Triggerii în PL/SQL

Triggerii reprezintă obiecte PL/SQL speciale, atașate la evenimente ce au loc în baza de date. Ei sunt, în esență, proceduri automate care se execută ca răspuns la operații asupra datelor sau la anumite evenimente ale sistemului. Triggerii sunt folosiți pentru a implementa reguli de integritate, auditare, validare, logare, replicare sau operații suplimentare ce trebuie realizate implicit la modificarea datelor.

Prin natura lor, triggerii nu sunt apelați explicit de programator — ei se declanșează automat atunci când evenimentul specificat apare în schema bazei de date.

Scopul triggerilor

Triggerii pot fi utilizați în situații precum:

- validarea suplimentară a datelor înainte de inserare, actualizare sau ștergere.
- auditarea operațiilor prin păstrarea unui istoric al modificărilor.
- menținerea integrității referențiale, în special atunci când constrângerile declarative nu sunt suficiente.
- calcularea automată a anumitor valori (loguri, timestamp-uri, valori derivate).
- limitarea sau blocarea anumitor operații, prevenind acțiuni nepermise.
- actualizarea automată a altor tabele la modificarea datelor sursă.
- înlocuirea comportamentului default al operațiilor asupra vizualizărilor prin triggerii INSTEAD OF.

Tipurile de triggeri

Triggerii pot fi clasificați în funcție de mai multe criterii:

1. Triggeri LMD (DML) - Se declanșează automat la executarea instrucțiunilor: INSERT, UPDATE, DELETE. Aceștia pot fi definiți:

a) la nivel de comandă (statement-level)

b) la nivel de linie (row-level)

2. Triggeri INSTEAD OF - Se folosesc pentru vizualizări (VIEW) care nu pot fi modificate direct.

Înlocuiesc operația originală prin comportamentul definit.

3. Triggeri de sistem (DDL și evenimente) - Se declanșează la evenimente precum: logon/logoff, shutdown/startup, before/after create table, drop, alter, înainte/după compilarea unui obiect

Triggerii DML la nivel de comandă și la nivel de linie

Triggeri la nivel de comandă

Se execută o singură dată pentru întreaga instrucțiune SQL, indiferent de câte rânduri sunt afectate. Utili pentru logare, contabilizarea operațiilor și validări globale.

Triggeri la nivel de linie

Se execută pentru fiecare rând afectat de comanda SQL. Utili pentru validarea individuală a fiecărui rând, copierea valorilor într-un tabel de audit sau actualizări coerente ale altor entități.

Sintaxa generală a triggerilor

```
--Trigger BEFORE/AFTER la nivel de linie
CREATE OR REPLACE TRIGGER trg_nume
BEFORE INSERT OR UPDATE ON angajati
```

```
FOR EACH ROW
BEGIN
  -- logica
END;

--Trigger la nivel de comandă
CREATE OR REPLACE TRIGGER trg_nume
AFTER DELETE ON produse
BEGIN
  -- logica
END;
```

Folosirea clauzelor INSERTING, UPDATING, DELETING

Acestea permit identificarea operației care a declanșat triggerul:

```
IF INSERTING THEN
  ...
ELSIF UPDATING THEN
  ...
ELSIF DELETING THEN
  ...
END IF;
```

:NEW – valorile noi (pentru INSERT sau UPDATE)

:OLD – valorile vechi (pentru DELETE sau UPDATE)

Exemplu complet de trigger la nivel de linie:

```
CREATE OR REPLACE TRIGGER trg_audit_angajati
AFTER UPDATE ON angajati
FOR EACH ROW
BEGIN
  INSERT INTO audit_angajati(id_angajat, salariu_vechi, salariu_nou, data_op)
  VALUES (:OLD.id_angajat, :OLD.salariu, :NEW.salariu, SYSDATE);
END;
```

Triggerii INSTEAD OF

Utilizați doar pentru vizualizări complexe (JOIN, agregări, subinterogări etc.) care nu permit actualizări implicite.

```
CREATE OR REPLACE TRIGGER trg_update_view
INSTEAD OF UPDATE ON v_angajati_view
FOR EACH ROW
BEGIN
  UPDATE angajati
  SET salariu = :NEW.salariu
  WHERE id = :OLD.id;
END;
```

Triggerii de sistem (DDL și evenimente)

Se declanșează la evenimente administrative.

Exemplu: logarea operațiilor DDL:

```
CREATE OR REPLACE TRIGGER trg_ddl_log
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
  INSERT INTO log_ddl(user_name, data_op, tip_op)
  VALUES (USER, SYSDATE, ORA_SYSEVENT);
END;
```

Exemplu: logare la conectare:

```
CREATE OR REPLACE TRIGGER trg_logon
AFTER LOGON ON DATABASE
BEGIN
  INSERT INTO log_sesiuni(user_name, data_logon)
  VALUES (USER, SYSDATE);
END;
```

Ordinea de execuție a triggerilor

Când mai mulți triggeri sunt definiți pe aceeași operație:

I. Triggerii BEFORE STATEMENT
II. Triggerii BEFORE ROW
III. Operația SQL propriu-zisă
IV. Triggerii AFTER ROW
V. Triggerii AFTER STATEMENT

Oracle nu garantează ordinea între triggeri de același tip; ordinea alfabetică nu este obligatorie. Dacă ordinea este esențială, logica trebuie combinată în același trigger.

Recursivitatea triggerilor

Oracle permite recursivitatea indirectă, dar triggerii trebuie folosiți atent: actualizarea unui tabel în interiorul unui trigger definit pe acel tabel poate duce la loop infinit. Pentru a preveni recursivitatea necontrolată, se folosesc: variabile de pachet, condiții de control, verificări ale coloanelor afectate.

Oracle previne în mod automat unele mecanisme recursive, dar nu pe toate.

Erorile specifice triggerilor

Triggerii pot genera erori diverse:

1. ORA-04091: mutating table - Apare când un trigger încearcă să citească sau să modifice același tabel pe care îl afectează în acel moment.

Problema apare în triggerii row-level și este caracteristică UPDATE/INSERT/DELETE. De exemplu: **SELECT ... FROM angajati** într-un trigger definit pe angajati.

Soluții:

- utilizarea triggerilor statement-level,
- memorarea datelor în tabele temporare,
- utilizarea pachetelor cu variabile globale,
- utilizarea unui trigger AFTER STATEMENT pentru procesare ulterioară.

2. ORA-04084 / ORA-04082: modificări nepermise asupra :NEW sau :OLD - încercarea de a seta :OLD în AFTER INSERT etc.

3. ORA-04098: Eroare la compilarea triggerului — trigger invalid și nerecompilat.

Exerciții

a) Să se creeze un nou tabel STUDENTS care reține informații despre studenții unei facultăți. Vor fi prezente coloanele: last_name, first_name, adress, country (foreign key catre tabelul COUNTRIES), grade și status (un VARCHAR).
b) Să se introducă un mecanism care calculează automat statusul fiecărui student în funcție de media sa. Dacă grade va fi 1, acesta va fi „excluded”, între 2 și 5 va fi „failed”, iar 6 sau mai sus va fi „passed”. Statusul va fi actualizat la fiecare inserție sau actualizare

c) Să se introducă o validare pentru studenți care să respingă orice actualizare care introduce caractere interzise în numele sau prenumele lor (orice caracter care nu este literă mare sau mică este invalid).
d) Pentru fiecare dată când se șterg linii din tabelul STUDENTS să se afișeze pe ecran câte intrări sunt la începutul și la finalul acțiunii.

a) Create a new „STUDENTS” table that stores info about students from a university. Add columns for: last_name, first_name, adress, country (foreign key to table COUNTRIES), grade and status (a VARCHAR).
b) Introduce a mechanism that calculates a student’s status based on his grade. If the grade is 1, the status should be „excluded”, between 2 and 5 it should be „failed” and 6 or higher should be „passed”. Modify the status after every update or insert.
c) Introduce a validation for students that nullifies any update that introduces invalid characters into their first or last names (any character that isn’t a lowercase or uppercase letter is invalid).
d) For every time that there are lines deleted in the STUDENTS table, output the number of entries in the table before or after the action.