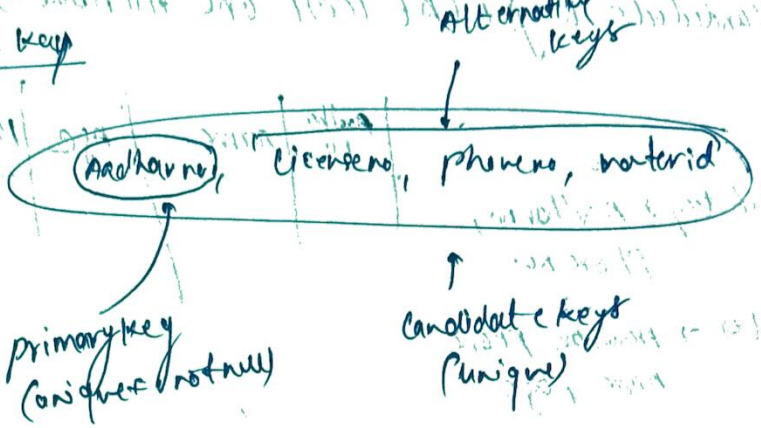
(DBMS) Database Managment system

candidate keys



primary key

→ unique & not null

Foreign keys

⇒ It is an attribute or set of attributes that refrences to primary key of same or other table.

Eg

Students

Rollno	name	address
1	A	delhi
2	B	jbp
3	C	indore

primary key

Refrenced table

Courses

Foreign key

courseid	course name	Rollno
C_1	DBMS	1
C_2	Networking	2
C_3	History	2
C_1	DBMS	3

Refrencing table

lect 13 # super key

↳ (candidate key) + (At least one Attribute)

Aadhar no.	name	age	phone no

candidate key ⇒ Aadhar no, phone no.

Attributes ⇒ Aadhar, phone, name, age

super key ⇒ combination of ① aadhar & name
② aadhar & age
③ aadhar & phone no
④ aadhar, name & age, etc

Ex $R(A_1, A_2, A_3 \ldots A_n)$ → Relation, Then how many superkeys possible

① If A_1 → candidate key (ck)

$A_1, \underbrace{A_2, A_3 \ldots A_n}_{n-1}$

$\Rightarrow 2^{n-1}$

② No ck

$\underbrace{A_1\ A_2\ A_3 \ldots A_n}_{n}$

$\Rightarrow 2^n$

③ A_1, A_2 → ck

A_1A_2, A_1A_3, A_1A_4, A_1A_5 ... $\Rightarrow 2^{n-1}$

A_2A_1, A_2A_3, A_2A_4, A_2A_5 ... $\Rightarrow 2^{n-1}$

$\Rightarrow 2^{n-2}$

$\Rightarrow (2^{n-1}) + (2^{n-1}) - (2^{n-2})$

④ $\underbrace{A_1, A_2}_{ck_1}$ $\underbrace{A_3, A_4}_{ck_2}$

A_1A_2, A_3
$A_1A_2A_4$
$A_1A_2A_5$
$A_1A_2A_3A_4$ $\Rightarrow 2^{n-4}$

$A_3A_4A_1$
$A_3A_4A_2$
$A_3A_4A_5$
$A_3A_4A_1A_2$ $\Rightarrow 2^{n-4}$

2^{n-2} 2^{n-2}

$\Rightarrow (2^{n-2}) + (2^{n-2}) - (2^{n-4})$

Lect 14

Entity - Relation (ER)

representation model of database

ER

Attributes

student (Rollno, name, age, address)

course (courseid, coursename, Rollno, duration)

Entity type schema

student — study — course

relationship

Entity →

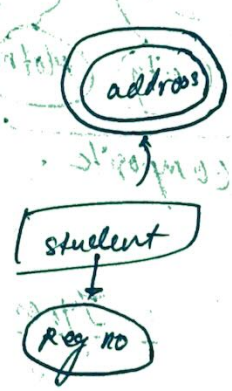
Relationship →

Attribute →

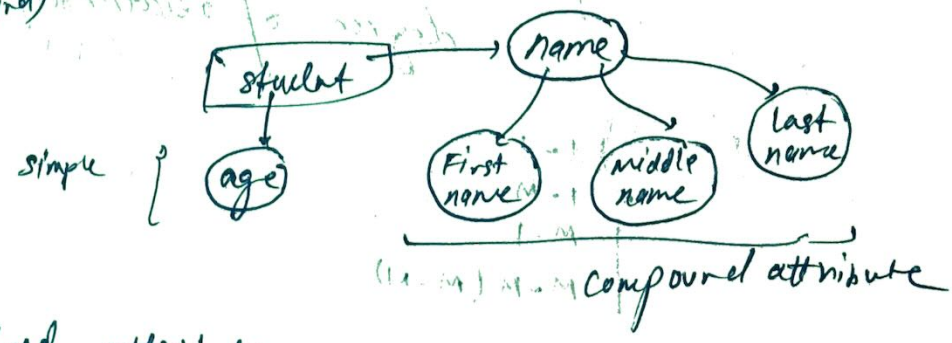
Lect 15

Types of Attributes

① single v/s multivalued attribute



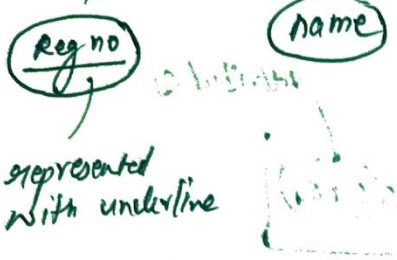
② simple v/s composite attribute (compound)



③ stored v/s Derived attribute

(dotted elipse) representation

④ key v/s non-key attribute



represented with underline

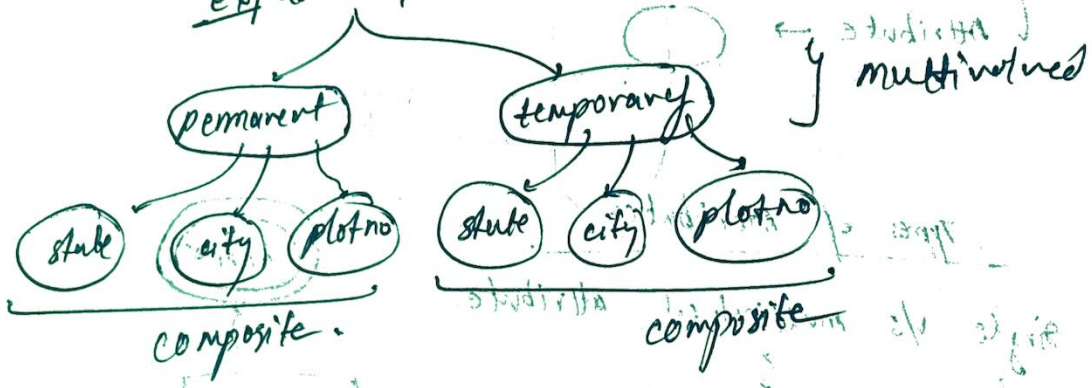
⑤ Required v/s optional attribute

notnull — could be null

⑥ complex

↳ mix of composite + multivalued.

Ex] address,



Lect 10

Types of relationship

degree of relationship (cardinality)

- 1-1
- 1-M
- M-1
- M-M (M-N)

① one to one

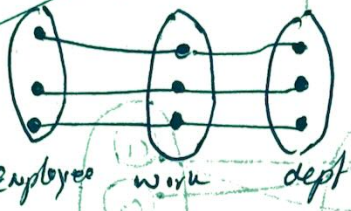
primary key

Eid	Ename	Eage
e_1	A	18
e_2	B	20
e_3	A	21

work

Eid	Did
e_1	D_1
e_2	D_2
e_3	D_3

Did	Dname	location
D_1	IT	Indore
D_2	prod	delhi
D_3	HR	Jabalpur



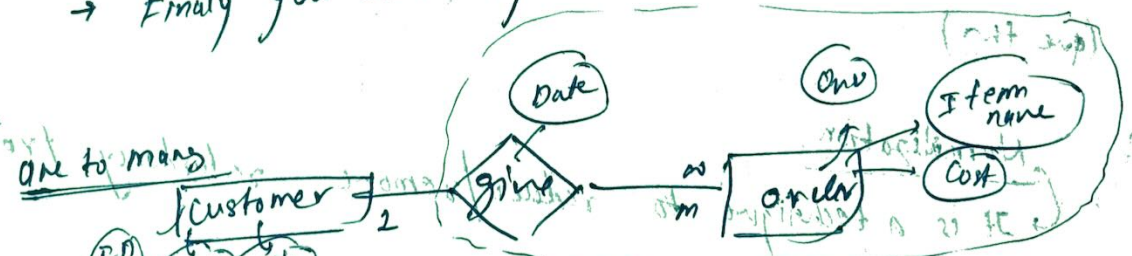
→ you can make any one of Eid/Did – primarykey

→ Then you can merge work with Employee (if Eid is pk) or work with dept (if Did is pk).

→ Finaly you will left with two tables.

Left 17

② one to many



ID	name	city
c_1	tanishq	jabalpur
c_2	Rashi	jabalpur
c_3	D	katni

fk fk

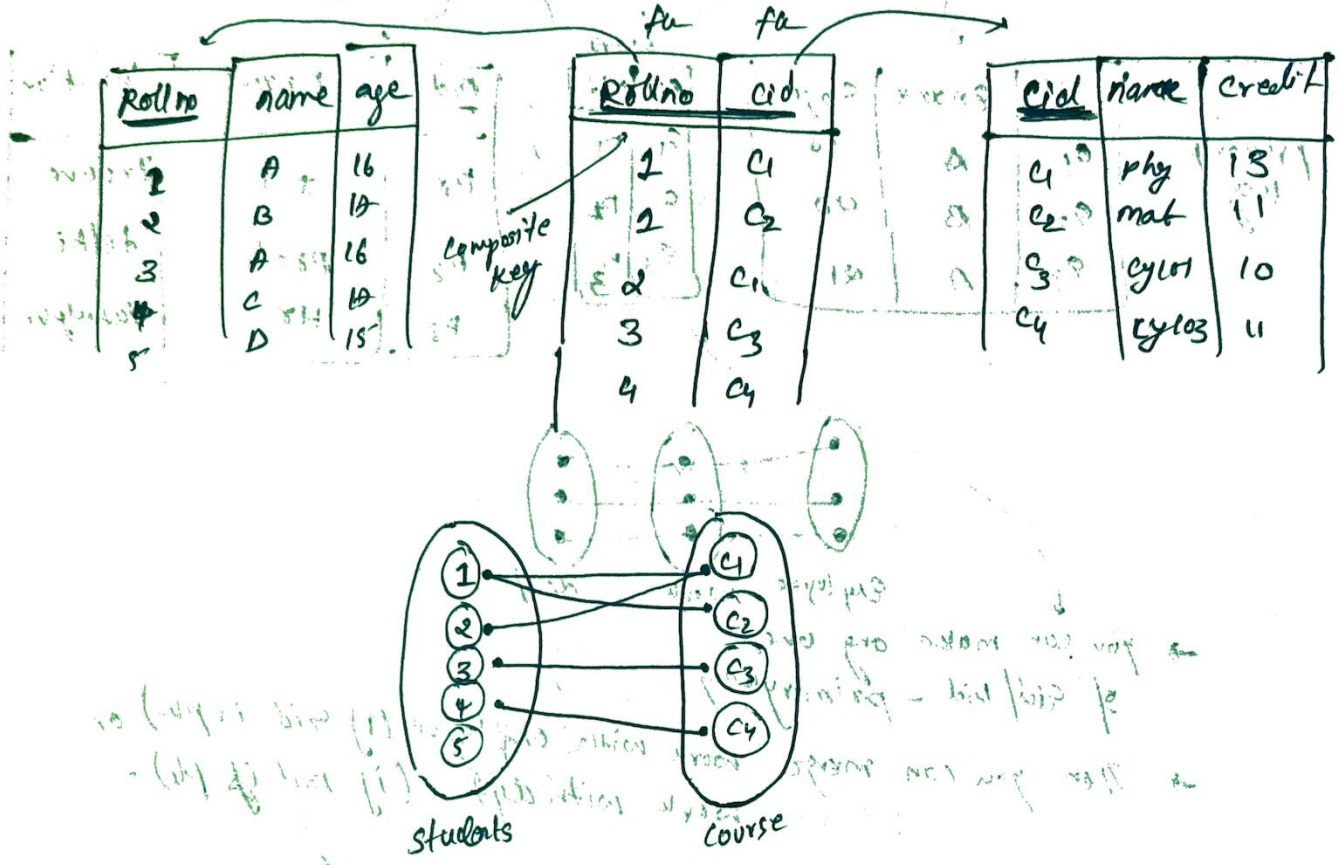
ID	o_no	Date
c_1	o_1	11/11
c_1	o_2	10/11
c_2	o_3	11/11

o_no	Itemname	cost
01	kaju	10
02	kishmish	20
03	Badam	30

merge

ono	Itemname	ID	Date	cost
01	kaju	c_1	11/11	10
02	kishmish	c_1	10/11	20
03	Badam	c_2	11/11	30

Lect 18 ③ many to many



Lect 19 (qve tha)

Lect 20 Normalization

↳ It is a technique to reduce/remove redundancy from a table.

Lect 21 First Normal form

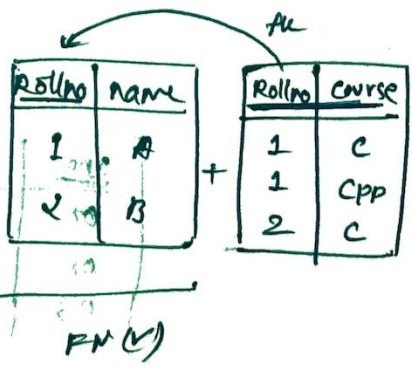
↳ It states that in a table there cannot exist att multivalued, attributes.

Eg

Roll no	name	course
1	A	C/CPP
2	B	C

(student)

(Not FN form)



lect 11

Closure method

method to find candidate key with some relation given about attributes with other attributes.

Ex: R(A B C D)

① FD { A → B, B → C, C → D, }

② FD { A → B, B → C, C → D, D → A }

A^+ = BCDA (4 attributes) ⇒ A = CK

B^+ = BCD, C^+ = CD, D^+ = D ⇒ NO CK

(AB)$^+$ = ABCD ⇒ CK (AB) (because it will then become a super key (CK is minimal))

A^+ = ABCD, B^+ = BCDA, C^+ = CDAB, D^+ = DABC ⇒ CK

CK = {A}

prime Attri = {A}

Non-prime Attri = {B, C, D}

CK = {A, B, C, D}

prime Attri = {A, B, C, D}

non-prime attri = { }

③ R(A B C D E)

FD { A → B, BC → D, E → C, D → A }

method to find ④: right Attri (B, C, D, A)

(E)$^+$ = BCDA (E)

method to find ⑤ other keys

(AE)$^+$ is a key ⇒ replace A/E with FD have on R.H.side.

eg ① D → A

AE$^+$ = (DE)$^+$

② BC → D

AE$^+$ = DE$^+$ → BCE$^+$

D → A

E → C

(BE)$^+$

✓ (AE)$^+$ = ECBDA

✓ (BE)$^+$ = BECDA

✓ (DE)$^+$

CK = {AE, BE, DE}

prime Attri = {A, B, D, E}

non-prime Attri = {C}

lect 23

Functional Dependency

① Reflexivity : If y is a sub-set of x
then $x \rightarrow y$.

② Argumentation : If $x \rightarrow y$
then $xz \rightarrow yz$

③ Transitive : If $x \rightarrow y$, and $y \rightarrow z$
then $x \rightarrow z$

④ Union : If $x \rightarrow y$, and $x \rightarrow z$
$x \rightarrow yz$

⑤ Decomposition : If $x \rightarrow yz$,
then $x \rightarrow y$, and
$x \rightarrow z$

⑥ ~~pseudo decomposition~~

⑥ pseudotransitivity : If $x \rightarrow y$ and $wy \rightarrow z$
then $wx \rightarrow z$

⑦ Composition : If $x \rightarrow y$, and $z \rightarrow w$
then $xz \rightarrow yw$

Ex

sid	name
1	A
1	B

not valid

sid	name
1	A
2	A

valid

lect 24

Second Normal Form

1. 1st normal form should be valid
2. All the non-prime attributes should be fully functional dependent on candidate key.
   (There should be no partial dependency in the table)

Ex

(X) (Y) (Z)

not valid 2nd NF

Customer ID	Store ID	Location
1	1	Delhi
1	3	Bombay
2	1	Delhi
3	2	Bangalore
4	3	Bombay

Delhi → 1
Bombay → 3
Bangalore → 2

Candidate key ⇒ {Customer ID, Store ID} xy

Prime Attri ⇒ {Customer ID, Store ID} ⇒ {x, y}

Non-prime Attri ⇒ {Location} = {z}

2nd rule ⇒ xy → z → But here it's not a valid 2nd NF

Hence

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

2nd NF (Y)

Store ID	Loc
1	Delhi
2	Bangalore
3	Bombay

2NF (Y)

Ex $R(ABCDEF)$

FD $\{C \to F, E \to A, EC \to D, A \to B\}$

method (a)

$(CE)^+ = A\,B\,C\,D\,E\,F$; $CK = \{CE\}$

prime Attri $= \{C, E\}$

non-prime Attri $= \{A, B, D, F\}$

$CE \to D$, But $C \to F$, $E \to A$

hence partial dep.

not in 2nd NF

(b) $CE^+ = CEADFB$

No method (b)

hence only one ck

Note] for 2nd Normal form

(a) All non-prime attri should be fully dependent on ck (not partially)

same

(b) There should be no partial dependency in the table.

~~(c) Non-NPA should determine NPA.~~

Lecture] # 3rd Normal form

⇒ (1) Table should be in 2nd NF.

⇒ (2) There should be no transitive dependency.

[non-prime attri → non-prime attri should be not there]

FD $\{A \to B, B \to C\}$

PA $= \{A\}$

NPA $= \{B, C\}$

$A \to B$, $B \to C$ → 3rd NF (✓)

Ex R(ABCD)

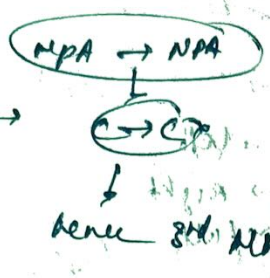
FD {AB → CD , D → A}

Soln

CK = {AB , DB}

PA = {AB, D}

NPA = {C}



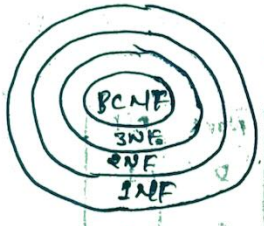
Hence 3rd NF (✓)

Lecture 28 | Boyce Codd Normal Form (BCNF)

special case of 3rd Normal form.

→ (1) 3rd NF

→ (2) LHS of each FD should be ck or super key.



Lect 29 | (3rd NF) ≠ (BCNF) → lossless decomposition

3rd NF → dependency preserving decomposition

Lect 38]

Introduction to Joins and it's types

* **Joins**
  - → Cross (product) join → all possible matches with two table (product multiply)
  - → Natural Join
  - → Conch Join
  - → Equi Join
  - → self Join
  - → outer join → left, right, full

Lect 39]

Natural Join

Join = Cross product + Condition

Emp

EID	Name
E1	A
E2	B
E3	C

Dept

DID	Name	EID
D1	D	E1
D2	E	E2

⇒ select name from (Emp, dept) where Emp.EID = Dept.DID;

this will do cross prod

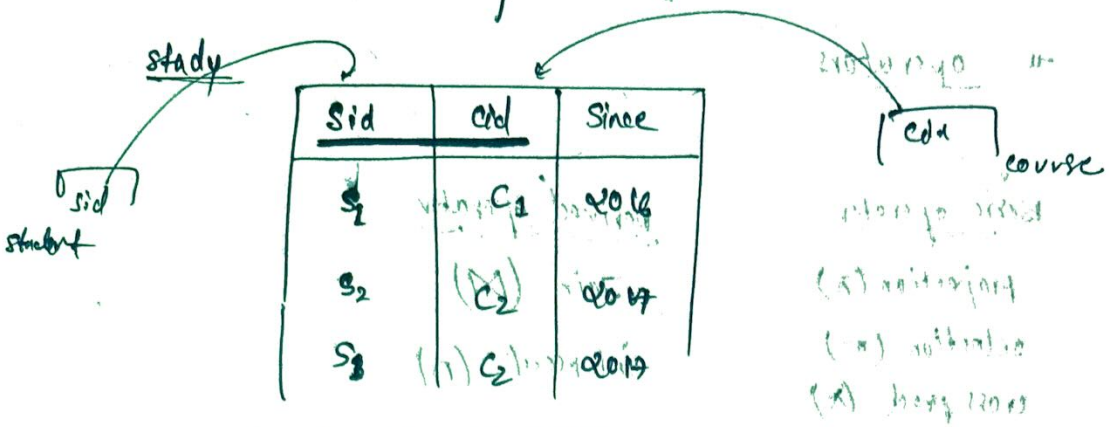
EID	Name	DID	Name	EID
E1	A	D1	D	E1
E2	B	D2	E	E2
E3	C	D1	D	E1
E1	A	D2	E	E2
E2	B	D2	E	E2
E3	C	D2	E	E2

out names → A, B

⇒ **select name from Emp natural join dept;**

Note* (common attribute should be same)

lect 40] ——— self Join ———

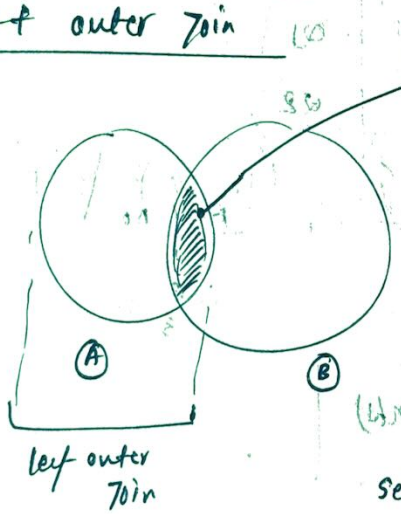


select ___ from study as T1, study as T2 where T1.Sid = T2.Sid ;

lect 41] ——— Equi Join ———

↳ same he hai yrr.

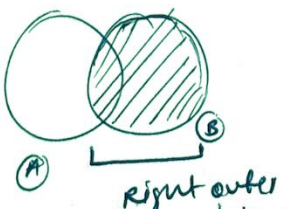
lect 42] left outer Join



Select ___ from A left outer join B

on (A.Id = B.Id) ;

lect 43] Right outer Join



lecture] Relational Algebra

operators

Basic operator
- projection (π)
- selection (σ)
- cross prod ($\times$)
- union ($\cup$)
- rename (ρ)
- set diff ($-$)

Derived operator
- Join ($\bowtie$)
- intersect ($\cap$)

$x \cap y = x - (x - y)$

- Division ($/$, $\div$)

lecture

projection (π) ⟶ to fetch the data. (only unique answers).

(students)

RollNo	Name	Age
1	A	20
2	B	21
3	A	22

⇒ $\pi_{RollNo}(students)$ ⟶

RollNo
1
2
3

⇒ $\pi_{RollNo, Name}(students)$

⇒ $\pi_{Name}(students)$ ⟶

Name
A
B

lect 46

selection (σ)

(Students)

Roll no	name
1	A
2	B
3	A

$\Rightarrow \Pi_{name}\left(\sigma_{rollno=2}(student)\right)$

lect 47

~~cross projection~~

cross product ($\times$)

R_1

A	B	C
1	2	3
2	1	4

R_2

C	D	E
3	4	5
2	1	2

$R_1 \times R_2$

A	B	C	C	D	E
1	2	3	3	4	5
1	2	3	2	1	2
2	1	4	3	4	5
2	1	4	2	1	2

lect 48] difference (−)

(A−B)

(student)

Roll no	name
1	A
2	B
3	C

(Emp)

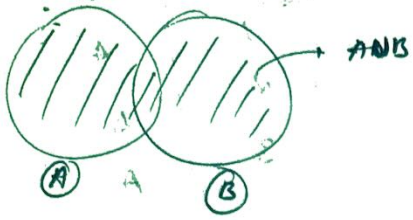
No	name
7	E
1	A

(student) − (Emp)

Roll no	Name
2	B
3	C

$\Pi_{name}(student - Emp)$

lect 49] union operator



⇒ student ∪ Employe

⇒ π_{name}(student) ∪ π_{name}(Employee)

lect 50] division Method

(enrolled)

sid	cid
s_1	c_1
s_1	c_2
s_2	c_1
s_3	c_2

(course)

cid
c_1
c_2

⇒ Enrolled(sid, cid) | Course(cid)

s_1

{ To derive name of student who is enrolled in every course.

A(x,y) / B(y) ⇒ It results x values of tuple such that there should be for every y value of relation B.

How this works

step① $(\pi_{sid}(Enrolled) \times \pi_{cid}(course))$

s_1 × c_1
s_2 c_2
s_3

step②

s_1	c_1
s_1	c_2
s_2	c_1
s_2	c_2
s_3	c_1
s_3	c_2

−

enrolled	
s_1	c_1
s_1	c_2
s_2	c_1
s_3	c_2

=

s_2	c_2
s_3	c_1

↑ Students who are not enrolled in atleast one course.

step③ $\pi_{sid}(\quad) - \pi_{sid}$(enrolled)

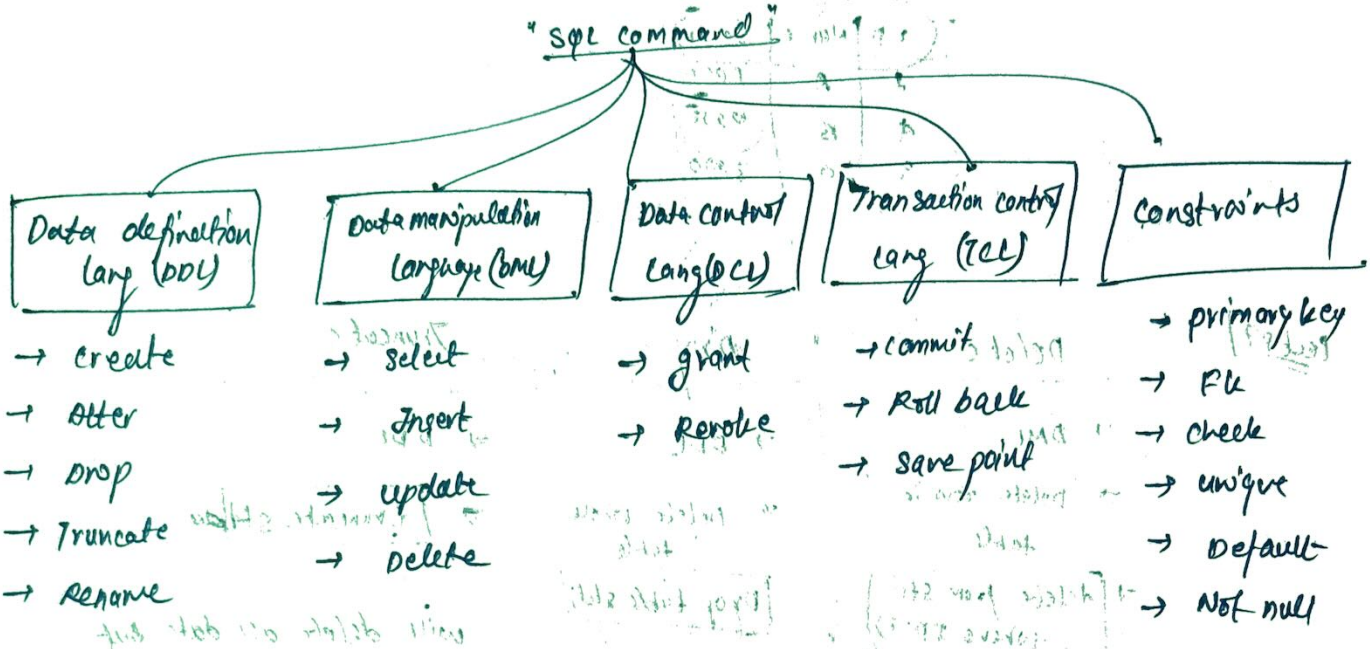
⇒ s_1

↑ students who are enrolled in every courses.

$\Rightarrow \pi_{sid}(enrolled) - \left(\pi_{sid}\left(\left(\pi_{sid}(enrolled) \times \pi_{cid}(course) \right) - (enrolled) \right) \right)$

lect 52/53

Structure query Language (SQL)



lect 54

(student)

id	name	Salary

```
Create table student
(
    id int,
    name varchar2(20),
    Salary number(8)
);

desc student;
```

lect 55

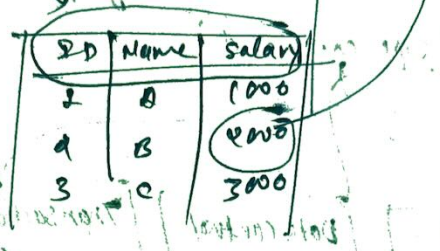
Alter

→ add, Remove, modify, rename
add constraint, remove constraint

```
Alter table student add addren varchar(30)
                  rename column id to sid;
                  rename to std;
```

lect 56]

Alter	update
⇒ DDL	⇒ DML
⇒ For structure change	⇒ For Data change.



ID	Name	salary
1	A	1000
2	B	2000
3	C	3000

lect 57]

Delete	Drop	Truncate
⇒ DML	⇒ DDL	⇒ DDL
⇒ Delete row in table	⇒ Delete whole table	⇒ truncate std
→ delete from std where ID=2;	Drop table std;	will delete all data but keep the table.

(std)

ID	name
1	A
2	B
3	C

delete from std ≠ truncate std

no diff same

But here we can use it with "where".

lect 58] constraints in sql

- ⇒ unique
- ⇒ not null
- ⇒ check (age > 18)
- ⇒ PK, FK
- ⇒ Default

lect 59/60/61/62/63/66

⇒

(emp)

eid	name	dept	salary
1	A	d_1	1000
2	B	d_2	2000
3	C	d_1	3000
4	D	d_2	4000
5	E	d_3	5000

⇒ select dept, count(*) from emp group by dept;

dept	count(*)
d_1	2
d_2	2
d_3	1

⇒ select dept from emp group by dept having count(*) < 2;

dept
d_3

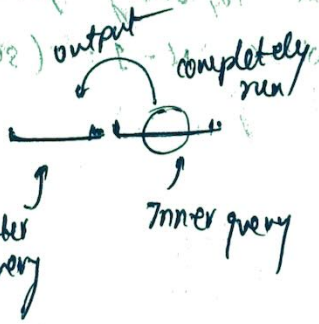
Note
- "having" → condn inside a group
- "where" → condn in whole table

lect 64

In / Not In

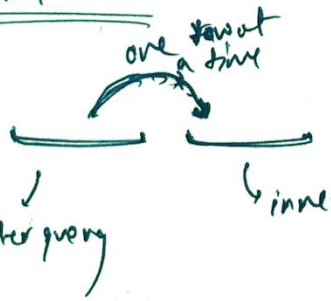
'A' in ('A', B, C) → ✓

'A' not in ('A', B, C) → ✗



lect 66

Exists / not Exists




```
select * from emp where eid exists (
        select eid from project where emp.eid = project.eid);
```

text 07] aggregate function

avg, min, max, sum, count

text 08] correlated subquery

output

inquery

outquery

text 09] find Nth highest salary

emp(e1)

ID	salary
1	1000
2	4000
3	3000
4	3000
5	4000
6	5000

emp(e2)

ID	salary
1	1000
2	4000
3	3000
4	3000
5	4000
6	5000

⇒ select id, salary from emp e1

where N − 1 = (select count(distinct(salary)) from emp e2

where e2.salary > e1.salary);

text 10] — PLSQL —

yt channel
"Data with baraa"
lect3

Window functions

(A) Aggregate fⁿ

- → count
- → sum
- → Avg
- → min
- → max

(B) Rank fⁿ

- → ROW_NUMBER()
- → RANK()
- → dense_rank()
- → cume_dist()
- → percent_rank()
- → NTILE(n)

(C) Value (Analytic fⁿ)

- → lead( )
- → lag( )
- → first_value
- → l

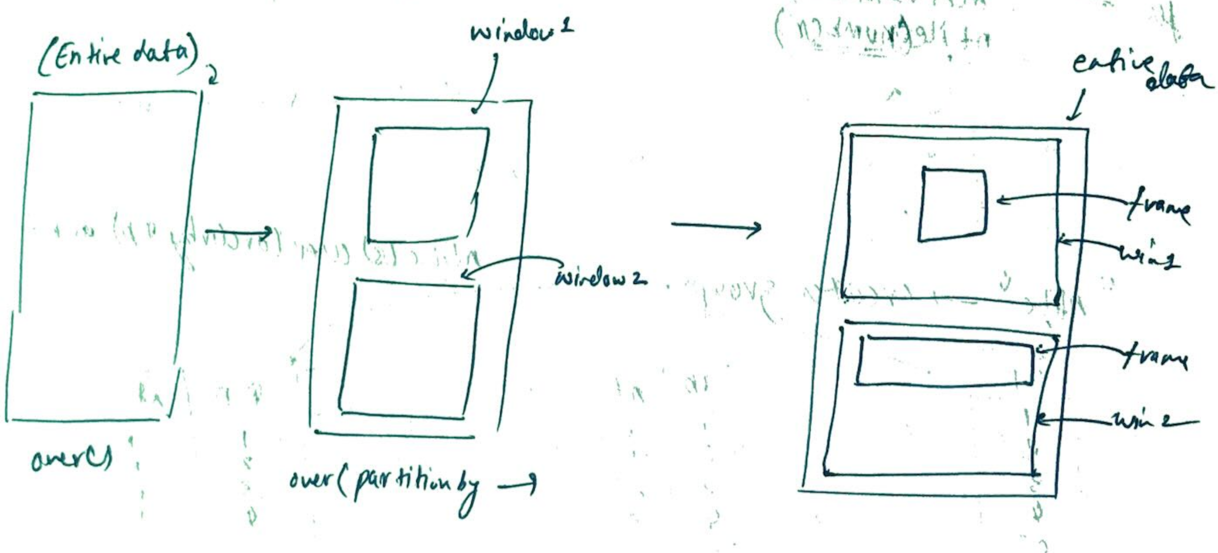
window syntax

AVG() over( )

- AVG() → window fⁿ
- over( ) → define the window

over(
- partition by ___ → define window
- order by ___ → sort within the window
- ROWS unbounded preceding → frame clause (defines subset of rows within each window).

)



Frame clause (order by is must to use)

ROWS Between current row and unbounded preceding

- ROWS → (Frame type)
  - → ROWS
  - → Range
- current row → (lower Frame Boundary)
  - → current row
  - → N preceding
  - → unbounded preceding
- unbounded preceding → (higher frame boundary)
  - →
  - →
  - →

[preceding (pahale)
→ current row
following (Badme)]

lect 4]

(A) Aggregate window functions

count (*) → count each row

count (col) → count only for which the value inside the f^n is (not null) (A)

Note] If we use order by in over clause,
it will by default come with frame clause
"row between unbounded preceding and current row".

lect 5]

(B) Ranking f^n (Frame clause not allowed, order by is must)

[Integer based ranking] → 1, 2, 3, 4, 5 ---

[percentage based ranking] → 0, 0.25, 0.5, 0.75, 1.0

f^n →
- row_number()
- rank() → creates gaps
- dense_rank()
- ntile(number)

f^n →
- cume_dist()
- percent_rank()

"ntile" → creates groups.

ID
1
2
3
4
5
6
7
8
9
10

ntile(2) over(order by ID) AS nt

ID	nt
1	1
2	1
3	2
4	2
5	3
6	3
7	4
8	4
9	5
10	5

groups of two.

ntile(3) over (order by ID) as nt

ID	nt
1	1
2	1
3	1
4	1
5	2
6	2
7	2
8	3
9	3
10	3

→ group of 3

But as 10/3 = 3.33
↓
(4), 3, 3 → com (a

higher group comes first

Lect 4]

(A) Aggregate window functions

count(*) → count each row

count(col) → count only # which the value inside the fn is (not null) (A)

Note] If we use order by in over clause,
it will by default come with frame clause
" row between unbounded preceding and current row ".

Lect 5] # (B) Ranking fn (Frame clause not allowed, order by is must)

Integer based ranking → 1, 2, 3, 4, 5 ...

percentage based ranking → 0, 0.25, 0.5, 0.75, 1.00

fn → row_number()
rank() → creates gaps
dense_rank()
ntile(number)

fn → cume_dist()
percent_rank()

" ntile " → creates groups.

ntile(3) over (order by ID) as nt

ID	nt
1	1
2	1
3	2
4	2
5	3
6	3
7	4
8	4
9	5
10	5

group 5 fm.

ntile(2) over (order by ID) AS nt

ntile(3) over (order by ID) as nt

ID	nt
1	1
2	1
3	1
4	1
5	2
6	2
7	2
8	3
9	3
10	3

group 3 fm

But as $\frac{10}{3} = 3.33$

(4) cases ↓ convert

Higher group comes first

→ cume_dist() over(order by sales desc) as dist

Sales	dist
100	1/5
80	3/5 (last position of same value)
80	
50	4/5
30	5/5

cume_dist = Position Nr / Number of Rows

→ percent_rank() over(order by sales desc) as dist

Sales	dist
100	0/4 = 0
80	2/4 = 1/4 (First position of same value)
80	
50	3/4
30	4/4 = 1

percent = Position Nr - 1 / No. of rows - 1

Lect-6] # ⑥ Value functions →

(lag/lead/First_value/Last_value)() over()

orderby — [must]

month	sales
1	10
2	20
3	30
4	40
5	50
6	60
7	70
8	80

First_value, lag, lead, last_value, default → 1, null

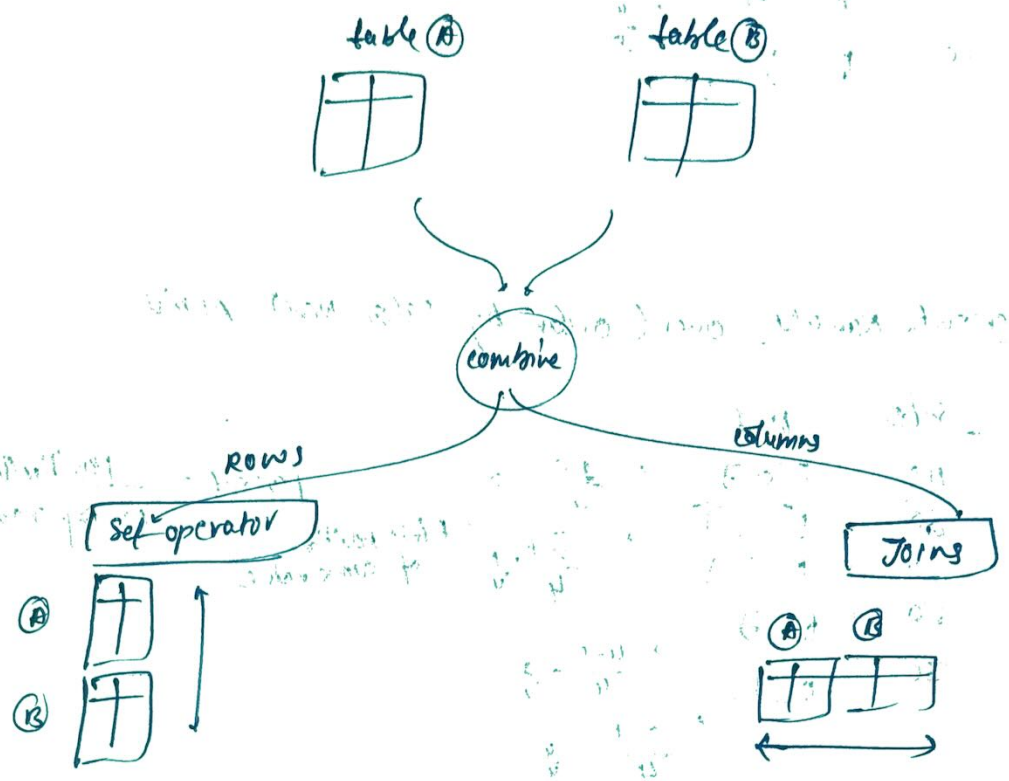
lead/lag (Attribute, offset, default) → Frame ⑧

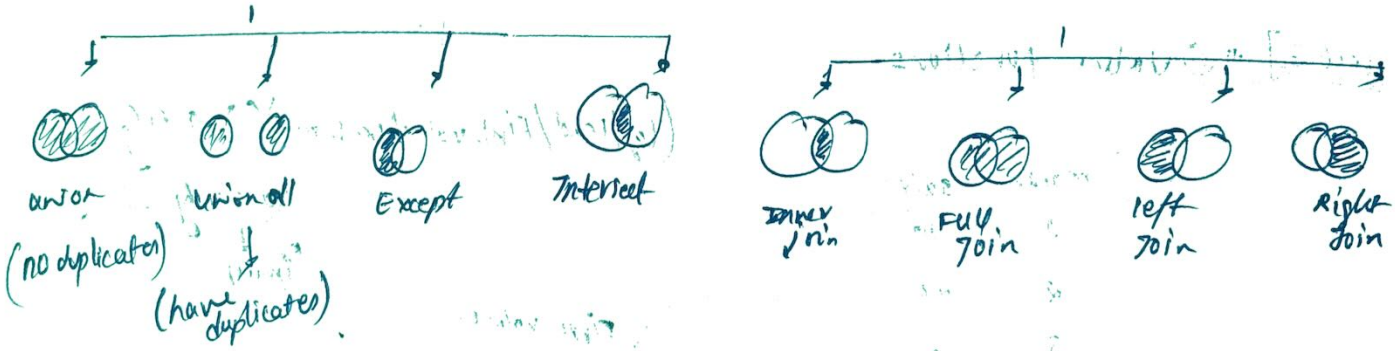
First_value/Last_value (Attribute) → Frame ①

But for last_value ↳ over(order by — Rows Between current row And unbounded following)

lect 7

Set operators





Note ① No. of columns must be same,
② Datatype must be same.

How to use ⇝ select * from A
union
select * from B;

Lect 8] —— Case statement ——

start of logic → Case

When cond1 Then res1

When cond2 Then res2

!

ELSE result

↳ optional

End

end of logic

→ like switch

Case Attri

When 'data1' Then res1

End

Lect 9] —— Null Functions ——

Check for Nulls

Null

Is Null → True

Is Not Null → False

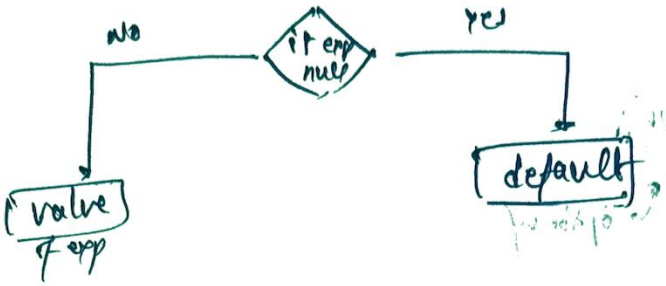
Replace values

Null —— IsNull(exp, default) / Coalesce ——→ 40

40 —— NullIf ——→ Null

ISNULL

ISNULL(exp , default)



coalesce (value1, value2, value3 ...)

Similar to ISNULL.

NULLIF

↳ NULLIF(Attribute, value)

NULLIF(sale, -1)

ID	Sale
1	1
2	2
3	→
4	3

ID	Sale
1	1
2	2
3	Null
4	3

[cont 11]

Date time

timestamp → oracle, postgres, mysql

datetime2 → sql server

get day, month, and year

Day(date) → 1 - 31

Month(date) → 1 - 12

Year(date) → xxxx

Datepart(part, date)

datepart(month, date) → 08

'2021-08-20' — datepart(year, date) → 2021

part = month, year, day, hour, minute, second, quater, week

Datename(part, date)

part = month, weekday, day

Jan / Feb / ...

Monday, tuesday ...

day → string 1/2/3 - - 31

EOmonth(date)

end of month

EOmonth(2021-07-20) → 2021-07-31

Sep. month last date

Datetrunc(part, date)

(month, year, day, minute, hour, seconds)

What ever part you'll give this fn will reset after that part

eg: Datetrunc(minute, '2021-08-21 21:02:58') → 21:02:00 (second)

Datetrunc(hour, '...') → 21:00:00 (hour, seconds)

Lect 12
Lect 13

—— Date Format Representation ——

2022-08-20 8:55:45

YYYY-MM-dd HH:mm:ss

format specifier

date

Format → MM/dd/YYYY → 08/20/22

Format → MMM YYYY → Aug 2022

number

Format → N → 123,123.45

Format → C → $ 123,123.45

Format → P → 123,123.45%

123123.45

Casting

string '123' ⟺ 123 number

Date 2022-08-20 ⟺ '2022-08-20' string

CAST(value AS data_type)

year/month/day

Date Add(part , interval, date)

Date Diff (part, date1, date2) ⟶ date1 − date2

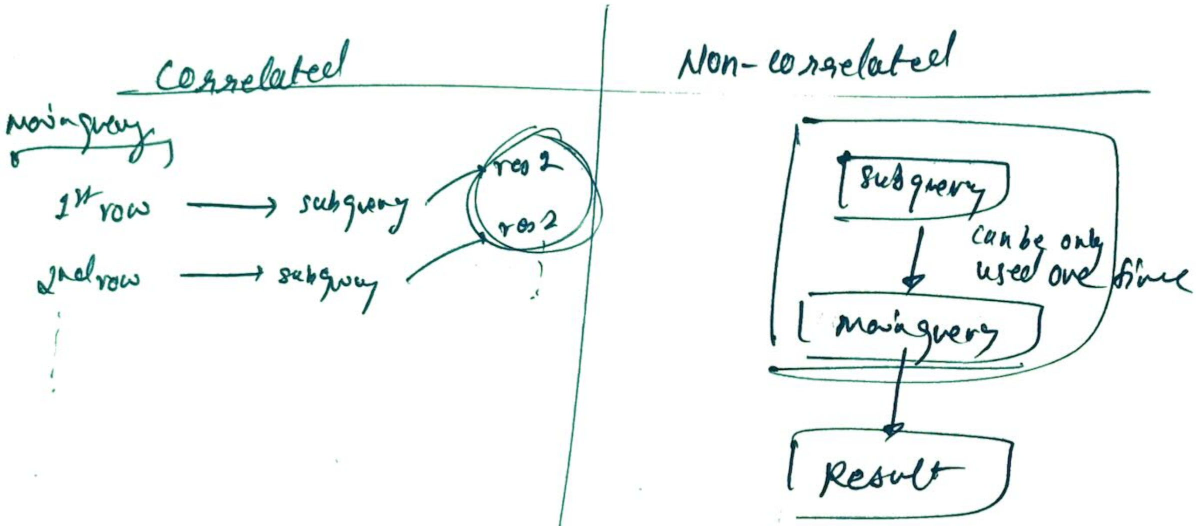
Is_date(date) → True / False (1 / 0)

Format (value, format, [, culture]) (optional)

eg ① Format (date, 'dd/MM/yyyy')
② Format (~~date~~ 1234.56, 'D', 'en-IN')

ddd → Mon/Tue/Wed--
MMM → Jan/Feb
MMMM → January/Febauray/March —

lect 15] — sub query —



Exists

If return anything then that row will be included in result, else will be removed

lect 16] Common Table Expression (CTE)