

Triggeri (declanșatori)

➤ Un trigger

- este un bloc PL/SQL asociat unui tabel, view, scheme sau unei baze de date.
- trigger-ul se executa implicit ori de câte ori are loc un anumit eveniment
- pot fi de următoarele tipuri:
 - trigger-i la nivel de aplicație: se declanșează odată un un anumit eveniment din aplicație;
 - trigger-i la nivel de bază de date: se declanșează atunci când un eveniment asupra datelor (de ex, LMD) sau un eveniment sistem (logon, shutdown) apare asupra unei scheme sau asupra bazei de date.

➤ Instrucțiunea pentru crearea trigger-ilor LMD conține următoarele informații:

- timpul declanșării trigger-ului în raport cu evenimentul:
 - pentru tabele: BEFORE, AFTER
 - pentru view-uri nemodificabile: INSTEAD OF
- evenimentul declanșator: INSERT, UPDATE, DELETE
- numele tabelului
- tipul trigger-ului – precizează de câte ori se execută corpul acestuia; trigger-ul poate fi la nivel de:
 - instrucțiune (statement): corpul triggerului se execută o singură dată pentru evenimentul declanșator. Un astfel de trigger se declanșează chiar dacă nici o linie nu este afectată.
 - linie (row): corpul triggerului se declanșează o dată pentru fiecare linie afectată de către evenimentul declanșator. Un astfel de trigger nu se execută dacă evenimentul declanșator nu afectează nici o linie.
- clauza WHEN – precizează o condiție restrictivă
- corpul trigger-ului (blocul PL/SQL)

➤ Sintaxa comenzii de creare a unui trigger LMD este:

```
CREATE [OR REPLACE] TRIGGER [schema.]nume_trigger
{BEFORE | AFTER}
[INSTEAD OF]
{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_trigger;
```

➤ Informații despre triggeri se găsesc în următoarele vizualizări ale dicționarului datelor: *USER_TRIGGERS*, *USER_TRIGGER_COL*, *ALL_TRIGGERS*, *DBA_TRIGGERS*.

➤ Modificarea unui declanșator constă din redenumirea, recompilarea, activarea sau dezactivarea acestuia și se realizează prin comenzi de forma:

```
ALTER TRIGGER nume_trigger ENABLE;
ALTER TRIGGER nume_trigger DISABLE;
ALTER TRIGGER nume_trigger COMPILE;
ALTER TRIGGER nume_trigger RENAME TO nume_nou;
```

Activarea și dezactivarea tuturor triggerilor asociați unui tabel se realizează prin comenzile:

```
ALTER TABLE nume_tabel  
DISABLE ALL TRIGGERS;  
ALTER TABLE nume_tabel  
ENABLE ALL TRIGGERS;
```

Eliminarea unui declanșator se face prin

```
DROP TRIGGER nume_trigger;
```

- Sintaxa pentru crearea unui declanșator sistem este următoarea
- ```
CREATE [OR REPLACE] TRIGGER [schema.]nume_declanșator
{BEFORE | AFTER}
{lista_evenimente_LDD | lista_evenimente_bază}
ON {DATABASE | SCHEMA}
[WHEN (condiție)]
corp_declanșator;
```

unde: *lista\_evenimente\_LDD* - CREATE, DROP, ALTER)

*lista\_evenimente\_bază* - STARTUP, SHUTDOWN, LOGON, LOGOFF, SERVERERROR, SUSPEND)

## Exerciții

1. Să se creeze un trigger care asigură ca inserarea de angajați în tabelul EMP\_PNU se poate realiza numai în zilele lucrătoare, între orele 8-18.

**Obs:** Trigger-ul nu are legătură directă cu datele => este un **trigger la nivel de instrucțiune**.

```
CREATE OR REPLACE TRIGGER b_i_emp_pnu
BEFORE INSERT ON emp_pnu
BEGIN
 IF (TO_CHAR(SYSDATE, 'dy') IN ('sat', 'sun')) OR
 (TO_CHAR(SYSDATE, 'HH24:MI')
 NOT BETWEEN '08:00' AND '18:00')
 THEN
 RAISE_APPLICATION_ERROR (-20500, 'Nu se pot introduce
 inregistrari decat in timpul orelor de lucru');
 END IF;
END;
/
```

Testați trigger-ul:

```
INSERT INTO emp_pnu (employee_id, last_name, first_name, email, hire_date,
 job_id, salary, department_id)
VALUES (300, 'Smith', 'Robert', 'rsmith', SYSDATE, 'IT_PROG', 4500, 60);
```

2. Modificați trigger-ul anterior, astfel încât să fie generate erori cu mesaje diferite pentru inserare, actualizare, actualizarea salariului, ștergere.

```
CREATE OR REPLACE TRIGGER b_i_emp_pnu
BEFORE INSERT OR UPDATE OR DELETE ON emp_pnu
BEGIN
 IF (TO_CHAR(SYSDATE, 'dy') IN ('sat', 'sun')) OR
 (TO_CHAR(SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00')
 THEN
 IF DELETING THEN
```

```

 RAISE_APPLICATION_ERROR (-20501, 'Nu se pot
 sterge inregistrari decat in timpul orelor de lucru');
 ELSIF INSERTING THEN
 RAISE_APPLICATION_ERROR (-20500, 'Nu se pot
 adauga inregistrari decat in timpul orelor de lucru');
 ELSIF UPDATING ('SALARY') THEN
 RAISE_APPLICATION_ERROR (-20502, 'Nu se poate
 actualiza campul SALARY decat in timpul orelor de
 lucru');
 ELSE
 RAISE_APPLICATION_ERROR (-20503, 'Nu se pot
 actualiza inregistrari decat in timpul orelor de
 lucru');
 END IF;
END IF;
END;
/

```

3. Să se creeze un trigger care să permită ca numai salariații având codul job-ului AD\_PRES sau AD\_VP să poată câștiga mai mult de 15000.

**Obs:** Trigger-ul se declanșează de un număr de ori = nr de înregistrări inserate sau al căror câmp salary este modificat (deci are legătură cu datele din tabel) => este un **trigger la nivel de linie**.

```

CREATE OR REPLACE TRIIGER b_i_u_emp_pnu
 BEFORE INSERT OR UPDATE OF salary ON emp_pnu
 FOR EACH ROW
 BEGIN
 IF NOT (:NEW>job_id IN ('AD_PRES', 'AD_VP'))
 AND :NEW.salary > 15000
 THEN
 RAISE_APPLICTION_ERROR (-20202, 'Angajatul nu poate
 castiga aceasta suma');
 END IF;
 END;
/

```

4. Să se implementeze cu ajutorul unui declanșator constrângerea că valorile salariilor nu pot fi reduse (trei variante). După testare, suprimați trigger-ii creați.

*Varianta 1:*

```

CREATE OR REPLACE TRIGGER verifica_salariu_pnu
 BEFORE UPDATE OF salary ON emp_pnu
 FOR EACH ROW
 WHEN (NEW.salary < OLD.salary)
 BEGIN
 RAISE_APPLICATION_ERROR (-20222, 'valoarea unui salariu nu poate fi micșorată');
 END;
/
Update emp_pnu
Set salary = salary/2;
Drop trigger verifica_salariu_pnu;

```

*Varianta 2:*

```
CREATE OR REPLACE TRIGGER verifica_salariu_pnu
 BEFORE UPDATE OF salary ON emp_pnu
 FOR EACH ROW
BEGIN
 IF (:NEW.salary < :OLD.salary) THEN
 RAISE_APPLICATION_ERROR (-20222, 'valoarea unui salariu nu poate fi micsorata');
 END IF;
END;
/
```

*Varianta 3:*

```
CREATE OR REPLACE PROCEDURE p4l6_pnu IS
 BEGIN
 RAISE_APPLICATION_ERROR (-20222, 'valoarea unui salariu nu poate fi
 micsorata');
END;
/
CREATE OR REPLACE TRIGGER verifica_salariu_pnu
 BEFORE UPDATE OF salary ON emp_pnu
 FOR EACH ROW
 WHEN (NEW.salary < OLD.salary)
 CALL p4l6_pnu;
```

5. Să se creeze un trigger care calculează comisionul unui angajat 'SA\_REP' atunci când este adăugată o linie tabelului emp\_pnu sau când este modificat salariul.

**Obs:** Dacă se dorește atribuirea de valori coloanelor utilizând NEW, trebuie creați triggeri BEFORE ROW. Dacă se încearcă scrierea unui trigger AFTER ROW, atunci se va obține o eroare la compilare.

```
CREATE OR REPLACE TRIGGER b_i_u_sal_emp_pnu
 BEFORE INSERT OR UPDATE OF salary ON emp_pnu
 FOR EACH ROW
 WHEN (NEW.job_id = 'SA_REP')
BEGIN
 IF INSERTING
 THEN :NEW.commission_pct := 0;
 ELSIF :OLD.commission_pct IS NULL
 THEN :NEW.commission_pct := 0;
 ELSE :NEW.commission_pct := :OLD.commission_pct * (:NEW.salary/:OLD.salary);
 END IF;
END;
/
```

6. Să se implementeze cu ajutorul unui declanșator constrângerea că, dacă salariul minim și cel maxim al unui job s-ar modifica, orice angajat având job-ul respectiv trebuie să aibă salariul între noile limite .

```
CREATE OR REPLACE TRIGGER verifica_sal_job_pnu
 BEFORE UPDATE OF min_salary, max_salary ON jobs_pnu
 FOR EACH ROW
DECLARE
 v_min_sal emp_pnu.salary%TYPE;
 v_max_sal emp_pnu.salary%TYPE;
```

```

 e_invalid EXCEPTION;
BEGIN
 SELECT MIN(salary), MAX(salary)
 INTO v_min_sal, v_max_sal
 FROM emp_pnu
 WHERE job_id = :NEW.job_id;
 IF (v_min_sal < :NEW.min_salary) OR
 (v_max_sal > :NEW.max_salary) THEN
 RAISE e_invalid;
 END IF;
EXCEPTION
 WHEN e_invalid THEN
 RAISE_APPLICATION_ERROR (-20567, 'Exista angajati avand salariul in afara
 domeniului permis pentru job-ul corespunzator');
END verifica_sal_job_pnu;
/

```

7. Să se creeze un trigger check\_sal\_pnu care garantează ca, ori de câte ori un angajat nou este introdus în tabelul EMPLOYEES sau atunci când este modificat salariul sau codul job-ului unui angajat, salariul se încadrează între minimul și maximul salariilor corespunzătoare job-ului respectiv. Se vor exclude angajatii AD\_PRES.

```

CREATE OR REPLACE TRIGGER check_sal_pnu
 BEFORE INSERT OR UPDATE OF salary, job_id
 ON emp_pnu
 FOR EACH ROW
 WHEN (NEW.job_id <> 'AD_PRES')
DECLARE
 v_min employees.salary %TYPE;
 v_max employees.salary %TYPE;
BEGIN
 SELECT MIN(salary), MAX(salary)
 INTO v_min, v_max
 FROM emp_pnu -- FROM copie_emp_pnu
 WHERE job_id = :NEW.job_id;
 IF :NEW.salary < v_min OR
 :NEW.salary > v_max THEN
 RAISE_APPLICATION_ERROR (-20505, 'In afara domeniului');
 END IF;
END;
/

```

Testați trigger-ul anterior:

```

UPDATE emp_pnu
SET salary = 3500
WHERE last_name= 'Stiles';

```

Ce se obține și de ce? Modificați trigger-ul astfel încât să funcționeze corect.

**Obs:** Tabelul este mutating. Pentru ca trigger-ul să funcționeze, utilizați o copie a tabelului emp\_pnu în instrucțiunea SELECT din corpul trigger-ului. (aceasta este doar una dintre solutii, se vor vedea ulterior si altele).

8. a) Se presupune că în tabelul *dept\_pnu* se păstrează (într-o coloană numită *total\_sal*) valoarea totală a salariilor angajaților în departamentul respectiv. Introduceți această coloană în tabel și actualizați conținutul.

```
ALTER TABLE dept_pnu
ADD (total_sal NUMBER(11, 2));

UPDATE dept_pnu
SET total_sal =
(SELECT SUM(salary)
FROM emp_pnu
WHERE emp_pnu.department_id = dept_pnu.department_id);
```

- b) Creați un trigger care permite reactualizarea automată a acestui câmp.

```
CREATE OR REPLACE PROCEDURE creste_total_pnu
(v_cod_dep IN dept_pnu.department_id%TYPE,
v_sal IN dept_pnu.total_sal%TYPE) AS
BEGIN
UPDATE dept_pnu
SET total_sal = NVL (total_sal, 0) + v_sal
WHERE department_id = v_cod_dep;
END creste_total_pnu;
/

CREATE OR REPLACE TRIGGER calcul_total_pnu
AFTER INSERT OR DELETE OR UPDATE OF salary ON emp_pnu
FOR EACH ROW
BEGIN
IF DELETING THEN
creste_total_pnu (:OLD.department_id, -1* :OLD.salary);
ELSIF UPDATING THEN
creste_total_pnu (:NEW.department_id, :NEW.salary - :OLD.salary);
ELSE /* inserting */
Creste_total_pnu (:NEW.department_id, :NEW.salary);
END IF;
END;
/
```

9. Să se implementeze cu ajutorul unui declanșator restricția că într-un departament pot lucra maximum 50 de angajați (comentat/rezolvat laborator)

10. Să se creeze un declanșator care:

- a) dacă este eliminat un departament, va șterge toți angajații care lucrează în departamentul respectiv;
- b) dacă se schimbă codul unui departament, va modifica această valoare pentru fiecare angajat care lucrează în departamentul respectiv.

```
CREATE OR REPLACE TRIGGER dep_cascada_pnu
BEFORE DELETE OR UPDATE OF department_id ON dept_pnu
FOR EACH ROW
BEGIN
IF DELETING THEN
DELETE FROM emp_pnu
```

```
WHERE department_id = :OLD.department_id;
END IF;
IF UPDATING AND :OLD. department_id != :NEW. department_id THEN
 UPDATE emp_pnu
 SET department_id = :NEW. department_id
 WHERE department_id = :OLD. department_id;
END IF;
END dep_cascada_pnu;
```

**Obs:** Declanșatorul anterior realizează constrângerea de integritate *UPDATE* sau *ON DELETE CASCADE*, adică ștergerea sau modificarea cheii primare a unui tabel „părinte” se va reflecta și asupra înregistrărilor corespunzătoare din tabelul „copil”.

Testați trigger-ul:

```
DELETE FROM dept_pnu
WHERE department_id = 10;
```

```
UPDATE dept_pnu
SET department_id = 12
WHERE department_id = 10;
```