SQL – Triggers

Un trigger est une action déclenchée lorsqu'un événement survient sur les données d'une table.

À ce titre, il est attaché à une table d'une base de données.

Ils sont généralement utilisés pour :

- Vérifier les données avant insertion ou modification
- Vérifier si les contraintes seront respectées après suppression, insertion ou modification d'une ligne
- Supprimer les données pour l'utilisateur mais pas dans la DB (positionnement d'un flag à 1, style SAP)
- Journalisation
- Etc.

SQL – Triggers

La requête de création d'un trigger est la suivante :

```
CREATE TRIGGER nom_trigger
moment_trigger action_trigger
ON nom_table
FOR EACH ROW
BEGIN
  instruction(s)
END;
```

SQL – Triggers

- `moment_trigger` indique si les instructions doivent être exécutées avant ou après l'action qui a déclenché le trigger : `BEFORE` ou `AFTER`
- `action_trigger` représente l'action qui déclenche le trigger : `INSERT`, `UPDATE` ou `DELETE`. Une seule action par trigger

Rem : `FOR EACH ROW` est le seul mode implémenté actuellement par MySQL. En SQL2, le choix est possible entre `FOR EACH ROW` qui exécute les instructions pour chaque enregistrement impacté par la requête et `FOR EACH STATEMENT` qui exécute une seule fois les instructions pour tous les enregistrements impactés par la requête

SQL – Triggers

- Sur une table donnée, il ne peut exister qu'un seul trigger pour une paire moment–action
- Par convention, le nom du trigger sera le nom de la table, suivi du moment, suivi de l'action, les éléments étant séparés par des underscores
- On ne peut pas utiliser de `SELECT`, à l'exception des `SELECT … INTO …`

SQL – Triggers

- La version 5.7.2 introduit la possibilité de définir plusieurs fois la même paire moment-action sur une même table.
- L'ordre d'exécution est celui de la création de ces triggers.
- Cependant, deux nouvelles options permettent de modifier cet ordre :
  - `FOLLOWS` : permet de préciser le trigger après lequel l'exécution doit s'effectuer
  - `PRECEDES` : permet de préciser le trigger avant lequel l'exécution doit s'effectuer

SQL – Triggers

```
CREATE TRIGGER nom_trigger
moment_trigger action_trigger
ON nom_table
FOR EACH ROW FOLLOWS nom_trigger_à_suivre
BEGIN
  instruction(s)
END;
```

```
CREATE TRIGGER nom_trigger
moment_trigger action_trigger
ON nom_table
FOR EACH ROW PRECEDES nom_trigger_à_précéder
BEGIN
  instruction(s)
END;
```

SQL – Triggers

La requête de suppression d'un trigger est la suivante :

```
DROP TRIGGER nom_trigger;
```

Rem : pour **modifier** un trigger, il faut d'abord le supprimer puis le recréer.

SQL – Triggers

Pour lister les triggers :

```
SHOW TRIGGERS;
```

Pour obtenir la requête de création d'un trigger :

```
SHOW CREATE TRIGGER nom_trigger;
```

SQL – Triggers

Exemple

Sur la table client, on crée un trigger à exécuter avant une insertion :

```
DROP TRIGGER IF EXISTS client_before_insert;
DELIMITER |

CREATE TRIGGER client_before_insert
BEFORE INSERT
ON client
FOR EACH ROW
BEGIN
  instruction(s)
END|

DELIMITER ;
```

SQL – Triggers

Afin de manipuler les valeurs proposées lors d'une mise à jour et de les comparer aux valeurs existantes, deux objets sont mis à disposition :

- `OLD` : représente les valeurs contenues dans la table (`UPDATE` ou `DELETE`)
- `NEW` : représente les valeurs proposées par la requête (`INSERT` ou `UPDATE`)

SQL – Triggers

Exemple 1

Création d'un trigger qui place la catégorie à A0 lors de l'insertion d'un client, quelle que soit la valeur passée pour celle-ci

```
DROP TRIGGER IF EXISTS client_before_insert;
DELIMITER |

CREATE TRIGGER client_before_insert
BEFORE INSERT
ON client
FOR EACH ROW
BEGIN
  SET NEW.cat := 'A0';
END|

DELIMITER ;
```

SQL – Triggers

Exemple 2

Création d'un trigger qui met à jour la catégorie en fonction de la valeur du compte lors de la mise à jour d'un client :

- Si compte < 1000, A0
- Si compte <5000, B0
- Sinon, C0

SQL – Triggers

Exemple 2

```
DROP TRIGGER IF EXISTS client_before_update;
DELIMITER |

CREATE TRIGGER client_before_update
BEFORE UPDATE
ON client
FOR EACH ROW
BEGIN
  IF NEW.compte != OLD.compte THEN
  -- mise à jour du compte
    IF NEW.compte < 1000 THEN
      SET NEW.cat := 'A0';
    ELSEIF NEW.compte < 5000 THEN
      SET NEW.cat := 'B0';
    ELSE
      SET NEW.cat := 'C0';
    END IF;
  END IF;
END|

DELIMITER ;
```

SQL – Triggers

Exercices

Créez un trigger :

1) pour formater les noms et les prénoms avec une majuscule, toutes les autres lettres étant en minuscules lors d'une insertion ou d'un update dans la table client

2) pour mettre à jour le stock d'un produit lorsqu'une commande est passée (cas trivial où le stock est supposé suffisant)

Triggers – Journalisation

Afin de sauvegarder les opérations réalisées sur une table, nous ajouterons généralement les quatre colonnes suivantes :

1. `created_at : DATETIME`
2. `created_by : VARCHAR(50)`
3. `updated_at : DATETIME`
4. `updated_by : VARCHAR(50)`

Chaque colonne étant facultative afin de ne pas contraindre l'utilisateur de les placer dans ses requêtes d'insertion et de mise à jour.

Cette convention est appliquée par la plupart des ORM.

Triggers – Journalisation

Ce qui donne sur la table client :

```
ALTER TABLE client ADD COLUMN created_at DATETIME NULL;
ALTER TABLE client ADD COLUMN created_by VARCHAR(50) NULL;
ALTER TABLE client ADD COLUMN updated_at DATETIME NULL;
ALTER TABLE client ADD COLUMN updated_by VARCHAR(50) NULL;
```

Triggers – Journalisation

Exemple 3

```
DROP TRIGGER IF EXISTS client_before_insert;
DELIMITER |

CREATE TRIGGER client_before_insert
BEFORE INSERT
ON client
FOR EACH ROW
BEGIN
  SET NEW.created_at := NOW();
  SET NEW.created_by := CURRENT_USER();
END|

DELIMITER ;
```

Triggers – Journalisation

Exemple 3 - suite

```
DROP TRIGGER IF EXISTS client_before_update;
DELIMITER |

CREATE TRIGGER client_before_update
BEFORE UPDATE
ON client
FOR EACH ROW
BEGIN
  SET NEW.updated_at := NOW();
  SET NEW.updated_by := CURRENT_USER();
END|

DELIMITER ;
```

Triggers – Historisation

Il est également possible de sauvegarder tout ce qui se passe sur une table :

- Toutes les valeurs avant chaque mise à jour
- Toutes les valeurs avant suppression

Pour cela, il faut créer une nouvelle table associée à la table à historiser et au schéma identique à celle-ci.

SQL – Triggers

Exercices

Créez un trigger :

3) pour vérifier s'il reste suffisamment de produits en stock pour la quantité commandée. Si ce n'est pas le cas, limitez la quantité commandée à ce qui reste en stock. N'oubliez pas de mettre à jour le stock (cas trivial où on ne passera pas de commandes s'il ne reste rien en stock)

4) pour passer une commande à un fournisseur lorsque le stock d'un produit est égal à 0. Ajoutez la colonne stock_minimal à la table produit et créez la table commande_externe avec :

- Le nom du fournisseur
- L'idproduit
- La quantité

SQL – Triggers

Exercices

Créez un trigger :

5) Pour historiser tous les mouvements de stock

6) Pour créer une commande lors d'une insertion d'une ligne de détail dont l'idcommande n'existerait pas ou serait positionné à `NULL`