# Le procédural

**MySql 5.x.x**

par Dureuil Eric

Date de publication : 21 janvier 2013

Dernière mise à jour : 21 janvier 2013

TOUT PUBLIC

Comme tout utilitaire, un SGBD doit une grande partie de sa puissance au possibilités avancées de programmation. Le procédural fait partie de ces aspects qui démultiplient la puissance et la sécurité des bases de données. Nous aborderons donc ici les différents aspects de l'utilisation du procédural dans MySql, c'est à dire le routines stockées, les déclencheurs et les événements.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

# 1 - Généralités

Commençons par un petit rappel. La programmation procédurale est le fait de pouvoir utiliser les fonctionnalités d'un utilitaire ou d'un langage en ayant la possibilité d'enchaîner différentes actions avec, éventuellement, une logique de contrôle et ce en créant des procédures ou des fonctions que l'on appellera dans un contexte particulier.

Dans les SGBD, le procédural se résume à 3 aspects :

- les routines stockées, appelées directement dans les requêtes (fonctions stockées) ou par l'utilisateur hors requête (procédures stockées).
- Les déclencheurs (« triggers » en anglais) qui servent à exécuter une action automatiquement lors d'actions qui correspondent à des tentatives de modifications sur une table.
- Les événements (« events » en anglais) qui servent à gérer le déclenchement d'action par rapport à une date donnée.

Avant MySQL 5.0.3, le seul code procédural que l'on nous permettait était la mise en place de déclencheurs, apparus à la version 5.0.2... À partir de cette version apparaissent les routines stockées.

On arrive alors à une nouvelle ère permettant la mise en place de traitement de plus en plus lourd côté MySQL et d'éviter parfois des allers-retours entre le langage serveur (php, java, c#, c/c++, etc...) et le SGBD.

Comme c'est déjà possible dans d'autres SGBD, l'utilisation d'autres langages (on parle de php ou c/c++ notamment) pourrait apparaître pour les routines stockées, dans un futur plus ou moins lointain... Cela passerait par l'incorporation d'un compilateur léger.

# 2 - Avantages et inconvénients

## 2-1 - Avantages du procédural

### 2-1-1 - Le contrôle de cohérence des données

Le procédural introduit diverses possibilités au niveau de la gestion des données d'une application selon ce que l'on emploie. La plus flagrante est une que l'on peut centraliser le contrôle de l'insertion, la modification ou la suppression des données au niveau du SGBD et garantir un comportement unique plutôt que de compter sur l'applicatif pouvant mixer différentes sources et donc règles et introduire alors un risque d'incohérence...

Les déclencheurs ont été conçus pour ça principalement.

Les procédures stockées permettent aussi cela, tout en ôtant le coté automatique et transparent que les déclencheurs permettent mais en permettant de n'avoir aucune limitation d'action par rapport à ce que permet la version de SGBD utilisée.

### 2-1-2 - Une meilleure isolation serveur - SGBD

Ce sont les procédures stockées qui vont introduire cette possibilité. L'isolation peut se concevoir à 2 niveaux :

- La sécurité, en gérant drastiquement les actions possibles sur le SGBD par le langage serveur et même aller jusqu'à quasi bloquer les risques liés à l'identification de l'utilisateur SQL coté langage serveur.
- Éviter autan que possible la présence de SQL dans le langage, ce qui qui permet une maintenance plus efficace des différentes parties ou de faire travailler des équipes plus spécialisées en parallèles. Cela permet de reprendre le principe de ce qui est fait dans le web où on sépare le XHTML, le style et/ou le JavaScript...


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

On peut alors constituer des API qui en aideront la mise. Cela permet alors de cacher complètement au langage serveur la structure réelle des données.

### 2-1-3 - Limitation de la consommation de la bande passante

Lors d'un échange entre le serveur et le SGBD, différentes actions vont se produire...

Phase de connexion et d'identification :

- requête DNS (si la connexion est faite sur un nom de domaine et pas une IP).
- Connexion TCP/IP.
- Bufferisation des infos d'identification côté serveur.
- Envoi des infos d'identification
- Bufferisation des infos d'identification côté SGBD.
- Vérification de l'identification.
- Acceptation ou rejet de l'identification.

Phase de traitement d'une requête :

- Bufferisation de la requête côté serveur.
- Envoi de la requête.
- Bufferisation de la requête côté SGBD.
- Analyse et exécution de la requête.
- Bufferisation des résultats côté SGBD.
- Envoi des résultats.
- Bufferisation des résultats côté serveur.

Les échanges, utilisant TCP/IP, impliquent la découpe des données en paquets dont ce protocole garantit la réception... ce processus est dépendant de la qualité du réseau... à savoir que si un paquet n'arrive pas dans les temps ou s'il est considéré comme altéré, il est réexpédié jusqu'à ce qu'il soit considéré comme valide...

Plus on échange de paquets, plus on risque des erreurs et donc des réexpéditions. On consomme alors de la bande passante et donc des ressources et du temps. Parfois l'analyse et l'exécution d'une requête est plus rapide que les échanges qu'elle engendre entre le serveur et le SGBD.

Dans nombre de cas, la taille de la requête est supérieure à celle des données qu'elle va gérer. En limitant les allers-retours entre eux, les procédures stockées peuvent parfois drastiquement réduire la consommation de bande passante.

## 2-2 - Inconvénients du procédural

Le procédural est un langage interprété dans MySQL, donc on ne pourra pas avoir des performances bien meilleures que celle de php...

La compilation des routines se fait lors du premier appel lors d'une session (connexion) et est mise en cache. Cela suit ce qui est fait pour les requêtes... Ce système favorise donc les appels multiples qui se verront donc accélérés à partir du deuxième.

L'implémentation des expressions régulières n'est pensé que dans le but de rechercher la présence d'une expression dans une chaine de caractère, pour l'instant, en aucun cas pour savoir sa position ou découper cette dernière. Donc ce genre d'opérations est préférable à faire ça côté langage serveur.

Aucun type ne permet de stocker une table dans une variable directement.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

Le seul langage permis est le SQL procédural mais la documentation nous prédit en d'autres... un jour (c/c++, php notamment)...

# 3 - Les règles d'écritures communes du procédural

## 3-1 - Les délimiteurs et blocs de code

Comme le procédural permet l'enchaînement des instructions, on va devoir avoir la possibilité de définir un bloc de code qui va encapsuler celles-ci pour former le corps de la ressource procédurale (routine, déclencheur, événement).

On va devoir empêcher l'interpréteur SQL de s'arrêter sur les « ; » terminant les instructions dans le corps de la ressource procédurale que l'on crée ou modifie. Pour cela, on utilise le mot clé « delimiter » qui va changer le délimiteur de fin de ligne courant. On le rétablira une fois qu'on a fini la création ou mise à jour de le (ou des) ressources procédurale souhaitées.

On utilise les mots clés « begin » et « end » pour générer un bloc. Comme dans la plupart des langages, un bloc peut aussi contenir des déclarations de variables locales (dont la porté est limité au bloc où elles sont déclarées mais aussi à tous les sous-blocs de code que celui-ci contiendra).

Les instructions de rupture de code (branchements conditionnel, boucles) ne nécessitent d'utiliser « begin » et « end » pour les blocs d'instructions qu'elles contiennent que si on a besoin de déclarer des variables à l'intérieur d'elles. C'est pratique pour limiter l'usage de la mémoire.

Une déclaration générique de ressource procédurale contenant plusieurs instructions aura cette forme :

```
#syntaxe générique de création d une ressource procédurale
drop type_de_ressource [if exists] nom_ressource;
delimiter $$
create type_de_ressource nom_ressource liste_paramètres_si_besoin
liste_options_si_besoin
begin #corps
instrution_1;
...
instruction_n;
end$$
delimiter ;
```

On pourra préciser une étiquette juste avant « begin » avec une expression, commençant par une lettre et ne contenant que des caractères alphanumériques, terminée par « : ». On mettra alors, éventuellement, cette étiquette derrière le « end » associé.

Une variante de déclaration de ressource procédurale montrant un label, un sous-bloc de code et la porté des variables locales en fonction de leur position :

```
#syntaxe générique de création d une ressource procédurale avec des sous-blocs et label
drop type_de_ressource nom_ressource;
delimiter $$
create type_de_ressource nom_ressource liste_paramètres_si_besoin
liste_options_si_besoin
begin #corps
liste_declarations; #déclarations visibles dans tout le corps
instrution_1;
...
instruction_n;
label:begin #sous bloc de code nommé "label"
liste_declarations; #déclarations visibles juste dans le bloc nommé "label"
instrution_1;
...
instruction_n;
end label;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

```
end$$
delimiter ;
```

On se référera à chaque type de ressource procédurale pour voir des exemples plus concrets. Une indentation intelligente sera d'une grande aide pour une lecture aisée du code, que se soit au niveau des séquences d'instructions dans les blocs ou des requêtes qu'elles peuvent contenir.

## 3-2 - Les transactions et blocs de code dans le procédural

Le mot clé « begin » sert aussi à ouvrir une transaction mais hors d'une ressource procédurale. Dedans il sera toujours considéré comme le début d'un bloc ou sous-bloc de code.

On utilisera toujours « start transaction » pour démarrer la transaction. Si un « begin » est rencontré à l'intérieur, il ne terminera pas celle-ci et n'en débutera pas une nouvelle.

Il faut, en effet, se rappeler que les transactions imbriquées ne sont pas supportées dans MySql.

## 3-3 - Les variables

### 3-3-1 - Les différents types de variables

#### 3-3-1-1 - Les variables définissant les réglages de MySql

Ces variables servent à modifier ou à accéder en lecture seule certains comportements de MySQL :

- au niveau du serveur en faisant précédé le nom de la variable par « @@global. ».
- au niveau de la session de l'utilisateur en faisant précédé le nom de la variable par « @@session. » ou « @@ ».

Leur liste, valeurs possibles et signification sont définies dans la documentation MySql. Certaines de ces variables ne sont peuvent n'avoir un porté qu'au niveau serveur ou session.

On peut citer à titre d'exemple une variable qui définit le nombre maximum de niveaux de récursion pour les procédures stockées et est modifiable :

- « @@global.max_sp_recursion_depth ».
- « @@session.max_sp_recursion_depth ».
- « @@max_sp_recursion_depth ».

Si on fait référence à une variable de réglage inexistante, une exception est déclenchée.

On peut les utiliser hors ou dans le corps d'une ressource procédurale.

#### 3-3-1-2 - Les variables utilisateur

##### 3-3-1-2-1 - Les variables globales de la session

Elles sont créées par l'utilisateur et ont une porté globale au niveau de la session. Elles ne sont pas typées donc on peut affecter des valeurs ayant différents types dans la même variable globale à plusieurs endroits du code... même si c'est plutôt déconseillé car on risque de ne plus savoir ce qu'on manipule.

Elles n'ont pas besoin d'être déclarées et leur nom commence toujours par « @ ».


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

On peut les utiliser hors ou dans le corps d'une ressource procédurale.

#### 3-3-1-2-2 - Les variables locales

Elles ne peuvent exister que dans un bloc de code dans le corps d'une ressource procédurale et doivent être déclarées, leur porté étant celle du bloc où on les a créées. On utilise le mot clé « declare » pour le faire. Sa syntaxe permet de déclarer plusieurs variables du même type et, d'éventuellement, préciser la valeur par défaut qu'on leur donne grâce au mot clé « default ».

les déclarations de variables locales doivent toujours être faites au tout début du bloc de code où on veut les créer, sous peine de déclencher une erreur de compilation...

exemple de syntaxe :

```
#code générique pour declare
declare nom_variable_1,...,nomvariable_n nom_type;

declare nom_variable_1,...,nomvariable_n nom_type default valeur;
```

Des exemples appliqués :

```
declare a,b,c int;

declare e,f,g,h varchar(255) default 'ok';
```

Les types autorisés pour une variable locale sont les même que ceux des colonnes sauf

- le type « set ».
- le type « enum ».

Il faudra tenir compte de cette limitation, ce sera la chaîne de caractères correspondant à un des éléments de ces types que l'on stockera dans une variable...

Si on tente d'utiliser une variable non déclarée, on obtient l'erreur :

« ERROR 1327 (42000): Undeclared variable: %s » avec « %s » qui est le nom de la variable qu'on a tenté d'utiliser.

Si on déclare une variable après un curseur ou un gestionnaire d'erreur dans le bloc de code courant, on a l'erreur :

« ERROR 1337 (42000): Variable or condition declaration after cursor or handler declaration ».

## 3-3-2 - Manipulation des variables

### 3-3-2-1 - L'instruction « Set »

C'est la méthode la plus simple pour manipuler une ou plusieurs variables à la fois. L'ordre d'exécution est séquentiel, de gauche à droite. On utilise « , » pour séparer chacune d'elles.

Par exemple :

```
set nom_variable=expression;

set nom_variable_1=expression_1,nom_variable_2=expression_2,...,nom_variable_n=expression_n;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

On peut mixer les types de variables (locales ou globales) comme on veut dans les expressions.

Par exemple :

```
set @a=2;

set @c='abc',b=5,@d=concat(@a,@c);
```

à noter qu'on n'utilise pas « := » pour l'affectation dans « set » met « = ».

### 3-3-2-2 - L'instruction « select … into »

On utilisera cette forme particulière de « select » principalement pour faire des traitements SQL et les stocker dans une ou plusieurs variables au lieu de les afficher (utilisation classique de « select »).

on peut donc faire tous les traitements que l'on veut avec 2 contraintes :

- le nombre d'expressions retournées par le « select » doit être le même que le nombre de variables derrière le « into » et leur types compatibles.
- le nombre de ligne retourné par « select » ne peut être que 0 ou 1.

On utiliser « select » de manière équivalente à « set » :

```
select expression into nom_variable;
#équivaut à:
set nom_variable=expression;
```

On peut aussi faire des affectations à partir d'un « select » sur différentes tables :

```
select expression_1,expression_2,...,expression_n
into nom_variable_1,nom_variable_2,...,nom_variable_n
from nom_table
liste_jointures
where liste_conditions
options_groupage
options_tri
options_limite_affichage;
```

Par exemple :

```
select concat('abc:',55.0) into @c
#équivaut à:
set @c=concat('abc:',55.0);

select c1,c2,d1,d2
into @c,@d,@e,@f
from tab1 t1
left join tab2 t2 on t2.id1=t1.id
where t1.id=3;
```

On utilisera souvent un « limit 1 » à la fin d'une requête risquant de renvoyer plus d'un résultat ou en cas de doute...


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

# 3-4 - Les instructions de rupture de code

## 3-4-1 - Les branchements conditionnels

### 3-4-1-1 - Le « if » et ses variantes

Le cas le plus simple utilise seulement les mots clés « if », « then » et « end if ». La condition à vérifier se trouvera entre « if » et « then » et la (ou les) instruction(s) à exécuter entre le « then » et le « end if ». Le principe est que si la condition est évaluée à « true » les instructions sont alors exécutées.

La syntaxe correspondante est :

```
if expression then
liste_instructions_si_expression_vraie
end if;
```

On peut aussi indiquer un ensemble d'instructions à exécuter si la condition vaut « false » grâce au mot clé « else ».

La syntaxe devient alors :

```
if expression then
liste_instructions_si_expression_vraie
else
liste_instructions_si_expression_fausse
end if;
```

À noter qu'on peut avoir des « if » imbriqués. On les utilise donc comme dans tous les langages.

Par exemple :

```
if expression then
if expression1 then
liste_instructions_si_expression1_vraie
else
liste_instructions_si_expression1_fausse
end if;
else
liste_instructions_si_expression_fausse
end if;
```

En fait, on peut imbriquer toutes les instructions de rupture de code comme on veut (excepté « iterate »)...

De même, on peut utiliser le mot clé « elseif » pour créer des enchaînements de tests mutuellement exclusifs...

par exemple :

```
if expression then
liste_instructions_si_expression_vraie
elseif expression1 then
liste_instructions_si_expression1_vraie
else
liste_instructions_si_expression_et_expression1_fausses
end if;
```

C'est donc l'instruction de rupture de code de base mais, sur des enchaînements de tests plus complexes (équivalent à un « switch » par exemple), d'autres instructions seront plus adaptées.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

## 3-4-1-2 - Le branchement conditionnel multiple « case »

Dans sa forme la plus souvent utilisée, c'est l'équivalent SQL de « switch » en c/c++, php, etc... On utilise alors le mot clé « case » suivit de l'expression que l'on veut comparer. Ensuite chaque mot clé « when » désigne une valeur ou expression de comparaison et le « then » associé la listes des instructions à exécuter si la comparaison est positive.

On a alors la syntaxe :

```
case expression
when valeur1 then
liste_instructions_si_expression_est_valeur1;
when valeur2 then
liste_instructions_si_expression_est_valeur2;
end case;
```

Dans son autre forme on ne précise pas d'expression derrière « case », chaque expression derrière les « when » seront testée et si l'une d'elle est vraie la liste d'instructions du « then » correspondant sera exécutée.

Cette variante a pour syntaxe :

```
case
when expression1 then
liste_instructions_si_expression1_vraie;
when expression2 then
liste_instructions_si_expression2_vraie;
end case;
```

Dans tous les cas, pour chaque cas à tester, une liste d'instructions vide n'est pas permise. On utilisera alors un bloc « begin »...« end ; » vide.

Un mot clé « else » peut définir une liste d'instructions à exécuter si aucune comparaison n'a marché. À noter qu'il vaut mieux toujours en mettre un car si on tombe dans ce cas alors on a une erreur :

« ERROR 1339 (20000): Case not found for CASE statement ».

Pour parer à ça il suffit de mettre un bloc « begin »...« end » vide dans le « else ».

Par exemple :

```
case expression
when valeur1 then
liste_instructions_si_expression_est_valeur1;
when valeur2 then
liste_instructions_si_expression_est_valeur2;
else
liste_instructions_si_expression_different_de_valeur1_et_valeur2_fausses;
end case;
```

Ou encore :

```
case
when expression1 then
liste_instructions_si_expression1_vraie;
when expression2 then
liste_instructions_si_expression2_vraie;
else
liste_instructions_si_expression1_et_expression2_fausses
end case;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

Cette instruction est plus optimisée que des enchaînements multiple de « if » et « elseif » pour avoir un ensemble de tests mutuellement exclusifs.

On peut imbriquer toutes les instructions de rupture de code comme on veut (excepté « iterate »)...

#### 3-4-1-3 - L'instruction « leave »

Le mot clé « leave » permet un branchement inconditionnel permettant de quitter :

- soit une boucle
- soit un bloc de code « begin »...« end ».

On peut éventuellement préciser une étiquette qui désigne la boucle ou le segment de code concerné.

Si le bloc de code concerné est le plus extérieur dans la ressource procédurale alors son exécution s'arrêtera.

### 3-4-2 - Les instructions pour les boucles

#### 3-4-2-1 - L'instruction « iterate »

Le mot clé « iterate » permet un branchement inconditionnel sur le début d'une boucle, sans en attendre la fin. On peut éventuellement le faire suivre par une étiquette si on on veut préciser la boucle concernée parmi plusieurs imbriquées. Il ne peut donc être utilisé que dans une boucle « while », « repeat » ou « loop ».

Ce type de branchement dans une boucle est dangereux. On risque de faire une boucle sans fin et de planter le thread correspondant en déclenchant un dépassement de temps d'exécution du script dans le langage serveur appelant.

Il faut donc tout particulièrement faire en sorte qu'à un moment ou un autre on sorte de la boucle.

#### 3-4-2-2 - La boucle « while »

Cette boucle correspond à un « tant que »...« faire » en algorithmique. On utilise le mot clé « while » suivit d'une expression qui correspond à la condition, qui tant qu'elle est vraie permet de rester dans la boucle. On utilise ensuite le mot clé « do » pour introduire la séquence de code à exécuter dans la boucle, dont la fin est délimitée par « end while ».

Exemple de syntaxe :

```
while expression do
liste_instructions
end while;
```

Par conception de cette boucle, si l'expression qui définit la condition pour rester en elle équivaut à « false » avant le premier passage dans la boucle, alors l'exécution ne passera jamais à l'intérieur de cette dernière.

On pourra éventuellement utiliser un bloc « begin »...« end » pour encadrer les instructions dans la boucle si on veut déclarer des variables dedans.

On pourra préciser un label en utilisant la même convention que pour les blocs « begin »...« end ». La syntaxe est alors :

```
boucle:while expression do
liste_instructions
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

```
end while boucle;
```

On peut imbriquer toutes les instructions de rupture de code comme on veut (excepté « iterate »)...

### 3-4-2-3 - La boucle « repeat »

Cette boucle correspond au « répéter »...« jusqu'à » en algorithmique. On utilise le mot clé « repeat » pour introduire la séquence de code à exécuter dans la boucle, dont la fin est délimitée par « until » qui est suivi d'une expression qui correspond à la condition, qui tant qu'elle est vraie permet de rester dans la boucle. La syntaxe se termine en utilisant « end repeat ».

Exemple de syntaxe :

```
repeat
liste_instructions
until expression
end repeat;
```

Dans cette forme de boucle, on effectuera toujours au moins un passage.

On pourra éventuellement utiliser un bloc « begin »...« end » pour encadrer les instructions dans la boucle si on veut déclarer des variables dedans.

On pourra préciser un label en utilisant la même convention que pour les blocs « begin »...« end ». La syntaxe est alors :

```
boucle:repeat
liste_instructions
until expression
end repeat boucle;
```

On peut imbriquer toutes les instructions de rupture de code comme on veut (excepté « iterate »)...

### 3-4-2-4 - La boucle « loop »

Cette instruction sert à générer une boucle sans condition. Pour la quitter, on utilisera l'instruction « leave ».

la syntaxe est :

```
loop
liste_instructions
end loop;
```

On pourra éventuellement utiliser un bloc « begin »...« end » pour encadrer les instructions dans la boucle si on veut déclarer des variables dedans.

On pourra préciser un label en utilisant la même convention que pour les blocs « begin »...« end ». La syntaxe est alors :

```
boucle:loop
liste_instructions
end loop boucle;
```

On peut imbriquer toutes les instructions de rupture de code comme on veut (excepté « iterate »)...


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

Ce type de boucle est dangereuse car on risque de planter le thread correspondant et de déclencher un dépassement de temps d'exécution du script dans le langage serveur appelant.

Il faut donc tout particulièrement faire en sorte qu'une instruction « leave » soit exécutée, au moins si un certain nombre d'itérations de la boucle est atteint, pour éviter ce genre de situation.

Il est donc plus raisonnable de lui préférer une boucle « while » ou « repeat » car elle intègre explicitement une condition de sortie, les rendant plus aisées à déboguer en cas de boucle sans fin.

# 4 - Les routines stockées

## 4-1 - Les différents types

On a 2 types de routines :

- les procédures qui ne retournent pas de valeur mais dont les paramètres sont en entrée, entrée et sortie ou sortie. Elles sont utilisées via un appel à l'instruction « call ».
- les fonctions qui retournent une valeur et n'ont que des paramètres en entrée. Elles peuvent être utilisée dans un « select » ou une expression avec d'affectation sur une variable comme les fonction native ou les UDF. On ne peut donc pas faire d'affichage via un « select » ou un « show » dedans.

On doit créer une routine stockée sur une base de données précise, elle utilisera nativement son contexte en interne sauf si on en change en interne. Le contexte d'exécution est stocké avant et restitué en sortie de la routine.

MySQL stocke le « SQL_MODE » qui est en vigueur au moment de la création de la routine et c'est celui-ci qui sera utilisé lors de son exécution et pas celui du contexte d'appel.

On peut préciser le fait qu'elle contiennent des instructions lisant ou modifiant des informations stockées dans les tables mais c'est n'est pas testé par le MySQL qui le prends comme une information donnée en toute conscience par le développeur. Donc cela peut avoir des conséquences sur la consistance des données ou l'utilisation des logs.

## 4-2 - Prérequis

Dans la base « mysql », qui est crée à l'installation de MySQL 5.0.x, la table « proc » doit être présente (ce qui n'est pas forcément le cas si on fait une mise à jour depuis une vieille version). Elle contient la description de toutes les routines.

On retrouve aussi ces informations, mais pas tout à fait sous la même forme, dans la table « routines » de la base « information_schema ». cette table date de l'époque où seules existaient les UDF.

À noter que la colonne « comment » a changé de type dans la table « mysql.proc » en passant de :

« char(64) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '' » avant MySQL 5.5.

à :

« text CHARACTER SET utf8 COLLATE utf8_bin NOT NULL » depuis MySQL 5.5.

Cela peut entraîner un message d'erreur qui apparaît si on met une sauvegarde des fichiers qui décrivent les bases de données d'une version antérieure :

« ERROR 1548 (HY000): Cannot load from mysql.proc. The table is probably corrupted ».


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

Elle disparaît lors de la rectification du type. Le problème ne se pose pas si vous exporter et importer les base avec l'utilitaire de mise à jour (pas forcément présent ou accessible selon les cas).

Le nombre maximal de récursion est défini par un variable globale :

- « @@global.max_sp_recursion_depth » pour le serveur entier qui sera utilisée à chaque début de session.
- « @@session.max_sp_recursion_depth » ou « @@max_sp_recursion_depth » pour la session.

Par défaut, cette valeur est à 0, le maximum possible est 255. si on atteint la limite définie dans ces variables globales, on aura l'erreur :

« ERROR 1456 (HY000): Recursive limit %d (as set by the max_sp_recursion_depth variable) was exceeded for routine %s » avec « %d » la valeur définie et « %s » le nom de la procédure concernée.

## 4-3 - Sécurité

### 4-3-1 - Les droits de création, modification et suppression

Un utilisateur MySql qui va créer, modifier, supprimer ou exécuter une routine stockée doit respectivement avoir les droits :

- « CREATE ROUTINE » pour créer une routine.
- « ALTER ROUTINE » pour modifier une routine.
- « DROP ROUTINE » pour supprimer une routine.
- « EXECUTE » pour exécuter une procédure stockée.

Dans certains cas, le droit « SUPER » sera nécessaire en plus.

### 4-3-2 - Les droits d'exécution

Les droits pour pouvoir exécuter les éventuelles requêtes SQL qui se trouvent dans une routine se basent sur le choix fait lors de la création ou de la modification de la routine grâce à l'option « sql security ». On peut alors décider que ceux-ci seront les droits de l'utilisateur qui a créé (ou modifié) la routine ou ceux de celui qui l'invoque.

Par exemple : si le créateur à un droit « select » sur une table ou base, que la routine a sa sécurité basé sur celui du créateur mais que l'utilisateur n'a pas le droit « select » alors si des « select » sont effectués dans la routine ils fonctionneront quand même.

Par défaut, c'est le créateur (ou modificateur) qui va être choisi si on ne précise pas. À noter que si le créateur n'est pas celui qui est définit avec « sql security » alors il faudra avoir le droit « SUPER ».

## 4-4 - La gestion d'erreurs et des warnings

### 4-4-1 - État SQL et condition

#### 4-4-1-1 - L'état SQL

Un état SQL (ou « SQL STATE » en anglais) va correspondre à une erreur ou un warning ou un un succès émis par MySQL. C'est un code numérique sur 5 chiffres, sous forme d'une chaîne de caractères, qui va le représenter et un message fournissant une description lui est associé.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

Ils sont introduits par le mot clé « sqlstate », éventuellement suivi du mot clé « value ».

Certains états SQL couvrent ainsi les codes d'erreur numériques qui sont propres à MySQL.

On parlera de classe pour désigner un ensemble d'états SQL de même nature (commençant par les 2 mêmes chiffres), il y en a 5 :

- '00' pour un état correspondant à un succès.
- '01' pour un warning.
- '02' pour « not found », le contenu cherché n'a pas été trouvé.
- '03' et suivant pour une erreur.
- '45' pour des états définis par l'utilisateur.

## 4-4-2 - Déclaration d'une condition

Une condition représente un nom donné à un état SQL ou une erreur MySQL. Certains noms ne peuvent être utilisés pour les déclarer :

- « sqlwarning » qui désigne la classe de codes d'états commençant par '01'.
- « sqlexception » qui désigne la classe de codes d'états commençant par '01', '02' ou '03'.

Pour déclarer une condition, on utilise le mot clé « declare » suivi du nom de la condition. Les mots clés « condition for » introduisent le code d'erreur numérique MySQL ou un état SQL.

La syntaxe est donc :

```
declare nom condition for expression;
```

Utiliser une condition permet de rendre plus lisible le code. Une condition ne peut être déclaré que pour un seul état SQL à la fois.

Si on tente de déclarer le même nom d'une condition plus d'une fois dans la même porté, on a l'erreur :

« ERROR 1332 (42000): Duplicate condition: %s » avec « %s » qui est le nom de la condition qu'on a tenté d'utiliser.

Si on déclare une condition après un curseur ou un gestionnaire d'erreur dans le bloc de code courant, on a l'erreur :

« ERROR 1337 (42000): Variable or condition declaration after cursor or handler declaration ».

## 4-4-3 - Le déclenchement d'une erreur ou d'un warning

### 4-4-3-1 - L'instruction « signal »

Introduite en MySql 5.5.0, elle sert à déclencher un état SQL particulier. On peut donc programmatiquement déclencher des états SQL à la demande...

la syntaxe utilise le mot clé « signal » suivi du code désignant l'état SQL :

- le nom d'une condition déclarée avant (mais seulement celles définies avec « sqlstate » pas celles définies par un numéro d'erreur) sous peine de déclencher une erreur.
- le mot clé « sqlstate » suivi, éventuellement, par le mot clé « value », d'une chaîne de caractères contenant 5 chiffres qui correspond au code d'un état.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

On peut ensuite, si besoin, utiliser le mot clé « set » pour définir le contexte qui sera passé au gestionnaire d'erreurs grâce à un ou des couples expression=valeur... Expression représente le nom d'une information décrivant le contexte et valeur peut être une variable ou un littéral.

On peut utiliser les noms suivants :

- class_origin.
- subclass_origin.
- message_text.
- mysql_errno.
- constraint_catalog.
- constraint_shema.
- constraint_name.
- catalog_name.
- shema_name.
- table_name.
- column_name.
- cursor_name.

Toutes les valeurs associées sont de type « varchar(64) » sauf mysql_errno qui est de type « smallint unsigned ».

La syntaxe général est donc :

```
signal code;

signal code set expression_1=valeur_1,...,expression_n=valeur_n;
```

Sur le principe, « signal » est pensé pour le déclenchement d'erreur programmatique, mais tous les états SQL sont permis, en fait, sauf ceux commençant par '00' car ils indiquent le succès d'une opération.

On aura la propagation de l'état SQL par « signal » quel que soit le réglage du serveur ou de la session contrairement à ceux générés lors de l'exécution de requêtes.

### 4-4-3-2 - L'instruction « resignal »

Cette instruction va servir, à l'intérieur d'un gestionnaire d'erreur, à propager l'erreur, générée par une instruction « signal » ou une erreur ou un warning lors d'une requête, en modifiant éventuellement son code ou tout ou partie de son contexte.

La syntaxe est la même que celle de « signal » hormis que seul « resignal » est obligatoire, tout le reste est optionnel selon ce qu'on veut modifier...

On a donc les 3 syntaxes suivantes qui sont possibles :

```
resignal;

resignal code;

resignal code set expression_1=valeur_1,...,expression_n=valeur_n;
```

Cette instruction est interdite hors d'un gestionnaire d'erreur actif. Par exemple, si « resignal » est dans une procédure stockée qui est appelé dans un bloc de code où un gestionnaire d'erreur est actif, c'est donc valide.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

## 4-4-4 - Les gestionnaires

### 4-4-4-1 - Rôle

Dans les routines, il peut être très utile de contrôler comment le code va réagir à une erreur ou un warning émis lors de son exécution. C'est ce que vont permettre ces gestionnaires. On va pour les définir pour réagir à un signalement particulier ou à une ensemble d'états SQL et déclencher un comportement particulier. C'est l'équivalent d'un « try »...« catch » dans nombre de langage.

### 4-4-4-2 - Déclaration d'un gestionnaire

Les gestionnaires doivent toujours être déclaré juste après la dernière déclaration de variables ou de conditions (s'il y en a) dans le bloc de code et juste avant la première instruction (ils sont donc actif sur tout le code du bloc où ils sont définis). Sinon une erreur est déclenchée.

On utilise le mot clé « declare » suivit du mot clé désignant le type d'action à faire par le gestionnaire :

- « continue » pour un gestionnaire qui va continuer l'exécution du code même si le ou les états voulus l'ont déclenché.
- « exit » pour un gestionnaire qui va arrêter l'exécution dans le bloc de code où il est défini si le ou les états voulus l'ont déclenché.

On utilise ensuite les mots clés « handler for » suivis d'une expression décrivant le ou les états à intercepter :

- le nom d'une condition déclarée avant.
- le code numérique d'erreur MySql.
- le mot clé « sqlstate » suivi, éventuellement, par le mot clé « value », d'une chaîne de caractères contenant 5 chiffres qui correspond au code d'un état.
- « not found » qui désigne la classe d'états dont le code commence par '02'.
- « sqlwarning » qui désigne la classe de codes d'états commençant par '01'.
- « sqlexception » qui désigne la classe de codes d'états commençant par '01', '02' ou '03'.

On met enfin l'instruction ou le bloc de code à exécuter en cas de déclenchement.

La syntaxe est donc :

```
declare type_handler handler for expression instruction_ou_code_a_executer;
```

L'état SQL, tant qu'il n'est pas intercepté par un gestionnaire, remonte jusqu'au bloc de code le plus extérieur de la routine.

## 4-4-5 - Limitations

On ne peut pas faire de curseur sur un « show errors » et aucune variable ou « select » sur une table système ne peut le remplacer... On ne pourra donc pas récupérer les informations de contexte pour faire un mécanisme de log personnalisé pour les erreurs ou warnings...


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

# 4-5 - Les curseurs

## 4-5-1 - Rôle

Les curseurs permettent de parcourir le jeu de résultats d'un « select » en permettant un traitement ligne par ligne via une boucle de lecture procédurale. Ils sont :

- en lecture seule.
- assensibles (le curseur peut faire une copie de la table résultat ou pas).
- le déplacement du curseur ne se fait que vers l'avant (du premier vers le dernier résultat renvoyé).

Ils restent donc relativement rudimentaires dans leur implémentation dans MySQL. Leur utilisation est, par contre, indispensable dans l'automatisation de certaines taches comme la copie, la suppression ou la modification de tables ou de bases existantes...

Ils ne sont utilisables que dans les routines stockées.

## 4-5-2 - Limitations

Utilisant une boucle de lecture procédurale les rend relativement lents. Si l'on peut faire une requête équivalente au curseur et à son traitement, il vaudra mieux passer par elle si :

- le jeu de résultats est important.
- l'appel à la routine le contenant est fréquent si la génération et/ou le traitement du curseur ne sont pas relativement courts.

Pour remettre la position du curseur sur le premier résultat, on doit fermer et rouvrir le curseur, ce qui va recréer le jeu de résultat.

Les requêtes avec « show » et « describe » sont interdites dans un curseur, depuis MySQL 5.1.23.

## 4-5-3 - Déclaration

On utilise le mot clé « declare » suivi du nom du curseur. Les mots clé « cursor for » introduisent la requête SQL associée au curseur.

La syntaxe est donc la suivante :

```
declare nom cursor for requete;
```

On peut déclarer plusieurs curseurs dans la même routine stockée et les utiliser de manière imbriquée (principalement).

Si la requête utilisée dans le curseur n'est pas un « select », on aura très certainement l'erreur :

« ERROR 1322 (42000): Cursor statement must be a SELECT ».

À noter que certains « show » (ceux équivalent à un « select ») sont tolérés mais leur support n'est pas garanti et peut disparaître sans préavis au gré des version de MySQL. Il est donc très conseillé d'éviter leur usage et de n'utiliser que des requête avec « select ».

Si la requête utilisée dans le curseur contient « into », on aura l'erreur :


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

« ERROR 1323 (42000): Cursor SELECT must not have INTO ».

Si on tente de créer un deuxième curseur avec le même non dans la même porté de variable, on a l'erreur :

« ERROR 1333 (42000): Duplicate cursor: %s » avec « %s » qui est le nom du curseur qu'on a tenté d'utiliser.

Si on déclare un curseur toujours avant les éventuels gestionnaires d'erreur et après les éventuelles déclaration de variables, suivies des éventuelles déclaration des conditions, sinon on aura les erreurs :

« ERROR 1337 (42000): Variable or condition declaration after cursor or handler declaration ».

ou

« ERROR 1338 (42000): Cursor declaration after handler declaration ».

Les requêtes « show » sont interdites dans les curseurs depuis MySQL 5.1.23, qu'ils soient remplaçables par un « select » équivalent ou non.

Toute requête interdite dans les curseurs déclenche une simple erreur de syntaxe :

« ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '%s' ».

## 4-5-4 - Utilisation

### 4-5-4-1 - Ouverture du curseur

Pour pouvoir utilisé un curseur, on doit l'ouvrir. Ceci exécute la requête définit dans celui-ci, génère le jeu de résultats correspondants et position le pointeur interne sur le premier d'entre eux.

On utilise alors simplement le mot clé « open » suivit du nom du curseur.

La syntaxe est donc :

```
open nom;
```

Si on tente d'utiliser un curseur non déclaré, on obtient l'erreur :

« ERROR 1324 (42000): Undefined CURSOR: %s » avec « %s » qui est le nom du curseur qu'on a tenté d'utiliser.

Si on tente d'ouvrir une seconde fois un curseur déjà ouvert, on obtient l'erreur :

« ERROR 1325 (24000): Cursor is already open ».

Si on tente d'utiliser un curseur sans l'ouvrir, on obtient l'erreur :

« ERROR 1326 (24000): Cursor is not open ».


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

## 4-5-4-2 - Lecture

Pour lire un curseur ouvert, on utilise le mot clé « fetch » suivi du nom du curseur puis du mot clé « into » qui introduit la liste des variables, qui recevront le résultat de la lecture et dont le nombre et le type doivent être cohérent avec ceux des colonnes retournées par le « select » déclaré.

La syntaxe est donc :

```
fetch nom into liste_variables;
```

Attention ! Si la lecture n'a rien récupéré (plus aucune ligne de résultat à lire), l'état des variables recevant les valeurs reste inchangé. Du coup, il vaut mieux tester toujours si quelque chose est bien retourné par la lecture du curseur dans la boucle qui le parcourt et n'autoriser le traitement que si besoin.

Si une ligne de résultat est lue, « fetch » fait avancer le pointeur du curseur d'un pas sinon il déclenche un état SQL « not found ».

Si on tente d'utiliser un curseur non déclaré, on obtient l'erreur :

« ERROR 1324 (42000): Undefined CURSOR: %s ». avec « %s » qui est le nom du curseur qu'on a tenté d'utiliser.

Si on tente d'utiliser un curseur sans l'ouvrir, on obtient l'erreur :

« ERROR 1326 (24000): Cursor is not open ».

Si on n'a pas la correspondance entre le nombre de colonnes retournées par la requête du curseur et celui de la liste de variable derrière le « into » dans « fetch », on a l'erreur :

« ERROR 1328 (HY000): Incorrect number of FETCH variables ».

Si on est avec le « SQL_MODE » qui est définit à strict et si on a oublié de définir un gestionnaire d'erreur pour « not found », on peut voir apparaître l'erreur :

« ERROR 1329 (02000): No data - zero rows fetched, selected, or processed ».

## 4-5-4-3 - Fermeture

Pour libérer les ressources liées au curseur ou pouvoir refaire une boucle de lecture dessus après avoir lu le dernier résultat (s'il en existe), on doit fermer le curseur.

On utilise alors simplement le mot clé « close » suivit du nom du curseur.

La syntaxe est donc :

```
close nom;
```

Si on ferme un curseur inexistant, de même que si on accède à un curseur fermé, une erreur se produira.

Si on tente d'utiliser un curseur non déclaré, on obtient l'erreur :

« ERROR 1324 (42000): Undefined CURSOR: %s ». avec « %s » qui est le nom du curseur qu'on a tenté d'utiliser.

Si on tente d'utiliser un curseur sans l'ouvrir ou après l'avoir fermé, on obtient l'erreur :


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

« ERROR 1326 (24000): Cursor is not open ».

### 4-5-5 - Exemple

En pratique, on va définir un gestionnaire d'erreur pour l'état SQL « not found » et faire un test qui autorise le traitement dans la boucle de lecture seulement s'il y a eu quelque chose à lire.

Cela donne un code comme celui-ci :

```
...
declare suivant bool default true;
declare nom cursor for requete;
declare continue handler for not found set suivant=false;
...
open nom;
while suivant do
fetch nom into liste_variables;
if suivant then
liste_instructions_traitement;
end if;
end while;
close nom;
...
```

Le « if » est important pour éviter le traitement des données si « fetch » n'a rien lu ce qui peut entrainer des conséquences dures à analyser pour comprendre que c'est ça le problème. C'est donc une sécurité nécessaire.

## 4-6 - Les procédures stockées

### 4-6-1 - Principe

Les procédures stockées sont des ressources procédurales dont le but être d'être appelées hors d'un « select ». Elles peuvent être appelées directement ou dans d'autres ressources procédurales (procédures, déclencheurs, événements). Leur paramètres sont en entrée, sortie, entrée et sortie mais elles ne retournent pas de valeur (d'où le nom de procédure).

Les procédures stockées sont toujours hébergées par une base, c'est à dire qu'elles appartiennent à une base comme une table par exemple.

### 4-6-2 - Limitations

L'interprétation de leur code n'est fait qu'à la première demande lors de chaque session (connexion). Le gain en performance n'apparaît donc qu'à partir du deuxième appel de la même session.

On ne peut pas utiliser « load data » dans une procédure stockées depuis MySQL 5.0.7.

### 4-6-3 - Syntaxe

#### 4-6-3-1 - Création

On utilise les mots clé « create procedure », suivi éventuellement des mots clé « if not exists », puis vient le nom de la procédure, éventuellement précédé du nom de la base et d'un point si on exécute la commande hors de la sélection de la base qui hébergera la procédure.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

On place après la liste des paramètres, séparés par une virgule, entre parenthèses. Ces dernières sont obligatoires même si il n'y a aucun paramètres. Chaque paramètre est décrit par son type d'accès (« in » pour en entrée, « out » pour en sortie, « inout » pour en entrée et sortie), son nom suivi de son type. Un paramètre précédé par « out » aura la valeur « null » jusqu'à son éventuelle affectation dans le corps de la procédure. On ne pourra pas utiliser le même nom qu'un paramètre pour la déclaration d'une variable sauf dans un sous bloc de de code sinon on pourrait l'erreur :

« ERROR 1326 (42000): Duplicate variable: %s ».

On place ensuite les éventuelles options :

- « language » suivi d'un nom de langage, précise celui qui est utilisé dans la procédure. Par défaut c'est « sql » et actuellement c'est le seul supporté.
- « deterministic » ou « not deterministic », permet de dire si le résultat de la procédure sera le même ou pas (si on utilise le « not ») pour un jeu de valeurs de paramètres donné... Actuellement, cet attribut n'est pas utilisé.
- « sql security » suivi de « definer » ou « invoker » permet de définir sur les droits de qui se base la sécurité pour l'exécution de la procédure. Par défaut, c'est « definer », c'est à dire son créateur. « invoker » permet de fixer les droit comme étant ceux de celui qui invoque la procédure.
- « comment » suivi d'une chaîne de caractères permet de spécifier un commentaire qui sera visible

Les options sont séparées par un espace, une tabulation ou un saut à la ligne, selon la lisibilité souhaitée.

On place enfin le corps de la procédure qui peut être une seule instruction ou un bloc. De façon générale, il est préférable de toujours utiliser un bloc de code même pour une seule instruction car dés qu'il y aura un bloc d'instructions, on devra utiliser « delimiter » pour empêcher l'interprétation des commandes dans celui-ci. Et cela permet de n'utiliser qu'une paire de « delimiter » pour encadrer la création de plusieurs procédures ainsi que d'avoir de bonne pratiques d'écriture.

À partir de MySql 5.0.20, on peut préciser les droits à utiliser lors de création de la procédure. On utilise alors le mot clé « definer » entre « create » et « procedure » suivi d'un égal puis du nom de l'utilisateur MySql voulu, au format : 'nom_utilisateur'@'host'.

Avec :

- nom_utilisateur, le nom de l'utilisateur dont on veut donner les pouvoirs.
- host, le nom de l'hôte ou son adresse IP.

Si le nom d'utilisateur pour « definer » est différent de celui qui exécute la requête de création alors ce dernier devra avoir le droit « SUPER ».

Un warning sera émis si le compte d'utilisateur spécifié n'existe pas.

La syntaxe est alors :

```
delimiter $$

#la syntaxe la plus simple possible
create procedure essai1()
begin
liste_instructions;
end$$

#la syntaxe la plus complète
create definer='nom'@'host' procedure if exists essai2(liste_parametres)
liste_options
begin
liste_instructions;
end$$
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

```
delimiter ;
```

Dans le corps d'une procédure on ne peut pas utiliser « load data » depuis MySQL 5.0.7 sinon on a l'erreur :

« ERROR 1314 (0A000): LOAD DATA is not allowed in stored procedures ».

À savoir que le « SQL_MODE » est stocké lors de la création et que c'est celui-ci et non celui qui est défini juste avant l'exécution du déclencheur qui sera utilisé.

### 4-6-3-2 - Suppression

On utilise les mots clé « drop procedure », suivi éventuellement des mots clé « if exists », puis vient le nom de la procédure concernée, éventuellement précédé du nom de la base et d'un point si on exécute la commande hors de la sélection de la base qui héberge la procédure.

On a ainsi les syntaxes suivantes :

```
#la syntaxe la plus simple possible
drop procedure essai;

#la syntaxe la plus complète
drop procedure if exists `base`.`essai`;
```

Si on veut ne faire la suppression que si la procédure existe, ce qui évite d'avoir un warning ou une erreur si elle n'existe pas, il vaut mieux toujours employer « if exists ».

### 4-6-3-3 - Utilisation

Pour l'utilisation, on a 2 syntaxes possibles :

- le mot clé « call » suivi du nom seulement si la liste des paramètres est vide (dans la déclaration de la procédure).
- le mot clé « call » suivi du nom avec, entre parenthèses, la liste des expressions passées en paramètres et dont le nombre doit être le même que le nombre de paramètres de la procédures.

Le nom de la procédure est éventuellement précédé du nom de la base et d'un point si on exécute la commande hors de la sélection de la base qui hébergera la procédure.

On a ainsi la syntaxe suivante :

```
#la syntaxe quand il n y a pas de paramètres
call essai;

#la syntaxe quand il y a des paramètres
call essai(liste_parametres);
```

On n'abordera pas ici le « alter procedure » qui correspond à un « drop procedure if exists » suivi du create procedure » avec le même nom, permettant d'être plus générique.

La syntaxe complète de déclaration avec suppression préalable si nécessaire est :

```
delimiter $$

#la syntaxe la plus simple possible
drop procedure if exists essai1$$
create procedure essai1()
begin
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

```
liste_instructions;
end$$

#la syntaxe la plus complète
drop procedure if exists essai2$$
create definer='nom'@'host' procedure if exists essai2(liste_parametres)
liste_options
begin
liste_instructions;
end$$

delimiter ;
```

On notera qu'il peut être bon de mutualiser ces actions si on a plusieurs procédures à créer ou modifier pour n'avoir qu'un seul « delimiter » au début et à la fin du code.

### 4-6-3-4 - Visualisation des informations

On utilise les mots clés « show create procedure » suivi du nom de la procédure, éventuellement précédé du nom de la base et d'un point en cas d'homonymie possible.

On obtient ainsi :

- le nom de la procédure.
- le « SQL_MODE » enregistré lors de la création.
- la chaîne de caractères servant à la création de celle-ci.

La syntaxe est donc la suivante :

```
show create procedure `nom`;
show create procedure `base`.`nom`;
```

On peut obtenir ces informations directement dans la table « proc » de la base « mysql ».

On obtient des informations plus générales avec les mots clés « show procedure status » éventuellement suivi de « like » introduisant une chaîne de caractères définissant un motif de recherche pour préciser le nom de la ou des procédure(s) dont on veut les informations.

Le motif utilise la même syntaxe que celle utilisée dans « like » dans un « select ». Si on utilise la syntaxe sans « like », on obtient les informations pour toutes les procédures.

La syntaxe est alors :

```
show procedure status;
show procedure status like 'motif';
```

On peut avoir toutes ces informations aussi dans la table « routines » de la base « information_schema ».

## 4-7 - Les fonctions stockées

### 4-7-1 - Principe

Ce sont des routines procédurales faites pour être appelées et se comporter comme les fonctions natives de MySQL. Elles retournent un résultat directement contrairement aux procédures. Les fonctions stockées sont toujours hébergées par une base (contrairement aux fonctions définies par l'utilisateur ou aux fonctions natives de MySQL), c'est à dire qu'elles appartiennent à une base comme une table par exemple.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

## 4-7-2 - Limitations

Aucune requête provoquant un affichage direct (comme un « select » ou « show ») ne peut être utiliser dans une fonction.

La syntaxe de création utilise « create function » comme pour les fonctions définies par l'utilisateur (UDF en anglais) permettant l'extension de MySQL grâce à des bibliothèques compilés (des fichiers « .dll » ou « .so » selon le système d'exploitation)... Ce qui peut prêter à confusion.

La valeur retournée doit être l'un des types possibles pour les variables, on ne peut donc pas retourner un jeu de résultats directement. Néanmoins on pourra les encoder sous la forme d'un texte genre CSV ou XML, ce qui permet de contourner cette limitation mais rendrait le traitement en MySql des ces résultat peu efficaces (on utilisera cette technique plus dans un but de restitution finale de ceux-ci).

Le temps d'exécution est largement plus grand qu'une UDF ou une fonction native.

## 4-7-3 - Syntaxe

### 4-7-3-1 - Création

On utilise les mots clé « create function », suivi éventuellement des mots clé « if not exists », puis vient le nom de la fonction, éventuellement précédé du nom de la base et d'un point si on exécute la commande hors de la sélection de la base qui hébergera la fonction.

On place après la liste des paramètres, séparés par une virgule, entre parenthèses. Ces dernières sont obligatoires même si il n'y a aucun paramètres. Chaque paramètre est décrit par son nom suivi de son type, ils sont obligatoirement en entrée.

On doit ensuite utiliser le mot clé « returns » qui introduit le type de la valeur retournée.

On place ensuite les éventuelles options :

- « language » suivi d'un nom de langage, précise celui qui est utilisé dans la procédure. Par défaut c'est « sql » et actuellement c'est le seul supporté.
- « deterministic » ou « not deterministic », permet de dire si le résultat de la procédure sera le même ou pas (si on utilise le « not ») pour un jeu de valeurs de paramètres donné... Actuellement, cet attribut n'est pas utilisé.
- « sql security » suivi de « definer » ou « invoker » permet de définir sur les droits de qui se base la sécurité pour l'exécution de la procédure. Par défaut, c'est « definer », c'est à dire son créateur. « invoker » permet de fixer les droit comme étant ceux de celui qui invoque la procédure.
- « comment » suivi d'une chaîne de caractères permet de spécifier un commentaire qui sera visible.

On place enfin le corps de la fonction qui peut être une seule instruction ou un bloc.

Dés qu'il y aura un bloc d'instructions, on devra utiliser « delimiter » pour empêcher l'interprétation des commandes dans celui-ci.

Le corps de la fonction stockée doit toujours au moins contenir une fois l'instruction « return » qui devra être suivi de « null » ou d'une expression d'un type compatible avec celle définie dans la clause « returns ».

la syntaxe est donc :

```
delimiter $$
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

```
#la syntaxe la plus simple possible
create function essai1()
returns type_retourné
begin
liste_instructions;
end$$

#la syntaxe la plus complète
create definer='nom'@'host' function if exists essai2(liste_parametres)
returns type_retourné
liste_options
begin
liste_instructions;
end$$

delimiter ;
```

À savoir que le « SQL_MODE » est stocké lors de la création et que c'est celui-ci et non celui qui est défini juste avant l'exécution du déclencheur qui sera utilisé.

On notera qu'il peut être bon de mutualiser ces actions si on a plusieurs procédures à créer ou modifier pour n'avoir qu'un seul « delimiter » au début et à la fin du code.

### 4-7-3-2 - Suppression

On utilise les mots clé « drop function », suivi éventuellement des mots clé « if exists », puis vient le nom de la procédure concernée, éventuellement précédé du nom de la base et d'un point si on exécute la commande hors de la sélection de la base qui héberge la fonction.

On a ainsi les syntaxes suivantes :

```
#la syntaxe la plus simple possible
drop function essai;

#la syntaxe la plus complète
drop function if exists `base`.`essai`;
```

Si on veut ne faire la suppression que si la fonction existe, ce qui évite d'avoir un warning ou une erreur si elle n'existe pas, il vaut mieux toujours employer « if exists ».

### 4-7-3-3 - Utilisation

En pratique, on peut créer une fonction stockée avec le même nom qu'une fonction native mais pour l'utiliser, on devra obligatoirement qualifier le nom en le faisant précédé du nom de la base qui héberge la fonction. Ceci n'est pas possible pour une UDF qui aurait le même nom.

Hormis cette éventuelle nécessité de qualification du nom de la fonction, on l'utilise dans un « select » ou avec une instruction « set » comme n'importe quelle autre fonction. On peut utiliser potentiellement une fonction appartenant à une base sur des données d'une autre (si on ne modifie pas pas des tables à l'intérieur de la fonction sous peine de risquer de générer une incohérence).

On peut donc avoir des syntaxes d'appel du genre :

```
set @a=essai();

select essai1(truc) from ma_table where recherche='truc':
```

On n'abordera pas ici le « alter function » qui correspond à un « drop function if exists » suivi du create function » avec le même nom, permettant d'être plus générique.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

La syntaxe complète de déclaration avec suppression préalable si nécessaire est :

```
delimiter $$

#la syntaxe la plus simple possible
drop function if exists essai1$$
create function essai1()
returns type_retourné
begin
liste_instructions;
end$$

#la syntaxe la plus complète
drop function if exists essai2$$
create definer='nom'@'host' function if exists essai2(liste_parametres)
returns type_retourné
liste_options
begin
liste_instructions;
end$$

delimiter ;
```

On notera qu'il peut être bon de mutualiser ces actions si on a plusieurs fonctions à créer ou modifier pour n'avoir qu'un seul « delimiter » au début et à la fin du code.

### 4-7-3-4 - Visualisation des informations

On utilise les mots clés « show create function » suivi du nom de la fonction, éventuellement précédé du nom de la base et d'un point en cas d'homonymie possible.

On obtient ainsi :

- le nom de la fonction.
- le « SQL_MODE » enregistré lors de la création.
- la chaîne de caractères servant à la création de celle-ci.

La syntaxe est donc la suivante :

```
show create function `nom`;
show create function `base`.`nom`;
```

On peut obtenir ces informations directement dans la table « routines » de la base « information_schema ».

On obtient des informations plus générales avec les mots clés « show function status » éventuellement suivi de « like » introduisant une chaîne de caractères définissant un motif pour préciser le nom de la ou des fonction(s) dont on veut les informations.

La syntaxe est alors :

```
show function status;
show function status like 'motif';
```

Le motif utilise la même syntaxe que celle utilisée dans « like » dans un « select ». Si on utilise la syntaxe sans « like », on obtient les informations pour toutes les fonctions.

On peut avoir ces informations aussi dans la table « routines » de la base « information_schema ».


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

# 5 - Le planificateur d'événements

## 5-1 - Principe

C'est un CRON, c'est à dire un gestionnaire d'exécution permettant le déclenchement d'une tâche à partir d'un moment donné et avec une possibilité de répétition à intervalles réguliers. L'idée à été d'éviter de passer par une gestion externe à MySQL (CRON du système d'exploitation par exemple) qui force l'utilisation d'un langage tiers (PHP par exemple ou d'un exécutable qui lancerait l'exécution du code SQL voulu.

Une tâche sera donc un bloc de code SQL procédural appelé à partir d'une date donnée et dont la ré-exécution éventuelle sera paramétrable (planification).

## 5-2 - Prérequis

### 5-2-1 - Version et données spécifiques

Le planificateur est introduit à partir de MySQL 5.1.6. On a alors certains ajouts à MySQL :

- une table « events » apparaît dans la base « information_schema » .
- une variable globale pour la configuration de l'état du planificateur.
- une table « event » apparaît dans la base « mysql ».
- 5 variables globales qui décomptent les actions liées au planificateur depuis son dernier démarrage.

Ces tables contiennent toutes les données relatives aux tâches.

Le parti pris a été de permettre de faire toutes les actions (création, modification, suppression) sur les tâches du planificateur indifféremment :

- soit par des requêtes classiques (« insert », « update », « delete »).
- soit par l'API d'instructions spécialisé (« create event », « alter event », « drop event », « show events », « show create event »).

Ce qui laisse une grande souplesse d'utilisation...

### 5-2-2 - La sécurité

Il faut avoir le droit « SUPER » pour pouvoir configurer dynamiquement le planificateur par une variable de configuration sinon on le fera à l'arrêt ou au lancement de celui-ci selon les cas.

Pour créer, modifier ou supprimer des événements planifiés, il faudra avoir le droit :« EVENT » à partir de MySQL 5.1.12, pour les version antérieures, il faut avoir le droit « SUPER ».

Le « grant » pour l'octroyer ou le « revoke » pour le supprimer doit être fait sur une base ou le serveur seulement (contrairement à d'autre droits qui peuvent toucher aussi les tables) sinon une erreur spécifique sera déclenchée :

« ERROR 1144 (42000): Illegal GRANT/REVOKE command; please consult the manual to see which privileges can be used ».

Les droits d'exécution dans une tâche sont ceux de sont créateur sauf en cas d'utilisation de procédures stockée


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

## 5-2-3 - Les différents modes de fonctionnement du planificateur

### 5-2-3-1 - Comportement actuel du planificateur

Les valeurs possibles de configuration à partir de MySQL 5.1.12 sont :

- 1 ou « on » pour démarrer le planificateur. Cela peut être fait à l'arrêt du serveur via le paramétrage de « event-scheduler=on » dans le fichier de configuration (« my.conf » ou « my.ini » selon le système d'exploitation, dans la section « [mysqld] » par exemple). En utilisant « set @@global.event_scheduler=on ; » (ou une syntaxe équivalente) pour modifier la variable globale. Un thread de type « daemon » apparaît alors en plus du processus du serveur (thread de type « query ») si on utilise « show processlist ; » (ce thread « daemon » n'existe que si le planificateur marche).
- 0 ou « off » pour arrêter le planificateur (même principe de configuration que pour la valeur précédente). C'est la valeur par défaut si rien n'est défini par les différents moyens possibles. Le thread n'apparaît plus dans la liste donnée par « show processlist ; ».
- « disable » pour désactiver le planificateur. Cela ne peut être fait que lorsque le serveur MySQL est à l'arrêt en modifiant le fichier de configuration via le paramétrage de « event-scheduler=disable » ou au lancement du serveur avec le paramètre de ligne de commande « --event-scheduler=disable ». On ne peut plus modifier l'état du planificateur dynamiquement. « disable » n'a pas d'équivalent numérique. Le thread n'apparaît plus dans la liste donnée par « show processlist ; ».

Si on tente d'accéder à event_scheduler comme une variable globale de session (« @@session.event_scheduler » ou « @@event_scheduler »), une erreur spécifique sera déclenchée :

« ERROR 1229 (HY000): Variable 'event_scheduler' is a GLOBAL variable and should be set with SET GLOBAL ».

### 5-2-3-2 - Évolution du comportement par rapport à MySQL 5.1.12

Avant MySQL 5.1.11, la valeur « disable » n'existait et la valeur par défaut est 0 ou « off » et on peut passer dynamiquement de l'un à l'autre.

Dans MySQL 5.1.11, on introduit la valeur 2 qui est alors la valeur par défaut et correspond à l'état « suspendu ». le planificateur ne peut alors passer que de « on » à 2 et inversement dynamiquement. Pour passer à « off » le serveur doit être arrêté.

Suite au bug n°17619 on va passer au comportement actuel dans MySQL 5.1.12.

La règle d'unicité des noms de tâche change aussi :

- avant MySQL 5.1.12, on pouvait avoir plusieurs fois le même nom de tâche sur une même base tant que l'utilisateur les ayant créés était différent.
- après MySQL 5.1.12, un nom de tâche doit être unique pour chaque base.

### 5-2-3-3 - Bug connu de risque de désactivation

À partir de MySQL 5.1.17, un bug (n°26807) fait que si on démarre le serveur avec l'option « --skip-grant-table » cause la désactivation automatique du planificateur quelque soit l'éventuelle valeur dans « event-scheduler ».


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

# 5-3 - Syntaxe

## 5-3-1 - Création d'une tâche

Dés que la tâche est crée, elle est prise en compte par le planificateur et apparaît dans la table « information_schema.events ».

On n'abordera ici que la syntaxe proposer par l'API MySQL, pour son équivalent sous forme de requêtes se référer à la documentation.

On utilise les mots clés « create event » suivi éventuellement des mots clés « if not exists » (pour éviter la création de la tâche si elle existe déjà), puis du nom de la tâche qui doit être unique au niveau de la base de données choisie. On peut préciser le nom d'une base devant le nom de la tâche en les séparant par un point sinon c'est le nom de la base sélectionnée qui sera utilisé par défaut.

On utilise ensuite les mots clés « on schedule », suivi de la définition des options de planification de la tâche. Dans la suite, les expressions définissant une date et une heure peuvent être :

- une expression de type datetime
- une expression utilisant les fonctions de date et heure de MySQL
- une chaîne de caractères qui sera convertible en datetime.

Les possibilités de planification sont donc :

- soit le mot clé « at » introduisant une expression définissant une date et une heure. Cela permet de définir une exécution unique à une date et heure particulière de la tâche.
- soit le mot clé « every » suivi d'une expression définissant un intervalle, c'est à dire une valeur entière suivi de l'unité de temps choisie (celles utilisées pour l'opérateur « interval » dans les fonctions de date et heure de MySQL). On peut éventuellement ensuite utiliser le mot clé « starts » introduisant une expression définissant une date et une heure pour préciser la date et l'heure de la première exécution. De même, pour préciser éventuellement la date et l'heure après quoi on n'exécutera plus la tâche, on peut utiliser le mot clé « ends » introduisant une expression définissant une date et une heure.

On peut ensuite préciser la persistance de la tâche une fois qu'elle a expirée :

- les mots clés « on completion preserve » permettent de ne pas supprimer la tâche une fois qu'elle a expiré.
- les mots clés « on completion not preserve » permettent de supprimer automatiquement la tâche une fois qu'elle a expiré, c'est le comportement par défaut si on ne précise rien.

À partir de MySQL 5.1.18, on peut ensuite préciser les options d'activation de la tâche :

- le mot clé « enable », valeur par défaut si on ne précise rien rend la tâche active, c'est à dire qu'elle est prise en compte par le planificateur.
- le mot clé « disable » qui permet de rendre la tâche inactive lors de sa création, son activation étant fait plus tard, c'est à dire qu'elle n'est pas prise en compte par le planificateur.
- les mots clés « disable on slave » qui permet de rendre la tâche inactive si elle est dupliquée sur le serveur esclave en cas de réplication.

On utilise enfin le mot clé « do » pour introduire l'instruction ou le bloc de code à exécuter.

À partir de MySql 5.1.17, on peut préciser les droits à utiliser lors de l'exécution de la tâche. On utilise alors le mot clé « definer » suivi d'un égal puis du nom de l'utilisateur MySql voulu, au format : 'nom_utilisateur'@'host'.

Avec :


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

- nom_utilisateur, le nom de l'utilisateur dont on veut donner les pouvoirs au déclencheur.
- host, le nom de l'hôte ou son adresse IP.

Si l'on omet d'utiliser « definer », c'est l'utilisateur courant qui crée la tâche. C'est comme si on avait utilisé explicitement : « definer=current_user ».

Si le nom d'utilisateur pour « definer » est différent de celui qui exécute la requête de création alors ce dernier devra avoir le droit « SUPER ».

On place alors « definer » entre le « create » et « event ».

La syntaxe est donc :

```
delimiter $$

#la syntaxe la plus simple
delimiter $$create event nom1
on schedule options_planification
do
begin
bloc_de_code_ou_instruction;
end$$

#la syntaxe la plus complète
create definer=nom_utilisateur event if not exists nom2
on schedule options_planification
options_persistance
options_activation
do
begin
bloc_de_code_ou_instruction;
end$$

delimiter ;
```

On n'abordera pas ici le « alter event » qui correspond à un « drop event if exists » suivi du create event » avec le même nom, permettant d'être plus générique.

La syntaxe complète de déclaration avec suppression préalable si nécessaire est :

```
delimiter $$

#la syntaxe la plus simple
drop event if exists nom1$$
create event nom1
on schedule options_planification
do
begin
bloc_de_code_ou_instruction;
end$$

#la syntaxe la plus complète
drop event if exists nom2$$
create definer=nom_utilisateur event nom2
on schedule options_planification
options_persistance
options_activation
do
begin
bloc_de_code_ou_instruction;
end$$

delimiter ;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

On notera qu'il peut être bon de mutualiser ces actions si on a plusieurs fonctions à créer ou modifier pour n'avoir qu'un seul « delimiter » au début et à la fin du code.

À savoir que le « SQL_MODE » est stocké lors de la création et que c'est celui-ci et non celui qui est défini juste avant l'exécution du déclencheur qui sera utilisé.

## 5-3-2 - Suppression d'une tâche

On utilise les mots clés « drop event », éventuellement suivi de « if exists » pour éviter une erreur si la tâche n'existe pas, et enfin du nom de la tâche. On peut préciser le nom d'une base devant le nom de la tâche en les séparant par un point sinon c'est le nom de la base sélectionnée qui sera utilisé par défaut.

La syntaxe est donc :

```
#la syntaxe la plus simple
drop event nom;

#la syntaxe la plus complète
drop event if exists `base`.`nom`;
```

En cas de tâche inconnue on a l'erreur :

« ERROR 1517 (HY000): Unknown event '%s' ».

## 5-3-3 - Informations sur les tâches et le planificateur

### 5-3-3-1 - Informations sur le planificateur

On peut avoir le décompte d'un certain nombre d'actions liées au planificateur depuis le dernier démarrage avec 5 variables globales :

- « @@global.com_create_event » est le nombre de création de tâches.
- « @@global.com_drop_event » est le nombre de suppressions de tâches.
- « @@global.com_alter_event » est le nombre de modifications de tâches.
- « @@global.com_show_create_event » est le nombre de demandes d'information sur le code de création de tâches.
- « @@global.com_show_events » est le nombre de demandes d'information sur les tâches.

On peut aussi, pour lister ces 5 valeurs, faire :

```
show status like '%event%';
```

On peut voir l'état du planificateur avec la variable « @@global.event_scheduler ».

### 5-3-3-2 - Informations sur les tâches

#### 5-3-3-2-1 - Syntaxe de « show events »

Pour voir la liste des tâches on va utiliser les mots clés « show events », éventuellement suivis par le mot clé « in » ou « from » introduisant le nom d'une base. On peut ensuite préciser un motif de recherche pour le nom de la tâche avec le mot clé « like » suivi par le motif. Enfin on peut rechercher avec la mot clé « where » suivi d'une expression celle-ci dans les différentes informations sur la tâche.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

La syntaxe est donc :

```
#la syntaxe la plus simple
show events;

#la syntaxe la plus complète
show events in nom_base like '%motif%' where 'expr';
```

Une date peut être au format :

- STZ : « session time zone », date en fonction de la valeur de « @@time_zone » au moment où une action se produit vis à vis de la tâche.
- UTC : « universal time coordinated », date en temps universel.
- ETZ : « event time zone » date en fonction de la valeur de « @@time_zone » au moment de la création (ou modification) de la tâche.

L'interprétation des dates dépend de la version de MySQL :

- la date de création et de dernière modification qui sont stockées dans « mysql.event » et « information_schema.events » sont toujours au format STZ quelque soit le version de MySQL.

- avant MySQL 5.1.17, toutes les dates stockées sont en UTC dans « mysql.event », « information_schema.events » et « show events ».
- après MySQL 5.1.17, les dates dans « information_schema.events » et « show events » sont en ETZ, celles dans « mysql.event » en UTC.

Par contre, c'est toujours la valeur de date en UTC qui sert à gérer l'action pour garantir un fonctionnement cohérent.

Les informations fournies sont :

- le nom de la base qui héberge la tâche.
- le nom de la tâche.
- la valeur de « @@time_zone » utilisée pour la planification.
- le nom de l'utilisateur qui a créé la tâche.
- le type d'exécution : « one time » pour une unique exécution ou « recurring » pour répétitive.
- la date en cas d'exécution unique.
- la valeur de l'intervalle de temps en cas de répétition.
- l'unité de l'intervalle en cas de répétition.
- la date de première exécution si elle est définie en cas de répétition.
- la date de dernière exécution si elle est définie en cas de répétition.
- le status (« enable » pour actif, « disable » pour inactif ou « slaveside_disable » pour inactif en cas de réplication, cette dernière valeur n'existe que depuis MySQL 5.1.18)
- l'identifiant du serveur sur lequel a été créée la tâche depuis MySQL 5.1.18.

Depuis MySQL 5.1.21 on a aussi :

- le jeu de caractères de la connexion lors de la création de la tâche.
- la collation de la connexion lors de la création de la tâche.
- la collation de la base hébergeant la tâche.

### 5-3-3-2-2 - Syntaxe de « show create event »

On utilise les mots clés « show create event » suivis par le nom de la tâche, éventuellement précédé par le nom de la base concernée avec un point pour les séparer.

La syntaxe est donc :


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

```
show create event nom;
```

Les informations fournies sont :

- le nom.
- le « SQL_MODE » au moment de la création.
- la valeur de « @@time_zone » au moment de la création.
- la chaîne de caractères contenant la requête de création de la tâche correspondant à son état d'activité actuelle et non celle de sa création initiale.

Depuis MySQL 5.1.21 on a aussi :

- le jeu de caractères de la connexion lors de la création de la tâche.
- la collation de la connexion lors de la création de la tâche.
- la collation de la base hébergeant la tâche.

# 6 - Les déclencheurs

## 6-1 - Rôle

Leur support a été introduit dans MySql 5.0.2 mais il reste rudimentaire. Actuellement, l'espace de nom est lié à la table pour laquelle ils sont créés (le même nom ne peut exister 2 fois pour une même table) mais il devrait évoluer pour être au niveau serveur. Il vaut mieux donc prendre l'habitude d'avoir des noms uniques pour les déclencheurs au niveau du serveur ou au moins de la base de données.

Ils peuvent servir :

- à garantir les règles d'insertion, de modification ou de suppression dans les tables où on les crée.
- à exécuter des traitements sur d'autres tables lors des ces événements.

Pour chacune de ses actions, le déclencheur peut être définit sur 2 moments particuliers :

- avant (« before » en anglais), on peut donc empêcher l'action ou modifier la façon dont il agit.
- après (« after » en anglais), on peut alors agir une fois l'action réalisée.

Attention il ne peut y avoir plus d'un déclencheur pour un événement sur une action.

## 6-2 - La sécurité

Pour pouvoir créer ou supprimer un déclencheur, il faut avoir le droit « SUPER », sauf à partir de MySql 5.1.6, où l'utilisateur doit avoir le droits « TRIGGER ».

Lors de l'exécution, à partir de MySql 5.1.6, l'utilisateur choisi pour l'exécution du déclencheur doit avoir les droits suffisants pour :

- les requêtes présentes dans le déclencheur sur d'autres tables que la table concernée.
- l'action correspondant au déclencheur sur la table concernée.

Dans certains cas, le droit « SUPER » sera nécessaire tout de même.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

# 6-3 - La syntaxe

## 6-3-1 - Création

La syntaxe générique utilise les mots clés « create trigger » suivis du nom du déclencheur. On doit ensuite mettre le type d'événements en utilisant les mots clés « before » ou « after », suivi du type d'action définie par « insert », « delete » ou « update ». On utilise, enfin, le mot clé « on » suivit du nom de la table concernée par le déclencheur. L'expression « for each row » introduit le bloc de code à exécuter (comprendre ici : pour chaque ligne à insérer, modifier ou supprimer selon l'action faire...).

On effectuera la suppression d'une version antérieure du déclencheur dans le cas ou elle existerait.

Par exemple :

```
#code générique pour la création d un déclencheur
delimiter $$
drop trigger if exists nom$$
create trigger nom evenement on nom_table$$
for each row
begin
listes_instructions;
end$$
delimiter ;
```

Il est plus clair d'utiliser un bloc de code même, pour une simple requête mais ce n'est pas obligatoire...

À partir de MySql 5.1.6, on peut préciser les droits à utiliser lors de l'activation du déclencheur. On utilise alors le mot clé « definer » suivi d'un égal puis du nom de l'utilisateur MySql voulu, au format : 'nom_utilisateur'@'host'.

Avec :

- nom_utilisateur, le nom de l'utilisateur dont on veut donner les pouvoirs au déclencheur.
- host, le nom de l'hôte ou son adresse IP.

La syntaxe est alors :

```
#code générique pour la création d un déclencheur
delimiter $$
drop trigger if exists nom$$
create definer=nom_utilisateur trigger nom evenement on nom_table$$
for each row
begin
listes_instructions;
end$$
delimiter ;
```

Si l'on omet d'utiliser « definer », c'est l'utilisateur courant qui crée le déclencheur. C'est comme si on avait utilisé explicitement : « definer=current_user ».

On ne peut précisé un autre utilisateur avec « definer » que celui qui crée le déclencheur que si et seulement si le créateur a le droit « SUPER ».

Ils sont normalement considérés comme du code procédural avec l'option « deterministic » même s'ils n'y a aucun moyen de la fixer... L'idée est qu'on attend, en principe, à ce qu'un déclencheur fasse la même chose à chaque fois que la même situation est rencontrée... Cette considération n'est pas valide dans certains cas comme l'utilisation de « uuid() » qui génère un code unique à chaque appel, par exemple... Ce qui peut poser des problème de réplication, notamment.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

À savoir que le « SQL_MODE » est stocké lors de la création et que c'est celui-ci et non celui qui est défini juste avant l'exécution du déclencheur qui sera utilisé.

### 6-3-2 - Suppression

On utilise le mots clés « drop trigger », éventuellement suivi par « if exists » si on veut tester l'existance du déclencheur à supprimer pour éviter un warning, suivi du nom du déclencheur à supprimer.

La syntaxe est donc :

```
drop trigger nom;

drop trigger if exists nom;
```

À noter qu'avant MySql 5.0.10, le nom de la table concernant le déclencheur à supprimer devait figurer obligatoirement devant le nom de celui-ci, séparé par un « . ».

Les déclencheurs sont automatiquement supprimés si la table sur laquelle ils portent est supprimée.

## 6-4 - Limitations

### 6-4-1 - Impossibilité de création

Un déclencheur ne peut être créé que sur une table permanente, on ne peut donc pas en créer sur une table avec le moteur « temporary » ou « memory » ou une vue ou l'une des tables système de la base « mysql ».

### 6-4-2 - Interdiction de références circulaires

On ne peut jamais faire une action qui se passe explicitement ou qui déclenche un réaction sur la table qui concerne un déclencheur dans celui-ci.

### 6-4-3 - Contenu interdit

On ne peut pas mettre dedans :

- une instruction produisant un affichage (« show » ou encore un « select » n'utilisant pas de « into », etc...).
- un appel à une procédure stockée, avant MySql 5.1.6.
- utiliser les instructions déclenchant les exceptions ou warnings, avant MySql 5.1.6.
- une instruction engendrant le début ou la fin d'une transaction pour les tables qui ont le moteur « innodb ».

### 6-4-4 - Comportement lors d'interactions entre tables

Lors d'une cascade de mise à jour, les déclencheurs des tables concernées par la cascade ne sont pas appelés (sauf ceux de la table initiale donc).

Les déclencheurs ne se déclenchent pas :

- lors d'action sur des clés étrangères.
- sur les serveurs esclaves, lors d'une réplication, quand il y a des modifications sur le serveur maître.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/

### 6-4-5 - Obligations lors de changement de version de MySql

En cas de mise à jour d'une version antérieure à MySql 5.0.10, 5.0.16, 5.1.0 ou 5.5.0 vers cette version ou supérieure, on devra supprimer tous les déclencheurs avant de pouvoir faire la mise à jour et les recréer ensuite sinon la commande pour les supprimer ne marchera plus.

## 6-5 - Utilisations

### 6-5-1 - Avec InnoDB

Les déclencheurs font parti de la transaction même si un « begin » est présent au début de son corps. Attention, un « begin » dans du procédural est toujours considéré comme un début de bloc de code.

### 6-5-2 - Cas de l'évènement « on insert »

Les déclencheurs sur cet événement s'exécuteront en cas de commandes :

- « insert ».
- « load data ».
- « replace ».

### 6-5-3 - Cas de l'évènement « on update »

Les déclencheurs sur cet événement s'exécuteront en cas de commandes « update ».

### 6-5-4 - Cas de l'évènement « on delete »

Les déclencheurs sur cet événement s'exécuteront en cas de commandes :

- « delete ».
- « replace ».

Attention, les instructions « truncate table » ou « drop table » ne généreront pas ces événement.

### 6-5-5 - Cas particulier

Si on utilise une commande « insert » avec l'option « on duplicate key update » alors on déclenchera des événements « on insert » ou « on update » selon le cas rencontré à chaque ligne.


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Dureuil Eric. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://ericd69.developpez.com/tutoriels/mysql/