View

- View is a database object which contains logical copy of data from table or tables.
- View does not store data hence it is also called virtual table.

Types of View

1. Simple View
2. Composite View / Join View

Simple View

- A view which is created based on single table data is called simple view.

<u>Create View</u>

<u>Exampl1:</u>

```
CREATE VIEW  employee_view1
AS
SELECT * FROM employees WHERE department_id=1;
```

<u>Example#2:</u>

```
CREATE OR REPLACE VIEW  employee_view1
AS
SELECT * FROM employees WHERE department_id=2;
```

<u>Get Data From View</u>

```
SELECT * FROM employee_view1;
```

Note

- View maintains data dynamically.

  ```
  INSERT INTO employees VALUES (107,'A',50000,2);
  ```

  ```
  SELECT * FROM employee_view1;
  ```

DML operation on simple view

- The view which is accepting DML operation on it is called as updatable view because table is reflecting.

  ```
  INSERT INTO employee_view1 VALUES (108,'B',50000,2);

  INSERT INTO employee_view1 VALUES (109,'B',50000,3);
  ```

Note

- We must include not null or primary key column into the view then only we are allowed to insert data from view to table.

Example#1

```
CREATE TABLE employee2(
        empid int primary key,
        name varchar(20),
        address varchar(20)
);

INSERT INTO employee2 values (1,'A','bbsr');

CREATE VIEW employee2_view2
as
SELECT name,address FROM employee2;
SELECT * FROM employee2_view2;

INSERT INTO employee2_view2(name,address) VALUES ('B','Address');
```

Composite View

- A view which is created based on multiple table data is called composite view or complex view or join.

  Example#1

  ```
  CREATE VIEW report_view1 AS
  SELECT employee_name,department_name from employees,departments
  WHERE employees.department_id = departments.department_id;
  ```

Example#2

Write a view to store 2nd highest salary employee details.

```
create view employeesalary
as
select * from employees where salary = (select max(salary) from employees where
salary<(select max(salary) from employees));
```

Note

- It is not possible perform insert operation.

Conclusion

- View is used to simplify complex query. We no need to write complex query multiple times.

TCL

- A set of one or more SQL statements that are logically related is called a Transaction.
- Atomicity property says that all the statements should execute successfully or none of the statements should execute.

TCL Commands

- Commit
- Rollback
- Savepoint

Note

```
SHOW VARIABLES LIKE 'autocommit';
SET autocommit = 0;
```

Commit

Commit command is used to permanently save any transaction into database.

```
INSERT INTO employees VALUES(111,'B',50000,1);
```

```
COMMIT;
```

Rollback

This command restores the database to last committed state.

```
INSERT INTO employees VALUES(111,'B',50000,1);

ROLLBACK;
```

Savepoint

Identify a point in a transaction to which you can later roll back.

```
SAVEPOINT sp1;

INSERT INTO employees VALUES(114,'B',50000,1);

INSERT INTO employees VALUES(115,'B',50000,1);

SAVEPOINT sp2;

INSERT INTO employees VALUES(116,'B',50000,1);

INSERT INTO employees VALUES(117,'B',50000,1);

ROLLBACK TO SAVEPOINT sp2;

COMMIT;

ROLLBACK;

SELECT * FROM employees;
```

Note

- DML statements can be rollbacked where DDL are auto commit.
- DML commands can't be rollback when a DDL command is executed immediately after a DML. DDL after DML means "auto commit".

  ```
  INSERT INTO employees VALUES(116,'B',50000,1);

  CREATE TABLE emp(eno int);

  ROLLBACK;
  ```

Enable Auto commit

```
SHOW VARIABLES LIKE 'autocommit';
SET autocommit = 1;
```

Disable Auto commit

```
SET autocommit = 0;
```

How to work with transaction when autocommit is on

START TRANSACTION

- It is used to create new transaction.
- When we apply commit or rollback then it will stop the transaction.