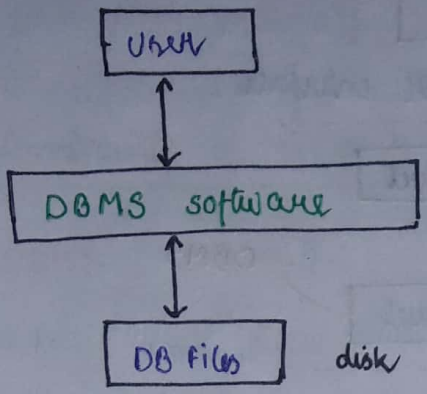
3/11/17

Brief Introduction —

Database — collection of related data.
eg- set of students info.

DBMS — s/w used to manage and access database files in an efficient way.



Interface b/w user and DB files.

Flat File System — (OS file system) DB files stored in disk are managed without DBMS (i.e using OS only)

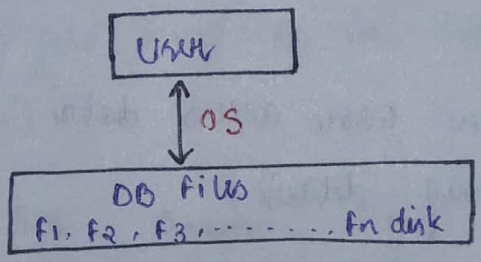


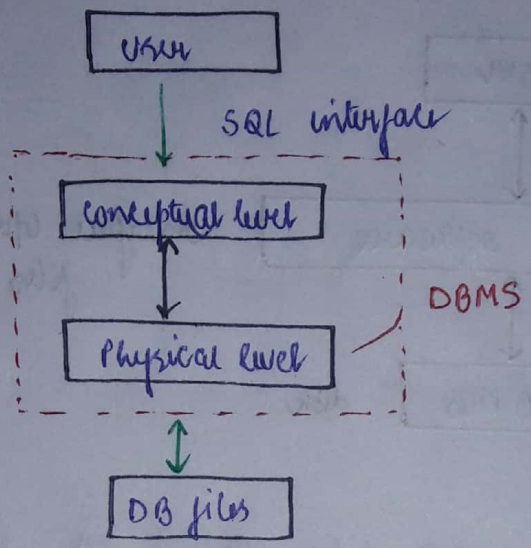
Fig: Flat file system

- It manages small DB : ~~fails to~~ manages ~~large~~ small DB : If DB size is in GB's / TB's then, flat file system fails to manage DB
- It manages small DB : If DB size is in KB's or MB's, then it can manage using flat files

Disadvantage —

→ Too complex to manage storage details of DB files by the user and too difficult to develop application programs and manage application programs.

Advantage of DBMS File system —



- No need to know storage details by user as all details are known to DBMS.
- Data Independency : User can access data without storage details
- Easy to access data.

#	Flat File System	DBMS File System
1)	Complex to manage storage details of DB files by user.	No need to manage storage details of DB files by the user.

2) more access cost, to access data from DB files.

3) Degree of concurrency is very less.

4) Too complex to implement different level of access control (security)

DBMS maintains indexing to reduce access cost.

More degree of concurrency

Easy to implement levels of access control because of virtual tables (views)

Integrity Constraints of RDBMS —

Relational DBMS [RDBMS]:

① Widely used data model

② R.J Codd (also called as Codd Data Model)

③ Codd proposed 12 rules to design RDBMS s/w.

Rule 1 —

- Data in DB file must be in tabular format (set of rows and columns).

student →

Sid	Sname	Age
S_1	A	20
S_2	B	20
S_3	C	21
S_4	A	22

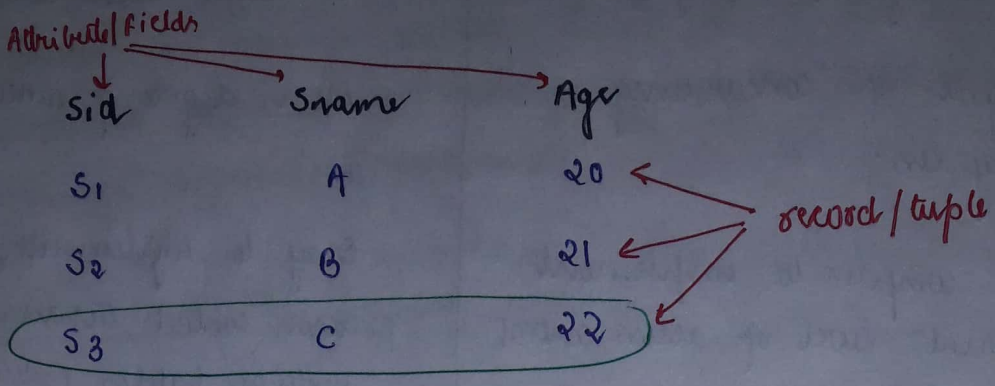
DBMS file

student

S1, A, 20 # S2, B, 20 # S3, C, 21 # S4, A, 22

Flat file (OS file)

- No 2 records of DB file is same



column name → Attribute / field

row → record / tuple.

Cardinality: No. of records of DB file.

Arity: No. of fields of DB file (table).

Relational Schema —

Definition of DB Table (or) Structure of DB table.

Student (Sid , Sname , age)

↑ table name

field name.

Relational instance:

- Record set of the DB file (also called as snapshot)
- It changes more frequently as compared to relational schema

Candidate Key —

Minimal set of attributes used to differentiate records of table uniquely.

① student (Sid , Sname , age)

↑		
S_1	A	20
S_2	B	21
S_3	C	22
S_4	D	20

{ Sid : candidate key }

{ [Sid , Sname] : Not candidate key }

↑ Not minimal

② Enroll (Sid , Cid , fee)

student can enroll many courses and course can be enrolled by many students . i.e

↓

Same sid with many cid and vice versa

candidate key → [Sid , Cid] as

Sid	Cid	fee
S_1	C_1	5000
S_1	C_2	6000
S_2	C_2	8000
S_3	C_2	2000

③ R

A	B	C
4	6	8
4	5	8
4	6	4
4	5	4
5	6	8

ABC : candidate key

candidate key → A ✗ AB ✗ ABC ✓

B ✗ BC ✗

C ✗ AC ✗

→ X is some set of attributes of R and
X is candidate key of relation R iff -

① No two records with same field values over "X" attributes (unique)

and ② No proper subset of X can differentiate the records uniquely (minimal)

if ABC is candidate key then

~~AB~~

R (X ----)

X ↑ candidate key

④ Emp -

Eid	Ename	DOB	panid	ppno	Bank	Acc No.
					SBI	1001
					BOI	1001
					SBI	1002

Assume No 2 emp's with same Bank and Acc No.

{ Eid , panid , ppno , Bank Acc No. } ← set of candidate keys of Emp relation

Eid ↑ Primary key

panid , ppno , Bank Acc No. ↓ Alternative keys.

- One of candidate keys is assigned as Primary key and remaining all candidate keys are called as Alternative keys

- NULL — Unknown value / Un existed value
  If any value is not assigned or given by user it is assigned as NULL.

Primary key —

- Any one candidate key of relation whose field values must be not NULL
- Every field of primary key must be not NULL (NULL value not allowed for any primary key field)
- For any DB relation atmost one primary key is allowed.

Alternative keys —

Any one candidate key of relation whose field values must be not NULL.

Every field of primary key must be not NULL (NULL value not allowed for any primary key field)

For any DB relation atmost one primary key is allowed.

- All candidate keys of relation except primary key.
- Alternative key fields allows NULL values.
- More than one alternative key is allowed to define.

not

Table Creation –

```
create Table Emp
(   Eid     varchar(10) Primary key,
    Ename   varchar(30),
    DOB     date,
    panid   varchar(10) UNIQUE NOT NULL,
    ppno    varchar(15) UNIQUE
    bank    varchar(20),
    Acc No  integer(15),
      UNIQUE ( bank, Acc no)
);
```

Keywords used

Primary key → Primary key

Alternative key → UNIQUE

Candidate keys

- Primary key (unique and Not NULL)
- Alternative keys (Unique)

Prime Attribute –

Attribute which belong to any candidate key of relation.

Prime attribute set : set of all candidate key's attribute of relation / table

Eg – In Table Emp.

Non prime attribute set = { Eid, panid, ppno, bank, accno }

Non prime attribute — (Non key attribute)

Attribute which does not belongs to any candidate key of the relation.

Non-prime attribute set — set of all non prime attribute of relation R.

Eg - in table Emp.

Non prime attribute set : { Ename, DOB }

→ For any RDBMS table, there must be atleast one candidate key whose field value must be NOT NULL.

For eg →

```
create table R
(   A int UNIQUE,
    B int UNIQUE,
    C int
);
```

✗ Not Allowed

↓

Two candidate keys but none is both unique and Not Null.

(Atleast one key should be unique and not null.

R (A , B , C)

2	NULL	-
4	3	-
6	4	-
NULL	5	-

} Relation R not in RDBMS.

Eg - Create table R
(
A int UNIQUE NOT NULL,
B int UNIQUE,
C int
);

✓ Allowed (for RDBMS)

Primary key → NO

candidate keys → 2 (A, B)

Eg - Create table R
(
A int primary key,
B int UNIQUE,
C
);

✓ Allowed for RDBMS (preferred)

primary keys → 1 (A)

Candidate keys → 2 (A, B)

Alternate keys → 1 (B)

Note - UNIQUE NOT NULL ≠ PRIMARY KEY

It means alternative key which does not allow NULL value

Super Key -

- Set of attributes of Relation (R) which can differentiate records uniquely but may not be the minimal attribute set.

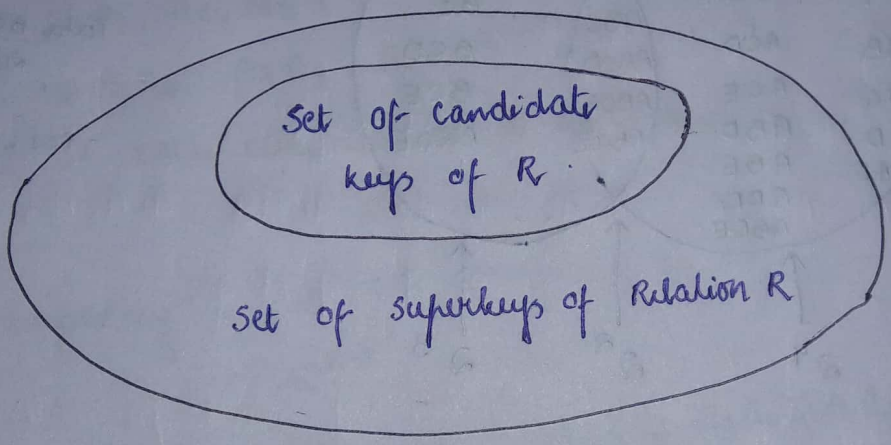
- Candidate key is minimal super key.
- Super key is not used in table implementation (theoritical approach), used only in design stage

Eg — stud (Sid , Sname , age)

if candidate key = Sid

↓

super key = { Sid,
Sid Sname,
Sid age,
Sid Sname age
}



↓

Every superkey is not candidate key while every candidate key is superkey.

Ques: R(A, B, C, D)

How many super-keys in relation R if A is candidate key?

$^3C_0 + {}^3C_1 + {}^3C_2 + {}^3C_3 = \frac{\lfloor 3}{\lfloor 0 \cdot \lfloor 3} + \frac{\lfloor 3}{\lfloor 1 \cdot \lfloor 2} + 1 + 1$

$= 3 + 3 + 1 + 1 = 8$ i.e 2^3

Superkey = A · { any subset of B, C, D }

$$= \left\{ \begin{array}{l} A, ABC, ABCD \\ AB, ACD \\ AC, ABD \\ AD \end{array} \right\} \Rightarrow 2^3 \text{ superkeys}$$ Ans

Ques: R(A, B, C, D, E). How many superkeys in Rel R if {A, BC} candidate keys.

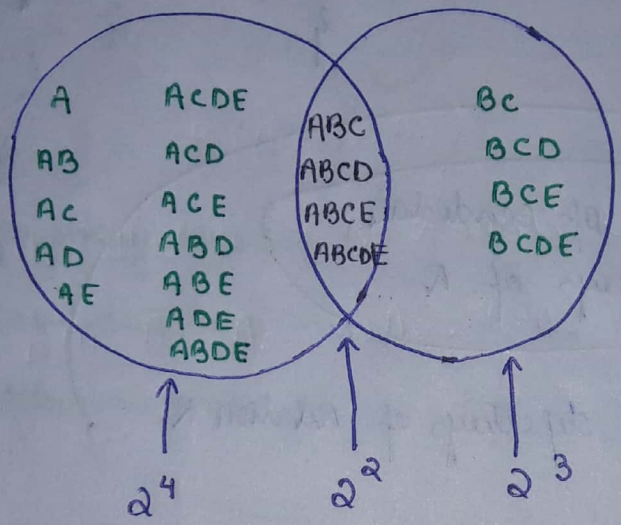
Soln:

A, B, C, D, E

Take A
Now 4 left
$\therefore 2^4$

Take BC
Now 3 left
$\therefore 2^3$

For common ABC
2 left
$\therefore 2^2$.



$$n(X \cup Y) = n(X) + n(Y) - n(X \cap Y)$$
$$= 2^4 + 2^3 - 2^2$$
$$= 16 + 8 - 4 \Rightarrow 20$$ Ans

Ques: R(A, B, C, D, E, F)

{AB, BCD} superkey = ?

Soln: No. of superkey = AB {CDEF} + BCD {AEF} − ABCD {EF}

$$= 2^4 + 2^3 - 2^2$$
$$= 20$$ Ans

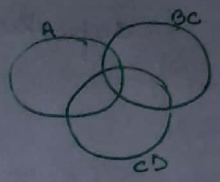
Ques: R(A, B, C, D, E). How many super keys in R; A, BC, CD are candidate key.?

superkey = A(BCDE) + BC(ADE) + CD(ABE) − ABC(DE) − BCD(AE) − ACD(BE) + ABCD(ϕ)

$= 2^4 + 2^3 + 2^3 - 2^2 - 2^2 - 2^2 + 2^1$

$= 18 + 4$

$= 22$ Ans



Ques: R(A1 A2 A3 A4 - - - - An)

How many super keys if

a) {A1} cand key.

b) {A1 A2 , A3 A4}

c) {A1, A2, A3}

d) {A1 A2 , A2 A3 , A4 A5}

e) If each attribute of R cand-key.

Solⁿ

(a) superkey = $A_1 \cdot \{A_2 A_3 - - An\}$

$= 2^{n-1}$ Ans

(b) superkey = $A_1A_2\{A_3A_4 - - An\} + A_3A_4\{A_1A_2 - - An\} - A_1A_2A_3A_4\{A5 - -\}$

$= 2^{n-2} + 2^{n-2} - 2^{n-4}$

$= 2^{n-4}(2^3 - 1)$

$= 7 \cdot 2^{n-4}$ Ans

(c) superkey = $A_1\{A_2A_3 \cdots An\} + A_2\{A_1A_3A_4 - An\} + A_3\{A_1A_2A_4A_5 - - An\} - A_1A_2\{A_3 - - An) - A_2A_3(- -) - A_3A_1(- -) + A_1A_2A_3(- -)$

$\Rightarrow 2^{n-1} + 2^{n-1} + 2^{n-1} - 2^{n-2} - 2^{n-2} - 2^{n-2} + 2^{n-3}$

$\Rightarrow 2^n - 2^{n-2} + 2^{n-3}$

$\Rightarrow 2^{n-3}\,[2^3 - 2 + 1]$

$\Rightarrow 7 \cdot 2^{n-3}$ Ans

d) superkey = $A_1A_2 + A_2A_3 + A_4A_5 - A_1A_2A_3 - A_2A_3A_4A_5 - A_1A_2A_4A_5 + A_1A_2A_3A_4A_5$

$= 2^{n-2} + 2^{n-2} + 2^{n-2} - 2^{n-3} - 2^{n-4} - 2^{n-4} + 2^{n-5}$

$= 2^{n-1} + 2^{n-2} - 2^{n-3} - 2^{n-3} + 2^{n-5}$

$= 2^{n-1} + 2^{n-2} - 2^{n-2} + 2^{n-5}$

$= 2^{n-5}\,[16 + 1] \Rightarrow 17 \cdot 2^{n-5}$ Ans

***e) Suppose, $R(A_1, A_2, A_3)$

If each attribute of R is cand. key,

$\{A_1, A_2, A_3, A_1A_2, A_2A_3, A_1A_3, A_1A_2A_3\}$

$\Rightarrow$ 7 superkeys

$\therefore$ for n attributes $\Rightarrow$ $2^n - 1$ max possible SK.

↓

for ϕ

Note –

→ No. of possible superkeys in R with n attribute is = $2^n - 1$ (If each attribute of R is cand. key)

→ If all attribute form one candidate key, then

No. of superkeys of R = 1

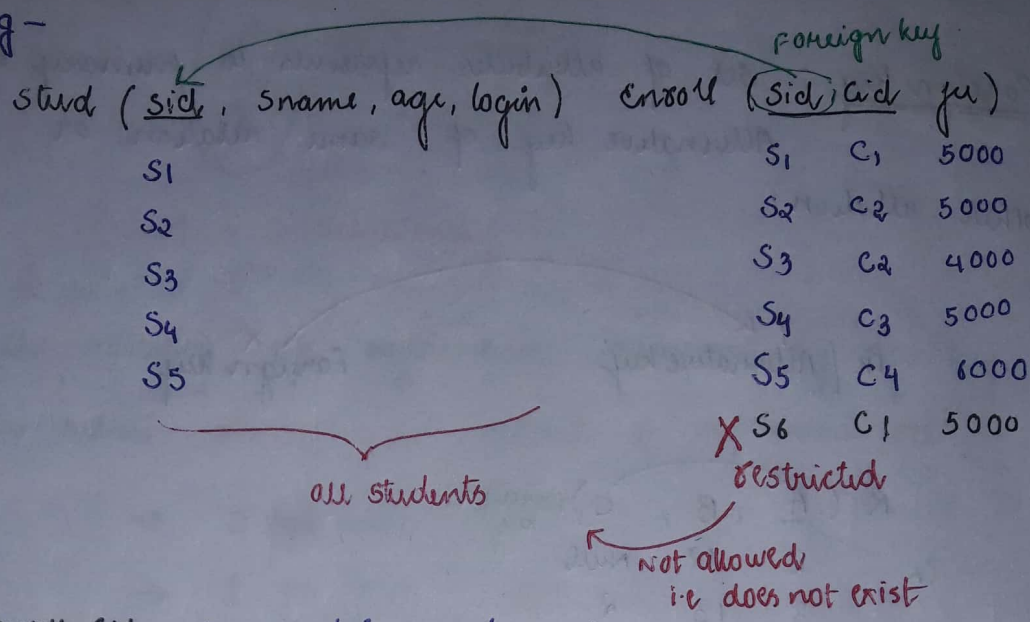
Eg – $R(A_1A_2A_3) \Rightarrow A_1A_2A_3\{\phi\}$

$\therefore 2^0 \Rightarrow$ 1 superkey

Foreign Key (Refrential Key)

"It is used to relate data b/w tables"

eg-



→ Enroll Sid always belongs to Stud Sid.

↑ Foreign key.

```
Create table stud
(
    Sid varchar(10) Primary key,
    Sname varchar(30),
    age integer,
    login varchar(30) UNIQUE
);
```

```
Create table Enroll
(
    Sid varchar(10);
    Cid varchar(10);
    Fee integer
    Primary key (Sid, Cid))
    FOREIGN KEY (Sid) REFERENCES
    Stud (Sid)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Referential constraint

↓

by default, it is NO ACTION

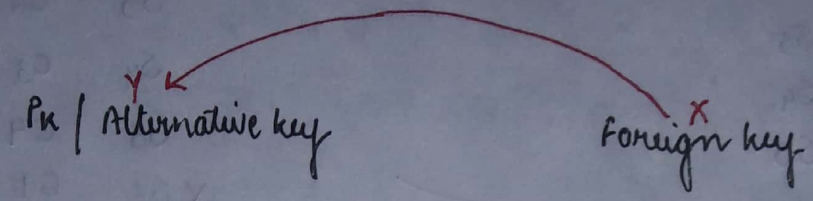
Either Primary key or alternative key

FOREIGN KEY (Sid) REFERENCES Stud / Sud (Sid)

↑ Enroll Sid ↑ Pk of Stud

→ **Foreign key**: Set of attributes references to primary key / Alternative key of same relation or some other relation.

⊙



⊙ R (A , B , C) Foreign key

Pk			
	2	12	Null
	3	13	2
	4	14	2
	5	15	3
	7	17	10

✗ cannot because $10 \notin A$

⊙ R (A , B) S (C , D)

2	12	Null	21
3	13	2	22
4	14	2	23
5	15	3	24
6	16	4	25

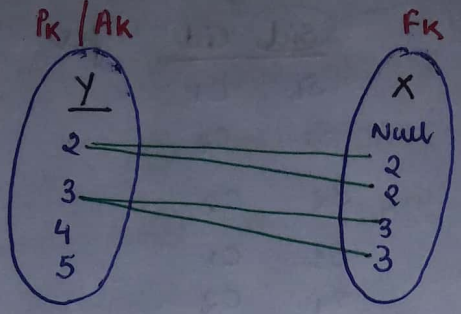
✗ does not reference any.

1 : Many Mapping

⊙ R (A , B_{Pk} , C)

alternate key			
2	12	Null	→ does not point to the Null of referenced key i.e Null of A.
3	13	2	
4	14	2	
5	15	3	
6	16	4	
Null	18	5	

⊙ Record with FK value as NULL is not in the relationship



{ 1:M + 0 relationship }

⊙ Each record of referencing relation (foreign key) can relate atmost one record of referenced rel^{n}.

→ 0 for NULL

→ 1 for rest

⊙ Each record referenced can be related by 0 or more (many) records of referencing relation.

Referencial Integrity Constraints —

4/11/17

Student —

Sid	Sname	age
S1		
S2		
S3		
S4		
S5		

↓
Referenced relation

Enroll —

Sid (foreign key)	Cid	fee
S1	C1	
S1	C2	
S2	C2	
S5	C1	
S4	C2	

Table which contain foreign key
Referencing Relation

Foreign key value must always be in corresponding primary key (i.e referenced relation)

→ Foreign key Constraints —

① Referenced Relation [Student]

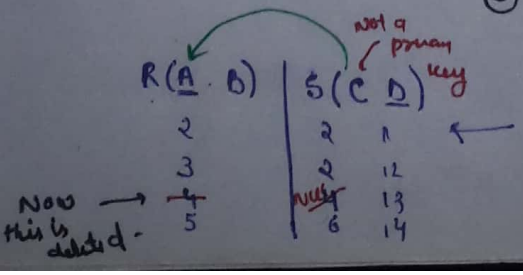
- Insertion — No violation
- Deletion — May cause violation

↓ solution for this

① ON DELETE NO ACTION (default)
deletion of referenced record restricted if foreign key violation occurs.

② ON DELETE CASCADE
Because of deletion of referenced record, DBMS deletes the related referencing record.

③ ON DELETE SET NULL
Allowed to delete referenced record if related FK field values are set to NULL (i.e not a primary key)



- Updation — may cause violation

↓ soln

① ON UPDATE NO ACTION (default)

② ON UPDATE CASCADE

③ ON UPDATE SET NULL

② Referencing Relation [Enroll]

- Insertion — may cause violation
- Deletion — no violation
- Updation — may cause violation.

eg—

```
create table Enroll
(  Sid
   Cid
   fee
   Primary ( )
   Foreign key (Sid) references student
   ON DELETE CASCADE
   ON UPDATE CASCADE
);
```

→ **Check Constraint :**

(constraint to attribute values)

It is used to specify set of accepted values of attribute (or) range of value to given attribute.

Eg —

```
create table stud
(
    sid varchar(10) Primary key,
    sname varchar(30),
    age integer check age >= 20 and age <= 30,
    games varchar(10) check ("cricket", "chess", NULL)
);
```

NULL is also Not allowed.

Here, NULL is allowed.

Ques: Assume a relation R(ABC) where AB combined is a primary key and another relation S(DEF) where D is primary key. Attribute 'E' of rel 'S' for ; foreign key reference to Rel R is allowed or not

R (A B C) — S (D E F)

4 5

4 6

5 4

5 6

X Not allowed to refer

2 fields are combined.

So the reffering i.e FK. must also be 2 fields

2) check constraint can be replaced by foreign key. T or F?

Not Allowed, False as check constraint is for range and FK is for values that belong to

Normalization

- It is used to eliminate or reduce redundancy in DB tables, or RDBMS table.
- Redundancy in DB table is present if two or more independent relations are stored in single RDBMS table.

Eg - Sid → Sname age (Student info) Cid → Cname Instructor (Course info) Sid Cid → fee (Mapping data)

Above are the 3 independent relations

↓ when these are maintained in a single table.

R

Sid	Sname	age	Cid	Cname	Instr	fee
S1	B	20	C1	DB	Korth	5000
S2	B	20	C1	DB	Korth	4000
S3	A	22	C1	DB	Korth	4000
S3	A	22	C2	Algo	Comman	6000
S3	A	22	C3	Algo	Allen	5000
			C4	CO	Hayes	

redundancy

① To delete student S3 → we need to delete this 3 record → but by doing so, we delete course data also (drawback)

redundancy as same student (enrolled with different course)

② we need to delete course C1, by doing so, we delete student data also (drawback)

③ If we need to insert a course, we need to add student info also (Insertion problem)

④ If we change the age of S2 in one 1 place ↓ may cause inconsistency

Problems because of redundancy

Because of redundancy, it may cause inconsistency

Database Anamalies

① **Updation Anamaly** —

If one redundent copy is updated and some other redundant copy is not updated, it causes inconsistency

② **Deletion Anamaly** —

Because of deletion of some data, we may loose some other important data

③ **Insertion Anamaly** —

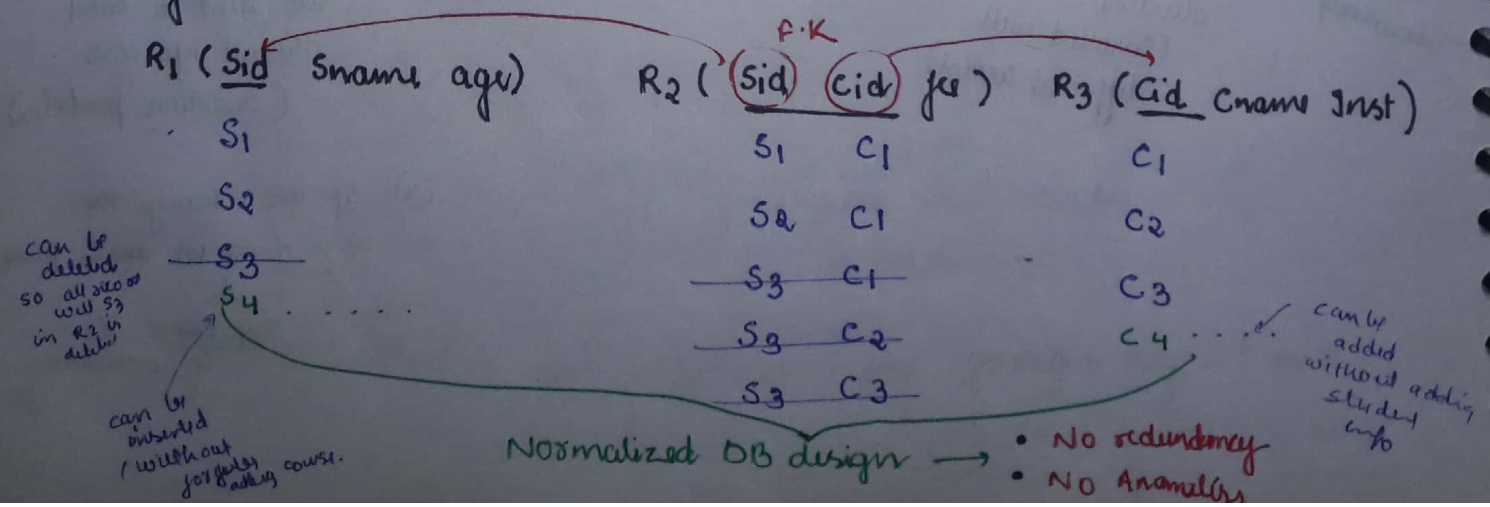
It is not possible to insert one independent data without other data (independent data)

If redundancy is reduced / eliminated, then DB anamalies are also reduced or eliminated. } Requirement of Normalization

↓ soln.

→ Decompose the DB relation into two or more sub relations to reduce / eliminate redundancy.

Eg. -



Functional Dependency (FD) —

(also called Single Valued Dependency)

- It is represented by '→' mark.

Eg. $X \longrightarrow Y$ (where X and Y are 2 different set of attributes)

- $X \longrightarrow Y$ implied in R

only if $t_1 \cdot X = t_2 \cdot X$ then $t_1 \cdot Y = t_2 \cdot Y$

(i.e for each 'X' value in relation R, there must be only one Y value)

Eg. –

R

	X	Y
t_1	x_1	y_5
t_2	x_1	y_5

R

	A	B	C
t_1:	4	8	2
	6	5	3
t_2:	4	8	4
	6	5	6
t_3:	4	8	5
	7	5	8
	9	6	7

C be the candidate key (i.e all the records are unique) → ∴ C → A ✓ Always, C → B ✓

Where ever A is same B must also be same ∴ A → B

A → B ✓

B → A ✗ (because for same B, different A)

Whereever A is not same B may be same or different.

C → A, C → B } Always ✓ if (C is candidate key or Super key)

↓

candidate key (or) Super key

$X \longrightarrow Y$

↓ determinant ↓ Determiner

① Trivial FD –

$X \to Y$ is trivial FD if and only if $X \supseteq Y$

(self dependency)

Eg- $A \to A$

$B \to B$

$C \to C$

$AB \to A$

$AB \to AB$

$X \supseteq Y$

[every possible trivial FD always exist in relation R

R(ABC) {AB → A}

45 6

45 7]

X should contain all elements from Y sides

② Non Trivial FD –

No common attribute over determinant and determiner i.e

$X \cap Y = \phi$, then $X \to Y$ is non trivial FD

Eg-

$A \to B$

$C \to A$

$C \to B$

$AC \to B$

③ Semi - Non Trivial FD –

combination of trivial and non trivial.

Eg -

$AC \to BC$

can be split as :

$AC \to B$ (non trivial)

$AC \to C$ (trivial)

$A \to AB$

$A \to A$ (trivial)

$A \to B$ (non trivial

$BC \longrightarrow AC$

$BC \rightarrow A$ (non trivial)

$BC \rightarrow C$ (trivial)

Ques:

R

A	B	C
2	4	6
3	4	7
2	4	5
5	4	7
6	5	7

Find set of Non trivial FD's ?

$A \rightarrow B$ ✓ $B \rightarrow A$ X $A \rightarrow BC$ X $BC \rightarrow A$ X

$B \rightarrow C$ X $C \rightarrow B$ X $B \rightarrow AC$ X $AC \rightarrow B$ ✓

$C \rightarrow A$ X $A \rightarrow C$ X $C \rightarrow AB$ X $AB \rightarrow C$ X

$\{A \rightarrow B, AC \rightarrow B\}$ Ans

If $A \rightarrow B$ and $A \rightarrow C$ true $\Longleftrightarrow A \rightarrow BC$ true

Ques: R (A B C)

4 4 7

4 7 7

4 4 5

4 7 5

6 4 7

Find set of non trivial FD's for given instance

$A \rightarrow B$ X $B \rightarrow A$ X $A \rightarrow BC$ X $BC \rightarrow A$ X

$B \rightarrow C$ X $C \rightarrow B$ X $B \rightarrow CA$ X $CA \rightarrow B$ X

$C \rightarrow A$ ~~X~~ $A \rightarrow C$ X $C \rightarrow AB$ X $AB \rightarrow C$ X

$\{\}$ Ans

→ Stud (Sid Sname age)

S_1	A	
S_2	B	
S_3	C	
S_4	C	→ Even though all Sname are same still we can't say Sname → Sid ✗ as in future S_4 with same name C can come

Here, { Sid → Sname, age }

↓ defined from constraint of DB design

→ R (MEid did)

e_1	d_1
e_1	d_2
e_2	d_3
e_3	d_4

MEid : emp who manages dept

"each dept must have only one manager"

"each emp can manage many dept" (constraints)

{ did → MEid }

↳ dependencies are defined based on given constraint

If all MEid are unique, then also we cant say MEid → did as in future same employee can manage any other dept.

→ R (A B C)

4	5	1
4	5	2
6	7	3
7	7	4
4	8	4

{ B → A not valid implied } True

{ A → B implied in R }

↓ we cannot say (because based on instance dependency cannot be defined)

∴ May or may not.

← as in future we could add (same A but diff B.

∴ Functional dependencies of relation R may or may not be defined over given data.

Armstrong Rules over FD's —

X, Y, Z are some set of attributes over the relation R.

① Reflexivity —

$X \rightarrow X$ always holds true
(as trivial FD is always valid)

② Transitivity —

If $X \rightarrow Y$ and $Y \rightarrow Z$ implied in relation R, then

$X \rightarrow Z$ also holds true in relation R.

Eg - R (A B C D) if $\{B \rightarrow C, C \rightarrow D\}$ exist in R

4	5	8	then, $B \rightarrow D$ also exist in R
4	5	8	
5	7	9	
5	7	9	
6	7	9	
6	7	9	

③ Augmentation —

If $X \rightarrow Y$ is true, then $XZ \rightarrow YZ$ will be true.

Eg - R (A B C D)

4	4	5	8
5	4	5	8
5	5	7	9
6	5	7	9
7	6	7	9
8	6	7	9

if $B \rightarrow C$
then
$AB \rightarrow AC$ is true
Whatever be the A value

but reverse is false
if $AB \rightarrow AC$, then $B \rightarrow C$ ✗ False.

④ Split and Merge Rule –

- If $X \rightarrow YZ$, then $X \rightarrow Y$, $X \rightarrow Z$ (Split rule)

But reverse i.e. $XY \rightarrow Z$ exist in R,
then $X \rightarrow Z$ or $Y \rightarrow Z$ may not exist in R
↓
they exist together.

- If $X \rightarrow Y$, $X \rightarrow Z$, then $X \rightarrow YZ$ (Merge Rule)

⑤ Attribute closure (X^+)

X^+ = set of all possible attributes which are determined by X recursively.

Ques: Given FD set

$\{ A \rightarrow B, BC \rightarrow D, CD \rightarrow E, B \rightarrow C, EF \rightarrow G \}$

Find A^+ ?

$A^+ = \{ A, B, C, D, E \}$ Ans $\quad A \rightarrow ABCDE$

$A \rightarrow A$ reflective — A
$A \rightarrow B$ — AB
$B \rightarrow C$ — ABC
$BC \rightarrow D$ — A, B, C, D
$CD \rightarrow E$ — A, B, C, D, E

$B^+ = \{ B, C, D, E \}$ Ans $\quad B \rightarrow BCDE$

$(AF)^+ = \{ A, F, B, C, D, E, G \}$ Ans

$AF \rightarrow ABCDEFG$

Super key —

(It can be identified by FD's)

Relation R with FD set (F),

R (X Y Z)

2
3
4
5
6

X : super key of R

then { X → YZ }

> X is superkey of relation R iff X^+ determines all attributes of relation R

Eg -

R (A B CD)

2 4
3 5
7 6 7 9
Not allowed ✗ 7 6 7 8
allowed ✓ 8 6 7 8
(A must be different)

given { A → B, B → CD }

B^+ = { B, C, D }

Since, it does not determine all the attributes, it does not determine A

∴ Not superkey

A^+ = { A, B, C, D }

A is Super key as it determines all

For superkey

↳ All the values are unique (different)

Candidate key –

(Minimal super key)

X is candidate key of rel R iff

① X must be super key of relation R

(i.e X^+ should determine all attribute of R)

and ② NO proper subset of X is super key

($\nexists$ $Y \subset X$ such that

Y^+ should not determine all attribute)

Eg- For relation R(A B C D E) with dependencies

{AB → C, B → D, A → E}

$(AB)^+ = \{A, B, C, D, E\}$

↑

AB is super key

Now, proper subset of AB → {A, B}

Not super key

$A^+ = \{A, E\}$

$B^+ = \{B, D\}$

∴ AB is candidate key.

* If one attribute is super key i.e A is super key, then it itself will be candidate key.

***

Ques: Find No. of candidate keys of given relation R?

① $R(A\,B\,C\,D)$ $\{A \to B,\ B \to C\}$ — not in FD ∴ will be part of candidate key.

$A^+ = \{A, B, C\}$

D is not determined

D cannot be determined by any

$\therefore (AD)^+ = \{A, B, C, D\}$ — superkey

Also, $A^+ = \{A, B, C\}$, $D^+ = \{D\}$ } Not super key

$\therefore$ (AD) is candidate key.

determinant	determiner
A	B
B	C

- Only in determinant but not in determiner (so it will be a part of candidate key)

Note ⊙ X must be a part of every candidate key of R, only if:

① X is not present in non trivial FD set of relation R (Here D)

(OR)

② X belongs to some determinant but does not belong to determiner of non trivial FD (Here A)

⊙ if $X \to Y$ is non trivial FD with 'Y' as prime attribute of relation R, then R has atleast 2 candidate key

$X \to Y$

Prime attribute (i.e Y belong to any of candidate key)

$X \longrightarrow Y$

Y → Prime attribute

cand key

$(WY)^+ = \{$ All attributes of $R\}$

if $X \rightarrow Y$;

⇓ (we can replace Y with X)

One more possible candidate key exist

$(WX)^+ = \{ W, X, Y,$ all $\}$

X determine Y and with W, Y all.

Here, in this ques,

prime attribute is $(AD)^+$

A or D is not present on RHS of any FD

∴ only 1 candidate key.

② $R(ABCDE)$

$\{AB \rightarrow C,\ C \rightarrow D,\ D \rightarrow EA\}$

B is not on RHS of any FD

∴ B will be in all candidate key.

super key → $(AB)^+ = \{ A\ B\ C\ D\ E \}$

Now Testing for minimal super key

a) Find proper subset of super key closure and it should not determine all attribute

$A^+ = \{A\}$ ✗

$B^+ = \{B\}$ ✗

AB is minimal. ⟹ candidate key.

b) some attribute of super key should not determine other attribute of super key.

AB

$A \rightarrow B$ ✗ i.e $A^+ = \{A\}$

$B \rightarrow A$ ✗ i.e $B^+ = \{B\}$ ∴ AB is minimal.

AB

↓

More candidate keys are

{ BD, BC }

∴ { AB, BD, BC }

3 candidate keys in R.

BD

We need to check each for minimal

B^+ = { B }

D^+ = { D, E, A }

Here it is minimal

↓

D is on RHS

C → D

∴ BC

③ R(ABCD)

{ AB → CD, C → A, D → B }

AB^+ = { A, B, C, D }.

check for minimal.

A^+ = { A }

B^+ = { B }

A^+ should not contain B and B^+ should not contain A

∴ AB

↓

CB = { C, B, A, D }

↓

CD = { C, D, A, B }

AD = { A, B, C, D }

B^+ = { B }

C^+ = { C, A } ✓

C^+ = { C, A }

D^+ = { D, B } ✓

{ AB, AB, CB, CD }

(4)

④ R(ABCDE)

{ A → BC, CD → E, E → A, B → D }

A^+ = { A, B, C, D, E }

E^+ = { A, B, C, D, E }

CD = { A, B, C, D, E }

CB = { C, B, D, E, A }

AD = { A, B, C, D, E }

ED = { A, B, C, D, E }

(4)

⑤ R (ABC DEF}

{ AB → C , C → DE , E → F , F → B }

A will be part of every.

AB = { A B C D E F }

AF = { A, F, B, C, D, E }

AE = { A, B, C, D, E, F }

AC = { A, C, D, E, F, B }

(4)

⑥ R (ABC DEF}

{ AB → C , C → D , CD → E , DE → F , EF → B }

A will be part of all

✓ AB = { A B C D E F } $A^+ = A$, $B^+ = B$ ✓

✓ AEF = { A, E, F, B, C, D }

$A^+ = \{A\}$, $E^+ = \{E\}$, $F^+ = \{F\}$.

$(AE)^+ = \{A, E\}$ $(EF)^+ = \{E, F, B\}$

$AF^+ = \{A\ F\}$

✓ AED = { A, E, D, F, B, C } $AD^+ = \{A, D\}$, $DE = \{D, E, F, B\}$, $AE = \{A, C\}$

✗ ACD = { A, C, D, E, F, B } $AC = \{A, C, D, E,$ contoi D

✓ AC = { A, C, D, E, F, B } $A = \{A\}$

~~$AC = \{A, C, D$~~

$C = \{C, D, E, F, B\}$

(4)

7) R(ABC)

No non trivial FD in R.

ABC is candidate key.

Note – if no non-trivial FD, then candidate key is all the attribute collectively.

Membership test –

Given FD set (F),

> $X \to Y$ FD is logically implied (member) in FD set iff X^+ must determine 'Y' in FD set (F).

$\{ \text{----------} \} \Leftarrow X \to Y$

if X^+ consist Y, then
$X \to Y$ is member of FD set
Otherwise $X \to Y$ not member.

Ques: $\{ AB \to C, BC \to D, D \to E, DE \to F \}$

a) Test $AB \to F$ member or not.

$(AB)^+ = \{ A, B, C, D, E, F \}$ — contain F

∴ Member.

b) Test $BC \to A$ member or not.

$(BC)^+ = \{ B, C, D, E, F \}$

Not Member
as A is not in $(BC)^+$

Closure of FD set — (NP complete problem / takes exponential time)

Given FD set (F)

F^+ : Set of all possible FD's which can be derived by using given FD's

Eg- $F = \{ A \to B , B \to C \}$

$A^+ = ABC$ $B^+ = BC$ $C^+ = C$ $(AB)^+ = ABC$ $(BC)^+ = BC$ $(AC)^+ = ABC$ $(ABC)^+ = ABC$

$F^+ =$ {
$A \to A$, $A \to B$, $A \to C$, $A \to AB$, ⋮, $A \to ABC$
$B \to B$, $B \to C$, $B \to BC$
$C \to C$
$AB \to A$, $AB \to B$, ⋮, $AB \to ABC$
$BC \to B$, $BC \to C$, $BC \to BC$
$AC \to A$, ⋮, $AC \to ABC$
$ABC \to A$, $ABC \to AB$
}

Note - ① FD set closure identification

→ NPC problem
(exponential TC Problem)

② $F \cong F^+$

↓

equal expressive power.

Equality of FD set —

Given FD set F and G

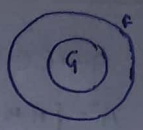
1) F and G have equal expressive iff $F^+ \cong G^+$ → Theoretically, and is complex method.

2) F and G are equal expressive iff —

a) F covers G :

Every FD of G set must be a member of F set i.e

$F \supseteq G$



and b) G covers F :

Every FD of F set must be member of G set. i.e

$F \subseteq G$

F covers G	G covers F	
True	True	$F \equiv G$
True	False	$F \supset G$
False	True	$F \subset G$
False	False	F and G are not comparable i.e (F) (G) or (F G)

Ques: $F = \{ A \to B,\ B \to C,\ C \to A \}$

$G = \{ A \to BC,\ B \to AC,\ BC \to A,\ AB \to C \}$

Which is true .

a) $F \subset G$

b) $F \supset G$

c) $F \equiv G$

d) None of them

F covers G (every G member of F)

~~$A^+ = \{A, B, C\}$~~

Take each member of G and check if present in F or not

From G: $A \to BC$ $B \to AC$ $BC \to A$ $AB \to C$

in F: $A^+ = \{A, B, C\}$ $B^+ = \{B, C, A\}$ $(BC)^+ = \{B, C, A\}$ $(AB)^+ = \{A, B, C\}$

∴ True.

G covers F

From F: $A \to B$ $B \to C$ $C \to A$

in G: $A^+ = \{A, B, C\}$ $B^+ = \{B, A, C, A\}$ $C^+ = \{C\}$

False

Ques: $F = \{A \to BCDEF,\ BC \to ADEF,\ B \to F,\ D \to E\}$

$G = \{A \to BC,\ BC \to AD,\ B \to F,\ D \to E\}$

F covers G –

From G in F

$A \to BC$: $A^+ = \{ABCDEF\}$ ✓

$BC \to AD$: $(BC)^+ = \{BCADEF\}$ ✓

$B \to F$: $B^+ = \{BF\}$ ✓

$D \to E$: $D^+ = \{D, E\}$ ✓

G covers F —

$A \to BCDEF$: $A^+ = \{ABCDEF\}$ ✓

$BC \to ADEF$ ✓

$B \to F$ ✓

$D \to F$ ✓

$\therefore F \cong G$

Minimal Cover (or) Canonical Cover —

Given FD set F,

Minimal cover of FD set (F):
Minimal possible FD's which are logically equal to F.

Procedure to find minimal cover:

Given FD set (F)

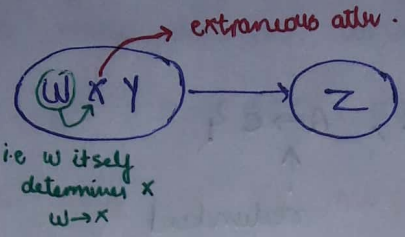
① Remove extraneous attributes from every determinant of FD set (F)

> if $wxy \to z$, $w^+ = x$; then attr 'x' is extraneous in $wxy \to z$: f_1

② Remove all redundant FD's from step ① result

> $X \to Y$ redundant in FD set(F_1) in above
> iff $X \to Y$ is member of
> $\{ F_1 - \{X \to Y\} \}$ in FD set

Note – Extraneous attribute over determinant –



$\{ WXY \to Z, W \to X \} \equiv \{ WY \to Z, W \to X \}$

we remove this extra.

Ques: ① $\{ ABCD \to E, AB \to F, F \to D \}$

$AB^+ = \{ A, B, F, D \}$ (D – extra)

extraneous

$\{ ABC \to E, AB \to F, F \to D \}$

② $\{ ABCD \to E, AC \to F, F \to B, C \to D \}$

$(AC)^+ = \{ A, C, F, B, D \}$

extraneous

$\therefore \{ AC \to E, AC \to F, F \to B, C \to D \}$

③ $\{ AB \to C, A \to B, B \to A \}$

$A^+ = \{ A, B \}$ Here B extraneous (or) $B^+ = \{ B, A \}$ A is extraneous

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

Either A or B is extraneous but not both.

Redundant FD in FD set —

$X \to Y$ in FD set (F) is redundant iff

$X \to Y$ must implied (number) in FD set $\{F - (X \to Y)\}$

$\{F - (X \to Y)\}$

After this also

$X \to Y$ is member of F

$\therefore$ $X \to Y$ is redundant in FD set (F)

Ques: Find redundant —

① $F = \{A \to B,\ B \to C,\ A \to C\}$

↑ redundant

② $F = \{AB \to C,\ BC \to D,\ AB \to D,\ CD \to E,\ BC \to E\}$

↑ hold this assume it is not there now (chapter 6) find $(AB)^+$ if it gives C, then redundant else not

redundant

Ques: $F = \{A \to BCDEF,\ BC \to ADEF,\ ACF \to D,\ D \to E,\ B \to F\}$

Find Minimal cover.

ACF → extraneous

Remove extraneous attr —

$F = \{A \to BCDEF,\ BC \to ADEF,\ A \to D,\ D \to E,\ B \to F\}$

Find redundant —

Here split the FD

$A \to B$	$BC \to A$ ✓	$A \to D$	$D \to E$	$B \to F$
$A \to C$ ✓	$BC \to D$ ✓	✗	✓	✓
$A \to D$ ✗	$BC \to E$ ✗	$(B)^+ = B, C, A, D, E$		
$A \to E$ ✗	$BC \to F$ ✗	$(BC)^+ = BC, A, D, E, F$		
$A \to F$ ✗		only $B \to F$		
		$\therefore BC \to F$		

∴ Minimal cover is

$F = \{ A \to BC,\ BC \to AD,\ D \to E,\ B \to F \}$ Ans.

Here we have 2 choices

$A \to D$: $BC \to D$

$A \to D$ ✗ removed — $BC \to D$ ✓

$A \to D$ ✓ retained : $BC \to D$ is redundant

∴ one more minimal cover exist.

$F = \{ A \to BCD,\ BC \to A,\ D \to E,\ B \to F \}$

Note ① Minimal cover of FD set (F) may not be unique but, all minimal covers are logically equal.

② Also, minimal cover of is equal to FD set (F)

$F m_1 \cong F m_2 \cong F$ expressive power remains same.

Properties of Decomposition —

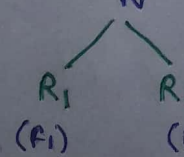
Decomposition of relation (R) into subrelations not causes any integrity violation if.

→ it is loss less join decomposition (and)

→ dependency preserving decomposition.

Dependency Preserving decomposition –

R with FD set (F)

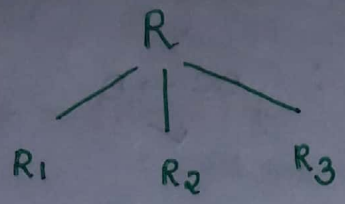


$F_1 \cup F_2 \cong F$

Because of the fact that decomposition should not lost any FD of FD set

Loss less Join decomposition —

- JOIN result of subrelations must be equal to the original relation.



$$R_1 \bowtie R_2 \bowtie R_3 \cong R$$

- Relation (R) is decomposed into R_1 R_2 R_3 ---- R_3 sub relations.

In general,

$$[R_1 \bowtie R_2 \bowtie \dots \bowtie R_n] \supseteq R$$

- If $(R_1 \bowtie R_2 \bowtie \dots \bowtie R_n) \cong R$, then it is loss less join decomposition.
- If $(R_1 \bowtie R_2 \bowtie \dots \bowtie R_n) \supset R$, it is called as lossy join decomposition

↓

Here, some extra tuples are generated because of wrong decomposition, these are called spurious tuples.

R

Sid	Sname	Cid
S_1	A	C_1
S_1	A	C_2
S_2	B	C_2
S_3	B	C_3

{ Sid → Sname }

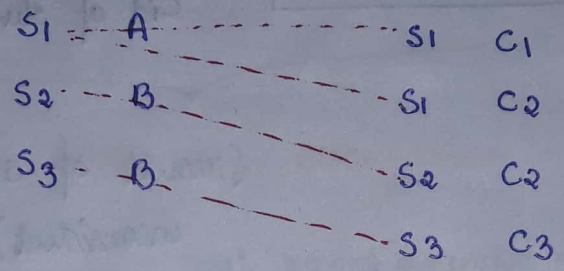
candidate key : Sid Cid

Cid of Student $S_2 = C_2$

① decomposed into

where both are =

R_1 (Sid Sname) R_2 (Sid Cid)



⇓ Join

Sid	Sname	Cid
S_1	A	C_1
S_1	A	C_2
S_2	B	C_2
S_3	B	C_3

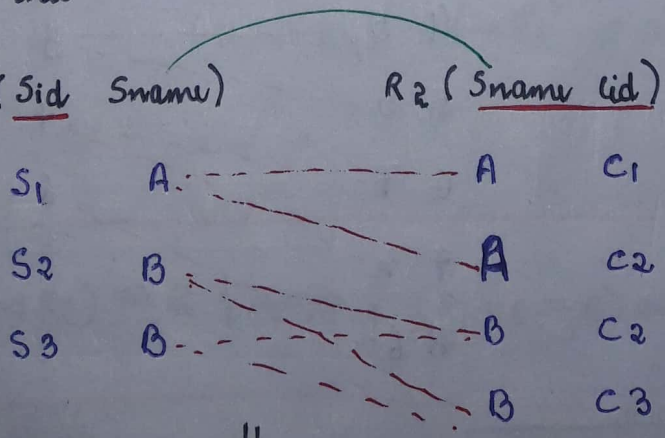
Cid of $S_2 = C_2$

$R_1 \bowtie R_2 \cong R$

loss less decomposition.

② decomposed into

R_1 (Sid Sname) R_2 (Sname Cid)



⇓

Natural Join $R_1 \bowtie R_2$

$R_1 \bowtie R_2$

Sid	Sname	Cid
S_1	A	C_1
S_1	A	C_2
S_2	B	C_2
S_2	B	C_3
S_3	B	C_2
S_3	B	C_3

extra tuple (spurious tuple)

$(R_1 \bowtie R_2) \supset R$

↓

Lossy Join decomposition (consistency is lost)

Cid of student S_2 = C_3, C_2

✗ wrong o/p

(result of SQL queries is inconsistent)

Ques: R (A B C) { A → B , B → C }

4	5	4
5	5	4
6	7	5
7	7	5
8	6	5

candidate key A

① decomposed into R_1 (A B) R_2 (B C)

A → B

B → C

candidate key of R_2 B

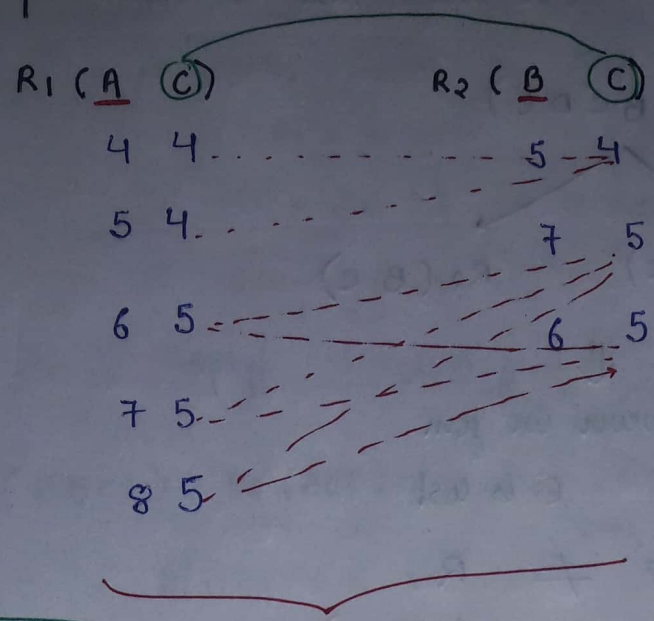
R_1 A	R_1 B	R_2 B	R_2 C
4	5	5	4
5	5	7	5
6	7	6	5
7	7		
8	6		

$R_1 \bowtie R_2 \supseteq R$

Total record generated = 6

Lossless join decomposition.

② decomposed into



C is common attribute and it is not the unique key of any relation

∴ decomposition is lossy

$(R_1 \bowtie R_2) \supset R$

lossy Join Decomposition

Total record generated = 8

Note-

Relation R with FD set (F) decomposed into R_1 R_2 sub relation

→ given decomposition is lossless join decomposition iff

① $(R_1 \cup R_2) = R$ // every attr of R is in some sub rel.

because of this no colour is lost

and ② $(R_1 \cap R_2) \longrightarrow R_1$ // $R_1 \cap R_2$ is superkey for R_1

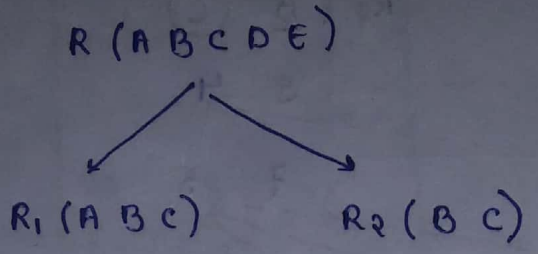
(or)

$(R_1 \cap R_2) \longrightarrow R_2$ // $R_1 \cap R_2$ is superkey for R_2

because of 2nd condition NO spurious record

$\{(R_1 \bowtie R_2) = R\} \iff \{(R_1 \cup R_2 = R)$ and $(R_1 \cap R_2 \to R_1$ or $R_1 \cap R_2 \to R_2)\}$

→ $R_1 \cup R_2 = R$ (It ensures that no column is lost).



⇓

However we join

E is lost.

$R_1 \bowtie R_2 \neq R$

(ABCD) (ABCDE)

Ques: R (A B C D E)

{ AB → C , C → D , B → E }

Test whether given decompositions are lossless or lossy.-

i) R_1 (A B C) R_2 (C D)

lossy. $R_1 \cup R_2 \neq R$ ✗ E is lost

ii) R_1 (A B C) R_2 (D E)

lossy. $R_1 \cup R_2 = R$ ✓ No common attribute

$R_1 \cap R_2 = \phi$ ✗

iii) R_1 (A B C) R_2 (C D E)

lossy

$R_1 \cup R_2 = R$ ✓ (A B C D E

$R_1 \cap R_2 = C$ ✗

$C^+ = \{C, D\}$

↑ Not super key of R_1 or R_2

iv) $R_1(ABCD)$ $R_2(BE)$

$R_1 \cup R_2 = R$ ✓

$R_1 \cap R_2 = B$

$B^+ = \{B, E\}$ i.e it determines all for R_2

↓

super key (R_2) ✓

~~lossy~~ loss less join

v) $R_1(ABCD)$ $R_2(ABE)$

$R_1 \cup R_2 = R$ ✓

$R_1 \cap R_2 = \{A, B\}$

$(AB)^+ = ABCDE$

∴ determines both R_1 and R_2

∴ super key in both relation

∴ loss less join

5/11/17

Ques: $R(ABCDEFG)$

$\{AB \rightarrow C, BC \rightarrow A, AC \rightarrow B, B \rightarrow D, DE \rightarrow F, F \rightarrow G\}$

i) $D_1 \{ABC, DEF, FG\}$

$R_1(ABC)$ $R_2(DEF)$ $R_3(FG)$

$R_1 \cup R_2 \cup R_3 \equiv R$ ✓

Join relations only if common attribute key of any subrelation

$\Rightarrow$ (LLJ)

lossless join

$R_1(ABC)$ $R_2(DEF)$ $R_3(FG)$

$R_1(ABC)$ $R_{23}(DEFG)$

$R_2 \cap R_3 = F$

$F^+ = \{FG\}$ ∴ F key for R_3

∴ join

No common attribute ∴ lossy

11) D2 { ABC, BD, DEF, FG, ABDE }

R_1 (ABC) R_2(BD) R_3(DEF) R_4(FG) R_5 (ABDE)

$R_1 \cap R_2 = \{B\}$

$B^+ = \{B, D\}$

$\therefore$ B is key for R_2

$\therefore$ Join

R_{12} (ABCD) R_3 (DEF) R_4 (FG) R_5 (ABDE)

$R_{12} \cap R_3 = D$

$D^+ = \{D\}$

does not determine any $\therefore$

Not allowed to join

$R_3 \cap R_4 = F$

$F^+ = \{F, G\}$

$\therefore$ key for R_4

↓ Join

R_{12} (ABCD) R_{34} (DEFG) R_5 (ABDE)

$R_{12} \cap R_5 = \{ABD\}$

$(ABD)^+ = \{ABCD\}$

$\therefore$ it is key for R_{12}

↓ Join

R_{125} (ABCDE) R_{34} (DEFG)

$R_{125} \cap R_{34} = \{DE\}$

$(DE)^+ = \{DEFG\}$

$\therefore$ Key for R_{34}

↓ Join

R_{12345} (ABCDEFG)

$\therefore$ Lossless Join Decomposition

Dependency Preserving decomposition —

- Relational schema R with FD set (F) decomposed into R_1 R_2 R_3 ---- R_n sub relations with F_1 F_2 F_3 ... F_n FD's.
- In general,

$$\{F_1 \cup F_2 \cup F_3 \cup \dots F_n\} \subseteq F$$

- If $[F_1 \cup F_2 \cup \dots F_n] \cong F$, then decomposition is dependency preserving decomposition (every FD of F is preserved in sub relation)
- If $[F_1 \cup F_2 \cup \dots F_n] \subset F$, then decomposition is not dependency preserving decomposition.

↓

Here, because of decomposition, some FD of F is lost.

Ques: R(A B C D E F G)

$\{AB \to C, BC \to A, AC \to B, B \to B, DE \to F, F \to G\}$

D_2 $\{ABC, BD, DEF, FG, ABDE\}$.

$R_1(ABC)$	$R_2(BD)$	$R_3(DEF)$	$R_4(FG)$	$R_5(ABDE)$
↓	↓	↓	↓	↓
$AB \to C$	$B \to D$	$DE \to F$	$F \to G$	$B \to D$
$BC \to A$				
$AC \to B$				

All the dependencies are preserved in some relation.

$\therefore [F_1 \cup F_2 \cup F_3 \cup F_4 \cup F_5] \cong F$

$\therefore$ Dependency Preserved.

Ques: R(ABCDE)

$\{A \to B,\ B \to C,\ C \to D,\ D \to BE\}$

decomposed to $\{AB,\ BC,\ CD,\ DE\}$

$R_1(AB)$ ↓ $A \to B$

$R_2(BC)$ ↓ $B \to C$

$R_3(CD)$ ↓ $C \to D$

$R_4(DE)$ ↓ $D \to E$

$B^+ = \{B, C, D, E\}$
A is not present so $B \to A$ will not be there.

$C^+ = CDBE$

$\therefore C \to B$

Here, $D \to B$ is not present directly.
So, we find all hidden FD.

$D^+ = DBEC$

$D \to C$

$E^+ = E$

NOW,

$A \to B$ ✓

$B \to C$ ✓

$C \to D$ ✓

$D \to BE$ ✓ as $D^+ = DBEC$

$\therefore F_1 \cup F_2 \cup F_3 \cup F_4 \cong F$

dependency preserved.

Ques: R(ABCD)

$\{AB \to CD,\ D \to B\}$

decomposed into $\{ACD,\ BD,\ ABC\}$

$R_1(ACD)$

F_1

compute closure of all proper subset of (ACD)

$A^+ = A$, $B^+ = B$, $C^+ = C$

$AB, BE; \ldots$

$(AD)^+ = AD, B, C$

$AD \to C$

$AB \to C$ ✓

$AB \to D$ ✗

$D \to B$ ✓

we find $(AB)^+$ in subrelation

$(AB)^+ = ABC$ (D is not here)

$\therefore$ lost

$R_2(BD)$

F_2

$D \to B$

$B^+ = \{B\}$

$R_3(ABC)$

F_3

$AB \to C$

$C^+ = \{C\}$

$(AB)^+ = ABCD$

All possible FD's

$\therefore$ Dependency Not preserved

($AB \to D$ is lost because of decomposition)

Ques: R (ABC DEF)

{ A → BCDEF , BC → ADEF , B → F , D → E }

decomposed to { ABC, BCD, BF, DE }

R_1 (ABC)	R_2 (BCD)	R_3 (BF)	R_4 (DE)
↓	↓	↓	↓
$A \rightarrow BC$	$BC \rightarrow D$	$B \rightarrow F$	$D \rightarrow E$
$(BC)^+ = BCADEF$	$D^+ = \{D, E\}$	$F^+ = \{F\}$	$E^+ = \{E\}$.
$BC \rightarrow A$			
(F_1)	(F_2)	(F_3)	(F_4)

~~$(AB)^+ = ABCDEF$~~

$A \rightarrow BCDEF$

A^+ in subrelation = A B C D E F

∴ all are satisfied ✓

$BC \rightarrow ADEF$

$(BC)^+$ in subrel = B C D E F A

all satisfied ✓

$B \rightarrow F$ ✓

$D \rightarrow E$ ✓

$F_1 \cup F_2 \cup F_3 \cup F_4 \cong F$

∴ Dependency Preserved.

Normal Forms —

- Used to identify degree of redundancy (also used to eliminate / reduce redundancy)

Redundancy in relation (R) can be because of

Non Trivial FD ($X \to Y$)
(Single valued dependency)

~~Used~~ to eliminate redundancy over FD's, relation should decomposed into "BCNF"

Non Trivial MVD ($X \to\to Y$)
(Multi valued dependency)

To eliminate redundancy over MVD's, relation should decompose into "4NF"

Various Normal Forms —

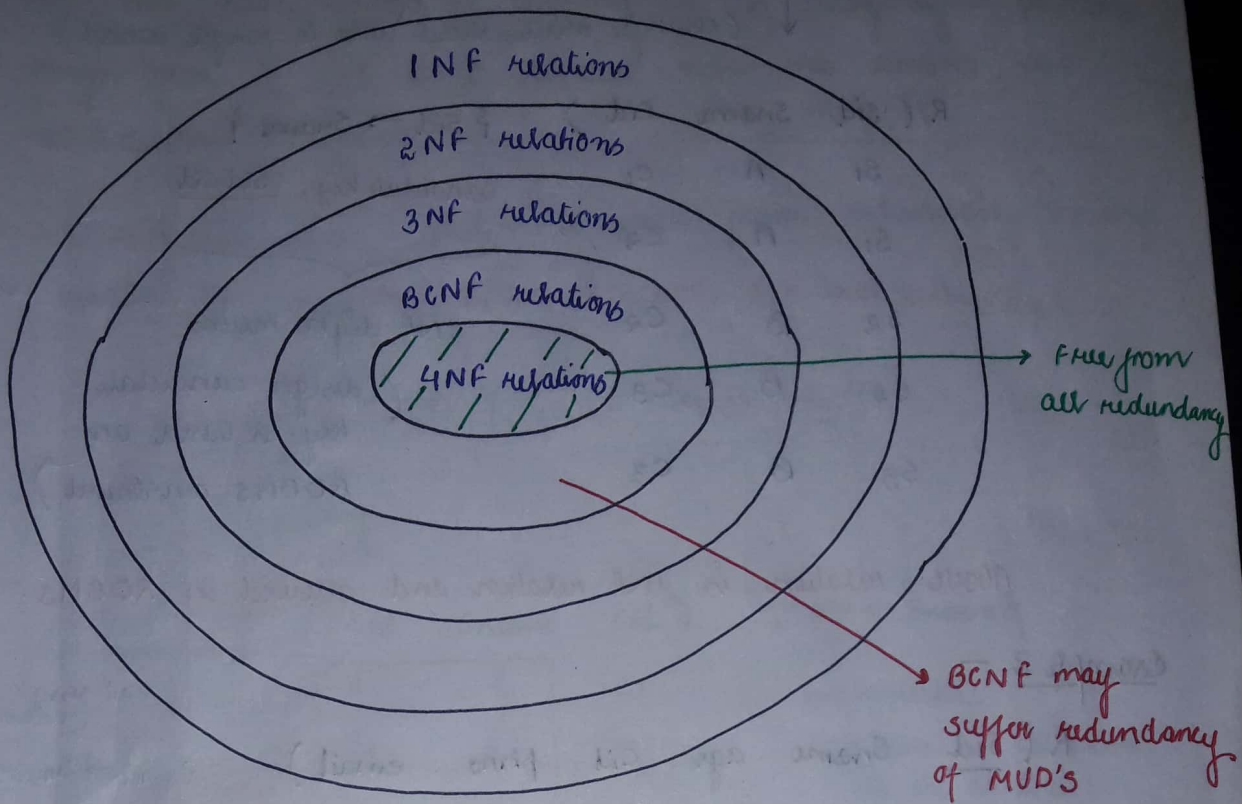
defined over FD's $X \to Y$:

1. 1NF
2. 2NF
3. 3NF
4. BCNF → if BCNF is satisfied then 0% redundancy over FD set of relation. (i.e if rel in BCNF)
   → BCNF rel may consist redundancy over MVD's
5. 4NF

defined over MVD's $X \to\to Y$

0% redundancy over FD's and MVD's if rel is in 4NF

If the rel^{n} is not in INF, then it does not satisfy RDBMS rules.



First Normal Form (1NF) —

- It is default NF of RDBMS Relation.
- definition : Relation R in 1NF iff No multivalued attribute in R

(OR)

Every attr of rel R must be atomic (single value).

- No 2 record of relation must be same (i.e the relation must have unique candidate key)

R (sid	Sname	Cid)
S_1	A	C_1/C_2
S_2	B	C_2/C_3
S_3	B	C_3

multivalued attribute (set of multiple value over attribute for any record)

{ sid → Sname } Not in 1NF (1-many or 1a^{n}) { sid → Cid } ✗

↓ converting to INF design
(change multivalued attr to single valued)

R (sid Sname Cid) { Sid → Sname }

S_1	A	C_1
S_1	A	C_2
S_2	B	C_2
S_2	B	C_3
S_3	B	C_3

candidate key: Sid Cid

INF design means (design candidate key is based on RDBMS constraint)

Above relation is INF relation and allowed in RDBMS.

Example 2 –

all single valued.

R (Sid Sname age Cid phno email) { Sid → Sname age }

S_1	A	20	$C_1 \mid C_2$ (2)	$P_1 \mid P_2 \mid P_3$ (3)	$e_1 \mid e_2 \mid e_3 \mid e_4$ (4) ⇒ 24
S_2	B	22	$C_2 \mid C_3$ (2)	P_4 (1)	$e_5 \mid e_6$ (2) ⇒ 4

↓ converting to INF design
(RDBMS design)

R (Sid Cid phno email Sname age) { Sid → Sname age }

cand key: Sid Cid phno email
all multivalued value.

S_1	C_1	P_1	e_1	A	20
S_1	C_1	P_2	e_2	A	20
S_1	C_1	P_3	e_3	A	20
⋮	⋮	⋮	⋮	⋮	⋮
S_1	C_2	P_1	e_1	A	20
⋮	⋮	⋮	⋮	⋮	⋮
S_2	C_2	P_4	e_5	B	22
S_2	C_2	P_4	e_6	B	22
S_2	C_3	P_4	e_5	B	22
S_2	C_3	P_4	e_6	B	22

Total record for S_1 = 2×3×4 = 24

for S_2 = 2×1×2 = 4

Note

In INF relⁿ degree of redundancy is very high because design goal of INF is to convert data into RDBMS, but not to decrease redundancy.

→ $X \to Y$ is non trivial FD which form redundancy in any relation R iff non trivial FD with X is not superkey

$X \longrightarrow Y$: non trivial FD

X → not superkey

Eg- R (Sid Sname Cid) { Sid → Sname }

S_1	A	C_1
S_1	A	C_2
S_1	A	C_3
S_2	B	C_3

redundancy

Sid → not superkey (as does not determine all the attributes)

∴ repeated in table (Eg- S_1)

→ $X \to Y$ FD in R does not form any redundancy iff

① Trivial FD

$X \supseteq Y$

(or)

② X must be a super key.

Eg- R (Sid Sname age) { Sid → Sname age }

Sid → superkey

S_1	A	20
S_2	A	20
S_3	B	22
S_4	B	20

No Redundancy.

Eg – R(A B C) $\{A \rightarrow BC\}$

A is a super key

No Redundancy

A	B	C
1	2	5
2	2	5
3	3	7
4	3	7
5	3	7

Here No relation b/w B and C.

Eg – R(A B C) $\{A \rightarrow B, B \rightarrow C\}$

1	2	5
2	2	5
3	3	7
4	3	7
5	3	7

A → B : A is super key

B → C : B is not a super key

↓

Forms redundancy in relation R

(i.e whatever will be B's value and C's value both will be same in all case)

Ques: Which of the dependency can form redundancy in R

R(A B C D E F) (Prime attr, Not prime attr)

cand key → AB, BC

$\{AB \rightarrow C, B \rightarrow D, D \rightarrow E, AE \rightarrow F, C \rightarrow A\}$

- $AB \rightarrow C$

  $(AB)^+ = ABCDEF$

  ∴ AB is a super key. ✓

- $B \rightarrow D$

  $B^+ = BDE$

  B is not a super key. Redundant

- $D \rightarrow E$

  $D^+ = DE$

  not a super key Redundant

- $AE \rightarrow F$

  $(AE)^+ = AEF$ not a super key Redundant

- $C \rightarrow A$

$C^+ = C.A$

Not a superkey

Redundancy

2nd Method

R(A B C D E F)

Prime attr — Not prime attr.

candidate key → {AB, BC}

{ $AB \rightarrow C$, $B \rightarrow D$, $D \rightarrow E$, $AE \rightarrow F$, $C \rightarrow A$ }

$AB \rightarrow C$: Prime → Prime (superkey)

$B \rightarrow D$: Prime → not prime (Not SK) ①

$D \rightarrow E$: Not prime → Not prime (Not SK) ②

$AE \rightarrow F$: Prime, Not prime → Not prime (Not SK) ③

$C \rightarrow A$: Proper subset of candidate key (BC) ∴ Not SK ④

Forms Redundancy

Note — X is not a superkey of relation R if — $X \rightarrow Y$

① X is proper subset of candidate key.

(or)

② X is non prime attribute

(or)

③ X is proper subset of candidate key combines with non prime attributes.

⇒

Non trivial FD $X \rightarrow Y$ with X not super key (forms redundancy)	1 NF	2 NF	3 NF	BCNF (does not allow any non trivial FD which form redundancy)
① [Proper subset of candidate key] → [non prime attribute]	Allowed	Not allowed	Not Allowed	Not allowed
② [Non prime attribute] → [non prime attr]	Allowed	Allowed	Not allowed	Not allowed

		1 NF	2NF	3 NF	BCNF
③	[Proper subset of candidate key && Non prime attri] → [non prime attri]	Allowed	Allowed	Not allowed	Not allowed
④	[Proper subset of one candidate key] → [Proper subset of other candidate key]	Allowed	Allowed	Not allowed	Not allowed

Ques: If rel R in 3NF but not in BCNF, then which FD must be in R.

Soln. (from above table)

[Proper subset of one candidate key] ⟶ [Proper subset of other candidate key]

must exist in R.

Ques: If rel R in 3NF and NO non trivial FD such that

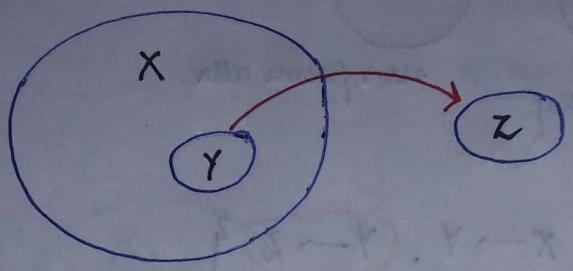
{Proper subset of Cand key} → {Proper subset of other cand key}, then rel R must also be in?

BCNF

Second Normal Form (2NF) —

Relational schema (R) is in 2NF iff No partial dependency in relation R.

Partial Dependency :



X : any candidate key
$Y \subset X$
Z : Non prime attribute

$Y \subset X$

$Y \rightarrow Z$ Partial dependency.

Third Normal Form (3NF) —

Relational schema R is in 3NF iff

Every nontrivial FD $X \rightarrow Y$ in R with.

① X must be superkey

(or)

② Y must be prime attribute.

$X \rightarrow Y$
(X: Super key, or Y: Prime attr.)

$\{ X \rightarrow Y , Y \rightarrow Z \}$
(X: super key, Z: prime attr)

(candidate key) ⟶ Non Prime ✓

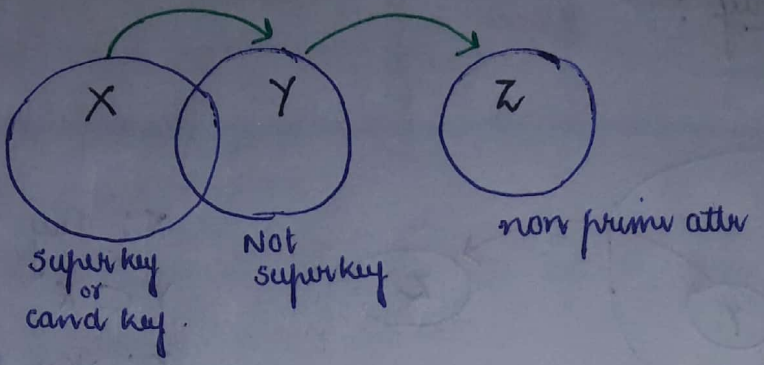
(superkey or Not superkey) ⟶ Prime ✓

} in 3NF

definition –

A relation R is in 3NF if there is no transitive dependency.

Transitive dependency:



$\{ X \rightarrow Y, Y \rightarrow Z \}$

↓

Transitive dependency

(Non prime attr (Z) transitively determined by superkey or cand key)

↓

Not Allowed in 3NF

→ If $X \rightarrow Y$, $Y \rightarrow Z$ (Non prime)

↓ super key

↓ super key

↳ If this Y is super key, then it is not transitive dependency as Z (non prime) is determined by super key.

↓

Allowed in 3NF

→ Prime attribute transitively determined by superkey is allowed in 3NF.

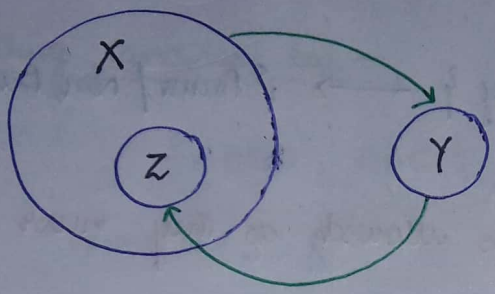
$\{ X \rightarrow Y , Y \rightarrow Z \}$

X — superkey
Y — not superkey
Z — prime

↓

Since it is prime attribute

Allowed in 3NF



X : candidate key
Y : Not superkey
Z : prime attribute

Example, $R(ABC)$ $\{ AB \rightarrow C , C \rightarrow A \}$

AB — superkey
A — prime attr.

candidate key : AB, BC

(∵ Allowed in 3NF)

But $C \rightarrow A$ forms redundancy in the relation as (proper subset of one cand key) determine (proper subset of another cand key)

↓

Soln is BCNF

Boyce Codd NF (BCNF) —

- Relational schema R is in BCNF iff every non trivial FD $X \rightarrow Y$ in R is with X as superkey.

All non trivial FD's whose determinant is a super key.

↓

Allowed in BCNF

- BCNF : { Super key } ⟶ { Prime / Non Prime }

Note - Trivial FD are always allowed as they never cause redundancy.

- { NO ~~relati~~ redundancy in R over FD's } ⟺ Rel R in BCNF

{ Redundancy exists in R over FD's } ⟺ Rel R not in ~~super~~ BCNF

Non trivial FD

$X \rightarrow Y$

↓

not super key

Ques: Find the highest NF of given relation R.

① R(ABCDE)

{ ABD → C , BC → D , CD → E }

Soln: Check for BCNF

$(ABD)^+ = ABDCE$ (super key)

$(BC)^+ = BCDE$ ✗

$(CD)^+ = CDE$ ✗

∴ Not in BCNF

Finding candidate key —

{ ABD , ABC }

Prime attr : A B C D

Non prime attr : E

Note

If there exist a trivial F.D, do not check for it, it is always allowed for all. Eg. AB → A

- Check for 3NF —

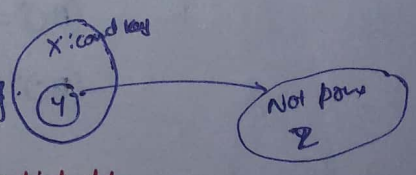
ABD → C
BC → D
] allowed as C, D are prime attri

CD → E ✗ (E: not prime)

∴ Not in 3NF

- Check for 2NF —

Here we check for partial dependency



ABD → C
BC → D
} prime attri — No partial dep

CD → E → non prime attribute — "

(CD: Not proper subset of any candidate key)

↓ All are having no partial dependency but still it could fail c/

But $BC \to D$ and $CD \to E$

$\Downarrow$

$BC \to E$ ∴ Partial dependency exist

proper subset — Not prime ∴ not in 2NF

don't check by this method

↓ use

2NF test : $\forall \left(\text{Proper subset of candidate key}\right)^+ =$ always prime

1st candidate key → ABD

$A^+ = A$
$B^+ = B$
$D^+ = D$
$(AB)^+ = AB$
$(BD)^+ = BD$
$(AD)^+ = AD$

} all prime

∴ No partial depend

2nd candidate key → ABC

$A^+ = A$
$B^+ = B$
$C^+ = C$
$(AB)^+ = AB$
$(AC)^+ = AC$

} prime

$(BC)^+ = BCDE$

↑ non prime ∴ Partial dependency exist

∴ Not in 2NF

Since All fails ∴ highest NF → 1NF Ans

② R(ABCDE)

{ AB → C , C → D , B → E }

candidate key → AB

prime attr : AB

non prime : CDE

B → E partial dependency.

(B: non prime)

$A^+ = A$

$B^+ = B, E$ (E: non prime)

∴ 2 NF fails.

⟹ highest NF = 1NF Ans

③ R(ABCD)

{ AB → C , C → A , AC → D }

- For BCNF :

$(AB)^+ = ABCD$

$C^+ = CAD$ ✗ Not BCNF

candidate key → (AB , BC)

Prime : ABC

Non prime : D.

- 3NF :

AB → C ✓ (prime)

C → A ✓ (prime)

AC → D ✗ (not prime; not candidate)

Not 3NF

- 2NF :

Candd. AB → $A^+ = A$, $B^+ = B$ ~~AB = ABCD~~

BC → $B^+ = B$, $C^+ = C, A, D$ (D: non pr)

Not 2NF

∴ 1NF Ans

④ R(ABCD)

$\{ AB \to C, BC \to D \}$

- **BCNF**

$(AB)^+ = ABCD$ ✓ super key

$(BC)^+ = BCD$ × not

✗ Not in BCNF

candidate key = AB

prime : A B

non prime : C D

- **3NF**

$AB \to C$ ✓

↑ candi ↑ not prime

$BC \to D$ ✗

↑ not cand ↑ non pr

✗ Not in 3NF

- **2NF**

candidate key AB

$A^+ = A$

$B^+ = B$

∴ 2NF

⑤ R(ABCDEF)

$\{ AB \to C, C \to D, D \to AE, DE \to F, EF \to B \}$

- **BCNF**

$(AB)^+ = ABCDEF$ ✓

$C^+ = CDAEB$ ✓

$(EF)^+ = EFB$ ✗

✗ Not in BCNF

candidate key = AB, AEf, ~~BD DF~~ D, C

prime : A B C D E F

Non prime : ✗

• 3NF

$AB \rightarrow C$ ✓

All are prime
∴ kuch bhi ho if RHS is prime ∴ 3NF

⑥ R (ABCDEF)

$\{AB \rightarrow C,\ C \rightarrow DE,\ E \rightarrow F,\ F \rightarrow B\}$

• BCNF

$(AB)^+ = \{ABCDEF\}$ ✓

$C^+ = CDEFB$ ✗ Not in BCNF

candidate key → AB, AF, AE, AC

prime : ABCEF

Non prime : D

• 3NF

$AB \rightarrow C$ prime ✓

$C \rightarrow DE$ ✗ NOT 3NF

C: Not candi; DE: Not prime

• 2NF

AB → $A^+ = A$

$B^+ = B$.

AF → $F^+ = F, B$

AE → $E^+ = E, F$

AC → $C^+ = C, D, E$ ✗ Not in 2NF

D: Not fr

∴ 1NF

(8) R(ABCD)

{ AB → C , BC → A , AC → B }

BCNF

ABC ✗ Not in BCNF

candidate key → ABD , BCD , ACD

prime → ABCD

Not prime → ✗

3NF

AB → C prime ✓ (kuch bhi ho.)

BC → A ✓

AC → B ✓

∴ 3NF

(9) R(ABCD)

{ AB → CE , BD → F , E → F }

BCNF

$(AB)^+$ = ABCE ✗ Not in BCNF

candidate key =

we remove all dependency which are not identified by relation R

AB → C ✓

AB → E ✗

BD → F ✗

E → F ✗

∴ only FD is { AB → C }

candidate key = ABD

prim = ABD
non prim = C

AB → C is partial dependency
∴ 2NF ✗

Hence 1NF

(7) R(ABCDE)
{ A → BC , CD → E , E → A , B → D }

candidate key → A, E, BC, CD

prim = ABCDE
all are prim
∴ 3NF

(19) R(ABC)
{ AB → E , DE → C }

Remove FD which are not in R.

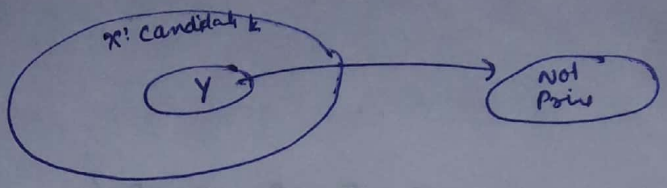
All are remove
i.e No dependency left

∴ R is in BCNF

Ques: If relation R with only simple candidate key, then which is true about relation R?

a) R in BCNF
b) R in 3NF but may not BCNF
c) R in 2NF but may or may not 3NF
d) R in 1NF but " " " " 2NF

If candidate key is simple



This condition is not possible as only simple candidate key in R.

i.e if simple candidate key → then proper subset = Null i.e ϕ only.

$\{\} \subset \{C\}$.

∴ 2NF

Eg- R(ABCD) $\{A \to B, B \to AC, C \to D\}$

candidate key → $\{A, B\}$

$C \to D$ Not prime

not candidate

X 3NF fails

But 2NF ✓

∴ 3NF may or may not but 2NF will always be True.

Eg- R(ABC) $\{A \to B, B \to AC\}$

candidate key = $\{A, B\}$

2NF ✓
3NF ✓
BCNF ✓

Ques: If every attribute of R is prime attribute then R is in 3NF but may or may not BCNF

a) 1NF
b) 2NF
c) 3NF ✓
d) 4NF

Ques: If R is in 3NF (and) at most one compound candidate key in rel R (remaining candidate keys are simple then ___

a) R also in BCNF ✓
b) R may not BCNF
c) R not BCNF
d) can not decide

R is in 3NF

Every LHS → prime
candidate key → Non prime

ABCDE

AB → C C → D D ↔ E
E → A
A → B

$(AB)^+ = ABCDE$
$C^+ = CDEAB$
$D^+ = DEABC$
$E^+ = ABCDE$

D, C

Ques: R (ABC)
with no non-trivial FD's, then highest NF of R **BCNF**

No non trivial FD's ⟶ No redundancy over FD's

↓

BCNF rel^n

Ques: Relation R with only 2 attributes, then relation R is always in **BCNF** (also in other higher NF)

R (AB)

↓

There will be 4 condition.

① R (AB) {A → B} ✓ BCNF

② R (AB) {B → A} ✓ BCNF

③ R (AB) {A → B, B → A} ✓ BCNF

④ R (AB) {No non trivial FD's} ✓ BCNF

Decomposition into higher NF —

Relation R decompose into '2NF', '3NF', 'BCNF' with

- lossless join decomposition
  and
- dependency preserving decomposition

① R(ABCDE)

{AB → C , C → D , B → E}

candidate key → AB

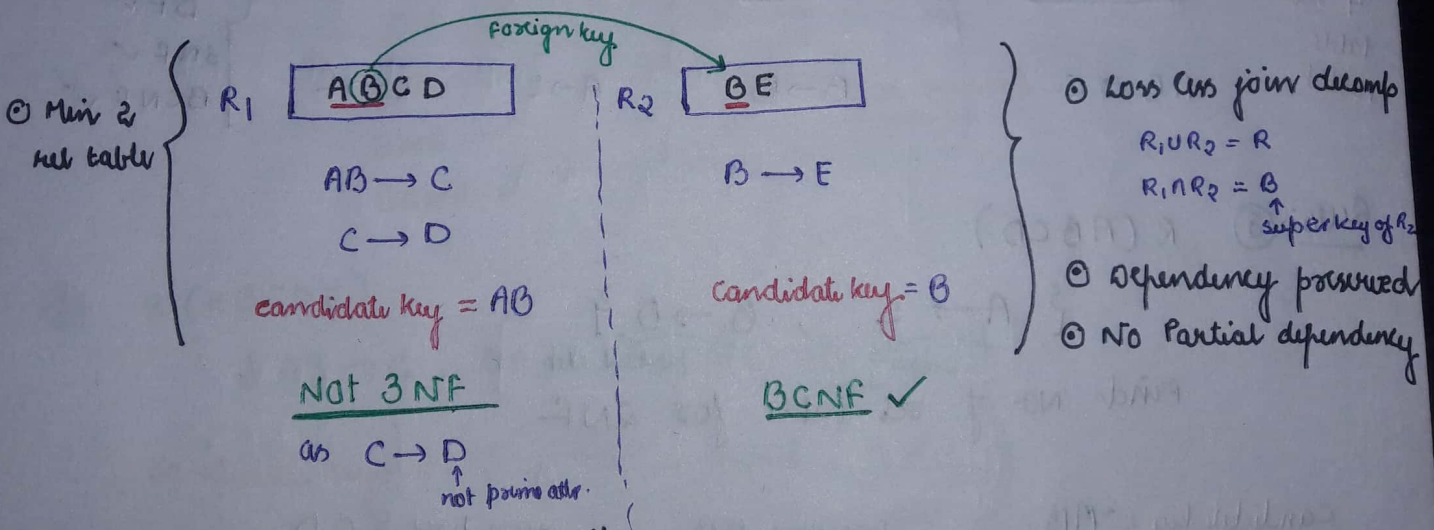
B → E : partial dependency

∴ R not in 2NF

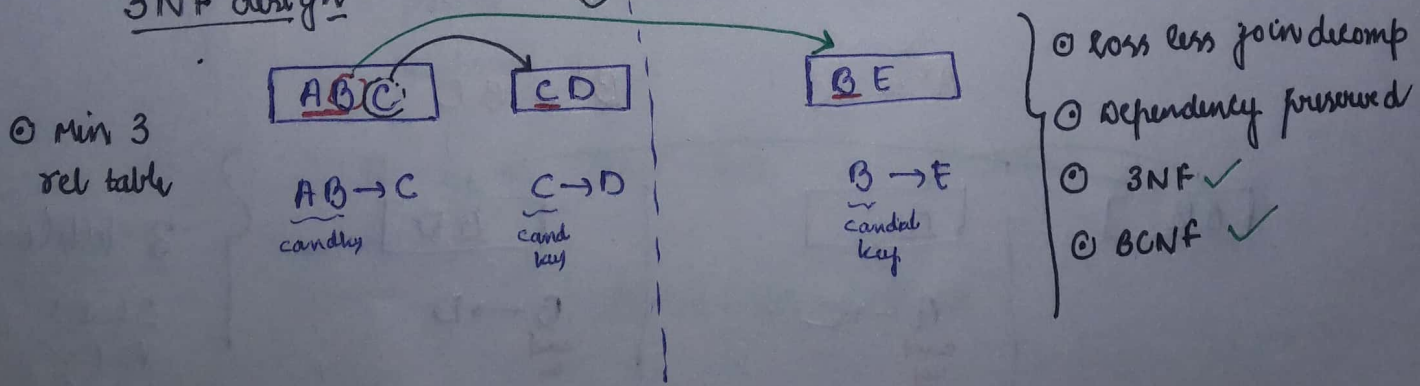
⇒ R in 1NF

2NF decomposition —

Now decompose B → E as it is partial dependency



3NF design —



② R(ABCDEF)

candidate key = AB
prime = AB non prime = CDEF

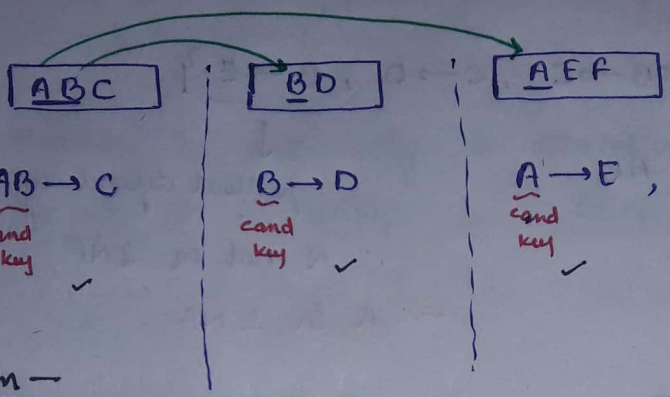
{ AB → C , B → D , A → E , E → F }

Partial dependency (B → D)

Partial dependency (A → E, A → F)

2NF design —

$B^+ = BD$ $A^+ = AEF$



3 rel tables

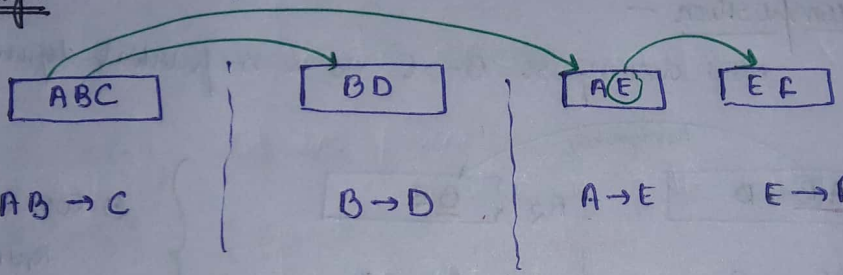
ABC: AB → C (cand key) ✓

BD: B → D (cand key) ✓

AEF: A → E (cand key) ✓ , E → F (✗ cand key) ∴ Not 3NF

Lossless dependency preserving
NO Partial dep
∴ 2NF

3NF design —



4 rel tables needed

AB → C B → D A → E E → F

LLJ ✓
DP ✓
3NF ✓
BCNF ✓

Ques: ③ R(ABCD)

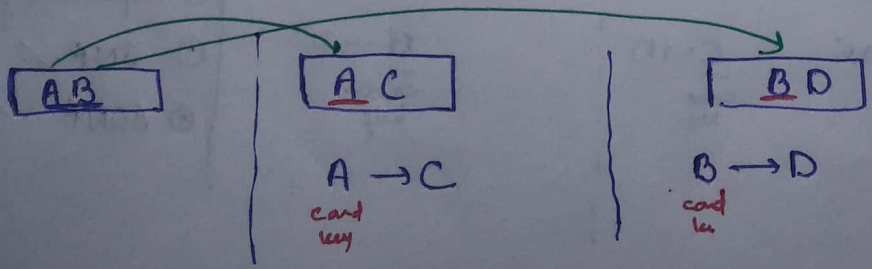
{ A → C , B → D }

partial dep partial dep

Find No. of rel table for 2NF

candidate key = AB prime = AB Non pri = CD

$A^+ = AC$ $B^+ = BD$



AC: A → C (cand key)

BD: B → D (cand key)

3 tables

LLJ ✓
DP ✓
2NF ✓
3NF ✓
BCNF ✓
(For only 2 table it is lossy)

④ R (ABCDEF)

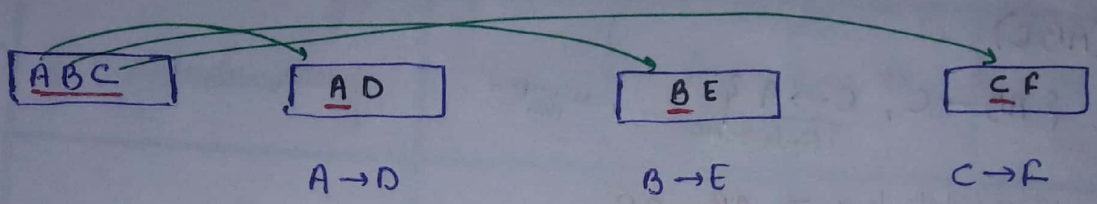
{ A → D partial dep
B → E partial "
C → F } " "

find No. of rel tables for 2NF ?

candidate key → ABC

prime = ABC not prime = DEF

$A^+ = AD$ $B^+ = BE$ $C^+ = CF$



A → D B → E C → F

Loss less join ✓
DP ✓
2NF ✓
3NF ✓
BCNF ✓

∴ 4 tables needed

⑤ R(ABCDE)

{ AB → C , BC → A , AC → B }

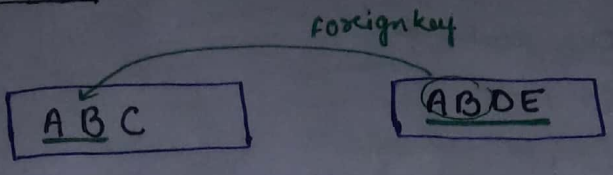
candidate keys = { ABDE , ACDE , BCDE }

prime = ABCDE
not prime = X

∴ R is in 3NF but Not BCNF
(all prime)

↓ BCNF decomposition

BCNF decomposition –



ABC

$AB \rightarrow C$

$BC \rightarrow A$

$AC \rightarrow B$

candidate key = AB, BC, AC

BCNF ✓

ABDE

No non trivial FDs

candidate key = ABDE

BCNF ✓

DP ✓
LLJ ✓
BCNF ✓

⑥ R(ABC)

$\{ AB \rightarrow C, \underbrace{C \rightarrow A}_{\text{Fails BCNF}} \}$ → ∴ decompose this.

candidate key = AB, CB

prime = ABC , non prime = X

All are prime

∴ It is in 3NF but not BCNF

↓ BCNF decomposition.

BC

~~$B \rightarrow C$~~

No non-trivial FD's

BCNF ✓

CA

$C \rightarrow A$

BCNF ✓

LLJ ✓
BCNF ✓
but
DP ✗
fails

↓

Re-design

ABC

$AB \rightarrow C$

$C \rightarrow A$

LLJ ✓

DP ✓

but Not BCNF ✗

∴ Above relation R is not possible to decompose BCNF with dependency preserving decomposition.

Most accurate NF

	DB design Goals	1NF	2NF	3NF	BCNF
①	0% redundancy	NO	NO	NO	Yes over FD's NO over MVDs
②	** Loss less Join decomposition	Yes	Yes	Yes	Yes Lossless decomp of 1NF, 2NF, 3NF, BCNF possible for every relation.
③	** Dependency Preserving decomposition	Yes	Yes	Yes	May Not Not every rel can decompose into BCNF with dependency preserving decompo

Every relation can decompose into 1NF, 2NF, 3NF, ~~BCNF~~ with dependency preserving decompo.

Queries

- Relational Algebra (mathematical formula) } Procedural Query lang
  Formulation of what data retrieved (and) how data is retrieved

---

- SQL (used to run in computer)
- Relational calculus (mathematical formula) } Non procedural Query lang
  Formulation of what data is retrieved.

Relational Algebra

→ Basic Operator :

- π → Projection } Unary operator
- σ → selection
- $\times$ → Cross product } Binary operator
- $\cup$ → Union
- $-$ → Minus (set difference)
- ρ → Rename } Unary operator

→ Derived operator –

Binary operator {
- $\cap$: Intersection (using "$-$")
- $\bowtie$: Join (using "$\pi, \sigma, \times$")
- $/$: division (using "$\pi, \times, -$")

Projection (π) –

It is used to project required list of attribute from relation R.

$$\pi_{attr\_list}(R)$$

Selection (σ) –

It is used to retrieve records which are satisfying the given predicate condition (P) from relation R.

$$\sigma_P(R)$$

Eg – R

A	B	C
4	6	8
7	5	8
5	6	8
8	3	5

$\pi_{BC}(R)$:

B	C
6	8
5	8
3	5

} distinct tuple in result

$\sigma_{A<=6}(R)$:

A	B	C
4	6	8
5	6	8

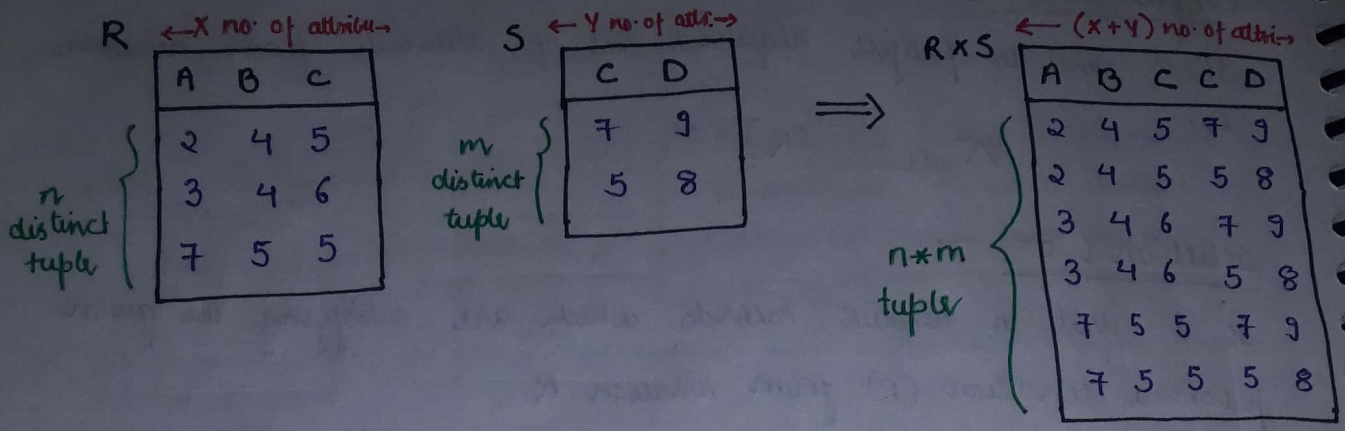
$\pi_{Bc}(\sigma_{A\leq 6}(R))$:

B	C
6	8

Cross Product (x) –

R x S results in all attribute of R followed by all attributes of S (and) each record of R pairs with every record of S.

Eg –



→ Relation R with n tuples and Relation S with 0 tuples, then no. of Tuples in R×S = 0 tuple (empty)

Eg – $A = \{1, 2, 3\}$

$B = \{\}$

$A \times B = \{\}$ Ans

Joins –

① Natural Join ($\bowtie$)

② Conditional Join ($\bowtie_c$)

③ Outer Joins
- Left outer join ($\bowtie$ or $\bowtie_c$)
- Right outer join ($\bowtie$ or $\bowtie_c$)
- full outer join ($\bowtie$ or $\bowtie_c$)

④ Equi Join ($\bowtie_=$)

conditional join with equal condition.

Natural Join –

$$R \bowtie S \equiv \pi_{\text{distinct attribute}} \left(\sigma_P (R \times S) \right)$$

P : equality of common attribute of R and S

Eg –

R

A	B	C
2	4	5
3	4	6

S

C	D
7	9
5	8

$\Rightarrow$

R×S

A	B	C	C	D
2	4	5	7	9
2	4	5	5	8
3	4	6	7	9
3	4	6	5	8

Now for σ_P i.e selection where common attribute are same.

A	B	C	C	D
2	4	5	5	8

Only 1 columns of C will be there

↓

Now π_{dist} i.e project it.

Eg – Now for above,

$$R \bowtie S \equiv \pi_{ABCD} \left(\sigma_{R.C = S.C} (R \times S) \right)$$

Eg – $T_1(\underline{A}\,\underline{B}\,\underline{C})$ $T_2(\underline{B}\,\underline{C}\,D)$

$$T_1 \bowtie T_2 \equiv \pi_{ABCD} \left(\sigma_{T_1.B = T_2.B \wedge T_1.C = T_2.C} (T_1 \times T_2) \right)$$

Eg- $T_1(A\ B)$ $T_2(C\ D)$ · Find natural join

$$T_1 \bowtie T_2 \equiv T_1 \times T_2$$

A	B
4	5
6	7

C	D
3	5
4	5

$\Rightarrow$

A	B	C	D
4	5	3	5
4	5	4	5
6	7	3	5
6	7	4	5

n×m

Natural join is equal to cross product if

Join condition is empty

Conditional Join — $(\bowtie_c)$

$$R \bowtie_c S \equiv \sigma_c (R \times S)$$

Eg-

R

A	B	C
2	4	5
3	4	6
7	5	5

S

C	D
7	9
5	8

$$R \underset{R.C<S.C}{\bowtie} S \equiv \sigma_{R.C<S.C}(R \times S)$$

$\downarrow$

A	B	C	C	D
2	4	5	7	9
3	4	6	7	9
7	5	5	7	9

Ans

Note —

$R \bowtie_{R.A<5} S \neq \sigma_{RA<5}(R \bowtie S)$

$\sigma_{R.A<5}(R \times S)$

$\sigma_{R.C=S.C \wedge R.A<5}(R \times S)$

- (Equality of common attribute is not check)
- Duplicate tuples exist

A	B	C	C	D
2	4	5	7	9
2	4	5	5	8
3	4	6	7	9
3	4	6	6	8

(Natural Join, ∴ equality of common attribute is also checked)

NO similar tuples.

A	B	C	D
2	4	5	8

R →

A	B	C
2	4	5
3	4	6
7	5	5

S →

C	D
7	9
5	8

Outer Join —

① Left Outer Join (⟕)

R ⟕ S ≡ R ⋈ S tuples
and
tuples of R, which failed the join condition.

Eg -

R

A	B	C
4	5	7
4	6	3
4	7	5

Failed join cond

=

S

C	D	E	F
7	4	3	5
7	7	5	6
6	9	8	9

R ⟕ S

A	B	C	D	E	F
4	5	7	4	3	5
4	5	7	7	5	6
4	6	3	Null	Null	Null
4	7	5	Null	Null	Null

left part

② Right Outer Join (R ⟖ S)

A	B	C	D	E	F
4	5	7	4	3	5
4	5	7	7	5	6
Null	Null	6	9	8	9

right part

③ Full Outer Join ($\overset{F}{\bowtie}$)

$R \overset{F}{\bowtie} S \equiv (R \bowtie S) \cup (R \overset{A}{\bowtie} S) \cup (R \overset{A}{\bowtie} S)$

Eg -

A	B	C	D	E	F
4	5	7	4	3	5
4	5	7	7	5	6
4	6	3	Null	Null	Null
4	7	5	Null	Null	Null
Null	Null	6	9	8	9

6/11/17

Ques: R(A---) S(B....)

Retrieve 'A' values of rel R, those are more than (some) ↓ any | atleast one 'B' values of S.

Soln:

R(A...)
2
4
6

S(B...)
3
5

⇒

RxS

A...	B...
2	3
2	5
4	3
4	5
6	3
6	5

⇒

A
4
6

$\therefore \pi_{R.A}\left(\sigma_{R.A>S.B}(R \times S)\right)$ Ans

(OR)

$\pi_{R.A}(R \underset{R.A>S.B}{\bowtie} S)$ Ans ← using conditional join

Note- $R \bowtie_c S$

$\sigma_c (R \times S)$

$R \bowtie S$

} Used to retrieve records of rel "R" which do satisfy the given specific condition with atleast one/any/some records of S

Ques: R(A...) S(B....)

Retrieve 'A' val of R those are more than every 'B' of rel (S)

every → all

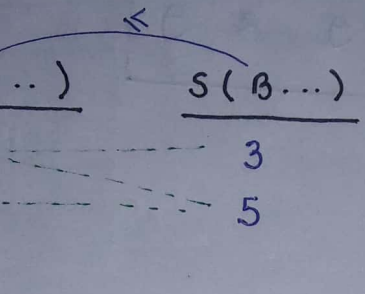
Soln

R(A--)

2

4

✓ 6



$\neg(>All) \equiv \leq$ some or any

$\Rightarrow$ (All — A value of R $\leq$ any B)

$$\pi_A(R) - \pi_A(R \underset{R.A \leq S.B}{\bowtie} S)$$ Ans

A val of R those are more than every B of S $= \left(\begin{array}{c}\text{All 'A' value}\\ \text{of relation}\\ R\end{array}\right) - \left(\begin{array}{c}\text{'A' value of rel R}\\ \text{those are less than}\\ \text{equal to some 'B' of S}\end{array}\right)$

$\Rightarrow$ ρ (Rename) -

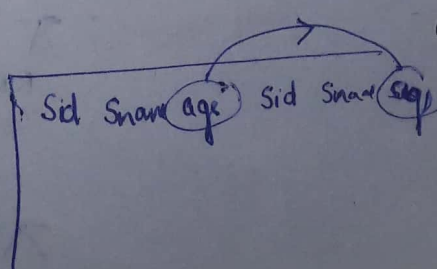
For condition where cross product of same table is done

σ (stud × stud)

stud.age > stud.age

↳ produced ambiguity

Hence we use rename operator



- Rename is used to rename table name or attribute.

Eg- There is a table named Stud (Sid Sname age)

For renaming table name –

ρ (Temp, Stud) : Stud is renamed as Temp (Sid, Sname age)

For renaming the attribute of table –

$\rho_{I,N,A}$ (Stud) : Attribute of table Stud are changed Stud (I, N, A)

For renaming some attribute of table –

$\rho_{1 \to I,\ 3 \to A}$ (Stud) : Stud (I, Sname, A)

↳ just by giving column no. we can change the name.

Ques: Stud (Sid Sname age)
Retrieve Sid's of student whose age is more than age of some student.

↓
For some we use cond. join

$\bowtie_c$

Rename this table name

Stud > Stud

Sid	Sname	age
S1		10
S2		20
S3		30
S4		40

Sid	Sname	age
S1		10
S2		20
S3		30
S4		40

$$\pi_{Stud.Sid}\left(Stud \bowtie_{Stud.age > Temp.age} \rho(Temp, Stud)\right)$$

OR

Method 2 By renaming column —

$$\pi_{Sid}\left(Stud \bowtie_{age > A} \rho_{I,N,A}(Stud)\right)$$

Ques: Retrieve Sid's of student whose age is maximum.

× 10		10
× 20	$\geq$ all →	20
× 30		30
✓ 40		40

$\geqslant$ Every = All value — < some i.e atleast one

Sid's with max age = Sid's of all Student — Sid's of student whose age is less than some student age → $\bowtie_c$ used

$$\therefore \pi_{Sid}(Stud) - \pi_{Sid}\left(Stud \bowtie_{age < A} \rho_{I,N,A}(Stud)\right)$$

Ans

Stud - Sid	age	<	Stud - Sid	age
✓ S_1	10		S_1	10
✓ S_2	20		S_2	20
✓ S_3	30		S_3	30
S_4	40		S_4	40

⇒

Sid	—	Sid
S_1		S_1
S_2		S_2
S_3		S_3
S_4		

⇒ S_4

Ques: Retrieve sid's of student whose age is minimum.

↓

$\leq$ all

✓ 10	$\leq$ all	10
× 20		20
× 30		30
× 40		40

$$\pi_{sid}(Stud) - \pi_{sid}\left(Stud \underset{age>A}{\bowtie} \rho_{I,N,A}(Stud)\right)$$

(OR)

$$\pi_{sid}(Stud) - \pi_{I}\left(Stud \underset{age<A}{\bowtie} \rho_{I,N,A}(Stud)\right)$$

Ques: Stud (sid sname age)

Retrieve sid's whose age is max for each name.

↳ i.e Sid's whose age is $\geq$ every student of same name.

Stud

	Sid	Sname	age
	S_1	A	10
✓	S_2	A	20
	S_3	B	30
✓	S_4	B	40

$\times$

Sid	Sname	age
S_1	A	10
S_2	A	20
S_3	B	30
S_4	B	40

Sid's whose age is max for each name = Sid's of all student − Sid's whose age < some student age of same name (some → cond join $\bowtie_2$)

$$= \pi_{sid}(Stud) - \pi_{sid}\left(Stud \underset{\substack{Sname=N \\ \wedge \\ age<A}}{\bowtie} \rho_{I,N,A}(Stud)\right)$$

$$\begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix} - \begin{bmatrix} S_1 \\ S_3 \end{bmatrix}$$

$$= \begin{bmatrix} S_2 \\ S_4 \end{bmatrix}$$ Ans

Ques: Stud (sid, marks, gender)

① Retrieve sid's of female student who scored more marks than some male student.

↓

atleast one

($\bowtie_C$)

Sid	Male	Gend	X	Sid	Male	Gend
S_1						
S_2						
S_3						
S_4						

$$\pi_{Sid.}\left(Stud. \bowtie_{\substack{Gender='f' \wedge \\ G='M' \\ \wedge \\ Marks > M}} \rho_{(I,M,G.)}(Stud)\right)$$

② Retrieve Sid's of female student who scored less marks than every male student.

all.

$\neg(< All) \equiv \geq Some$

$$\Rightarrow \pi_{Sid}\left(\sigma_{gender='f'}(Stud)\right) - \pi_{Sid}\left(Stud \bowtie_{\substack{Gender='f' \\ \wedge \\ G='M' \\ \wedge \\ Marks \geq M}} \rho_{(I,M,G)}(Stud)\right)$$

SET Operators –

① $\cup$: Union
② $-$: Minus (set difference) } basic operators
③ $\cap$: Intersection → derived operator.

→ To apply set operations, relations must be union compatable.

→ R and S are union compatable iff.

⊙ No. of attributes of R $\cong$ No. of attribute of S

and ⊙ Domain of each attribute of R must be same as that of S

R (A_1 A_2) S (B_1 B_2)

domain (A_1) → possible value accepted by attribute A_1

Eg- $\pi_{Sid}(\ldots\ldots)$ Set Operator ($\cup, \cap, -$) $\pi_{Sid\ Sname}(\ldots\ldots)$

✗ Not allowed

(as no. of attributes as different in one [Sid] ≠ [Sid | Sname])

Eg- $\pi_{Sid\ Sname}(\ldots)$ Set operator $\pi_{Sid\ age}(\ldots)$

✗ Not allowed

(as domain is different sname ≠ age)

Eg- $\pi_{Sid\ Sname}(\ldots)$ Set Operator $\pi_{IN}(\ldots)$

IN ↑ set of student id ; ↖ name of student

✓ Allowed

Note –

① R ∪ S, R ∩ S, R − S results are always treated as distinct tuples.

R

A
2
2
2
3
3
4

S

B
2
2
3
3
5
5

$\Longrightarrow$

R ∩ S

A
2
3

} distinct tuple

② R ∪ S, R ∩ S, R − S expressions resulted schema (i.e structure) is same schema of left side relation R.

For R set operator S

↙ name of result table = R

↘ Column's name = column name of R.

Eg –

R

A	B	C
2	4	6
3	5	7
4	6	5

S

D	E	F
3	5	7
4	6	5
2	4	8

Union –

$$R \cup S = \{ x \mid x \in R \underbrace{\vee}_{OR} x \in S \}$$

R ∪ S

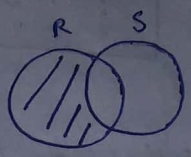
A	B	C
2	4	6
3	5	7
4	6	5
2	4	8

Minus —

$$R - S = \{ x \mid x \in R \wedge x \notin S \}$$

(But not)

(Records of R but not S) $\cong$ Records of only R



$\therefore R - S =$

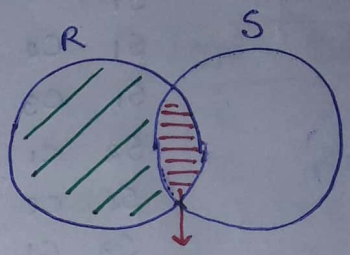
A	B	C
2	4	6

Intersection —

$$R \cap S \cong R - (R - S)$$

$$R \cap S = \{ x \mid x \in R \wedge x \in S \}$$

(AND)



$R - (R - S) \cong R \cap S$

$R \cap S =$

A	B	C
3	5	7
4	6	5

Note → If T_1 and T_2 are with same (no. and name of) attribute, then

$$T_1 \cap T_2 = T_1 \bowtie T_2$$

T_1 (A B)

A	B
2	4
6	8

T_2 (A B)

A	B
2	4
6	5

$T_1 \cap T_2 = 24$

$T_1 \bowtie T_2 = 24$ → matches A to A from T_1 and T_2 also B to B.

$T_1 \cup T_2 = T_1 \bowtie T_2$

$T_1 \cap T_2 = T_1 - (T_1 - T_2) \equiv T_1 \bowtie T_2$

Division (// or ÷)

Enroll (sid, cid)

sid	cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_2
S_3	C_1

Course (cid, ...)

cid
C_1
C_2
C_3

all courses

Retrieve Sid's enrolled for all courses.

$\pi_{sid\ cid}(Enroll) \,/\, \pi_{cid}(course) \equiv$

cid
C_1
C_2
C_3

sid	cid
✓ S_1	C_1
✓ S_1	C_2
✓ S_1	C_3
✗ S_2	C_1
✗ S_2	C_2
✗ S_3	C_1

Sid
S_1

Sid which is mapped or paired with every course

↓

represented by ÷

Expansion of "÷"

Sid's enrolled for every course

$\pi_{sid\ cid}(Enroll) \,/\, \pi_{cid}(course)$

Step 1 : Sid's of student not enrolled every course
(Sid's enrolled proper subset of all courses)

$$\pi_{Sid}\left(\pi_{Sid}(Enroll) \times \pi_{Cid}(course) - \pi_{Sid\,Cid}(Enroll)\right)$$

Every student enrolled for every course

Actual pairs enrolled

{c1}
{c2}
{c3}
{c1, c2}
{c2, c3}
{c3, c1}

if it is any of it ↓ then not enrolled for all courses

Sid	Cid
s1	c1
s1	c2
s1	c3
s2	c1
s2	c2
s2	c3
s3	c1
s3	c2
s3	c3

—

Sid	Cid
s1	c1
s1	c2
s1	c3
s2	c1
s2	c2
s3	c1

⟹

Sid	Cid
s2	c3
s3	c2
s3	c3

⟹

Sid
s2
s3

Sid's not enrolled every course

Step 2 :

[Sid's enrolled every course] = [Sid's enroll some course] − [Sid's not enrolled every course]

$$\pi_{Sid\,Cid}(Enroll) / \pi_{Cid}(course) = \pi_{Sid}(Enroll) - \pi_{Sid}\left(\pi_{Sid}(Enroll) \times \pi_{Cid}(course) - \pi_{Sid\,Cid}(Enroll)\right)$$

$$\pi_{AB}(R) / \pi_{B}(S) \equiv \pi_{A}(R) - \pi_{A}(\pi_{A}(R) \times \pi_{B}(S) - \pi_{AB}(R))$$

↓ 'A' value of rel R which are paired by every 'B' value of S.

$\pi, -, \times$ basic operators are used to express '/'

Eg – R(ABCD) / S(CD)

$\Rightarrow$ R(AB) Ans

Ques:

Stud (Sid Sname age) [all student]

Course (Cid cname instructor) [all courses]

Enroll (Sid Cid fee) [Mapping] Sid enrolled courses

① Retrieve Sid's enrolled some course taught by KORTH (instructor).

π_{Sid} (~~Stud~~ (~~Enroll~~ Enroll ⋈ Course))

~~instructor = "KORTH"~~

Enroll.Cid = Course.Cid

OR

Enroll

Sid	Cid
S_1	C_1
S_1	C_2
S_2	C_2
S_3	C_3
S_3	C_4

X Course

Cid	Instr
C_1	KORTH
C_2	KORTH
C_3	Navathe
C_4	Navathe

$$\pi_{Sid}\left(\sigma_{Enroll.cid = course.cid}\left(Enroll \times Course\right)\right)$$

Instructor = KORTH

Cost of query = 2 * (7 * 4) = 56

OR

$$\pi_{sid}\left(\sigma_{Enroll.cid = Course.Cid}\left(Enroll \times \sigma_{Instr = KORTH}(Course)\right)\right)$$

For cross product 7×2

4 compares (2 records returned)

1 compare

$$\text{cost of query} = 4 + (7 \times 2) \times 1$$
$$= 18$$

OR

$$\pi_{sid}\left(\sigma_{Inst = KORTH}\left(Enroll \bowtie Course\right)\right)$$

Natural Join

(OR)

Here also instead of joining with full courses, if we retrieve KORTH course and then Join

$$\pi_{sid}\left(Enroll \bowtie \left(\sigma_{Inst = KORTH}(Course)\right)\right)$$

(2) Retrieve Sid's enrolled by every course taught by KORTH

every → division is used

$$\pi_{sid\ Cid}(Enroll) \,/\, \pi_{Cid}\left(\sigma_{Inst = KORTH}(Course)\right)$$

(OR)

$$\pi_{Sid}(Enroll) - \pi_{Sid}\left(\pi_{Sid}(Enroll) \times \pi_{Cid}\left(\sigma_{Inst = KORTH}(Course)\right) - \pi_{(Enroll\ Sid,Cid)}\right)$$

Sid's not enrolled every KORTH course

Every sid enrolled every KORTH Course.

③ Retrieve Cid's enrolled by some student whose age more than 20)

$$\pi_{cid}\left(\sigma_{age=20}(Enroll \bowtie Stud)\right)$$

(OR) $$\pi_{cid}\left(Enroll \bowtie \sigma_{age>20}(Stud)\right)$$

④ Retrieve Cid's enrolled by every stud whose age more than 20

$$\pi_{sid\,cid}(Enroll) \;/\; \pi_{sid}\left(\sigma_{age=20}(Stud)\right)$$

Note —

Pair with every	⟹	use ÷
> every ..	⟹	All val − { ≤ some }

⑤ Retrieve ~~Sem~~ Sid's enrolled some course taught by Korth or Navathe.

$$\pi_{Sid}\left(Enroll \bowtie \sigma_{\substack{Inst=KORTH \\ \vee \\ Inst=Navathe}}(course)\right)$$

OR

$$\pi_{Sid}\left(Enroll \bowtie \sigma_{Inst=KORTH}(course)\right) \cup \pi_{SId}\left(Enroll \bowtie \sigma_{Inst=Navath}(Couse)\right)$$

⑥ Retrieve sid's enrolled some course taught by Korth and Navathe

$\pi_{sid}(Enroll \bowtie \sigma_{Inst = KORTH \wedge Inst = Navathe}(course))$ ✗ Wrong as Never true for AND cond.

(OR)

$\pi_{sid}(Enroll \bowtie \sigma_{Inst = KORTH}(course)) \cap \pi_{sid}(Enroll \bowtie \sigma_{Ins = Nav}(course))$

⑦ ~~Retrieve sid's every course taught by Navathe and Korth.~~

Enroll

Sid	Cid
S_1	C_1
S_1	C_2
S_2	C_2

Course

Cid	Inst
C_1	Korth ✗
C_2	Navathe ✗
C_3	Ullman ✗

Condition taken is
↓
isn = Korth ∧
isn = Navathe
⇓
This will never be true (as instr. can't be same Korth & Navathe together)
⇓
Result of this is empty.

⑦ Retrieve sid's where every course is taught by Korth and Navathe.

↓
÷

Cid	Inst
C_1	Korth
C_2	Korth
C_3	Navathe
C_4	Navathe

Enroll

Sid	Cid
S_1	C_1
S_1	C_2
S_2	C_3
S_2	C_4
S_3	C_1
S_3	C_2
S_3	C_3
S_3	C_4
S_4	C_1
S_4	C_3

⇒ o/p is S_3

S_3 $\pi_{Sid\ Cid}(Enroll) \div \pi_{cid}(\sigma_{Inst = Korth \vee Inst = Navathe}(course))$

C_1
C_2
C_3
C_4

OR

$$\{ \pi_{sid,cid}(Enroll) / \pi_{cid}(\sigma_{Inst=Korth}(course)) \} \cap$$

common s_3 $\Leftarrow$ $\{ \pi_{sid,cid}(Enroll) / \pi_{cid}(\sigma_{Inst=Navathe}(course)) \}$

⑧ Retrieve sid's enrolled only Korth courses.

↓ set diff

↓ It should be only student of KORTH and nobody else

Enroll

Sid	Cid
s_1	c_1
s_1	c_2
s_2	c_2
s_3	c_1
s_3	c_3
s_4	c_1

Course

Cid	Inst
c_1	Korth
c_2	Navathe
c_3	Ullman

$\Rightarrow$ O/p will be s_4

{ Sid's enrolled some course } — (set diffrn) { Sid's enrolled some (atleast 1) non KORTH course }

(— behave as but not)

$s_4 \Leftarrow$ s_1, s_2, s_3, s_4 — s_1, s_2, s_3

$$\pi_{Sid}(Enroll) - \pi_{Sid}(Enroll \bowtie \sigma_{Inst \neq Korth}(course))$$

⑨ Sid's whose age only 20

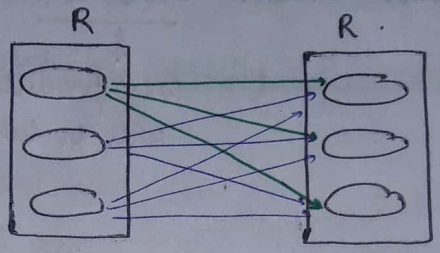
↳ does not relate to set diff

$$\pi_{Sid}(\sigma_{age=20}(stud))$$

Self Cross Product —

$\sigma_c(R \times R)$: To compare every possible 2 records of R.

$\sigma_c(R \times R \times R)$: To compare every possible 3 records of R.



Ques: ① Retrieve Sid's who enrolled atleast 2 courses.

Enroll

	Sid	Cid	fee
t_1	S_1	C_1	
t_2	S_1	C_2	
	S_1	C_3	
	S_3	C_1	
	S_3	C_2	
×	S_2	C_1	

≠

Sid	Cid	fee
S_1	C_1	
S_1	C_2	
S_1	C_3	
S_3	C_1	
S_3	C_2	
S_2	C_3	

Here we need to compare every possible 2-2 records

↓

For comparing using cross product.

$$\left(t_1 \cdot Sid = t_2 \cdot Sid \;\wedge\; t_1 \cdot Cid \neq t_2 \cdot Cid \right) \Longrightarrow$$

ensures that $t_1 \cdot Sid$ enrolled atleast 2 courses

$$\pi_{Enroll.Sid}\left(\sigma_{\substack{Enroll.Sid = T.Sid \\ \wedge\ Enroll.Cid \neq T.Cid}}\left(Enroll \times \rho(T, Enroll)\right)\right)$$

(OR)

$$\pi_{Sid}\left(Enroll \underset{\substack{Sid=S \\ \wedge\, Cid=C}}{\bowtie} \rho_{S,C,f}(Enroll)\right)$$

② Retrieve Sid's enrolled atleast three courses.

↓

cross product of 3 instances will be taken.

Firstly Rename it,

$\rho(T_1, Enroll)$

$\rho(T_2, Enroll)$

$\rho(T_3, Enroll)$

$$\sigma\ (T_1 \times T_2 \times T_3)$$

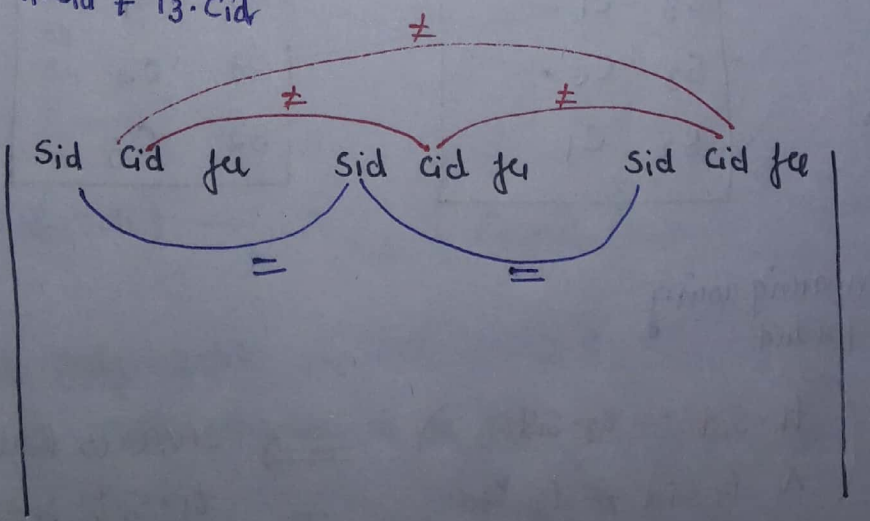
$T_1.Sid = T_2.Sid \wedge$

$T_2.Sid = T_3.Sid \wedge$

$T_1.Cid \neq T_2.Cid \wedge T_2.Cid \neq T_3.Cid \wedge$

$T_1.Cid \neq T_3.Cid$

Taken because equality does not follow equality rule



③ Retrieve Sid's enrolled only one course.

↓

(Sid's enrolled atleast one course but not enrolled atleast 2 course)

⇒ All enrolled — Enrolled in atleast 2 courses

$$\pi_{sid}(Enroll) - \pi_{sid}\left(Enroll \underset{\substack{sid=S \\ \wedge\, cid \neq C}}{\bowtie} \rho_{S,C,f}(Enroll)\right)$$

④ Retrieve Sid's enrolled exactly 2 courses

(Sid's enrolled atleast 2 courses) — (Sid's enrolled atleast 3 courses)

⑤ Sid's enrolled atmost one course.
(zero or 1)

⇒ Sid of all student — enrolled in atleast 2 courses.

$$\pi_{sid}(Stud) - \pi_{sid}\left(Enroll \underset{\substack{sid=S \\ \wedge\, cid \neq C}}{\bowtie} \rho_{S,C,f}(Enroll)\right)$$

⑥ Sid's enrolled atmost 2 courses

↓

enrolled in 0, 1 or 2 courses

All student Sid's — Sid's enrolled atleast 3 courses

$\pi_{Sid}(Stud) - \pi_{Sid}($ $)$

atleast 3

⑦ Retrieve Sid's who paid max fees.

Enroll

Sid	Cid	fees
S_1		80
S_1		70
S_1		50
S_3		50
S_3		60
S_2		80

Enroll

Sid	Cid	fees
S_1		80
S_1		70
S_1		50
S_3		50
S_3		60
S_2		80

$\pi_{Sid}(Enroll) - \pi_{Sid}(Enroll \bowtie_{Fee<F} \rho_{S,C,F}(Enroll))$

S_1, S_2, S_3 − S_1, S_3

$S_2 \Leftarrow$

↑

Error

(as S_1 also paid max fees)

o/p should be S_1 and S_2

(This is because Sid is not unique) $\Longrightarrow$ So we will project Sid Cid and these will be subtracted from all Sid Cid record.

$$\pi_{Sid}\left(\pi_{Sid\,Cid}(Enroll) - \pi_{Sid\,Cid}\left(Enroll \underset{Fee<F}{\bowtie} \rho_{S,C,F}(Enroll)\right)\right)$$

$$\pi_{Sid}\left\{\begin{matrix} S_1 & C_1 \\ S_2 & C_1 \end{matrix}\right\} \Leftarrow \left\{\begin{matrix} S_1 & C_1 \\ S_1 & C_2 \\ S_1 & C_3 \\ S_3 & C_1 \\ S_3 & C_2 \\ S_2 & C_1 \end{matrix}\right\} - \left\{\begin{matrix} S_1 & C_2 \\ S_1 & C_3 \\ S_3 & C_1 \\ S_3 & C_2 \end{matrix}\right\}$$

$\Downarrow$

S_1
S_2 (required ans)

⑧ Retrieve Sid's who paid maximum fee for each course.

	Sid	Cid	Fee
	S_1	C_1	50
✓	S_2	C_1	80
	S_3	C_1	70
✓	S_4	C_2	70
	S_5	C_2	60
	S_2	C_2	50

Sid's paid fee $\geq$ every stud for same course $\Rightarrow$ All Stud − Sid who paid less fees for same course

$$\pi_{Sid}\left(\pi_{Sid\,Cid}(Enroll) - \pi_{Sid\,Cid}\left(Enroll \underset{Fee<F \,\wedge\, Cid=C}{\bowtie} \rho_{S,C,F}(Enroll)\right)\right)$$

Ques: Stud (Sid Sname) with 200 tuples and Enroll (Sid Cid with 100 tuples. How many max and min records in result of stud ⋈ Enroll ?

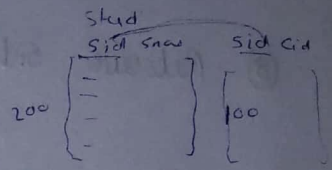
a) (200, 100) b) (200, 0)

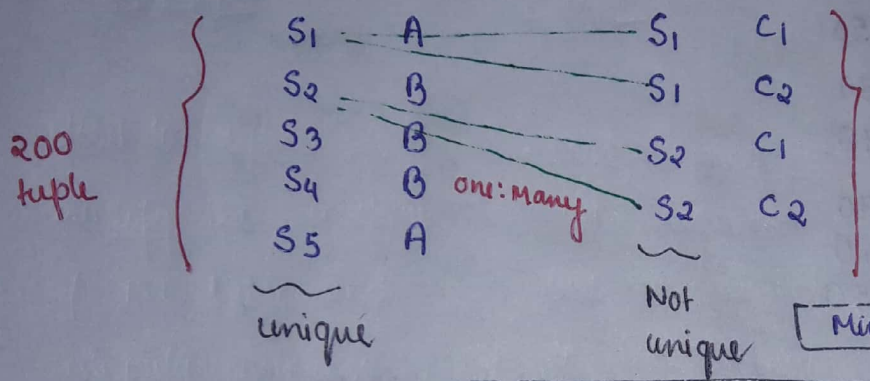
c) (100, 100) d) (20,000, 0)

=

Stud (Sid Sname) ⋈ Enroll (Sid Cid)

Primary key
∴ it is unique





as foreign key reference exist

Min tuple of stud ⋈ enroll = 100 tuple

if Foreign key does not exist ⇒ 0 (from 0 to 100)

Max tuple of stud ⋈ enroll = 100 tuples

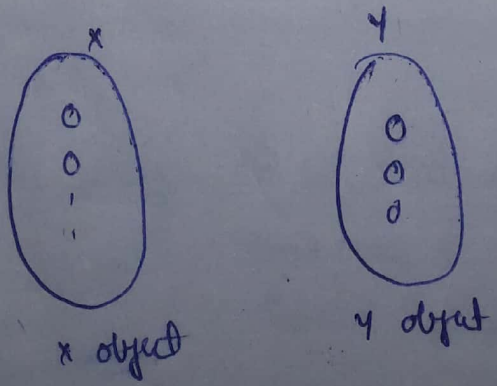
Note → In one to many mapping b/w A and B sets

① Max no. of pair at any-time = Y pairs

② M:1 , Max no. of pair = X pairs

③ 1:1 , Max no. of pair = min (X, Y) pairs

④ M:N Max no. of pair = X · Y pairs



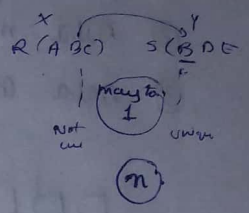
Ques: R(A B C) with n tuples and
S(B D E) with m tuples. What is

Max tuples in R ⋈ S : n tuples

many:1

Min tuples in R ⋈ S : { n tuples with foreign key
0 tuples with no foreign key }

(Assume there is no NULL value)



Ques: R(A B C) with n tuple and
S(A D E) with m tuple.

1:1

Max tuple in R ⋈ S : min (n, m) tuple

Min tuple in R ⋈ S : { min (n, m) tuple with FK
0 without FK

Ques: R(A B C) with n tuples
S(D B E) with m tuples.

many:many

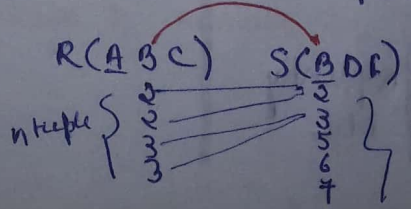
Max tuple in R ⋈ S : n × m tuples.

Min tuple in R ⋈ S : 0 (because NO 'B' is candidate key)

Ques: R(A B C) S(B D F) with n and m tuples and
{A → B, B → C, B → C, B → D, D → F} are FD's.

we could find primary key
A⁺ = ABCDF
∴ A as primary key and B for 2nd

Max tuple in R ⋈ S : n-tuples



SQL (Structured Query Language)

Sublanguages of SQL –

1. Data definition language (DDL)
2. Data manipulation lang (DML)
3. Data control lang (DCL)

DDL

- Used to define or modify the structure of DB table and integrity constraints (Primary key, alternative key, FK, check, etc)
- DDL commands –
  - → CREATE TABLE <tname>...
  - → DROP TABLE <tname>...
  - → ALTER TABLE <tname>
    - Add/remove attribute
    - Modify integrity constraint
  - → CREATE INDEX..., etc.
- Not modifies frequently and control of DDL under DBA

DML

- Used to modify data records but not allowed to modify structure of tables (and) Used to access data from DB table
- DML Commands –
  - INSERT INTO <tname>
  - DELETE FROM <tname>
  - UPDATE <tname> SET...
  - SELECT * from table where Condition;

DCL

- Data control for consistency
  - → Lock()
  - → Shared()
  - → Exclusive lock()
  - → Commit
  - → Rollback

  (Used for transaction and concurrency control)
- Data control for security
  - → grant
  - → Revoke

SQL query (VS) R.A expression —

① SQL :

selection operator (π)

③ SELECT DISTINCT $A_1, A_2, \ldots A_n$

Cross Product (X) ← ① FROM $R_1, R_2, \ldots R_n$

Selection operator (σ_P) ← ② WHERE P ;

Execution order is ①, ②, ③

↓ In R.A form

$$\pi_{A_1, A_2, \ldots A_n} \left(\sigma_P \left(R_1 \times R_2 \times \ldots R_n \right) \right)$$

Example,

① Retrieve Sid's enrolled some course taught by KORTH

In R.A →

$$\pi_{Sid} \left(\sigma_{\substack{Enroll.id = Course.id \\ \wedge\ Inst = KORTH}} \left(Enroll \times Course \right) \right)$$

In SQL →

```
SELECT DISTINCT Sid
FROM Enroll AS T1, Course AS T2
                ↑ used to rename
WHERE T1.Cid = T2.Cid and
      T2.Inst = "KORTH";
```

Note - DISTINCT keyword is used to remove the duplicate records.

Basic SQL clauses —

⑤ SELECT ⑥ [DISTINCT] $A_1, A_2, \ldots A_n$

① FROM $R_1, R_2, R_3, \ldots R_m$

② [WHERE P]

③ [GROUP BY (Attribute)

④ [HAVING condition]]

⑦ [ORDER BY (Attributes) [DESC]] ;

* By default ORDER By is Ascending ord

* Bracket denote optional part

* Execution Flow —

① → ② → ③ → ④ → ⑤ → ⑥ → ⑦

i.e

FROM : Cross Product (×)

↓

WHERE : Selection operator (σ)

↓

GROUP BY

↓

HAVING

↓

SELECT

↓

DISTINCT } : Projection operator (π)

↓

ORDER BY

SELECT of SQL ↓ π

Selection operator of RA ↓ σ_P

Aggregate functions of SQL —

- COUNT ( )
- SUM ( )
- AVG ( )
- MIN ( )
- MAX ( )

} 'NULL' value is not considered for aggregation.

↓

discard NULL value

→ Aggregate function computes aggregation of 'NON NULL' values.

Eg — Create table R
(A int
B int
);

R

Row_id	A	B
1	6	8
2	8	7
3	NULL	10
4	8	Null
5	5	5
6	Null	Null
7	5	6

- One extra attribute is defined by.
- It is defined by default for every RDBMS table.
- User cannot modify or manipulate it.
- It is unique it default Primary key.

Ques: Write SQL query to delete duplicate record ?
FAQ Interview

Count —

count (*) : Counts no. of record of table

count (A) : Counts no. of NON NULL value of attribute A

COUNT (DISTINCT A) : Counts no. of distinct NON Null values of attr 'A'.

Eg –

```
SELECT COUNT(*) AS A1 , COUNT(A) AS A2 ,
       COUNT (DISTINCT A) AS A3
FROM R;
```

↓ output

For table →

Rowid	A	B
1	6	8
2	8	7
3	Null	10
4	8	Null
5	5	5
6	Null	Null
7	5	6

⇒

A1	A2	A3
7	5	3

Ques: SELECT Sum (A) AS A1 , SUM (DISTINCT A) A2 ,
AVG (A) AS A3 , AVG (DISTINCT A) AS A4 ,
MIN (A) AS A5 , MAX (A) AS A6.
FROM R;

Solⁿ.

↓ o/p

A1	A2	A3	A4	A5	A6
32	19	6.4	6.33	5	8

↓ 32/5 — Sum/count(A)

↓ 19/3 — dist(Sum)/count(A)

Ques: Select Avg(A), B
From R;

Soln:

Avg(A)	B
6.4	□
	□
	□

Single value → 6.4

Many values → wrong ✗ → gives Error

∴ If aggregate func^n is used in SELECT class, then any other attribute is not allowed to SELECT ~~by~~ if GROUP BY clause does not exist

Ques:
```
SELECT A
FROM R
WHERE A = MAX(A);
```

defined for condition for each record

∴ does not give 8 as answer

It computes as —

A = MAX(A)
6 = Max(6)
8 = Max(8)
8 = Max(8)
⋮

Aggregation is computed for each record
↓
∴ No use of using it
(but does not give error)

∴ O/p →

6
8
8
5
5

Eg -

```
Select sid
from stud
where age = max(age);
```

✗ Avoid using Aggregate in 'where clause'

↓

does not gives error but computes each record

∴ returns sid of each record.

7/11/17

GROUP BY clause -

Used to group data records based on specified attribute values.

R

A	B	C
a_1	b_1	80
Null	b_2	70
a_3	b_3	50
Null	b_2	90
a_1	b_1	60
a_3	b_3	60
a_1	b_5	Null

Group by A →

R

A	B	C
a_1	b_1	80
a_1	b_1	60
a_1	b_5	Null
Null	b_2	70
Null	b_2	90
a_3	b_3	50
a_3	b_3	60

Group by (AB)
(more than 1 attr is used)

↓

Used for OLAP application (Data Mining Queries)

R

A	B	C
a_1	b_1	80
a_1	b_1	60
a_1	b_5	Null
Null	b_2	70
Null	b_2	90
a_3	b_3	50
a_3	b_3	60

SQL Standard for group by -

Every attribute of group by must be in select clause and is allowed to use aggregation func^{n} in SELECT clause.

↓

(Aggregation of SELECT clause computes aggregation of each group)

But it is not allowed to select attributes which are not in group.

Eg - 3. Select A, Avg(C)
1. From R
2. Group by (A);

✓ Aggregation allowed as group by (A) → So A can be selected

Soln.

A	Avg (C)
a_1	70
NULL	80
a_3	55

- for last table

Ques:

Select A
From R
Group by (A);

A
a_1
Null
a_3

Ques: Select B
From R
Group by A; Error
(Not allowed)

↑ It is group by A
but is selecting B
∴ Error.

Ques: Select A, B
From R
Group by A;

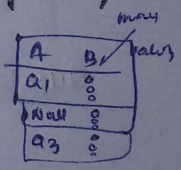
B is selected
and
it is group by A

∴ Not allowed
Error.

Ques: Select Avg(C)
From R
Group by A;

In SQL standard –
It is not allowed as group by attribute i.e 'A' is not in select clause. ∴ Error

(data can't be placed in tabular format)



But, in Oracle SQL –
It is allowed.

Avg (c)
70
80
55

But we don't know which is avg of which groups.

same functionality when implemented in SQL standard can be written as —

```
Select C1
From ( select A, Avg(c) as C1
       From R
       Group by A)
```

↓

Inner query o/p is

A	Avg(c)
a_1	~~30~~
a_1	~~70~~
a_1	~~70~~
Null	80
Null	80
a_3	55
a_3	55

after outer query →

C1
70
80
55

Ques: select A
From R
Group by (A, B)

Error as Group by attribute B is not in the select

In ~~SQL~~ Oracle SQL —

Allowed ✓

A
a_1
a_1
Null
a_3

We cannot say which value is group by A and which is group by B.

Ques: which is not allowed as per SQL standard.

a) Set A, Avg(B) ✗
From R;

b) Set Avg(C)
From R ✗
Group by A;

c) Select A,
From R ✗
Group by (A,B);

option a), b), c) all are wrong, but
↓
(b) and (c) are accepted by Oracle SQL i.e some compiler but not a)

d) Select A, count(*). ✓
From R
Group by A

Ques: which is not allowed.

a) Select A
From R
Group by (A,B);
↑ group by B
but B not projected

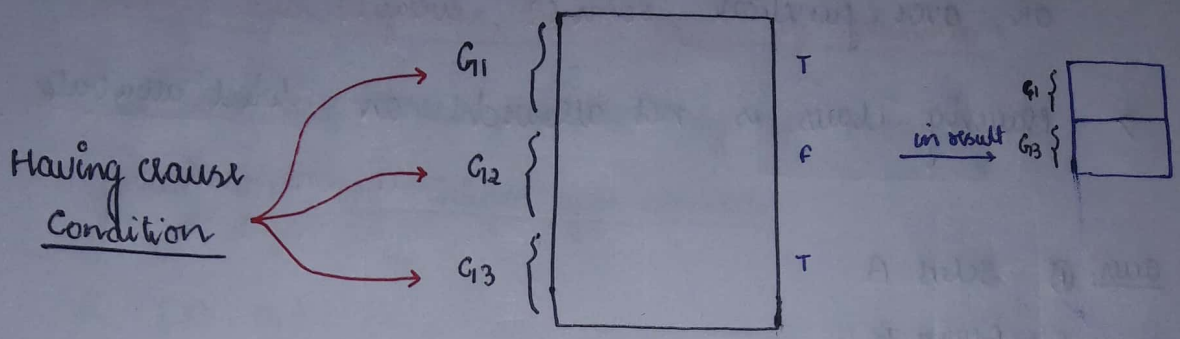
b) Select A
From R
Group by A; ✓

c) Select A, count(*)
From R
Group by A; ✓

d) Select A,B
From R
Group by (A,B) ✓

Having Clause —

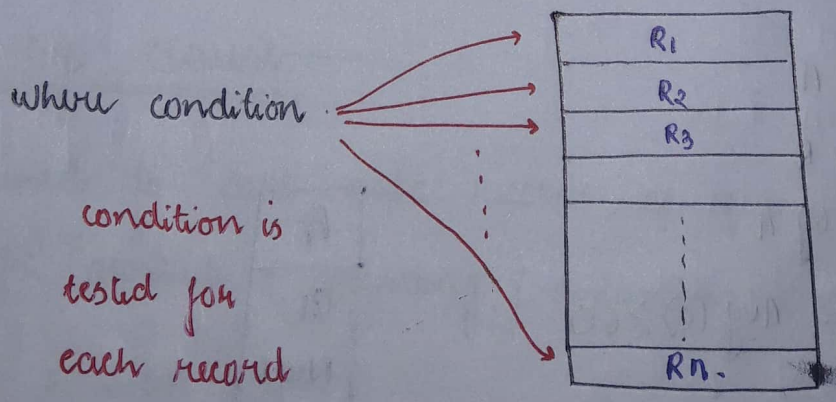
- It is used to select group based on some specified condition.
- Having clause condition is tested for each group.



condition is tested for each group

(If any group fails the condition, then that group is not in result)

In 'where' clause



condition is tested for each record

SQL Standard for Having Clause —

→ Having clause is allowed to use only if Group by clause exist.

→ Condition of having clause must be over aggregation or over function some(), every()

→ Having claus is not allowed on direct attribute

Ques: ① Select A
From R
Group by A
having C > 70 ;

↑ direct attribute

∴ Error

C > 70 cannot be applied as condition of group.

② Select A
1. From R
2. Group by A
3. having Avg(C) > 60

↓ aggregation is allowed

A
a1
Null

③ Select A
1. From R
2. Group by A
3. having sum(C) ≥ 70;

Allowed

A
a1
Null

Note — Having clause can use direct group by attribute

It can use direct attribute only if Group by is based on one integer type attribute

R (A B)

{ 2, 2 }
{ 4, 4, 4 } ✓
{ 8, 8 } ✓

Select A
From R
Group by A
having A > 3

o/p →

A
4
8

Allowed

Order by clause —

It is used to sort data records of query result, based on specific attribute (ascending / descending order)

Eg —
5. Select X, Y, Z
1. From ——
2. Where ——
3. Group by ——
4. having ——
6. Order by (X, Y);

↳ ordering can be done base on 1 or more attribute

X	Y	Z
2	6	10
2	8	8
3	5	0
3	4	1
Null	6	4
Null	5	3

o/p for → order by (X, Y)

X	Y	Z
2	6	10
2	8	8
3	4	1
3	5	0
Null	5	3
Null	6	4

NULL values will always be at the last

Nested Query —

① Nested Query (without Co-relation)

② Co-related Nested Query.

→ **Nested Query :**

- Select Attr
  From Tables
  Where condition
  Group by Attri
  Having Condition
  Order by attri

(Inner Query)

← Inner query result treated as table/cond for outer query

- In nested query without co-relation,
  - → Inner query is independent (∵ executed 1st)
  - → Outer query uses the result of inner query

Eg -

```
Select T1.Sid
From Stud T1
where marks = ( Select max(marks)
                From Stud T2 );
```

result of it is 80 (scalar value)

① Inner query / Bottom

② Outer query / Top

- ⊙ Execution flow is from Bottom to Top.
- ⊙ Inner query executes only once.

→ *** Co-related Nested Queries:

- Inner query is not independent.
- Inner query uses the attributes defined in outer query.
- Outer query uses the result of inner query.
- WHERE clause and HAVING clause of outer query is used to correlate with inner query.

Eg- Select A

① FROM R

③ WHERE (select count(*)

② From S

where (R.A < S.B) ≤ 2

10 20 30 40 50

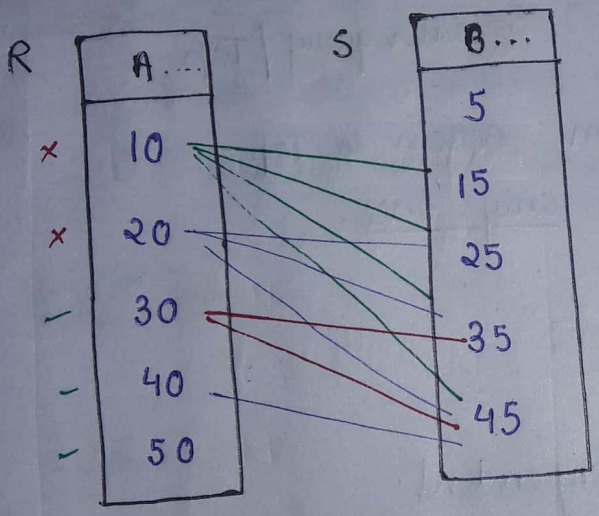
Inner query

To compute inner query attr. of outer query is required

(It's result is compared with 2)

↓

if its less, then outer Where clause is True else False



Result of inner query	where cond ◯ ≤ 2
4	F
3	F
2	T
1	T
0	T

⇓

o/p will be

A
30
40
50

No. of Inner query execution = No. of record in outer query

= 5

- If co-relation is present in where clause, then for each record of outer query, from clause computes inner query.
- Execution flow is (Top – Bottom – Top) in loop.

Eg- Select A
① From R
② Group by A
③ having Avg(C) > (Select Avg(C)
From S
where S.A = R.A or a3);
↳ group name of relation R

Soln:

R

	A	C
X	a_1	40
X	a_1	70
✓	a_3	80
✓	a_3	90

S

A	C
a_1	80
a_1	70
a_3	60
a_3	50

- If co-relation is in having clause, then for each group of outer query, inner query recomputes.

No. of inner query execution	=	No. of groups in outer query

Eg-

```
Select Sid
From Stud
where marks = ( Select Marks
                From Stud
                where branch = CS );
```

Inner query

↓

The result of this inner query is set of marks i.e not a single scalar value

∴ Direct comparison not allowed

"Error"

Stud

Sid	branch	marks
S_1	CS	70
S_2	IT	80
S_3	CS	90
S_4	IT	95
S_5	CS	85

→ Result of inner query

Marks
70
90
85

↑

Now here direct comparison is not possible

i.e ↓

S_1 CS 70 = {70, 90, 85}

set of values

To avoid this problem, SQL support "SQL func^n"

SQL func^{n} used in Nested Queries —

① IN / NOT IN
② ANY
③ ALL
} Best suited for Nested Queries with no correlation

④ EXISTS / NOT EXISTS → Best for co-related Nested Query.

These SQL func^{n} are used to compare set of values of inner query result with records of value of outer query in where / having clause

* So, for last ques, instead of '=' we can use 'IN'.

Note — Above func^{n} are not used when inner query results in single scalar value.

① IN — (Membership Test)

X IN [Set of Y value of inner query result]

↓

Returns True if x value is member of set of Y value

Eg — X IN {2, 3, 5, 6}

⇓

True if X ∈ set

X IN {Empty Set} : False
X NOT IN {Empty Set} : True

→ IN can be used in place of Equi Join i.e (works similar to Equi Join)

Eg —

① R(AB) S(CD)

$\pi_{A,B}(R \underset{R.B=S.C}{\bowtie} S) \equiv$

```
Select *
From R
where B IN ( select C
             From S )
```

Works same as equi Join

(Equal to atleast one value of C or some value of any.)

② $T_1(ABC)$ $T_2(DEF)$

$\pi_{ABC}(T_1 \underset{T_1.B=T_2.D \wedge T_1.C=T_2.E}{\bowtie} T_2) \equiv$

```
Select *
FROM T1
where (B,C) IN ( select D,E
                 From T2 )
```

comparison b/w equal no. of columns must be there b/w inner and outer query.

③ R(AB) S(CD)

$\pi_{A,B}(R \underset{R.B>S.C}{\bowtie} S)$

Not possible to express using IN func^n

② ANY and ALL

→ X operator ANY [set Y values]

operator: <, >, <=, >=, =, <>

TRUE if X satisfies the given comparison i.e operation with some (↓ atleast 1) value of set Y

<>: Not Equal

X op ANY {Empty set} : False

→ X operator ALL [Set of Y values]

True if X satisfies the given comparison (operator) with every value of set Y.

X op ALL {Empty set} : True.

Eg- Select *
FROM R
WHERE A IN (Select B
From S)

Here = ANY can be used.

R	A
x	10
✓	30
✓	40

S	B
	20
	30
	40

⟹

A
30
40

Here, IN can be replaced by (= ANY)

IN ≡ (= ANY)

Rewriting it using = ANY

```
Select *
FROM R
where A = ANY ( select B
                From S )
```

- But, 2 field comparison of IN cannot be replaced by = ANY i.e IN support more than one field comparison.

```
Eg -  select *
      FROM T1
      where (B, C) IN ( select D, E
                        from T2 )
```

cannot be done by = ANY

- Also, = ANY support many more operations like <, >, <=, >=, etc that are not supported by IN.

- < > ALL can be replaced by NOT IN func^n

Ques:

R

A	B
4	2
6	4
7	5

S

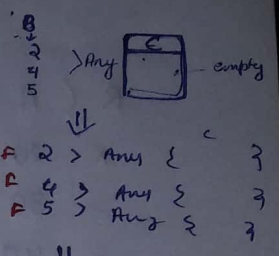
C	D
4	3
7	6
9	8

What is the no. of tuples in the result of SQL query?

① Select *
From R
Where B > ANY (Select C
From S
Where D > 10);

a) 0 c) 2

b) 1 d) 3



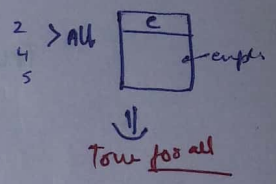
ANY = ∃

② Select *
From R
Where B > ALL (Select C
From S
Where D > 10);

ALL = ∀

a) 0 c) 2

b) 1 d) 3



$\exists x (P(x))$ ⟶ True for some x (atleast 1), P(x) is true
False for every x, P(x) is false.

$\forall x (P(x))$ ⟶ True for every x, P(x) is true
False for some (atleast 1) x, P(x) is false.

Eg - X = {5, 10, 15}

$\exists x (x > 10) = T$
$\exists x (x > 20) = F$
$\forall x (x > 2) = T$
$\forall x (x > 5) = F$

Eg - X = { }

$\exists x (x > 10) =$ False
$\forall x (x > 10) =$ True

③ EXISTS

Function returns **True** if inner query result is **not empty**, otherwise returns false

R (A....)	S (B....)
✗ 20	25
✓ 30	35
✓ 40	45

```
Select A
From R
Where EXISTS ( Select *
               From S
               Where R.A > S.B )
```

Empty ∴ return false ⇒ record not in o/p

{4, 1, 2} — T, T

True: if some record of S, it satisfy R.A > S.B

F
T
T

20
30
40

o/p →

A
30
40

Ques: R

A	B
4	2
6	4
7	5

S

C	D
4	3
7	6
9	8

Find No. of Tuples in result of -

①
```
Select *
From R
Where Exists ( Select C
               From S
               Where D>10 and R.B<S.C );
```

{ } empty → ∴ false

∴ 0 Tuples

② Select *
From R
Where EXISTS (Select count (*)
From S
Where D > 10 and R.B < S.C);

returns 0 i.e {0}

Since it has a value
∴ Not empty
⇒ True
(true for all)

∴ 3 records in the o/p.

Ques: R(A...) S(B...)

Retrieve 'A' values of R which are more than some 'B' of Relation S.

In R.A —

$$\pi_A (R \underset{R.A > S.B}{\bowtie} S)$$

In SQL —

Select distinct A
From R, S
Where R.A > S.B

(OR)

We cant write it using ANY.

Select distinct A

Ques: Emp (Eid, dno, sal, gender)

① Retrieve Eid's who get highest salary

```
Select Eid
From Emp
                                  ] All Eid's
-
~~where~~ EXCEPT
Select T1.eid
From Emp T1, Emp T2
Where T1.sal < T2.sal             ] Eid's who do not get highest salary
```

EXCEPT keyword is used for SET difference

In R.A —

$$\pi_{Eid}(Emp) - \pi_{Eid}(Emp \underset{Sal<S}{\bowtie} \rho_{I,D,S,G}(EMP))$$

(OR)

```
Select Eid
From Emp
where sal = ( Select Max (sal)
              From Emp );
```

② Retrieve Eid's who get 2nd highest salary.

In R.A —

$$\rho\left(Temp, \pi_{Eid,Sal}\left(Emp \underset{Sal<S}{\bowtie} \rho_{I,D,S,G}(Emp)\right)\right)$$

↑ This gives Eid, sal of all employee except who gets highest salary.

$$\pi_{Eid}(Temp) - \pi_{Eid}\left(Temp \underset{Sal<S}{\bowtie} \rho_{I,S}(Temp)\right)$$

In SQL —

(Join Query)

'Temp' used in R.A needed to be computed 3 times, for this SQL supports WITH clause.

```
WITH Temp ( Eid , Sal) as
        (Select Distinct T1.Eid , T1.Sal
         From Emp T1 , Emp T2
         Where T1.Sal < T2.Sal )
SELECT Eid
From Temp
EXCEPT
Select T1.Eid
From Temp T1 , Temp T2
Where T1.Sal < T2.Sal ;
```

If with clause is not used then, this is written 3 times in place of Temp.

(OR)

Using Aggregation (Nested Query)

```
Select Eid
From Emp
Where Sal = ( Select max (Sal)        <-- 2nd Max
              From Emp
              Where Sal < (Select max(Sal)   <-- 1st Max
                           From Emp ))
```

Above is non-correlated query

(For this if we want Kth highest, then it become
- (K+1) levels of nested query
- K+1 instance of Emp Table is used.

OR

Emp (Eid sal)

T_1

E_1	80
E_2	80
E_3	70
E_4	70
E_5	60
E_6	50

Emp (Eid sal)

T_2

E_1	80
E_2	80
E_3	70
E_4	70
E_5	60
E_6	50

For each record of T_1 count no. of distinct record of T_2 whose salary is more than sal of T_1

↓

count (distinct) = 0

T_1 sal is 1st max

count (distinct) = 1

T_1 sal is 2nd max

⋮

```
Select T1.Eid
From Emp T1
where ( Select Count (Distinct T2.sal)
        From Emp T2
        Where T1.sal < T2.sal ) = 1
```

inner query

* For Eid's of all Emp with 1st, 2nd and 3rd max sale.

where $T_1.sal < T_2.sal \leq 2$

For kth highest

→ Only 2 level of query

→ 2 instance of Emp table

* for Eid who get Top 3 least salary.

where cond will be changed as

$T_1.sal > T_2.sal \leq 2$

* for Eid who get 3rd highest

$T_1.sal < T_2.sal = 2$

* Eid's who get 3rd last salary

where $T_1 \cdot sal > T_2 \cdot sal = 2$

Ques: Emp (Eid, dno, sal, gender)

(1) Retrieve dno which consist of atleast 3 employees.

In R.A —

$\rho (T_1, Emp)$

$\rho (T_2, Emp)$ atleast 3

$\rho (T_3, Emp)$

$\sigma (T_1 \times T_2 \times T_3)$

$T_1 \cdot dno = T_2 \cdot dno \wedge$
$T_2 \cdot dno = T_3 \cdot dno \wedge$
$T_1 \cdot Eid \neq T_2 \cdot Eid \wedge$
$T_2 \cdot Eid \neq T_3 \cdot Eid \wedge$
$T_1 \cdot Eid \neq T_3 \cdot Eid$;

In SQL —

```
select distinct T1.dno
from Emp T1, Emp T2, Emp T3
where T1.dno = T2.dno and T2.dno = T3.dno
and T1.Eid <> T2.Eid and
    T2.Eid <> T3.Eid and
    T1.Eid <> T3.Eid ;
```

Ans

(OR) By using GROUP by —

This is easier and has less computation.

Eid	dno
e_1	d_1
e_2	d_1
e_3	d_1
e_4	d_2
e_5	d_2
e_6	d_3
e_7	d_3
e_8	d_3
e_9	d_3

≥ 3 (for d_1)

≥ 3 (for d_2)

≥ 3 (for d_3)

✓ ✗ ✓

```
Select dno
From Emp
Group by dno
having count(*) ≥ 3
```

Ans

having clause can be replaced by where clause.
(∵ Having clause is not mandatory)

```
Select dno
From ( select dno, count(*) c
       From Emp
       Group by dno) temp
where c ≥ 0
```

The computation by these 2 queries i.e with having clause and where clause are same

Temp

dno	count(*)
d_1	3
d_2	2
d_3	4

② Retrieve d.no such that average sal of female emp's of department more than avg sal of all male employees of comp

```
Select dno
1. From Emp
2. Where gender = female
3. Group by dno
4. having Avg(sal) > (Select Avg(sal)
                      From Emp
                      where gender = male);
```

Avg sal of all male employees of all dept.

Avg sal of female employees of each dno.

Emp

Eid	dno	Sal	gender
e1	d1		f
e3	d1		f
e5	d2		f
e7	d2		f

o/p

d no
d1

OR By cross producting

T2

dno	Avg of Fem
d1	80
d2	50

dept wise

T2

Avg of Mal
60

```
Select T1.dno
From (select dno, Avg (sal) sal
      From Emp
      where gender = female
      Group by dno) T1,

     (Select Avg (sal) sal
      From Emp
      where gender = male ) T2
Where T1.sal > T2.sal ;
```

(3) Retrieve dno such that avg sal of female emp of each dept more than avg sal of male emp of same dept

Co-related query with having clause —

```
Select dno
From Emp T1
where T1.gender = female
```

Top ① Group by dno

Top ③ having Avg (sal) > (

```
Select Avg (sal)
From Emp T2
② where T2.gender = male and
      T2.dno = T1.dno ;
```

)

bottom

(OR)

```
Select T1.dno
From ( select dno, Avg (sal) sal
       From Emp
       where gender = Female
       Group by dno) T1 ,

     (select dno, Avg (sal) sal
      From Emp
      Group by dno
      where gender = Male ) T2
Where T1.sal > T2.sal and
      T1.dno = T2.dno ;
```

(3) Retrieve Eid's whose salary more than ~~some sabar~~ salary of some Emp of dept 5.

$\bowtie_C$

In R.A —

$$\pi_{Eid}\left(Emp \underset{Sal>S \wedge D="5"}{\bowtie} \rho_{I,D,S,G}(Emp) \right)$$

In SQL —

directly cross producting

```
SELECT (Eidistinct) T1.Eid
FROM Emp T1, Emp T2
where T1.sal > T2.sal
      and T2.dno = 5 ;
```

(OR)

OR Using ANY function — Complexity = n

```
Select Eid
From Emp
where Sal > ANY ( Select Sal
                  From Emp
                  where dno = 5 )
```

Here, we can replace

Sal > ANY (Sal of dept 5)

↓ equivalent to

Sal > (min Sal of dept 5)

OR Using Co-related query -

```
Select T1.Eid
From Emp T1
where EXISTS ( Select *
               From Emp T2
               where T2.dno = 5
               and T1.Sal > T2.Sal );
```

10 ~~20~~ 30 40 50

T1:

	Eid	dno	Sal
✗	e1	4	10
✗	e2	5	20
✓	e3	4	30
✓	e4	5	40
✓	e5	3	50

n tuples

T2:

Eid	dno	Sal
e1	4	10
e2	5	20
e3	4	30
e4	5	40
e5	3	50

T2.Sal: 10 20 30 40 50

✗	✗	✗	✗	✗
✗	✗	✓	✓	✓
✗	✗	✗	✗	✗
✗	✗	✗	✓	✓
✗	✗	✗	✗	✗
↓	↓	↓	↓	↓
F	F	exist is T	T	T

Atleast 1 record of inner query should be True

↓

So Exist is True else False

Complexity = $O(n^2)$

④ *** Retrieve Eid's whose sal more than every emp of dept 5

↓ 'ALL' can be used

In R.A —

$$\pi_{Eid}(Emp) - \pi_{Eid}(Emp \underset{Sal \leq S \wedge D=5}{\bowtie} \rho_{I,D,S,G}(Emp))$$

In SQL -

Using cross product and set difference (JOIN Query)

```
Select Eid from Emp          <- All
EXCEPT                       <- -
Select T1.Eid
From Emp T1, Emp T2
where T1.sal <= T2.Sal and
      T2.dno = 5;
```

All employee whose sal is less than atleast 1

(OR) Using ALL funcn - (Nested Query)

```
Select Eid
From Emp T1
where Sal > ALL ( Select Sal
                  From Emp
                  where dno = 5 );
```

Sal > All {Sal: 20, 40}

* Instead of 'ALL', max can also be used

OR *** Using Co-related Nested Query –

¬(T1.sal ≤ some sal of dept) ≡ T1 sal is more than every sal of dept=5

```
Select T1.Eid
From Emp T1
where NOT EXISTS ( Select *
                   From Emp T2
                   where T2.dno = 5
                         T2.sal ≥ T1.sal );
```

T1.sal ≤ some sal of dept 5

T1

Eid	dno	Sal
e1	4	10
e2	5	20
e3	4	30
e4	5	40
e5	3	50

✗	✓
✗	✓
✗	✓
✗	✓
✓	✗

exists (T1.sal ≤ T2.sal)

i.e atleast one T1.sal ≤ T2.sal

Not exist (T1.sal ≤ T2.sal)

i.e

T1.sal > every T2.sal.

Ques: R(A....) S(B....)

① Retrieve 'A' value of R which are more than some 'B' value in relation S.

In R.A —

$$\pi_A (R \underset{R.A > S.B}{\bowtie} S)$$

In SQL —

Join Query:

```
Select distinct R.A
From R, S
Where R.A > S.B;
```

(OR) Nested Query: Using ANY

```
Select R.A
From R
Where R.A > ANY (select B
                 From S);
```

(OR) Co-related Nested query.

```
Select R.A
From R
Where EXISTS (select *
              From S
              where S.B < R.A);
```

(EXISTS ↓ Some)

↳ R.A > Some value of B in relation S

Ques: R(A--) S(B--)

Retrieve 'A' value which are more than every 'B' of S
↓
ALL

In R·A →

$$\pi_A(R) - \pi_A(R \underset{R.A \leq S.B}{\bowtie} S)$$

In SQL →

Join Query:

```
select A
from R
EXCEPT
select A
 from R,S
 where R.A ≤ S.B
```

Nested Query: Using ALL

```
select A
From R
where A > ALL ( select B
                From S );
```

Co-related Nested Query:

```
Select A
from R
where NOT EXISTS ( select *
                   from S
                   where R.A ≤ S.B )
```

SQL queries equal to "/" of RA —

Ques: Stud (Sid Sname age) Course (Cid Cname Inst) Enroll (Sid Cid fee)

- Retrieve Sid's of student who enrolled every course taught by KORTH.

In R.A :

$$\pi_{Sid\,Cid}(Enroll) \,/\, \pi_{Cid}\left(\sigma_{Inst = KORTH}(Course)\right)$$

In SQL :

Co-related Query —

Enroll T_1

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1

Course

Cid	Inst
C_1	KORTH
C_2	KORTH
C_3	KORTH
C_4	Navathe

Enroll T_2

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1

Retrieve $T_1.Sid$ from Enroll (T_1) only if —

$$\left[\begin{array}{c}\text{All Cid's of}\\ \text{KORTH}\end{array}\right] - \left[\begin{array}{c}\text{Cid's of Enroll } (T_2)\\ \text{which are enrolled}\\ \text{by } T_1.Sid\end{array}\right] = \phi \quad \text{Empty}.$$

when $T_1 = S_1$ $\quad \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} - \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \phi$ ✓ in result

when $T_1 = S_2$ $\quad \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} - \begin{bmatrix} C_1 \end{bmatrix} = \begin{bmatrix} C_2 \\ C_3 \end{bmatrix}$ Not empty ✗

```
Select T1.Sid
① From Enroll T1
③ Where NOT EXISTS (  Select Cid
                      From Course
                      Where Inst = KORTH
                      EXCEPT
                    ② Select T2.Cid
                      From Enroll T2
                      Where T2.Sid = T1.Sid )
```

OR

Not Co-related Query —

$$\pi_{Sid\,Cid}(Enroll) \,/\, \pi_{Cid}\left(\sigma_{Inst = KORTH}(Course)\right)$$

Above can be expanded as:

$$\equiv \pi_{Sid}(Enroll) - \pi_{Sid}\Big(\underbrace{\pi_{Sid}(Enroll) \times \pi_{Cid}\left(\sigma_{Inst = KORTH}(Course)\right) - \pi_{Sid\,Cid}(Enroll)}_{R}\Big)$$

Now writing above equn in SQL form.

```
(Select Sid
 From Enroll
 Except
 Select Sid
```

```
From ( Select Enroll.sid , Course.Cid
       From Enroll, Course
       where Inst = KORTH
       except
       Select Sid , Cid
       From Enroll
     ) R ;
```

(2) Retrieve Cid's enrolled by every student whose age is more than 20.

All -

Retrieve T_1.Cid from Enroll only if -

$$\left[\begin{array}{l}\text{All Sid's whose}\\ \text{age more than 20}\end{array}\right] - \left[\begin{array}{l}\text{Sid of enroll } (T_2) \text{ who}\\ \text{are enrolled by } T_1.\text{Cid}\end{array}\right] = \phi$$

Now implementing above in SQL

```
Select T1.Cid
① From Enroll T1
③ where NOT EXISTS ( Select Sid
                     From Stud
                     where age > 20
                     Except
                  ② Select T2.Sid
                     ~~Enroll~~
                     From Enroll T2
                     where T2.Cid = T1.Cid );
```

Ques: Catalog (Sid Pid) parts (pid Col)

Sid	Pid
S_1	P_1
S_1	P_2
S_1	P_3
S_2	P_2

pid	Col
P_1	Red
P_2	Red
P_3	Red
P_4	Green

Retrieve Sid's ~~of~~ supplied every red part.

~~A Sid~~

In SQL,

All — (P_1, P_2, P_3) Not Red

(S_1, S_2) − (S_2)

(S_1) ✓

(All Sid's of Red) — (Sid's whose Pid's are not red).

Retrieve T_1.Sid From Catalog only if

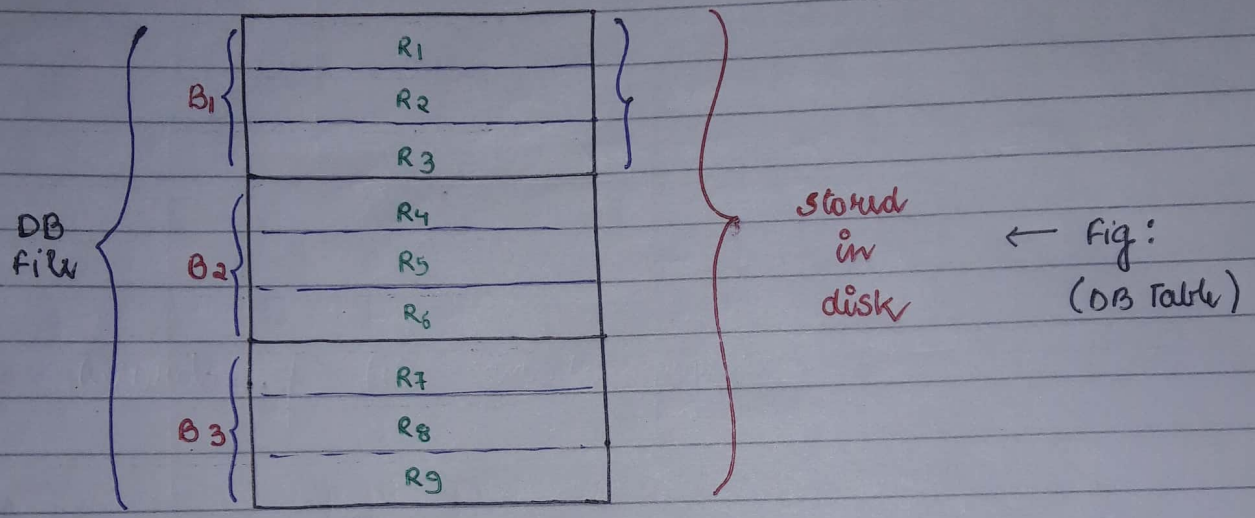
[all red parts ID's] − [Pid's of Catalog (T_2) which are supplied by T_1.Sid] = ϕ

```
Select T1.Sid
From Catalog T1
Where Not exist (
    Select Pid
    From Parts
    Where Color = Red
    Except
    Select T2.pid
    from Catalog T2
    Where T2.Sid = T1.Sid )
```

11/11/17

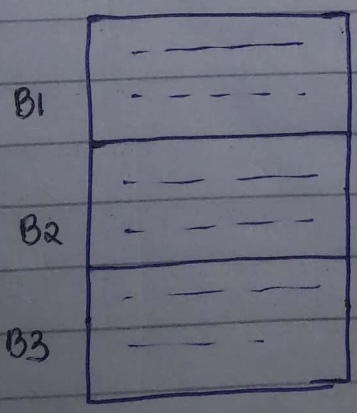
File Organization and Indexing

- Database is collection of files (tables)
- File is a collection of pages (blocks)
- Block is a collection of records.

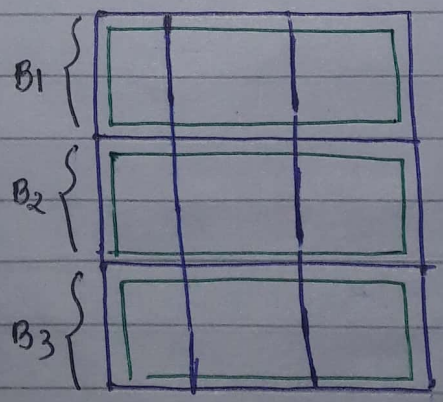


← Fig: (DB Table)

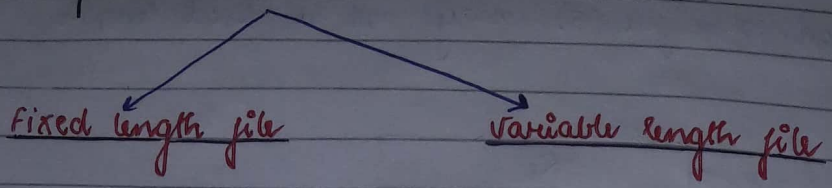
- Data is accessed from disk, block by block.
- In Operating System
  ↓
  (flat file)



In DBMS file
↓



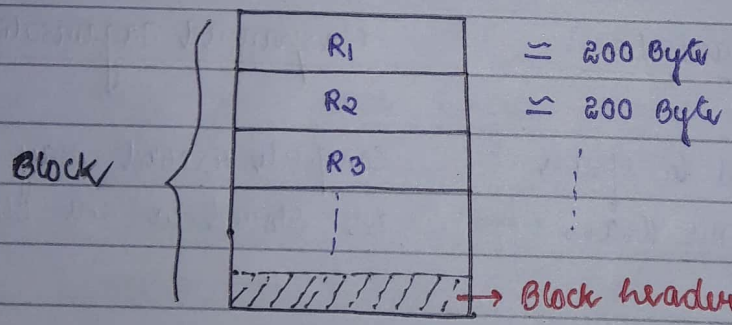
Records of Database File —



① Fixed length file :

```
Create table R
(
    A char(100),
    B char(50),
    C char(50)
);
```

Size of the record ≃ 200 Bytes



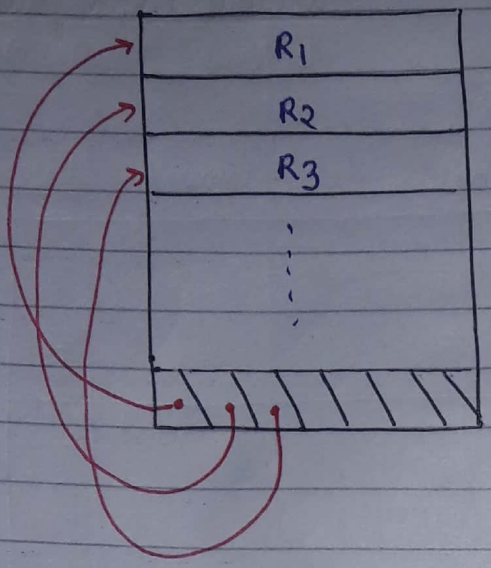
(offset table to access records within the block)

② Variable length file :

```
Create table S
(
    D char(100),
    E char(50),
    F text
);
```

size of F could be any ↓ it depends on data we store

Record of file (s) may not be fixed length

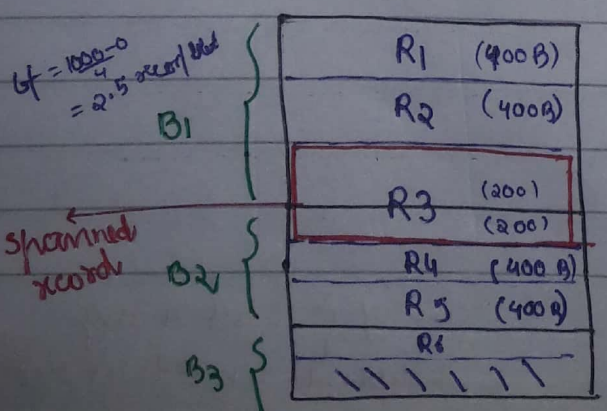


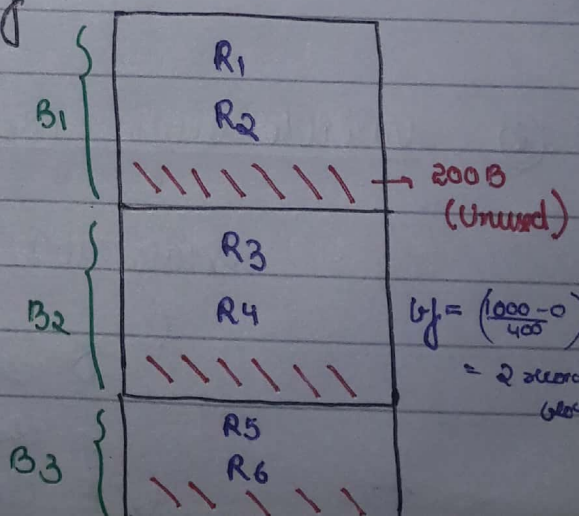
Records of file can be organised as —

variable length record

for fixed length record

Spanned organisation	Unspanned organisation
• Record is allowed to span or store in more than one block.	Complete record ~~may or~~ must be stored in one block.
• Eg- Block size = 1000 Byte Record size = 400 Byte	Eg —





• Possible to allocate file without internal fragmentation	may not be possible to allocate file without internal fragmentation.
• More access cost (as record is in more than one block) and is complex to organize (drawback)	less access cost and easy to organize.
• DB uses spanned by default for variable length record.	DB used unspanned by default for fixed length record.

Block Factor —

→ Maximum possible records per block is called block factor

→ It is only for fixed length records.

Block size : B bytes
Block header size : H bytes
Record size : R bytes

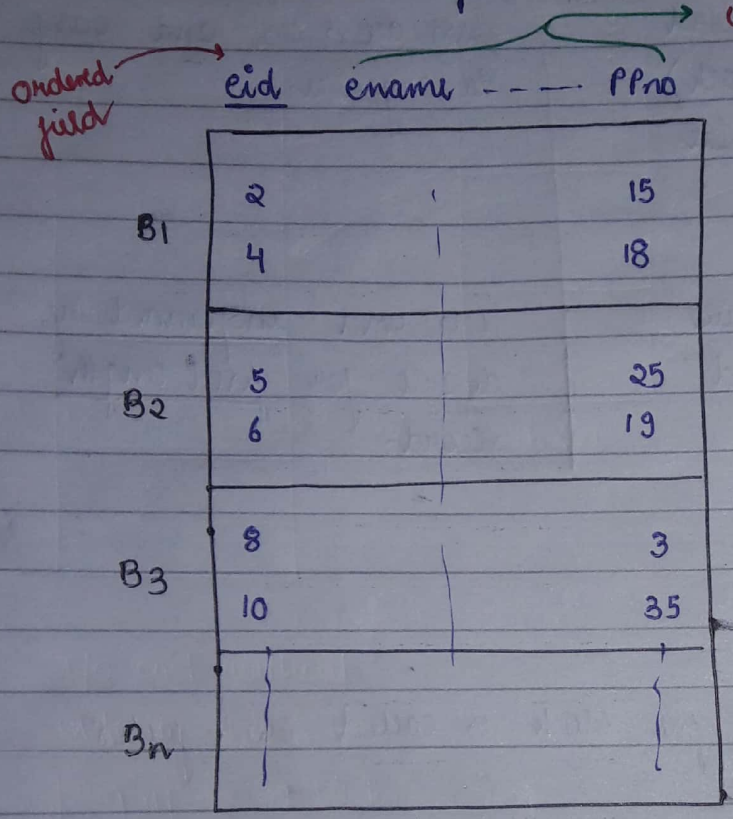
(*) by default use this

Block factor for unspanned organization → $\left\lfloor \frac{B-H}{R} \right\rfloor$ records/block

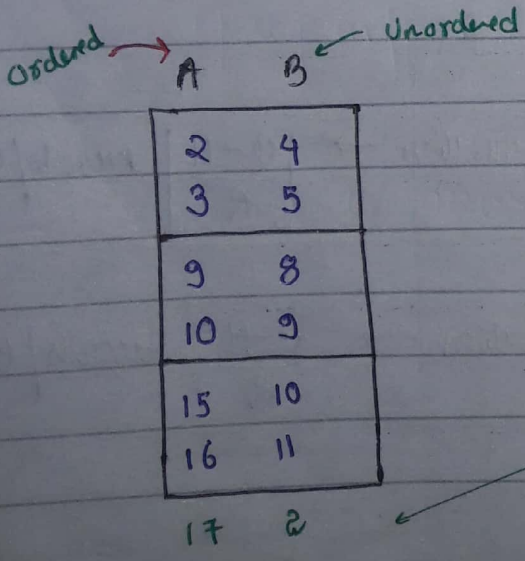
Block factor for spanned organization → $\frac{B-H}{R}$ records/block.

Indexing

DB File [Emp]



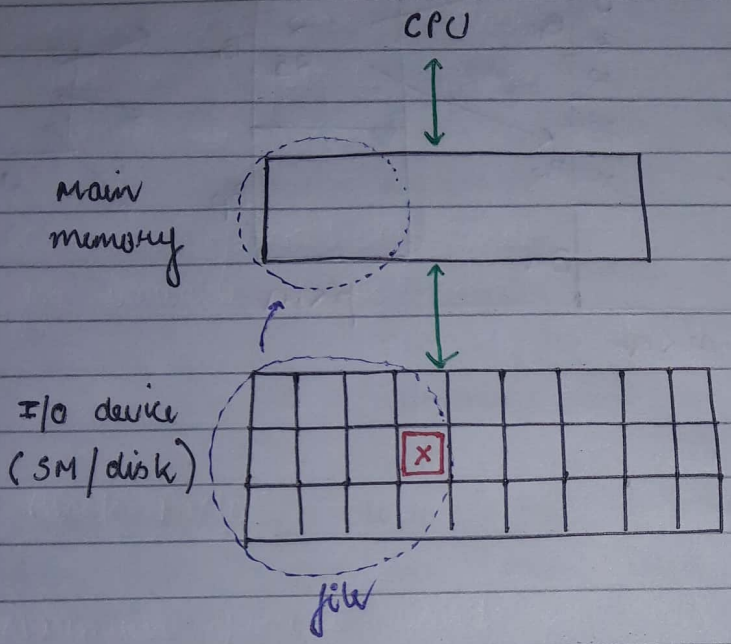
- Indexing is used to reduce the access cost.



But, here B is also ordered, but this is not known to DB, there is no gaurentee that in future, same type of record will be there

→ I/O cost (Access cost) :

Number of secondary memory blocks (disk blocks) required to transfer data from disk to main memory in order to access some record is called I/O cost.



- I/O cost to access record from database file of n blocks without index

(a) Based on ordered field access :

```
select *
From emp
where eid = x;
```

$\lceil \log_2 n \rceil$ block access cost

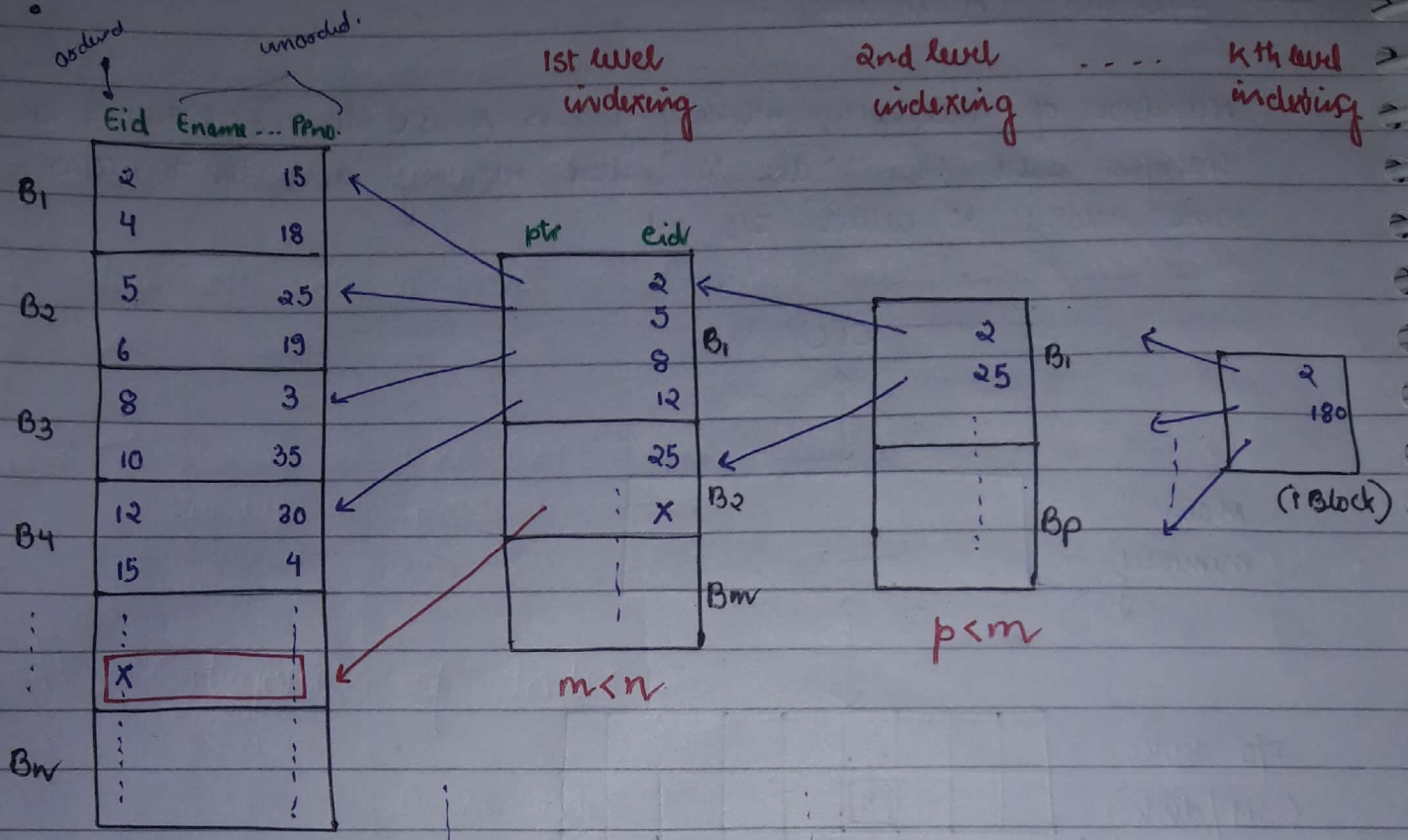
with assumption that middle of the file can be found by DBMS.

(b) Based on unordered field access :

```
Select *
From emp
where Pno = x;
```

'n' block access cost

(drawback of indexing)



I/O cost without indexing is : $\lceil \log_2 n \rceil$ or n blocks

I/O cost with indexing using 1st level indexing

$\lceil \log_2 m \rceil + 1$ blocks

I/O cost to access records using multilevel index

$(k+1)$ blocks.

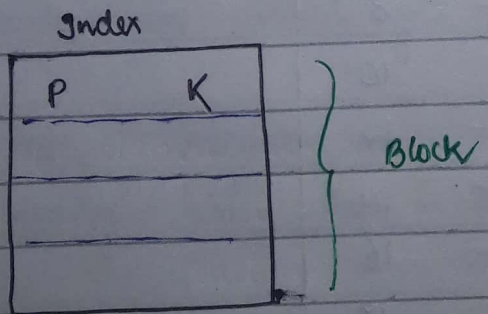
Index file —

- Each entry of index file is 2 fields.
  < search-key , pointer >
  - → field used for indexing.
  - → It can be a key (or) non key.
  - → can be an ordered field (or) unordered field

- Index file block factor is max possible index entries [searchkey, ptr] pairs in index block.

> Block factor of index file = $\left\lfloor \frac{B - H}{P + K} \right\rfloor$ entries/block

B: size of block
H: size of block header
K: size of search key
P: size of pointer.



Multilevel Index —

- Index to index until, at last level, there is one block.
- Ideal I/O cost to access record using multi level index is (k+1) blocks.

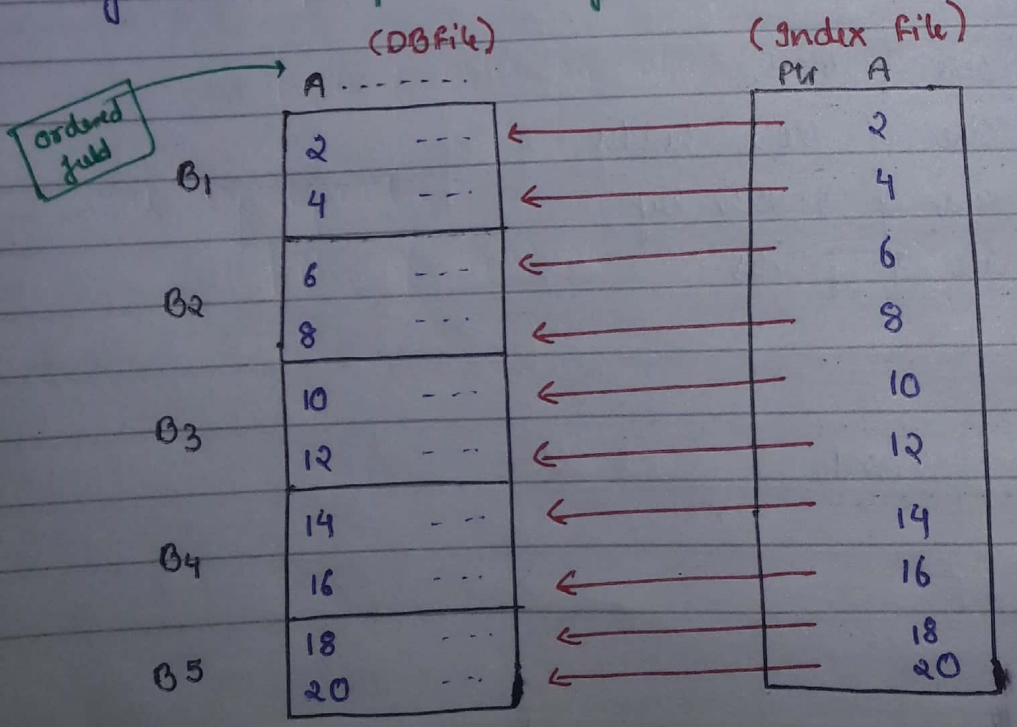
Note — Ideally we take (k+1) blocks, but it may increase slightly in practical conditions.

Categories of Index :

1. Dense index
2. Sparse index

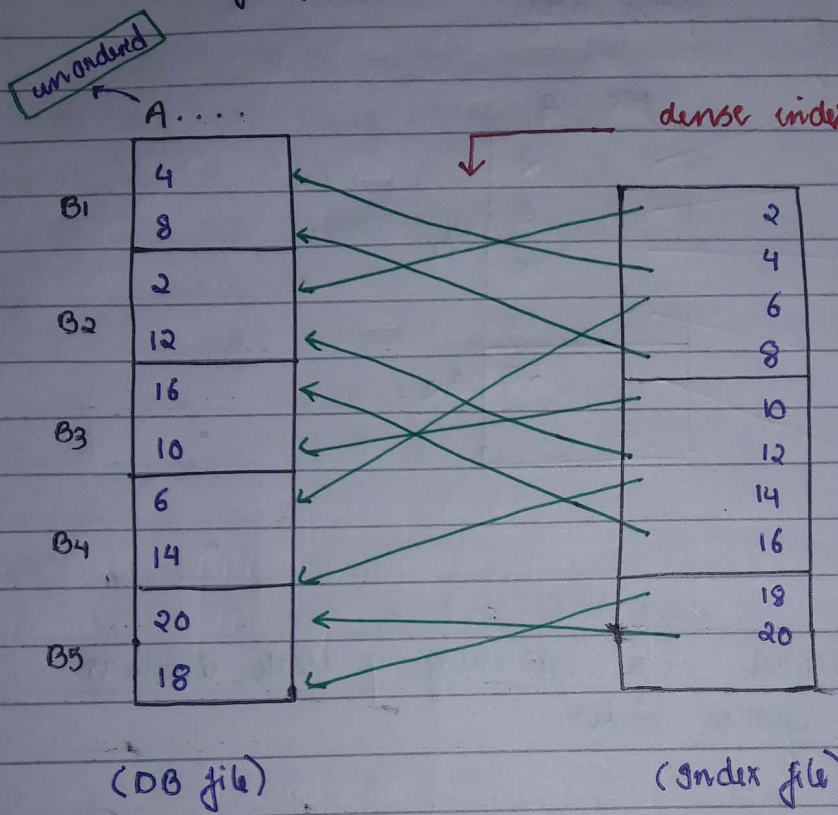
Dense Index —

- More entries are used for the indexing of database files. (max possible entry in index)



- For each record of DB file, there exist an entry in index file.

Indexing for unordered field of DB file —



No. of index entry = No. of records

- Dense indexing can be done for ordered as well as unordered field.

Sparse Index —

→ Less entries in index file. (objective)

→ For set of record there exist an entry in index file (i.e. for many ~~entry~~ record there is one entry)