Bien débuter : les déclencheurs

Pré-requis

Les déclencheurs ne peuvent s'appliquer que sur des tables dont le moteur de stockage est transactionnel.

Le moteur de stockage par défaut lorsqu'une table est créée est `MyIsam`, généralement qui ne supporte pas les transactions. L'autre moteur de stockage le plus utilisé et qui supporte les transactions est le moteur `InnoDB`, il faut donc utiliser celui-ci.

> Pour changer le moteur de stockage d'une table, exécuter la requête

```
ALTER TABLE nom_table ENGINE=InnoDB;
```

Définition

Un déclencheur est une procédure stockée spéciale qui s'exécute automatiquement lorsqu'un événement se produit dans la base de données. Les événements sont des instructions `INSERT`, `UPDATE` ou `DELETE`. Ces déclencheurs s'activent automatiquement au déclenchement (avant ou après) d'un événement valide, que des lignes de table soient affectées ou non.

> Dans d'autres systèmes de bases de données (SQL Server, Oracle, PostGres), il est possible d'effectuer une commande d'annulation (instruction `ROLLBACK`), dans un trigger, mais pas dans MySql ce qui pose un problème lorsqu'on veut effectuer ce type d'opération. Il existe néanmoins des alternatives qui consistent à générer une erreur volontairement dans le trigger (voir plus loin dans l'exemple) .

Création d'un déclencheur

Schéma type d'un déclencheur :

```
CREATE TRIGGER nom
[MOMENT] [EVENEMENT]
ON [table]
FOR EACH ROW
BEGIN
   -- [requête]
END
```

Autres instructions

- `OLD` et `NEW` : lors de l'exécution d'un trigger, les mots-clés `NEW` et `OLD` sont créés de façon implicite et virtuelle afin de pouvoir manipuler l'état des données avant (`OLD`) et après (`NEW`) son exécution. La syntaxe est `OLD.nom_colonne` / `NEW.nom_colonne`. `OLD` et `NEW` représentent donc la table sur laquelle le déclencheur s'applique.
- `DECLARE` : déclare une variable
- `SET` : affecte une valeur à une variable
- `IF…THEN` : écriture de conditions

Premier exemple

Sur la base `hotel`, prenons un exemple d'une règle de gestion.

> Nous voulons interdire l'insertion des stations dont l'altitude est inférieur à 1000m.

Donc nous allons trouver un moyen d'interdire ce genre de requête:

```
insert into station (sta_nom, sta_altitude) values ('station du bas', 666);
```

Nous pouvons mettre en place un `trigger` en insertion sur la table `station`.

```
CREATE TRIGGER insert_station AFTER INSERT ON station
    FOR EACH ROW
    BEGIN
        DECLARE altitude INT;
        SET altitude = NEW.sta_altitude;
        IF altitude<1000 THEN
            SIGNAL SQLSTATE '40000' SET MESSAGE_TEXT = 'Un problème est survenu. Règle de gestion altitude !';
        END IF;
END;
```

Dans cet exemple, nous utilisons une fonctionnalité particulière des déclencheurs.

> Lorsqu'un déclencheur provoque une erreur, il annule la requête qui a déclenchée le trigger.

Ici, nous générons volontairement une erreur

```
SIGNAL SQLSTATE '40000' SET MESSAGE_TEXT = 'Un problème est survenu. Règle de gestion altitude !';
```

Ce qui provoque l'annulation de la requête qui a déclenchée le trigger

```
insert into station (sta_nom, sta_altitude) values ('station du bas', 666);
```

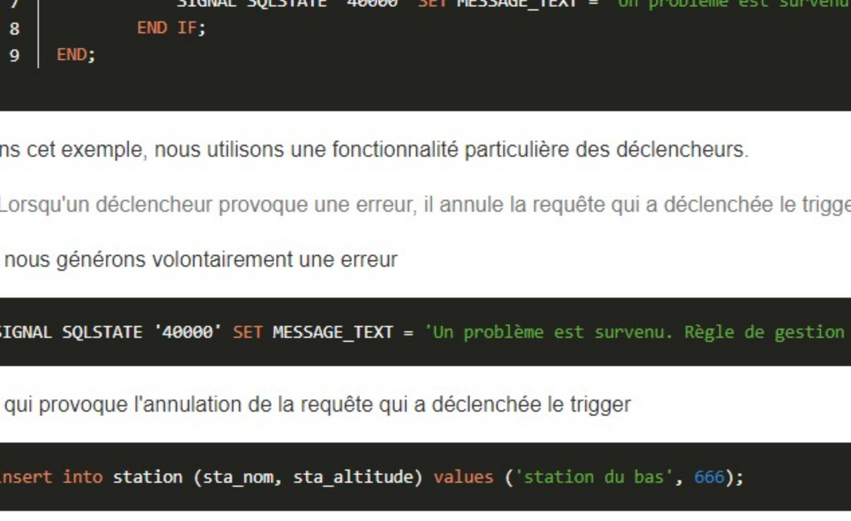
Exercices

A partir de l'exemple suivant, créez les déclencheurs suivants :

1. **modif_reservation** : interdire la modification des réservations (on autorise l'ajout et la suppression).
2. **insert_reservation** : interdire l'ajout de réservation pour les hôtels possédant déjà 10 réservations.
3. **insert_reservation2** : interdire les réservations si le client possède déjà 3 réservations.
4. **insert_chambre** : lors d'une insertion, on calcule le total des capacités des chambres pour l'hôtel, si ce total est supérieur à 50, on interdit l'insertion de la chambre.

Cas pratique

Création de la base de données de test.



Uitlisez ce script pour créeer la base de données.

Dans cette base de données, la table `commande` possède un champ `total`, il est prévu pour stocker le total de la commande !

Lorsque vous ajoutez, supprimez ou modifiez des produits dans la commande (avec la table `lignedecommande`), vous devez penser à recalculer le total de la commande et mettre à jour le champ `total`.

C'est exactement pour ça que les déclencheurs ont été créés.

Nous pouvons mettre en place un déclencheur sur la table `lignedecommande`, qui va se charger de recalculer le total puis mettre à jour la table commande.

```
CREATE TRIGGER maj_total AFTER INSERT ON lignedecommande
    FOR EACH ROW
    BEGIN
        DECLARE id_c INT;
        DECLARE tot DOUBLE;
        SET id_c = NEW.id_commande; -- nous captons le numéro de commande concerné
        SET tot = (SELECT sum(prix*quantite) FROM lignedecommande WHERE id_commande=id_c); -- on recalcul le total
        UPDATE commande SET total=tot WHERE id=id_c; -- on stocke le total dans la table commande
END;
```

Travail à réaliser

1. Mettez en place ce trigger, puis ajoutez un produit dans une commande, vérifiez que le champ total est bien mis à jour.
2. Ce trigger ne fonctionne que lorsque l'on ajoute des produits dans la commande, les modifications ou suppressions ne permettent pas de recalculer le total. Modifiez le code ci-dessus pour faire en sorte que la modification ou la suppression de produit recalcul le total de la commande.
3. Un champ remise était prévu dans la table commande. Prenez en compte ce champ dans le code de votre trigger.

Requêtes utiles

- Pour vérifier l'existence d'un déclencheur, utiliser la commande `SHOW TRIGGERS;` qui affiche la liste des déclencheurs existants dans le système.
- Modification et suppression : il n'est pas possible de modifier un déclencheur; la solution est donc le supprimer puis le recréer avec de nouvelles règles applicables. Utilisez la commande de suppression `DROP TRIGGER nom_trigger;`

Ressources

- les moteurs de stockage
- https://dev.mysql.com/doc/refman/5.7/en/trigger-syntax.html
- http://www.mysqltutorial.org/create-the-first-trigger-in-mysql.aspx
- https://openclassrooms.com/fr/courses/1959476-administrez-vos-bases-de-donnees-avec-mysql/1973090-triggers