SQL dans un langage de programmation

Ressource R3.07

Cours 3 : Les triggers (déclencheurs)

2023-2024

Triggers

Définition

- Un trigger (déclencheur) est un programme qui se déclenche automatiquement suite à un événement qui modifie l'instance d'une table : `INSERT`, `DELETE` ou `UPDATE`.
- Les triggers s'exécutent de manière cachée. Ils ne sont pas appelés explicitement, contrairement aux procédures stockées.

Utilité

- Gestion de contraintes d'intégrité complexes.
- Gestion de la redondance d'information.
- Gestion automatique de certains événements sur la base de données (*auditing*).

Triggers

Principe

A chaque trigger, on associe :

- un (des) événement(s) sur **une table** : `INSERT`, `DELETE` ou `UPDATE`.
- une condition : expression booléenne qui s'évalue à vrai ou faux.
- une action : un ensemble d'instructions SQL exécutées avec le langage procédural PL/SQL.

Déclenchement

- Quand : événements LMD sur une table.
- Comment : sur chaque tuple manipulé par l'événement ou sur chaque événement.
- Quoi : action exécutée **avant** ou **après** l'événement déclencheur.

Triggers

Trigger en PostgreSQL

Ecriture en 2 temps :

1. Ecriture d'une fonction trigger (en langage PLpgSQL)
2. Écriture du trigger appelant la fonction.
   On définit :
   - La table à laquelle le trigger est lié.
   - Les instructions du LMD qui déclenchent le trigger.
   - Le moment où le trigger se déclenche par rapport à l'instruction LMD (avant ou après).
   - Si le trigger se déclenche une seule fois pour toute l'instruction (trigger instruction), ou bien une fois pour chaque ligne modifiée/insérée/supprimée (trigger ligne).
   - Eventuellement une condition supplémentaire de déclenchement (clause `WHEN`) pour chaque ligne vérifiant la condition.

Triggers

Création de la fonction trigger

```
CREATE FUNCTION fct_trig() RETURNS TRIGGER AS
$$
DECLARE
-- declarations;
BEGIN
-- statements;
END;
$$ LANGUAGE plpgsql;
```

Triggers

Création du trigger

```
CREATE [OR REPLACE] TRIGGER nom_trig
{BEFORE | AFTER | INSTEAD OF} {événement [OR ...]}
ON nom_table
[FOR [EACH] {ROW | STATEMENT}]
[WHEN (condition)]
EXECUTE {FUNCTION | PROCEDURE} nom_fonction (arguments);
```

où événement fait partie de :

```
INSERT
UPDATE [OF nom_colonne [, ... ]]
DELETE
```

La commande `INSTEAD OF` s'utilise uniquement avec les vues.

Triggers

AFTER ou BEFORE

- Si le trigger doit déterminer si l'instruction LMD est autorisée, utiliser `BEFORE`.
- Si le trigger doit générer la valeur d'une colonne pour pouvoir ensuite la mettre dans la table, utiliser `BEFORE`.
- Si on a besoin que l'instruction LMD soit terminée pour exécuter la fonction, utiliser `AFTER`.

Triggers

Trigger de niveau ligne ou instruction

- Un trigger instruction (option `for each statement`) se déclenche une fois, suite à une instruction LMD.
- Un trigger ligne (option `for each row`) se déclenche pour chaque ligne modifiée par l'instruction LMD.

Exemple

Une instruction `update` sur une table `ma_table` modifie 10 lignes.

- Un trigger instruction se déclenchera une seule fois
- Un trigger ligne se déclenchera 10 fois, une fois par ligne modifiée

(déclenchement avant ou après la modification, en fonction de `after/before`).

Triggers

Variables globales

Dans un trigger de niveau ligne, on peut faire référence à la ligne courante, celle pour laquelle le trigger s'exécute.
Deux variables de type `RECORD` :

- `OLD` : contient la valeur de la ligne **avant** l'instruction LMD.
- `NEW` : contient la valeur de la ligne **après** l'instruction LMD.

	OLD	NEW
INSERT	NULL	ligne insérée
DELETE	ligne supprimée	NULL
UPDATE	ligne avant modification	ligne après modification

Triggers

Variables globales

- `TG_NAME` : contient le nom du trigger qui est en train d'être déclenché.
- `TG_WHEN` : type texte, contient `BEFORE` ou `AFTER` ou `INSTEAD OF`.
- `TG_LEVEL` : type texte, contient `ROW` ou `STATEMENT`.
- `TG_OP` : type texte, contient `INSERT` ou `UPDATE` ou `DELETE`.
- `TG_TABLE NAME` : contient le nom de la table qui a déclenché le trigger.
- `TG_NARGS` : nombre d'arguments transmis à la fonction liée au trigger.

Triggers

Modification de trigger

- Renommer un trigger :
  `ALTER TRIGGER ancien_nom_trig on nom_table rename to nouveau_nom_trig ;`
- Mettre hors service sans le détruire :
  `ALTER TABLE nom_table DISABLE TRIGGER {nom_trig | ALL | USER};`
- Remettre en service :
  `ALTER TABLE nom_table ENABLE TRIGGER {nom_trig | ALL | USER};`
- Détruire un trigger :
  `DROP TRIGGER [IF EXISTS] nom_trig on nom_table;`

Triggers

Exemple : mise à jour des notes

```
CREATE TABLE tbl_note (num_etudiant int, num_controle
int, note numeric);

SELECT * FROM tbl_note;

num_etudiant | num_controle | note
-------------+--------------+-------
       12345 |            1 | 13.50
(1 row)
```

Triggers

Exemple : mise à jour des notes

```
CREATE FUNCTION fct_maj_note() returns TRIGGER as
$$
BEGIN
IF (NEW.note < OLD.note) THEN RETURN NULL;
END IF;
RETURN NEW;
END;
$$ language plpgsql ;

CREATE TRIGGER trig_maj_note
BEFORE UPDATE on tbl_note
FOR EACH ROW
EXECUTE PROCEDURE fct_maj_note();
```

Triggers

Exemple : mise à jour des notes

```
UPDATE tbl_note set note=note+2;
UPDATE 1

SELECT * FROM tbl_note;

num_etudiant | num_controle | note
-------------+--------------+-------
       12345 |            1 | 15.50
(1 row)
```

Triggers

Exemple : mise à jour des notes

```
UPDATE tbl_note set note=note-2;
UPDATE 0

SELECT * FROM tbl_note;
-------------+-------------+-------
num_etudiant | num_controle | note
-------------+-------------+-------
       12345 |           1 | 15.50
(1 row)
```

Triggers

Exemple : compléter une ligne de facture

On considère le schéma simplifié ci-dessous :

```
1  CREATE TABLE Article(
2      id_article    int primary key,
3     nom           varchar,
4      prix          money
5  );
6
7  CREATE TABLE Facture(
8     id_article     int  references   Article ,
9    Quantite       int,
10    Prix_total     money
11   );
```

Et l'instance suivante :

```
 id_article |          nom          | prix
------------+-----------------------+-------
       1231 | Petit pain fromage    | €0,35
       1232 | Petit pain graines    | €0,35
       1233 | Petit pain céréales   | €0,35
       1234 | XXL Papier toilette   | €4,99
       1235 | Batonnets quate       | €0,32
       1236 | Mouchoirs blancs      | €0,79
(6 rows)
```

Triggers

Exemple : compléter une ligne de facture

```
1  CREATE or replace FUNCTION Exemple1()
2    returns TRIGGER as
3  $$
4    DECLARE
5       aprix  money;
6    BEGIN
7     IF TG_OP='UPDATE' or TG_OP='INSERT' THEN
8       SELECT prix into aprix FROM Article
9         WHERE  id_article=NEW.id_article;
10      NEW.Prix_total =aprix*NEW.Quantite;
11    END IF;
12    IF TG_OP='DELETE' THEN
13      SELECT prix into aprix FROM Article
14        WHERE  id_article=OLD.id_article;
15    END IF;
16    IF (FOUND) THEN
17       IF TG_OP='UPDATE' or TG_OP='DELETE' THEN
18        raise notice ' OLD Article % : % * % = % ',
19         OLD.id_article,aPrix,OLD.quantite,OLD.prix_total ;
20       END IF;
21       IF TG_OP='UPDATE' or TG_OP='INSERT' THEN
22        raise notice ' NEW Article % : % * % = % ',
23         NEW.id_article,aPrix,NEW.quantite,NEW.prix_total;
24       END IF;
25       IF TG_OP='UPDATE' or TG_OP='INSERT' THEN
26          RETURN NEW;
27       ELSE
28          RETURN OLD;
29       END IF;
30     END IF;
31     RETURN NULL;
32   END;
33 $$ language plpgsql ;
34
35 CREATE TRIGGER exemple1
36      BEFORE
37      UPDATE or INSERT or DELETE ON facture
38         FOR EACH ROW
39         EXECUTE PROCEDURE exemple1();
```

Triggers

Exemple : compléter une ligne de facture

```
INSERT INTO facture values
         (1231,1),(1232,2),(1233,3),(1234,1),(1235,1),(1236,3);
NOTICE:   NEW Article 1231 : €0,35 * 1 = €0,35
NOTICE:   NEW Article 1232 : €0,35 * 2 = €0,70
NOTICE:   NEW Article 1233 : €0,35 * 3 = €1,05
NOTICE:   NEW Article 1234 : €4,99 * 1 = €4,99
NOTICE:   NEW Article 1235 : €0,32 * 1 = €0,32
NOTICE:   NEW Article 1236 : €0,79 * 3 = €2,37
INSERT 0 6

SELECT a.nom Article,a.prix "Prix U",
       f.quantite "Quantite",f.prix_total "Total Ligne"
    FROM article a, facture f
    WHERE a.id_article=f.id_article;
       article       | Prix U | Quantite | Total Ligne
---------------------+--------+----------+-------------
 Petit pain fromage  |  €0,35 |        1 |       €0,35
 Petit pain graines  |  €0,35 |        2 |       €0,70
 Petit pain céréales |  €0,35 |        3 |       €1,05
 XXL Papier toilette |  €4,99 |        1 |       €4,99
 Batonnets quate     |  €0,32 |        1 |       €0,32
 Mouchoirs blancs    |  €0,79 |        3 |       €2,37
(6 rows)

SELECT sum(prix_total) "A payer" FROM facture;
 A payer
---------
   €9,78
(1 row)
```

Triggers

Exemple : compléter une ligne de facture

```
UPDATE Facture SET Quantite=4
   WHERE id_article=1231;
NOTICE:   OLD Article 1231 : €0,35 * 1 = €0,35
NOTICE:   NEW Article 1231 : €0,35 * 4 = €1,40
UPDATE 1

DELETE FROM Facture
    WHERE id_article=1234;
NOTICE:   OLD Article 1234 : €4,99 * 1 = €4,99
DELETE 1

UPDATE Facture SET prix_total='1'
    WHERE id_article=1231;
NOTICE:   OLD Article 1231 : €0,35 * 4 = €1,40
NOTICE:   NEW Article 1231 : €0,35 * 4 = €1,40
UPDATE 1

SELECT a.nom Article,a.prix "Prix U",
       f.quantite "Quantite",f.prix_total "Total Ligne"
    FROM article a, facture f
    WHERE a.id_article=f.id_article;
       article       | Prix U | Quantite | Total Ligne
---------------------+--------+----------+-------------
 Petit pain graines  |  €0,35 |        2 |       €0,70
 Petit pain céréales |  €0,35 |        3 |       €1,05
 Batonnets quate     |  €0,32 |        1 |       €0,32
 Mouchoirs blancs    |  €0,79 |        3 |       €2,37
 Petit pain fromage  |  €0,35 |        4 |       €1,40
(5 rows)
```

Triggers

Exemple : ne pas autoriser la modification d'un article

```
1  CREATE or replace FUNCTION Exemple2()
2    returns TRIGGER as
3  $$
4    BEGIN
5      IF TG_OP='UPDATE' or TG_OP='DELETE' THEN
6       raise notice ' OLD Article % : % % ',
7          OLD.id_article,OLD.nom,OLD.Prix ;
8      END IF;
9      IF TG_OP='UPDATE' or TG_OP='INSERT' THEN
10      raise notice ' NEW Article % : % % ',
11         NEW.id_article,NEW.nom,NEW.Prix ;
12     END IF;
13      RETURN NULL;
14   END;
15 $$ language plpgsql ;
16
17 CREATE TRIGGER exemple2
18      BEFORE
19      UPDATE or INSERT or DELETE ON Article
20         FOR EACH ROW
21         EXECUTE PROCEDURE exemple2();
```

Triggers

Exemple : ne pas autoriser la modification d'un article

```
INSERT INTO article VALUES(44444,'Vin','35');
NOTICE:   NEW Article 44444 : Vin €35,00
INSERT 0 0

DELETE FROM Article WHERE id_article=44443;
NOTICE:   OLD Article 44443 : Voiture €9 980,00
DELETE 0

UPDATE Article SET prix='2000' WHERE id_article=44443;
NOTICE:   OLD Article 44443 : Voiture €9 980,00
NOTICE:   NEW Article 44443 : Voiture €2 000,00
UPDATE 0

select * from article;
 id_article |      nom      |   prix
------------+---------------+-----------
      44441 | Cote de Boeuf |    €45,50
      44442 | Foie Gras     |   €120,15
      44443 | Voiture       | €9 980,00
(3 rows)
```