Cours de bases de données, , http://sql.bdpedia.fr

Triggers

Philippe Rigaux

le cnam

Les *triggers*

Un **trigger** est une procédure stockée qui se déclenche automatiquement sur certains événements.

Les possibilités offertes par les triggers sont très intéressantes. Citons :

- la gestion des redondances ; l'enregistrement automatique de certains événements (auditing) ;
- la spécification de contraintes complexes liées à l'évolution des données (exemple : le prix d'une séance ne peut qu'augmenter) ;
- toute règle liée à l'environnement d'exécution (restrictions sur les horaires, les utilisateurs, etc.).

Inconvénient (fort) : les *triggers* s'exécutent de manière cachée, et peuvent mener à des cycles sans fin.

Un exemple simple

Voici un trigger qui maintient la capacité d'un cinéma à chaque mise à jour sur la table `Salle`.

```
CREATE TRIGGER CumulCapacite
AFTER UPDATE ON Salle
FOR EACH ROW
WHEN (new.capacite != old.capacite)
BEGIN
  UPDATE Cinema
  SET capacite = capacite - :old.capacite  + :new.capacite
  WHERE  nom = :new.nomCinema;
END;
```

Pour garantir la validité du cumul, il faudrait créer des triggers sur les événements UPDATE et INSERT.

Les composants d'un trigger

Le modèle d'exécution des triggers est basé sur la séquence événement-Condition-Action (ECA) que l'on peut décrire ainsi :

- un trigger est déclenché par un évènement, spécifié par le programmeur, qui est en général une insertion, destruction ou modification sur une table ;
- la première action d'un trigger est de tester une condition : si cette condition ne s'évalue pas à TRUE, l'exécution s'arrête ;
- enfin l'action proprement dite peut consister en toute ensemble d'opérations sur la base de données, effectuée si nécessaire à l'aide du langage procédural supporté par le SGBD (PL/SQL).

Un trigger peut manipuler simultanément les valeurs ancienne et nouvelle de la donnée modifiée, ce qui permet de faire des tests sur l'évolution de la base.

Syntaxe générale

```
CREATE [OR REPLACE] TRIGGER <nomTrigger>
  {BEFORE | AFTER}
  {DELETE | INSERT | UPDATE [of column, [, column] ...]}
     [ OR {DELETE | INSERT | UPDATE [of column, [, column] ...]}] ...
  ON <nomTable> [FOR EACH ROW]
[WHEN <condition]
   <blocPLSQL>
```

Evénement, peut être BEFORE ou AFTER, suivi de DELETE, UPDATE ou INSERT séparés par des OR.

Condition, FOR EACH ROW est optionnel. En son absence le trigger est déclenché une fois pour toute requête modifiant la table, et ce sans condition.

Action est une procédure PL/SQL. Elle peut contenir des ordres SQL mais pas de mise à jour de la table courante.

Attention avec les triggers !

On peut des triggers qui **ralentissent** tout. Exemple

```
CREATE TRIGGER CumulCapaciteGlobal
AFTER UPDATE OR INSERT OR DELETE ON Salle
BEGIN
  UPDATE Cinema C
  SET capacite = (SELECT SUM (capacite)
                  FROM   Salle S
                  WHERE  C.nom = S.nomCinema);
END;
```

Attention avec les triggers !

On peut des triggers qui **bloquent** tout. Exemple d'un trigger à la fois idiot et assassin.

```
CREATE TRIGGER MAJSalle
AFTER UPDATE OR INSERT OR DELETE ON Cinema
BEGIN
  UPDATE Salle C
  SET capacite = (SELECT :new.capacite / SUM (capacite)
                  FROM   Salle S
                  WHERE  C.id = :new.id);
END;
```

Boucle infinie garantie, et très difficile à comprendre....