Practical No. 5

Aim : Implementing SQL commands for triggers.

Theory :-

* SQL Triggers :-

In SQL Server, triggers are database objects, actually, a special kind of stored procedure, which "reacts" to certain actions we make in the database. The main idea behind triggers is that they always perform an actions in case some event happens.

* Types of SQL Triggers :-

In SQL Server, we have 3 groups of triggers :

· DML (data manipulation language) triggers

We use DML SQL Triggers in the case when we want to assure that a certain control shall be performed before or after the defined statement on the defined table. This could be the case when your code is all over the place.
e.g. Database is used by different applications, code is written directly in applications and you don't have well documented.

They react to DML commands. These are - INSERT, UPDATE and DELETE.

· DDL (data definition language) triggers

As expected, triggers of this type shall react to DDL commands like - CREATE, ALTER and DROP.

Chhatrapati Shahu Maharaj Shikshan Sanstha's
CHH. SHAHU COLLEGE OF ENGINEERING
Kanchanwadi, Paithan Road, Aurangabad.
Date :

· Logon triggers:
The name says it all This type reacts to LOGON events.

* DML Triggers:
We know that, DML Triggers is a special type of stored procedure that automatically takes effect when a data manipulation language (DML) event takes place that affects the table or view defined in the trigger.
DML events include INSERT, UPDATE, or DELETE statements. DML triggers can be used to enforce business rules and data integrity, query other tables, and include complex Transact-SQL statements. The trigger and the statement that fires it are treated as a single transaction which can be rolled back from within the triggers.
If a severe error is detected (for example, insufficient disk space), the entire transaction automatically rolls back.

Syntax :

```
CREATE TRIGGER [schema_name.] trigger_name
ON table_name
{FOR | AFTER | INSTEAD OF} {[INSERT] [,] [UPDATE] [,] [DELETE]}
AS
{sql_statements}
```

Chhatrapati Shahu Maharaj Shikshan Sanstha's
CHH. SHAHU COLLEGE OF ENGINEERING
Kanchanwadi, Paithan Road, Aurangabad.

Date :

Practical No. 6.

Aim : Normalization (1NF, 2NF, 3NF, BCNF) in database.

Theory :

* Normalization :

Normalization is the process of organizing data in the database. It is used to minimize redundancy from a relation. Normalization divides the larger table into smaller table and link them using relations. It is used to eliminate, insert, update, delete anomalies.

* 1NF :

Relation will be in 1NF if it contains an atomic values. An attribute of a table can not hold multiple values. It must hold single value attribute.

1NF doesn't allow multivalued attributes, composite attribute and their combinations.

e.g. Table 1

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	INDIA
		9877117778		
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

↓ Conversion to 1NF

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	INDIA
1	RAM	9877117778	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

* 2NF :-

1NF does not eliminate redundancy but eliminating repeating values. 2NF is based on the concept of full functional dependancies. The relation is in 2NF.

① If it is in 1NF.

② It should not contain partial dependancy. If the proper subset of candidate key determine the non-prime attributes then it is called as partial dependancy. The normalization of 1NF to 2NF involve removal of partial dependancy. If partial dependancy exist we remove partial dependant attribute from the relation by placing them in a new relation.

Stud-no	Course-no	Fee
1	C_1	1000
2	C_2	1500
1	C_4	2000
4	C_3	1000
4	C_1	1000
2	C_5	2000

It is not in 2NF.

f.d = { {stud.no, course-no} → fee, courseno → fee} }

Candidate key = {courseno, studno}

Prime attributes = { courseno, studno}

NPA = {fee}

as course-no → fee it is not in 2NF.

To convert this relation into 2NF we need to split this table into 2 tables.

Chhatrapati Shahu Maharaj Shikshan Sanstha's
CHH. SHAHU COLLEGE OF ENGINEERING
Kanchanwadi, Paithan Road, Aurangabad.
Date :

Table 1 = { studno, courseno}
Table 2 = { courseno, fee} } 2NF

Table 1

studno	courseno
1	C_1
2	C_2
1	C_4
4	C_3
4	C_1
2	C_5

Table 2

courseno	fee
C_1	1000
C_2	1500
C_4	2000
C_3	1000
C_1	1000
C_5	2000

* 3NF :

A relation is in 3NF if it is in 2NF and doesn't contain any transitive partial dependancy. If there is no transitive dependancy for non-prime attributes then relation must be in 3NF. A relation is in 3NF if it holds one of the following conditions for every non-trivial functional dependancy

① x is a super key } anyone
② y is prime attributes (each elements of y is part of some candidate key).

The normalization of 2NF to 3NF involve removal of transitive depandancies if transitive dependancy exist we remove the transitively dependent attributes from the relation by placing the attribute in a new relation.

studno	name	state	country	age
1	aaa	Punjab	India	20
2	bbb	Maharashtra	India	19
3	ccc	Maharashtra	India	20

fd : { studno → {name, state, country, age} state → {country} }

sk = ck = studno.

To convert this relation into 3NF we will decompose the relation in two tables

Table 1

studno	name	state	age
1	aaa	Punjab	20
2	bbb	Maharashtra	19
3	ccc	Maharashtra	20

Table 2

state	country
Punjab	India
Maharashtra	India
Maharashtra	India

* **Boyce Code Normal Form (BCNF):**

It is advanced version of 3NF the relation is in BCNF. If it in 3NF and non-trivial functional dependancy $x \to y$, x must be super

eg: R(A, B, C)

fd = {A → B, B → C, C → A}

→ ABC^+ = {A, B, C}

sk = A^+ = {A, B, C} = ck

sk = C^+ = {A, B, C} = ck

sk = B^+ = {B, C, A} = ck

Ck = {A, B, C}

P.A = {A, B, C}

$$\left.\begin{array}{l} 1)\ A \to B\ ✓ \\ 2)\ B \to C\ ✓ \\ 3)\ C \to A\ ✓ \end{array}\right\}\ \text{BCNF}$$

update became permanent and stored in non-volatile memory

· Commands of Transactions in SQL

1) Commit :- To save the changes.

2) Rollback :- To roll back the changes.

3) Savepoint :- Creates points within the groups of transactions in which to Rollback.

4) Set Transaction :- Places a name on a transaction.

Chhatrapati Shahu Maharaj Shikshan Sanstha's
CHH. SHAHU COLLEGE OF ENGINEERING
Kanchanwadi, Paithan Road, Aurangabad.

Date :

Practical No. 7

Aim : Implementing Transactions in SQL

Theory :

· Transaction :-

Transaction is a single logical unit of work which access and possibaly modifies the content of database. Transaction are accessed using read and write operation.

for. eg :

Account A
Read (x)
x = x - 500
write (x)

Account B
Read (y)
y = y + 1000
write (y)

· Properties of Transaction :-

a) **Automicity :**

This property states that either the entire transaction takes place at once or doesn't happen at all. Each transaction is considered as one unit and either run to completion or it is not executed at all. It involve 2 operation.

1) **Abort** : If the transaction abort changes made to database are not visible.

2) **Commit** : If transaction commit changes made by transaction are visible in database.

b) **Consistency :**

Database must be in consistent state before and after the transaction.

e.g.

A	B
$x = 2000$	$y = 3000$
read(x)	read(y)
$x = x - 1000$	$y = y + 1000$
write(x)	write(y)

$x + y = 1000 + 4000 = 5000$

c) Isolation :-

Isolation property ensures that multiple transaction executed conqurrently without leading to inconsistency of database state changes occurring in perticular transaction will not be visible to any other transaction until that perticular changes in that transaction is written to memory or has been committed.

e.g:

T_1	T_2	
read(x)	read(x)	$x = 500$
$x = x + 100$	read(y)	$y = 500$
write(x)	$z = x + y$	
read(y)	write(z)	
$y = y - 50$		
write(y)		

Transaction must take place in isolation and changes should be visible only after they have been made to the main memory.

d) Durability :-

This property ensures that once the transaction has completed execution. The update and modification to the database are stored and written to the disk and persist even if system failure occurs.