

Database: a collection of related data which represents some aspect of real world.

DBMS (Database Management System): is a software for storing and retrieving data. It is used to do the operation like (insertion, updation, deletion).

- Structure data is stored ~~like~~ a particular structure. Most used is Relation DBMS. It stores the data in form of table.

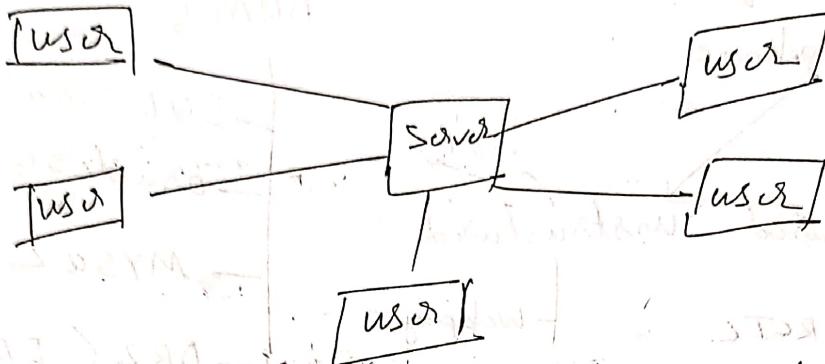
• Unstructure data has no structure. Web pages has no specific structure like there may be different types of photos, videos. Suppose there maybe photos, videos, text in webpage first.

Nowadays 90% data are unstructure.

Ex- Big data, Hadoop.

File System & DBMS \Rightarrow

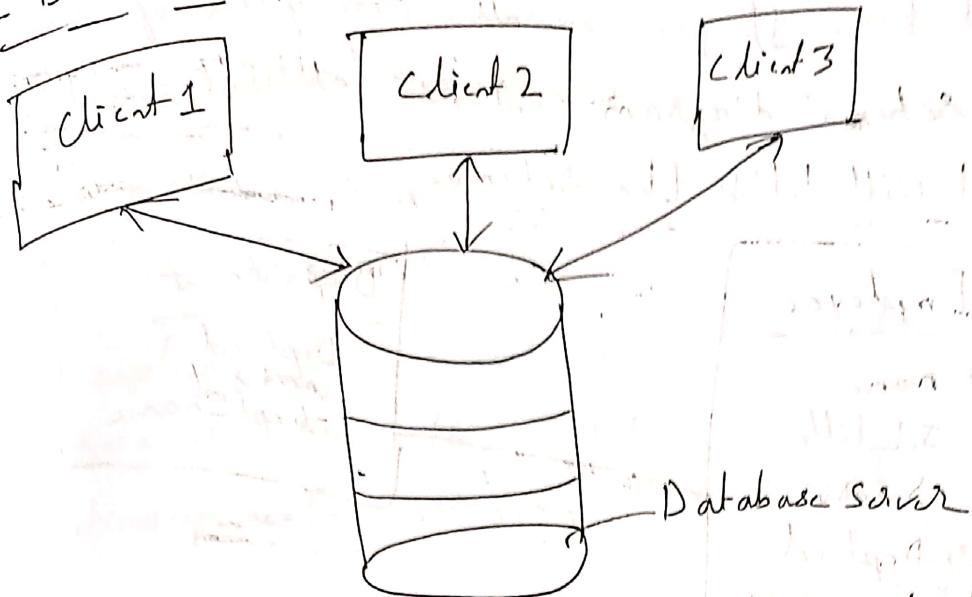
Earlier we use ~~some~~ file system for storing and accessing data. Though still we use file system in our computer. But in client ~~server~~ architecture we are using DBMS.



Drawback of file system / Benefit of DBMS

- i) Difficulty: If we want just 1 KB data (train detail) but it was stored in a big file. In file system we will get the whole file (50 KB). It will take a lot of time as well as memory.
- ii) Attributes: In file system we have to know file name as well as location. DBMS, we just have to write a query.
- iii) Concurrency: In file system concurrent access is difficult. But DBMS helps concurrent access by multiple users. DBMS has protocols for it. Multiple user booking ticket.
- iv) Security: File system does not provide role based system. DBMS gives role based access control. It enhances the security (user, admin, student, faculty)

2 tier Architecture

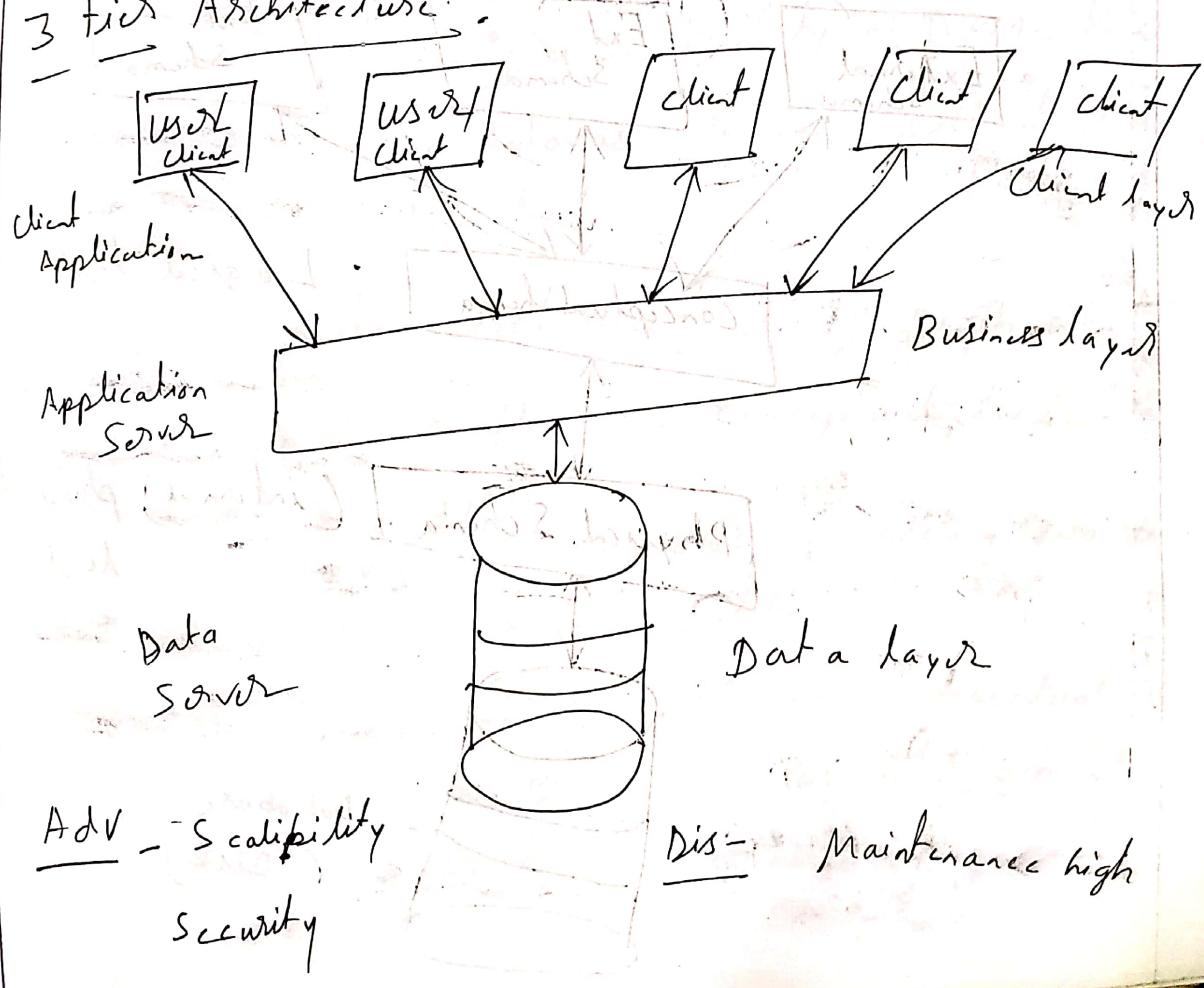


client have an API (JDBC, ODBC) to connect to the database.

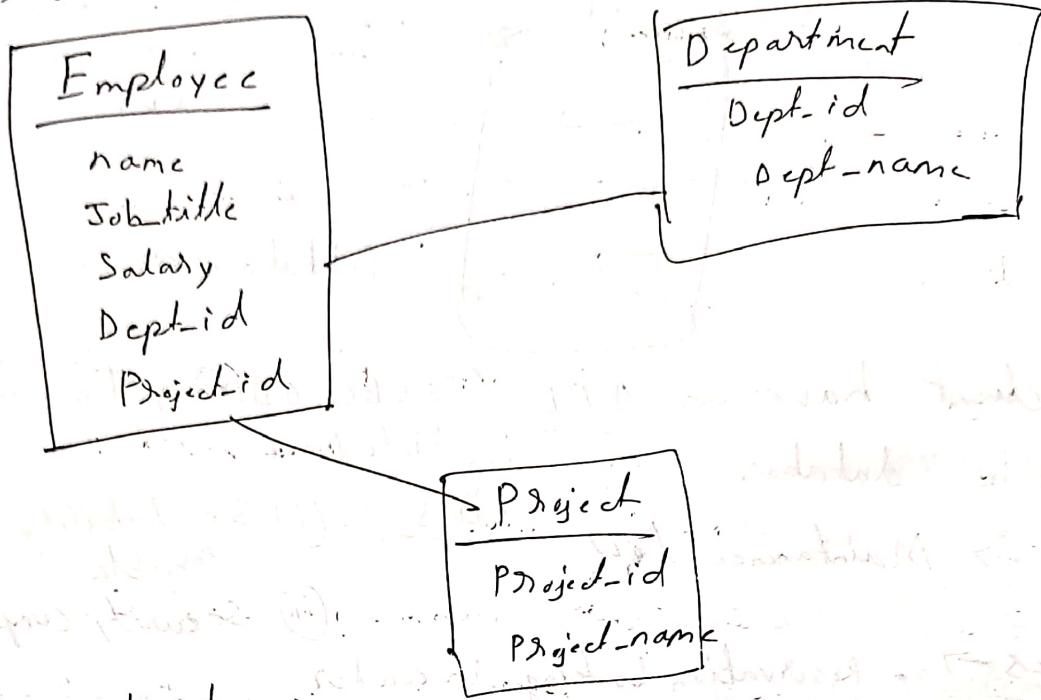
Adv → Maintenance low. Dis :- (i) Scalability not possible
(ii) Security compromised

- Reservation booking in counter
- Bank Physically money deposit or withdraw via form.

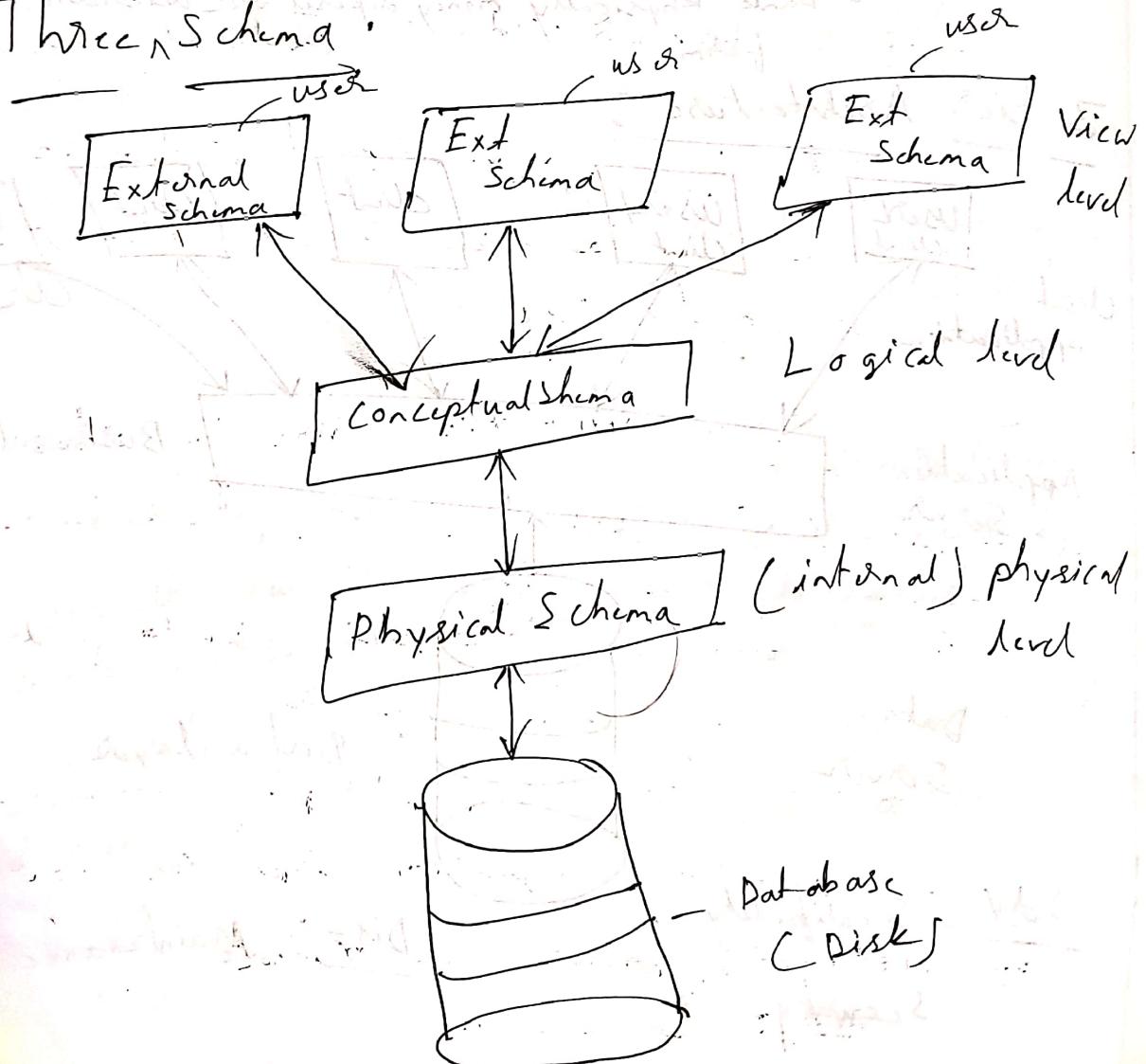
3 tier Architecture



Schema: It is a logical representation of database. It gives overall description of database. A schema diagram contains attributes, entities that will define the schema.



Three Schema level



main motive is to data independence (User and database should not connect directly). User does not know their data where is stored or how.

- External Schema is like how user will view their data. Student, Faculty, Dean has separate view.
- Conceptual/ logical schema define the design of database. We define entities, attributes & their relationships. ER diagram is its example.
- Physical schema tells how the data will be stored on storage device. The data can be stored in form of files.

* Database administrator handles physical Schema
Database designer works on conceptual schema
Interface designer works on External schema.

Data independence \Rightarrow a property of DBMS that helps to change a database schema without changing a schema at a higher level.

• Physical data independence: changing physical structure will not affect on conceptual level. This is achieved by mapping.

Ex - • using new storage device • switching data structure
• change location of data.

• Logical data independence: changing conceptual schema without any effect on view level. achieved by mapping.

Ex - • Add/modify/delete near attribute
• Merging two record.

(6)

Keys \Rightarrow Key is a set of attributes that can identify each tuple uniquely in a table.

- Candidate Key: A set of minimal attributes that can identify each tuple uniquely in the table.

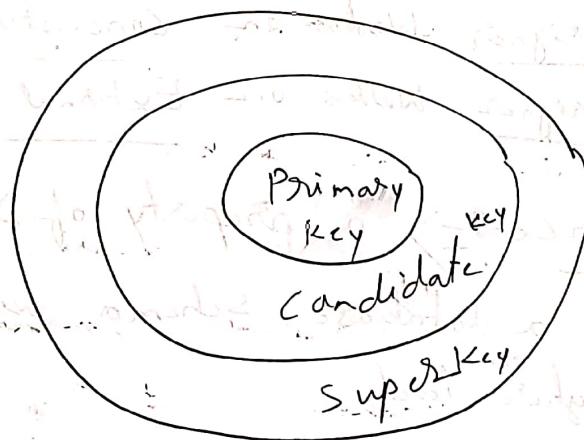
A & AB both identify tuple unique then A will be candidate & AB will be super key.

- Primary Key: A candidate key that the database designer selects while designing the database.

- Primary keys are unique and not null.

(*) only 1 primary key is possible (made of attribute or set of attributes).

- Alternate Key: Candidate keys that are left unused after implementing the primary key are called Alternate key.



Candidate key in student table while designing.

- Adhar card

- Roll no] - Primary key for Adhar card

- Reg. no]

- Voter id

- Phone no

- Pan no

- Email id

Adhar
card
key

• Primary key always provided by the system.

(7)

Foreign key: It is an attribute or set of attributes that references to primary key of same table or another table (relation).

- It maintains referential integrity.

Student		Courses	
Roll-no	name	Course id	Course name
1	Ani	C1	C8
2	Ani	C2	Maths
3	Jhili	C3	Physics

+ Referenced table + Referencing table

↑ primary key name may not be seen

- There may be more than one foreign key.

creation - Create table course

{ courseid varchar(10),

course name varchar(20),

Rollno int references

student(Rollno)

if we make foreign key in already created table

Alter table course

ADD constraint fk

foreign key (Rollno)

references student(Rollno)

Referential Integrity of Foreign Key \Rightarrow

integrity - being honest

- Referential integrity is a constraint applied on foreign key which requires the foreign key to have a matching primary key which makes sure from a row in one table to another table is valid.
- We will get error if we violate referential integrity.

Reference table

- i) insert: no violation
- ii) Delete: May cause ~~deletion~~ violation
 - If we delete a data which is not available in foreign key so no problem but if it is in foreign key then it will violate.

Solution: \Rightarrow On delete cascade (automatically all the rows which were taking reference will also deleted)

- On delete set null (automatically all the rows which were taking reference will be null)

The problem occurs when the foreign key is also primary key of table then nullify is not possible
- On delete no action: It is by default. It will give error.

(iii) update: May cause violation

Solution - On update cascade
On update noaction

Referencing table

(i) insert - may cause violation

(ii) deletion - no violation

(iii) updation → may cause violation (if we update foreign key value)

a) Let $R_1(a, b, c)$ and $R_2(x, y, z)$ be two relations in which 'a' is foreign key in R_1 that refers to the primary key of R_2 . Consider options (a) Insert into R_1 (b) Insert into R_2 (c) Delete from R_1 (d) Delete from R_2 which is true about referential integrity.

(a) option a & b will cause violation

(b) b & c will cause violation

(c) c & d will cause violation

(d) a & d will cause violation

\Rightarrow (d) ✓

fk R_1 spk R_2

a	b	c

x	y	z

(10)

Superkey \Rightarrow Superkey is a combination of all possible ^{attribute} which can uniquely identify two tuple in a table.

Superset of any candidate key is Superkey.

Rollno.	name	age

- Superkey must have one candidate key

Roll no \rightarrow candidate key - Superkey.

Rollno, name \rightarrow Superkey

Rollno, age \rightarrow Superkey

Rollno, name, age - Superkey

name, age - ~~Superkey~~

(x) If $P(A_1, A_2, \dots, A_n)$ then how many superkey

\rightarrow if A_1 is candidate key then Superkey possible 2^{n-1} ($\frac{A_1}{\text{fixed}} \frac{A_2}{\text{fixed}} \dots \frac{A_n}{\text{fixed}}$) 2^{n-1} possibility

\rightarrow if A_1, A_2 both candidate key

$A_1, A_2, \dots, A_3 - A_n$

\swarrow fixed fixed

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

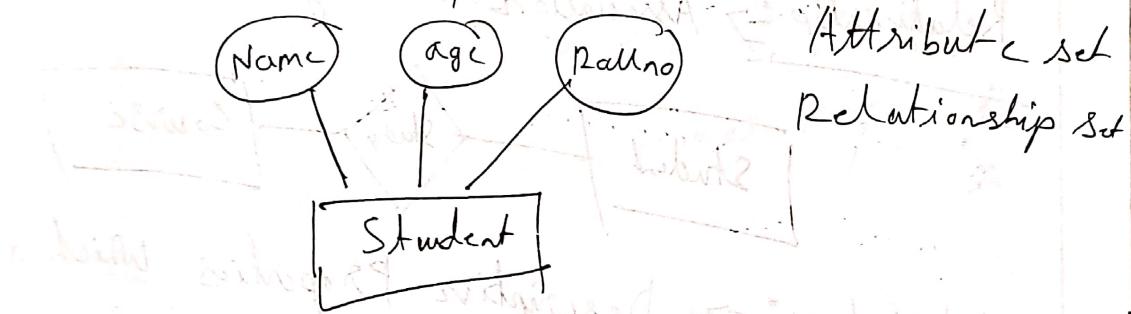
if A_1 fixed : if A_2 fixed common part

\rightarrow if $A_1 A_2 \neq A_3 A_4$ candidate key then
 $2^{n-2} + 2^{n-2} - 2^{n-9}$

Introduction to ER Model \Rightarrow

ER diagram or Entity relationship model is a conceptual model that gives the graphical representation of the logical structure of database.

- Before doing house we make design
- It shows all the constraints and relationships that exist among the different components
- ER diagram mainly composed of Entity sets



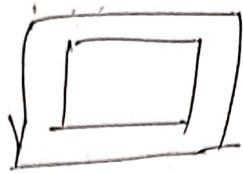
Entity \Rightarrow a real world object with an existence. In a college database student, professor, course is an entity.

- Strong entity : if it has a primary key. Represented by single rectangle.

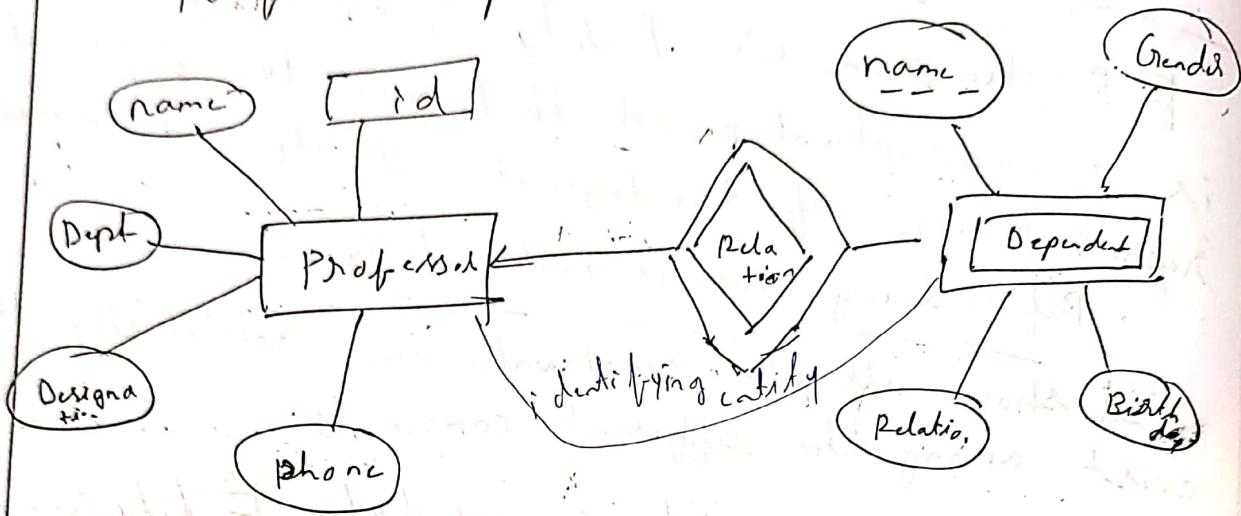


Professor is a strong entity, it has a professor id as primary key.

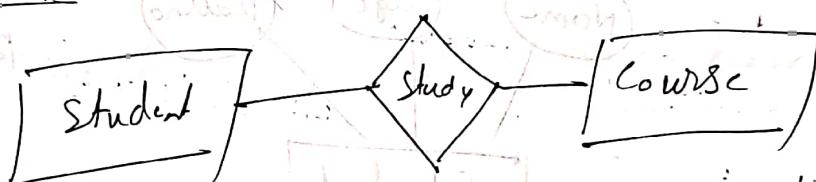
- (12)
- Weak entity : Do not have Primary key.
Dependent on strong entity. Represented by double rectangle.



Professor-dependent is weak entity.



Relationship \Rightarrow Association among several entities



Attributes \Rightarrow Descriptive Properties which are owned by entity.
like - Age, name, class is attribute of students

Single attribute: attribute which can take only one value

ex - Age is single valued (we can't write two different age.)

Multivalued: attribute which can take

multiple values. ex - Mobile no can take multiple values

• Simple: attribute which can not be divided

Ex- Age is simple

• Composite: attribute which can be divided further

Ex- Name can be divided first-
name, middle name, last name.

• Stared: attribute which can not be derived.

Ex- DOB is stared

• Derived: attribute which can be derived by other attribute

Ex- Age can be derived from DOB.
dotted line:

• key: attribute which can identify an entity in unique way.

Ex- Roll no is key attribute.

~~Roll no~~ - underlined is not key attribute.

• non key: attribute which is not key attribute.

Ex- Name, Mobile No.

• Required: mandatory value

Ex- Name, Mobile No.

• optional: Not mandatory

Ex- Name, Mobile No.

• Complex: Composite + multivalued

Ex- Residential address with 2 diff. numbers

• Abstract: combination of other fields

Ex- Date of Birth, Address, etc.

• derived: based on other fields

Ex- Name, Address, etc.

Degree of Relationship (cardinality) \Rightarrow

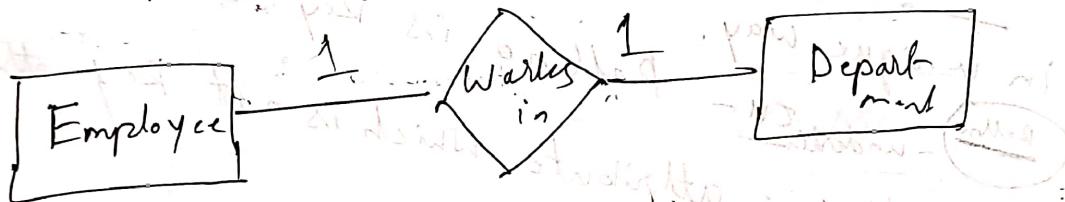
maximum number of relationship instances in which ~~can~~ an entity can participate.

- (i) one to one (ii) one to many
- (iii) Many to one (iv) Many to many

(i) One to one \Rightarrow An entity in Set A can be associated with atmost 1 entity in Set B and vice versa.

A record in one entity is associated with a record in other entity and vice versa.

e.g. Country - capital
Person - fingerprint



Employee can work in only 1 department

Department can have 1 Employee only

Employee			Work			Department		
PK	Name	age	PK	FK	FK	PK	Name	Loc
E1	Ari	20	E1	D1		D1	CS	Paris
E2	Say	21	E2	D3		D2	nurse	UK
E3	Ani	22	E3	D2		D3	CS	Paris

Employee

Work

Department

- In work there will be minimum 2 attributes.
- Both can be primary key.

(11) PK = Either E-ID or DIP

(15)

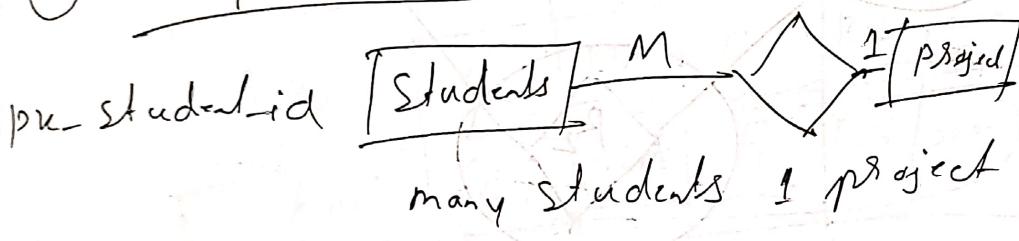
Can we reduce the tables
→ Yes we can make 2 table from 3.
If E-ID is primary key then we will just merge the Employee & Work.

(ii) One to many, \Rightarrow PK will be project-id

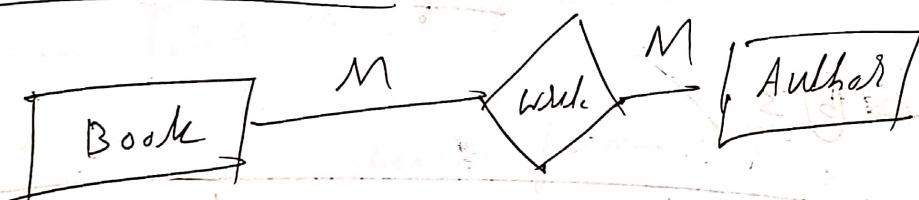


can reduce from 3 to 2 1 student many project

(iii) Many to one ^{table also}



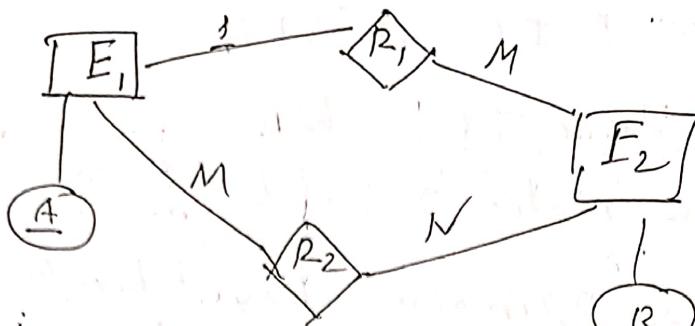
(iv) Many to many



PK will be combination of bookid & Authorid

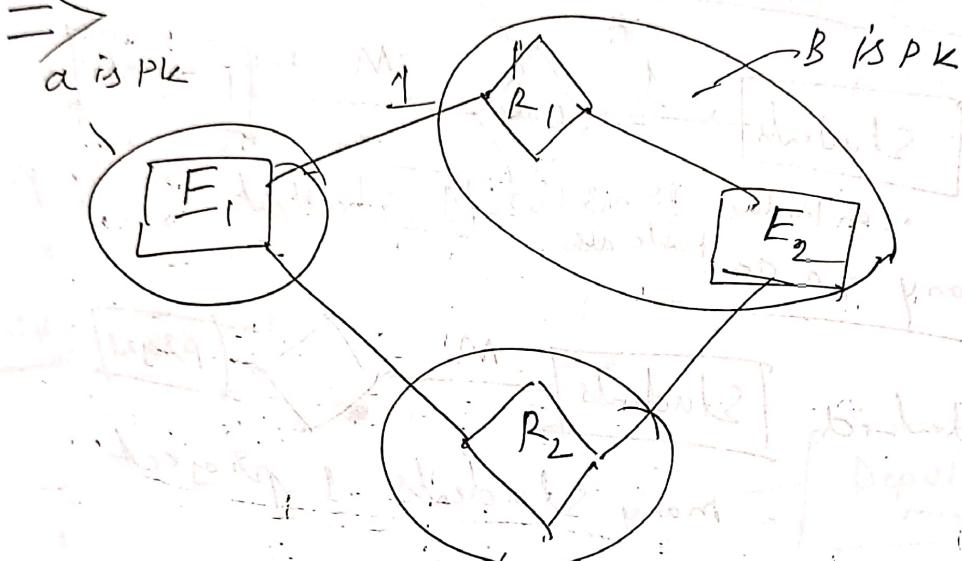
(6)

Q1



minimum no. of tables required to represent this E-R model into Relational model?

- (a) 2 (b) 3 (c) 4 (d) 5



\rightarrow (b) 3 ✓

Normalization \Rightarrow Process for making the data consistent by

- Reducing ~~Redundancies~~ or removing Redundancies (similar values)
- Ensuring the integrity of data

Row level duplicacy:

Roll	Name	Age
1	Ani	20
2	Ani	22
1	Ani	20

We remove row level duplicacy with the help of Primary key. We can make Roll primary key.

Column level Duplicacy:

Sid	Sname	Lid	Cname	FID	Fname	Salary
1	Ani	C1	DBMS	F1	Bob	30000
2	Ani	C2	Java	F2	Don	40000
3	Paltu	C1	DBMS	F1	Bob	30000
4	Hari	C1	DBMS	F1	Bob	30000

just a small example

Column 1, 3, 4 is copy of each other except

Sid, 4 Sname.

This type of duplicacy form major 3 problem

(i) insertion anomaly (problem occurred in specific case)

We can easily insert details of student (Sid, sname) but we can't just add any details of faculty & course as we have no student id (but as it is pk it ~~can't~~ can't be null).

(ii) Deletion anomaly: We can delete details of 1 or 3 or 4

but if we delete 2 (Sid) then we will lost the details of that particular course & faculty details as it was available only 1 time in database.

(iii) updation anomaly: \Rightarrow we can update the sname but if we try to salary of bob then it will update the salary 3 times. (as the data is written 3 times.) time consuming

Simple solution : divide the table into 3 tables

<u>Sid</u>	<u>sname</u>
<u>cid</u>	<u>course</u>
<u>Employee</u>	<u>Employee</u>

First Normal Form (1NF) \Rightarrow

- Each cell of the table contains only an atomic value. (either singular or null value)

<u>P.K.</u>	<u>Roll</u>	<u>Name</u>	<u>course</u>
1	Anil	C/C++	
2	Anil	Java	
3	Shili	C/DBMS	

is not in 1NF

as it contains multiple values

Solution \rightarrow we can represent this in 3 way to make it 1NF.

Solution 1 \Rightarrow

<u>Roll</u>	<u>name</u>	<u>course</u>
1	Anil	C
1	Anil	C++
2	Anil	java
3	Shili	C
3	Shili	DBMS

Roll + course

combined

P.K.

Solution 2

PK

Roll	name	course1	course2
1	Ari	C	C++
2	Ari	Java	null
3	jihili	C	DBMS

But the problem is when only a student is enrolled in a lot of course & other are enrolled only 1. So there will be so much null (as not good representation)

Solution 3 : Divide table into 2

Roll no	name
1	Ari
2	Ari
3	jihili

Rollno	course
1	C
1	C++
2	java
3	C
3	DBMS

Base table / Reference table

Rollno + course
TPK

Closure Method

If we want to find candidate key we use closure method

(i) $R(ABCD)$

FD { $A \rightarrow B, B \rightarrow C, C \rightarrow D$ }
 meaning of A closure means what A can determine

$$A^+ = ABCD$$

As B also determine C then we can say A also determine C

C again determine D

A will determine itself Also

So A is determining all the attribute then A is the candidate key

$$B^+ = BCD \times$$

$$C^+ = CD \times$$

$$C_K = \sum A^3$$

$$D^+ = DX$$

$$AB^+ = ABCD$$

AB determine AB

B also determine C

Here AB is not candidate key as it is not minimal. AB is super key. prime attribute = $\{A\}$ non prime attribute = $\{B, C, D\}$

(ii) $R(ABCD)$

FD = { $A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A$ }

$$\Rightarrow A^+ = ABCD$$

$$B^+ = BCDA$$

$$C^+ = CDAB$$

$$D^+ = ABCD$$

$$C_K = \{ A, B, C, D \}$$

Prime attribute are the attribute that makes candidate key

The prime attributes are $\{A, B, C, D\}$

iii) R (ABCDE)

$$FD = \{ A \rightarrow B, BC \rightarrow D, E \rightarrow C, D \rightarrow A \}$$

\Rightarrow short side - the attributes that are on right sides is determined ~~and~~ is determined

How B, D, C, A is being determined

So we can conclude that each candidate key must contain E (as E can be determined by itself)

$$\checkmark (\underline{AE})^+ = \underline{ABE} = ABE \subset \underline{ABCDE}$$

$$\checkmark (BE)^+ = BE = BE^C = BE^D = BE^DA$$

Suppose we get AE is candidate then we can check

if A is in right hand side of any FP
right hand

$H \cup C \rightarrow A$

$$(DE)^+ = ABCDE \checkmark$$

$$(CE)^+ = CE = eC \times$$

$$C_K = \{ A_E, B_E, \underline{B_E} \}$$

- Prime attribute = { A, B, D, E }

non prime attribute = {C}

Function Dependency \Rightarrow

~~X~~ \rightarrow $X \rightarrow Y$ - dependent

it says X determines Y

or
 y is dependent/determined by x

Cn- Sid name
 Anisha

1 2

We have a confusion that whether there is 2 name or same people, we will check Sid. If Sid different then both Anisha different else if Sid same both are same person

Sid → name

Ans X not valid as Sid is
determinate and it says
Same student but name same

Two type of FD if there is 2 student

(i) trivial FD: $X \rightarrow Y$ is said to be trivial if

if $Y \subseteq X$

$\{S_{id}, S_{name}\} \rightarrow S_{id}$

Lits n/p

† φ

ii) non trivial : $X \rightarrow Y$ is said to be non trivial if $y \notin X$
 $LHS \cap RHS = \emptyset$

Properties of FD

i) Reflexivity : if $Y \subseteq X$ then $X \rightarrow Y$

also. $X \rightarrow Y$ is sid \rightarrow sid ✓

ii) Augmentation : if $X \rightarrow Y$ then $XZ \rightarrow YZ$

e.g. sid \rightarrow sname then {sid, phone} \rightarrow {sname, phone}

iii) Transitive : if $X \rightarrow Y$ & $Y \rightarrow Z$ then

Ex - sid \rightarrow sname and sname \rightarrow salary then
 sid \rightarrow salary

iv) Union : if $X \rightarrow Y$ & $X \rightarrow Z$ then $X \rightarrow YZ$

v) Decomposition : if $X \rightarrow YZ$ then $X \rightarrow Y$ & $X \rightarrow Z$

vi) $XY \rightarrow Z$ the. $X \rightarrow Z$ & $Y \rightarrow Z$ X X not valid

vii) Pseudo transition : $X \rightarrow Y$ & $WY \rightarrow Z$ then
 $WX \rightarrow Z$

viii) Composition : $X \rightarrow Y$ & $Z \rightarrow W$ the.
 $XZ \rightarrow YW$

(24)

Second normal form (2NF) \Rightarrow

- Relation/table must be in First normal form (1NF)
- No partial dependency should not be there
 $\xrightarrow{\text{mean}}$
 All the non prime attribute fully only functional dependent

proper subset of \rightarrow non prime attribute
candidate key

Partial dependency

Eliminates hidden dependency

Suppose $CR = \{AB\}$

$$R = \{A, B, C, D\}$$

prime attribute = $\{A, B\}$

non prime attribute = $\{C, D\}$

if $\begin{array}{l} AB \rightarrow D \\ AB \rightarrow C \end{array} \} \checkmark$ fully dependent

$A \rightarrow C \} \quad B \rightarrow D \} \quad A \rightarrow D \quad \text{partial dependency}$

$$\overline{a} \rightarrow R = \{A, B, C, D, E, F\}$$

$$FD = \{C \rightarrow F, E - A, EC \rightarrow D, A \rightarrow B\}$$

\Rightarrow In order to check CR we will check right hand side, we notice EC is not determined
So in CR there must EC

$$\checkmark EC^+ = EC = EC \setminus A = EC \setminus AD = EC \setminus A \setminus B$$

$= EC \setminus A \setminus B$

25

Now we check whether E/C is available in right hand or not. In this case there is no available.

: So $Ck = \{El\}$

∴ Prime attribute = { E, C }

non prime attribute = { A, B, D, F }

Now we will check whether partial dependency exist or not

$C \rightarrow F$ non prime attribute
 " Subsets of candidate key partial dependency ✓

So this is not in 2NF

Third Normal Form (3NF)

- table must be in ~~2NF~~ 2NF

- No transitive Dependency

(*) to be in 3NF LHS should be Superkey/Candidate key and RHS is prime attribute

a) $R[ABCD]$, FD: $AB \rightarrow C$, $C \rightarrow D$

$$\Rightarrow CK = \{AB\}$$

$$\text{Prime add'l} = \{A, B\} \quad \text{non prime} = \{C, D\}$$

$A B \rightarrow C$

$C \rightarrow D$ X it is transitive
dependency.

a) R(ABCD) FP: $AB \rightarrow CD, D \rightarrow A$

$$\Rightarrow LK = \{AB, BD\}$$

$$AB^+ = ABCD \checkmark$$

As A is dependent on D

$$DB^+ = ABD = ABCD \checkmark$$

Prime attribute = {A, B, D}

non prime attribute = {C}

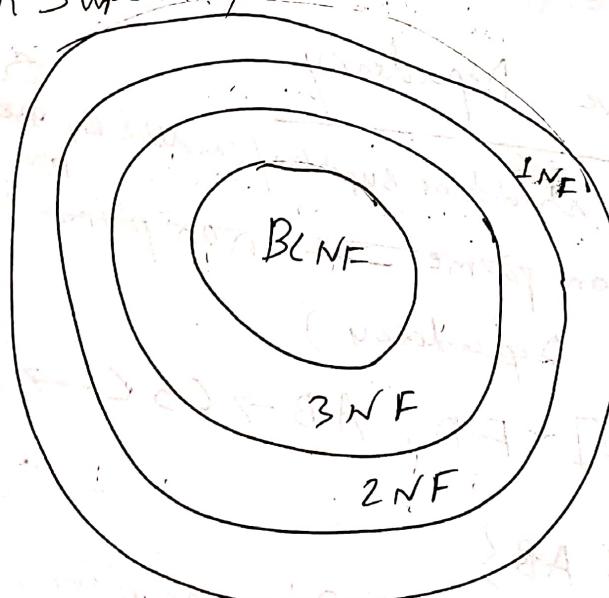
$$AB \rightarrow CD \checkmark \quad D \rightarrow A \checkmark$$

\therefore This is in 3NF.

BCNF (Boyce-Codd Normal Form)

\therefore to be in 3NF

- Left Hand Side (LHS) should be always candidate key or Super key



- Third Normal (3NF) always ensure Dependency preserving decomposition
but not in BCNF

⇒ Dependency preservation ensures - none of the FD that hold on the original relation are lost

- the sub relation still hold or satisfy FD of the original relation.

- Both 3NF & BCNF ensures lossless decomposition

Types of Decomposition ⇒

- i) Lossless Join Decomposition ⇒ if a Relation R is decomposed into sub relations R_1, R_2, \dots, R_n . This decomposition is called lossless join decomposition when the join of the sub relation results the same relation R that was decomposed.

$$R_1 \bowtie R_2 \bowtie R_3 \dots \bowtie R_n = R$$

⋈ - is a natural join operator

Natural join =
Cross product + Common attributes
 $R_1 \bowtie R_2$ (Value same)

R		C
A	B	C
1	2	1
2	2	2
3	3	2

R ₁	
A	B
1	2
2	2
3	3

R ₂	
B	C
2	1
2	2
3	2

R ₁		R ₂		
A	B	B	C	
1	2	2	1	✓
1	2	2	2	✓
1	2	2	3	✗
2	2	2	1	✓
2	2	2	2	✓
2	2	2	3	✗
3	3	2	1	✗
3	3	2	2	✗
3	3	3	2	✓

Natural join table

A	B	C
1	2	1
1	2	2
2	2	1
2	2	2
3	3	2

Lossy in terms of
duplicacy/inconsistency

The above join is called lossy join as the common attribute was not primary key or maintable.

Lossy Join Decomposition \Rightarrow if Relation R is decomposed into sub relation R_1, R_2, \dots, R_s

This decomposition is called lossy join decomposition. When the join of relation does not result the same relations R.

(*) if common attribute is ~~subset of other~~
should be candidate key or superkey of main table then there will be lossless join decomposition.

$$\text{Ex} - R_1(A, B) \quad R_2(A, C)$$

$$(i) R_1 \cup R_2 = R$$

$$(ii) R_1 \cap R_2 \neq \emptyset$$

(iii) common should be R_1 CK or R_2 CK or both

- 1NF \Rightarrow
- no multivalued attribute
 - only single attribute
- 2NF \Rightarrow
- In 1st NF must
 - + No partial dependency, only full dependency
 - if $AB \rightarrow CK$ & $C \rightarrow D$ non prime attribute
 - $A \rightarrow C \times$ Partial dependency
 - $A-B \rightarrow C \checkmark$

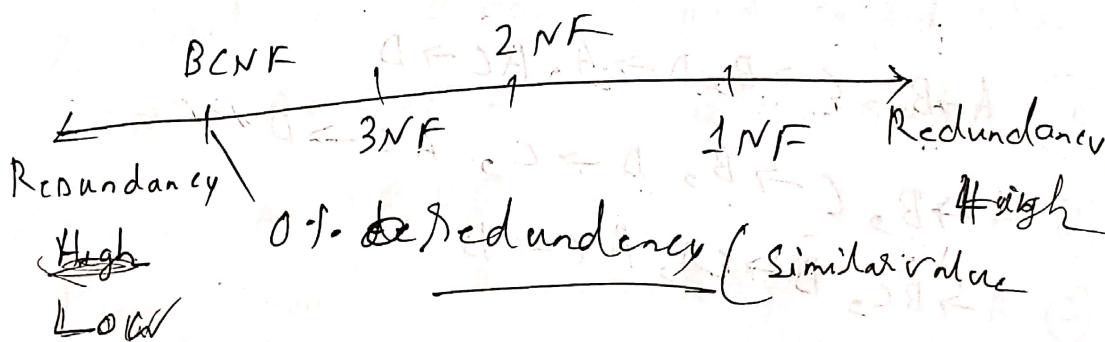
- 3NF \Rightarrow
- In 2nd NF
 - + no transitive Dependency
 - Either LHS candidate key or Superkey or PHS
 - prime attribute

- BCNF \Rightarrow
- In 3rd NF
 - + LHS must be CK or SK

- 4th NF \Rightarrow
- In BCNF
 - + No multivalued Dependency

Example ($X \rightarrow Y$) X \rightarrow $3Y$ \rightarrow $3Z$

- 5th NF \Rightarrow
- In 4th NF
 - + Lossless Decomposition



Minimal Cover \Rightarrow A minimal cover is a simplified and reduced version of the given set of functions dependency.

It is also called irreducible set.

It is also called as canonical cover.

Steps to find right hand attribute of all FDs

(i) split

$$X \rightarrow YZ$$



$$X \rightarrow Y, X \rightarrow Z$$

$\textcircled{2} A \rightarrow B \rightarrow C$ then
we can't write
 $A \rightarrow C, B \rightarrow CX$

(ii) remove all redundant (Same FDs)

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

Here $A \rightarrow C$ can be removed as it can be achieved

with $A \rightarrow B, B \rightarrow C$

(iii) Find the extraneous attribute in remove it (left side)

$$AB \rightarrow C$$

if $A + C$ contain B then we can remove B

if $B + C$ contain A then we can remove A

Ex - Find minimal cover

$$\{ A \rightarrow B, C \rightarrow B, D \rightarrow ABC, AC \rightarrow D \}$$

a) $A \rightarrow B, C \rightarrow B, D \rightarrow A, AC \rightarrow D$

b) $A \rightarrow B, C \rightarrow B, D \rightarrow C, AC \rightarrow D$

c) $A \rightarrow BC, D \rightarrow CA, AC \rightarrow D$

\Rightarrow Step 1 -

$$A \rightarrow B, C \rightarrow B, D \rightarrow A, D \rightarrow B, D \rightarrow C, AC \rightarrow D$$

Step 2 : \Rightarrow we will see closure of every LHS omitting this FD

$A^+ = A \oplus \text{so } A \rightarrow B \text{ can't be removed}$

$$C^+ \supseteq C \text{ accepted}$$

$$D^+ = DB \subset C \text{ accepted}$$

$D^+ = DABC$ so D is determining B so it will be removed

Now further we will not include this for further consideration

$$A \rightarrow B, C \rightarrow B, D \rightarrow A, D \rightarrow C, AC \rightarrow D$$

$$D^+ = DAB$$

$$AC^+ = ACB$$

Step 3 - $A^+ = AB$ so ~~we can't remove any~~ of $A \notin C$ but we can remove any

Then the minimal cover,

$$A \rightarrow B, C \rightarrow B, D \rightarrow AC, AC \rightarrow D$$

$\rightarrow \text{Q} \checkmark$

Q R(A B C D E F) check the highest Normal Form \Rightarrow FD: $\{AB \rightarrow C, C \rightarrow DE, E \rightarrow F, F \rightarrow A\}$

\Rightarrow First we will have to find candidate key

$$AB^+ = AB C D E F \quad CB^+ = C E D F A B$$

$$CK = \{AB, FB, EB, CB\}$$

$$\text{Prime attribute} = \{A, C, E, FB\}$$

$$\text{non prime} = \{D\}$$

First we will check upper to lower \Rightarrow

BCNF \rightarrow $AB \rightarrow C \checkmark$ AB is CK

$C \rightarrow DE X$ C is not CK/SK

3NF \rightarrow $AB \rightarrow C \checkmark$ AB is SK/CK

$C \rightarrow DE X$ C is not CK/SK

DE is not also non-prime

(C is prime but DE not prime)

2NF \rightarrow $AB \rightarrow C \checkmark$ AB is CK/SK full dependency

$C \rightarrow DE X$ C is subset of CK & DE is non-prime

By default every table is 1NF in this case as there is no table to check whether it contains multivalue or not.

Making a Relation into 2NF, 3NF, BCNF \Rightarrow

R (ABCDEF)

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

\Rightarrow Candidate keys are $= \{AB, FB, EB, CB\}$

Prime attribute $= \{A, B, C, E, F\}$

non-prime attribute $= \{D\}$

The table itself is in 1NF (As there is no value)

Now we will check if it is in 2NF

$AB \rightarrow C \checkmark$

Partial

$C \rightarrow D X$

Full

Full

Full

So it is not in 2NF

Full

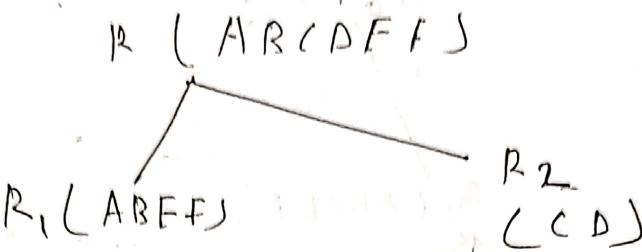
Full

Full

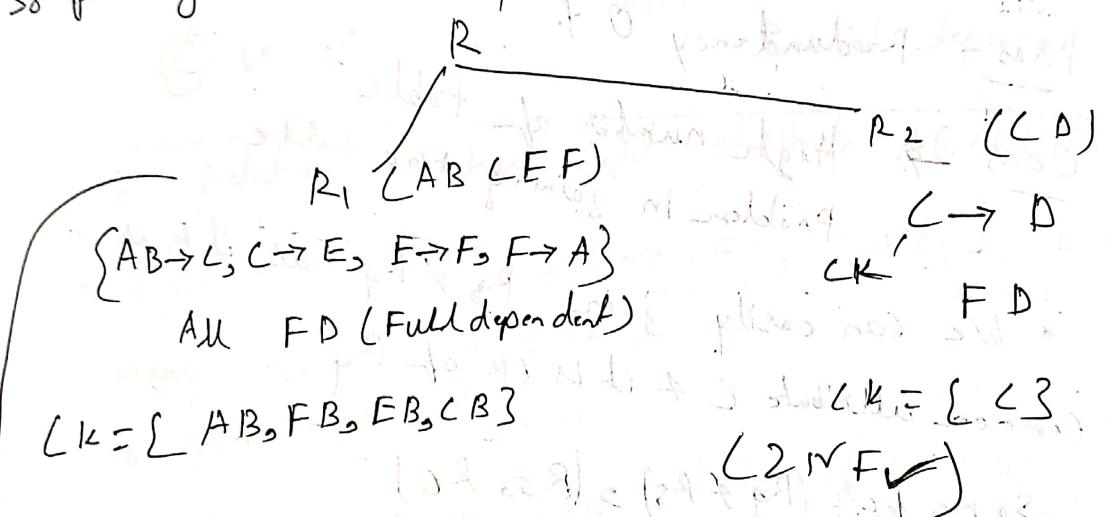
Full

To convert the table in 2NF we have to divide the table.
 And we have to make sure
 - Lossless decomposition
 - Dependency preserving decomposition

We will divide the table (Full dependency together & partial dependency).



To make it lossless decomposition we have to keep some common attribute on R_1 & R_2 . The common attribute should be CK / SK of R_1 or R_2 remain. In this case R_2 has CD it is comparatively little so finding candidate key is quite easy.



Now we will check whether it is in 3NF or not. If it is in 3NF ok otherwise we will convert.

$$A B \rightarrow C \checkmark \quad C \rightarrow E \checkmark \quad E \rightarrow F \checkmark \quad F \rightarrow A \checkmark$$

$\uparrow PA \quad \uparrow PA \quad \uparrow PA$

$\{PA \text{ (prime attribute)}\}$

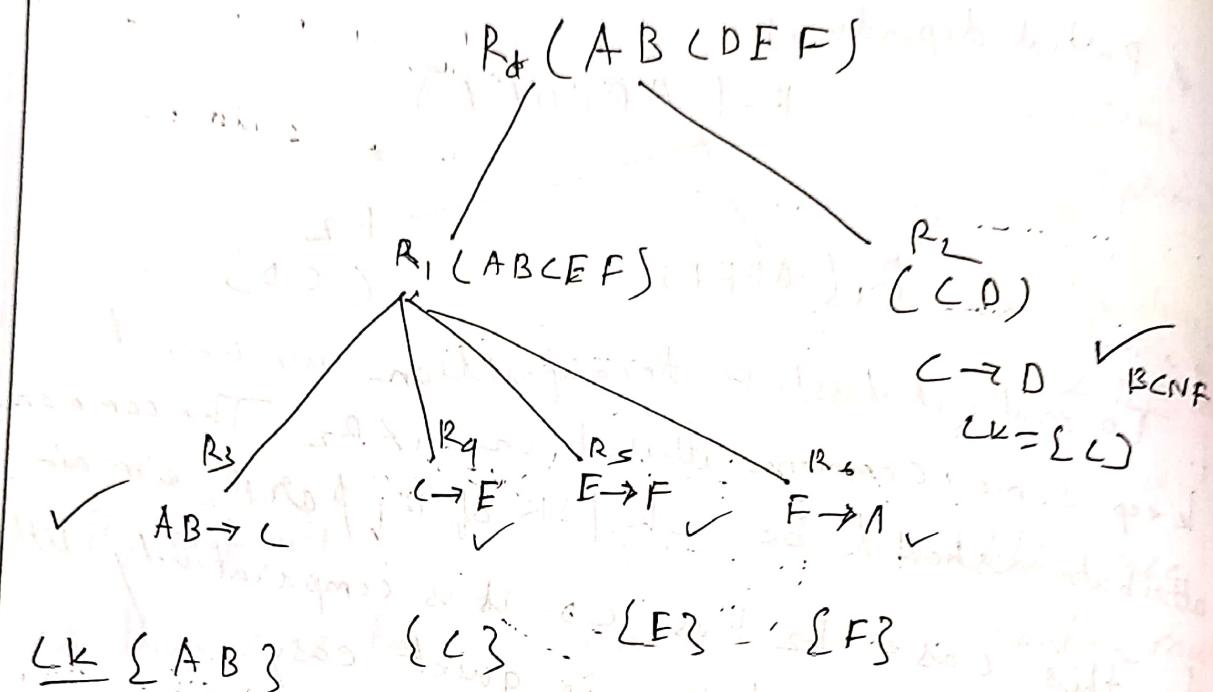
$$C \rightarrow P \checkmark$$

The table is in 3NF also

Now we will check whether it is in BCNF.
If not then we will convert it into BCNF.

$A \text{ } B \rightarrow C \checkmark \text{ (BCNF)} \quad C \rightarrow E \times \quad E \rightarrow F \times \quad F \rightarrow A \times$

$C \rightarrow D \checkmark \text{ (BCNF)}$



$$\underline{CK} \{AB\} \{C\} \{E\} \{F\}$$

Pros :- Redundancy of f.

Cons → High number of table

Problem in joining two table

- We can easily join R_3 & R_4 as it has a common attribute C & it is CK of R_4 .

Same for $(R_4 \text{ } \& \text{ } R_5) \rightarrow (R_5, R_6)$

But if we want to join R_3 & R_5 then we can't do directly as there is no common attribute. We have to take help of other attribute (R_4 in this case).

Q1 Which of the following statement is true

- (a) A relation is in 3rd NF then it is always in BCNF
- (b) A relation is in 2nd NF then it is not in 1st NF
- (c) A relation is in BCNF then it is in 2NF
- (d) A relation is in 2NF then it contains partial dependency.



\Rightarrow 2NF \rightarrow not contain any partial dependency

(c) ✓

Q2 A relation R has eight attribute AB CDEFGH

$$F = \{CH \rightarrow G, A \rightarrow BC, B \rightarrow CFH, E \rightarrow A, F \rightarrow EG\}$$

How many candidate keys in R

- (a) 3
- (b) 4
- (c) 5
- (d) 6

\Rightarrow A B C D E F G H

So D must be in Candidate key $D^+ = D$ so it is not alone

$$BD^+ = BDLFHGEA \quad CK = \{BD, AD, FD, FD\}$$

Now we will check if B is available in RHS of any FD

Yes in $A \rightarrow BC$ it is available so we will check AD

$$AD^+ = ADBCFEGH$$

Now we will check whether B is available in any other (in this case there is none). Next we will check if A is

in any RAS $E \rightarrow A$

$$S^1: ED^+ = AEDB \subset FGH$$

Now we will check whether E is available or not.

$F \rightarrow E$ in this E is in RHS

$$FD^+ = EGDAABCDFH$$

$\rightarrow (b) \checkmark$

Q1 Schema-1: Registration (rollno, courses)

non-trivial FD { $rollno \rightarrow courses$ }

Schema-2: Registration (rollno, courseid, email)

Non-trivial FD { $rollno, courseid \rightarrow email$ }
 $email \rightarrow rollno$

Schema-3: Registration (rollno, courseid, marks, grade)

non-trivial FD { $rollno, courseid \rightarrow marks, grade$ }
 $marks \rightarrow grade$

Schema-4: Registration (rollno, courseid, credit)

non-trivial FD { $rollno, courseid \rightarrow credit$ }

$courseid \rightarrow credit$

Which is 3NF but not in BCNF?

\rightarrow Schema-1: BCNF \checkmark means 3NF, 2NF, 1NF \checkmark

Schema-2: BCNF X as email is not CK

2) $rollno, courseid$ is CK
rollno is prime attribute
so obviously it is in 2NF, 1NF

Schemar-3: BCNF X
 3NF X *marked Is not CK, grade is not prime*
 2NF X *attribute*

Schemar-4: BCNF X

3NF X

2NF X

→ Schemar-2 (Ans)

Equivalence of Functional Dependency \Rightarrow

$$X = \{ A \rightarrow B, B \rightarrow C \} \quad | \quad Y = \{ A \rightarrow B, B \rightarrow C, A \rightarrow C \}$$

If X covers Y $n \geq Y$

Y covers X $n \leq Y$

then $n = Y$ *an check*

① X covers Y : we will take dependency from Y *an check*

according to X $A \rightarrow B \quad A \rightarrow C$

$$A^+ = ABC \quad | \quad A \rightarrow B \quad A \rightarrow C$$

$$B^+ = BC \quad | \quad B \rightarrow C$$

② Y covers X : we will take LHS of X and take closure from Y

$$A^+ = ABC \quad | \quad A \rightarrow B$$

$$B^+ = BC \quad | \quad B \rightarrow C$$

$$\therefore B^+ = BC$$

$$\therefore n = Y$$

$$\text{a) } X' = \{AB \rightarrow CD, B \rightarrow C, C \rightarrow D\}$$

$$Y_1 = \{AB \rightarrow C, AB \rightarrow D, C \rightarrow D\}$$

$$\Rightarrow X \text{ covers } Y: AB^+ = ABCD \quad \begin{array}{l} AB \rightarrow CL \\ AB \rightarrow DR \end{array}$$

$$C^+ = CD \quad C \rightarrow CL$$

Y covers X

$$AB^+ = ABCD$$

$$B^+ = B \quad B \rightarrow CL \times$$

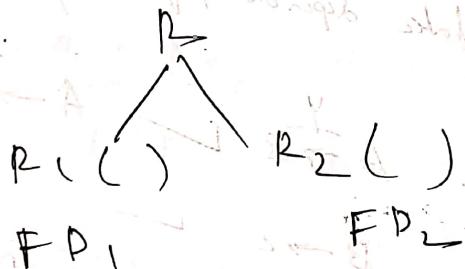
So $Y \not\models X$

Dependency Preserving Decomposition \Rightarrow

$$R(ABCD) \quad FD \rightarrow A \rightarrow B, A \rightarrow C$$

$$FD^+ \rightarrow A \rightarrow C$$

$$FD_1 \cup FD_2 = FD^+ / FD$$



a) Let $R(ABCD)$ with Functional dependency

$$\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow B\}$$

R is decomposed into $R_1(AB), R_2(BC), R_3(BD)$

with dependency will preserved



First we will determine the possible dependencies (non-trivial) and will check if it satisfies the Dependencies of R.

$R_1 (AB)$	$R_2 (BC)$	$R_3 (BD)$
$A \rightarrow B \checkmark$ $B \rightarrow A \times$	$B \rightarrow C \checkmark$ $C \rightarrow B \checkmark$ $C = CD \text{ (in } R\text{)}$	$B \rightarrow D \checkmark$ $D \rightarrow B \checkmark$ $B^T = BCD \text{ (in } R\text{)}$
We have to check if B can determine A in main dependency R		

in this case

$$B^T = BCD$$

Now we will write the union of the possible dependencies -

$$A \rightarrow B, B \rightarrow C, C \rightarrow B, B \rightarrow D, D \rightarrow B$$

Now we will check whether dependency in R' can be possible while taking closer from the possible dependencies.

$$A \rightarrow B \checkmark, B \rightarrow C \checkmark, C \rightarrow D \checkmark, D \rightarrow B \checkmark$$

∴ Dependency is preserved.

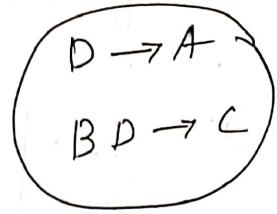
a) Let $R(ABCD)$ with Functional Dependencies

$$\{ AB \rightarrow CD, D \rightarrow A \} \text{ Decompose } R_1(AD) + R_2(BCD).$$

$R_1 (AD)$	$R_2 (BCD)$
$A \rightarrow D \times$ $D \rightarrow A \checkmark$ It is not in R or can be derived from R	We have to write only non-trivial dependencies $B \rightarrow CD \times$ $C \rightarrow BD \times$ $D \rightarrow BC \times$ $BC \rightarrow D \times$ $CD \rightarrow B \times$ $BD \rightarrow C \checkmark$

Now we will check LHS of Dependencies R and take closer from possible combination

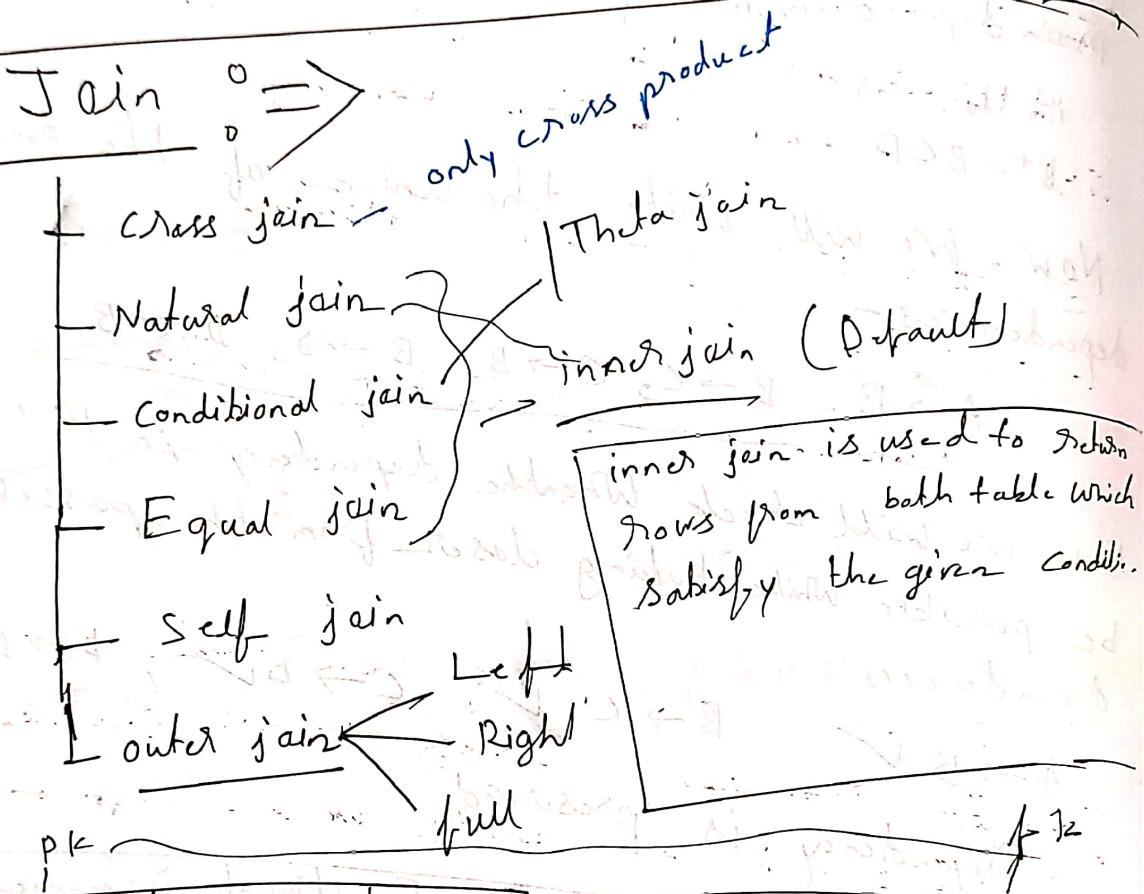
Possible combination are -



$AB^T = AB$ so we can't achieve $AB \rightarrow CD$

$D \rightarrow A$ achieved

So Dependency is not preserved.



E-no	Ename	Address
1	Aristit	Tamluk
2	Sayan	Ghatal
3	Amisha	Balurghat
4	Sorangshae	Karca

Deptno	Name	Eno
D1	CS	1
D2	IT	2
D3	Health	3
D4	BTS	4

Suppose we just want to know the address of a Employee of named Arjit then we can get it only from Employee table

Select Address from Employee Where Ename = 'Arjit' »

But when we want to display Employee name who is working in Health department.

⇒ we can't get answer with just only 1 table but we have to use two tables.

Here we use join easily. We can also do with subquery but it is complex.

Atleast one common attribute needed for Join

Join = Cross product + condition.

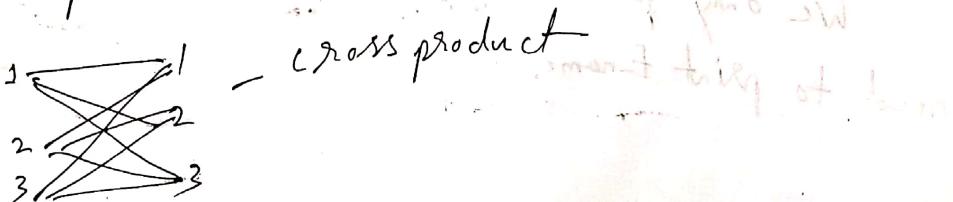
Natural Join \Rightarrow

Whenever there is any query firstly we have to find How many table we need to give the result

if there is only one table we use simple command like Select

If there is two table needed we will use cross product

Cross product is done using table1, table2



(x) A natural join is a type of join operation that creates an implicit join by combining tables based on columns with the same name and datatype.

- There should be one common attribute.
- There is no need to specify the column names to join.
- The resultant table always contains unique values.
- The resultant table always contains unique columns.
- It is possible to perform a natural join on more than two tables.
- Do not use on clause.

Syntax: $\text{Select } \underset{\text{Actual}}{\text{column names}} \text{ from }$
 $\text{Table1 Natural Join Table2;}$

Explanation: \Rightarrow Actually What happens.

$\Rightarrow \text{Select E-name from employee Department Where}$
 $\text{Employee-E-no} = \text{Department-E-no;}$

\downarrow
 it will first cross multiply 2 table there will be 16 rows but will select only where E-no are same.

There is only 4 column where E-no are same.
 we only print E-name as select command only need to print E-name.

Self Join \Rightarrow

Used to join a table to itself as if the table were two tables temporally renaming at least one table in the SQL statement.

Syntax: Selection a.column_name, b.column_name ...

From table1 a, table2 b

Where a.common_field = b.common_field;

Referencing a student table's fk
primary key

S-id	C-id	Since
S1	C1	2016
S2	C2	2017
S1	C2	2017

Referencing a course table's primary key

it is an intermediate table

We have to find student id who is enrolled in at least two courses \rightarrow We will use Self Join to solve this

Select S-id from study as T₁, study as T₂

Where T₁.S-id = T₂.S-id and T₁.C-id \neq T₂.C-id

What if no student is enrolled?



Equal Join \Rightarrow ... combine table based on
(Equi) matching values in specified columns.

- The column names do not need to be same.
- The resultant table contains repeated column Emp can perform on more than 2 table.

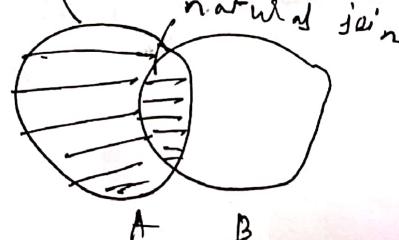
E-no	E-name	Address	Dept_no	Locatn.	E-no
1	Ram	Delhi	D ₁	Delhi	1
2	Axi	WB	D ₂	Karca	2
3	Ani	Karca	D ₃	Baniga lak	3
4	Priya	Japan			4

Suppose we want to find emp name who worked in a department having location same as their address

\Rightarrow Select E-name from Emp, Dept Where
Emp.Eno = Dept.Eno and Emp.address
= Dept.location

Left outer Join \Rightarrow it gives the matching rows and the rows which are in left table but not in right table

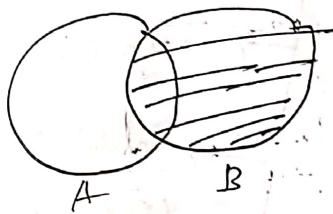
Any outer Join = Natural join + Something



Syntax - Select column-names from table1
 left outer join table2 On {table1.common attri =
 table2.common attri}

Condition

right outer join \Rightarrow it gives the matching
 rows and the rows which are in right table
 but not in left table.



Right

Select columns from table1 Right outer joins table2
 on {table1.common = table2.common}

Full outer Join \Rightarrow Left outer join + Right
 outer join or
 Matching values + only available on table1
 + only available on table2.

+ only available on both tables

Table1	Table2	Table1 & Table2
1	1	1
1	2	1
2	1	1
2	2	1

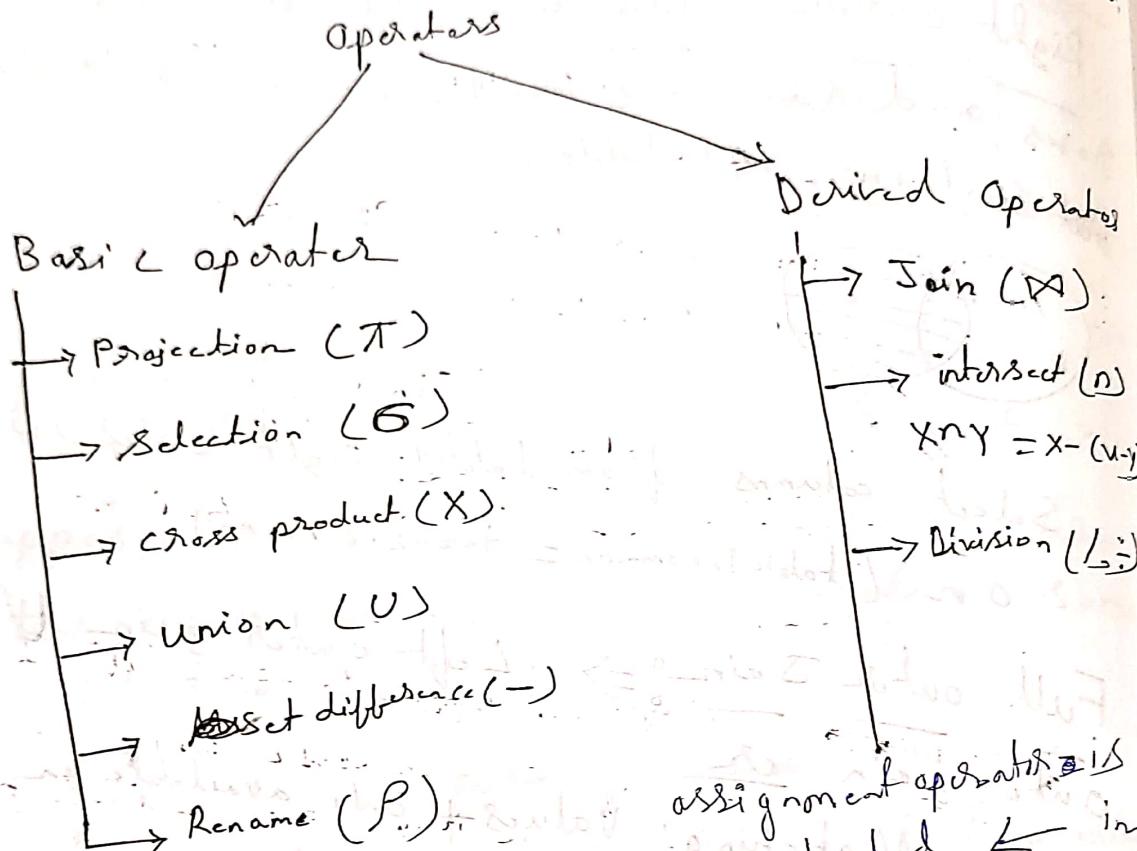
Final output of below query

What do we see about the result?

Relational Algebra \Rightarrow E.F coded 1970

It is also called Procedural Query Language.
 What to do? \nearrow How to do it?

Which takes relation as input & give relation as output
 Formal query language



Projection \Rightarrow Project some columns.

Student		
Rollno	Name	Age
1	Arijit	18
2	Anisha	19
3	Sayan	22

If we want to fetch Rollno then we will

write $\Pi_{\text{Rollno}} (\text{Student}) \rightarrow$ Show only Rollno column
 table name

$\Pi_{\text{Rollno}, \text{name}} (\text{Student}) \rightarrow$ Show only Rollno, name.

Projection always give distinct (unique result)

Name
A
B
A

$\Pi_{Name} (\text{student})$

It will only give A distinct elements

Selection (σ) : \Rightarrow Select rows based on given condition

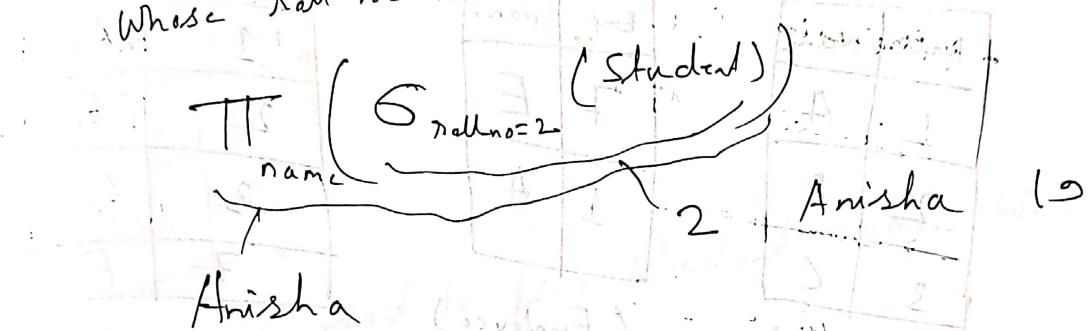
If we want to retrieve the name of student

whose roll no. is 2

whose roll no. is 2 means Projection

\Rightarrow Retrieve the name means Selection

whose roll no is 2 = Selection



We always use Π as last because it always

show distinct values so we can loose any value.

if name is same then we may loose desired data,

Projection = column

Selection = rows / tuples

Cross Product (X) \Rightarrow cross product of relations

returns $m \times n$ rows where m and n are

numbers of rows in R_1 & R_2

$m+n$ - number of columns

A	B	C
1	2	3
2	1	4

C	D	E
3	4	5
2	1	2

R_1

R_2

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

Union (\cup) \Rightarrow tuples which are in R_1 or in R_2 .

Max no of rows returned = $m+n$

min no of rows returned = $\max(m, n)$

- No of columns should be same on both tables in order to do union
- Domain of every column should be same.

Roll	Name
1	A
2	B
3	C

(Student)

Emp_no	Name
7	E
1	A

(Employee)

Roll	Name
1	A
2	B
3	C
7	E

(Student) \cup (Employee)

- By default the attribute name of union will be first table attribute name.
- There will be no duplicate values.

$(\pi_{name}(\text{Student}) \cup \pi_{name}(\text{Employee}))$

Name attribute is

A

B

C

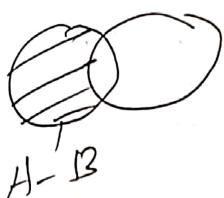
E

- * $(\text{Student}) \cup (\text{Employee})$ give the result age order of data is matched

numerical, char \cup numerical, char

else it will give error

Set Difference $\Rightarrow A - B = \text{available in } A \text{ but not in } B$



- no of columns must be same
- Domain of every column must be same

$(\text{Student}) - (\text{Employee})$

roll	name
2	B
3	C

Suppose we want to find student names who are not employee

$(\text{tt_name}(\text{Student}) - \text{tt_name}(\text{Employee}))$

Rename(P) \Rightarrow Renaming a relation to another relation.

Intersection(\cap) \Rightarrow Returns those tuple which are in both R_1 & R_2

$$\text{Max Rows} = \min(M, n)$$

$$\text{Min Rows} = 0$$

Conditional Join (\bowtie_C): Selection from two or more tables based on some condition

$\text{Cross product + condition.}$

Equi Join (\bowtie) \Rightarrow Special case of Condition join when only equality condition are applied.

Left outer join \rightarrow $\Delta\bowtie$

Right outer join - $\bowtie\Delta$

Full outer join - $\Delta\bowtie\Delta$

Division operator $\Rightarrow A (X, Y) / B (Y)$ returns X values for that there should be tuple $\langle x, y \rangle$ for every y value of relation B

or
 A / B returns those tuple in A which are associated with every tuple of B .

note \rightarrow Attribute of B should be a proper subset of attributes of A .

Sid	cid
S ₁	C ₁
S ₂	C ₁
S ₁	C ₂
S ₃	C ₂

cid
C ₁
C ₂

Enroll_id

(E)

Query - Retrieved sid of students who enrolled in every (all course)

\Rightarrow

$E(S_{\text{sid}}, C_{\text{id}}) / C(C_{\text{id}}) = S_1$ (as every student is enrolled in every course of id)

- It is an derived operator.

$$\text{lab of } \Pi_{\text{sid}}(E) \times \Pi_{\text{cid}}(C) \Rightarrow S_2 \times C_2$$

it means every student is enrolled in every course

now we will use set difference (-) to know the students who are not enrolled on all courses.

$$\text{lab of } \Pi_{\text{sid}}(E) \times \Pi_{\text{cid}}(C) - (E)$$

S_2

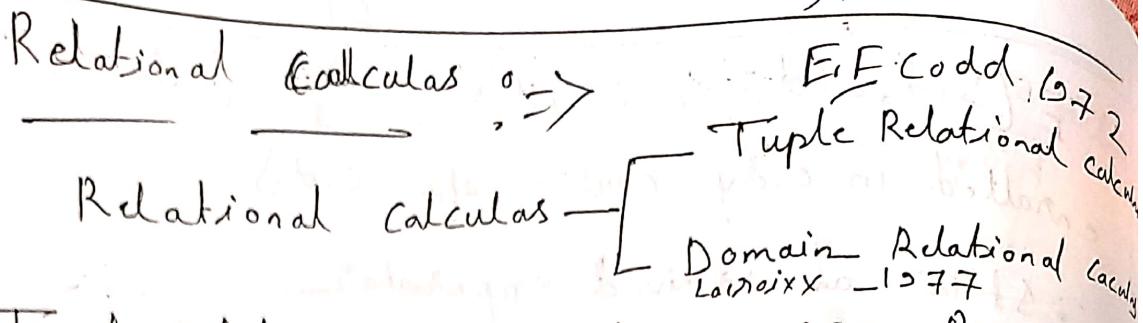
S_3

now we get the students who are not enrolled in every course now we will just minus from students of Enrolled

$$\Pi_{\text{sid}}(E) - \Pi_{\text{sid}}((\Pi_{\text{sid}}(E) \times \Pi_{\text{cid}}(C)) - (E))$$

$$\downarrow$$

S_1



Tuple Relation calculus is a non Procedural language. (We only say what to do)

Procedural language \leftarrow What to do (Language) \leftarrow How to do

- A query is expressed as,

$\{ t \mid P(t) \}$ where $t =$ resulting tuples & $P(t)$ is known as predicate and these are condition that are used to fetch it. if it gives all tuples if $P(t)$ is true.

Operations

AND - \wedge

OR - \vee

NOT - (\neg)

If also use quantifiers

- $\exists t \in r(Q(t)) \Rightarrow$ there exist a tuple in t in relation r such that predicate $Q(t)$ is true.

ex - $\exists t \text{ age} > 18$
 if there is any tuple where $age > 18$ then it will give true

$\forall t \in r(\text{Q}(t))$ - gives true if all tuples follows $a(t)$

ex- $\forall t (\text{age} > 20)$

If all ages > 20 then it will give true.

Unsafe expression \Rightarrow

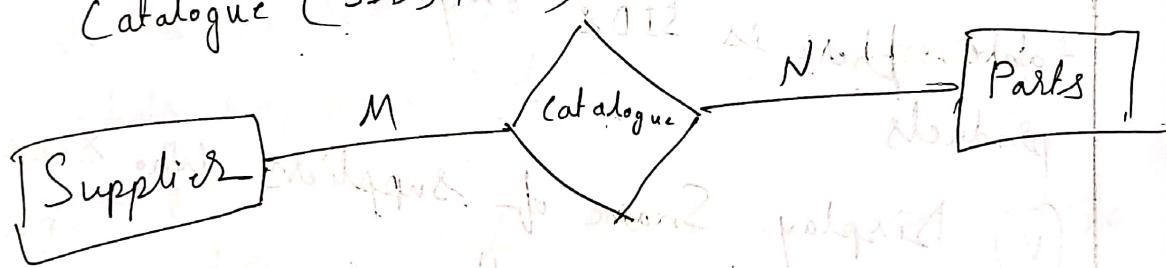
$\{ S.\text{name} \mid \neg \text{Supplier}(S) \}$

it is unsafe as it is telling to print names which are not in supplier table. There is infinite name. So it is unsafe.

(*) Supplier (SID, Sname, address)

Parts (PID, Pname, color)

Catalogue (SID, PID)



(1) Write a query in SQL to display Sname of suppliers?

$\Rightarrow \{ S.\text{Sname} \mid \text{Supplier}(S) \}$

We make variable S of Supplier table now we will check all the tuples and print Sname.

(ii) Display Pname of parts whose color is Red

$$\Rightarrow \{ \$p.pname \mid \text{Parts}(p) \wedge p.color = "Red" \}$$

(iii) to display SID of suppliers whose name is "Arjith" and address is "tamiluk"

$$\Rightarrow \{ s.SID \mid \begin{array}{l} \text{Supplier}(s) \wedge s.Sname = "Arjith" \\ \wedge s.address = "tamiluk" \end{array} \}$$

(iv) Display SID of suppliers who supplied some parts

$$\Rightarrow \{ c.SID \mid \text{Catalogue}(c) \}$$

We can get this with the help of catalogue table. There is SIDs only which supply some products.

(v) Display Sname of suppliers who supplied some parts pre variable

$$\Rightarrow \{ s.Sname \mid \begin{array}{l} \text{Supplier}(s) \wedge \exists c \in (\text{catalogue}(c)) \wedge \\ s.SID = c.SID \end{array} \}$$

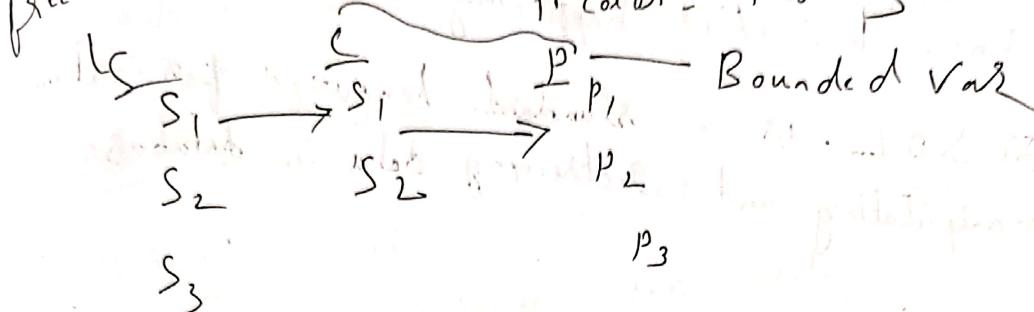
Here we need two tables {Supplier} and {Catalogue}

We use \exists that exist which checks if any

Supplier exist in catalogue where $s.SID = c.SID$

(vi) Display Sname of Suppliers who supplied some red color parts

$\Rightarrow \{ S.Sname \mid \text{Supplies}(S) \wedge (\exists c) (\text{Catalogue}(c) \wedge (\exists p) (\text{parts}(p)) \wedge S.sid = c.pid \wedge p.color = "Red" \}$



SQL (Structure Query Language)

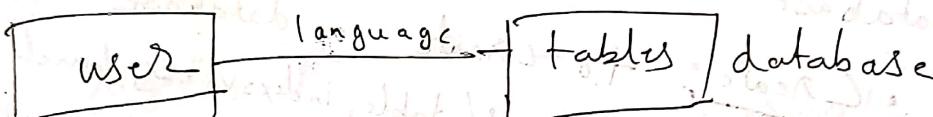


Table is stored in database in a structured way that's why we called it Structure Query language.

EE Codd published Relational model in 1970 with Relational Algebra and Tuple relational Calculus (TRC).

Previously it is called SEQUEL

(Simple English Query Language)

Later it is named SQL

- SQL is a domain specific language as it is only used in dbms / is used in various places.
- SQL is declarative language ~~as~~ non procedural language. We just say what to do, we don't know how it is happening inside.
- ④ SQL is a standard language for storing, manipulating and retrieving data in database.

DDL ⇒

Data Definition language which deals with database Schemas (tables)

- Create: To create a database and its objects (table, index, view, function, ...)
- Alter: Alters the structure of the existing database.
- Drop: Delete the whole structure.
- Truncate: Remove all records from a table, including all the spaces.
- Rename: Rename the table or object.

- DML \Rightarrow Data Manipulation languages used to store, modify, retrieve, delete and update data.
- Select: retrieve data from a database
 - Insert: insert data into table
 - Update: update existing data within the table.
 - Delete: Delete record
 - Merge: combining all three main statement
(insert, update, delete)

DCL \Rightarrow Control Language mostly concerned with rights, permission

- Grant — allows users access privileges
- Revoke — withdraw user's access given by using grant

TCL \Rightarrow Transaction Control language deals with transactions

- Commit: commit a transaction
- Rollback: rollback a transaction in case of error
- Savepoint: save the points in between transaction

Constraints \Rightarrow Condition in attribute value.

- Unique: All the values in a attribute is unique. No two data can't match.

- Not Null: Mandatory field. It can't be empty.

- Primary key: Unique + not null

- Check: checking values in time of entering

- check($age > 18$), if user enters 10 at 19 , it will show a message instantly and will say to enter the correct age.

- Foreign key: key which is taking reference from primary key

- Default: if user does not enter any value then by default it will take a value.

- Suppose i does not enter dob, it will take default "01.01.1990".

(Q) We can use multiple combination of constraints.

Create Database :-

Used to create new SQL database

Syntax

`Create database database-name;`

Drop Database :-

Used to drop an existing SQL database

`Drop database database-name;`

Create Table :-

Used to create a new table in a database

Syntax: `Create table table-name`

(Column1name datatype,

Column2name datatype,

Column3name datatype

) ;

like- `Create table emp`

(id int,

name varchar(20),

Salary number(10)

) ;

To know the details of the column we use.

descriptions `desc tablename;` `desc emp;`

60

Drop table \Rightarrow used to drop (delete) an existing table in a database.

Drop table tablename;

Truncate table: \Rightarrow used to delete the data inside a table but not the table itself.

Truncate table tablename;

Alter table: \Rightarrow used to add, delete or modify column names in an existing table.

- used to add & drop various constraints on an existing table.

- Add column
- Modify datatype
- Add constraints
- Rename column/table.
- Remove Column
- Modify datatype length
- Remove constraint

Ex-

• Alter table tablename

Add columnname datatype;

• Alter table tablename

Drop column column_name;

• Alter table tablename

modify column_name datatype;

Alter table tablename

filename column oldname to newname

Alter table tablename rename to
newname;

Alter table tablename

add primary key (column-name);

Alter VS Update =>

Alter is Data Definition Command

Update is Data Manipulation Command

Alter can change add, remove the
table structure (column's name, constraints)

update can change only the data inside
columns.

like update emp

Set salary = salary * 2

Delete VS Drop VS Truncate =>

Delete is DML command but Drop, Truncate
is DDL command

Delete is used to delete rows.

Drop is used to delete the entire table including
schemas.

Truncate just clear the record / data in
the schema but the structure remains
unchanged.

Delete can delete specific rows according to condition but truncate delete rows entirely. Table locking the data.

There is option of rollback in delete before committing as it keeps a log file. Slower than truncate.

There is no option in rollback in truncate. It is faster.

Select \Rightarrow Used to select data (columns) from database.

Syntax: Select column₁, column₂, ... from tablename;

If we want to select all fields then use asterisk (*)

Select * from emp;

Select distinct \Rightarrow Used to return only distinct (different) values

Syntax: Select distinct column₁, column₂, ... from tablename;

Where \Rightarrow Used to filter records in terms of condition

Syntax: Select column₁, column₂, ... from tablename where condition;

operators:

$=$: Equal

$<$: Less than

$>$: greater than

\geq : greater than or equal

\leq : less than or equal

\neq : Not equal

\neq : Not equal.

AND \rightarrow display record if all conditions are true

OR \rightarrow if any of the conditions is true

Not \rightarrow if the condition is not true.

Roll	Name	Section
1	Arijit	A
2	Sayan	B
3	Anisha	C
4	Priyanka	A
5	Soumen	B

Select Section from student:

O/P - Section

A

B

C

-A

B

Select distinct Section from student:

Select Name from student

Where Roll = 3

O/P - Name
Anisha

Order by : \Rightarrow Used to sort the result set in Ascending or Descending order
By default it sort Ascending order.

Syntax - Select column₁, column₂ -

from table-name

Order by column₁, column₂, ... Asc/_{DE}

Ex - Select * from student

order by name ; // Asc
by

Select * from student

order by name Desc ;

Insert into : \Rightarrow Used to insert new Records
in a table

Syntax : • Insert into table-name

Values (value₁, value₂, ...) ;

• We have to maintain the order of column
and its value)

• Insert into table-name (column₁, column₂, ...)
Values (value₁, value₂, ...) ;

Ex: Insert into student

Values (6, 'Sabit', 'A') ;

Null Value \Rightarrow It is not possible to test for null values with comparison operators such as $=$, \neq , $<$, $>$.
We will use IS NULL or IS NOT NULL

Syntax - Select column names
from tablename

Where columnname IS NULL ;

Ex - Select * from student

Where section IS NOT NULL ;

Update \Rightarrow Used to modify existing table's record

Syntax : update tablename
Set column1 = value1, column2 = value2
Where condition ;

Ex - update student

Set name = "Salonghae"

Where roll = 3 ;

Delete \Rightarrow Used to delete existing record in a table

Syntax : Delete from tablename ; // delete all rows.

Delete from tablename

Where condition ; // delete rows that matched the condition.

Select top :> used to specify the numbers of record to return.

Syntax - Select top number ~~of column~~
from table-name
Where condition

* Select column-names
from table-name
Where condition
limit number;

* Select column-names
from table-name
Fetch First number rows only;

* Select column-names
from table-name
Where Rownum <= number;

Ex * Select top 3 * from student;

* Select * from student
limit 3;

* Select * from student
Fetch first 3 rows only;

Aggregate Function: \Rightarrow NULL is not counted
 in a aggregate function
 \Rightarrow return the smallest value
 (i) Min(): \Rightarrow return the smallest value
 of selected column (numeric)

Syntax: Select min(column-name)
 from table-name;

Ex- Select min(Salary)
 from emp;

(ii) Max(): \Rightarrow return the largest value
 of the selected column (numeric)

Syntax: Select max(column-name)
 from table-name;

Ex- Select max(Salary)
 from emp;

(iii) Avg(): \Rightarrow return the average value
 of the selected column (numeric)

Syntax: Select Avg(column-name)
 from table-name;

Ex- Select Avg(Salary)
 from emp;

(iv) Sum(): \Rightarrow return total sum of a numeric
 column

Syntax: Select sum(column-name) from table-name;

① Count(s) \Rightarrow returns the number of rows that matched the specified criteria.

Syntax: Select count(column-name)
from table-name;

ex- Select count(name) from student;

IN \Rightarrow allows you to specify multiple values in a where clause.

- Starts and for multiple or

Syntax: Select column-names
from table-name

Where column-name IN (Value₁, Value₂, ...)

• Select column-names
from table-name

Where column-names IN (Select Statement₁)

ex- Select * from students

Where section IN ('A', 'B');

Between \Rightarrow selects values within given range

Value can be numbers, text / dates

- it is inclusive: both start and end value is included

Syntax - Select column-names from table-name
Where column-name Between value1 and

Like \Rightarrow Used in a where clause to search for specified pattern in a column.

- The percentage sign (%) represent zero, one or multiple characters
- The underscore ($_$) represent one or multiple single character

Syntax: Select column names
from tablename

Where columnname like pattern;

Patterns -

- $a\%$ \Rightarrow Values start with a
- $a_a\%$ \Rightarrow Values end with a
- $a_a\%$ \Rightarrow Values
- $a_a.c\%$ \Rightarrow Values that have ac in any position
- $-a_a\%$ \Rightarrow Values that have a in 2nd position
- $a_a_a\%$ \Rightarrow Values start with a at least 2 characters in length
- $a_a_a\%$ \Rightarrow Values start with a at least 3 characters in length
- $a_a_0\%$ \Rightarrow Starts with a and ends with 0.

Join \Rightarrow Used to combine rows from two or more tables based on related columns between them.

Inner Join $\xrightarrow{\text{by default}}$ Select records that have matching values in both tables

Syntax - Select column names

from table1

Inner join table2

On table1.column name = table2.column name

Don't preserve
non-matched
tuples

Cx- Select orders.orderID \rightarrow customers.customerName
from orders

inner join customers

on orders.customerID = customers.customerID

Left/Right/Full outer Join \Rightarrow

Left = common rows + left in left table only (table 1)

Right = common rows + left in right table rows (table 2)

full = common rows & left in both tables

Select column names

from table1

Left/Right/full join table2

on table1.column name = table2.column name

- Union \Rightarrow Used to combine the result
 of select command
- there should be same number of columns
 - columns must have similar data type.
 - must in same order

Syntax — Select column-names
 from table1
 Union

Select column-names from table2;

- (*) Union — select only distinct value
 . union all — select duplicate value.

Group By \Rightarrow Group rows that have same
 value (identical data)

Syntax — Select column-name
 from tablename
 group by column-name

- (*) We can only use the same column name in select, but can use aggregate function of other column.

- (*) Where is used in every row but in group by we use having

Having \Rightarrow Where clause place condition on select statement but having clause places on groups created by group by clause

Syntax - Select column-names
from table
where condition
group by column-names
Having condition; \rightarrow aggregated functions

Ex - Select count(customerid) as country
from customers
group by country
having count(customerid) > 5;

Aliases \Rightarrow used to give a stable temporary name
AS - keyword is used.

Syntax - Select column-name as alias-name
from tablename
Select column-names
from tablename as alias-name;

Subqueries \Rightarrow Emp

Eid	Ename	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	40000
5	Vaishnavi	IT	50000

- ① Display Max salary from emp table
 \Rightarrow first we have to know for find the result which table is needed
 We will use aggregate function

Select max(salary) from emp;

50000 (o/p)

- ② Display employee name who is taking maximum salary
 \Rightarrow We will use nested query

Select Ename from emp Where salary

= (Select max(salary) from emp)
 inner query

In this first inner query is executed
 then the outer query will execute.
 Where salary = 50000 (inner query result)

(3) Display 2nd highest salary from cmp table

⇒ Suppose first we get the highest salary, then we will exclude it then again will find the highest salary

④ Always use dummy table

Select max(salary) from cmp where salary < $(\text{select max(salary) from cmp})$

First inner query will execute then the outer

④ Display name who is taking 2nd highest salary

⇒ We will just have to write a outer query for name

Select Ename from cmp where salary

= $(\text{select max(salary) from cmp where salary} < \text{(select max(salary) from cmp)})$;

⑤ Display all the departments name along with number of employees working with them?

⇒ first we will use group by for gathering identical data then use count() aggregate function

Select dept, count(dept);

from emp	HR 2
group by dept;	MRKT 2
	IT 1

- ⑥ Display all the dept names | Where no of employees less than 2

\Rightarrow Select dept
from emp
group by dept
having count(*) < 2;

- ⑦ Display employee name Where no of employees less than 2

\Rightarrow Select Ename from emp Where dept in (Select dept from emp group by dept having count(*) < 2);

- ⑧ Display highest salary department wise and name of emp who is taking that salary

\Rightarrow Select Ename from emp Where salary in (Select max(salary) from emp group by dept);

Eid	Ename	Address
1	Ravi	Chd
2	Vazun	Delhi
3	Nitin	Pune
4	Robin	Bangalore
5	Ammy	Chd

Project

Eid	Pid	Pname	Location
1	P ₁	IoT	(Bangalore)
5	P ₂	Big data	Delhi
3	P ₃	Retail	Mumbai
4	P ₄	Android	Hyd

Emp

- ⑨ Details of employee whose address is either Delhi or Chd or Pune

⇒ Select * from emp where Address in ('Delhi', 'Chd', 'Pune')

- ⑩ find the name of employee who are working on a project

⇒ Select ename from emp where eid in (Select distinct eid from project)

- ⑪ Find the details of emp who is working on at least one project?

⇒ Select * from emp where eid exists (Select eid from project where emp id = project id);

(*) In nested query we do first inner query then compare with outer query but in correlated nested query first we execute outer query then search on inner query. It is top to down approach.

In nested query inner query just executed one time but in correlated nested query inner query checked several times.

- Exist / not exist always return true or false checking condition

Correlated Subquery \Rightarrow

• It is a subquery that uses values from outer query

Top to down approach

In (1) we use correlated subquery.

Here we can use exist, not exist, Any, all operator

for every outer row the inner query will execute every time.

1 row selected - if will selected by outer query then it will be checked with inner query then 2nd row will be selected and so on.

(*) Correlated Delete update also happens.

Nested Subquery vs correlated Subquery vs Joins

- Nested Subquery: Bottom up approach

Ex- Select * from emp Where cid in
 (Select cid from dept)

inner query execute first only on time.

- Correlated Subquery: Top down approach
 comparison number high

Select * from emp Where exists

(Select eid from dept Where emp.cid=dept.cid)

- Joins: cross product + condition

Select * from emp, dept Where
 emp.cid = dept.cid;

cid	name
1	Arijit
2	Sayan
3	Anisha
4	Priyanka
5	Suvamay

emp

dno	dname	cid
D1	HR	1
D2	IT	2
D3	MRKT	3
D4	IT	5

dept

Find the detail of all employee who works
 in any department

Find Nth highest Salary \Rightarrow

We can solve it with the help of
Correlated query

Select id, salary from cmp c1

Where $N-1 = (\text{Select count}(\text{distinct salary})$

from cmp c2 Where $c2.\text{salary}$
 $> c1.\text{salary})$

id	salary
1	10000
2	20000
3	20000
4	30000
5	40000
6	50000

cmp

id	salary	
1	10,000	$> 30000 \times$
2	20,000	$> 30000 \times$
3	20000	$> 30000 \times$
4	30000	$> 30000 \checkmark$
5	40000	$> 30000 \checkmark$
6	50,000	$> 30000 \checkmark$

c2 count = 2

Suppose we want 3rd highest $\rightarrow 30000 \checkmark$

$$N-1 = 3-1 = 2$$

- ④ We recognize it to be correlated query by
we use outer query value (c1) in inner
query.

Q | You need to display last name of employees who have 'A' as second character in their name.

(a) Select last_name from emp where last_name like '_ - A %';

(b) Select last_name from emp where last_name = '_ * A %';

(c) Select last_name from emp where last_name like '_ , A %';

\Rightarrow (a) ✓

Q | A command to remove relations from SQL Database

(a) Delete table tablename

(b) Drop table tablename

(c) Erase table tablename

(d) Alter table tablename

\Rightarrow Remove relation means remove SQL database table

We use Drop command (DDL)

Q1 In the following Schema R (a_1, b)

a₁: Select * from R

a₂: (Select * from R) intersect (Select * from R)

a₃: Select distinct * from R

a₁, a₂, a₃ produce same result

(a) a₁, a₂, a₃ produce same result

(b) only a₁, a₂ produce same result

(c) only a₂, a₃ produce same result

(d) a₁, a₂, a₃ produce different result

$\Rightarrow a_1$ select * from R \Rightarrow

a	b
1	1
2	2
3	3
2	3

a	b
1	1
2	2
3	3
2	3

a	b
1	1
2	2
3	3

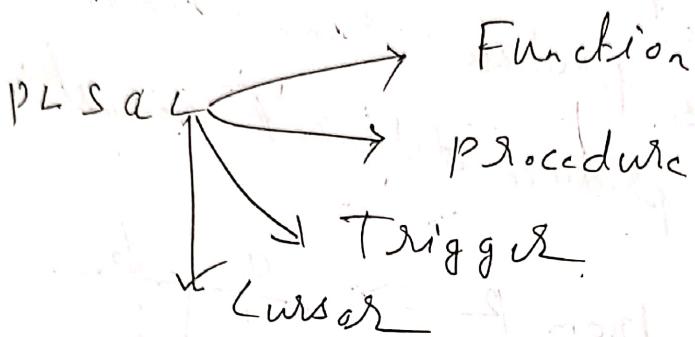
(c) ✓

PL SQL (Procedural SQL) \Rightarrow

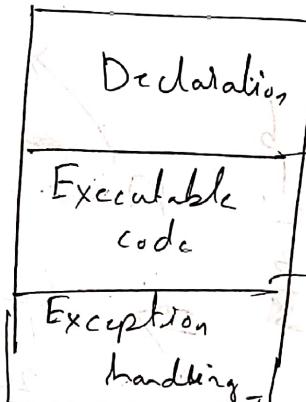
SQL is declarative/non procedural — only what-to-do is said.

but PLSQL is procedural SQL
what-to-do we use function here

How to do



PLSQL
Block
code



a int. b int c int

adding number

declare

a int;

b int;

c int;

begin

a := 10;

b := 20;

c := a + b;

end

Transaction =>

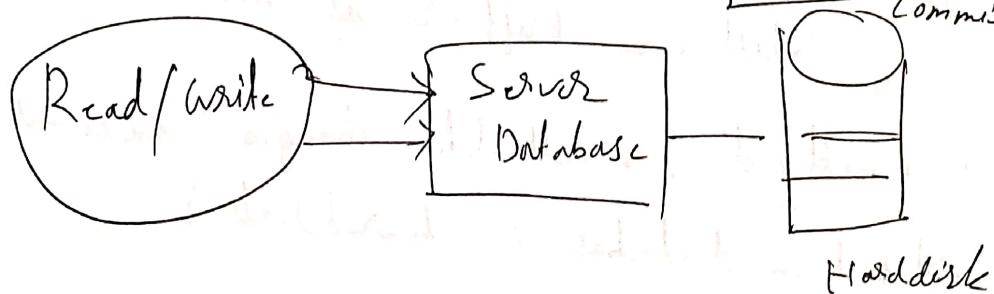
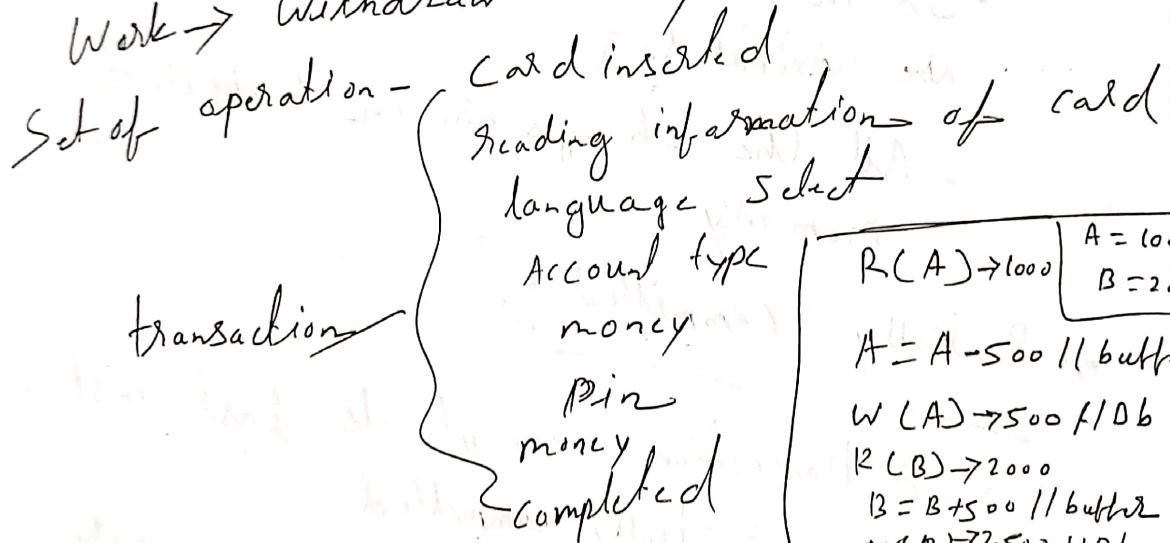
Transaction is a set of operation used to perform a logical unit of work. It generally represent change in database.

Operations =>

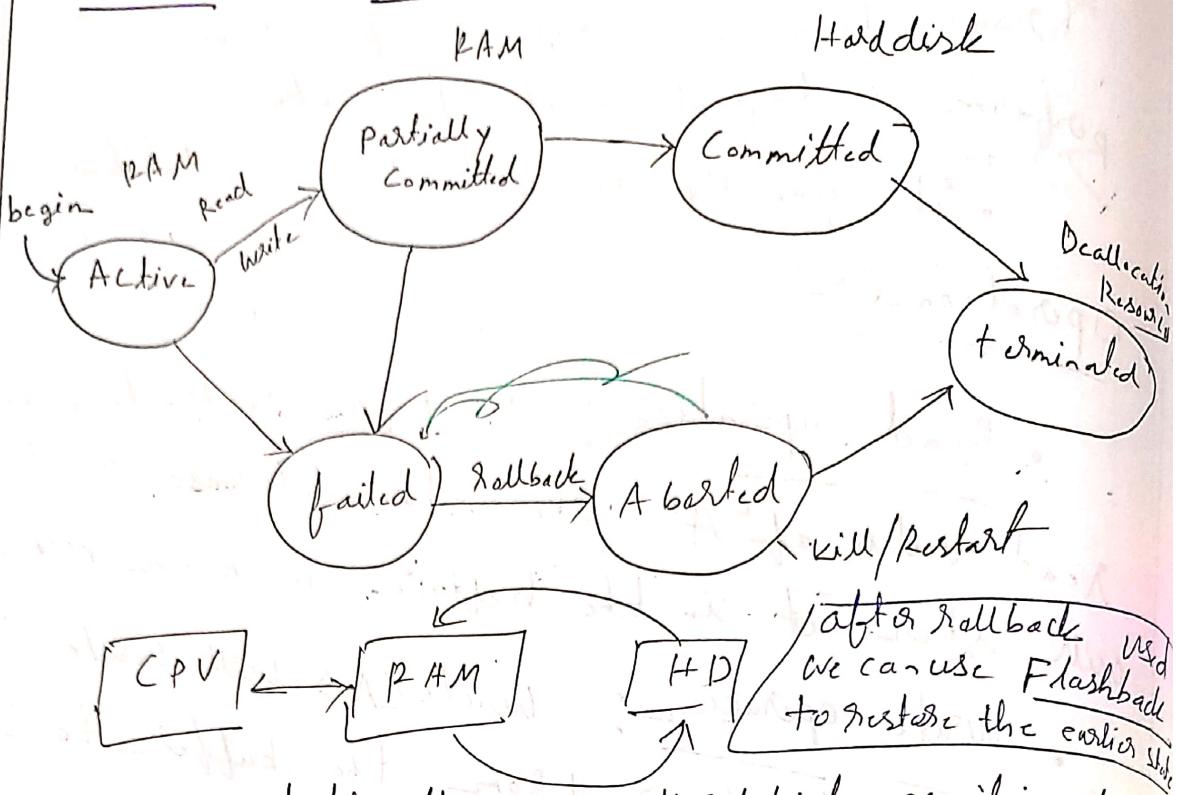
- Read operation: Read(A) instruction will read value of A^3 from the database and will store it in the buffer in main memory.
- Write operation: write(A) will write the updated value ' A^3 ' from the buffer to the database.

Ex Suppose we want to withdraw money then we use ATM transaction.

Work \rightarrow Withdraw money



Transaction States \Rightarrow



CPU can't directly access Harddisk as it is slow
That's why we use buffer in main memory

Active State:

- Is in active state as long as instructions are executed.
- All the changes are in buffer of main memory

Partially Committed:

- A transaction execute its last instruction it is in Partially committed
- Still in buffer of main memory
- Committed:
 - All the changes successfully stored in database (harddisk)

Failed: When transaction is in active or partially committed state and some failure occurs (power off, back button click) then it is in failed state.

Aborted: if transaction failed then recovery manager roll back all of its write operations to the database and bring the database to original state. Recovery module select one of two operation

- Restart the transaction
- Kill the transaction

ACID Properties of transaction =>

To maintain consistency of database certain properties are followed during transaction

ACID → Isolation
Consistency
Atomicity
Durability

Atomicity: Either the transaction occurs completely or it does not occur at all.

No transaction occurs partially

Consistency: The database must be consistent before and after the transaction.

Before and after the transaction the sum of money should be same.

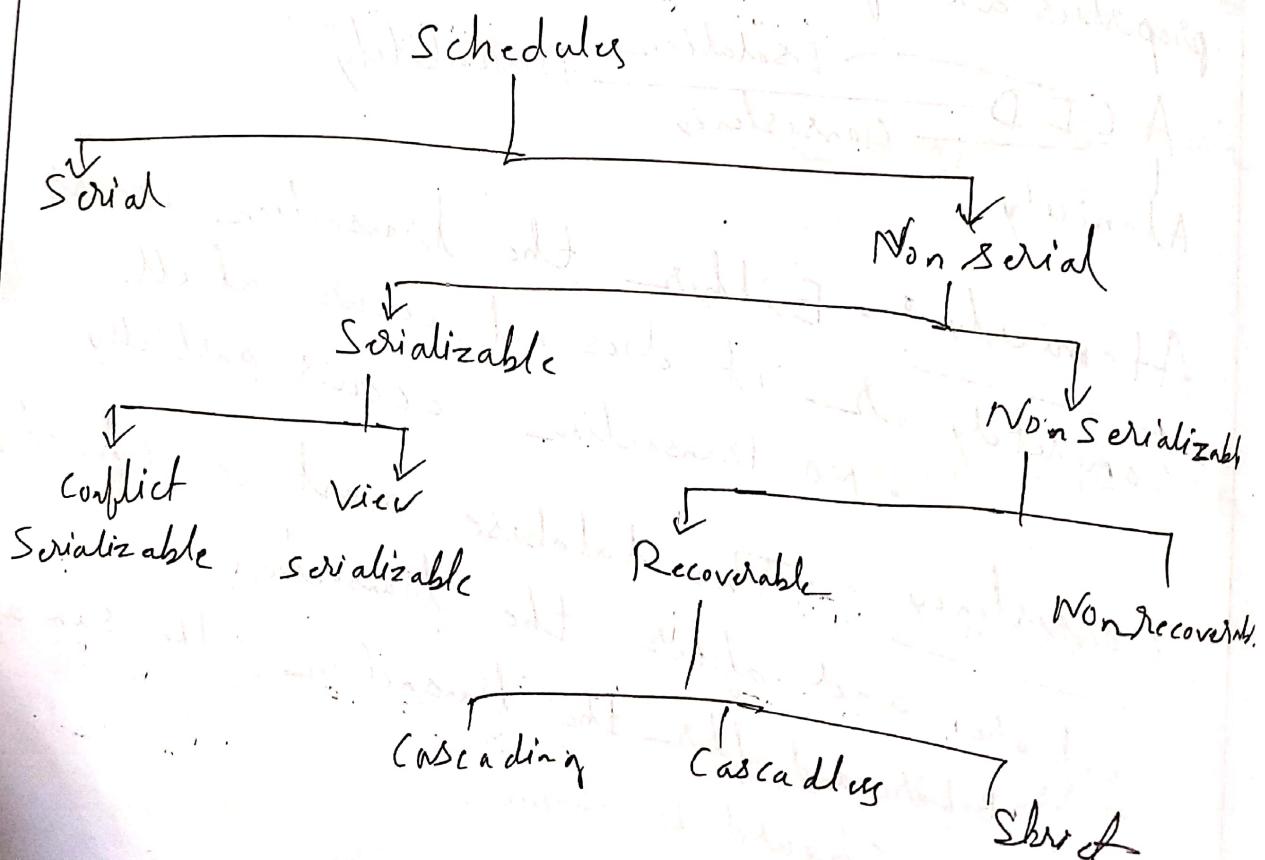
Isolation =>

- Multiple transaction occurs independently without interference
- changes occurring in a particular transaction will not be visible to other until it committed/written to memory.

Durability =>

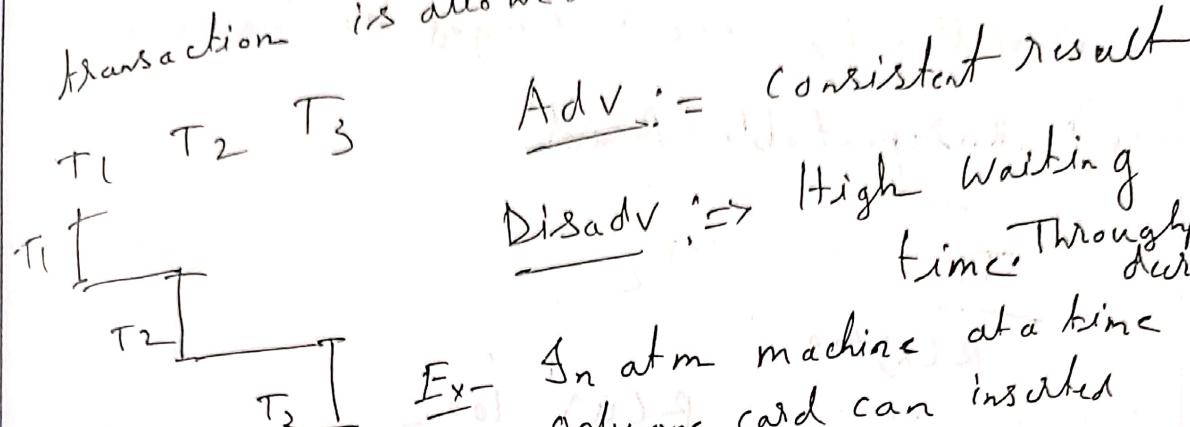
- If a transaction committed then it can't be lost no matter what.
- if there is failure, but changes will not be lost if committed.

Schedules => The order in which the operation of multiple transaction appear for execution



Serial Schedule:

- All the transaction execute serially one after the other.
- When one transaction executes no other transaction is allowed to execute.



Non Serial / Parallel \Rightarrow Multiple transaction execute concurrently.

Adv \rightarrow High throughput & performance

Disadv - not consistent

Ex- Online banking, IRCTC ticket booking

Types of problem in Parallel

① Dirty Read :

T₁

R(A) 60

W(A) 60-10

Rollback

↑ failure

T₂

R(A) 50

Dirty Read

W(A) 50-10

if reads

commit

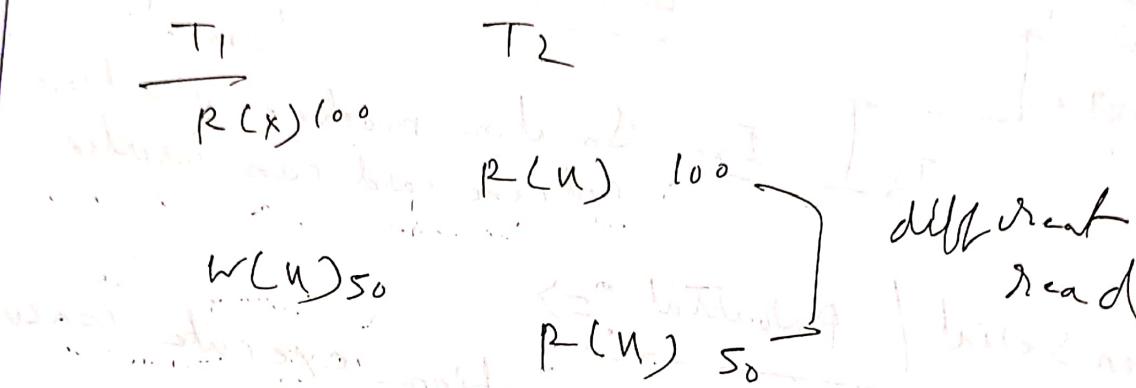
wrong value

as transaction is roll backed

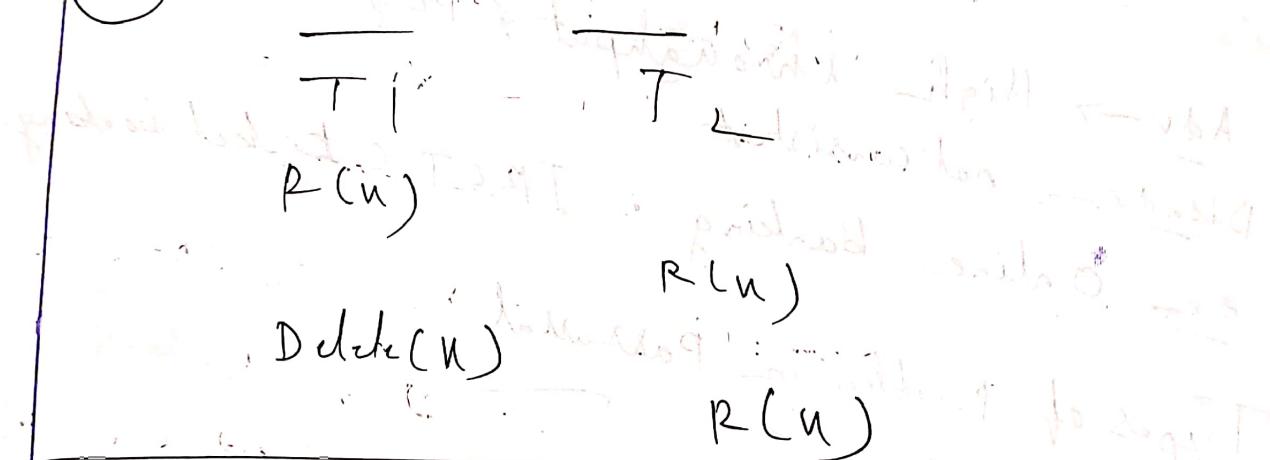
(ii) incorrect summary :- in between a transaction a other transaction come, Read, Commit an aggregate function

(iii) Last update :- older changes are overwritten by newer changes by transaction.

(iv) Unrepeatable Read :



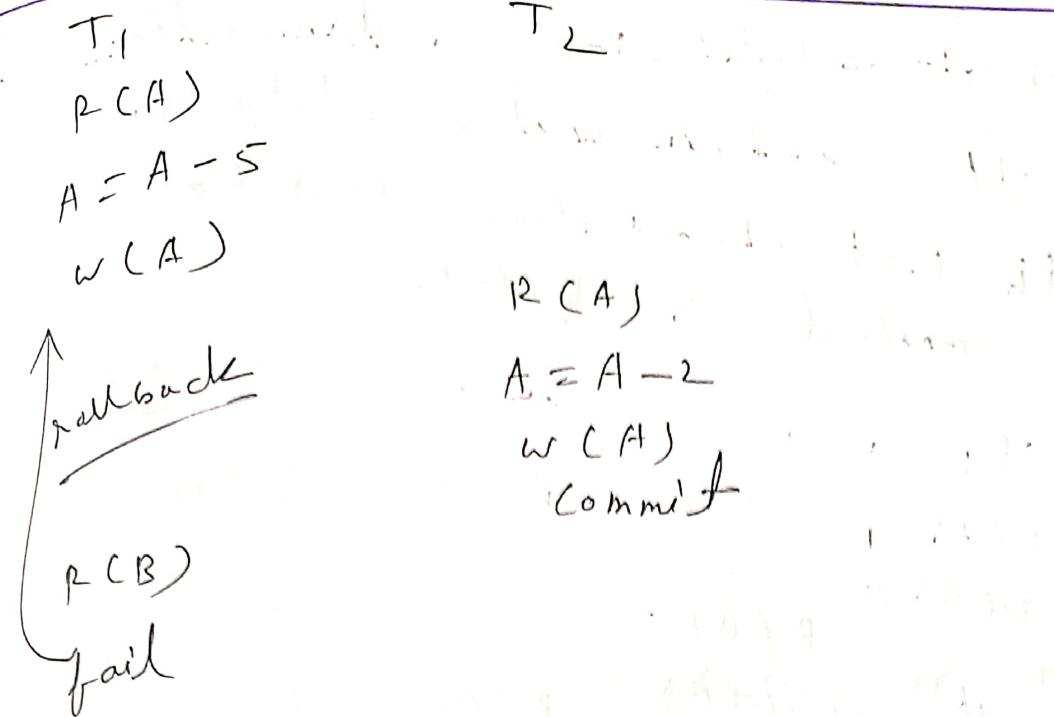
(v) Phantom Read



Non Recoverable Schedule

If in a schedule

A transaction perform dirty read operation from an uncommitted transaction & commit before the transaction from which it read dirty read.

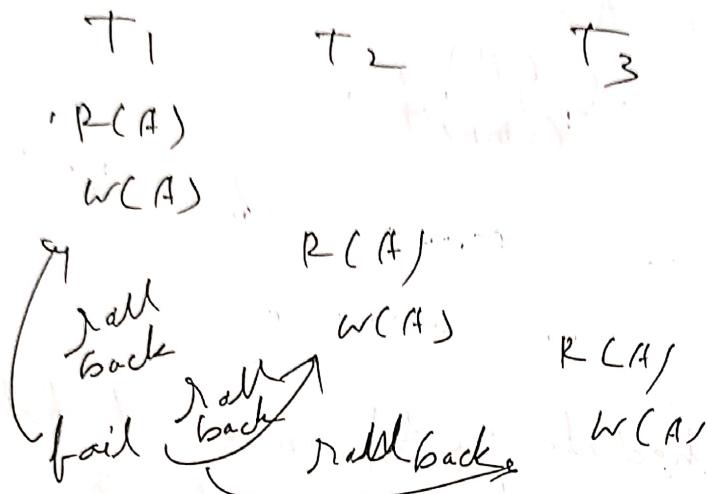


Recoverable Schedule \Rightarrow If in a Schedule

- A transaction perform a dirty read operation from an uncommitted transaction.
- Its commit operation is delayed till the uncommitted transaction either commit or roll back. It is called Recoverable Schedule
- Cascading Schedule: failure of one transaction cause several dependent transaction to roll back or abort. Big - CPU performance low no helpful
- Cascadeloss Schedule: transaction is not allowed to read a data until the last transaction that has written it is committed & aborted. (but we can write) this is also problem

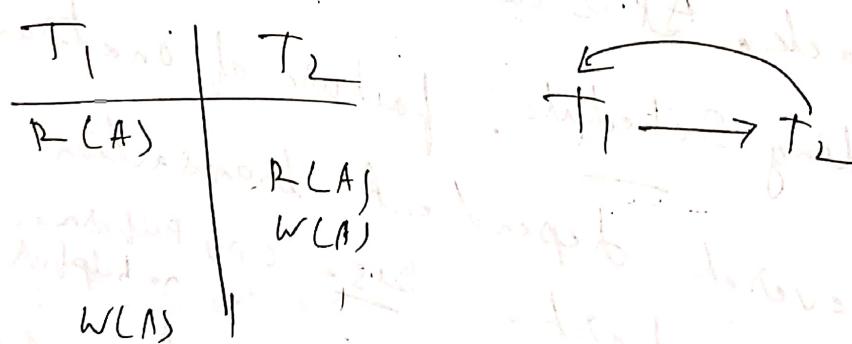
Analogy - Copying others answer in exam hall

Strict Schedule : \Rightarrow A transaction neither read nor write a data until the last transaction that has written is committed or aborted.



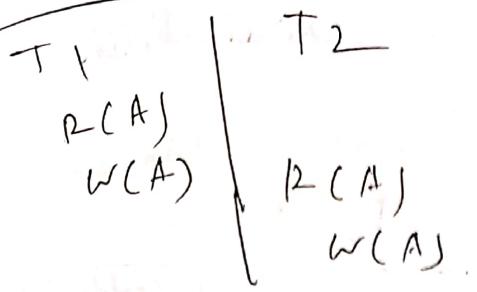
Serializability : \Rightarrow

helps to identify whether a non serial schedule can be made serial or not to maintain consistency of data.



This is non-serial. To make it serial we have to do $T_1 \rightarrow T_2$ or $T_2 \rightarrow T_1$.

We have to find a clone schedule which is serial.



$T_1 \rightarrow T_2$
This is done transaction
of previous

Types -

① Conflict Serializability

non serial can be converted into serial
by swapping its nonconflicting operation.

R(A) S R(A) — Non conflict

R(A) W(A) }
W(A) R(A) } conflict

W(A) W(A) → consider for addition

R(B) R(A) R(A)
W(B) W(A) } non conflict
R(B) W(B)

W(A) S

T₁ | T₂

R(A)
W(A)

R(A)

W(A)

R(B)

T₁ | T₂

R(A)

W(B)

R(B) W(A)

non conflicting

R(B) W(A)

Swap

again

R(B) R(A)

W(A)

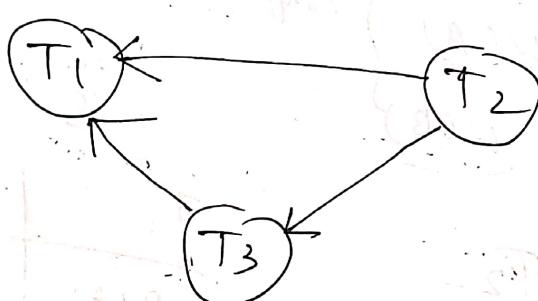
Precedence graph to check Serializability

S is a schedule which has 3 transactions $\rightarrow \Rightarrow$

T_1	T_2	T_3
$R(x) \&$		
		$R(y)$
		$R(u)$
	$R(y)$	
	$R(z)$	
		$w(y)$
	$w(z)$	
$R(z)$		
$w(u)$		
$w(z)$		

Conflict pair
 R-W
 W-R
 W-W
 or Same var

- Number of vertices $\phi =$ number of transaction
- check conflict pair in other transaction and draw edges



- $R(x)$ has a conflict pair $w(x)$ but there is no $w(u)$ in T_2 or T_3
- now we will check $R(y)$ of T_3 (we have to check sequentially)

$R(y)$ also has no conflict pair

• Next $R(u)$ of T_3 has a conflict pair
of $w(u)$ of T_1
So we will draw an edge $T_3 \rightarrow T_1$

• Next $R(y)$ of T_2 has a conflict pair with
 $w(y)$ of T_3 $T_2 \rightarrow T_3$

• Next $R(z)$ of T_2 has a conflict pair
with $w(z)$ of T_1

• Next $w(y)$ of T_3 has no conflict pair

• Next $w(z)$ of T_2 has a pair with $R(z)$

• Next $w(z)$ of T_1 already drawn so no

$T_2 \rightarrow T_1$ already drawn so no
need to draw.

note - don't check conflict pair
in own transaction.

(*) We have to check any loop or cycle
cycle. If there is no loop or cycle
then it's conflict serializable
if there is any loop then it is not
serializable

loop / cycle - if we start from T_1 & whether

we can return to T_1

- No need to cover all nodes b.wt

we have to back starting point.

94

Serializable Possibility Sequence

$T_1 \rightarrow T_2 \rightarrow T_3$ $T_3 \rightarrow T_1 \rightarrow T_2$

$T_1 \rightarrow T_3 \rightarrow T_2$ $T_3 \rightarrow T_2 \rightarrow T_1$

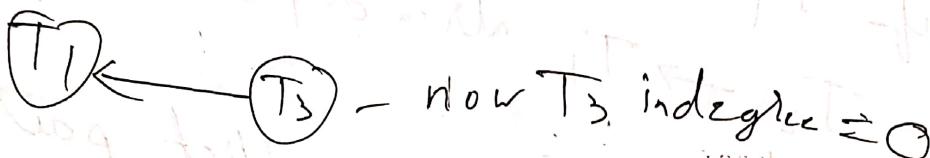
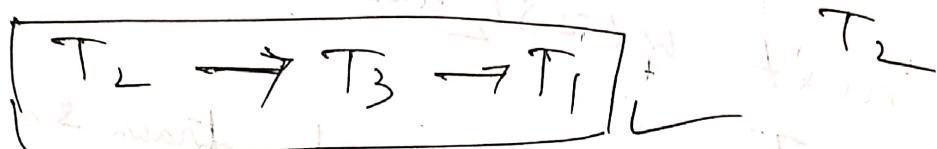
$T_2 \rightarrow T_1 \rightarrow T_3$

$T_2 \rightarrow T_3 \div T_1$

but we will choose from the graph on
the basis of in degree = 0

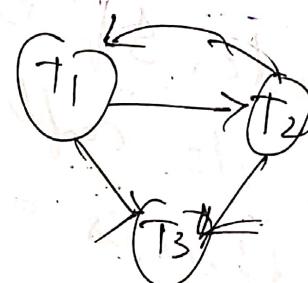
no \rightarrow is going to that vertices.

in this case T_2 in degree = 0 then we will choose



Check whether Schedule is conflict
Serializable

	T_1	T_2	T_3
$R(A)$			
$W(A)$			
$W(A)$			

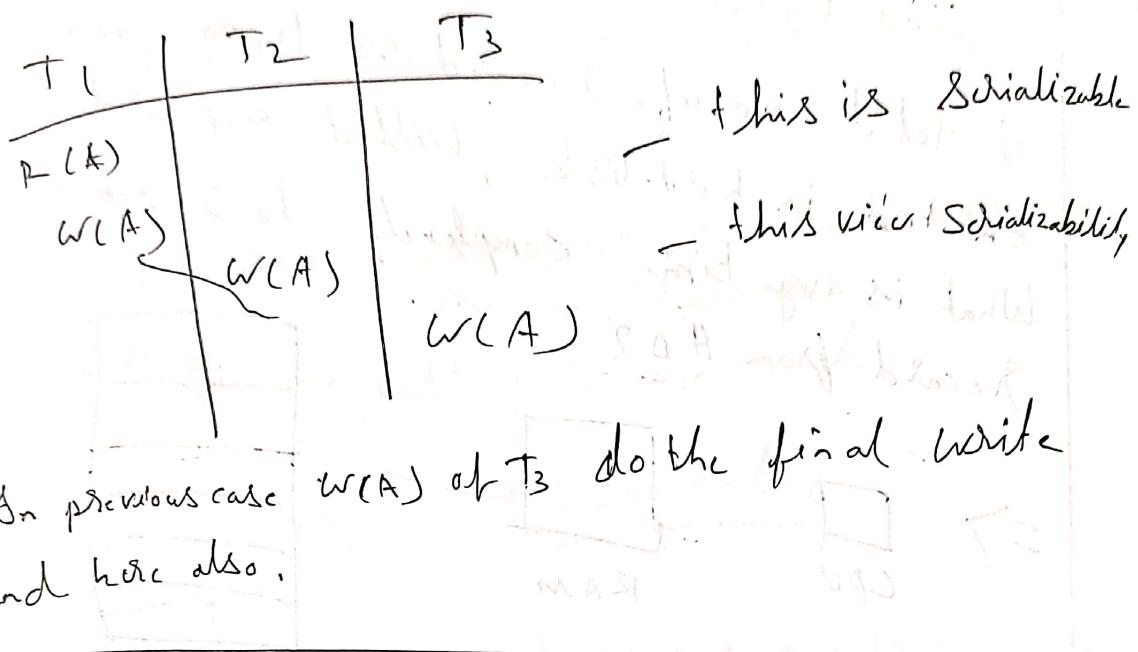


$w(A)$ so there is no loop
~~as cycle~~

if there is no loop then we can say this
is Serializable.

But if there is loop, then we can't say that it is not serializable. This is not conflict serializable, But it can be serializable.

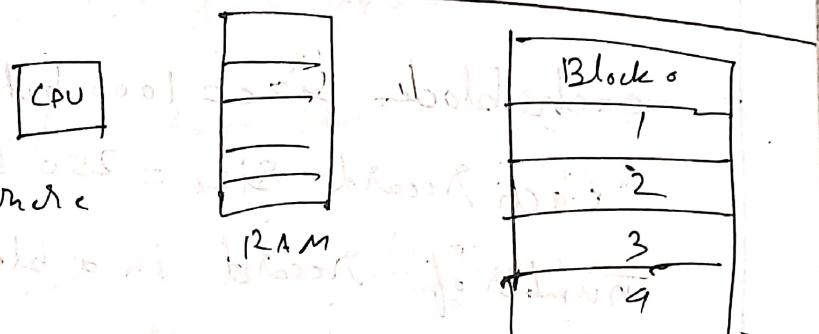
That's why we use Victr Serializability to check whether it is serializable or not.



Indexing \Rightarrow

Select * from stu where

roll no = 1;



(CPU can't directly communicate with Harddisk as it is slow (Harddisk))

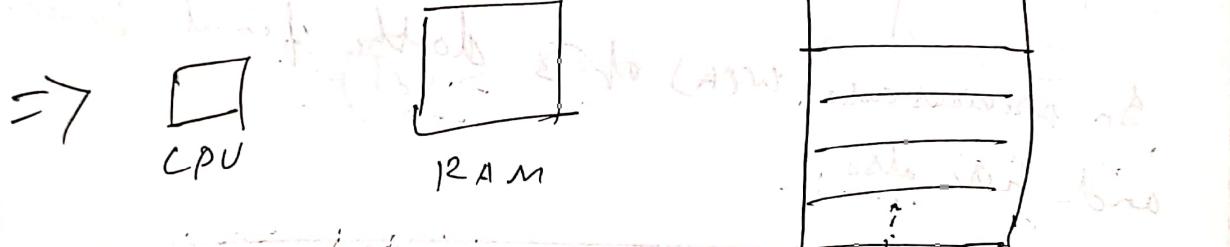
First the stu file will come to RAM, then CPU will check for roll no = 1 data.

So we need to maintain index so that we can take the file of stu at very minimum time. else we have to take each block and search whether our required data is in that or not.

* Indexing reduce to cost (as Number of transferring block from harddisk to Ram is minimum)

Ex → searching topic on book without index vs with index.

Q1 Consider a harddisk in which block size = 1000 bytes. each record is of size = 250 bytes. if total no of records is 10,000 and data entered in harddisk without any order. What is avg time complexity to search a record from H.D?



each block size = 1000 byte

each record size = 250 bytes

$$\text{number of record in a block} = \frac{1000}{250} = 4$$

$$\therefore \text{Total record of block} = \frac{1000}{4} = 250$$

Best case = 1. time : we get required file

worst case = 250. time (Linear Search)

$$\therefore \text{Avg } \cancel{\text{time}} = \frac{250}{2} = 125 \text{ Search}$$

if it was ordered then $\log_2^n = \log_{10}^{250}$

$$\text{Binary Search} = 12$$

What is avg time complexity to search a record from Index table of index table entry of 20 byte (key + pointer) of 10 B

\Rightarrow index also stored in secondary table.

\Rightarrow in index table there is key (20 B, asc/nan) and pointer (Page no/block no)

each block of harddisk can have = $\frac{1000 \text{ byte}}{20 \text{ byte}} = 50$

index directly

Dense \rightarrow same number of entry in index with number of entry in harddisk

Sparse \Rightarrow one entry for one block.

\Rightarrow 10000 entry \rightarrow 2500 entry

\oplus index is ordered (binary search) ~~unique(key)~~

So ~~avg~~ we need to take block in ram in case of Dense

~~avg~~ Total blocks needed = $\frac{10000}{50} = 200$

~~avg~~ $\log_2 200 \leq 8 + 1$ (for page taking + ram)

\Rightarrow Search

In case of Sparse

~~avg~~ $\log_2 50 + 1 = 6 + 1 = 7$ search

Types of indexing

(i) Primary index order

(ii) clustering index unordered

(iii) Secondary index key non-key

Primary (Sparse)	clustered (Sparse)
Second ary (Dense)	Second ary

Primary index \Rightarrow

Roll Name
1 A
2 B
3 C
4 D
5 E
6 F
7 G
8 H
9 I
10 J

pointer key

1
2
3
4
5

- it is sparse
- ordered
- one entry for one block

Average number of block access using

$$\text{index} = \log_2 N + 1 \quad N = \text{number of block}$$

in index table

clustered index

Dept Name
1 A
2 B
3 C
3 D
4 E
4 F
5 G
5 H
5 I
5 J

pointer key

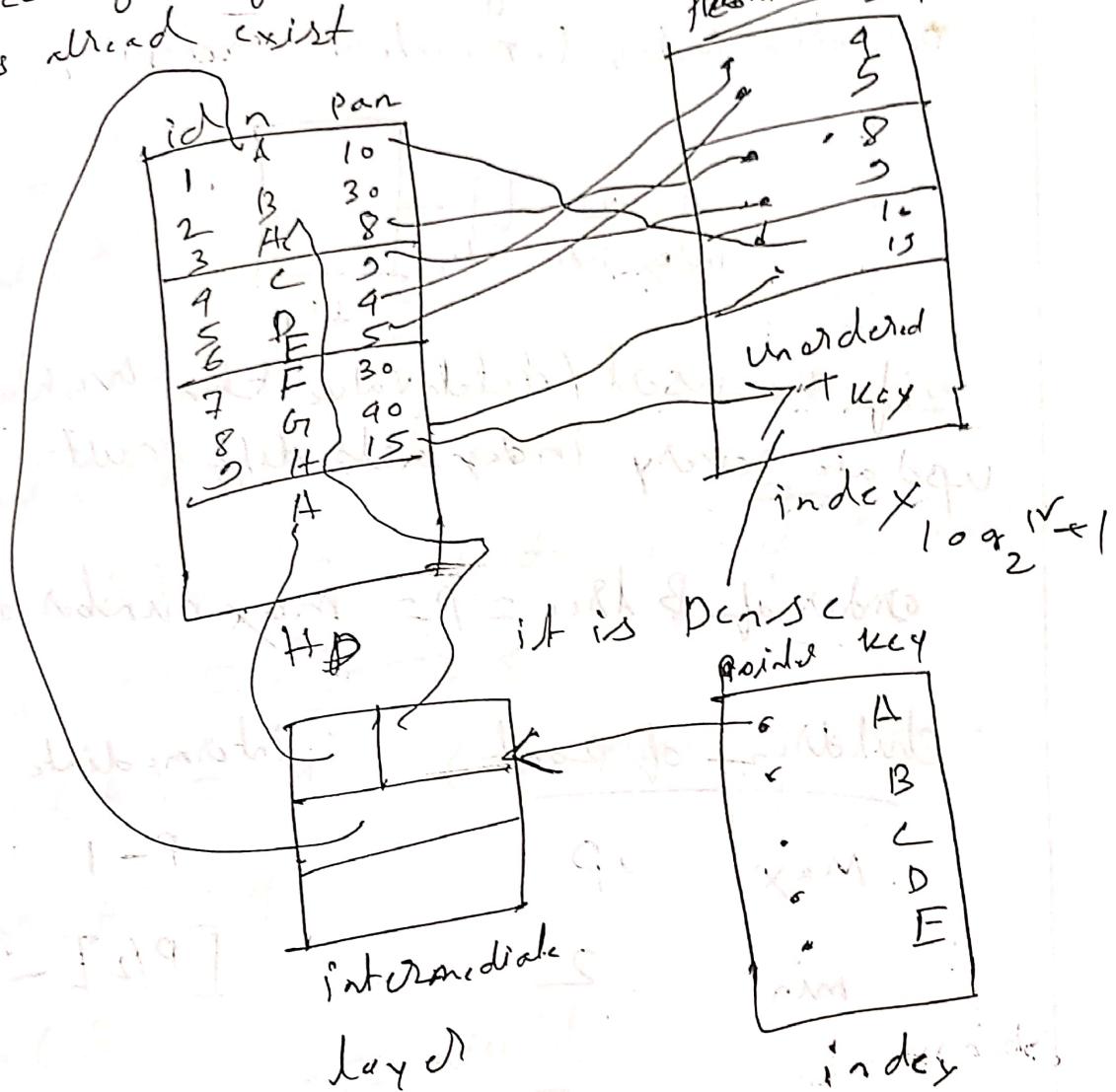
1
2
3
4
5

- it is sparse
- ordered
- one entry for one block

$$\text{access} = \log_2 N + 1 + \text{no of block range}$$

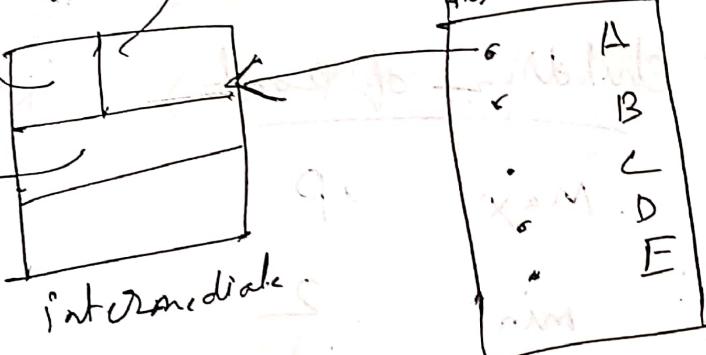
30 95

Secondary index \Rightarrow
 Secondary index provides secondary means
 of accessing a file for which primary
 access already exist



$$\log_2^{N+1}$$

it is described
pointer key

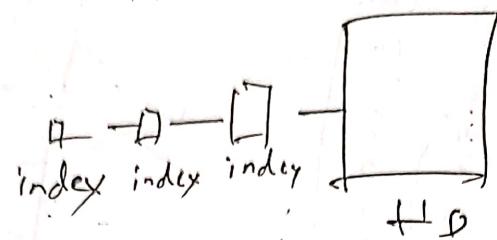


$$\log_2^N + 1 + \text{other index}$$

block
pointer

B tree \Rightarrow we use balance tree
for multilevel index

if index is big then we maintain
an other index to maintain index



if we insert / delete value then we have to
update every index. its difficult

order of B tree = p = max number of child.

children of root

Max

P

intermediate node

$P - 1$

min

2

$\lceil \frac{P}{2} \rceil - 1$

max children = order = p

$\therefore \text{key} = P - 1$

Record Pointer = $P - 1$

A	B
1	null
2	null
3	null
4	null
null	null

$\text{Count}(*) = 5$ number of rows (it count null)

$\text{Count}(A) = 4$ (it does not count null)

$\text{Count}(B) = 0$ (count of null value is 0)

$\text{Sum}(A) = 10$ (does not include null)

$$\text{Avg}(A) = \frac{10}{4} = 2.5$$

$\text{Min}(A) = 1$ (as null means no value)

$\text{Max}(A) = 9$

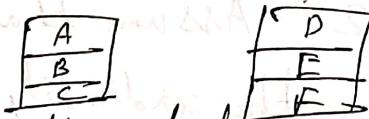
$\text{Sum}(B) = \underline{\hspace{2cm}} = \text{null}$

$\text{Min}(B) = \underline{\hspace{2cm}}$

$\text{Max}(B) = \underline{\hspace{2cm}}$

RAID - Redundant array of independent disks

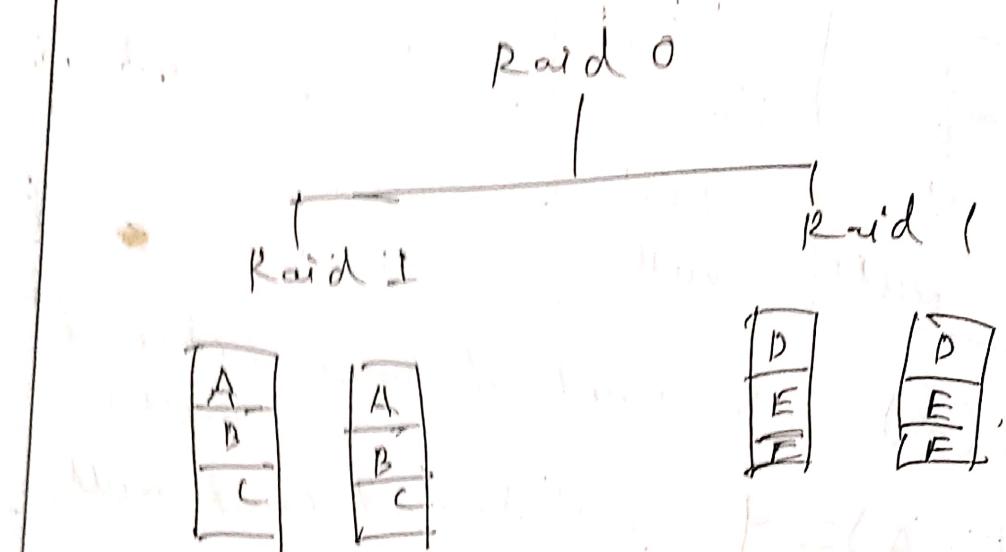
RAID 0 → different data in different disk



RAID 1 → mirror data. Copy data



Raid 0-1-1 : we splitting of missing



SQL function

Single row

Round()

Lower()

Upper()

Multi row

Sum()

Avg()

Min()

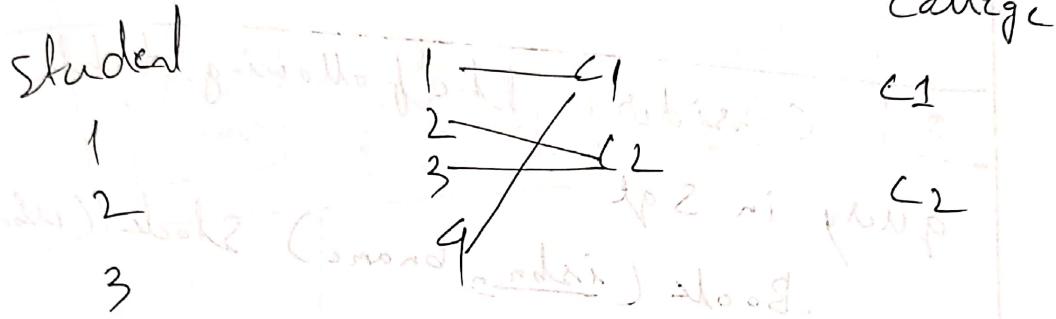
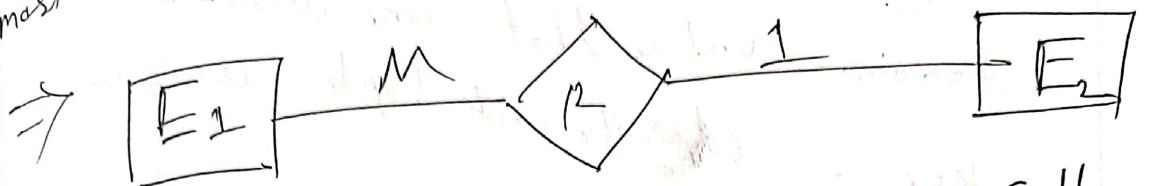
Q1) In an ER model, suppose R_y is many to one relationship from entity set E₁ to entity set E₂. Assume that E₁ & E₂ participate totally in R. The cardinality of E₁ is greater than the cardinality of E₂. Which one is true about R?

- (a) Every entity in E₁ is associated with exactly one entity in E₂.

(b) Some entity in E₁ is associated with more than one entity in E₂

① Every entity in E₂ is associated with exactly one entity in E₁.

② Every entity in E₂ is associated with at most one entity in E₁.



③ ✓ ④ X ⑤ X ⑥ X

(c) The view of total database content is

- ① conceptual view
- ② internal view
- ③ external view
- ④ physical view

→ ① ✓

(d) In relational model cardinality is

- ① no of tuples
- ② No of attribute
- ③ No of tables
- ④ No of constraints

→ ④ ✓
 ~~No of degree = no of constraints~~

- a) constraint is used to maintain consistency among tuples in two relations
- (a) key (b) domain (c) referential integrity
(d) entity integrity

→ c ✓

domain - value that one attribute take
key - classify each tuple uniquely

- a) consider the following table and query in SQL

Book (isbn, bname) Stock (isbn copy)

query 1 - inner join

query 2 - left outer

query 3 - right outer

Which query is certain to have an output that is a superset of the output of the other three queries?

- (a) Q1 (b) Q2 (c) Q3 (d) Q4

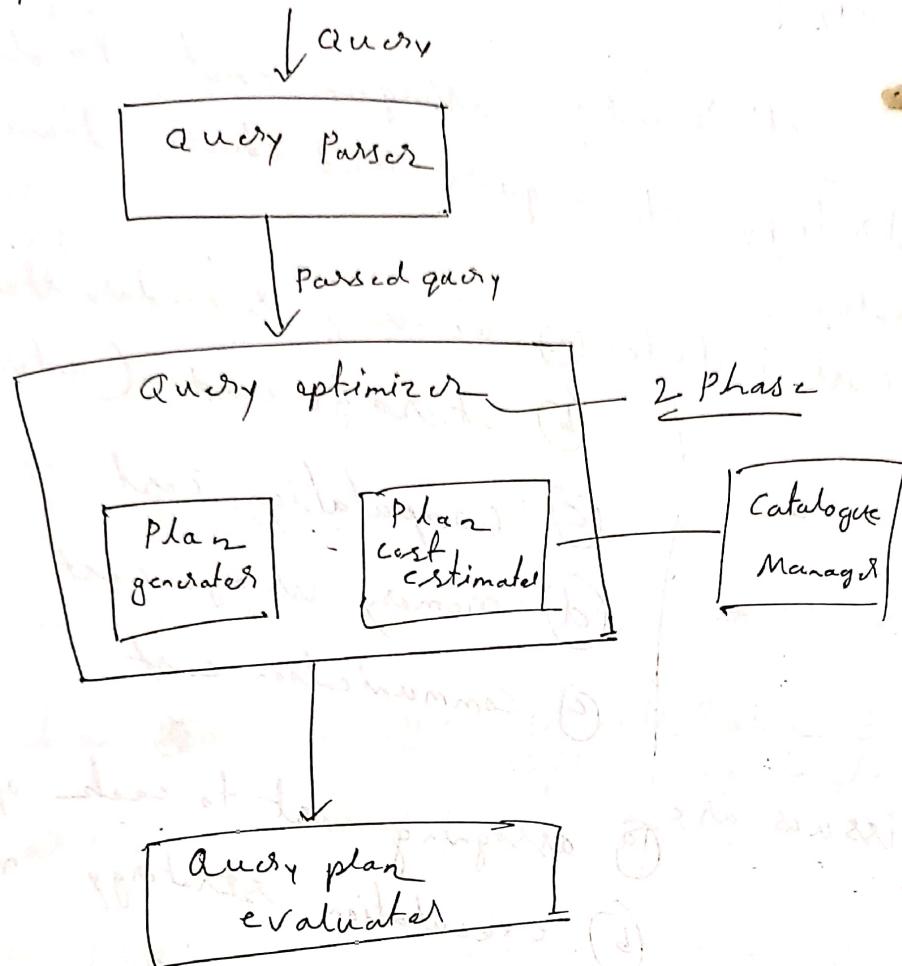
⇒ inner join = natural join

l_{oj} = inner join + left table left

r_{oj} = inner join + Right table left

full oj = inner join + left & right table

Query Optimization \Rightarrow It is the process of selecting an efficient execution plan for evaluating the query.



optimization is used for accessing the database in an efficient manner.

Ex → Select Sno from Student Where Sno = '101'

This is translated into $\sigma_{Sno='101'} (\pi_{Sno} (\text{Student}))$

$\pi_{Sno} (\sigma_{Sno='101'} (\text{Student}))$ efficient

Two techniques are

(i) Heuristic (logical) optimization.

(ii) Cost based optimization.

(ii) cost based \Rightarrow assign cost to different strategy of query and choose lowest cost strategy.

Cost include (a) accessing secondary storage (disk block)

(b) storage cost (intermediate result)

(c) computation cost

(d) memory usage cost

(e) communication cost

Issues are

(a) assigning cost to each operation

(b) execution strategy not fixed

(c) time consuming

(d) do not guarantee best optimized

(i) Logical \Rightarrow We use some predefined rules. It usually helps to optimize cost but not always. It is based on query tree.

(a) apply Select operation before project, join or other binary operation

(b) apply project operation before join or other binary operation

It reduces size of files.

- (*) Field means columns } Keys are constraints
 records mean rows }
- (*) A relational database consist of a collection of table.
- (*) column has only set of values.
 row has different set of values.
- (*) Column = attribute = Field
 Row = Record = tuple

(*) Relational Algebra = Procedural language

Domain relational + tuple relational = non procedural

(*) Varchar (20) ~~is mis~~ it will take 3 space as it changes length according to it.
 char (20) has fixed length

(*) Delete from R — Delete command remove the entries in the table.

(*) Trim is used to remove spaces at the end of the string

|| — Concatenation operator (append two string)

(*) Union eliminate duplicate
 Union all - retain duplicate

(*) Degree = number of attribute in a relation
 • inner join are SQL Server default join

- aggregate functions take a collection of values as input
- $\text{Avg}(\text{ }) = \text{mean value}$
- We use distinct to represent unique values
- $\text{count}(\text{x})$ only include null value, others don't include it (aggregate function)

(*) With clause: It is used for temporary relation.

(*) To include aggregate function having clause must be included after where we can use only in where only.

(*) Lateral \Rightarrow Used to access attribute of preceding table

(*) Subqueries can not gain table

(*) do not match some specified condition — double use of not exists

ER diagram: Set of entity = entity set
Relationships are — unary, binary, Ternary,
(relation between (two) (Three) same set)

Participation constraint \Rightarrow Specify the presence of an entity when it is related to another entity

(i) Total — all entity is in relationship — double line

(ii) Partial : — Some entity involve in relationships —