SQL : Procédures stockées & triggers

Afpa - Stéphane PONTONNIER 2022

Plan

- Procédures stockées
- Triggers (déclencheurs)

Procédures stockées

SQL

Procédures stockées

- Disponibles depuis la version 5 de **MySQL**
- Ensemble d'instructions SQL portant un nom, qu'on peut utiliser pour l'appeler :
  CALL nomProcedure()
- Facilitant certains traitements sur une ou plusieurs tables (possibilité d'effectuer des tests, des boucles, …)
- Minimisant les échanges entre le client et le serveurs de données

Procédures stockées

SQL

Pour créer une procédure stockée

CREATE PROCEDURE nomProcedure (les paramètres)
BEGIN
    – traitements = les instructions SQL
END

Remarques

- Chaque instruction SQL d'une procédure doit se terminer par **;** (délimiteur)
- Chaque requête SQL doit aussi se terminer par **;**
- Il faut changer le délimiteur avant le début de la procédure et le remettre à la fin

Procédures stockées

SQL

Pour créer une procédure stockée

```
DELIMITER |
CREATE PROCEDURE nomProcedure (les paramètres)
BEGIN
        – traitements = les instructions SQL
END |
DELIMITER;
```

Procédures stockées

SQL

Exemple

- Créez une procédure stockée qui augmente le salaire de la personne ayant le plus petit salaire de la table personne
- Le montant à ajouter au salaire est passé en paramètre

Déclaration de la procédure

```
DELIMITER |
CREATE PROCEDURE augmenterSalaireMin (somme int)
BEGIN
        DECLARE id int;
        SELECT num INTO id FROM personne WHERE salaire = (SELECT MIN(salaire) FROM
personne);
        UPDATE personne SET salaire = salaire + somme WHERE num = id;
END |
DELIMITER ;
```

Procédures stockées

SQL

Explication

- **DECLARE** permet de déclarer une variable et **DEFAULT** de l'initialiser
- **INTO** permet d'indiquer le nom de la variable dans laquelle on va placer le contenu du **SELECT**

Remarque

SELECT INTO **permet de sélectionner seulement une ligne. Une erreur sera générée si la requête sélectionne plusieurs lignes. En cas de doute, pensez à ajouter** LIMIT 1

Procédures stockées

SQL

Appel de la procédure (exécution)
CALL augmenterSalaireMin(50);

Pour consulter le code de la procédure augmenterSalaireMin
SHOW CREATE PROCEDURE augmenterSalaireMin;

Pour supprimer la procédure augmenterSalaireMin
DROP PROCEDURE IF EXISTS augmenterSalaireMin;

Remarque

On ne peut pas modifier une procédure avec MySQL. Il faut donc la supprimer et la recréer.

Procédures stockées

SQL

Il est possible d'utiliser une structure de contrôle de type IF … THEN … ELSE

```
DELIMITER |
CREATE PROCEDURE augmenterSalaireMin(somme int)
BEGIN
        DECLARE id INT;
        DECLARE smic INT;
        DECLARE mini INT;
        SET smic = 1200;
        SELECT MIN(salaire) INTO mini FROM personne;
        SELECT num INTO id FROM personne WHERE salaire = mini LIMIT 1;
        IF mini > smic THEN
                UPDATE personne SET salaire = salaire + somme WHERE num = id;
        ELSE
                UPDATE personne SET salaire = smic + somme WHERE num = id;
        END IF;
END |
DELIMITER ;
```

Procédures stockées

SQL

Autre structure de contrôle : CASE … WHEN … THEN … ELSE

```
CASE nomVariable
        WHEN value1 THEN traitement1;
        WHEN value2 THEN traitement2;
        …
        ELSE autreTraitement;
END CASE;
```

Exercice

En utilisant CASE … WHEN … THEN … ELSE, écrire une **procédure stockée** qui permet d'augmenter

- de 200 € le salaire de la personne qui habite à Marseille et qui a un véhicule si son salaire est égal au SMIC
- de 100 € sinon

Procédures stockées

SQL

LA boucle : WHILE … DO

```
WHILE condition(s) DO
        – traitements
END WHILE;
```

Exercice

En utilisant WHILE … DO, écrire une **procédure stockée** qui

- prend deux paramètres : *n* et *somme*
- permet d'ajouter *n* fois *somme* au salaire de la personne qui habite à Marseille et qui a un véhicule

Procédures stockées

SQL

La boucle : REPEAT … UNTIL

```
REPEAT
      – traitements
UNTIL condition(s)
END REPEAT;
```

Exercice

Refaire l'exercice précédent avec REPEAT … UNTIL

Procédures stockées

SQL

On peut aussi définir des libellés et utiliser ITERATE … LEAVE

```
label_loop: boucle -- peut être WHILE, REPEAT ou autre
        -- traitements
        IF conditions THEN
                LEAVE label_loop;
        END IF;
        IF autres_conditions THEN
                ITERATE label_loop;
        END IF;
END LOOP;
```

Explication

- ITERATE permet de relancer une itération en ignorant le reste du code (de la boucle)
- LEAVE permet de quitter la boucle en ignorant le reste du code (de la boucle)

Procédures stockées

SQL

La boucle LOOP … LEAVE

```
label_loop: LOOP
        -- traitements
        IF conditions THEN
                LEAVE label_loop;
        END IF;
END LOOP;
```

Triggers

SQL

Triggers (déclencheurs)

- Un ensemble d'instructions SQL attaché à une table
- Exécuté avant ou après un évènement sur la table de type (insertion, modification ou suppression)

Remarques

- Pour une table donnée, un seul trigger par événement et par moment
- Possibilité d'avoir un un trigger ***before insert*** et un ***after insert***
- Possibilité d'avoir un un trigger ***before update*** et un ***after update***

Triggers

SQL

OLD et NEW

- OLD.colonne désigne l’ancienne valeur de colonne
- NEW.colonne désigne la nouvelle valeur de colonne

Exemple

Avant chaque insertion d’un nouveau tuple dans la table personne, si une valeur pour la colonne ville n’a pas été renseignée, on attribue la valeur Marseille à la colonne ville

Triggers

SQL

Le trigger

```
DELIMITER |
CREATE TRIGGER setVilleMarseille BEFORE INSERT ON personne
FOR EACH ROW
BEGIN
        IF NEW.ville IS NULL THEN
                SET NEW.ville = 'Marseille';
        END IF;
END |
DELIMITER ;
```

Triggers

SQL

Exercice

Ecrire un trigger qui vérifie avant chaque augmentation de salaire si la différence entre l'ancien et le nouveau salaire dépasse 200 euros.
Si c'est bien le cas, on annule l'augmentation.
Sinon, l'augmentation est acceptée.

Triggers

SQL

Solution

```
DELIMITER |
CREATE TRIGGER augmenterSalaire BEFORE UPDATE ON personne
FOR EACH ROW
BEGIN
        IF NEW.salaire >= OLD.salaire + 200 THEN
                SET NEW.salaire = OLD.salaire;
        END IF;
END |
DELIMITER ;
```