MIT World Peace University

Database Management Systems
Second Year B. Tech, Semester 4

Create Triggers using PL/SQL

Assignment No. 7

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

April 30, 2023

Contents

1 Aim

Write PL/SQL Triggers for the creation of insert trigger, delete trigger and update trigger on the given problem statements.

2 Objectives

1. To study and use Triggers using MySQL PL/SQL block

3 Problem Statement

Create tables and solve given queries

4 Theory

4.1 Triggers in PL/SQL

A trigger is a named PL/SQL unit that is stored in the database and can be invoked repeatedly. A trigger automatically executes whenever an event associated with a table occurs. There are two types of triggers based on the which level it is triggered. They are:

1. Row Level Trigger
2. Statement Level Trigger
3. Database Level Trigger
4. Instead of Trigger
5. DDL Trigger
6. System Trigger
7. Compound Trigger

4.2 Advantages of Triggers

1. Triggers can be used to enforce complex business rules.
2. Triggers can be used to enforce complex integrity constraints.
3. Triggers can be used to propagate data to other tables.
4. Triggers can be used to log all changes to a table.
5. Triggers can be used to prevent invalid transactions.
6. Triggers can be used to notify external applications of database changes.

4.3 Disadvantages of Triggers

1. Triggers are hidden within the database and can be difficult to find.
2. Triggers are not visible in the SQL source code.
3. Triggers can be difficult to debug.
4. Triggers can cause performance issues.
5. Triggers can cause infinite loops.
6. Triggers can cause locking issues.
7. Triggers can cause cascading failures.
8. Triggers can cause unpredictable results.
9. Triggers can cause security issues.

4.4 Usage of Triggers

Usages of Triggers

1. Auditing
2. Data Integrity
3. Data Validation
4. Notification
5. Replication
6. Security
7. Synchronization

4.5 Difference between Stored Procedure and Trigger

1. Triggers are invoked automatically in response to the associated DML statement.
2. Stored procedures are invoked explicitly by users or applications.
3. Triggers are attached to tables and are implicitly invoked.
4. Stored procedures are stand-alone and are explicitly invoked.
5. Triggers execute implicitly in response to DML statements on the table with which they are associated.
6. Stored procedures execute explicitly when they are invoked by a user or application.
7. Triggers execute under the security privileges of the owner of the trigger.
8. Stored procedures execute under the security privileges of the user who invokes them.

9. Triggers are attached to tables and are implicitly invoked.
10. Stored procedures are stand-alone and are explicitly invoked.
11. Triggers execute implicitly in response to DML statements on the table with which they are associated.
12. Stored procedures execute explicitly when they are invoked by a user or application.
13. Triggers execute under the security privileges of the owner of the trigger.
14. Stored procedures execute under the security privileges of the user who invokes them.

Syntax of Triggers:

```
CREATE TRIGGER Trigger_Name
[ BEFORE | AFTER ]  [ Insert | Update | Delete]
ON [Table_Name]
[ FOR EACH ROW | FOR EACH COLUMN ]
AS
Set of SQL Statement
```

4.6 Types of Triggers

1. Row Level Trigger
   - Row level trigger is triggered when a row is inserted, updated or deleted from a table.
   - Row level trigger is declared at row level.
   - Row level trigger is used to perform an action when a DML event (INSERT, UPDATE, DELETE) occurs.
   - Row level trigger can be used to enforce complex business rules or integrity constraints.
2. Statement Level Trigger
   - Statement level trigger is triggered when a DML event (INSERT, UPDATE, DELETE) occurs.
   - Statement level trigger is declared at statement level.
   - Statement level trigger is used to perform an action when a DML event (INSERT, UPDATE, DELETE) occurs.
   - Statement level trigger can be used to enforce complex business rules or integrity constraints.
3. Database Level Trigger
   - Database level trigger is triggered when a DDL event (CREATE, ALTER, DROP, RENAME, TRUNCATE) occurs.
   - Database level trigger is declared at database level.
   - Database level trigger is used to perform an action when a DDL event (CREATE, ALTER, DROP, RENAME, TRUNCATE) occurs.
   - Database level trigger can be used to enforce complex business rules or integrity constraints.

4. Instead of Trigger
    - Instead of trigger is triggered when a DML event (INSERT, UPDATE, DELETE) occurs.
    - Instead of trigger is declared at view level.
    - Instead of trigger is used to perform an action when a DML event (INSERT, UPDATE, DELETE) occurs.
    - Instead of trigger can be used to enforce complex business rules or integrity constraints.
5. DDL Trigger
    - DDL trigger is triggered when a DDL event (CREATE, ALTER, DROP, RENAME, TRUNCATE) occurs.
    - DDL trigger is declared at database level.
    - DDL trigger is used to perform an action when a DDL event (CREATE, ALTER,

4.7 Trigger Level

Trigger Levels are used to specify when the trigger should be fired. There are two types of trigger levels. They are:

1. Before Trigger
    - Before trigger is triggered before the triggering statement is executed.
    - Before trigger is declared using BEFORE keyword.
    - Before trigger is used to perform an action before the triggering statement is executed.
    - Before trigger can be used to enforce complex business rules or integrity constraints.
2. After Trigger
    - After trigger is triggered after the triggering statement is executed.
    - After trigger is declared using AFTER keyword.
    - After trigger is used to perform an action after the triggering statement is executed.
    - After trigger can be used to enforce complex business rules or integrity constraints.

4.8 Trigger Events

Trigger Events are used to specify when the trigger should be fired. There are three types of trigger events. They are:

1. Insert Trigger
    - Insert trigger is triggered when an insert event occurs.
    - Insert trigger is declared using INSERT keyword.
    - Insert trigger is used to perform an action when an insert event occurs.
    - Insert trigger can be used to enforce complex business rules or integrity constraints.
2. Update Trigger

- Update trigger is triggered when an update event occurs.
- Update trigger is declared using UPDATE keyword.
- Update trigger is used to perform an action when an update event occurs.
- Update trigger can be used to enforce complex business rules or integrity constraints.

3. Delete Trigger

   - Delete trigger is triggered when a delete event occurs.
   - Delete trigger is declared using DELETE keyword.
   - Delete trigger is used to perform an action when a delete event occurs.
   - Delete trigger can be used to enforce complex business rules or integrity constraints.

An SQL example for trigger events is given below:

```sql
CREATE TRIGGER trigger_name
    BEFORE INSERT ON table_name
    FOR EACH ROW
    BEGIN
        -- trigger body
    END;
```

4.9 NEW and OLD Clause /Trigger Variables

Clauses are used to specify the columns of the table. There are two types of clauses. They are:

1. NEW Clause

   - NEW clause is used to specify the columns of the table that are affected by the triggering statement.
   - NEW clause is used in INSERT and UPDATE triggers.
   - NEW clause is used to specify the columns of the table that are affected by the triggering statement.
   - NEW clause is used in INSERT and UPDATE triggers.

2. OLD Clause

   - OLD clause is used to specify the columns of the table that are affected by the triggering statement.
   - OLD clause is used in UPDATE and DELETE triggers.
   - OLD clause is used to specify the columns of the table that are affected by the triggering statement.
   - OLD clause is used in UPDATE and DELETE triggers.

An SQL Example for NEW and OLD clause would be:

```sql
    CREATE TRIGGER trigger_name
    BEFORE INSERT ON table_name
    FOR EACH ROW
    BEGIN
        -- trigger body
        IF NEW.column_name = 'value' THEN
            -- trigger body
        END IF;
    END;
```

4.10 Dropping Triggers

Dropping Triggers is used to drop the trigger from the database. An SQL Example for this would be:

```sql
    DROP TRIGGER trigger_name;
```

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Draw.io for Drawing the ER diagram.

6 Input

Given Database from the Problem Statement for the Assignment for our batch. (A1 PA 20)

7 Queries

```sql
-- Assignment 7 Triggers

-- Consider the following relational schema: BOOK (Isbn, Title,
-- SoldCopies) WRITING (Isbn, Name) AUTHOR (Name, SoldCopies)

-- Define a set of triggers for keeping SoldCopies in AUTHOR updated
-- with respect to: updates on SoldCopies in BOOK insertion of new tuples
-- in the WRITING relation

-- Create Database and Tables

CREATE database if not exists store;
use store;
CREATE TABLE BOOK (
    Isbn VARCHAR(10) PRIMARY KEY,
    Title VARCHAR(100),
    SoldCopies INT
);

CREATE TABLE WRITING (
    Isbn VARCHAR(10),
    Name VARCHAR(50),
    PRIMARY KEY (Isbn, Name),
    FOREIGN KEY (Isbn) REFERENCES BOOK(Isbn)
);

CREATE TABLE AUTHOR (
    Name VARCHAR(50) PRIMARY KEY,
    SoldCopies INT
);

CREATE TABLE Customer (
    cust_id INT PRIMARY KEY,
    Principal_amount DOUBLE,
    Rate_of_interest DOUBLE,
    Years INT
);


INSERT INTO BOOK (Isbn, Title, SoldCopies)
VALUES ('9783161', 'Crime and Punishment', 500);

INSERT INTO WRITING (Isbn, Name)
VALUES ('9783161', 'Fyodor Dostoevsky');

INSERT INTO AUTHOR (Name, SoldCopies)
VALUES ('Fyodor Dostoevsky', 500);

INSERT INTO Customer (cust_id, Principal_amount, Rate_of_interest, Years)
VALUES (1, 1000, 0.05, 5);

-- Create Triggers

DELIMITER //
CREATE TRIGGER update_author_soldcopies
AFTER UPDATE ON BOOK
FOR EACH ROW
BEGIN
    IF NEW.SoldCopies != OLD.SoldCopies THEN
        UPDATE AUTHOR
        SET SoldCopies = SoldCopies + (NEW.SoldCopies - OLD.SoldCopies) WHERE Name
     IN (SELECT Name FROM WRITING WHERE Isbn = NEW.Isbn);
    END IF;
END//
DELIMITER ;

delimiter $$
CREATE TRIGGER insert_author_soldcopies
AFTER INSERT ON WRITING
FOR EACH ROW
BEGIN
    UPDATE AUTHOR
    SET SoldCopies = SoldCopies + (SELECT SoldCopies FROM BOOK WHERE Isbn = NEW.
    Isbn)
    WHERE Name = NEW.Name;
END $$
delimiter ;
```

8 Outputs

```


MariaDB [(none)]>
MariaDB [(none)]> CREATE database if not exists store;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use store;
Database changed
MariaDB [store]> CREATE TABLE BOOK (
    ->     Isbn VARCHAR(10) PRIMARY KEY,
    ->     Title VARCHAR(100),
    ->     SoldCopies INT
    -> );
Query OK, 0 rows affected (0.006 sec)
```

```

MariaDB [store]>
MariaDB [store]> CREATE TABLE WRITING (
    ->     Isbn VARCHAR(10),
    ->     Name VARCHAR(50),
    ->     PRIMARY KEY (Isbn, Name),
    ->     FOREIGN KEY (Isbn) REFERENCES BOOK(Isbn)
    -> );
Query OK, 0 rows affected (0.006 sec)

MariaDB [store]>
MariaDB [store]> CREATE TABLE AUTHOR (
    ->     Name VARCHAR(50) PRIMARY KEY,
    ->     SoldCopies INT
    -> );
Query OK, 0 rows affected (0.005 sec)

MariaDB [store]>
MariaDB [store]> CREATE TABLE Customer (
    ->     cust_id INT PRIMARY KEY,
    ->     Principal_amount DOUBLE,
    ->     Rate_of_interest DOUBLE,
    ->     Years INT
    -> );
Query OK, 0 rows affected (0.005 sec)

MariaDB [store]>
MariaDB [store]>
MariaDB [store]> INSERT INTO BOOK (Isbn, Title, SoldCopies)
    -> VALUES ('9783161', 'Crime and Punishment', 500);
Query OK, 1 row affected (0.001 sec)

MariaDB [store]>
MariaDB [store]> INSERT INTO WRITING (Isbn, Name)
    -> VALUES ('9783161', 'Fyodor Dostoevsky');
Query OK, 1 row affected (0.001 sec)

MariaDB [store]>
MariaDB [store]> INSERT INTO AUTHOR (Name, SoldCopies)
    -> VALUES ('Fyodor Dostoevsky', 500);
Query OK, 1 row affected (0.001 sec)

MariaDB [store]>
MariaDB [store]> INSERT INTO Customer (cust_id, Principal_amount, Rate_of_interest
   , Years)
    -> VALUES (1, 1000, 0.05, 5);
Query OK, 1 row affected (0.001 sec)

MariaDB [store]>
MariaDB [store]> -- Create Triggers
MariaDB [store]>
MariaDB [store]> DELIMITER //
MariaDB [store]> CREATE TRIGGER update_author_soldcopies
    -> AFTER UPDATE ON BOOK
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.SoldCopies != OLD.SoldCopies THEN
    ->         UPDATE AUTHOR
    ->         SET SoldCopies = SoldCopies + (NEW.SoldCopies - OLD.SoldCopies)
   WHERE Name IN (SELECT Name FROM WRITING WHERE Isbn = NEW.Isbn);
    ->     END IF;
    -> END//
Query OK, 0 rows affected (0.003 sec)

MariaDB [store]> DELIMITER ;
MariaDB [store]>
MariaDB [store]> delimiter $$
MariaDB [store]> CREATE TRIGGER insert_author_soldcopies
    -> AFTER INSERT ON WRITING
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE AUTHOR
    ->     SET SoldCopies = SoldCopies + (SELECT SoldCopies FROM BOOK WHERE Isbn =
     NEW.Isbn)
    ->     WHERE Name = NEW.Name;
    -> END $$
Query OK, 0 rows affected (0.003 sec)

MariaDB [store]> delimiter ;
MariaDB [store]>
MariaDB [store]> select * from BOOK;
+---------+---------------------+------------+
| Isbn    | Title               | SoldCopies |
+---------+---------------------+------------+
| 9783161 | Crime and Punishment |       500 |
+---------+---------------------+------------+
1 row in set (0.002 sec)

MariaDB [store]> select * from AUTHOR;
+-------------------+------------+
| Name              | SoldCopies |
+-------------------+------------+
| Fyodor Dostoevsky |        500 |
+-------------------+------------+
1 row in set (0.001 sec)

MariaDB [store]> select * from WRITING;
+---------+-------------------+
| Isbn    | Name              |
+---------+-------------------+
| 9783161 | Fyodor Dostoevsky |
+---------+-------------------+
1 row in set (0.001 sec)

MariaDB [store]> select * from Customer;
+---------+------------------+------------------+-------+
| cust_id | Principal_amount | Rate_of_interest | Years |
+---------+------------------+------------------+-------+
|       1 |             1000 |             0.05 |     5 |
+---------+------------------+------------------+-------+
1 row in set (0.001 sec)

-- Test Triggers


MariaDB [store]> select * from AUTHOR;
+-------------------+------------+
| Name              | SoldCopies |
+-------------------+------------+
| Fyodor Dostoevsky |        500 |
+-------------------+------------+
1 row in set (0.001 sec)

MariaDB [store]> UPDATE BOOK SET SoldCopies = 600 WHERE Isbn = '9783161';
Query OK, 1 row affected (0.004 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [store]> select * from AUTHOR;
+-------------------+------------+
| Name              | SoldCopies |
+-------------------+------------+
| Fyodor Dostoevsky |        600 |
+-------------------+------------+
1 row in set (0.000 sec)
```

9 Conclusion

Thus, we have learned creating and using triggers in SQL. We have also learned the advantages and disadvantages of using triggers in SQL.

10 FAQ

1. **Enlist Advantages of Triggers ?**

   Advantages of Triggers:

   - Data Integrity: Triggers can help to ensure that data in a database remains consistent and accurate, by automatically enforcing data validation rules.
   - Automation: Triggers can automate repetitive or complex database operations, such as updating related records or sending notifications based on specific events.
   - Security: Triggers can be used to implement security policies, such as preventing unauthorized access or monitoring user activity.
   - Performance: Triggers can improve database performance by reducing the number of queries needed to perform certain operations.
   - Audit trail: Triggers can be used to create an audit trail of changes made to data, which can be useful for compliance or troubleshooting purposes.

2. **Enlist Disadvantages of Triggers ?**

   Disadvantages of Triggers:

   - Complexity: Triggers can add complexity to database design and maintenance, making it harder to understand and modify the database schema.
   - Debugging: Debugging triggers can be difficult, as they are executed automatically and can be triggered by a wide range of events.
   - Performance: Triggers can also have a negative impact on database performance, particularly if they are poorly designed or executed frequently.
   - Scalability: Triggers can make it more difficult to scale a database system, as they can increase the number of transactions and the amount of data being processed.

3. **What are the Applications of Triggers.**

   Applications of Triggers:

   - Data validation: Triggers can be used to enforce data validation rules, such as checking that certain fields are not left blank or that certain values fall within a specified range.
   - Data synchronization: Triggers can be used to synchronize data between different tables or databases, ensuring that related records are always up-to-date.
   - Notifications: Triggers can be used to send notifications or alerts based on specific events, such as when a new record is added or an existing record is updated.
   - Security: Triggers can be used to implement security policies, such as preventing unauthorized access or monitoring user activity.
   - Audit trail: Triggers can be used to create an audit trail of changes made to data, which can be useful for compliance or troubleshooting purposes.