Bases de données

Ecole Marocaine des Sciences de l'Ingénieur

© Yousra Lembachar

Déclencheurs

Qu'est-ce qu'un déclencheur?

Un déclencheur ou une règle ECA est de la forme:

> «Quand un Evénement se produit, si une Condition est satisfaite, une Action est exécutée.»

Exemple: Quand un employé est évalué, si l'évaluation est > 70, lui attribuer une augmentation de salaire de 10%.

Autres exemples

1. Log des utilisateurs/changements
2. Exécuter des règles métier
3. Renforcer les contraintes d'intégrité
4. Dupliquer des données
5. Améliorer les performances et les changements

Déclencheurs dans SQL

```
CREATE TRIGGER déclencheur
(événement)
BEFORE | AFTER

INSERT | DELETE | UPDATE ON nom_table

[REFERENCING variables]

[FOR EACH ROW]
(condition)
[WHEN] [condition]
 (action)
action
```

FOR EACH ROW

Quand «FOR EACH ROW» est spécifié, le déclencheur est exécuté pour chaque ligne insérée, mise à jour ou supprimée par le déclencheur => Déclencheur par lignes (row-level)

Quand «FOR EACH ROW» n'est pas spécifié, le déclencheur est exécuté une seule fois => Déclencheur par opération (statement-level)

Dans MySQL, «FOR EACH ROW» est obligatoire. Si on insère 3 lignes, le déclencheur est exécuté 3 fois.

VARIABLES

OLD ROW: Pour l'ancienne ligne (delete, update, for each row)

NEW ROW: Pour la nouvelle ligne (insert, update, for each row)

OLD TABLE: Pour l'ancienne table (delete, update)

NEW TABLE: Pour la nouvelle table (insert, update)

Dans MySQL, «FOR EACH ROW» est obligatoire => OLD référence la ligne ancienne, et NEW la nouvelle ligne.

FOR EACH ROW

Il y a des déclencheurs par opération (statement-level) qui sont difficiles (voir même impossibles) à reproduire avec des déclencheurs par lignes(row-level)

Exemple: Si tous les étudiants ont une meilleure moyenne lors de la mise à jour de la table NOTE, leur ajouter un bonus.

```
AFTER UPDATE ON NOTE

REFERENCING OLD TABLE AS o, NEW TABLE AS n

WHEN o.note < n.note

UPDATE NOTE SET note = note + 2;
```

Exemple: Mise à jour en cascade

```
AFTER UPDATE ON ETUDIANT

REFERENCING OLD ROW AS old, NEW ROW as new

FOR EACH ROW

UPDATE NOTE SET idEtudiant = new.idEtudiant

    WHERE idEtudiant = old.idEtudiant;
```

NOTE est mise à jour s'il y a une référence entre ETUDIANT et NOTE. Ceci requiert de ne pas avoir de contraintes d'intégrité référentielle «RESTRICT» ou «SET NULL» entre NOTE et ETUDIANT.

Plusieurs déclencheurs

- Les implémentations différent d'un système à un autre => Il n'y a pas deux systèmes avec le même standard SQL
- MySQL n'autorise qu'un seul déclencheur par opération (INSERT, DELETE, UPDATE) par table, d'autres systèmes autorisent plusieurs => Comportement différent avec résultats différents
- Possibilité d'avoir plusieurs déclencheurs qui s'exécutent à la fois => Différents résultats, des cycles d'exécution infinies, des déclencheurs qui se déclenchent eux-mêmes, des actions qui déclenchent des déclencheurs...
- Faire attention à l'interaction avec les contraintes

Différentes implémentations

- PostGreS
  - Déclencheurs par ligne et par opération
  - OLD/NEW ROW + OLD/NEW TABLE
- SQLite
  - Déclencheurs par ligne - ~~OLD/NEW TABLE~~
  - OLD/NEW ROW qui devient OLD et NEW
- MySQL
  - Déclencheurs par ligne - ~~OLD/NEW TABLE~~
  - OLD/NEW ROW qui devient OLD et NEW
  - Un seul déclencheur par événement
  - Enchainement limité

Déclencheurs dans MySQL

Déclencheurs dans MySQL

CREATE TRIGGER déclencheur

(événement)

BEFORE | AFTER

INSERT | DELETE | UPDATE ON nom_table

FOR EACH ROW

(condition)

[condition]

(action)

action

Base de données

etudiant | matiere | note

Filter: Edit: File:

idEtudiant	nom
1	Samantha
2	Francis
3	Charles
4	Penelope
5	Craig
6	Steve
7	Dave
9	Cameron
10	Patricia
11	Patricia
	NULL

- universite
  - Tables
    - etudiant
    - matiere
    - MOYENNE
    - NOTE
    - RATTRAPAGE

etudiant | matiere | note

Filter: Edit: File:

idMatiere	nomMatiere
1	Maths
2	Sociologie
3	Nutrition
4	Art Moderne
5	Art Contempo…
6	Art Classique
	NULL

MOYENNE

Filter: Edi

idEtudiant	moyenne
1	18
2	20
3	13
4	14
5	10
6	8
10	5
11	7
12	15

RATTRAPAGE | Query 1

Filter: Edi

idEtudiant	idMatiere
1	4
6	1
10	1
10	2
11	3
	NULL

etudiant | matiere | note

Filter: Edit: File:

idEtudiant	idMatiere	note
6	1	8
11	3	7
10	1	5
10	2	4
2	1	20
1	2	20
1	1	15
4	1	14
3	1	13
5	1	10
	NULL	NULL

AFTER INSERT

```sql
DROP TRIGGER IF EXISTS AFTER_INSERT_ETUDIANT; -- Supprime le déclencheur
                                              -- s'il existe

CREATE TRIGGER AFTER_INSERT_ETUDIANT -- Création du déclencheur
AFTER INSERT ON ETUDIANT             -- qui s'exécute après chaque insertion
FOR EACH ROW                         -- pour chaque ligne
INSERT INTO MOYENNE VALUES (NEW.idEtudiant, 0);

INSERT INTO ETUDIANT VALUES (8, 'Charlie'); -- Insertion de Charlie

SELECT
    *
FROM
    ETUDIANT,
    MOYENNE
WHERE
    ETUDIANT.idEtudiant = MOYENNE.idEtudiant;
```

idEtudiant	nom	idEtudiant	moyenne
1	Samantha	1	18
2	Francis	2	20
3	Charles	3	13
4	Penelope	4	14
5	Craig	5	10
6	Steve	6	8
8	Charlie	8	0
10	Patricia	10	5
11	Patricia	11	7
12	Alba	12	15

Nouvel étudiant => Moyenne = 0

BEFORE INSERT

```sql
DROP TRIGGER IF EXISTS BEFORE_INSERT_ETUDIANT;

DELIMITER $$
CREATE TRIGGER BEFORE_INSERT_ETUDIANT    --  Création du déclencheur
BEFORE INSERT ON ETUDIANT                -- Avant chaque insertion
FOR EACH ROW BEGIN                       -- Pour chaque ligne/Début du bloc
    DELETE FROM NOTE WHERE idEtudiant = NEW.idEtudiant;    -- 1ere requete
    DELETE FROM MOYENNE WHERE idEtudiant = NEW.idEtudiant; -- 2eme requete
END;
$$
DELIMITER ;

INSERT INTO NOTE VALUES (12, 1, 20);
INSERT INTO MOYENNE VALUES (12, 20);
INSERT INTO ETUDIANT VALUES (12, 'Damien');

SELECT
    *
FROM
    ETUDIANT,
    NOTE,
    MOYENNE
WHERE
    ETUDIANT.idEtudiant
        = NOTE.idEtudiant
        AND ETUDIANT.idEtudiant
        = MOYENNE.idEtudiant
;
```

Utiliser DELIMITER dans le cas de BEGIN et END

Nouvel étudiant => Supprimer les notes et la moyenne de cet étudiant si elles existent.

idEtudiant	nom	idEtudiant	idMatiere	note	idEtudiant	moyenne
1	Samantha	1	1	15	1	18
1	Samantha	1	2	20	1	18
2	Francis	2	1	20	2	20
3	Charles	3	1	13	3	13
4	Penelope	4	1	14	4	14
5	Craig	5	1	10	5	10
6	Steve	6	1	8	6	8
10	Patricia	10	1	5	10	5
10	Patricia	10	2	4	10	5
11	Patricia	11	3	7	11	7

AFTER INSERT

```sql
DROP TRIGGER IF EXISTS AFTER_INSERT_NOTE; -- Supprime le déclencheur
                                          -- s'il existe

DELIMITER $$
CREATE TRIGGER AFTER_INSERT_NOTE     --  Création du déclencheur
AFTER INSERT ON NOTE                 -- Après chaque insertion ds note
FOR EACH ROW BEGIN                   -- Pour chaque ligne/Début du bloc
    IF                               -- Début du bloc IF
        NEW.note < 10                -- Si la note < 10
    THEN                             -- L'étudiant a un rattrapage
        INSERT INTO RATTRAPAGE VALUES (NEW.idEtudiant, NEW.idMatiere);
    END IF;                          -- Fin du bloc IF
END;                                 -- Fin du bloc
$$
DELIMITER ;

INSERT INTO NOTE VALUES (1, 4, 8);

SELECT * FROM RATTRAPAGE;
```

Filter: Edi

idEtudiant	idMatiere
1	4
6	1
10	1
10	2
11	3
	NULL

Nouvel note => Si note < 10,
l'étudiant a un rattrapage

AFTER DELETE

```sql
DROP TRIGGER IF EXISTS AFTER_DELETE_ETUDIANT; -- Supprime le déclencheur
                                              -- s'il existe

DELIMITER $$
CREATE TRIGGER AFTER_DELETE_ETUDIANT     --  Création du déclencheur
AFTER DELETE ON ETUDIANT                 -- Après chaque suppression
FOR EACH ROW BEGIN                       -- Pour chaque ligne/Début du bloc
    DELETE FROM NOTE WHERE idEtudiant = OLD.idEtudiant;
    DELETE FROM MOYENNE WHERE idEtudiant = OLD.idEtudiant;
    DELETE FROM RATTRAPAGE WHERE idEtudiant = OLD.idEtudiant;
END;                                     -- Fin du bloc
$$
DELIMITER ;

INSERT INTO ETUDIANT VALUES (15, 'Edward'); -- Insertion d'un nouvel étudiant
INSERT INTO NOTE VALUES (15,1,8);           -- Crée un rattrapage pour Edward
DELETE FROM ETUDIANT
WHERE
    idEtudiant = 15; -- Déclenche AFTER_DELETE_ETUDIANT

SELECT
    *
FROM
    ETUDIANT,
    NOTE,
    RATTRAPAGE
WHERE
    ETUDIANT.idEtudiant = NOTE.idEtudiant
        AND RATTRAPAGE.idEtudiant = ETUDIANT.idEtudiant
```

Equivalent à
ON DELETE CASCADE

idEtudiant	nom	idEtudiant	idMatiere	note	idEtudiant	idMatiere
1	Samantha	1	1	15	1	4
1	Samantha	1	2	20	1	4
1	Samantha	1	4	8	1	4
6	Steve	6	1	8	6	1
10	Patricia	10	1	5	10	1
10	Patricia	10	2	4	10	1
10	Patricia	10	1	5	10	2
10	Patricia	10	2	4	10	2
11	Patricia	11	3	7	11	3

Exemple: Contrainte de clé unique

```sql
DROP TRIGGER IF EXISTS UNIQUE_NOM;          -- Supprime le déclencheur s'il existe

DELIMITER $$
CREATE TRIGGER UNIQUE_NOM                   --  Création du déclencheur
BEFORE INSERT ON ETUDIANT                   -- Avant chaque insertion
FOR EACH ROW BEGIN
IF EXISTS (SELECT nom FROM ETUDIANT WHERE nom = NEW.nom) THEN -- Si le nom existe
    SIGNAL SQLSTATE '45000'                 -- Une erreur est levée
        set message_text = 'Tentative d''insertion d''un nom qui existe deja!';
END IF;
END;
$$
DELIMITER ;


INSERT INTO ETUDIANT VALUES (20, 'Samantha');
```

46:16

Action Output

		Time	A	Response
✓	1	17:16:20	D	0 row(s) affected
✓	2	17:16:20	C	0 row(s) affected
✗	3	17:16:20	I.	Error Code: 1644. Tentative d'insertion d'un nom qui existe deja!

Si le nom existe, une erreur est levée et l'insertion n'est pas effectuée.

Exemple: Chaînage des déclencheurs

```
CREATE TRIGGER AFTER_DELETE_ETUDIANT     --  Création du déclencheur
AFTER DELETE ON ETUDIANT                 -- Après chaque suppression
FOR EACH ROW
    DELETE FROM NOTE WHERE idEtudiant = OLD.idEtudiant;


DELIMITER $$
CREATE TRIGGER AFTER_DELETE_NOTE     --  Création du déclencheur
AFTER DELETE ON NOTE                 -- Après chaque suppression
FOR EACH ROW BEGIN                    -- Pour chaque ligne/Début du bloc
    DELETE FROM MOYENNE WHERE idEtudiant = OLD.idEtudiant;
    DELETE FROM RATTRAPAGE WHERE idEtudiant = OLD.idEtudiant;
END;                                  -- Fin du bloc
$$
DELIMITER ;

INSERT INTO ETUDIANT VALUES (15, 'Edward'); -- Insertion d'un nouvel étudiant
INSERT INTO NOTE VALUES (15,1,8);           -- Crée un rattrapage pour Edward
DELETE FROM ETUDIANT
WHERE
    idEtudiant = 15; -- Déclenche AFTER_DELETE_ETUDIANT qui déclenche AFTER_DELETE_NOTE

SELECT
    *
FROM
    ETUDIANT,
    NOTE,
    RATTRAPAGE
WHERE
    ETUDIANT.idEtudiant = NOTE.idEtudiant
        AND RATTRAPAGE.idEtudiant = ETUDIANT.idEtudiant
;
```

Le premier déclencheur entraîne le déclenchement du second

Un autre exemple de déclencheurs

```sql
DROP TRIGGER IF EXISTS AFTER_INSERT_NOTE;

DELIMITER $$
CREATE TRIGGER AFTER_INSERT_NOTE     --  Création du déclencheur
AFTER INSERT ON NOTE                 -- Après chaque insertion ds note
FOR EACH ROW BEGIN                   -- Pour chaque ligne/Début du bloc
    IF                               -- Début du bloc IF
        NEW.note >= 10               -- Si la note >= 10
    THEN
        IF exists (SELECT idEtudiant, idMatiere FROM RATTRAPAGE
            WHERE idEtudiant = NEW.idEtudiant AND idMatiere = NEW.idMatiere) -- L'étudiant avait un rattrapage dans
        THEN                         -- Le rattrapage est supprimé et la note est insérer
            DELETE FROM RATTRAPAGE WHERE idEtudiant = NEW.idEtudiant AND idMatiere = NEW.idMatiere;
        END IF;
    ELSEIF NEW.note < 10 -- L'etudiant a eu moins de 10 dans la matiere
        AND not exists (SELECT idEtudiant, idMatiere FROM RATTRAPAGE
            WHERE idEtudiant = NEW.idEtudiant AND idMatiere = NEW.idMatiere) -- mais n'avait pas un rattrapage dans
    THEN -- L'étudiant a un rattrapage
        INSERT INTO RATTRAPAGE VALUES (NEW.idEtudiant, NEW.idMatiere);
        ELSE -- L'etudiant avait un rattrapage et a eu moins de 10
            signal sqlstate '45000' set message_text = 'L''etudiant a deja passe un rattrapage dans la matiere.';
    END IF; -- Fin du bloc IF
END;                                 -- Fin du bloc
$$
DELIMITER ;

INSERT INTO NOTE VALUES (1, 4, 5);
```

35:27

tion Output

	Time	A	Response
1	18:19:03	I.	Error Code: 1644. L'etudiant a deja passe un rattrapage dans la matiere.