Version : **2022.01**

Dernière mise-à-jour : 2022/11/10 07:14

SER803 - Procédures, Fonctions, Déclencheurs, Vues et le Planificateur d'Evénements

Contenu du Module

Routines Stockées

Procédures stockées

Les procédures stockées et les fonctions sont créées avec les commandes **CREATE PROCEDURE** et **CREATE FUNCTION**. Une procédure est appelée avec la commande **CALL** et peut appeler une autre **routine stockée**. Une routine est une procédure stockée ou une fonction.

Une routine hérite de la base de données par défaut de l'utilisateur appelant, et donc, il est recommandé de commencer les routines avec la commande USE dbname, ou de spécifier explicitement les tables et les bases de données qui sont utilisées : i.e. base.table.

Pour illustrer les procédures stockées, vous allez travailler sur une base de données conçue pour suivre des rencontres et résultats d'un ligue d'équipes de football. Commencez par créer la base **ligue1** et les deux tables **equipe** et **rencontre**:

```
[root@centos8 ~]# mysql -uroot -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.3.28-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE ligue1;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]>
```

Utilisez les requêtes suivantes pour créer les tables **equipe** et **rencontre** :

```
  CREATE TABLE ligue1.equipe (
  id_equipe int primary key auto_increment,
  nom varchar(50) not null,
  stade varchar(50) not null,
  ville varchar(30) not null,
  points int not null default 0,
  buts int not null default 0
  ) ENGINE=MyISAM;
```

```
  CREATE TABLE ligue1.rencontre (
  id_domicile int,
  id_visiteurs int,
  date_match DATETIME,
  score_domicile tinyint,
```

```
  score_visiteurs tinyint,
  arbitre varchar(80),
  primary key (id_domicile, id_visiteurs, date_match)
  ) ENGINE=MyISAM;
```

Vous obtiendrez :

```
MariaDB [(none)]>   CREATE TABLE ligue1.equipe (
    ->   id_equipe int primary key auto_increment,
    ->   nom varchar(50) not null,
    ->   stade varchar(50) not null,
    ->   ville varchar(30) not null,
    ->   points int not null default 0,
    ->   buts int not null default 0
    ->   ) ENGINE=MyISAM;
Query OK, 0 rows affected (0.052 sec)

MariaDB [(none)]>   CREATE TABLE ligue1.rencontre (
    ->   id_domicile int,
    ->   id_visiteurs int,
    ->   date_match DATETIME,
    ->   score_domicile tinyint,
    ->   score_visiteurs tinyint,
    ->   arbitre varchar(80),
    ->   primary key (id_domicile, id_visiteurs, date_match)
    ->   ) ENGINE=MyISAM;
Query OK, 0 rows affected (0.038 sec)

MariaDB [(none)]>
```

Sélectionnez la base **ligue1**:

```
MariaDB [(none)]> USE ligue1;
Reading table information for completion of table and column names
```

```
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [ligue1]>
```

Ensuite créez la première procédure dans la base ligue1:

```
  DELIMITER //
  CREATE PROCEDURE ligue1.INIT_EQUIPE (nom_eq varchar(50), stade_eq varchar(50), ville_eq varchar(30))
  BEGIN
  INSERT INTO equipe (nom, stade, ville) VALUES (nom_eq, stade_eq, ville_eq);
  END//
  DELIMITER ;
```

Cette procédure a pour but l'insertion d'une nouvelle équipe, le stade et la ville. Vous obtiendrez un résultat similaire à celui-ci:

```
MariaDB [ligue1]>   DELIMITER //
MariaDB [ligue1]>   CREATE PROCEDURE ligue1.INIT_EQUIPE (nom_eq varchar(50), stade_eq varchar(50), ville_eq
varchar(30))
    ->   BEGIN
    ->   INSERT INTO equipe (nom, stade, ville) VALUES (nom_eq, stade_eq, ville_eq);
    ->   END//
Query OK, 0 rows affected (0.016 sec)

MariaDB [ligue1]>   DELIMITER ;
MariaDB [ligue1]>
```

Appelez la procédure pour saisir trois équipes:

```
MariaDB [ligue1]> CALL ligue1.INIT_EQUIPE('FC Mandriva', 'Parc des Princes', 'Paris');
Query OK, 1 row affected (0.000 sec)

MariaDB [ligue1]> CALL ligue1.INIT_EQUIPE('Debian AC', 'Yankee Stadium', 'New York');
Query OK, 1 row affected (0.000 sec)
```

```
MariaDB [ligue1]> CALL ligue1.INIT_EQUIPE('Vista FC', 'Qwest Field', 'Redmond');
Query OK, 1 row affected (0.000 sec)

MariaDB [ligue1]>
```

Visualisez les entrées de la table **equipe**:

```
MariaDB [ligue1]> SELECT * FROM ligue1.equipe;
+-----------+-------------+------------------+----------+--------+------+
| id_equipe | nom         | stade            | ville    | points | buts |
+-----------+-------------+------------------+----------+--------+------+
|         1 | FC Mandriva | Parc des Princes | Paris    |      0 |    0 |
|         2 | Debian AC   | Yankee Stadium   | New York |      0 |    0 |
|         3 | Vista FC    | Qwest Field      | Redmond  |      0 |    0 |
+-----------+-------------+------------------+----------+--------+------+
3 rows in set (0.000 sec)

MariaDB [ligue1]>
```

Créez une deuxième procédure pour initialiser les rencontres:

```
  DELIMITER //
  CREATE PROCEDURE ligue1.SAISIR_RENCONTRE (id_dom INTEGER, id_vis INTEGER, date_m DATETIME, arbitre_m
VARCHAR(80))
  BEGIN
  INSERT INTO rencontre (id_domicile, id_visiteurs, date_match, arbitre)
  VALUES (id_dom, id_vis, date_m, arbitre_m);
  END//
  DELIMITER ;
```

Cette procédure crée un rencontre entre deux équipes à une date précise et avec un arbitre précis. Vous obtiendrez un résultat similaire à celui-ci:

```
MariaDB [ligue1]>   DELIMITER //
MariaDB [ligue1]>   CREATE PROCEDURE ligue1.SAISIR_RENCONTRE (id_dom INTEGER, id_vis INTEGER, date_m DATETIME,
```

```
arbitre_m VARCHAR(80))
    ->   BEGIN
    ->   INSERT INTO rencontre (id_domicile, id_visiteurs, date_match, arbitre)
    ->   VALUES (id_dom, id_vis, date_m, arbitre_m);
    ->   END//
Query OK, 0 rows affected (0.043 sec)

MariaDB [ligue1]>   DELIMITER ;
MariaDB [ligue1]>
```

Appelez la procédure pour créer trois rencontres:

```
MariaDB [ligue1]> CALL ligue1.SAISIR_RENCONTRE(1, 2, '2016-12-25', 'Paul Unix');
Query OK, 1 row affected (0.000 sec)

MariaDB [ligue1]> CALL ligue1.SAISIR_RENCONTRE(2, 3, '2016-12-26', 'Joseph Bash');
Query OK, 1 row affected (0.000 sec)

MariaDB [ligue1]> CALL ligue1.SAISIR_RENCONTRE(3, 4, '2016-12-27', 'Jean Fenêtre');
Query OK, 1 row affected (0.000 sec)

MariaDB [ligue1]>
```

Visualisez les entrées de la table **rencontre**:

```
MariaDB [ligue1]> SELECT * FROM ligue1.rencontre;
+-------------+--------------+---------------------+----------------+-----------------+---------------+
| id_domicile | id_visiteurs | date_match          | score_domicile | score_visiteurs | arbitre       |
+-------------+--------------+---------------------+----------------+-----------------+---------------+
|           1 |            2 | 2016-12-25 00:00:00 |           NULL |            NULL | Paul Unix     |
|           2 |            3 | 2016-12-26 00:00:00 |           NULL |            NULL | Joseph Bash   |
|           3 |            4 | 2016-12-27 00:00:00 |           NULL |            NULL | Jean Fenêtre  |
+-------------+--------------+---------------------+----------------+-----------------+---------------+
3 rows in set (0.000 sec)
```

```
MariaDB [ligue1]>
```

Les matchs ayant eu lieu, il faut maintenant mettre à jour cette table. Créez une procédure pour la mise à jour des résultats:

```
DELIMITER //
CREATE PROCEDURE `ligue1`.`SAISIR_RESULTAT`(id_dom INTEGER, id_vis INTEGER, score_dom TINYINT(4), score_vis
TINYINT(4))
BEGIN
UPDATE ligue1.rencontre
SET score_domicile = score_dom, score_visiteurs = score_vis
WHERE id_domicile=id_dom AND id_visiteurs=id_vis;
IF score_dom = score_vis THEN
UPDATE equipe SET points = points+1
WHERE id_equipe=id_dom OR id_equipe=id_vis;
ELSEIF score_dom > score_vis THEN
UPDATE equipe SET points = points+3 WHERE id_equipe = id_dom;
ELSE UPDATE equipe SET points = points+3 WHERE id_equipe = id_vis;
END IF;
END//
DELIMITER ;
```

Vous obtiendrez un résultat similaire à celui-ci:

```
MariaDB [ligue1]> DELIMITER //
MariaDB [ligue1]> CREATE PROCEDURE `ligue1`.`SAISIR_RESULTAT`(id_dom INTEGER, id_vis INTEGER, score_dom
TINYINT(4), score_vis TINYINT(4))
    -> BEGIN
    -> UPDATE ligue1.rencontre
    -> SET score_domicile = score_dom, score_visiteurs = score_vis
    -> WHERE id_domicile=id_dom AND id_visiteurs=id_vis;
    -> IF score_dom = score_vis THEN
    -> UPDATE equipe SET points = points+1
    -> WHERE id_equipe=id_dom OR id_equipe=id_vis;
    -> ELSEIF score_dom > score_vis THEN
```

```
    -> UPDATE equipe SET points = points+3 WHERE id_equipe = id_dom;
    -> ELSE UPDATE equipe SET points = points+3 WHERE id_equipe = id_vis;
    -> END IF;
    -> END//
Query OK, 0 rows affected (0.034 sec)

MariaDB [ligue1]> DELIMITER ;
MariaDB [ligue1]>
```

Appelez la procédure pour les trois matchs:

```
MariaDB [ligue1]> CALL ligue1.SAISIR_RESULTAT(1, 2, 2, 1);
Query OK, 2 rows affected (0.001 sec)

MariaDB [ligue1]> CALL ligue1.SAISIR_RESULTAT(2, 3, 1, 3);
Query OK, 2 rows affected (0.000 sec)

MariaDB [ligue1]> CALL ligue1.SAISIR_RESULTAT(3, 4, 4, 2);
Query OK, 2 rows affected (0.000 sec)

MariaDB [ligue1]>
```

Visualisez les entrées de la table **rencontre**:

```
MariaDB [ligue1]> SELECT * FROM ligue1.rencontre;
+-------------+--------------+---------------------+----------------+-----------------+---------------+
| id_domicile | id_visiteurs | date_match          | score_domicile | score_visiteurs | arbitre       |
+-------------+--------------+---------------------+----------------+-----------------+---------------+
|           1 |            2 | 2016-12-25 00:00:00 |              2 |               1 | Paul Unix     |
|           2 |            3 | 2016-12-26 00:00:00 |              1 |               3 | Joseph Bash   |
|           3 |            4 | 2016-12-27 00:00:00 |              4 |               2 | Jean Fenêtre  |
+-------------+--------------+---------------------+----------------+-----------------+---------------+
3 rows in set (0.000 sec)
```

```
MariaDB [ligue1]>
```

Visualisez la table ligue1.equipe et vérifiez que les buts et les points ont été mis à jour :

```
MariaDB [ligue1]> SELECT * FROM equipe ;
+-----------+-------------+------------------+----------+--------+------+
| id_equipe | nom         | stade            | ville    | points | buts |
+-----------+-------------+------------------+----------+--------+------+
|         1 | FC Mandriva | Parc des Princes | Paris    |      3 |    0 |
|         2 | Debian AC   | Yankee Stadium   | New York |      0 |    0 |
|         3 | Vista FC    | Qwest Field      | Redmond  |      6 |    0 |
+-----------+-------------+------------------+----------+--------+------+
3 rows in set (0.000 sec)

MariaDB [ligue1]>
```

La nouvelle saison arrivée, il serait utile d'avoir une procédure qui vous crée les rencontres automatiquement. Créez donc une procédure pour initier les rencontres:

```
DELIMITER //
CREATE PROCEDURE `ligue1`.`INIT_RENCONTRES`()
BEGIN
DECLARE fini INT default 0;
DECLARE domicile, visiteur INT;
DECLARE cur_domicile CURSOR FOR SELECT id_equipe FROM equipe;
DECLARE cur_visiteur CURSOR FOR SELECT id_equipe FROM equipe;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET fini = 1;
OPEN cur_domicile;
WHILE fini <> 1 DO
FETCH cur_domicile INTO domicile;
IF fini=0 THEN
OPEN cur_visiteur;
WHILE fini <> 1 DO
FETCH cur_visiteur INTO visiteur;
```

```
IF domicile <> visiteur AND fini <> 1 THEN
INSERT IGNORE INTO rencontre (id_domicile, id_visiteurs) VALUES (domicile, visiteur);
END IF;
END WHILE;
CLOSE cur_visiteur;
SET fini=0;
END IF;
END WHILE;
CLOSE cur_domicile;
END//
DELIMITER ;
```

Vous obtiendrez un résultat similaire à celui-ci:

```
MariaDB [ligue1]> DELIMITER //
MariaDB [ligue1]> CREATE PROCEDURE `ligue1`.`INIT_RENCONTRES`()
    -> BEGIN
    -> DECLARE fini INT default 0;
    -> DECLARE domicile, visiteur INT;
    -> DECLARE cur_domicile CURSOR FOR SELECT id_equipe FROM equipe;
    -> DECLARE cur_visiteur CURSOR FOR SELECT id_equipe FROM equipe;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET fini = 1;
    -> OPEN cur_domicile;
    -> WHILE fini <> 1 DO
    -> FETCH cur_domicile INTO domicile;
    -> IF fini=0 THEN
    -> OPEN cur_visiteur;
    -> WHILE fini <> 1 DO
    -> FETCH cur_visiteur INTO visiteur;
    -> IF domicile <> visiteur AND fini <> 1 THEN
    -> INSERT IGNORE INTO rencontre (id_domicile, id_visiteurs) VALUES (domicile, visiteur);
    -> END IF;
    -> END WHILE;
    -> CLOSE cur_visiteur;
```

```
    -> SET fini=0;
    -> END IF;
    -> END WHILE;
    -> CLOSE cur_domicile;
    -> END//
Query OK, 0 rows affected (0.149 sec)

MariaDB [ligue1]> DELIMITER ;
```

Un **curseur** est un objet permettant de parcourir d'une manière séquentielle le jeu de résultats retourné par une requête SQL. Dans notre cas, nous avons besoin de créer deux curseurs **cur_domicile** et **cur_visiteur** portant sur la même requête **SELECT id_equipe FROM equipe**. Le parcours du deuxième curseur est placé à l'intérieur du premier pour passer en revue toutes les combinaisons de couples (equipe1, equipe2). De cette façon, pour chaque équipe domicile on parcourt toutes les équipes visiteuses qui vont jouer contre elle (c'est à dire toutes sauf elle-même) et on insère un nouveau match à chaque fois.

Le curseur est ouvert via l'ordre **OPEN** et fermé avec **CLOSE**. La syntaxe d'itération sur un curseur repose sur une simple boucle WHILE dans laquelle on fait avancer le curseur avec **FETCH**, et sur un **HANDLER**.

Un **HANDLER** sert à déterminer ce qui se passe lorsque le **SQLSTATE NOT FOUND** est atteint, autrement dit quand le curseur est au bout du jeu de résultats. En général on sort de la boucle WHILE.

Supprimez maintenant les enregistrements de la saison précédente se trouvant dans la table ligue1.rencontres :

```
MariaDB [ligue1]> DELETE FROM ligue1.rencontre WHERE id_domicile IS NOT NULL;
Query OK, 3 rows affected (0.000 sec)

MariaDB [ligue1]>
```

Appelez la procédure:

```
MariaDB [ligue1]> CALL ligue1.INIT_RENCONTRES();
Query OK, 0 rows affected, 1 warning (0.00 sec)

MariaDB [ligue1]>
```

Visualisez les entrées de la table **rencontre**:

```
MariaDB [ligue1]> SELECT * FROM ligue1.rencontre;
+-------------+--------------+---------------------+----------------+------------------+---------+
| id_domicile | id_visiteurs | date_match          | score_domicile | score_visiteurs | arbitre |
+-------------+--------------+---------------------+----------------+------------------+---------+
|           2 |            3 | 0000-00-00 00:00:00 |           NULL |             NULL | NULL    |
|           2 |            1 | 0000-00-00 00:00:00 |           NULL |             NULL | NULL    |
|           1 |            3 | 0000-00-00 00:00:00 |           NULL |             NULL | NULL    |
|           1 |            2 | 0000-00-00 00:00:00 |           NULL |             NULL | NULL    |
|           3 |            1 | 0000-00-00 00:00:00 |           NULL |             NULL | NULL    |
|           3 |            2 | 0000-00-00 00:00:00 |           NULL |             NULL | NULL    |
+-------------+--------------+---------------------+----------------+------------------+---------+
6 rows in set (0.000 sec)

MariaDB [ligue1]>
```

Réinitialisez la table ligue1.equipe :

```
MariaDB [ligue1]> UPDATE ligue1.equipe SET points=0, buts=0;
Query OK, 2 rows affected (0.000 sec)
Rows matched: 3  Changed: 2  Warnings: 0

MariaDB [ligue1]>
```

Visualisez les entrées de la table **equipe** :

```
MariaDB [ligue1]> SELECT * FROM ligue1.equipe;
+-----------+-------------+------------------+----------+--------+------+
| id_equipe | nom         | stade            | ville    | points | buts |
+-----------+-------------+------------------+----------+--------+------+
|         1 | FC Mandriva | Parc des Princes | Paris    |      0 |    0 |
|         2 | Debian AC   | Yankee Stadium   | New York |      0 |    0 |
|         3 | Vista FC    | Qwest Field      | Redmond  |      0 |    0 |
```

```
+----------+------------+-----------------+----------+--------+------+
3 rows in set (0.000 sec)

MariaDB [ligue1]>
```

Le code d'une procédure est stocké dans la table interne **INFORMATION_SCHEMA.ROUTINES**. Pour visualiser les procédures créées, il convient d'utiliser la commande suivante :

```
MariaDB [ligue1]> SELECT * FROM information_schema.routines WHERE ROUTINE_TYPE='PROCEDURE'\G
*************************** 1. row ***************************
           SPECIFIC_NAME: INIT_EQUIPE
         ROUTINE_CATALOG: def
          ROUTINE_SCHEMA: ligue1
            ROUTINE_NAME: INIT_EQUIPE
            ROUTINE_TYPE: PROCEDURE
               DATA_TYPE:
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
          DTD_IDENTIFIER: NULL
            ROUTINE_BODY: SQL
      ROUTINE_DEFINITION: BEGIN
  INSERT INTO equipe (nom, stade, ville) VALUES (nom_eq, stade_eq, ville_eq);
  END
           EXTERNAL_NAME: NULL
       EXTERNAL_LANGUAGE: NULL
         PARAMETER_STYLE: SQL
        IS_DETERMINISTIC: NO
         SQL_DATA_ACCESS: CONTAINS SQL
                SQL_PATH: NULL
```

```
           SECURITY_TYPE: DEFINER
                 CREATED: 2022-11-08 05:01:26
            LAST_ALTERED: 2022-11-08 05:01:26
                SQL_MODE:
STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
        ROUTINE_COMMENT:
                 DEFINER: root@localhost
    CHARACTER_SET_CLIENT: utf8
    COLLATION_CONNECTION: utf8_general_ci
      DATABASE_COLLATION: latin1_swedish_ci
*************************** 2. row ***************************
           SPECIFIC_NAME: INIT_RENCONTRES
         ROUTINE_CATALOG: def
          ROUTINE_SCHEMA: ligue1
            ROUTINE_NAME: INIT_RENCONTRES
            ROUTINE_TYPE: PROCEDURE
               DATA_TYPE:
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
          DTD_IDENTIFIER: NULL
            ROUTINE_BODY: SQL
      ROUTINE_DEFINITION: BEGIN
DECLARE fini INT default 0;
DECLARE domicile, visiteur INT;
DECLARE cur_domicile CURSOR FOR SELECT id_equipe FROM equipe;
DECLARE cur_visiteur CURSOR FOR SELECT id_equipe FROM equipe;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET fini = 1;
OPEN cur_domicile;
WHILE fini <> 1 DO
```

```
FETCH cur_domicile INTO domicile;
IF fini=0 THEN
OPEN cur_visiteur;
WHILE fini <> 1 DO
FETCH cur_visiteur INTO visiteur;
IF domicile <> visiteur AND fini <> 1 THEN
INSERT IGNORE INTO rencontre (id_domicile, id_visiteurs) VALUES (domicile, visiteur);
END IF;
END WHILE;
CLOSE cur_visiteur;
SET fini=0;
END IF;
END WHILE;
CLOSE cur_domicile;
END
           EXTERNAL_NAME: NULL
       EXTERNAL_LANGUAGE: NULL
         PARAMETER_STYLE: SQL
        IS_DETERMINISTIC: NO
         SQL_DATA_ACCESS: CONTAINS SQL
                SQL_PATH: NULL
           SECURITY_TYPE: DEFINER
                 CREATED: 2022-11-08 11:52:13
            LAST_ALTERED: 2022-11-08 11:52:13
                SQL_MODE:
STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
         ROUTINE_COMMENT:
                 DEFINER: root@localhost
    CHARACTER_SET_CLIENT: utf8
    COLLATION_CONNECTION: utf8_general_ci
      DATABASE_COLLATION: latin1_swedish_ci
*************************** 3. row ***************************
           SPECIFIC_NAME: SAISIR_RENCONTRE
         ROUTINE_CATALOG: def
```

```
          ROUTINE_SCHEMA: ligue1
            ROUTINE_NAME: SAISIR_RENCONTRE
            ROUTINE_TYPE: PROCEDURE
               DATA_TYPE:
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
          DTD_IDENTIFIER: NULL
            ROUTINE_BODY: SQL
      ROUTINE_DEFINITION: BEGIN
  INSERT INTO rencontre (id_domicile, id_visiteurs, date_match, arbitre)
  VALUES (id_dom, id_vis, date_m, arbitre_m);
  END
           EXTERNAL_NAME: NULL
       EXTERNAL_LANGUAGE: NULL
         PARAMETER_STYLE: SQL
        IS_DETERMINISTIC: NO
         SQL_DATA_ACCESS: CONTAINS SQL
                SQL_PATH: NULL
           SECURITY_TYPE: DEFINER
                 CREATED: 2022-11-08 05:03:39
            LAST_ALTERED: 2022-11-08 05:03:39
                SQL_MODE:
STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
         ROUTINE_COMMENT:
                 DEFINER: root@localhost
    CHARACTER_SET_CLIENT: utf8
    COLLATION_CONNECTION: utf8_general_ci
      DATABASE_COLLATION: latin1_swedish_ci
*************************** 4. row ***************************
```

```
           SPECIFIC_NAME: SAISIR_RESULTAT
         ROUTINE_CATALOG: def
          ROUTINE_SCHEMA: ligue1
            ROUTINE_NAME: SAISIR_RESULTAT
            ROUTINE_TYPE: PROCEDURE
               DATA_TYPE:
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
          DTD_IDENTIFIER: NULL
            ROUTINE_BODY: SQL
      ROUTINE_DEFINITION: BEGIN
UPDATE ligue1.rencontre
SET score_domicile = score_dom, score_visiteurs = score_vis
WHERE id_domicile=id_dom AND id_visiteurs=id_vis;
IF score_dom = score_vis THEN
UPDATE equipe SET points = points+1
WHERE id_equipe=id_dom OR id_equipe=id_vis;
ELSEIF score_dom > score_vis THEN
UPDATE equipe SET points = points+3 WHERE id_equipe = id_dom;
ELSE UPDATE equipe SET points = points+3 WHERE id_equipe = id_vis;
END IF;
END
           EXTERNAL_NAME: NULL
       EXTERNAL_LANGUAGE: NULL
         PARAMETER_STYLE: SQL
        IS_DETERMINISTIC: NO
         SQL_DATA_ACCESS: CONTAINS SQL
                SQL_PATH: NULL
           SECURITY_TYPE: DEFINER
```

```
                CREATED: 2022-11-08 05:06:32
           LAST_ALTERED: 2022-11-08 05:06:32
               SQL_MODE:
STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
        ROUTINE_COMMENT:
                DEFINER: root@localhost
    CHARACTER_SET_CLIENT: utf8
    COLLATION_CONNECTION: utf8_general_ci
      DATABASE_COLLATION: latin1_swedish_ci
*************************** 5. row ***************************
          SPECIFIC_NAME: AddGeometryColumn
        ROUTINE_CATALOG: def
         ROUTINE_SCHEMA: mysql
           ROUTINE_NAME: AddGeometryColumn
           ROUTINE_TYPE: PROCEDURE
              DATA_TYPE:
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
          DTD_IDENTIFIER: NULL
            ROUTINE_BODY: SQL
      ROUTINE_DEFINITION: begin
  set @qwe= concat('ALTER TABLE ', t_schema, '.', t_name, ' ADD ', geometry_column,' GEOMETRY REF_SYSTEM_ID=',
t_srid); PREPARE ls from @qwe; execute ls; deallocate prepare ls; end
          EXTERNAL_NAME: NULL
       EXTERNAL_LANGUAGE: NULL
         PARAMETER_STYLE: SQL
        IS_DETERMINISTIC: NO
         SQL_DATA_ACCESS: CONTAINS SQL
                SQL_PATH: NULL
```

```
           SECURITY_TYPE: INVOKER
                 CREATED: 2022-11-07 06:02:06
            LAST_ALTERED: 2022-11-07 06:02:06
                SQL_MODE:
         ROUTINE_COMMENT:
                 DEFINER: root@localhost
    CHARACTER_SET_CLIENT: utf8
    COLLATION_CONNECTION: utf8_general_ci
      DATABASE_COLLATION: latin1_swedish_ci
*************************** 6. row ***************************
           SPECIFIC_NAME: DropGeometryColumn
         ROUTINE_CATALOG: def
          ROUTINE_SCHEMA: mysql
            ROUTINE_NAME: DropGeometryColumn
            ROUTINE_TYPE: PROCEDURE
               DATA_TYPE:
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
          DTD_IDENTIFIER: NULL
            ROUTINE_BODY: SQL
      ROUTINE_DEFINITION: begin
  set @qwe= concat('ALTER TABLE ', t_schema, '.', t_name, ' DROP ', geometry_column); PREPARE ls from @qwe;
execute ls; deallocate prepare ls; end
           EXTERNAL_NAME: NULL
       EXTERNAL_LANGUAGE: NULL
         PARAMETER_STYLE: SQL
        IS_DETERMINISTIC: NO
         SQL_DATA_ACCESS: CONTAINS SQL
                SQL_PATH: NULL
```

```
           SECURITY_TYPE: INVOKER
                 CREATED: 2022-11-07 06:02:06
            LAST_ALTERED: 2022-11-07 06:02:06
                SQL_MODE:
         ROUTINE_COMMENT:
                 DEFINER: root@localhost
    CHARACTER_SET_CLIENT: utf8
    COLLATION_CONNECTION: utf8_general_ci
      DATABASE_COLLATION: latin1_swedish_ci
6 rows in set (0.001 sec)

MariaDB [ligue1]>
```

Fonctions Stockées

Vous allez maintenant créer une **Fonction Stockée** que vous utiliserez ultérieurement. La différence entre une procédure stockée et une fonction stockée est que cette dernière n'a pas besoin d'être appelée en utilisant la commande CALL. La fonction est appelée à partir d'une commande SQL. Cette fonction va vous identifier les équipes relégables en fin de saison.

Créez donc la fonction intégrée :

```
DELIMITER //
CREATE FUNCTION EST_RELEGABLE(idequipe INTEGER) RETURNS TINYINT
BEGIN
DECLARE relegable TINYINT default 0;
DECLARE id_courante INTEGER;
DECLARE fini TINYINT DEFAULT 0;
DECLARE cur1 CURSOR FOR SELECT id_equipe FROM equipe ORDER BY points ASC LIMIT 3;
DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET fini = 1;
OPEN cur1;
BOUCLE: REPEAT
FETCH cur1 INTO id_courante;
IF idequipe=id_courante THEN
```

```
SET relegable=1;
LEAVE BOUCLE;
END IF;
UNTIL fini END REPEAT BOUCLE;
CLOSE cur1;
RETURN relegable;
END//
DELIMITER ;
```

Vous obtiendrez un résultat similaire à celui-ci:

```
MariaDB [ligue1]> DELIMITER //
MariaDB [ligue1]> CREATE FUNCTION EST_RELEGABLE(idequipe INTEGER) RETURNS TINYINT
    -> BEGIN
    -> DECLARE relegable TINYINT default 0;
    -> DECLARE id_courante INTEGER;
    -> DECLARE fini TINYINT DEFAULT 0;
    -> DECLARE cur1 CURSOR FOR SELECT id_equipe FROM equipe ORDER BY points ASC LIMIT 3;
    -> DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET fini = 1;
    -> OPEN cur1;
    -> BOUCLE: REPEAT
    -> FETCH cur1 INTO id_courante;
    -> IF idequipe=id_courante THEN
    -> SET relegable=1;
    -> LEAVE BOUCLE;
    -> END IF;
    -> UNTIL fini END REPEAT BOUCLE;
    -> CLOSE cur1;
    -> RETURN relegable;
    -> END//
Query OK, 0 rows affected (0.038 sec)

MariaDB [ligue1]> DELIMITER ;
```

```
MariaDB [ligue1]>
```

A la fin de la saison, il conviendrait de saisir la commande suivante pour identifier les équipes relégables (NE SAISISSEZ PAS LES DEUX REQUETES SUIVANTES) :

```
> SELECT * FROM equipe WHERE EST_RELEGABLE(id_equipe)=1; [Entrée]
```

Ensuite il conviendrait de les supprimer :

```
> DELETE FROM equipe WHERE EST_RELEGABLE(id_equipe)=1; [Entrée]
```

Le code d'une fonction est stocké dans la table interne **INFORMATION_SCHEMA.ROUTINES**. Pour visualiser les fonctions stockées, utilisez la commande suivante :

```
MariaDB [ligue1]> SELECT * FROM information_schema.routines WHERE ROUTINE_TYPE='FUNCTION'\G
*************************** 1. row ***************************
           SPECIFIC_NAME: EST_RELEGABLE
         ROUTINE_CATALOG: def
          ROUTINE_SCHEMA: ligue1
            ROUTINE_NAME: EST_RELEGABLE
            ROUTINE_TYPE: FUNCTION
               DATA_TYPE: tinyint
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: 3
           NUMERIC_SCALE: 0
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
          DTD_IDENTIFIER: tinyint(4)
            ROUTINE_BODY: SQL
      ROUTINE_DEFINITION: BEGIN
DECLARE relegable TINYINT default 0;
DECLARE id_courante INTEGER;
```

```
DECLARE fini TINYINT DEFAULT 0;
DECLARE cur1 CURSOR FOR SELECT id_equipe FROM equipe ORDER BY points ASC LIMIT 3;
DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET fini = 1;
OPEN cur1;
BOUCLE: REPEAT
FETCH cur1 INTO id_courante;
IF idequipe=id_courante THEN
SET relegable=1;
LEAVE BOUCLE;
END IF;
UNTIL fini END REPEAT BOUCLE;
CLOSE cur1;
RETURN relegable;
END
           EXTERNAL_NAME: NULL
       EXTERNAL_LANGUAGE: NULL
         PARAMETER_STYLE: SQL
        IS_DETERMINISTIC: NO
         SQL_DATA_ACCESS: CONTAINS SQL
                SQL_PATH: NULL
           SECURITY_TYPE: DEFINER
                 CREATED: 2022-11-08 11:56:30
            LAST_ALTERED: 2022-11-08 11:56:30
                SQL_MODE:
STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
         ROUTINE_COMMENT:
                 DEFINER: root@localhost
    CHARACTER_SET_CLIENT: utf8
    COLLATION_CONNECTION: utf8_general_ci
      DATABASE_COLLATION: latin1_swedish_ci
1 row in set (0.001 sec)

MariaDB [ligue1]>
```

Notez que vous ne pouvez pas utiliser les commandes suivantes dans des routines stockées : LOCK TABLES, UNLOCK TABLES, ALTER VIEW, LOAD DATA/TABLE, PREPARE, EXECUTE, DEALLOCATE PREPARE.

Déclencheurs

Le support des déclencheurs (*triggers*) est inclus dans les versions de MySQL à partir de la version 5.0.2. Un déclencheur est un objet de base de données nommé, qui est associé à une table et qui s'active lorsqu'une **ACTION** de type **INSERT**, **DELETE** ou **UPDATE** sont effectuées sur une table. Un déclencheur peut être appelé avec une **POSITION**. La valeur de la POSITION est **BEFORE** ou **AFTER** l'**ACTION**.

Créez un trigger pour convertir la ville en majuscules lors de son insertion:

```
DELIMITER //
CREATE TRIGGER TGR_BI_EQUIPE BEFORE INSERT ON equipe FOR EACH ROW
BEGIN
SET new.ville = UPPER(new.ville);
END//
DELIMITER ;
```

Vous obtiendrez un résultat similaire à celui-ci:

```
MariaDB [ligue1]> DELIMITER //
MariaDB [ligue1]> CREATE TRIGGER TGR_BI_EQUIPE BEFORE INSERT ON equipe FOR EACH ROW
    -> BEGIN
    -> SET new.ville = UPPER(new.ville);
    -> END//
Query OK, 0 rows affected (0.055 sec)

MariaDB [ligue1]> DELIMITER ;
MariaDB [ligue1]>
```

Insérez une nouvel enregistrement:

```
MariaDB [ligue1]> INSERT INTO equipe (nom, stade, ville) VALUES ('Racing Club Strasbourg', 'La Meinau',
'strasbourg');
Query OK, 1 row affected (0.000 sec)

MariaDB [ligue1]>
```

```
MariaDB [ligue1]> SELECT nom, stade, ville FROM equipe;
+------------------------+------------------+------------+
| nom                    | stade            | ville      |
+------------------------+------------------+------------+
| FC Mandriva            | Parc des Princes | Paris      |
| Debian AC              | Yankee Stadium   | New York   |
| Vista FC               | Qwest Field      | Redmond    |
| Racing Club Strasbourg | La Meinau        | STRASBOURG |
+------------------------+------------------+------------+
4 rows in set (0.000 sec)

MariaDB [ligue1]>
```

Créez un trigger pour faire le total des buts:

```
DELIMITER //
CREATE TRIGGER TGR_BU_RENCONTRE BEFORE UPDATE ON rencontre FOR EACH ROW
BEGIN
IF old.score_domicile IS NULL and new.score_domicile IS NOT NULL THEN
UPDATE equipe SET buts = buts + new.score_domicile WHERE id_equipe=new.id_domicile;
END IF;
IF  old.score_visiteurs IS NULL and new.score_visiteurs IS NOT NULL THEN
UPDATE equipe SET buts = buts + new.score_visiteurs WHERE id_equipe=new.id_visiteurs;
END IF;
END//
DELIMITER ;
```

Vous obtiendrez un résultat similaire à celui-ci :

```
MariaDB [ligue1]> DELIMITER //
MariaDB [ligue1]> CREATE TRIGGER TGR_BU_RENCONTRE BEFORE UPDATE ON rencontre FOR EACH ROW
    -> BEGIN
    -> IF old.score_domicile IS NULL and new.score_domicile IS NOT NULL THEN
    -> UPDATE equipe SET buts = buts + new.score_domicile WHERE id_equipe=new.id_domicile;
    -> END IF;
    -> IF  old.score_visiteurs IS NULL and new.score_visiteurs IS NOT NULL THEN
    -> UPDATE equipe SET buts = buts + new.score_visiteurs WHERE id_equipe=new.id_visiteurs;
    -> END IF;
    -> END//
Query OK, 0 rows affected (0.053 sec)

MariaDB [ligue1]> DELIMITER ;
MariaDB [ligue1]>
```

Appelez la procédure pour inscrire un résultat:

```
MariaDB [ligue1]> CALL SAISIR_RESULTAT(1, 2, 2, 1);
Query OK, 4 rows affected (0.001 sec)

MariaDB [ligue1]>
```

Appelez la procédure une deuxième fois pour inscrire un autre résultat:

```
MariaDB [ligue1]> CALL SAISIR_RESULTAT(2, 1, 2, 1);
Query OK, 4 rows affected (0.000 sec)

MariaDB [ligue1]>
```

Examinez les résultats obtenus :

```
MariaDB [ligue1]> SELECT * FROM ligue1.equipe;
+-----------+----------------------+------------------+------------+--------+------+
```

```
| id_equipe | nom                     | stade            | ville      | points | buts |
+-----------+-------------------------+------------------+------------+--------+------+
|         1 | FC Mandriva             | Parc des Princes | Paris      |      3 |    3 |
|         2 | Debian AC               | Yankee Stadium   | New York   |      3 |    3 |
|         3 | Vista FC                | Qwest Field      | Redmond    |      0 |    0 |
|         4 | Racing Club Strasbourg | La Meinau        | STRASBOURG |      0 |    0 |
+-----------+-------------------------+------------------+------------+--------+------+
4 rows in set (0.000 sec)

MariaDB [ligue1]>
```

L'utilisation des déclencheurs est soumise à des limites :

- Limitation à un seul trigger par table, par ACTION et par POSITION.
- Uniquement triggers FOR EACH ROW (déclencheurs niveau ligne).
- Pas de déclencheurs sur des événements système autres que des requêtes SQL,
- Les déclencheurs ne peuvent pas être mis sur les tables de la base mysql,
- Les déclencheurs ne sont pas déclenchés par les ACTIONS de clés étrangères.

Vues

Les vues, y compris les vues modifiables, sont disponibles dans les versions binaires depuis la version 5.0.1.

- Création ou modification de vues avec les commandes CREATE VIEW ou ALTER VIEW
- Destruction de vues avec DROP VIEW
- Affichage des méta-données de vues avec SHOW CREATE VIEW

Ajoutez d'abord des champs à notre table **equipe** pour stocker le nom des entraîneurs ainsi que leurs numéros de téléphone :

```
ALTER TABLE equipe
ADD entraineur varchar(100) default 'inconnu',
ADD tel_entraineur varchar(20) default 'inconnu';
```

Vous obtiendrez un résultat similaire à celui-ci :

```
MariaDB [ligue1]> ALTER TABLE equipe
    -> ADD entraineur varchar(100) default 'inconnu',
    -> ADD tel_entraineur varchar(20) default 'inconnu';
Query OK, 4 rows affected (0.025 sec)
Records: 4  Duplicates: 0  Warnings: 0

MariaDB [ligue1]>
```

Mettez à jour la table:

```
MariaDB [ligue1]> UPDATE equipe SET entraineur='Ricardo GOMES', tel_entraineur='06-56-56-56-56' WHERE
id_equipe=1;
Query OK, 1 row affected (0.000 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [ligue1]> UPDATE equipe SET entraineur='Gérard Houllier', tel_entraineur='06-57-57-57-57' WHERE
id_equipe=2;
Query OK, 1 row affected (0.000 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [ligue1]>
```

Consultez le résultat :

```
MariaDB [ligue1]> SELECT * FROM ligue1.equipe;
+-----------+------------------------+------------------+------------+--------+------+------------------+--------
--------+
| id_equipe | nom                    | stade            | ville      | points | buts | entraineur       |
tel_entraineur |
+-----------+------------------------+------------------+------------+--------+------+------------------+--------
--------+
|         1 | FC Mandriva            | Parc des Princes | Paris      |      3 |    3 | Ricardo GOMES    |
```

```
06-56-56-56-56 |
|         2 | Debian AC              | Yankee Stadium   | New York   |      3 |    3 | Gérard Houllier  |
06-57-57-57-57 |
|         3 | Vista FC               | Qwest Field      | Redmond    |      0 |    0 | inconnu          | inconnu
|
|         4 | Racing Club Strasbourg | La Meinau        | STRASBOURG |      0 |    0 | inconnu          | inconnu
|
+-----------+------------------------+------------------+------------+--------+------+------------------+--------
--------+
4 rows in set (0.000 sec)

MariaDB [ligue1]>
```

Créez une vue qui **ne montre pas** le numéro de téléphone:

```
MariaDB [ligue1]> CREATE VIEW V_EQUIPE AS SELECT id_equipe, nom, stade, ville, points, buts, entraineur FROM
equipe;
Query OK, 0 rows affected (0.062 sec)

MariaDB [ligue1]>
```

Le code d'un view est stocké dans la table interne **INFORMATION_SCHEMA.VIEWS** :

```
MariaDB [ligue1]> SELECT * FROM information_schema.views\G
*************************** 1. row ***************************
       TABLE_CATALOG: def
        TABLE_SCHEMA: ligue1
          TABLE_NAME: V_EQUIPE
     VIEW_DEFINITION: select `ligue1`.`equipe`.`id_equipe` AS `id_equipe`,`ligue1`.`equipe`.`nom` AS
`nom`,`ligue1`.`equipe`.`stade` AS `stade`,`ligue1`.`equipe`.`ville` AS `ville`,`ligue1`.`equipe`.`points` AS
`points`,`ligue1`.`equipe`.`buts` AS `buts`,`ligue1`.`equipe`.`entraineur` AS `entraineur` from `ligue1`.`equipe`
        CHECK_OPTION: NONE
        IS_UPDATABLE: YES
             DEFINER: root@localhost
```

```
       SECURITY_TYPE: DEFINER
CHARACTER_SET_CLIENT: utf8
COLLATION_CONNECTION: utf8_general_ci
          ALGORITHM: UNDEFINED
1 row in set (0.004 sec)

MariaDB [ligue1]>
```

Donnez les droits sur cette vue à user1:

```
MariaDB [ligue1]> GRANT SELECT, INSERT, DELETE, UPDATE ON V_EQUIPE TO user1@localhost IDENTIFIED BY 'user1';
Query OK, 0 rows affected (0.000 sec)

MariaDB [ligue1]>
```

Déconnectez-vous et reconnectez-vous en tant qu'user1:

```
MariaDB [ligue1]> exit
Bye
[root@centos8 ~]# mysql -uuser1 -p
Enter password: user1
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 10
Server version: 10.3.28-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Notez que user1 ne voit que les bases de données **information_schema** et **ligue1** :

```
MariaDB [(none)]> show databases;
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ligue1             |
+--------------------+
2 rows in set (0.000 sec)

MariaDB [(none)]>
```

Changez de base de données:

```
MariaDB [(none)]> use ligue1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [ligue1]>
```

Ensuite, visualisez la vue V_EQUIPE:

```
MariaDB [ligue1]> SELECT * FROM V_EQUIPE;
+-----------+------------------------+------------------+------------+--------+------+------------------+
| id_equipe | nom                    | stade            | ville      | points | buts | entraineur       |
+-----------+------------------------+------------------+------------+--------+------+------------------+
|         1 | FC Mandriva            | Parc des Princes | Paris      |      3 |    3 | Ricardo GOMES    |
|         2 | Debian AC              | Yankee Stadium   | New York   |      3 |    3 | Gérard Houllier  |
|         3 | Vista FC               | Qwest Field      | Redmond    |      0 |    0 | inconnu          |
|         4 | Racing Club Strasbourg | La Meinau        | STRASBOURG |      0 |    0 | inconnu          |
+-----------+------------------------+------------------+------------+--------+------+------------------+
4 rows in set (0.000 sec)

MariaDB [ligue1]>
```

Notez cependant qu'user1 n'a pas accès à la table **equipe** :

```
MariaDB [ligue1]> SELECT * FROM equipe;
ERROR 1142 (42000): SELECT command denied to user 'user1'@'localhost' for table 'equipe'
MariaDB [ligue1]>
```

Déconnectez-vous et reconnectez-vous en tant que root :

```
MariaDB [ligue1]> exit
Bye
[root@centos8 ~]# mysql -uroot -p
Enter password: fenestros
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.3.28-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Planificateur d'Evénements

Pour visualiser l'état du planificateur d'évènements, utilisez la requête suivante :

```
MariaDB [(none)]> SHOW VARIABLES LIKE 'event_scheduler';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| event_scheduler | OFF   |
+-----------------+-------+
```

```
1 row in set (0.001 sec)

MariaDB [(none)]>
```

Pour activer le planificateur d'évènements, modifiez la valeur de la variable **event_scheduler** :

```
MariaDB [(none)]> SET GLOBAL event_scheduler = 1;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> SHOW VARIABLES LIKE 'event_scheduler';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| event_scheduler | ON    |
+-----------------+-------+
1 row in set (0.001 sec)

MariaDB [(none)]>
```

L'état du planificateur d'évènements, **Waiting on empty queue** peut être visualisé en saisissant la requête suivante :

```
MariaDB [(none)]> SHOW PROCESSLIST;
+----+-----------------+-----------+------+---------+------+-------------------------+------------------+-------
---+
| Id | User            | Host      | db   | Command | Time | State                   | Info             |
Progress |
+----+-----------------+-----------+------+---------+------+-------------------------+------------------+-------
---+
|  2 | system user     |           | NULL | Daemon  | NULL | InnoDB purge worker     | NULL             |
0.000 |
|  1 | system user     |           | NULL | Daemon  | NULL | InnoDB purge coordinator | NULL             |
0.000 |
|  3 | system user     |           | NULL | Daemon  | NULL | InnoDB purge worker     | NULL             |
0.000 |
```

```
|  4 | system user     |           | NULL | Daemon  | NULL | InnoDB purge worker      | NULL             |
0.000 |
|  5 | system user     |           | NULL | Daemon  | NULL | InnoDB shutdown handler  | NULL             |
0.000 |
| 12 | event_scheduler | localhost | NULL | Daemon  |  193 | Waiting on empty queue   | NULL             |
0.000 |
| 13 | root            | localhost | NULL | Query   |    0 | Init                     | SHOW PROCESSLIST |
0.000 |
+----+-----------------+-----------+------+---------+------+--------------------------+------------------+-------
---+
7 rows in set (0.000 sec)

MariaDB [(none)]>
```
