Eventos en MySQL. Scheduled Events.

Eventos programados en MySQL

Los eventos MySQL son tareas que se ejecutan de acuerdo con un cronograma especificado. Por lo tanto, a veces los eventos de MySQL se denominan eventos programados.
Los eventos MySQL se denominan objeto que contiene una o más instrucciones SQL. Se almacenan en la base de datos y se ejecutan en uno o más intervalos.
Por ejemplo, puede crear un evento que optimice todas las tablas en la base de datos que se ejecuta a la 1:00 a.m.todos los domingos.
Los eventos MySQL también se conocen como "disparadores temporales"porque se disparan por tiempo, no por eventos DML como los disparadores normales. Los eventos de MySQL son similares a un cronjob en Linux o un programador de tareas en Windows.
MySQL Event Scheduler gestiona la programación y ejecución de eventos.
MySQL Events puede ser muy útil en muchos casos, como optimizar tablas de bases de datos, limpiar registros, archivar datos o generar informes complejos durante el período de menor actividad.

Configuración del planificador de eventos MySQL

MySQL usa un hilo especial llamado hilo del planificador de eventos para ejecutar todos los eventos programados. Puede ver el estado del hilo del planificador de eventos ejecutando el comando SHOW PROCESSLIST:

```
SHOW PROCESSLIST;
```

Si el programador de eventos no está habilitado, puede configurar la variable de sistema **event_scheduler** para habilitarlo e iniciarlo:

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

Ejecute nuevamente el comando SHOW PROCESSLIST para ver el estado del subproceso del planificador de eventos:

```
SHOW PROCESSLIST;
```

Para deshabilitar y detener el hilo del planificador de eventos, establezca la variable del sistema event_scheduler en OFF:

```
SET GLOBAL event_scheduler = OFF;
```

Crear nuevos eventos MySQL

La instrucción CREATE EVENT crea un nuevo evento. Aquí está la sintaxis básica de la instrucción CREATE EVENT:

```
CREATE EVENT [IF NOT EXIST] event_name
ON SCHEDULE schedule
DO
event_body
```

En esta sintaxis:
Primero, especifique el nombre del evento en el que desea crear con las palabras clave CREATE EVENT. Los nombres de los eventos deben ser únicos dentro de la misma base de datos.
En segundo lugar, especifique una programación después de las palabras clave ON SCHEDULE.
Si el evento es único, use la sintaxis:

```
AT timestamp [+ INTERVAL]
```

Si el evento es un evento recurrente, use la cláusula EVERY:

```
EVERY interval STARTS timestamp [+INTERVAL] ENDS timestamp [+INTERVAL]
```

En tercer lugar, coloque las instrucciones SQL después de la palabra clave **DO**. Y puede llamar a un **procedimiento almacenado** dentro del cuerpo de un evento. En caso de que tenga declaraciones compuestas, puede envolverlas en un bloque **BEGIN END**.

MySQL CREATE EVENT ejemplos

Veamos algunos ejemplos de creación de nuevos eventos.

A) Crear un ejemplo de evento único

El siguiente ejemplo crea un evento puntual que inserta una nueva fila en una tabla.
Primero, cree una nueva tabla llamada mensajes:

```
CREATE TABLE messages (
    id INT PRIMARY KEY AUTO_INCREMENT,
    message VARCHAR(255) NOT NULL,
    created_at DATETIME NOT NULL
);
```

Segundo, cree un evento usando la instrucción CREATE EVENT:

```
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());
```

Tercero, verifique la tabla de mensajes y verá una fila. Significa que el evento se ejecutó cuando se creó.

```
SELECT * FROM messages;
```

Para mostrar todos los eventos en la base de datos, utilice la siguiente instrucción:

```
SHOW EVENTS FROM classicmodels;
```

La salida no muestra ninguna fila porque el evento se descarta automáticamente cuando caduca. En este caso, es un evento único y expiró cuando se completó su ejecución.
Para mantener el evento después de que caduque, use la cláusula **ON COMPLETION PRESERVE**. La siguiente instrucción crea otro evento de una sola vez que se ejecuta después de su tiempo de creación 1 minuto y no se descarta después de la ejecución.

```
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());
```

Espere 1 minuto, verifique la tabla de mensajes, se agregó otro registro:

```
SELECT * FROM messages;
```

Si ejecuta la instrucción SHOW EVENTS nuevamente, verá que el evento está allí debido al efecto de la cláusula ON COMPLETION PRESERVE:

```
SHOW EVENTS FROM classicmodels;
```

Crear un ejemplo de evento recurrente

La siguiente instrucción crea un evento recurrente que se ejecuta cada minuto y expira en 1 hora desde su hora de creación:

```
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());
```

Tenga en cuenta que utilizamos las cláusulas **STARTS** y **ENDS** para definir el período de vencimiento del evento. Puede probar este evento recurrente esperando unos minutos y verificando la tabla de mensajes.

```
SELECT * FROM messages;
```

Sentencia DROP EVENT

Para eliminar un evento existente, use la instrucción **DROP EVENT** de la siguiente manera:

```
DROP EVENT [IF EXIST] event_name;
```

Por ejemplo, para descartar el evento **test_event_03**, utiliza la siguiente instrucción:

```
DROP EVENT IF EXIST test_event_03;
```