Procédures stockées

Ne pas utiliser du SQL embarqué.
Optimiser la charge Client/Serveur en réduisant le trafic réseau et la charge serveur.
Faciliter la maintenance.
Accroître la sécurité.

Création d’une procédure

```
DELIMITER $$

CREATE PROCEDURE [bd.]nomDeLaProcedure ([DIRECTION argument1 TYPE [, DIRECTION argument2 TYPE]])
BEGIN
  Instruction SQL ou autre;
  Instruction SQL ou autre;
END $$

DELIMITER;
```

Les directions sont IN, OUT, INOUT.

IN : en entrée (facultatif).

OUT : en sortie.

INOUT : en entrée/sortie.

Les types sont ceux de MYSQL (INT, CHAR(n), VARCHAR(n), ...) ou STRING, INTEGER, REAL.

Suppression d’une procédure

```
DROP PROCEDURE IF EXISTS
nomDeLaProcedure;
```

PS sans paramètres

```
DELIMITER $$

DROP PROCEDURE IF EXISTS hotelsSelect $$

CREATE PROCEDURE hotelsSelect()
BEGIN
  SELECT * FROM hotels;
END $$

DELIMITER;



CALL hotelsSelect ;
```

PS avec paramètres IN

```
DELIMITER $$

DROP PROCEDURE IF EXISTS hotelInsert $$

CREATE PROCEDURE hotelnsert (IN asNom CHAR(5),
IN asEtoile INT(6))
BEGIN
  INSERT INTO hotels(libelle, etoile) VALUES(asNom, asEtoile);
END $$

DELIMITER;
```

Exemple de procédure d'Insertion d'enregistrement

```
DELIMITER $$

DROP PROCEDURE IF EXISTS hotelDelete $$
CREATE PROCEDURE hotelDelete(IN asId
INT(6))
BEGIN
   DELETE FROM hotels WHERE idHotel = asId;
END $$

DELIMITER;
```

Exemple de procédure de suppression d'enregistrement

Exécution d'une procédure stockée

```
CALL hotelInsert('le Magnifique',3);
```

```
CALL hotelDelete(25);
```

```
CREATE TABLE pays (
   id_pays char(3) NOT NULL default '',
   nom_pays varchar(45) NOT NULL default '',
   PRIMARY KEY (id_pays)) ;

INSERT INTO pays (id_pays, nom_pays)
VALUES ('033', 'France'),
('035', 'Angleterre'),
('039', 'Italie'),('057', 'Arabie saoudite');
```

Créez une procédure stockée en insertion (angleterre), suppression (arabie saoudite) et mise à jour (irlande à la place d’angleterre).

PS avec OUT

>Pour déclarer une variable :
DECLARE nom_de_variable type DEFAULT valeur;

>Pour affecter une valeur à une variable :
SET nom_de_variable = expression;

>Pour affecter une valeur à une variable dans un ordre SELECT : SELECT ... INTO variable FROM ...;

```
DELIMITER $$

DROP PROCEDURE IF EXISTS hotelNb $$

CREATE PROCEDURE hotelNb(OUT tot INT)
BEGIN
   SELECT COUNT(*) INTO tot FROM hotels;
END $$

DELIMITER;
```

Execution

mysql>CALL hotelNb(@Resultat);
mysql>SELECT @Resultat;

Fonctions stockées

Une fonction stockée renvoie un seul résultat.

Les arguments des fonctions stockées ne peuvent être que de direction IN. De ce fait cette clause de direction ne doit pas être précisée.
Pas de COMMIT ni de ROLLBACK dans les fonctions.

CREATE [AGGREGATE] **FUNCTION** nom_de_fonction([argument1 type [, ...]])
RETURNS type
BEGIN
  DECLARE variable Type **DEFAULT** valeur_par_defaut;
  SET variable = expression;
  RETURN variable | expression;
END …

Le type des paramètres ou retourné par une fonction peut être STRING, INTEGER ou REAL ou un type SQL (CHAR(5), ...).
Une fonction AGGREGATE se comporte comme une fonction agrégat native (SUM, COUNT,

```
DELIMITER $$

DROP FUNCTION IF EXISTS addition $$

CREATE FUNCTION addition(aiX INT , aiY INT)
RETURNS INT
BEGIN
   DECLARE nTotal INT DEFAULT 0;
   SET n_total = aiX + aiY;
   RETURN nTotal;
END $$

DELIMITER;
```

Execution : select addition(3,4) ;

```
DELIMITER $$

DROP FUNCTION IF EXISTS TTC $$

CREATE FUNCTION TTC(aiPrixHT DOUBLE)
RETURNS DOUBLE
[DETERMINISTIC]
BEGIN
   RETURN aiPrixHT * 1.2;
END $$

DELIMITER;
```

Execution :

```
SELECT prix "Prix HT", TTC(prix) "Prix TTC"
    FROM produits;
```

```
DELIMITER $$

DROP FUNCTION IF EXISTS hotelNbCh $$

CREATE FUNCTION hotelNbCh() RETURNS INT
BEGIN
   DECLARE nbch INT DEFAULT 0;
   SELECT COUNT(*) INTO nbch FROM
chambres;
   RETURN nbch;
END $$

DELIMITER;
```

Exécution : SELECT hotelNbCh();

```
CREATE TABLE villes (
   cp char(5) NOT NULL,
   nom_ville varchar(50) NOT NULL,
   id_pays char(3) NOT NULL default '',
   PRIMARY KEY (cp),
   KEY FK_villes_id_pays (id_pays));


INSERT INTO villes (cp, nom_ville, id_pays) VALUES
('14000', 'CAEN', '033'),
('75011', 'PARIS 11', '033'),
('75012', 'PARIS 12', '033'),
('75019', 'PARIS 19', '033'),
('94100', 'ST MANDÉ', '033'),
('34000', 'MONTPELLIER', '033'),
('35000', 'RENNES', '033'),
('35400', 'ST MALO', '033');
```

A résoudre via une fonction :
Compter le nombre de villes de la table Villes.

Récupérer le nom d'une ville en fonction de son CP.

Récupérer le nom du pays à partir de la ville (prendre la table Pays et la table Villes).

Les instructions de contrôles

Le IF, fonctionne comme pour les triggers :

```
IF condition THEN
    Action;
ELSE
    Action;
END IF;
```

```
DELIMITER $$

DROP FUNCTION IF EXISTS villesUneAvecIf $$

CREATE FUNCTION villesUneAvecIf(asCp CHAR(5)) RETURNS VARCHAR(50)
BEGIN
  DECLARE nbVilles INT(1);
  DECLARE lsNomVille VARCHAR(50);

  SELECT COUNT(*) INTO nbVilles FROM villes WHERE cp = asCp;
  IF nbVilles = 1 THEN
    SELECT nom_ville INTO lsNomVille FROM villes WHERE cp = asCp;
  ELSE
    SET lsNomVille = 'Pas de ville pour ce CP';
  END IF;
  RETURN lsNomVille;
END $$

DELIMITER;
```

La même fonction que précédemment qui renvoie le nom de la ville en fonction du CP mais avec un message si le CP n'existe pas.

Exemple : SELECT villesUneAvecIf('75011')
"Pas de ville pour ce CP";

CASE variable
WHEN valeur THEN action;
[WHEN valeur THEN action;]
[ELSE action;]
END CASE;

Voir les superbes tutos de Mister WP

Et faire la même fonction que précédemment qui renvoie le nom de la ville en fonction du CP mais avec un message si le CP n'existe pas.

La boucle WHILE

Réalise une boucle de type TANT QUE. Tant que la condition est à Vrai on boucle.

```
WHILE condition DO
      Instructions;
   END WHILE;
```

```
DELIMITER $$

DROP FUNCTION IF EXISTS fctWhile $$

CREATE FUNCTION fctWhile(aiN INT) RETURNS INT
BEGIN
   DECLARE liC INT;
   DECLARE liFactorielle INT;

   SET liC = 2;
   SET liFactorielle = 1;

   WHILE liC <= aiN DO
     SET liFactorielle = liFactorielle * liC;
     SET liC = liC + 1;
   END WHILE;

   RETURN liFactorielle;
END $$

DELIMITER;
```

Test: select fctWhile(3);

La boucle REPEAT

Réalise une boucle de type FAIRE JUSQU'A. On boucle jusqu'à ce que la condition soit à VRAI.

```
REPEAT
    Instructions;
  UNTIL condition END REPEAT;
```

```
DELIMITER $$
DROP FUNCTION IF EXISTS fctRepeat $$

CREATE FUNCTION fctRepeat(aiN INT) RETURNS INT
BEGIN
   DECLARE liC INT;
   DECLARE liFactorielle INT;

   SET liC = 1;
   SET liFactorielle = 1;

   REPEAT
     SET liFactorielle = liFactorielle * li_c;
     SET liC = liC + 1;
   UNTIL liC > aiN END REPEAT;

   RETURN liFactorielle;
END $$

DELIMITER;

Test:       SELECT fctRepeat(3);
```