Name	Adwait S Purao
UID no.	2021300101
Experiment No.	8

AIM:	To implement triggers in MySQL
Program 1	
PROBLEM STATEMENT :	Implement multiple triggers in hotel database
Theory :	A trigger in MySQL is a set of SQL statements that reside in a system catalog. **It is a special type of stored procedure that is invoked automatically in response to an event**. Each trigger is associated with a table, which is activated on any DML statement such as **INSERT, UPDATE**, or **DELETE**. A trigger is called a special procedure because it cannot be called directly like a stored procedure. The main difference between the trigger and procedure is that a trigger is called automatically when a data modification event is made against a table. In contrast, a stored procedure must be called explicitly. Generally, **triggers are of two types** according to the SQL standard: row-level triggers and statement-level triggers. **Row-Level Trigger:** It is a trigger, which is activated for each row by a triggering statement such as insert, update, or delete. For example, if a table has inserted, updated, or deleted multiple rows, the row trigger is fired automatically for each row affected by the insert, update, or delete statement. **Statement-Level Trigger:** It is a trigger, which is fired once for each event that occurs on a table regardless of how many rows are inserted, updated, or deleted. **Why we need/use triggers in MySQL?**

We need/use triggers in MySQL due to the following features:

- Triggers help us to enforce business rules.
- Triggers help us to validate data even before they are inserted or updated.
- Triggers help us to keep a log of records like maintaining audit trails in tables.
- SQL triggers provide an alternative way to check the integrity of data.
- Triggers provide an alternative way to run the scheduled task.
- Triggers increases the performance of SQL queries because it does not need to compile each time the query is executed.
- Triggers reduce the client-side code that saves time and effort.
- Triggers help us to scale our application across different platforms.
- Triggers are easy to maintain.

Limitations of Using Triggers in MySQL

- MySQL triggers do not allow to use of all validations; they only provide extended validations. **For example**, we can use the NOT NULL, UNIQUE, CHECK and FOREIGN KEY constraints for simple validations.
- Triggers are invoked and executed invisibly from the client application. Therefore, it isn't easy to troubleshoot what happens in the database layer.
- Triggers may increase the overhead of the database server.

Types of Triggers in MySQL?

We can define the maximum six types of actions or events in the form of triggers:

1. **Before Insert:** It is activated before the insertion of data into the table.

2. **After Insert:** It is activated after the insertion of data into the table.

3. **Before Update:** It is activated before the update of data in the table.

4. **After Update:** It is activated after the update of the data in the table.

5. **Before Delete:** It is activated before the data is removed from the table.

6. **After Delete:** It is activated after the deletion of data from the table.

When we use a statement that does not use INSERT, UPDATE or DELETE query to change the data in a table, the triggers associated with the trigger will not be invoked.

Naming Conventions

Naming conventions are the set of rules that we follow to give appropriate unique names. It saves our time to keep the work organize and understandable. Therefore, **we must use a unique name for each trigger associated with a table**. However, it is a good practice to have the same trigger name defined for different tables.

The following naming convention should be used to name the trigger in MySQL:

1. (BEFOR | **AFTER**) table_name (**INSERT** | **UPDATE** | **DELETE**)

Thus,

Trigger Activation Time: BEFORE | AFTER

Trigger Event: INSERT | UPDATE | DELETE

How to create triggers in MySQL?

We can use the **CREATE TRIGGER** statement for creating a new

trigger in MySQL. Below is the syntax of creating a trigger in MySQL:

```
CREATE TRIGGER trigger_name
   (AFTER | BEFORE) (INSERT | UPDATE | DELETE)
      ON table_name FOR EACH ROW
      BEGIN
     --variable declarations
     --trigger code
     END;
```

Queries

Trigger 1: Before Insert

Statement : Make the salary 0 if a non-negative salary is inserted

Code :

```
DELIMITER $$
CREATE TRIGGER Discard_Neg_Sal
BEFORE INSERT
ON employee FOR EACH ROW
BEGIN
IF new.e_salary<0
THEN set new.e_salary=0;
END IF;
END $$
DELIMITER ;

insert into employee values("Jay Nadkarni","Permanent",6,1234,"Nadkarni","Jay","Andheri","Mumbai",12346,-900);

Select * from employee;
```

Output:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

E_Name	E_Type	E_ID	H_ID	LastName	FirstName	Address	City	E_Contact	E_Salary
Adwait Purao	Permanent	1	1234	Purao	Adwait	Kurla	Mumbai	12345	10000
Akshay Kumar	Temporary	3	3456	Kumar	Akshay	Ram chowk	Ramgad	12347	30000
Ranbir Kapoor	Permanent	4	2345	Kapoor	Ranbir	Roopnagar	Agra	12348	40000
Angelina Jolie	Permanent	5	8970	Jolie	Angelina	Beverly Hills	Los Angeles	12349	50000
Jay Nadkarni	Permanent	6	1234	Nadkarni	Jay	Andheri	Mumbai	12346	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Trigger 2: After Insert

Statement :Add a message to the reminder column if number of employees is zero

Code :

```
DELIMITER $$
CREATE TRIGGER Negative_Emp
AFTER INSERT
ON hotel_info FOR EACH ROW
BEGIN
IF new.h_num_emp<0
THEN set new.reminder="Pls enter number of employees";
END IF;
END $$
DELIMITER ;

insert into hotel_info values("Vrindavan",8766,"Andheri",-89,7,NULL);

SELECT * FROM HOTEL_INFO;
```

Output:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

H_Name	H_ID	H_Address	H_Num_Emp	H_vacancies	reminder
marriot	1234	Pune	3456	5	NULL
The Plaza	2345	New York	4567	7	NULL
Claridge's	3456	London	5678	7	NULL
Raffles	5678	Singapore	6789	8	NULL
Taj Mahal Palace	6789	Mumbai	789 7890	9	NULL
Vrindavan	8766	Andheri	-89	7	Pls enter number of employees
Beverly Hills Hotel	8970	Los Angeles	8907	2	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Trigger 3: Before Update

Statement : Give a remark message if there is abnormal hike in price

Code :

```
DELIMITER $$
CREATE TRIGGER Pre_Price_Update
```

```
BEFORE UPDATE
ON room FOR EACH ROW
BEGIN
IF new.R_price>old.R_price * 5
THEN set new.remark="Abnormal price hike";
END IF;
END $$
DELIMITER ;

update room set r_price=10000 where r_no=12;

select * from room;
```

Output:

Result Grid | Filter Rows: | Edit: | Export/Import:

R_no	R_vacany	R_price	R_type	H_ID	Reservation_no	remark
12	1	10000	Basic	1234	1	Abnormal price hike
13	0	2000	Deluxe	2345	2	NULL
14	1	1500	Suite	5678	3	NULL
15	0	2500	Luxury Suite	6789	4	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Trigger 4: After update

Statement : Give a error message if there is an abnormal rise in salary

Code :

```
DELIMITER $$
CREATE TRIGGER After_salary_Update
AFTER UPDATE
ON employee FOR EACH ROW
BEGIN
IF new.e_salary>old.e_salary * 5 Then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT="Abnormal salary hike";
END IF;
END $$
DELIMITER ;

update employee set e_salary=100000 where e_id=1;
```

Output:

29 08:14:18 update employee set e_salary=100000 where e_id=1 Error Code: 1644. Abnormal salary hike

Trigger 5: Before Delete

Statement : Put records in employee archives table befored deleting an entry from employee table

Code :

```
DELIMITER $$
CREATE TRIGGER Before_emp_delete
BEFORE DELETE
ON employee FOR EACH ROW
BEGIN
INSERT INTO employee_Archives
VALUES(OLD.e_id,OLD.e_name,OLD.e_contact,OLD.city);
END $$
DELIMITER ;

delete from employee where e_id=1;

select * from employee_archives;
```

Output:

E_id	e_name	E_contact	city
1	Adwait Purao	12345	Mumbai
NULL	NULL	NULL	NULL

Trigger 6: After Delete

Statement : Make the salary 0 if a non-negative salary is inserted

Code :

```
DELIMITER $$
CREATE TRIGGER After_emp_delete
AFTER DELETE
ON employee FOR EACH ROW
BEGIN
INSERT INTO Salary_Archives
```

```
VALUES(OLD.e_id,OLD.e_salary);
END $$
DELIMITER ;

delete from employee where e_id=3;
select * from salary_archives;
```

Output:

E_id	E_salary
3	30000
NULL	NULL

Trigger 7: Before insert(Room)

Statement : Show an error message if room price is negative

Code :

```
DELIMITER $$
CREATE TRIGGER Neg_room_price
BEFORE INSERT
ON room FOR EACH ROW
BEGIN
IF new.r_price<0
THEN SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT="Negative room price";
END IF;
END $$
DELIMITER ;

Insert into room values(16,1,-500,"Deluxe",1234,5,NULL);
```

Output:

52 08:52:42 Insert into room values(16,1,-500,"Deluxe",1234,5,NULL) Error Code: 1644. Negative room price 0.000 sec

Trigger 8: After Insert

Statement : Set reminder if vacancies are negative

Code :

```
DELIMITER $$
CREATE TRIGGER Negative_Vacancies_hotel
AFTER INSERT
ON hotel_info FOR EACH ROW
BEGIN
IF new.h_vacancies<0
THEN set new.reminder="Vacancies cannot be negative";
END IF;
END $$
DELIMITER ;

Insert into hotel_info values("Sawali",7564,"Kurla",88,-4,NULL);

Select * from hotel_info;
```

Output:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

H_Name	H_ID	H_Address	H_Num_Emp	H_vacancies	reminder
marriot	1234	Pune	3456	5	NULL
The Plaza	2345	New York	4567	7	NULL
Claridge's	3456	London	5678	7	NULL
Raffles	5678	Singapore	6789	8	NULL
Taj Mahal Palace	6789	Mumbai	7890	9	NULL
Sawali	7564	Kurla	88	-4	Vacancies cannot be negative
Vrindavan	8766	Andheri	-89	7	Pls enter number of employees
Beverly Hills Hotel	8970	Los Angeles	8907	2	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Trigger 9: After Update

Statement : Display error message if H_id is changed

Code :

```
DELIMITER $$
CREATE TRIGGER After_Room_Update
AFTER UPDATE
ON room FOR EACH ROW
BEGIN
IF new.h_id!=old.h_id then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT="Pls enter the correct Hotel id";
END IF;
```

```
END $$
DELIMITER ;
```

Update room set h_id=2345 where r_no=12;

Output:

76	09:38:04	Update room set h_id=2345 where r_no=12	Error Code: 1644. Pls enter the correct Hotel id	0.000 sec

Trigger 10: Before Update

Statement : Set reminder if hotel name is changed

Code :

```
CREATE TRIGGER Name_update
BEFORE UPDATE
ON hotel_info FOR EACH ROW
BEGIN
IF new.h_name!=old.h_name
THEN set new.reminder="Name of hotel changed";
END IF;
END $$
DELIMITER ;
```

update hotel_info set h_name= "roshan" where h_id=7564;

Output:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

H_Name	H_ID	H_Address	H_Num_Emp	H_vacancies	reminder
marriot	1234	Pune	3456	5	NULL
The Plaza	2345	New York	4567	7	NULL
Claridge's	3456	London	5678	7	NULL
Raffles	5678	Singapore	6789	8	NULL
Taj Mahal Palace	6789	Mumbai	7890	9	NULL
roshan	7564	Kurla	88	-4	Name of hotel changed
Vrindavan	8766	Andheri	-89	7	Pls enter number of employees
Beverly Hills Hotel	8970	Los Angeles	8907	2	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Conclusion

In this experiment we learnt about triggers and how they help us in enforcing rules and validating data before insertion , deletion and updation . We implemented triggers on MySql

workbench on hotel database.