EXPERIMENT NO. 9

AIM:- Implementation of Views and Triggers

OBJECTIVES :-Implementation of Views and Triggers

THEORY:- Views are stored queries. A view acts as a virtual table. A view consists of rows & columns just like the table. The difference between table and view is that view are definitions built on top of other tables (or views).

There are some points to define a View as follows:

- It is used to restrict access to the database.
- Hide data complexity.
- A view is stored as a select statement in the database.
- DML operations on a view like Insert, Update, Delete effects.

There are several rules which SELECT statement has to follows:

- In the SELECT statement, subquery cannot be included.
- Variables such as local, user, and session variables cannot be used in the SELECT statement.
- A prepared statement cannot be used in the view.
- Temporary tables or views cannot be used in the SELECT statements and any tables or views which referred by views must exist.
- View cannot be associated with triggers.

Advantages of MySQL Views

MySQL has various advantages as

1. It simplifies complex queries.
2. It adds an extra security layer.
3. Make business login consistent

1) Creating View: -

Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables or another view.

Syntax of create view is as,

Create view view_name as select column(s) from table where condition.

2) Displaying a View: -

The syntax for displaying the data in a view is similar to fetching data from a table using a SELECT statement.

Select * from view_name

3) Updating a View: -
UPDATE command for view is same as for tables.
Create or replace view view_name as select column_name(s) from table_name where condition

4) Drop View: -
Drop command is used for dropping the view same as table.
Drop view view_name;

5) SHOW View
To show all the present "Views" inside the database, type the following command in your Workbench.
SHOW FULL tables;

6) Rename View
Rename is used to change the name of the view table, views and tables share the same namespace and may create confusion.
Rename table original_view_name to new_view_name

SQL Triggers :-

The SQL CREATE TRIGGER statement provides a way for the database management system to actively control, monitor, and manage a group of tables whenever an insert, update, or delete operation is performed.

The statements specified in the SQL trigger are executed each time an SQL insert, update, or delete operation is performed. An SQL trigger may call stored procedures or user-defined functions to perform additional processing when the trigger is executed. Unlike stored procedures, an SQL trigger cannot be directly called from an application. Instead, an SQL trigger is invoked by the database management system on the execution of a triggering insert, update, or delete operation. A trigger or database trigger is a stored program executed automatically to respond to a specific event e.g., insert, update or delete occurred in a table.

The database trigger is powerful tool for protecting the integrity of the data in your MySQL databases. In addition, it is useful to automate some database operations such as logging, auditing, etc.

Advantages of Triggers

1. Triggers offer another way to verify data integrity.
2. Triggers may be useful for auditing table adjustments in results.
3. Triggers offer an easy method to execute planned activities. When using triggers, you don't have to wait for the planned events to start because the triggers are triggered immediately before or after an adjustment is made to the data in a table.

Disadvantages of Triggers

1. Triggers can increase MySQL Server overhead.
2. Triggers may be difficult to handle because they operate in the database automatically.

An SQL trigger can be created by specifying the CREATE TRIGGER SQL statement.

Parts of a Trigger

A trigger has three basic parts:

- A triggering event or statement
- A trigger restriction
- A trigger action

The Triggering Event or Statement: -

A triggering event or statement is the SQL statement, database event, or user event that causes a trigger to fire. A triggering event can be one or more of the following:

- An INSERT, UPDATE, or DELETE statement on a specific table (or view, in some cases)
- A CREATE, ALTER, or DROP statement on any schema object

Trigger Restriction :-

A trigger restriction specifies a Boolean expression that must be true for the trigger to fire. The trigger action is not run if the trigger restriction evaluates to false or unknown.

Trigger Action :-

A trigger action is the procedure (PL/SQL block, Java program, or C callout) that contains the SQL statements and code to be run when the following events occur:

- A triggering statement is issued.
- The trigger restriction evaluates to true. Seps :-
  - Put the trigger name after the `CREATE TRIGGER` statement. The trigger name should follow the naming convention `[trigger time]_[table name]_[trigger event]`, for example `before_employees_update`.
  - Trigger activation time can be `BEFORE` or `AFTER`. You must specify the activation time when you define a trigger. You use the `BEFORE` keyword if you want to process action prior to the change is made on the table and `AFTER` if you need to process action after the change is made.
  - The trigger event can be `INSERT`, `UPDATE` or `DELETE`. This event causes the trigger to be invoked. A trigger only can be invoked by one event. To define a trigger that is invoked by multiple events, you have to define multiple triggers, one for each event.
  - A trigger must be associated with a specific table. Without a table trigger would not exist therefore you have to specify the table name after the `ON` keyword.
  - You place the SQL statements between `BEGIN` and `END` block. This is where you define the logic for the trigger.

Trigger Syntax:

```
CREATE OR REPLACE TRIGGER trigger_name
BEFORE / AFTER / INSTEAD OF
INSERT / DELETE / UPDATE ON table_name
[REFERENCING [NEW AS new_cols] [OLD AS old_cols] ]
[FOR EACH ROW [WHEN (where_condition)] ] [DECLARE]
BEGIN EXCEPTION
END;
```

Drop Trigger:-

It removes the trigger.

Syntax: drop trigger trigger_name

Some trigger examples are as follows

```
mysql> select * from people;
+------+-------+
| age  | name  |
+------+-------+
|   50 | amit  |
|   55 | sumit |
|  -45 | rahul |
|  -42 | akash |
|   45 | nayan |
|   43 | ajay  |
|   79 | rajan |
|   40 | aryan |
|   28 | raj   |
+------+-------+
9 rows in set (0.00 sec)

mysql> Delimiter $$
mysql> create trigger before_people_insert
    -> before insert on people
    -> for each row
    -> begin
    -> if new.age<0 then
    -> set new.age=0;
    -> end if;
    -> end
    -> $$
Query OK, 0 rows affected (0.09 sec)

mysql> insert into people values(56,'soniya');
    -> $$
Query OK, 1 row affected (0.07 sec)

mysql> insert into people values(-54,'abhishek');
    -> $$
Query OK, 1 row affected (0.05 sec)
```

The output of this trigger is

```
mysql> select * from people;
+------+----------+
| age  | name     |
+------+----------+
|   50 | amit     |
|   55 | sumit    |
|  -45 | rahul    |
|  -42 | akash    |
|   45 | nayan    |
|   43 | ajay     |
|   79 | rajan    |
|   40 | aryan    |
|   28 | raj      |
|   56 | soniya   |
|    0 | abhishek |
+------+----------+
11 rows in set (0.00 sec)
```

Conclusion :- Hence studied the triggers in SQL.