⑥ SELECT {To get data}

⇒ Select <column names> ← Projection

from <table name>

where <condition> ← Selection

⇒ select * from Emp ;

↳ get all records from Emp table.

⇒ select ename, desig from emp

↳ list of all ename, design from emp.

⇒ Select distinct ename, desig from emp

↳ duplicate records will be removed from result.

⇒ Select * from emp where dept = 'Health';

↳ get all employee from emp where department is health.

19/2/24

To add multiple condition with where clause.

<, >, <=, >=, =, <>

↳ not equal to.

```
Select ename, salary, desig      ← Projection.
from emp
                                 ← Selection
where salary > 25000 AND location = 'Jaipur';
```

OR

NOT location = 'Jaipur';

is Null & is not null ⇒

⇒ select * from emp where location is null;

location is not null;

Like clause ⇒

⇒ String pattern match

⇒ we have 2 wildcard characters.

% _

% ⇒ 0, 1 or more than 1 character.

_ ⇒ exactly for 1 character.

Q ⇒ List all the employees whose location starts with 'J'

⇒ Select * from emp where location like 'J%';

Q => List all employees whose name starts with 'A' & is of only 4 characters.

=> select * from Emp where name like 'A___';

'A__M%';

'A___%';

Emp

eid	ename	dept.no.
1	A	101
2	B	102
3	C	101

Dept

d_id	d_name
101	CS
102	IT

Q we want

1	A	CS
2	B	IT
3	C	CS

If we do, Select * from Emp, Dept, we will get.

eid	ename	dept.no.	d_id	d_name
1	A	101	101	CS
1	A	101	102	IT
2	B	102	101	CS
2	B	102	102	IT
3	C	101	101	CS
3	C	101	102	IT

Our output will be Cartesian product of two tables.

If we have table R1 with 3-records & R2 with 2-records then Cartesian product will give us a table with 6 records

A In case of attributes we can have max of (R1 + R2)
(R1 → no. of attr, R2 → no. of attr in R2)

Select eid, ename, dname from emp, dept
where did = d.no;

1	A	CS
2	B	IT
3	C	CS

20/2/24

Select * from emp // we cannot predict the order.

To get data in some orders we have to use "Order by" clause.

=> Select * from emp order by salary;
{by default ascending}

=> Select * from emp order by salary desc;
↳ for descending order.

Emp

ename	Salary
aa	50000
ab	50000
bb	45000
bc	45000

Output

There is not order predictable for both

b b	45000
b c	45000
a a	50000
a b	50000

{bb, bc} ∄ (aa, ab)

⇒ Select * from emp order by Salary, ename desc;

{ order will be according to Salary descending and in case Salary is same we will have ename in descending orders }

↓

Output

a b	50000
a a	50000
b c	45000
b b	45000

In SQL we have 5 Aggregate Funtions

Select max (Salary) from Emp;

Output ⇒ 50000

Select min (salary) from emp;

Output = 45000

Select Sum (salary) from emp;

Output ⇒ 190000

Select Avg (salary) from emp;

Output ⇒ 47,500

Select Count (*) from emp;

Output ⇒ no. of records.

⇒ Select avg (salary) from emp where desig = 'IA';

Group By {Clause}

Q ⇒ I want to find total Salary given to all specific posts in the department.

Expected Output ⇒

IA	25000000
AP	100000000
Dso	—
ACP	—
SA	—

Select desig, Sum(salary) from Emp Group by desig;

In group by clause at projection we can only have attribute that we group by (eg desig) and any aggregate function.

group by with sub group ⇒

Q ⇒ we want to group emp with desig and we also want to group desig by location. and find total salary accordingly

⇒ ~~Select~~

⇒ Select desig, location, sum(salary) from Emp group by desig, location;

eg ⇒

desig	location	sum(salary)
IA	BKN	2500000
IA	JPR	10000000
IA	AJM	5000000
AP	JJN	—
AP	BKN	—
—	—	—

Let's say we want group desig by location and want only groups with sum of salary higher than 5 lakh

⇒ Select desig, Location, Sum(salary) from Emp group by desig, location having sum(salary) > 500000;

21/2/24

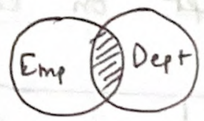
If we want to use where clause with group by it will used before it & in case of Order clause will be after group by & group by - having?.

Emp

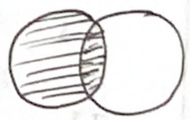
eid	ename	did
1	A	101
2	B	102
3	C	101
4	D	103
5	E	Null
6	F	Null

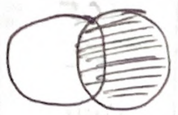
Dept

did	dname
101	CS
102	IT
103	EC
104	EE



← Inner Join


← Left Outer Join


← Right Outer Join


← Full Outer Join

① Inner Join →

Select eid, ename, dname from Emp

inner join Dept on Emp.did = dept.did;

Output-

eid	ename	dname
1	A	CS
2	B	IT
3	C	CS
94	D	EC

② Left Join →

Select eid, ename, dname from Emp

Left join Dept on Emp.did = Dept.did;

eid	ename	dname
1	A	CS
2	B	IT
3	C	CS
4	D	EC
5	E	NULL
6	F	NULL

③ Right Join

⇒ Select eid, ename, dname from Emp Right

join Dept on Emp.did = Dept.did;

eid	ename	dname
1	A	CS
2	B	IT
3	C	CS
4	D	EC
Null	Null	EE

④ Full Outer Join

⇒ Select eid, ename, dname from Emp Outer

join Dept on Emp.did = Dept.did;

eid	ename	dname
1	A	CS
2	B	IT
3	C	CS
4	D	EC
5	E	NULL
6	F	NULL
-	-	EE

In and NoT In

for, Set matching

⇒ Select * from Emp where desig = 'IA' and location in { 'Bharatpur', 'Karauli', 'Dausa' };

⇒ Select * from Emp where desig = 'IA' and location ~~is~~ not in { 'Bharatpur', 'Karauli', 'Dausa' };

Set Comparison ⇒

Q⇒ get all peons list whose salary is greater than Programmer.

⇒ Select * from Emp where desig = 'Peon' and salary > Some (Select salary from Emp where desig = 'Prog');

Older edition used to use 'Any' instead of 'Some' for same meaning/use

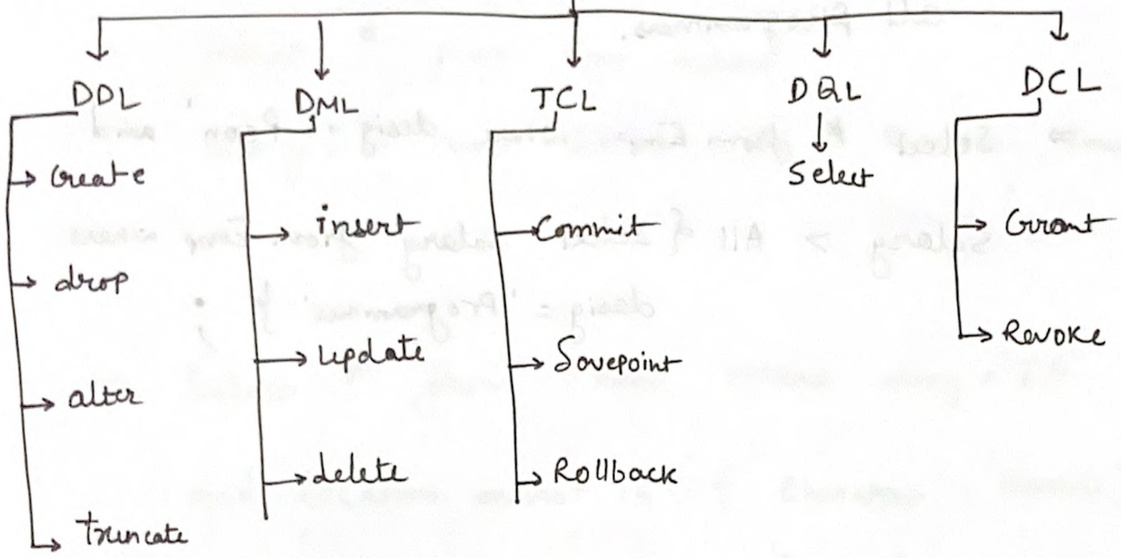
gSAL

Q. Get all peons list whose salary is greater than all programmers.

⇒ Select * from Emp where desig = 'Peon' and salary > All { select salary from Emp where desig = 'Programmer' } ;

23/2/24

SQL Commands



DDL ⇒ Data Definition Language

DML ⇒ Data Manipulation ~~Language~~ Language

TCL ⇒ Transaction Control Language

DCL ⇒ Data Control Language

DQL ⇒ Data Query Language

SDL ⇒ Storage definition language

① DDL ⇒ To define Schema (Table / Database's structure)

② DML ⇒ To Modify data in Table

③ TCL ⇒ Queries related to the transaction.

④ DQL ⇒ # To Access the data from database.

⑤ ~~DQL~~ SDL ⇒ for defining methods of storing data.

⑥ ~~DCL~~ DCL ⇒ # To give & take permission to users;

DCL

① Grant ⇒

```
# grant select, update on emp to Ramesh,
  Suresh;
```

② Revoke ⇒

```
# Revoke select, update on emp from Ramesh;
```

Between Clause ⇒

- We want to identify all employees whose salary is between 50000 & 70000

(1) Select * from Emp where

Salary >= 50000 And Salary <= 70000;

(2) Select * from Emp where

Salary between 50000 and 70000

{Both values will be inclusive}

We can also use (not between) as well

(3) Select * from Emp where

Salary Not between

50000 and 70000;

Views ⇒

Subset of database which can be used as table

↳ Table on fly {That does n't get stored}

↳ Virtual Table

~~⇒ To work with~~

→ for security purposes

→ To avoid accidental damages.

⇒ For updatable Views {if DBA allows updation from View}

→ View should not be created by group by or having clauses.

→ The view should not be created by using more than one tables.

→ For insertion of record through View, all the attributes other than attributes of View should not be "Not Null";

Command ⇒

⇒ Create view V

as

(select ename, salary, emobile

from emp);

⇒ select ename from V;