

Assignment No. 8

Aim: Write and execute suitable database triggers .Consider row level and statement level triggers.

Objective:

- To study and implement PL/SQL triggers.

Theory :

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events.

- A **database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
- A **database definition (DDL)** statement (CREATE, ALTER, or DROP).
- A **database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

Benefits of Triggers

Triggers can be written for the following purposes –

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

Creating Triggers

The syntax for creating a trigger is –

```
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.

Conclusion:-

We have studied and executed different types of database triggers.

Output:

1. CREATE TABLE:

a) EMP:

```
CREATE TABLE EMP(
    EMP_NUM INT(5) AUTOINCREMENT,
    FIRSTNAME VARCHAR(20),
    LASTNAME VARCHAR (20),
    CHANGEDATE DATE
);
```

The screenshot shows the DB Browser for SQLite interface. The top menu bar includes File, Edit, View, Tools, Help, New Database, Open Database, Write Changes, Revert Changes, Open Project, Save Project, Import Database, and Close Database. Below the menu is a toolbar with icons for New Database, Open Database, Write Changes, Revert Changes, Open Project, Save Project, Import Database, and Close Database. The main window has tabs for Database Structure, Browse Data, Edit Progress, and Summary SQL. On the left, there's a tree view of the database structure. In the center, a SQL editor window contains the following code:

```
SELECT 1 FROM EMP;
```

Below the SQL editor is a table named 'EMP' with columns EMP_NUM, FIRSTNAME, LASTNAME, and CHANGEDATE. The data is as follows:

EMP_NUM	FIRSTNAME	LASTNAME	CHANGEDATE
1 101	Wise	Sylvia	30-06-2005
2 101	Pig	Pell	27-09-1998
3 102	Vikes	Guru	15-04-1997
4 103	SureB	Odeval	05-09-1990
5 104	Presti	Pugal	26-03-1998

At the bottom of the table area, it says "Results: 5 rows affected in 14ms" and "Ex. line 1:" followed by the SQL query "SELECT * FROM EMP;". To the right of the table, there's a "SQL Log" window showing the history of queries:

```
507 SELECT * FROM `EMP` ORDER BY 1
508 SELECT COUNT() FROM (SELECT `empid` , 1 FROM
509 EMP , `EMPAUDIT` SET `TRANSACTION` = 1
510 SELECT COUNT() FROM (SELECT `empid` , 1 FROM
511 `empid` , 1 FROM `EMP` ORDER BY
512 SELECT COUNT() FROM (SELECT `empid` , 1 FROM
513 `EMP` , `EMPAUDIT` SET `TRANSACTION` = 1
514 UPDATE `EMP` , `EMPAUDIT` SET `TRANSACTION` = 1
515 SELECT COUNT() FROM (SELECT * FROM EMP
516 SELECT * FROM EMP LIMIT 1, 12345)
```

b) EMPAUDIT:

```
CREATE TABLE "EMPAUDIT" (
    "ID" INTEGER AUTOINCREMENT,
    "EMP_NUM" INTEGER,
    "LASTNAME" TEXT,
    "CHANGEDAT" DATE
);
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 8\Assignment 8.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```
1  SELECT * FROM EMPAUDIT;
```

ID	EMP_NUM	LASTNAME	CHANGEDAT
1	100	Shrivastava	30-04-2005
2	101	Patil	27-08-1998
3	102	Guru	13-04-1997
4	103	Chavan	05-06-1980
5	104	Pusad	28-03-1996

Result: 5 rows returned in 20ms
At line 1:
SELECT * FROM EMPAUDIT;

Edit Database Cell

Mode: Text Import Export Set as NULL

NULL

Type of data currently in cell: NULL
0 byte(s)

Apply

SQL Log

Show SQL submitted by Application Clear

```
609 UPDATE "main"."EMPAUDIT" SET "CHANGEDAT"=? WE ^  
610 SELECT COUNT(*) FROM (SELECT "_rowid_",'*'  
611 SELECT "_rowid_",'*' FROM "main"."EMP" ORDER BY  
612 SELECT COUNT(*) FROM (SELECT "_rowid_",'*'  
613 SELECT "_rowid_",'*' FROM "main"."EMPAUDIT" OR  
614 UPDATE "main"."EMPAUDIT" SET "CHANGEDAT"=? WE  
615 UPDATE "main"."EMPAUDIT" SET "CHANGEDAT"=? WE  
616 SELECT COUNT(*) FROM (SELECT '*' FROM EMP);  
617 SELECT '*' FROM EMP LIMIT 0, 49999;  
618 SELECT COUNT(*) FROM (SELECT '*' FROM EMPAUDIT)  
619 SELECT '*' FROM EMPAUDIT LIMIT 0, 49999;  
620
```

SQL Log Plot DB Schema Remote

UTF-8

2. CREATE TRIGGER (BEFORE INSERTING NEW VALUE IN TABLE):

```
CREATE TRIGGER insert_trigger BEFORE INSERT ON EMP
FOR EACH ROW
BEGIN
    UPDATE EMP SET EMP_NUM = EMP_NUM-400 ;
END;
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 8\Assignment 8.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```
1  CREATE TRIGGER insert_trigger BEFORE INSERT ON EMP
2  FOR EACH ROW
3  BEGIN
4      UPDATE EMP SET EMP_NUM = EMP_NUM-400 ;
5  END;
6
```

Result: 'query executed successfully. Took 1ms'
At line 1:
CREATE TRIGGER insert_trigger BEFORE INSERT ON EMP
FOR EACH ROW
BEGIN
 UPDATE EMP SET EMP_NUM = EMP_NUM-400 ;
END;

Edit Database Cell

Mode: Text Import Export Set as NULL

NULL

Type of data currently in cell: NULL
0 byte(s)

Apply

SQL Log

Show SQL submitted by Application Clear

```
698 PRAGMA recursive_triggers
699 PRAGMA secure_delete
700 PRAGMA synchronous
701 PRAGMA temp_store
702 PRAGMA user_version
703 PRAGMA wal_autocheckpoint
704 SELECT '*' NOT LIKE '%'
705 DROP TRIGGER "main"."insert_trigger";
706 PRAGMA database_list;
707 SELECT type,name,sql,tbl_name FROM "main".sql
708 SELECT type,name,sql,tbl_name FROM sqlite_temp
709 PRAGMA database_list;
710 SELECT type,name,sql,tbl_name FROM "main".sql
```

SQL Log Plot DB Schema Remote

3. INSERT VALUE IN EMP TABLE:

INSERT INTO EMP VALUES(105,'Praharsh','Kumar',1995-10-20);

The screenshot shows the DB Browser for SQLite interface. In the SQL tab, the command `SELECT * FROM EMP;` is run, resulting in 6 rows returned. A new row is inserted with the values: EMP_NUM = 105, FIRSTNAME = Praharsh, LASTNAME = Kumar, and CHANGEDATE = 1995-10-20. The database structure pane shows the EMP table with columns: EMP_NUM, FIRSTNAME, LASTNAME, and CHANGEDATE. The data grid displays the updated table with 7 rows.

EMP_NUM	FIRSTNAME	LASTNAME	CHANGEDATE
1 -300	Nikhil	Shrivastava	30-04-2005
2 -299	Raj	Patil	27-08-1998
3 -298	Vikas	Guru	13-04-1997
4 -297	Suresh	Chavan	05-06-1980
5 -296	Preeti	Pusad	28-03-1996
6 105	Praharsh	Kumar	1965

Result: 6 rows returned in 38ms
At line 1:
SELECT * FROM EMP;

SQL Log pane shows the executed SQL statements, including the insertion command at line 713.

4. DROP TRIGGER:

`DROP TRIGGER insert_trigger;`

The screenshot shows the DB Browser for SQLite interface. In the SQL tab, the command `DROP TRIGGER insert_trigger;` is run, resulting in a query executed successfully message. The database structure pane shows the EMP table with columns: EMP_NUM, FIRSTNAME, LASTNAME, and CHANGEDATE. The data grid displays the updated table with 7 rows.

Result: query executed successfully. Took 0ms
At line 1:
DROP TRIGGER insert_trigger;

SQL Log pane shows the executed SQL statements, including the trigger drop command at line 714.

5. UPDATE TRIGGER:

a) BEFORE UPDATE:

`CREATE TRIGGER update_trigger BEFORE UPDATE ON EMP FOR EACH ROW`

```

BEGIN
    UPDATE EMP SET EMP_NUM = EMP_NUM-200 ;

```

The screenshot shows the DB Browser for SQLite interface. In the SQL tab, the following SQL code is entered:

```

1 CREATE TRIGGER update_trigger BEFORE UPDATE ON EMP
2 FOR EACH ROW
3 BEGIN
4     UPDATE EMP SET EMP_NUM = EMP_NUM-200 ;
5 END;

```

The result pane shows the trigger was created successfully:

```

Result: query executed successfully. Took 1ms
At line 1:
CREATE TRIGGER update_trigger BEFORE UPDATE ON EMP
FOR EACH ROW
BEGIN
    UPDATE EMP SET EMP_NUM = EMP_NUM-200 ;
END;

```

END;

```

UPDATE EMP
SET FIRSTNAME = 'Raj'
WHERE EMP_NUM = 106;

```

The screenshot shows the DB Browser for SQLite interface. In the SQL tab, the following SQL code is entered:

```

1 SELECT * FROM EMP;

```

The results pane displays the contents of the EMP table:

EMP_NUM	FIRSTNAME	LASTNAME	CHANGEDATE
1 -500	Nikhil	Shrivastava	30-04-2005
2 -499	Raj	Patil	27-08-1998
3 -498	Vikas	Guru	13-04-1997
4 -497	Suresh	Chavan	05-06-1980
5 -496	Preeti	Pusad	28-03-1996
6 -95	Praharsh	Kumar	1965
7 -94	Raj	Pillai	1964

The result pane shows the query returned 7 rows:

```

Result: 7 rows returned in 11ms
At line 1:
SELECT * FROM EMP;

```

DROP TRIGGER update_trigger;

6. AFTER:

CREATE TRIGGER after_insert AFTER INSERT ON EMP FOR EACH ROW

BEGIN

 UPDATE EMP SET EMP_NUM = EMP_NUM - 1;

The screenshot shows the DB Browser for SQLite interface. In the SQL tab, the following SQL code is entered:

```
1 CREATE TRIGGER after_insert AFTER INSERT ON EMP
2 FOR EACH ROW
3 BEGIN
4     UPDATE EMP SET EMP_NUM = EMP_NUM - 1;
5 END;
```

The result pane shows the trigger was created successfully:

```
Result: query executed successfully. Took 1ms
At line 1:
CREATE TRIGGER after_insert AFTER INSERT ON EMP
FOR EACH ROW
BEGIN
    UPDATE EMP SET EMP_NUM = EMP_NUM - 1;
END
```

On the right side, there are panes for "Edit Database Cell" (showing NULL), "SQL Log" (listing recent database operations), and "Database Structure" (showing the EMP table structure).

END

INSERT INTO EMP VALUES(106,'Tarun','Sonawane',19-08-1998);

The screenshot shows the DB Browser for SQLite interface. In the SQL tab, the following SQL code is entered:

```
1 SELECT * FROM EMP;
```

The result pane shows the current data in the EMP table:

EMP_NUM	FIRSTNAME	LASTNAME	CHANGEDATE
1 -501	Nikhil	Shrivastava	30-04-2005
2 -500	Raj	Patil	27-08-1998
3 -499	Vikas	Guru	13-04-1997
4 -498	Suresh	Chavan	05-06-1980
5 -497	Preeti	Pusad	28-03-1996
6 -96	Praharsh	Kumar	1965
7 -95	Raj	Pillai	1964
8 105	Tarun	Sonawane	-1987

The SQL log on the right shows the history of operations, including the insertion of the new row.

```
DROP TRIGGER after_insert;
```

7. AFTER UPDATE:

```
CREATE TRIGGER after_update after UPDATE ON EMP
FOR EACH ROW
BEGIN
    UPDATE EMP SET EMP_NUM = EMP_NUM-50 ;
END
```

```
INSERT INTO EMP VALUES (100, 'Viraj', 'Kate', 25-03-2001);
```

The screenshot shows the DB Browser for SQLite interface. In the top navigation bar, the database is set to 'Assignment 8.db'. The main window has tabs for 'Database Structure', 'Browse Data', 'Edit Pragmas', and 'Execute SQL'. The 'Execute SQL' tab is active, displaying the following SQL code:

```
1 CREATE TRIGGER after_update after UPDATE ON EMP
2 FOR EACH ROW
3 BEGIN
4     UPDATE EMP SET EMP_NUM = EMP_NUM-50 ;
5 END
```

Below the code, the results of the execution are shown:

```
Result: query executed successfully. Took 3ms
At line 1:
CREATE TRIGGER after_update after UPDATE ON EMP
FOR EACH ROW
BEGIN
    UPDATE EMP SET EMP_NUM = EMP_NUM-50 ;
end
```

To the right of the main window, there is an 'Edit Database Cell' panel with a text input field containing 'NULL'. Below it, a message says 'Type of data currently in cell: NULL 0 byte(s)'. There is also a 'SQL Log' panel showing the history of SQL queries:

```
744 SELECT "_rowid_," FROM "main"."EMP" ORDER BY ...
745 SELECT COUNT(*) FROM (SELECT * FROM EMP);
746 SELECT * FROM EMP LIMIT 0, 49999;
747 SELECT "_rowid_," FROM "main"."EMP" ORDER BY ...
748 SELECT COUNT(*) FROM (SELECT "_rowid_," FROM ...
749 PRAGMA database_list;
750 SELECT type,name,sql,tbl_name FROM "main".sql
751 SELECT type,name,sql,tbl_name FROM sqlite_tes...
752 PRAGMA database_list;
753 SELECT type,name,sql,tbl_name FROM "main".sql
754 SELECT type,name,sql,tbl_name FROM sqlite_tes...
755 PRAGMA database_list;
756 SELECT type,name,sql,tbl_name FROM "main".sql
```

```
UPDATE EMP
SET FIRSTNAME = 'Viru'
WHERE EMP_NUM = 100;
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 8\Assignment 8.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```
1 SELECT * FROM EMP;
```

EMP_NUM	FIRSTNAME	LASTNAME	CHANGEDATE
1 -551	Nikhil	Shrivastava	30-04-2005
2 -550	Raj	Patil	27-08-1998
3 -549	Vikas	Guru	13-04-1997
4 -548	Suresh	Chavan	05-06-1980
5 -547	Preeti	Pusad	28-03-1996
6 -146	Praharsh	Kumar	1965
7 -145	Raj	Pillai	1964
8 50	Viru	Kate	-1979
9 55	Tarun	Sonawane	-1987

Result: 9 rows returned in 25ms
At line 1:
SELECT * FROM EMP;

Edit Database Cell

Mode: Text Import Export Set as NULL

NULL

Type of data currently in cell: NULL
0 byte(s) Apply

SQL Log

Show SQL submitted by Application Clear

```
751 SELECT type,name,sql,tbl_name FROM sqlite_temp^
752 PRAGMA database_list;
753 SELECT type,name,sql,tbl_name FROM "main".sql
754 SELECT type,name,sql,tbl_name FROM sqlite_temp
755 PRAGMA database_list;
756 SELECT type,name,sql,tbl_name FROM "main".sql
757 SELECT type,name,sql,tbl_name FROM sqlite_temp
758 SELECT COUNT(*) FROM (SELECT * FROM EMP);
759 SELECT * FROM EMP LIMIT 0, 49999;
760 SELECT COUNT(*) FROM (SELECT * FROM EMP);
761 SELECT * FROM EMP LIMIT 0, 49999;
762
```

SQL Log Plot DB Schema Remote

UTF-8

The screenshot shows the DB Browser for SQLite interface. In the main window, a query `SELECT * FROM EMP;` is run, returning 9 rows of data from a table with columns EMP_NUM, FIRSTNAME, LASTNAME, and CHANGEDATE. The results are displayed in a grid. Below the grid, the output shows 9 rows returned in 25ms and the executed SQL statement. To the right, there are two panes: 'Edit Database Cell' (currently showing NULL) and 'SQL Log' (listing several SQL statements, likely from a previous session or a script). The SQL Log pane includes dropdowns for 'Show SQL submitted by' (set to Application) and 'Clear'. The bottom right corner of the interface shows the character encoding as UTF-8.

DROP TRIGGER after_update;