

## 23) Introduction aux Procédures stockées en MySQL

### 23.1 Qu'est-ce qu'une procédure stockée ?

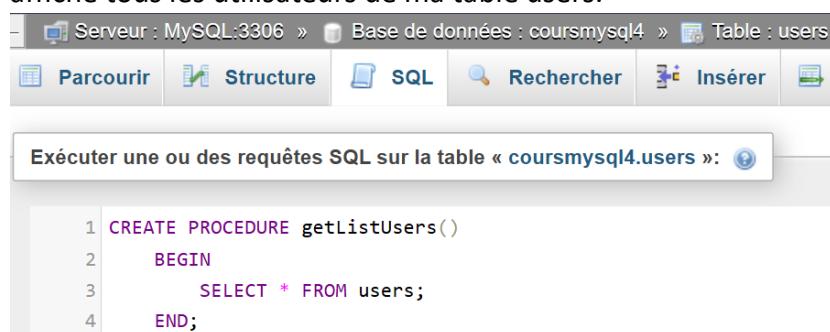
Une **procédure stockée** est un ensemble d'instructions SQL qui est sauvegardé dans la base de données et peut être exécuté ultérieurement. Elle permet d'automatiser des traitements et d'améliorer la performance en évitant d'envoyer plusieurs requêtes séparées.

Voici quelques avantages des procédures stockées :

- **Réutilisabilité** : Une fois créée, elle peut être appelée plusieurs fois.
- **Performance** : Moins de trafic entre l'application et le serveur, car tout est exécuté côté base de données.
- **Sécurité** : Peut restreindre l'accès direct aux tables en forçant les utilisateurs à passer par la procédure.

### 23.2 Création d'une procédure

En MySQL, une procédure est créée avec la commande **CREATE PROCEDURE**. Pour des soucis de visibilité, je vais le faire dans ce cours sur PHPMYADMIN pour que vous voyez clairement, mais on le fera sur le terminal lors de la pratique. Je vais créer une procédure qui affiche tous les utilisateurs de ma table users.



The screenshot shows the PHPMyAdmin interface with the following details:

- Serveur : MySQL:3306
- Base de données : coursmysql4
- Table : users
- Toolbar buttons: Parcourir, Structure, SQL, Rechercher, Insérer, Supprimer
- SQL query editor:

```
Exécuter une ou des requêtes SQL sur la table « coursmysql4.users »: 
1 CREATE PROCEDURE getListUsers()
2 BEGIN
3     SELECT * FROM users;
4 END;
```

Testez, que remarquez-vous ?

Vous remarquerez qu'on a un petit problème... On a 2 points virgules « ; ». Celui de la requête et celui qui termine la procédure. Du coup quand il va lire ce code il ne va pas vraiment savoir à quel moment s'arrêter. C'est un peu l'inconvénient des procédures mais il va falloir commencer avant une procédure par faire un délimiteur (avec le mot clé **DELIMITER**), et changer le délimiteur des requêtes. Attention MariaDb avant de créer la procédure, il faut être en admin(sudo) et faire : « mariadb-upgrade ».

On va mettre que le délimiteur sera \$\$, j'aurai pu mettre ce que je veux. Voici du coup une structure de base en 3 étapes :

```
- DELIMITER $$  
- CREATE PROCEDURE Nom_Procedure()  
  
BEGIN  
  
    -- Requêtes SQL ;  
  
END $$  
  
- DELIMITER ;
```

#### Explication :

- **DELIMITER \$\$** : Change temporairement le délimiteur (; devient \$\$) pour éviter les conflits avec les instructions internes.
- **CREATE PROCEDURE Nom\_Procedure()** : Définit la procédure.
- **BEGIN ... END** : Bloc d'instructions SQL.
- **DELIMITER ;** : Remet le délimiteur à ;.

Ce qui maintenant donnera plutôt ceci pour qu'on crée notre procédure de la liste des utilisateurs :

```
mysql> DELIMITER $$  
mysql> CREATE PROCEDURE getListUsers()  
        -> BEGIN  
        -> SELECT * FROM users;  
        -> END $$  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> DELIMITER ;
```

Si vous le faites en une seule ligne il ne vous dira rien. Par contre si vous le faites étape par étape il vous dira query ok.

### 23.3 Voir les procédures

Voilà, nous avons créé une procédure stockée. Maintenant on va voir comment les voir.

Pour voir la procédure qu'on vient de créer, vous pouvez faire :

- **SHOW PROCEDURE STATUS LIKE 'getListUsers';**

The screenshot shows the results of a MySQL query in the SQL editor:

```
SHOW PROCEDURE STATUS LIKE 'getListUsers';
```

Below the query, there are several buttons: Profilage [Éditer en ligne] [Éditer] [Créer le code source PHP] [Actualiser].

Options supplémentaires

Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_set_client	collation_connection	Database Colla
coursmysql4	getListUsers	PROCEDURE	root@localhost	2025-02-11 11:00:38	2025-02-11 11:00:38	DEFINER		utf8mb4	utf8mb4_unicode_ci	utf8mb3_general_ci

On sait aussi faire de manière plus générale, pour voir les procédures qui sont stockées pour ma base de données coursmysql, je peux faire :

- **SHOW PROCEDURE STATUS WHERE DB = 'coursmysql';**

Il nous reste maintenant à voir comment on utilise une procédure qu'on a créée. C'est très simple, c'est avec **CALL** suivi du nom de la procédure :

- **CALL nomProcédure(0 à n paramètres) ;**

Dans notre cas, on fera donc :

The screenshot shows the result of the query:

```
CALL getListUsers();
```

Below the query, there are several buttons: Éditer en ligne [Éditer] [Créer le code source PHP].

Options supplémentaires

id_user	firstname	lastname	gender	date_of_birth	city	weight_kg
1	James	Bond	M	2007-07-07	Londres	87
2	Jack	Bauer	M	2004-12-24	New-York	124
3	Lara	Croft	F	2005-08-01	Washington	55
4	Beyonce	Knowles	F	1981-09-04	Houston	70
5	Black	Blue	X	1978-05-10	Anvers	87
6	Kevin	DeBruyne	M	1991-06-28	Anvers	77
7	Eden	Hazard	M	1988-10-10	Brussels	82
8	Mario	Bros	X	1980-01-30	Brussels	110
9	Alicia	Knowles	F	1985-05-09	New York	67

On voit bien qu'il a montré tous les utilisateurs de notre table users. Ça fonctionne un peu comme les fonctions que vous avez vu dans vos langages de programmations. On peut aussi voir comment on a créé une procédure, avec **SHOW CREATE PROCEDURE nomProcédure** :

The screenshot shows the result of the query:

```
SHOW CREATE PROCEDURE getListUsers;
```

Below the query, there are several buttons: Profilage [Éditer en ligne] [Éditer] [Créer le code source PHP] [Actualiser].

Options supplémentaires

Procedure	sql_mode	Create Procedure	character_set_client	collation_connection	Database Collation
getListUsers		CREATE DEFINER='root' '@'localhost' PROCEDURE `getListUsers`() BEGIN SELECT * FROM users; END	utf8mb4	utf8mb4_unicode_ci	utf8mb3_general_ci

Il faudra cliquer sur option supplémentaire puis texte complet pour tous voir.

### 23.4 Création de procédure stockée avec paramètres

Les procédures peuvent accepter des paramètres d'entrée (**IN**), de sortie (**OUT**) ou les deux (**INOUT**). On va créer une autre procédure stockée qui cette fois ci permet d'afficher les utilisateurs en fonction du genre introduit.

```
1 DELIMITER $$  
2 CREATE PROCEDURE getListUsersByGender(IN genre VARCHAR(5))  
3 BEGIN  
4     SELECT * FROM users WHERE gender = genre;  
5 END $$  
6 DELIMITER ;
```

Ensuite si j'appelle cette procédure, en mettant bien en paramètre ce que je souhaite, il me ressort bien la liste de tous les hommes :

```
CALL getListUsersByGender("M");
```

[ Éditer en ligne ] [ Éditer ] [ Crée le code source PHP ]

Tout afficher | Nombre de lignes : 25 ▾ Filtrer les lignes: Chercher dans cette table

Options supplémentaires

<b>id_user</b>	<b>firstname</b>	<b>lastname</b>	<b>gender</b>	<b>date_of_birth</b>	<b>city</b>	<b>weight_kg</b>
1	James	Bond	M	2007-07-07	Londres	87
2	Jack	Bauer	M	2004-12-24	New-York	124
6	Kevin	DeBruyne	M	1991-06-28	Anvers	77
7	Eden	Hazard	M	1988-10-10	Brussels	82
14	Romelu	Lukaku	M	2000-12-29	Anvers	101
16	Bill	Gates	M	1968-12-18	Chicago	79

On voit bien qu'il nous a affiché tous les hommes, on peut changer le M en paramètre par F et il affichera toutes les femmes de notre table users etc...

On peut aussi utiliser un paramètre OUT pour récupérer un résultat. Imaginons que je veuille récupérer le nombre d'utilisateurs qui viennent de Bruxelles.

```
DELIMITER $$  
CREATE PROCEDURE getNumberBruxellois(OUT nbBruxellois INT)  
BEGIN  
    SELECT COUNT(*) INTO nbBruxellois FROM users WHERE city = "Bruxelles";  
END $$  
DELIMITER ;
```

Ici je mets le résultat de ma requête dans une variable de sortie qui sera nbBruxellois.

Pour maintenant bien appeler la procédure, il faudra le faire en deux étapes :

- **CALL nomProcedure(@nomDeLaVariable) ;**
- **SELECT @nomDeLaVariable ;**

Ce qui donne :

```
1 CALL getNumberBruxellois(@nombreDeBruxellois);  
2 SELECT @nombreDeBruxellois;
```

Le nom de variable que vous mettez avec le @ peut avoir un nom différent de ce que vous avez mis dans la procédure stockée.

### 23.5 Modification et suppression d'une procédure

**Pour modifier une procédure avec MySQL c'est d'abord l'effacer puis la recréer (DROP + CREATE).** En effet MySQL ne permet pas de modifier directement une procédure. Il faut la supprimer puis la recréer.

- **DROP PROCEDURE IF EXISTS nomDeLaProcédure;**

Puis, recréez-la avec les nouvelles modifications.

### 23.6 Conclusion

Les **procédures stockées** en MySQL sont un excellent moyen d'optimiser et de sécuriser les requêtes. Elles permettent :

- **D'automatiser** des traitements SQL.
- **D'améliorer la performance** en réduisant le trafic entre l'application et la base.
- **De sécuriser** l'accès aux données en limitant les droits d'accès directs aux tables.

Je ne vais pas aller plus loin, c'est une introduction aux procédures stockées. Il faut savoir qu'on peut rajouter des conditions (IF, CASE) ou encore des boucles (WHILE, etc.) ou même rajouter des transactions (COMMIT, ROLLBACK). Vous pouvez toujours aller plus loin de votre côté. Je rappel qu'on vous ouvre les portes mais c'est à vous d'aller plus loin. Tous ce qui concerne les procédures stockées poussée, c'est souvent les admins de Base de Données qui utilisent principalement. Nous dans le cadre de votre formation de développeur web, on va plutôt faire tous ce qui est logique du côté du code et non dans la base de données, mais c'est toujours bon de savoir que ça existe.

**(Exos)**

- 1) Créez une procédure stockée sans paramètre qui permet d'afficher tous les prénoms et noms d'utilisateurs qui sont de Bruxelles, Anvers et New-York.
- 2) Créez une procédure stockée sans paramètre qui permet d'afficher les femmes qui ont plus de 60 kilos.
- 3) Créez une procédure stockée sans paramètre qui permet d'afficher les utilisateurs qui ont moins de 30 ans.
- 4) Créez une procédure stockée sans paramètre qui permet d'afficher les utilisateurs qui n'ont pas écrits d'article.
- 5) Créez une procédure stockée qui reçoit en paramètre d'entrée une lettre et qui me sort tous les utilisateurs qui ont cette lettre dans leur nom.
- 6) Créez une procédure stockée qui reçoit en paramètre d'entrée une date de naissance et qui m'affiche toutes les personnes qui sont nées avant cette date.
- 7) Créez une procédure stockée qui reçoit en paramètre d'entrée le nom d'une ville et qui affiche tous les utilisateurs de cette ville.
- 8) Créez une procédure qui reçoit une ville en paramètre d'entrée et qui me ressort en paramètre de sortie, le nombre de personnes qui sont de cette ville.
- 9) Créez-moi une procédure stockée qui ne reçoit rien en paramètre d'entrée par contre qui reçoit un paramètre de sortie. Qui m'affiche le prénom, nom et poids le tout dans la variable de sortie.
- 10) Créez une procédure stockée qui reçoit en paramètre d'entrée une ville et un nombre qui représente un âge. Qui me ressort donc tous les utilisateurs qui sont d'une ville donnée en paramètre et qui ont plus que l'âge qu'on donne en paramètre.
- 11) (BONUS) Créez une procédure qui reçoit un âge en paramètre d'entrée et qui me sort en paramètre de sortie si oui ou non, on a une personne qui a cet âge dans notre table users.