**Working with MySQL Scheduled Event**

**Summary**: About MySQL event scheduler and how to create MySQL events to automate database tasks.

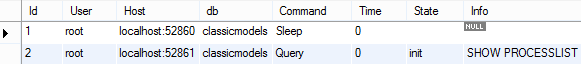
A MySQL event is a task that runs based on a predefined schedule therefore sometimes it is referred to as a scheduled event. MySQL event is also known as “temporal trigger” because it is triggered by time, not by table update like a [trigger](http://www.mysqltutorial.org/mysql-triggers.aspx). A MySQL event is similar to a cron job in UNIX or a task scheduler in Windows.

You can use MySQL events in many cases such as optimizing database tables, cleaning up logs, archiving data, or generate complex reports during off-peak time.

**MySQL event scheduler configuration**

MySQL uses a special thread called event schedule thread to execute all scheduled events. You can see the status of event scheduler thread by executing the following command:

|  |  |
| --- | --- |
|  | SHOW PROCESSLIST; |



By default, the event scheduler thread is not enabled. To enable and start the event scheduler thread, you need to execute the following command:

|  |  |
| --- | --- |
|  | SET GLOBAL event\_scheduler = ON; |

Now to see the status of event scheduler thread, you execute the SHOW PROCESSLIST command again.

|  |  |
| --- | --- |
|  | SHOW PROCESSLIST; |



To disable and stop the event the event scheduler thread, you execute the SET GLOBAL command with value of the event\_scheduler is OFF:

|  |  |
| --- | --- |
|  | SET GLOBAL event\_scheduler = OFF; |

**Creating new MySQL events**

Creating an event is similar to creating other database objects such as stored procedures or triggers. An event is a named object that contains SQL statements.

A [stored procedure](http://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx) is only executed when it is invoked directly; a [trigger](http://www.mysqltutorial.org/mysql-triggers.aspx) is executed when an event associated with a table such as [an insert](http://www.mysqltutorial.org/mysql-insert-statement.aspx), [update](http://www.mysqltutorial.org/mysql-update-data.aspx), or [delete](http://www.mysqltutorial.org/mysql-delete-statement.aspx) event occurs while an event can be executed at once or more regular intervals.

To create and schedule a new event, you use the  CREATE EVENT statement as follows:

|  |  |
| --- | --- |
|  | CREATE EVENT [IF NOT EXIST]  event\_name  ON SCHEDULE schedule  DO  event\_body |

Let’s examine the statement in more detail.

* First, you specify the event name after the  CREATE EVENT clause. The event name must be unique within a database schema.
* Second, you put a schedule after the  ON SCHEDULE clause. If the event is a one-time event, you use the syntax:AT timestamp [+ INTERVAL] If the event is a recurring event, you use the EVERY clause:EVERY interval STARTS timestamp [+INTERVAL] ENDS timestamp [+INTERVAL]
* Third, you place the SQL statements after the DO keyword. It is important to notice that you can call a stored procedure inside the body of the event. In case you have compound SQL statements, you can wrap them in a  BEGIN END block.

Let’s look at few examples of creating events to understand the syntax above.

To create and schedule a new one-time event that inserts a message into a table called messages you do the following steps:.

First, create a new table named messages by using the CREATE TABLE statement as follows:

|  |  |
| --- | --- |
|  | CREATE TABLE IF NOT EXISTS messages (      id INT PRIMARY KEY AUTO\_INCREMENT,      message VARCHAR(255) NOT NULL,      created at DATETIME NOT NULL  ); |

Second, create an event by using the CREATE EVENT statement:

|  |  |
| --- | --- |
|  | CREATE EVENT IF NOT EXISTS test\_event\_01  ON SCHEDULE AT CURRENT\_TIMESTAMP  DO    INSERT INTO messages(message,created\_at)    VALUES('Test MySQL Event 1',NOW()); |

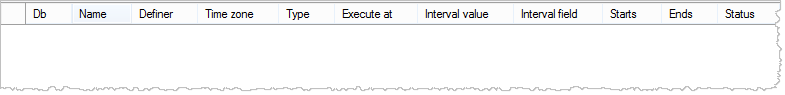
Third, check the messages table; you will see that we have 1 record. It means the event was executed when it is created.

|  |  |
| --- | --- |
|  | SELECT \* FROM messages; |

mysql event log entry

To shows all events of a database schema, you use the following statement:

|  |  |
| --- | --- |
|  | SHOW EVENTS FROM classicmodels; |



We don’t see any row returned because an event is automatically dropped when it is expired. In our case, it is a one-time event and expired when its execution completed.

To change this behavior, you can use the ON COMPLETION PRESERVE clause. The following statement creates another one-time event that is executed after its creation time 1 minute and not dropped after execution.

|  |  |
| --- | --- |
|  | CREATE EVENT test\_event\_02  ON SCHEDULE AT CURRENT\_TIMESTAMP + INTERVAL 1 MINUTE  ON COMPLETION PRESERVE  DO     INSERT INTO messages(message,created\_at)     VALUES('Test MySQL Event 2',NOW()); |

Wait for 1 minute, check the messages table, another record was added:

|  |  |
| --- | --- |
|  | SELECT \* FROM messages; |

message table records

If we execute the  SHOW EVENTS statement again, we see the event is there because the effect of the  ON COMPLETION PRESERVE clause:

|  |  |
| --- | --- |
|  | SHOW EVENTS FROM classicmodels; |

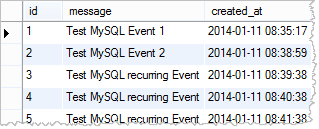
mysql event list

The following statement creates a recurring event that executes every minute and is expired in 1 hour from its creation time:

|  |  |
| --- | --- |
|  | CREATE EVENT test\_event\_03  ON SCHEDULE EVERY 1 MINUTE  STARTS CURRENT\_TIMESTAMP  ENDS CURRENT\_TIMESTAMP + INTERVAL 1 HOUR  DO     INSERT INTO messages(message,created\_at)     VALUES('Test MySQL recurring Event',NOW()); |

Notice that we used STARTS and ENDS clauses to define expiration period for the event. You can test this recurring event by waiting for few minutes and check the messages table.

|  |  |
| --- | --- |
|  | SELECT \* FROM messages; |



**Drop MySQL events**

To remove an existing event, you use the DROP EVENT statement as follows:

|  |  |
| --- | --- |
|  | DROP EVENT [IF EXIST] event\_name; |

Taken from - <http://www.mysqltutorial.org/mysql-triggers/working-mysql-scheduled-event/>