Triggers in SQL

1

Introduction to Triggers

- The SQL standard defines two types of triggers: row-level triggers and statement-level triggers.
- A row-level trigger is activated for each row that is inserted, updated, or deleted.
  - For example, if a table has 100 rows inserted, updated, or deleted, the trigger is automatically invoked 100 times for the 100 rows affected.

2

Introduction to Triggers

- A statement-level trigger is executed once for each transaction regardless of how many rows are inserted, updated, or deleted.
- MySQL supports only row-level triggers. It doesn't support statement-level triggers.

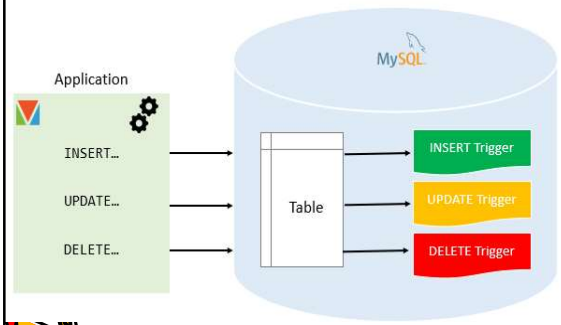
3

SQL Triggers

- Can be used to monitor a database and take a corrective action when a condition occurs
  - Examples:
    - Charge $10 overdraft fee if the balance of an account after a withdrawal transaction is less than $500
    - Limit the salary increase of an employee to no more than 5% raise

4

SQL Triggers



5

Advantages of triggers

- Triggers provide another way to check the integrity of data.
- Triggers handle errors from the database layer.
- Triggers give an alternative way to run scheduled tasks.
  - you don't have to wait for the scheduled events to run because the triggers are invoked automatically *before* or *after* a change is made to the data in a table.
- Triggers can be useful for auditing the data changes in tables.

6

Disadvantages of triggers

- Triggers can only provide extended validations, not all validations. For simple validations, you can use the NOT NULL, UNIQUE, CHECK and FOREIGN KEY constraints.
- Triggers can be difficult to troubleshoot because they execute automatically in the database, which may not invisible to the client applications.
- Triggers may increase the overhead of the MySQL Server.

7

SQL Triggers

```
CREATE TRIGGER trigger-name
   trigger-time trigger-event
   ON table-name
   FOR EACH ROW
        trigger-action;
```

trigger-time ∈ {BEFORE, AFTER}
trigger-event ∈ {INSERT,DELETE,UPDATE}

8

SQL Triggers

```
1 CREATE TRIGGER trigger_name
2 {BEFORE | AFTER} {INSERT | UPD
3 ATE| DELETE }
4 ON table_name FOR EACH ROW
  trigger_body;
```

9

SQL Triggers

- First, specify the name of the trigger that you want to create after the CREATE TRIGGER keywords. Note that the trigger name must be unique within a database.
- Next, specify the trigger action time which can be either BEFORE or AFTER which indicates that the trigger is invoked before or after each row is modified.

10

SQL Triggers

- Then, specify the operation that activates the trigger, which can be INSERT, UPDATE, or DELETE.
- After that, specify the name of the table to which the trigger belongs after the ON keyword.
- Finally, specify the statement to execute when the trigger activates.
  - If you want to execute multiple statements, you use the BEGIN END compound statement

11

SQL Triggers: An Example

- We want to create a trigger to update the total salary of a department when a new employee is hired

```
mysql> select * from employee;
+----+------+---------+--------+------------+------+
| id | name | superid | salary | bdate      | dno  |
+----+------+---------+--------+------------+------+
|  1 | john |       3 | 100000 | 1960-01-01 |    1 |
|  2 | mary |       3 |  50000 | 1964-12-01 |    3 |
|  3 | bob  |    NULL |  80000 | 1974-02-07 |    3 |
|  4 | tom  |       1 |  50000 | 1970-01-17 |    2 |
|  5 | bill |    NULL |   NULL | 1985-01-20 |    1 |
+----+------+---------+--------+------------+------+
5 rows in set (0.00 sec)

mysql> select * from deptsal;
+---------+-------------+
| dnumber | totalsalary |
+---------+-------------+
|       1 |      100000 |
|       2 |       50000 |
|       3 |      130000 |
+---------+-------------+
3 rows in set (0.00 sec)
```

12

SQL Triggers: Example

- Create a trigger to update the total salary of a department when a new employee is hired:

```
mysql> delimiter !
mysql> create trigger update_salary
    -> after insert on employee
    -> for each row
    -> begin
    ->       if new.dno is not null then
    ->          update deptsal
    ->          set totalsalary = totalsalary + new.salary
    ->          where dnumber = new.dno;
    ->       end if;
    -> end !
Query OK, 0 rows affected (0.06 sec)

mysql> delimiter ;
```

The keyword "new" refers to the new row inserted

Triggers: Example – Part 2

```
mysql> select * from deptsal;
+----------+-------------+
| dnumber  | totalsalary |
+----------+-------------+
|        1 |      100000 |
|        2 |       50000 |
|        3 |      130000 |
+----------+-------------+
3 rows in set (0.00 sec)

mysql> insert into employee values (6,'lucy',null,90000,'1981-01-01',1);
Query OK, 1 row affected (0.08 sec)

mysql> select * from deptsal;
+----------+-------------+
| dnumber  | totalsalary |
+----------+-------------+
|        1 |      190000 |
|        2 |       50000 |
|        3 |      130000 |
+----------+-------------+
3 rows in set (0.00 sec)

mysql> insert into employee values (7,'george',null,45000,'1971-11-11',null);
Query OK, 1 row affected (0.02 sec)

mysql> select * from deptsal;
+----------+-------------+
| dnumber  | totalsalary |
+----------+-------------+
|        1 |      190000 |
|        2 |       50000 |
|        3 |      130000 |
+----------+-------------+
3 rows in set (0.00 sec)

mysql> drop trigger update_salary;
Query OK, 0 rows affected (0.00 sec)
```

totalsalary increases by 90K

totalsalary did not change

Triggers: Example – Part 3

- A trigger to update the total salary of a department when an employee tuple is modified:

```
mysql> deliniter !
mysql> create trigger update_salary2
    -> after update on employee
    -> for each row
    -> begin
    ->       if old.dno is not null then
    ->          update deptsal
    ->          set totalsalary = totalsalary - old.salary
    ->          where dnumber = old.dno;
    ->       end if;
    ->       if new.dno is not null then
    ->          update deptsal
    ->          set totalsalary = totalsalary + new.salary
    ->          where dnumber = new.dno;
    ->       end if;
    -> end !
Query OK, 0 rows affected (0.06 sec)
```


Triggers: An Example – Part 4

```
mysql> deliniter ;
mysql> select * from employee;
+----+--------+---------+--------+------------+------+
| id | name   | superid | salary | bdate      | dno  |
+----+--------+---------+--------+------------+------+
|  1 | john   |       3 | 100000 | 1960-01-01 |    1 |
|  2 | mary   |       3 |  50000 | 1964-12-01 |    3 |
|  3 | bob    |    NULL |  80000 | 1974-02-07 |    3 |
|  4 | tom    |       1 |  50000 | 1970-01-17 |    2 |
|  5 | bill   |    NULL |   NULL | 1985-01-20 |    1 |
|  6 | lucy   |    NULL |  90000 | 1981-01-01 |    1 |
|  7 | george |    NULL |  45000 | 1971-11-11 | NULL |
+----+--------+---------+--------+------------+------+
7 rows in set (0.00 sec)

mysql> select * from deptsal;
+----------+-------------+
| dnumber  | totalsalary |
+----------+-------------+
|        1 |      190000 |
|        2 |       50000 |
|        3 |      130000 |
+----------+-------------+
3 rows in set (0.00 sec)

mysql> update employee set salary = 100000 where id = 6;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from deptsal;
+----------+-------------+
| dnumber  | totalsalary |
+----------+-------------+
|        1 |      200000 |
|        2 |       50000 |
|        3 |      130000 |
+----------+-------------+
3 rows in set (0.00 sec)
```


Triggers: Example – Part 5

- A trigger to update the total salary of a department when an employee tuple is deleted:

```
mysql> delimiter !
mysql> create trigger update_salary3
    -> before delete on employee
    -> for each row
    -> begin
    ->       if (old.dno is not null) then
    ->          update deptsal
    ->          set totalsalary = totalsalary - old.salary
    ->          where dnumber = old.dno;
    ->       end if;
    -> end !
Query OK, 0 rows affected (0.08 sec)

mysql> delimiter ;
```


Triggers: Example – Part 6

```
mysql> select * from employee;
+----+--------+---------+--------+------------+------+
| id | name   | superid | salary | bdate      | dno  |
+----+--------+---------+--------+------------+------+
|  1 | john   |       3 | 100000 | 1960-01-01 |    1 |
|  2 | mary   |       3 |  50000 | 1964-12-01 |    3 |
|  3 | bob    |    NULL |  80000 | 1974-02-07 |    3 |
|  4 | tom    |       1 |  50000 | 1970-01-17 |    2 |
|  5 | bill   |    NULL |   NULL | 1985-01-20 |    1 |
|  6 | lucy   |    NULL | 100000 | 1981-01-01 |    1 |
|  7 | george |    NULL |  45000 | 1971-11-11 | NULL |
+----+--------+---------+--------+------------+------+
7 rows in set (0.00 sec)

mysql> select * from deptsal;
+----------+-------------+
| dnumber  | totalsalary |
+----------+-------------+
|        1 |      200000 |
|        2 |       50000 |
|        3 |      130000 |
+----------+-------------+
3 rows in set (0.00 sec)
```

SQL Triggers: Another Example – Part 6

```
mysql> delete from employee where id = 6;
Query OK, 1 row affected (0.02 sec)

mysql> delete from employee where id = 7;
Query OK, 1 row affected (0.03 sec)

mysql> select * from deptsal;
+---------+-------------+
| dnumber | totalsalary |
+---------+-------------+
|       1 |      100000 |
|       2 |       50000 |
|       3 |      130000 |
+---------+-------------+
3 rows in set (0.00 sec)
```

19

SQL Triggers

- To list all the triggers you have created:

  mysql> show triggers;

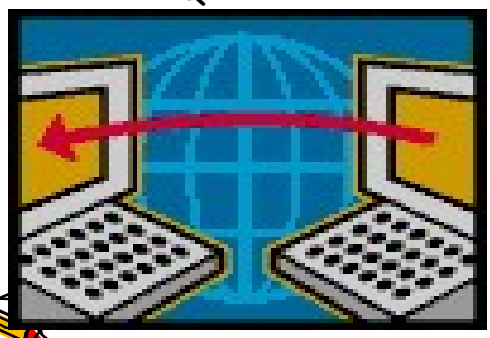
20

A Few Things to Note

- A given trigger can only have one event.
- If you have the same or similar processing that has to go on during insert and delete, then it's best to have that in a procedure or function and then call it from the trigger.
- A good naming standard for a trigger is <table_name>_event if you have the room for that in the name.
- Just like a function or a procedure, the trigger body will need a begin … end unless it is a single statement trigger.

21

Questions



22