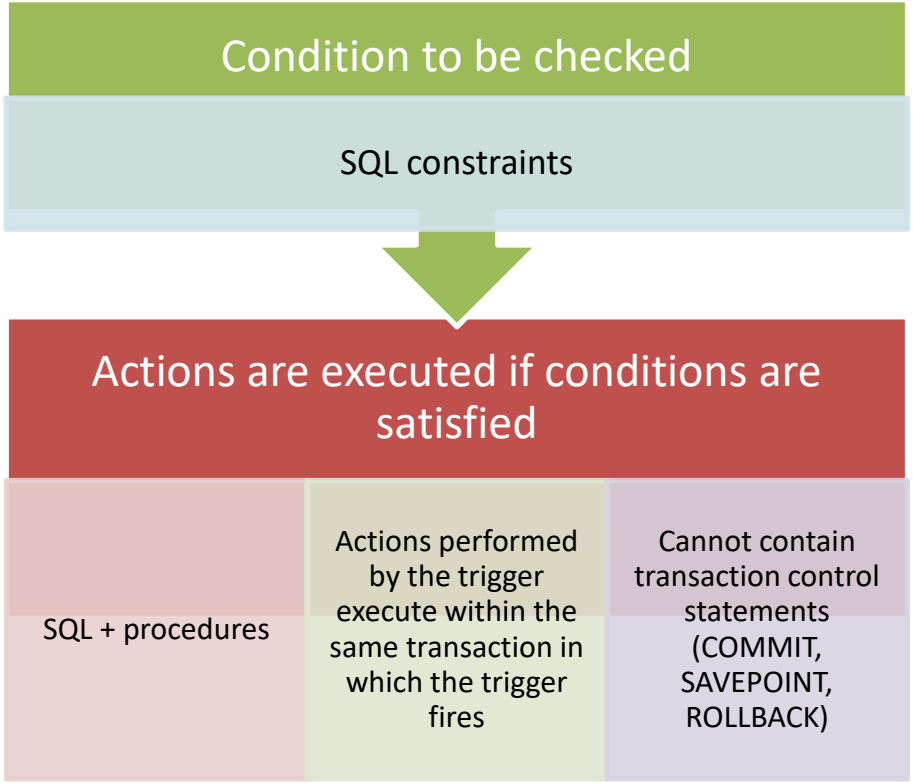
Triggers

COURSE : 23CAT- Database Management System
UNIT I : Introduction
CLASS : I Semester / I MCA

- Event occurs in databases when
  - addition of new row, deletion of row by DBMS



Triggers

It is specialized category of stored procedure that is called automatically when a database server event occurs

- Not specified in SQL-92, but standardized in SQL3 (SQL1999)
- Available in most enterprise DBMSs
- Some vendors
  - permit native extensions to SQL for specifying the triggers
  - Use general purpose programming language instead of SQL
  - extend the triggers beyond tables

Purpose of Trigger

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

Differences

Triggers	Procedure
It is called automatically when a data modification event occurs against table	It must be invoked directly
it can't take the input parameters	It can take the input parameters
It can't return a value	It can return a value
It can't use the transaction statements	Use the transaction statements like begin transaction, commit transaction and rollback inside a stored procedure
It can't schedule a trigger	It can be scheduled to execute on a predefined time,

Trigger - Syntax

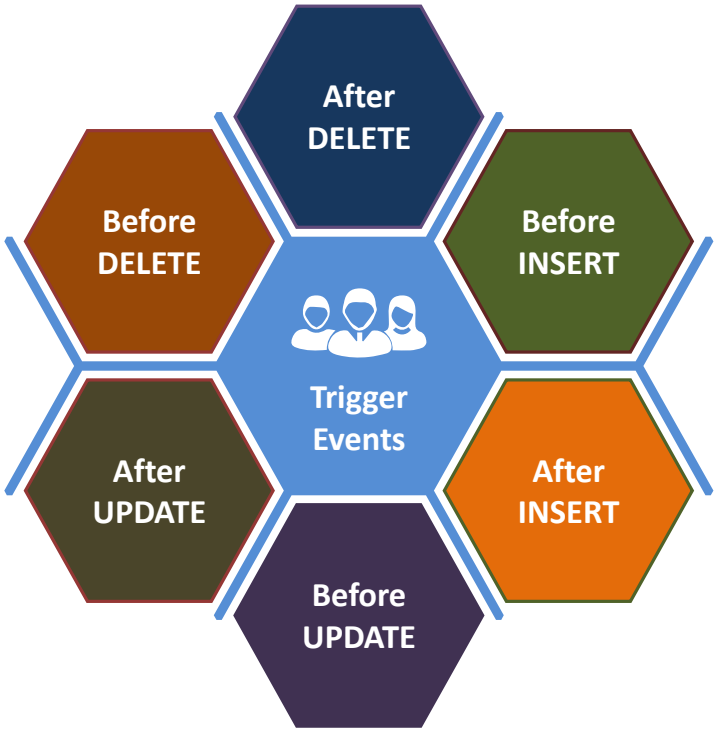
```
CREATE TRIGGER schema.trigger_name
ON table_name
AFTER  {INSERT/UPDATE/DELETE}
[NOT FOR REPLICATION]
AS  {SQL_Statements}
```

schema: optional, schema the new trigger belongs to
trigger_name: name for the new trigger
table_name: the table to which the trigger applies
SQL_Statements: one / more SQL statements that are used to perform actions in response to an event that occurs

```
CREATE TABLE Employee
(
  Id INT PRIMARY KEY,
  Name VARCHAR(45),
  Salary INT,
  Gender VARCHAR(12),
  DepartmentId INT
)
```

Firing Sequence of Trigger

1. Execute all BEFORE STATEMENT triggers
2. Disable temporarily all integrity constraints recorded against the table
3. Loop for each row in the table
   - Execute all BEFORE ROW triggers
     - Execute the SQL statement against the row and perform integrity constraint checking of the data
   - Execute all AFTER ROW triggers
4. Complete deferred integrity constraint checking against the table
5. Execute all AFTER STATEMENT triggers



Trigger Example

```
SQL>CREATE OR REPLACE TRIGGER derive_commission_trg
  2 BEFORE UPDATE OF sal ON emp
  3 FOR EACH ROW
  4 WHEN (new.job = 'SALESMAN')
  5 BEGIN
  6   :new.comm := :old.comm * (:new.sal/:old.sal);
  7 END;
  8 /
```

Trigger name:	derive_commission_trg
Timing:	BEFORE executing the statement
Triggering event:	UPDATE of sal column
Filtering condition:	job = 'SALESMAN'
Target:	emp table
Trigger parameters:	old, new
Trigger action:	calculate the new commission to be updated

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
  sal_diff number;
BEGIN
  sal_diff := :NEW.salary  - :OLD.salary;
  dbms_output.put_line('Old salary: ' || :OLD.salary);
  dbms_output.put_line('New salary: ' || :NEW.salary);
  dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

Triggering Event

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS, SALARY) VALUES (7, 'Kriti', 22, 'HP', 15000.00);

Output

```
Old salary:
New salary: 15000
Salary difference:
```

UPDATE customers SET salary = salary + 3000
WHERE id = 7;

Output

```
Old salary: 15000
New salary: 18000
Salary difference: 3000
```