Laborator 6 PL/SQL
Declanșatori

Un declanșator este un bloc PL/SQL care se execută automat ori de câte ori are loc un anumit eveniment "declanșator" (de exemplu, inserarea unei linii într-un tabel, ștergerea unor înregistrări etc.)

Tipuri de declanșatori:

- la nivel de bază de date – pot fi declanșați de o comandă *LMD* asupra datelor unui tabel; o comandă *LMD* asupra datelor unei vizualizări; o comandă *LDD* (*CREATE*, *ALTER*, *DROP*) referitoare la anumite obiecte ale schemei sau ale bazei de date; un eveniment sistem (*SHUTDOWN*, *STARTUP*); o acțiune a utilizatorului (*LOGON*, *LOGOFF*); o eroare (*SERVERERROR, SUSPEND*).
- la nivel de aplicație – se declanșează la apariția unui eveniment într-o aplicație particulară.

- Sintaxa comenzii de creare a unui declanșator *LMD* este următoarea:

```
CREATE [OR REPLACE] TRIGGER [schema.]nume_declanșator
{BEFORE | AFTER}
{DELETE | INSERT | UPDATE [OF coloana[, coloana …] ] }
  [OR {DELETE|INSERT|UPDATE [OF coloana[, coloana …]] …}
ON [schema.]nume_tabel
[REFERENCING {OLD [AS] vechi NEW [AS] nou
                | NEW [AS] nou OLD [AS] vechi } ]
[FOR EACH ROW]
[WHEN (condiție) ]
corp_declanșator;
```

- În cazul declanșatorilor *LMD* este important să stabilim:
  - momentul când este executat declanșatorul: BEFORE, AFTER
  - ce fel de acțiuni îl declanșează: INSERT, UPDATE, DELETE
  - tipul declanșatorului: la nivel de instrucțiune sau la nivel de linie (*FOR EACH ROW*).

- Sintaxa comenzii de creare a unui declanșator *INSTEAD OF* este următoarea:

```
CREATE [OR REPLACE] TRIGGER [schema.]nume_trigger
--momentul când este declanșat
INSTEAD OF
--comanda/comenzile care îl declanșează
{ DELETE|INSERT|UPDATE [OF coloana[, coloana …] ] }
  [OR {DELETE|INSERT|UPDATE [OF coloana[, coloana …] ] …}
ON [schema.]nume_vizualizare
[REFERENCING {OLD [AS] vechi NEW [AS] nou
              | NEW [AS] nou OLD [AS] vechi } ]
FOR EACH ROW
[WHEN (condiție) ]
corp_trigger (bloc anonim PL/SQL sau comanda CALL);
```

- Sintaxa comenzii de creare a unui declanșator sistem este următoarea:

```
CREATE [OR REPLACE] TRIGGER [schema.]nume_trigger
{BEFORE | AFTER}
{comenzi_LDD | evenimente_sistem}
ON {DATABASE | SCHEMA}
[WHEN (condiție) ]
corp_trigger;
```

- Informații despre declanșatori se pot obține interogând vizualizările
  - *USER_TRIGGERS, ALL_TRIGGERS, DBA_TRIGGERS*
  - *USER_TRIGGER_COL*

- Dezactivarea, respectiv activarea declanșatorilor se realizează prin următoarele comenzi:

```
ALTER TABLE nume_tabel
DISABLE ALL TRIGGERS;

ALTER TABLE nume_tabel
ENABLE ALL TRIGGERS;

ALTER TRIGGER nume_trig ENABLE;

ALTER TRIGGER nume_trig DISABLE;
```

- Eliminarea unui declanșator se face prin

```
DROP TRIGGER nume_trig;
```

1. Definiți un declanșator care să permită lucrul asupra tabelului emp_*** (INSERT, UPDATE, DELETE) decât în intervalul de ore 8:00 - 20:00, de luni până sâmbătă (**declanșator la nivel de instrucțiune**).

```
CREATE OR REPLACE TRIGGER trig1_***
      BEFORE INSERT OR UPDATE OR DELETE ON emp_***
BEGIN
 IF (TO_CHAR(SYSDATE,'D') = 1)
     OR (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 20)
 THEN
 RAISE_APPLICATION_ERROR(-20001,'tabelul nu poate fi actualizat');
 END IF;
END;
/
DROP TRIGGER trig1_***;
```

2. Definiți un declanșator prin care să nu se permită micșorarea salariilor angajaților din tabelul emp_*** (**declanșator la nivel de linie**).

Varianta 1

```
CREATE OR REPLACE TRIGGER trig21_***
  BEFORE UPDATE OF salary ON emp_***
  FOR EACH ROW
BEGIN
  IF (:NEW.salary < :OLD.salary) THEN
  RAISE_APPLICATION_ERROR(-20002,'salariul nu poate fi micsorat');
```

```
  END IF;
END;
/
UPDATE emp_***
SET    salary = salary-100;

DROP TRIGGER trig21_***;
```

Varianta 2

```
CREATE OR REPLACE TRIGGER trig22_***
  BEFORE UPDATE OF salary ON emp_***
  FOR EACH ROW
  WHEN (NEW.salary < OLD.salary)
BEGIN
  RAISE_APPLICATION_ERROR(-20002,'salariul nu poate fi micsorat');
END;
/
UPDATE emp_***
SET    salary = salary-100;

DROP TRIGGER trig22_***;
```

3. Creați un declanșator care să nu permită mărirea limitei inferioare a grilei de salarizare 1, respectiv micșorarea limitei superioare a grilei de salarizare 7 decât dacă toate salariile se găsesc în intervalul dat de aceste două valori modificate. Se va utiliza tabelul *job_grades*_***.

```
CREATE OR REPLACE TRIGGER trig3_***
  BEFORE UPDATE OF lowest_sal, highest_sal ON job_grades_***
  FOR EACH ROW
DECLARE
  v_min_sal  emp_***.salary%TYPE;
  v_max_sal  emp_***.salary%TYPE;
  exceptie EXCEPTION;
BEGIN
  SELECT MIN(salary), MAX(salary)
  INTO   v_min_sal,v_max_sal
  FROM   emp_***;

  IF (:OLD.grade_level=1) AND  (v_min_sal< :NEW.lowest_sal)
     THEN RAISE exceptie;
  END IF;

  IF (:OLD.grade_level=7) AND  (v_max_sal> :NEW.highest_sal)
     THEN RAISE exceptie;
  END IF;

EXCEPTION
  WHEN exceptie THEN
    RAISE_APPLICATION_ERROR (-20003, 'Exista salarii care se
                             gasesc in afara intervalului');
END;
/
```

```
UPDATE job_grades_***
SET    lowest_sal =3000
WHERE  grade_level=1;

UPDATE job_grades_***
SET    highest_sal =20000
WHERE  grade_level=7;

DROP TRIGGER trig3_***;
```

4. **a.** Creați tabelul *info_dept_**** cu următoarele coloane:
- id (codul departamentului) – cheie primară;
- nume_dept (numele departamentului);
- plati (suma alocată pentru plata salariilor angajaților care lucrează în departamentul respectiv).

b. Introduceți date în tabelul creat anterior corespunzătoare informațiilor existente în schemă.

c. Definiți un declanșator care va actualiza automat câmpul *plati* atunci când se introduce un nou salariat, respectiv se șterge un salariat sau se modifică salariul unui angajat.

```
CREATE OR REPLACE PROCEDURE modific_plati_***
          (v_codd  info_dept_***.id%TYPE,
           v_plati info_dept_***.plati%TYPE) AS
BEGIN
  UPDATE  info_dept_***
  SET     plati = NVL (plati, 0) + v_plati
  WHERE   id = v_codd;
END;
/
```

```
CREATE OR REPLACE TRIGGER trig4_***
  AFTER DELETE OR UPDATE  OR  INSERT OF salary ON emp_***
  FOR EACH ROW
BEGIN
  IF DELETING THEN
     -- se sterge un angajat
     modific_plati_*** (:OLD.department_id, -1*:OLD.salary);
  ELSIF UPDATING THEN
    --se modifica salariul unui angajat
    modific_plati_***(:OLD.department_id,:NEW.salary-:OLD.salary);
  ELSE
    -- se introduce un nou angajat
    modific_plati_***(:NEW.department_id, :NEW.salary);
  END IF;
END;
/
```

```
SELECT * FROM  info_dept_*** WHERE id=90;

INSERT INTO emp_*** (employee_id, last_name, email, hire_date,
                     job_id, salary, department_id)
VALUES (300, 'N1', 'n1@g.com',sysdate, 'SA_REP', 2000, 90);
```

```
SELECT * FROM  info_dept_*** WHERE id=90;

UPDATE emp_***
SET    salary = salary + 1000
WHERE  employee_id=300;

SELECT * FROM  info_dept_*** WHERE id=90;

DELETE FROM emp_***
WHERE  employee_id=300;

SELECT * FROM  info_dept_*** WHERE id=90;

DROP TRIGGER trig4_***;
```

5. **a.** Creați tabelul *info_emp_**** cu următoarele coloane:
- id (codul angajatului) – cheie primară;
- nume (numele angajatului);
- prenume (prenumele angajatului);
- salariu (salariul angajatului);
- id_dept (codul departamentului) – cheie externă care referă tabelul *info_dept_****.

b. Introduceți date în tabelul creat anterior corespunzătoare informațiilor existente în schemă.

c. Creați vizualizarea *v_info_**** care va conține informații complete despre angajați și departamentele acestora. Folosiți cele două tabele create anterior, *info_emp_****, respectiv *info_dept_****.

d. Se pot realiza actualizări asupra acestei vizualizări? Care este tabelul protejat prin cheie? Consultați vizualizarea *user_updatable_columns.*

e. Definiți un declanșator prin care actualizările ce au loc asupra vizualizării se propagă automat în tabelele de bază (**declanșator INSTEAD OF**). Se consideră că au loc următoarele actualizări asupra vizualizării:
- se adaugă un angajat într-un departament deja existent;
- se elimină un angajat;
- se modifică valoarea salariului unui angajat;
- se modifică departamentul unui angajat (codul departamentului).

f. Verificați dacă declanșatorul definit funcționează corect.

g. Modificați declanșatorul definit astfel încât să permită și următoarele operații:
- se adaugă un angajat și departamentul acestuia (departamentul este nou);
- se adaugă doar un departament.

h. Verificați dacă declanșatorul definit funcționează corect.

i. Modificați prin intermediul vizualizării numele unui angajat. Ce observați?

j. Modificați declanșatorul definit anterior astfel încât să permită propagarea în tabelele de bază a actualizărilor realizate asupra numelui și prenumelui angajatului, respectiv asupra numelui de departament.

k. Verificați dacă declanșatorul definit funcționează corect.

```
CREATE OR REPLACE TRIGGER trig5_***
    INSTEAD OF INSERT OR DELETE OR UPDATE ON v_info_***
    FOR EACH ROW
BEGIN
IF INSERTING THEN
    -- inserarea in vizualizare determina inserarea
    -- in info_emp_*** si reactualizarea in info_dept_***
    -- se presupune ca departamentul exista
```

```
ELSIF DELETING THEN
   -- stergerea unui salariat din vizualizare determina
   -- stergerea din info_emp_*** si reactualizarea in
   -- info_dept_***
```

```
ELSIF UPDATING ('salariu') THEN
   /* modificarea unui salariu din vizualizare determina
      modificarea salariului in info_emp_*** si reactualizarea
      in info_dept_***    */
```

```
ELSIF UPDATING ('id_dept') THEN
    /* modificarea unui cod de departament din vizualizare
       determina modificarea codului in info_emp_***
       si reactualizarea in info_dept_***  */
```