

**Exemplul 4.6**

```

DECLARE
  v_categ categorii%ROWTYPE;
  v_categ2 categorii%ROWTYPE;
  v_categ_modific v_categ%ROWTYPE;
  v_categ_null categorii%ROWTYPE;
BEGIN
  v_categ.denumire := 'Categorie noua';
  v_categ.nivel :=1;
  SELECT MAX(id_categorie)+1 INTO v_categ.id_categorie
  FROM   categorii;

  INSERT INTO categorii VALUES v_categ;

  SELECT * INTO v_categ2
  FROM   categorii
  WHERE  id_categorie= v_categ.id_categorie;

  DBMS_OUTPUT.PUT_LINE ('Ati inserat: '||
    v_categ2.id_categorie || ' ' || v_categ2.denumire ||
    ' ' || v_categ2.nivel || ' ' ||
    NVL(v_categ2.id_parinte,0));

  v_categ_modific := v_categ;

  v_categ_modific.id_categorie := v_categ.id_categorie + 1;

  UPDATE categorii
  SET ROW = v_categ_modific
  WHERE id_categorie= v_categ.id_categorie;

  SELECT * INTO v_categ2
  FROM   categorii
  WHERE  id_categorie= v_categ_modific.id_categorie;

  DBMS_OUTPUT.PUT_LINE ('Ati modificat in: '||
    v_categ_modific.id_categorie || ' ' ||
    v_categ_modific.denumire || ' ' ||
    v_categ_modific.nivel || ' ' ||
    NVL(v_categ_modific.id_parinte,0));

  v_categ2 := v_categ_null;

  DELETE FROM categorii
  WHERE  id_categorie= v_categ_modific.id_categorie
  RETURNING id_categorie, denumire, nivel, id_parinte
  INTO v_categ2;

  DBMS_OUTPUT.PUT_LINE ('Ati sters linia: '||
    v_categ2.id_categorie || ' ' || v_categ2.denumire ||
    ' ' || v_categ2.nivel || ' ' ||
    NVL(v_categ2.id_parinte,0));

END;

```

**Exemplul 4.9**

```

DECLARE
    TYPE tab_ind IS TABLE OF tip_plata.descriere%TYPE
        INDEX BY PLS_INTEGER;
    t    tab_ind;
BEGIN
    -- atribuire valori
    DELETE FROM tip_plata
    WHERE id_tip_plata NOT IN (SELECT id_tip_plata FROM facturi)
    RETURNING descriere BULK COLLECT INTO t;

    --parcuregere
    DBMS_OUTPUT.PUT_LINE('Tabloul are ' || t.COUNT ||
        ' elemente:');
    FOR i IN t.FIRST..t.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(t(i));
    END LOOP;
    ROLLBACK;
END;

```

**Exemplul 4.10**

```

DECLARE
    TYPE rec IS RECORD (id tip_plata.id_tip_plata%TYPE,
        den tip_plata.descriere%TYPE);
    TYPE tab_ind IS TABLE OF rec
        INDEX BY PLS_INTEGER;
    t    tab_ind;
BEGIN
    -- atribuire valori
    DELETE FROM tip_plata
    WHERE id_tip_plata NOT IN (SELECT id_tip_plata
        FROM facturi)
    RETURNING id_tip_plata, descriere BULK COLLECT INTO t;

    --parcuregere
    DBMS_OUTPUT.PUT_LINE('Tabloul are ' || t.COUNT
        || ' elemente:');
    FOR i IN t.FIRST..t.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(t(i).id || ' ' || t(i).den);
    END LOOP;
    ROLLBACK;
END;

```

**Exemplul 4.11**

```

DECLARE
    TYPE tab_ind IS TABLE OF NUMBER INDEX BY VARCHAR2(1);
    t tab_ind;
    i varchar2(1);

BEGIN
    -- initializare
    t('a') := ASCII('a');
    t('A') := ASCII('A');
    t('b') := ASCII('b');
    t('B') := ASCII('B');
    t('x') := ASCII('x');
    t('X') := ASCII('X');

    -- parcurgere
    i := t.FIRST;
    WHILE i IS NOT NULL LOOP
        DBMS_OUTPUT.PUT_LINE('t('||i||')='||t(i));
        i := t.NEXT(i);
    END LOOP;
END;

```

**Exemplul 4.12**

```

DECLARE
    TYPE tab_imb IS TABLE OF NUMBER;
    t tab_imb := tab_imb(1,2,3,4,5);
    t_null tab_imb;

BEGIN
    -- atribuire valori
    t.EXTEND(5);
    FOR i IN 6..10 LOOP
        t(i):=i;
    END LOOP;
    --parcurgere
    DBMS_OUTPUT.PUT('Tabloul are ' || t.COUNT || ' elemente:');
    FOR i IN t.FIRST..t.LAST LOOP
        DBMS_OUTPUT.PUT(t(i) || ' ');
    END LOOP;
    DBMS_OUTPUT.NEW_LINE;

    t := t_null;
    IF t IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('Colectie atomic null');
    END IF;
END;

```

**Exemplul 4.14**

```
DECLARE
  -- tipul a fost definit la ex4.13
  v_grupe      t_imb_categ := t_imb_categ();
  v_id_categ    raion_grupe_imb.id_categorie%TYPE;
  v_den        raion_grupe_imb.denumire%TYPE;
BEGIN
  SELECT * INTO v_id_categ, v_den, v_grupe
  FROM   raion_grupe_imb
  WHERE  id_categorie = 1;

  DBMS_OUTPUT.PUT_LINE(v_id_categ || ' ' || v_den);
  DBMS_OUTPUT.PUT_LINE('-----');
  FOR i IN 1..v_grupe.LAST LOOP
    DBMS_OUTPUT.PUT_LINE(v_grupe(i));
  END LOOP;
END;
```

**Exemplul 4.18**

```

DECLARE
    TYPE t_imb IS TABLE OF NUMBER(2);

    t t_imb := t_imb();
    t1 t_imb := t_imb(1,2,1,3,3);
    t2 t_imb := t_imb(1,2,4,2);
    t3 t_imb := t_imb(1,2,4);
    t4 t_imb := t_imb(1,2,4);
    t5 t_imb := t_imb(1,2);

BEGIN
    -- IS EMPTY
    IF t IS EMPTY THEN
        DBMS_OUTPUT.PUT_LINE('t nu are elemente');
    END IF;

    -- CARDINALITY
    DBMS_OUTPUT.PUT('t1 are ' || CARDINALITY(t1) ||
        ' elemente: ');
    FOR i IN 1..t1.LAST LOOP
        DBMS_OUTPUT.PUT(t1(i) || ' ');
    END LOOP;
    DBMS_OUTPUT.NEW_LINE;

    DBMS_OUTPUT.PUT('t2 are ' || CARDINALITY(t2) ||
        ' elemente: ');
    FOR i IN 1..t2.LAST LOOP
        DBMS_OUTPUT.PUT(t2(i) || ' ');
    END LOOP;
    DBMS_OUTPUT.NEW_LINE;

    -- SET
    t := SET(t1);
    DBMS_OUTPUT.PUT('t1 fara duplicate: ');
    FOR i IN 1..t.LAST LOOP
        DBMS_OUTPUT.PUT(t(i) || ' ');
    END LOOP;
    DBMS_OUTPUT.NEW_LINE;

    -- MULTISET EXCEPT
    t := t1 MULTISET EXCEPT t2;
    DBMS_OUTPUT.PUT('t1 minus t2: ');
    FOR i IN 1..t.LAST LOOP
        DBMS_OUTPUT.PUT(t(i) || ' ');
    END LOOP;
    DBMS_OUTPUT.NEW_LINE;

    -- MULTISET UNION
    DBMS_OUTPUT.PUT('t1 union distinct t2: ');
    t := t1 MULTISET UNION DISTINCT t2;
    FOR i IN 1..t.LAST LOOP
        DBMS_OUTPUT.PUT(t(i) || ' ');
    END LOOP;
    DBMS_OUTPUT.NEW_LINE;

```

```
-- MULTISSET INTERSECT
t := t1 MULTISSET INTERSECT DISTINCT t2;
DBMS_OUTPUT.PUT('t1 intersect distinct t2 : ');
FOR i IN 1..t.LAST LOOP
    DBMS_OUTPUT.PUT(t(i) || ' ');
END LOOP;
DBMS_OUTPUT.NEW_LINE;

-- test egalitate
IF t2=t3 THEN
    DBMS_OUTPUT.PUT_LINE('t2 = t3');
ELSE
    DBMS_OUTPUT.PUT_LINE('t2 <> t3');
END IF;

IF t3=t4 THEN
    DBMS_OUTPUT.PUT_LINE('t3 = t4');
ELSE
    DBMS_OUTPUT.PUT_LINE('t3 <> t4');
END IF;

-- IN
IF t4 IN (t1,t2,t3) THEN
    DBMS_OUTPUT.PUT_LINE('t4 in (t1,t2,t3)');
ELSE
    DBMS_OUTPUT.PUT_LINE('t4 not in (t1,t2,t3)');
END IF;

-- IS A SET
IF t4 IS A SET THEN
    DBMS_OUTPUT.PUT_LINE('t4 este multime');
ELSE
    DBMS_OUTPUT.PUT_LINE('t4 nu este multime');
END IF;

-- MEMBER OF
IF 2 MEMBER OF t4 THEN
    DBMS_OUTPUT.PUT_LINE('2 este in t4');
ELSE DBMS_OUTPUT.PUT_LINE('2 nu este in t4');
END IF;

-- SUBMULTISSET OF
IF t5 SUBMULTISSET OF t4 THEN
    DBMS_OUTPUT.PUT_LINE('t5 este inclus in t4');
ELSE
    DBMS_OUTPUT.PUT_LINE('t5 nu este inclus in t4');
END IF;
END;
```

**Exemplul 4.25**

```
INSERT INTO CATEGORII(id_categorie)
VALUES(9999);

INSERT INTO produse (id_produs, denumire, id_categorie)
VALUES(9999, 'D9999', 9999);
COMMIT;

SELECT id_produs, denumire, id_categorie
FROM   produse
WHERE  id_categorie in (800, 900, 9999)
ORDER BY 3;

DECLARE
    TYPE tip_vec IS VARRAY(3) OF NUMBER(4);
    v tip_vec := tip_vec(800, 900, 9999);
    eroare EXCEPTION;
    PRAGMA EXCEPTION_INIT(eroare, -24381);
    nr_erori NUMBER;
BEGIN
    FORALL i IN 1..3    SAVE EXCEPTIONS
        DELETE FROM produse
        WHERE id_categorie = v(i);
    END LOOP;

    EXCEPTION
        WHEN eroare THEN
            nr_erori := SQL%BULK_EXCEPTIONS.COUNT;
            DBMS_OUTPUT.PUT_LINE('Numar comenzi esuate: ' || nr_erori);
            FOR i IN 1..nr_erori LOOP
                DBMS_OUTPUT.PUT_LINE('Eroare ' || i ||
                    ' aparuta in timpul iteratiei ' ||
                    SQL%BULK_EXCEPTIONS(i).ERROR_INDEX);
                DBMS_OUTPUT.PUT_LINE('Mesajul erorii: ' ||
                    SQLERRM(-SQL%BULK_EXCEPTIONS(i).ERROR_CODE));
            END LOOP;
END;
/

SELECT id_produs, denumire, id_categorie
FROM   produse
WHERE  id_categorie in (800, 900, 9999)
ORDER BY 3;
```