Introduction aux bases de données
Cours 6 : Variables, procédures stockées

Jean Francis Michon [1]

[1]Université Nationale d'Economie, Kharkov

16 Décembre 2021

Création de variables

Pour créer une **variable** dans MySQL on utilise la commande SET et le nom doit commencer par @:

```
SET @mavariable = 'Bonjour' ;
```

- Vous pouvez lui affecter une valeur chaîne de caractères, ou numérique entière ou flottante, ou NULL. Il y a quelques limites (valeur temporelles, dates, type JSON - voir la documentation MySQL).
- Vous pouvez l'employer ensuite comme paramètre dans une requête.
- **Attention :** Cette variable n'existe que pendant votre session MySQL. Elle disparaît ensuite. Avec une version récente de PhpMyAdmin, vous pouvez exécuter plusieurs requêtes à la suite dans l'onglet SQL. Mais si vous exécutez les requêtes une par une avec la touche *Exécutez*, votre variable disparaît car chaque exécution est une session différente.

Exemple d'utilisation de variables

Si vous tapez les deux lignes :

```
SET @recherche='A%';
SELECT nom FROM clients WHERE nom LIKE @recherche ;
```

dans l'onglet SQL et si vous cliquez sur le bouton *Exécutez* les deux requêtes seront exécutées dans la même session de MySQL. Donc vous obtenez tous les champs **noms** de la table **clients** qui commencent par la lettre A.
Vous pouvez évidemment faires des opérations ou appliquer des fonctions sur vos variables.

Affectations d'une variable par SELECT

On peut initialiser une variable avec un résultat de requête d'une façon plus complexe :

```
SELECT @lenom := nom  FROM ‘clienttous‘ WHERE nom LIKE ’AL%
```

Attention : S'il y a plusieurs réponses à la requête @lenom contiendra la dernière valeur de nom satisfaisant les conditions WHERE.
Voir aussi plus loin les SELECT... INTO...
Voici un autre exemple avec UPDATE.

Affectation de champs par UPDATE et une variable

On affecte le champ **numero** de la table **conducteurs** à la valeur **@entier** et on incrémente pour chacun des enregistrements !

```
SET @entier =5;
UPDATE conducteurs SET numero = @entier := @entier+1
        WHERE true;
```

Cette pratique est **déconseillée** car elle dépasse le cadre d'application de MySQL. Si on veut faire des opérations plus complexes, il vaut mieux utiliser une connexion de mySQL avec un langage de programmation (C, PHP, Python, Java,...).

Variables du système

Le serveur MySQL utilise de nombreuses variables système prédéfinies. Certaines peuvent être modifiées par une instruction SET d'autres non. Pour les afficher faire

`SHOW VARIABLES;// ou SELECT @@var_name;`

Vous verrez alors les nombreuses variables et leur valeurs. Certaines variables ne sont pas modifiable par SET. Elles doivent être modifiées directement sur le moniteur de MySQL et sont validées après un arrêt suivi d'un redémarrage de MySQL.

Procédures stockées

Si vous devez répéter souvent certaines séquences de requêtes vous devez employer les procédures stockées. Créons en une : elle est très simple : pas de paramètre et une seule requête exécutée :

```
DELIMITER | -- On change le délimiteur
CREATE PROCEDURE noms_conducteurs()
    -- pas de paramètre
BEGIN
    SELECT idc, nom  FROM conducteurs;
END|            -- utilise  le  nouveau délimiteur
DELIMITER ;  -- reprendre le délimiteur usuel
```

La procédure sera stockée dans la base de données. Elle en fait partie.
Pour supprimer la procédure utiliser: DROP PROCEDURE noms_conducteurs;

Procédures stockées dans PhpMyAdmin

Dans PhpMyadmin vous devez

- Sélectionner la base de donnée
- Cliquer sur l'onglet "Procédures Stockées". Vous aurez alors des outils pour construire et enregistrer votre procédure.

Dans la suite on n'utilise pas PhpMyAdmin.

Paramètres des procédures stockées

Evidemment on veut pouvoir rentrer des paramètres ou extraire des valeurs. On définit les paramètres entrant avec le mot IN (pas obligatoire), sortant avec OUT, et les deux avec INOUT :

```
DELIMITER | -- On change le délimiteur
CREATE PROCEDURE noms_conducteurs(IN recherche VARCHAR(20))
    -- un paramètre entrant
BEGIN
   SELECT idc, nom  FROM conducteurs
         WHERE nom LIKE recherche;
END|              -- utilisez  le  nouveau délimiteur
DELIMITER ;
```

Pour lancer la procédure stockée en SQL on écrit, par exemple :

```
CALL nom_conducteurs('A%');
```

Affectations de variables par des procédures

On peut utiliser les variables SQL. La syntaxe

```
SELECT attributs  INTO variables FROM ...
```

vous permet de stocker les valeurs retournées par la procédure dans des variables :

```
CREATE PROCEDURE nomville(OUT @var1, OUT @var2)
BEGIN
SELECT nom, ville INTO @var1, @var2 FROM conducteurs WHERE
SELECT(@var1, @var2);
END |
```