Stored Procedure

(Docker setup)

|  |  |
| --- | --- |
| Elément | Description |
| **Compétences** | Comprendre l’utilité d’une procédure stockée et en implémenter |
| **Objectifs** | Apprendre ce qu’est une procédure stockée, en créer avec différents types de paramètres. |
| **Durée estimée** | 2x 45 min |
| **Répertoire de travail** | Libre |
| **Fichiers sources** |  |
| **A produire** | Répondre aux questions directement dans ce document |
| **Moyens d’aide** |  |

# Informations générales

Le but de cet exercice est de vous apprendre ce qu’est une procédure stockée et comment lui appliquer des permissions.

# Théorie

Les procédures stockées sont des programmes informatiques qui sont stockés dans une base de données (côté serveur) et peuvent être appelés par d'autres programmes ou applications pour effectuer des opérations sur la base de données.

Les procédures stockées peuvent inclure des instructions de contrôle, des requêtes SQL, des paramètres d'entrée et de sortie, des variables locales, des curseurs et des fonctions.

## Définition des paramètres d’une procédure stockée

En MySQL, il est possible de définir les paramètres d'une procédure stockée en utilisant les mots-clés IN, OUT et INOUT. Voici ce que chacun de ces mots-clés signifie :

IN : ce mot-clé est utilisé pour définir un paramètre d'entrée pour la procédure stockée. Les valeurs peuvent être passées à la procédure stockée en utilisant des arguments au moment de l'appel de la procédure. Les paramètres IN ne peuvent pas être modifiés par la procédure stockée.

OUT : ce mot-clé est utilisé pour définir un paramètre de sortie pour la procédure stockée. Les valeurs de ces paramètres sont définies par la procédure stockée et retournées à l'appelant à la fin de l'exécution de la procédure. Les paramètres OUT ne sont pas utilisés pour passer des valeurs à la procédure.

INOUT : ce mot-clé est utilisé pour définir un paramètre qui est à la fois d'entrée et de sortie pour la procédure stockée. Les valeurs peuvent être passées à la procédure stockée en utilisant des arguments au moment de l'appel, et la procédure stockée peut également modifier les valeurs de ces paramètres. Les paramètres INOUT sont utiles pour les scénarios où la procédure stockée doit renvoyer une valeur calculée qui dépend de la valeur d'entrée.

# Première procédure stockée : paramètre OUT

Ouvrez PHPMyAdmin, créez une base de données « db\_test » et allez dans l’onglet SQL de cette base de données. Ajoutez cette procédure stockée :

DELIMITER //

CREATE PROCEDURE sp\_helloWorld(OUT message VARCHAR(50))

BEGIN

SET message = 'Bonjour le monde';

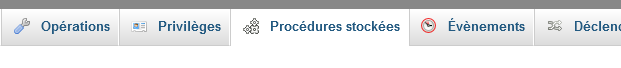
END //

DELIMITER ;

Tester votre procédure en utilisant :  
CALL sp\_helloWorld(@message) ;

SELECT @message ;

Les procédures stockées sont visibles dans l’onglet **Procédures stockées.**



Depuis cet onglet, repérez votre procédure et éditez là. Constatez que le type de sécurité peut être **DEFINER** ou **INVOKER**. Quelle peut être la différence entre ces deux types ?

Graphical user interface, text, application

Description automatically generated

# Question « Délimiter »

Selon vous, qu’est-ce qu’un délimitateur et à quoi cela est utile dans le contexte actuel ?

Cela sert à ouvrir la possibilité de créé une procédure. Le deuxième délimiter sert à la fermé

# Procédure stockée : paramètre IN

Créer une procédure stockée nommée « sp\_helloWorld2 » qui affiche « Bonjour *Nom\_saisi ».* Le nom est passé en paramètre de la procédure. Pour concacténer, vous pouvez utiliser la fonction **CONCACT().**

DELIMITER //

CREATE PROCEDURE sp\_helloWorld2(OUT message VARCHAR(50), IN nom\_saisi)

BEGIN

SET message = CONCAT("Hello ", nom\_saisi);

END //

DELIMITER ;

Tester votre procédure en utilisant :  
CALL sp\_helloWorld2(@message, ‘Bob’) ;

SELECT @message ;

**Note :** @message crée une variable utilisateur et le select permet d’en afficher le contenu.

# Troisième procédure stockée : accès à l’utilisateur

Pour accéder à l’utilisateur actuel, vous pouvez utiliser la fonction USER().

DELIMITER |

CREATE PROCEDURE sp\_helloWorld3(out message)

BEGIN

DECLARE currentUser « à compléter »

SET message = « à compléter »

END |

DELIMITER |

Tester votre procédure en utilisant :

CALL sp\_helloWorld3(@message) ;

SELECT @message ;

DELIMITER //

CREATE PROCEDURE sp\_helloWorld3(out message)

BEGIN

DECLARE currentUser VARCHAR(50);

SET currentUser = USER();

SET message = currentUser;

END

DELIMITER //

# Questions : Création d’une table t\_audit\_access\_absences

Nous souhaitons garder une trace de toutes les lectures faites sur la vue des absences, par quel utilisateur et à quelle date. Veuillez respecter ces conventions :

t\_audit\_access\_absences(idAuditAbsences INT, audAbsDate DATETIME, audAbsUser VARCHAR(50))

La clé primaire est un identifiant artificiel autoincrémenté.

CREATE TABLE t\_audit\_access\_absences(

idAuditAbsences INT AUTO\_INCREMENT,

audAbsDate DATETIME,

audAbsUser VARCHAR(50),

PRIMARY KEY(idAuditAbsences)

);

# Questions : Création d’une procédure stockée pour auditer les accès en lecture

Créer une procédure stockée (sp\_get\_absences) qui :

* Lit toute la vue v\_absentStudent
* Insère dans t\_audit\_absences :
  + La date et l’heure courante
  + Le nom de l’utilisateur connecté

DELIMITER //

CREATE PROCEDURE sp\_get\_absences()

BEGIN

INSERT INTO t\_audit\_access\_absences(t\_audit\_access\_absences.audAbsDate, t\_audit\_access\_absences.audAbsUser) VALUES (CURRENT\_DATE(), USER());

END //

DELIMITER //

# Questions : Définir les permissions

1. Créez un compte utilisateur John dont le mot de passe est aussi John
2. John peut se connecter depuis n’importe quel ordinateur.
3. John peut **uniquement** exécuter la procédure sp\_get\_absences
4. Testez votre sécurité en vous connectant avec le compte **John** et lancez (CALL) votre procédure stockée

CREATE USER 'John'@'%' IDENTIFIED BY 'John';

GRANT EXECUTE ON PROCEDURE sp\_get\_absences TO 'John'@'%';