Lab-Week08

Trigger Example

- **Example 01:** In this example, initially we have created two tables named “Salaries” and “Salary_Archives”. In the salary table we will hold the salary information of the present employees only but we want to hold our past employee’s salary information also that’s why we have created another table “Salary_Archives” that will hold the salary information of these employees who left the job. To achieve this, we use a Trigger ‘Before Delete’ on ‘Salaries’ table to its each row so that when any row from “Salaries” table task to be deleted, MySQL will be triggered automatically before deletion and will insert all the value of that row with the deletion time in ‘Salary_Archives’ then it will delete that row from ‘Salaries’ table.

- **Example 02:** In the example 2 we have created another table named ‘Salary_Budget’ that will hold the total salary of all employees in total attributes. Here we get the total salary by using the group function SUM (amount) where amount is an attribute of ‘Salary’ table that hold the salary of each employee so basically this group function will add all the amount and will return one row from ‘Salaries’ and this will be insert in the ‘Salary_Budget’ table. After that we created 3 new Triggers in ‘Salaries’ table, the description is given below.

    - 1. Trigger ‘After Update’ for each row: After Update the ‘salaries’ table MySQL get triggered and will update the ‘Salary_Budget’ if there present any difference between old amount and new amount of ‘salaries’ table.
    - 2. Trigger ‘After Delete’ for each row: After deleting a row from ‘Salaries’ table MySQL will be triggered again and will Update the ‘Salary_Budgets’ table’s total by subtract the old amount (The deleted row’s amount) from the total of what we have till now.
    - 3. Trigger ‘After Insert’ for each row: Now if we insert any rows in “Salaries” table MySQL will be triggered now also and will Update the ‘Salary_Budgets’ table’s total by add the new amount (The newly inserted row’s amount) to the total of what we have till now.

- **Example 03:** In this example we created two tables one is Bank_account and another one is Account_Audit. Now we will do multiple things onwards. At first, we will create or replace (If there exist any by manual as MySQL don’t support this replace query) trigger before Insert, Delete, and Update on ‘Bank_Account’ table for each row. Now if we insert values into ‘Bank_Account’ table then MySQL will get triggered and before insert the values in ‘Bank_account’, it will insert the new Account, new balance, Sysdate (System’s date), ‘Insert’ (Operation type) into ‘Account_Audit’ table’s Account, Curr_balance, Audit_Date, OP_Type attributes. Now if we update values of any row of ‘Bank_Account’ table then MySQL will get triggered also and before update the values in ‘Bank_account’, it will insert the new Account, old balance (Balance after insertion in ‘Bank-Account’), new balance (Balance that will get update now in ‘Bank_Account’), Sysdate (System’s date), ‘Update’ (Operation type) into ‘Account_Audit’ table’s Account, Prev_Balance, Curr_balance, Audit_Date, OP_Type attributes. And now if we delete any rows from ‘Bank_Account’, then MySQL will get triggered also and

before delete the rows from ‘Bank_account’, it will insert the old Account (What we will delete now), old balance (Balance stored in ‘Bank-Account’), Sysdate (System’s date), ‘Delete’ (Operation type) into ‘Account_Audit’ table’s Account, Curr_balance, Audit_Date, OP_Type attributes.

Store Procedure Example

In this example initially we created 2 tables named ‘Bookshelf’ and ‘Book_Order’ and then we create a store procedure named as ‘New_Book’ where we stored 2 queries one to insert values in the ‘Bookshelf’ table and the 2nd one for delete that book’s information from ‘Book_Order’ table that is now stored in the ‘Bookshelf’ table by the first query of this procedure. So now if we call the ‘New_Book’ procedure with its parameter then it will do exactly what I described above.