01– Triggers et procédures stockées

Slam 3 – SIO 2ème année

Introduction

- Nous allons voir ici:
  - Les procédures stockées
  - Les déclencheurs

Les procédures stockées: Pourquoi?

- Comme en programmation : code SQL souvent répété.
- Nécessité d'exécuter plusieurs requêtes pour une seule tâche.
- Solution :
  - Créer (comme en programmation):
    - Des fonctions
    - Des procédures
- Conséquences:
  - Allégement maxi des traitements sur le client
  - Structuration plus propre du code SQL

Les procédures stockées : Où?

- Elles sont compilées puis stockées sur le serveur de BD.
- Implique une vitesse d'exécution accrue.

Les procédures stockées : Comment?

- Pour créer une procédure stockée :
- CREATE PROCEDURE nomProc (listeParam)
  - BEGIN
    - Traitements
  - END $$
- Attention : les traitements contenant des instructions délimitées par le point-virgule:
  - Obligation de changer de délimiteur avant le CREATE PROCEDURE

Exemple de création

- Dans la base CaisseEnregistreuse, on veut créer une proc. stockée afin de calculer le nombre de produit dans le rayon passé en paramètre, puis qui va mettre à jour la table Rayon:

```
DELIMITER $$
CREATE PROCEDURE compteProduit ( idR int)
BEGIN
    declare nb int ;
    select count(*) into nb from produit where idRayon=idR ;
    update rayon set nbProd=nb where idRayon = idR ;
END$$
DELIMITER ;
```

Appel d'une procédure stockée

- Une fois la procédure créée, on peut l'appeler directement ou bien dans une autre procédure:

```
call compteProduit( 1 ) ;
```

- Va exécuter notre procédure avec le paramètre idRayon égale à 1.

Retrouver sa procédure

- Pour consulter le code de la procédure stockée, on utilise SHOW CREATE PROCEDURE suivi du nom de la procédure à consulter:

```
show create procedure compteProduit ;
```

Modifier une procédure

- On peut avoir besoin de modifier une procédure stockée.
- On utilise alors ALTER PROCEDURE
- Exemple : Modifier les commentaires de la procédure précédemment créée:

```
ALTER PROCEDURE compteProduit
Comment "Compte les produits du rayon en paramètre" $$
```

Suppression d'un procédure

- La procédure ayant un nom dans la BD, nous pouvons la supprimer avec l'instruction DROP PROCEDURE.
- Comme toute suppression, on peut utiliser la condition IF EXISTS afin d'éviter une erreur en cas de non existence:

```
DROP PROCEDURE if exists compteProduit;
```

Les variables dans les procédures

- Comment utiliser des variables dans une procédure stockée:
  - Déclaration :
  - Affectation :
    - Directe :
    - À partir d'une requête :

```
declare a int ;
declare nom varchar(45) ;
```

```
SET a=3 ;
SET nom="Toto" ;
```

```
SELECT count(*) INTO b FROM Produit ;
```

Instructions de contrôle: IF

- IF condition THEN
  - traitements
- END IF;
- Permet d'exécuter un traitement SQL si la condition est vérifiée. Sinon, on peut en exécuter un autre:

```
if b>10 then
    update Produit set pxProd = pxProd+1 where idRayon =r;
else
    update Produit set pxProd = pxProd/2 where idRayon =r;
end if;
```

Instructions de contrôle: CASE

- Selon une valeur, on peut exécuter un traitement:
- syntaxe MYSQL :
  - CASE uneVariable
    - WHEN uneValeur THEN unTraitement
    - WHEN uneValeur THEN unTraitement
    - WHEN uneValeur THEN unTraitement
    - ELSE unTraitement
  - END CASE;

Exemple CASE

```
CASE b
    WHEN 3 THEN update Produit set pxProd = pxProd+1
                where idRayon =r;
    WHEN 12 THEN update Produit set pxProd = pxProd/2
                where idRayon =r;
    ELSE update Produit set pxProd = pxProd/3
                where idRayon =r;
END CASE;
```

Boucler avant de sortir

- La commande LOOP…END LOOP permet de réitérer un bloc de traitement jusqu'à ce que l'instruction LEAVE soit trouvée
- syntaxe MYSQL :
  - nomLabel: LOOP
    - …
    - …
    - LEAVE nomLabel
    - …
  - END LOOP

Exemple LOOP/LEAVE

```
labelLoop: loop
    if b>2 then
        update Produit set pxProd=pxProd*2 ;
        select count(*) into b
            from produit where idRayon=2 ;
    else
        leave labelLoop;
        end if ;
end loop ;
```

Boucle avec REPEAT

- Les commandes à l'intérieur d'une commande REPEAT sont répétées jusqu'à ce que la condition soit vraie:
- syntaxe MYSQL:
  - REPEAT
    - ...
    - ...
  - UNTIL maCondition
  - END REPEAT

La boucle avec WHILE

- Les commandes dans l'instruction WHILE sont répétées tant que la condition est vraie.
- syntaxe MYSQL:
  - WHILE maCondition DO
    - ...
    - ...
  - END WHILE ;

Exemple WHILE:

```
CREATE PROCEDURE dowhile()
BEGIN
  DECLARE v1 INT DEFAULT 5;

  WHILE v1 > 0 DO
    update Produit set pxProd=pxProd+1 where idProd=v1 ;
    SET v1 = v1 - 1;
  END WHILE;
END
```

Remarques Procédures Stockées

- Les procédures stockées sont très puissantes.
- Presque toutes les instructions disponibles.
- Possibilité (évidemment) d'imbriquer les structures de contrôles les unes aux autres.
- La plupart des traitements peuvent alors être exécuter directement sur la BD.
- Le client ne sert plus qu'à l'affichage

Les fonctions

- Comme vous le savez, une fonction est une procédure qui renvoie 1 valeur.
- Syntaxe MYSQL:
  - CREATE FUNCTION nomFonction(listeParam)
  - RETURNS typeDeRetour
    - BEGIN
      - …
      - …
      - RETURN laValeurARetourner ;
    - END $$

Exemple fonction+appel

```
CREATE FUNCTION quelAge(ddn date) returns int
BEGIN
    DECLARE age INT ;
    set age = (year(curdate())-year(ddn) ) ;
    return age ;
END$$
```

```
select quelage('2006-3-3') ;
```

Remarques Fonctions

- Toutes les structures de contrôle vues pour les procédures stockées sont (bien évidemment) valables pour les fonctions.
- On peut utiliser les fonctions dans :
  - Des procédures stockées
  - D'autres fonctions
  - Les SELECT, UPDATE, INSERT…
  - Des triggers
- Beaucoup de fonctions existent déjà sous MYSQL.

Les Triggers (déclencheurs)

- Un déclencheur est un objet de base de données associé à une table, qui s'active lorsqu'un événement particulier survient.
- Il existe 3 types d'évènements déclencheurs:
  - L'insertion
  - La suppression
  - La modification

Syntaxe du CREATE TRIGGER

- CREATE TRIGGER *nomDuTrigger*
- *quand évènement*
- ON *surLaTable*
- FOR EACH ROW *Traitements*

Trigger : Quand?

- C’est le moment d'action du déclencheur.
- Il peut être:
  - BEFORE (avant) : le déclencheur s'active avant
  - AFTER (après): le déclencheur s'active après

Trigger : Evènement

- Indique le type de commande qui active le déclencheur.
- Il peut y avoir:
  - INSERT
  - UPDATE
  - DELETE
- Ex: BEFORE INSERT pour vérifier valeurs avant insert
- 1 seul trigger par table et par évènement et par moment.
- Possibilité de BEFORE UPDATE et BEFORE INSERT
- Possibilité de BEFORE UPDATE et AFTER UPDATE.

Trigger : surLaTable

- Un trigger se « pose » sur une table et seulement sur une table.
- Impossible sur une vue ou une table temporaire.

Trigger : Traitements

- Les traitements peuvent contenir tout ce qui a été vu dans les fonctions et procédures stockées.
- Peuvent contenir des procédures et des fonctions.
- Particularité des triggers:
  - OLD représente l'ancienne occurrence
  - NEW représente la nouvelle occurrence
  - OLD.nomColonne représente l'ancienne valeur du champ nomColonne.
  - NEW.nomColonne représente la nouvelle valeur

Exemple màj montant du passage

- On veut, à chaque insertion dans CONTENIR, màj le montant du passage en caisse:

Màj Montant du passage

```
CREATE TRIGGER majMontant AFTER INSERT ON contenir
FOR EACH ROW
BEGIN
    declare ancienMontant float ;
    declare nouveauMontant float ;
    declare pxAAjouter float ;
    declare pxProduit float ;
    select pxUnit into pxProduit
        from Produit
        where idProduit = NEW.idProduit ;
    select montantTotal into ancienMontant
        from PassageCaisse
        where idPassageCaisse = NEW.idPassageCaisse ;
    set pxAAjouter = NEW.qte * pxProduit ;
    set nouveauMontant = ancienMontant + pxAAjouter ;
    update PassageCaisse set montantTotal = nouveauMontant
    where idPassageCaisse=NEW.idPassageCaisse ;
END$$
```

TP à faire

- Reprendre la bd Tour de France:
  - Comment renseigner les points obtenus?
  - Par rapport au temps effectué sur une étape, renseigner une position

FIN

- Questions