invalid');          END CASE;          CASE v\_directie              WHEN 1 THEN v\_order\_asc\_desc := 'ASC';              WHEN 2 THEN v\_order\_asc\_desc := 'DESC';              ELSE raise\_application\_error(-20000, 'v\_directie invalid');          END CASE;          OPEN v\_cursor FOR              'WITH project\_counts AS (                  SELECT employee\_id, COUNT(\*) nr\_proiecte FROM works\_on                  GROUP BY employee\_id              )              SELECT employee\_id, first\_name || '' '' || last\_name emp\_name, salary, job\_id, nr\_proiecte              FROM employees              JOIN project\_counts              USING (employee\_id)              ORDER BY ' || v\_order\_by || ' ' || v\_order\_asc\_desc;          LOOP              FETCH v\_cursor INTO v\_emp\_id, v\_emp\_name, v\_emp\_salary, v\_emp\_job, v\_nr\_proiecte;              EXIT WHEN v\_cursor%NOTFOUND;              dbms\_output.PUT\_LINE(v\_emp\_id||': '||v\_emp\_salary||', cu salariu '||v\_emp\_salary||', cu job '||v\_emp\_job||', nr. proiecte la care lucreaza: '||v\_nr\_proiecte);          END LOOP;          CLOSE v\_cursor;      END print\_employees\_dynamic\_sorting;      PROCEDURE proc\_ex1(v\_id produse.id\_produs%TYPE, v\_procent NUMBER)      IS      BEGIN          UPDATE produse          SET pret\_unitar = pret\_unitar + pret\_unitar \* v\_procent          WHERE id\_produs = v\_id;      EXCEPTION          WHEN NO\_DATA\_FOUND THEN              RAISE\_APPLICATION\_ERROR (-20001,'Nu exista produsul');      END;      PROCEDURE forall\_in\_category\_proc\_ex1(id\_categorie\_ales id\_categorie\_t)      IS      BEGIN          FOR produs IN (              SELECT id\_produs              FROM PRODUSE              WHERE id\_categorie = id\_categorie\_ales          )          LOOP              DBMS\_OUTPUT.PUT\_LINE('Scădere preț - produs '||produs.id\_produs);              proc\_ex1(produs.id\_produs, -0.05);          END LOOP;      END;  END hw\_s9; |

# Testare pachet

|  |
| --- |
| DECLARE      v\_activitati\_case hw\_s9.lista\_activitate\_case;  BEGIN      v\_activitati\_case := hw\_s9.get\_activitate\_case();      FOR v\_row IN (          SELECT id\_casa FROM case      )      LOOP          IF NOT v\_activitati\_case.exists(v\_row.id\_casa)          THEN              dbms\_output.PUT\_LINE('Fara date pt. casa '||v\_row.id\_casa);              CONTINUE;          END IF;          dbms\_output.PUT\_LINE('--- Casa '||v\_row.id\_casa||': ---');          FOR i IN 1..v\_activitati\_case(v\_row.id\_casa).last          LOOP              DECLARE                  v\_act hw\_s9.activitate\_client := v\_activitati\_case(v\_row.id\_casa)(i);              BEGIN                  dbms\_output.PUT\_LINE('    Client '||v\_act.id\_client||': '||v\_act.achitate.count||' achitate ('||v\_act.suma\_achitate||' lei'                      ||'), '||v\_act.neachitate.count||' neachitate ('||v\_act.suma\_neachitate||' lei)');              END;          END LOOP;      END LOOP;  END;  BEGIN      hw\_s9.print\_achizitii\_client\_facturi\_max();  END;  BEGIN      hw\_s9.print\_employees\_dynamic\_sorting(1, 1);      hw\_s9.print\_employees\_dynamic\_sorting(1, 999);  END;  BEGIN      FOR i IN 1..10      LOOP          hw\_s9.forall\_in\_category\_proc\_ex1(i);      END LOOP;  END;  ROLLBACK;  DROP PACKAGE hw\_s9; |

# Screenshot-uri

