MySQL Events

Topic 4 Lesson 6
Using time to determine actions in the database

MySQL events

- MySQL Events are tasks that run according to a time schedule.
- An event performs a specific action
- This action consists of an SQL statement, which can be a compound statement in a BEGIN END block
- An event's timing can be either one-time or reoccurring
  - If reoccurring, it can state an interval that determines how often it gets run
  - Can specify a time window to state when the event is active
- An event is uniquely identified by its name and the schema to which it is assigned
- An event is executed with the privileges of its definer/author
- Errors and warnings from an event are written to the log

Structure of an event DB object

```
CREATE EVENT `event_name`
        ON SCHEDULE schedule
        [ON COMPLETION [NOT] PRESERVE]
    -- next option is specific to a CLUSTER DB
        [ENABLE | DISABLE | DISABLE ON SLAVE]
DO BEGIN
  -- event body
END

DROP EVENT `event_name`
ALTER EVENT `event_name`
```

Scheduler status

```
SHOW VARIABLES LIKE 'event_scheduler';
```

Turn the scheduler on if you plan to use the event scheduler

```
SET GLOBAL event_scheduler = ON;
```

SHOW EVENTS;
Determine the events that are scheduled for a database

```
SHOW EVENTS IN ap;
```

Options for a schedule

- Run once on a **specific date/time:**
  AT ‘YYYY-MM-DD HH:MM.SS’
  e.g. AT ‘2011-06-01 02:00.00’
- Run once **after a specific period has elapsed:**
  AT CURRENT_TIMESTAMP + INTERVAL n [HOUR|MONTH|WEEK|DAY|MINUTE]
  e.g. AT CURRENT_TIMESTAMP + INTERVAL 1 DAY
- Run at **specific intervals forever:**
  EVERY n [HOUR|MONTH|WEEK|DAY|MINUTE]
  e.g. EVERY 1 DAY
- Run at **specific intervals during a specific period:**
  EVERY n [HOUR|MONTH|WEEK|DAY|MINUTE] STARTS date ENDS date
  e.g. EVERY 1 DAY STARTS CURRENT_TIMESTAMP + INTERVAL 1 WEEK ENDS ‘2017-01-01 00:00.00’

Event example 1

```
DELIMITER $$

CREATE EVENT `archive_blogs`
            ON SCHEDULE EVERY 1 WEEK STARTS '2015-07-24 03:00:00'

DO BEGIN -- copy deleted posts

 INSERT INTO blog_archive (id, title, content)
   SELECT id, title, content FROM blog WHERE deleted = 1;
   -- copy associated audit records
 INSERT INTO audit_archive (id, blog_id, changetype, changetime)
   SELECT audit.id, audit.blog_id, audit.changetype, audit.changetime
     FROM audit JOIN blog ON audit.blog_id = blog.id WHERE blog.deleted = 1;
     -- remove deleted blogs and audit entries
  DELETE FROM blog WHERE deleted = 1;

 END $$


-- reset the delimiter
DELIMITER ;
```

Event example 2

A statement that creates a one-time event

```
DELIMITER //

CREATE EVENT one_time_delete_audit_rows
ON SCHEDULE AT NOW() + INTERVAL 1 MONTH
DO BEGIN
  DELETE FROM invoices_audit
  WHERE action_date < NOW() - INTERVAL 1 MONTH;
END//
```

Event example 3

A statement that creates a recurring event

```
CREATE EVENT monthly_delete_audit_rows
ON SCHEDULE EVERY 1 MONTH
STARTS '2015-06-01'
DO BEGIN
  DELETE FROM invoices_audit
  WHERE action_date < NOW() - INTERVAL 1 MONTH;
END//
```

Managing events

A statement that disables an event

```
ALTER EVENT monthly_delete_audit_rows DISABLE
```

A statement that enables an event

```
ALTER EVENT monthly_delete_audit_rows ENABLE
```

A statement that renames an event

```
ALTER EVENT one_time_delete_audit_rows
    RENAME TO one_time_delete_audits
```

A statement that drops an event

```
DROP EVENT monthly_delete_audit_rows
```

A statement that drops an event only if it exists

```
DROP EVENT IF EXISTS monthly_delete_audit_rows
```

Summary

- DB **function** allows you to create functions that can be called from an SQL statement. It is stored with a specific database and extends the definition of the database
- DB **procedure** allows you to bundle up a collection of SQL statements, store them with the database, and any application that has access to the DB can use them
- A **trigger** respond to changes in the database
  - Allows you to define constraints on the tables
- An **event** allows you to schedule tasks to be done by a calendar date or an interval
- A **prepared statem**ent allows you to specify the structure of a SQL statement and change literal values passed to the statement. It provides a layer of security to the system