Mysql Query Rewriter

Pour installer le plug-in de réécriture de requête Rewriter, exécutez install_rewriter.sql situé dans le répertoire de share de votre installation MySQL.

```
mysql -h localhost -u root --protocol=tcp -p < /home/pilou/Formation/mysql-8.0.20-linux-glibc2.12-x86_64/share/install_rewriter.sql
```

L'installation se constate :

```
mysql> SELECT * FROM mysql.plugin;
+----------+-------------+
| name     | dl          |
+----------+-------------+
| rewriter | rewriter.so |
+----------+-------------+
1 row in set (0.00 sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'rewriter%';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| rewriter_enabled | ON    |
| rewriter_verbose | 1     |
+------------------+-------+
2 rows in set (0.00 sec)
```

La table rewrite_rules est une table persistante pour le plugin query_rewrite:

```
mysql> SHOW CREATE TABLE query_rewrite.rewrite_rules\G
*************************** 1. row ***************************
       Table: rewrite_rules
Create Table: CREATE TABLE `rewrite_rules` (
  `id` int NOT NULL AUTO_INCREMENT,
```

```
  `pattern` varchar(5000) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL,
  `pattern_database` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  `replacement` varchar(5000) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL,
  `enabled` enum('YES','NO') CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL DEFAULT 'YES',
  `message` varchar(1000) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  `pattern_digest` varchar(64) DEFAULT NULL,
  `normalized_pattern` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.01 sec)
```

Vous pouvez activer le plugin soit via le my.ini soit

```
mysql> SET GLOBAL rewriter_enabled = ON;


mysql> SET GLOBAL rewriter_enabled = OFF;
```

Le plug-in de réécriture ne fonctionne qu'avec les instructions SELECT.

Commencons avec un exemple:

```
mysql> INSERT INTO query_rewrite.rewrite_rules (pattern, replacement) VALUES('SELECT ?',
'SELECT ? + 1');
```

Il est possible de les afficher sachant quel ne sont pas "compilé"

```
mysql> SELECT * FROM query_rewrite.rewrite_rules\G
*************************** 1. row ***************************
                id: 1
           pattern: SELECT ?
  pattern_database: NULL
       replacement: SELECT ? + 1
           enabled: YES
           message: NULL
    pattern_digest: NULL
normalized_pattern: NULL
1 row in set (0.00 sec)
```

Puis nous allons les compiler:

```
mysql> CALL query_rewrite.flush_rewrite_rules();
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM query_rewrite.rewrite_rules\G
*************************** 1. row ***************************
                id: 1
           pattern: SELECT ?
  pattern_database: NULL
       replacement: SELECT ? + 1
           enabled: YES
           message: NULL
    pattern_digest: d1b44b0c19af710b5a679907e284acd2ddc285201794bc69a2389d77baedddae
normalized_pattern: select ?
1 row in set (0.00 sec)
```

La requete est ainsi mise :

```
mysql> select 1;
+-------+
| 1 + 1 |
+-------+
|     2 |
+-------+
1 row in set, 1 warning (0.00 sec)
```

Usage 1: Optimization

Pour des raisons de performances, il est parfois souhaitable de réécrire une requete sans pouvoir le faire

-> SELECT count(distinct emp_no) FROM employees.employees INNER JOIN employees.salaries USING(emp_no) WHERE DATEDIFF(to_date, from_date) < {integer};

<= SELECT count(emp_no) FROM employees.employees WHERE emp_no IN (SELECT emp_no FROM employees.salaries WHERE DATEDIFF(to_date, from_date) < {integer});

```
INSERT INTO query_rewrite.rewrite_rules
(
pattern,
replacement
)
VALUES
(
'SELECT count(distinct emp_no) FROM employees.employees INNER JOIN employees.salaries
USING(emp_no) WHERE DATEDIFF(to_date, from_date) < ?',
'SELECT count(emp_no) FROM employees.employees WHERE emp_no IN ( SELECT emp_no FROM
employees.salaries WHERE DATEDIFF(to_date, from_date) < ?)'
);


CALL query_rewrite.flush_rewrite_rules();
```

Usage 2: Optimization

Il est possible de rajouter des optimizations en commentaires dans le SQL de MySQL. Par exemple,

-> SELECT count(distinct emp_no) FROM employees.employees INNER JOIN employees.salaries USING(emp_no) WHERE salary = {integer};

<= SELECT /*+ MAX_EXECUTION_TIME(10000)*/ count(distinct emp_no) FROM employees.employees INNER JOIN employees.salaries USING(emp_no) WHERE salary = {integer};

Revision #2
Created 30 May 2020 06:31:30 by ggpilou2
Updated 9 June 2020 06:20:54 by ggpilou2