In DBMS, there are several types of languages used to interact with the database.

1) DDL (data defined language)

it is used to define and manage the structure of the database.

- create
- Alter
- drop
- truncate

2) DML (data manipulation language)

it is used for manipulate the data within the database.

- insert
- update
- delete

3) DQL (data query language)

it is used to query and retrieve data from the database

- select

4) TCL (transaction control language)

it is used for managing transaction in database.

roll back

commit

5) DCL (data control language)

it is used to control access of data within the database.

grant :- it allow to do some operations on database objects.

revoke :- to remove the permission on database objects.

grant privileges on table_name to 'user_name';

revoke privileges on table_name from 'username';

privileges : select, update, delete, Alter.. etc.

* if we only give update privileges then we can't update without select priveleges

views

views in database are like virtual table generated by SQL query.

student, instructor, department

(st_id, ct_name, instructor_id, instructor_name, depart_name)

```
create view inform_1
( query 1 );
```

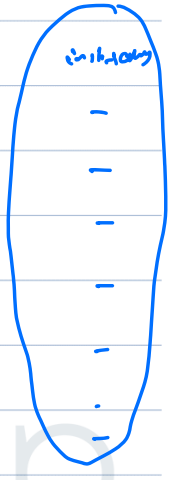
- views can simplify complex queries by encapsulating in single.
- each time a view is queried, the underlying query is executed to fetch the data
- it can be used for restrict sensitive data to other users;

break till 9:23

window function

instructor

inst_id	inst_name	inst_salary	dept_id



aggregate fn

```
select avg(salary)
from instructor;
```

```
select inst_n, avg(salary)
from instructor;
```

over()

partition by()

rank() over( )

dense_rank() over( )

row_number() over( )

```sql

grant select on instructors to 'lokesh';
grant update , delete on instructors to 'lokesh';
grant update on departments to 'lokesh';

revoke delete on instructors from 'lokesh';

-- display st_id , st_name , instructor_id , instructor_name
-- department_id  , department_name

select st_id , st_name , i.instructor_id , instructor_name,
d.department_id , department_name
from students s
join instructors i
on  i.instructor_id =  s.instructor_id
join departments d
on d.department_id = i.department_id;


 create or replace view st_inst_dept
 as (
    select st_id , st_name , i.instructor_id , instructor_name,
d.department_id , department_name
from students s
join instructors i
on  i.instructor_id =  s.instructor_id
join departments d
on d.department_id = i.department_id
 );

 select * from st_inst_dept;

 update st_inst_dept
 set st_name = 'bharat'
 where st_id = 4;

 start transaction;
  update st_inst_dept
 set st_name = 'bharat';

 rollback;
 -- get avg of salary from instructor;
 select avg(salary)
 from instructors;
 -- wrong way
 select instructor_name , avg(salary)
 from instructors;
 -- wrong answer
 select instructor_name , avg(salary)
 from instructors
 group by instructor_name;

 -- subquery
  select instructor_name ,(select avg(salary) from instructors)
 from instructors;
-- window function
select instructor_name , avg(salary)
over()
from instructors;

-- fetch all the inst_name along with avg salary of
-- all instructor in their respective department

```

```sql
-- correalted subquery
select instructor_name ,
(select avg(salary) from instructors where department_id = a.department_id)
from instructors a;
-- window function
select instructor_name ,
   avg(salary) over(partition by department_id)
from instructors;
-- sorting
select st_id , f_name , marks
from students
order by marks desc;

-- rank()
select st_id , f_name , marks,
rank() over(order by marks desc)
from students
order by marks desc;
-- default sparse ranking
-- 1,1,3,4,4,4,4,8 ....

select st_id , f_name , marks,
dense_rank() over(order by marks desc)
from students
order by marks desc;
-- dense ranking
-- 1,1, 2, 3, 3, 3, 3, 4.....

-- row number
select st_id , f_name , marks,
row_number() over(order by marks desc)
from students
order by marks desc;
```