**Plan de formation sur les procédures stockées**

1. **Introduction**

Les procédures stockées sont disponibles depuis la version 5 de MySQL. Elles permettent d'automatiser des actions qui peuvent être très complexes.

Une procédure stockée est en fait une série d'instructions SQL désignée par un nom. Lorsque l'on crée une procédure stockée, on l'enregistre dans la base de données que l'on utilise, au même titre qu'une table, par exemple. Une fois la procédure créée, il est possible d'appeler celle-ci par son nom. Les instructions de la procédure sont alors exécutées.

Contrairement aux requêtes préparées, qui ne sont gardées en mémoire que pour la session courante, les procédures stockées sont, comme leur nom l'indique, stockées de manière durable, et font bien partie intégrante de la base de données dans laquelle elles sont enregistrées.

1. **Création et utilisation d'une procédure**

Voyons tout de suite la syntaxe à utiliser pour créer une procédure :

**CREATE PROCEDURE nom\_procedure ([parametre1 [, parametre2, ...]])**

**Corps de la procédure;**

* 1. **CREATE PROCEDURE**

Sans surprise, il s'agit de la commande à exécuter pour créer une procédure. On fait suivre cette commande du nom que l'on veut donner à la nouvelle procédure.

* 1. **([parametre1 [, parametre2, ...]])**

Après le nom de la procédure viennent des parenthèses. **Celles-ci sont obligatoires !** À l'intérieur de ces parenthèses, on définit les éventuels paramètres de la procédure. Ces paramètres sont des variables qui pourront être utilisées par la procédure.

* 1. **Corps de la procédure**

C'est là que l'on met le **contenu** de la procédure, ce qui va être exécuté lorsqu'on lance la procédure. Cela peut être soit **une seule requête**, soit **un bloc d'instructions**.

1. **Classification d’une procédure stockée**

**On a deux classes de procédure stockée :**

* 1. **Procédure avec une seule requête**

Voici une procédure toute simple, sans paramètres, qui va juste afficher toutes les utilisateurs.

**Exemple**

**CREATE PROCEDURE afficher\_utilisateur\_requete()**

**-- pas de paramètres dans les parenthèses**

**SELECT id, nom, prenom,telephone FROM utilisateur;**

* 1. **Procédure avec un bloc d'instructions**

Pour délimiter un bloc d'instructions (qui peut donc contenir plus d'une instruction), on utilise les mots **BIGIN et END**

* + 1. **Delimiter**

Ce que l'on appelle délimiteur, c'est tout simplement (par défaut), le caractère ;. C'est-à-dire le caractère qui permet de délimiter les instructions.

Or, il est tout à fait possible de définir le délimiteur manuellement, de manière à ce que ; ne signifie plus qu'une instruction se termine. Auquel cas le caractère ; pourra être utilisé à l'intérieur d'une instruction, et donc dans le corps d'une procédure stockée.

**Exemple**

**DELIMITER |**

**CREATE PROCEDURE user\_sexe()**

**BEGIN**

**SELECT \* FROM `user` WHERE 1;**

**SELECT \* FROM sexe;**

**END|**

1. **Utilisation d'une procédure stockée**

Pour appeler une procédure stockée, c'est-à-dire déclencher l'exécution du bloc d'instructions constituant le corps de la procédure, il faut utiliser le mot-clé CALL, suivi du nom de la procédure appelée, puis de parenthèses (avec éventuellement des paramètres).

**Exemple :** CALL user\_sexe();

1. **Les paramètres d'une procédure stockée**

**Maintenant que l'on sait créer une procédure et l'appeler, intéressons-nous aux paramètres.**

* 1. **Sens des paramètres**

Un paramètre peut être de trois sens différents : entrant (IN), sortant (OUT), ou les deux (INOUT).

IN : c'est un paramètre "entrant". C'est-à-dire qu'il s'agit d'un paramètre dont la valeur est fournie à la procédure stockée. Cette valeur sera utilisée pendant la procédure (pour un calcul ou une sélection, par exemple).

OUT : il s'agit d'un paramètre "sortant", dont la valeur sera établie au cours de la procédure et qui pourra ensuite être utilisé en dehors de cette procédure.

INOUT : un tel paramètre sera utilisé pendant la procédure, verra éventuellement sa valeur modifiée par celle-ci, et sera ensuite utilisable en dehors.

* 1. **Syntaxe**

Lorsque l'on crée une procédure avec un ou plusieurs paramètres, chaque paramètre est défini par trois éléments :

Son sens : entrant, sortant, ou les deux. Si aucun sens n'est donné, il s'agira d'un paramètre IN par défaut.

Son nom : indispensable pour le désigner à l'intérieur de la procédure.

Son type : INT, VARCHAR(10)…

1. **Suppression d'une procédure**

Vous commencez à connaître cette commande : pour supprimer une procédure, on utilise DROP (en précisant qu'il s'agit d'une procédure).

DROP PROCEDURE nom\_procedure;

1. **Avantages, inconvénients et usage des procédures stockées**

#### Avantages

Les procédures stockées permettent de **réduire les allers-retours entre le client et le serveur MySQL**. En effet, si l'on englobe en une seule procédure un processus demandant l'exécution de plusieurs requêtes, le client ne communique qu'une seule fois avec le serveur (pour demander l'exécution de la procédure) pour exécuter la totalité du traitement. Cela permet donc un certain **gain en performance**.

Elles permettent également de **sécuriser**une base de données. Par exemple, il est possible de **restreindre les droits des utilisateurs** de façon à ce qu'ils puissent **uniquement exécuter des procédures**. Finis les DELETE  dangereux ou les UPDATE  inconsidérés. Chaque requête exécutée par les utilisateurs est créée et contrôlée par l'administrateur de la base de données par l'intermédiaire des procédures stockées.

Cela permet ensuite de**s'assurer qu'un traitement est toujours exécuté de la même manière**, quelle que soit l'application/le client qui le lance. Il arrive par exemple qu'une même base de données soit exploitée par plusieurs applications, lesquelles peuvent être écrites avec différents langages. Si on laisse chaque application avoir son propre code pour un même traitement, il est possible que des différences apparaissent (distraction, mauvaise communication, erreur ou autre). Par contre, si chaque application appelle la même procédure stockée, ce risque disparaît.

#### Inconvénients

Les procédures stockées**ajoutent évidemment à la charge du serveur de données**. Plus on implémente de logique de traitement directement dans la base de données, moins le serveur est disponible pour son but premier : le stockage de données.

Par ailleurs, certains traitements seront toujours plus simples et plus courts à écrire (et donc à maintenir) s'ils sont développés dans un langage informatique adapté. A fortiori lorsqu'il s'agit de traitements complexes. **La logique qu'il est possible d'implémenter avec MySQL permet de nombreuses choses, mais reste assez basique**.

Enfin, **la syntaxe des procédures stockées diffère beaucoup d'un SGBD à un autre**. Par conséquent, si l'on désire en changer, il faudra procéder à un grand nombre de corrections et d'ajustements.

#### Conclusion

Comme souvent, tout est question d'**équilibre**. Il faut savoir utiliser des procédures quand c'est utile, quand on a une bonne raison de le faire. Il ne sert à rien d'en abuser.  
Pour une base contenant des données ultrasensibles, une bonne gestion des droits des utilisateurs couplée à l'usage de procédures stockées peut se révéler salutaire.  
Pour une base de données destinée à être utilisée par plusieurs applications différentes, on choisira de créer des procédures pour les traitements généraux et/ou pour lesquels la moindre erreur peut poser de gros problèmes.  
Pour un traitement long, impliquant de nombreuses requêtes et une logique simple, on peut sérieusement gagner en performance en le faisant dans une procédure stockée (a fortiori si ce traitement est souvent lancé).

À vous de voir quelles procédures sont utiles pour **votre application et vos besoins**.

* Une procédure stockée est un **ensemble d'instructions** que l'on peut exécuter sur commande.
* Une procédure stockée est un objet de la base de données **stocké de manière durable**, au même titre qu'une table. Elle n'est pas supprimée à la fin de la session comme l'est une requête préparée.
* On peut passer des **paramètres** à une procédure stockée, qui peuvent avoir trois sens : IN(entrant), OUT  (sortant) ou INOUT  (les deux).
* SELECT ... INTO  permet d'assigner des données sélectionnées à des variables ou des paramètres, à condition que le SELECT  ne renvoie qu'une seule ligne, et qu'il y ait autant de valeurs sélectionnées que de variables à assigner.
* Les procédures stockées peuvent permettre de **gagner en performance** en diminuant les allers-retours entre le client et le serveur. Elles peuvent également aider à **sécuriser une base de données** et à s'assurer que les traitements sensibles sont toujours exécutés de la même manière.
* Par contre, elle **ajoute à la charge du serveur** et sa syntaxe n'est **pas toujours portable** d'un SGBD à un autre.