SGBD-Anul 3
Laborator 5 PL/SQL

Triggeri (declanşatori)

- 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 declanşează odată un un anumit eveniment din aplicație;
    - trigger-i la nivel de bază de date: se declanşează 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 declanşării trigger-ului în raport cu evenimentul:
    - pentru tabele: BEFORE, AFTER
    - pentru view-uri nemodificabile: INSTEAD OF
  - evenimentul declanşator: 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 declanşator. Un astfel de trigger se declanşează chiar dacă nicio linie nu este afectată.
    - linie (row): corpul triggerului se declanşează o dată pentru fiecare linie afectată de către evenimentul declanşator. Un astfel de trigger nu se execută dacă evenimentul declanşator nu afectează nicio 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 declanşator constă din redenumirea, recompilarea, activarea sau dezactivarea acestuia şi 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 şi 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 declanşator se face prin
DROP TRIGGER nume_trigger;

- Sintaxa pentru crearea unui declanşator sistem este următoarea

CREATE [*OR REPLACE*] *TRIGGER* [*schema.*]*nume_declanşator*
{*BEFORE* | *AFTER*}
{*lista_evenimente_LDD* | *lista_evenimente_bază*}
ON {*DATABASE* | *SCHEMA*}
[*WHEN* (*condiție*)]
corp_declanşator;

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 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.***

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

3. Să se creeze un trigger care să permită ca numai salariații având codul job-ului AD_PRES sau AD_VP să poată câştiga mai mult de 15000.
Obs: Trigger-ul se declanşează 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***.

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

5. Să se creeze un trigger care calculează comisionul unui angajat 'SA_REP' atunci când este adăugată o linie tabelului emp sau când este modificat salariul.
Obs: Dacă se doreşte 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.

6. Să se implementeze cu ajutorul unui declanşator constrângerea că, dacă salariul minim şi cel maxim al unui job s-ar modifica, orice angajat având job-ul respectiv trebuie să aibă salariul între noile limite (*RAISE* eroare; *EXCEPTION*).

7. Să se creeze un trigger check_sal 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 şi maximul salariior corespunzătoare job-ului respectiv. Se vor exclude angajatii AD_PRES.

Testați trigger-ul anterior:
UPDATE emp
SET salary = 3500
WHERE last_name= 'Stiles';
Ce se obține şi 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 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* se păstrează (într-o coloană numită *total_sal*) valoarea totală a salariilor angajaților în departamentul respectiv. Introduceți această coloană în tabel şi actualizați conținutul.

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

9. Să se creeze două tabele noi new_emp şi new_dept pe baza tabelelor employees şi departments. Să se creeze un view view_emp, care selectează codul, numele, salariul, codul departamentului, email-ul, codul job-ului, numele departamentului şi codul locației pentru fiecare angajat.
Să se creeze un ***trigger de tip INSTEAD OF*** care, în locul inserării unei linii direct în view, adaugă înregistrări corespunzătoare în tabelele new_emp şi new_dept. Similar, atunci când o linie este modificată sau ştearsă prin intermediul vizualizării, liniile corespunzătoare din tabelele new_emp şi new_dept sunt afectate.

10. Să se implementeze cu ajutorul unui declanşator restricția că într-un departament pot lucra maximum 50 de angajați.

Obs: Declanşatorul *TrLimitaDep* consultă chiar tabelul (*emp*) la care este asociat declanşatorul (*mutating*).
Tabelul *emp* este *mutating* doar pentru un declanşator la nivel de linie.
O solutie pentru acestă problemă este crearea a doi declanşatori, unul la nivel de linie şi altul la nivel de instrucțiune :
- în declanşatorul la nivel de linie se înregistrează valoarea lui *:NEW.department_id*, dar nu va fi interogat tabelul *emp*.
- interogarea va fi făcută în declanşatorul la nivel de instrucțiune şi va folosi valoarea înregistrată în declanşatorul la nivel de linie.

O modalitate pentru a înregistra valoarea lui :*NEW.department_id* este ***utilizarea unui tablou indexat în interiorul unui pachet***.

```
CREATE OR REPLACE PACKAGE PdepDate AS
  TYPE t_cod IS TABLE OF dept.department_id%TYPE
            INDEX BY BINARY_INTEGER;
  v_cod_dep   t_cod;
  v_NrIntrari  BINARY_INTEGER := 0;
END PdepDate;
```

Obs: Această soluție funcționează pentru departamentele nou introduse.
Pentru testare:
- introduceți un nou departament; introduceti mai mult de 50 de angajați în departamentul inserat anterior (eventual, cu o comandă de tip INSERT INTO ... (SELECT ...));

11. Să se creeze un declanşator care:

a) dacă este eliminat un departament, va şterge 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.

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

Obs : Se presupune că asupra tabelului *emp* există o constrângere de integritate:
FOREIGN KEY (department_id) REFERENCES dept(department_id)

În acest caz sistemul *Oracle* va afişa un mesaj de eroare prin care se precizează că tabelul *dept* este *mutating*, iar constrângerea definită mai sus nu poate fi verificată.
ORA-04091: *table MASTER.DEPT is mutating*, *trigger/function may not see it*

12. Să se creeze un ***declanşator la nivelul bazei de date*** prin care să nu se permită ştergerea informațiilor din tabelul emp de către utilizatorul curent. Dezactivați, iar apoi activați trigger-ul creat. Testați, iar apoi suprimați acest trigger.

13. Să se creeze un tabel care conține următoarele câmpuri: user_id, nume_bd, eveniment, nume_obj, data. Să se creeze un **trigger sistem** (la nivel de schemă) care să introducă date în acest tabel după ce utilizatorul a folosit o comandă LDD. Testați, iar apoi suprimați trigger-ul.