UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Proiect nr. 154/323 cod SMIS – 4428 cofinanțat de prin Fondul European de Dezvoltare Regională “Investiții pentru viitorul dumneavoastră”.

Programul Operaţional Sectorial Creşterea Competitivităţii Economice - POS CCE

UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Baze de date

26. Triggeri

Introducere

- Un trigger este un bloc PL/SQL stocat pe server, care se execută la apariția unui eveniment care modifică starea anumitor obiecte ale bazei de date. Termenul corespondent în literatura de specialitate românească este *declanşator*, dar este rar folosit şi de aceea, în continuare se va folosi termenul din limba engleză.
- Tipurile de evenimente care pot determina execuția unui trigger sunt:
- comenzi *INSERT, UPDATE, DELETE* pe o tabelă;
- comenzi *INSERT, UPDATE, DELETE* pe un view (cu opțiunea INSTEAD OF);
- comenzi *CREATE, ALTER, DROP* la nivel de schemă sau bază de date;
- comenzi *SHUTDOWN, LOGON, LOGOFF* la nivel de schemă sau bază de date.

Utilitatea şi avantajele triggerilor

În general, triggerii se folosesc pentru:

- gestionarea restricțiilor complexe de integritate;
- monitorizarea tranzacțiilor;
- efectuarea de replicări de tabele situate în diferite noduri ale unei baze de date distribuite;
- păstrarea semnăturii userilor care au efectuat operații pe baza de date;
- prelucrarea de informații statistice în legătură cu accesul tabelelor;
- jurnalizarea transparentă a evenimentelor.

Printre avantajele utilizării triggerilor, se pot menționa:

- declanşarea automată, la apariția evenimentului monitorizat;
- lansarea în execuție a unor proceduri stocate specifice;
- posibilitatea modificării în cascadă a mai multor obiecte corelate în baza de date;
- transparența față de utilizator.

Sintaxa de creare a unui trigger

- Sintaxa unui trigger este:

CREATE [OR REPLACE] TRIGGER trigger_name
[BEFORE| AFTER| INSTEAD OF]
[INSERT| [OR] | UPDATE [OF column,…]| [OR] | DELETE]
ON schema_name
[FOR EACH ROW]
[WHEN condition]
DECLARE
trigger_variables
BEGIN
trigger_body
END;

Cuvinte cheie

unde:

- ***trigger _name*** – este numele triggerului;
- ***schema_name*** – defineşte userul şi tabela/view, pe care se monitorizează evenimentul;
- ***column*** – defineşte coloanele tabelei/view-lui pe care se monitorizează evenimentul;
- ***condition*** – reprezintă o condiție pentru executarea triggerului, fiind admise corelări dar nu şi interogări;
- ***trigger_variables*** – reprezintă secțiunea de declarare de variabile locale ale triggerului;
- ***trigger_body*** – reprezintă corpul triggerului;
- ***CREATE*** – comanda de creare;
- ***REPLACE*** - recreazã triggerul, dacă acesta există deja;
- ***BEFORE| AFTER*** – specifică momentul executării triggerului: înainte sau după apariția evenimentului;
- ***INSTEAD OF*** – specifică că este permisă o operație de inserare, ştergere, modificare pe view-uri, la care nu este permisă operația în mod firesc;
- ***INSERT | UPDATE | DELETE*** – specifică evenimentul pe care se declaşează triggerul;
- ***FOR EACH ROW*** – specifică că execuția triggerului se face pentru fiecare linie afectată, cu respectarea condiției din clauza *WHEN*.

Triggeri pe o comandă şi pe o linie

Există două tipuri de triggeri:

- triggeri pe o comandă – sunt executați o singură dată pentru evenimentul declanşator. De exemplu, dacă este executată o comandă *INSERT* de mai multe linii, triggerul este executat o singură dată. În acest caz, nu este limitare la numărul de linii afectate de eveniment.
- triggeri pe o linie – sunt executați ori de câte ori o linie a unei tabele este afectată de evenimentul declanşator. De exemplu, dacă este executată o comanda *UPDATE* care actualizează k linii, triggerul este executat de k ori.

Triggeri pe evenimente

- Triggerul poate fi executat înainte ca evenimentul să aiba loc (opțiunea *BEFORE*), sau după ce evenimentul s-a consumat(opțiunea *AFTER*).

  În general, triggerii de tip BEFORE sunt folosiți pentru :

- a salva valorile coloanelor înaintea executării unei comenzi *UPDATE*;
- a decide dacă acțiunea triggerului trebuie sau nu executată(acest lucru poate îmbunătăți performanțele serverului prin eliminarea procesărilor inutile).

  Triggerii de tip *AFTER* sunt, în general, folosiți atunci când:

- se doreşte ca executarea triggerului să se facă după ce comanda s-a efectuat cu succes;
- nu au apărut erori de procesare, care ar impune o comandă *ROLLBACK* pentru tranzacțiile parțiale deja efectuate;
- trebuie alterate şi alte date corelate cu cele deja afectate.

Triggeri de tip BEFORE

- Aceşti triggeri se declanşează la apariția unui eveniment, dar înainte ca evenimentul să se termine.
- Exemplu de trigger de tip *BEFORE* care afişează un mesaj ori de câte ori se face o inserare de date în tabelul angajați:

```
CREATE OR REPLACE TRIGGER inserare
BEFORE INSERT ON angajati
BEGIN
        dbms_output.put_line ('S-a facut o inserare noua in tabelul angajati!');
END;
```

- Să verificăm cum lucrează, facând o inserare în tabel:

```
SQL> INSERT INTO angajati(id_ang,nume,salariu)
        VALUES (999,'PREDA MIHAI', 1500);
SQL> S-a facut o inserare nouă în tabelul angajati!
        1 row created.
```

Triggeri de tip AFTER

- Sunt triggeri care se declanşează după ce evenimentul declanşator se termină. În exemplul următor, se creează un trigger care afişează un mesaj, ori de câte ori s-a făcut o modificare de date în tabela angajați:

```
CREATE OR REPLACE TRIGGER modificare
AFTER UPDATE ON angajati
BEGIN
        dbms_output.put_line('S-a făcut o modificare de date în
                              tabelul angajati!');
END;
```

- Să verificăm cum lucrează triggerul, făcând o modificare de comision în tabelul angajați:

```
SQL> UPDATE angajati SET comision=100 WHERE id_ang=7369;
SQL> S-a făcut o modificare de date în tabelul angajati!
        1 row updated.
```

Predicate condiționale

- În cazul în care se execută mai multe comenzi DML, se pot folosi predicate condiționale în corpul triggerului.
- Predicate condiționale sunt:
- ***INSERTING*** – returnează TRUE, dacă triggerul se declanşează pe o comandă INSERT;
- ***UPDATING*** – returnează TRUE, dacă triggerul se declanşează pe o comandă UPDATE;
- ***UPDATING (‘column_name’)*** – returnează TRUE, dacă triggerul se declanşează pe o comandă UPDATE pe o coloană specificată;
- ***DELETING*** – returnează TRUE, dacă triggerul se declanşează pe o comandă DELETE.

Exemplu:

```
CREATE OR REPLACE TRIGGER monitor AFTER INSERT OR DELETE OR UPDATE OF salariu,comision
    ON angajati FOR EACH ROW
    DECLARE
    salariu angajati.salariu%TYPE;
    comision angajati.comision%TYPE;
    BEGIN
    IF INSERTING THEN
        INSERT INTO mesaje VALUES ('Inserare in tabela angajati',to_char(sysdate,'dd-mm- yyyy
                                                hh:mi:ss'));
    ELSIF DELETING THEN
        INSERT INTO mesaje VALUES ('Stergere in tabela angajati',to_char(sysdate,'dd-mm- yyyy
                                                hh:mi:ss'));
    ELSIF UPDATING ('salariu') THEN INSERT INTO mesaje VALUES ('Salariu modificat in tabela
    angajati',:old.salariu ||'/' ||:new.salariu||'/'||to_char(sysdate,'dd-mm-yyyy hh:mi:ss'));
    END IF;
    END;
```

Triggere cu opțiunea INSTEAD OF

- Aceste triggere se definesc numai pe view-uri, nu şi pe tabele. Unele view-uri nu pot fi modificate prin comenzi DML, dar folosind un trigger cu opțiunea *INSTEAD OF*(in loc de) acest lucru este realizabil. View-urile care nu pot fi modificate prin comenzile *UPDATE, INSERT* sau *DELETE* sunt cele create printr-o interogare care conține în construcție:
- un operator *SET* sau *DISTINCT*
- o funcție de agregare, sau o funcție analitică
- clauzele *GROUP BY, ORDER BY, CONNECT BY* sau *START WITH*
- o expresie tip colecție într-o listă *SELECT*
- o subcerere într-o listă *SELECT*
- unele metode de *JOIN*

Exemplu:

```
CREATE OR REPLACE TRIGGER manager INSTEAD OF INSERT ON sefi
REFERENCING NEW AS n
FOR EACH ROW
DECLARE
nr number;
BEGIN
SELECT COUNT(*) INTO nr FROM angajati  WHERE id_ang = :n.id_ang;
      IF nr = 0  THEN
        INSERT INTO angajati (id_ang,nume,id_dep)
                VALUES (:n.id_ang, :n.nume, :n.id_dep);
        UPDATE angajati  SET salariu=:n.salariu, comision=:n.comision,
        data_ang=:n.data_ang,  id_sef=:n.id_ang WHERE id_ang = :n.id_ang;
END IF;
END;
```

Trigger created.