DBMS Assignment A8

```
mysql> create table library(bno integer,bname varchar(40),author varchar(30),issue_for
integer);
Query OK, 0 rows affected (1.95 sec)

mysql> insert into library values(1,"Harry Potter","J.K. Rowling",15);
Query OK, 1 row affected (0.10 sec)

mysql> insert into library values(2,"Diary of a Wimpy Kid","Jeff Kenney",20);
Query OK, 1 row affected (0.07 sec)

mysql> insert into library values(3,"The Time Machine","H.G. Wells",8);
Query OK, 1 row affected (0.06 sec)

mysql> insert into library values(4,"Invisible Man","Ralph Ellison",25);
Query OK, 1 row affected (0.21 sec)

mysql> insert into library values(5,"The War of the Worlds","H.G. Wells",10);
Query OK, 1 row affected (0.06 sec)

mysql> insert into library values(6,"The Diary of a Young Girl","Anne Frank",8);
Query OK, 1 row affected (0.07 sec)

mysql> select * from library;
+------+--------------------------+------------------+-----------+
| bno  | bname                    | author           | issue_for |
+------+--------------------------+------------------+-----------+
|    1 | Harry Potter             | J.K.Rowling      |        15 |
|    2 | Diary of a Wimpy Kid     | Jeff Kenney      |        20 |
|    3 | The Time Machine         | H.G.Wells        |         8 |
|    4 | Invisible Man            | Ralph Ellison    |        25 |
|    5 | The War of the Worlds    | H.G.Wells        |        10 |
|    6 | The Diary of a Young Girl | Anne Frank       |         8 |
+------+--------------------------+------------------+-----------+


mysql> create table library_audit(bno integer,prev_issue_days integer,upd_issue_days
integer);
Query OK, 0 rows affected (0.79 sec)

mysql> desc library_audit;
+----------------+------+------+-----+---------+-------+
| Field          | Type | Null | Key | Default | Extra |
+----------------+------+------+-----+---------+-------+
| bno            | int  | YES  |     | NULL    |       |
| prev_issue_day | int  | YES  |     | NULL    |       |
| upd_issue_days | int  | YES  |     | NULL    |       |
+----------------+------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

```
mysql> delimiter &&
mysql> create trigger t1 before update on library for each row
-> begin
-> insert into library_audit values(old.bno,old.issue_for,new.issue_for);
-> end
-> &&
Query OK, 0 rows affected (0.67 sec)

mysql> select * from library_audit;
Empty set (0.08 sec)

mysql> update library set issue_for=16 where bno=3;
Query OK, 1 row affected (0.11 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from library_audit;&&
+------+-----------------+----------------+
| bno  | prev_issue_days | upd_issue_days |
+------+-----------------+----------------+
|   3  | 8               | 16             |
+------+-----------------+----------------+
1 row in set (0.00 sec)

mysql> update library set issue_for=25 where bno=2;
Query OK, 1 row affected (0.28 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from library_audit;&&
+------+-----------------+----------------+
| bno  | prev_issue_days | upd_issue_days |
+------+-----------------+----------------+
|   3  | 8               | 16             |
|   2  | 20              | 25             |
+------+-----------------+----------------+
2 rows in set (0.00 sec)

mysql> create trigger t2 before delete on library for each row
-> begin
-> insert into library_audit values(old.bno,old.issue_for,old.issue_for);
-> end&& Query OK,
0 rows affected (0.43 sec)

mysql> delete from library where bno=6;
Query OK, 1 row affected (0.23 sec)
mysql> select * from library;
```

```
+------+-----------------------+------------------+-----------+
| bno  | bname                 | author           | issue_for |
+------+-----------------------+------------------+-----------+
|    1 | Harry Potter          | J.K.Rowling      |        15 |
|    2 | Diary of a Wimpy Kid  | Jeff Kenney      |        25 |
|    3 | The Time Machine      | H.G.Wells        |        16 |
|    4 | Invisible Man         | Ralph Ellison    |        25 |
|    5 | The War of the Worlds | H.G.Wells        |        10 |
+------+-----------------------+------------------+-----------+
```

mysql> select * from library_audit;

```
+------+----------------+----------------+
| bno  | prev_issue_day | upd_issue_days |
+------+----------------+----------------+
|    3 |              8 |             16 |
|    2 |             20 |             25 |
|    6 |              8 |              8 |
+------+----------------+----------------+
```

3 rows in set (0.03 sec)

MES College of Engineering Pune-01

Department of Computer Engineering

Name of Student: Dhruvil Shah	**Class: TE Comp 1**
Semester/Year: 5th/2020	**Roll No: 047**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part A-08**

GROUP: A ASSIGNMENT NO: 08

AIM: Write a PL/SQL block of code for Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).

PROBLEM STATEMENT:

Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.

OBJECTIVES:

- To study the concept a trigger.
- **T**o learn various Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).

PRE - REQUISITES:

Interactive SQL commands, PL/SQL programming, use of oracle 11g database Editor.

APPARATUS:

- Operating System recommended: 64-bit Open source Linux or its derivative
- Front End :- Oracle Editor
- Back end: Oracle 11g

SYNTAX:

Trigger:

- CREATE [OR REPLACE] TRIGGER trigger_name
- {BEFORE | AFTER | INSTEAD OF }
- {INSERT [OR] | UPDATE [OR] | DELETE}
- [OF col_name]

- ON table_name
- [REFERENCING OLD AS o NEW AS n]
- [FOR EACH ROW]
- WHEN (condition)
- DECLARE
- Declaration-statements
- BEGIN
- Executable-statements
- EXCEPTION
- Exception-handling-statements
- END;

Where

- CREATE [OR REPLACE] TRIGGER trigger_name − Creates or replaces an existing trigger with the *trigger_name*.
- {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
- [OF col_name] − This specifies the column name that will be updated.
- [ON table_name] − This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) − This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.
- :old and :new are the system variables used to get the older and newer values before or after the changes are made to the master table.

CONCLUSION:

QUESTIONS:

1. What is a trigger?
2. What are Benefits of Triggers?
3. What are **Row** triggers and **Statement** triggers?
4. Why are we using **Before** and **After** triggers?
5. What is Insert, Update and Delete triggers?

Q1 What is trigger?

Ans A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

Syntax:

```
create trigger [trigger_name]
[ before / after ]
{ insert / update / delete } on [table_name] [for each row]
[ trigger body ].
```

Q2 What are benefits of Trigger?

Ans The benefits of trigger are

1] Generating some derived column values automatically
2] Enforcing referential integrity
3] Auditing
4] Synchronous replication of tables.
5] Imposing security authorizations
6] Prevent invalid transactions

Q3 What are ROW triggers and STATEMENT triggers?

Ans ROW Triggers:
Row level triggers execute once for each row in transaction. Row level triggers are the most common type of triggers, they are often used in data auditing applications. Row-level trigger is identified by each row row clause in the CREATE Trigger command.

STATEMENT Trigger:
Statement trigger executes once for each transaction. Statement triggers are not often used for data related activities, they are normally used to enforce additional security measures on the types of transactions that may be performed on a table. Statement triggers are the default type of triggers created and are identified by omitting FOR EACH ROW clause in the CREATE TRIGGER command.

Q4 Why are we using Before and After trigger?

Ans We use before triggers when we want to update any field or validate any record before they are save to the database
After triggers are used when we wish to access any field values after they are saved to the data base

Q5 What is insert, update and delete triggers?

Ans DML triggers execute when a user tries to modify data through a DML event. They can be either BEFORE or AFTER triggers. Triggers on DML statements include following triggers

BEFORE INSERT, BEFORE UPDATE, BEFORE DELETE, AFTER INSERT, AFTER UPDATE, AFTER DELETE.