

### Exemplul 8.1

```
CREATE OR REPLACE TRIGGER trig_ex1
  BEFORE INSERT OR DELETE OR UPDATE on facturi
BEGIN

  IF (TO_CHAR(SYSDATE,'D') = 1)
    OR (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 20)
  THEN
    RAISE_APPLICATION_ERROR(-20001,'Operatiile asupra
      tabelului sunt permise doar in
      programul de lucru!');
  END IF;

END;
/

--stornare factura status = -2:

INSERT INTO facturi(id_factura, id_casa, id_client,
                    data, status, id_tip_plata)
VALUES (171, 402,94, SYSDATE,-2,10);

DROP TRIGGER trig_ex1;
```

### Exemplul 8.2

```
CREATE OR REPLACE TRIGGER trig_ex2
  BEFORE INSERT OR DELETE OR UPDATE on facturi
BEGIN

  IF (TO_CHAR(SYSDATE,'D') = 1)
    OR (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 20)
  THEN
    IF INSERTING THEN
      RAISE_APPLICATION_ERROR(-20001,'Inserarea in tabel
        este permisa doar in timpul programului de lucru!');
    ELSIF DELETING THEN
      RAISE_APPLICATION_ERROR(-20002,'Stergerea din tabel
        este permisa doar in timpul programului de lucru!');
    ELSE
      RAISE_APPLICATION_ERROR(-20003,'Actualizarile in tabel
        sunt permise doar in timpul programului de lucru!');
    END IF;
  END IF;

END;
/
```

### Exemplul 8.3

#### --varianta1

```
CREATE OR REPLACE TRIGGER trig1_ex3
BEFORE UPDATE OF serie ON case
FOR EACH ROW
WHEN (NEW.serie <> OLD.serie)
BEGIN
    RAISE_APPLICATION_ERROR (-20000, 'Nu puteti modifica
                                seria casei fiscale!');
END;
/
```

```
UPDATE case
SET     serie = serie||'_';
```

#### --varianta2

```
CREATE OR REPLACE PROCEDURE proc_trig_ex3
IS
BEGIN
    RAISE_APPLICATION_ERROR (-20000, 'Nu puteti modifica
                                seria casei fiscale!');
END;
/
```

```
CREATE OR REPLACE TRIGGER trig2_ex3
BEFORE UPDATE OF serie ON case
FOR EACH ROW
WHEN (NEW.serie <> OLD.serie)
BEGIN
    proc_trig_ex3;
END;
/
```

#### --varianta3

```
CREATE OR REPLACE TRIGGER trig3_ex3
BEFORE UPDATE OF serie ON case
FOR EACH ROW
WHEN (NEW.serie <> OLD.serie)
CALL  proc_trig_ex3
/
```

#### --varianta4

```
CREATE OR REPLACE TRIGGER trig4_ex3
BEFORE UPDATE OF serie ON case
FOR EACH ROW
BEGIN
    IF :NEW.serie <> :OLD.serie THEN
        RAISE_APPLICATION_ERROR (-20000, 'Nu puteti
                                modifica seria casei fiscale!');
    END IF;
END;
/
```

#### Exemplul 8.4

```
CREATE OR REPLACE TRIGGER verifica_stoc
BEFORE INSERT OR UPDATE OF cantitate on facturi_produce
FOR EACH ROW
DECLARE
    v_limita        produse.stoc_curent%TYPE;
BEGIN
    SELECT stoc_curent-stoc_impus
    INTO   v_limita
    FROM   produse
    WHERE  id_produc IN (:NEW.id_produc, :OLD.id_produc);

    IF :NEW.cantitate - NVL(:OLD.cantitate,0) > v_limita
    THEN
        RAISE_APPLICATION_ERROR(-20000,'Se depaseste '||
            'stocul impus. Cantitate permisa '||v_limita);
    END IF;
END;
/
```

#### Exemplul 8.5

```
CREATE OR REPLACE PROCEDURE modifica_stoc
(v_id produse.id_produc%TYPE, v_cantitate NUMBER)
IS
BEGIN
    UPDATE produse
    SET     stoc_curent = stoc_curent + v_cantitate
    WHERE  id_produc = v_id;
END;
/

CREATE OR REPLACE TRIGGER actualizeaza_stoc
AFTER INSERT OR DELETE OR UPDATE OF cantitate
ON facturi_produce
FOR EACH ROW

BEGIN
    IF INSERTING THEN
        modifica_stoc(:NEW.id_produc,-1*NEW.cantitate);
    ELSIF DELETING THEN
        modifica_stoc(:OLD.id_produc,:OLD.cantitate);
    ELSE
        modifica_stoc(:OLD.id_produc,
            :OLD.cantitate-:NEW.cantitate);
    END IF;
END;
/
```

### Exemplul 8.6

```
-- coloana este populata cu null sau cu 0?
ALTER TABLE categorii
ADD nr_produce NUMBER DEFAULT 0;

-- coloana este populata cu null sau cu 0?
UPDATE categorii c
SET     nr_produce =
        (SELECT COUNT(*)
         FROM   produse
         WHERE  id_categorie = c.id_categorie);

CREATE OR REPLACE VIEW info_categorii_produce
AS
SELECT p.*, c.denumire AS categ_denumire, nivel,
       id_parinte, nr_produce
FROM   produse p, categorii c
WHERE  p.id_categorie = c.id_categorie;

CREATE OR REPLACE TRIGGER actualizeaza_info
INSTEAD OF INSERT OR DELETE OR UPDATE
        ON info_categorii_produce
FOR EACH ROW

DECLARE
    v_nr NUMBER(1);
BEGIN
    IF INSERTING THEN

        SELECT COUNT(*) INTO v_nr
        FROM   categorii
        WHERE  id_categorie = :NEW.id_categorie;

        IF v_nr = 0 THEN
            INSERT INTO categorii
            VALUES (:NEW.id_categorie, :NEW.categ_denumire,
                    :NEW.nivel, :NEW.id_parinte, 1);

            INSERT INTO produse
            VALUES (:NEW.id_produc, :NEW.denumire,
                    :NEW.descriere, :NEW.stoc_curent,
                    :NEW.stoc_impus, :NEW.pret_unitar,
                    :NEW.greutate, :NEW.volum, :NEW.tva,
                    :NEW.id_zona, :NEW.id_um,
                    :NEW.id_categorie, SYSDATE, SYSDATE,
                    :NEW.activ);
        ELSE
            INSERT INTO produse
            VALUES (:NEW.id_produc, :NEW.denumire,
                    :NEW.descriere, :NEW.stoc_curent,
                    :NEW.stoc_impus, :NEW.pret_unitar,
```

```

:NEW.greutate, :NEW.volum,
:NEW.tva, :NEW.id_zona, :NEW.id_um,
:NEW.id_categorie, SYSDATE, SYSDATE,
:NEW.activ);

UPDATE categorii
SET    nr_produce = nr_produce+1
WHERE  id_categorie = :NEW.id_categorie;
END IF;

ELSIF DELETING THEN
DELETE FROM produce
WHERE  id_produs = :OLD.id_produs;

UPDATE categorii
SET    nr_produce = nr_produce-1
WHERE  id_categorie = :OLD.id_categorie;

ELSIF UPDATING('id_categorie') THEN
UPDATE produce
SET    id_categorie = :NEW.id_categorie
WHERE  id_produs = :OLD.id_produs;

UPDATE categorii
SET    nr_produce = nr_produce+1
WHERE  id_categorie = :NEW.id_categorie;

UPDATE categorii
SET    nr_produce = nr_produce-1
WHERE  id_categorie = :OLD.id_categorie;

ELSIF UPDATING('denumire') THEN
UPDATE produce
SET    denumire = :NEW.denumire
WHERE  id_produs = :OLD.id_produs;

ELSE
RAISE_APPLICATION_ERROR(-20000,'Ai voie sa
actualizezi doar categoria sau denumirea
produsului!');
END IF;

END;
/

```

### Exemplul 8.7

```
CREATE TABLE audit_user
(nume_bd          VARCHAR2(50),
 user_logat       VARCHAR2(30),
 eveniment        VARCHAR2(20),
 tip_obiect_referit VARCHAR2(30),
 nume_obiect_referit VARCHAR2(30),
 data            TIMESTAMP(3));

CREATE OR REPLACE TRIGGER audit_schema
  AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
  INSERT INTO audit_user
  VALUES (SYS.DATABASE_NAME, SYS.LOGIN_USER,
          SYS.SYSEVENT, SYS.DICTIONARY_OBJ_TYPE,
          SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP(3));
END;
/

CREATE TABLE tabel (coloana_1 number(2));
ALTER TABLE tabel ADD (coloana_2 number(2));
INSERT INTO tabel VALUES (1,2);
CREATE INDEX ind_tabel ON tabel(coloana_1);
SELECT * FROM audit_user;
```

### Exemplul 8.8

```
CREATE TABLE log_user(nume_user  VARCHAR2(30),
                      data        TIMESTAMP,
                      moment       VARCHAR2(20));

CREATE OR REPLACE PROCEDURE insert_log IS
BEGIN
  INSERT INTO log_user
  VALUES (SYS.LOGIN_USER, SYSTIMESTAMP, 'after logon');
END;
/

CREATE OR REPLACE TRIGGER logon_logoff_after
  AFTER LOGON ON SCHEMA
  CALL insert_log
/

CREATE OR REPLACE TRIGGER logon_logoff_before
  BEFORE LOGOFF ON SCHEMA
BEGIN
  INSERT INTO log_user
  VALUES (SYS.LOGIN_USER, SYSTIMESTAMP, 'before logoff');
END;
/

SELECT * FROM log_user;
```

### Exemplul 8.9

```
CREATE TABLE erori
(nume_bd          VARCHAR2(50),
 user_logat       VARCHAR2(30),
 data             TIMESTAMP(3),
 eroare           VARCHAR2(2000));

CREATE OR REPLACE TRIGGER log_erori
  AFTER SERVERERROR ON SCHEMA
BEGIN
  INSERT INTO erori
    VALUES (SYS.DATABASE_NAME, SYS.LOGIN_USER,
            SYSTIMESTAMP,
            DBMS_UTILITY.FORMAT_ERROR_STACK);
END;
/

CREATE TABLE a (id NUMBER(2));
INSERT INTO a VALUES (123);
ALTER TABLE a DROP (b);
SELECT * FROM abc;
SELECT * FROM erori;
```

### Exemplul 8.10

```
CREATE TABLE erori
(nume_bd          VARCHAR2(50),
 user_logat       VARCHAR2(30),
 data             TIMESTAMP(3),
 eroare           VARCHAR2(2000));

CREATE OR REPLACE TRIGGER log_eroare
  AFTER SERVERERROR ON SCHEMA
BEGIN
  IF IS_SERVERERROR(942) THEN
    INSERT INTO erori
      VALUES (SYS.DATABASE_NAME, SYS.LOGIN_USER,
              SYSTIMESTAMP,
              DBMS_UTILITY.FORMAT_ERROR_STACK);
  END IF;
END;
/

ALTER TABLE ab DROP (b);
SELECT * FROM abc;
SELECT * FROM erori;
```

### Exemplul 8.11

```
CREATE OR REPLACE PROCEDURE create_trigger
(v_nume VARCHAR2)
IS
    sir1 VARCHAR2(4000);
    sir2 LONG;
BEGIN
    SELECT  DESCRIPTION, TRIGGER_BODY
    INTO    sir1,sir2
    FROM    USER_TRIGGERS
    WHERE   TRIGGER_NAME = UPPER(v_nume);

    DBMS_OUTPUT.PUT('CREATE OR REPLACE TRIGGER ' ||
                    sir1);
    DBMS_OUTPUT.PUT_LINE(sir2);
end;
/

EXECUTE create_trigger('verifica_stoc')
```

### Exemplul 8.12

```
-- Trigger-ul realizeaza actualizari in cascada:
-- actualizarea cheii primare din tabelul parinte
-- determina
-- actualizarea cheii externe din tabelul copil

CREATE OR REPLACE TRIGGER modifica_copil
AFTER UPDATE OF id_categorie ON categorii
FOR EACH ROW
BEGIN
    UPDATE produse
    SET     id_categorie = :NEW.id_categorie
    WHERE  id_categorie = :OLD.id_categorie;
END;
/

-- constrangerea de cheie externa este definita
-- (cu optiuni la stegere sau nu)
-- exista produse in categoria 428
-- actualizarea urmatoare este permisa

UPDATE  categorii
SET     id_categorie = 7000
WHERE   id_categorie = 428;
```



### Exemplul 8.13

```
-- Trigger-ul realizeaza actualizari in cascada:
-- actualizarea cheii externe din tabelul copil
-- determina
-- actualizarea cheii primare din tabelul parinte

CREATE OR REPLACE TRIGGER modifica_parinte
  BEFORE UPDATE OF id_categorie ON produse
  FOR EACH ROW
BEGIN
  UPDATE categorii
  SET    id_categorie = :NEW.id_categorie
  WHERE  id_categorie = :OLD.id_categorie;
END;
/

--actualizarea urmatoare este permisa
UPDATE  produse
SET      id_categorie = 7000
WHERE    id_categorie = 428;
```

### Exemplul 8.14

```
-- daca ambii trigger-i definiti anterior ar fi activi
-- simultan, atunci urmatoarele comenzi nu ar fi
-- permise
-- eroarea aparuta
-- "table is mutating,
-- trigger/function may not see it"

UPDATE  categorii
SET      id_categorie = 7000
WHERE    id_categorie = 428;

UPDATE  produse
SET      id_categorie = 7000
WHERE    id_categorie = 428;
```

### Exemplul 8.15

```
CREATE OR REPLACE TRIGGER trig_ex15
BEFORE DELETE ON categorii
FOR EACH ROW
DECLARE
    v_denumire VARCHAR2(50);
BEGIN
    SELECT denumire INTO v_denumire
    FROM    categorii
    WHERE   id_categorie = :OLD.id_categorie;
END;
/

-- trigger-ul consulta tabelul de care este asociat
-- comanda urmatoare nu este permisa
-- eroarea aparuta
-- "table is mutating,
-- trigger/function may not see it"

DELETE FROM categorii WHERE id_categorie = 428;

-- comanda urmatoare este permisa
-- (categoria 7000 nu exista in tabel)
DELETE FROM categorii WHERE id_categorie = 7000;
```

### Exemplul 8.16

```
-- Trigger-ul realizeaza stergeri in cascada:
-- stergerea unei inregistrari din tabelul parinte
-- determina
-- stergerea inregistrarilor copil asociate

CREATE OR REPLACE TRIGGER sterge_copil
BEFORE DELETE ON categorii
FOR EACH ROW
BEGIN
    DELETE FROM produse
    WHERE id_categorie = :OLD.id_categorie;
END;
/

-- Cazul 1 - constrangerea de cheie externa nu are
-- optiuni de stergere specificate
-- urmatoarea comanda este permisa

DELETE FROM categorii WHERE id_categorie = 428;

-- Cazul 2 - constrangerea de cheie externa are
-- optiuni de stergere (CASCADE/SET NULL)
-- comanda urmatoare nu este permisa
-- eroarea aparuta
-- "table is mutating,
-- trigger/function may not see it"

DELETE FROM categorii WHERE id_categorie = 428;
```

### Exemplul 8.17

```
--Varianta 1

CREATE OR REPLACE TRIGGER trig_17
BEFORE INSERT OR UPDATE OF id_client_j
ON pret_preferential
FOR EACH ROW
DECLARE
    nr NUMBER(1);
BEGIN
    SELECT COUNT(*) INTO nr
    FROM   pret_preferential
    WHERE  id_client_j = :NEW.id_client_j
    AND    EXTRACT(YEAR FROM data_in) =
           EXTRACT(YEAR FROM SYSDATE);

    IF nr=3 THEN
        RAISE_APPLICATION_ERROR(-20000,'Clientul are
        deja numarul maxim de promotii permis anual');
    END IF;

END;
/
-- clientul 10 are deja 3 promotii asociate
-- apare mesajul din trigger

INSERT INTO pret_preferential
VALUES (101,0.1, sysdate,sysdate+30, 500, 10);

-- clientul 20 are doar 2 promotii asociate
-- linia este inserata

INSERT INTO pret_preferential
VALUES (101,0.1, sysdate,sysdate+30, 500, 20);
rollback;

--comenzile urmatoare determina eroare mutating

INSERT INTO pret_preferential
SELECT 101,0.1, sysdate,sysdate+30, 500, 20
FROM   DUAL;

UPDATE pret_preferential
SET    id_client_j = 120
WHERE  id_client_j = 70;
```

## --Varianta 2

```
CREATE OR REPLACE PACKAGE pachet
AS
    TYPE tip_rec IS RECORD
        (id pret_preferential.id_client_j%TYPE,
         nr NUMBER(1));
    TYPE tip_ind IS TABLE OF tip_rec
        INDEX BY PLS_INTEGER;
    t tip_ind;
    contor NUMBER(2) := 0;
END;
/

CREATE OR REPLACE TRIGGER trig_17_comanda
BEFORE INSERT OR UPDATE OF id_client_j
ON pret_preferential
BEGIN
    pachet.contor := 0;
    SELECT id_client_j, COUNT(*)
        BULK COLLECT INTO pachet.t
    FROM   pret_preferential
    WHERE  EXTRACT(YEAR FROM data_in) =
        EXTRACT(YEAR FROM SYSDATE)
    GROUP BY id_client_j;
END;
/

CREATE OR REPLACE TRIGGER trig_17_linie
BEFORE INSERT OR UPDATE OF id_client_j
ON pret_preferential
FOR EACH ROW
BEGIN
    FOR i in 1..pachet.t.last LOOP
        IF pachet.t(i).id = :NEW.id_client_j
            AND pachet.t(i).nr + pachet.contor=3 THEN
            RAISE_APPLICATION_ERROR(-20000,'Clientul '||
                :NEW.id_client_j||' depaseste numarul '||
                'maxim de promotii permis anual');
        END IF;
    END LOOP;
    pachet.contor := pachet.contor+1;
END;
/
```

```
-- linia este inserata
INSERT INTO pret_preferential
VALUES (102,0.1, sysdate,sysdate+30, 500, 120);

-- linia este inserate
INSERT INTO pret_preferential
SELECT 103,0.1, sysdate,sysdate+30, 501, 120
FROM    DUAL;

-- se depaseste limita impusa
-- apare mesajul din trigger
INSERT INTO pret_preferential
SELECT * FROM pret_pref;

UPDATE pret_preferential
SET    id_client_j = 120
WHERE  id_client_j = 40;

UPDATE pret_preferential
SET    id_client_j = 210
WHERE  id_client_j in (40, 130,140);
```