MySQL创建定时任务

一、前言

自MySQL5.1.6起，增加了一个非常有特色的功能-事件调度器（Event Scheduler），可以用做定时执行某些特定任务（例如：删除记录、对数据进行汇总、数据备份等等），来取代原先只能由操作系统的计划任务来执行的工作。更值得一提的是MySQL的事件调度器可以精确到每秒钟执行一个任务，而操作系统的计划任务（如：Linux的cron或Windows下的任务计划）只能精确到每分钟执行一次。对于一些对数据实时性要求比较高的应用（例如：股票、赔率、比分等）就非常适合。

事件调度器有时也可以称为临时触发器（temporal triggers），因为事件调度器是基于特定时间周期触发来执行某些任务，而触发器（Triggers）是基于某个表所产生的事件触发的，区别也就在这里。

1、在使用这个功能之前必须确保event_scheduler已开启，可执行

SET GLOBAL event_scheduler = 1;
---或我们可以在配置my.cnf文件 中加上 event_scheduler = 1

或

SET GLOBAL event_scheduler = ON;
来开启，也可以直接在启动命令加上"--event_scheduler=1"，例如：
mysqld ... --event_scheduler=1

注：将事件计划关闭：SET GLOBAL event_scheduler = 0;

2、要查看当前是否已开启事件调度器，可执行如下SQL：
SHOW VARIABLES LIKE 'event_scheduler';
或
SELECT @@event_scheduler;
或
SHOW PROCESSLIST;

注：

（1）关闭事件任务： ALTER EVENT eventName ON COMPLETION PRESERVE DISABLE;

（2）开启事件任务：ALTER EVENT eventName ON COMPLETION PRESERVE ENABLE;

（3）查看事件任务：SHOW EVENTS ;

二、创建事件

先来看一下他的语法：

```
CREATE EVENT [IFNOT EXISTS] event_name
        ONSCHEDULE schedule
        [ONCOMPLETION [NOT] PRESERVE]
        [ENABLE | DISABLE]
        [COMMENT 'comment']
        DO sql_statement;
```

```
schedule:
    AT TIMESTAMP [+ INTERVAL INTERVAL]
    | EVERY INTERVAL [STARTS TIMESTAMP] [ENDS TIMESTAMP]

   INTERVAL:
    quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |
    WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |
    DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
```

1) 首先来看一个简单的例子来演示每秒插入一条记录到数据表

```
USE td2;

CREATE TABLE aaa(timeline TIMESTAMP);
CREATE EVENT e_test_insert
ON SCHEDULE EVERY 1 SECOND
DO INSERT aaa VALUE(CURRENT_TIMESTAMP);
```

等待3秒之后，再执行查询看看

```
SELECT * FROM aaa;
```

timeline
2018-09-17 18:06:29
2018-09-17 18:06:30
2018-09-17 18:06:31
2018-09-17 18:06:32
2018-09-17 18:06:33
2018-09-17 18:06:34

2) 5天后清空test表：

```
CREATE EVENT e_test
ON SCHEDULE AT CURRENT_TIMESTAMP+INTERVAL 5 DAY
DO TRUNCATE TABLE aaa;
```

3) 2007年7月20日12点整清空test表：

```
CREATE EVENT e_test1
ON SCHEDULE AT TIMESTAMP '2018-09-17 18:16:00'
DO TRUNCATE TABLE aaa;
```

4) 每天定时清空test表：

```
CREATE EVENT e_test2
ON SCHEDULE EVERY 1 DAY
DO TRUNCATE aaa;
```

5) 5天后开启每天定时清空test表：

```
CREATE EVENT e_test3
ON SCHEDULE EVERY 1 DAY
STARTS CURRENT_TIMESTAMP+INTERVAL 5 DAY
DO TRUNCATE aaa;
```

6) 每天定时清空test表，5天后停止执行：

```
CREATE EVENT e_test4
ON SCHEDULE EVERY 1 DAY
ENDS CURRENT_TIMESTAMP+INTERVAL 5 DAY
DO TRUNCATE aaa;
```

7) 5天后开启每天定时清空test表，一个月后停止执行：

```
CREATE EVENT e_test5
ON SCHEDULE EVERY 1 DAY
STARTS CURRENT_TIMESTAMP+INTERVAL 5 DAY
ENDS CURRENT_TIMESTAMP+INTERVAL 1 MONTH
DO TRUNCATE aaa;
```

[ON COMPLETION [NOT] PRESERVE]可以设置这个事件是执行一次还是持久执行，默认为NOT PRESERVE。

8) 每天定时清空test表(只执行一次，任务完成后就终止该事件)：

```
CREATE EVENT e_test6
ON SCHEDULE EVERY 1 DAY
ON COMPLETION NOT PRESERVE
DO TRUNCATE aaa;
```

[ENABLE | DISABLE]可以设置该事件创建后状态是否开启或关闭，默认为ENABLE。
[COMMENT 'comment']可以给该事件加上注释。

三、修改事件

先来看一下他的语法：

```
ALTER EVENT event_name
         [ONSCHEDULE schedule]
         [RENAME TOnew_event_name]
         [ON COMPLETION [NOT] PRESERVE]
         [COMMENT 'comment']
         [ENABLE | DISABLE]
         [DO sql_statement]
```

1) 临时关闭事件

```
ALTER EVENT e_test DISABLE;
```

2) 开启事件

```
ALTER EVENT e_test ENABLE;
```

3) 将每天清空test表改为5天清空一次：

```
ALTER EVENT e_test
ON SCHEDULE EVERY 5 DAY;
```

四、删除事件

先来看一下他的语法：

```
DROP EVENT [IF EXISTS] event_name
```

例如删除前面创建的e_test事件

```
DROP EVENT e_test;
```

当然前提是这个事件存在，否则会产生ERROR 1513 (HY000): Unknown event错误，因此最好加上IF EXISTS

```
DROP EVENT IF EXISTS e_test;
```

注意：如果你将event执行了Alter event event_name disable.那么当你重新启动mysql服务器后，该event将被删除（测试版本：5.1.30）

备注：在event事件中：ON SCHEDULE 计划任务，有两种设定计划任务的方式：

（1）AT 时间戳，用来完成单次的计划任务

（2）EVERY 时间（单位）的数量实践单位[STARTS 时间戳] [ENDS时间戳]，用来完成重复的计划任务。

在两种计划任务中，时间戳可以是任意的TIMESTAMP 和DATETIME 数据类型，时间戳需要大于当前时间。
在重复的计划任务中，时间（单位）的数量可以是任意非空（Not Null）的整数式，时间单位是关键词：YEAR，MONTH，DAY，HOUR，MINUTE 或者SECOND。

提示：其他的时间单位也是合法的如：QUARTER, WEEK, YEAR_MONTH,DAY_HOUR,DAY_MINUTE,DAY_SECOND,HOUR_MINUTE,HOUR_SECOND, MINUTE_SECOND，不建议使用这些不标准的时间单位。

[ON COMPLETION [NOT] PRESERVE]：ON COMPLETION参数表示"当这个事件不会再发生的时候"，即当单次计划任务执行完毕后或当重复性的计划任务执行到了ENDS阶段。而PRESERVE的作用是使事件在执行完毕后不会被Drop掉，建议使用该参数，以便于查看EVENT具体信息。

五、应用案例

本案例是利用 event scheduler 的特性，每秒钟调用一次存储过程，用于判断 SLAVE 是否正常运行，如果发现 SLAVE 关闭了，忽略 0 次错误，然后重新启动 SLAVE。

1）首先创建存储过程

```
CREATE PROCEDURE Slave_Monitor()
BEGIN
SELECT VARIABLE_VALUE INTO @SLAVE_STATUS
FROM information_schema.GLOBAL_STATUS
WHERE VARIABLE_NAME='SLAVE_RUNNING';
IF('ON'!=@SLAVE_STATUS) THEN
SET GLOBAL SQL_SLAVE_SKIP_COUNTER=0;
SLAVE START;
END IF;
END;
```

由于存储过程中无法调用类似 SHOW SLAVE STATUS 这样的语句，因此无法得到确切的复制错误信息和错误代码，不能进一步的处理 SLAVE 停止的各种情况。

2）接着，创建任务

```
CREATE EVENT IF NOT EXISTS Slave_Monitor
ON SCHEDULE EVERY 5 SECOND
ON COMPLETION PRESERVE
DO
CALL Slave_Monitor();
```

创建了一个任务，每 5秒钟执行一次，任务结束后依旧保留该任务，而不是删除。当然了，在本例中的任务不会结束，除非将它手动禁止了。

如果在运行中想要临时关闭一下某个任务，执行 ALTER EVENT 语句即可：

```
ALTER EVENT Slave_Monitor ON COMPLETION PRESERVE DISABLE;  ##关闭事件
```

```
ALTER EVENT Slave_Monitor ON COMPLETION PRESERVE ENABLE;  ##开启事件
```