

Jai Shri Ram

Name : Anurag Singh

Standard : Division : Roll :

Subject : DBMS

INDEX

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 & 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" of Referential Integrity	22	
13.		Super key in DBMS		22-25
14.		E-R Model		25-26
15.		Types of attributes in ER Model		27-29
16.		Degree of Relationship (Cardinality)		29-30
17.		One to One Relationship		31-32
18.		Many to many Relationship		32-33
19.		Ques" 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 SOL).

S. No.	Date	Title	Page No.	Teacher's Sign/Remarks
29.		Minimal cover	38 - 60	
30.		Question on Normalization	61 - 64	
31.		Q1:- Find normal form of a Rel?	65 - 69	
32.		Normalization Questions	69 - 70	
33.		Ques? Explained on Normalisation	70 - 74	
34.		Cover & Equivalence of F.D.	74 - 76	
35.		Dependency preserving Decomposit?	76 - 79	
36.		— " — Example	79 - 80	
④		37. Joins & Its Types	81 - 82	
38.		Natural Join	82 - 84	
39.		Self Join	84 - 86	
40.		EQUI-Join	86 - 87	
41.		Left Outer Join	88 - 89	
42.		Right Outer Join	89 - 90	
43.		Relational Algebra (R.A.)	90 - 91	
44.		projection in Relational Algebra	91 - 92	
45.		Selection in — " —	92 - 93	
46.		Cross/ Cartesian product in — " —	93 - 94	
47.		Set Diff. in R.A.	94 - 95	
48.		Union oper? in R.A.	96 - 97	
49.		Division oper? — " —	97 - 99	
50.		Tuple Calculus in DBMS	99 - 103	
51.		Intro to SQL	104 - 106	
52.		All types of SQL Commands	107 - 108	
53.		Create Table in SQL with execution	108 - 109	
54.		ALTER Command (DDL) in SQL	109 - 110	
55.		D/LW ALTER & UPDATE	111 -	
56.		D/LW DELETE, DROP & TRUNCATE	112 - 113	
57.		Constraints in SQL	113 - 115	
58.		SQL Queries & Sub-Queries (i) & (ii)	115 - 116	
59.		(iii) & (iv) part.	116 - 117	
60.		(v) part	117 - 118	
61.		(vi) part	119	
62.		(vii) part	120	

S. No.	Topic Name	page. No.
63.	Use of IN and Not IN	121 - 122
64.	Use of IN and Not IN in Subquery	122 - 123
65.	Exist & Not Exist Subqueries	123 - 124
66.	Aggregate func's in SQL	124 - 126
67.	Correlated Subquery in SQL	126 - 127
68.	DB Join, Nested Subquery & Co-related Subquery	127 - 129
69.	Find N th Highest Salary using SQL	129 - 132
70.	3 Imp Questions on SQL Basic Concepts	133 - 134
71.	PL - SQL	135
72.	Transac ⁿ Concurrency	136 - 137
73.	ACID properties of a Transac ⁿ	137 - 139
74.	Transaction States	139 - 141
75.	SCHEDULE (Serial Vs Parallel Schedule)	141 - 142
76.	Types of problems in Concurrency	142 - 144
77.	Read-Write Conflict (OR) Unrepeatable Read problem	144 - 145
78.	Irrecoverable Vs Recoverable Schedule in Trans	146 -
79.	Cascading VS Cascadeless Schedule	147 - 149
80.	SERIALIZABILITY	150 - 152
81.	Conflict Equivalent Schedules	152 - 154

EDD → Father of DBMS

DB
Page

①

F

Database Management System (DBMS) :-

→ Basic Intro → 2 tier, 3 tier, 3 schema,
(3 level of abstraction),
graph.

graph.

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 — " — .

graph.

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.

↑ 3-PL, (3 phase lock)
| timestamp).

→ SQL & Relational algebra.

SQL → Structured Query Language.

(SQL is a programming lang.).

D D L command. (Data - Definition).

D M L (Data - Manipulation).

D C L

→ Constraint.

→ Aggregate func.

→ Joins

→ Nested Query.

→ In, Not in, Any, All.

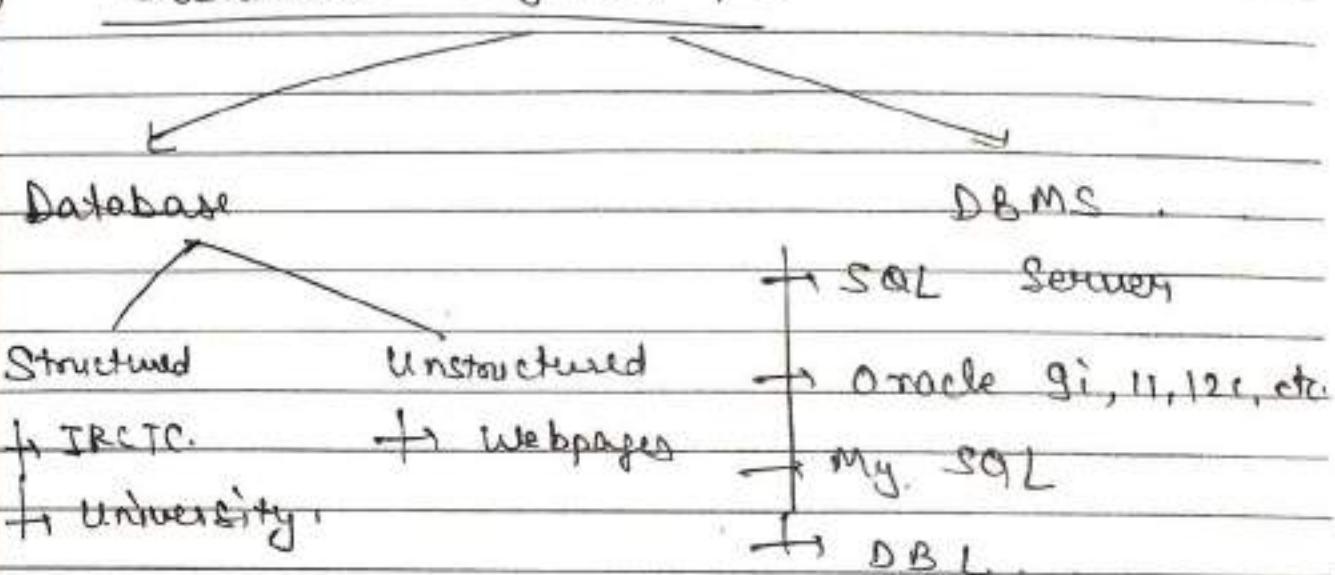
→ Indexing: → (Single level indexing).

→ primary, cluster & Secondary Indexing.

→ B tree, B⁺ tree. (In Multi-level).

(2)

Database System !



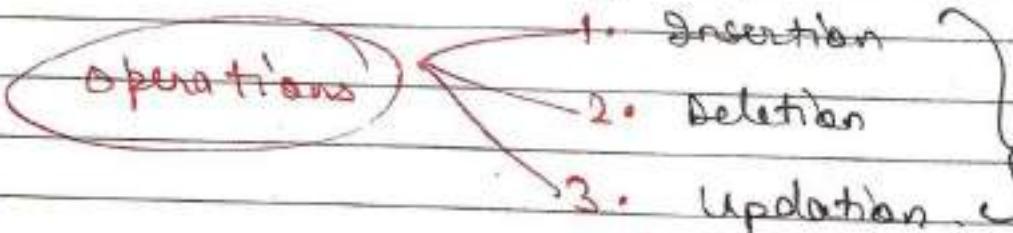
⇒ Database : → "Collector" of Related Data.

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

⇒ Structured : →

RDBMS → Relational Database Management System

⇒ DBMS : → collection of operations



It provides easiness to perform opera's.

⇒ Diff. companies have made diff. DBMS.

Ex: Microsoft Co. ⇒ SQL Server
(Sequential Server).

⇒ Oracle ⇒ 9i, 11, 12 c, etc.
My SQL.

⇒ IBM ⇒ DBL.

(H) Structured Data ⇒ RDBMS

Mean,

We stored in the table form.



Now, Table is technically called as Relo?

"Relo" is most usable form.

Now

→ Store Relo's & access Relo's is done by the Management System i.e. DBMS.

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

Hence,

→ We perform Insert, Delete & update opns on Relo's.

(I) Unstructured Data This is not predefined structure.

A webpage is a collection of photos, videos, chats, etc. i.e.

There is not a particular format in these.

- Hence, RDBMS works only on the structured data.

[Note: Govt. of data on the Earth is unstructured.]

- i.e., There be more technologies on unstructured data.
Ex: Bigdata, Hadoop, etc.

(3)

file system vs DBMS ↗

- file system → 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 use DBMS?

- bcz, we are using the client-server architecture, means,
→ Our data is at a centralised loc' & all over the world users can access that.



↑
I Server
my data



Client (User)

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 KB).

∴

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. i.e., Attributes (name, loco, permission, etc.)

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

In DBMS, no loco", it is totally ~~compt~~ independent. We don't require any attributes here.
(Easiness provide).

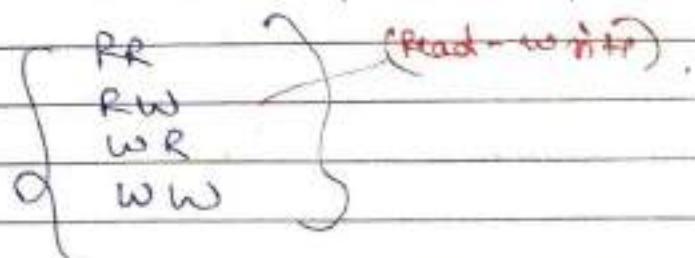
3.) Concurrency: → means Concurrent Access.

Multiple persons can access the data at the same time.

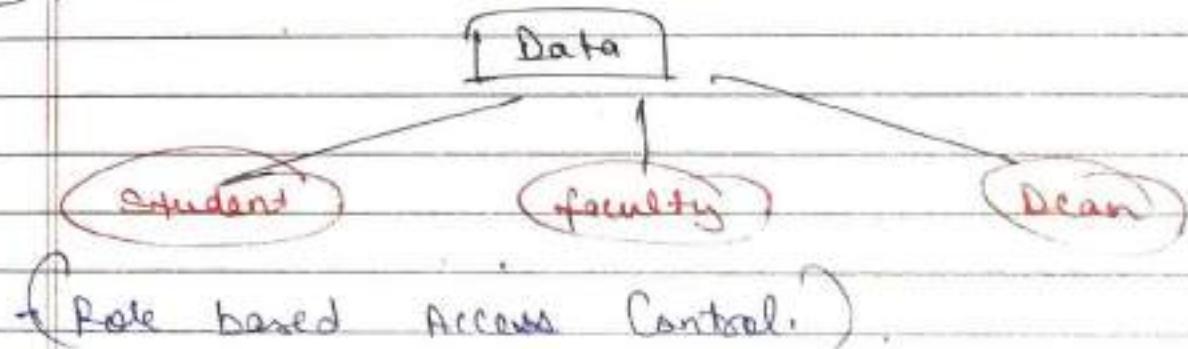
Ex: IRCTC (Indian Railway)
System

File system may show inconsistency when multiple users want access of a file at the same time.

DBMS, have proper protocol for concurrency.



4.) Security: \rightarrow Role based security



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

(c.s)
 File system, has no security for this.
 No level-by-level. No hierarchical.
 DBMS has this role-based security system

5.) Data Redundancy: \rightarrow (Duplication).

In file system, we can store same content ~~no~~ by diff file names.

In DBMS, has many constraints to ensure unique data. Stop redundancy of data.

∴ DBMS is at back end of Client Server & Web Applic.

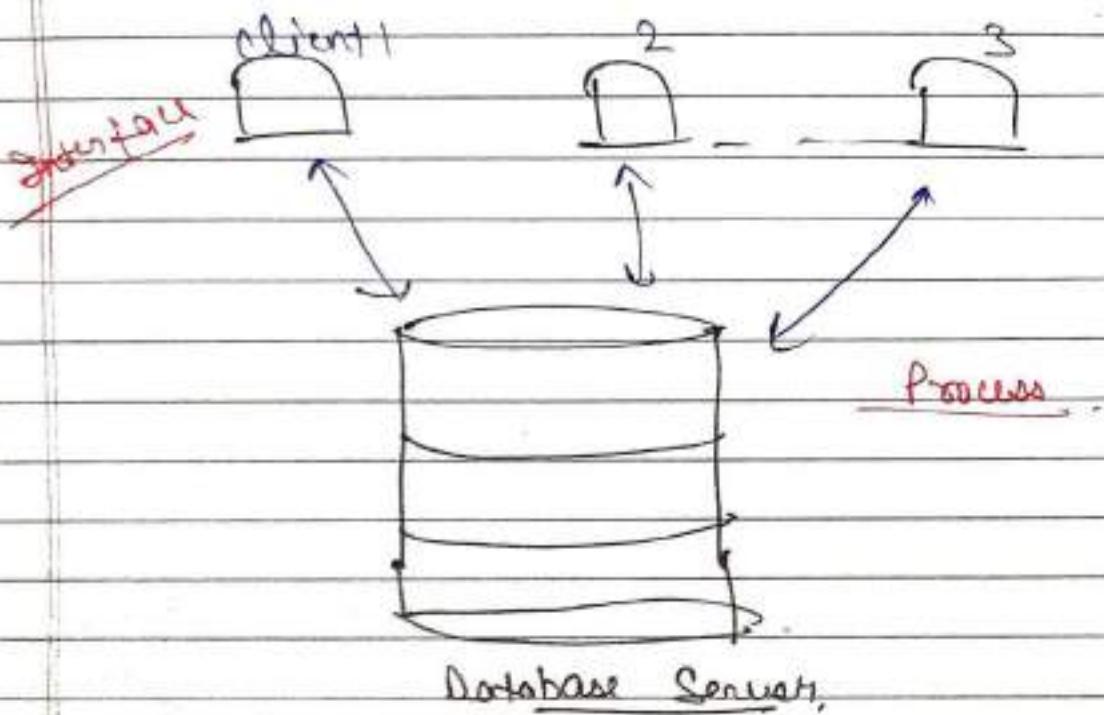
(4)

2 tier \times 3 tier architecture in DBMS !

(+)

2 tier means 2 layers.

1. Client (Machine) layer.
2. Database Server, (Data layer).



→ 2 tier also called as Client- Server architecture.

Ex: i) ~~Indian Railway~~ Indian Railway if we desire ticket by going to railway Sta' at ticket window.

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

(Client \rightarrow Request \rightarrow Database Server).

gives info

→ Hence, limited clients & limited database to which we access. i.e., 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 good, b/c clients are directly interact. with the database.

✓ Advantage: Maintenance is easy.

(*) 3-tier: → (Now, mostly used).

- 1.) Client layer
- 2.) Business layer
- 3.) Data layer.

→ In 2-tier, query is process in Database Server
But,

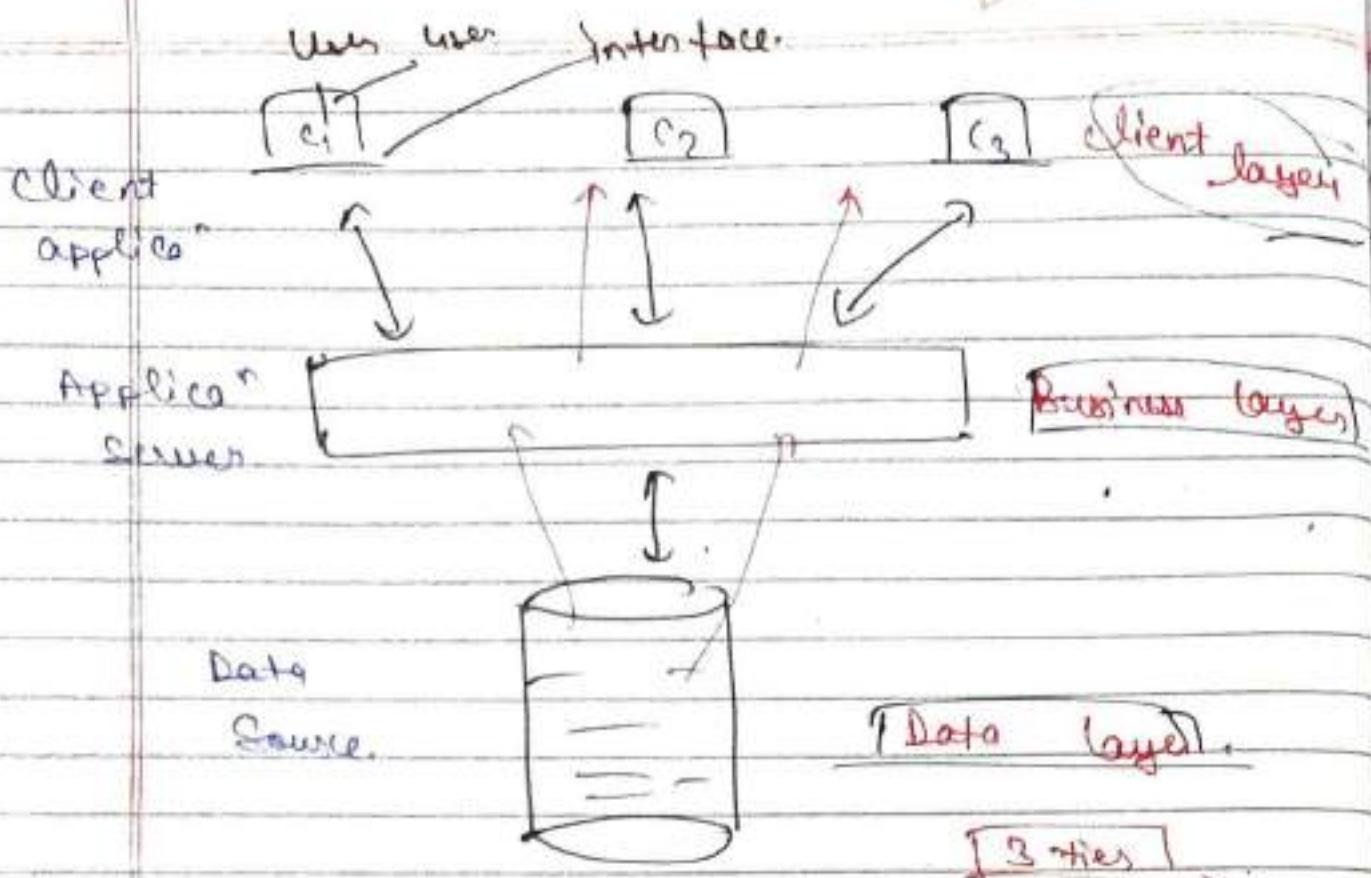
→ here, Business layer supports interface.
(Our query is process in Business layer)
Hence,
we don't give load to Database Server here.

Hence, Business layer acts as Intermediate.

Ex:- TRCTC & banking app. & Gmail app

Advantage:

- 1.) Scalability.
- 2.) Security. (No direct interact of data b/w user.)



(Here, Maintenance is not easy bcz.
it is complex.)

Ex: Web Applics (APPS) are kind of
3-tier Architecture.

If we go physically to any bank or
Railway station, then it is 3-tier architecture.

(S) Schema → Logical Representaⁿ
of a database.

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

D BMS Access & manage the data in this
Schema (Table) form. It is not actually stored
in this table form in drives.

Entity Schema of a Student: + E-R

Pell. No.	name	address
-----------	------	---------

✓ Relationship

(iii) Course: (Schema, Entity)

C. ID	name	Dur."
-------	------	-------

✓ (Relationship)

↳ Logical Representation (structure)

→ But, we implement it by SQL.
(integer, character - -). (Sequential)

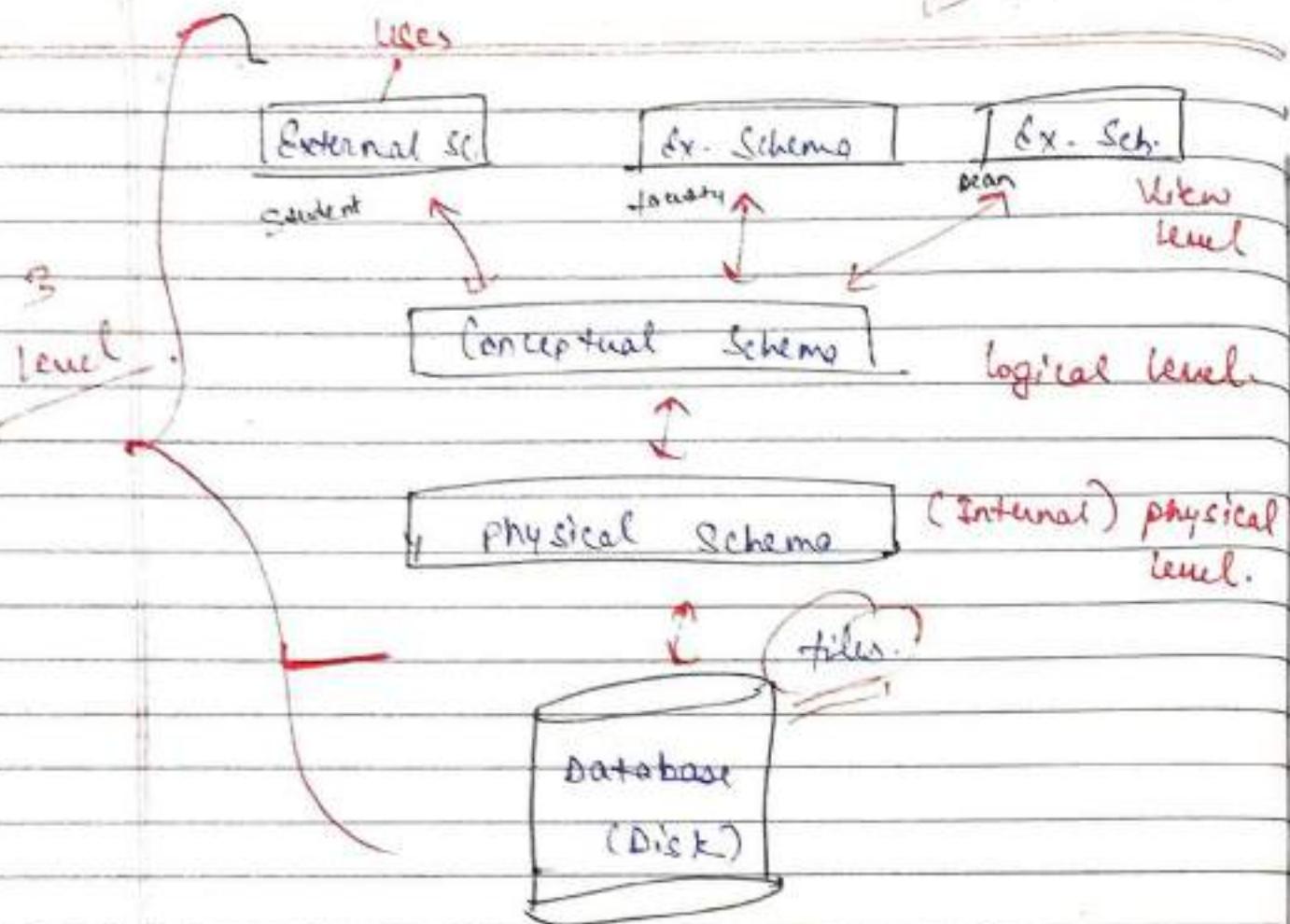
Data Defin' language (SQL) → to implement
Schema. (or to design)

Schema to simply a structure (table form)

6. (Three Level of Abstraction) Cor) (Three Schema Aschi.)

→ Rule hide the "locn" of where the data is stored, from the user).

Ex: Our Mails, we don't know physically (exact locn) where our mails are stored. Ex- Delhi, UK, US, etc.



External Sch! (View level): \rightarrow View that will be given to the User.

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

Ex: \rightarrow after login, which view comes to us is External Schema.

Conceptual Sch! E-R Model

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

\Rightarrow "In-forma" of all tables that we use, & their relationship. A type of Blueprint is this.

physical Schema: → where the data is actually physically present. The loc' of data.

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

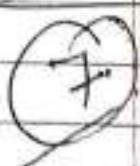
→ front End or Interface Developers are at level of Ext. Schema.

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

↳ Centralised — Data at one place.
Multiple — Data all over the world.

~~Graph~~
~~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.

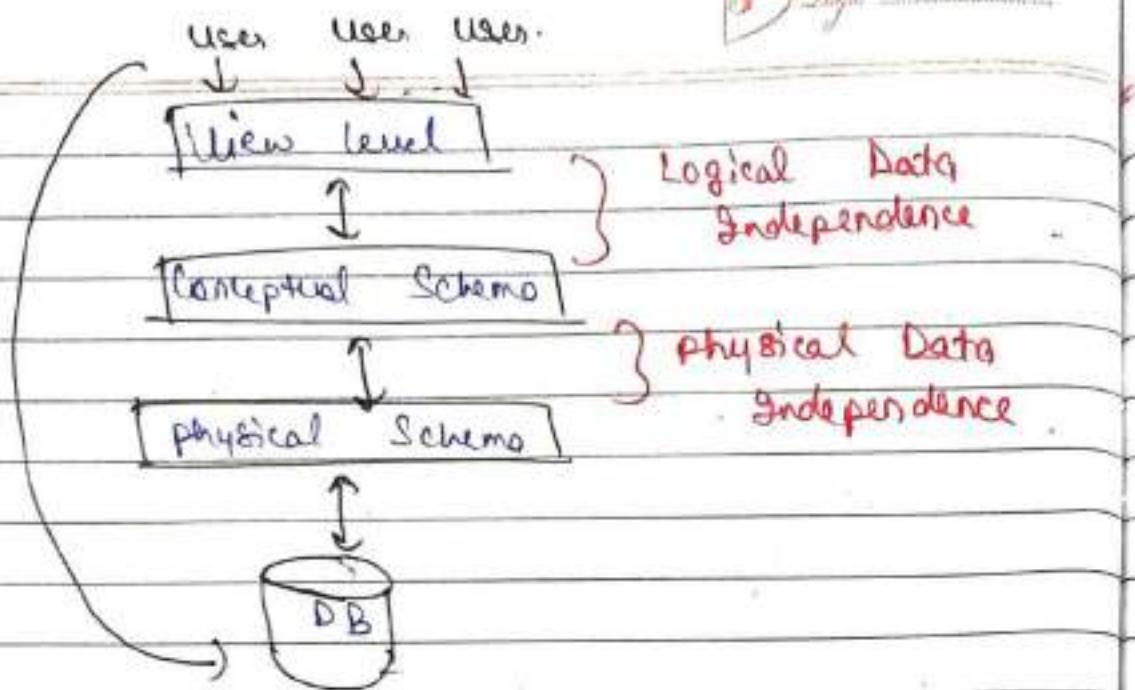
There are 3 layers b/w the user & the data



Data INDEPENDENCE : →

→ Make user independence of data. Aka its loc' & etc.

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



② Logical Data Independence! →

Let,

- Student Table. → add by user 1.

Name	Age	Mob. No.

It will not affect the application program.

- we don't need to write the App^ pgm again. Whatever 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. column is only for user 1.

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

- In actual Table, maybe we have so col's but user can only see S-T col's. So, user think there is only S-T col's. But, there are many.

→ Hence, view level don't change by user changes by user 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, won't 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	Ridley	Shamli	20	1) Same 2 student represents or
2	Anurag	Tanpur	21	2) 2 diff. students with same attr.
3	Ridley	Shamli	20	

4 To identify this, we must have a key.

Ex:-

Student Table

- 1] Aadhar Card
- 2] Roll No.
- 3] Reg. No.
- 4] License 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.

X — X

Q)

PRIMARY KEY : →

(Every lock has one unique key).

i.e.

use uniquely identify 2 things.

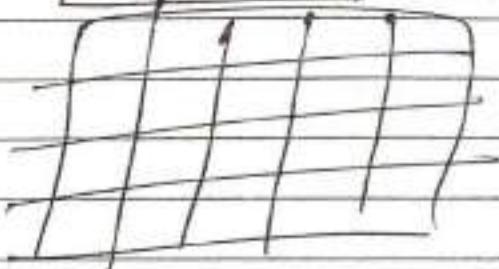
Ex:- Any 2 student can have same name, age, D.O.B. but, some attributes like phone No., Aadhar Card — are always unique.

2) Candidate keys: These are Unique.

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

Candidate keys.

Not NULL



→ phone no., aadhar card numbers can't be NULL & ID should be unique & not NULL.

Case 1: We fill wrong. Aadhar card no.

Case 2: We don't take adm's without Aadhar card no.

In student case, most appropriate key
 to be 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 ~~can~~ have only 1 primary key in a database. Software don't allow us this.
(Why not 2 or more, when we only need 1)

X X

(15)

Foreign key in DBMS : →

(F.K.)

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

- It maintains referential integrity.

Ex: Student →

	P.R.N.	Roll no.	Name	Add.	C. ID	C. name	Roll.no.
(primary key).	1	A	Delhi		C ₁	DBMS	1
	2	B	Chennai		C ₂	networks	2
	3	A	Mumbai				

Course → (F.K.)

- Roll.no is same b/w the student & course. It shows relationship b/w them.

- In Table 2, Roll no' column value ^{take} references from Roll no' attribute (primary key) in Table 1.

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

- No, bcs, If it is not in Table 1 (Roll.no). Until now.

(with f.k.)

2 Table 2 is called Referencing Table.

3 Table 1 is called Referenced Table.
(with p.k.). or Base Table.# Create Table Course
(Course_id varchar (10),
Course_name varchar (20),Code:
Ex:

Rollno int references

Student (roll no.).

);

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

ALTER TABLE course

ADD constraint fk.

foreign key (roll no.)

references student (roll no);.

Note: (1) We don't have to keep the name 'Roll'.
 Atm. some 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) (i)

In mobile phones,
 diff. price at flipkart, Amazon & store.
 So, not Integrity.

(Ex 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 3

P.T.Y

F.K.

	roll.no.	name	add.		C. Id	C.name	Roll.no.
Student	→ 1	A	Betti	Deletion	C ₁	DBMS	1
(Base Table or Referenced Table)	2	B	Munshi		C ₂	networks	2
	3	A	Chd.		C ₃	Car	7
	4	O	Ced.				

Cause. (Referencing table).

(ii) 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).

~~Referencing~~ 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.

SPM

Date _____
Page _____

(21)

But, if same colⁿ is also a primary key in that other table. Then, we can't use this colⁿ. b/c,

primary key cannot be NULL.
(unique, not NULL)

3.) on delete No Action.

(In this colⁿ, our row don't get deleted, first we have to delete from other tables, then we can delete in our table).

3.) updation: → may cause viola' (b/c, if we make '20' of '2', then now how we can get ROLL NO '2' in referencing table).

Roll#:
1) On update Cascade

2) On update Set NULL

3) On update No Action.

#. Referencing table: →

1.) Insert: → May cause viola' (b/c, it's not base table, then how it in referencing table).

2.) Delete: → No violation.

3.) updation: → May cause violation. (b/c, we can't make '20' of 2 if '20' is not in base table).

Not!

- 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.

Q12

Q17

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 \rightarrow

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

Which is correct regarding referential integrity?

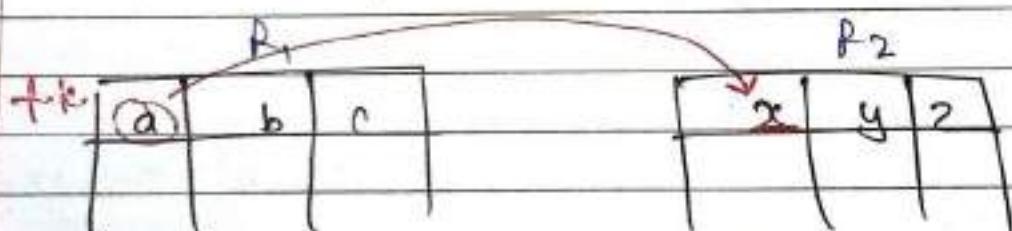
1.) option a & b cause "violation"

2.) option b & c will cause "violation"

3.) option c & d " "

4.) option d & a " " \rightarrow (Ans).

Let x be p.k. in R_2 .



Referencing Table.

Base Table

(or) Referenced Table.

* Note! If f.p. is not there, then we can do anything (insert, delete, update) without any violation.

Q13. SUPER KEY in DBMS? \rightarrow

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

→ Candidate key is minimal.

~~Super Key~~

Candidate key = Roll.no.	Roll no. name age
(C.K.)	
(also ↑ Super key)	
Roll no. name Roll no. age Roll no. name age	} all are Super key.

Candidate key (C.K.) is ~~not~~ at add one ~~it~~, at super key ~~is~~ will.

2). name, age. ~~x (not Super key)~~.

3) Super set of any Candidate key is Super key.

Q: If $R(A_1, A_2, \dots, A_n)$ then how many Super keys are possible.
 If $\rightarrow A_1$ is Candidate key.
 $\rightarrow A_1, A_2$ are candidate keys.

Ans: power set's how many subsets can be possible of given set.

~~Super Key~~ $\rightarrow A_1 \ A_2 \ A_3$ (Either take or not take)
 $\rightarrow 2 \times 2 \times 2 = 8$ possibilities
 $\rightarrow 2^3$.

Now, $\text{Sol}^n \rightarrow R(A_1, A_2, \dots, A_n)$.

i) \exists (exist A_i ने आइया है).

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

Q.

$R(A, A_2, A_3, \dots, A_n)$.

Complementary.

$1 \times \underbrace{2 \times 2 \times \dots}_{n-1}$

So,

$\lceil 2^{n-1} \rceil$ Ans.

ii) $R(A_1, A_2, A_3, \dots, A_n)$.

~~if A_1, A_2 both C.R.~~

A_1, A_2 both C.R.

when $A_1 \rightarrow$

A_1

A_1, A_2

$\underline{A_1, A_2, A_3}$

$\underline{n-1}$

$\underline{2}$

when $A_2 \rightarrow A_2$

$\underline{A_2, A_1}$

$\underline{A_2, A_1, A_3}$

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

But, Some sets are common (btw, $A_1 A_2 = A_2 A_1$).

So, Common are those who has both $\underline{A_1, A_2}$.

So,

those are $\underline{2^{n-2}}$.

Now,

$$\Rightarrow \boxed{2^n + 2^{n-1} - 2^{n-2}} \quad \text{ok}$$

$$\Rightarrow \boxed{2^n - 2^{n-2}}$$

iii) $A_1 \bullet A_2$, combined is C.K.

Now,
 $\underline{2^{n-2}}$ ok.

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

$$\rightarrow \underline{2^{n-2}} \quad 2^{n-2}$$

Now,

$$\begin{array}{c} A_1 A_2 A_3 A_4 \\ \hline A_1 A_2 A_3 A_4 \\ A_1 A_2 A_3 A_4 - - \\ A_3 A_4 A_1 A_2 - - \end{array}$$

To remove these common elements.

$$\begin{array}{c} A_1 A_2 A_3 A_4 - \\ \hline \underline{2^{n-1}} \end{array} \quad \begin{array}{c} A_n \\ \hline \underline{\text{only } 2} \end{array}$$

$$\Rightarrow \boxed{2^{n-2} + 2^{n-2} - 2^{n-4}}$$

$$\text{Q.E.D.} \quad \boxed{2^{n-1} - 2^{n-4}}$$

(4.) E-R Model! \rightarrow (Entity Relationship Model)

\rightarrow Used for logical representation.

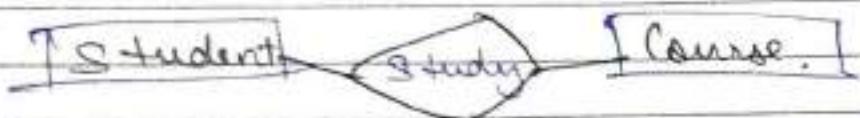
\rightarrow To see logical structure before implementation (Design).

→ 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.

→ Attribute - characteristics of Entity.
 ↗ types

→ Relationship.

→ types → 1 to 1
 1 to many - } 2 types
 many to 1
 many to many } 4 types

* Entity → Student, represented by rectangle []

* Attribute →

* Relationship →

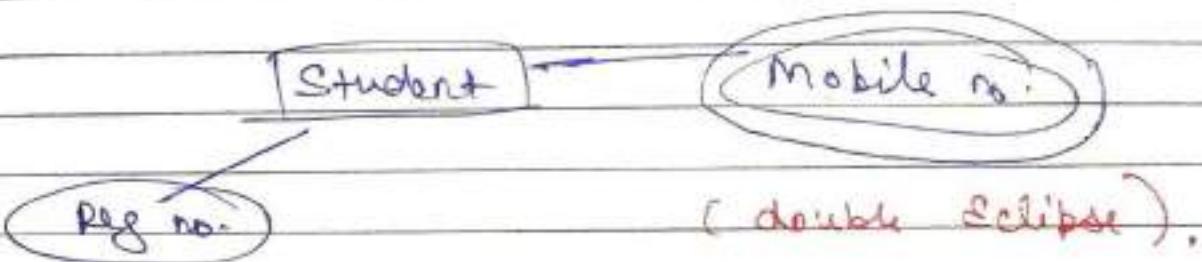
1.S.

Types of Attributes in ER Model:

T student L

1.) Single vs Multivalued attributes :-

↓ ↓
 Reg no. mobile no. (May be more than 1)
 or address (C.No. of a student)

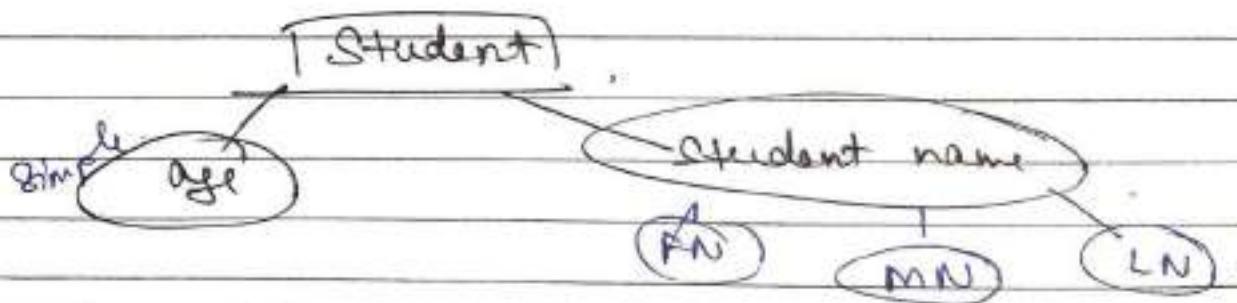


2.) Simple vs Composite Attributes.

Simple - Can't be further divided.

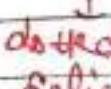
Composite - Composed of more than 1 value.

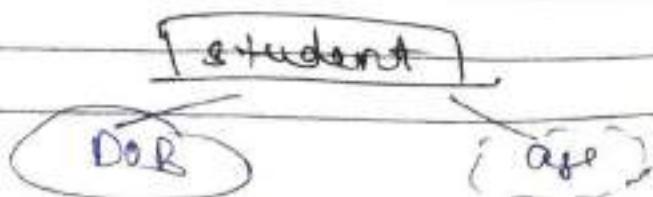
Ex:- name (first name, middle name, last name)



3.) Stored & Derived Attributes :-

Stored → These are stored & can't be derived
Ex - D.O.B.

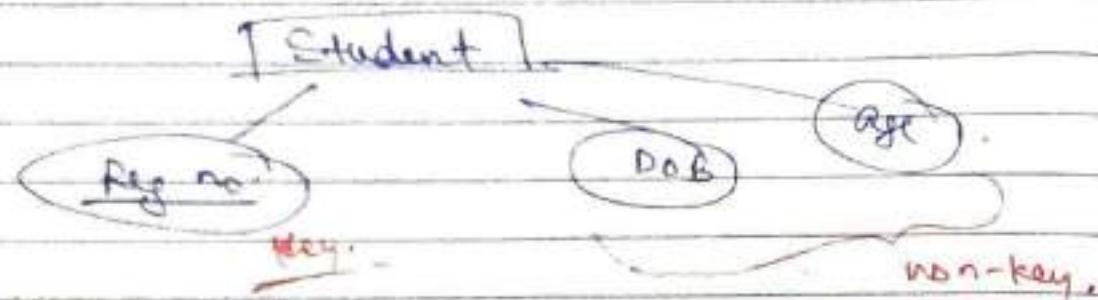
Derived → Ax-Age. (Derived from D.O.B.) 



~~4.)~~ Key vs Non-key Attributes : →
 key - used to uniquely identify.
Unique (No Repetition)

Ex - Reg No is always unique for a student.

Represent with underline (—).



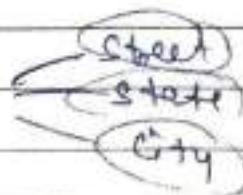
2) Required vs optional Attributes: →

Required → These are mandatory (*) Name
 Optional → can also be leave D.o.B., Add., etc.

3) Complex Attribute: →
 (Composite + Multivalued)

Ex: If a student have 2 Residential Add,
 In each Add., he have 2 phone no.

Add. is Composite



(16).

Degree of Rel' ship! → (Cardinality).

→ how the Entities are connected with each other.

4 types:-

1) 1-1

one to one

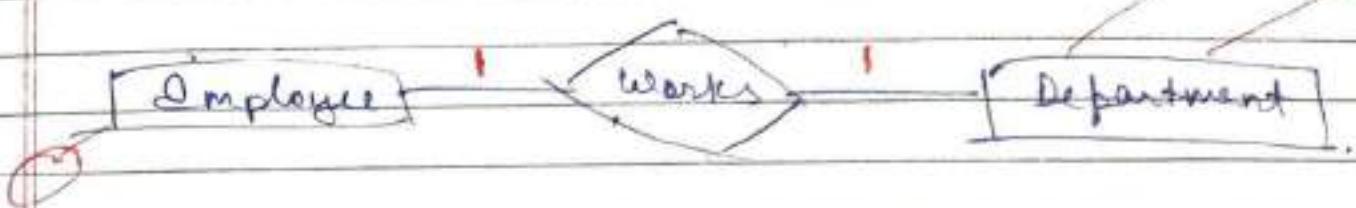
2) 1-n

3) m:1

4) M-M (M-N)

many to many

(1) One - to - One (1-1) :-



Convert Entity into Table :-

(Relationship on all table exists, so what)

Employee & Department int Rel " Works "

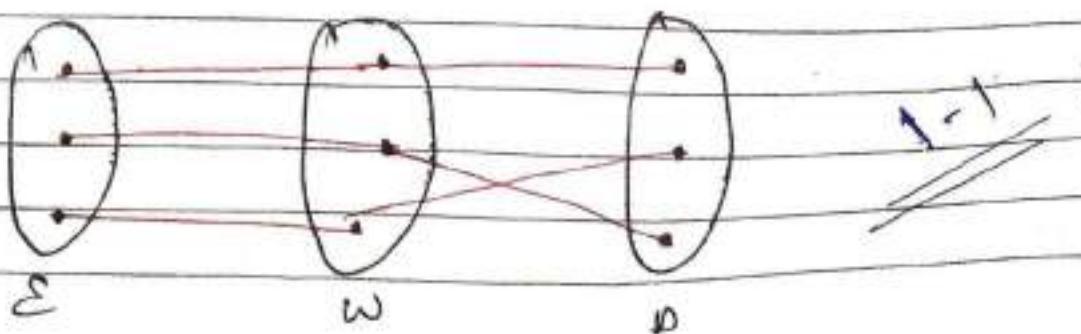
Rel'ship Table → Attributes?

→ ? 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 relationship table, then we have to see relationship (1-1, 1-M, ...).



→ P.K. = Either E.ID or D.ID
(primary key)

E.ID	E.name	age	D.ID	D.name	loc
E ₁	A	20	D ₁	D ₁	IT Bay
E ₂	B	25	D ₃	D ₂	Prod. Delhi
E ₃	C	28	D ₂	D ₃	HR Delhi
E ₄	A	24			
E ₅	B	25			

↓
tech
PK = 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. i.e.
(Merge Table & Department Table)

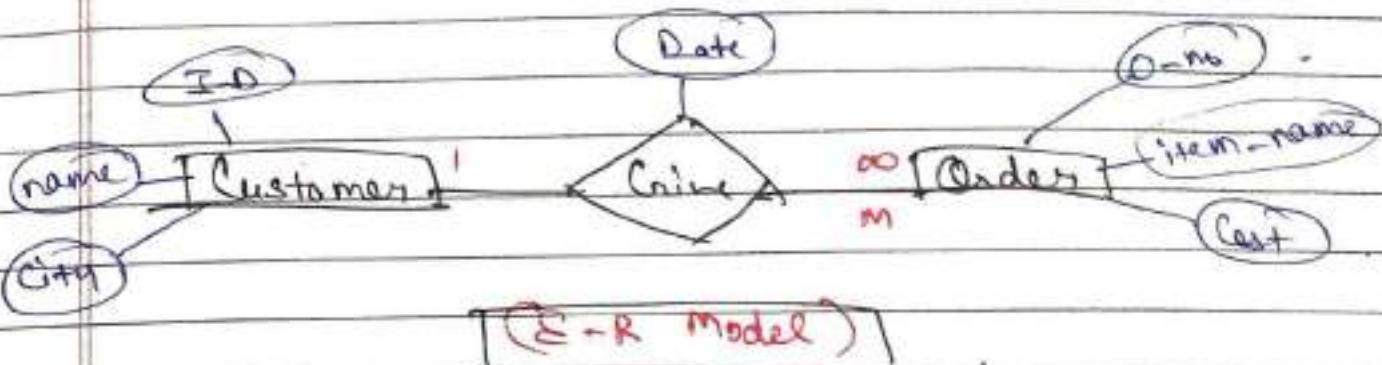
Every Table must have its primary key.



(31)

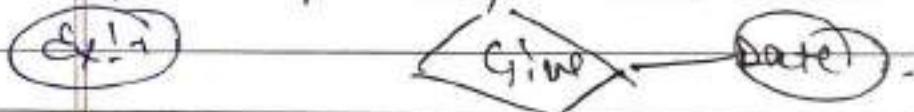
(15)

One to Many Relationship : →
(1 - M).



When we physically implement the E-R Model then we need Relational Model. & we use Tables in Relational Model.

1) Relationship may have its attribute.



2) we call it Descriptive Attribute.

Id	Name	City	I-D	f.k.		O.no	Item name	Cart
				O.no	Date			
C ₁	A	Tal.	C ₁	O ₁	~	O ₁	Pizza	100
C ₂	B	Delhi	C ₂	O ₂	~	O ₂	Burger	200
C ₃	C	Mumbai	C ₃	O ₃	~	O ₃	Pasta	300
C ₄	A	Mumbai	C ₄	O ₄	~	O ₄	Cold-Drink	400

Here, O.no is always diff. & is unique.
So,

P.R. = (O-nos).

Note: Always P.R. of the (many side) in (1-m) is also the P.R. of the relationship table.

* Can we Merge Tables? (many diff side merge)
Yes,

By Relationship \rightarrow Order Table.

(here both have same P.R.).

ID	O-nos	item name	Cost	Date
~	~	~	~	~

Now,

2 Tables.

(M-1) is also same like this.

18.

Many \rightarrow Many Relationship \rightarrow (M-N)

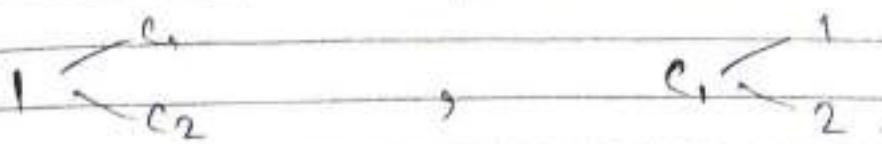
Student (M) \leftrightarrow Study \leftrightarrow Course (N)

f.p. f.l.

Roll no	Name	Age	Roll no	C-id	C-id	Name	Credit
1	A	16	1	C	C ₁	Maths	4
2	B	17	2	C ₂	C ₂	phy.	4
3	A	16	1	C ₂	C ₃	Chem.	4
4	C	17	2	C ₁	C ₄	Hindi	4
5	D	15	3	C ₃			

Base Table Referencing Table Base Table

Many - Many.



⇒ P.K. In Referencing Table (Relationship Table): →

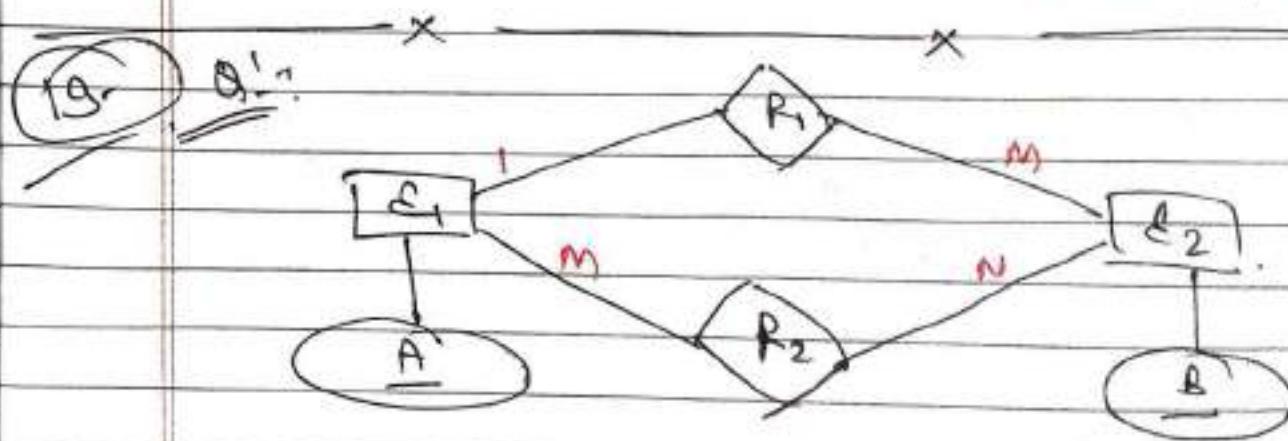
Roll. No repeats &
C-id also repeats

So, Roll no. & C-id both make P.K. combinerly,
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-N)



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

Q1

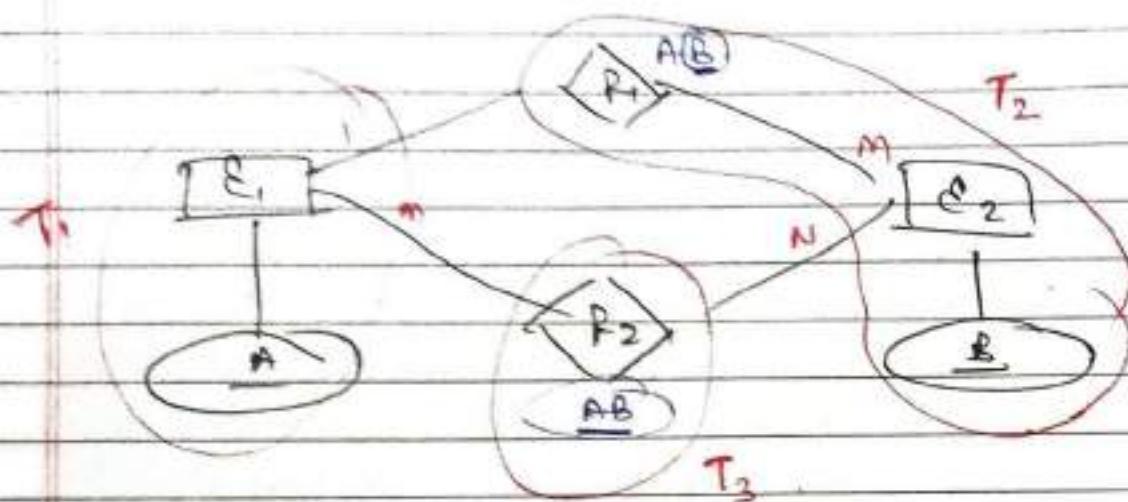
Q2

(a) 3
(b) 5

R_1
 R_2
 R_3
 R_4

4 Tables

But
minimum?



Hence,

$$\left\{ \begin{array}{l} T_1 = E_1 \\ T_2 = P_1 E_2 \\ T_3 = R_2 \end{array} \right\} \quad \text{3 Tables} \quad \frac{\text{sh}}{\text{sh}}$$

E_2 की सभी attributes $P_1 E_2$ in combined table
में आते जाते हैं। So now, we also don't need
separate E_2 table. $[Min^m=3]$ ↳

Q2

Normalization : →

a) It is a technique to remove or
reduce redundancy (duplicacy) from a
Table.

- 1 There are 2 types of duplicacy in Database:-
- 1.) Row level
 - 2.) Column "

(1) Row Level:-

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

Same. (Dupl. Ent.)

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.

(2) Column Level! →

	student		course		faculty		
P.K.	S-Id	S-name	C-Id	C-name	F-Id	F-name	Salary
	1	Ram	C ₁	DB MS	F ₁	John	30,000
	2	Rawi	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	20,000
	5	Mohan	C _{1,0}	MBBS			

→ 4 columns are same in many rows.

⇒ Anomaly

- Insertion Anomaly,
- Deletion "
- updation "

(⇒ Anomaly means problem, occurs on special occasion.)

Now,

S.Id	S-name	Cid	Cname	Fid	Fname	Salary
1	Ram	C ₁	DBMS	F ₁	John	30k
2	Ravi	C ₂	Java	F ₂	Bob	40k
3	Nitin	C ₁	DBMS	F ₁	John	30k
4	Amritpal	C ₁	DBMS	F ₁	John	30k
5	Maria	m	m	m	m	m
		C ₁₀	MRBS			

1) Insertion Anomaly:-

- We want to add data of a new st.
Let, Maria
- Let

University starts a new Course,

C₁₀ - MRBS

- We can't insert this info in table,
Even, we " " — the new faculty data
bcz,

- We only introduce the new course C₁₀,
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.P.
- ∴ we can't insert directly.

3.1 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. \rightarrow 3.1 will delete the whole row.

We don't face any problem 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.

\Rightarrow We here, only delete the detail of student but, bcz of him, all the info get deleted.

Course Info - last & }
Faculty Info - last }.

ie, extra info is remained here & we can't recover it later.

3) updation Anomaly: →

Simple query → [S-Id-4] change name from Amit 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 its 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, Normalization removes Redundancy.

How?

A simple 2NF may be, if we divide that table into multiple tables. Like,

<u>Ex</u>	S-id	S-name
-----------	------	--------

<u>F.K.</u>	C-id	C-name
-------------	------	--------

<u>P.F.</u>	F-id	F-name	salary
-------------	------	--------	--------

This can be 1 of the 2NF.

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

(21.) First Normal Form: \rightarrow (1 NF)

EF Could \rightarrow Father of A.R.M.S.

Table should not contain any multivalued attribute.

student

Roll No.	Name	Course
1	Sai	c/c++
2	Anurag	Tuna
3	Onkar	c/o BMS

\rightarrow Not in 1st NF

Null means not available

501

Now, how to convert in 3rd NF? →

1st way

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

primary key (P.K.) = Rollno. 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	Omkar	C	DBMS

P.K. = Rollno.

3rd way

Roll no.	Name
1	Sai
2	Anurag
3	Omkar

Base Table

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

Referencing Table

P.K. = Roll no.

P.K. = Rollno Course

F.K. = Roll no.

22

Closure Method : →

helps

- To find all the Candidate keys in the Table.

Ex:

Candidate key (C.K.). $R(ABCD)$.(Functional
Dependency).FD of $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$ } .

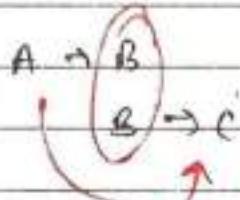
[meaning of closure is that what 'A' can determine]

Here, A is determining B (from FD ①).

closure $\leftarrow A^+ = B$

Ex:-

$$A^+ = BCDA$$



(A can determine itself also)

Ex:- Roll no. can determine itself.

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,

$$B^+ = BCD$$

Hence, A not determine B.

(ii) B cannot be a C.R.

Q

$$C^+ = CD$$

$$D^+ = D$$

prime att. = {A}

C7

only A is C.R.

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

$$\boxed{CR = \{A\}}$$

Note!

$$(AB)^+ = ABCD$$

(AB - itself)
B & C
can.

Here,

AB can be a C.R.

But

it's not a C.R.

bcz

C.R. is always minimal.

In AB only A is C.R.

AB is super key (S.K.).

AB
Super key

(A is Saath
add anything
becomes S.K.)

AB is a Super key.

Ex!

R(ABCD).

FD = {A → B, B → C, C → D, D → A}.

$$A^+ = ABCD$$

$$B^+ = BCDA$$

$$C^+ = CDAB$$

$$D^+ = DABC$$

∴ C.R. = {A, B, C, D}

all +.

prime Attribute ~ are Attribute which is used in making of the C.R.

Q. prime att. = {A, B, C, D} . { bcr, all r }
are C.R.

Non-prime att. = { } . NULL.

Q. R(A B C D E).

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

Now, we have to check that which attribute are coming on the Right side. bcr, attributes on the Right side, it will determine at at \neq ,

$= BCA$ (C & not BCR),
 $\& = BCAS$

Note: Each & Every candidate key must contain bcr, & if present on left side, then only it be written in right side also.

उत्तर: Right side में नहीं आ रहा। यानि उस Left side से होना ही चाहिए। जो candidate key एवं उस होना ही होगा।

Now,

$E^+ = EC$ (E alone is not candidate key
but, it is used in forming C.R.)

Now, Solve!

With n?

$A E^+ A B C D$ $\Rightarrow A \neq A E$

R^+ : $BECDA$

C^+ : CE

DE^+ : $DEABC$

x

Q.

C.R. : { A \rightarrow E, B \rightarrow E, D \rightarrow E }

→ Trick! - First, we get AE as C.R. So, check (A or E) on the right side of FD.

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

Q.

So,

directly, DE is also becomes your C.R.

now check +D, it is depend on 2. So, check that with

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

(Used in making C.R.).

non-prime attr. = { C }

Ans:-

23)

Functional Dependency : \rightarrow (F.D.)

which describes the relationship b/w the attributes.

Determinant \rightarrow $X \rightarrow Y$ \rightarrow Dependent Attr.
 X determines Y (or)
 Y is determined by X .

Ex:-

S-id \rightarrow Sname \rightarrow valid.

1 \rightarrow Ranjit

2 \rightarrow Ranjit

} These 2 are diff.

Ex:-

1 \rightarrow Ranjit

1 \rightarrow Ranjit

}

Same Student

Valid case.

Ex: 1 \rightarrow Parikh
 2 \rightarrow Warren } Valid.

Ex: 1 \rightarrow Parikh } Not Valid.
 1 \rightarrow Warren

(A) F.D. are of 2 types: \rightarrow

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

1) Trivial F.D.: \rightarrow

If

$$x \rightarrow y$$

then,

y is subset of x .

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

Ex: $\frac{\text{Sid}}{x} \rightarrow \frac{\text{Sid}}{y}$

Note: \rightarrow

$$x \rightarrow y$$

LHS \cap RHS $\neq \emptyset$ (Never Null).

Ex:

$$\text{Sid Snow} \rightarrow \text{Sid.}$$

\cap Sid.

Valid.

2) Non-Trivial F.D.: \rightarrow

If
 then $x \rightarrow y$

y is not a subset of x .

i.e.

$$[x \cap y = \emptyset] \text{ (NULL)}$$

Ex:-

 $Sid \rightarrow Sname$ $Sid \rightarrow phone\ no.$ $Sid \rightarrow Loca^*$

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

Properties of F.O! →

1) Reflexivity: If y is subset of x . Trivial

then

$$y \rightarrow y \quad (Sid \rightarrow Sid)$$

2) Augmentation:

If $x \rightarrow y$, then

$$x_2 \rightarrow y_2$$

$Sid \rightarrow Sname$
 $Sid\ phone \rightarrow Sname\ phone$

3) Transitivity:

$$x \rightarrow y \quad \& \quad y \rightarrow z$$

thus,

$$x \rightarrow z$$

$Sid \rightarrow Sname$ & $Sname \rightarrow City$
 $Sid \rightarrow City$.

4) Union!-

$$x \rightarrow y \quad \& \quad x \rightarrow z$$

thus

$$x \rightarrow yz$$

5.) Decomposition:

$\frac{2}{2}$

$x \rightarrow yz$

then,

$x \rightarrow y$

and

$y \rightarrow z$

but,

$x \rightarrow yz$

$y \rightarrow z$, & $y \rightarrow z$

x

6.) Pseudo Transitive:

$\frac{2}{2}$

$x \rightarrow y$

& $wy \rightarrow z$

then,

$wx \rightarrow z$

7.) Composition:

$\frac{2}{2}$

$x \rightarrow y$

& $yz \rightarrow w$

then

$xz \rightarrow yw$

(24.)

2nd Normal Form:

(2nd NF).

2 rules

- Table as Relo' must be in 1st NF.
- All the non-prime attributes should be fully functional dependent on Candidate Key (C.R.) or (C.P.K.).

(There should be no partial dependency in the Relo').

Part of
CK

Non prime

AB

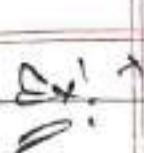
an.

A
(a part)

C
(non-prime)

↑
partial depend.

not 2nd NF

Ex:  

Customer - Id	Store - Id	Locality
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

C.R. : Customer - Id Store - Id

Prime attributes: C - Id
 Store - Id.

Non prime: Locality

Here, Locality is only depend on Store - Id.
i.e., partial dependency.

(to be in 2nd NF it should depend on the both C-id & S-id (as both are PK)).

Now →

Convert it into 2nd NF →

Hence, use make 2 Table.

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

Store-Id	Locality
1	Delhi
2	Bangalore
3	Mumbai

Now, it is fully depended,
but only 1 P.K. is here).

2nd NF

Q1: R (ABCD E F)

FD: {C → F, C → A, C → D, A → B}

Sol: (C.R.)

First, check Right hand side.

Step 1:

F A D B

means
(Now, these attr are
determined by some of
the values.)

Q2:

[On LHS, there must be CE.]

CE = FADB

(C.R. जो हो जाएगी,
तभी CE रह जाएगा)

Now,

EC⁺ = EC F A D B

(All 6 are present)

∴, EC is C.R.

Now, use trick

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

but

not, neither E nor C, none is present.

Q3,

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

i.e.

{C.R. = {E C}}

✓

Proof: check

Step 1 to comp. here.

After finding C.R. = {E C}

$$\left. \begin{array}{l} A^+ = AB \\ B^+ = B \\ C^+ = CF \end{array} \right\}$$

∴, proved.

proper subset is
always less than
a set.

$X \subset X \setminus Y \rightarrow$ proper subset

$X \subseteq X \setminus Y \rightarrow$ subset

Step 3 prime Attributes: {E, C}

non-prime attributes: {A, B, D, F}

Step 3: C.R. = {E, C}

What is proper subset of {C}

↳ either 'E' or 'C'.

Now
check:-

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

for partial dependency check on LHS - either 'E' or 'C' (AND)
check on RHS - whether it is non-prime attr.

not in 2nd NF

C \rightarrow F

↳ partial dependency.

Q. Table is not in 2nd NF.

$C \rightarrow F$ α P.D.
 $E \rightarrow A$ α P.D.
 $EC \rightarrow D$ α P.D.
 $A \rightarrow B$ α P.D.

1 st p.o.

∴ Table is not in 2nd NF.

25. 3rd NF:

Table or Rel must be in 2nd NF.

2.

→ There should be no transitive dependency in table.

Non prime or Non-unique
prime or unique.

SM Date: _____
Page: _____

(3)

not sufficient condn.

(NPA)!

* Mean, Non-prime att. and att. Non-prime att.
determine att. के लिए सकारा ।
(C.R. & Prime att. (P.A.) की दो घटने ने determining
NPA की).

Ex:

Roll no.	State	City	C.R. = α Roll no. }
1	Punjab	Mohali	F.D → α Roll no + state,
2	Haryana	Ambala	State → City }
3	Punjab	Mohali	
4	Haryana	Ambala	
5	Bihar	Patna	

⇒ P.A. = α Roll no. 3

⇒ N.P.A. = α State, City 3.

Ans.

Here, $\text{Roll} \rightarrow \text{State}$ and $\text{State} \rightarrow \text{City}$.

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

It is not in 3rd NF.

Ex: R (ABCDEF)

FD: {AB → C, C → D}

⇒ C.R. = α AB }

P.A. = α A, B }

N.P.A. = α C, D }

Transitive

AB = ABCD.

α → β
N.P.A. N.P.A.

It is not in 3rd NF

C.R. + anything = S.R.

Super Key



Date: _____
Page: _____

~~Ex:~~ R (ABCD).

FD: (AB \rightarrow CD, D \rightarrow A).

Soln: C.R.: {AB, DB}.

PA: {A, B, D}.

NPA: {C}.

(B not on RHS)

$$B^+ = B$$

$$AB^+ = ABCD$$

Now, A on RHS.

$$DB^+ = DBAC$$

is also C.R.

~~Ex:~~ Now, for each F.D.

This \rightarrow C.R. on S.R.

RHS \rightarrow L.O.P.A.

check only 1 bcz [OR].

FD: (AB \rightarrow CD, D \rightarrow A).

✓ ✓

Table is in 3rd NF.

bcoz,

(NPA \rightarrow NPA') is not present here.

26

BCNF. (Boyce Codd Normal Form) \rightarrow

\Rightarrow also called as special case of 3rd NF.

~~Ex: Rollno, Name, Address, Student~~
~~Only one key~~

Student

Rollno	Name	Address	Age
1	Ravi	K0123	20
2	Warun	M034	21
3	Ravi	K786	23
4	Rahul	D2B6	21

Table is in
3rd NF
already.

C.R. = & Roll no, Voter - Id 3.

f.D. :- $\left\{ \begin{array}{l} \text{Roll no} \rightarrow \text{name} \\ \text{Roll no} \rightarrow \text{matrid} \\ \text{matrid} \rightarrow \text{age} \\ \text{matrid} \rightarrow \text{Roll no} \end{array} \right\}$

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

Here, 3NF 's की (OR) और cond' तो हटा दी, जिसमें RHS के P.A. को नहीं भी आए चल जाए वाला).

Here,

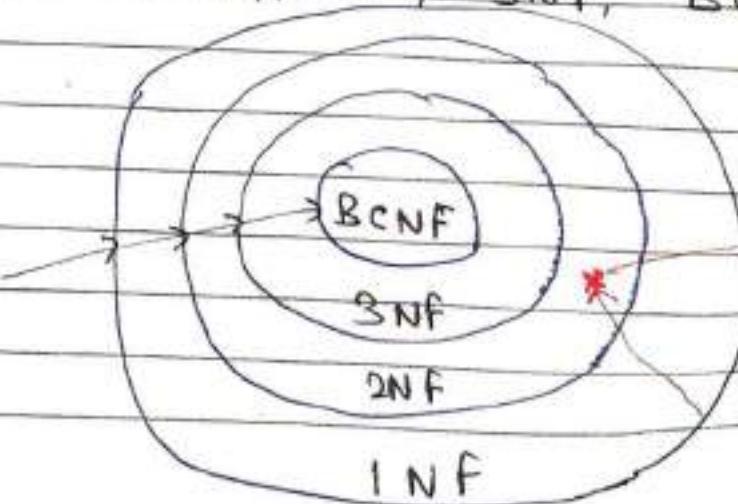
we only want C.R. or S.R. in L.H.S. & RHS के दोनों लोगों देना नहीं।

So, check all the f.D. one by one.

In all the 4 f.D, the LHS is C.R.

It is in BCNF form.

Compn:- INF, 2NF, 3NF, BCNF, ...



It is outside
3NF &
in 2NF &
INF both.

Tc

INF.

$$2NF = INF + \text{cond}'$$

$$3NF = 2NF + \text{cond}'$$

$$BCNF = 3NF + \text{cond}'$$

{ }

X

X

2*

Lossless & Lossy "Decompositi" !

Join

We normalize table \rightarrow or we decompose
table into INF, ... 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.R₁

A	B
1	2
2	2
3	3

R₂

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 R₂.C from R₂ Natural Join R₁
where R₁.A = '1'.

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

cross product! If R₁ has x rows &
R₂ has y rows }

then

their join has x.y rows.

k

condi': common ~~element~~ col^m of both
Tables (R₁ & R₂) - here (B) has the
same value in join Table.

k

Natural Join = Cross product + Condi'.

Now,

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

Now,

R₁

(state).
Spurious of
tuples.

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

→ table after
Joining.

In original Table (R), we have only 3 tuples (rows).

but,

After Joining, in R' , we have 5 tuples.

\rightarrow It is a ^(partial) 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.

3. In original, for $A=1$, $C = 1$.

but

In join table, for $A=1$, $\left. \begin{array}{l} C=1 \\ C=2 \end{array} \right\}$

④ Why we get Extra's (C_2 tuples more)

∴ here, we take B as common in both Table.
But

Criteria for Common Attribute should be C.R. or S.R. of either R_1 or R_2 or both.

So, we have to C.R. or S.R. of original Table.

- R has duplicacy in Table. We have to choose attr. 'A' for Right Ave. bcs, A is unique. $\{1, 2, 3\}$.

R_1	$\{AB\}$
R_2	$\{AC\}$

→ We get 3 tuples also in joining table.

- # Cond' for lossless Joining Decompos'! →

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

$AB \cup AC = ABC$.

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

$AB \cap AC$

$A \neq \emptyset$

(C.R. of original table)

3.) R_1 C.R. (or) R_2 C.R. (or) Both

To take common attribute

28. fill normal forms with real life Examples →

	1st NF	2nd NF	3rd NF
1	→ No multivalued Attribute.	→ In 1st NF +	→ In 2nd NF +
2	→ only single valued.	→ No partial dependency. * only full dependency.	→ No Transitive dependency → No, NP A. should determine N.F. A.

$AB \rightarrow C$

If A & B are 2 FDD of C, then both will use the Emply. C.

SM

B C.N.F.

\rightarrow In 3rd N.F.

+

- L.H.S must \Rightarrow No multi-valued
be C.R. or dependency.

S.K.

$$X \rightarrow Y$$

4th N.F.

\Rightarrow In BCNF

+

5th N.F.

\rightarrow In 4th N.F.

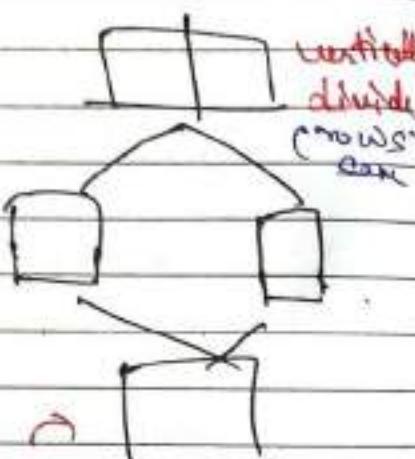
+

lossless
decomposition

Name \rightarrow 3 Phone no.

\rightarrow 3 Mail Id

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

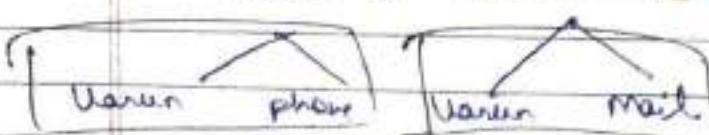


table

Name	M ₁	E ₁	
Name	M ₁	E ₁	
1	M ₁	E ₂	Very
	M ₁	E ₃	long
	M ₂	E ₁	site
	M ₂	E ₂	also
			don't able
			to form a key

May be extra tuples
comes. So,
make C.R. as
common attribute in
both tables.

Make 2 table.



(Now, no multi-valued dependency).

28)

Minimal Cover : \rightarrow (Irreducible).

Q! For the following functional dependencies,
find the correct Minimal Cover \rightarrow

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

- a) $A \rightarrow B, C \rightarrow B, D \rightarrow A, AC \rightarrow D$.
- b) $A \rightarrow B, C \rightarrow B, D \rightarrow C, AC \rightarrow D$.
- c) $A \rightarrow BC, D \rightarrow CA, AC \rightarrow D$.
- d) $A \rightarrow B, C \rightarrow B, D \rightarrow AC, AC \rightarrow D$. ~~Ans~~

Quesn: Our RHS in F.D. must be single.

Step 1: ~~$A \rightarrow B, C \rightarrow B, D \rightarrow ABC, AC \rightarrow D$~~ .

By "decomps" prop, separate them.

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

Step 2: Remove the redundant F.D. \rightarrow .

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

\rightarrow Let, $(A \rightarrow B)$ be R.F.D., Now check the closure of A,

$$\{ A^+ = A \} \quad (\text{not all } +).$$

i.e.,

$(A \rightarrow B)$ is not redundant. We can't remove it.

\rightarrow Do same check this for every F.D. \rightarrow .

but, $D \rightarrow B$ is redundant.

\rightarrow Let, $(D \rightarrow B)$ ~~x~~ then, $D^+ = \overline{DABC}$ (all +).

\rightarrow remove, $D \rightarrow B$.

Now we will check $D \rightarrow B$, at which set R.F.D.,
which at SET 2 remove the R.F.D. 1.

Now, for

$$\overbrace{AC \rightarrow D}^{\times}$$

$$AC^+ = ACB$$

So,

also includes

$$\overbrace{AC \rightarrow D}^{\times}$$

Now, we get

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

Step B: Now, we only want 1 Attrib in RHS.

Here,

$$\overbrace{AC \rightarrow D}^{\times}$$

Now, check by removing A. & then
check closure of C.

$$C^+ = CB$$

So, C+ is not 'A' yet which means
A must be included in addition to C.
Hence,

$$\overbrace{AC \rightarrow D}^{\text{at least}}$$

Same check w/ A, by removing C.

$$A^+ = AB$$

So, can't remove C.

So, $\overbrace{AC \rightarrow D}^{\times}$ can't be reduced.

So,

$$\{ A \rightarrow B, C \rightarrow B, \boxed{D \rightarrow A}, \overbrace{D \rightarrow C}^{\text{complex}}, AC \rightarrow D \}$$

So,

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

Q:

30) Question on Normalization : →

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

F.D. ! $\delta_{AB \rightarrow C}, C \rightarrow DE, E \rightarrow F, F \rightarrow A \}$.

~~Soln:~~ Find all C.F.s in Reln? →
~~Step 1:~~

By closure method! → .

(B is not on RHS.) So, & compuls.

$$B^+ = B \text{ } \underline{\quad}, A^+ = \underline{A} \text{ } .$$

- $AB^+ = ABCDEF$ (Call +).

C.

[AB is C.F.]

Now, check if A on RHS.

We get,

$$F \rightarrow \textcircled{A}$$

C.

[FB is also C.F.]

Now, check F on RHS!

$$E \rightarrow \textcircled{F}$$

∴,

[EB is also C.F.]

Now, check E on RHS.

$$C \rightarrow D(\textcircled{E})$$

∴,

[CB is also C.F.]

Now check + C on RHS.

$AB \rightarrow C$

AB is already in c.t.
So, we get all c.p.

C.P. = { AB, FB, CB, CB } \vdash 4 CP

Step 2: Write all prime attr & NPA! \Rightarrow

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

NPA = { D }

Step 3: Now, check FD: 1.

{ $AB \rightarrow C, C \rightarrow DC, E \rightarrow F, F \rightarrow A$ }

No = 1

Check 1-by-1.

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

Redundancy \rightarrow decreases

Mean, When table is in BCNF, then redundancy is lowest. Δ in 1NF redundancy is highest.

So, to check highest NF, use Start from BCNF!.

In BCNF, we know, all LHS of all FD's should be CP or SP.

$\{AB \rightarrow C, C \rightarrow DE, D \rightarrow F, F \rightarrow A\}$	X	X	X
	↓		

It is not in BCNF form.

→ Now, 3NF →

check 1st transitive dependency

(NPA → NPA)

$LHS \rightarrow C.P. \text{ or } S.P.$	then	it is 3NF.
$RHS \rightarrow \text{is a f.a.}$		

Now, 1st Cond'n is already checked in BCNF.

so, here check only 2nd cond'n that whether RHS is P.A. or not.

	$AB \rightarrow C$	$C \rightarrow DE$	$D \rightarrow F$	$F \rightarrow A$
BCNF	✓	X	✓	X
3NF	✓	X	✓	✓

∴ 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. It already 2NF if it is 3NF. Check only for 'X' tick.

Check!

LHS is proper subset of C.R.
 RHS is non-prime R.H.

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

	$AB \rightarrow c$	$c \rightarrow de$	$e \rightarrow f$	$f \rightarrow A$
BCNF	✓	✗	✗	✗
3NF	✓	✗	✓	✓
2NF	✓	✗	✓	✓
1st NF	✓	✗ <small>(bcz both cond' are not in 2NF).</small>	✓	✓

Ans is 1st NF. Why,

bcs

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

Ex:

Table \rightarrow R (ABCDEF)

all are general R.H.,

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

Table Already 1st NF \rightarrow 2nd NF.
 (assume already).

1st NF.

Ans

proper subset is always less than a set.



65

Q1

Find out Normal Form of a Reln! →
(from INF - BCNF).

Q1:- $R(ABCDEF)$,

FD's: $\{AB \rightarrow C, C \rightarrow D, C \rightarrow E, E \rightarrow F, F \rightarrow A\}$

C.R. = $\{AB, FB, CB, CB\}$.

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

NPA = $\{D\}$

Soln! Now, let we assume that $R(ABCDEF)$ is already in 1st NF.

→ Check \rightarrow 2nd NF! →

(partial) P.D.
(super)

Cond'n $\Rightarrow \left\{ \begin{array}{l} \text{LHS must be proper subset} \\ \text{of any C.R.} \\ \text{[and]} \\ \text{RHS must be a Non-P.A.} \end{array} \right\}$

FD's of $\{AB \rightarrow C, C \rightarrow D, C \rightarrow E, E \rightarrow F, F \rightarrow A\}$

(full dependency).

∴ not in 2nd NF.

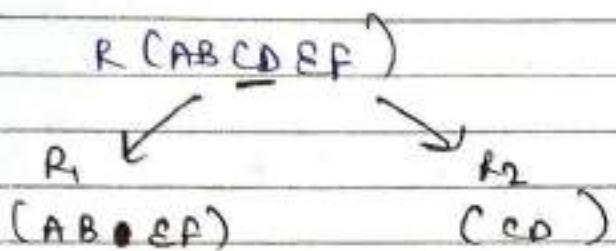
→ Make it in 2nd NF! →

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

(Cond'n! -)

Common attribute must be C.R.

- 1.) Lossless Decomposition
2.) Dependency should be preserved.

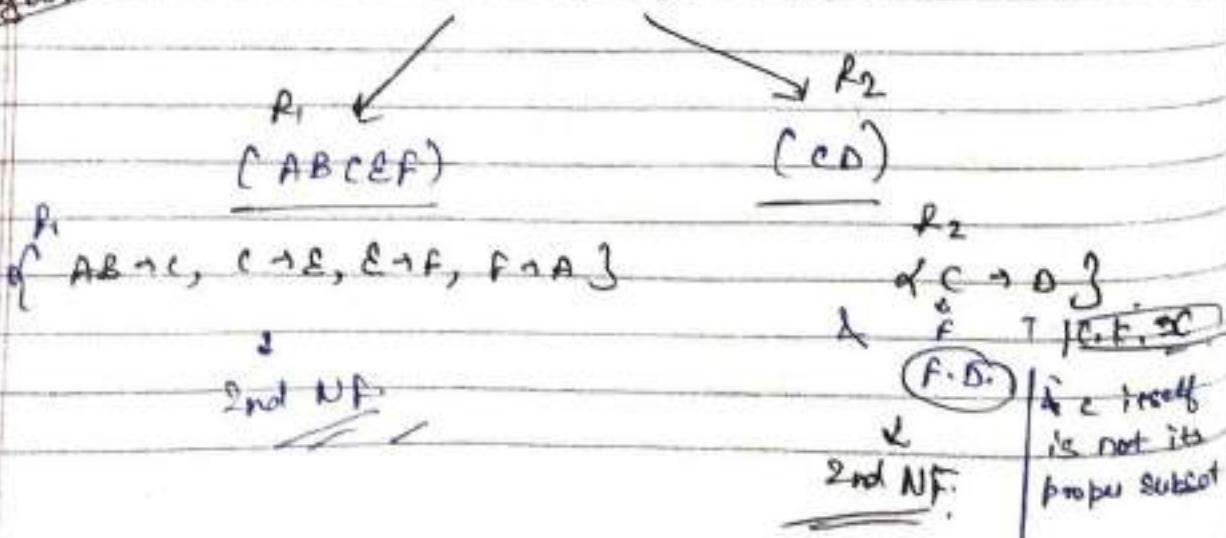
Now, w.r.t problem $\bar{C} \in \bar{R}$, ($C \rightarrow D$, 3rd NF)Play on \bar{C} , Relational Table will $\bar{C} \bar{I}$.Now,

Now, we have to make a common attr. b/w these 2 tables:-

Criteria # common! - can be C.R. of any R_1 & R_2 .

Ex. In R_2 ($C D$)
 $C \rightarrow D$ $C + = CD$ (all 2).
 C [C is C.R.].

Ex.

make 'C' common in both R_1 & R_2 .Now,
again $R(A B C D E F)$.

Now!

Check w. 3NF! →

$\left\{ \begin{array}{l} \text{LHS must be C.R.} \\ \text{RHS } \rightarrow \text{ be P.A.} \end{array} \right\} \rightarrow 3NF$

80,

R₁

(ABCDEF)

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

$\checkmark \quad \checkmark \quad \checkmark \quad \checkmark$

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

PA = {A, B, C, E, F}

R₂

(CD)

$CC \rightarrow D^3$

C.R. = {C}

PA = {C}

1.

R₁ is 3rd NF.

R₂ is also 3rd NF.

Now!

Check w. BCNF! →

[Cond'n → L.H.S. must be a C.R.]

R₁ {AB → C, C → E, E → F, F → A}

$\checkmark \quad \times \quad \times \quad \times$

not in BCNF form.

R₂ {C → D}

R₂ is BCNF form.

∴ There is Redundancy.

Now,

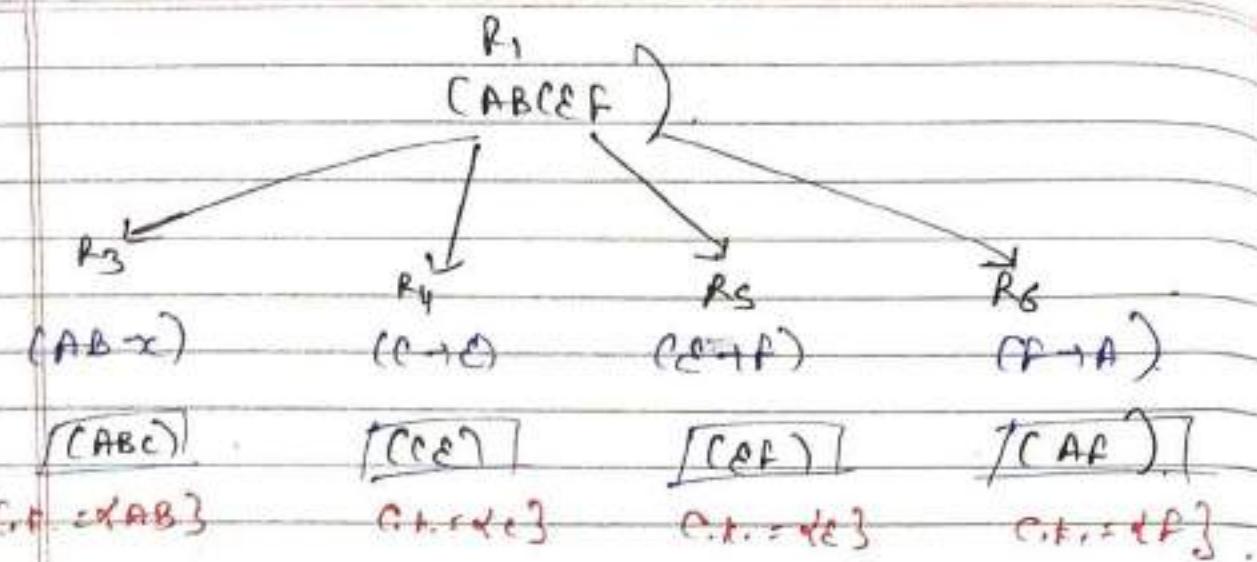
Again Decompose R₁.

(ABCDEF).

Redundancy - 0%.

(Civil problem on 2nd Dec, Total May check 1)

Normalisation's aim → To make redundancy 0%.



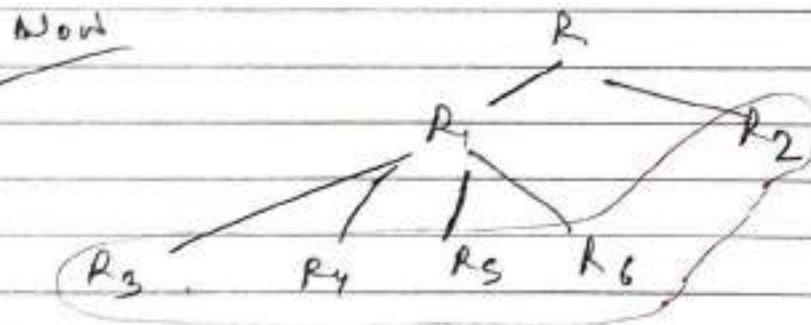
Now, these all 4 tables have C.R.

Q. 2,

Now all 4 are in BCNF now.

↳ (CE) -> F common & (F) <-> A

Work

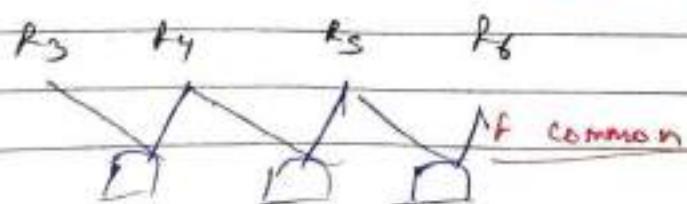


Now, in all these 5 tables,

Redundancy is 0%.

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

So, have to join R_3 & R_4 (both have common F)



→ 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 $R_3 \Delta R_4$, Let, $R_3 R_4$
(ABCE)

Now,

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

$R_3 R_4 R_5$ (ABCEF).

→ E is C.I.C. now

32-

Normalisation Questions: →

→ If a reln is in BCNF, then it is in 2NF also

Q: If a Reln R has 8 att. (ABCDEFGH)

$F = \emptyset$ ch $\rightarrow G$, $A \rightarrow BC$, $B \rightarrow CFH$, $E \rightarrow A$, $F \rightarrow EG$,

How many Candidate keys in R.

a) 3

b) 4

c) 5

d) 6

Soln: First, check on RNS, which att is absent.

D

∴ So, now it comes with every.

$N^+ = D$.

✓ $AD^+ = AD \cup BC \cup FG \cup E$

(all 8),

Now, check A on RNS,

so,

ED^+ also c.t.p.

Now, check B on RNS.

so,

FD^+ also a c.t.p.

Now, check C on RNS.

so,

BD^+ also a c.t.p.

Now, check D on RNS.

Now, completed.

so,

$\{AD, ED, FD, BD\} \nsubseteq C.P.$ ✓

33.

"One" Explained on Normalisation! →

Scheme is a structure of a Table.

→ So, Scheme (or) Table → same.

→ Non-trivial F.D. means in which.

$LHS \cap RHS = \emptyset \rightarrow$ 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 \rightarrow Courses ?

Ans:- We have to check that this table be in which form.

→ So, as always, start from higher form.

→ Check \rightarrow BCNF!

cond'n : LHS of every FD must be C.R. or S.R. or P.P.

and,

Roll.no be already given as a C.R.

So,

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

So,

Table be in BCNF form.

∴ also in 1st NF, 2nd NF & 3NF

Schema 2 : Registration (roll.no, Course Id, email)

Non-trivial F.D. of roll.no, Course Id \rightarrow email
email \rightarrow roll.no

Ans \rightarrow CR = { roll.no CourseId } .

PR = { roll.no, Co.Id } .

NRA = { email } .

→ Check BCNF! \rightarrow 1st FD. LHS is C.R.
but not in 2nd F.D.

So, not in BCNF form .

→ Check for 3NF! ~

Cond'n: LHS must be a PK or st. on PR
 OR -
 RHS must be a f. A.

Now, let FD be already valid,

1.

2nd FD: email \rightarrow roll no.

P.A.

f

T.

3.

(T)

It is in 3NF. ✓

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

Non-trivial FD: { roll no, Co-Id } \rightarrow marks, grade }
 marks \rightarrow grade . }

Ans: C. t. = { roll no, Co-Id } }

PA = { roll no, Co-Id }

NPA = { marks, grade } .

→ check & BCNF! ~

1st FD \rightarrow valid.

2nd FD \rightarrow not valid.

not in BCNF.

→ check & 3rd NF! ~

1st FD \rightarrow already valid

2nd FD \rightarrow marks \rightarrow grade.

F

F

∴ not in 3rd NF.

→ ↙

→ check \forall 2nd NF! →

cond'n:

LHS must be a proper subset of CK.
TAND]

LHS must NPA.

↳ for p.D. → i.e. not in 2nd NF.
if both cond'n are true.

→ 1st FD → already valid.

2nd FD → marks → grade
f t

(nf)

∴ not in partial dependency.

∴ go to in 2nd NF. \therefore

Scheme 4: Register (rollno, C-Id, Credit).

Non-trivial FD, { rollno, C-Id } → Credit }

C-Id → Credit }

As, 1 C.R. = { rollno (C-Id) }

PA = { rollno, C-Id } .

NPA = { credit } .

→ check \forall BCNF! →

1st FD → ✓

2nd FD → ✗

} \therefore not in BCNF.

→ check \forall 3NF! -

1st FD → ✓

2nd FD → C-Id → Credit
f f

(F)

\therefore not in 3rd NF.

→ Check \rightarrow 2NF? \rightarrow

1st FD \rightarrow \leftarrow

2nd FD \rightarrow C-Idl \rightarrow Credit
 \uparrow \uparrow
 (+ ?)

∴ It is Partial Dependency (P.D.)

So,

It is not in 2nd NF.

" Table is in 1st NF.
 (We already assume it)."

$X \longrightarrow Y$

(34) Cover & Equivalence of F.D. \rightarrow

$$\left\{ \begin{array}{ll} \text{if } X \text{ covers } Y & Y \subseteq X \\ \text{if } Y \text{ covers } X & X \subseteq Y \end{array} \right. \quad \text{if both are true, then, } [X = Y] \rightarrow \text{Equivalent.}$$

$$\text{Q: } X = \{A \cap B, B \rightarrow C\} \quad | \quad Y = \{A \cap B, B \rightarrow C, A \rightarrow C\}.$$

i) X covers Y : $\rightarrow (Y \subseteq X)$

(इसमें Y की FD. में LHS नहीं closure ले रही है ताकि X तक से है).

$$A^+ = ABC$$

$$B^+ = BC$$

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

∴ all 3 covers in these closure forms.

X covers Y.

i) Y covers X:

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

$$A^+ = AB$$

$$B^+ = BC$$

(X not closure)
(Y not closed)

∴ Y also covers X.

both symbols cancels each other & become equivalent.

$$X \neq Y \quad Y \neq X$$

$$X = Y$$

$$\Leftrightarrow$$

e.g. i) $X = \{ AB \rightarrow CD, B \rightarrow C, C \rightarrow D \}$.

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

X covers Y:

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

$$AB^+ = ABCD$$

$$C^+ = CD$$

∴ True

$\therefore Y$ covers $X \rightarrow ?$

$$X = K_{AB \rightarrow CD}, B \rightarrow C, C \rightarrow D \}$$

$$\underline{AB^+ = ABCD} \quad \checkmark \quad B^+ = B \quad \checkmark \quad \underline{C^+ = CD} \quad \checkmark \quad (\text{from } Y)$$

$\therefore Y$ not covers X .
It becomes false.

$$[X \neq Y] \quad \text{ok.}$$

$$[X \supseteq Y] \quad \leftarrow [Y \supseteq X].$$

\checkmark , true. \times , false. \checkmark .

35. Dependency Preserving Decompositon

Def'n: Let, any table $R(ABCD)$

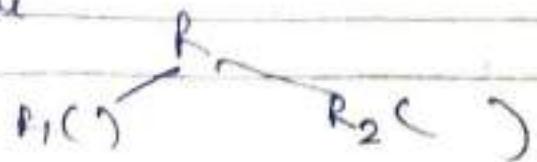
also same F.D., $\{FD\{A \rightarrow B, B \rightarrow C\}\}$

Now,

we find closure λ , then find C.K., λ
then we also find hidden dependencies like.

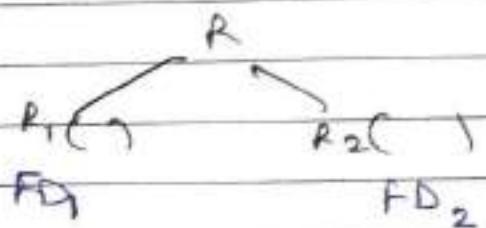
$FD^+ \nsubseteq A \rightarrow C \}$ by transition prop.

In normalisaⁿ, we do decompositⁿ,
then, we divide



↳ We distribute attr. of R_1 in R_1 & R_2
 union of attr. of $R_1 \rightarrow R_2$, must be
 equal to the attr. of R .

Now, f.d. also divides. like.



Now,

check! - Dependency should be preserved,
 (can't this f.d. lost or gain attr. while
 preserve other attr. like $A \rightarrow C$).

i.e.

$$\boxed{FD_1 \cup FD_2 = FD}$$

(original f.d.
 are equal)

Q: Let $R(ABCD)$

with f.d.

$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow B$ 3.

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

$R_1(AB)$	$R_2(BC)$	$R_3(BD)$
$A \rightarrow A$	$B \rightarrow C$ ✓	$B \rightarrow D$ ✓
$B \rightarrow B$	$C \rightarrow B$ ✓	$D \rightarrow B$ ✓
$A \rightarrow B$ ✓		
$B \rightarrow A$ ✗		
$B^* = BCD$		
(don't have A).		

union of all $R_1, R_2 \wedge R_3$ are equal to R . So, true. (Right decomposition)

Now,

→ Check Dependency preservation: →

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

Ques, Whatever dependency we can make from these 2, make them & write in table.

* We don't want trivial F.D. →

→ Trivial F.D.'s in which "intersec" (n) is not null.

i.e. $A \rightarrow A$ has $n \neq \emptyset$
Ex-1 (common is \neq).

Ex-2 $AB \rightarrow A$ has $n \neq \emptyset$
(common is \neq).

We don't want these bcs, trivial are by default, true. It isn't like this.

$A \rightarrow A$ is definitely false.

→ Only take non-trivial F.D.'s →
 $(n = \emptyset)$.

Now:

F.D.'s of $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow B$.

$$A^+ = ABCD$$

$$B^+ = BCD$$

$$C^+ = CBD$$

$$D^+ = DB$$

} So, acc. to these,
Select from the
~~non-trivial~~ non-trivial.

F.D. (which we finally
select).

from table. =

Now

F.D. which we get from Table: →

$$A \rightarrow B \quad \checkmark$$

$$B \rightarrow C \quad \checkmark$$

$$C \rightarrow B$$

$$B \rightarrow D$$

$$D \rightarrow B. \quad \checkmark$$

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

So, F.D.'s of $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow B$.

Now,

$C \rightarrow D$ is not direct.

so, now take ' C ' closure.

$$C^+ = CBD$$

$\Rightarrow C \rightarrow D$ also preserved.

∴ all F.D. are preserved.

True. ✓

Q. 2:

Dependency Preserving Decomposition Example

Let, P(ABCD)

with F.D.

$$LAB \rightarrow CD, D \rightarrow A \}$$

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

$R_1(AB)$	$R_2(BCD)$
$A \rightarrow B$ ✗	$B \rightarrow CD$ ✗
$D \rightarrow A$ ✓	$C \rightarrow BD$ ✗
	$D \rightarrow CB$ ✗
	$BC \rightarrow D$ ✗
attributes of $(R_1 \cup R_2)$ make R .	
Now,	
F.O: $\{AB \rightarrow CD, D \rightarrow A\}$.	

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

$$BC^+ = BC$$

$$BD^+ = BDA$$

$$CD^+ = CDA$$

Now,

F.D. we get from table: 7

$$\{ \begin{array}{l} D \rightarrow A \\ BD \rightarrow C \end{array} \}$$

Now, check w.r.t original F.D.:

$$\{ AB \rightarrow CD, D \rightarrow A \} \quad \text{from table}$$

$$AB^+ = AB$$

$$D^+ = DA$$

We 'can't' get

$AB \rightarrow CD$, from our decomposed F.D.

Hence,

Dependency preserved not

Why

No

34

Joins & Its Types : →

1. Join: When we have to join 2 or more tables, to get result.

C. No	E-name	Add.	Dep. No	Name	L-name
1	Ram	Delhi	D ₁	HR	1
2	Mann	Chd.	D ₂	IT	2
3	Rani	Chd.	D ₃	MRKT	4
4	Amrit	Delhi	D ₄	Finance	5
5	Nitin	Noida			

'Employee'



'Department'

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

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

Now:-

Q. Find Ename of Emp who is working in HR.
Here, HR is in department Table

Ename is in Employee Table.

Ans:-

Here, we need Joins.

Note:- There must be same common attr. in Tables to use Join. Here, C-no is common.

→ Join is Cross product

+

Select Statement (Condition)

→ Its Types:-

→ Cross Join (or) Cross product.

~~→~~ Natural Join

~~→~~ Conditional Join

→ Equi " "

→ Self " "

→ Outer " "

→ Inner " "

→ Left Outer " "

→ Right " "

→ Full " "

(38)

" Natural Join " ! →

→ Join = Cross product + Condi?

Employee			Department		
E-No	E-name	Add.	Dep No	Name	E-no
1	Ram	Delhi	D ₁	HR	1
2	Manu	Chd.	D ₂	IT	2
3	Ravi	Chd.	D ₃	MKT	3
4	Ankit	Delhi			

→ first find! - Table name

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

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

Also do it by "Natural Join".

Common Attr - 'E-No.'

Ques,

Whenever we have to equalize (笛卡尔积) the values of the common Attr, then, we always use NATURAL JOIN.

Now, Write Query ! ~

Select Ename from Emp, Dept Where

Emp. Eno = Dept. Eno ;

Output :- Ram (Vasim, Amrit)

'E-No' - must be same in both the tables.

i.e. ($\frac{E\text{-No.}}{\text{Emp. No.}}$) ✗

Emp		Dept	
E-no	Ename	Dept No.	D-no
1	Ram	D ₁	1
	"	D ₂	2
	v	D ₃	4
2	Vasim	D ₁	1
	"	D ₂	2
	v	D ₃	4
3	Ram	D ₁	1
	"	D ₂	2
	"	D ₃	4
4	Amrit	D ₁	1
	"	D ₂	2
	"	D ₃	4

so, finally we get 3 rows

Dno	Ename	DeptNo	Eno
1	Ram	D1	1
2	Varun	D2	2
4	Amit	D3	4

↗ Ram
 { Varun
 Amit } → Ans.
 = =

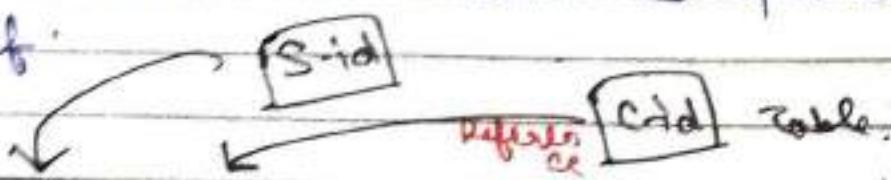
Now, how to write Actual Query? →

→ Select E-name from Emp Natural join Dept;

39)

Self Join! →

→ in which the Table is joined to itself.



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

student course

Study

→ find student id who is enrolled at Dept D3

→ Now, we do Self Join.

- SQL query always start from - "from" (equal)

Query: →

Select (from Study as T₁, study as T₂;

T₂ → means Cross product.

- + So, here we make same Table 2 times
1. name as T₁ & T₂.

S ₁	C ₁	2016
S ₂	C ₂	2017
S ₁	C ₂	2017

T₂ (m)

S-id	C-id	Since
S ₁	C ₁	2016
S ₂	C ₂	2017
S ₁	C ₂	2017

(n)

T₁

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

Condition:

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

$\leftarrow \rightarrow$ different
(not equal to).

Q1

Page

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 T₁ & T₂ both have S-id.

So,

we can use any T₁.Sid or T₂.Sid.

Aus - S₁

T₂.Sid.

Q10

EQUI - Join :- \rightarrow (=)

(कोई एक अंतर के बीच (=) तभी सही है।)

E-No	Surname	Add.
1	Ram	Delhi
2	Mohan	Chd.
3	Rani	Chd.
4	Anurit	Delhi

Employee

Dep No.	Loco	E no
D ₁	Delhi	1
D ₂	Pune	2
D ₃	Patna	4

Department
Dept.

$$f \Delta f = f$$

$$f \Delta T = f$$

$$T \Delta T = T$$

$$f \Delta f = f$$

$$f \Delta T = T$$

$$T \Delta T = T$$

8 अप्रैल

Q: find the Emp name who worked in a department having locaⁿ same as their address?

Ans: By just seeing the 2 Table: →
[Ram]

(Q) Query: →

Select Sname from Emp, Dept where

Emp. E-no = Dept. E-no , # common

and

Emp. Addl = Dept. locaⁿ ; ;

Emp.

Dept.

			D ₁	Delhi	1
Delhi	1	Ram	D ₁	Delhi	1
"	1	"	D ₂	Pune	2
"	1	"	D ₃	Patna	4
Chennai	2	Harish	D ₁	Delhi	1
"	2	"	D ₂	Pune	2
"	2	"	D ₃	Patna	4
"	3	Ram	D ₁	Delhi	1
"	3	"	D ₂	Pune	2
"	3	"	D ₃	Patna	4
Delhi	4	Amrit	D ₁	Delhi	1
"	4	"	D ₂	Pune	2
"	4	"	D ₃	Patna	4

Note:-

Q1 Table का common. add. की तो है

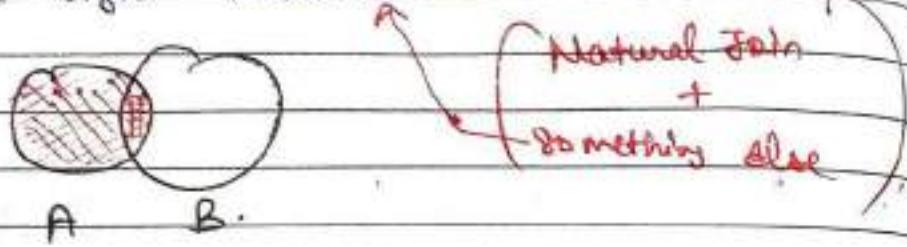
→ नहीं (=) नहीं है लेकिन फिर भी 2 एक

अब यह क्या करता है ?

(41)

Left Outer Join ! →

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



Emp			Dept.		
Empno	Sname	Deptno	Deptno	Dname	Loc.
E ₁	Varun	D ₁	D ₁	IT	Delhi
E ₂	Amit	D ₂	D ₂	HR	Hyd.
E ₃	Ravi	D ₁	D ₃	Finance	Pune
E ₄	Nitin	-	-	-	-

Query!

Select empno, sname, dname, loc from
emp left outer join, dept on

left

(emp.deptno = dept.dept no.)

→ cond'n of

Natural Join

(common attr. dno;
(D1, D2, D3))

Relational Database always shows output in Table form.

SM Date _____
Logo _____

89-

Output Table:

Emp-no	E-name	D-name	Loc^n	→ (Left की तरफ सही Row 3rd की तरफ)
E ₁	Umar	IT	Delhi	
E ₂	Amit	HR	Hyd.	
E ₃	Ravi	IT	Delhi	
E ₄	Nitin	-	-	

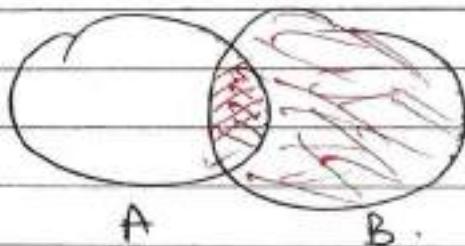
X

X

Q2

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	Umar	D ₁ C
E ₂	Amit	D ₂ C
E ₃	Ravi	D ₃ C

(Dept)

Dept-No	D-name	Loc^n
D ₁	IT	Delhi
D ₂	HR	Hyd.
D ₃	Finance	Hyd
D ₄	Testing	Ne�de

Query:-

Select emp-no, E-name, d-name, loc from emp Right Outer Join, dept on cemp.dept-no = dept-dept-no.)

natural join.

→ Outer Table : →

Emp-no	L-name	R-name	Loc
E ₁	Umar	IT	Delhi
E ₂	Ankit	HR	Hyd.
E ₃	Pani	Finance	Lune.
-	-	Testing	Maida.

Right
Table
All
Rows.

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

$$(\text{Left O.J.}) \cup (\text{Right O.J.})$$

Y X

43) 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 in Relational Algebra.

("collection" of mathematical Expressions).

Relational algebra → SQL → No SQL

In today's time,

"operators"

Basic operator

- projection (π)
- Selection (σ)
- Cross product (\times)
- Union (\cup)
- Rename (ρ)
- Set Difference ($-$)

Derived operators

- Join (\bowtie)
 - intersect (\cap)
- $(x \bowtie y) = y - (x - y)$
- Division ($/, \div$)

∴ If we understand Relational Algebra very clearly, then we don't face any problem in understanding SQL.

(44)

Projection in Relational algebra : →

Projection (π): →

π func. is to retrieve the data.

Roll no.	Name	Age
1	A	26
2	B	21
3	A	19
4	C	18

π (Student)

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

Q). $\pi_{\text{RollNo.}}(\text{Student})$.

Roll no.
1
2
3

Ans: $\pi_{\text{RollNo., Name}}(\text{Student})$.

We fetch the 2 colⁿ (rollno & name) from the student (table).

Output →

RollNo	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 select & before this we use other operators.

45

Selection in Relational algebra → σ (Sigma).

RollNo	Name	Age
1	A	20
2	B	21
3	A	19

→ It works on tuples (rows).

1. First, It found rows.

π operator.

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

$\pi_{\text{rollno} = '2'} (\text{student})$.

$\rightarrow [2, \text{B}, 2]$

Final.

$\pi_{\text{name}} (\sigma_{\text{rollno} = '2'} (\text{student}))$.

$\rightarrow [8]$

π always last it operate \Rightarrow all.

* We can also find, 2 at a time.

Ex:-

$\pi_{\text{name}, \text{age}} (\sigma_{\text{rollno} = '2'} (\text{student})) \rightarrow [6, 21]$

col.
 $=$

Note:- Project (π) always works on Columns.

Selection (σ) always works on Rows. (tuple).

(46)

Cross / Cartesian product in Relational algebra

A	B	C
1	2	3
2	1	4

(P₁)

C	D	E
3	4	5
2	1	2

(P₂)

To Train, we must have to Cross product.

→ $3+3 = 6 \quad (\text{m+n}),$

$2 \times 2 = 4 \quad (\text{m} \times \text{n}).$

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

→ $(R_1 \times R_2)$

(A)

(B)

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

Q7

Set Difference in R. A →

$$(A - B) = A \text{ but not } B.$$

$$= A \cap B'$$

Ex:

1, 2, 3

S₁

3, 4

S₂

$$S_1 - S_2 = 1, 2$$

$$S_2 - S_1 = 4$$

→ It is not Commutative! →

i.e.

$$\boxed{A - B \neq B - A}$$

Cond'n:-

- 1.) No. of columns must be same in no.
- 2.) Domain of every column must be same.
(data of same type). (Ex- numeric). I+A X.

Rollno	Name
1	A
2	B
3	C

(Student)

Emp-no	Name
1	E
2	A

(Employee)

→ (Student) - (Employee)

(: A is staff)

Rollno	Name
2	B
3	C

→ By defining, what
Table in left column
will do

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

→ (Student - Employee).

How to write?

(Tname (student) - Tname (Employee)).

Output:-

Name
B
C

((C, A, B, C) = (E, A))

48

Union Oper. in R.A. : →

a) Same as in Sect. (ii).

(1, 2, 3)

S₁

(3, 4, 5)

S₂

(1, 2, 3, 4, 5)

b)

Ex:-

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

(Student)

(Employee)

Cond'n :-

1.) No. of col^m must be same in r.2.) Domain of every col^m must be same.

(student) ∪ (employee)

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

Table :-

(left side "and rt")

Long trick!

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

Name.	
A	
B	
C	
E	

\rightarrow (Student also & who
is Employee also
or both).

Note: 1

All no.	Name	Name	Dept No.
1	A	E	2
2	B	A	1
3	C		

Numeric

Alphabet

(π_{name} , by not same domain,
e.g., Default order pick that).

So, here, not Union "operator" applies.
(NULL).

Q3)

Division Operator in R.A. : →

→ "Div" operator is actually a Derived
(We derived them, from normal operators).

l, \div

$(\times, -, \pi, \tau)$.

→ We use these

Query: Retrieve list of students who
enrolled in all course.

2) Every ($\forall t$) all t_{std} of Dini's method.

Std	Cid
S ₁	C ₁
S ₂	C ₁
S ₁	C ₂
S ₃	C ₂

'Enrolled'

Cid
C ₁
C ₂

'Course'

(*) Note of Dini's Method →

$$A(x, y) / B(y) \leftarrow \text{it results 'x' values}$$

for that there should be tuple $\langle x, y \rangle$
for every y value of Relⁿ B.

Here,

$$[\in (Std, Cid) / C(Cid). = S_1] . . .$$

i.e.,

S_1 enrolled in every course (C₁ & C₂).

How it happens?

We let, all students are enrolled in
every course.

(for this, we do the Cross product),

$$\Rightarrow (\pi_{cid}(\text{Enrolled})) \times (\pi_{cid}(\text{Course})).$$

S₁ x C₁

S₂ C₂

S₃

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

(Enrolled).

Note:

We subtract actual Scenario i.e., (Actual Table) from this cross product. \rightarrow we get

S_2, S_3
(who didn't enroll in all courses)

* Now, final \rightarrow (Query).

$$\pi_{std}(\text{Enrolled}) - (\pi_{std}((\pi_{std}(\text{Enrolled}) \times \pi_{std}(\text{course})) - \text{Enrolled}))$$

$$\frac{S_1}{S_2} - \frac{S_2}{S_3}$$

$$S_5$$

$$2 S_1 \downarrow .$$

\times

so Tuple Calculus in DBMS \rightarrow .

Relational Calculus (RC)

↓
TRC
(tuple or rows)

↓
DRC
(domain or column)
Att.

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

Q) 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(V),
AND(A), NOT(¬).
- Atomic func's:- (We use one variable here, fix -
(Supply Ids. from a supply table)) → only 1 cond'n.
- It also uses quantifiers!
- $\exists t \in r(P(t))$: "there exists" a tuple
in r in rule" or such that predicate
 $P(t)$ is true.
- $\forall t \in r(P(t))$: $P(t)$ is true "for all"
tuples in rule" or .

* Unsafe expression: \rightarrow (U.E.)

Supplier (Table).

- $\{ s.name \mid \neg \text{Supplier}(s) \}$
- (not sign)

Here, It means

that, all the supplier name who are not
in the Supplier Table.

Qn,
here, ans is ∞ . (∞ loop \Rightarrow other attempt).

"unsafe Expr".

(These are not in the Rel'el algebra)

- Both TRC & RA have same expressive power.

unsafe Expr \leftrightarrow $\{ \text{U.E.} (x) \}$.
all stat
with D.

(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 extra func.

* Examples: -

Q-1. Write a query in SQL to display Sname of Suppliers.

Q-2. Write a query in SQL to display Pname of parts whose color is Red?

- Q-3. Write a query in SQL to display
 SID of suppliers whose name is
 'Varun' & address is 'chandigarh'.
- Q-4. Write a query to SQL to display
 SID of suppliers who supplied some
 parts?
- Q-5. Write a query in SQL to display
Name of suppliers who supplied
 some parts?
- Q-6. Write a query in SQL to display
Name of suppliers who supplied
 some red color parts?

(1)

$$\{ \text{S.Sname} | \text{Supplier}(\text{s}) \}$$

S-1
 (examining
 attr 1)

Supplier
 Table

(2)

$$\{ \text{P.Pname} | \text{Parts}(\text{p}) \wedge \text{p.color} = \text{'red'} \}$$

(3)

$$\{ \text{S.SID} | \text{Supplier}(\text{s}) \wedge \text{s.name} = \text{'Varun'} \wedge \\ \text{s.add} = \text{'chandigarh'} \}$$

(4)

Here, it extract ans. from the Catalog Table

$$\{ \text{C.SID} | \text{Catalog}(\text{c}) \}$$

Sid & Catalog Table in SQL Table (10%).
But, Sname Only in Supplier Catalog (10%)

10%

(5)

Here, we need 2 Table

Supplier
Catalog

&

final one from supplier Table.
(There exists).

{ S.Sname | Supplier (S) } \cap {c | Catalog (C)}
 \wedge S.Sid = C.Sid }

Here,

{ S \rightarrow free variable
c \rightarrow bounded variable }

(Forwards STAR WITH
+ AND + ONCE).

(6)

Here, we need 3 Table.

Supplier
Catalog
parts

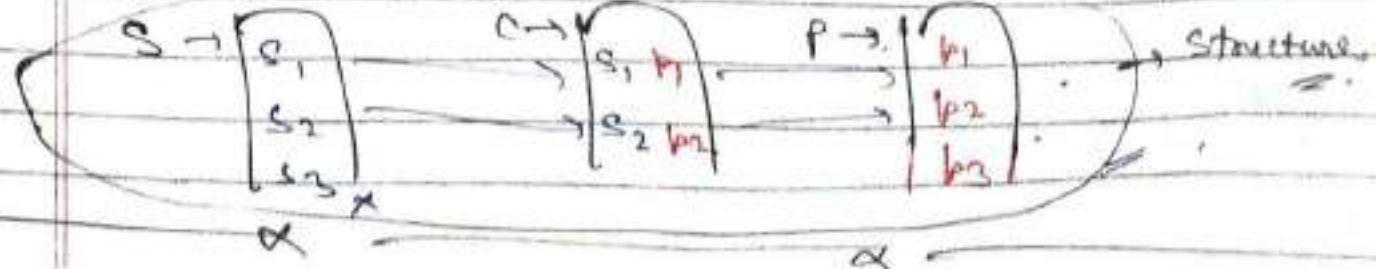
But,

one from supplier Table.

{ S.Sname | Supplier (S) } \cap {c | Catalog (C)}
 \cap {p | parts (P)} \wedge S.Sid = C.Sid \wedge
C.pid = p.pid \wedge p.color = 'red' }

S variable for supplier Table.
c \rightarrow "catalog"
p \rightarrow "parts"

S \rightarrow free variable
p \rightarrow bounded "



SI- Intro to SQL → (structured query language)

(1970)

User

language

Table or Rel.

→ EF code & the theory of Database.

store & fetching of data.

Relational Model:

by the relational algebra
Wf of

TR: (Tuple Relational calculus)

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

→ Before,

SEQUEL → Simple English Query language
then,

SQL → Structured Query lang.

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

→ Now, No SQL is come into market.

100% data → Structured

90% data → Unstructured (spart, noSQL)

④ Properties:-

5/1

Date: _____
Page: _____

105

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 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
Δ how to do.
↳ (There is procedure that how to do)
Later on,
[PL-SQL] → procedural language.

- 3) DDL, DML, DCL, TCEL
These are diff. commands in SQL to create, insert, fetch, control data. etc.
- 4) Keys & constraints. (Ex. P.K., F.K., C.constraint)
↳ (primary key (P.K.) constraint,
f.k.
Not, NULL, default)

aggregate → ~~other~~, family, ~~and~~ ~~or's~~, ~~where~~
Date _____
Page _____
Q1

exist / not exist.

5.) operators (like, between, ~~In, Not In~~,
(conditional)).

→ We use these operators internally to
use Query.

Ex:- IRCTC : - we query on this app,
by APIs (Application programming interface).

Train no., slot no ;
IRCTC based on structured Data.

(Train की Info. Tables in form of डी
शेव डिटी एच ई).

6.) clauses (distinct, order by, group by,
from ^{also} having).

We ^{mostly} use this in query.

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

8.) Joins & Nested Query : -

9.) PL SQL (Triggers, func, cursor, procedure)

same as in C lang.

④ what to do! -

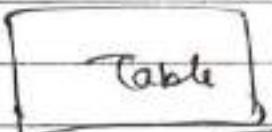
In libraries of oracle or like SQL
Server, it is already defined that what
select do, & what other commands 'do'.
so, we don't need to give procedure that
how to do.

(52)

All Types of SQL Commands :-

- * DDL Commands :- (Data Definition language).
 - Deals with Schema (Structure | Table).

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



- * DML Commands :- (Data manipulation 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 permission)
- Remove (take back those permissions).

- * TCL Commands :- (Transaction Control lang.)
 - Mainly used in Transaction.

Part - Shopping, Bill payment - online

- Commit (~~if Transaction~~ to commit (done), then save ^{data})
- Roll back. (~~if Transaction~~ fails).
- Save point. (~~Ex:~~ we can save after 20-30% of transacⁿ).

Ex:- Like in Downloading & Buffering,

We have save points. Ex: If downloads fails on 90%, then if it starts from '0' again as from 90%. This is Save point.

Motivation: To execute these commands, we can use Oracle, MySQL, 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.

Create Table in SQL with

execution →

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

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

→ Create table < table-name >
(
Column 1 name datatype,
Column 2 name datatype,
Column 3 name datatype
);
desc table-name ;

|| no space

Ex:- abc-xyz

|| describe

Ex:-

Create table emp
(
id int,
name varchar(20),
Salary number (10)
);
desc emp;

Ex:-
(2nd fixed)

→ fixed type of datatype
variable -||-
" (can be changed)

54.

Alter Command (DDL) in SQL :→

→ Use → (Add or drop or change)
Delete → (Delete)

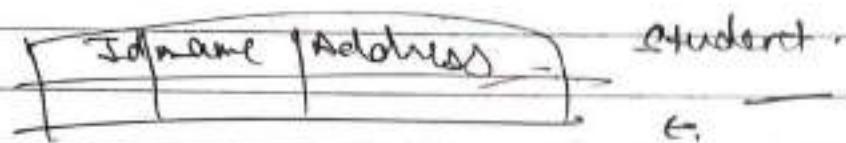
Ex:-

Student

Int	ID	name	varchar(20)

→ already 2nd

- Add or remove columns.
- modify datatype.
- Modify datatype length.
- Add constraints.
- Remove constraints.
- Rename column / table.



② 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 emp;

alter table emp drop column address;
 " " " modify id varchar(30);

" " " rename column id to seg-no;

" " " rename to emp1;

alter table emp1 add primary key (seg-no);

(CS) DDL Alter & Update : →

② Alter: → only change in structure,
 (DDL) & not in data.

Ex: Emp

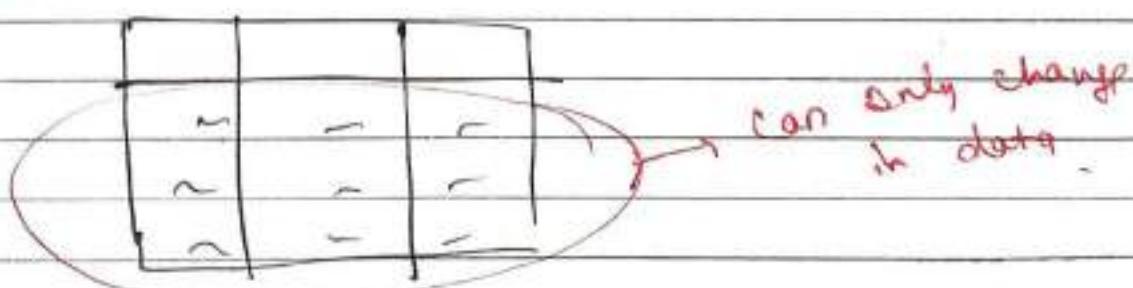
ID	Name	Salary	Email
1	A	10k	1
2	B	20k	1
3	C	30k	1

→ can only change in this store.

- Int → varchar
- name varchar (10)
- 30 → Eid.
- Emp → Emp_detail

colⁿ
 (name change)
 (table name -)

* update: → can only change in data,
 (DML) & not in structure.



→ update Emp
 Set Salary = Salary * 2 ;] → for all student

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

S6

D/B

Delete, Drop & Truncate:(1) Delete: →

(DML).

Syntax: →

Delete from table name

Ex. [Delete from Student], → All rows deleted.

Desc Student;

Student:

ID	name
1	A
2	B
3	C

ID	name
1	A
2	B
3	C

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

Ex. [Delete from Student
Where id = 1;]

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

→ Log! → Every file completely delete कर देते हैं, Computer ने temporary file बनाते हैं, to restore data, get log and use it.

Ex. on delete, our data comes into recycle bin. (as it log file).

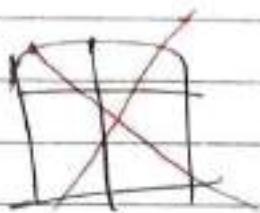
Rollback',

* possible, but it creates log.

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

Drop table student;
Desc student;

→ no object found.
(all deleted).



Rollback',

* (not possible).

* 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) definition.

Ex: In gmail: → (2 account IDs can't be same).

anurag@gmail.com × (not present)?

anurag123@gmail.com ✓ (present). ✓

1) Cond' \rightarrow that new mail-id must be unique.

2) Cond' \rightarrow length of password must be 6 digit, Capital letter etc.

Note:- We apply conditions on Attributes (column).

and the data which fulfill these cond', only comes into the column.

1.) Unique: \rightarrow ex: Mobile no (no duplicate can exist in that m.no colⁿ).

2.) Not Null: *

(we can't skip that colⁿ),

i.e., * - mandatory. (value can't be null)

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

4.) Check:

age int
 $x > 10$

check (age > 18).

i.e., $\rightarrow 10 \times$ (ii)

User Id (1001, 1002, ...)

Domain - fixed.

1001

5). Foreign key:

6). Default:

Salary isn't default 10k.

If we don't fill anything, they default
Salary colⁿ take value \rightarrow 10k

7) We can use multiple constraint in a colⁿ also.

Q8)

SQl Queries & Sub-Queries \rightarrow
(from Beginner to end).

Emp.

Q8:	E_id	E-name	Dept	Salary
	1	Ram	HR	10k
	2	Ankit	MRKT	20k
	3	Ravi	HR	30k
	4	Nitin	MRKT	40k
	5	Harun	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 maxⁿ Salary.

Note:- Aggregate func ! \rightarrow for max, count, average,

$<>$ → not equal to . , = , ≠

SM Date _____
Page _____

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

Output → 50000

(ii) 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 → Name]

→ Inner query execute first 1 time

Outer	Inner
Ex -	10k = Sof. ✗
	20k = Sof. ✗
	30k = Sof. ✗
	Sof. = Sof. ✓

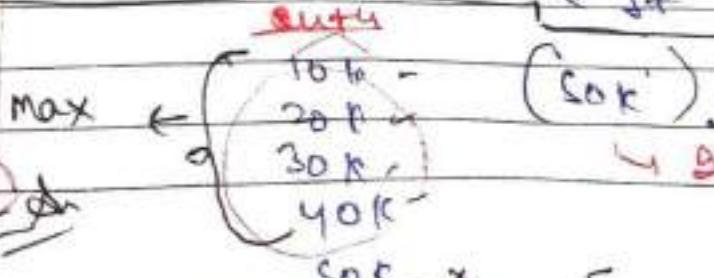
(53)

(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?

(v)

(Logic) Highest Salary - remove 1
Rest Salary - find highest



It is 2nd highest in table

40k ✗

(iii) Query:-

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

Task 1 Sol.

(iv). $\{$ = , when we have to compare only }
with 1 value.

$\{$ in , when we have to compare with)
multiple values.

Ex:-)

$$2 = 2$$



$\begin{cases} 2 = 1, 3, 2, 4, 5 \\ 2 \text{ in } (1, 3, 2, 4, 5) \end{cases}$ (Wrong way) (True)

Query:- only sald name in (A)

Select E-name from Emp where

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

Output Statement .

x

x

(60)

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

Output:-

HR	2
M.R.K.T.	2
I.T.	1

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

A.I.M.T.O
C.S.E - 100

Note:-1 Here we use a special func.

group by } Change
2
(form grp of dept).

HR	2
HR	2
MRKT	2
MRKT	2
IT	1

Note:-2 (and) of group by → wt att. of IT
in group by in SQL use att at &
similar att in select _____ in SQL it will work

If,

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

* 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

Q61. (iii) Write a Query to display all the dept names where no. of Emps are less than 2.

HR ?

MKT ?

IT IT sh.

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

Query:-

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

Output → IT.

Q7. If they ask of Employee name in this query, then → (Query → output)

Query →

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

↳ Warren.

* HR IT

* MKT

* IT sh.

(62)

(iii) 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, ... d.



Steps

HR	HR - 30,000
HR	HR - 30,000
MRKT	MRKT - 40,000
MKT	IT - 50,000
IT	IT - 50,000

30k
40k
50k

Outer

10k x
20k x
30k x
40k y

30k
40k
50k

50k

6.3) 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	Aranyak	Chd.

Project

E_id	P_id	P-name	Locn
1	P ₁	IOT	Bangalore
2	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
4	Aranyak	Chd.

Chd ✓

Delhi ✓

Pune ✓

Bangalore ✗

Chd. ✓

~~Nested Query or Query (Query)~~

37

Now, NOT IN :- (rows not included)

Query:-

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

Output:-

4	Robin	Bangalore
---	-------	-----------

Chd x
Delhi x
Pune x
Bangalore ✓
Delhi x

Q64 - Use of IN & Not IN in Subquery :-
(Same 2 tables as 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 → 3rd at 342

Means first write main query & then write ORA

Inner Query in Nested Query - runs 1 time.

SM
Date: _____
Page: _____

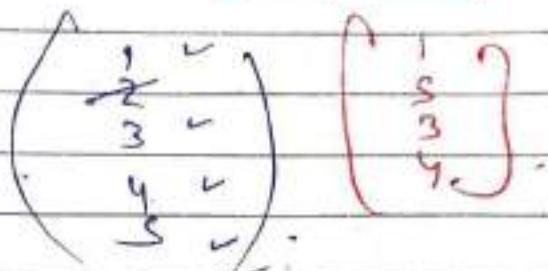
123

Query:-

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

Output:- Rani
Nitin
Robin
Anny.

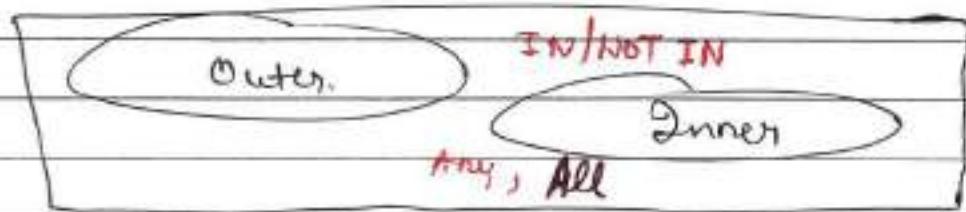
Inner Query.



(65)

Exist & NOT Exist Subqueries :-

→ We use these in Correlated Nested Query.



⇒ In nested query, - we use IN / NOT IN.

In correlated 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 Correlated Nested Query: → the one row of Outer
Query is compared with all rows of Inner Query.
ie, Top to Down Approach.

Null → means value is not available
Empty → (S.M)
→

A) Query:

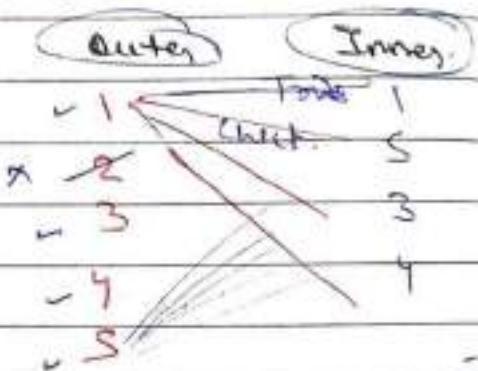
Select * from Emp where S_id

Exists (Select S_id from Project where Emp.S_id = project.E_id);

Note: Exist / not Exist gives True or False.

Output:

1	Ram	Chd.
3	Nitin	Pune
4	Robin	Bang.
5	Ananya	Chd.



66.

Aggregate Funcs in SQL: →

max, min, Count, Avg, Sum.

(Total no. of values in a Table).

Emp

S_id	E_name	Dept	Salary
1	Ram	HR	10k
2	Ankit	MR&IT	20k
3	Ran	HR	30k
4	Nitin	MR&IT	30k
5	Varun	IT	50k
6	Sandy	Testing	Null.

* Query for Max. Salary: -

→ Select max(Salary) from Emp;

Output → 50k

of

→ If SQL repeats 2 times, then it also gives output 2 times.

* 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 distinct).

* SUM:-

→ Select Sum (Salary) from Emp;
 [Output → 140k]. (sum of all salary)

→ Select Distinct (Sum (Salary)) from Emp;
 [Output → 110k] (30k repeats)

(*) AVG :-

$$\text{Avg (Salary)} = \frac{\text{Sum (Salary)}}{\text{Count (Salary)}}$$

$$= \frac{140k}{5}$$

$$= 28k$$

Ans.

→ Select Avg (Salary) from Emp;
 Output: → 28k.

→ Select Distinct (Avg (Salary)) from Emp;
 Output: → 27,500.

$$\begin{aligned} \text{Distinct (Avg (Salary))} &= \frac{\text{Distinct (Sum (Salary))}}{\text{Distinct (Count (Salary))}} \\ &= \frac{110k}{4} \\ &= 27,500 \end{aligned}$$

(*) Correlated Subquery in SQL: →
 (Synchronized Query).

- It is a subquery that uses value 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	L-id
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 Eid एवं उसी Employee detail के लिए।

→ Query:-

Select * from Emp where exists (Select * from Dept where Dept.Did = Emp.Eid);

Output:-

Eid	Name	Address
1	A	Delhi
2	B	Pune
3	D	Chd.
4	B	Delhi

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

68

DB Joins, Nested Subquery & Correlated Subquery:-

→ Nested → Bottom up. (Stage 2 at Stage 1)

→ Correlated → Top down Approach (Stage 1 at Stage 2)

→ Joins → Cross product + Cond.

SM

Emp	
Eid	Name
1	A
2	B
3	C
4	D
5	E

Dept.		
Dept No	Name	Eid
D ₁	IT	1
D ₂	HR	2
D ₃	MKT	3

Q1. Detail of all emp who works in any dept.

(Here, we need both 2 Tables)

→ Nested Subquery →

Select * from Emp where Eid in
(Select eid from Dept);

Output ↗

1	A
2	B
3	C

→ Correlated Subquery: →

Select * from emp where exists
(Select eid from dept where
emp.eid = dept.eid);

"id is common in both table"

Output ↗

1	A
2	B
3	C

True

True

False

True

True

False

False

- 1st Table $\rightarrow m$ rows
- 2nd Table $\rightarrow n$ rows

then, Total Comparison $\rightarrow m \times n$

\rightarrow JOINS : \rightarrow

Cross product + Condⁱn

It creates a new Table with $(m \times n)$ rows.

then,

check [Condⁱ \rightarrow emp_id = dept_id]

Notes

Joins is faster than Correlated Subquery
 bcz it made a table of rows $(m \times n)$
at once. While in Correlated, 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
 $(m \times n)$ power).

Ques

Find N^{th} Highest Salary using SQL : \rightarrow

Emp.

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

→ Highest Salary →

Select max (Salary) from Emp;
Output a sort.

→ 2nd Highest Salary →
(Correlated Query).

→ Select Max (Salary) from Emp where
Salary Not In (Select Max (Salary) from
Emp);
Output - 4 OR

Now

How to recognise Correlated Subquery?

जबकि Outer Query की ओर Inner

Value देने वाले use करती है।

→ How to find Nth Highest Salary? →

Query:-

[Best method learn it]

★ Select id, Salary from Emp e, where
 $N-1 = (\text{Select Count(Distinct Salary)} \text{ from Emp } e_2 \text{ where } e_2.\text{Salary} > e.\text{Salary})$;

here

ii, Correlated Subquery

we make 2 slice of Emp is, E, & E₂
 (Dumps)

Ex- on 1st case :-

E ₁	E ₂
30	30
1 Salary	1 Salary
2 10k	2 10k \Rightarrow count = 1 (false)
3 20k	3 20k \Rightarrow count = 1 (true)
4 30k	4 30k \Rightarrow count = 2
5 40k	5 40k \Rightarrow count = 3
6 50k	6 50k \Rightarrow count = 4

\Rightarrow why we make 2 (E_1 & E_2): -

bcz Employee use use 2 times. (E_1 , E_2)

If we don't create E_1 & E_2 , then
 E_2 . Salary \geq E_1 . Salary

\hookrightarrow It means, Employee Salary \geq Employee Salary.

\hookrightarrow

E_1 & E_2

\hookrightarrow no mean, X.

Ex:-

10k \geq 20k f

20k \geq 20 f

30k \geq 20 f

40k \geq 20 count = 1

50k \geq 20 count = 2

50k \geq 20 count = 3

} count = 3

Ex-2 Let we have to find 4th highest, then

$$N-1 \Rightarrow 4-1 = 3$$

&

Select. Id. Salary

$$3 =$$

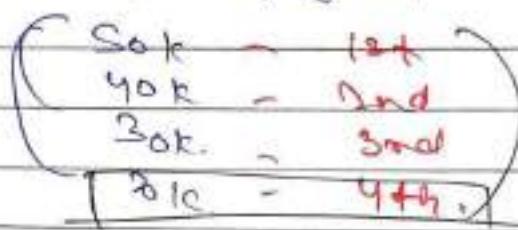
When we get 3 as Count from this inner query, then, our output comes.

And, we get, Count = 3
 from the 2nd row. b/c,
 from 1st row we get Count = 4.
 Hence, Ans is data of 2nd Row.

Output ->

ID	Salary
2	20k

20k is our 4th highest Salary.



Ex! for 3rd highest Salary!

$$N=3, \quad N-1 = 2$$

Hence,

for which row we get Count = 2,
 that row will be our output.

$$10k > 30 \quad f$$

$$20k > 30 \quad f$$

$$20k > 30 \quad }$$

$$30k > 30 \quad f$$

$$40k > 30 \quad T, \text{ count} = 1$$

$$50k > 30, T, \text{ count} = 2 \quad }$$

4th Row is our output

Output: 4, 30k. &

30k is our 3rd highest salary. &

(20.) Q. 3 Imp. Questions on SQL basic Concepts.

Q. 3 You need to display last name of Employee 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%' X

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) Y_A% anything AABA gut st ud

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

iii) 4th character must be A → '___A%'

iv) 2nd last letter must be A → '_.A_'

(6) A command to remove data from SQL database \rightarrow (Table).

- (A) Delete table < table name >.
- (B) Drop table < _____ > 3. Obj
- (C) Erase table < _____ >
- (D) Alter table < _____ >.

\rightarrow DDL Commands deal with Schema (Table Structure).

Create, drop, Alter \rightarrow DDL Commands.

\rightarrow Alter table \rightarrow to change anything in table.

{Delete table, Erase table} \times (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 \rightarrow 1. 3 obj.

c) only Q2, Q3 \rightarrow 1. 3 obj.

d) Q1, Q2, Q3 produce diff. results.

Ex:

T	Q1 (Q3)	Q2	Promissory Note with Data 'c'.
1	1	1	
2	2	2	
3	3	3	1, 2, 3
3	3	3	Q2 & Q3

Issue Table
Table {
Select (With Some data)}

(*)

PL-SQL

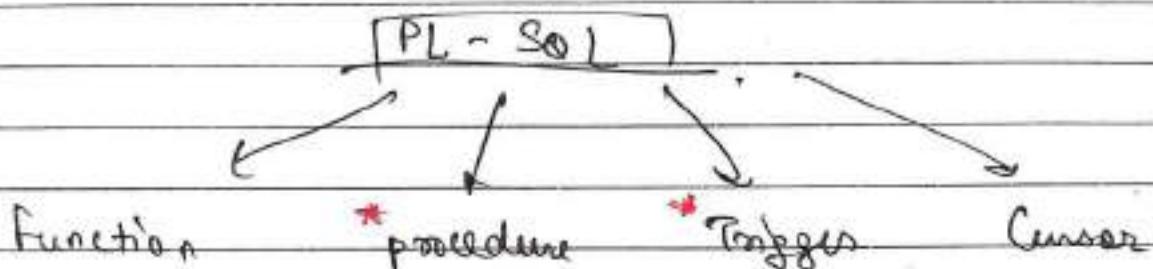
o →

(procedural - SQL)

→ SQL → declarative in nature. (under 'What to do').

→ PL-SQL → procedural (1. What to do →
2. How to do. →).

(programming flavour Std PL/SQL).



→ In SQL → (Write a query).

→ In PL-SQL → (Write a program, or write a PL-SQL code
(program write one at a time)).

→ One Block Code has 3 parts:-

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

declare	a int;
begin	
Executable	in C++
Code	
exception	
handling (error);	}

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

CPU always performs in RAM. CPU - fast.
CPU never works on Hard Disk. Hard Page Disk → slow

SM

Slow
Hard

Page
Disk → slow

Q2.

TRANSACTION CONCURRENCY :-

- # Transaction: It is a set of operations used to perform a logical unit of work.

(जब तक हम charge करे रहे, Database का
उपयोग, तभी तक Database का Read करे
रहे, तब तक यह भी part of transaction है.)

Ex:-

- When we withdraw our money from ATM,
then we have to perform a ~~set~~ of operaⁿs.
These set of operaⁿs 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 operaⁿs).

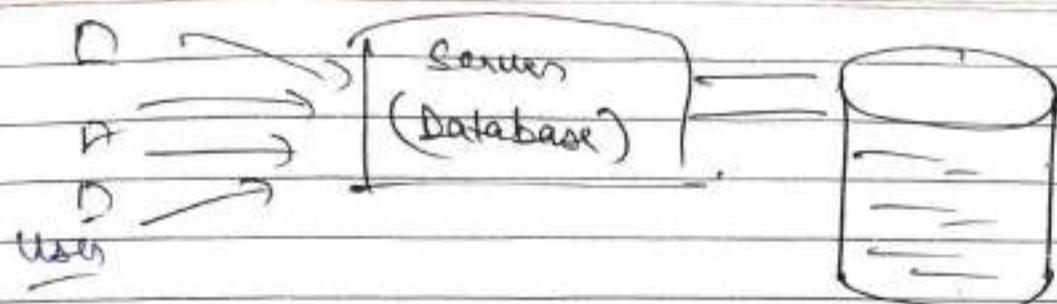
- # Commit - Whatever changes we made, save that permanent in Hard Disk.
(Update data in HDD)

In C++ for assignment, =
In PLSQL, :=

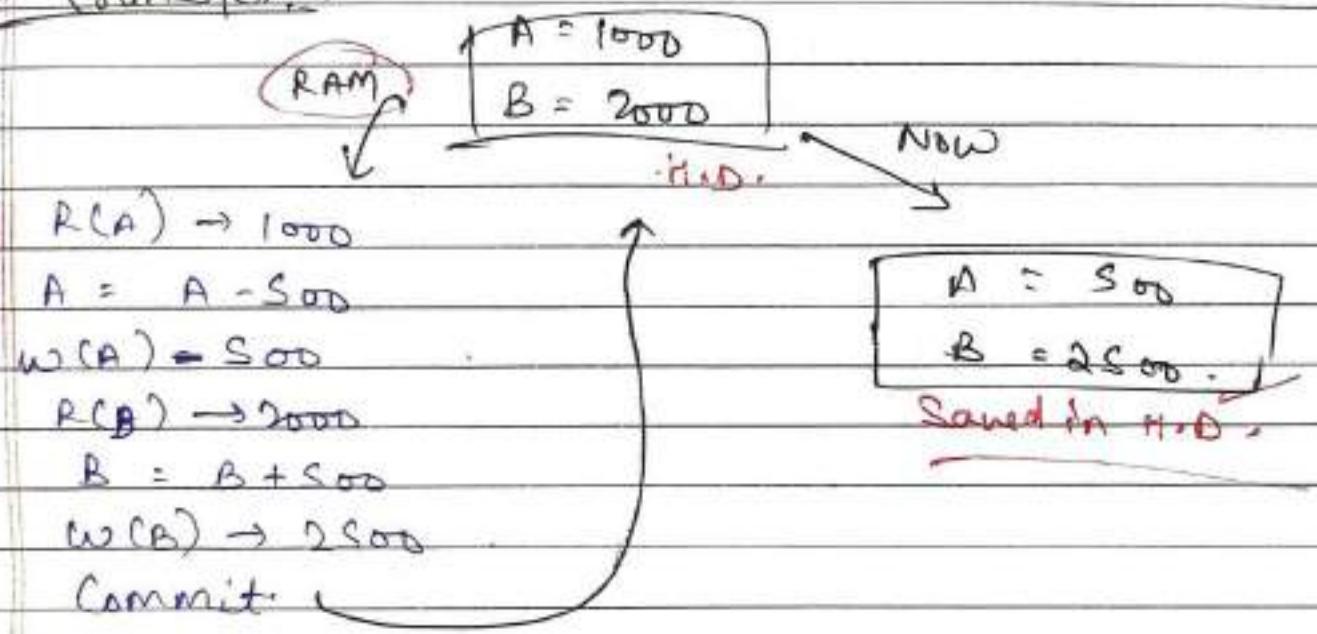
SM
Mutha
Durga

C++, =
PLSQL, :=

(137)



Transfer:

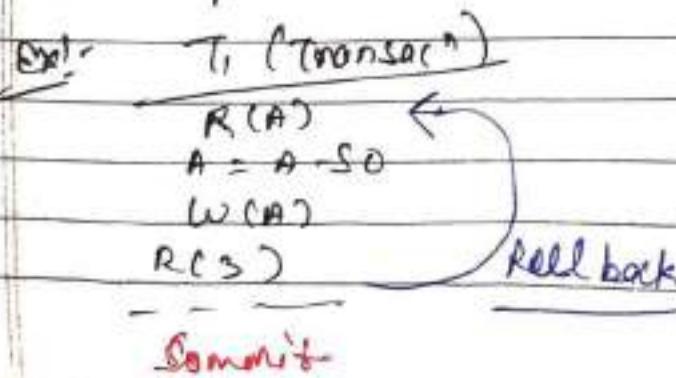


(73) - Acid properties of a Transaction :-

A → Atomicity
 C → Consistency
 I → Isolation
 D → Durability

} At back End

Atomicity → Either all or None:



(start commit at 4cm
 and all fail at start,
 then roll back)

→ At first open execute, commit, rollback.
After this roll back.

Note: A failed transaction cannot be resumed.
A failed transaction will always restart.

Consistency : →

Before trans. start And,
After the trans. completed,

Sum of money should be same.

Ex)

$$\boxed{\begin{array}{l} A = 2000 \\ B = 3000 \end{array}} \quad A \xrightarrow{1000} B$$

T₁

$$R(A) 2000$$

$$A = A - 1000$$

$$W(A) 1000$$

$$R(B) 3000$$

$$B = B + 1000$$

$$W(B) 4000$$

Commit.

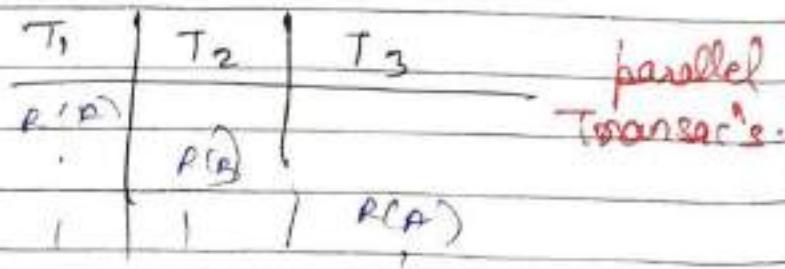
5000

$$\boxed{\begin{array}{l} A = 1000 \\ B = 4000 \end{array}}$$

3000 ✓

Sum is same

Isolation : →



→ We try to convert a parallel schedule into a serial schedule. (Conceptually)

CPU speed is in MIPS (million instrucⁿ per second).

Q Hard Disk \rightarrow 10/20 instrucⁿ per second.

CPU needs compatibility with HD for execution.

SM

Data

Prog

139

\Rightarrow Then,

Serial Schedule is always consistent

* Parallel



Schedule

T₁ \rightarrow T₂

T₂ \rightarrow T₁

* Durability : \rightarrow

Whatever changes are made, they must be permanent. i.e.

(Update for lifetime until we again update the data)

? That's why, we save data in HD for durability.

74)

Transaction States : \rightarrow

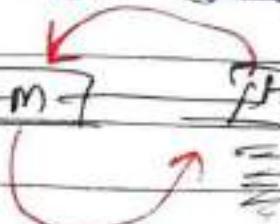
\Rightarrow At now, Transaction is in passive state. and as we start executing it, it comes into Active State.

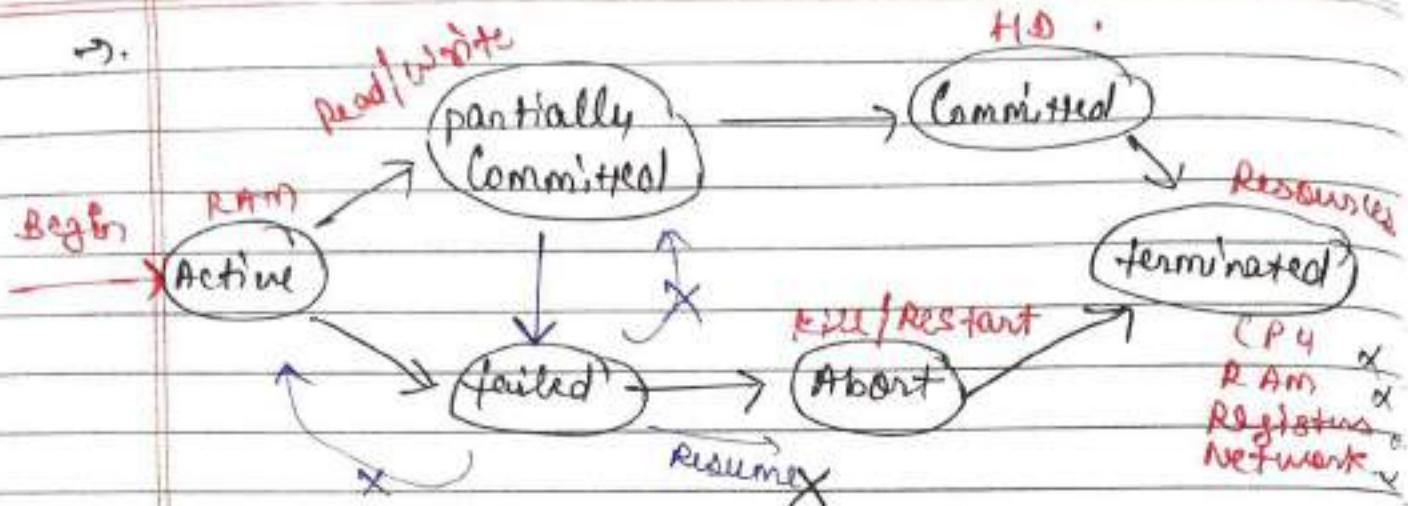
* In terms of C++ : \rightarrow When we write a program in C++ & save it. Now, the program is in Hard Disk in idle state. But, when we start executing or compiling, it comes into RAM that we called ACTIVE STATE.

ICPU

FRAM

FHD





→ partially Committed :-

All op's are done except Commit.

i.e. If N operation,
then

$(N-1)$ is done.

All op's are stored in local memory /
shared memory until now.

→ Committed :-

Changes are now saved to shared disk.

→ terminated :-

Here we deallocate our resources
i.e. free all Resources. bc,

Resources are United).

Now, they move to anyone else.

→ Failed → power failure, switch damage, etc.

(Failed either from Active or partially committed State)

→ Aabort : → Rollback the operation & restart the operation.

* - - - - *

(25.)

Schedule : → (Serial vs parallel schedule).

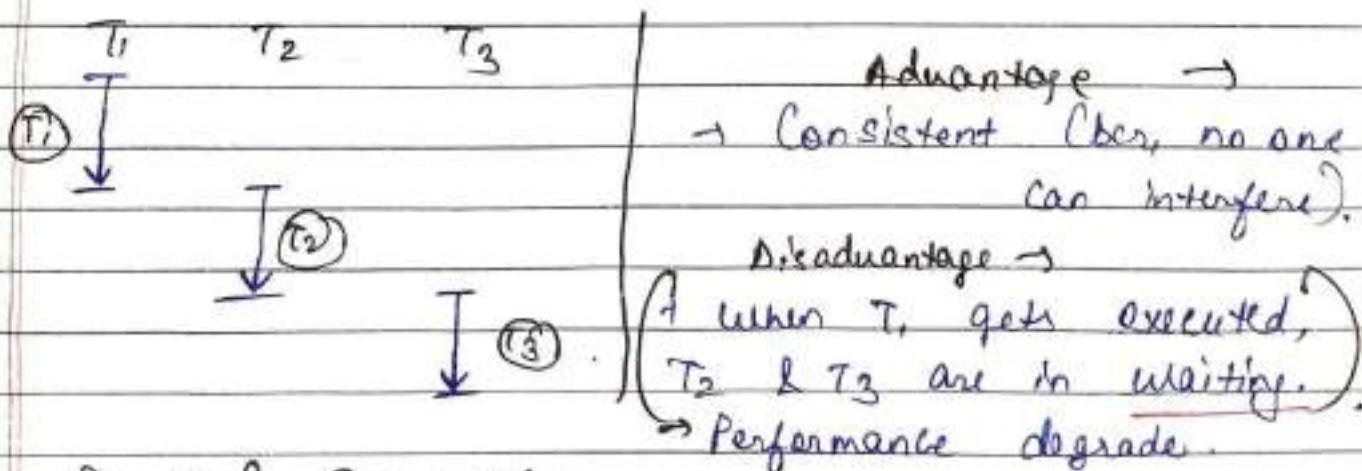
→ Schedule : It is chronological execution sequence of multiple transactions.

T₁ T₂ T₃ - - - - - T_n

Q. What is the sequence that these transac. are getting executed, that's called Schedule.

Serial Schedule : → Until a transaction gets completed, no other transac. can interfere.

All transac.'s executed by a ~~one~~ serial sequence.



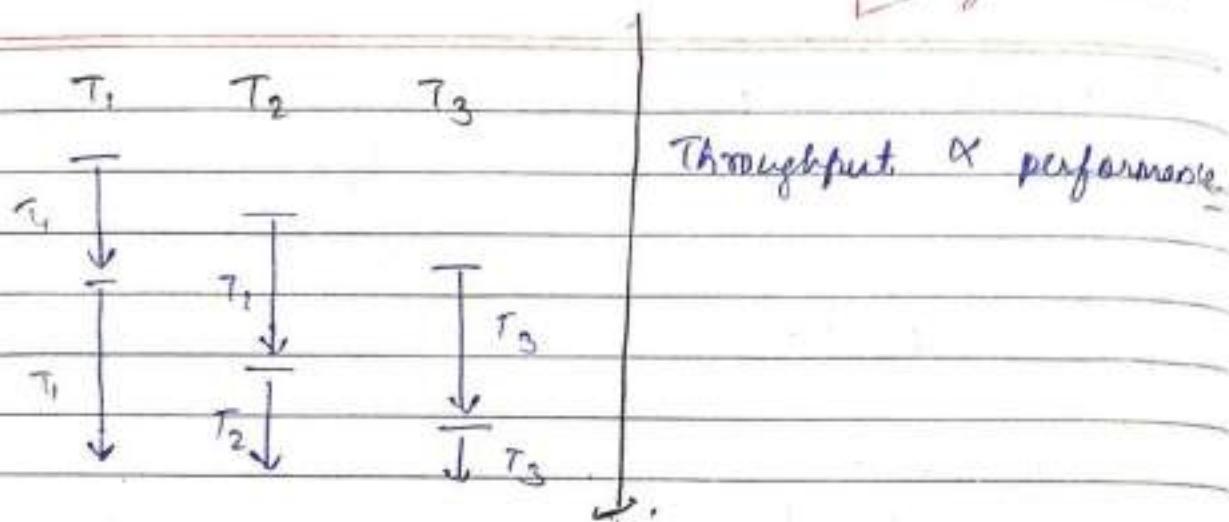
Parallel Schedule : →

- We can switch to multiple transac.'s at a time. i.e,
- Multiple transac.'s can execute at a same time.

Ex:- Banking Online System : → Multiple users can use at a time.

Throughput → No. of transactions executed / time.

SM Date _____
Page _____



Throughput \propto performance

→ Advantage :-

→ performance increased i.e., (throughput is high).

→ Disadvantage :-

→ problem may occur & (Inconsistent)

(*) Nowadays,

(Parallel Schedule is more preferred.)

↳ better good performance in less time.

→ →

(*) Types of problems in Concurrency :-

→ Concurrency means when multiple trans. executed at a same time i.e., Parallel schedule.

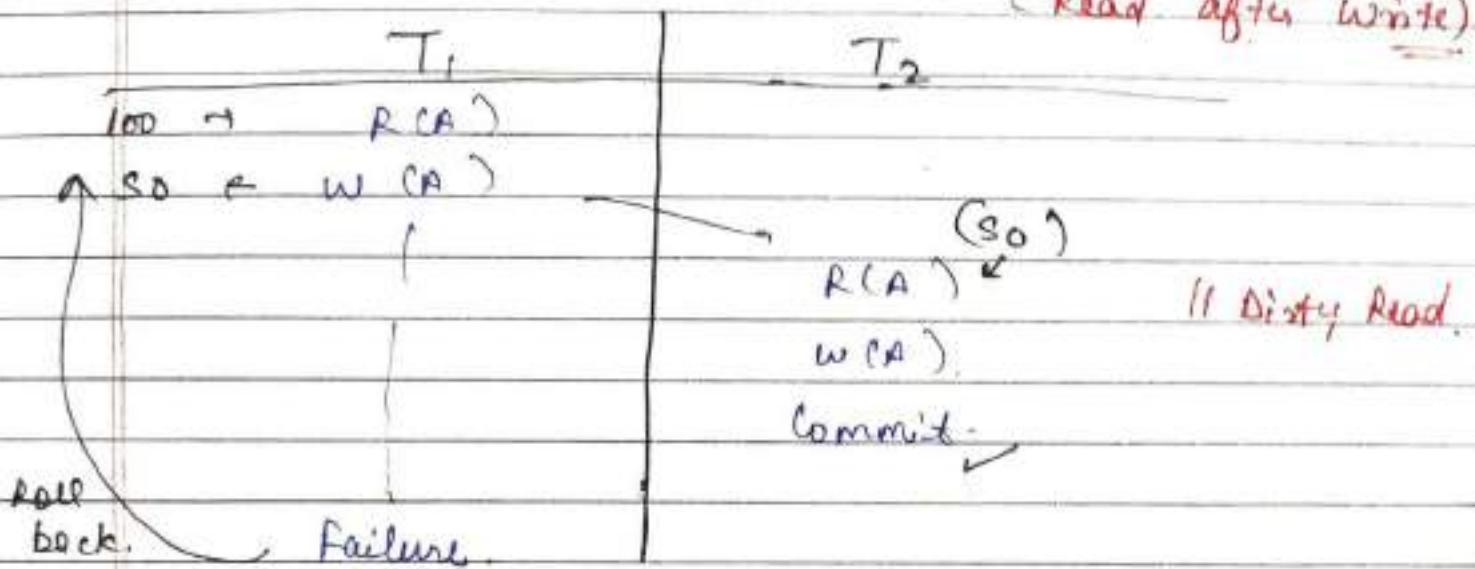
→ (mean, no problem is in Serial Schedule.
These are problems only in Parallel Schedule)

1) Dirty Read

2) Incorrect Summary

- 3.) Lost update
- 4.) Unrepeatable Read
- 5.) phantom Read.

1.) Dirty Read: \rightarrow or Uncommitted Read or PNR.
 (Read after write)



Hence, when T_1 gets failed. Then, how T_2 can use the A-so. So, Dirty Read.

2.) Incorrect Summary problem: \rightarrow (T₁)
 It occurs mostly when a transacⁿ start & T_2 comes b start performing its aggregate funcn's.
then, we get incorrect value of sum, average etc.

3.) lost update: \rightarrow

\rightarrow if 1st T_1 changes And 2nd changes at T_2 &
 2nd update the first 1st & 2nd don't changes \therefore this
 lost \therefore at 2nd i.e., [update for 2nd still lost at 2nd]

Ex: T_1 | T_2 (id-1)
 likes | likes
 5 likes | 5 likes

\Rightarrow & we finally get 5 likes on the same product instead of 10 likes.

4.) Unrepeatable Read →

T_1	T_2
$R(x) - 100$	$R(x) - 100$
$w(x)$	$R(x) - 50$
50	↑ Unrepeatable Read

- T_2 got diff. values on diff. Read. (100 & 50).
- So, It is also a problem.

5.) phantom Read →

T_1	T_2
$10 \leftarrow R(x)$	$R(x) \rightarrow 10$
Delete(x)	\rightarrow Read(x). (It also takes to Delete)
(0)	

7.2 Read-Write Conflict (or) Unrepeatable Read Problem →

↓ Cases are there →

Same Data.

$R(A)$	$R(A)$	Problem
$R(A)$	$W(A)$	
$W(A)$	$R(A)$	
$W(A)$	$W(A)$	

Ex-CP

User 1

User 2

 T_1 T_2 2. $R(A)$ $A = 210$ problem occurs
to this. $R(A) \quad ?$ $w(A) \quad ?$

Commit:

① $R(A)$ $w(A)$

Commit.

 $R(A) \quad ?$ $w(A) \quad ?$

Commit:

#

Ex- IRCTC

Let User 2 reserve both the seats. Then,
 $A = A - 2 = 0$ → If user 1 remained in painting for resource
or not. & when he does again.

Now,

[Seats becomes '0']

So,

Now, User 1 has to be roll back. (Abort)

Ex- 2

 T_1 T_2 $A = 10$ 10 $R(A)$ 4 $A = A - 1$ $R(A) \quad 10$ $A = A - 1$ $w(A) \quad 9$

Commit

9 $w(A)$

Commit

c. We issued 2

books, but

value still is 9.

(Instead of 8)

18.

Irrrecoverable Vs RecoverableSchedule In Transaction(S) Schedule

$$\begin{array}{|c|c|} \hline T_1 & T_2 \\ \hline 10 & \\ \hline \end{array}$$

~~10 R(A)~~

~~S A = A - S~~

~~S W(A)~~

$$\begin{cases} A = 10 \\ B = 20 \end{cases} \quad \text{10.}$$

R(A) S

A = A - 2

W(A) 3

Commit.

~~or R(B).~~

* fail.

(due to any reason,
(let, T₁ fails))

Now, (T₁ Rolls back due
to Atomicity property).

(either all or None)

Or roll back, everything happens in T₁, is
gone. So,

Again Now, $\boxed{A \leftarrow 10}$

But, here T₂ also done something &
that is lost now.

\Rightarrow T₂ Change is lost. we can't recover it
now. \Rightarrow It is Irrrecoverable Schedule.

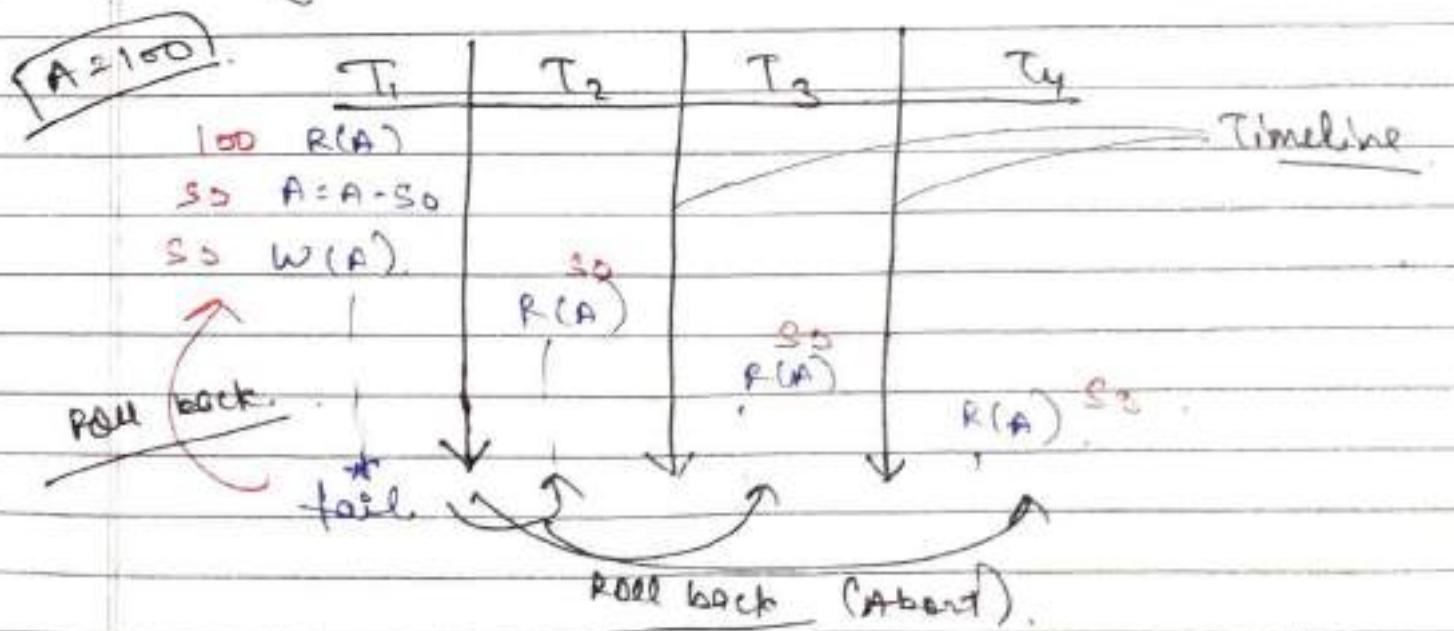
Q5. Cascading (Vs) Cascadeless Schedule :-

In recoverability, these 5 are important:-

- 1.) Recoverable
- 2.) Non-recoverable
- 3.) Cascadeless
- 4.) Cascading
- 5.) Strict Recoverable.



Cascading: means due to occurrence of one event, multiple events are automatically occurring.



Here, let T₁ fails due to any reason. Then, it will backs automatically due to Atomicity & again (A is 100). But, now T₂, T₃, T₄ has ($A \rightarrow S_0$). So, they are working on losing data. So, now, we also forcefully Roll back (Abort) the T₂, T₃ & T₄.

~~so, This is cascading.~~

(If T_1 fails, then we also have to roll back T_2, T_3 & T_4).

-1 Here,

CPU utilization gone waste by (T_2, T_3 & T_4).

Performance is bad / poor.

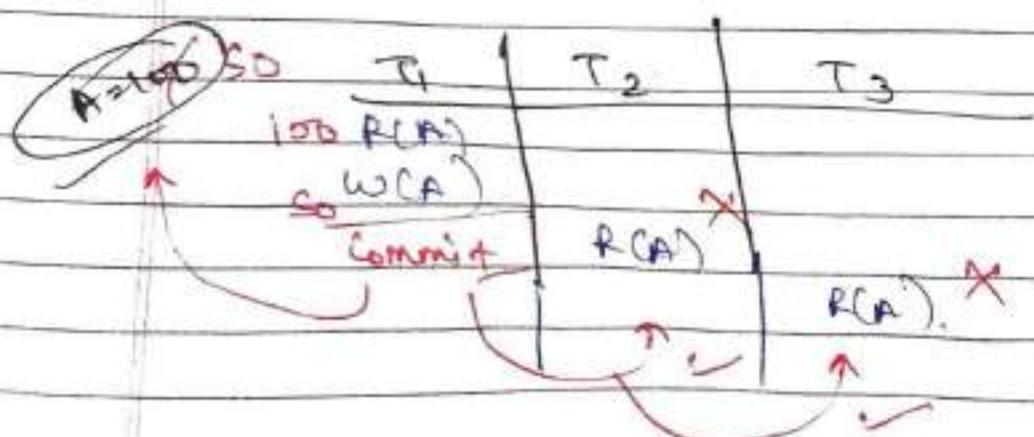
Ex: If a intelligent student (S_1) removes a answer. Then, other 3 students (S_2, S_3 & S_4) also cuts that answer. But, it is wrong. So, their work has gone waste. Cascading.

* Cascadeless: →

→ How to remove the problem of cascading?

→ Sol: T_2 & T_3 can't read the (A) value from T_1 until A value gets committed or roll back in T_1 .

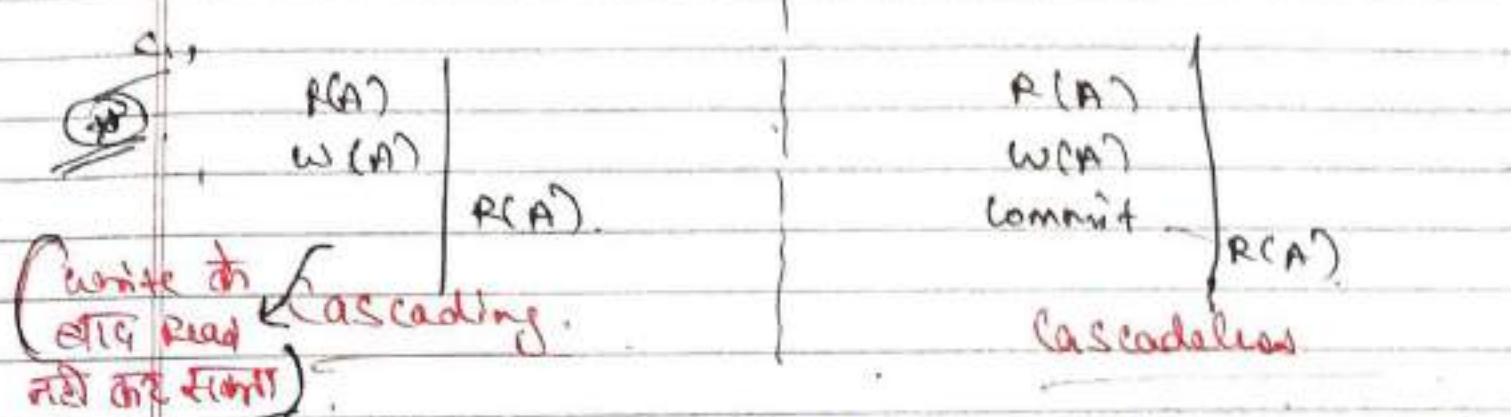
Commit → It is done at this point (at T_1)



→ ii. Don't allow Read in T_2 & T_3 .

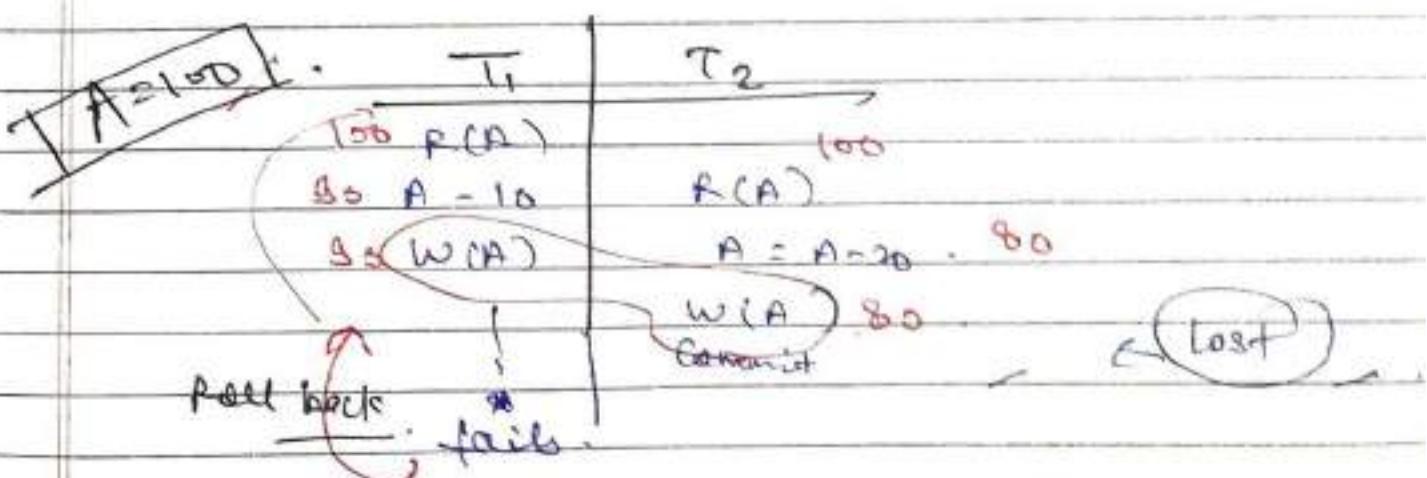
Ques,

∴ automatically becomes Cascadedless.



→ But, there is still W-W (Write-Write) problem in Cascadedless, bc,

(Read allows नहीं है, write तो कर सकता है)



→ Now, when T_1 fails (A becomes 100 again)

The work of T_2 automatically gets lost.
(i.e., Write-Write problem (or) lost update problem)

bc T_2 value of $W(A) \rightarrow 80$ at end & reflect

पॉस्ट की नहीं। → & finally $A = 100$

→ Strict Reliability tells that we also can't write along with Read.

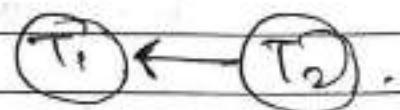
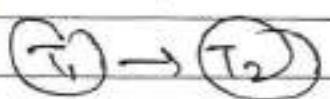
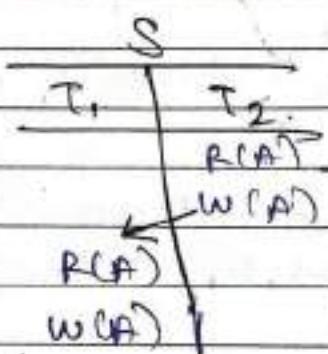
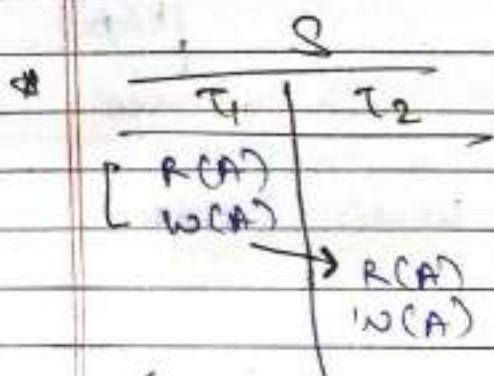
80

SERIALIZABILITY : →

Serializability means that a schedule has ability to become a serializable.

Mean,

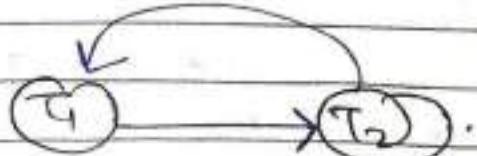
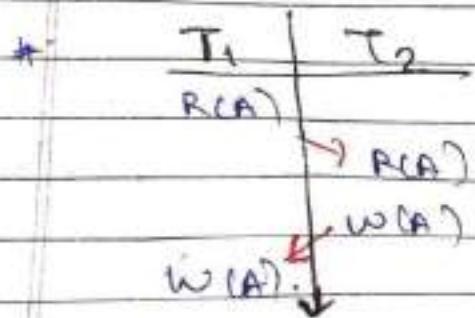
Schedule - call "of transaction" (T_1, T_2, \dots)



By seeing these, we can tell that they are already in serial schedule.

To, serial Schedule तो ये already हैं, (यहाँ नहीं कर सकते।)

→ We want, Parallel Schedule →



Convert it to Serial Schedule →

2 ways :

(1) $T_1 \rightarrow T_2$

(OR)

(2) $T_2 \rightarrow T_1$

→ To check Serializable? Check that if there exists an serial schedule equivalent to parallel scheduler 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. →
(clone of 11 schedule) Serializable

(#) List, a schedule has 3 transac's.

S			
<u>parallel</u> <u>Schedule</u>	T_1	T_2	T_3
	R(A)		
		R(A)	
		w(A)	
	w(B)		
		R(B)	
			w(B)

⇒ Serial Schedule
16 ways

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

⇒ 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 get 6 forms:-

Ans:-

we have 3 values ($T_1, T_2 \& T_3$)

Ans:-

3! $= 6$ ways.

$\times \quad \quad \quad \times \quad \quad \quad \times$

(Q) Conflict Equivalent Schedules :-

$R(A)$ $R(A)$ } Non-Conflict pair.

$R(A)$	$W(A)$	}	Conflict pairs
$W(A)$	$R(A)$		
$W(A)$	$W(A)$		

$R(B)$ $W(B)$ $R(B)$ $W(A)$	$R(A)$	}	Non-Conflict pair.
	$R(A)$		
	$W(A)$		
	$W(B)$		

(Q) How to Convert :-

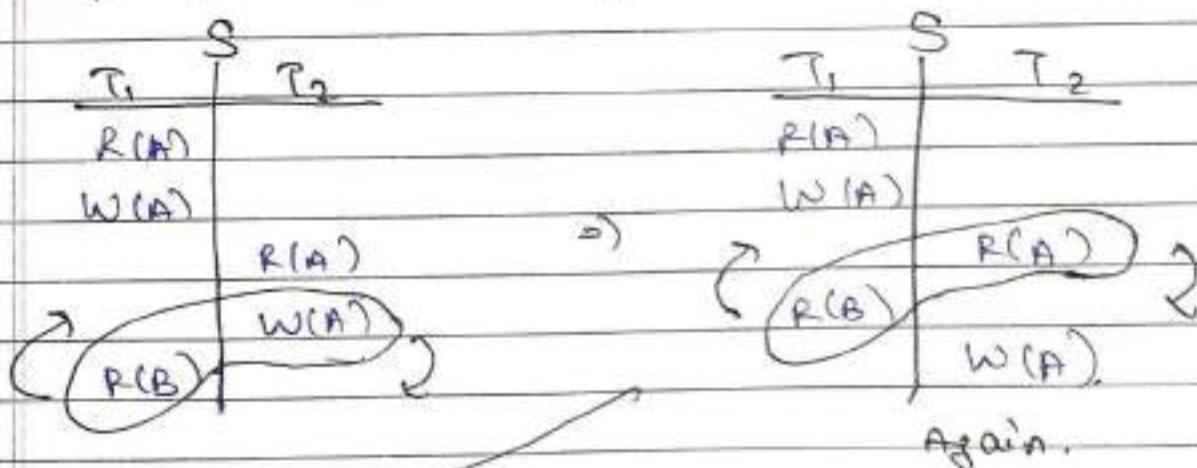
If we have adjacent non-conflict pair, then swap their pair.

Ex:- $T_1 | T_2$ $R(A) | R(B)$ $\Rightarrow R(B) | R(A)$

Q: To check Conflict Equivalent \rightarrow

		$S \equiv S'$ check	
S		S'	
T_1	T_2	T_1	T_2
R(A)		R(A)	
W(A)		W(A)	
	R(A)		R(B)
	W(A)		
P(B)			W(A)

Sol: So, In S, we have adjacent non-conflict pair, so swap them.

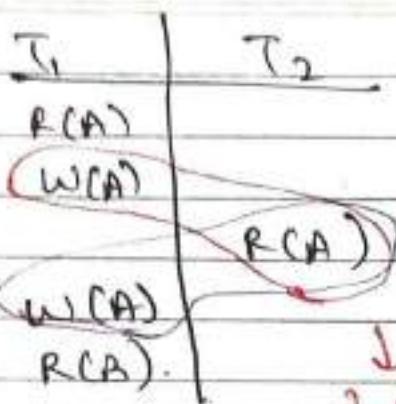


S		$= S'$	Hence,
T_1	T_2		$S \equiv S'$
R(A)			
W(A)			
P(B)		R(A)	
		W(A)	

Hence,

$S \& S'$ are conflict equivalent Schedule.

Ex:



2 adjacent pairs, But

they are conflict pairs. i.e., no change
in positions.

Note: \rightarrow

$S \xrightarrow{\text{CE}} S' \rightarrow$ Serializable
(Conflict Equivalent)
(Serial Schedule)

(1 - 81) videos end here,

Tai Shri Ram

INDEX

Anurag Singh DBMS STD: _____ SEC: _____ ROLL NO: _____

S.NO.	Date	TOPIC	Page No.	Topic Sign Number
82.		Conflict Serializability, precedence Graph	1-4	
83.		View " " "	4-6	
84.		Concurrency Control protocols	7-8	
85.		Drawbacks in Shared / Exclusive	9-11	
86.		phase locking (2PL) protocol	12-15	
87.		Drawbacks in 2PL protocol	15-17	
88.		Strict 2PL, Rigorous 2PL & Conservative 2PL	17-21	Sched.
89.		Basic Timestamp Ordering Protocol	21-24	
90.		Ques" on " " " "	24-25	
91.		INDEXES	26-28	
92.		Ques" on I/O Cost in Indexing	29-31	
93.		Numerical on " " " "	31-35	
94.		Types of INDEXES	35	
95.		Primary INDEX	36-37	
96.		CLUSTERED Index	37-38	
97.		Secondary Index (multilevel Indexing)	39-42	
98.		Intro to B-tree & its Structure	42-45	
99.		Insertion in B-Tree	45-47	
100.		How to find order of B-Tree	48-49	
101.		DB B-Tree & B+ Tree	49-53	
102.		Ques" on order of B+ Tree	53-54	
103.		Immediate Database Modification	55-57	
104.		O" on DBMS Basic Concepts & Data Modelling	57-59	
105.		Imp Ques" on Advance DBMS	60-63	
106.		Deferred Database Modification	63-66	
107.		LIKE Command in SQL	66-67	
108.		Basic PL-SQL programming with Execution	68-69	

103.	PL-SQL (while, for loop).	69 - 70.
110.	Single & MultiRow func's in SQL	70 - 71.
111.	Character func's in SQL	72 - 73.
112.	Views in Database.	74 - 77.
*	<u>SQL cheatSheet</u>	<u>78 - 81</u>

82

Conflict Serializability, Precedence Graph

T ₁	T ₂	T ₃
R(x)		
	R(y)	
	R(x)	
	R(y)	
	R(z)	
		w(y)
	w(z)	
	w(x)	
	w(z)	

↓ — Timeline.

First, we have to make precedence graph → (mean, just graph with edges & vertices).

Vertex → No of transaction (T₁, T₂ & T₃).

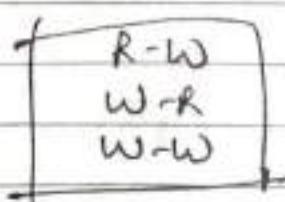
Now

* Edges

Check the conflict pairs in other transactions
and draw edges.

like * T₁ → (check in T₂ & T₃)

* Conflict pairs: -



of same variable
lets. (x).

* Start!

New, Start with "first open", i.e.,

T₂ in T₁ → R(x)

L check the conflict pair of $R(x)$
in other Transac's i.e. $(T_2, \Delta T_3)$.

→ Conflict pair of $R(x) \rightarrow w(x)$

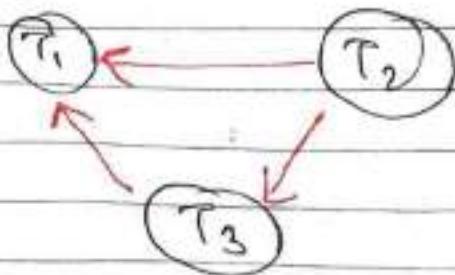
now, similarly,

conflict pair of $w(x) \rightarrow R(x), w(x)$.

Now, do check for every opno's in
all transac's & after checking list
these opno's b/c if you find any
conflict pair, then draw edge acc. to
vertex (Transac').

T_1	T_2	T_3	C.P. we find →
$R(x)$		$R(y)$	$R(x) - w(x) (T_3 \rightarrow T_1)$
		$R(x)$	$R(y) - w(y) (T_2 \rightarrow T_3)$
	$R(y)$		$R(z) - w(z) (T_2 \rightarrow T_1)$
	$R(z)$		$w(z) - R(z) (T_2 \rightarrow T_1)$
		$w(y)$	$w(z) - w(z) (T_2 \rightarrow T_1)$
$R(z)$	$w(z)$		
$w(x)$			
$w(z)$			

Precedence Graph:-



- # Now, come on Graph! -
Check that if there is any loop/Cycle in the graph.

How to check cycle: → This is the first step
Node at first, at second step if it is same
It might be 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? →

$$\left. \begin{array}{l} T_1 \rightarrow T_2 \rightarrow T_3 \\ T_1 \rightarrow T_3 \rightarrow T_2 \\ T_2 \rightarrow T_1 \rightarrow T_3 \\ T_2 \rightarrow T_3 \rightarrow T_1 \\ T_3 \rightarrow T_1 \rightarrow T_2 \\ T_3 \rightarrow T_2 \rightarrow T_1 \end{array} \right\}$$

6
Cases

→ Now, which case?

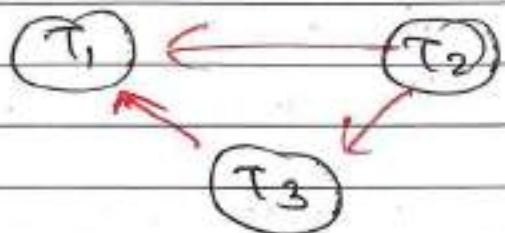
Again, see the graph
check

Indegree = 0

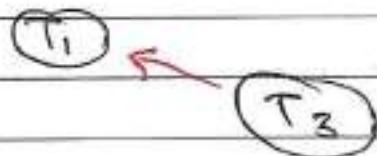


* Indegree = 0, means a vertex with '0' indegree.

Mean, T_1, T_2 don't get Edge (arrow) to T_3 from T_1 .



T_2 has, Indegree = 0,
Now, remove T_2 .



$T_2 \rightarrow T_3 \rightarrow T_1$ ✓

T_1	T_2	T_3
	0	
0		0

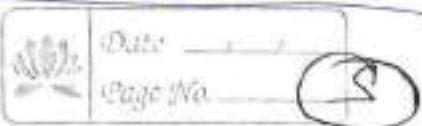
(83)

View Serializability : →

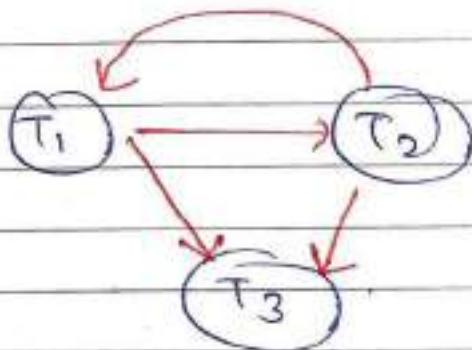
Q:- Check whether schedule is conflict serializable or not?

→ First make Precedence graph.

Loop \rightarrow No Answerable, then, view Serializable Answers.



S		
T ₁	T ₂	T ₃
R(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 Serializable.

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,

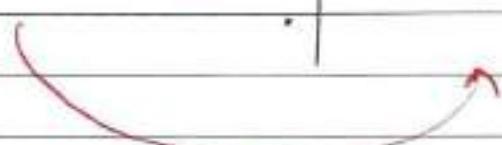
We have to check whether they match each other or not.

With A = 100.



T_1	T_2	T_3
100 R(A)	$A = A - 40$	
$W(A)$ $A = A - 40$ <u>20</u>	$W(A)$ $A - 20$ <u>0</u>	

T_1	T_2	T_3
100 R(A)	W(A)	
50 W(A)	W(A)	20
	$W(A)$	<u>0</u>



Now, apply this same here.

Here, finally
 A value = 0

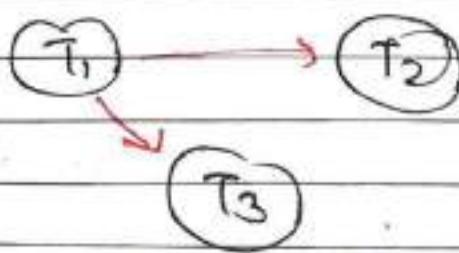
Here also
finally, A is '0'

Hence, Both are Equivalent.

They are View Equivalent (\equiv).

Note: When, finally output is given by A of T_3 . So, if adjust the pos' of A of T_1 & T_2 . So, there is no problem.

Now,



(By : Rep'ency
of only
 T_1)

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.



84.

(parallel schedule)

(C.C.P.)

Concurrency Control Protocols! →

(Shared - Exclusive Locking Protocol)

→ C.C.P. is that how to make them Serializable, or how to make them recoverable. Schedules 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).

T₁

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.

T₂

X(A)

→ Exclusive lock.

R(A)

W(A)

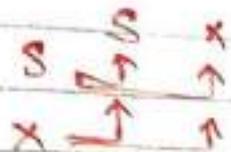
U(A)

→ unlock.

Compatibility

Table : →

		Request	
		S	X
Grant	S	Yes	No
	X	No	No



Expln:-

S has only R(A)

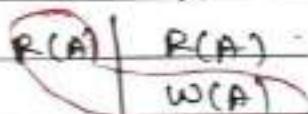
↳ X has both R(A), W(A)

So

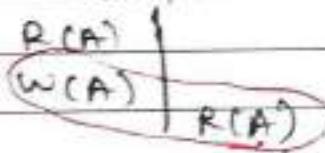
⇒ S - S (No conflict) b/w R(A)-R(A).

⇒ S - X (Conflict).

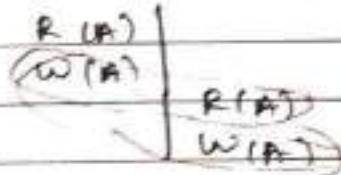
R - R



⇒ X - S (Conflict)



⇒ X - X



→ Conflict.

* Problems in S/X Locking : →

- 1.) May not sufficient to produce only Serializable Schedule.
- 2.) May not free from In-recoverability.
- 3.) May not free from dead lock.
- 4.) May not free from starvation.

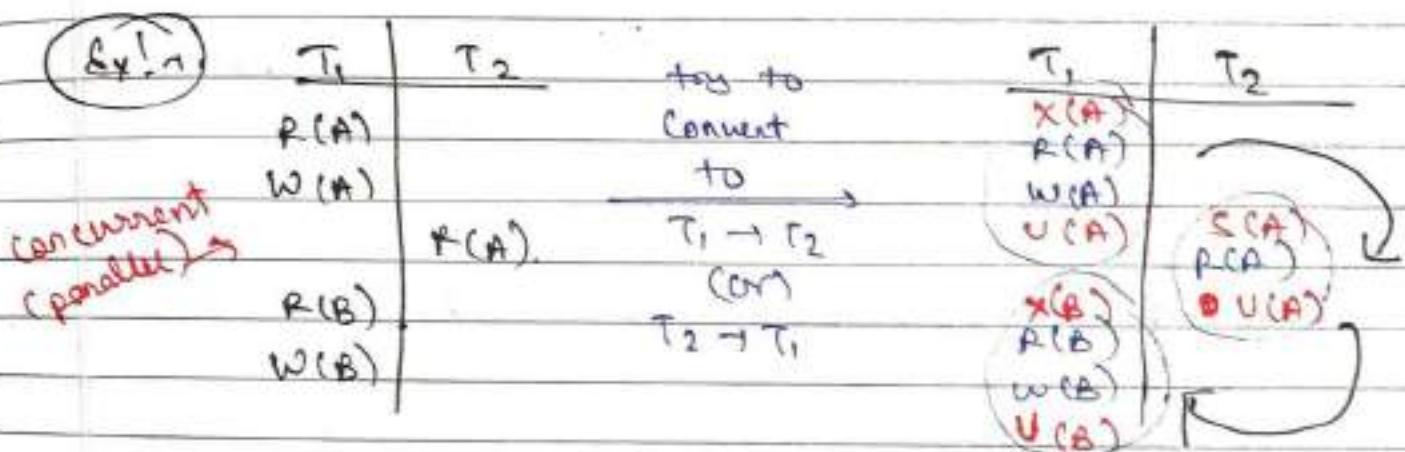
Q. 9

85.

Drawbacks

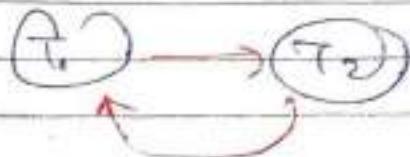
in Shared | Exclusive :
S/X Locking protocol

- locking gives us serializable schedule
→ Consistent
- i.) may not sufficient to produce only serializable schedule.
→ it get Hold, etc. fit.

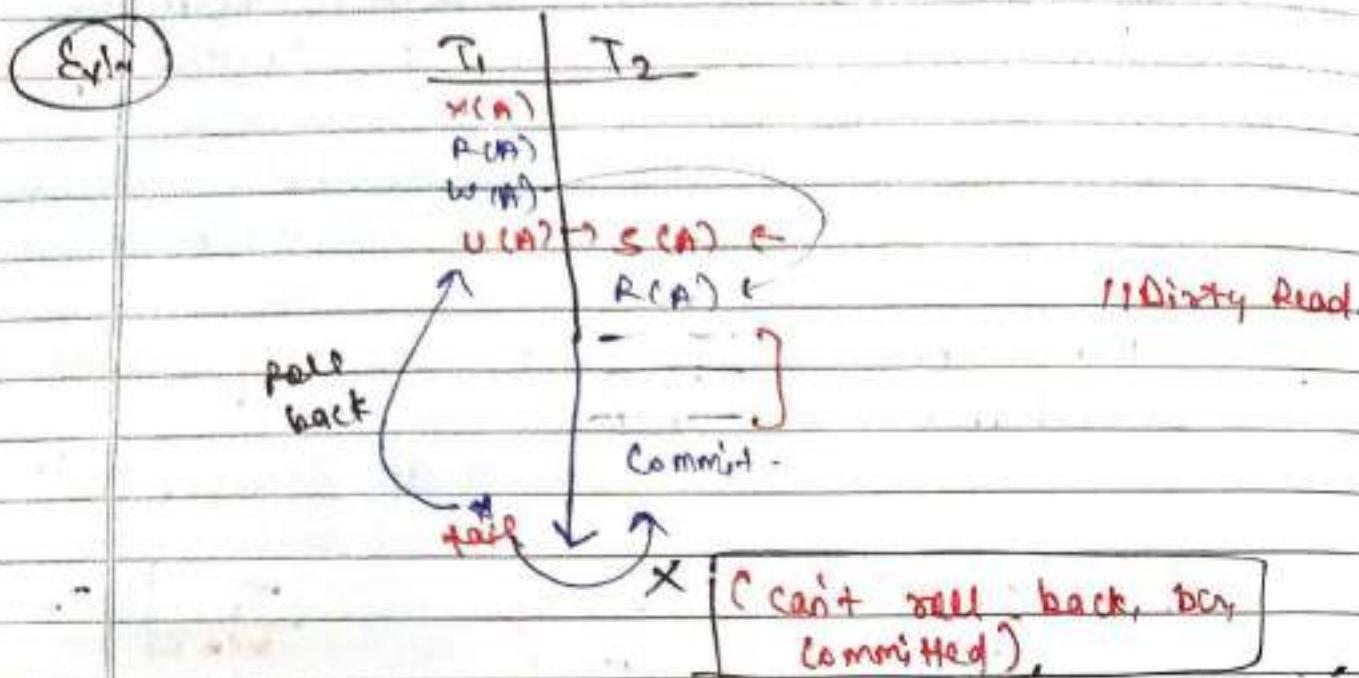


Note:- Until we do unlock, U(A) in T₁, we don't be able to put Shared lock S(A) on T₂. b/c by Compatibility Table. $(X-S) \rightarrow NO$, so first we unlock X(A). & then, use S(A) on T₂ data.

- a) Here, we don't get Serializable Schedule even after applying S/X locking. As u can see in Table.



2.) May not free from non-recoverability.



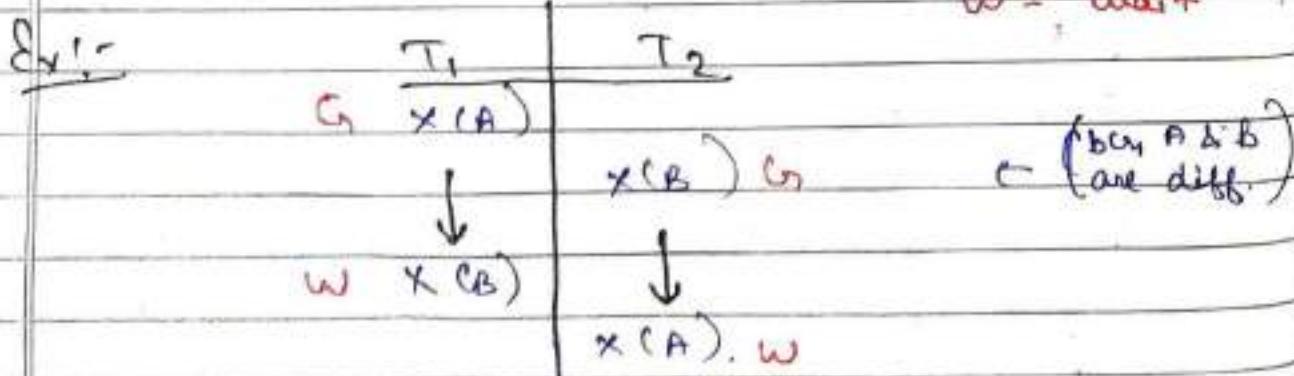
i. It can't recover here.

3.) May not free from deadlock.

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.

c - current
w - wait



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 to ~~by~~ Shared & Right ~~to~~, without unlock.

Ex:	T_1	T_2 S(A)	T_3	T_4	S → Shared lock X → Exclusive lock U → unlock.
w X(A)					
Waiting Time.	{	{	{	{	
	↓	↓	↓	↓	
		U(A)		S(A).C	
			U(A)		
				U(A).	

So, here, T_1 waits for $X(A)$ until T_4 unlocks.

So, here starvation also. (because shared by T_2 & T_3)
(Exclusive, P.M. unlock by T_4)
Grant after grant!

86

phase locking (2PL) protocol in
Transacⁿ Concurrency Control :-

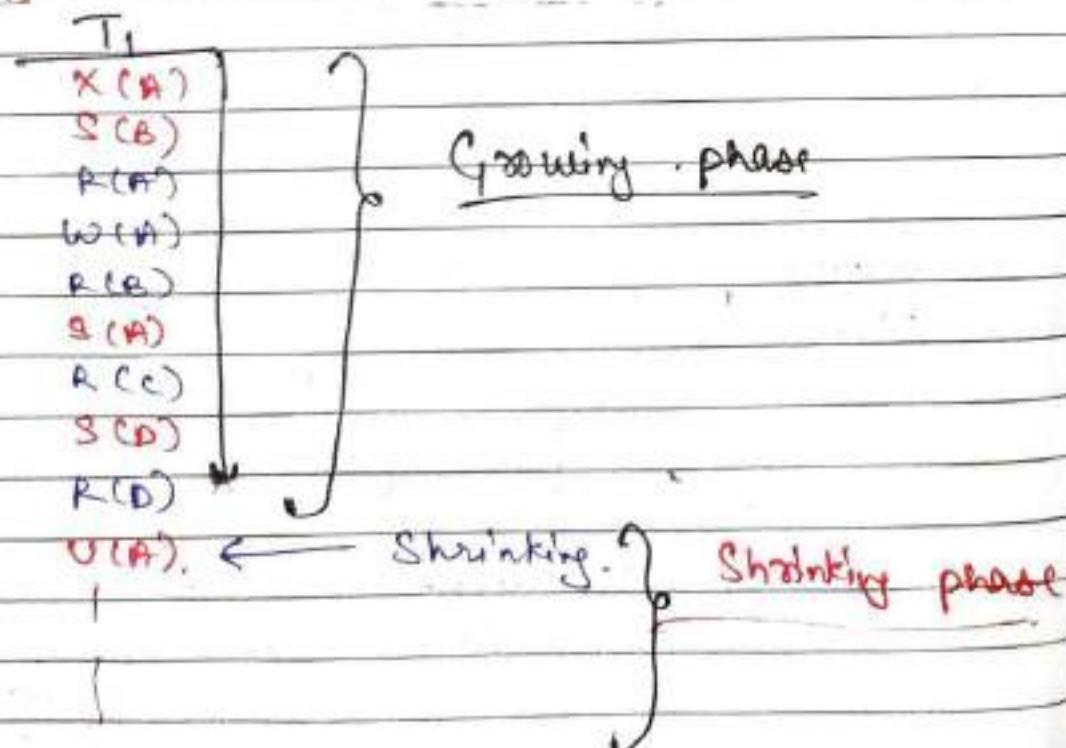
- 2PL (2 phase locking) is just the extension of simple shared/exclusive locking.
We just do modifications in them.

#

2-Phase Locking (2PL) :-

- Growing phase : → locks are acquired
↳ no locks are released.
- Shrinking phase : → locks are released.
↳ no locks are acquired.

TS/XS

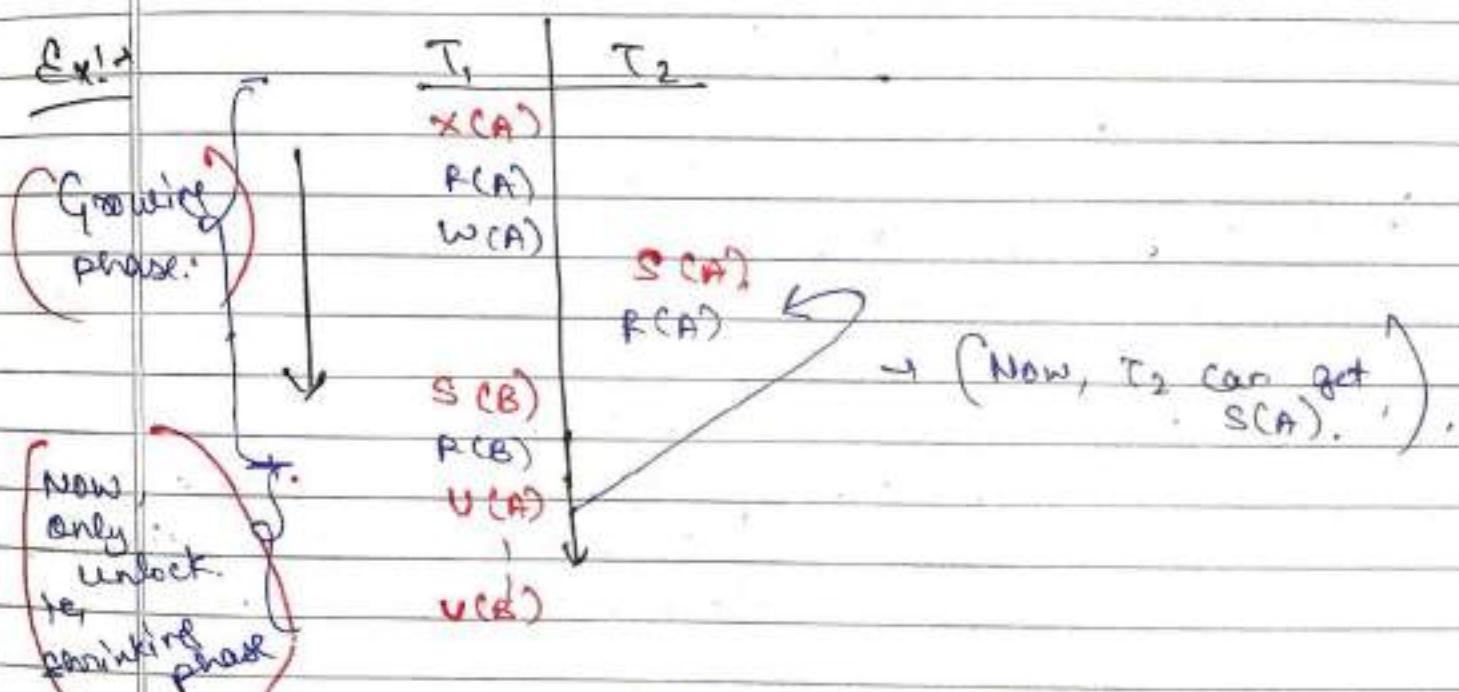


Note! When growing phase starts, then only apply locks.

[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.
BD,
we made (2-PL) protocol for this.



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.

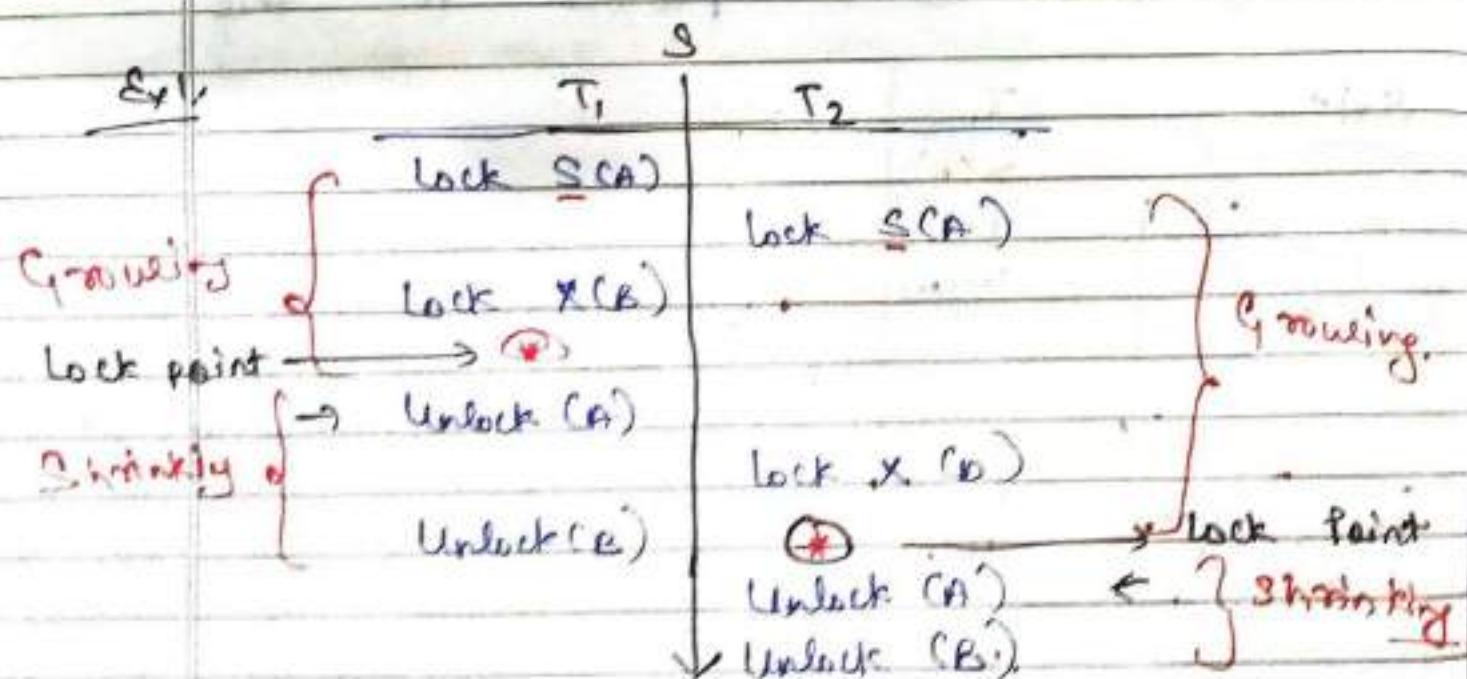
* Note:

Note: If the transaction which follow (2-PL), then it is always serializable.

Ex: 1)

Consider T1 starts at 3:42 Started at 4:01 and T2 Means,

While T1 is in growing phase by using S(A) [Shared Lock], then at same time T2 also starts its growing phase by using S(A). See by Compatibility Table.



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.

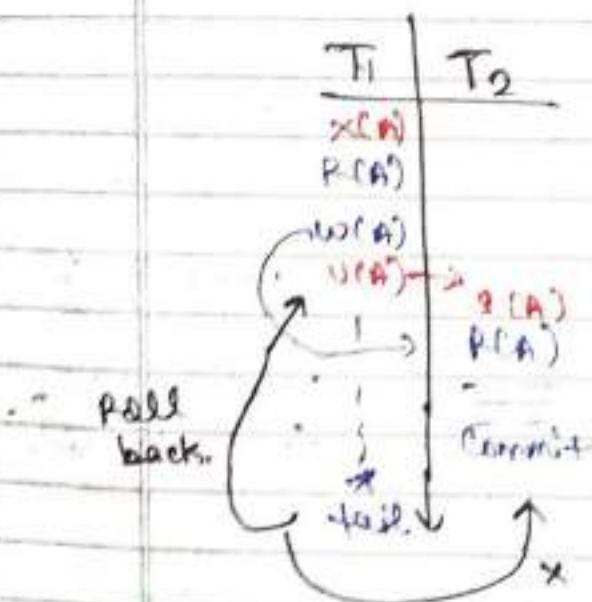
Commit at तीर्त्त roll back. ऐसे करें।

15

- ⇒ Lock Point: → where trans. is taking the last lock. (or) where trans. is unlock first time.
 - ⇒ function lock Point → it will happen at trans. used ~~at~~ before it.

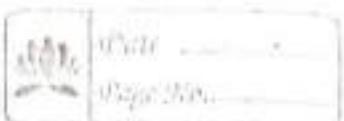
87) Drawbacks in (2-PL) protocol: →

- + **Advantage:** Always Ensures Serializability.
 - **Drawbacks:**
 - (1.) May not free from Sr. unrecoverability.

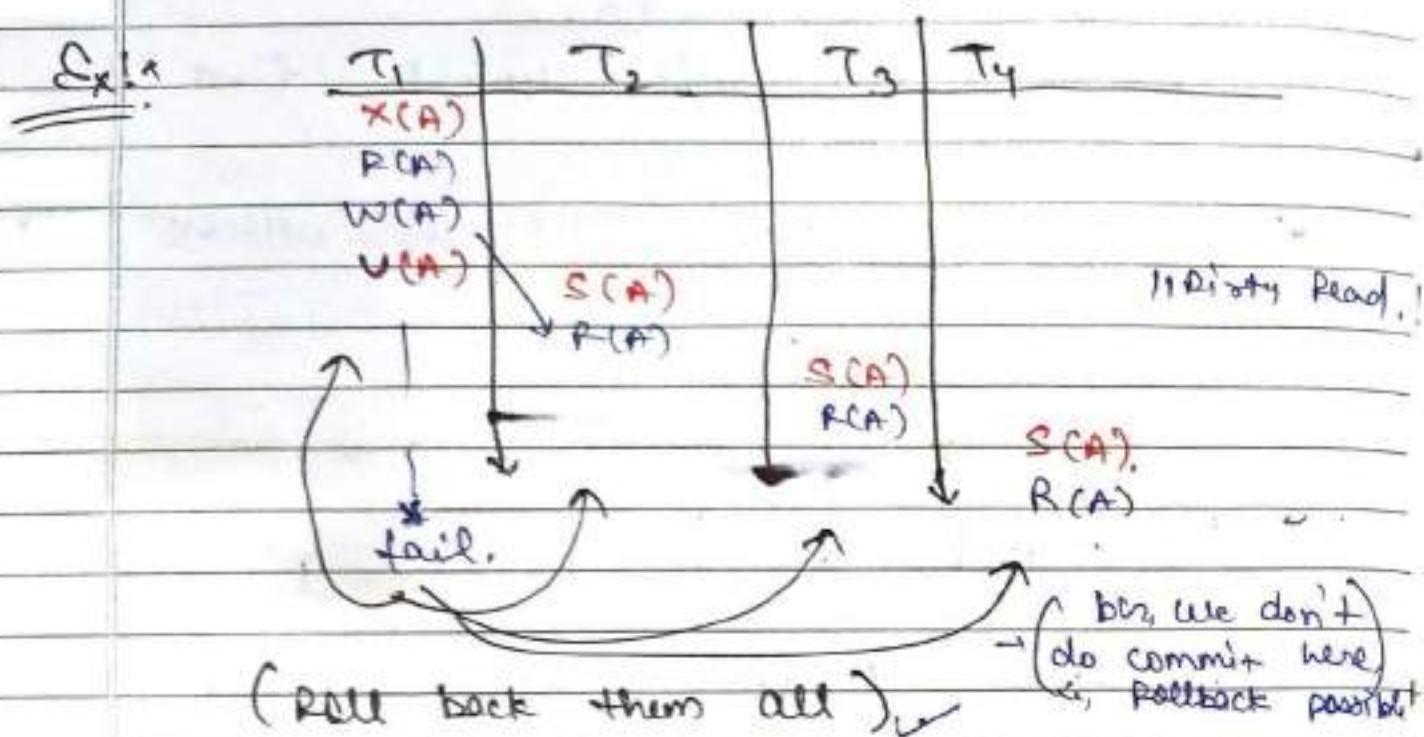


Hand

It is by Bf Irrecoverable Schedule.
We can't recover it (Roll back it).
(cancel).

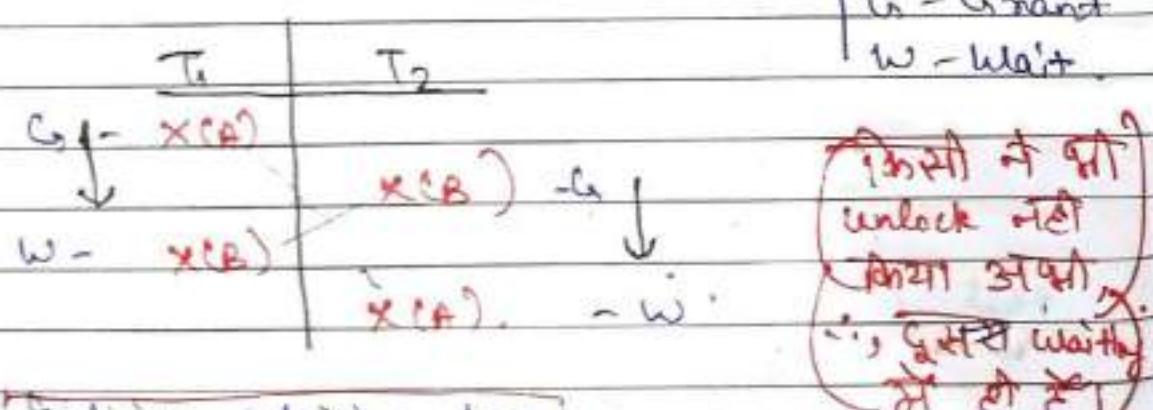


(2.) Not free from Cascading rollback \rightarrow



Bad performance. \rightarrow Cascading Rollback is possible here.

(3.) Not free from Deadlocks.



(T_1 & T_2 both waiting here to complete their transac's.).

(4.) Not free from Starvation.

(Wait for limited time).

On 2PL

Here, the problem of Serializability removes.

(12)

T ₁	T ₂	T ₃	T ₄
	S(A)		
w — X(A)	!	S(A)	
	U(A)	!	S(A)
		U(A)	!
			U(A)

Wait

(88) Strict 2PL, Rigorous 2PL & Conservative 2PL Schedule :-

→ These are the extension (enhancement) 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.

T ₁	T ₂	T ₃
X(A)		
R(A)		
W(A)		
	U(A)	S(A)
		UR(A)
unlock		
roll back	*	
	*	fail

i.e. (Cascading roll back)

→ also have to roll back T₂ & T₃



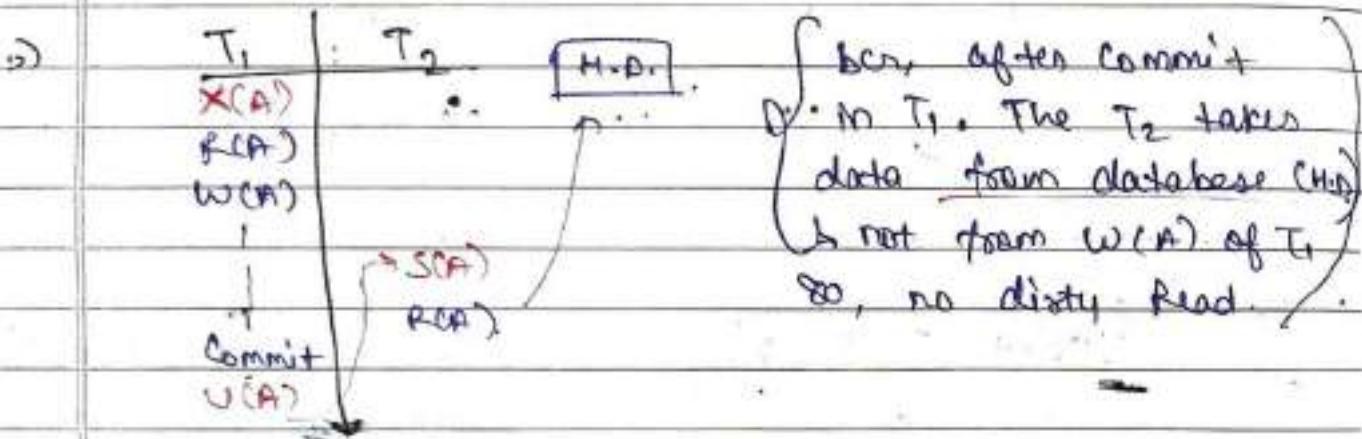
Date _____

Page No. _____

So, To Stop this →.

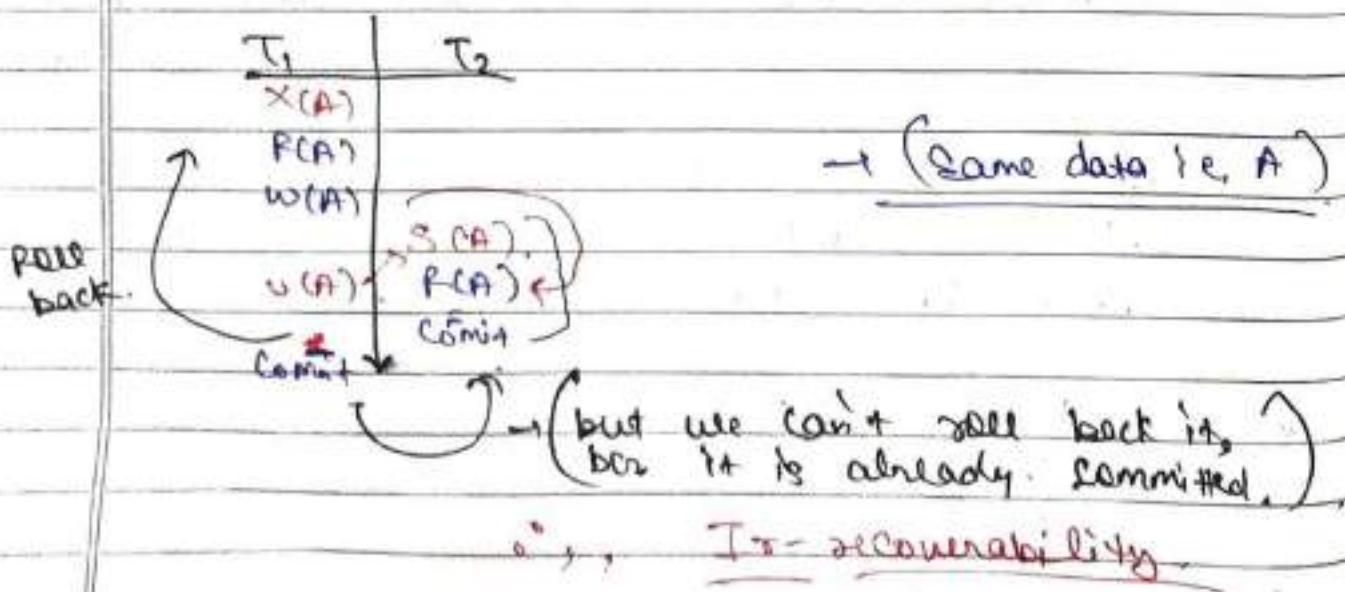
→ We have to unlock Exclusive lock after Commit.

then, this problem of Cascading Semantics.



Hence, Here, Cascading Rollback is Removed,
it will always produce Cascadeless.

⇒ Now, & To-recoverability! →

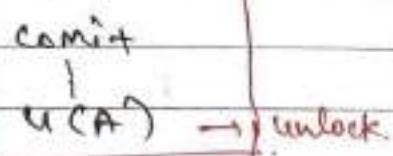


→ Hence, it is right to unlock it with $commit$.
we unlock it after $commit$ is done.

⇒ Hence, T_2 को S/X तुला ही नहीं मिलेगा,
जबकि वह T_1 commit नहीं हो जाए।

और अगर commit होने के बाद S/X भी नहीं होता
(T_2 को), then No problem.

⇒ Hence, In recoverability demands →
here,



वह file at Database (H.D.) से ही Read करेगा।

⇒ Set procedure Strict Recoverable Schedule.

Note:

2 problems demands here →

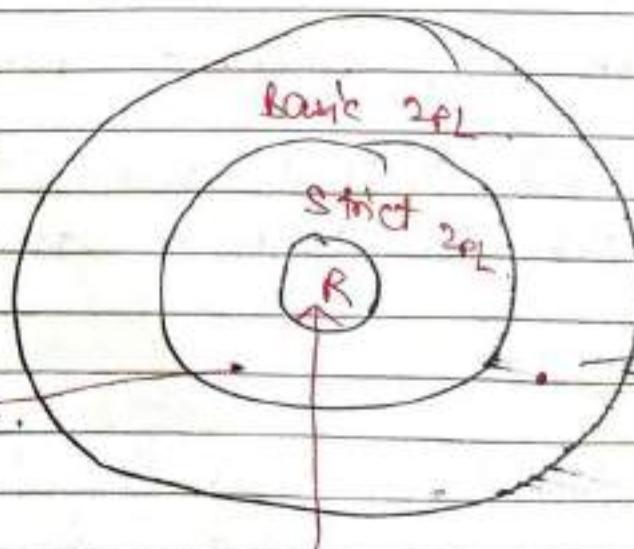
1.) Cascadeness

2.) Strict Recoverable

* Rigorous 2PL, इस ओर Strict हो जाया | ऐसे
में S/X दोनों तरफ होते हैं।

(इसमें हम shared lock का release नहीं
कर सकते।)

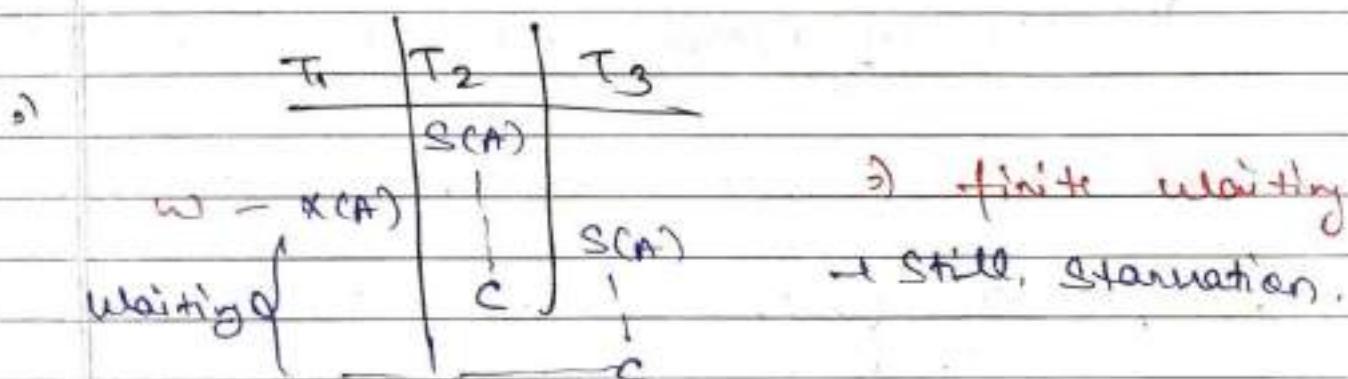
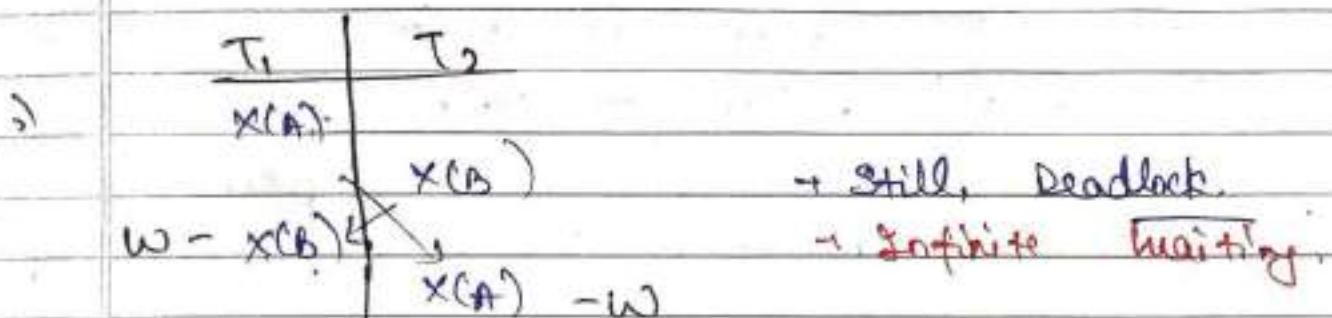
#



In both 2PL &
Strict 2PL. But
Not in Rigorous 2PL.

In Basic 2PL only

But, problem of deadlock & starvation still are there.

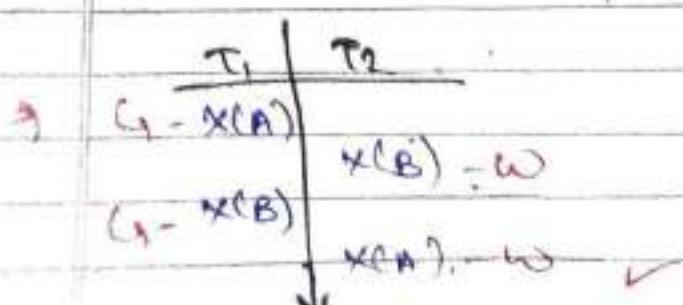


Conservative 2PL →

practically, it is not possible. In this, किंकोई भी trans. start होने से पहले ही साइंक lock लेता है। ($A, B, C \rightarrow$ सब ये लेता है).

→ अब तक T_2, T_3 को ज्ञान भी lock नहीं मिलेगा।
i.e., problem of deadlock removes here.

→ (T_1 को पहले ही ज्ञान resource के लिए T_2 , जिसकी T_2 को ज्ञान भी resource के लिए).



C - Current
 w - Waiting.

i, T₁ completes here first. So,
 no Infinite waiting. b/c

(T₁ is comp. & T₂ & T₃ use the same)

[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	→ Time stamp		
_↑ _{older} (read TS)	_↓ _{Younger} (write TS)			_↑ _{Youngest}	

- Read-TS (RTS) = last (latest) transaction no. which performed Read successfully
- Write-TS (WTS) = last (latest) transaction no. which performed Write successfully

Ex:-

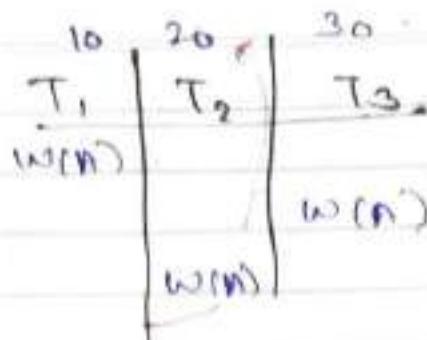
10 20 30 T.S. (T_i)

T ₁	T ₂	T ₃
R(A)		
	R(A)	
		R(A)

$$\Rightarrow RTS(A) = 30$$

↳ b/c, latest read is
 of T₂ & Time stamp (T.S)
of T₃ is 30.

Ex:-



$$\Rightarrow WT(A) = 20$$

* (4) Rules :-

- * In timestamp list, (wt timestamp) \leq (ts timestamp), all TS should complete first.
- * i.e.,

Serializability: Older \rightarrow Younger

Failure
Conflict
Consistent

1. Read

1. Trans. T_i issues a "read (A) opera"

a) If $WTS(A) \geq TS(T_i)$, Rollback T_i.

b) otherwise Execute R(A) opera

Set RTS(A) = Max of RTS(A), TS(T_i) }

2. Write

2. Trans. T_i issues "Write (A) opera"

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) opera

Set WTS(A) = TS(T_i).

* Understanding these :-

$$\text{Ex} \rightarrow \text{(old)} \quad \left| \begin{array}{c} T_1 \\ R(A) \end{array} \right. \quad \left| \begin{array}{c} T_2 \\ w(A) \end{array} \right. \quad \text{(Scaling)} \quad \left| \begin{array}{c} T_1 \\ w(A) \end{array} \right. \quad \left| \begin{array}{c} T_2 \\ R(A) \end{array} \right. \quad \left| \begin{array}{c} T_1 \\ w(A) \end{array} \right. \quad \left| \begin{array}{c} T_2 \\ w(A) \end{array} \right.$$

\checkmark (NP).

(No Conflict).

bcr.

$(\text{old} \rightarrow \text{young})$ or $(T_1 \rightarrow T_2)$. So,

(T₁ पर को execute की जाएगी, तो T₂ में कोई
Richard नहीं आएगी ।)

$$\frac{100}{T_1} \quad \frac{200}{T_2} \quad R(A) = 10$$

W(A)

! comit

Both back:

X

Pole Back. (Not possible)
(तो नहीं), (नहीं देना).

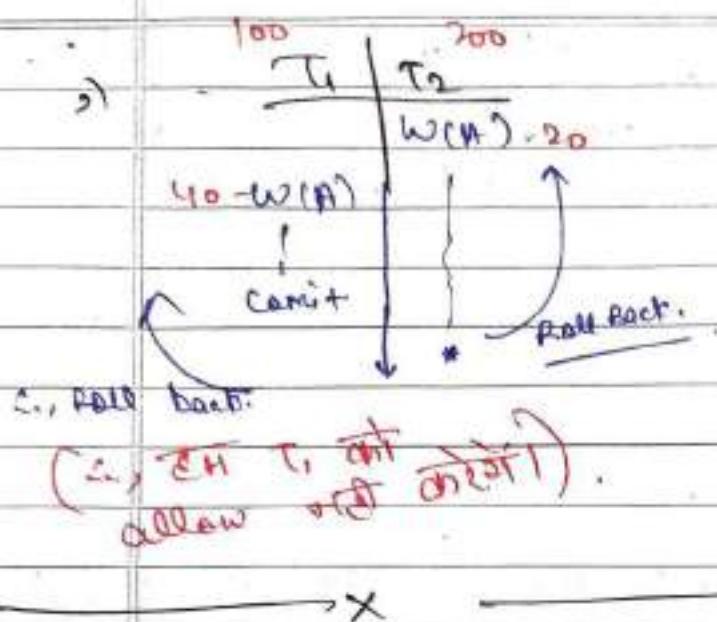
Case 1

(ber, now at T_2 ft.
A = 10, कोर - जी value
Read $\tan \theta$ के लिए वह ber,
 T_1 के लिए Database में 20
Update करें दिया।

(इस दो को बुद्धि
सारे करने देंगे।)

Hence, at 20 km/hr की
पहली दूरी । i.e,
T₁ में से Data पर कोई
one नहीं ।

11 Dirty Read



If T_2 fails,
then, our update
will be lost).

II Lost update.

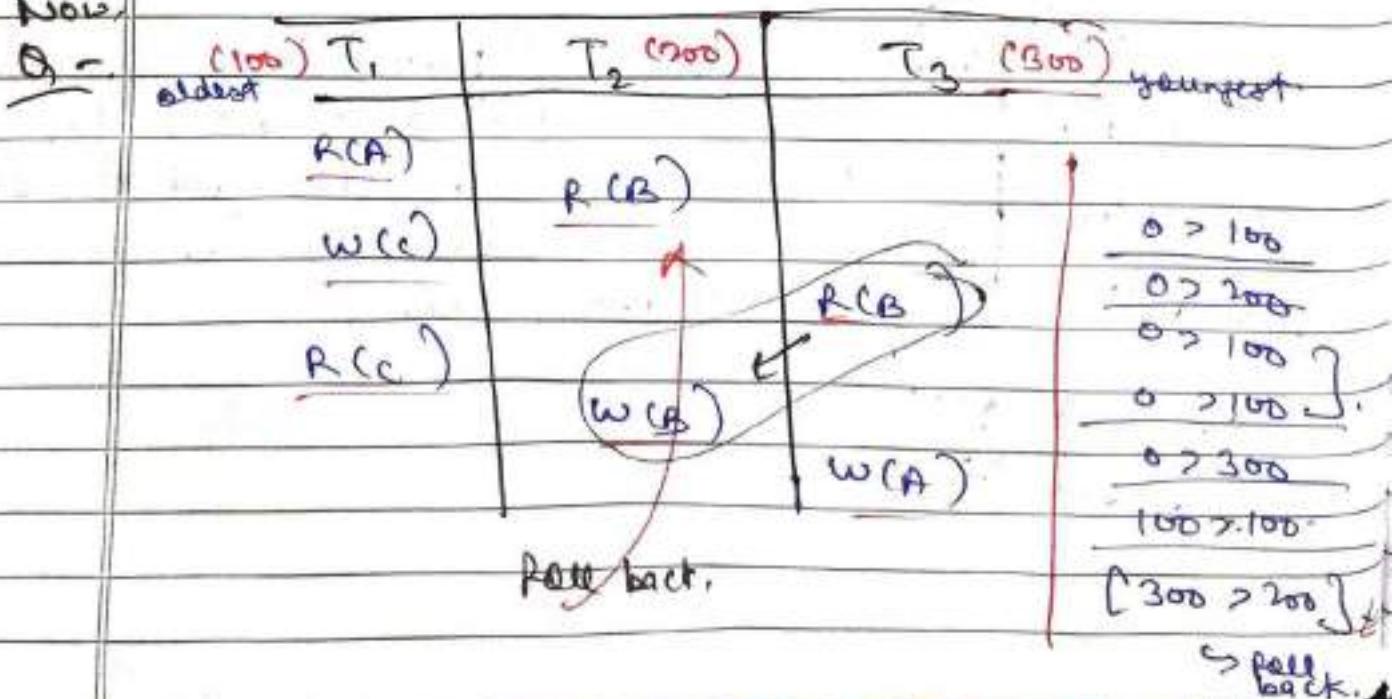
Q6. Solve Ques on Timestamp Ordering Protocol.

use same 2 rules →

1.) 1st Rule is of Read → has only 'cond's'
buz, only (R-W) conflict.

2.) 2nd Rule is of Write → has 2 cond's,
buz (W-W) → 2 conflicts.

Note:





100 > 300.
0 > 300.

	A	B	C	
R TS	100	200 300		100
W TS	0	0	0	100

⇒ Initially,
all values
are 'zero'

(Final Table) ah.

(Turns)

⇒ 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 T₁, then
R(B) of T₂, then
W(C) of T₁, then
R(B) of T₃, then
R(C) of T₁, then
then
here Roll back. ← W(B) of T₂, then
W(A) of T₃ → then

i.e., pattern wise as in Table

8

Then, make the final Table of :
(R TS) & (W TS) ah.

[OR]

We also do the Ques" direct, without using
these 2 rules with just the [older → younger]
concept & cases (we discussed in prev. video).



Q1.

INDEXING : →

- CPU → processor (who process).
- Query comes to CPU. (CPU process others.)

→ ~~Select * from student where Roll no = 1;~~

~~Select * from student where Roll no = 1;~~

- CPU has to execute it, But Data is in the memory.

- In general architecture, we only get,
2 types of Memory →

- 1. RAM
 - primary memory
- 2. H.D.
 - Secondary memory

- Ram is volatile, Ram works directly with CPU.

• CPU Speed → 20 MIPS (Million Instructions per Second)

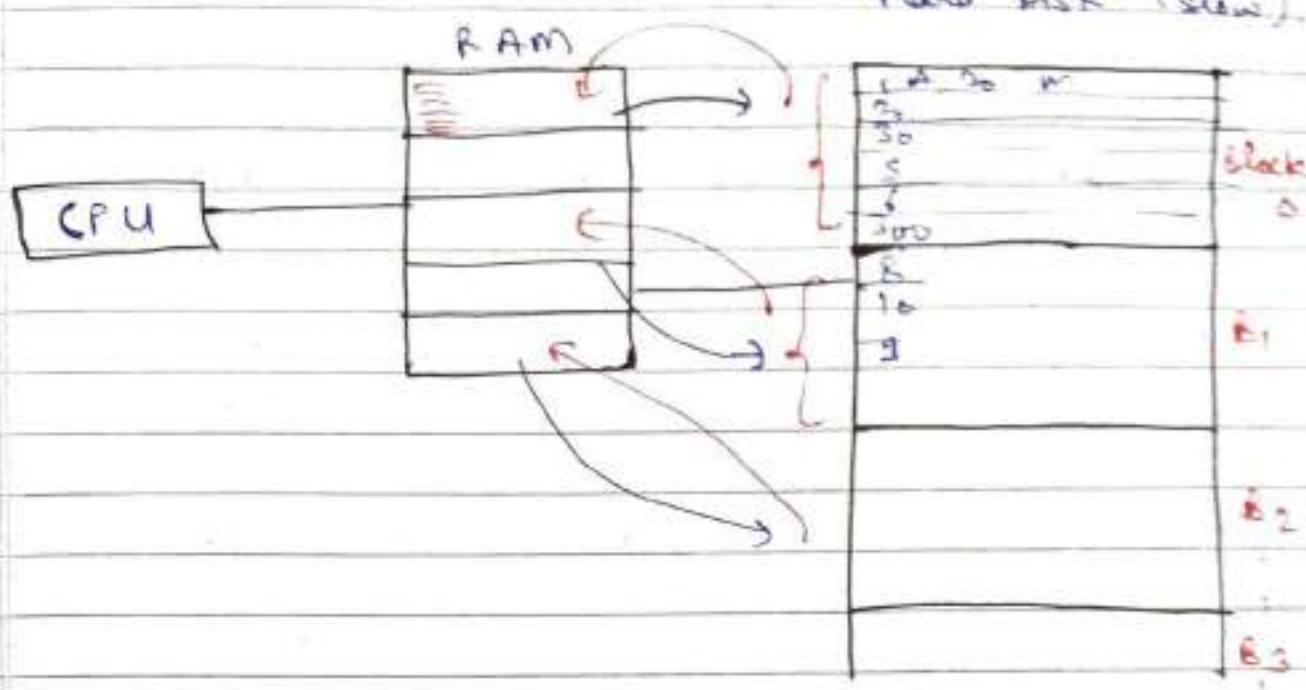
→ H.D. Speed → 10 Iops (Instructions per second)

i. these both are not compatible with each other.

SO,

- Ram comes b/w them.

→ Ram speed counts in milliseconds.
CPU → nanoseconds.



- # We divide Hard Disk in blocks.
Logical Blocks (but not physical blocks)

Ex:-

Logical Drives \rightarrow C Drive, D Drive, etc.

- # O.S. divides hard drive into fixed sized blocks \rightarrow then insert Data. (blocks / page)

Ex:- If we have record of 10k students

&

Block size is of 100 Records in Hard Disk.

so,

we need 100 Blocks in H.D. ($100 \times 100 = 10k$)

- \Rightarrow Now, data may also be stored in 2 ways
1.) Sorted (ordered) 2.) Unsorted (unordered).

\Rightarrow Let,



- 1) Unordered Data: + then, we send 1-by-1 every block of H.D. into RAM. & if we get our data, then OK. Otherwise, RAM send back that block & next blocks come into RAM. & go on.

If we get data → HIT

If we don't get → Miss

(*) So, Here, INDEXING is used.

(~~Ent~~ Ram ~~H~~ H.D. in Block ~~in~~ searching time in Hit ~~chart~~,).

→ We transfer data from H.D. to Ram. So,

there is ~~etc~~ transfer cost. i.e.,

[I/O cost]

→ Input / Output

For Data ~~in~~ call the ~~at~~, ~~select~~ ~~block~~
I/O cost.

• 2) If call more blocks, then I/O cost more.
& then time also increases of searching.

(*) 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 \rightarrow 500 pages.

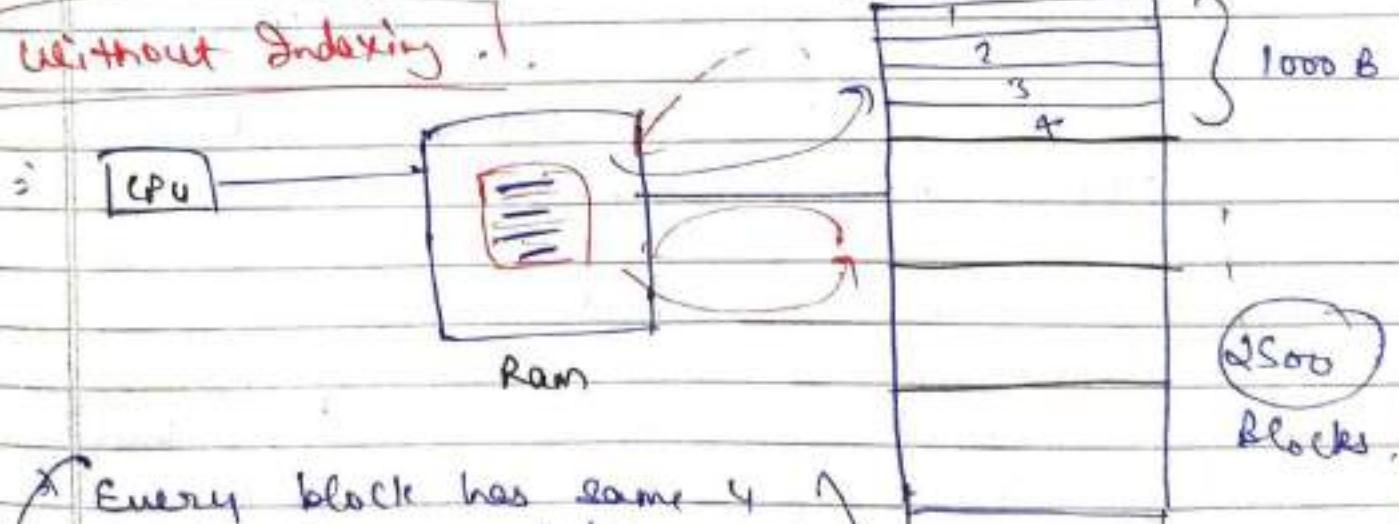
- \Rightarrow If we use Index, then the no. of pages in shuffle case decreases.
- \Rightarrow (Let, Index is of 2-3 page, then just see that topic on Index & then directly go to that page).

Q2- Numerical Ex. on I/O Cost in Indexing:

Q1 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 i.e., always 4.

HD

(2500
Blocks)

3) No. of records we can put

$$\text{in every block} = \frac{1000}{4} = 250$$

4)

$$7) \text{No. of blocks required} = \frac{10000}{2500} = 4$$

→ I/O Cost :-

5) Best Case = 1

Worst Case = 2500 = N.

$$\text{Avg. Case} = \frac{2500}{2} \Rightarrow 1250$$

$$\frac{N}{2}$$

Avg Time Complexity: $O(N)$

& This is all for unordered Data.

Hence, we used Linear Search here.

6) 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, $\Rightarrow \log_2 2500$

≈ 12 approx

Hence,

we need approx 12 blocks in ordered data.

b, this both case are without Indexing.

b) When we used Indexing, we even need less blocks than these.

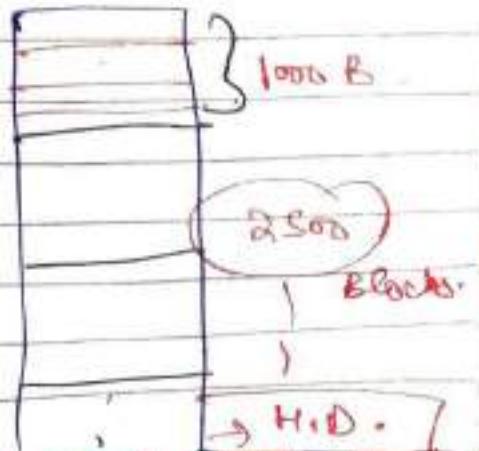
I/O cost is low.

Q3)

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 = 20 B (key + pointers) = 10 B + 10 B.

i) 2500 blocks



i) 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 = Block size in H.D.



To understand: In a book, all pages are of same size. Whether it is index page or anything.

② Index Table Entry = 20 B (key + pointer).

key value → $\frac{20}{2}$ Value 1st as search size 2nd
(Topic name), roll no. etc.
pointer → page No.

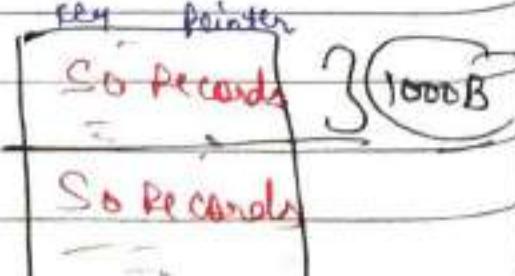
Each block in Index Table = $\frac{\text{Block Size}}{\text{Index Entry Size}}$
contains Record

$$\frac{50}{2}$$

$$I = 50$$

Hence, we can put 50 records in a block of Index.

(i.e., contains only topic name)
 \rightarrow 50 records in 1000 B.



→ 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: 1

Here, all 10k records are to be entered in Index.

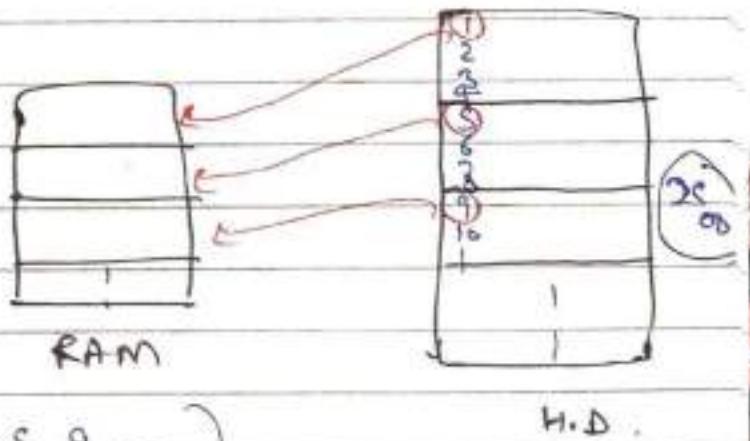
2) Sparse! used when data is ordered [sorted].

Here, we just Enter 1 record from the block into the RAM like,

we entered ~~100~~

Anchor data (header)

into the Index. (1, 5, 9, ...).



Ex: 1 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,
ie, Sparse →

No. of ~~records~~ in Index = $\frac{2500}{50}$ [→ 50]

Then, search time $\rightarrow \log_2 50$

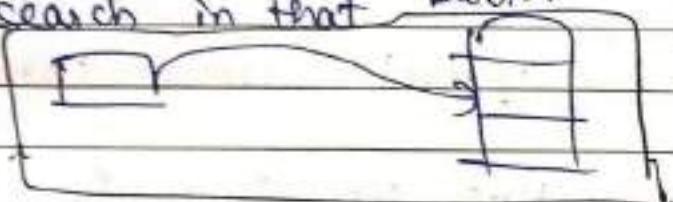
$$\approx 6 \text{ approx}$$

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



$$\text{Total no. of Search} = 6 + 1$$

$$= 7$$

6 search in Index.

1st (1) search in H.D.

But

\rightarrow Our data is unordered \rightarrow .

Here, Dense \rightarrow ,

No. of blocks in Index = All records

Record in one block of Index

$$= \frac{200}{10,000}$$

Sp

Now,

$$\approx 200$$

$$\text{Searching} \rightarrow \log_2 200 + 1$$

$$\approx 8 + 1$$

≈ 9 approx

$$\log_2 2^8 + 1$$

$$\approx 8 + 1 (9)$$

$$2^8 = 256$$

→ Here, we use $\lceil \log_2 n \rceil$ also, b/c
Index is to unordered data get ordered/
sorted (in ASCII form).

∴ we use $\lceil \log_2(n) \rceil$. here.

Q4.

Types of Indexes:

- 1.) Primary Index
- 2.) Clustered " "
- 3.) Secondary " "

Key → uniqueness.
Non Key → not unique.

Q5.

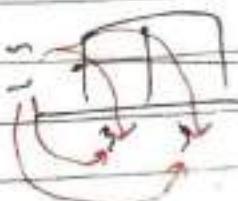
Table:

ordered file	Primary Index	clustered Index	at most 1.
unordered file	Secondary Index	Secondary Index	

key

Non key.

1
2
3
4
5
1
1



- Q6.) 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. _____

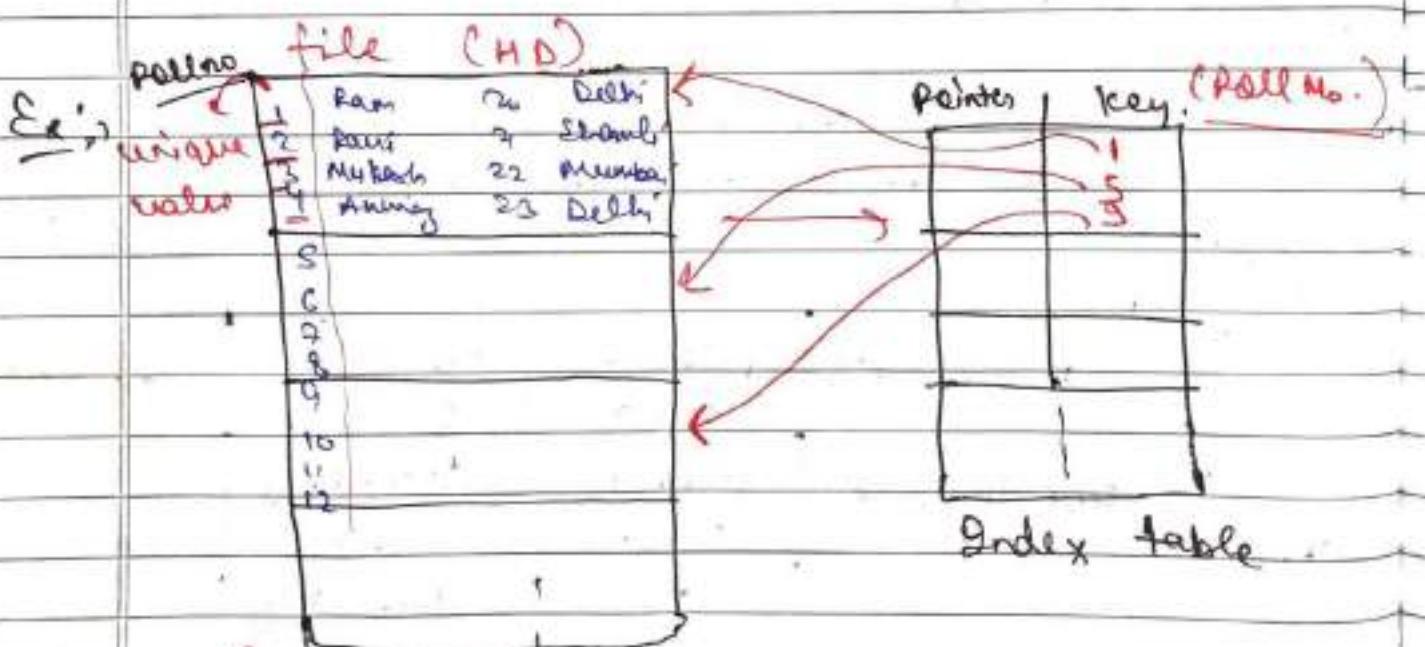


Primary Index:

- ⇒ In a table, if we made any of att. primary keys, then by default a primary index is created on them automatically.
- ⇒ Advantage:-
 - 1. Data is ordered.
 - 2. It is also unique. (key value is present).

Ex:- In IRCTC,

We just find everything, by Train No.



→ 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 (Reader).

(1, 5, 9, 13, ...).

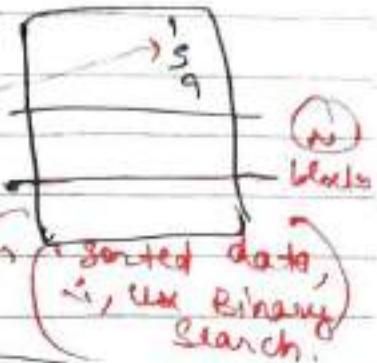
Primary Index is Sparse.

Ex: 1) We have to find 3,

then,

We know 3 is found in
block of '1'.

Hence,



$$\text{Total Search Time} = (\log_2 N + 1).$$

Where,

N is the No. of blocks in Index Table.

35.

Clustered Index : →

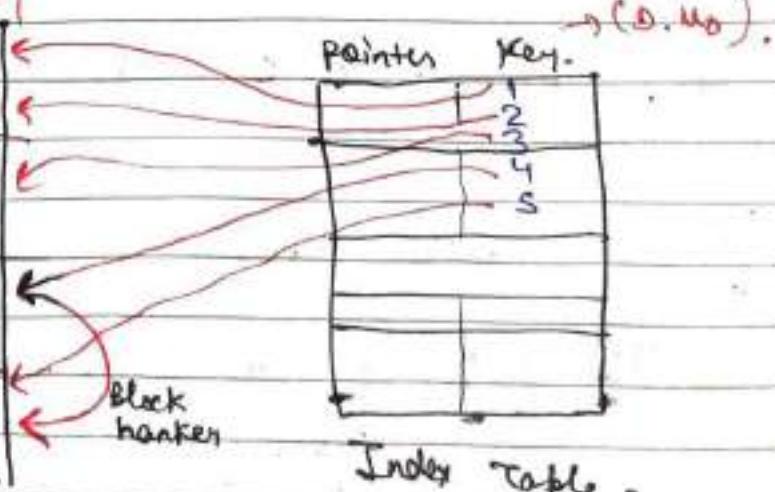
→ data must be ordered & with Non Key,
(i.e., not unique).

i.e.,

→ (value may be repeated but must be sorted.)

→

Non Key	ID/No	Ename	P-No
	1	A	
	1	B	
$B_1 \rightarrow$	2	C	
	2	A	
	3	-	
$B_2 \rightarrow$	3	-	
	3	-	
	4	-	
$B_3 \rightarrow$	4	-	
	4	-	
<i>ordered but not unique!</i>	5	-	



Index Table.

→ If our data is multiple times (not unique) in R.D. then, it also comes only 1 time in Index.

→ (In Index, no repetition).  unique

→ Here, 4 is not only in one block.
Some 4's are also in next block.
So, how we point to them.
So,

Now, we use [Block pointer] to point to next block for same value.

→ Here, searching criteria slightly increased (\uparrow).

→ Clustered Index is also sparse, bcz always we don't need to make pointer for every value. \uparrow Repetition - there is only 1 pointer.

→ So with 2) primary index & cluster index it's still right, bcz no need. \uparrow
1 Table has only 1 primary key \uparrow IT 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.

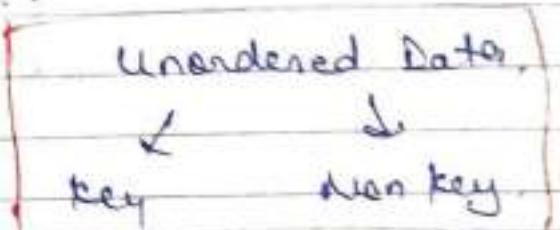
(These extra are \uparrow Block pointers.
for, each block pointer, there is $+1$).

(17)

Secondary Index, (Multilevel Indexing):

→ (Already there is an index in database, we have to make another index).

→ Where use S.I.?



→ Why Another Index?

fails

(E10)

key pointers

key	pointer
1	•
5	•
9	•
13	•

[Employee]

Eid	Name	Panno	H.D.
1	A	40	IT
2	B	S1	
3	A	62	
4	C	33	
5	A	71	
6	D	82	
7	E	B1	
8	F	23	
9	K	100	
10	L	120	
11	M	150	
12	C	136	
13	A	+	
	B	•	

(CPAN no.)

key	pointer
23	•
33	•
40	•
S1	•
62	•
71	•
82	•
•	•

(with key)

[Name]

.	.	.
-	-	-
.	.	.

key	pointer
A	•
B	•
C	•
D	•
E	•

(with non-key)

Intermediate layer
(Block of Record pointers)

Secondary Index (S.I.)

→ Why Another Index? -

→ Let, H.D. has data of Employee table
 ↳ data is already sorted on basis
 of E-id (primary key). → [Unique + Not Null]

↳ bcoz most of the query are on E-id.

→ Find * of Employee where E-id = ...
 ↳ Then we use primary index.
 (which is already tree formed on basis
 of E-id.)

→ But, sometimes we also use
 name & for ex.
 → If primary index fails, or get is
 made on E-id.

So,

we use Secondary Index here.

② Case when we also have key.
 → We use PAN no.

(unique + unsorted)

(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.

Date _____

Page No. _____

41-

* Secondary Index on key, is always DENSE.

bcz, we don't have any anchors (leaders) here.
like in primary key.

→ 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: Key, $N + 1$]

where,

N is the no. of blocks in Index Table.

* Case II When we have Non key with unsorted data.

It is the worst case.

then,

we use NAME,

(or neither ordered, nor key)

(unkey).

In (Secondary Indx in Disk.)

→ In Index, we don't have to take multiple values of A... (Only once).

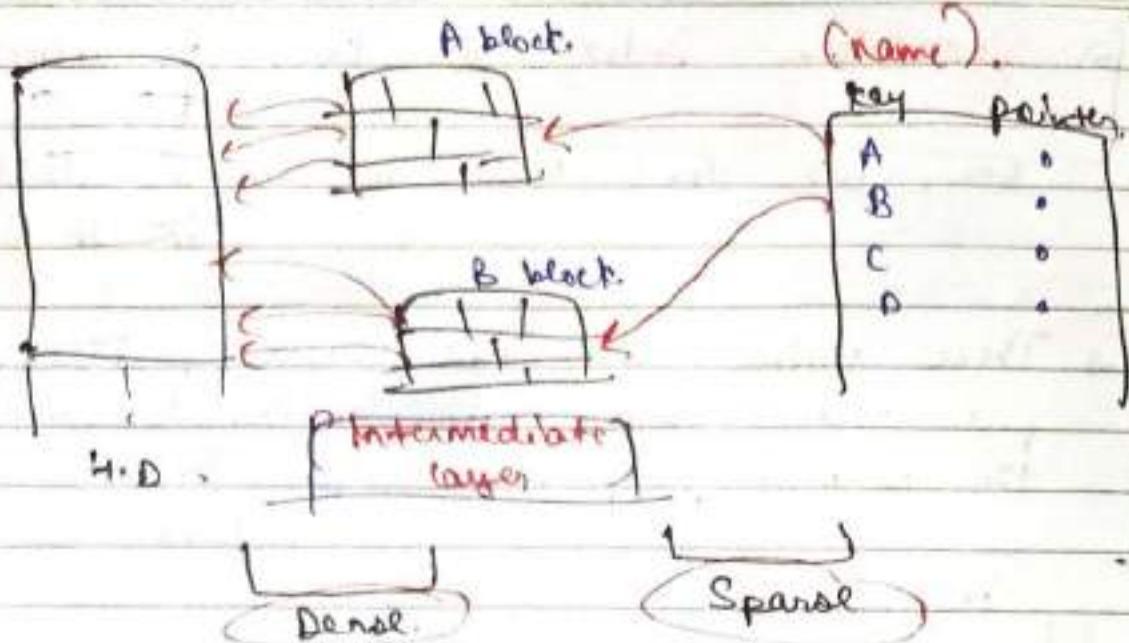
But, A is many times in Hard Disk

so,

here we made an Intermediate layer.

(It is a block of record pointers).

Expl.)



∴ Hence, it is Mix of Dense & Sparse.

So,

∴ We can say it DENSE Overall.

∴ Time Complexity : $\log_2 N + 1 + 1 \dots$

(And this is only for 'N'
It may be more than
 $T \log_2 N + 2$)

([↑] for intermediate layer)

(iii) Secondary Index is always dense.

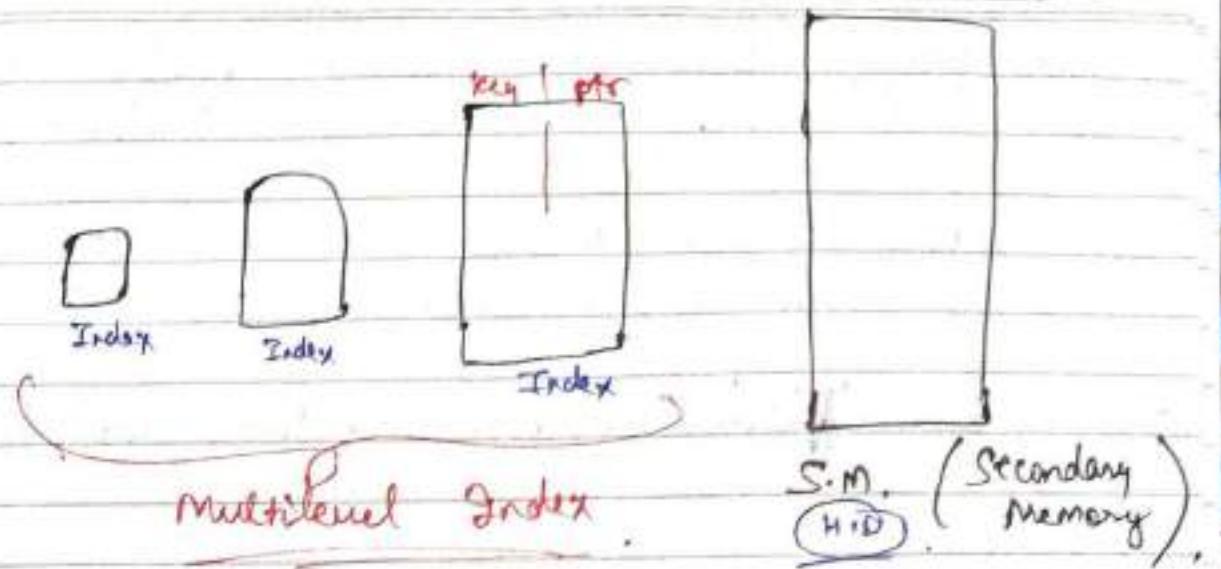
n ————— k —————

(iv) Intro to B-Tree & its Structure :-

B-Tree (Dynamic Multilevel Index)
(Balanced Tree)

graph \rightarrow cycle
Tree \rightarrow Acyclic.

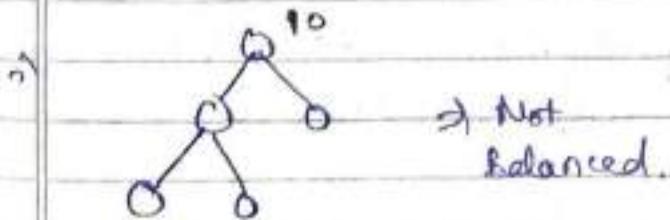
13.



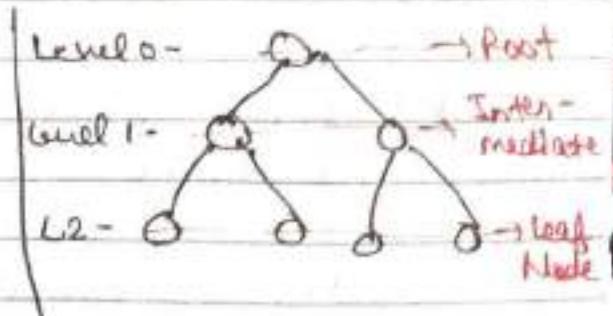
- ⇒ So this, it is hard to manage. b/c.
If we insert data in S.M. then we have
also insert in all the Indexes & same for
delete.

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! \rightarrow
When node denotes his child. Then, we
use Block pointers (B.P.).

- * More! \rightarrow A node can contain multiple values
here:



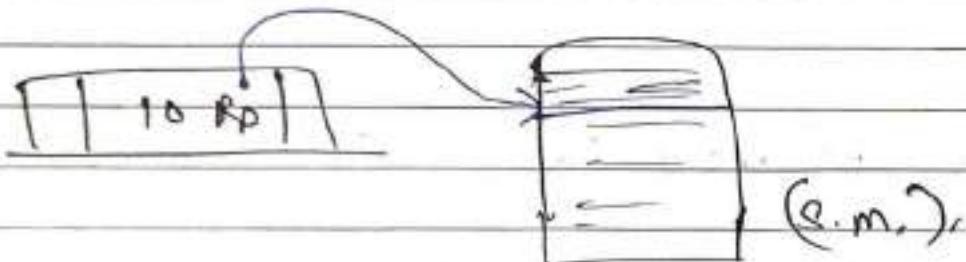
→ Keys: - that on which basis we have to search. Searching criteria - key.

(Key or in Nodes in B-tree (Sect 1)).

(We can insert multiple keys in a node).

→ Data pointer (or) Record pointer (R.P.): → These are which 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 of 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.

$$\text{Keys} = p-1 \quad \text{Max}$$

$$R.P. = p-1$$

$$\text{Min Keys} = \lceil \frac{p}{2} \rceil - 1$$

Table:-

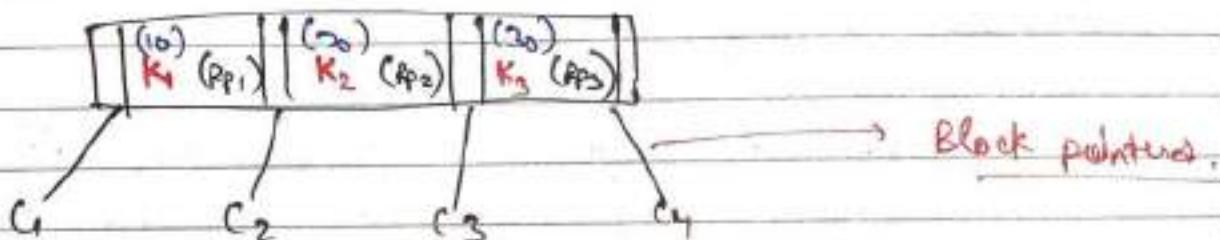
Children of max min	Root	Intermediate Node
	p	p
	2	$\lceil \frac{p}{2} \rceil$

→ Ceiling value

Explan.

Let

$p = 4$ = order of a Tree.



- # We always insert keys in the Sorted Order. (bcz, it follows the prop. of Binary Search Tree).

Q9) Insertion in B-Tree! →

Q1. Insert the following keys into B-Tree, if order of B-Tree = 4.

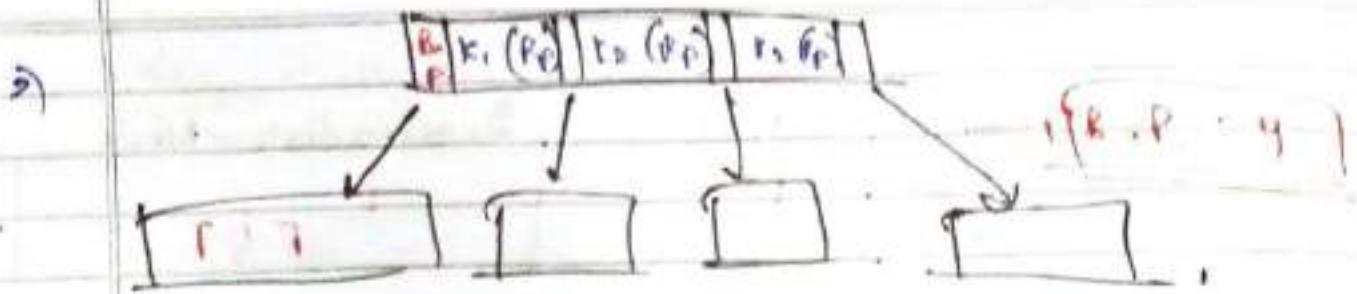
1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

Ans. 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.

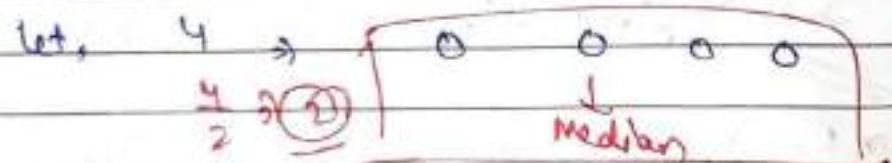
(#) In Binary Search Tree,

$\boxed{\text{Root} \rightarrow \text{Left} = \text{Small Element}}$

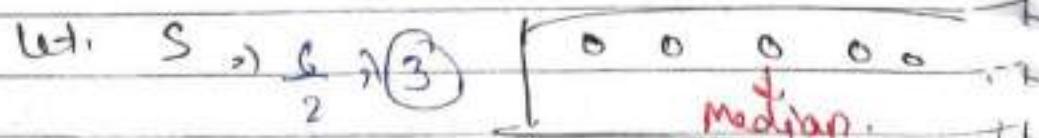
$\boxed{\text{Root} \rightarrow \text{Right} = \text{Big Element}}$

(#) overflow sit & Median find the break point itself,

→ When, $n - \text{Even} \Rightarrow \frac{n}{2} \rightarrow$

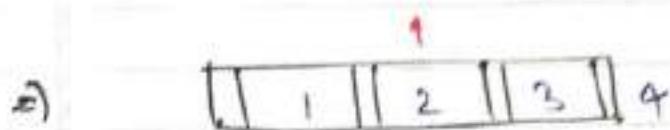


→ When, $n - \text{Odd} \Rightarrow \frac{n+1}{2} = \text{Median}$

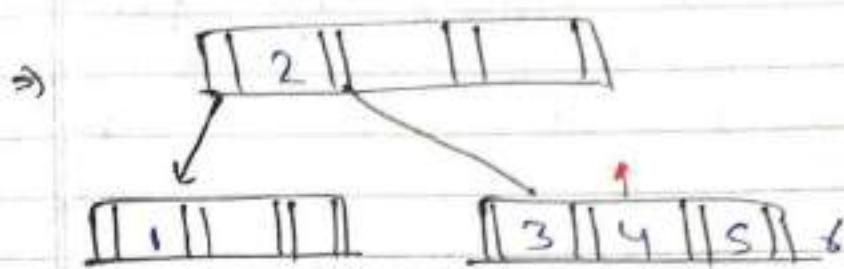


→ and, shift Median to upward (\uparrow), & Both left & Right Elements ~~shift~~ sit till until 1 & ye, now we break.

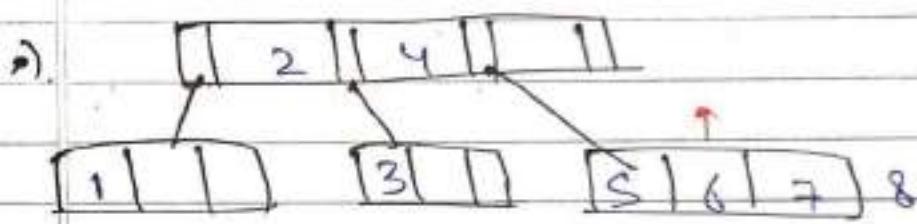
Q. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10



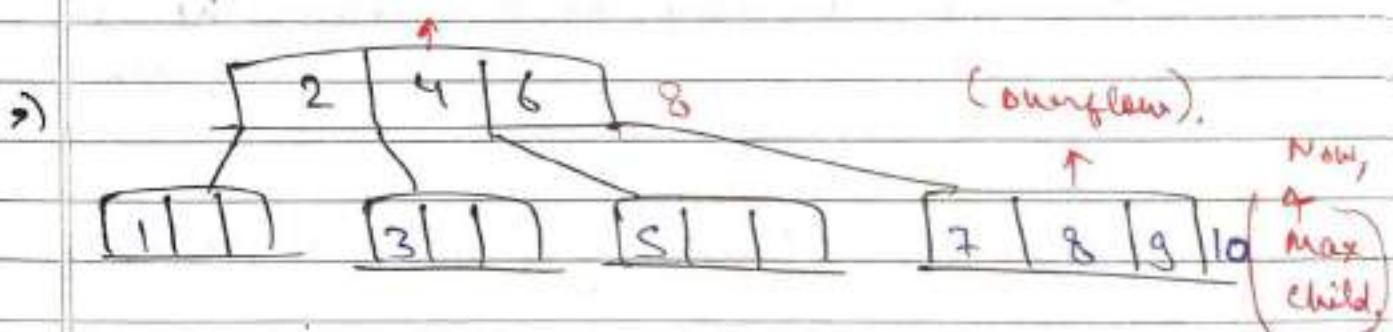
(overflow). $\Rightarrow \frac{1}{2} \Rightarrow \frac{3}{2} \Rightarrow 2$



(overflow).

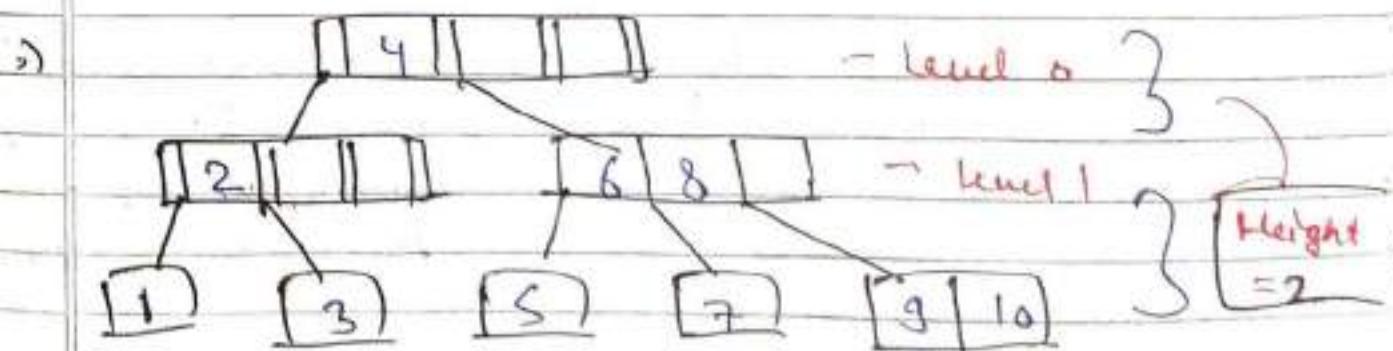


(overflow).



(overflow).

Now,
Max
child.



- level 0 } 3

- level 1 }

Height
= 2

level 2

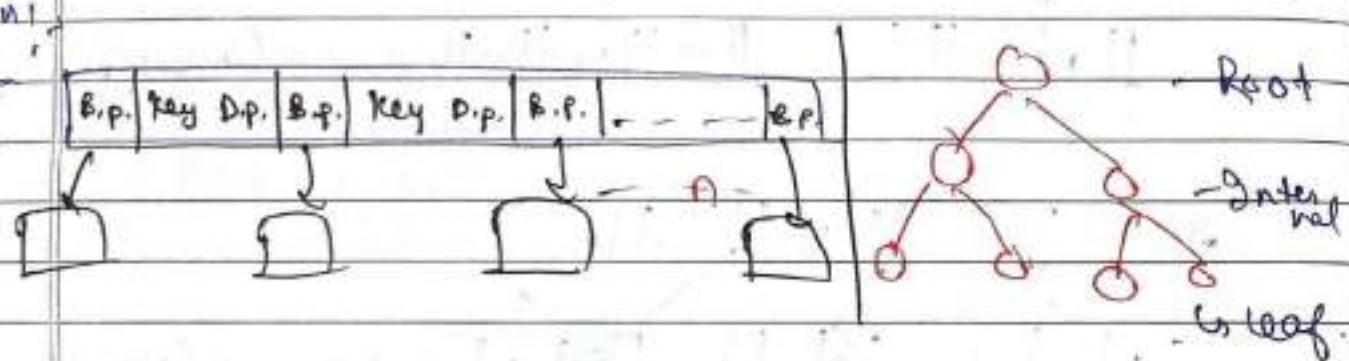
→ weight = 2 }
levels - 3 }

Q. 100.

How to find Order of B-Tree ! →

- Q. Consider a B-tree with key size = 10 bytes, block size = 12 bytes, data pointer is of size 8 bytes and block pointer is 5 bytes. Find the order of B-tree?

Soln:

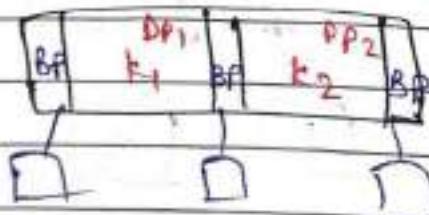


→ Let, In a node / block, has 'n' no. of Block pointers.

$$\text{Total size of B.p.} = n \times (\text{size of 1 B.p.})$$

$$T = n \cdot B_p$$

If n B.p. (or children a node can have) then, $(n-1)$ keys, & 1 record pointer.



$$n \cdot B_p + (n-1) \cdot \text{key size} + (n-1) \cdot R_p \leq \text{block size}$$

(or)
Node size.

$$2) n \times 5 + (n-1)(10+8) \leq 512$$

$$5n + 18n - 18 \leq 512$$

$$23n \leq 530$$

$$\boxed{n \leq \frac{530}{23}}$$

$$n \leq 23.04$$

$$n=23$$

- Max. 23 children.

→ Every F.P. represent 9 children.

SD,

Max. Order = 23.

col

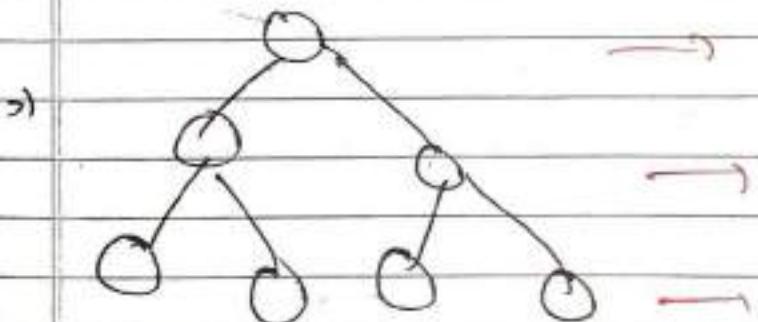
Max | Min Children

23 | 2

23 | $\lceil \frac{23}{2} \rceil \Rightarrow [11, 12]$

* 12

leaf has no children.



- Keys = Order - 1

22

col

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.

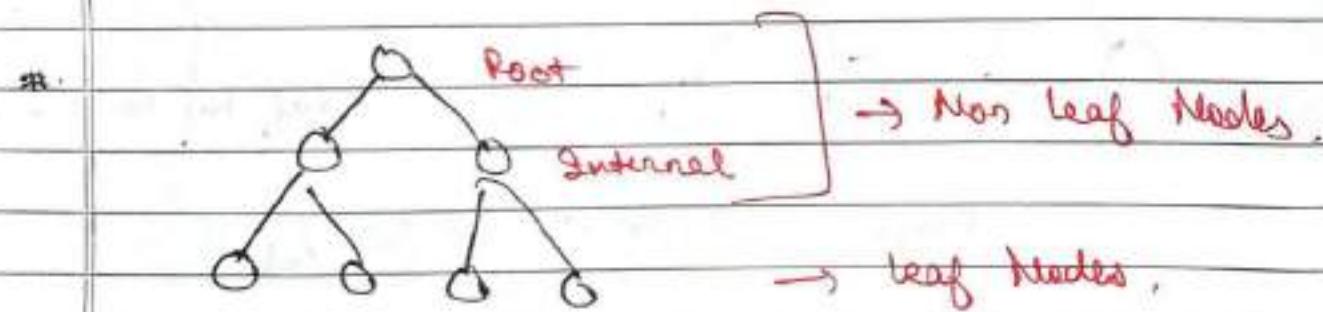
- 3.). No Redundant (Duplicate) Search key present.
- 4.) Leaf Nodes not linked together.

② B⁺ Tree's

- 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**.

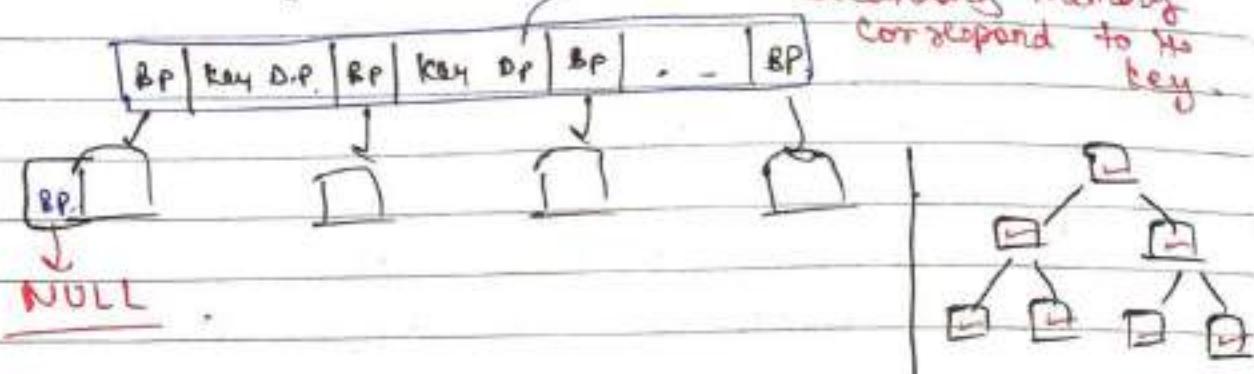
* **Index Record**  **Key** **Data pointers (Record pointer)**,



- * 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.

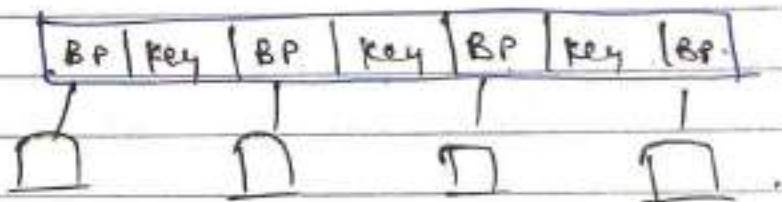
B.P - Block pointers.

Structure of B tree,



Structure of B+ tree :-

→ [Non leaf structure] → or Internal Node →.



→ There is no Data pointers (D.P.) in Internal Node structure / Non-leaf Node structure.

R.P / D.P - ~~(X)~~

Hence,

We have more space in Internal nodes.
Hence, we can create more children & put more keys.

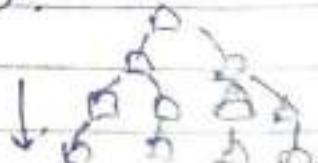
So, that's why,

B+ tree → Breadthwise longer.

B tree → Depthwise longer.

↳ (as, less no. of children breadthwise as compared to the B+ tree).

So, depth more.



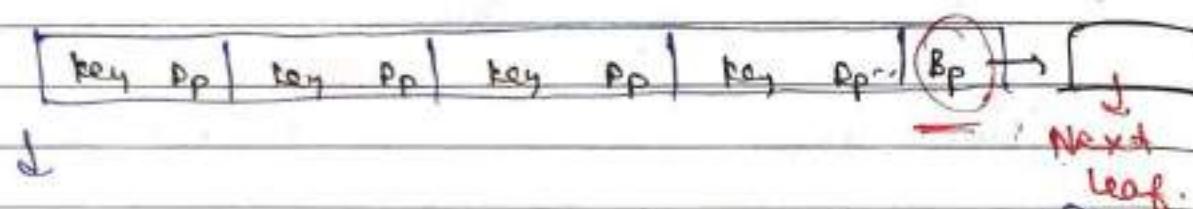


#. Q8, that's why searching is slower in B tree as compared to B+ tree.

#. B⁺ tree Structure →

[Leaf Node Structure] →

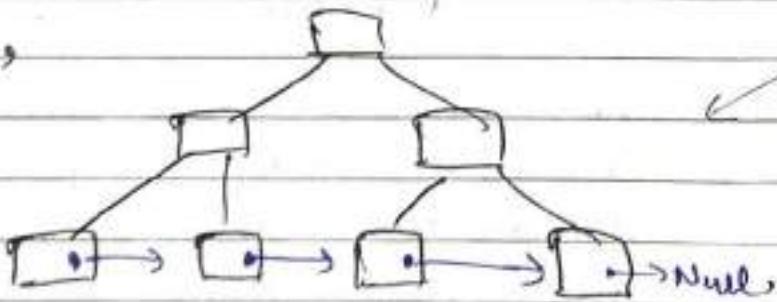
(Structure of Leaf & Non Leaf Nodes
are different.)



⇒ (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,

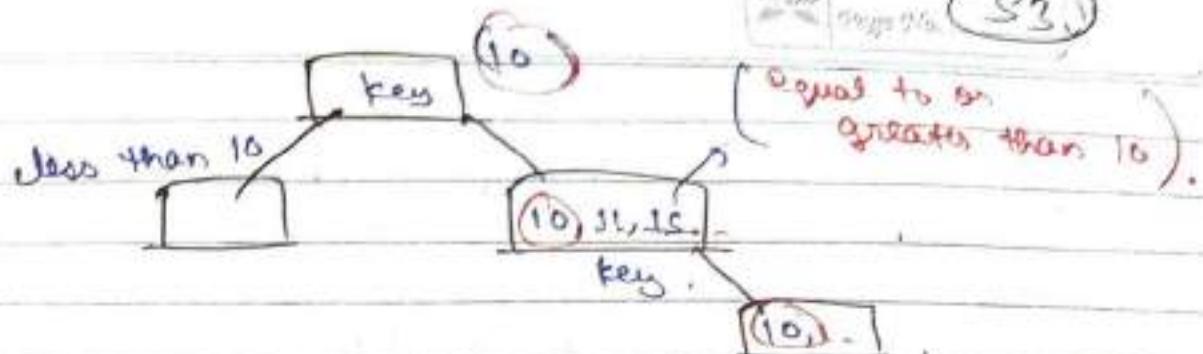


⇒ In this, ~~less~~ value → its key is in left node & the greater value to key one in right node. (But here, in right side, we also have to put the key ~~value~~ with its greater value).

, i.e.,

Date _____
Page No. _____

(S3)



Reason: → bcz, we only search in leaf Node, bcz Only leaf Node has all keys with Data pointers are present.

Q2:

we also have to search the D.P. of the key, so, that's why we also carry the key value upto leaf Node.

- Searching is that's very faster in B+ tree, bcz have to search only in leaf Node.

2

bcz of this, Redundant keys are also present in B+ tree.

Leaf Nodes are also linked together.



(102.)

Ques: on Order of B+ Tree : →

Q1: Consider a B+ Tree with key size = 10 bytes, block size = 512 bytes, data pointer = 8 bytes & block pointer = 3 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 | Key DP,]



→ Non-leaf: →

$$n \times B_p + (n-1) \times \text{key} \leq \text{Block Size}$$

$$\rightarrow S \times n + (n-1)_{10} \leq S_{12}$$

$$S_n + 10n_{10} \leq S_{12}$$

$$15n \leq 522$$

$$n \leq \frac{522}{18} 34.8$$

$$\boxed{n \leq 34.8}$$

$$\boxed{n = 34 \text{ th}}$$

$$\boxed{\text{order} = 34}$$

[(Max BP.) as (Max children possible)].

→ Leaf: →

Let

x pairs

$$x(\text{key} + \text{pp}) + \text{BP} \leq \text{Block Size}$$

$$x(10+8) + S \leq S_{12}$$

$$18x \leq S_{07}$$

$$x \leq \frac{S_{07}}{18} \boxed{28.18}$$

$$\boxed{x \leq 28.18}$$

$$\boxed{x = 28}$$

$$\boxed{\text{order} = 28}$$

Note: Order of a leaf Node in B+ tree
is the no. of (key, p.p.) pairs.

103.

Immediate Database Modification : → (Log Based Recovery Methods),

⇒ Immediate means ~~3rd~~, ~~4th~~ CH 1.

Sohail

$$\begin{array}{|c|c|} \hline A & 100 \\ \hline B & 200 \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline 200 \\ \hline 400 \\ \hline \end{array}$$

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 >

Tredo

⇒ यदि हम Ram में अंतर्गत value (200) देते हैं, तो वह Time पर ही Database से अंतर्गत ($A=200$) करवता है। Commit करता है।

⇒ i.e., 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 was both start & commit or not. If yes, then it Tredo.

→ Redo, means saves the latest value in the database. i.e.

→ Recovery Manager don't see value in the H.D., it only sees in the Trans. log & checks (starts & Commit) & then saves in the H.D. database. & if already saved, then over-write & fixed them.

→ But, If

		Transac' log	
Database		T ₁	
A = 100	B = 200	R(A)	< T ₁ , start >
		A = A + 100	< T ₁ , A, 100, 200 >
		W(A) - 200	old new
		R(B)	< T ₁ , B, 200, 400 >
		B = B + 200	
		W(B) - 400	

* fail.

→ 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 & saves them in the Database.

- 3 In Immediate, we store both old & new value.
- 4 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.

Extn → Transaction log →

$\langle T_1, \text{start} \rangle$
 $\langle T_1, A, 1000, 2000 \rangle$
 $\langle T_1, B, 5000, 6000 \rangle$
 $\langle T_1, \text{Commit} \rangle$

REDO

$\langle T_2, \text{start} \rangle$
 $\langle T_2, C, 700, 800 \rangle$
 old.

UNDO

Ques on DBMS basic Concepts & Data Modelling:

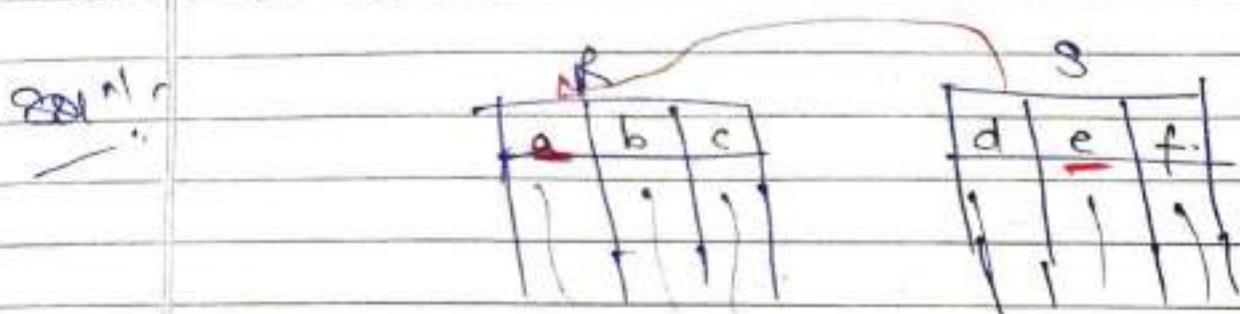
Q:- Let, R (a,b,c) and S (d,e,f) be 2 rel's.
'd' is foreign key of S that refers its primary key of R. Consider 4 operations on 'R' & 'S'

- 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) i & iv~~ → ~~Ans.~~
- d) i & iv.



→ Understand from ~~intelligent student~~ ~~if they don't delete then it's not fine~~ ~~then what will happen~~ ~~if they don't delete then what will happen~~
 ∵ (If they don't delete → then, violation.)

2. The view of total database content is

- ~~a) Conceptual view~~
- ~~b) Internal view~~
- ~~c) External view~~
- ~~d) physical view.~~

→ This is on 3 schema architecture.

* Internal view \rightarrow External view,

External view is basically related to outer view.

*) User ~~not~~ Total view ~~not~~ ~~not~~.

User has partial view (data only he need).

→ physical view at ~~next~~ 3rd level view
(Database storage Related).

→ In partial case, External view is Ans.

Q3. 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.

Q4. 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, selected with primary key.

→ key deals with uniqueness.



Ques. Comp. Ques' on Advance DBMS :-

(Big Data & Data Warehouse)

Q1. What do data warehouse support?

- a) OLAP
- b) OLTP
- c) OLAP & OLTP
- d) spatial database.

Ans: Data Warehouse, where we integrate data at one place from all diff sources.

Ex: Big BAZAR, - 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 AD's - to diff users acc. to their data of purchasing items.

(Apply Mining Algorithms on these data)

→ OLAP → Online Analytical processing.

→ OLTP → Online Transaction processing.

→ works on current data.

→ (that where to remove item, when open new store, remove or add items) —
all these based on Analysis.

2. Hadoop is framework that works with variety of related tasks. Common ones include —

- ~~a) Map Reduce, Hive & Hbase~~
- b) Map Reduce, MySQL and Google Apps
- c) Map Reduce, HDFS, Pig
- D. Map Reduce, HDFS, TruSight

→ Hadoop basically work on Big Data.
(It is a tool of big data).

↳ Use Google, Facebook. Big data deals with the multiple petabytes of the data. Now, to process this much of data, we don't use normal tools like SQL Server, Oracle. We use Hadoop. (as, 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.

Q) Google Apps, it is part of cloud.

Q.) All of the following accurately describe Hadoop, except:

- A) open Source → (listed in Apache, install it).
- B) Real Time ~~N~~.
- 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).

Q.) Which of the following does not comes under five V's of Big Data?

- V) Volume (how much amount of data).
- V) Velocity (with which velocity, data ↑).
- V) Variety (Structured, Unstructured, Semi-Structured).
- V) Visualization.

To Note:- At, 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, trustworthiness).
- ↓
(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.

→ Visualizing, 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 latter 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 \rightarrow 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 Modification
 - 2.) Immediate

(24)

		Transac Log	
$A = 100$	200	T_1	
$B = 300$	400	$R(A)$	$\langle T_1, \text{Start} \rangle$
Database (H.O.)		$A = A + 100$	$\langle T_1, A, 200 \rangle$
		$W(A) - 200$	<u>new value</u>
		$R(B)$	$\langle T_1, B, 400 \rangle$
		$B = B + 200$	<u>new value</u>
		$W(B) - 400$	$\langle T_1, \text{Commit} \rangle$
		Commit	<u>Redo</u>

→ Here, It don't update in Database hand-to-hand. It updates in Database after Commit. (by, deferred-late, postponed).

After commit, database updated.

- How use Recovery in Deferred? (if, say system fails after Commit.)
 - So, when Recovery Manager comes, it first check trans. log file & check (abort \rightarrow Commit) in it.

then, Recovery Manager.

REDO

- Means, update the new values in the database.

due to

→ (let, "fail → our database also not there").
then,

Recovery Manager gets the A & B value
from Log in database.

$$A = 200$$

$$B = 400$$

→ (If it already in database, then overwrite
it). **WRAPUP**

Case II

$$\begin{cases} A = 100 \\ B = 200 \end{cases}$$

T₁

w(A) - 200

w(B) - 400

* fail.

"Transac" log.

<T₁, start>

<T₁, A, 200>

<T₁, B, 400>

Roll back.

- Now, fails before Commit. So, now
database value is same as before.

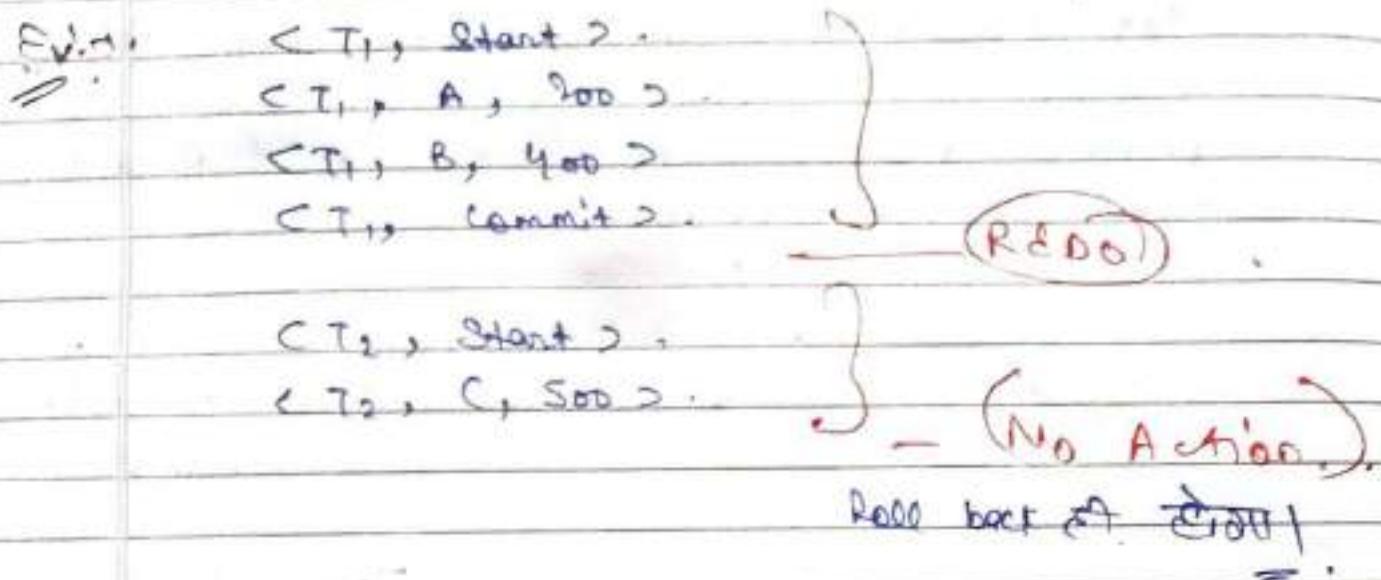
Now,

After failure occur, when Recovery Manager
opens the log file. → It sees T₁ starts
but not commit. So, here Recovery
Manager don't do anything. He simply
Roll back.

Means, get value out of var, and all
Open's out of mind till t'.



→ So, Deferred Modification also known as No UNDO/REDO method.



ToT like Command in SQL.

→ We use Like Command, generally to search the Data.

- 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 'o' in 2nd place.
- 6.) name should contain total five characters.

'%' → Any value (of any character).
In with 'o' character is also, it's problem will be [].

(6.2)

Emp.

ID	Name
1.	Varun
2.	Arun
3.	Karan
4.	Ankit
5.	Ranjeet
6.	Ajeet

'%' → any value
length.
- → reserved for
a value.

1.) Select * from Emp where name like 'A%';
Output → Arun, Ankit, Ajeet

2.) --- like '%n%';
Output → Varun, Karan.

3.) --- like '%ee%';
Output → Ranjeet, Ajeet

Note:- If Name → Sar, Arne, — anything
then, these also comes, bcz size ('ee') we
need.
'%' → also 'o' character include.

4.) --- like '_ay.';
Output → Varun, Karan, Ranjeet.

5.) --- like 'a____';
Output → Varun.

a

a



108.

Basic PL-SQL programming with Execution :-

→ Program 1: Find the Sum of 2 numbers.

```
→ declare
  a int ;
  b int ;
  c int ;
```

Event
able
part

```
begin
  a := b a ;
  b := b b ;
  c := a+b ;
```

" q
// value given by user
(to take input from user).

```
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 := b a ;
b := b b ;
```

```
if (a>b)
```

```
then
```

```
dbms_output.put_line ('a is greater'); //a
else
```

```
dbms_output.put_line ('b is greater'); //b
```

SQL line से हमें value पहले ही देनी पड़ती है तो लाइन में user value नहीं दे सकता।

69

end if;
end;

(both code works fine).

X X —————— X —————— X —————— X —————— X

Q3. 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 1.

program 4:-

while loop :-

```
→ declare  
    a int;  
    b int;  
begin  
    a := 0;  
    b := 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

↳ If stack sometimes will 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 output corresponding to that row. → then, It is Single Row func.

⇒ Multi-Row :- func.

func. that apply on more than one row,

→ Gives an output corresponding to all these rows.

⇒ Round

(Round-off value)

⇒ Mod.

(Gives remainder after division)

⇒ Lower

(Convert a string into lower letters)

⇒ InitCap.

(Capitalize the initial letter)

⇒ Concat

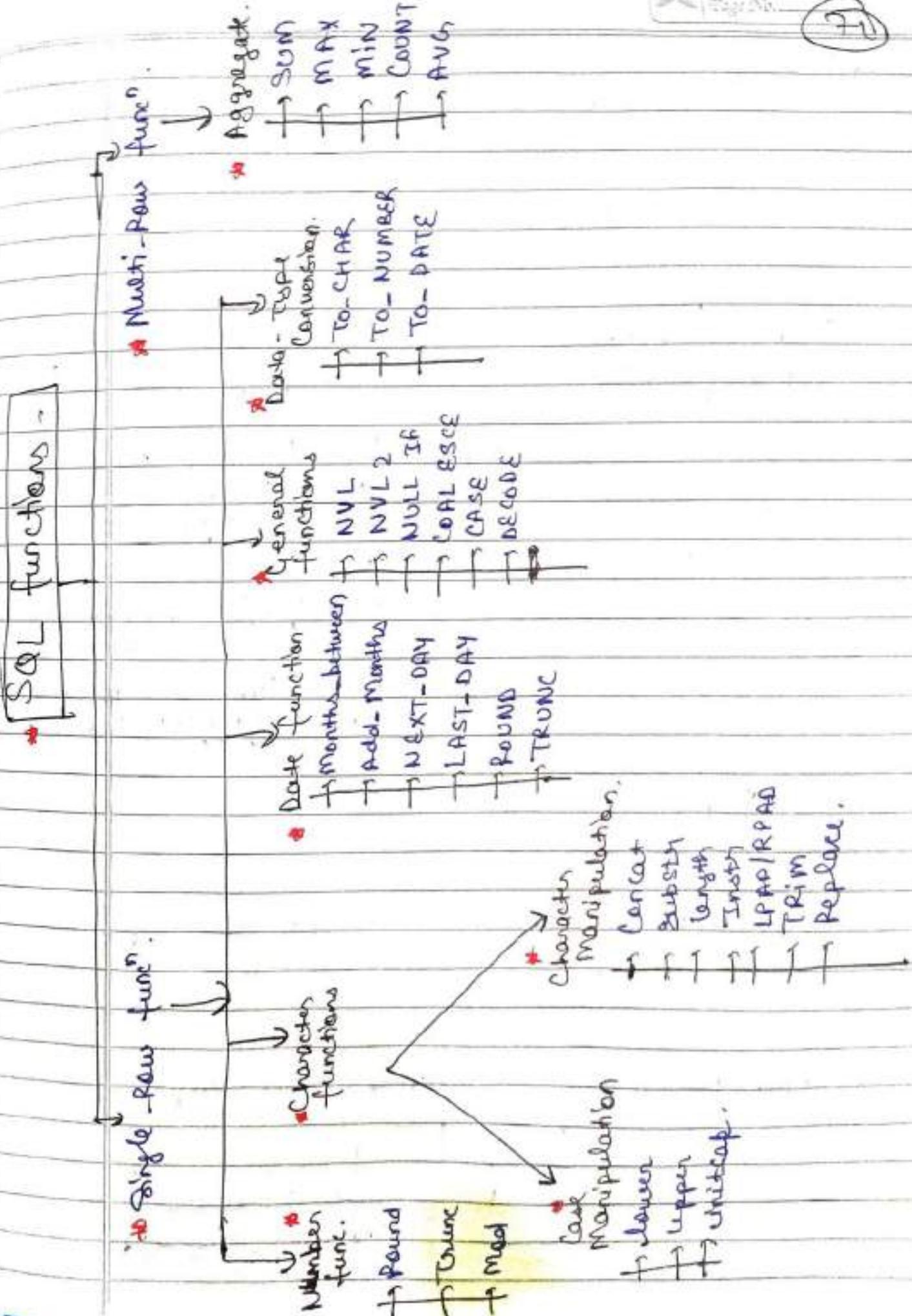
(Add 2 string & make 1)

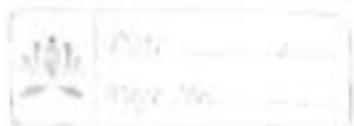
⇒ LPAD/RPAD

(for left & right padding)

⇒ NVL, NVL2

(to take care of NULL values)





III. Character functions in SQL with

Execution : →

→

Character functions

See function
Manipulation

Character
Manipulation

- Lower
- Upper
- Init Cap.

- Concat ('Varun', 'single') Varun single
- Substr ('Varun', 2, 4) anu
- Instr ('Varun', 'u') 4th
- length ('Varun') 5
- LPAD ('Varun', 10, '*') *****Varun
- RPAD — " — (10 - length)
- TRIM ('V' from 'Varun') Arun
- REPLACE ('Varun', 'V', 'T') Tarun

④

ANURAG
1 2 3 4 5 6

④

Execution → If we want to implement these func., so, first we have to make schema (table).

→ Output Table → Select * from emp;

ID	NAME
1	Create Smashers
2	Varun Single

⇒ Create table emp
(
id int,
name varchar(20))

);
insert into emp values (1, 'GATE SMASHERS'); || 2 rows
insert into emp values (2, 'Varun Single'); || 1 row.

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;]

↓
Output:-

REPLACE(NAME, 'V', 'T').

GATE SMASHERS

Varun Single.

↳ Output:-

LTRIM(NAME, 'V')

** GATE-SMASHERS

*** Varun_Singla.

Here, it counts (-)
Space also.

Output:-

SUBSTR(NAME, 2, 5)	INSTR(NAME, 'V')
ate S	0
Varun	1

↳ It don't count (-) here).

CREATE	SMASHERS
1 2 3 4	5 6 7 8 9 10 11 12

→ ate S

(12)

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.)

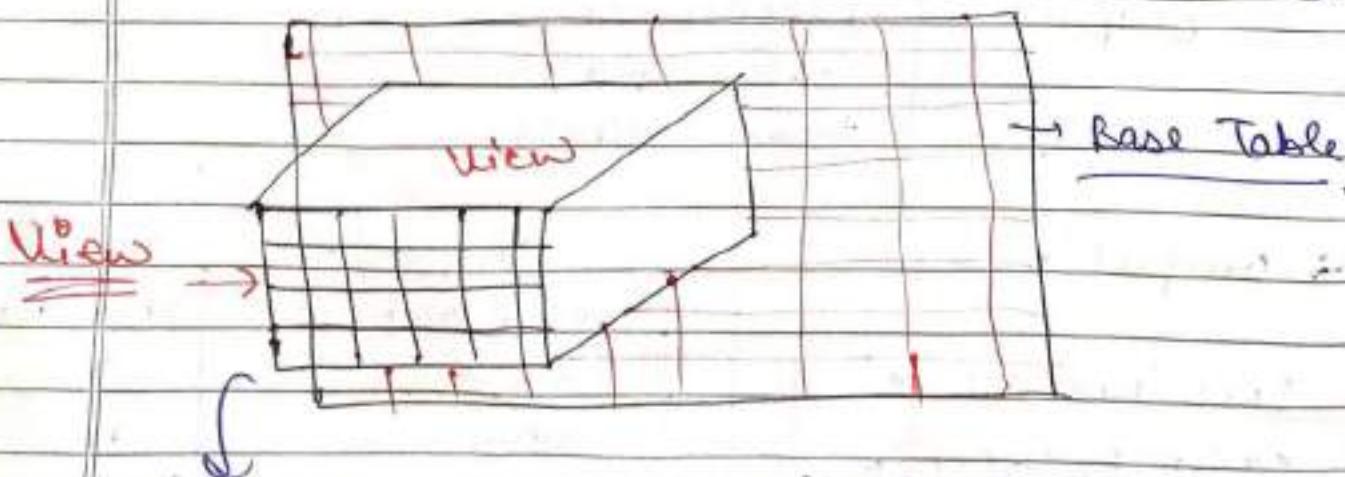
⇒ But, View is the Virtual Table.

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 V, as Select id from Student;



This view has no physical existence,
we don't store it anywhere.

VI - View

23

- (#) 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 (No) updatable Views! →

→ If we made any changes in Base Table
↳ that col^m 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) DDL 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.

3. Materialized View:-

→ type of updated version.

(If our data is on the remote server,
→ I want a copy of that on my local server / machine. i.e., 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.

i.e.

View can also take data from **Multiple Tables**.

We also can take any particular row from Table to make view, like,

→ where address = 'Delhi';
→ all Delhi students come into the view.

Advantages of View :-

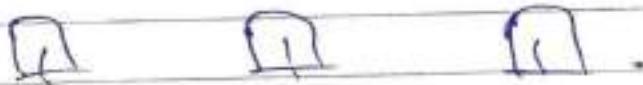
1) To restrict the Data Access.

(Here, 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.

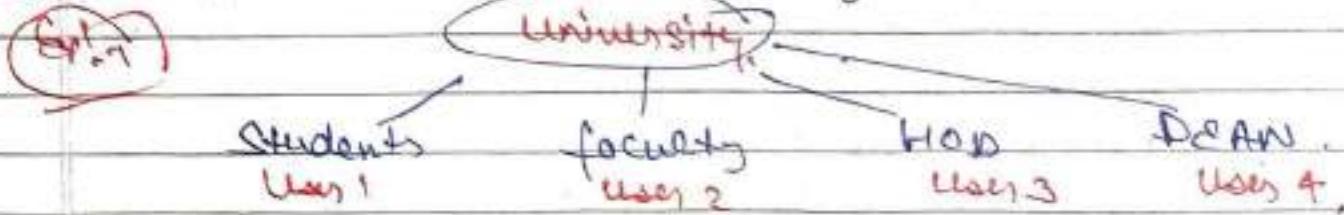
So,

It's better to just make the view from these tables. Rather, than to write complex query.

3.) To provide data independence,

(We just 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.



(They all have diff. privileges).

So,

→ We give diff. view to all of them of the same Data (Table).

X X

① SQL CHEAT SHEET :→

② Examples →

1.) Select all rows from table with filter applied

→ Select * From tbl where Col1 > 5;

2.) Select first 10 rows for 2 columns

Select Col1, Col2 fromm tbl limit 10;

3.) Select all rows with multiple filters applied

→ Select * froms tbl where Col1 > 5 AND Col2 < 2;

4.) Select all rows from col1 and col2 ordering by col1

→ Select col1, col2 froms tbl order By 1;

5.) Return count of rows in table

→ Select Count (*) froms tbl ;

6.) Return sum of col1

→ Select SUM (col1) froms tbl ;

7.) Returns max value from col1

→ Select MAX (col1) froms tbl ;

8.) Computer Summary Statistics by grouping col2

→ Select AVG (col1) froms table Group By col2;

a) 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 NewColumnName
FROM tbl;

(*) ORDER OF EXECUTION →

FROM
WHERE
GROUP BY
HAVING
SELECT
ORDER BY
LIMIT

④ 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;

X

OK