Jai Shree Ram.

Name : Anurag Singhal

INDEX

Standard : ..... Division : ..... Roll : .....

Subject : DBMS

S. No.	Date	Title	Page No.	Teacher's Sign/Remarks
1.		DBMS Intro	1-2	
2.		Database System & its types.		3-5.
3.		File System VS DBMS		5-7.
4.		2 tier & 3 tier architecture		8-10.
5.		Schema		10-11
6.		3 Level of Abstraction (or) 3 schema Arch.		11-13
7.		Data Independence		13-15.
8.		Candidate key & Primary key		15-16
9.		Primary key		16-18
10.		Foreign key		18-19
11.		Foreign key (Referential Integrity)		19-21
12.		Ques^n of Referential Integrity		22
13.		Super key in DBMS		22-25
14.		E-R Model		25-26
15.		Types of attributes in ER Model		27-28.
16.		Degree of Relo^ship (Cardinality)		28-30.
17.		One to one Relo^ship		31-32
18.		Many to many Relo^ship.		32-33.
19.		Ques^n practice.		33-34
20.		Normalization		34-39
21.		First normal form (1NF)		39-40
22.		Closure method		41-44
23.		Functional Dependency		44-47
24.		2nd Normal form (2NF)		47-50.
25.		3rd Normal form (3NF)		50-52.
26.		BCNF (Boyce Codd Normal form)		52-54
27.		Lossless & Lossy Join Decomposition		54-57.
28.		All normal forms with Real Life Examples		57-58.

(Live SQL).

(1) Database Management System (DBMS) :→

→ Basic Introduc° → 2 tier, 3 tier, (3 schema, 3 level of abstraction) Imp.
(comes in Data Independence)

→ Various Data models :→
Network, Hierarchical,
(Relational, ER, object oriented.

(DBMS) or (RDBMS) — same.
(Relational)

→ ER MODEL :→
Conceptual. (Entity - Relationship)

Attributes & its types,
Relationship — " — Imp.

Imp →
→ Basics of keys :→
- primary key & its characteristic.
Candidate key — " —
Super key — " —
&
Foreign key — " —

→ Normalization →
closure method → to find candidate key
functional dependencies
1st NF (normal form), 2NF,
3NF
BCNF

→ Transaction Control & Concurrency →
ACID properties.
R-W problem (Read-write).
W-R "
W-W "
Conflict Serializability.
Recoverability.

Concurrency → locks.
↑ 2-PL, timeStamp. (2 phase lock).

→ SQL & Relational algebra.

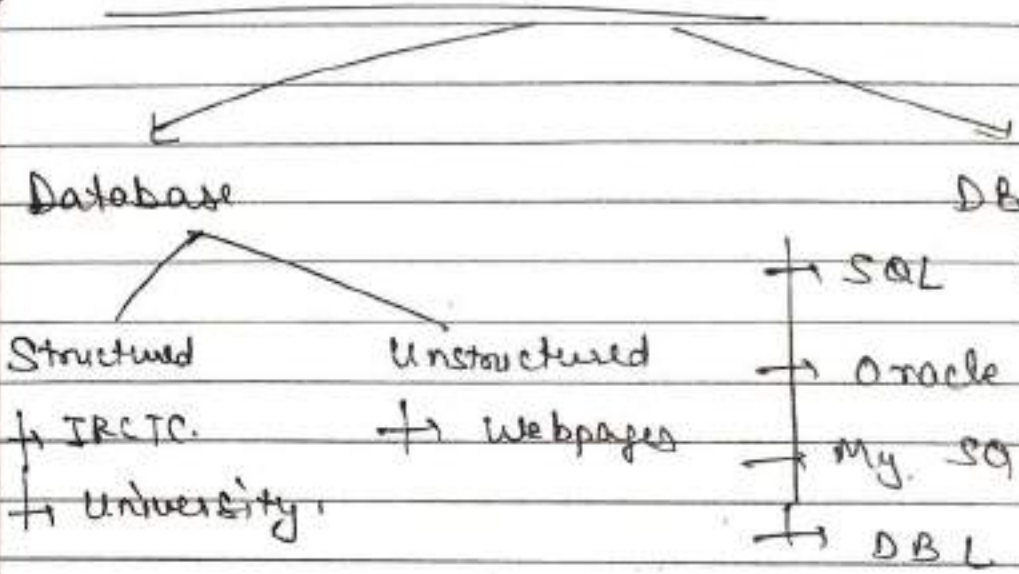
SQL → Structured Query language.
(It is a programming lang.).

DDL command. (Data-Definition)
DML (Data-Manipulation)
DCL
→ Constraint.
→ Aggregate func.
→ Joins
→ Nested Query.
→ In, Not in, any, all.

→ Indexing: → (Single level Indexing).
→ primary, cluster & Secondary Indexing.

→ B tree, B+ tree. (in Multilevel).

(2) Database System :→

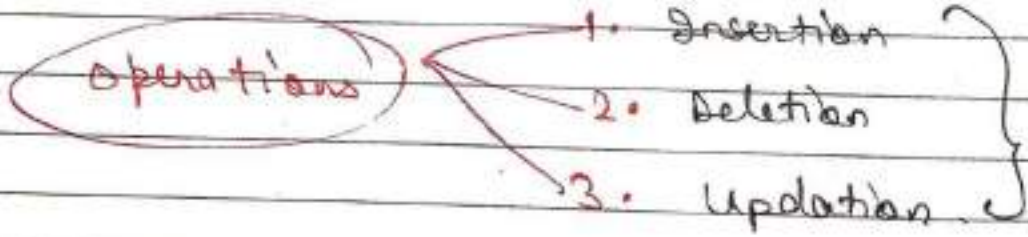


a) Database :→ Collec^n of Related Data.

Ex:- Indian Railways & Indian passport have their different data.

b) Structured :→
RDBMS → Rela^al Database Management Sys^m

c) DBMS :→ collection of operations



It provide easyness to perform opera's.

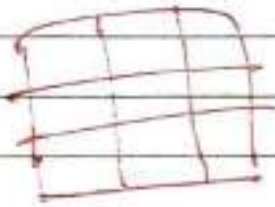
d) Diff. companies have made diff. DBMS.
Ex:-
Microsoft Co. ⇒ SQL Server
(Sequal Server).

⇒ Oracle ⇒. 9i, 11, 12 c, etc.
My SQL.

⇒ IBM ⇒. DB2.

(#) Structured Data! → RDBMS
↳ Relation

Mean,
We stored in the table form.



→ (Table).

Now, Table is technically called as Rela".

→ Rela" is most usuable form.

Now
→ Store Rela"s & access Rela"s is done by the Management System i.e. RDBMS.

⇒ When it is used for Relations, it is called as RDBMS.

Hence,
→ We perform Insert, Delete & updation opra"s on Rela"s.

(#) Unstructured Data! → There is not predefined structure.

Ex! →
A webpage is a collec" of photos, videos, chats, etc. i.e.
There is not a particular format in these.

⇒ Hence, RDBMS works only on the structured Data.

[Note:→] 90% of data on this Earth is unstructured.

⇒ i, There is more technologies on unstructured data.
Ex:→ Bigdata, Hadoop, etc.

---

3. File System Vs DBMS :→

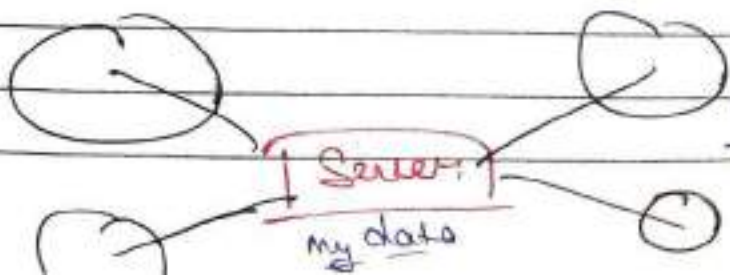
⇒ File System is used before DBMS.

⇒ User always manages its data in file form.
→ OS has inbuilt File System.

Ex:→ CIFS, NFS File systems.

⇒ File System:→ We stores our data in file form. & then into our drives.

→ Why we use DBMS?
for we are using the client-server architecture, means.
⇒ Our data is at a centralised loc^n & all over the world users can access that.



→ Hence, we can't use file system here.
Now,
DBMS comes into picture.

1.) If we have to search only for 1 KB, then in file system, complete file comes to us (approx of 25 GB).
↳,
More memory usage. Now, DBMS → gives us only of 1KB data from the server.

(Searching is fast & Memory utilization is Efficient).

2.) In file system, we require attributes to search data. ie,
Attributes (name, loca, permission, etc.)

Meta Data → Data about Data.
(Data of a particular file).

In DBMS, no loca, it is totally independent. We don't require any attributes here.
(Easeness provide).

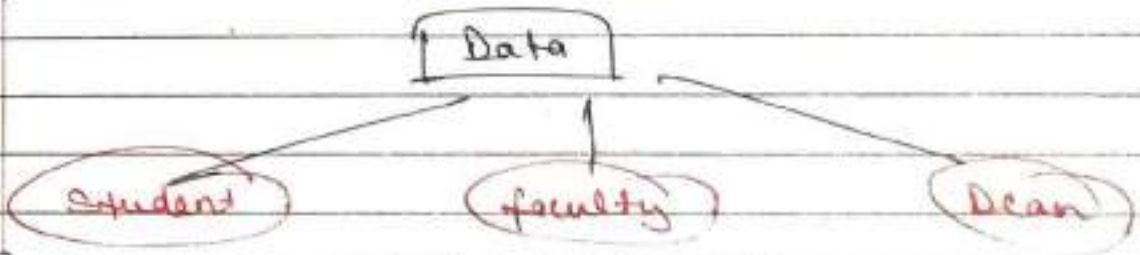
3.) Concurrency → means Concurrent Access
- Multiple persons can access the data at the same time.

Ex: IRCTC (Indian Railway) System

File system may show inconsistency when multiple users want access of a file at the same time.
DBMS, have proper protocol for concurrency.

RR
RW
WR
WW
(Read - write)

4.) Security! → Role based Security



(Role based Access Control.)

If we are user, we only get user data.
" - faculty, " - faculty

(O.S)
File system, has no security for this.
No level-by-level. No Hierarchical.
DBMS has this role-based security feature.

5.) Data Redundancy! → (Duplicacy).

In file system, we can save same content by diff file names.
In DBMS, has many constraints to ensure unique data. Stops redundancy of data.

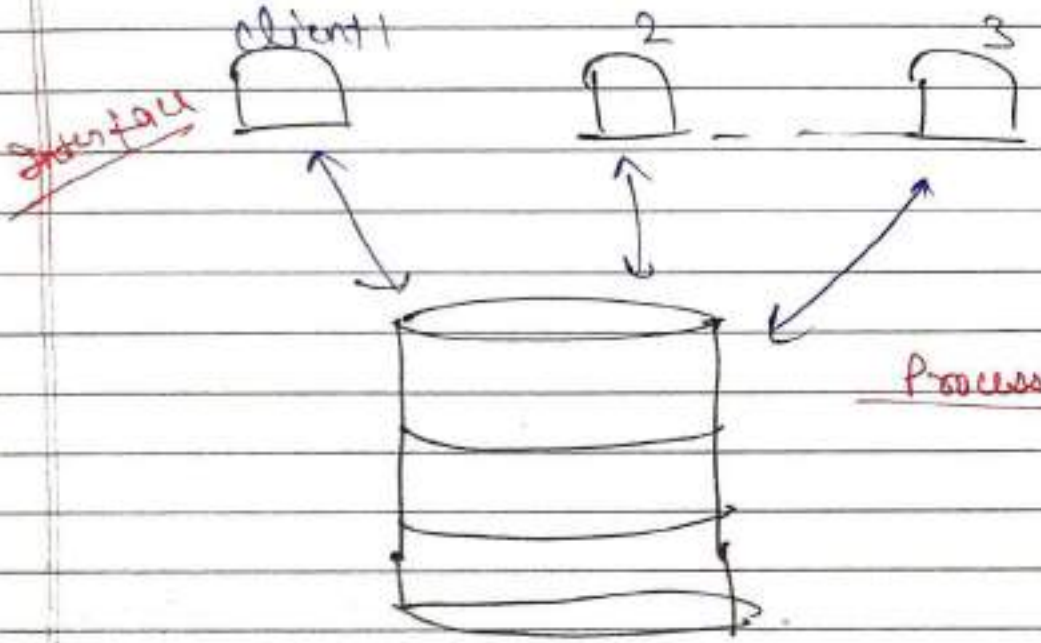
→ DBMS is at back End of every client Server & Web Applications.

Date: Page:

(4.) 2 tier & 3 tier architecture in DBMS :-

(H) 2 tier :- means 2 layers.

1. client (machine) layer.
2. Database Server. (Data layer).



Database Server.

→ 2 tier also called as client-Server Architecture.

Ex:- (i) Indian Railway :- If we reserve ticket by going to railway Stn at ticket window.

(ii) Bank :- When physically we draw or post some money.

(client → Request → Database Server → gives info)

→ Hence, limited clients & limited Database to which we access. ie, why maintenance is very Easy.
But, the users are not limited. They are in big nos. Then, this 2 tier system fails. We call it Scalability.

Security :- → not gud. bcz clients are directly interact with the database.

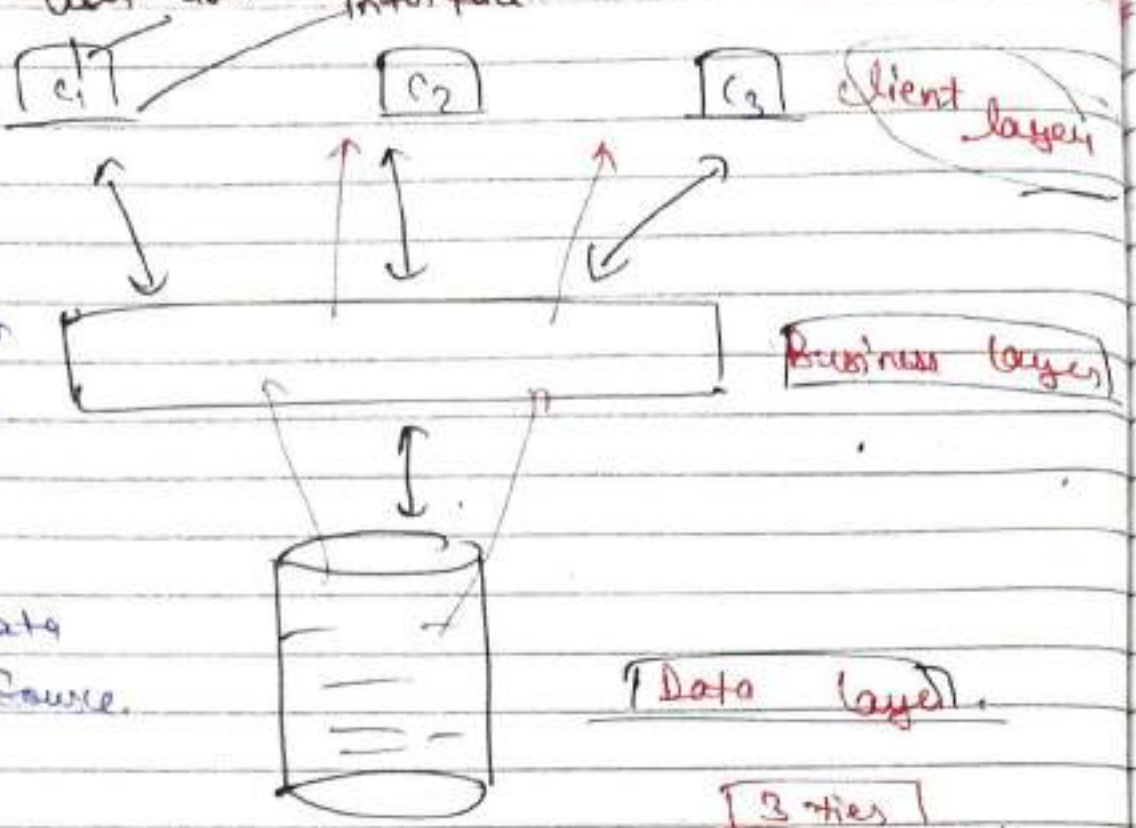
Advantage :- Maintenance is Easy.

(#) 3-tier :- → (Now, mostly uses).
1.) client layer
2.) Business layer
3.) Data layer.

→ In 2-tier, Query is process in Data-base Server.
But,
→ here Business layer supports interface. (Our Query is process in Business layer)
Hence,
We don't give load to Database Server here.
Hence, Business layer acts as Intermediate.

Ex:- IRCTC & banking app & Gmail app

Advantage :- 1) Scalability.
2) Security (no direct interact of data & user)



Here, Maintence is not Easy bcz. it is Complex.

Ex:- Web Applica's (APPS) are kind of 3-tier Architecture.

If we go physically to any bank or Railway Sta", then it is 2-tier Architecture.

(5) Schema → Logical Representa" of a database.

⇒ Ex:- In RDBMS, data is stored in the form of Tables. (Rela"s).

DBMS access & manage the data in this Schema (table) form. It is not actually stored in this table form in drives.

Ex: (i) Schema of a Student: → (Entity) E-R

Roll. No.	name	addresses

✓ Relationships.

(ii) Course: (Schema, Entity).

C. ID	name	Dura^n

✓ (Relationship).

↳ Logical Representa^n (structure).

→ But, we implement it by SQL. (Sequel).
(Integer, character - - -).
Data Defi^n language (DDL) → to implement Schema. (Or to design)

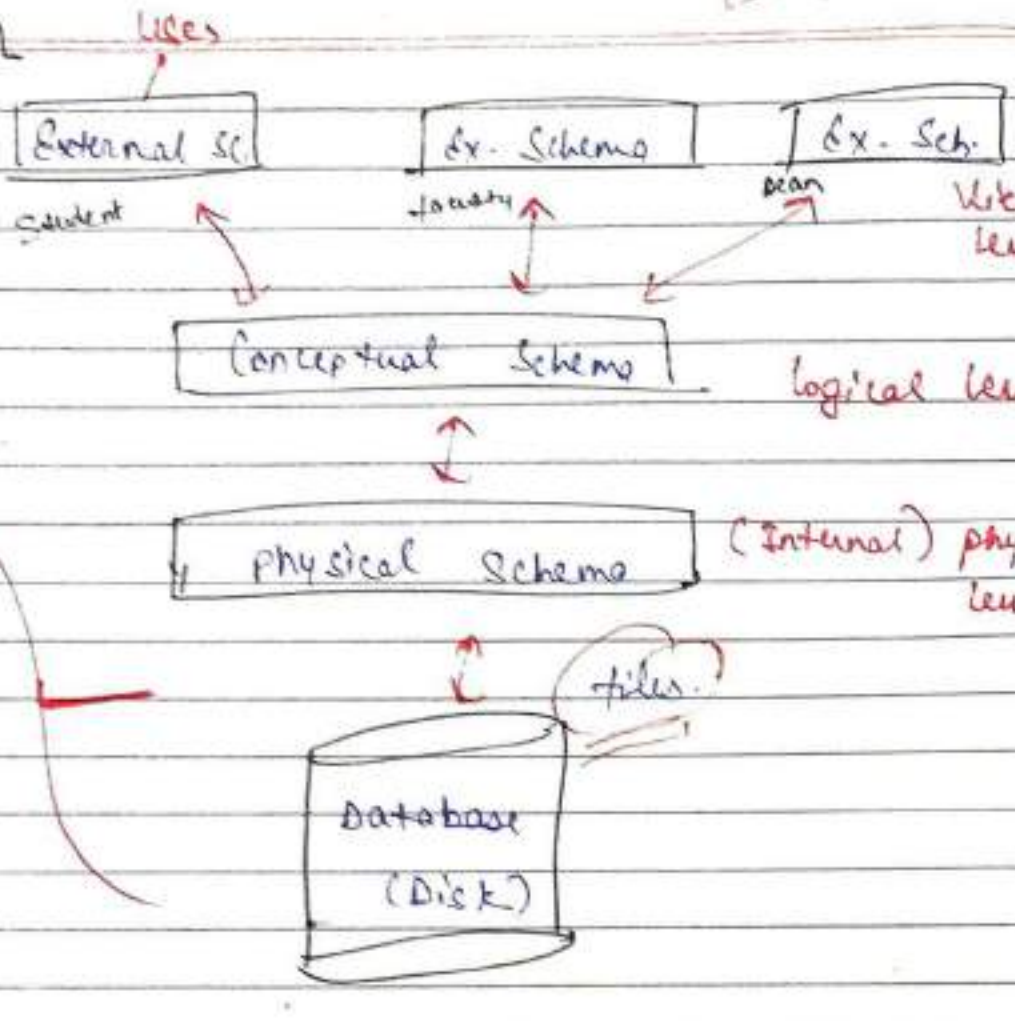
Schema is simply a structure (table form).

---

6. (Three Level of Abstraction) (or) (Three Schema Arch^n.)

→ (We hide the loca^n of where the data is stored, from the user)

Ex: Gmail, we don't know physically (exact loca^n) where our mails are stored. Ex- Delhi, UK, US, - - - etc.



External Sch: (View level) :→ View that will be given to the User.

{ Students have their View.
Faculty have —"— . }

Ex:→ After Login, which View comes to us is External Schema.

Conceptual Sch: E-R Model

Ex: Student (Roll No, age, add, ...)

→ In-form" of all tables that we use & their Relo^ship. A type of Blueprint is this.

physical Schema :→ Where the data is actually, physically present. The loca^ of data.

→ A Normal Data Base Designer is working at level of Conceptual Schema.

→ front End or Interface developers are at level of Ext. Schema.

→ Database Administrator (Who has all control of data) is at physical Schema.

→ Centralised — Data at one place.
Multiple — Data all over the World.

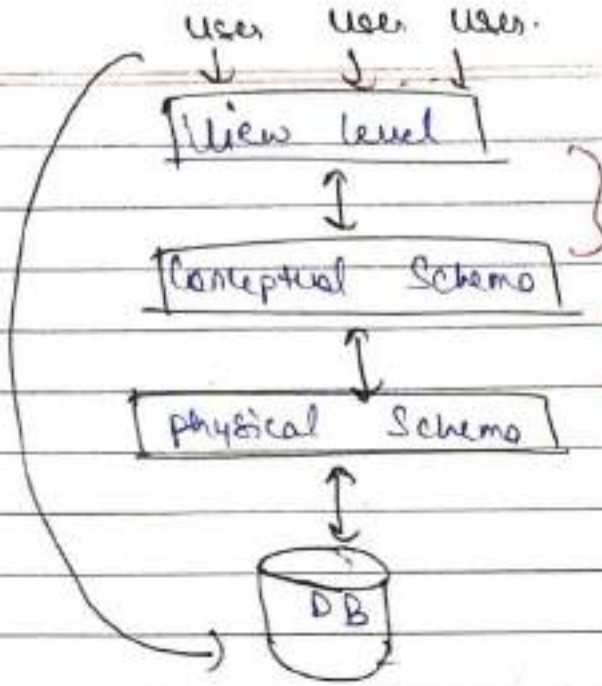
Note:→ When we see the data as a user, then we see it in Table form. But, Actually in hard disk, data is stored as files. and we apply the layer of DBMS on it.

There are 3 layers b/w the user & the data

7. Data INDEPENDENCE :→

→ Make user independence of data. Hide its loca^ & etc.

→ Conceptual Schema :→ which table we use, how many tables are there. how many attributes, relationship b/w them.



(#) Logical Data Independence! →

Let,

Student Table. → add by user 1.

Name	Age	Mob. No.

↓

It will not affect the Applica^n program.

→ We don't need to write the App^n pgm again. What user 1 changes want, we give him. But, It will not affect the actual logical structure.

mean, User 2 still can see only col 1 & col 2. Mob No. Colu^m is only ∀ user 1.

→ We use this concept, by Views. (Virtual Table).

→ In actual Table, may be we have 50 col's but user can only see 5-7 col's. So, user think there is only 5-7 col's. But, there are many.

→ Hence, View level don't change by your changes by users in Conceptual Schema.
Ex:- UMS (University Management System), Shopping website → They can add or delete any col's.

Hence,
It is logical Data Independence.

physical Data Independence :→ Schema
Any change in physical ~~Data Independence~~ won't affect our Conceptual Schema.

Ex:- If we take data from Hard Disk 1 to 2, then data not changes. Tables & Structures remains the same.

Any change in Back End, wont affect the user. It is Data Independence.

---

8. Candidate key & Primary key :→

→ key :→ It is one of the attribute in the table.
Use of key: To uniquely identify any 2 tuples in the table.

Roll. No.	S. name	City	Age
1	Reddy	Shamli	20
2	Anurag	Kanpur	21
3	Reddy	Shamli	20

1.) Same 1 student repeats or
2.) 2 diff. students with same attri.

4 To identify this, we must have a key.

Ex:) Student Table

1. Aadhar Card
2. Roll No.
3. Reg. No.
4. Licence No.
5. Voter Id
6. Phone No.
7. Email - Id.

These all attributes can uniquely identify any 2 rows.

The set of all 1-7 values. If we make a set of all of them. Then we call it Candidate key.

{Aadhar Card, Roll No, Reg No ——— Email-Id}

Now, from above Candidate key set, we choose the one most appropriate & call it Primary key. & rest all keys known as Alternative keys.

× ×

9. PRIMARY KEY :→

(Every lock has its unique key.)

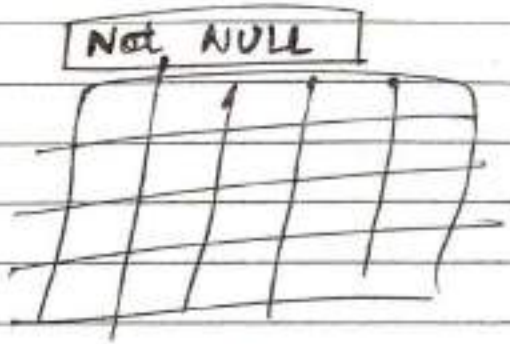
i.e.
ule uniquely identify 2 things.

Ex! Any 2 student can have same name, age, D.O.B. but, some attributes like phone No., Aadhar Card — are always unique

Candidate keys! → These are Unique.

→ phone no
→ aadhar card
→ Pan
→ Reg no.
→ Roll no.

} Candidate keys.



→ phone no, aadhar card Value को NULL नहीं रख सकते। भरना ही पड़ेगा।

Case 1: We fill wrong. aadhar card no.
Case 2: We don't take admn" without aadhar card no.

In student case, most appropriate key is Reg no.
Roll no.

→ PRimary key = { unique + NOT NULL }

→ We don't give primary key to them, they give to us.

Ex:- university give us Reg. No.
passport office give us passport No.
license office give us license No.

- We have only 1 primary key in a Database. Software don't allow us this.

(Why need 2 or more, when we only need 1)

10. Foreign key in DBMS :→

(F.k.)

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

→ It Maintains referential Integrity.

Ex:- Student →

P.K (primary key)	Roll no	name	Add.
	1	A	Delhi
	2	B	Chandi
	3	A	Mumbai

Course →

(F.k.)

C. ID	C. name	Roll. No.
C_1	DBMS	1
C_2	networks	2

→ Roll. No is same b/w the student & course. It shows Relation'ship b/w them.

→ In Table 2, Roll No. colm value 'references from Roll no. attribute (primary key) in Table 1. take

Q:- Can I write Roll. No. '10' in Foreign key (F.k.)?

→ No, bcz It is not in Table 1 (Roll. No) Until now.

(with f.k.)

⇒ Table 2 is called Referencing Table.

⇒ Table 1 is called Referenced Table. (with p.k.). Or Base Table.

```
* Create table Course
  (
      Course_id varchar (10),
      Course_name varchar (20),

      Rollno int references
                  Student (Roll no.).
  );
```

⇒ Code.

* Now, how to write a query after the table is created.

```
Alter table Course
ADD constraint f.k.
foreign key (Roll no.)
references Student (Roll no);
```

Note: (1.) We don't have to keep the name 'Roll. No.' same of both the attributes necessarily. We can also take diff. names.

(2.) In a table, there can be more than 1 foreign key.

(11.) Foreign key :- (Referential Integrity) :-

→ Integrity means Same value. for the database.

Ex:1 (1)

→ In mobile phone, diff. price at Flipkart, Amazon & store. So, not Integrity.

(2.) In university, a Student Reg no. is same at every office in the university, in library, in dean office. So, Integrity present here.

Ex:1 -

Student (Base Table or Referenced Table)	Roll No. (P.K.)	name	add.
	1	A	Delhi
	2	B	Mumbai
	3	A	Ghd.
	4	C	Gzd.

Deletion

C. Id	C. name	Roll. No (F.K.)
C_1	DBMS	1
C_2	networks	2
C_3	C++	7

Course (Referencing table).

(#) Referenced Table →

1.) Insert :→ No violation (We can easily add).

2.) Delete :→ May cause violation (bcz, if we delete a row & with same roll. no. a row is in referencing table. Then, it's not possible).

Sol^n :→ 1.) On delete Cascade
(Also delete from every table)

2.) On delete set Null
(Insert NULL on other tables).

~	~	Roll. No NULL

f.k. can. take reference from P.K. of same table. also.

But, if same col" is also a primary key in that other table. Then, we can't use this col". bcz.

primary key cannot be NULL.
<unique, not NULL>

3.) on delete No Action.
(in this col", our row dont get deleted, first we have to delete from other tables, than we can delete in our table).

3.) updation: → may cause viola" (bcz, if we make '20' of '2', then now how we can get Roll No '2' in referencing table).
Sol"!: →
1.) On update Cascade
2.) On update Set NULL.
3) On update No Action.

(#). Referencing table: →

1.) Insert: → May cause viola" (bcz, if '1' is not base table, then how it in Referencing table)

2.) Delete: → No violation

3.) updation: → May cause violation. (bcz we cant make '20' of 2 if '20' is not in base table).

Note: 1.) Same key can be foreign & primary key in a table.
2.) Many table can have a foreign key from a single table by take reference of its P.K.

(12) Q:-7)

Let $R_1(a, b, c)$ and $R_2(x, y, z)$ be 2 relations in which 'a' is foreign key in R_1 that refers to primary key of R_2. Consider, 4 options →

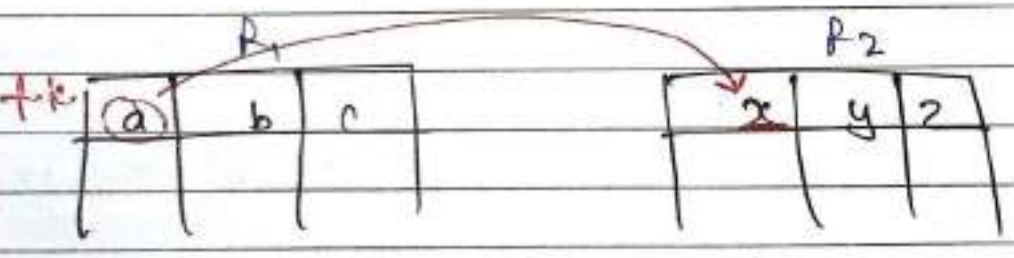
a) Insert into R_1 — ✗
b) Insert into R_2 — ✓
c) Delete from R_1 — ✓
d) Delete from R_2 — ✗

Which is correct regarding referential Integrity ?

1.) option a & b cause violation
2.) option b & c will cause violation
3.) option c & d — " —
4.) option d & a — " — → (Ans).

Let, x be p.k. in R_2.

Sol'n :-



Referencing Table. Base Table (or) Referenced Table.

Note :- If f.k. is not there, then we can do anything (insert, delete, update) without any violation.

(13) SUPER KEY in DBMS :-

1. Super key is a combination of all possible attributes which can uniquely identify 2 tuples (row) in a table.

→ Candidate key is minimal.

Ex:- Candidate key (C.K.) = Roll.No. (also Super key)

Roll no.	name	age

Roll no, name
Roll no, age
Roll no, name, age
} all are Super key.

Candidate key (C.K.) के साथ कुछ भी add कर दो, तो super key हो जाएगी।

⇒ name, age × (not Super key).

⇒ Super set of any Candidate key is Super key.

Q:- R(A1, A2, ... An) then how many Super keys are possible.
If → A1 is Candidate key.
→ A1, A2 are candidate keys.

Ans:- power set :→ how many subsets can be possible of given set.

Ex:- → A1 A2 A3 (Either take or not take A1, 2 possibility)
2 × 2 × 2 = 8
⇒ 2^3.

→ Now, Sol^n → $R(A_1, A_2, \ldots A_n)$.

i) ⇒ (इसमें A_1 तो आएगा ही).

A_1
A_1, A_2
A_1, A_3
A_1, A_2, A_3, A_4.

$R(A_1, \underbrace{A_2, A_3, \ldots A_n}_{n-1})$.

Compulsory.

$1 \times \underbrace{2 \times 2 \times \ldots 2}_{n-1}$.

So, $\boxed{2^{n-1}}$ Ans.

ii) $R(A_1, A_2, \underbrace{A_3, \ldots A_n}_{n-2})$.

[redacted] Ans.

→ A_1, A_2 both C.K.

When A_1 →	When A_2 →
A_1	A_2
A_1, A_2	A_2, A_1
A_1, A_2, A_3	A_2, A_1, A_3
—	—
2^{n-1}	2^{n-1}

But, some sets are common (bcz $A_1A_2 = A_2A_1$).

So, Common are those who has both A_1, A_2.

So, those are 2^{n-2}.

Now,

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

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

iii) $A_1 \cdot A_2$ combined is C.K.

2^{n-2} ans.

iv) $A_1 A_2$, $A_3 A_4$ are C.K.

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

Now,

$A_1 A_2 A_3 A_4$	$A_1 A_2 A_3 A_4$
$A_1 A_2 A_3 A_4 - -$	$A_3 A_4 A_1 A_2 - -$

To remove these common Elements.

$A_1 A_2 A_3 A_4 - - - A_n$

2^{n-4}.

$2^{n-2} + 2^{n-2} - 2^{n-4}$ ans.

$2^{n-1} - 2^{n-4}$

Q6.

14. E-R Model: → (Entity Relationship Model)

→ used for logical representation.

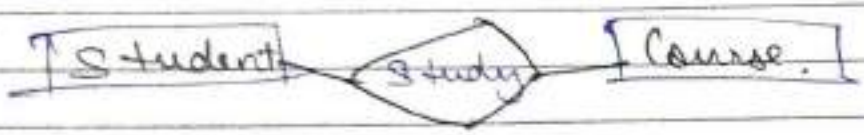
→ To see logical structure before implementation (Design).

→ It does the job of database design.

Entity :- Any object which has physical existence is Entity.

Ex:- Student (roll no, age, address)
(Entity) attributes.

→ Relationship → Relationship b/w 2 or more Entities



Relationship b/w student & course is of Study.

⇒ Student (Roll no, age, address)
Entity type (Schema).

⇒ We implement these by using SQL.
(Structured Query lang).

→ Entity.
→ Attributes - Characteristics of Entity.
↳ types.
→ Relationship.
↳ types → 1 to 1
1 to many
many to 1
many to many } 4 types

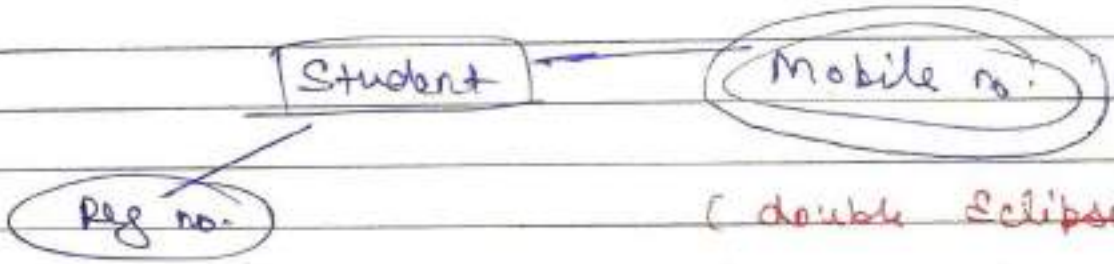
Entity → Student → represented by rectangle
Attribute →
Relationship →

(15.) Types of Attributes in ER Model :-

Student

1.) Single vs Multivalued attributes :- →

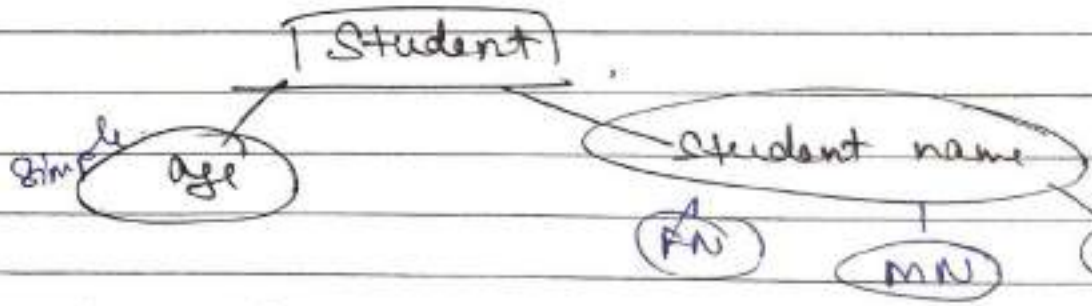
Reg no. — Mobile no. or address. (May be more than 1 C.No. of a Student).



(double Eclipse).

2.) Simple vs Composite Attributes.

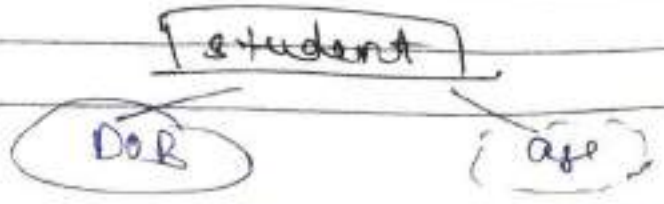
Simple - Can't be further divided.
Composed - Composed of more than 1 value.
Ex:- name (first name, middle name, last name)



3.) Stored & Derived Attributes :- →

Stored → These are stored & can't be derived.
Ex - DOB.
Derived → Ex - Age. (derived from D.O.B.)

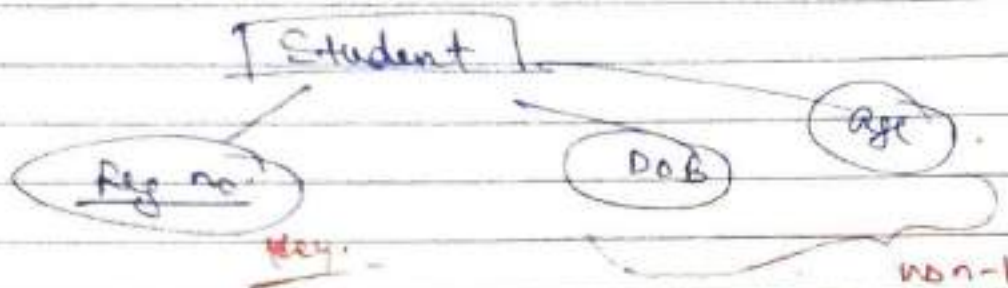
dotted Eclipse.



4) Key vs Non-key Attributes :→

Key - used to uniquely identified.
Unique (No Repitition)

Ex - Reg No is always unique for a student.
Represent with underline (___).



5) Required vs optional Attributes :→

Required → These are Mandatory (*) Name
optional → can also be leave. DoB, Add., ~~ph~~

6) Complex Attribute :→
(Composite + Multivalued)

Eg :→ If a student have 2 Residential Add, & in Each Add., he have 2 phone no:

Add. is Composite (Street, State, City) Multivalued.

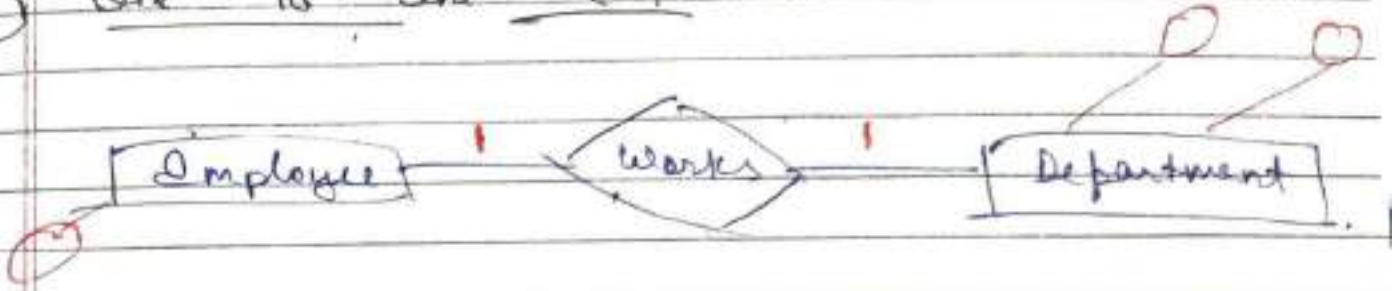
16. Degree of Relationship :→ (Cardinality).

→ how the Entities are connected with Each other.

4 types! →

1) 1-1 — One to one
2) 1-M
3) M:1
4) M-M (M-N) — Many to many

(1) One-to-One (1-1) :→



Convert Entity into Table :→

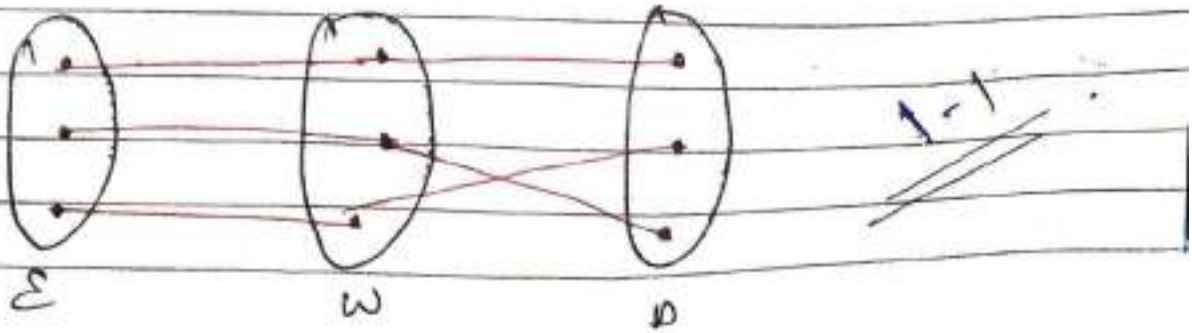
(Relo'ship को भी table बनेगा, so हमें Employee & Department का Rel' वाला)

Rel'ship Table! → Attributes → 2 always (primary keys of both the table)

(E.ID) & (D.ID)

These, E.ID & D.Id works as a foreign key (F.K).

→ When we have to enter data in this Relo" ship table, then we have to see Rel'ship (1-1, 1-M, ---).



→ P.K. = Either E_Id or D_Id.
(primary key)

E_Id	E name	age
E_1	A	20
E_2	B	25
E_3	C	28
E_4	A	24
E_5	B	25

f.k.

EID	D_ID
E_1	D_1
E_3	D_2
E_2	D_3

f.k.

D_ID	Dname	loc
D_1	I.T.	Banglore
D_2	Prod.	Delhi
D_3	HR	Delhi

both

P.K = E_ID

* Can we Merge?
Now, In Table 1 & Table 2, E_ID is the primary key.
Hence, we can merge Table 1 & 2.

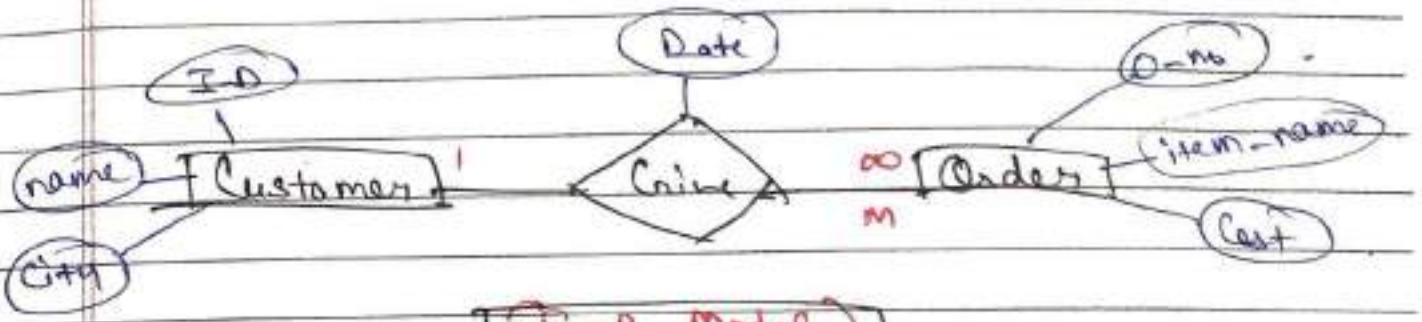
E_ID	E name	age	D_ID
E_1	A	20	D_1
E_2	B	25	D_3
E_3	C	28	D_2
E_4	A	24	—
E_5	B	25	—

* Now, we have 2 Table at final.
(Merge Table & Department Table)

× ×

Every Table must have its primary key.

(1-P) One to Many Rela^n ship : →
(1-M).

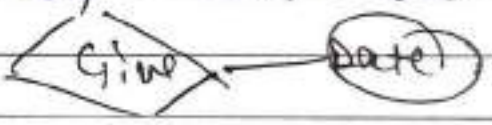


(E-R Model)

→ When we physically implement the E-R Model, then we need Relational Model. & we use Tables in Rela'al Model.

a) Rela'ship may have its attribute.

Ex:-



& we call it Descriptive Attribute.

f.k. f.k.

Id	Name	City
C_1	A	Tal.
C_2	B	Delhi
C_3	C	Mumbai
C_4	A	Mumbai

I-D	O-no	Date
C_1	O_1	~
C_1	O_2	~
C_2	O_3	~
C_2	O_4	~

O-no	Item name	Cost
O_1	Pizza	100
O_2	Burger	200
O_3	pasta	300
O_4	Cold-Drink	400

Here, O-no is always diff. & so unique too.

P.K. = (O_no).

Note:- Always P.K. of the many side in (1-M) is also the P.K. of the Relationship Table.

Can we Merge Tables? (many right side merge)

Yes,

By Relationship & Order Table.
(bcz both have same P.K.).

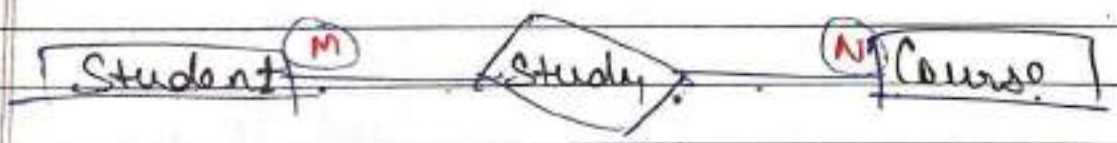
ID	O_no	item name	Cost	Date
~	~	~	~	~

Now,

2 Tables.

(M-1) is also same like this.

18. Many-to-Many Relationship :→
(M-N)



Roll no	name	age
1	A	16
2	B	17
3	A	16
4	C	17
5	D	15

Base Table

f.p. f.k.

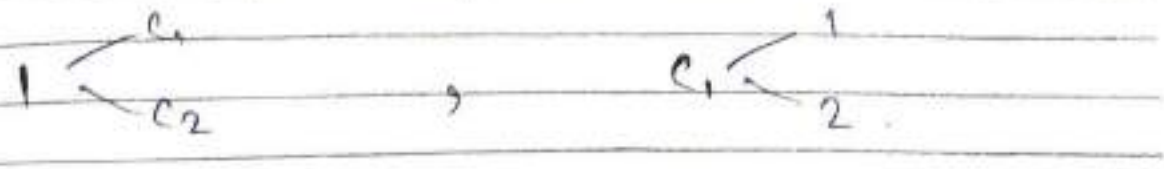
Rollno	C-id
1	C_4
2	C_2
1	C_2
2	C_4
3	C_3

Referencing Table

C-id	name	Credit
C_1	Maths	4
C_2	phy.	4
C_3	Chem.	4
C_4	Hindi	4

Base Table

Many - Many



P.K. in Referencing Table (Relationship Table) :→

Roll. No repeats &
C-id also repeats

So, Roll no. & C-id both make P.K. Combinely.

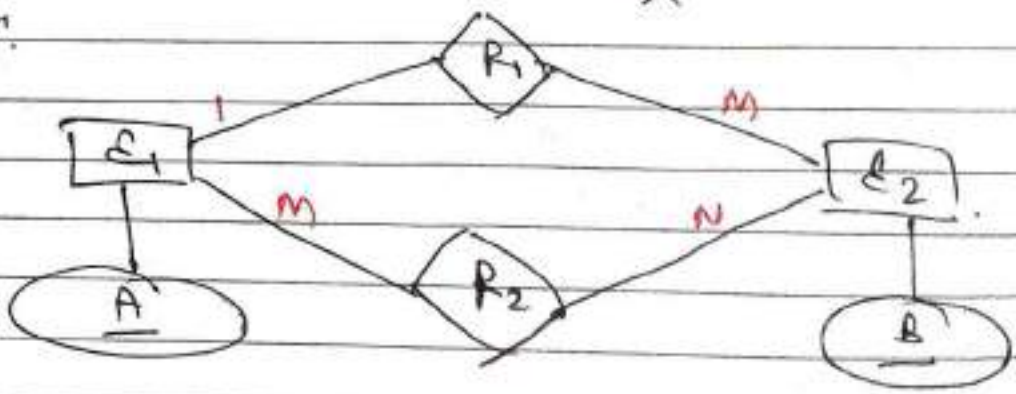
i.e, **Composite key = Roll no. C-id**

Can we Reduce Tables ?

→ No. bcz, p.k. is combined.

Note:→ p.k. in Relationship Table depends on Relationship (1-1, 1-M, ... M-M).

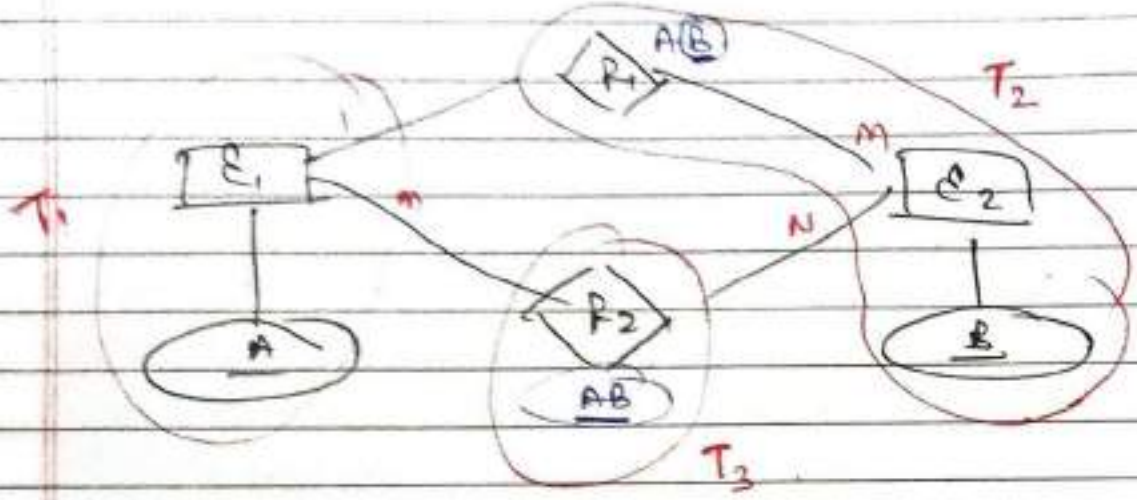
Q:→



* What is the min^m no. of tables required to represent this E-R model into Relational Model ?

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

E_1
E_2
R_1
R_2
} 4 Tables . But minimum ?



Hence,

$T_1 = E_1$
$T_2 = R_1 E_2$
$T_3 = R_2$
} 3 Tables ✓

E_2 का सभी attributes $R_1 E_2$ के combined table में आ जाएगा। So Now, we also don't need separate E_2 Table. [Min^M = 3] ✓

---

20. **Normalization** :→

It is a technique to remove or reduce Redundancy (Duplicacy) from a Table.

There are 2 types of duplicacy in Database!→
1.) Row level
2.) Column "

(#) Row level!→

S-Id	S-name	Age
1	Ram	20
2	Varun	25
1	Ram	20

Same. (Duplicate) =

Row level.

→ We use the concept of primary key (P.K.)
We set a P.K. to any appropri. attribute.

primary key (Unique + Not Null).

P.K. will take care of this duplicacy.

(#) Column-level!→

student course faculty

P.K.	S-Id	S-name	C-Id	C-name	F-Id	F-name	Salary
	1	Ram	C_1	DBMS	F_1	John	30,000
	2	Ravi	C_2	Java	F_2	Bob	40,000
	3	Nitin	C_1	DBMS	F_1	John	30,000
	4	Anurag	C_1	DBMS	F_1	John	30,000
	5	Varun	C_{10}	MBBS			

⇒ 4 columns are same in many rows.

Anomaly

→ Insertion Anomaly
→ Deletion "
→ updation "

(→ Anomaly means problems, occurs on special occasion.)

Now,

S.Id	S.name	C.id	C.name	F.id	F.name	Salary
~~1~~	~~Ram~~	~~C_1~~	~~DBMS~~	~~f_1~~	~~John~~	~~30k~~ 40k
2	Ravi	C_2	Java	f_2	Bob	40k
3	Nitin	C_1	DBMS	f_1	John	~~30k~~ 40k
4	Amritpal	C_1	DBMS	f_1	John	~~30k~~ 40k.
5	Varun	—	—	—	—	—
		C_{10}	MBBS			

1) Insertion Anomaly :→
→ We want to add data of a new St.
Let, Varun
Let,
University starts a new Course,
C_{10} – MBBS
→ We can't insert this info. in table.
Even, we — " — the new faculty data be,
→ We only introduce the new Course C_{10}.
We don't talk about the student and we don't have S. Id.
→ We also remain it (S.id) NULL - bcz it is a P.K.
→ So, we can't insert directly.

It is the our Insertion Anomaly.

2.) Deletion Anomaly :->

We have simple query = Remove the database of Roll. No - 1.

```
Delete from student
where S-id = 1.
```

-> It will delete the whole row.

We don't face any prblm here.

Now,

We have to delete the data of Roll. No 2.

```
Delete from student
where S-id = 2
```

-> Row 2 is deleted fully from database.

Now,

Row 2 is blank there.

Now,

tell us who is teaching to Roll No. 2 & what was the Course name of Roll. No 2.

Likely be, It was only one Student who was studying that particular Course. & that particular faculty is teaching that Course.

-> We here, only delete the detail of Student but, bcz of him, all the info got deleted.

Course Info - lost
faculty Info - lost. } &

i.e, extra Info is removed here & we can't recover it later.

3.) updation Anomaly :→

Simple query → [S-Id-4] change name from Amrit to Amritpal.

```
Update student
Set Sname = 'Amritpal'
Where S-Id = 4
```
→ Code.

No problem here.

Now,
If we want to change the salary of faculty F1 from 30K to 40K.

Change Salary of F1 from 30K to 40K.

Now, how many times the F1 repeats in table, the same no. of times updation query runs to changes them all from 30K to 40K.

Note:) There is only 1 faculty F1, then his salary must also changes 1 times. But, due to the column level duplicacy, it runs no. of times. Hence, It takes more time.
It is updation anomaly.

* Now, Normalizn removes Redundancy.

How ?

→ A simple Rel^n may be, it use divide that table into multiple tables. Like,

P.K.	S-id	S-name

P.K.	C-id	C-name

P.K.	F-id	F-name	Salary.

This can be 1 of the Rel^n.

→ Now, we don't get any anomaly in insertion, deletion & updaⁿ. There is no effect on others. Easy.

---

21. # First Normal Form :→ (1 NF)

Ef Codd → father of D.B.M.S

→ Table should not contain any multivalued attribute.

Student

Roll No.	Name	Course
1	Sai	C/C++
2	Anuraj	Java
3	Onkar	C/DBMS

→ Not in 1st NF

Null mean not available

Now, how to convert in 1st NF?

1st way

Roll no	Name	Course
1	Sai	C
1	Sai	C++
2	Anurag	Java
3	Onkar	C
3	Onkar	DBMS

primary key (P.K.) = Roll no. Course

(Combined, it is composite P.K.)

2nd Way

Roll no.	Name	Course 1	Course 2
1	Sai	C	C++
2	Anurag	Java	Null
3	Onkar	C	DBMS

P.K. = Roll no.

3rd way

Roll no	Name
1	Sai
2	Anurag
3	Onkar

Base Table

P.K. = Roll no

F.K.

Roll no.	Course
1	C
1	C++
2	Java
3	C
3	DBMS

Referencing Table

P.K. = Roll no Course

F.K. = Roll no.

22. Closure Method :→

helps
↳ To find all the Candidate keys in the Table.

Ex: Candidate key (C.K.)

R(ABCD).

FD of { A→B, B→C, C→D }.

(Functional Dependency).

Meaning of closure is that What 'A' can determine

Here, A is determing B (from FD ①)

closure Sign. ← $A^+ = B$

$A^+ = BCDA$.

(A can determine itself also)

Ex: Roll no. can determine itself.

A → B
B → C
↳ Transitive.

Now, R(ABCD) has all 4 Attributes that are in $A^+ = BCDA$.

∴ A can determine all the Attributes of Table. This is the prop. of the Candidate key.

Now, $B^+ = BCD$

Here, B not determine A.

B cannot be a C.K.

$C^+ = CD$

$D^+ = D$.

Prime att. $= \{A\}$

Non prime att $= \{B, C, D\}$.

Only A is C.K.

$CK = \{A\}$

Note:

$(AB)^+ = ABCD$

(AB → itself, B → C, C → D.)

Here,

AB can be a C.K.

But

It be not a C.K.

bcz

C.K. be always minimal.

In AB only A be C.K.

∴ XtoA is Super key (S.K.)

AB

Super key

(A se saath add any thing, becomes S.K.)

AB is a Super key.

Ex:

R(ABCD).

$FD = \{A \to B, B \to C, C \to D, D \to A\}$.

$A^+ = ABCD$

$B^+ = BCDA$

$C^+ = CDAB$

$D^+ = DABC$

∴ C.K. $= \{A, B, C, D\}$ all ✓.

prime Attribute! → are Attribute which is used in making of the C.K.

Ex, prime att. = {A, B, C, D} . { bcz, all ↑ are C.K. }

Non-prime att. = {ϕ} . → NULL .

Q! R(ABCDE).

FD = {A → B, BC → D, E → C, D → A} .

⇒ Now, we have to check that which attribute are coming on the Right side. bcz, Attributes on the Right side, वो ↑ को determine तो कर रहे हैं।

= BDCA . (E नहीं आया)

E = BDCAE

Note! → Each & Every candidate key must contain. i.e. bcz, E if present on left side, then only it is written in Right side also.

जो Right side में नहीं आ रहा। मानि तो Left side में होना ही चाहिए। तो Candidate key बनाने में use होगा ही होगा।

Now,

E+ = EC → (E alone is not candidate key but, it is used in making C.K.)

Now, Start! → with A!

AE → ABECD ∴ A is C.K.

$BE^+ = BECDA$

$CE^+ = CE$ ×

$DE^+ = DEABC$

C.K. = {AE, BE, DE}

Trick!- First, we get AE as C.K. So, Check (A or E) either on the Right side of F.D.

F.D = {A → B, BC → D, E → C, D → A}.

So, directly DE is also becomes the C.K., now check if D, it is depend on 2. So, check with (used in making C.K.).

prime Att. = {A, B, D, E}

non-prime att. = {C}. Ans.

× ×

(23) Functional Dependency :→ (F.D.)

It is the method which describes the rela'ship b/w the attributes.

X → Y → Dependent Att.

Determinant → X determines Y (or) Y is determined by X.

Ex!-

S-id	→	Sname	→ valid.
1	→	Ranjit	} ∵ These 2 are diff.
2	→	Ranjit	

Ex!-

1	→	Ranjit	} Same Student
1	→	Ranjit	

Valid case.

Ex:- 1 → Ranjit
2 → Raman
} Valid.

Ex:- 1 → Ranjit
1 → Raman
} Not Valid.

(A) F.D. are of 2 types:→

1.) Trivial F.D.
2.) Non-Trivial F.D.

1.) Trivial F.D.:→

If $x \rightarrow y$

then,

y is subset of x.

These Trivial F.D. are Valid. (Always True).

Ex:- $\underset{x}{Sid} \rightarrow \underset{y}{Sid}$ ✓

NOTE:→ $x \rightarrow y$

$LHS \cap RHS \neq \phi$ (Never NULL)

Ex:- Sid Sname → Sid.

$\cap$ Sid. ✓ Valid.

2.) Non-Trivial F.D.:→

If $x \rightarrow y$

then,

y is not a subset of x.

i.e. $X \cap Y = \phi$ (NULL)

Ex:- Sid → Sname
Cid → phone no
Eid → Loca"

Now, for this we have to check cases & find which is valid or not.

(*) properties of F.D :→

1.) Reflexivity: If Y is subset of X, then $X \to Y$. (Trivial)
(Sid → Sid).

2.) Augmentation:
If $X \to Y$, then
$XZ \to YZ$

Sid → Sname
Sid phone → Sname phone

3.) Transitive: If
$X \to Y$ & $Y \to Z$
then,
$X \to Z$.

Sid → Sname & Sname → City
Sid → City.

4.) Union:- If $X \to Y$ & $X \to Z$
then $X \to YZ$

5.) Decomposition: If,

$X \to YZ$ then

$X \to Y$ and $X \to Z$

But:

$XY \to Z$ $\quad X \to Z$ & $Y \to Z$ ✗

6.) Pseudo Transitive:

If

$X \to Y$ & $WY \to Z$ then,

$WX \to Z$

7.) Composition:-

If $X \to Y$ & $Z \to W$

then

$XZ \to YW$

2nd Normal Form :→ (2nd NF).

2 Rules

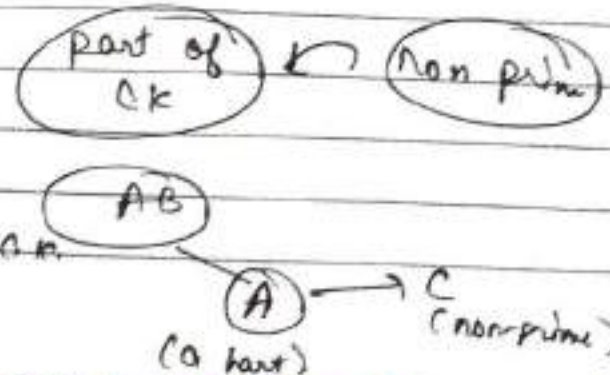
→ Table or Rel^n must be in 1st NF.

→ All the non-prime attributes should be fully functional dependent on Candidate key (C.K. or P.K.)

(or)

[There should be no partial dependency in the Rel^n].

Eg:



Ex! →

Customer

Custome -Id	Store - Id	Loca^n
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

C.K. : Customer -Id Store Id

Prime attributes: C - Id
Store - Id.

Non prime : Loca^n

Here, Loca^n is only depend on Store - Id.
i.e. partial dependency.
bcz,
(to be in 2nd NF it should depend on the both C-id & S-id (bcz both are C.K.).

Now →

Convert it into 2nd NF :→

Here, we make 2 Table.

C-Id	Store-Id
1	1
1	3
2	1
3	2
4	3

Store-Id	Loca^n
1	Delhi
2	Bangalore
3	Mumbai

(Here, it is fully dependent, bcz only 1 C.K. is here)

2nd N.F.

Q:- R (A B C D E F)

FD: { C → F , E → A , EC → D , A → B }

Sol^n:- [C.K.]

First, check Right hand side.

Step 1):- F A D B (means Now, these att. are determined by some of the values.)

So, on LHS, there must be CE.

(C.K. जो भी बनेगी, उसमें CE तो होना ही)

CE : FADB

Now,

EC^+ = ECFADB (all 6 are present)

∴, EC is C.K.

Now, Use Trick

Either E or C must be present at the RHS of any F.D.

but

not, neither E nor C, noone is present.

So,

there is only 1 C.K. in this table.

i.e. [C.K. = {EC}] ans.

Proof:- check

Step 1 is comp. here.

After finding C.K. = {EC}

$A^+ = AB$
$B^+ = B$
$C^+ = CF$

∴, proved.

proper subset is always less than a set.

$X \subset XY$ → proper subset
$X \subseteq XY$ → subset

Step 2 :→ prime Attributes : $\{E, C\}$

non-prime Attributes : $\{A, B, D, F\}$.

Step 3 :→ C.L. = $\{EC\}$

What is proper subset of EC
↳ either 'E' or 'C'.

Now check :-

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

Partial
(Either proper subset of C.K.)
Check on LHS → either 'E' or 'C' AND
Check on RHS → whether it is non-prime attr.

for partial dependency i.e, not in 2nd NF rule gets

$C \to F$

↳ partial dependency,

So,

Table is not in 2nd NF.

$C \to F$	α PD	partial
$E \to A$	α PD	fully.
$EC \to D$	α FD	
$A \to B$	α FD	

→ 1 भी P.D. मिलती है, Table is not in 2nd NF.

Q5. 3rd NF :→

→ Table or Rel' must be in 2nd NF.

&

→ There should be no transitive dependency in table.

Non prime ⇒ Non-unique
prime ⇒ unique.

Date:
Page:

not sufficient cond^n.

(NPA) →

* Mean, Non-prime att. को कोई Non-prime att. determine नहीं कर सकता।

(C.K. & Prime att. (P.A.) तो कर सकते हैं determine NPA को।)

Ex:-

Rollno.	State	City
1	punjab	Mohali
2	Haryana	Ambala
3	punjab	Mohali
4	haryana	Ambala
5	Bihar	patna

C.K. = {Roll no.}
F.D → {Roll no → state, state → city}

⇒ PA = {Roll no.}

⇒ NPA = {state, city}.

Also, (NPA → NPA).

Here, Roll → State and State → City.

It is transitive dependency & we don't want that.

So,

It is not in 3rd NF.

Ex:- R (ABCD)

FD: {AB → C, C → D}

⇒ C.K. = {AB}

PA = {A, B}

NPA = {C, D}

AB → ABCD. (Transitive)

C → D
NPA NPA

It is not in 3rd NF.

C.K. + anything = S.K.
Super Key.

Ex:- R (ABCD)
FD: (AB → CD, D → A).

(B not on RHS)

Soln: C.K.: {AB, DB}.
PA: {A, B, D}.
NPA: {C}.

$B^+ = B$

$AB^+ = ABCD$

Now, A on RHS.

so, $DB^+ = DBAC$

is also C.K.

3rd: Now, for each F.D.

LHS → C.K. or S.K
[OR]
RHS → ∈ o P.A.

→ check only 1 bcz [OR].

Then, Table is in 3rd NF.

FD: (AB → CD, D → A)

∴ Table is in 3rd NF.

bcz, (NPA → NPA) is not present here...

---

(26) BCNF. (Boyce Codd Normal form):-

⇒ also called as special case of 3rd NF.

[illegible]

Student

Roll-no	Name	Voter-id	Age
1	Rani	K0123	20
2	Hanna	M034	21
3	Rani	K786	23
4	Rahul	D286	21

Table is in 3rd NF already.

C.K. = { Roll no, voter-Id }.

F.D. :- { Roll no → name, Roll no → voterid, voterid → age, voterid → Roll no. }

Note:- LHS of each FD should be C.K. or S.K.

Here, 3NF 'a' की (OR) वाली condi" ही हटा दी, जिसमें RHS के P.A. होने से भी काम चल जाता था).

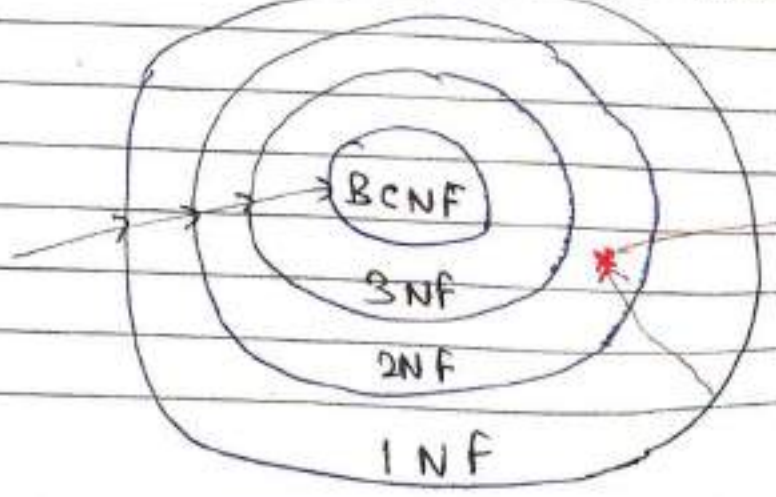
Here,
we only want C.K. or S.K. in L.H.S. & RHS में कोई लेना देना नहीं।

Sol":- So, check all the F.D. one-by-one.

In all the 4 F.D, the LHS is C.K.

So, It is in BCNF form. Ans.

(#) Compare:- 1NF, 2NF, 3NF, BCNF :-



1NF.

2NF = 1NF + condi^n

3NF = 2NF + condi^n

BCNF = 3NF + condi^n

Join

27 Lossless & Lossy - Decomposi^n :→

We normalise table → or we decompose table into 1NF, ... forms.

R

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

→ R_1 (AB)

→ R_2 (BC)

We divide this table into R_1 & R_2

Q. B is common in both the Table.

R_1

A	B
1	2
2	2
3	3

R_2

B	C
2	1
2	2
3	2

⇒ find the value of C if the value of A = '1'.

Now, for this we have to join R_1 & R_2 tables.

So,

Select $R_2.C$ from R_2 Natural Join R_1 where $R_1.A = '1'$.

(1st row of R_1 multiply all rows of Table 2).

Cross product:- If R_1 has x rows & R_2 has y rows then their join has x.y rows.

Condi'n:- Common col^m of both Tables (R_1 & R_2) - here (B) has the same value in join Table.

Natural Join = Cross product + Condi'n.

Now,

R_1 R_2

A	B	B	C	
1	2	2	1	✓
1	2	2	2	✓
~~1~~	~~2~~	~~3~~	~~2~~	
2	2	2	1	✓
2	2	2	2	✓
~~2~~	~~2~~	~~3~~	~~2~~	
~~3~~	~~3~~	~~2~~	~~1~~	
~~3~~	~~3~~	~~2~~	~~2~~	
3	3	3	2	✓

Now.

R'

A	B	C	
1	2	1	→ table after joining.
1	2	2	
2	2	1	
2	2	2	
3	3	2	

(take). Spurious of tuples.

- In original Table (R), we have only 3 tuples (rows).
but,
After Joining, in R', we have 5 tuples.

It is a flaw. It is called the Lossy Decomposition.

- Why lossy? →
Here, we get 2 extra rows. then, why lossy.
Here,
We don't talk about rows. We call it lossy because of inconsistency. There is a problem in Database.

⇒ In original, for A=1, C = 1.
but
In join table, for A=1 { C = 1, C = 2 }.

(#) Why we get Extra? → (2 tuples more)

⇒ bcz, we take B as common in both Table.
but

Criteria for Common. → Common Attribute should be C.K. or S.K. of either R_1 or R_2 or both.
So, we have to C.K. or S.K. of original Table.

∴ B has duplicacy in Table, We have to choose attri. 'A' for Right Rel.

bcz, A is unique. <1, 2, 3>.

i.e.

R_1 (AB)
R_2 (AC)

→ We get 3 tuples also in joining table.

Condi" for lossless Joining Decompos"! →

1.) $R_1 \cup R_2 \equiv R$

$AB \cup AC \equiv ABC$.

2.) $R_1 \cap R_2 \neq \phi$

$AB \cap AC$

$A \neq \phi$

3.) R_1 C.K. (or) R_2 C.K. (or) Both (C.K. of original Table)

To take Common attribute ↗

---

28. All normal forms with Real life Examples →

6 forms here

1st NF	2nd NF	3rd Nf.
→ No multivalued Attribute.	→ In 1st NF +	→ In 2nd NF +
→ only single valued.	→ No partial Dependency.	→ No Transitive dependency
	* only full dependency.	→ No, NPA. should determine N.S.A.
	AB → C	
	If A & B are 2 key of a Co., then both will use the Emplo. 'C'.	

BCNF	4th N.F.	5th N.F.
→ In 3rd NF + → LHS must be C.K. or S.K. $X \to Y$	⇒ In BCNF + ⇒ No multivalued Dependency. $X \to\to Y$	→ In 4th NF + → lossless Decompos'n.

Varun → 3 Phone no.
→ 3 Mail Id

(Varun depends on multiple att. i.e. phone & mail.)

table

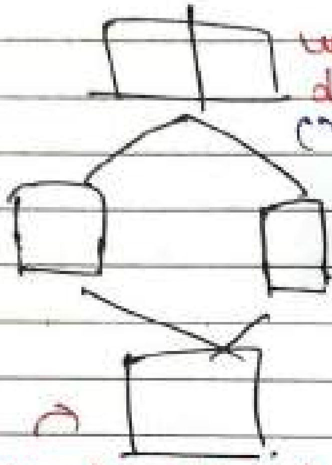
Varun	M_1	E_1
Varun	M_1	E_2
|	M_1	E_3
|	M_2	E_1
|	M_2	E_2
|	;	

→ Very long.
→ We also don't able to form a key.

So,
Make 2 table.

[Varun — phone] [Varun — Mail]

(Now, no multivalued dependency).


vertically divide (rows) can

May be extra tuples Come. So, make C.K. as common attribute in both tables.

(29) Minimal Cover :→ (Irreducable):

Q:- For the following functional dependencies, find the correct Minimal Cover →

$\{A \to B,\ C \to B,\ D \to ABC,\ AC \to D\}$.

a) $A \to B$, $C \to B$, $D \to A$, $AC \to D$.
b) $A \to B$, $C \to B$, $D \to C$, $AC \to D$.
c) $A \to Bc$, $D \to CA$, $AC \to D$.
d) $A \to B$, $C \to B$, $D \to AC$, $AC \to D$ Ans

Ques:- Our RHS in F.D. must be single.

Step 1: $\{A \to B, C \to B, D \to ABC, AC \to D\}$.

By decomposition prop, separate them.

$A \to B$, $C \to B$, $D \to A$, $D \to B$, $D \to C$, $AC \to D$.

Step 2: Remove the Redundant F.D. :-

$A \to B$, $C \to B$, $D \to A$, $D \to B$, $D \to C$, $AC \to D$.
Left: ✗ ✗ ✓ ✓ ✓ ✓

→ Let, $A \to B$ को हटा दिया, Now check the closure of A,

$A^+ = A$ (not all ✓).

i.e., $A \to B$ is not Redundant. We can't remove it.

→ Same check this for every F.D. :-

$D \to B$ is redundant.

Let, $D \to B$ ✗ then, $D^+ = DABC$ (all ✓).

i.e., ↓

Remove, $D \to B$.

अब जब आप check करेंगे, तो इसको नहीं लेंगे, क्योंकि तो पहले ही remove कर दिया।

Now, for $AC \to D$ ✗

$AC^+ = ACB$

∴ Also include $AC \to D$ ✓

Now, we get

$\{A \to B, C \to B, D \to A, D \to C, AC \to D\}$.

Step 3. Now, use only want 1 Att. in LHS.

Here, $AC \to D$

Now, Check By remove A. & then check, closure of C.

$C^+ = CB$

• अगर C^+ में हमें 'A' भी मिल जाता, तो हम A को हटा सकते थे। लेकिन नहीं मिला तो,

$AC \to D$ ही रहेगा।

Now check ✗ A, by removing C. ✓

$A^+ = AB$.

∴ Can't remove C.

∴ $AC \to D$ can't be reduced.

∴ $\{A \to B, C \to B, \underbrace{D \to A, D \to C}_{\text{complex}}, AC \to D\}$.

∴ $\{A \to B, C \to B, D \to AC, AC \to D\}$. Ans.

30) Question on Normalization :→

Q:- R (ABCDEF), check the highest normal form ?

F.D. : { AB→C, C→DE, E→F, F→A }.

Soln:- Step 1: Find all CKs in Relⁿ :→

By closure Method :→

(B is not on RHS.) So, B compulsory.

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

- $AB^+ = ABCDEF$ (all ✓)

↳ [AB is C.K.]

Now, Check for A on RHS.

we get, F → (A)

↳ [FB is also CK.]

Now, check A on RHS :→

E → (F)

So, [EB is also C.K.]

Now, Check E on RHS.

C → D(E)

So, [CB is also C.K.]

Now check for C on RHS.

AB → C.

AB is already our C.K.

So, we get all C.K.

C.K. = { AB, FB, EB, CB } ∴ 4 CK.

Step 2 :- Write all prime att. & NPA :-

P.A. = { A, B, C, E, F }.

NPA = { D }.

Step 3 :- Now, check FD :-

{ AB → C, C → DC, E → F, F → A }.

Now,

Check 1-by-1.

Highest NF :- 1NF, 2NF, 3NF, BCNF.

Redundancy decrease.

* Mean, when table is in BCNF, then redundancy is lowest. & in 1NF redundancy is highest.

* So, to check highest NF, we start from BCNF :-

→ In BCNF, we know, all LHS of all FD's should be CK. or SK.

{ AB → C , C → DE , E → F , F → A }

(x) (x) (x)

↓

It is not in BCNF form.

→ Now, 3NF :→

Check :→ transitive dependency (NPA → NPA) ✗

LHS → C.K. or S.K.
OR
RHS → is a P.A.
→ then → It is 3NF.

Now, 1st Condⁿ is already checked in BCNF. So, here check only 2nd ✗ condⁿ that whether RHS is P.A. or not.

{ AB → C | C → DE | E → F | F → A }

	AB → C	C → DE	E → F	F → A
BCNF	✓	✗	✗	✗
3NF	✓	✗	✓	✓

∴ not in 3NF :→

→ Now, 2NF :→

Same thing → if there is already a tick in 3NF, then we don't have to check that for 2NF. जो already 2NF में होगा तो। Check only for 'x' tick.

Check! →

LHS is proper subset of C.K. →
RHS is non-prime Att.

↓

for partial dependency i.e,
if true then not in 2nd NF.

	AB → C	C → DE	E → F	F → A.
BCNF	✓	✗	✗	✗
3NF	✓	✗	✓	✓
2NF	✓	✗	✓	✓
		(bcz both condⁿ true). ∴ not in 2NF.		
1st NF	✓	✓	✓	✓

↳ Ans is 1st NF. Ans

bcz

→ for 1st NF → that we don't want any multivalued attribute in the table. All att. must be single (atomic).

& in

Table → R (ABCDEF)

all are general att,

we can't tell them as atomic or multivalued by seeing them.

↳

Table already 1st NF में ही दिखती है.
(↳ Assume already).

∴

1st NF. Ans

proper subset is always less than a set.

31. Find out Normal Form of a Rel^n :→ (from 1NF - BCNF).

Q:- R(ABCDEF),

FD's : { AB → C, C → D, C → E, E → F, F → A }

C.K. = { AB, FB, EB, CB }.

P.A. = { A, B, C, E, F }

NPA = { D }

Sol^n:- Now, let we assume that R(ABCDEF) is already in 1st NF.

→ Check ⊕ 2nd NF :→ (partial dependency) P.D.

Condi^n ⇒ { LHS must be proper subset of any C.K. [and] RHS must be a Non-P.A }

F.D's { AB → C, C → D, C → E, E → F, F → A }

F & F (FD) | T & T (PD) | T & F (FD) | T & F (FD) | T & F (FD)

(full dependency).

∴ not in 2nd NF.

⇒ make it in 2nd NF :→

We do it by decompose the table. (Divide into 2 parts).

Cond^n! →

Common attribute must be C.K.

1.) Lossless Decomposi^n
2.) Dependency should be preserved.

(Now, यहाँ problem है यहाँ. C→D उठाकर Alag करते हैं, Alag Table बनाते हैं।)

Now.

R (AB CD EF)

R_1 (AB EF) ↙ ↘ R_2 (CD)

Now, we have to make a common att. b/w these 2 table! —
Criteria → common! - can be C.K. of any R_1 & R_2.

So, In R_2 (CD)
C → D $C^+ = CD$ (all 2).
i.e. [C is C.K.].

So,
make 'C' common in both R_1 & R_2.

Now, again

R (ABCD EF).

R_1 (ABCEF) ↙ ↘ R_2 (CD)

R_1
{AB → C, C → E, E → F, F → A}
↓
2nd NF.

R_2
{C → D}
& C is C.K. [C.K. is C itself is not its proper subset]
F.D.
↓
2nd NF.

Now:-
→ Check ∀ 3NF:→

{ LHS must be C.K.
(or).
RHS [redacted] be P.A. } → 3NF.

so,

R_1
(ABCEF)
{AB→C, C→E, E→F, F→A}
✓ ✓ ✓ ✓

C.K. = {AB, FB, EB, CB}
PA = {A, B, C, E, F}.

∴ → It is 3rd NF.

R_2
(CD)
{C→D}
✓

C.K = {C}.
PA = {C}.

It is also 3rd NF.

Now:-
→ Check ∀ BCNF:→

Condi^n → L.H.S. must be a C.K.

R_1 {AB→C, C→E, E→F, F→A}
✓ × × ×

not in BCNF form.

→ There is Redundancy.

R_2 {C→D}

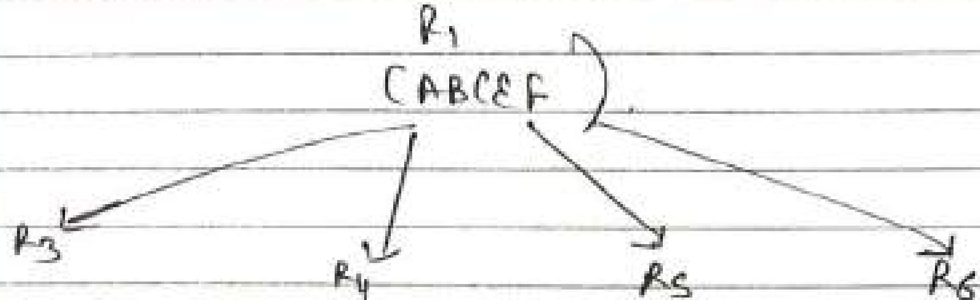
It is BCNF form.
↓
Redundancy — 0%.

Now,
Again Decompose R_1
(ABCEF).

(यही problem चल रहे, उसको Alag करेंगे।)

SK
self
its
subset

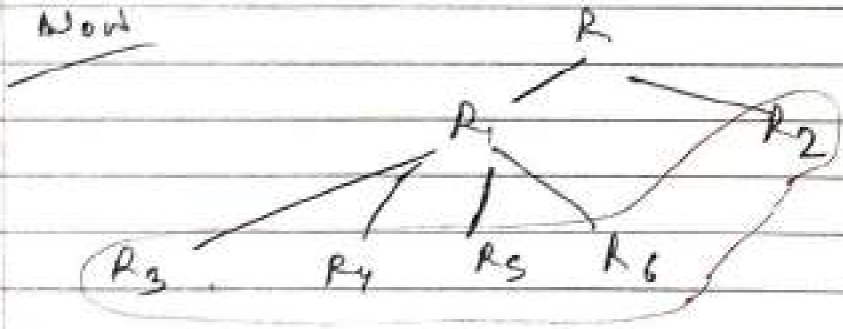
Normalisation aim → To make redundancy 0%. | SM (lossless decomposition) to normalise



R3	R4	R5	R6
(AB→C)	(C→E)	(E→F)	(F→A)
(ABC)	(CE)	(EF)	(FA)
C.K. = {AB}	C.K. = {C}	C.K. = {E}	C.K. = {F}

Now, these all 4 tables have C.K.
So,
these all 4 are in BCNF now
& (कुछ-न-कुछ common तो है ही सब में)

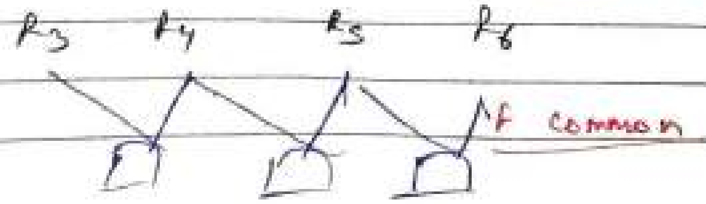
& Now



Now, In all these 5 tables,
Redundancy is 0%.

- But, there are now multiple tables & if we join them again, it is Complex.

(as) have to Join R3 & R4 (both have common C)



⇒ But, Join R_3 & R_5.
(ABC) (EF). (there nothing is common)

We can't join them directly. So, take help from R_4.

⇒ So, Combine R_3 & R_4, let, R_3R_4 (ABCE)

Now,
We can combine it with R_5 bcz both have now 'E' as common att.

$R_3R_4R_5$ (ABCEF)
↳ E is C.K. now.

(32) Normalisation Questions :→

→ A rel^n is in BCNF, then it is in 2NF also

Q: A Rel^n R has 8 att. (ABCDEFGH)

F = { CH → G, A → BC, B → CFH, E → A, F → EG }

how many candidate keys in R.

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

Sol^n: First, check on RHS, which att is absent.

D ⇒ So, now it comes with every.

$D^+ = D$

$AD^+ = ADBCFHG$ (all 8).

Now, check A on RHS,
So,
ED^+ also a C.K.
Now, check E on RHS.
So,
FD^+ also a C.K.
Now, check F on RHS.
So,
BD^+ also a C.K.
Now, check B on RHS.
Now, completed.
So,

AD, ED, FD, BD ✓ C.K.

33. Ques" Explained on Normalisation! →

Schema is a structure of a Table.

→ So, Schema (or) Table → same.

→ Non trivial F.D. means in which.

LHS ∩ RHS = ϕ → we have to check that it is valid or not.

→ Trivial F.D. are always valid.

Schema 1 : Registration (Roll. No, Courses)

Non trivial F.D of roll no → Courses }

Ans → We have to check that this table is in which form.

→ So, as always, start from higher form.

→ Check for BCNF :-

Condi" : LHS of every FD must be C.K. or S.K. or P.K.

and,

Roll. no is already given as a C.K.

So,

LHS is a C.K. of F.D.

So,

→ Table is in BCNF form.

i, also in 1st NF, 2nd NF & 3NF.

Schema 2 : Registra" (roll no, Course id, email)

Non trivial FD of roll no, Course id → email
email → roll no.

Ans → C.K. = { roll no Course id }.

PA = { roll no, Co. Id }.

NPA = { email }.

→ Check BCNF → 1st F.D. LHS is C.K. but not in 2nd F.D.

So, not in BCNF form.

→ Check for 3NF:-

Condⁿ:- LHS must be a CK or SK or PK
OR
RHS must be a P.A.

Now, 1st FD is already valid,

2nd FD: email → roll no.
P.A.
F T.

(T)

It is in 3NF.

Scheme 3:- Reg'n (roll no, Co-Id, marks, grade).

Non trivial FD: { roll no, Co-id → marks, grade
marks → grade. }

Ans:- C.K. = { roll no Co-id }.
PA = { roll no, Co-id }
NPA = { marks, Grade }.

→ check for BCNF:-

1st FD → valid.
2nd FD → not valid.

Not in BCNF.

→ check for 3rd NF:-
1st FD → already valid
2nd FD → marks → grade.
F F ∴ not in 3rd NF.

→ check ∀ 2nd NF ! →

Condⁿ ! - [LHS must be a proper subset of CK. AND RHS must NPA.]

↳ for P.D. → i.e, not in 2nd NF. if both Condⁿ are True.

→ 1st FD → already valid.
2nd FD → marks → grade
f T
(→ f)

↳ Not in partial Dependency
∴ It is in 2nd NF. Ans :-

Schema 4 ! → Register (roll no, C-Id, Credit).

Non trivial FD { roll no, C-Id → Credit
C-Id → Credit }.

Ans → C.K. : { roll no C-Id }
PA : { roll no, C-Id }.
NPA : { Credit }.

→ check ∀ BCNF ! →
1st FD → ✓
2nd FD → ✗
} ∴ not in BCNF.

→ Check ∀ 3NF ! -
1st FD → ✓
2nd FD → C-Id → Credit
f f → (F.)

So, not in 3rd NF.

Check ∀ 2NF! →

1st FD → ✓

2nd FD → C-Id → Credit
T T
(∀)

∴ It is Partial Dependency (P.D.)

So,

It is not in 2nd NF. ∴

∴

Table is in 1st NF. ∴

(We already assume it).

(34.) Cover & Equivalence of F.D. : →

If X covers Y $Y \subseteq X$

& If Y covers X $X \subseteq Y$

∴ if both are True.

then, $[X \equiv Y]$ → Equivalent.

Q:- $X = \{A \to B, B \to C\}$ | $Y = \{A \to B, B \to C, A \to C\}$.

i) X covers Y: → $(Y \subseteq X)$

(इसमें Y की FD. में LHS का closure लेंगे लेकिन X वाले में से।).

So,

$A^+ = ABC$ $\quad B^+ = BC$.

$Y = \{A \to AB,\ B \to C,\ A \to C\}$.

∴ all 3 covers in these closure forms.

∴ X covers Y. ✓

(i) Y covers X :→

$X = \{A \to B,\ B \to C\}$.

(X का closure Y से लेंगे।)

$A^+ = ABC$

$B^+ = BC$

∴ Y also covers X. ✓

∴ both Symbols cancels each other & become Equivalent.

$X \not\subset Y$ $\quad Y \not\subset X$

$\boxed{X \equiv Y}$ Ans. ✓

Q:- $X = \{AB \to CD,\ B \to C,\ C \to D\}$.

$Y = \{AB \to C,\ AB \to D,\ C \to D\}$.

(i) X Covers Y :→

$Y = \{AB \to C,\ AB \to D,\ C \to D\}$.

$AB^+ = ABCD$ $\quad C^+ = CD$

∴ True.

ii) Y covers X :-

$X = \{AB \to CD,\ B \to C,\ C \to D\}$.

$AB^+ = ABCD$, $B^+ = B$, $C^+ = CD$ (from Y).

∴ Y not covers X.
It becomes false.

∴

[$X \neq Y$] Ans.

&

[$X \supseteq Y$] ✓, true & [$Y \supseteq X$] ✗, false ✓.

(35.) Dependency Preserving Decomposition :-

Defin :- Let, any table [R(ABCD)]

& also some F.D., [FD {A → B, B → C}]

Now,
we find closure & then find C.K., & then we also find hidden dependencies :-
like.

$FD^+ \{A \to C\}$ by Transitive prop.

and,

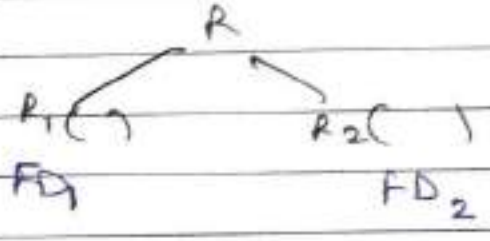
→ In normalisation, we do decomposition, then, we divide

R → $R_1(\ )$, $R_2(\ )$

Use distribute attri. of, i in R_1 & R_2

union of attri of R_1 & R_2 must be equal to the attri of R.

Now, F.D. also divides, like



NOW,

Check! - Dependency should be preserved, (कहीं कोई F.D. lost ना हो जाए हो, सबको preserve करके रखना है।)

i.e. $FD_1 \cup FD_2 = FD'$ (original F.D. के Equal.)

Q: Let R(ABCD) with F.D.

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

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

$R_1(AB)$

~~A → A~~

~~B → B~~

A → B ✓

B → A ✗

$R_2(BC)$

B → C ✓

C → B ✓

$R_3(BD)$

B → D ✓

D → B ✓

$B^+ = BCD$

(don't have A).

union of attr R_1, R_2 & R_3 are Equal to R. So, true. (Right decomposition)

Now

→ Check Dependency preservation :→

- R_1 (AB), So, It has only A, B attrs.

So, whatever Dependency we can made from these 2, make them & write in table.

We don't want Trivial F.D. :→

→ Trivial F.D. :→ In which intersection (∩) is not null.

i.e.

Ex-1 $A \to A$ has $\cap \neq \phi$ (common is A).

Ex-2 $AB \to A$ has $\cap \neq \phi$ (common is A).

We don't want these bcz, trivial are by default, true. तो दोनों ही हैं like

$A \to A$ is definitely true.

So,

→ Only take non trivial F.D. :→ $(\cap = \phi)$

→ Now:

F.D :- { A → B, B → C, C → D, D → B }.

$A^+ = ABCD$
$B^+ = BCD$
$C^+ = CDB$
$D^+ = DB$

So, acc. to these. Select from the non-trivial F.D. (which we finally select) from table. =>

Now
F.D. which we get from Table :- =>

A → B ✓
B → C ✓
C → B
B → D
D → B. ✓

Now, check whether original F.D. are possible through these or not.

So, F.D : { A → B, B → C, C → D, D → B }.
✓ ✓ ✓ ✓

Now,
C → D is not direct.
So, now take 'C' closure

$C^+ = CBD$ ∴ C → D also preserved. =>

All F.D. are preserved, True. Ans.

× ×

36. Dependency Preserving Decomposition Example

Q.2) Let R(ABCD) with F.D.
{AB → CD, D → A}.

Decompose $R_1(AD)$ & $R_2(BCD)$.

$R_1(AD)$		$R_2(BCD)$	
$A \to D$	✗	$B \to CD$	✗
$D \to A$	✓	$C \to BD$	✗
		$D \to CB$	✗
		$BC \to D$	✗
attributes of $(R_1 \cup R_2)$ make R_1	✓	$BD \to C$	✓
		$CD \to B$	✗

Now,

F.D: $\{AB \to CD, D \to A\}$.

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

$BC^+ = BC$,
$BD^+ = BDAC$
$CD^+ = CDA$.

So,

F.D. we get from table! →

$\{ D \to A, \; BD \to C \}$

Now, check ∀ original F.D! →

$\{AB \to CD, D \to A\}$. from table

✗ ✓

$AB^+ = AB$
$D^+ = DA$

We, 'can' get

$AB \to CD$, from our decomposed f.D.

Hence,

Dependency preserved not. Ans

37. Joins & Its Types :→

1. **Join:** When we have to 'join' 2 or more table, to get result.

E_No	Ename	Add.
1	Ram	Delhi
2	Varun	Chd.
3	Ravi	Chd.
4	Amit	Delhi
5	Nitin	Noida

'Employee'

Dep_No	Name	E_no
D_1	HR	1
D_2	IT	2
D_3	MRKT	4
D_4	Finance	5

'Department'

↓

Select address from Employee where Ename = 'Varun'
Output → Chd.

2. In these basic ques, we don't need Join, We easily done those by their separate tables.

Now' -

Q. Find Ename of Emp whose working in HR.
Here, HR is in department Table
& Ename is in Employee Table
So,
Here, we need Joins.

Note: There Must be some common att. in Tables to use Join. Here, E_no is common.

→ Join is Cross product
+
Select Statement (condition).

Its Types! →

- Cross Join (or) Cross product.
- Natural Join.
- Conditional Join
- Equi "
- Self "
- Outer "
  - Left Outer.
  - Right "
  - full "

(38) "Natural Join" ! →

Join = Cross product + Condi?.

(m x n).

Employee (m)

E_No	Ename	Add.
1	Ram	Delhi
2	Varun	chd.
3	Ravi	Chd.
4	Amrit	Delhi

Department (n)

Dep No.	Name	Eno.
D_1	HR	1
D_2	IT	2
D_3	MRKT	4

→ first find! - Table name

Q. "find the Emp. Names who is working in a department".

→ Here, we need both, - Table Employees & Department Table.

We do it by "Natural Join".

→ Common Att - 'E_No.'

So,

Whenever we have to equalise (d1 & d2) the values of the common Att. , then, we always use NATURAL JOIN.

→ Now, Write Query :→

```
Select Ename from Emp, Dept where
Emp.Eno = Dept.Eno ;
```

Output :→ Ram, Varun, Amrit.

'E_No' - must be same in both the Tables.
i.e (E_No / Emp_No.)

Emp		Dept	
E_No	Ename	Dept No.	E_No
1	Ram	D_1	1
1	"	D_2	2
1	"	D_3	4
2	Varun	D_1	1
2	"	D_2	2
2	"	D_3	4
3	Ravi	D_1	1
3	"	D_2	2
3	"	D_3	4
4	Amrit	D_1	1
4	"	D_2	2
4	"	D_3	4

So, finally we get 3 rows.

Eno	Ename	DeptNo	Eno
1	Ram	D_1	1
2	Varun	D_2	2
4	Amit	D_3	4

Ram
Varun
Amit

→ Ans:

Now, how to write Actual Query ! →

→ Select E.name from Emp Natural Join Dept.;

39. Self Join ! →

→ in which the Table is joined to itself.

S-id

Reference

Cid Table.

S-id	C-id	Since
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

Student Course

Study

Q: find student id who is enrolled

→ Now, we do Self Join.

- SQL query always start from = 'from' (keyword)

Query :→

```
Select * from study as T1, study as T2;
```

↑ ',' → means Cross product.

- So, here we make same Table 2 times & name as T_1 & T_2.

S-id	C-id	Since
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

T_2 (m)

J-id	S-id	Since
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2012

(n) T_1

m×n

T_1		T_2	
S_1	C_1	S_1	C_1
S_1	C_1	S_2	C_2
S_1	C_1	S_1	C_2 ✓
S_2	C_2	S_1	C_1
S_2	C_2	S_2	C_2
S_2	C_2	S_1	C_2
S_1	C_2	S_1	C_1 ✓
S_1	C_2	S_2	C_2
S_1	C_2	S_1	C_2

Condⁿ :-

1 student (S-id) need 2 course (C-id)

< > → different (not equal to).

Now, Complete Query :-

```
Select T1.Sid from Study as T1, Study as T2
where
        T1.Sid = T2.Sid      (Student name same)
            and
        T1.Cid <> T2.Cid.;   (Course diff)
```

So, final output Table :-

S_1	C_1	S_1	C_2
S_1	C_2	S_1	C_1

→ Now, we want S-id from this Table.
But T_1 & T_2 both have S-id.
So,
we can use any $\Pi_{T_1.Sid}$ or $\Pi_{T_2.Sid}$.

Ans - S_1

(Q.) EQUI - Join : → (=)

(कोई भी attr के बीच (=) लगा सकते हैं)

Employee

E_No	E_name	Add.
1	Ram	Delhi
2	Mohan	Chd.
3	Ravi	Chd.
4	Amrit	Delhi

Department / Dept.

Dept No.	Loc^n	E no.
D_1	Delhi	1
D_2	Pune	2
D_3	Patna	4

T ∧ T	=	T
T ∧ F	=	F
F ∧ T	=	F
F ∧ F	=	F

T or T	=	T
T or F	=	T
F or T	=	T
F or F	=	F

Q:- find the Emp name who worked in a department having loca" same as their address ?.

Ans:- By just seeing the 2 Table's →, Ram.

(A) Query :→

```
Select Ename from Emp, Dept where
        Emp.E-no = Dept.E-no      # common
              and
        Emp.Add = Dept.Loca" ;
```

Emp.

Delhi	1	Ram
"	1	"
"	1	"
Pune	2	Varun
"	2	"
"	2	"
"	3	Ravi
"	3	"
"	3	"
Delhi	4	Amit
"	4	"
"	4	"

Dept.

D_1	Delhi	1
D_2	pune	2
D_3	patna	4
D_1	Delhi	1
D_2	pune	2
D_3	patna	4
D_1	Delhi	1
D_2	Pune	2
D_3	patna	4
D_1	Delhi	1
D_2	pune	2
D_3	patna	4

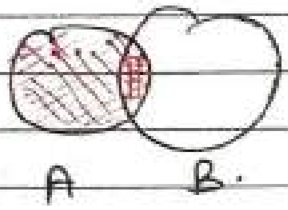
→ Ram.

Ans.

Note:- दो table के common attr. के बीच में बराबर (=) लगाते हैं & किसी भी 2 attr के बीच हम (=) लगाते हैं।

(47) <u>Left Outer Join</u> :→

- It gives the matching rows & the rows which are in left table but not in the right table.



(Natural Join + something else)

Emp		
Emp-no	E name	Dept no.
E1	Varun	D1
E2	Amrit	D2
E3	Ravi	D1
E4	Nitin	—

Dept		
Dept no.	Dname	Loc.
D1	IT	Delhi
D2	HR	Hyd.
D3	Finance	Pune

Query :-

Select emp-no, e-name, d-name, loc from emp left outer Join dept on (emp.dept no = dept.dept no.)

left

→ Cond'n of Natural Join (common att. को बराबर करेंगे।)

Rela'al Database always shows output in Table form.

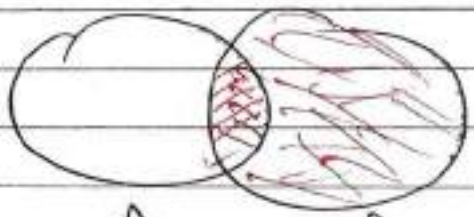
output Table:-

Emp no.	E name	d-name	Loc^n
E_1	Varun	I T	Delhi
E_2	Amrit	H R	Hyd.
E_3	Ravi	I T	Delhi
E_4	Nitin	—	—

→ (Left की तो सभी Row आती ही हैं)

92. Right Outer Join :→

It gives the matching rows & the rows which are in Right Table but not in left Table.



(Emp)

Emp-no	E name	Dept-no
E_1	Varun	D_1
E_2	Amrit	D_2
E_3	Ravi	D_3

(Dept)

Dept. No	D-name	Loc^n.
D_1	IT	Delhi
D_2	HR	Hyd
D_3	Finance	Rural
D_4	Testing	Noida

Query:-

```
Select emp-no, e-name, d-name, loc from
emp Right Outer join dept on
(emp.dept-no = dept.dept-no.)
```

(natural join.)

⇒ Output Table : →

Emp-no	E-name	D-name	Loc'n
E_1	Varun	IT	Delhi
E_2	Amit	HR	Hyd.
E_3	Rani	Finance	Pune.
–	–	Testing	Noida.

Right Table all rows.

'Full-Outer Join' :- Take the union of left outer join & Right outer join rows.

(Left O.J.) ∪ (Right O.J.)

(93) Relational Algebra : → (1970).
(procedural lang. or formal Query lang.) also.

→ have to do these things in Query : →
1.) What to do :
2.) How to do :

⇒ SQL language base is Rela'al Algebra.
(Collection of mathematical Expressions).

Rela'al algebra → SQL → No SQL
In Today's time.

"operators"

Basic operator	Derived operator.
→ projection (π)	→ Join (⋈)
→ Selection (σ)	→ intersect (∩)
→ Cross product (×)	(x ∩ y) = x − (x − y).
→ Union (∪)	
→ Rename (ρ)	→ Division (/, ÷)
→ Set Difference (−)	

» If we understand Rela"al Algebra very clearly, then we don't face any problem in understanding SQL.

(44) Projection in Rela"al algebra :→

→ Projection (π) : →

π func. is to Retrieve the data.

Roll no.	Name	Age
1	A	20
2	B	21
3	A	19
!	!	!

& (Student)

→ Query: Retrieve the roll no. from table (Student).

Date: Page:

2). $\pi_{\text{Roll no.}}(\text{Student})$.

Roll no.
1
2
3

→ Query:- $\pi_{\text{Roll no, name}}(\text{Student})$.

We fetch the 2 col^m (roll no. & name) from the student (Table).

Output →

Roll no	Name
1	A
2	B
3	A

Note: It only gives unique Answer.

Q2: # Query:- $\pi_{\text{name}}(\text{Student})$.

Name
A
B

(only name here).

→ project, we use it in last & before this we use other Operators.

(45) Selection in Relational algebra → (CR.A.).

σ (Sigma).

Roll no.	Name	Age
1	A	20
2	B	21
3	A	19.

⇒ It works on Tuples (rows).

→ First, it found rows.

π operator.

* Query: Retrieve the name of student whose Roll no = '2'

⇒ $\pi_{\text{Rollno}='2'}(\text{Student})$ → [2, B, 21]

↓ Final.

⇒ $\pi_{\text{name}}(\sigma_{\text{Rollno}='2'}(\text{Student}))$ → B

π always last में operate होता है।

* We can also find 2 at a time.

Eg:- $\pi_{\text{name, Age}}(\sigma_{\text{Rollno}='2'}(\text{student}))$ → B, 21

Note:- Projection (π) always works on Columns.
&
Selection (σ) always works on Rows. (Tuples)

46. Cross / Cartesian product in Rel'al algebra

⇒

A	B	C
1	2	3
2	1	4

R_1

C	D	E
3	4	5
2	1	2

R_2

To join, we MUST have to cross product.

→ $3 + 3 = 6$ (m+n).

$2 \times 2 = 4$ (m×n).

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

→ $(R_1 \times R_2)$

(R_1) (R_2) =

→ We need a common att. to join 2 tables. (Here, C).

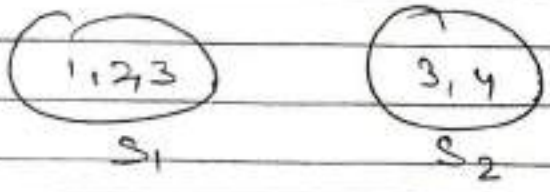
---

(47) Set Difference in R.A :→

$(A - B)$ = A but not B.

$= A \cap B'$

Ex:



$S_1 - S_2 = 1,2$. $S_2 - S_1 = 4$

* 2) It is not Commutative :→

i.e.

$A - B \neq B - A$

Condⁿ :-

1.) No. of columns must be same in both.
2.) Domain of every column must be same. (data of same type). (Ex- Numeric).

Ex :-

Rollno	Name
1	A
2	B
3	C

(Student)

Emp-No	Name
7	E
1	A

(Employee)

⇒ (Student) − (Employee)

Rollno	Name
2	B
3	C.

⇒ By default, पहले table के first column रहता है

Q:- Find the name of a person who is a student but not employee.

⇒ (Student − Employee).

→ How to Write ?

$(\Pi_{name}(student) - \Pi_{name}(Employee))$.

→ Output :-

Name
B
C

((A(B,C)) − (E,A))

(48) Union Opera" in R.A. :→

→ Same as in Set. (∪).

S_1 = {1, 2, 3} S_2 = {3, 4, 5}

$S_1 \cup S_2$ = {1, 2, 3, 4, 5}

Ex:→

Roll no	Name
1	A
2	B
3	C

(Student)

Emp. No.	Name
7	E
1	A

(Employee)

Condⁿ :→

1. No. of colᵐ must be same in no.
2. Domain of every colᵐ must be same.

(Student) ∪ (Employee)

Roll no.	Name.
1	A
2	B
3	C
7	E.

→ (Left side" Table. and ⊆).

Easy Trick!

2) $(\Pi_{name}(Student) \cup \Pi_{name}(Employee))$.

Name.
A
B
C
E.

→ (Student also & who is Employee also or both.)

Note:-1

Roll no	name
1	A
2	B
3	C

Numeric

Name	Emp No.
E	2
A	1.

alphabet

X

not Same domain.

(ye, By Default order pick होता है).

Here, not Union operation applies. (NULL).

---

(9) Division Operation in R.A. :→

→ Divis" operator is actually a Derived operator.

(Ise derived them, from normal operators).

$/, \div$ $(\times, -, \Pi, \sigma)$.

→ Ise use these

→ Query: Retrieve Sid of Students who enrolled in every Course.

all

⇒ Every (सभी) all ~~[illegible]~~ → Divis^n method.

Sid	Cid
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

'Enrolled'

Cid.
C_1
C_2

'Course'

Notat^n of Divis^n Method :→

A(x, y) / B(y) = it results 'x' values

for that there should be tuple <x, y> for every y value of Rel^n B.

Here,

[E (Sid, Cid) / C(Cid). = S_1]. Ans

i.e,

S_1 enrolled in every course (C_1 & C_2).

How it happens ?

We let, all Students are enrolled in every Course.

(for this, we do the Cross product).

⇒ $(\pi_{sid}$ (Enrolled)) × $(\pi_{cid}$ (Course)).

S_1	×	C_1
S_2		C_2.
S_3		

S_1	C_1
S_1	C_2
S_2	C_1
S_2	C_2
S_3	C_1
S_3	C_2

(Enrolled)

Now, (we subtract actual Scenario i.e. (Actual Table) from this cross product. → we get S_2, S_3 (who don't enrolled in all courses)

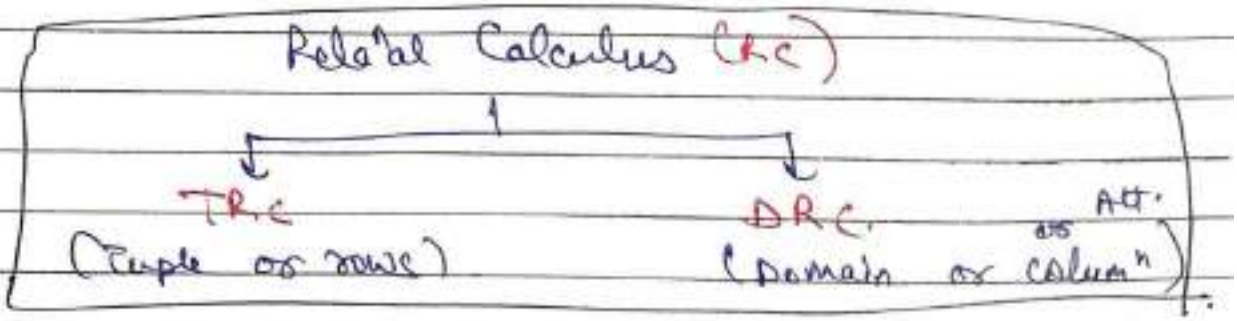
(*) Now, final ! → (Query).

$$\Pi_{sid}(Enrolled) - \left(\Pi_{sid}\left(\left(\Pi_{sid}(Enrolled)\right) \times \Pi_{cid}(course)\right) - (Enrolled)\right)$$

S_1 ~~S_2~~ ~~S_3~~ − S_2 S_3

⇒ S_1

So- Tuple Calculus in DBMS :→



⇒ Tuple Relational Calculus is a non-procedural query lang. (only tells what to do & not that how to do) unlike Relational algebra.

Date:
Page:

⇒ In Tuple Calculus, a query is expressed as

$\{ t \mid P(t) \}$

where, t = resulting tuples,
P(t) = known as predicate & these are the conditions that are used to fetch it, Thus, it generates set of all tuples t, such that predicate P(t) is true for it.

* Operations :→

→ P(t) may have various conditions logically combined with OR (∨), AND (∧), NOT (¬).

→ Atomic func's :- (We use one variable here, Ex - (Supply Ids. from a Supply table) → only 1 condⁿ.

→ It also uses quantifiers :

• ∃ t ∈ r (Q(t)) = "there exists" a tuple in t in relⁿ r such that predicate Q(t) is true.

• ∀ t ∈ r (Q(t)) = Q(t) is true "for all" tuples in relⁿ r.

* Unsafe Expression :→ (U, E).

Supplier (Table).

- { S.name | ¬ Supplier (S) }.
  (not sign)

Here, It means that, all the Supplier name who are not in the Supplier Table.

So, here, ans is ∞. (So, ∞ loop ही ans)

↳ Unsafe Expres".

(These are not in the Rel'al algebra)

- Both TRC & RA have same Expressive powers.

↓

unsafe exp" ↔ [U, E (x).]

(the query which we can write in RA, can also write in TRC & even in DRC.)

But, SQL has more powers than these. SQL has more xtra func".

* Examples :→

Q-1. Write a query in SQL to display {name of Supplier}.

Q-2. Write a query in SQL to display {name of Parts whose color is Red}.

('Catalog' सूची)

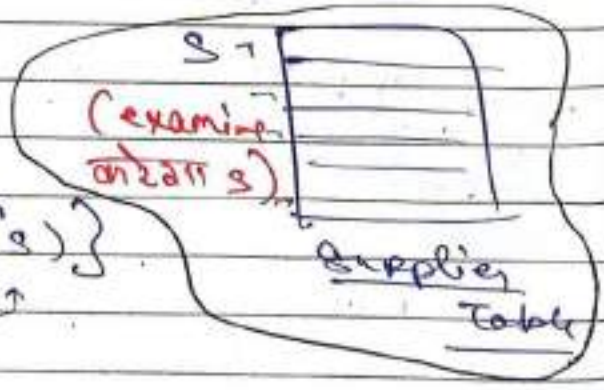
Q-3 Write a query in SQL to display SID of Suppliers whose name is 'Varun' & address is 'Chandigarh'.

Q-4 Write a query in SQL to display SID of suppliers who supplied same parts.

Q-5 Write a query in SQL to display Sname of suppliers who supplied same parts.

Q-6 Write a query in SQL to display Sname of Suppliers who supplied some red color parts.

(1-)



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

(2-)

$\{ p.Pname \mid Parts(p) \wedge p.color = 'Red' \}$.

(3-) $\{ S.SID \mid Supplier(S) \wedge S.name = 'Varun' \wedge S.add = 'Chandigarh' \}$.

(4-) Here, it extract ans. from the Catalog Table.

$\{ C.SID \mid Catalog(C) \}$.

Sid तो Catalog Table में भी दिया गया।
But, Sname only in Supplier table.

(5) Here, we need 2 Table → Supplier, Catalog.

&

final ans. from supplier Table.

(There exists).

$$\{ S.Sname \mid Supplier(S) \wedge \exists c\,(Catalog(c)) \wedge S.Sid = C.Sid \}$$

Here,

S	→ free variable
c	→ bounded variable

(जिसके साथ हम ∃ या ∀ लगाते।)

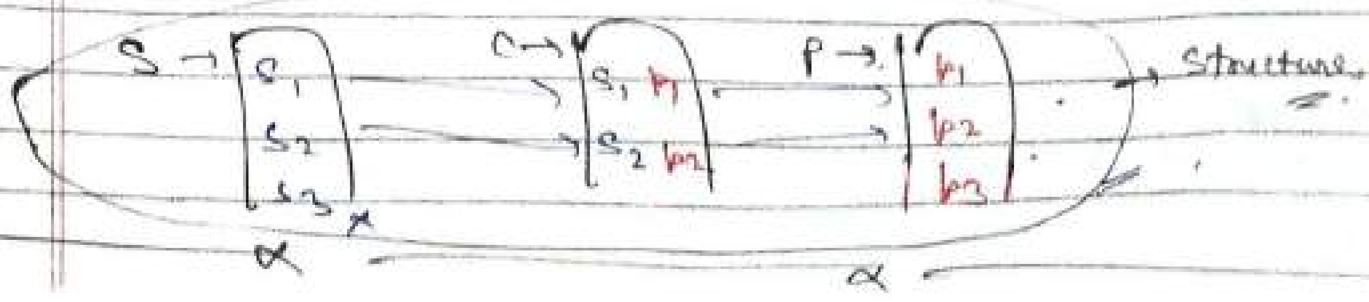
(6) Here, we need 3 Table → Supplier, Catalog, parts.

But,
ans from Supplier Table.

$$\{ S.Sname \mid Supplier(S) \wedge \exists c\,(Catalog(c)) \wedge \exists p\,(parts(p)) \wedge S.Sid = C.Sid \wedge C.pid = p.pid \wedge p.Color = 'Red' \}$$

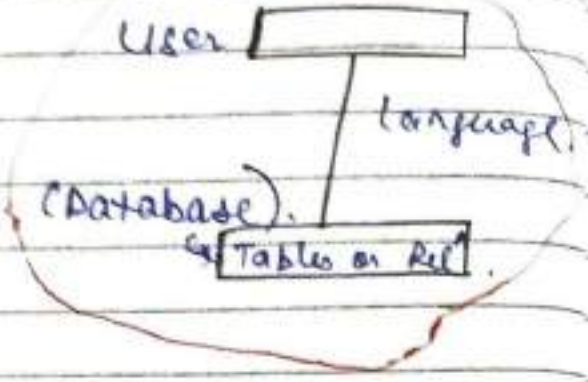
(S variable for Supplier Table.
c —— " —— Catalog ——.
p —— " —— parts ——.)

(S → free variable
c, p → bounded ".)



1) Intro to SQL :- (Structured Query language)

(1970)



→ EF codd gave theory of store & fetching of data.

Rel'al Model:

By the help of Relational algebra & TR: (Tuple Rel'al calculus)

→ IBM converts this concept of database of EF codd into the Structure Query language (SQL).

→ Before,
SEQUEL → Simple English Query language

Then,
SQL → Structure Query lang.

→ then, Oracle & Microsoft, etc. comes into this field of databases.

→ Now, No SQL is come into market.

→ 10% data → Structure
90% data → Unstructure (Apart, Mp4 etc)

2) Properties :-

→ General purpose! → Applicability on multiple places. Ex- C/C++, but Domain Specific! → only use in any particular field. Ex!-
SQL is in only Relational Database. (When we have to store & fetch data from the Relations (Tables), they only we use SQL). and except this, there is no general use of SQL.

1) SQL is a domain-Specific language.
2) SQL is a declarative language.

→ declarative lang → what to do.
→ procedural lang → what to do & how to do.
↳ (There is procedure that how to do)

later on,
PL SQL → procedural language.

3) DDL, DML, DCL, TCL
These are diff commands in SQL etc to create, insert, fetch, control data. etc.

4) Keys & constraints. (Ex- P.K, F.K, C.key).
↓
(primary key (P.K.) constraint,
F.K. " ,
Not, NULL, default).

aggregate → माना, family, कितने लोग, संख्या करना.

Exist / Not Exist.

5.) operators (like, between, In, Not in, Conditional).

→ We use these operators interally to use Query.

Ex:- IRCTC :- We query on this app by APIs (Applica' programming interface).

Train no', seat no',
IRCTC based on structured Data.
(Train की Info. Tables के form में ही show होती है).

6.) clauses (distinct, order by, group by, from having).
We also majorly use this in query.

7.) aggregate func's :- (max, average, count, min, sum,)

8.) Join & Nested Query :-

9.) PL SQL (Triggers, func', Cursor, procedure)
same as in Java.

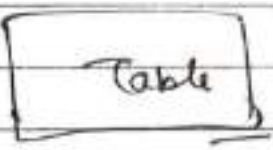
(#) What to do :-
In libraries of oracle or like SQL series, it is already defined that what select do, & what other commands 'do' so, we don't need to give procedure that how to do.

(52) All Types of SQL Commands :-

* DDL Commands :- (Data Definition language).

→ Deals with Schema (structure | Table).

- Create
- Alter (change Table Specifica's).
- Drop. (Remove Table)
- Truncate. (Remove Data).
- Rename.



* DML Commands :- (Data manipula^n language).

→ If we have to change anything in data, then we use DML. (Manipulation).

- Select
- Insert.
- Update.
- Delete.

* DCL Commands :- (Data Control language)

→ who has control over the data.
who is authorised on that data.

- Grant (give permis^n)
- Revoke. (take back that permis^n).

* TCL commands :- (Transac^n control lang)

→ Mainly used in Transac^n's

S31

Ex:- Shopping, Bill payment - online.

- Commit (If Transac" is commit (done), then save data)
- Roll back. (If Transac" fails).
- Save point. (Ex: we can save after 20-30% of transac").

Ex:- Like in Downloading & Buffering, we have save points. Ex: If downloads fail on 90%, then it it starts from '0' again as from 90%). This is save point.

Note:- To Execute these commands, we can use Oracle, My SQL, Server, etc.

(*) Constraints :- → We mostly use these Constraints in DDL & DML.

- Primary key (for uniqueness)
- Foreign key (" reference).
- Check
- Unique
- Default
- Not Null (mandatory *)

→ Constraints are rules which we made for store the data.

S3- Create Table in SQL with execution :- →

like SQL

→ In DDL Commands, very first command is Create Table.

→ Now, we use Oracle Syntax. (In other like My SQL, SQL Server, there is only little diff.).

```
Create table <table-name>          // no space
(                                  Ex:- abc_xyz.
 Column name  datatype ,
 Column 2 name  datatype ,
 Column 3 name  datatype
);
desc table-name ;                  // describe.
```

Ex:-

```
Create table emp
(
id int ,                           → fixed type of datatype
name varchar2 (20) ,               → variable  // ──→
Salary number (10)                     (can be increased)
);
desc emp ;
```

ex- (2 bits fixed)

54. Alter Command (DDL) in SQL :→

Use → (बनी हुई टेबल में अगर कुछ change करना है तो।)

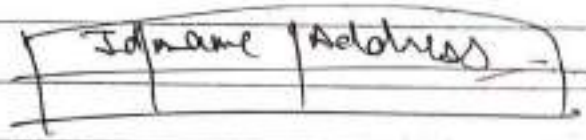
Ex:-

Student

id (int)	name (varchar(20))

→ already है।

→ Add & Remove column/s.
→ Modify datatype.
→ Modify datatype length.
→ Add constraints
→ Remove Constraints
→ Rename col^m n / table.



(1) Alter table student Add address varchar(30);

↳ (,) ;.

* multiple columns.

⇒ Modify datatype:-

Ex:- from int to varchar

```
* Create table emp
(
id int,
name varchar(10)
);
alter table emp add address varchar(10);
desc emp1;
alter table emp drop column address;
  "   "   " modify id varchar(30);
  "   "   " rename column id to reg_no;
  "   "   " rename to emp1;
alter table emp1 add primary key (reg_no);
```

(55) M/B Alter & Update : →

(*) Alter :- only Change in structure, (DDL) & not in data.

Ex:-

Emp

ID	name	Salary	Email
1	A	10k	1
2	B	20k	1
3	C	30k	1

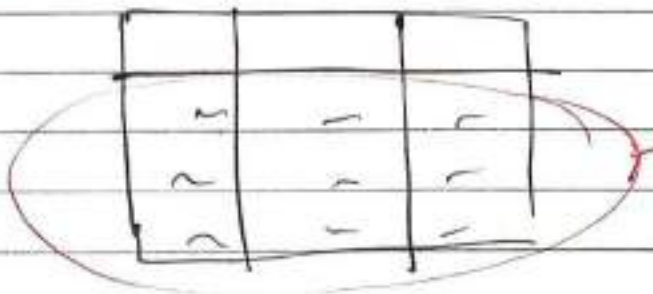
} → can only change in this structure.

→ Id → varchar
→ name varchar (10)
↓
20

→ ID → Eid.
→ Emp → Emp_detail

Col^m (name change)
(table name =)

(*) Update :- can only change in data, (DML) & not in structure.



→ can only change in data.

```
update Emp
Set Salary = Salary * 2;
```
→ for all students

```
update Emp
Set Salary = Salary * 2
Where id = 1;
```
→ for only student of id = 1.

S6 D|B Delete, Drop & Truncate :-

(1) Delete :-
(DML).

Syntax :-
```
Delete from table name
```

Ex1: `Delete from Student` → All rows deleted.

`Desc Student;`

Student.

ID	name
1	A
2	B
3	C

ID	name

Note:- But here, also flexibility. We can apply condⁿ for delete some particular rows.

Ex:
```
Delete from Student
Where id = 1;
```

⇒ It is a slower process. - bcz, it also creates log.

> log :- Error file Completely delete होने से पहले, Computer में temp. file बनती है, to restore data, इसे log कहते हैं।

Ex: On delete, our data comes into Recycle Bin. (It is log file).

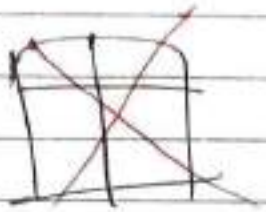
Rollback;

(possible, but it creates log).

(v) Drop: (DDL) Delete the complete table at once.

```
Drop table student;
Desc student;
```

→ no object found. (all delete).



Rollback; ✗ (not possible).

(vi) Truncate: (DDL)

```
Truncate student;
Desc student;
```

All rows delete at once.

ID	name

→ faster process.

Rollback; ✗ (but no log).

S7. Constraints in SQL:

→ Constraints means (conditions) restrict.

Ex: In gmail: (2 account Ids can't be same)

anurag@gmail.com ✗ (not present)

anurag13@gmail.com ✓ (present).

Condi^n → that new mail-id must be unique.

Condi^n → length of password must be 6 digit, Capital letter etc.

Note:- We apply conditions on Attributes (column). and the data which fulfill these condi^n, only comes into the column.

1.) Unique :→ Ex:- Mobile no (no duplicate can exist in that m.no. col^n.)

2.) Not Null :→ *

(We can't skip that col^n),

i.e, * - Mandatory. (value देनी ही पड़ेगी)

* 3.) Primary key = Unique + Not Null
Ex:- Reg. no.

4.) Check :→

age int

? → x (-10)

check (age > 18) → 10 ✗
→ 20 ✓

i.e, (2 particular domain में से ही User Id बनानी है)

Domain - fixed.

5.1. Foreign key :-

6.1. Default :-

Salary int default 10k.

If we don't fill anything, they default Salary col^m take value → 10k

7) We can use multiple constraint in a col^m also.

---

58. SQL Queries & Sub-Queries :- → (from Beginning to end).

Q:-

Emp.

E_id	E_name	Dept	Salary
1	Ram	HR	10k
2	Amrit	MRKT	20k
3	Rani	HR	30k
4	Nitin	MRKT	40k
5	Varun	IT	50k.

(i) Write a SQL Query to display maximum Salary from Emp table.

(ii) Write a SQL Query to Display Employee name, who is taking max^m Salary.

Note:- aggregate func :- for max, count, average,

<> → not equal to ·, = ≠, ≠

(i). Select max (salary) from Emp;

Output → 50000

(ii) We here, take help of Nested Query or Sub-Query. (Query inside a Query).

```
Select E.name from Emp where
Salary = (Select max(Salary) from Emp);
```

Outer Query. Inner Query

Output → Varun

→ Inner Query execute first 1 time

Outer: 10k = 50k ✗
20k = 50k ✗
50k = 50k ✓

(iii) Write a SQL query to display 2nd highest Salary from Emp table ?

(iv) Write a SQL query to display Employee name who is taking 2nd highest Salary ?

(iii)

Logic → Highest Salary - remove &
Rest Salary - find highest
(It is 2nd highest in table)

Outer: 10k, 20k, 30k, 40k, 50k (50k)
Max ← Inner
40k

(iii) Query :-

```
Select max (Salary) from Emp where
Salary <> (Select max (Salary) from Emp);
```

Sal.

Ask it.

(iv). { = , when we have to compare only with 1 value.
{ in , when we have to compare with multiple values.

Ex:-

2 = 2 ✓

{ 2 = 1,3,2,4,5 (Wrong way)
{ 2 in (1,3,2,4,5) ✓ (True)

Query :- ✗ only solid name in (iii)

```
Select E-name from Emp where
Salary = ( Select max (Salary) from Emp where
Salary <> (Select max (Salary) from Emp));
```

Output written.

60. (v) Write a Query to display all the dept names along with no of Emps. working in that ?

Output :-

HR	2
MRKT.	2
I T.	1

Ex:- In a university, find how many students are there in diff branches.

AI → 101
CSE → 100

Note:- Here, we use a special func.

[group by] clause.

(forms grp of dept.).

HR HR	2
MRKT MRKT	2
IT.	1

Note:- Condin of group by → जो att. को नाम हम group by के साथ use कर रहे हैं, सिर्फ वही select ____ के साथ लिख सकते हैं

If,
we have to write other att. in Select other than the att of group by func, then we have to use aggregate funcs.

- Query:-
  Select dept from emp group by dept;

  [Output] →

  HR
  HR
  MRKT
  MRKT
  IT

→ Same.

Note:- Count (dept) or Count (*).

- Query:- (aggregate func.)
  Select dept, Count(*) from emp group by dept;

Output :-
HR - 2
MRKT - 2
IT - 1

— X —

61. (ii) Write a Query to display all the dept names where no of Emps are less than 2.

HR 2
MRKT 2
IT 1 ✓

Note:- 'Where' clause works on complete Table but now we group by this table. Hence, group by & where are independent. Not help each other.
Hence,
we use 'having'.

Query:-

```
Select dept from Emp group by dept
having Count(*) < 2;
```

Output → IT.

Q ⇒. If they ask of Employee name in this query, then → (Varun → output)

Query →

```
Select E-name from Emp where dept In
(Slect dept from Emp group by dept
having Count(*) <2);
```

↳ Varun.

In
× HR IT
✓ MRKT
✓ IT ✓

62

Que) Write a Query to display highest Salary department wise and name of emp who is taking that Salary.

Note: A SQL query always starts from 'from'

Select ... from table name

Note: Select max (Y) from ... group by X;

Here, It is Right, bcz we use aggregate func 'max' with Y i.e. max(Y) & if we don't use max, then we can only use X, bcz group by X. (same).

Query:-

```
Select Ename from Emp where Salary
In (Select max (Salary) from Emp
group by dept );
```

Output → Rani, Nitin, Varun ... so on.

Steps →

			Outer	Inner
HR	HR - 30,000	30k	40k x	30k
HR			30k ✓	40k
MRKT	MRKT - 40,000	40k	40k ✓	50k
MRKT	IT - 50,000	50k	50k ✓	
IT				

* means All attributes.

Simple in Query.

(63) Use of IN and Not IN :-

1= (1,2,3,4,5) X (wrong way).

1= 2 false. (but, Right way).

Emp

E_id	E_name	Address
1	Ravi	Chd
2	Varun	Delhi
3	Nitin	Pune
4	Robin	Bangalore
5	Ammy	Chd.

Project

E_id	P_id	P_name	Loc^n
1	P_1	IoT	Bangalore
5	P_2	Big Data	Delhi
3	P_3	Retail	Mumbai
4	P_4	Android	Hyderabad

Q:- Detail of Emp whose address is either Delhi or Chd or Pune ;

Query:-

```
Select * from Emp where Address = 'Delhi';
```

Output:- | 2 | Varun | Delhi |

⇒ But, We have 3 cities. So,

Query:-

```
Select * from Emp where
        Address In ('Delhi', 'Pune', 'Chd');
```

Output:

1	Ravi	Chd
2	Varun	Delhi
3	Nitin	Pune.
5	Ammy	Chd.

Chd ✓
Delhi ✓
Pune ✓
~~Bangalore~~ ✗
Chd. ✓

Next Query → Query (Query)

NOW, NOT IN :- (Rows not included)

Query :-

```
Select * from Emp Where
Address Not In ('Delhi', 'Pune', 'Chd');
```

Output :-

4	Robin	Bangalore

Chd ×
Delhi ×
Pune ×
Bangalore ✓ th
Chd, ×

× ——— ×

64- Use of IN & NOT IN in Subquery :-
(Same 2 tables of before).

Query :- find the name of Emps who are working on a project ?

⇒ Suggestion :- first make Dummy Tables.

⇒ Here, we need both 2 Tables

Both tables have common - EID
so, we compare by taking EID.

⇒ Nested → Bottom up → नीचे से ऊपर

Means पहले ऊपर वाली query & then बाहर वाली

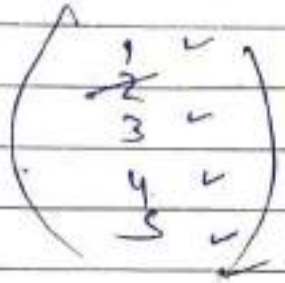
Inner Query in Nested Query - Runs 1 time.

* Query :-

```
Select Ename from Emp where Eid
In ( Select Distinct (Eid) from Project );
```

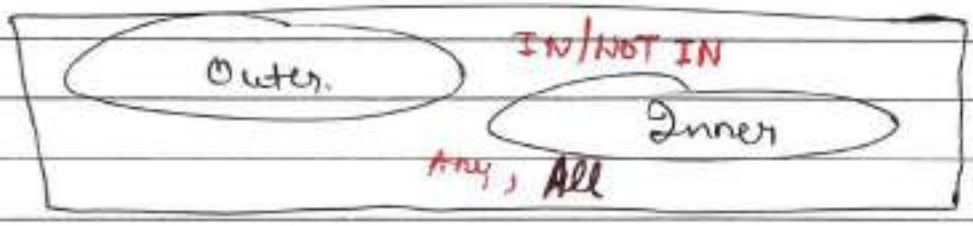
Inner Query.

Output :- Ravi
Nitin
Robin
Anuj.



Q5. EXIST & NOT EXIST Subqueries :-

↳ We use these in Correlated Nested Query.



⇒ In nested query. - We use IN / NOT IN
&
In Corelated nested Query - We use EXIST / NOT EXIST

Query → Find the detail of Emp who is working on at least one Project ?
(Same 2 Tables)

Note :- In nested Query → Inner Query executes first, & then compare its output with Outer Query.
In Corelated Nested Query :- the one row of Outer Query is compared with all rows of Inner Query.
i.e. Top to Down Approach.

Null → means value is not available.
Empty

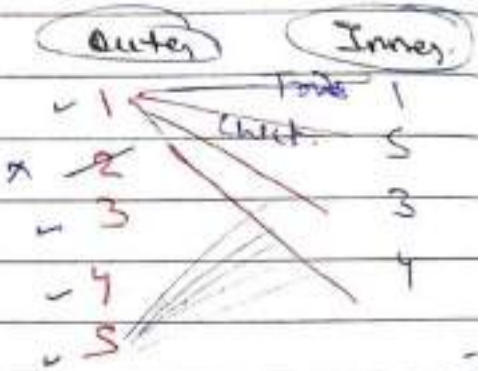
Query:

```
Select * from Emp where Eid
Exists (Select Eid from Project where
        Emp.Eid = project.Eid);
```

Note: Exist / not Exist gives True or False.

Output:

1	Ravi	Chd
3	Nitin	Pune
4	Robin	Bang.
5	Ammy	Chd.



66. Aggregate Funcs in SQL: →

max, Min, Count, Avg, Sum.
↳ (Total no of values in a Table).

Emp

E_id	E_name	Dept	Salary
1	Ram	HR	10k
2	Amit	MRKT	20k
3	Ravi	HR	30k
4	Nitin	MRKT	30k
5	Varun	IT	50k
6	Sandy	Testing	Null

- Query for Max. Salary: →

```
Select max(Salary) from Emp;
```

Output → 50k

⇒ If 50k repeats 2 times, then it also give output 2 times.

4 Same for Min

→ Select Min (Salary) from Emp;

Output → 10k.

* COUNT :-

Count (*) means no of rows in the Table

→ Select Count (*) from Emp;

Output → 6 . 6 rows.

→ Select Count (Salary) from Emp;

Output → 5 (bcz one value is Null)

* Distinct :-

→ Select Distinct (Count (Salary)) from Emp;

Output → 4 . (bcz 30k is 2 times).

(*) SUM :-

Select Sum (Salary) from Emp;

Output → 140 k . (Sum of all salary)

→ Select Distinct (Sum (Salary)) from Emp;

Output → 110 k (30k repeats).

(*) AVG :-

Avg (salary) = Sum (salary) / Count (salary)

= 140K / 5

= 28K ✓

→ Select Avg (Salary) from Emp;

Output :- 28K.

→ Select Distinct (Avg (Salary)) from Emp;

Output :- 27,500.

Distinct (Avg (Salary)) = Distinct (Sum (Salary)) / Distinct (Count (Salary))

= 110K / 4

= 27,500 ✓

---

(6) Co-related Subquery in SQL :- (Synchronized Query).

- It is a subquery that uses values from Outer Query.
- Top to Down Approach :- (Outer to Inner)
- for One row of outer Table it compare with every row of Inner Table.

→ Returns True / False.

Emp

Eid	name	address
1	A	Delhi
2	B	Pune
3	A	Chd.
4	B	Delhi
5	C	pune
6	D	Mumbai
7	E	Hyd.

Dept

D-id	D-name	Eid
D_1	HR	1
D_2	IT	2
D_3	MRKT	3
D_4	Testing	4

Query:- Find all Employee detail who work in a department.

→ True होने पर ही उन्हें Employee Detail दी जाएगी।

→ Query:-

```
Select * from Emp where
    exists ( Select * from dept where
        dept.Eid = Emp.Eid );
```

Outputs:-

Eid	name	address
1	A	Delhi
2	B	Pune
3	A	Chd
4	B	Delhi

Inner

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

False

---

68 DB Joins, Nested Subquery & Corelated Subquery:-

→ Nested → Bottom up. (अंदर से बाहर)
→ Corelated → Top down Approach (बाहर से अंदर)
→ Joins → Cross product + Condi.

Emp.

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

Dept.

Dept no	name	E_id
D_1	IT	1
D_2	HR	2
D_3	MKT	3

Q:- Detail of all Emp who works in any dept.

(Here, we need both 2 Tables).

→ Nested Subquery →

```
Select * from Emp where E_id in
(Select e_id from dept);
```

Output →

1	A
2	B
3	C

→ Correlated Subquery :→

```
Select * from emp where exists
(select id from dept where
emp.e_id = dept.e_id);
```

e_id is common in both Table

Output :→

1	A
2	B
3	C

Inner	Outer
1 = 1 → True	1 = 5
2 = 1	2 = 5
3 = 1	3 = 5

False

→ If 1st Table → m rows
& 2nd Table → n rows

then, Total Comparison → m x n

→ JOINS :→

Cross product + Condi°

It creates a new Table with (m x n) rows.

→ then,
check [Condi° → emp.eid = dept.eid]

Note:→ Joins is faster than Related Subquery bcz it made a table of rows (m x n) at Once. While in Related, we again & again compare one by one row of Outer Table with all Rows of Inner Table. So, it takes time. △ But, Joins take more space. (bcz, big Table of (m x n) rows).

---

69. Find Nth Highest Salary using SQL :→

Emp.

ID	Salary
1	10k
2	20k
3	20k
4	30k
5	40k
6	50k.

→ Highest Salary :-

```
Select max (Salary) from Emp;
```

Output → 80k.

→ 2nd Highest Salary :-
(Nested Query).

```
Select max (Salary) from Emp where
Salary Not In (Select max (Salary) from
Emp);
```

Output :- 40K

Now

How to recognise Correlated Subquery ?
जहाँ पर outer query का कोई ना कोई value हमने inner use किया है।

How to find N^{th} Highest Salary ! →

Query :- Best method → learn it.

```
Select eid, Salary from Emp e1 where
N-1 = (Select Count (Distinct Salary) from
Emp e2 where
e2.Salary > e1.Salary);
```

↳ Co-related Subquery

Here
we make 2 alias of Emp i.e. e1 & e2
(Dummy)

Ex:- on Self :-

E_1

ID	Salary
1	10k
2	20k
3	30k
4	30k
5	40k
6	50k

E_2

ID	Salary	
1	10k	> 10k (false)
2	20k	> 10k count = 1
3	20k	> 10 count = 1 (dup. occ.)
4	30k	> 10 count = 2
5	40k	> 10 c = 3
6	50k.	> 10 c = 4

⇒ Why we make 2 (E_1 & E_2) :-
bcz employee use ur 2 times. (E_1 & E_2).

If we don't create E_1 & E_2, then
$e_2.salary > e_1.salary$

↳ It means, employee salary > employee salary.
↳ no Mean, ×.

↳ E_1 & E_2.

Ex:-
1st row: 10k > 20k F
2nd row: 20k > 20 F
3rd row: 30k > 20 F
4th row: 30k > 20 count = 1
5th row: 40k > 20 count = 2
6th row: 50k. > 20 count = 3 } count = 3.

Ex:- Let's we have to find 4th highest, then
N-1 ⇒ 4-1 [= 3].
&

```
Select id, Salary
3 =
```

When we get 3 as count from this Inner query, then, our output comes.

And, we get, Count = 3
from the 2nd row. bcz,
from 1st Row we got count = 4.
Hence,
Ans is data of 2nd Row

Output:→

ID	Salary
2	20k.

&

20k is our 4th highest Salary.

50k - 1st
40k - 2nd
30k. - 3rd
20k - 4th.

Ex:1 for 3rd highest Salary!?

N=3, N-1 = 2

Hence,
for which Row we got Count = 2,
that row will be our output.

10k > 30 F
20k > 30 F
20k > 30 F
30k > 30 F
40k > 30 T , count = 1
50k. > 30. T , count = 2

4th Row is our output.

Output: 4, 30k.

& 30k is our 3rd highest Salary.

3 Imp. Questions on SQL basic concepts :-

Q.1 You need to display last name of Employees who have 'A' as 2nd character in their names. Which SQL statement display the required result?

a) Select last_name from Emp where last_name like '_A%' ;

b) — " — last name = '*A%' ×

c) — " — name like '%A%'

d) — " — like '*A%'

Note :- Questions like, 2nd letter same, Salary be of only 5 nos, like that.

— based on 'Like' command.

% → Any value.
_ → fixed a place for a value.

Ex :- i) %A% Anything AABA ✓

ii) 2nd letter → _A% (one pair is fixed)

iii) 4th character must be A → ___A%

iv) 2nd last letter must be A → %A_

(2.) A Command to remove relation from SQL database → [Table].

a) Delete table < table name > .
b) Drop table < ——— > 3. Ans
c) Erase table < ——— >
d) Alter table < ——— > .

→ DDL Commands deal with Schema (Table structure).

Create, drop, Alter → DDL Commands.

→ Alter table → to change anything in table.

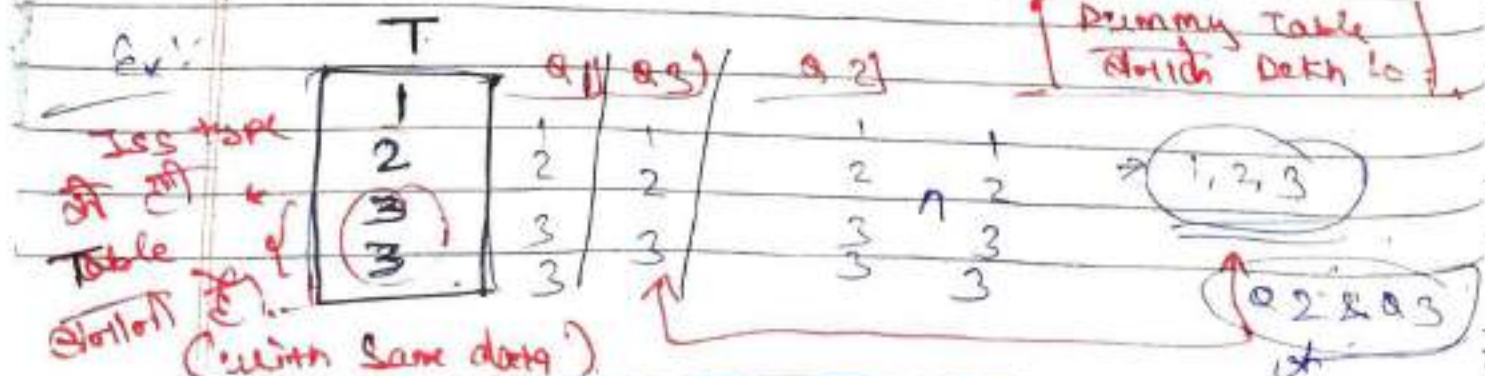
Delete table, Erase table. ✗ (Even not a command)

(3.) In the following, Scheme R is R(a,b).

Q1: Select * from R
Q2: (Select * from R) Intersect (Select * from R)
Q3: Select distinct * from R

a) Q1, Q2, Q3 produce same result
b) only Q1, Q2 ——— "
c) only Q2, Q3 ——— " ——— 3 Ans.
d) Q1, Q2, Q3 produce diff. result.

Ex:



71. PL-SQL :-

(procedural - SQL).

- SQL → Declarative in nature (only 'What to Do').
- PL-SQL → procedural (1.) What to Do, 2.) How to Do.).

(programming flavour देता है।).

[PL - SQL]

- Function
- *procedure
- *Trigger
- Cursor

- In SQL → (Write a query).
- In PL-SQL → (Write a program, or Write a PL-SQL Code)

(program write करते हैं।).

- Its Block Code has 3 parts :-

```
declare
a int
b int
c int
begin
a := 10;
b := 20;
c := a+b;
end;
```

Declaration (a int)
Executable Code (begin … end)
Exception handling (error).

in C++ { }

↳ means (If manually, we have to raise any errors), like $\frac{x}{0}$, we define it in Exception handling.

CPU always performs in RAM.
CPU never works on Hard Disk

CPU - fast.
Hard Disk → slow

---

72. TRANSACTION CONCURRENCY :→

Transac°: It is a set of operations used to perform a logical unit of work.

(कोई भी डेटा change करे देते, Database के अंदर, कोई भी डेटा Database को Read करे देते, that is also a part of transac°.)

Ex-1

→ When we withdraw our money from ATM, then we have to perform a ~~test~~ set of opera°, these set of opera's called as Transaction.

→ [Work - Withdraw Money.]

A transaction generally represent change in database.

Database Transactions has 2 operations :→ Read & Write. (Commit) - we also use this.

[Read is the access of Database.
Write is change in database, we made.]

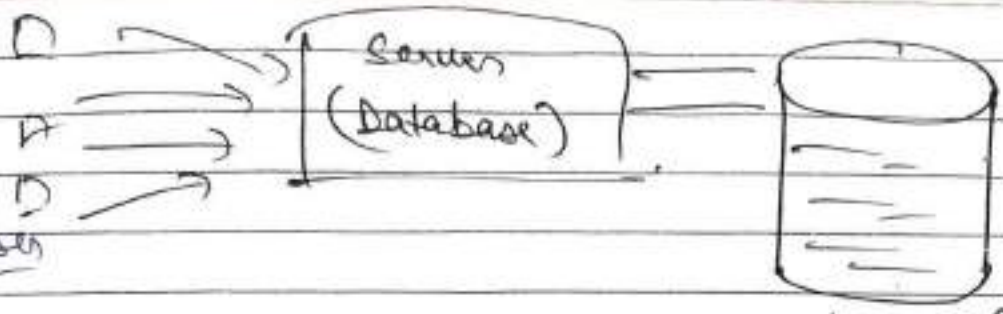
→ When we read or access any data from HDD (Hard drives), then it comes to RAM (where we perform opera's).

Commit → Whatever changes we made, save that permanent in Hard Disk.
(Update data in HDD)

In C++ for assignment, =
In PL SQL , : =

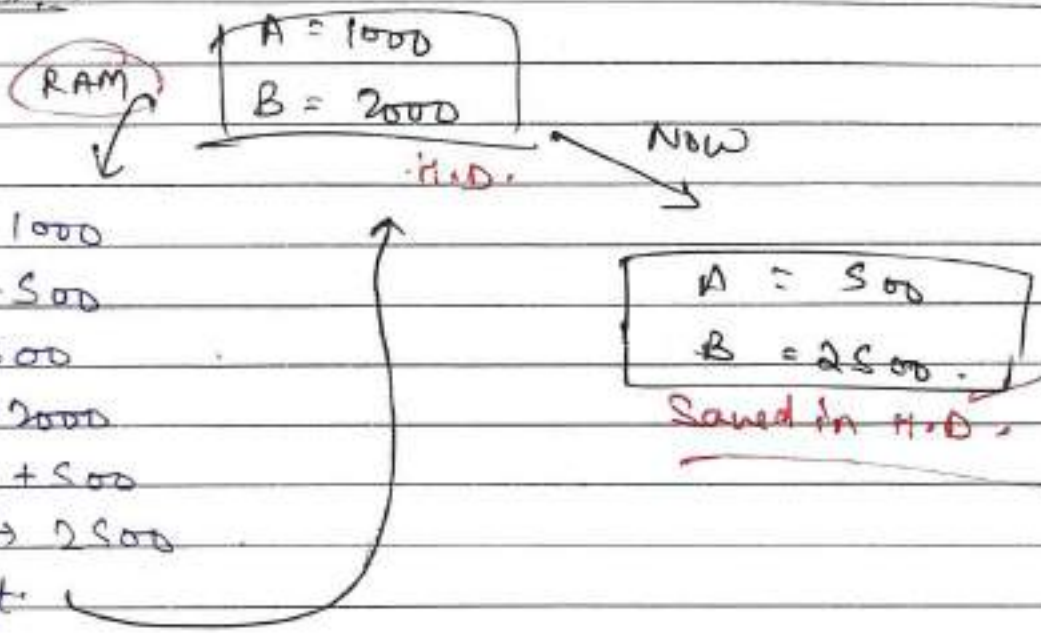
C++, = =
PLSQL, =

(37)



H.D. (Hard Disk)

Transfer :-



(73) ACID properties of a Transaction :-

A → Atomicity
C → Consistency
I → Isolation
D → Durability

} At back End

Atomicity → Either all or None.

Ex:- T_1 (Transaction)

R(A)
A = A - 50
W(A)
R(B)
- - -
Commit

Rollback

(अगर Commit से पहले कहीं भी fail हो गया, then Roll Back)

या तो सभी oper° Execute होते, Commit होंगे।
वरना कुछ न हो Roll back होगा।

Note:- A failed Transaction cannot be Resumed.
A failed Transac° will always restart.

Consistency :→

Before trans. start And,
After the trans. completed,

Sum of money should be same.

Ex:

A = 2000
B = 3000
5000

A ——1000——→ B

T_1

R(A) 2000
A = A - 1000
W(A) 1000
R(B) 3000
B = B + 1000
W(B) 4000
Commit

A = 1000
B = 4000
5000

Sum is same

Isolation :→

T_1	T_2	T_3
R(A)		
	R(B)	
		R(A)

parallel Transac's.

→ We try to convert a parallel schedule into a Serial schedule. (Conceptually)

CPU Speed is in MIPS (Million instruc" per second).
Q Hard Disk → 10/20 instruc" per second.
Ans CPU never compatible with HDD for execution.

⇒ Then,
Serial Schedule is always Consistent

# Parallel	# Schedule
$T_1 \rightleftarrows T_2$	$T_1 \rightarrow T_2$
	$T_2 \rightarrow T_1$

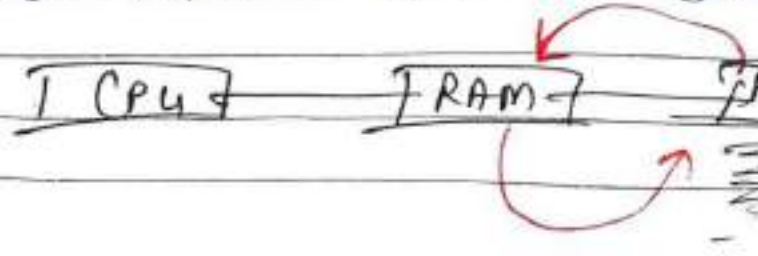
* Durability :→
Whatever changes we made, they must be permanent. i.e.
(Update for lifetime until we again update the data).

⇒ That's why, we save data in HDD for Durability.

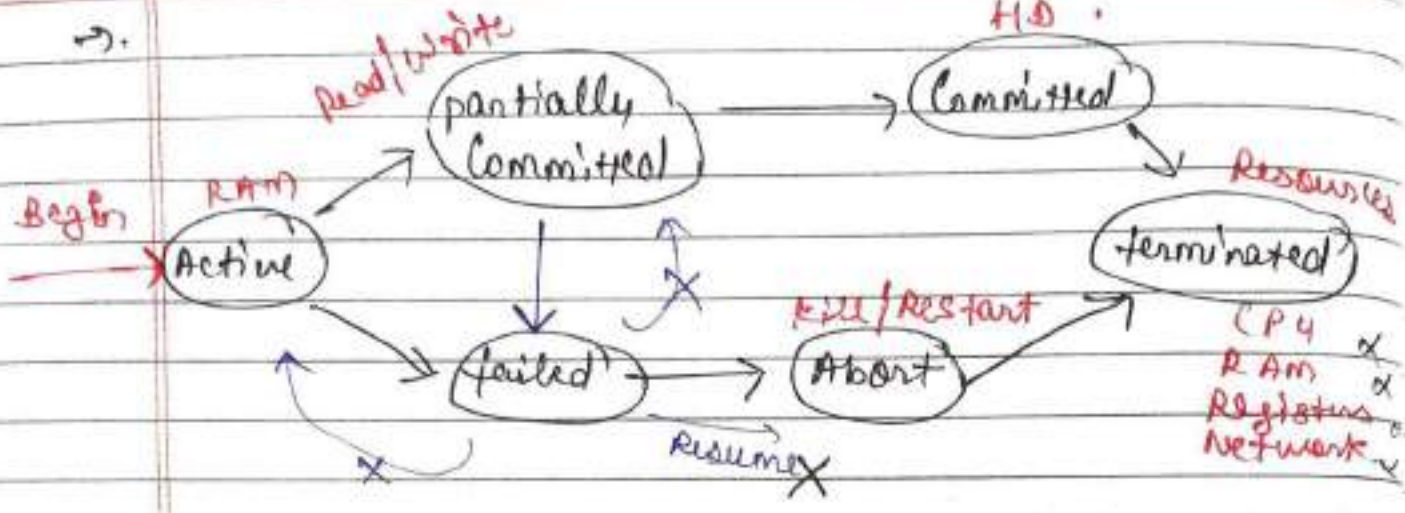
74. Transaction States :→

⇒ At now, Transac" is in passive state. And as we start Executing it, it comes into Active State.

* In terms of O.S. :→ When we write a program in C++ & save it. Now, the program is in Hard Disk in ideal state. But, when we start executing or compiling, it comes into RAM that we called ACTIVE STATE.



→



→ partially Committed! →
All oper's are done except Commit.
ie. If N operations.
then
(N-1) is done.

All opers are stored in local memory/ram/ shared memory until now.

→ Committed! -
Changes are now saved to Hard Disk.

→ terminated! →
Here, we deallocate our resources
ie. free all Resources. bc,
(Resources are limited).
↓
Now, they move to anyone else.

→ failed: → power failure, switch damage, etc.
(failed either from Active or partially committed state)

→ Abort! → Rollback the opera's & Restart the opera's.

---

(25) SCHEDULE! → (Serial vs parallel Schedule).

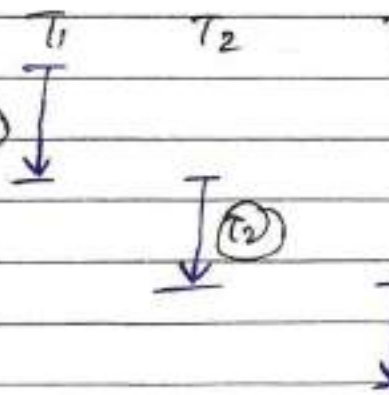
→ Schedule! → It is chronological execution sequence of multiple transactions.

T_1 T_2 T_3 — — — T_n

↳ What is the sequence that these transac. are getting executed, that's called Schedule.

Serial Schedule! → Until a transac gets completed, no other trans. can interfere.

All transa's executed by a serial sequence.



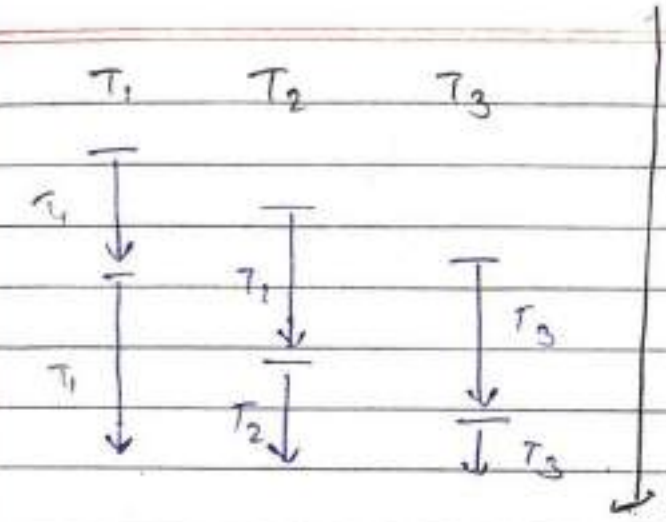
Advantage →
→ Consistent (bcz, no one can interfere).

Disadvantage →
→ When T_1 gets executed, T_2 & T_3 are in waiting.
→ Performance degrade.

Parallel Schedule! →
→ We can switch to multiple transac's at a time. Ie,
→ Multiple transac can execute at a same time.

Ex!→ Banking Online System! → Multiple users can use at a time.

Throughput → No. of transc's executed / time.



→ Advantage :-
→ performance Increased i.e, (Throughput is high.).

→ Disadvantage :-
→ problm may occur . & (Inconsistent)

(*) Nowadays,
(Parallel Schedule is more performed.)
↳ bcz good performance in less time.

(Q6) Types of problems in Concurrency :→

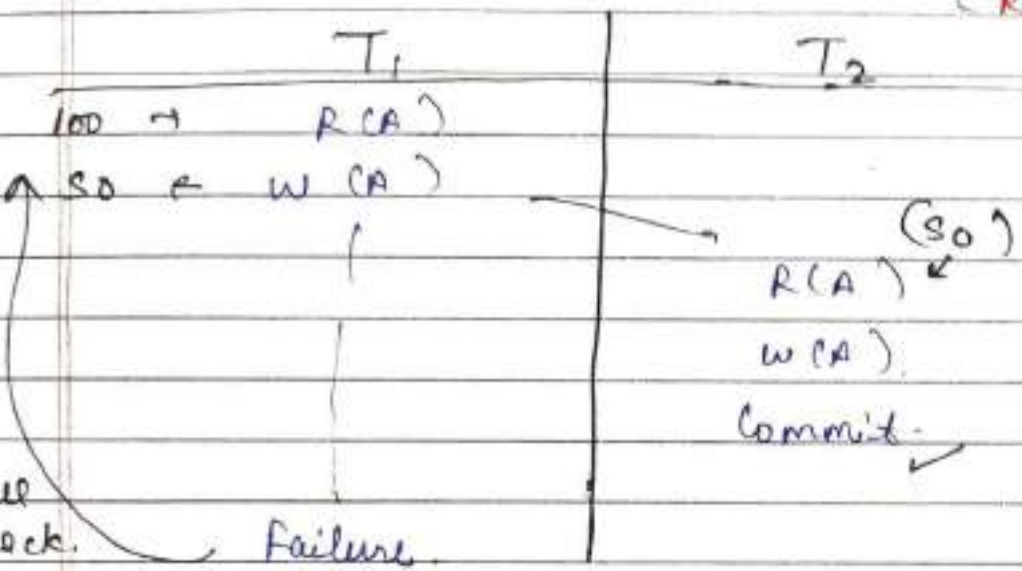
→ Concurrency, means when multiple trans. executed at a same time i.e, Parallel schedule.

→ (Mean, No problm is in Serial Schedule. There are prblms only in Parallel Schedule)

1) Dirty Read
2) Incorrect Summary

3.) Lost update
4.) Unrepeatable Read
5.) phantom Read.

1.) Dirty Read :→ or Uncommited Read or RAW.
(Read after write)



Hence, when T_1 gets failed. Then, how T_2 can use the A - 50. So, Dirty Read.

2.) Incorrect Summary problem :→ (T_1)
It occurs mostly when a transac^n start & T_2 comes & start performing its aggregate func^s.
(Sum, average)
then,
we get incorrect value of Sum, average etc.

3.) Lost Update :→

⇒ जो पहले (T_1) ने changes किये / उन changes को (T_2) ने over update कर दिया / & फिर बाद वाले changes दिखाई Lost हो गए / i.e, [update किया हुआ Lost हो गया]

Ex:→

Cid-1	T_1	T_2	Cid-1
5 Likes			5 Likes

⇒ & we finally get 5 Likes on the same product instead of 10 Likes.

Phantom - Simple, Shadow.

4.) * Unrepeatable Read :→

T_1	T_2
R(x) - 100	
	R(x) - 100.
W(x) - 50	
	R(x) - 50.

// Unrepeated Read.

⇒ T_2 got diff values on diff Read. (100 & 50).

∴, It is also a problm.

5.) * phantom Read :→

T_1	T_2
R(x) ← 10	
	R(x) → 10.
Delete (x) [0] ✗	→ Read (x). ('It also takes to Delete '0')

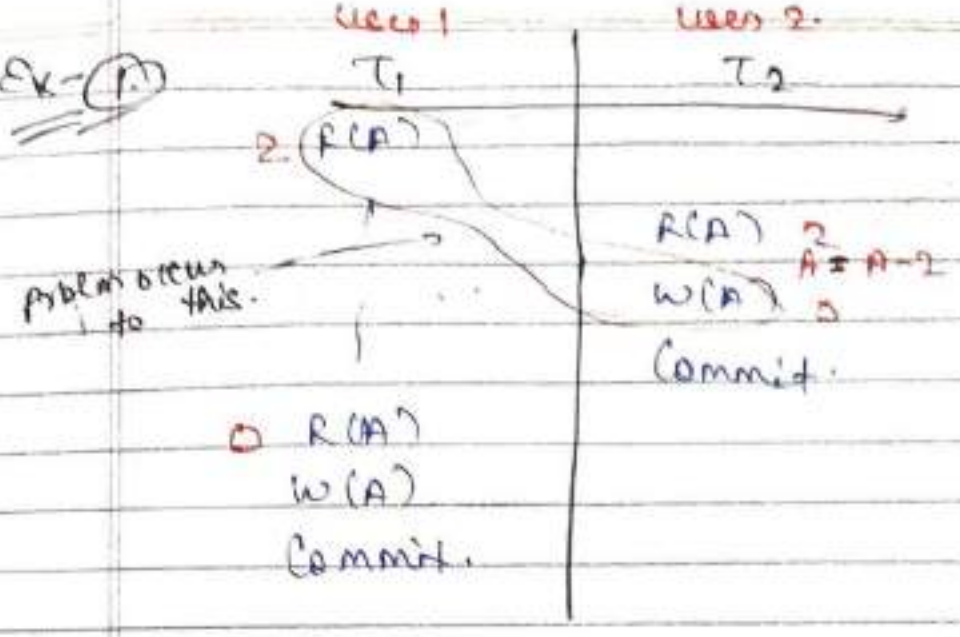
(7*) Read - Write (Problem) conflict (OR) Unrepeatable Read Problem :→

∴ 4 cases are there →

Same Data.

R (A)	R (A)	✓
R (A)	W (A)	Problem.
W (A)	R (A)	
W (A)	W (A)	

Ex-(1)



(#) Ex- IRCTC

Let, User 2 reserve both the seats. Then,

$A = A - 2 \quad = 0$

- & user 1 remained in waiting to reserve or not. & when he sees again.

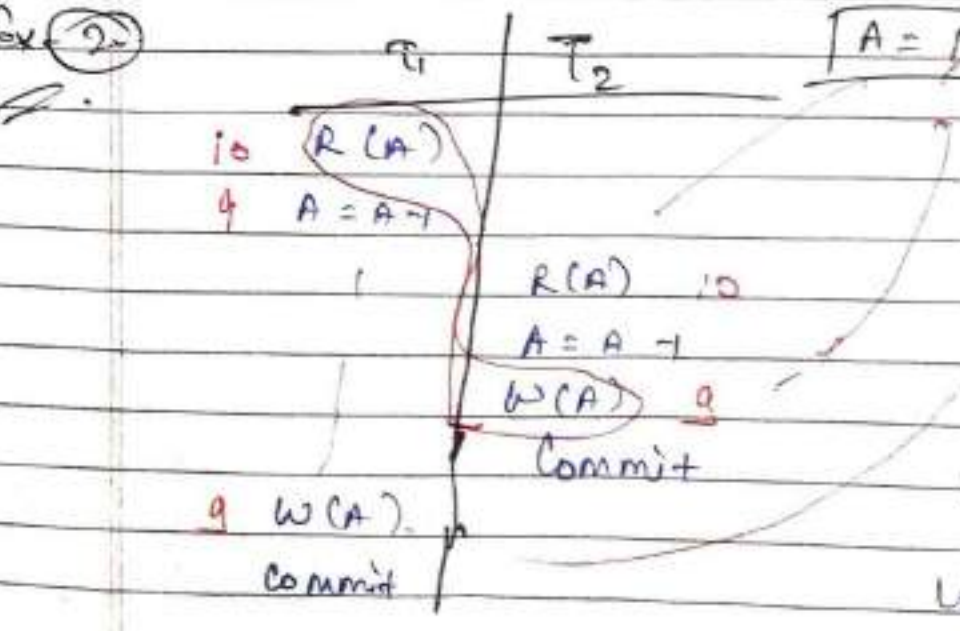
Now,

Seats becomes '0'.

So,

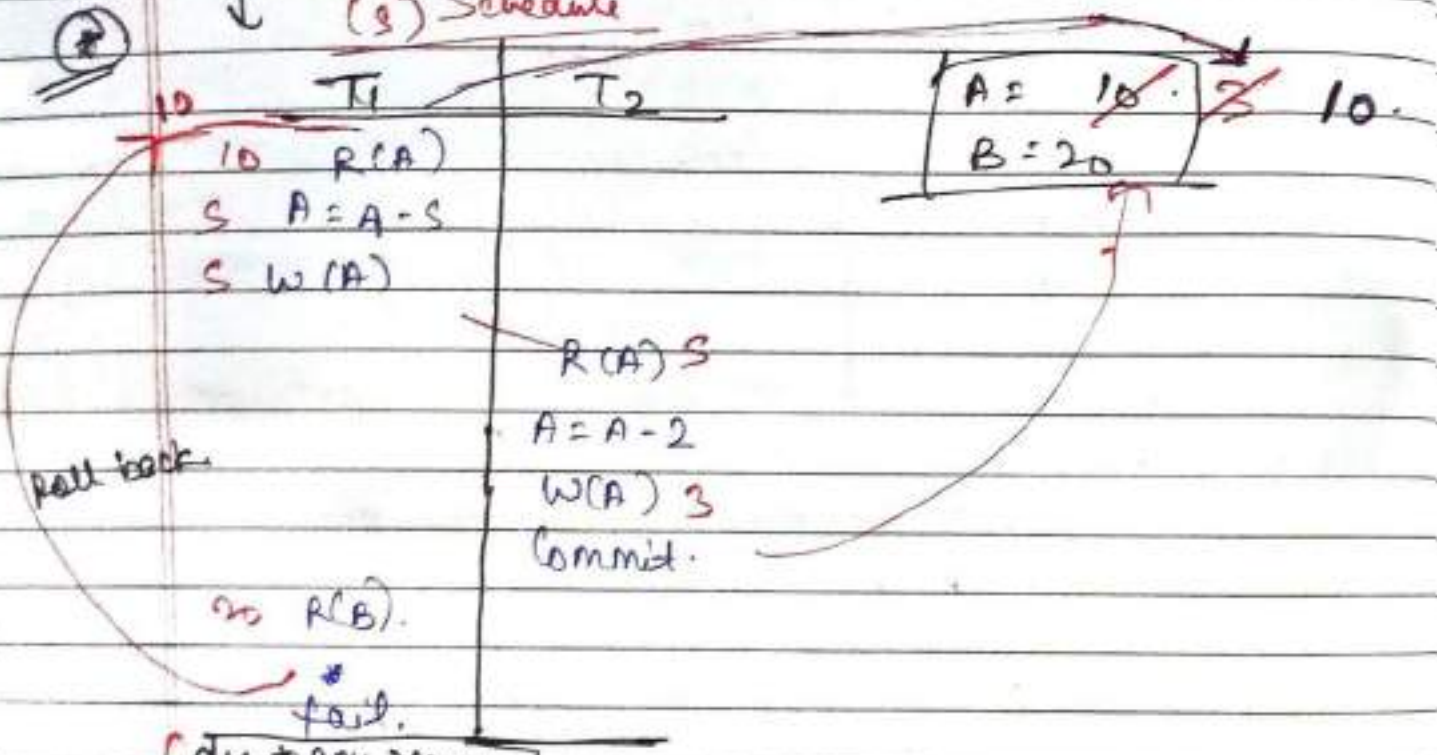
Now, User 1 have to be Roll back. (Abort)

Ex-(2)



(18.) Irrecoverable Vs Recoverable Schedules in Transaction's :-

↓ (3) Schedule



↳ Now, (T_1 Rolls back due to Atomicity property. ↓ (Either all or None)).

∴ On Roll back, everything happens in T_1 is gone. So, Again Now, [A is 10].

∴ But, here T_2 also done something & that is Lost Now.

⇒ T_2 Change is lost. We can't recover it now. ⇒ It is Irrecoverable Schedule.

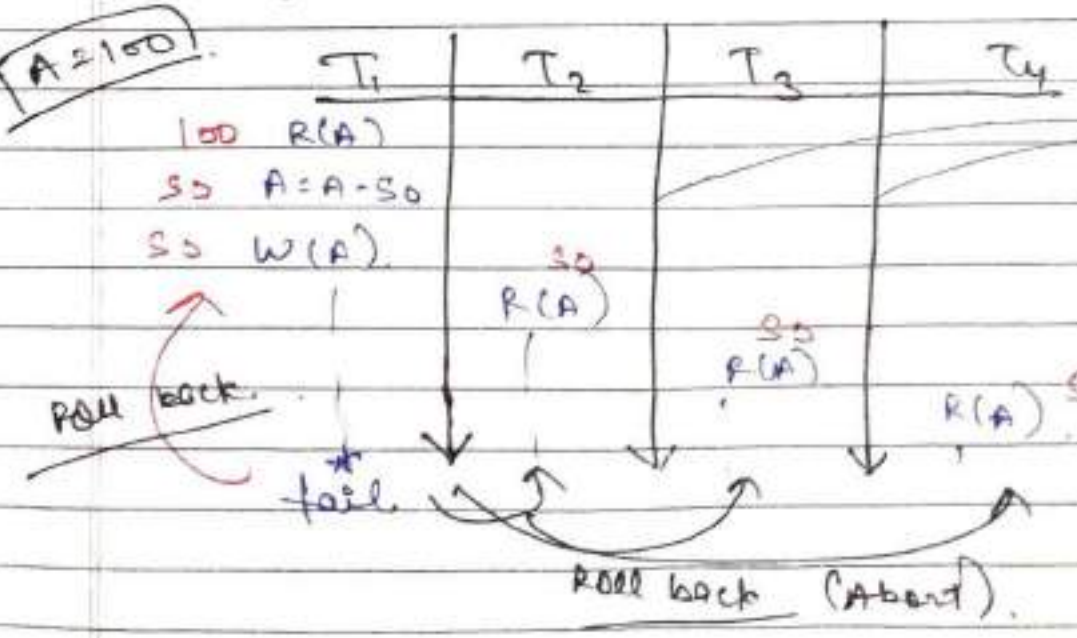
Cascade → झरना

79. Cascading (Vs) Cascadeless Schedule :-

↳ In Recoverability, these 5 are Important :-

1.) Recoverable
2.) Non-recoverable
3.) Cascadeless
4.) Cascading
5.) Strict Recoverable.

⇒ Cascading :- Means Due to occurence of one event, multiple events are automatically occuring.



↳ Here, if T_1 fails due to any reason, then it roll backs automatically due to Atomicity & again (A is 100). But, now T_2, T_3, T_4 has (A → 50). So, they are working on wrong data. So, now,
We also forcefully Roll back (Abort) the T_2, T_3 & T_4.

So, This is cascading.
(If T_1 fails, then we also have to roll back T_2, T_3 & T_4).

→ Here,
CPU utilisation gone waste by (T_2, T_3 & T_4).

performance is bad/poor.

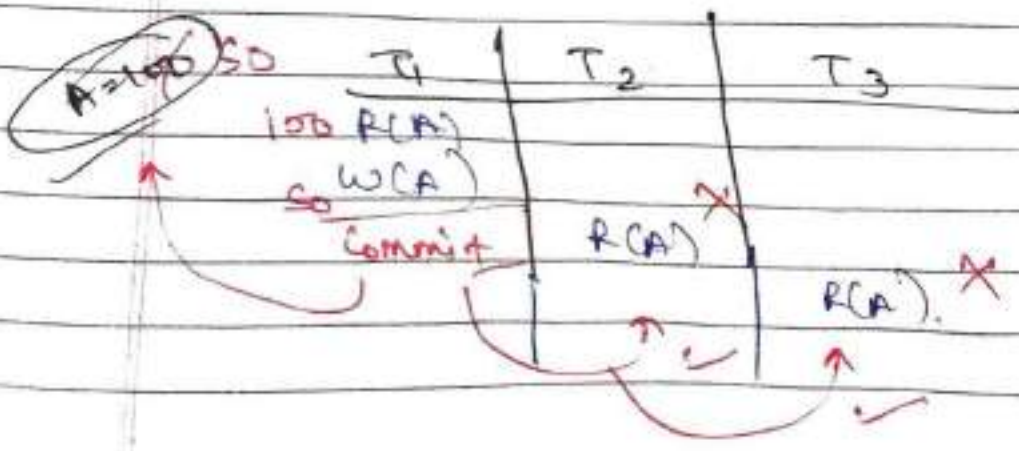
Ex:- If a Intelligent student (S_1) remains a answer. Then, other 3 students (S_2, S_3 & S_4) also cuts that answer. Bcz. it is wrong. So, their work has gone waste. Cascading.

* Cascadeless :→

→ How to remove the problem of cascading?

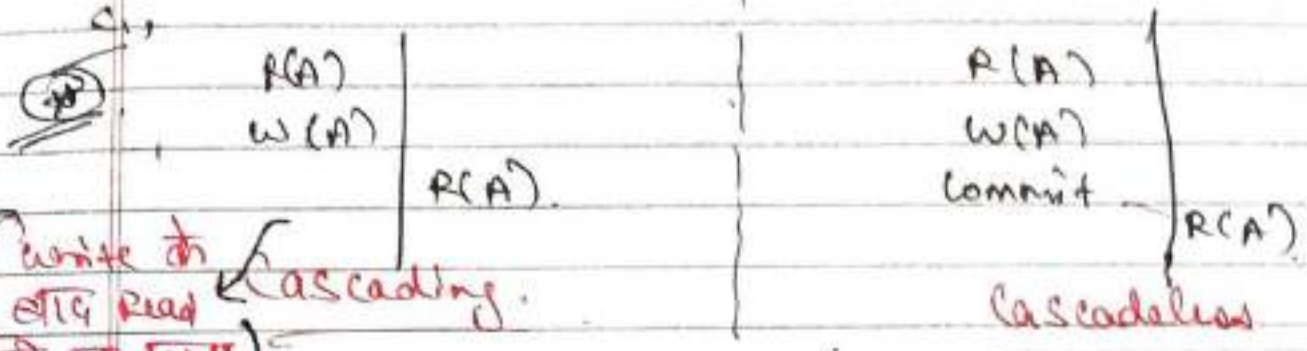
→ So ∴ T_2 & T_3 can't Read the (A) value from T_1 until A value gets committed or roll back in T_1.

(Commit → होने के बाद तो कोई दिक्कत ही नहीं है।)



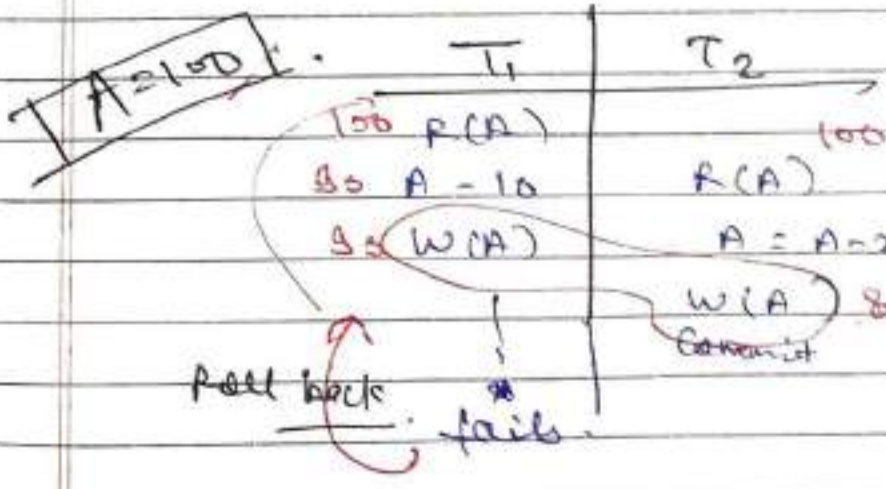
→ i, Don't allow Read in T_2 & T_3.

Qn,
It automatically becomes Cascadeless.



→ But, there is still W-W (Write-write) problem in Cascadeless. bcz,

(Read allow नहीं है, write तो कर सकता है)



→ Now, when T_1 fails. (A becomes 100 again)
Qn
The work of T_2 automatically gets lost.
i.e, Write-Write problem (or) lost updation problem.
bcz T_2 value of W(A) → 80 तो कहीं पे Reflect होती ही नहीं। & finally A = 100

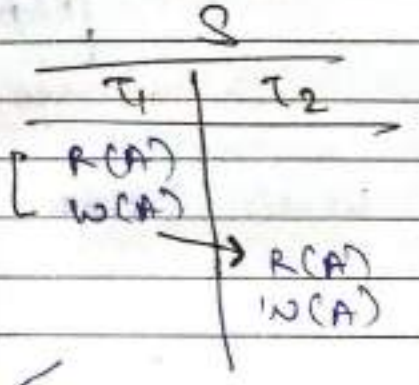
⇒ Strict Recoverable tells that we also can't write along with Read.

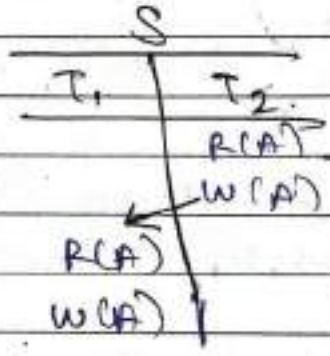
80. SERIALIZABILITY :→

Serializability means that a Schedule has ability to become a Serializable.

Mean,

Schedule - Collection of transaction. $(T_1, T_2, \ldots T_n)$





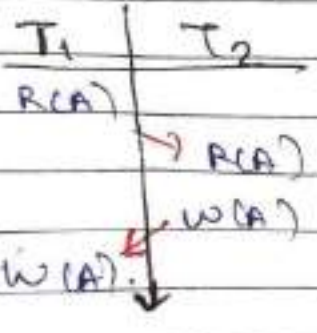
$T_1 \rightarrow T_2$

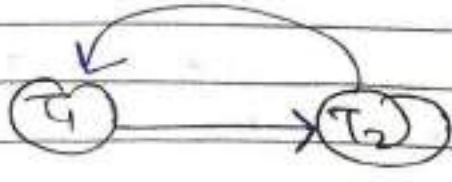
$T_1 \leftarrow T_2$

By seeing these, we can tell that they are already in serial schedule.

तो, Serial Schedule तो ये already है, इसको कुछ नहीं कर सकते।)

→ We want, Parallel Schedule →





→ Convert it to Serial Schedule →

2 ways :-

(1) $T_1 \longrightarrow T_2$

(OR)

(2) $T_2 \longrightarrow T_1$

⇒ To check Serializable? Check that if there Exists an Serial Schedule Equivalent to Parallel Schedule or not. This Concept is known as Serializability.

(#) Serializability (2 method).

- Conflict Serializable
- View Serializable

→ We check that a parallel Schedule can convert to a Serial Schedule or not. → Serializable (Class of 11 Schedule).

(#) Let, a Schedule has 3 transac^n's.

S

parallel Schedule →

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

⇒ Serial Schedule

16 ways !

$T_1 \to T_2 \to T_3$
$T_1 \to T_3 \to T_2$
$T_2 \to T_3 \to T_1$
$T_2 \to T_1 \to T_3$
$T_3 \to T_1 \to T_2$
$T_3 \to T_2 \to T_1$

⇒ If we able to convert that parallel Schedule in any of the 6 forms of serial schedule, then we can say that It is a Serializable.

⇒ Why we got 6 form :→
bcz,
we have 3 values (T₁, T₂ & T₃).
So,
3! = 6 ways.

(Q) Conflict Equivalent Schedules :→

R(A)	R(A)	} Non-Conflict pair.

R(A)	W(A)	
W(A)	R(A)	} Conflict pairs
W(A)	W(A)	

Here, A & B a 2 diff so, np.

R(B)	R(A)	
W(B)	R(A)	} Non - Conflict pair.
R(B)	W(A)	
W(A)	W(B)	

(#) How to Convert :→
- If we have adjacent non-conflict pair, then swap their pair.

Ex:

T_1	T_2
	R(A)
R(B)	

⇒

T_1	T_2
R(B)	
	R(A)

✓

Q:- To check Conflict Equivalent :-→

$S = S'$ → check

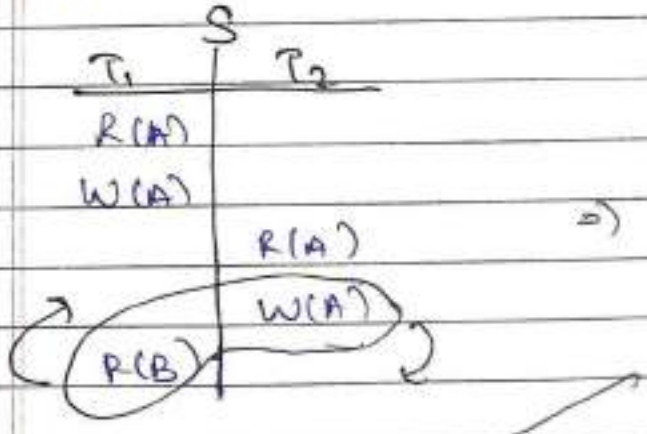
S

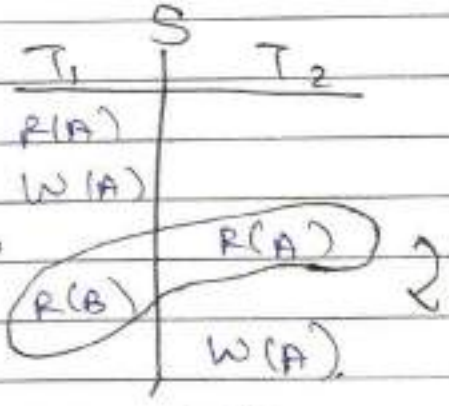
T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	

S'

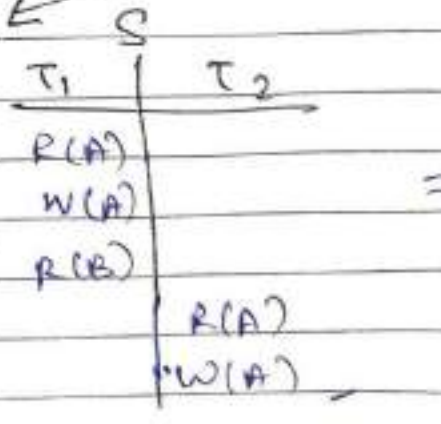
T_1	T_2
R(A)	
W(A)	
R(B)	
	R(A)
	W(A)

Sol":- So, In S, we have adjacent non-conflict pair, so swap them.





Again.



Now Serial Schedule.

$= S'$

Hence

$S \equiv S'$

Hence.
S & S' are Conflict Equivalent Schedule.

Ex:-

T_1	T_2
R(A)	
W(A)	
	R(A)
W(A)	
R(B).	

↓

2 adjacent pairs, But they are conflict pairs. ∴, no change in positions.

Note:-> $S \xrightarrow{CE} S' \longrightarrow$ Serializable ↓ i.e, (Serial Schedule)

↓ conflict equivalent

(1 - 81) videos End here.

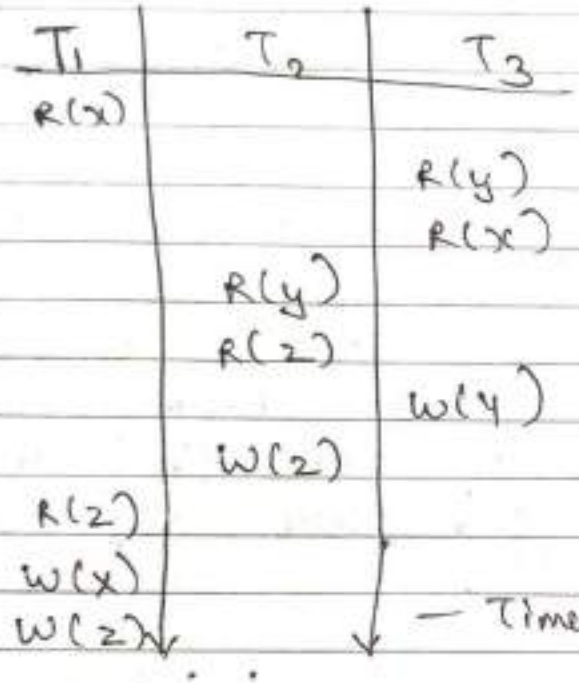
Jai Shree Ram

INDEX

Anurag Singhal DBMS STD: ___ SEC: ___ ROLL NO:

(82) Conflict Serializability, Precedence Graph :-

Q:-



→ First, we have to make precedence Graph → (mean, just graph with edges & vertices).

Vertex → No of Transac's (T_1, T_2 & T_3).

Now

* Edges:

Check the conflict pairs in other transactions and draw Edges.

Like * T_1 → (check in T_2 & T_3)

* Conflict pairs :→

R-W
W-R
W-W

of same variable. i.e. (x).

* Start :→

Now, Start with first opern. i.e,

In T_1 → R(x)

& check the conflict pair of R(x) in other Transac's i.e. (T_2 & T_3).

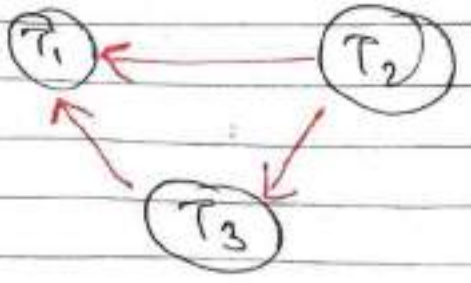
→ Conflict pair of R(x) → W(x) (C.P.)
&
now, Similarly,
conflict pair of W(x) → R(x), W(x).

→ Now, do check for every opero's in all transac's & after checking wit these opero^. & If you find any conflict pair, then draw edge acc. to vertex (Transac^).

→

T_1	T_2	T_3	C.P we find →
~~R(x)~~			
		~~R(y)~~	R(x) - W(x) ($T_3 \to T_1$)
		~~R(x)~~	R(y) - W(y) ($T_2 \to T_3$)
	~~R(y)~~		R(z) - W(z) ($T_2 \to T_1$).
	~~R(z)~~		W(z) - R(z) ($T_2 \to T_1$)
		~~W(y)~~	W(z) - W(z) ($T_2 \to T_1$)
	~~W(z)~~		
~~R(z)~~			
~~W(x)~~			
~~W(z)~~			

Precedence Graph:-



Now, come on Graph :-
check that if there is any loop / cycle in the Graph.

How to check cycle :- अगर हम एक किसी भी Node से चले, तो क्या हम घूम के वापस आ सकते हैं। If yes → loop / Cycle Exists.

(It is not must that the cycle must covers all the nodes. May be possible, it comes just after visiting one node). → also a cycle.

(# In our graph, there is no loop / cycle.)

No loop / Cycle → Conflict Serializable Schedule
If loop / Cycle Exists → Not C.S.S.

Conflict Serializable → Serializable → Consistent
(Serial Schedule)

Now, how to make Serializable ! →

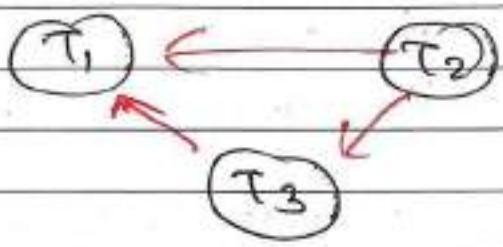
$T_1 \to T_2 \to T_3$
$T_1 \to T_3 \to T_2$
$T_2 \to T_1 \to T_3$
$T_2 \to T_3 \to T_1$
$T_3 \to T_1 \to T_2$
$T_3 \to T_2 \to T_1$

6 Cases → Now, which Case ?

Again, see the graph & check
Indegree = 0.

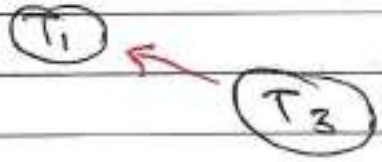
Indegree = 0, means a vertex with '0' indegree.

Mean, जहाँ (vertex) पर कोई भी Edge (arrow) नहीं आ रहा हो।



T_2 has, Indegree = 0,

Now, remove T_2.



∴ $T_2 \rightarrow T_3 \rightarrow T_1$ Ans.

T_1	T_2	T_3
	0	
		0
0		

(83.) View Serializability :→

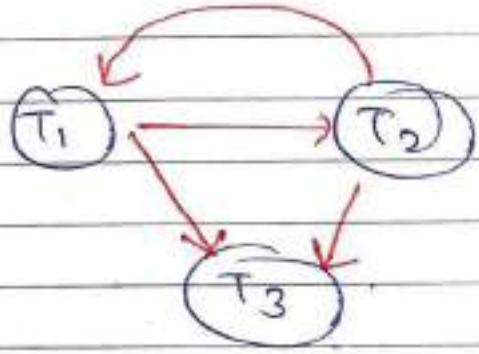
Q:- Check whether schedule is conflict Serializable or not?

→ First make Precedence graph.

Loop → No Answerable. then, view serializable Answers.

9

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



→ Here, we get a loop.

So,

It is Non Conflict Serializable.

↳ Here, we can't tell that it Serializable or not.

↳ Now,

(*) To check that, we use View Serializability.

→ Now, we arrange this Table,

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

We change the position

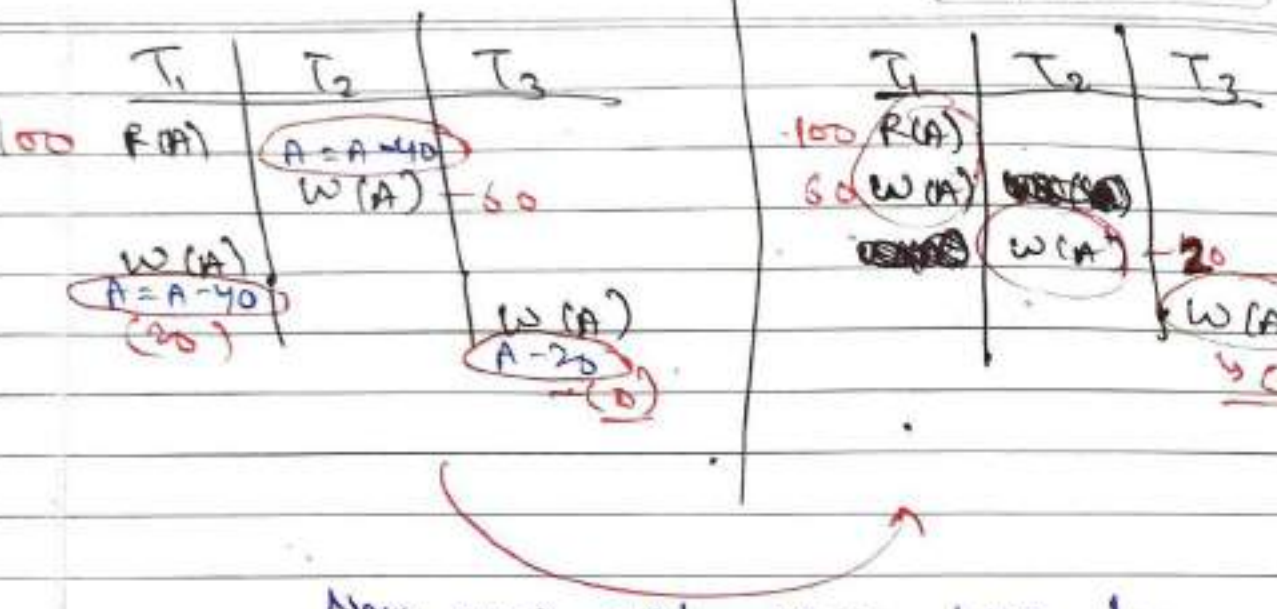
T_1	T_2	T_3
R(A)		
W(A)		
	W(A)	
		W(A).

Now this is a Serial Schedule. $T_1 \rightarrow T_2 \rightarrow T_3$.

But,

We have to check whether they match each other or not.

with A = 100



Now, apply this same here.

Here, finally A value = 0

Here also finally A is '0'

Hence, Both are Equivalent.

They are View Equivalent (≡).

Note:- bcz, finally output is given by A of T_3. So, If adjust the pos^n of A. of T_1 & T_2. So, there is no problem.

Now, $T_1 \rightarrow T_2$, $T_1 \rightarrow T_3$ (By : Refering of only T_1)

$\therefore$

Hence, $T_1 \rightarrow T_2 \rightarrow T_3$ (By Table).

Note:- Conflict Serializable tell that it is not Serializable (no ans.). But, View Serializable Checks it & tell that it is Serializable.

—x— —x—

(parallel schedule)

84. Concurrency Control Protocols :→ (C.C.P.)

(Shared - Exclusive Locking Protocol)

→ C.C.P. is that how to make them Serializable, or how to make them recoverable Schedule are concurrent, but how we can make them serializable & recoverable, this comes under Concurrency Control Protocol (C.C.P.).

→ We achieve this, By using Locking Protocols.

(#) 'Shared - Exclusive locking' :→

We use 2 locks here,

→ Shared lock(s) → if trans. locked data item in Shared mode, then allowed to Read only.

U → means unlock.

T_1	
S(A)	→ Shared lock.
R(A)	→ only Read
U(A)	→ Unlock.

→ Exclusive lock(x) → if trans. locked data item in Exclusive mode then allowed to Read & write both.

T_2	
X(A)	→ Exclusive lock.
R(A)	
W(A)	
U(A).	→ unlock.

* Compatibility Table :- →

→ Request

grant ↓	S	X
S	Yes	No
X	No	No

S S X
S ↗ ↗
X ↗ ↗

Explo" :-

S has only R(A)

X has both R(A), W(A)

So,

⇒ S - S (No Conflict) bcz R(A)-R(A).

⇒ S - X (Conflict).

R - R

R(A)	R(A)
	W(A)

⇒ X - S (Conflict)

R(A)	
W(A)	R(A)

⇒ X - X

R(A)	
W(A)	R(A) → Conflict.
	W(A)

* Problems in S/X Locking :- →

1.) May not sufficient to produce only Serializable Schedule.

2.) May not free from Irrecoverability.

3.) May not free from deadlock.

4.) May not free from starvation.

85. Drawbacks in Shared | Exclusive :→ S/X locking protocol

→ locking gives us Serializable Schedule
→ Consistent

i) May not sufficient to produce only Serializable Schedule.
→ हो भी सकता, नहीं भी।

Ex!→

T_1	T_2
R(A)	
W(A)	
	R(A)
R(B)	
W(B)	

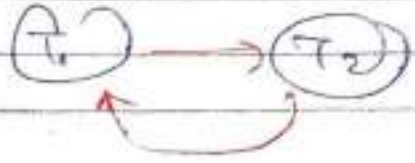
(concurrent (parallel) →)

try to convert to → $T_1 \to T_2$ (or) $T_2 \to T_1$

T_1	T_2
X(A)	
R(A)	
W(A)	
U(A)	S(A)
	R(A)
X(B)	U(A)
R(B)	
W(B)	
U(B)	

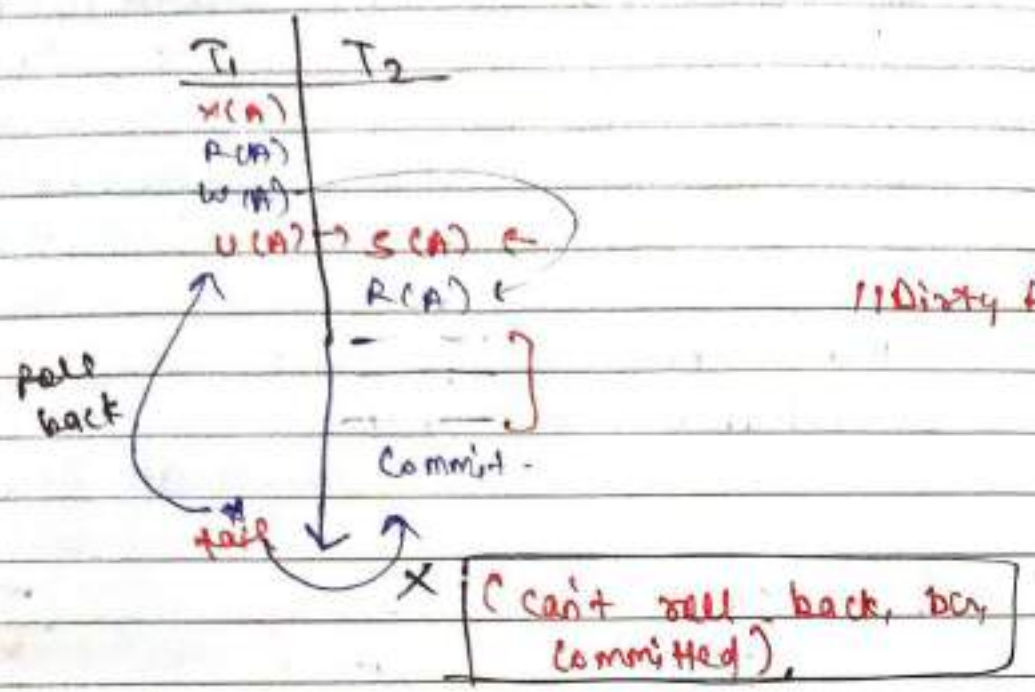
Note:→ Until, we do unlock, U(A) in T_1, we don't be able to put Stared lock S(A) on T_2. bcz by Compatibility Table. [(X-S) → No], so first we unlock X(A). & then, use S(A) on T_2 data.

ii) Here, we don't get Serializable Schedule even after appling S/x locking. As u can see in Table.



2.) May not free from irrecoverability.

Ex:-



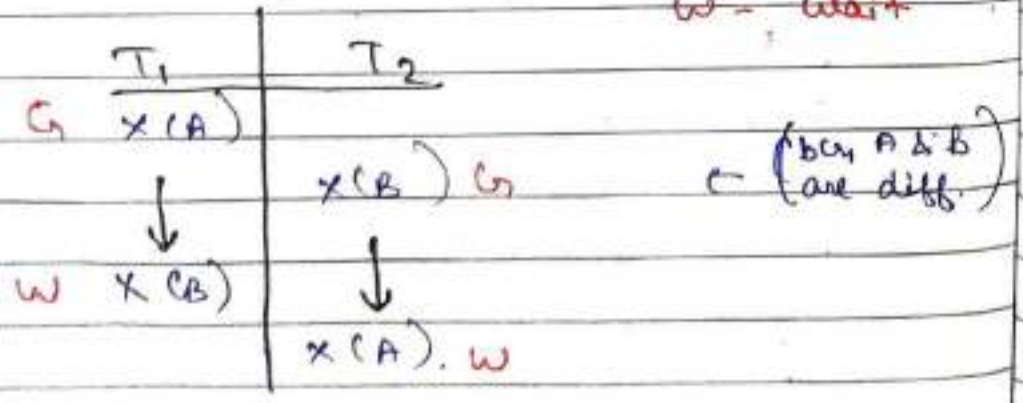
∴, It can't recover here.

3.) May not free from dead lock.

Deadlock :- When 2 person wait for resources, & they both are waiting in an infinite loop, So, when we wait infinitely, it is called Deadlock state.

G - Grant
W - Wait

Ex:-



2) Here, both are in waiting bcz, They are not unlock after use. So,

→ T_1 also waits that when T_2 unlocks, he can use on X(B).

Same

→ T_2 also waits that when T_1 unlocks, he can use on X(A).

So, Both are waiting in an Infinite loop.

4.) May not free from Starvation.

[In deadlock, waiting upto Infinite time
But,
In Starvation, waiting not upto the Infinite time.]

a) [Shared के ऊपर Shared दे सकते हैं, without unlock
. by compatibility Table.]

Ex:-

S → Shared Lock
X - Exclusive "
U - Unlock.

T_1	T_2	T_3	T_4
	S(A)		
W X(A)			
		S(A) G	
	U(A)		
			S(A). G
		U(A)	
			U(A).

Waiting Time (for T_1's X(A))

So, here, T_1 waits for X(A) until T_4 unlocks.
So, here starvation also. (by Shared के ऊपर Exclusive, बिना unlock के grant नहीं होता।)

86. **Two phase locking (2PL) protocol in Transaction Concurrency Control :→**

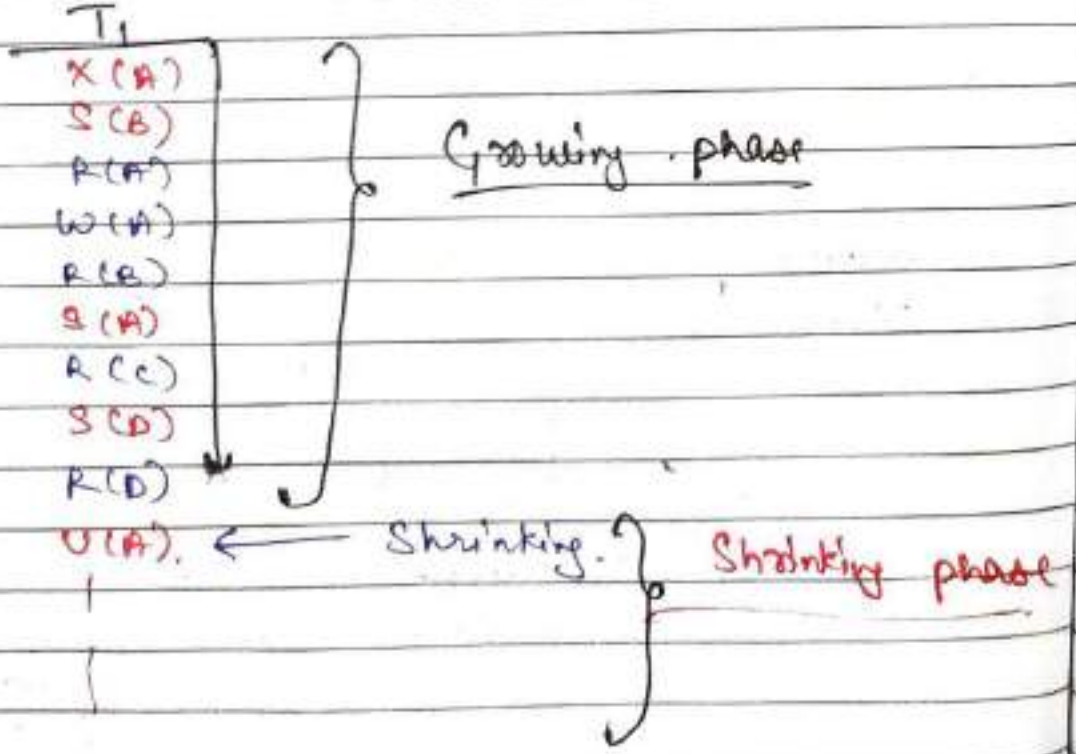
⇒ 2PL (2 phase locking) is just the extension of simple Shared/Exclusive locking. We just do modifications in them.

(#) **2-Phase locking (2PL) :→**

→ **Growing phase :→** Locks are acquired & no locks are released.

→ **Shrinking phase :→** locks are released & no locks are acquired.

[S/X]



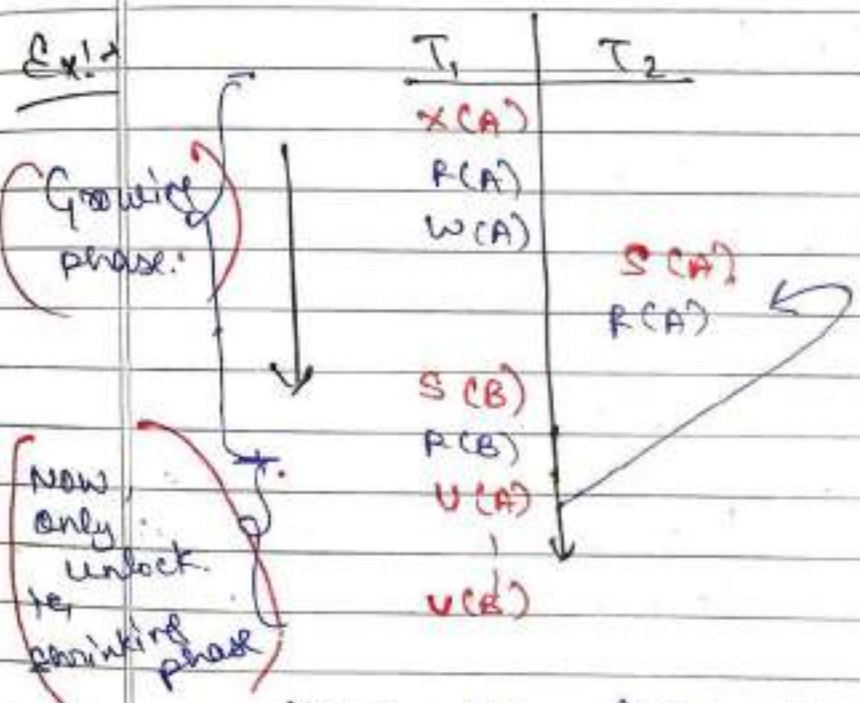
Note!:- When Growing phase starts, then only apply locks.

and,

When we first unlock, i.e, Shrinking phase starts, then we only unlock. (and not able to apply any lock).

⇒ We achieve Serializability by this, that we don't achieve in Simple Shared/Exclusive protocol.
So,
We made (2-PL) protocol for this.

Ex!-



Hence, we first starts T_1, & if T_2 comes in b/w then we don't entertain him. First, we complete T_1 & then goes to T_2.

$[T_1 \rightarrow T_2]$,

Serializability Achieved.

↓

Consistency automatically achieved.

Note:- The transaction which follows (2-PL), then it is always serializable.

Ex:- यहाँ पर दो Shared के बाद Shared तो दे ही सकते हैं। Means,

While T_1 is in growing phase by using S(A) [Shared lock], then at same time T_2 also starts its growing phase by using S(A). bcz by Compatibility Table.

Ex:-

S

T_1	T_2
Lock S(A)	
	Lock S(A)
Lock X(B)	
Unlock (A)	
	Lock X (B)
Unlock (B)	
	(*)
	Unlock (A)
	Unlock (B)

Growing → Lock point → Shrinking

Growing → Lock point → Shrinking

Now,

* Serializability Schedule, How formed?

i.e., $T_1 \rightarrow T_2$ (or)

$T_2 \rightarrow T_1$

So, This is done by (Lock Point).

Commit के बाद Roll back. नहीं कर सकते।

⇒ Lock Point :→ where trans. is taking the last lock. (or)
where trans is unlock first time.

⇒ जिसका Lock Point पहले आ जाएगा। वो trans. पहले आ जाएगी।

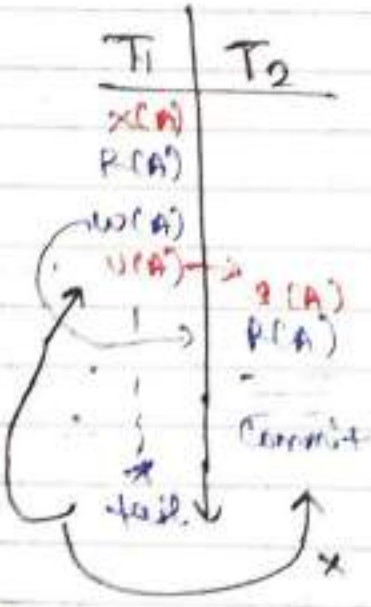
i.e., $[T_1 \rightarrow T_2]$

(87.) Drawbacks in (2-PL) protocol :→

→ Advantage: Always Ensures Serializability.

⇒ Drawbacks :→

(1.) May not free from Irrecoverability.

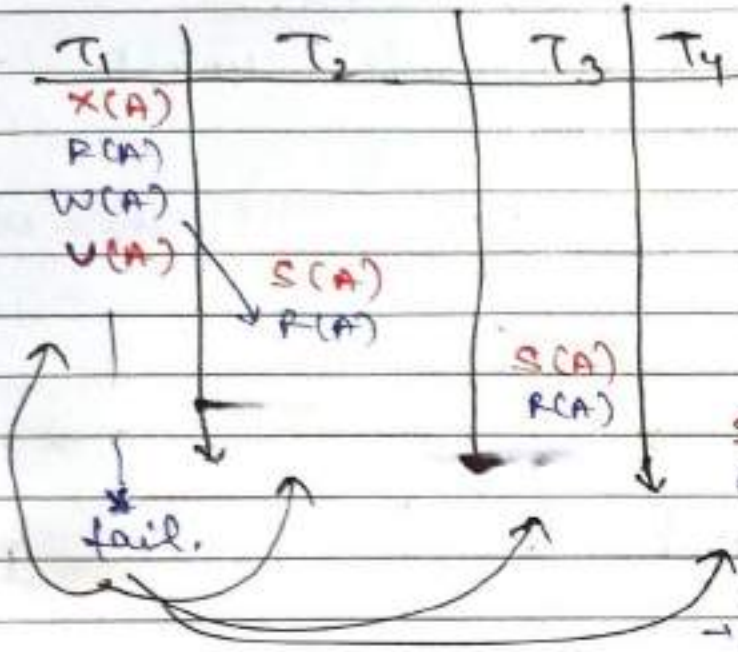


→ (We can't roll back it, bcz it is already commited)

Hence,
It is Ex. of Irrecoverable Schedule.
We can't recover it (Roll back it).
(rarely).

(2.) Not free from Cascading Rollback →

Ex:-



// Dirty Read.

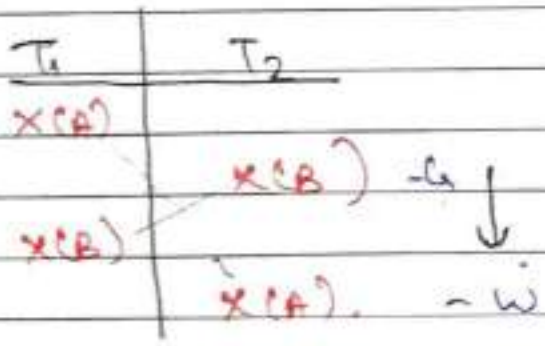
(Roll back them all)

→ (bcz, we don't do commit here ∴, Rollback possible)

Bad performance. & Cascading Rollback is possible here.

(3.) Not free from Deadlocks.

| Cs - Grant
| W - Wait.



(किसी ने भी unlock नहीं किया अभी ∴, दूसरे waiting में ही है)

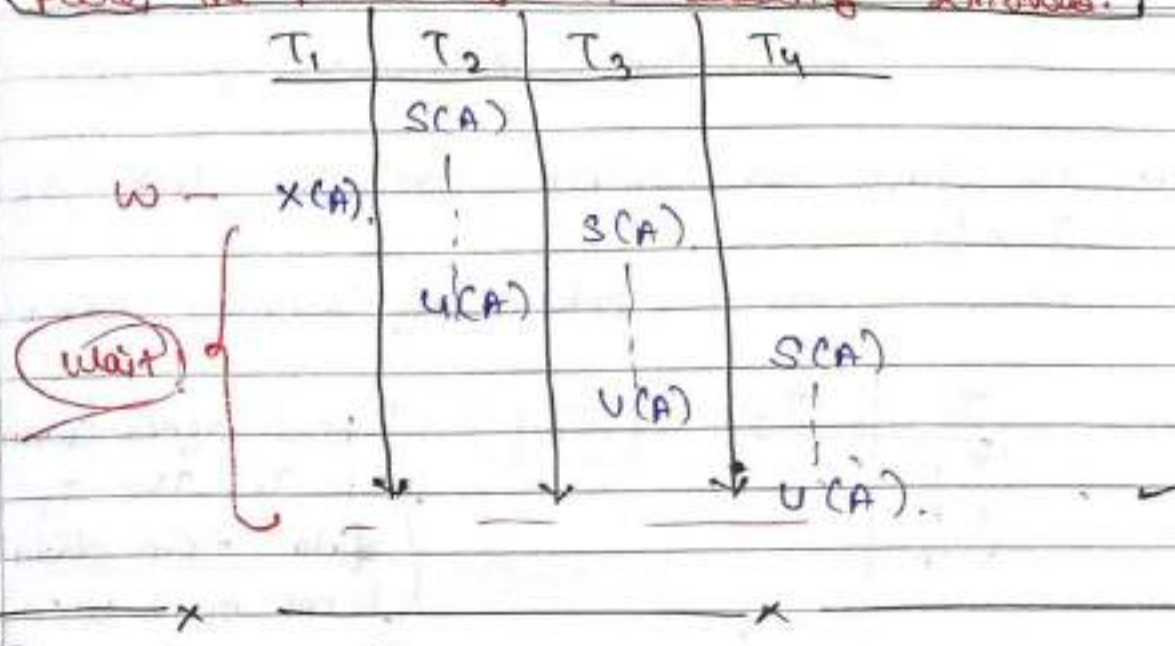
Infinite waiting here

(T1 & T2 both waiting here to complete their transcr's).

(4.) Not free from Starvation.

(wait for limited time).

In 2PL
Here, the problem of Serializability removed.



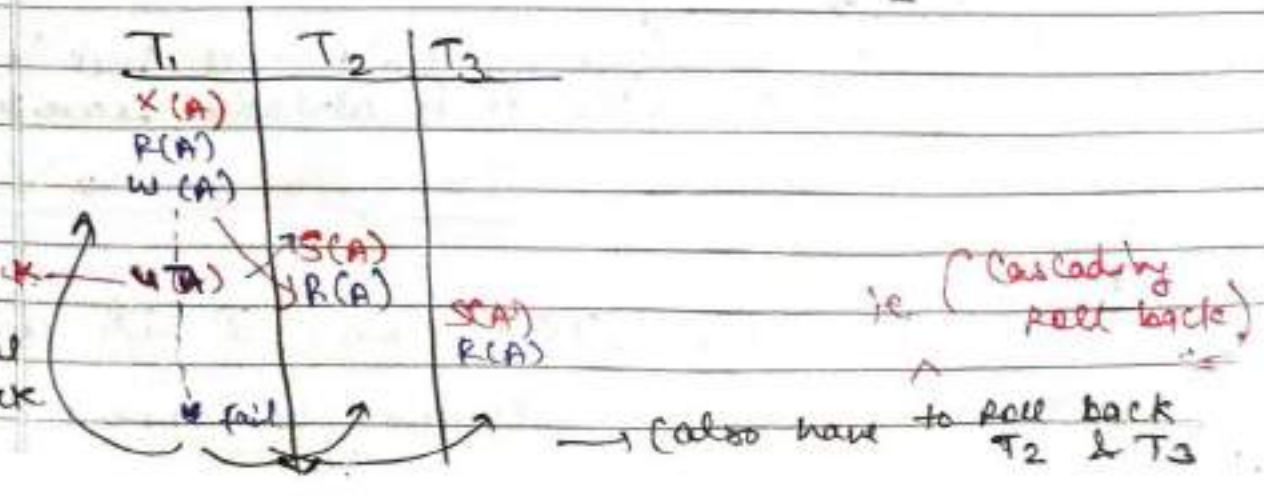
(18) Strict 2PL, Rigorous 2PL & Conservative 2PL Schedule :→

→ These are the Extension (Advance) of 2PL :→ basic.

Strict 2PL :→
It should satisfy the basic 2PL and all exclusive locks should hold until commit/Abort.

Rigorous 2PL :→ It should satisfy the basic 2PL and all shared, exclusive locks should hold until commit/Abort.

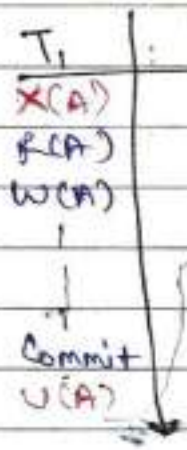
Ex:-



So, To stop this →.

→ (We have to unlock Exclusive lock after Commit.
then, this problm of Cascading removes.

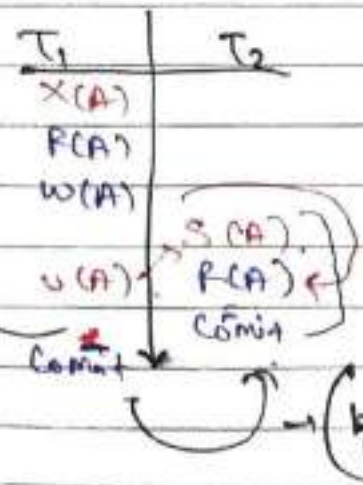
⇒



(DCR, after Commit in T_1. The T_2 takes data from database (H.D) & not from W(A) of T_1 So, no dirty Read.

Hence, Here, Cascading Rollback is Removed, & It will always produce Cascadeless.

⇒ Now, ∀ Irrecoverability :→



→ (Same data i.e, A)

→ (but we can't roll back it, DCR it is already Committed.

∴, Irrecoverability.

→ Hence, हम इसको पहले unlock ही नहीं करेंगे।
We unlock it after commit is done.

⇒ (Hence, T_2 को S/X कुछ भी नहीं मिलेगा, जब तक T_1 commit नहीं हो जाता। ∆

और अगर commit होने के बाद S/X मिल रहा है (T_2 को), then No problem.

⇒ (Hence, Irrecoverability remains here.) commit → U(A) → unlock

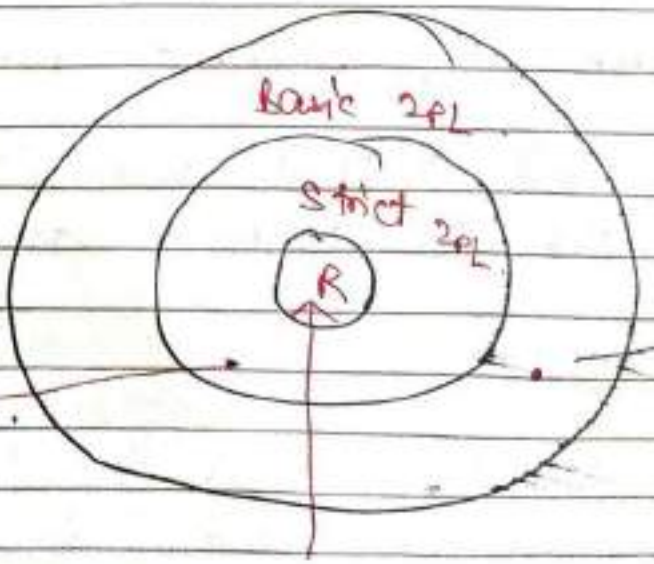
फिर वो (T_2) Database (H.D.) से ही Read करेगा।

⇒ It produce Strict Recoverable Schedule.

Note:- 2 problems remains here →
1.) Cascadeless
2.) Strict Recoverable

- Rigorous 2PL (R), बस और strict हो गया। RC, इसमें S/X दोनों आ गए।
(इसमें हम Shared lock को भी Release नहीं कर सकते।)

(H)



In both 2PL & Strict 2PL. But not in Rigorous 2PL.

→ But, problem of deadlock & starvation still are there.

T_1	T_2
X(A)	
	X(B)
W - X(B)	
	X(A) - W

→ Still, Deadlock.
→ Infinite waiting.

T_1	T_2	T_3
	S(A)	
W - X(A)		
waiting		S(A)
	C	
		C

→ finite waiting
→ Still, starvation.

Conservative 2PL :→

practically, It is not possible. In this, कि कोई भी trans. start होने से पहले ही सारे Lock लेले। (A, B, C → सब पे लेले।).

→ इससे T_2, T_3 को एक भी lock नहीं मिलेगा।

∴ problem of Deadlock Removes here.

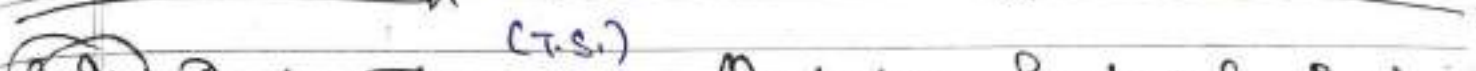
→ (T_1 को पहले ही सभी resource दे दो, ताकि T_2 को एक भी resource ना मिले।).

T_1	T_2
G - X(A)	
	X(B) - W
G - X(B)	
	X(A) - W

G - Grant
W - waiting.

i, T_1 completes were first. So, no infinite waiting. bcz
(T_1 के comp. होने पर T_2 use करे लेगा।)

[No Deadlock here.]



89. Basic Timestamp Ordering Protocol :→

→ unique value assign to every transaction.
→ Tells the order (when they enters into System)

Let,

10:00	10:10	10:15 → Time	T.S. (T_i)
T_1	T_2	T_3	
100	200	300 → Timestamp	
↓	↓	↳ youngest.	
elder (पहले आई)	younger (बाद में आई)		

→ Read_Ts (RTS) = last (latest) transaction no. which performed Read successfully
→ Write_Ts (WTS) = last (latest) transaction no. which performed write successfully

Ex:1

10	20	30	T.S. (T_i)
T_1	T_2	T_3	
R(A)			
	R(A)		
		R(A)	

=> RTS(A) = 30

↳ bcz, latest Read is of T_3 & Time stamp (T.S) of T_3 is 30.

Ex:-

10	20	30
T_1	T_2	T_3
W(A)		
		W(A)
	W(A)	

⇒ WT(A) = 20

(ii) Rules :-

In Timestamp, let,

(जो Trans. पहले आई, वही पहले complete होगी)

Serializability : older → younger. → follows conflict Serializable

Tx Read

1. Trans. T_i issues a Read (A) operation

a) If WTS (A) > TS (T_i), Rollback T_i

b) otherwise Execute R(A) operation

Set RTS (A) = Max of { RTS (A), TS (T_i) }

(0, 200 ; 100, 300) → in next rule use

Tx Write

2. Trans. T_i issues Write (A) operation

a) If RTS (A) > TS (T_i) then Rollback T_i

b) If WTS (A) > TS (T_i) then Rollback T_i

c) otherwise execute write (A) operation.

Set WTS (A) = TS (T_i).

(iii) Understanding these :-

Ex → (old) 100 T_1 | 200 T_2 (young)

T_1	T_2
R(A)	
	W(A).

✓ (NP). (No Conflict).

T_1	T_2
W(A)	
	R(A)

✓

T_1	T_2
W(A)	
	W(A).

✓

bcz,
(old → young) or (T_1 → T_2). So,
(T_1 पहले Execute हो जाएगी, तो T_2 में कोई दिक्कत नही आएगी।).

Case I

Ex:- (old) 100 T_1 | 200 T_2 (young)

T_1	T_2
	R(A) -10
(20) W(A).	
commit	

→ (bcz, now से T_2 ने A-10, कौन सी value Read करके बैठी है। bcz, T_1 ने तो Database में 20 update कर दिया।).

Roll back. (X) (Not possible)

(T_1 को होने नही देना।).

Case II

T_1	T_2
	W(A) - (20).
20 → R(A)	
commit	

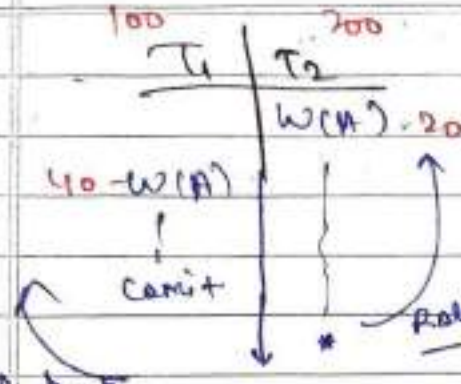
(Hence, से 20 अब कहीं पे है ही नहीं। i.e, T_1 गलत Data पे काम कर गयी।).

Roll back.

// Dirty Read

(हम T_1 को Read नही करने देंगे।)

Case III.



∴, Roll back.

(∴, इस T1 को allow नहीं करेंगे).

⇒ (If T2 fails, then, our updation will be lost).

⇒ It lost updation.

(90.) Solve Ques on Timestamp Ordering Protocol.

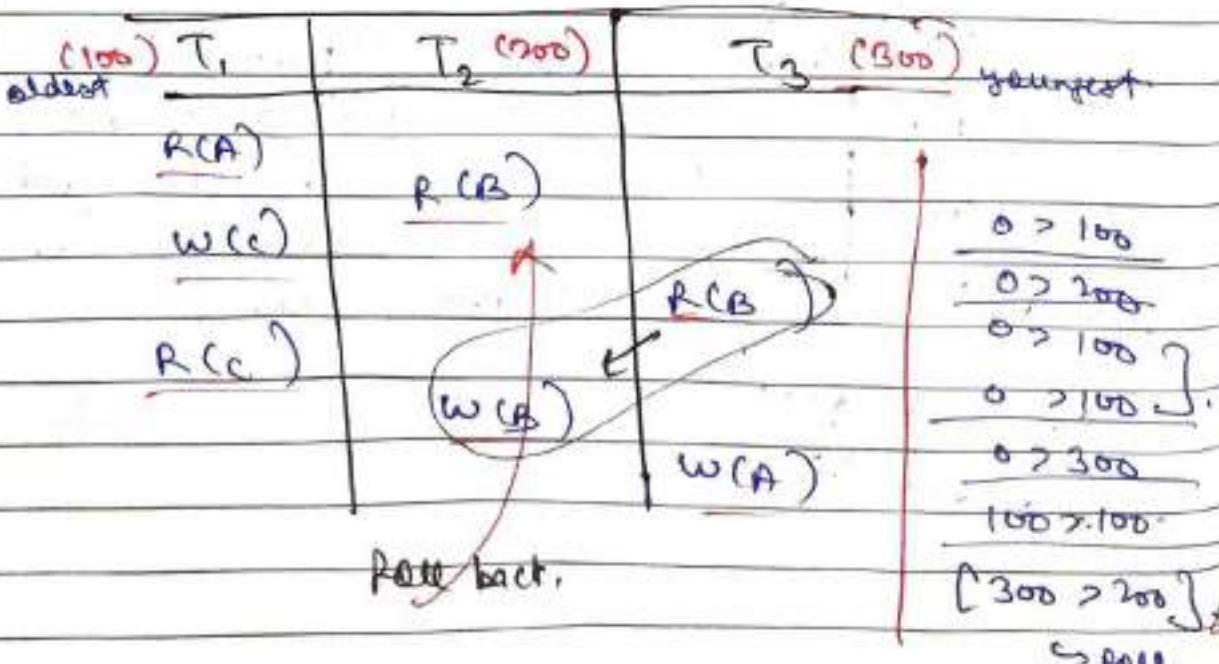
We have 2 Rules →

1.) 1st Rule is of Read → has only 1 cond" bcz, only (R-W) conflict.

2.) 2nd Rule is of Write → has 2 cond"s, bcz (W-R, W-W) → 2 conflicts.

Now,

Q -



100 > 300.
0 > 300.

	A	B	C
RTS	100 ~~200~~ 300. 0	0	0 100
WTS	0 300	0	0 100

=) Initially, all values are 'zero'.

(Final Table) ✍

(Thomas)

→ Check, for every values in the Table & put that values in Rules & then make the Table.

- for R(A) use 1st Rule of Read
- for W(B) use 2nd Rule of Write.

→ first check R(A) of T_1, then
R(B) of T_2, then
W(C) of T_1, then.
R(B) of T_3, then.
R(C) of T_1, then
Here Roll back. ← W(B) of T_2, then
W(A) of T_3

i.e, pattern wise as in Table

&

then, make the final Table of :

(RTS) & (WTS) ✍

OR

We also do the Ques^n direct, without using these 2 Rules with just the [older → younger] concept & cases (we discussed in prev. video).

[Volatile – अस्थायी, परिवर्तनशील]

31. INDEXING :→

⇒ CPU → processor (who process).

→ Query comes to CPU. CPU process this.

⇒ Eg →

```
Select * from student where Roll no = 1;
```

⇒ CPU has to execute it, but Data is in the Memory.

⇒ In general architecture, we only let, 2 types of Memory →

1. RAM - primary Memory
2. H.D. - Secondary Memory

⇒ Ram is volatile, Ram works directly with CPU.

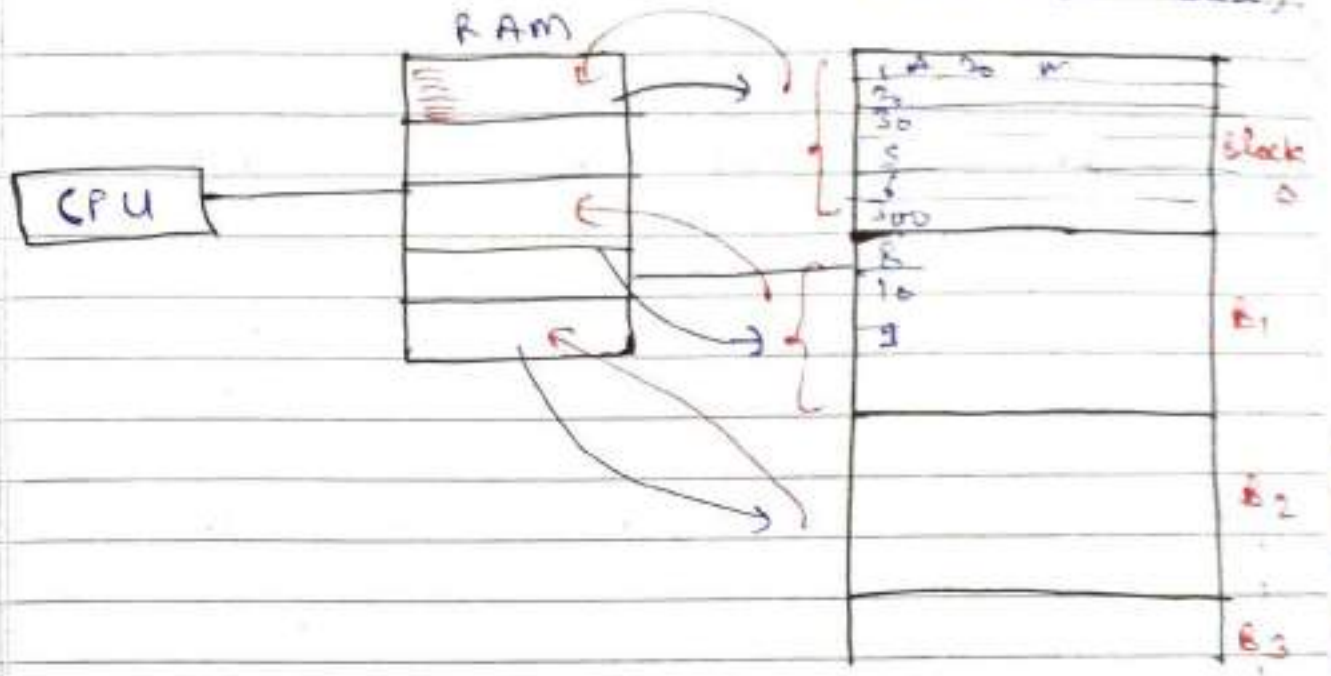
- CPU Speed → In MIPS (Million Instr per Second)
- ⇒ H.D. Speed ⇒ 10/20 Instruc per second.

-i, these both are not compatible with each other.

So,

⇒ Ram comes b/w them.

⇒ Hard speed counts in Milliseconds.
CPU ——v—— nano/picoseconds.



We divide Hard Disk in Blocks.
Logical Blocks (& not physical Blocks)

Ex:
Logical Drives → C Drive, D Drive, — etc.

O.S. divides hard drive into fixed Sized blocks & then Insert Data. (Blocks / pages)

Ex:) If we have record of 10 K students
&
Block Size is of 100 Records in Hard Disk.
So,
we need 100 Blocks in H.D (100 x 100 = 10K)

⇒ Now, data may also be Stored in 2 ways
1.) Sorted (ordered) 2.) Unsorted (unordered)

⇒ Let,

Date
Page No.

2) Unordered Data! → then, we send 1-by-1 every block of H.D. into RAM. & if we get our data, then o.k. Otherwise, RAM send back that block & next blocks come into RAM. & so on.

If we get data → HIT
If we don't get → miss

(#) So, Here, INDEXING is used.

(ईस Ram में H.D. के Block का Searching time को नहीं करते,).

3) We transfer data from H.D. to Ram. So, there is a transfer cost. i.e,

[I/O Cost]
→ Input/ out-put.

इस Data को call कर रहे, इसको बोलते I/o cost.

3) & If call more blocks, then I/o cost more. & then time also increases of searching.

So,

(*) Indexing, that we have to call min^m no. of blocks. i.e, I/o cost is reduced.

Ex:- Let, a book has 1000 pages. & we have to search a page.
Worst Case → 1000 (find in last).
Best Case → 1 (find on 1st page).

Average → 500 pages.

∴ (If we use Index, then the no. of pages we shuffle are decreases.)

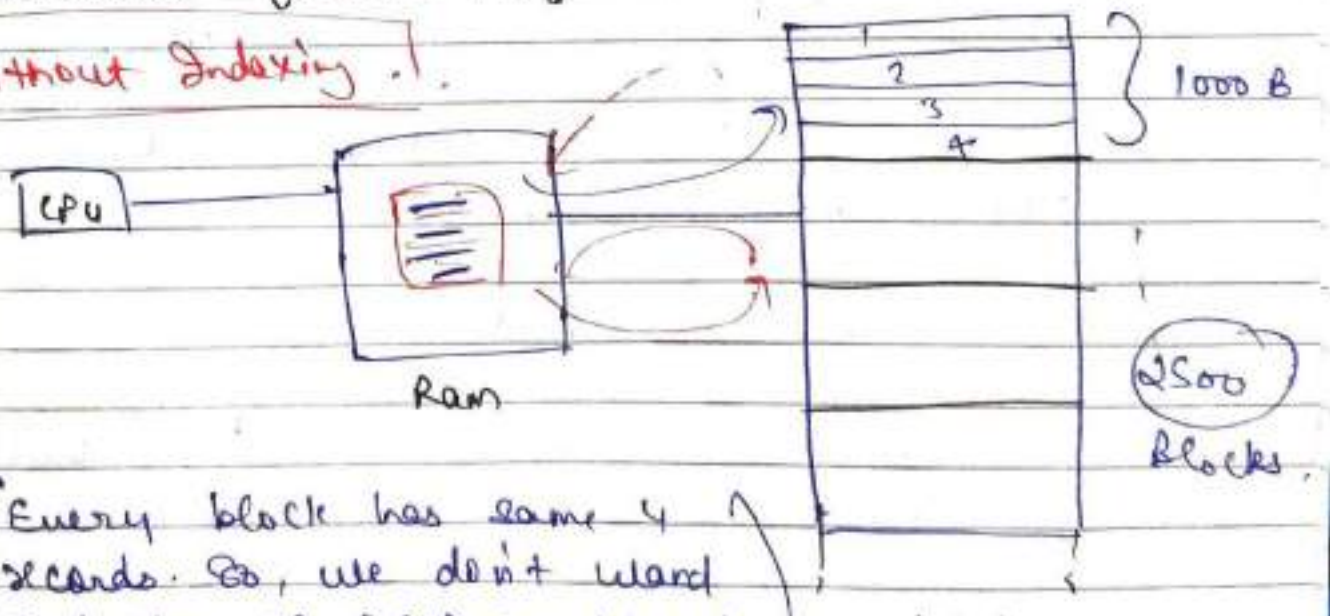
⇒ (Let, Index is of 2-3 page, then just see that topic an Index & then directly go to that page).

32- Numerical Ex. on I/O Cost in Indexing :→

Q:→ Consider a Hard Disk in which Block Size = 1000 Bytes, each record is of size = 250 Bytes. If total no. of records are 10,000 and the data Entered in Hard Disk without any order (unordered).

→ What is avg time Complexity to search a record from HD?

Without Indexing :



∴ Every block has same 4 records. So, we don't want that time of RAM to search in a block bcz ie, always 4.

⇒ No. of records we can put in every block = $\frac{1000\ B}{250\ B} = 4$

&

⇒ No. of Blocks Required = $\frac{10000}{4} = 2500$

⇒ **I/o Cost :-**

(#) Best Case = 1
Worst Case = 2500 = (N).

Avg Case = $\frac{2500}{2} \Rightarrow 1250$ $\left(\frac{N}{2}\right)$.

Avg Time Complexity : O(N)

& This is all for unordered Data.
Hence, we used linear Search here.

(#) Let, **Ordered Data :-**
then,
we use Binary Search here.
but it Searches on sorted data
(either ascending or descending).
& then,
Time Complexity : $O(\log_2 N)$

Here, $\Rightarrow \log_2 2500$.

$\Rightarrow$ 12 approx

Hence,
we need approx 12 blocks in ordered data.

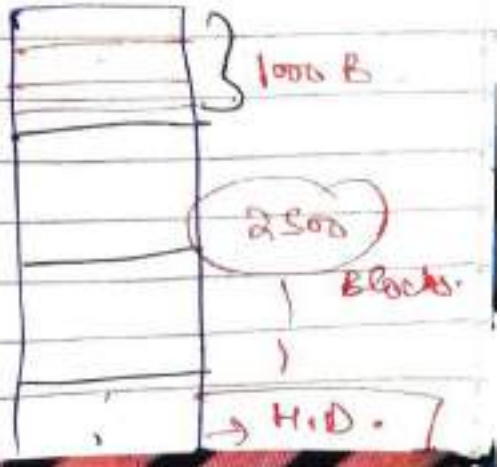
$\therefore$, this both case are without Indexing

$\therefore$ When we used Indexing, we even need less blocks than these.

I/O cost is low.

33. Numerical on I/O Cost in Indexing :-

Q:- Consider a H.D. in which Block Size = 1000 Bytes, each record is of size = 250 Bytes. If total no. of records are 10,000 and the data Entered in Hard disk without any order (Unordered Data). What is the avg time Complexity to search a record from Index Table if Index Table entry = 20B (key 10B + pointer 10B).



2500 Blocks

$\therefore$ Index is also stored in the H.D. When we search any data, then Index come into the Ram.

→ Then, we first search into the Index.

Note:- Block size in Index (page) = Block size in H.D. (page)

1000 B — 1000B

↓

→ **understanding:-** In a book, all pages are of same size, whether it is Index page or anything.

① Index Table Entry = 20 B (key + pointer)
key = 10B, pointer = 10B.

key value → जिस value से हम search करते हैं (Topic name), Roll no. etc.

pointer → page. No.

→ Each block in Index Table contains Record = $\frac{\text{Block Size}}{\text{Index Entry size}}$

$= \frac{1000}{20} = 50$

= 50

Hence, we can put 50 records in a block of Index.

(bcz, contains only topic name ∴ ज्यादा आ जाएंगे).

key | pointer
50 Records } 1000B
50 Records

⇒ How we can Enter records in an Index! →

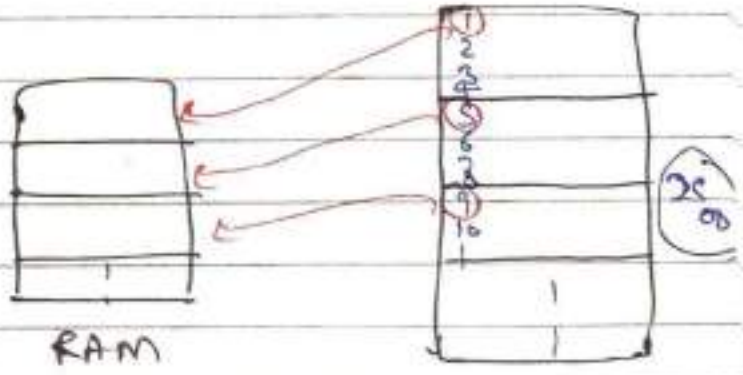
1) **Dense!** → used when unordered data.

Here, we have to put all the records of H.D. into the Index.

Ex!, → Here, all 10k records are to be Entered in Index.

2) **Sparse!** → used when data is ordered / sorted.

Here, we just Enter 1 record from the block into the RAM. Like, we entered Anchor data (leader) into the Index. (1, 5, 9 ...).



Ex!, → So, here → we just have to enter the no. of records in RAM equal to no. of blocks in H.D.

Here, 2500 records are to be Entered in Index.

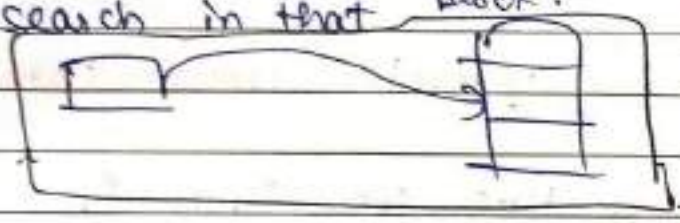
(*) Now, come to ques!,
If our data is ordered → then,
i.e, [Sparse] →
no. of ~~records~~ Blocks in Index = $\frac{2500}{50}$ [→ 50]

Then, search time → $\log_2$ so

→ 6 approx

Hence,
We have to search 6 times to find the desired page no.

(then that page no. takes us to the desired block in H.D & then, we just only have to search in that block.



So

Total no of Search = 6 + 1

= 7

6 search in Index.
last (1) search in H.D.

But,
→ our data is unordered →
Here, Dense →

→ No of blocks in Index = $\frac{\text{All records}}{\text{record in one block of Index}}$

$= \frac{10,000}{50}$

$= 200$

Now,

Searching → $\log_2 200 + 1$

$+ 8 + 1$

19 approx

$\log_2 2^8 + 1$

$8 + 1 = 9$

$2^8 = 256$

→ Here, we use $\lceil \log_2 200 \rceil$ Also, for Index में तो unordered data ही ordered/Sorted होकर ही आएगा ना।

↳ we use $\lceil \log_2(n) \rceil$ here. ✓

94. ✱ Types of Indexes :→

1.) Primary Index
2.) Clustered "
3.) Secondary "

| key → uniqueness.
| non key → not unique.

Ref.

(A) Table :-

	key	Non key
ordered file	primary Index	clustered Index
unordered file	Secondary Index	Secondary Index

✱✱ Most ↑.

key	Non key
1	1
2	1
3	2
4	2
5	3
⋮	3
	⋮

(B) We all have primary Index in our Books ✓
In last of Book → Secondary Index.

If we want I/O cost to be less, then we made Indexes.

Date
Page No.

95. Primary Index :→

1) In a table, if we made any of att. primary key, then By default → primary Index is operated on them Automatically.

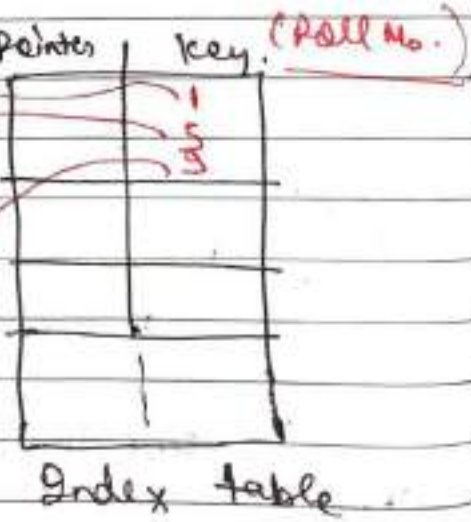
2) ~~Advantage~~ :-
1.) data is Ordered
2.) & also unique. (key value is present)

Ex:- In IRCTC, we first find everything, by Train No.

Ex:- Pollno (unique value)

file (HD)

1	Ram	24	Delhi
2	Ravi	7	Chennai
3	Mukesh	22	Mumbai
4	Anurag	23	Delhi
5			
6			
7			
8			
9			
10			
11			
12			



Index table

↳ Data is (Sorted + unique).

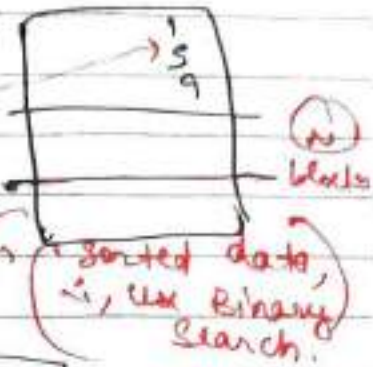
So, we use Sparse here.

→ No. of Entries in Index Table = No. of blocks in H.D.

and, we call it Anchor. (Leader).
(1, 5, 9, 13 . . . .).

Primary Index is Sparse.

Ex.) we have to find 3,
then,
we know it is found in block of '1'.



Hence,

Total Search Time = $(\log_2 N + 1)$.

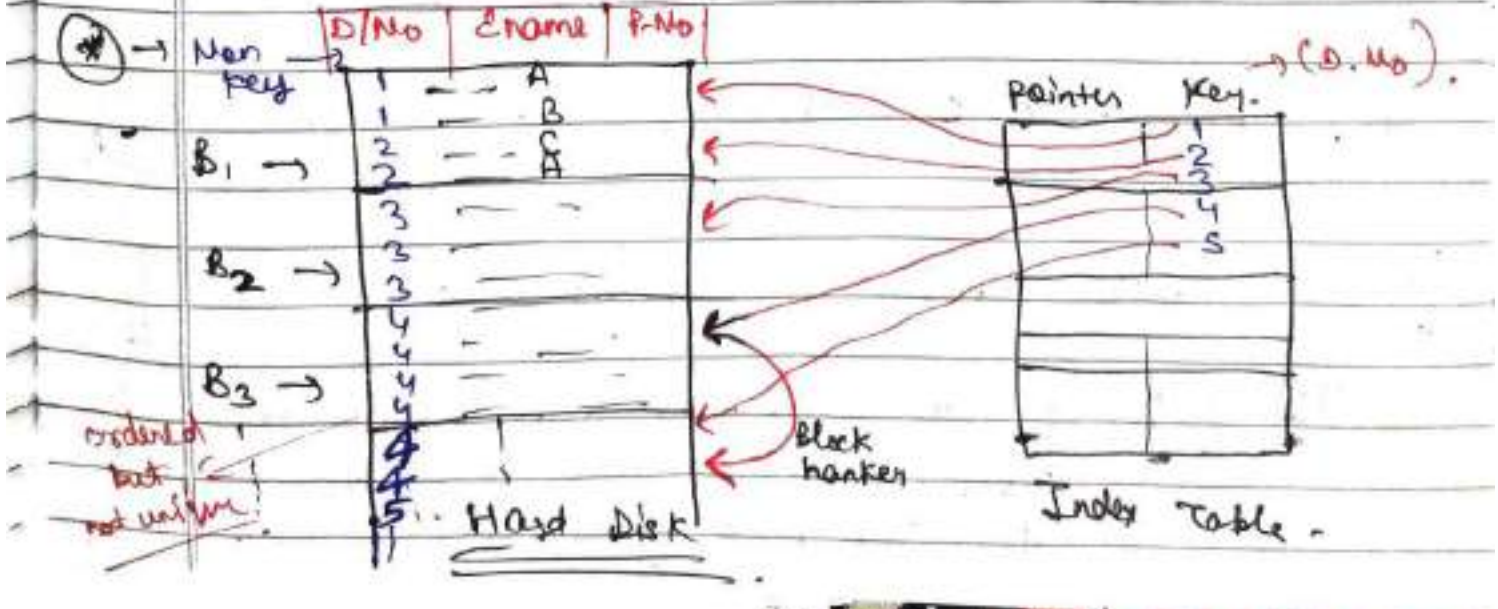
where,
N is the No. of blocks in Index Table.

---

(36) Clustered Index :→

→ data must be ordered & with Non key. (ie, not unique).

i.e,
(value may be repeated but must be sorted.)



→ If our data is multiple times (not unique) in H.D. then, It also comes only 1 time in Index.

→ (In Index, no repetition). i.e, (unique)

* Here, 4 is not only in one block. Some 4s are also in next block. So, How we point to them.
So,
Now, we use [Block pointers] to point to next block for same value.

) Here, Searching criteria slightly increased (↑).

) Clustered Index is also sparse, bcz always. we don't need to make pointer for every value. & Repetition - there is only 1 pointer.

) जो एक से ज्यादा primary Index & Cluster Index नहीं होता सकते, bcz no need. & 1 Table has only 1 primary key ही 1.
So,

* At most 1, (both primary & Cluster Index)

* Total Search Time = $(\log_2 N + 1) + 1$

Where, N is No. of blocks in Index Table.
(These extras are of Block pointers. for, Each block pointer there is +1).

× ×

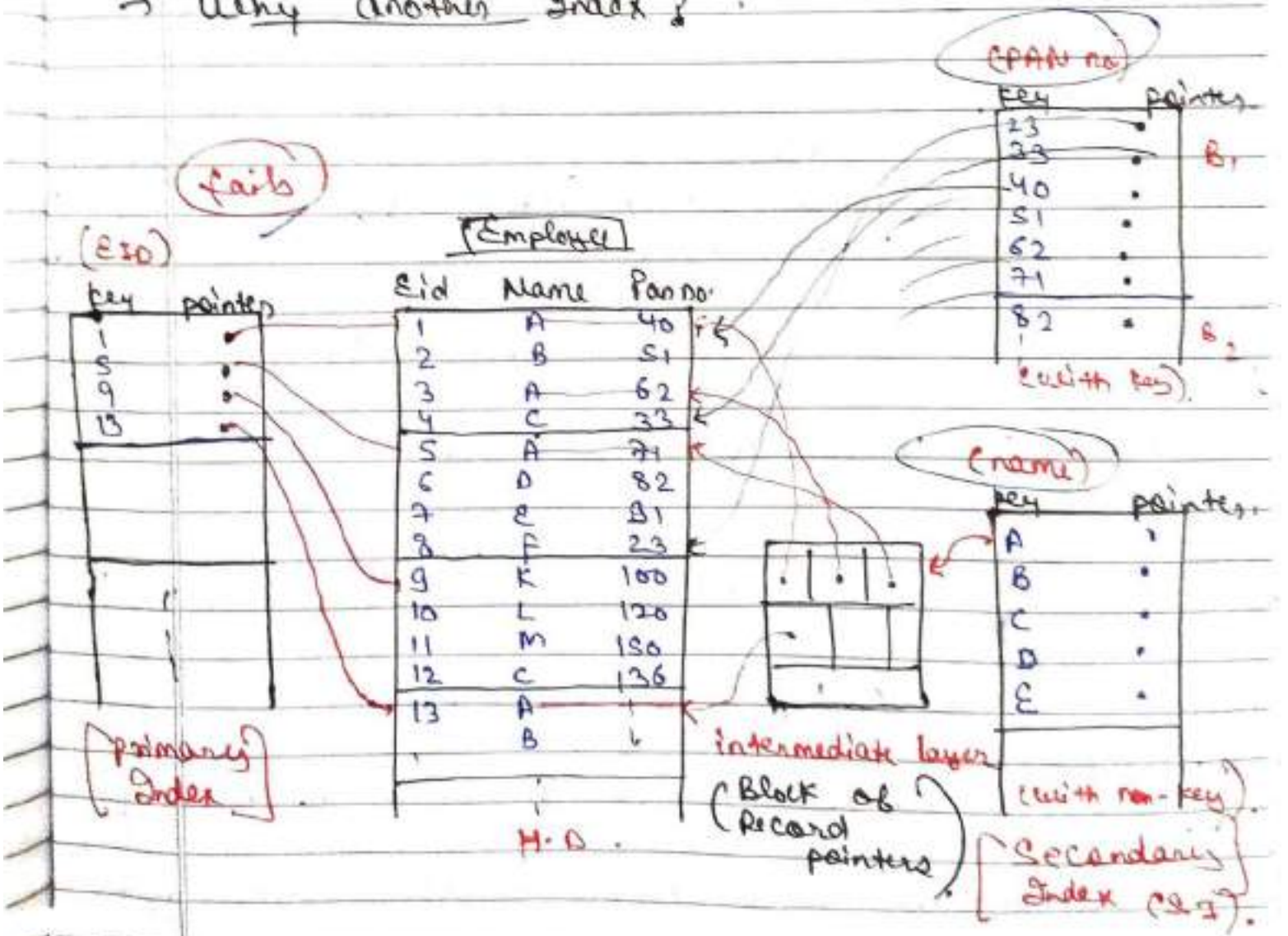
(17.) Secondary Index, (Multilevel Indexing):-
(S.I.).

→ (Already there is an Index in database, we have to made another Index).

→ Where use S.I. ?

Unordered Data.
↙ ↘
Key Non key

→ Why another Index ?



→ Why Another Index?

⇒ Let, H.D. has data of Employee table & data is already sorted on basis of Eid (primary key) → [Unique + Not Null]

↳ bcz most of the query are on E-Id.

↓

(Find * of Employee where E-Id = ---)

↳ then we use, primary Index.
(which is already there, formed on basis of E-Id.)

↳ But, sometimes we also use name & pan no.
then, primary Index fails, bcz it is made on E-Id.

So,
we use Secondary Index here.

(*) Case I: When we also have key.

then
we use PAN bcz
(unique + unordered).

(As u can see in Diag.)

(#) We always use sorted values in Index.

↳ then we apply binary search → Time Saving.

* Index is always sorted & unique.

* (5) Secondary Index on key, is always DENSE.

bcz, we don't have any anchor (leader) here like in primary In.

→ These values are not sorted. Hence, we put them sorted in Index & write all the records of H.D. in Index Table. So, Dense.

i.e.
[No. of records in Index = No. of records in H.D.]

⇒ [Search time: $\log_2 N + 1$]

where,
N is the no. of blocks in Index Table.

(*) Case II when we have Non key with unordered data.

It is the worst case.

then,

we use NAME,

(neither ordered, non key) (unique).

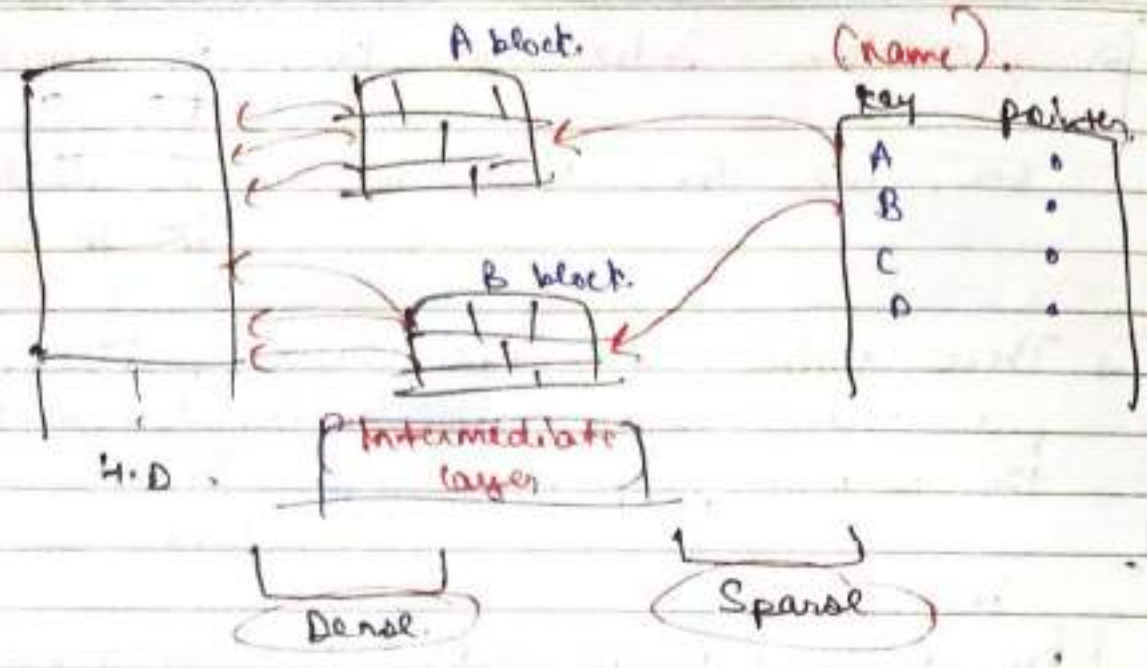
(Secondary Ind in Dbg.)

⇒ In Index, we don't have to take multiple values of A. (only once).
But, A is many times in Hard Disk.
So,
here we made an Intermediate layer.
(It is a block of record pointers).

book में तीसरे Secondary Index (S.I.) भी होता है।

Date
Page No.

Ex:-



⇒ Hence, It is Mix of Dense & Sparse.

So,

⇒ We can say it DENSE Overall.

⇒ Time Complexity : $\log_2 N + 1 + 1$ + ...

(Ans for Intermediate layer)

(And this is only for 'N'. It may be more than $\lceil \log_2 N + 2 \rceil$)

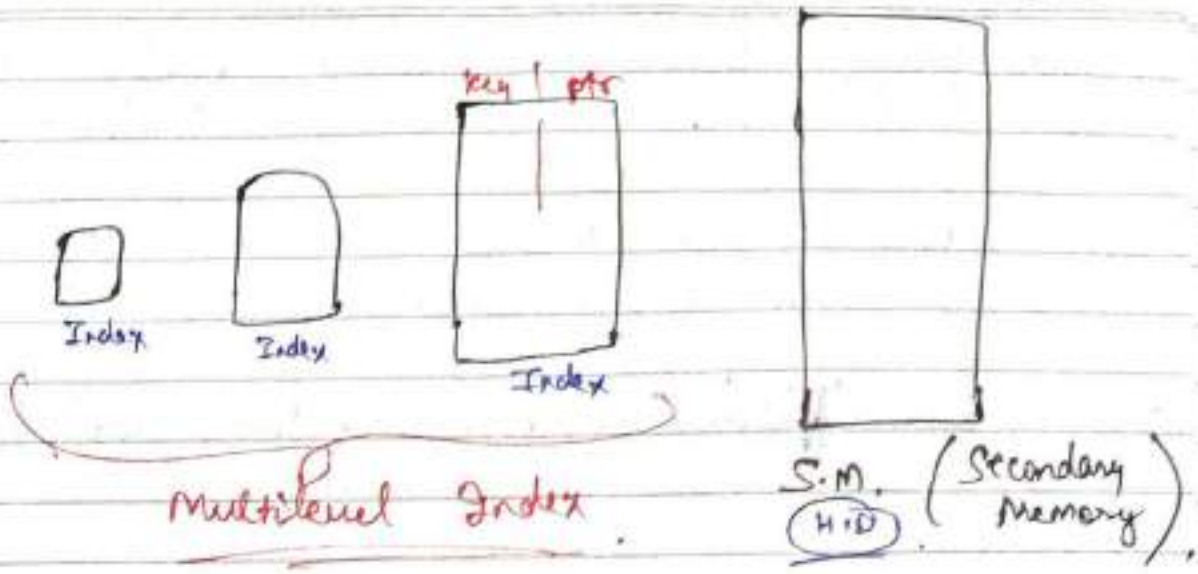
(*) Secondary Index is always dense.

(9th) Intro to B-Tree & its Structure :-

⇒ B-Tree (Dynamic Multilevel Index) :-

(Balanced Tree)

graph → cycle
Tree → acyclic.

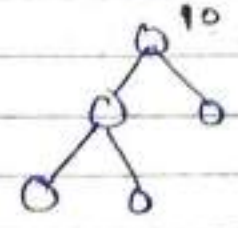


Multilevel Index

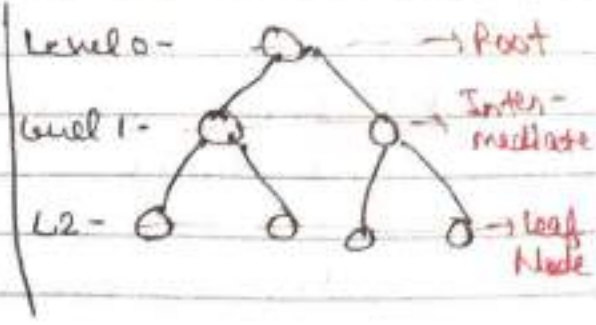
⇒ So this, it is hard to manage. bcz, If we insert data in S.M. then we have also insert in all the Indexes & same for delete.

(#) Balanced Tree (B-Tree!)

→ means, all the Elements are at the same level of leaf Node.

⇒ 

⇒ Not Balanced.



⇒ Block pointer (B.P.) or Tree pointer! → When node denotes his child. Then, we use Block pointers (B.P.).

*Note! → A node can contain multiple values here.

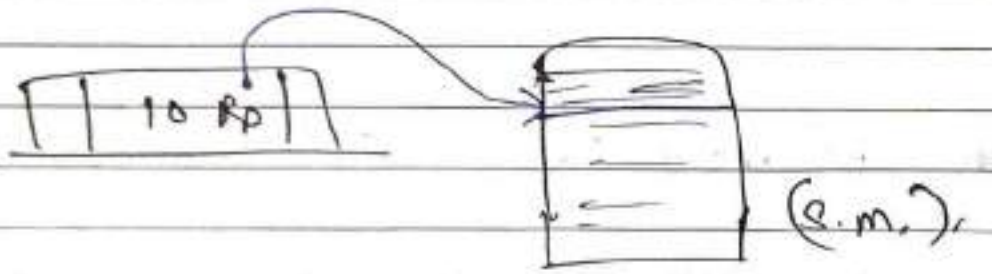
Date / /
Page No.

⇒ Keys :- that on which basis we have to search. Searching criteria - key.

(key को हम Nodes के अंदर रखेंगे।)
(We can insert multiple keys in a node).

⇒ Data pointer (or) Record pointer (D.P.) (R.P.) :→
These are with correspond to keys.

→ This record pointers points to in the Secondary Memory (Hard Disk) where are record is present of that key.



(#) keys = Record pointers.

(#) Block pointer depends on the how many children are there in a Node.

No of children = No of B.P.

(#) Order = p (of a B-tree)
: Max. no of Block pointers.

Order = p = Max no of children a node can have.

(#) Keys = p-1 (Max)
Rp = p-1

Min keys = $\lceil \frac{p}{2} \rceil - 1$

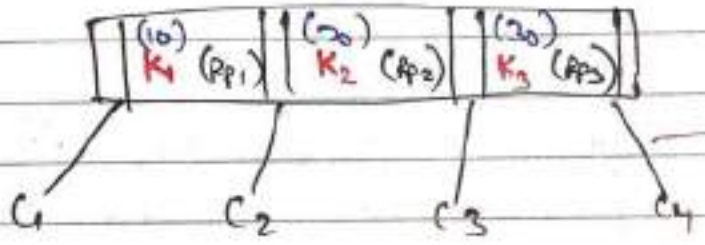
Table :-

Children	Root	Intermediate Node
max	p	p
Min	2	$\lceil \frac{p}{2} \rceil$

↳ ceiling value

Ex :-

Let $p = 4$ = order of a Tree.



(#) We always insert keys in the Sorted Order. (bcz it follows the prop. of Binary Search Tree).

---

99. Insertion in B-Tree :-→

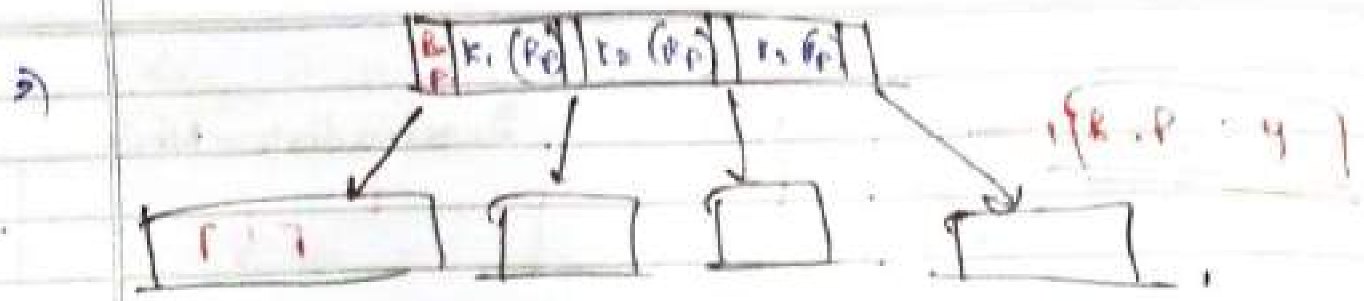
Q:- Insert the following keys into B-Tree, if order of B-tree = 4.

1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

⇒ Order = 4 = max. no of children.

Max keys = $p - 1 = 3$

Min keys = $\lceil \frac{p}{2} \rceil - 1 \Rightarrow 1$.



⇒ We follows the prop of Binary Search Tree in Insertion.

(i) In Binary Search Tree,

[Root → Left = Small Element
Root → Right = Big Element.]

(ii) overflow होते ही, Median निकाल के break करना पड़ेगा,

→ When, n - Even ⇒ $\frac{n}{2}$ →

let, 4 → [o o o o]

$\frac{4}{2}$ → (2) ... Median

→ When, n - odd ⇒ $\frac{n+1}{2}$ = Median

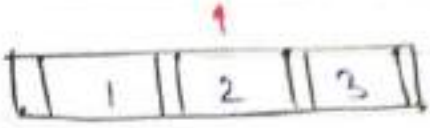
let, 5 ⇒ $\frac{6}{2}$ ⇒ (3) [o o o o o]

Median.

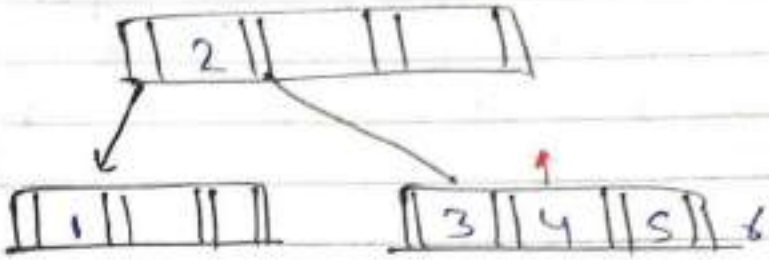
⇒ and, shift Median to upward (↑), & उसके Left & Right Elements उसके नीचे आ जाएँगे । & ie, how we break.

42

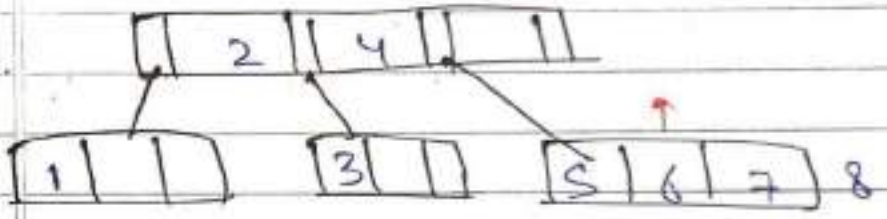
(Ans), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

a) 

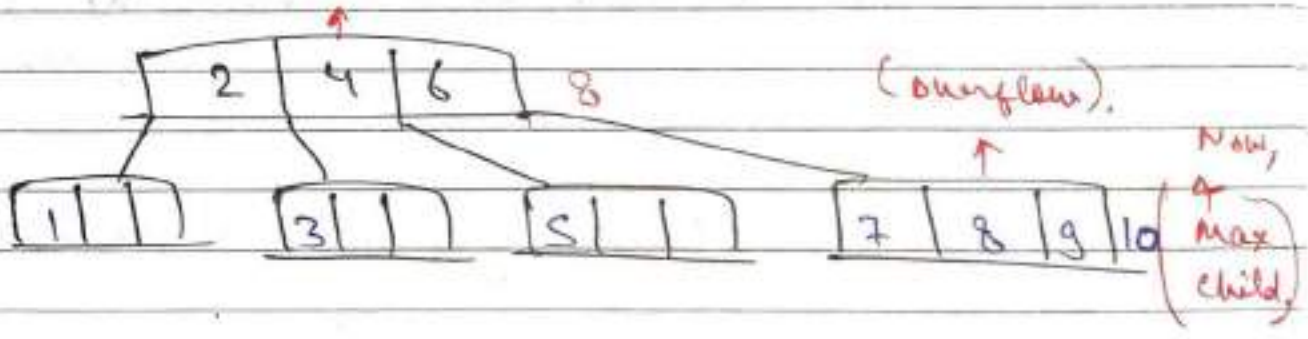
(overflow). $\Rightarrow \frac{3}{2} \Rightarrow \frac{3}{2} \Rightarrow 2$.

b) 

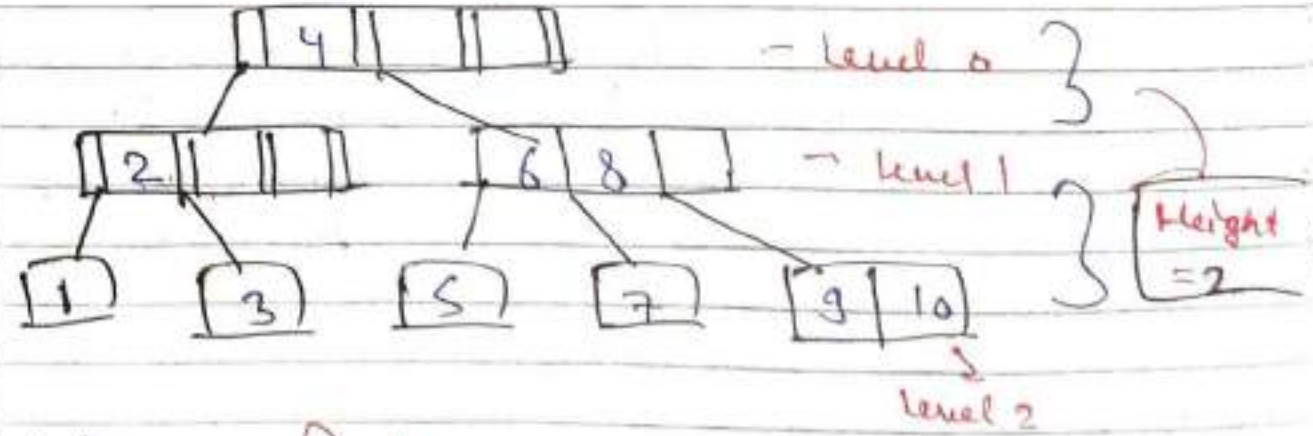
(overflow).

c) 

(overflow).

d) 

(overflow).

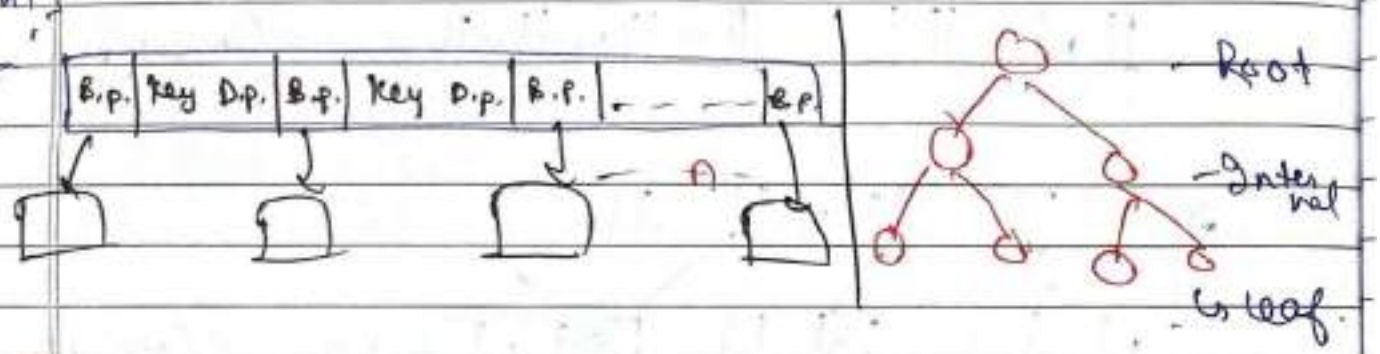
e) 

f) height - 2
levels - 3

100. How to find Order of B-Tree :→

Q. Consider a B-tree with key size = 10 bytes, block size = 512 bytes, data pointer is of size 8 bytes and block pointer is 5 bytes. Find the order of B-tree?

Sol^n :→



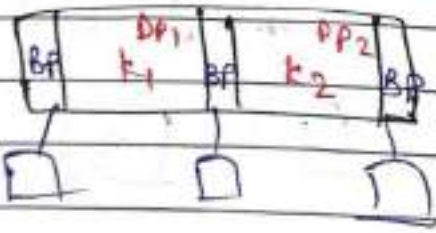
→ Let, In a node / Block has 'n' no of Block pointers.

So,

Total size of B.p. = n × (size of 1 B.p.)

= n.Bp

→ If n B.p.s (or children a node can have) then, (n-1) keys, & Record pointer.



* n × Bp + (n-1) key size + (n-1) Rp ≤ Block size (or) Node size.

2). $n \times 5 + (n-1)(10+8) \leq 512$.

$\Rightarrow 5n + 18n - 18 \leq 512$

$\Rightarrow 23n \leq 530$

$$n \leq \frac{530}{23}$$

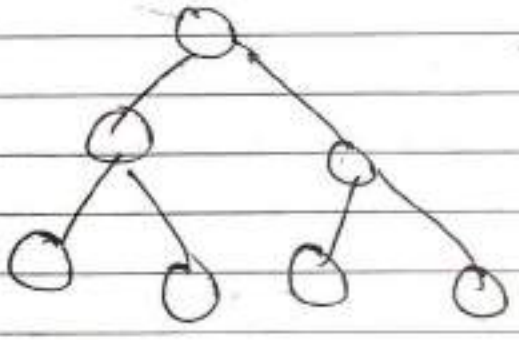
$\Rightarrow n \leq 23.04$

$\therefore n = 23$ ✓. — Max. 23 children.

⇒ Every B.T. represent a children.

SO,

Max. Order = 23. ✓



Max	Min Children
23	2
23	$\lceil \frac{23}{2} \rceil \Rightarrow \lceil 11.5 \rceil$ ✗ (12)
Leaf has no children.	

✓

- Keys = Order - 1 (22) ✓.

---

101. DIB — B-Tree & B+ Tree :→

* B-Tree :→

1) Data is stored in leaf as well as internal nodes.
2) Searching is slower, deletion complex.

3.) No Redundant (Duplicate) search key present.
4.) Leaf Nodes not linked together.

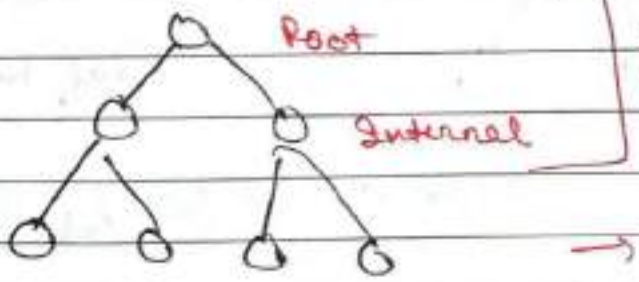
B+ Tree :-

1.) Data is stored only in leaf nodes.
2.) Searching is faster, deletion easy. (directly from leaf Node).
3.) Redundant keys may present.
4.) Linked together like linked list.

* We use B & B+ Tree, to put Index Record. These trees actually contain Index Record.

* Index Record → Key
Data pointer (Record pointer).

*

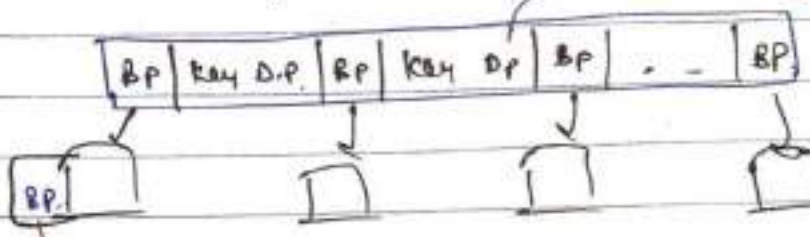


* In B tree, the structure of every node is same. (Either root, internal or leaf).

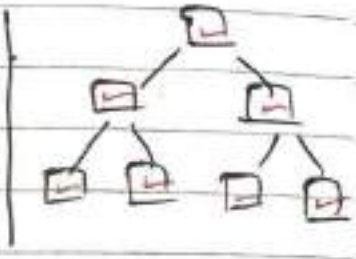
* Leaf Node, has no Children. So, what's the role of B.P. Then, they points to NULL.

B.P - Block pointer.

Structure of B tree,



→ points to data in Secondary memory corresponding to the key.



Structure of B+ tree :→

⇒ Non leaf structure :→ or Internal Node →.

↳ There is no Data pointer (D.P.) in Internal Node structure / Non-leaf Node structure.

P.P / D.P → (X)

Hence,

We have more space in Internal node. Hence, we can create more children & put more keys.

So, that's why,

B+ tree → Breathwise longer.

B tree → Depthwise longer.

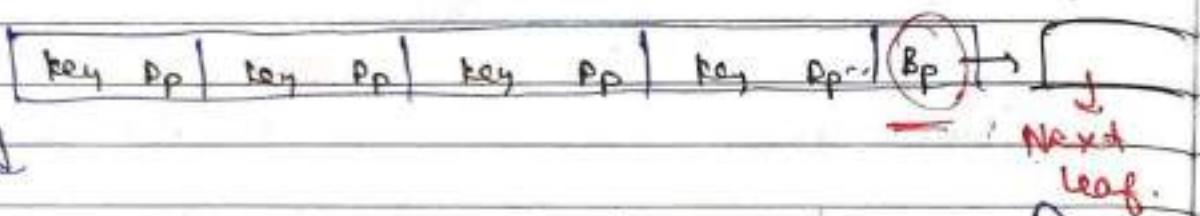
↳ (Coz, less no. of children breathwise as compared to the B+ tree).

So, depth more.

#. So, that's why searching is slower in B tree as compared to B^+ tree.

#. B^+ tree Structure →

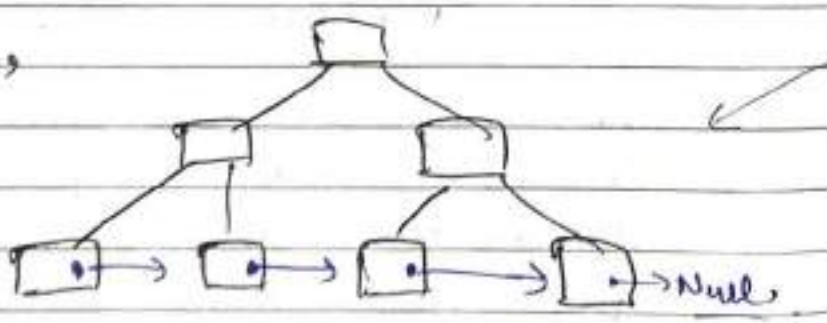
Leaf Node Structure →
(Structure of leaf & Non leaf Nodes are different.)



⇒ (There are no Block pointers in leaf Node Except 1 in last which points to Next leaf.)

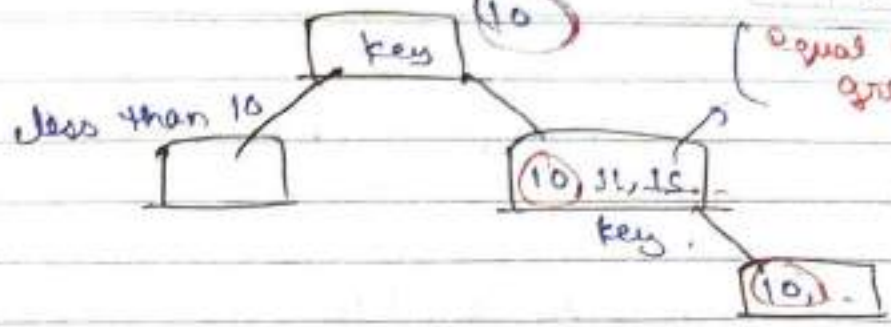
Both, B & B^+ are balanced, i.e, all leaf Nodes are at same level.

In B^+ structure,



⇒ In this, less value to key is in left node & the greater value to key are in Right node. (But here, in Right side, we also have to put the key value with its greater values).

⇒ i.e,



→ Reason :→ bcz, we only search in leaf Node, bcz only leaf Node has all keys with Data pointers (D.P.) are present.
So,
(We also have to reach the D.P. of the key)
so, that's why we also carry the key value upto leaf Node.

→ Searching is that's why faster in B+ tree, bcz have to search only in leaf Node.

& # bcz of this, Redundant keys are also present in B+ tree.
# Leaf Nodes are also linked together.

---

102. Ques on Order of B+ Tree :→

Q!→ Consider a B+ Tree with key size = 10 bytes, block size = 512 bytes, data pointer = 8 bytes & block pointer = 5 bytes. What is the order of leaf & non leaf node ?

Sol!→ Non leaf :→

BP	key	BP	key	BP	key	BP

Leaf Node :→

key	DP	key	DP	key	DP	N.P.

Date / /
Page No.

⇒ Non-leaf :-

$$n \times Bp + (n-1) \times key \leq \text{Block size}$$

→ $5 \times n + (n-1)10 \leq 512$

$5n + 10n - 10 \leq 512$

$15n \leq 522$

$n \leq \frac{522}{15}$ 34.8

$n \leq 34.8$

$\therefore$ $n = 34$ ✓

Order = 34

(Max BP.) or (Max children possible).

⇒ leaf :-

Let x pairs

$$x(key + pp) + BP \leq \text{Block size}$$

$x(10+8) + 5 \leq 512$

$18x \leq 507$

$x \leq \frac{507}{18}$ 28.18

$x \leq 28.18$

$\therefore$ $x = 28$ ✓ Order = 28 ✓

Note :- Order of a leaf node in B+ tree is the no of (key, D.P.) pairs.

H.D.D. - Hard Disk Drive.

103. Immediate Database Modification : →
(Log Based Recovery Methods).

⇒ ↳ Immediate means तुरंत, Instantly.

Ex-1)

A = ~~100~~ 200	
B = ~~200~~ 400	

Hard Drive.

T_1
R(A)
A = A+100
W(A) -200
R(B)
B = B+200
W(B) -400
Commit.

Transaction Log.

< T_1, Start >
< T_1, A, 100, 200 > (old, new)
< T_1, B, 200, 400 > (old, new)
< T_1, Commit >

Redo

⇒ जैसे ही Ram के अंदर value (200) हुई, उसी time पे ही Database में भी (A=200) कर दो। बिना Commit हुए।

⇒ Ie, At time, when we write in Memory (RAM), At same time also update in the H.D. We don't wait for commit.

⇒ But, when we see in Trans. Log,

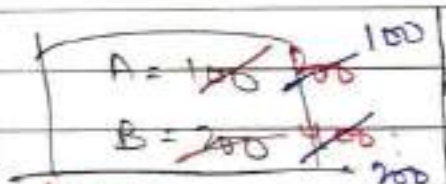
Our Recovery Manager sees the log & check whether T_1 has both Start & Commit or not. & If yes, then it REDO

⇒ Redo, means saves the latest value in the Database. i.e.,

⇒ (Recovery Manager don't see value in the H.D., it only see in the Trans. log & check. (starts & Commit) & then saves in the H.D. database. & if already saved, then Over-write & fixed them.

⇒ But, If

Ex.1



T_1
R(A)
A = A + 100
W(A) - 200
R(B)
B = B + 200
W(B) - 400
↓
fail.

Transac^n log -

$< T_1$, start >
$< T_1$, A, 100, 200 > (old, new)
$< T_1$, B, 200, 400 >.
UNDO

⇒ Here, Recovery Manager don't find Commit in Trans. log. So, he UNDO.

⇒ UNDO, means it saves the old value. But, In database there are updated values now. So, Recovery Manager takes the old values from the Trans. log & saved them in the Database.

In Immediate, we store both old & new value but.
In Deferred, we only store new value. (i.e. only REDO, not UNDO).

that's why
Immediate Database Modification is known as UNDO - REDO strategy.

Ex:- ⇒ Transac' log ⇒

< T_1, start >
< T_1, A, 1000, 2000 >
< T_1, B, 5000, 6000 >
< T_1, Commit > — REDO.

< T_2, start >
< T_2, C, 700, 800 > — UNDO.
old.

---

(104) Ques on DBMS basic Concepts & Data Modelling :-

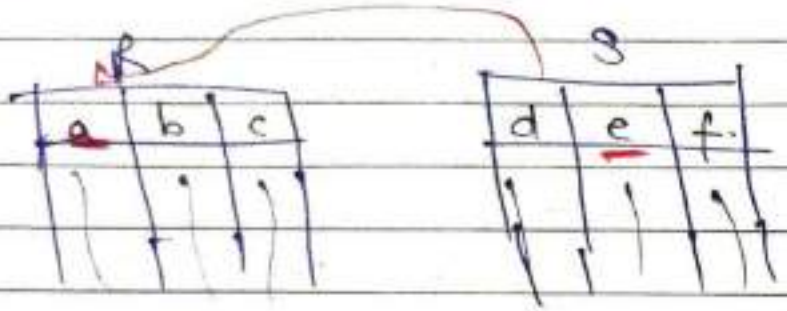
Q:- Let, R (a, b, c) and S (d, e, f) be 2 Rel's. 'd' is foreign key of S that refers to primary key of R. Consider 4 opns on 'R' & 'S'

i) insert into R
ii) insert into S
iii) delete from R
iv) delete from S

→ Which of these can violate Referential Integrity ?

a) i & ii
b) i & iii
c) ii & iii — ✓
d) i & iv

Sol n :-



⇒ Understand from → ये Intelligent student कुछ भी delete कर सकता है लेकिन फिर इससे बच्चों को भी तो Delete करना पड़ेगा।
∴ (If they don't delete → then, violation.)

(2.) The views of total database content is —

a) Conceptual view b) Internal view
c) External view d) physical view.

⇒ This is On 3 schema architecture.

* Internal views & External views,

external view is basically related to outer view.

⇒ User को Total view नहीं होता।
User has partial views (data only he need)

(Mitral - basic, original, fundamental.)

Date
Page No. 51

⇒ physical view तो सबसे 3rd level view हो गया।
(Database storage Related).

⇒ In partial case, External view is Ans.

(3.) In Relational Model, Cardinality is →.

a) No. of tuples
b) No. of attributes.
c) No. of Tables.
d) No. of Constraints.

⇒ If degree, then no. of attributes.

The no. of rows in a Table, called as Cardinality

(4.) _____ Constraint is used to maintain Consistency among tuples in two relations.

a) key
b) domain.
c) Referential Integrity
d) Entity integrity.

⇒ domain basically deals with that which type of data we put into Table. (integer, varchar, character, -etc.)

⇒ Entity integrity, related with primary key.

⇒ key deals with uniqueness.

(10s) Imp. Ques' on Advance DBMS :→
(Big Data & Data Warehouse)

Q-1: What do data warehouse support?

a) OLAP
b) OLTP
c) OLAP & OLTP
d) Relational Database.

⇒ Data Warehouse, where we integrate data at one place from all diff sources.
Ex:- Big BAZAAR, → their outlets mostly found in Every City.
&
(at the end of day, all data integrated from all cities & kept) i.e, data Warehouse.

- & we store this data, to analyse on it later. So that, they can also play ADS - to diff users acc. to their data of purchasing items.
(Apply Mining algorithms on these data)

⇒ OLAP → online Analytical processing.
⇒ OLTP → online Transac' processing.
↳ It works On Current Data.

⇒ (that where to remove item, where open new store, remove or add items) — all these based on Analysis.

2. Hadoop is framework that works with variety of related tools. Common group includes ——

A) Map reduce, Hive & Hbase
B) Map Reduce, MySQL and Google Apps.
C) Map Reduce, Hummer, Iguana
D) Map Reduce, Heron, Trumpet

⇒ Hadoop, basically work on Big Data. (It is a Tool of Big Data).

↳ like Google, Facebook. Big data deals with the multiple petabytes of the data. Now, to process this much of data. We don't use normal tool like SQL Server, Oracle. We use Hadoop. (Bcz, this data is also unstructured).

&

↳ (Hadoop Ecosystem / framework includes many small tools like map reduce. (used to process the data, reduce the data (divide) & then works on batch processing — i.e. works on Multi-processing).

⇒ Hive → (If we write to SQL commands in Hadoop, then use Hive).

⇒ Hbase → (helps in data storage).

→ Also tool like Zookeeper, flume, pig, etc.

2) Google Appo, it is part of cloud.

3. All of the following accurately describe Hadoop, Except :

A) open Source — (listed in Apache, we also install it).
B) Real time Data.
C) Java based
d) Distributed Computing approach.

→ Hadoop, It is basically batch processing. Means we first need data & then we analyse on it.

→ (In Real time, we use SPARK).

4. Which of the following does not comes under five V's of Big Data ?

a) Volume (how much amount of data).
b) Velocity (with which velocity, data ↑).
c) Variety (Structured, unstr., semi-structu).
d) Visualization.

Note :- Let, if there is 'x' amount of total data.
Where, x - is all data on Earth, then,
(80% - 90%) of x is created in last 5-6 years. Mean,
Data is increasing with so much speed.

Unstructured → photos, videos,
Semi-Structured → XML based data.

→ Value → (ie, value of our data).
→ Veracity → (means, trustworthyness).
(how much capable our data to believe on it.)

→ 5 V's of Big DATA :→
1.) Volume
2.) Velocity
3.) Variety
4.) Value
5.) Veracity.

→ Visualize, is a part of Data Analytics. where we visualise our data with help of graphs (pie chart, bar chart, -- etc.)

---

106. Deferred Database Modification :→

→ This topic comes under log based Recovery.
↓
(If there is any failure inside our System, then Either we can recover that System or not).

→ log, is basically a file (small sized file). in which we store our actions. that (what Trans. performs, we stored in logs)

deferred → delayed, postponed, belated

Date
Page No.

→ (like, History in our browser) → when our system fails, to recover Trans.
⇒ (Either we have to Roll back them or Modify them).

– We do that, by seeing the log.

⇒ 2 Methods, by seeing the log :→
1.) Deferred Database Modifica°
2.) Immediate

(2nd)

A : ~~100~~ 200 B : ~~300~~ 400 Database (H.D).	T_1 R(A) A = A + 100 W(A) – 200 R(B) B = B + 200 W(B). – 400 Commit.	Transac° log :- < T_1, Start >. < T_1, A, 200 >. new value. < T_1, B, 400 >. new value. < T_1, Commit >.

Redo

⇒ Here, It don't update in Database hand-to-hande. It updates in Database after Commit. (Bcz deferred – late, postponed).

After Commit, database updates.

⇒ How we Recovery in Deferred ? Let', Say system fails after Commit.
→ So, when Recovery Manager comes, It first check trans. log file. & check. (Start & Commit) in it.

Then, Recovery Manager. (REDO).

- Means, update the new values in the database.

+ (let, due to fail → our database also not there).
then,
- Recovery Manager puts the A & B value from Log in database.

A = 200
B = 400

→ (If it already in database, then overwrite हो जाएगा।)

Case II.

T_1

A = 100
B = 200

W(A). — 200

W(B). — 400

* fail.

Transac" log.

<T_1, start>.
<T_1, A, 200>.
<T_1, B, 400>.

(Roll back.)

- Now, fails before Commit. So, now database value is same as before.

Now,
after failure occur, when Recovery Manager opens the log file. → It sees T_1 starts but not commit. So, Here Recovery Manager don't do anything. He Simply Roll back.

Means, पुरानी value को ही रखो, कुछ भी Opera" करने की जरूरत नहीं है।

⇒ So, Deferred Modification also known as No UNDO/REDO Method.

Ex:- <T1, Start>.
<T1, A, 300>
<T1, B, 400>
<T1, commit>. ——— (REDO)

<T2, Start>.
<T2, C, 500>. ——— (No Action).
Roll back ~~का होगा~~.

⇒ In Deferred, we store only new values.

---

(107) Like Command in SQL :-

⇒ We use Like Command, generally to search the Data.

Q:- 1.) Find Employee detail whose name starting with 'A'.
2.) Find Emp detail whose name ending with 'n'.
3.) whose name contains 'ee'.
4.) whose name contain 'a' in 2nd place.
5.) whose name contain 'a' in 2nd place & name should contain total five characters.

% → Any value (कुछ भी chara. हो)
चाहे 'a' character से हो, कोई problem नहीं।

Emp.

ID	Name
1.	Varun
2.	Arun
3.	Tarun
4.	Amrit
5.	Ranjeet
6.	Ajeet

% → any value in length.
_ → reserved for a value.

1.) Select * from Emp where name like 'A%';
Output → Arun, Amrit, Ajeet

2.) —"— like '%n';
Output → Varun, Arun.

3.) —"— like '%ee%';
Output → Ranjeet, Ajeet

Note:→ If Name → Eesar, Aruneee, — any thing
then, these also comes, bcz sirf 'ee' we read.

% → also 'a' characters include.

4.) —"— like '_a%';
Output → Varun, Tarun, Ranjeet.

5.) —"— like '_a___';
Output → Varun.

In, C++ Here
=> =
=> :=
11 → 05

Date
Page No.

10 8: Basic PL-SQL programming with Execution :→

→ Program 1: Find the Sum of 2 numbers.

```
=> declare
Declarative.  a int ;
              b int ;
              c int ;
              begin
Executable part  a := &a ;        // value given by user. (& to take input from user).
              b := &b ;
              c := a+b ;
              dbms_output.put_line ('Sum of a and b =' ||c);
              end ;
```

// cout in C++
// ; in C++

(#) Program 2: Greatest of 2 numbers →

```
→ declare
  a int ;
  b int ;
  begin
  a := &a ;
  b := &b ;
  if (a>b)
  then
  dbms_output.put_line ('a is greater') ; //a
  Else
  dbms_output.put_line ('b is greater') ; //b.
```

SQL Live में हमें value पहले ही देनी पड़ती है इसमें बाद में user value नहीं दे सकता।.


```
end if;
end;
```

→ (both code works fine). ✓

---

109. PL-SQL : (while, for loop):→

program 3:→ for loop :→

```
declare
a number (2);
begin
for a in 0..10
loop
dbms_output.put_line (a);
end loop;
end;
```

// = 0 to 10.
// By default, increment of 1 by!

program 4:→ while loop :

```
declare
a int;
b int;
begin
a:=0;
b:=&b;
while a<b
loop
a:=a+1;
dbms_output.put_line (a);
end loop;
end;
```

// print from a to b.

→ Output → 1 to 10

then,
→ output → 0 to 9
↓
(Now, first print then increment)
→ output → 10

* In oracle Sometimes there shows Error in output - then write.

⇒ Set Serveroutput on // but in Live SQL It is By default.

Code

⇒ (both code works fine). ✓

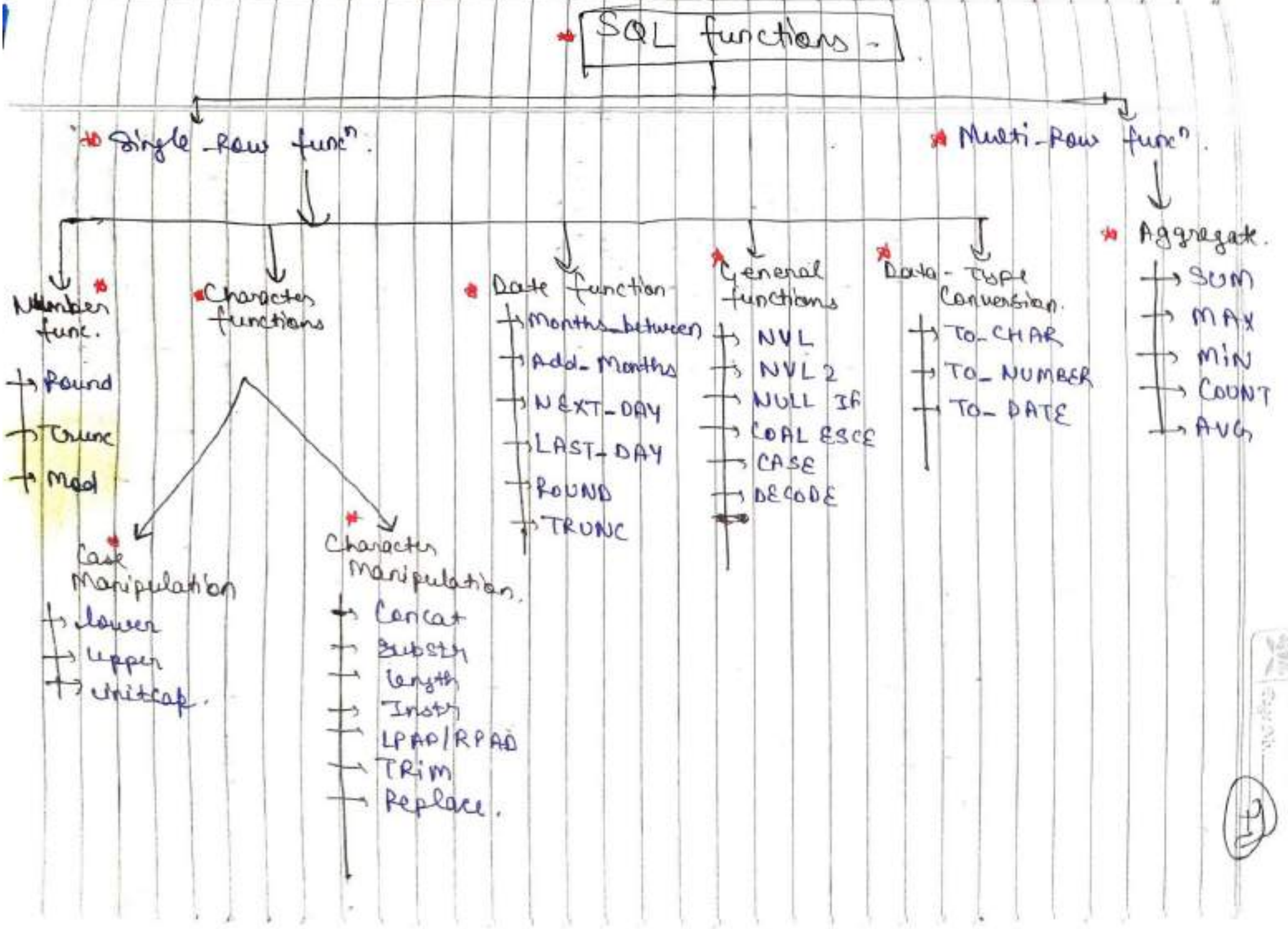
110- Single Row & Multi Row functions in SQL :-

⇒ Single Row :-
If our func. is applicable on single row, and only apply on single row. & gives an single output corresponding to that row. - then, It is Single row funcs.

⇒ Multi-Row :- func.
func. that apply on more than one rows, & gives an single output corresponding to all these rows.

⇒ Round (Round-off value)
⇒ Mod. (gives remainder after divison).
⇒ lower (Convert a string into lower letters).
⇒ InitCap. (Capital the Initial letter).
⇒ Concat (Add 2 string & make 1).
⇒ LPAD/ RPAD (for left & Right padding).
⇒ NVL, NVL2 (to take care of NULL values).

SQL functions :



(111.) Character functions in SQL with Execution :→

⇒ [Character functions]

- Case function Manipulation
- Character Manipulation.

Case function Manipulation:
- → Lower
- → Upper
- → Init Cap.

Character Manipulation:
- → Concat ('Varun', 'Singla') Varun Singla
- → Substr ('Varun', 2, 4) arun
- → Instr ('Varun', 'u') 4th
- → Length ('Varun') 5
- → LPAD ('Varun', 10, '*') *****Varun (10 - length)
- → RPAD —"— —
- → TRIM ('V' from 'Varun') Arun
- → REPLACE ('Varun', 'V', 'T') Tarun.

(*) ANURAG
1 2 3 4 5 6

(#) Execution → If we want to implement these funcs, so, first we have to make schema (table).

⇒ [Output table:→] select * from emp;

ID	NAME
1	Code Smashers
2	Varun Singla.

⇒

```
Create table emp
(
id int,
name varchar2(20)
);
insert into emp values (1, 'Gate Smashers');
insert into emp values (2, 'Varun Singla');
Select * from emp;
Select lower(name) from emp;
Select upper(name) from emp;
Select initcap(name) from emp;
Select concat(id, name) from emp;
Select SUBSTR(name, 2, 5), INSTR(name, 'v')
    from emp;
Select length(name) from emp;
Select lpad(name, 15, '*') from emp;
Select rpad(name, 15, '*') from emp;
Select trim('v' from name) from emp;
Select replace(name, 'v', 't') from emp;
```

Table Created.

2 rows

2 rows.

Select 1 by 1 Each row, & see the output

Output:

REPLACE(NAME, 'V', 'T')
Gate Smashers
tarun Singla.

Output:

LPAD(NAME, 15, '*')
****Gate_Smashers
*****Varun_Singla.

Here, ~~3t~~ counts(_) Space also.

Output:

SUBSTR(NAME, 2, 5)	INSTR(NAME, 'v')
ate S	0
arun	1

(~~3t~~ don't count (_) here).

i.e.,

G	A	T	E		S	M	A	S	H	E	R	S
1	2	3	4		5	6	7	8	9	10	11	12

↑ ate S

(112) # View in Database :→

(Oracle, SQL Server Views)

- What is View in Database ?

→ Virtual Table :→

The Table we create,
Create Table xyz

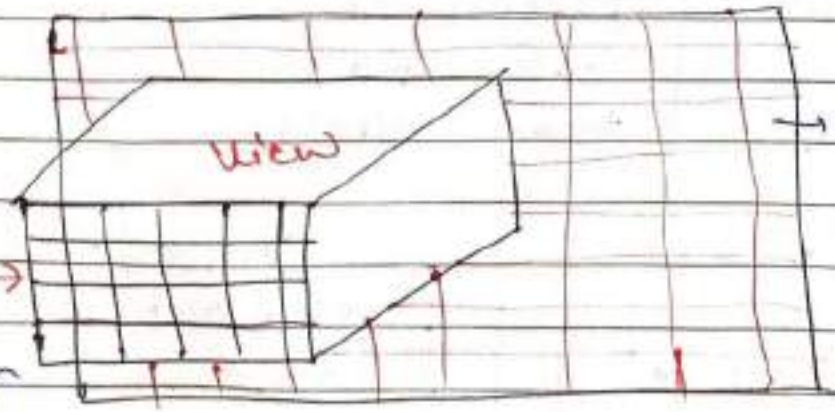
It takes physical space in memory (H.D.)

⇒ But, View is the virtual Table.
It looks like a Table, but it is not actually a table. It don't take space. any.

⇒ View is the result set of a stored query.

Query :→

Create view V1 as Select id from Student;



View →

This view has no physical existence, we don't store it anywhere.

V1 - View) -

(#) The Execute code of this query, Stores after compile. &

After compile, when we write
Select * from V1
then,
it shows the data like a table from the View

(#) So, Actually we just store this little query, rather than Result. We don't Store Result.
That's why it's a Virtual Table.

⇒ Read-only (vs) updatable Views ! →

⇒ If we made any changes in Base Table & that colm is also in our view, then, obviously that also changes.

⇒ Same, If we delete from Base Table. Then also deletes from View.

(#) ⇒ But, if we change Anything in View ! →

And we want that changes to execute also in the Base Table, then updatable Views. &

If we Shut down the (Insert, Delete & update) (DML commands) on view, i.e, we disable these commands. So, that it can't Operate on View. Then, we made Read-Only Views. for it.

(Snapshot – जल्दी से photo लेना।)

Date ______
Page No. ______

2. Materialised View :-
- type of updated version.

(If our data is on the remote server, & I want a copy of that on my local server / Machine. ie, I want a snapshot of that remote data on my local Server. So, that is called Materialised View.

→ (This takes space, but takes less space as comparatively to that data).

(#) We can't apply any DDL Command (Alter, etc.) on view.
But, apply only DML Commands if we make the updatable view.

(#) We can insert the data of more than 1 table, in a View.
ie.
(View can also take data from Multiple Tables.)

(#) We also can take any particular row from Table to make view. like,
———— where address = 'Delhi';
So, all Delhi students come into the view.

(#) Advantages of View :-

1.) To restrict the Data Access.
(like, we don't give the access of original Table to our user, we just give view to them) of some data.

2.) To make complex queries Easy.

Ex. 1) like, we some data from 3 tables.

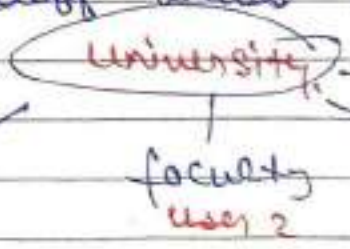
then,
By default, we apply Join or Nested Query.

So,
Its better to just make the views from these tables. Rather, than to write Complex Queries

3.) To provide data Independence,
↓
(We just a give a particular access to a user of a Table, rather than giving the full database access).

4.) To present diff views of the same data.

Ex. 2



(They all have diff privileges).

So,
→ We give diff views to all of them of the same Data (Table).

(#) SQL CHEAT SHEET :→

(#) Examples →

1.) Select all rows from table with filter applied
→ Select * FROM tbl where Col 1 > 5;

2.) Select first 10 rows for 2 columns
Select Col1, Col2 fprom tbl limit 10;

3.) Select all rows with multiple filters applied
→ Select * from tbl where Col 1 > 5 AND Col 2 < 2;

4.) Select all rows from Col1 and Col2 ordering by Col1
→ Select Col1, Col2 from tbl Order By 1;

5.) Return count of rows in table
→ Select Count (*) from tbl;

6.) Return Sum Of Col1
→ Select SUM (col1) from tbl;

7.) Return max value from Col1
→ Select MAX (col1) from tbl;

8.) Computer summary statistics by grouping Col2
→ Select AVG (col1) from tbl GROUP By Col2;

9.) Combine data from 2 tables using a left Join

→
```
Select * from tbl1 As t1
LEFT JOIN tbl2 As t2 ON t2.col1 = t1.col1
;
```

10.) Aggregate & filter Results

→
```
SELECT
    col1,
    AVG (col2) = AVG (col3) AS total
FROM tbl
GROUP BY col1
HAVING total > 2 .
```

11.) Implementation of CASE statement →

→
```
Select col1,
CASE
    When col1 > 10 THEN 'more than 10',
    When col1 < 10 THEN 'less than 10'
    ELSE '10'
END AS NewColumnsName
FROM tbl ;
```

(*) ORDER OF EXECUTION →

```
FROM
WHERE
GROUP BY
HAVING
SELECT
ORDER BY
LIMIT
```

(5). Create

```
CREATE DATABASE Mydatabase ;

CREATE INDEX IndexName
ON TableName (col1) ;

CREATE TABLE OurTable (
        id int ,
        name varchar (12)
);
```

* UPDATE TABLE

```
UPDATE OurTable
SET col1 = 56
where col2 = 'Something' ;
```

* DELETE

```
DROP DATABASE OurDatabase ;

DROP TABLE OurTable ;
```

* DELETE Records

→ Delete from OurTable
where col1 = 'something' ;

* Add / Remove Column.

```
Alter Table OurTable
ADD cols int;
```

```
ALTER TABLE OurTable
DROP Column cols;
```