

BansilalRamnathAgarwalCharitableTrust's

VishwakarmaInstituteofTechnology, Pune-37

(AnautonomousInstituteofSavitribaiPhulePuneUniversity)



Department of Multidisciplinary Engineering

Division	C
Batch	3
GR-no	12320192
Rollno	77
Name	Aditya Janardan Jilla

Title: Write and execute all types of database triggers in PL/SQL for your course project

Trigger types : Before insert, after insert, before update, after update, before delete, after delete. implement all types of triggers for your course project.

OPERATIONAL TABLE :

```
mysql> select * from students;
```

StudentID	FirstName	LastName	Age	Gender	CollegeID	Location
1	Aditya	Jilla	20	Male	5	Pune
2	Swapnil	Pamu	21	Male	6	Pune
3	Atharva	Chavan	20	Male	3	Vellore
4	Amit	Barde	23	Male	5	Pune
5	Sapna	Wargaj	22	Female	4	Pune
6	Piyush	Warde	24	Male	5	Pune
7	Dipali	Katave	21	Female	2	New Delhi
8	Kunal	Patil	20	Male	4	Pune
9	Neha	Shinde	21	Female	5	Pune
10	Rahul	Gavde	22	Male	6	Pune
11	Ankita	Pawar	23	Female	3	Vellore
12	Yogesh	Gore	24	Male	5	Pune
13	Pooja	Patil	20	Female	2	New Delhi
14	Raj	More	21	Male	1	Mumbai

```
14 rows in set (0.00 sec)
```

```
mysql> select * from colleges;
```

CollegeID	CollegeName	Location	EstablishedYear
1	Indian Institute of Technology Bombay	Mumbai	1958
2	Indian Institute of Technology Delhi	New Delhi	1961
3	Vellore Institute of Technology	Vellore	1984
4	Vishwakarma Institute of Information Technology	Pune	1983
5	New College Name	Pune	1854
6	New College Name	Pune	1983

```
6 rows in set (0.00 sec)
```

```
mysql> desc auditlog;
```

Field	Type	Null	Key	Default	Extra
log_id	int	NO	PRI	NULL	auto_increment
operation	varchar(10)	YES		NULL	
table_name	varchar(20)	YES		NULL	
operation_time	datetime	YES		NULL	
data	varchar(255)	YES		NULL	

```
5 rows in set (0.00 sec)
```

1) Before Insert

SYNTAX:

```
DELIMITER //
CREATE TRIGGER trigger_name
BEFORE INSERT ON table_name
FOR EACH ROW
BEGIN
    -- Trigger logic here
END;
//
DELIMITER ;
```

SCREENSHOT/ OUTPUT:

```
mysql> DELIMITER //
mysql> CREATE TRIGGER before_students_insert
-> BEFORE INSERT ON Students
-> FOR EACH ROW
-> BEGIN
-> IF NEW.Age < 17 OR NEW.Age > 100 THEN
-> SIGNAL SQLSTATE '45000'
-> SET MESSAGE_TEXT = 'Age must be between 17 and 100';
-> END IF;
-> END;
-> //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql> INSERT INTO Students (StudentID, FirstName, LastName, Age, Gender, CollegeID, Location)
-> VALUES (15, 'Aditya', 'Jilla', 15, 'Male', 1, 'Mumbai');
ERROR 1644 (45000): Age must be between 17 and 100
```

DESCRIPTION: This SQL trigger ensures that when inserting a new row into the "Students" table, the age value must fall within the range of 17 to 100. If the age is outside this range, it triggers an error preventing the insertion, maintaining data integrity.

2) After Insert

SYNTAX:

```
DELIMITER //
CREATE TRIGGER trigger_name
AFTER INSERT ON table_name
FOR EACH ROW
BEGIN
    -- Trigger logic here
END;
//
DELIMITER ;
```

SCREENSHOT/OUTPUT:

```
mysql> DELIMITER //
mysql> CREATE TRIGGER after_students_insert
-> AFTER INSERT ON Students
-> FOR EACH ROW
-> BEGIN
-> INSERT INTO AuditLog (operation, table_name, operation_time, data)
-> VALUES ('INSERT', 'Students', NOW(), CONCAT('Inserted Student: ', NEW.FirstName, ' ', NEW.LastName));
-> END;
-> //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql> INSERT INTO Students (StudentID, FirstName, LastName, Age, Gender, CollegeID, Location)
-> VALUES (18, 'Aditya', 'Jilla', 22, 'Male', 1, 'Mumbai');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM AuditLog;
+-----+-----+-----+-----+-----+
| log_id | operation | table_name | operation_time | data |
+-----+-----+-----+-----+-----+
| 1 | INSERT | Students | 2024-04-18 18:48:10 | Inserted Student: Aditya Jilla |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

DESCRIPTION: This SQL trigger, executed after each insertion into the "Students" table, logs the operation in an "AuditLog" table. It records the operation type ('INSERT'), table name ('Students'), operation time (current time), and details of the inserted student, including first and last name. This facilitates tracking and monitoring of database changes for auditing purposes.

3) BEFORE UPDATE

SYNTAX:

```
DELIMITER //
CREATE TRIGGER trigger_name
BEFORE UPDATE ON table_name
FOR EACH ROW
BEGIN
    -- Trigger logic here
END;
//
DELIMITER ;
```

SCREENSHOT/OUTPUT:

```
mysql> DELIMITER //
mysql> CREATE TRIGGER before_students_update
-> BEFORE UPDATE ON Students
-> FOR EACH ROW
-> BEGIN
-> IF NEW.Age < 17 OR NEW.Age > 100 THEN
-> SIGNAL SQLSTATE '45000'
-> SET MESSAGE_TEXT = 'Age must be between 17 and 100';
-> END IF;
-> END;
-> //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql> UPDATE Students
-> SET Age = 15
-> WHERE StudentID = 1;
ERROR 1644 (45000): Age must be between 17 and 100
```

DESCRIPTION:

This SQL trigger, executed before updating a row in the "Students" table, verifies that the new age value falls within the range of 17 to 100. If the age is outside this range, it raises an error, enforcing data integrity by preventing invalid updates.

4) AFTER UPDATE

SYNTAX:

```
DELIMITER //
CREATE TRIGGER trigger_name
AFTER UPDATE ON table_name
FOR EACH ROW
BEGIN
    -- Trigger logic here
END;
//
DELIMITER ;
```

SCREENSHOT/OUTPUT:

```
mysql> DELIMITER //
mysql> CREATE TRIGGER after_students_update
  -> AFTER UPDATE ON Students
  -> FOR EACH ROW
  -> BEGIN
  -> INSERT INTO AuditLog (operation, table_name, operation_time, data)
  -> VALUES ('UPDATE', 'Students', NOW(), CONCAT('Updated Student: ', OLD.FirstName, ' ', OLD.LastName, ' to ', NEW.FirstName, ' ',
NEW.LastName));
  -> END;
  -> //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> UPDATE Students
  -> SET FirstName = 'Aditya123', LastName = 'Jilla'
  -> WHERE StudentID = 1;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM AuditLog;
```

log_id	operation	table_name	operation_time	data
1	INSERT	Students	2024-04-18 18:48:10	Inserted Student: Aditya Jilla
2	UPDATE	Students	2024-04-18 19:01:31	Updated Student: Aditya Jilla to Aditya123 Jilla

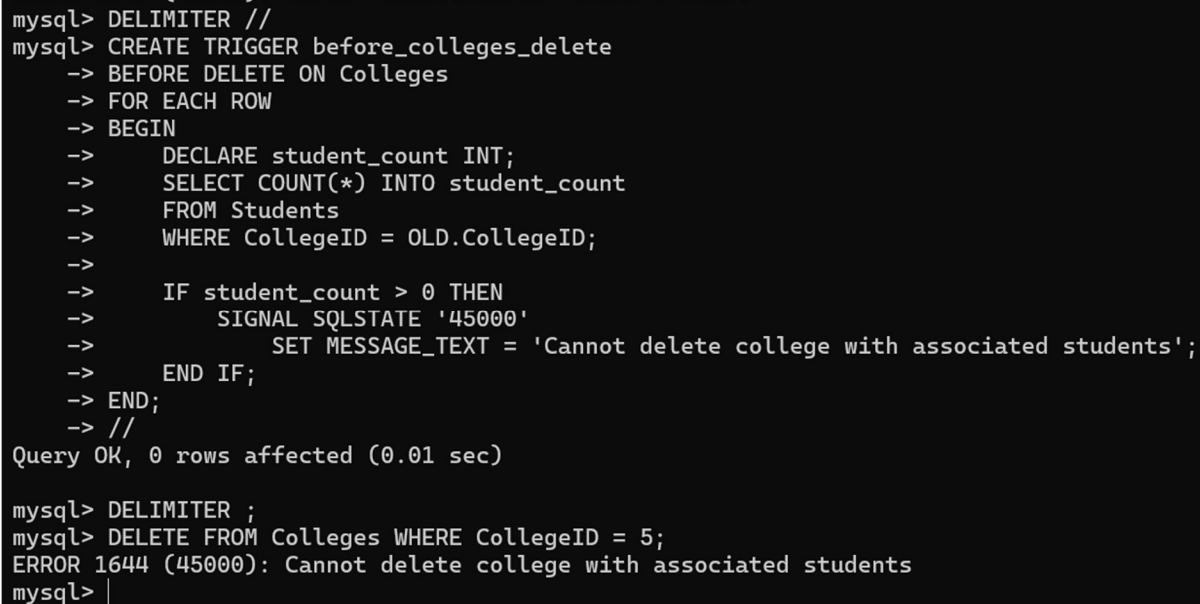
DESCRIPTION: This SQL trigger, executed after each update on the "Students" table, records the operation in the "AuditLog" table. It captures the operation type ('UPDATE'), table name ('Students'), operation time (current time), and details of the updated student, including both old and new first and last names. This enables tracking and auditing of database modifications.

5) BEFORE DELETE

SYNTAX:

```
DELIMITER //
CREATE TRIGGER trigger_name
BEFORE DELETE ON table_name
FOR EACH ROW
BEGIN
    -- Trigger logic here
END;
//
DELIMITER ;
```

SCREENSHOT/OUTPUT:



```
mysql> DELIMITER //
mysql> CREATE TRIGGER before_colleges_delete
-> BEFORE DELETE ON Colleges
-> FOR EACH ROW
-> BEGIN
->     DECLARE student_count INT;
->     SELECT COUNT(*) INTO student_count
->     FROM Students
->     WHERE CollegeID = OLD.CollegeID;
->
->     IF student_count > 0 THEN
->         SIGNAL SQLSTATE '45000'
->         SET MESSAGE_TEXT = 'Cannot delete college with associated students';
->     END IF;
-> END;
-> //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> DELETE FROM Colleges WHERE CollegeID = 5;
ERROR 1644 (45000): Cannot delete college with associated students
mysql> |
```

DESCRIPTION: The `before_colleges_delete` trigger fires before a delete operation on the `Colleges` table. It checks if there are any students associated with the college being deleted in the `Students` table. If any students are found, the trigger raises an error with the message "Cannot delete college with associated students," preventing the deletion of the college.

5) AFTER DELETE

SYNTAX:

```
DELIMITER //
CREATE TRIGGER trigger_name
AFTER DELETE ON table_name
FOR EACH ROW
BEGIN
    -- Trigger logic here
END;
//
DELIMITER ;
```

SCREENSHOT/OUTPUT:

```
mysql> DELIMITER //
mysql> CREATE TRIGGER after_students_delete
-> AFTER DELETE ON Students
-> FOR EACH ROW
-> BEGIN
-> INSERT INTO AuditLog (operation, table_name, operation_time, data)
-> VALUES ('DELETE', 'Students', NOW(), CONCAT('Deleted Student: ', OLD.FirstName, ' ', OLD.LastName));
-> END;
-> //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> DELETE FROM Students
-> WHERE StudentID = 1;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM AuditLog;
+-----+-----+-----+-----+-----+
| log_id | operation | table_name | operation_time | data |
+-----+-----+-----+-----+-----+
| 1 | INSERT | Students | 2024-04-18 18:48:10 | Inserted Student: Aditya Jilla |
| 2 | UPDATE | Students | 2024-04-18 19:01:31 | Updated Student: Aditya Jilla to Aditya123 Jilla |
| 3 | DELETE | Students | 2024-04-18 22:24:00 | Deleted Student: Aditya123 Jilla |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

DESCRIPTION: This SQL trigger, executed after each deletion from the "Students" table, logs the operation in the "AuditLog" table. It records the operation type ('DELETE'), table name ('Students'), operation time (current time), and details of the deleted student, including their first and last names. This mechanism facilitates monitoring and auditing of database deletions.