|  |  |
| --- | --- |
| **Name** | **Hatim Yusuf Sawai** |
| **UID no.** | **2021300108** |
| **Experiment No.** | **8** |

|  |  |
| --- | --- |
| **AIM:** | **Implement Triggers in MySql** |
| **Program 1** | |
| **PROBLEM STATEMENT:** | Implement different types of triggers on tables in the existing database in mysql |
| **THEORY:** | **TRIGGERS:**  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**.  **Limitations of Using Triggers in MySQL**  1. 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.  2. Triggers are invoked and executed invisibly from the client application. Therefore, it isn't easy to troubleshoot what happens in the database layer.  3. Triggers may increase the overhead of the database server.  **Types of Triggers in MySQL:**  **1. BEFORE INSERT:**  It is activated before the insertion of data into the table.  **Syntax:**  **CREATE** **TRIGGER** trigger\_name  **BEFORE INSERT** **ON** table\_name  **FOR** EACH ROW  **BEGIN**  **trigger** code  **END;**  **2. AFTER INSERT:**  It is activated after the insertion of data into the table.  **Syntax:**  **CREATE** **TRIGGER** trigger\_name  **AFTER INSERT** **ON** table\_name  **FOR** EACH ROW  **BEGIN**  **trigger** code  **END;**  **3. BEFORE UPDATE:**  It is activated before the update of data in the table.  **Syntax:**  **CREATE** **TRIGGER** trigger\_name  **BEFORE UPDATE** **ON** table\_name  **FOR** EACH ROW  **BEGIN**  **trigger** code  **END;**  **4. AFTER UPDATE**  It is activated after the update of the data in the table.  **Syntax:**  **CREATE** **TRIGGER** trigger\_name  **AFTER UPDATE** **ON** table\_name  **FOR** EACH ROW  **BEGIN**  **trigger** code  **END;**  **5. BEFORE DELETE**  It is activated before the data is removed from the table.  **Syntax:**  **CREATE** **TRIGGER** trigger\_name  **BEFORE DELETE** **ON** table\_name  **FOR** EACH ROW  **BEGIN**  **trigger** code  **END;**  **6. AFTER DELETE**  It is activated after the deletion of data from the table.  **Syntax:**  **CREATE** **TRIGGER** trigger\_name  **AFTER DELETE** **ON** table\_name  **FOR** EACH ROW  **BEGIN**  **trigger** code  **END;** |
| **QUERIES:** | **Before Insert Trigger**  1. Create a trigger to check if new record contains patient age less than 18, if so, throw an error:  CREATE trigger patient\_trigger  BEFORE INSERT ON patient  FOR EACH ROW  BEGIN  IF NEW.Age<18 THEN  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = 'Age must be greater than 18';  END IF;  END;  2. Create a trigger to check if new salary being inserted is always over 0 (+ve) and if not then default the salary column to 0:  CREATE trigger doc\_salary\_trigger  BEFORE INSERT ON doctor  FOR EACH ROW  BEGIN  IF NEW.Salary<0 THEN  SET NEW.Salary = 0;  END IF;  END;  **Before Update Trigger:**  3. Create a trigger to change the address to “Andheri West” whenever the user updates the address to “Andheri”:  CREATE trigger pat\_update\_trigger  BEFORE UPDATE ON patient  FOR EACH ROW  BEGIN  IF NEW.address='Andheri' THEN  SET NEW.address='Andheri West';  END IF;  END;  **Before delete trigger:**  4. Create a trigger to check if patient is not assigned to any doctor before deletion:  CREATE trigger pat\_delete\_trigger  BEFORE DELETE ON patient  FOR EACH ROW  BEGIN  If OLD.D\_id is not null THEN  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = 'Cannot delete patient as he is assigned to a doctor';  END IF;  END;  5. Create a trigger to check if a doctor is assigned to a patient before deleting the doctor record:  CREATE trigger doc\_delete\_trigger  BEFORE DELETE ON doctor  FOR EACH ROW  BEGIN  If OLD.D\_id in (SELECT D\_id FROM patient) THEN  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = 'Cannot delete doctor as he is assigned to a patient';  END IF;  END;  **After Delete Trigger:**  6. Create a trigger to increase the salary of a doctor who’s assigned patient was removed from the database after treatment:  CREATE trigger del\_trigger  AFTER DELETE ON patient  FOR EACH ROW  BEGIN  IF OLD.D\_id is not null THEN  UPDATE doctor SET Salary=Salary+1000 WHERE D\_id=OLD.D\_id;  END IF;  END; |
| **CONCLUSION:** | In this experiment, we learned how to implement different type of triggers in MySQL and how to use thr triggers to maintain consistent data across all tables |