Les fonctions

procédure stockée ou fonction?

Ils semblent être similaires, mais une fonction a plus de limites.

Je me trompe probablement, mais il semble qu'une procédure stockée peut tout faire et davantage qu'une fonction stockée peut. Pourquoi / quand utiliserais-je une procédure vs une fonction?

La plus générale, la différence entre les procédures et les fonctions qu'ils sont appelés différemment et à des fins différentes:

- Une procédure ne retourne pas de valeur. Au lieu de cela, elle est invoquée avec une instruction d'APPEL pour effectuer une opération telle que la modification d'une table ou de traitement de l'extrait des registres.
- Une fonction est appelée dans une expression et renvoie une valeur unique directement à l'appelant à être utilisé dans l'expression.
- Vous ne pouvez pas appeler une fonction avec une instruction d'APPEL, et vous ne pouvez pas appeler une procédure dans une expression.

La syntaxe pour la routine de création est quelque peu différent pour les procédures et les fonctions:

- Procédure de paramètres peuvent être définis comme entrée seule, sortie seule, ou les deux. Cela signifie qu'une procédure peut transmettre des valeurs à l'appelant en utilisant les paramètres de sortie. Ces valeurs peuvent être consultées dans les instructions qui suivent l'instruction d'APPEL. Les fonctions ont seulement des paramètres d'entrée. En conséquence, bien que les deux procédures et fonctions peuvent avoir des paramètres, des paramètres de la procédure de déclaration diffère de celle des fonctions.
- Les fonctions renvoient une valeur, donc il doit y avoir une clause RETURNS dans une définition de fonction pour indiquer le type de données de la valeur de retour. Aussi, il doit y avoir au moins une instruction de RETOUR dans le corps de la fonction pour renvoyer une valeur à l'appelant. Les RETOURS et le RETOUR n'apparaissent pas dans les définitions de procédure.

1. Pour appeler une procédure stockée, utilisez l' CALL statement. Pour appeler une fonction stockée, reportez-vous à lui dans une expression. La fonction renvoie une valeur lors de l'évaluation de l'expression.

- Une procédure est invoquée à l'aide d'une instruction d'APPEL, et ne peut transmettre des valeurs à l'aide des variables de sortie. Une fonction peut être appelée à partir de l'intérieur d'un énoncé, tout comme toute autre fonction (qui est, en invoquant le nom de la fonction), et peut retourner une valeur scalaire.
- La spécification d'un paramètre IN, OUT ou INOUT est valable que pour une PROCÉDURE. Pour une FONCTION, les paramètres sont toujours considérés comme des paramètresd'entrée.

Si aucun mot-clé est donné devant un nom de paramètre, il est un paramètre par défaut. **Paramètres pour les fonctions stockées ne sont pas précédés par IN, OUT ou INOUT.** Tous les paramètres de la fonction sont traités comme des paramètres.

Pour définir une procédure stockée ou une fonction, utilisez CREATE PROCEDURE ou FONCTION CREATE respectivement:

CREATE PROCEDURE **proc_name ([parameters]) [characteristics]**
routine_body

CREATE FUNCTION **func_name ([parameters])**
RETURNS **data_type // diffrent [characteristics]**
routine_body

Une extension mysql pour une procédure stockée (pas de fonctions) est qu'une procédure peut générer un ensemble de résultats, ou même plusieurs ensembles de résultats, l'appelant processus de la même manière que le résultat d'une instruction SELECT. Cependant, le contenu de ces ensembles de résultats ne peuvent pas être utilisés directement dans l'expression.

Routines Stockées sont associés à une base de données particulière, tout comme des tables ou des vues. Lorsque vous supprimez une base de données, toutes les routines stockées dans la base de données sont également supprimés.

Les procédures stockées et les fonctions ne partagent pas le même espace de noms. Il est possible d'avoir une procédure et une fonction avec le même nom dans une base de données.

Dans les procédures Stockées SQL dynamique peut être utilisé, mais pas dans des fonctions ou des déclencheurs.

SQL préparées (préparation, l'EXÉCUTION, DÉSALLOUER PRÉPARER) peuvent être utilisés dans des procédures stockées, mais pas stockées, des fonctions ou des déclencheurs. Ainsi, les fonctions stockées et les déclencheurs ne peuvent pas utiliser le SQL Dynamique (où vous construisez des déclarations comme des chaînes de caractères, puis les exécuter).

Certaines plus intéressantes différences entre la FONCTION et la PROCÉDURE STOCKÉE:

2. Une procédure stockée est précompilé plan d'exécution où que les fonctions ne sont pas. Fonction Analysées et compilées lors de l'exécution. Stockées les procédures Stockées sous la forme d'un pseudo-code dans la base de données c'est à dire forme compilée.
3. Les fonctions sont normalement utilisés pour les calculs où, comme les procédures sont normalement utilisés pour l'exécution de la logique métier.
4. Les fonctions Ne peuvent pas affecter l'état de la base de données (Déclaration, explicite ou implicite, commit ou rollback sont interdits dans la fonction) Alors que Les procédures stockées Peuvent affecter l'état de la base de données à l'aide de commettre etc. refrence: J. 1. Les Restrictions sur les Routines Stockées et les Déclencheurs
5. Les fonctions ne pouvez pas utiliser la CHASSE d' états alors que les procédures Stockées peuvent le faire.
6. Fonctions stockées ne peuvent pas être récursive, alors que les procédures Stockées peuvent être. Remarque: Récursive procédures stockées sont désactivés par défaut, mais peut être activé sur le serveur par le réglage de la max_sp_recursion_depth système de serveur de variable à une valeur différente de zéro. Voir la Section 5.2.3, "Variables Système", pour plus d'informations.
7. Au sein d'une fonction stockée ou d'un déclencheur, il n'est pas autorisé à modifier une table qui est déjà utilisé (pour la lecture ou l'écriture) par l'instruction qui a appelé la

fonction ou le déclencheur. Bon Exemple: Comment mettre à Jour même table sur la suppression dans MYSQL?

Exmple

Créer une fonction qui retourne le nombre de livres

```
delimiter $$
create function f1()
RETURNS INt
BEGIN
     declare x int default 0;
    select count(*) into x FROM livre;
    return x;
end$$;
```

Pour exécuter la fonction :

```
set @x=f1();
select @x as 'Nombre';
```

Créer une fonction qui retourne le titre d'un livre donné.

```
delimiter $$
create function f2(c int)
returns varchar(80)
BEGIN
     declare t varchar(80) default '';
    select titre into t from livre where code_Livre=c;
    return t;
end$$
```

Pour exécuter cette fonction :

```
set @x=f2(1);
select @x as 'Titre';
```

Exercice

Soit la base de données suivante :

Départements :(DNO, DNOM, DIR, VILLE)
Employés : (ENO, ENOM, PROF, DATEEMB, SAL, COMM, #DNO)

1. Écrire une fonction qui retourne le nombre des départements.

2-Écrire une fonction qui permet de calculer le nombre d'employés dans un département donné. Cette fonction reçoit comme paramètre le code du département et retourne le nombre d'employés dans ce département.

3. Écrire une fonction qui permet d'insérer un enregistrement dans la table employés et qui retourne le numéro d'employé inséré. l'Insertion utilise une séquence.

4. Écrire une fonction qui reçoit comme paramètre le code du département et retrourne la moyenne des salaires des employés dans ce département.

Rmq :

Pour corriger l'erreur :

#1418 - This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled (you *might* want to use the less safe log_bin_trust_function_creators variable)

Taper la commande dans le console MySQL:

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

Fonction sans paramètre

Créer une fonction qui va retourner le nombre des étudiants:

```
delimiter $$
create function f()
returns int
begin
   set @x=0;
   select count(*) into @x
   from etudiant;
   return @x;
end$$
```

Fonction avec paramère

Créer une fonction qui va retourner le nombre des étudiants d'une classe donnée:

```
delimiter $$
create function f2(cc int)
returns int
begin
    set @x=0;
    select count(*) into @x
    from etudiant
    where code_cl =cc;
    return @x;
end$$
```

Créer une fonction qui va retourner l'email d'un étudiant donné:

```
delimiter $$
```

```
create function f3(ce int)
returns varchar(60)
begin
    set @x='';
    select email into @x
    from etudiant
    where code =ce;
    return @x;
end$$
```