MySQL perfecionnement

Les déclencheurs

Imane Ait Yahiatene

Définition

Un déclencheur est un ensemble d'actions qui sont exécutées automatiquement lorsqu'une opération de modification spécifiée (instruction SQL INSERT, UPDATE ou DELETE) est effectuée sur une table donnée. Les déclencheurs sont utiles pour des tâches telles que l'application de règles commerciales, la validation de données d'entrée et la conservation d'une piste d'audit.

Utilisation des déclencheurs

- Appliquer les règles de gestion
- Valider les données d'entrée
- Générer une valeur unique pour une ligne nouvellement insérée dans un autre fichier.
- écrire dans d'autres fichiers à des fins de vérification
- Interroger d'autres fichiers à des fins de référencement croisé.
- Accéder aux fonctions du système
- Répliquer des données dans différents fichiers pour assurer la cohérence des données.

Pourquoi les déclencheurs?

- Développement plus rapide des applications. Comme la base de données stocke les déclencheurs, vous ne devez pas coder les actions de déclenchement dans chaque application de la base de données.
- Application globale des règles de gestion. Définissez un déclencheur une seule fois, puis réutilisez-le pour toutes les applications qui utilisent la base de données.
- Maintenance simplifiée. Si une règle de gestion change, vous devez modifier uniquement le programme de déclenchement correspondant au lieu de chaque programme d'application.
- Amélioration des performances dans un environnement client/serveur. Toutes les règles s'exécutent sur le serveur avant le retour du résultat.

Clauses utilisées pour les déclencheurs

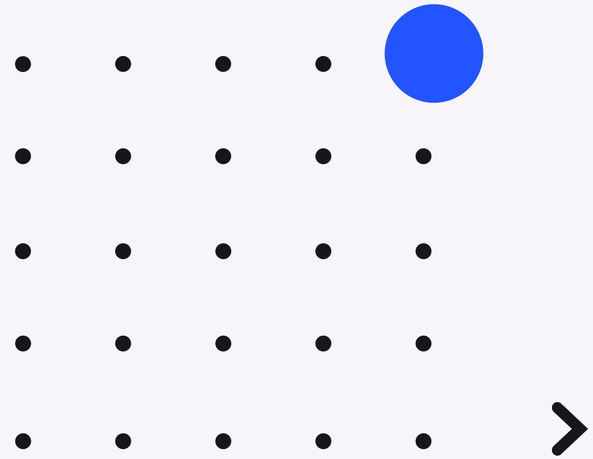
- Compound statements (BEGIN / END)
- Variable declaration (DECLARE) and assignment (SET)
- Flow-of-control statements (IF, CASE, WHILE, LOOP, WHILE, REPEAT, LEAVE, ITERATE)
- Condition declarations
- Handler declarations

Déclencheur au niveau des lignes

ils sont activé pour chaque ligne, par exemple, si une table a inséré, mis à jour ou supprimé plusieurs lignes, le déclencheur est activé automatiquement pour chaque ligne affectée par l'instruction de déclenchement.

Déclaration d'un déclencheur

```
CREATE TRIGGER nom_trigger
moment_trigger evenement_trigger
 ON nom_table
 FOR EACH ROW
BEGIN
 --variable declarations
 --trigger code
 END;
```



Explication

Les conventions d'appellation sont utilisées pour organiser les déclencheurs, c'est pourquoi il est recommandé d'utiliser un nom unique pour chaque déclencheur associé à la même table, cependant nous pouvons donner le même nom aux déclencheurs pour différentes tables, voici une bonne convention d'appellation :
(BEFORE | AFTER) nom_table (INSERT | UPDATE | DELETE)

- **moment_trigger**

  Il s'agit du moment du déclenchement de l'événement, qui peut être avant ou après.

  (BEFORE|AFTER)

- **evenement_trigger**

  il s'agit de l'événement qui a déclenché la procédure, dans MYSQL, il peut s'agir d'une insertion, d'une mise à jour ou d'une suppression.

  (INSERT|DELETE|UPDATE)

Exemple: AFTER INSERT

```
ql> SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID, SALARY, COMMISSION_PCT
----------+------------+-----------+---------+----------+----------------+
MPLOYEE_ID | FIRST_NAME | LAST_NAME | JOB_ID  | SALARY   | COMMISSION_PCT |
----------+------------+-----------+---------+----------+----------------+
       100 | Steven     | King      | AD_PRES | 24000.00 |           0.10 |
       101 | Neena      | Kochhar   | AD_VP   | 17000.00 |           0.50 |
       102 | Lex        | De Haan   | AD_VP   | 17000.00 |           0.50 |
       103 | Alexander  | Hunold    | IT_PROG |  9000.00 |           0.25 |
       104 | Bruce      | Ernst     | IT_PROG |  6000.00 |           0.25 |
       105 | David      | Austin    | IT_PROG |  4800.00 |           0.25 |
----------+------------+-----------+---------+----------+----------------+
ows in set (0.00 sec)
```

Exemple: AFTER INSERT

```
mysql> SELECT * FROM log_emp_details;
+-------------+----------+---------------------+
| emp_details | SALARY   | EDTTIME             |
+-------------+----------+---------------------+
|         100 | 24000.00 | 2011-01-15 00:00:00 |
|         101 | 17000.00 | 2010-01-12 00:00:00 |
|         102 | 17000.00 | 2010-09-22 00:00:00 |
|         103 |  9000.00 | 2011-06-21 00:00:00 |
|         104 |  6000.00 | 2012-07-05 00:00:00 |
|         105 |  4800.00 | 2011-06-21 00:00:00 |
+-------------+----------+---------------------+
6 rows in set (0.02 sec)
```

Exemple: AFTER INSERT

```
DELIMITER
$$
USE `hr`
$$
CREATE
TRIGGER `hr`.`emp_details_AINS` AFTER INSERT ON `hr`.`employees `
FOR EACH ROW
-- Edit trigger body code below this line. Do not edit lines above this one
BEGIN
INSERT INTO log_emp_details
VALUES(NEW.employee_id, NEW.salary, NOW());
END$$
```

Exemple: AFTER INSERT

```
mysql> INSERT INTO emp_details VALUES(236, 'RABI', 'CHANDRA', 'RABI','590.423.45700',
Query OK, 1 row affected (0.07 sec)

mysql> SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID, SALARY, COMMISSION_PCT FROM
+-------------+------------+-----------+---------+----------+----------------+
| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | JOB_ID  | SALARY   | COMMISSION_PCT |
+-------------+------------+-----------+---------+----------+----------------+
|         100 | Steven     | King      | AD_PRES | 24000.00 |           0.10 |
|         101 | Neena      | Kochhar   | AD_VP   | 17000.00 |           0.50 |
|         102 | Lex        | De Haan   | AD_VP   | 17000.00 |           0.50 |
|         103 | Alexander  | Hunold    | IT_PROG |  9000.00 |           0.25 |
|         104 | Bruce      | Ernst     | IT_PROG |  6000.00 |           0.25 |
|         105 | David      | Austin    | IT_PROG |  4800.00 |           0.25 |
|         236 | RABI       | CHANDRA   | AD_VP   | 15000.00 |           0.50 |
+-------------+------------+-----------+---------+----------+----------------+
7 rows in set (0.00 sec)
```

Exemple: AFTER INSERT

```
mysql> SELECT * FROM log_emp_details;
+-------------+----------+---------------------+
| emp_details | SALARY   | EDTTIME             |
+-------------+----------+---------------------+
|         100 | 24000.00 | 2011-01-15 00:00:00 |
|         101 | 17000.00 | 2010-01-12 00:00:00 |
|         102 | 17000.00 | 2010-09-22 00:00:00 |
|         103 |  9000.00 | 2011-06-21 00:00:00 |
|         104 |  6000.00 | 2012-07-05 00:00:00 |
|         105 |  4800.00 | 2011-06-21 00:00:00 |
|         236 | 15000.00 | 2013-07-15 16:52:24 |
+-------------+----------+---------------------+
7 rows in set (0.00 sec)
```

Exemple: BEFORE INSERT

```
USE `hr`;
DELIMITER
$$
CREATE TRIGGER `emp_details_BINS`
BEFORE INSERT
ON employees FOR EACH ROW
-- Edit trigger body code below this line. Do not edit lines above this
one
BEGIN
SET NEW.FIRST_NAME = TRIM(NEW.FIRST_NAME);
SET NEW.LAST_NAME = TRIM(NEW.LAST_NAME);
SET NEW.JOB_ID = UPPER(NEW.JOB_ID);END;
$$
```

Exemple: BEFORE INSERT

```
mysql> INSERT INTO emp_details VALUES (334, ' Ana ', ' King', 'ANA', '690.432.45701',
Query OK, 1 row affected (0.04 sec)
```

Now list the following fields of emp_details :

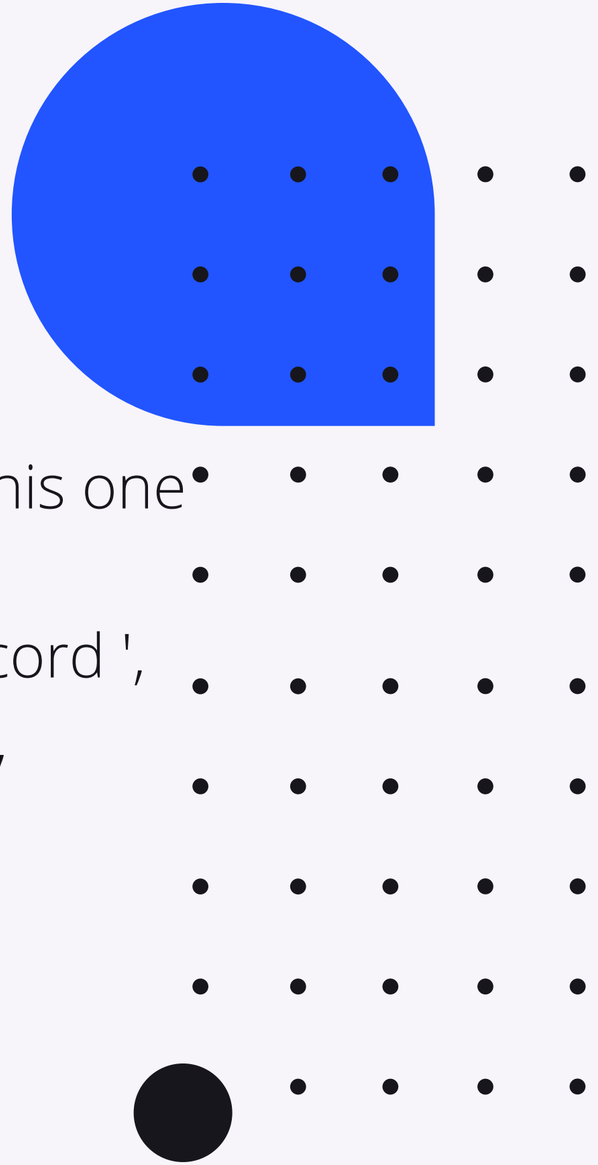
```
mysql> SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID, SALARY, COMMISSION_PCT FROM
+-------------+------------+-----------+---------+----------+----------------+
| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | JOB_ID  | SALARY   | COMMISSION_PCT |
+-------------+------------+-----------+---------+----------+----------------+
|         100 | Steven     | King      | AD_PRES | 24000.00 |           0.10 |
|         101 | Neena      | Kochhar   | AD_VP   | 17000.00 |           0.50 |
|         102 | Lex        | De Haan   | AD_VP   | 17000.00 |           0.50 |
|         103 | Alexander  | Hunold    | IT_PROG |  9000.00 |           0.25 |
|         104 | Bruce      | Ernst     | IT_PROG |  6000.00 |           0.25 |
|         105 | David      | Austin    | IT_PROG |  4800.00 |           0.25 |
|         236 | RABI       | CHANDRA   | AD_VP   | 15000.00 |           0.50 |
|         334 | Ana        | King      | IT_PROG | 17000.00 |           0.50 |
+-------------+------------+-----------+---------+----------+----------------+
8 rows in set (0.00 sec)
```

Example AFTER UPDATE

```
mysql> SELECT * FROM STUDENT_MAST;
+------------+------------------+----------+
| STUDENT_ID | NAME             | ST_CLASS |
+------------+------------------+----------+
|          1 | Steven King      |        7 |
|          2 | Neena  Kochhar   |        8 |
|          3 | Lex  De Haan     |        8 |
|          4 | Alexander Hunold |       10 |
+------------+------------------+----------+
4 rows in set (0.00 sec)
```

Example AFTER UPDATE

```
TRIGGER `test`.`student_mast_AUPD`
AFTER UPDATE
ON `test`.`student_mast`FOR EACH ROW
-- Edit trigger body code below this line. Do not edit lines above this one
BEGIN
INSERT into stu_log VALUES (user(), CONCAT('Update Student Record ',
        OLD.NAME,' Previous Class :',OLD.ST_CLASS,' Present Class ',
        NEW.st_class));
END
$$
```



Example AFTER UPDATE

```
mysql> UPDATE STUDENT_MAST SET ST_CLASS = ST_CLASS + 1;
Query OK, 4 rows affected (0.20 sec)
Rows matched: 4
Changed: 4
Warnings: 0
```

Example AFTER UPDATE

```
mysql> SELECT * FROM STUDENT_MAST;
+------------+------------------+----------+
| STUDENT_ID | NAME             | ST_CLASS |
+------------+------------------+----------+
|          1 | Steven King      |        8 |
|          2 | Neena  Kochhar   |        9 |
|          3 | Lex  De Haan     |        9 |
|          4 | Alexander Hunold |       11 |
+------------+------------------+----------+
4 rows in set (0.00 sec)mysql> SELECT * FROM STU_LOG;
+----------------+-----------------------------------------------------------------
| user_id        | description
+----------------+-----------------------------------------------------------------
| root@localhost | Update Student Record Steven King Previous Class :7 Present Class
| root@localhost | Update Student Record Neena  Kochhar Previous Class :8 Present Cla
| root@localhost | Update Student Record Lex  De Haan Previous Class :8 Present Class
| root@localhost | Update Student Record Alexander Hunold Previous Class :10 Present
+----------------+-----------------------------------------------------------------
4 rows in set (0.00 sec)
```

Exemple: BEFORE UPDATE

```
mysql> SELECT * FROM STUDENT_MARKS;
+------------+------------------+------+------+------+------+------+-------+--------
| STUDENT_ID | NAME             | SUB1 | SUB2 | SUB3 | SUB4 | SUB5 | TOTAL | PER_MAR
+------------+------------------+------+------+------+------+------+-------+--------
|          1 | Steven King      |    0 |    0 |    0 |    0 |    0 |     0 |      0.
|          2 | Neena  Kochhar   |    0 |    0 |    0 |    0 |    0 |     0 |      0.
|          3 | Lex  De Haan     |    0 |    0 |    0 |    0 |    0 |     0 |      0.
|          4 | Alexander Hunold |    0 |    0 |    0 |    0 |    0 |     0 |      0.
+------------+------------------+------+------+------+------+------+-------+--------
4 rows in set (0.00 sec)
```

Exemple: BEFORE UPDATE

Maintenant que l'examen est terminé et que nous avons reçu les notes de toutes les matières, nous allons mettre à jour le tableau, les notes totales de toutes les matières, le pourcentage des notes totales et la note seront automatiquement calculés. Pour cet exemple de calcul, les conditions suivantes sont supposées :

Total des notes (sera stocké dans la colonne TOTAL) : TOTAL = SUB1 + SUB2 + SUB3 + SUB4 + SUB5

Pourcentage des notes (sera stocké dans la colonne PER_MARKS) : PER_MARKS = (TOTAL)/5

Note (sera stocké dans la colonne GRADE) :
- Si PER_MARKS>=90 -> 'EXCELLENT'.
- Si PER_MARKS>=75 ET PER_MARKS<90 -> 'TRES BON'.
- Si PER_MARKS>=60 ET PER_MARKS<75 -> 'BON'.
- Si PER_MARKS>=40 ET PER_MARKS<60 -> 'MOYEN'.
- Si PER_MARKS<40-> 'NON PROMOTIONNÉ'.

Exemple: BEFORE UPDATE

```
CREATE TRIGGER `student_marks_BUPD`
BEFORE UPDATE
ON student_marks FOR EACH ROW
-- Edit trigger body code below this line. Do not edit lines above this one
BEGIN
SET NEW.TOTAL = NEW.SUB1 + NEW.SUB2 + NEW.SUB3 + NEW.SUB4 + NEW.SUB5;
SET NEW.PER_MARKS = NEW.TOTAL/5;
IF NEW.PER_MARKS >=90 THEN
SET NEW.GRADE = 'EXCELLENT';
ELSEIF NEW.PER_MARKS>=75 AND NEW.PER_MARKS<90 THEN
SET NEW.GRADE = 'VERY GOOD';
ELSEIF NEW.PER_MARKS>=60 AND NEW.PER_MARKS<75 THEN
SET NEW.GRADE = 'GOOD';
ELSEIF NEW.PER_MARKS>=40 AND NEW.PER_MARKS<60 THEN
SET NEW.GRADE = 'AVERAGE';
ELSESET NEW.GRADE = 'NOT PROMOTED';
END IF;
END;
$$
```

Exemple: BEFORE UPDATE

```
mysql> UPDATE STUDENT_MARKS SET SUB1 = 54, SUB2 = 69, SUB3 = 89, SUB4 = 87, SUB5 = 59
Query OK, 1 row affected (0.05 sec)
Rows matched: 1
Changed: 1
Warnings: 0
```

Exemple: BEFORE UPDATE

```
mysql> SELECT * FROM STUDENT_MARKS;
+------------+------------------+------+------+------+------+------+-------+---------
| STUDENT_ID | NAME             | SUB1 | SUB2 | SUB3 | SUB4 | SUB5 | TOTAL | PER_MARK
+------------+------------------+------+------+------+------+------+-------+---------
|          1 | Steven King      |   54 |   69 |   89 |   87 |   59 |   358 |     71.6
|          2 | Neena  Kochhar   |    0 |    0 |    0 |    0 |    0 |     0 |      0.0
|          3 | Lex  De Haan     |    0 |    0 |    0 |    0 |    0 |     0 |      0.0
|          4 | Alexander Hunold |    0 |    0 |    0 |    0 |    0 |     0 |      0.0
+------------+------------------+------+------+------+------+------+-------+---------
4 rows in set (0.00 sec)
```

Example AFTER DELETE

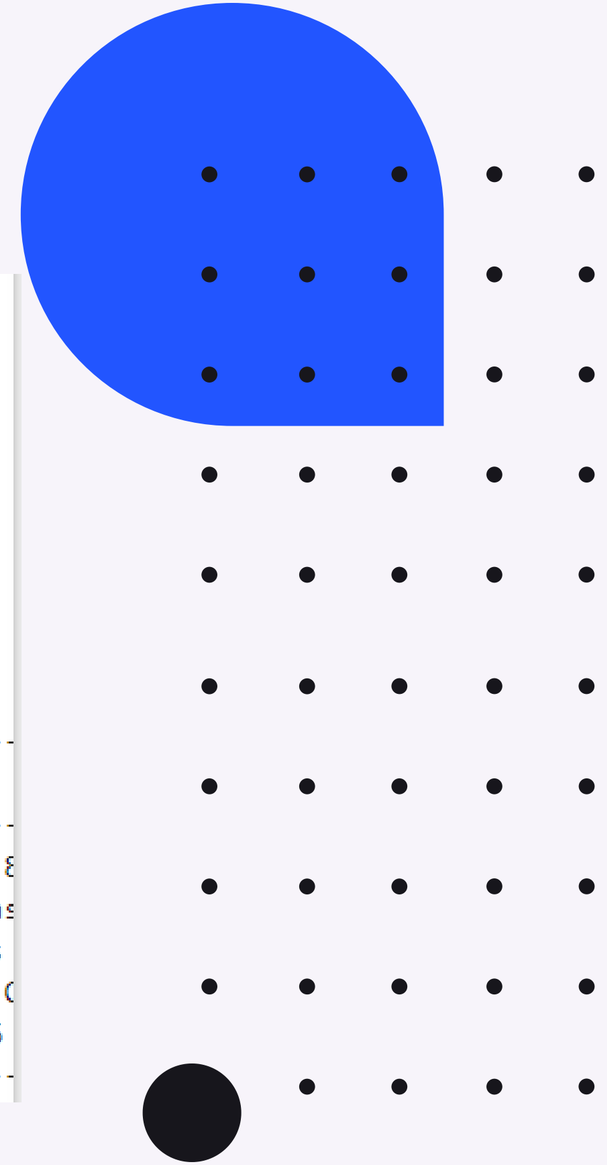
```
CREATE TRIGGER `student_mast_ADEL`
AFTER DELETE ON student_mast FOR EACH ROW
-- Edit trigger body code below this line. Do not edit lines above this one
BEGIN
INSERT into stu_log VALUES (user(), CONCAT('Update Student Record ',
        OLD.NAME,' Clas :',OLD.ST_CLASS, '-> Deleted on ', NOW()));
END;
$$
```

Example AFTER DELETE

```
mysql> DELETE FROM STUDENT_MAST WHERE STUDENT_ID = 1;
Query OK, 1 row affected (0.06 sec)
```

Example AFTER DELETE

```
mysql> SELECT * FROM STUDENT_MAST;
+------------+------------------+----------+
| STUDENT_ID | NAME             | ST_CLASS |
+------------+------------------+----------+
|          2 | Neena  Kochhar   |        9 |
|          3 | Lex  De Haan     |        9 |
|          4 | Alexander Hunold |       11 |
+------------+------------------+----------+
3 rows in set (0.00 sec)
mysql> SELECT * FROM STU_LOG;
+----------------+-----------------------------------------------------------------
| user_id        | description
+----------------+-----------------------------------------------------------------
| root@localhost | Update Student RecordSteven King Previous Class :7 Present Class 8
| root@localhost | Update Student RecordNeena  Kochhar Previous Class :8 Present Clas
| root@localhost | Update Student RecordLex  De Haan Previous Class :8 Present Class
| root@localhost | Update Student RecordAlexander Hunold Previous Class :10 Present C
| root@localhost | Update Student Record Steven King Clas :8-> Deleted on 2013-07-16
+----------------+-----------------------------------------------------------------
```



Comment MySQL gère-t-il les erreurs pendant l'exécution du trigger ?

- Si un déclencheur BEFORE échoue, l'opération sur la ligne correspondante n'est pas effectuée.
- Un déclencheur BEFORE est activé par la tentative d'insertion ou de modification de la ligne, que cette tentative aboutisse ou non par la suite.
- Un déclencheur AFTER n'est exécuté que si tous les déclencheurs BEFORE et l'opération sur la ligne s'exécutent avec succès.
- Une erreur au cours d'un déclencheur BEFORE ou AFTER entraîne l'échec de toute l'instruction qui a provoqué l'invocation du déclencheur.
- Pour les tables transactionnelles, l'échec d'une instruction doit entraîner un retour en arrière de toutes les modifications effectuées par l'instruction.

Suppression d'un déclencheur

Nous pouvons supprimer un déclencheur dans MySQL en utilisant l'instruction DROP TRIGGER. Vous devez être très prudent lorsque vous supprimez un déclencheur de la table. En effet, une fois que nous avons supprimé le déclencheur, il ne peut pas être récupéré. Si un déclencheur n'est pas trouvé, l'instruction DROP TRIGGER génère une erreur. Si nous supprimons une table, tous ses déclencheurs sont également supprimés.

Syntaxe

DROP TRIGGER [IF EXISTS] [nom_schéma.]nom_trigger

Si nous omettons le nom du schéma, l'instruction supprimera le déclencheur de la base de données actuelle.