Jai Shree Ram.

Name : Anurag Singhal

INDEX

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

Subject : DBMS

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

(Live SQL).

E F codd → Father of DBMS

Date: Page:

(1) Database Management System (DBMS) :→

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

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

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

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

Attributes & its types,
Relationship — " — Imp.

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

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

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

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

→ SQL & Relational algebra.

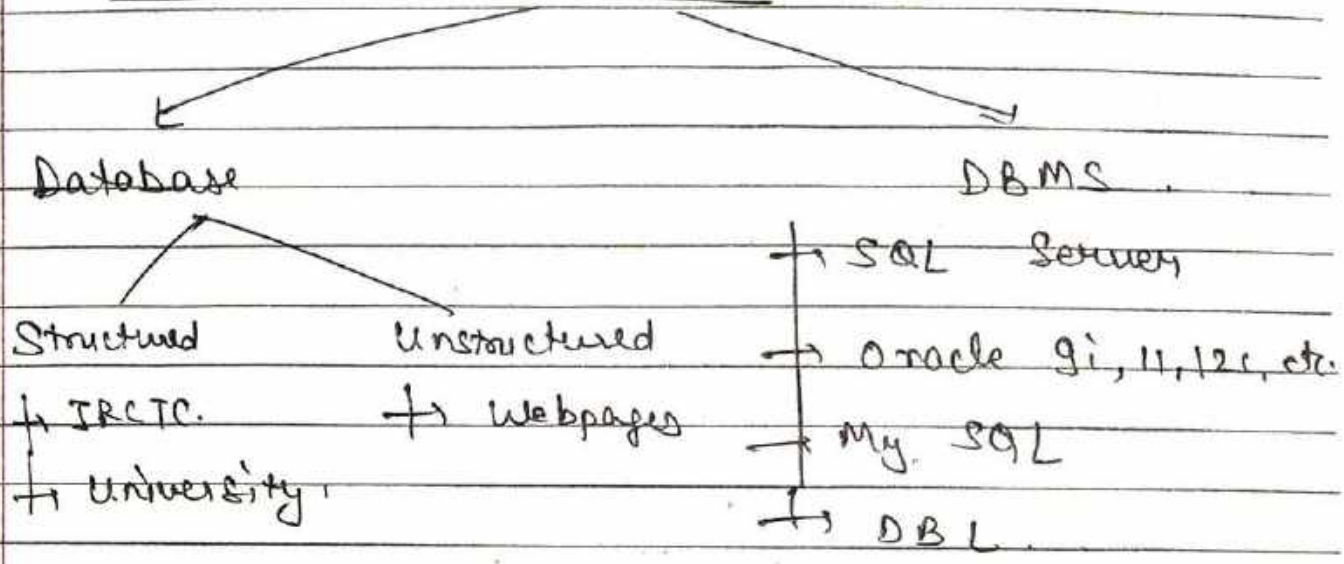
SQL → Structured Query language.
(It is a programming lang.).

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

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

→ B tree, B^+ tree. (in Multilevel).

2) Database System :→

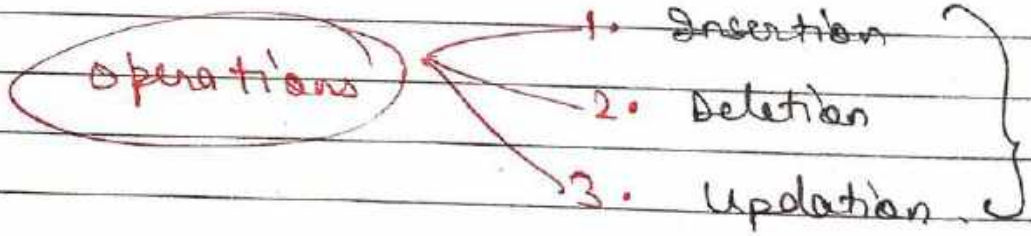


⇒ Database :→ Collec" of Related Data.

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

⇒ Structured :→
RDBMS → Rela"al Database Management Sys".

⇒ DBMS :→ collection of Operations



It provide easyness to perform opera"s.

⇒ Diff. Companies have made diff. DBMS.

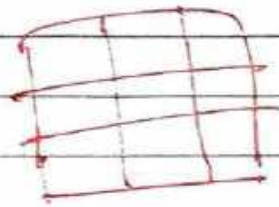
Ex:-
Microsoft Co. ⇒ SQL Server
(Sequal Server).

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

⇒ IBM ⇒. DB2.

(#) Structured Data! → RDBMS
↳ Relation

Mean,
We stored in the table form.


(Table).

Now, Table is technically called as Rela".

→ Rela" is most usuable form.

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

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

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

(#) Unstructured Data! → There is not predefined Structure Ex! →
A webpage is a collec" of photos, videos, Chats, etc. i.e.
There is not a particular format in these.

⇒ Hence, RDBMS works only on the structured Data.

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

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

---

3. File System Vs DBMS :→

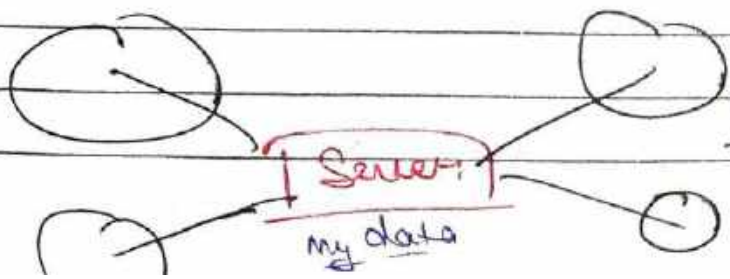
⇒ File System is used before DBMS.

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

Ex:→ CIFS, NFS File Systems.

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

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



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

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

(Searching is fast & Memory utilize is Efficient).

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

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

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

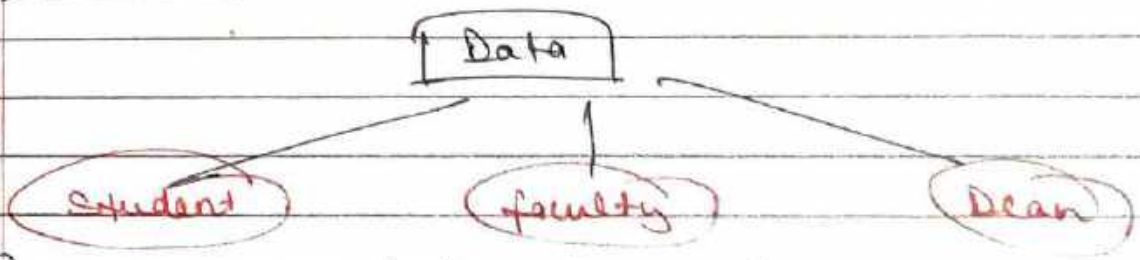
3.) Concurrency :→ means Concurrent Access. Multiple persons can access the data at the same time.

Ex:- IRCTC (Indian Railway System)

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

RR
RW
WR
WW
(Read - write)

4.) Security :→ Role based security



(Role based Access Control.)

If we are users, we only get user data.
" - faculty, " - faculty
(O.S.)
File system, has no security for this.
No level-by-level. No Hierarchical.
DBMS has this role-based security.

5.) Data Redundancy :→ (Duplicacy).

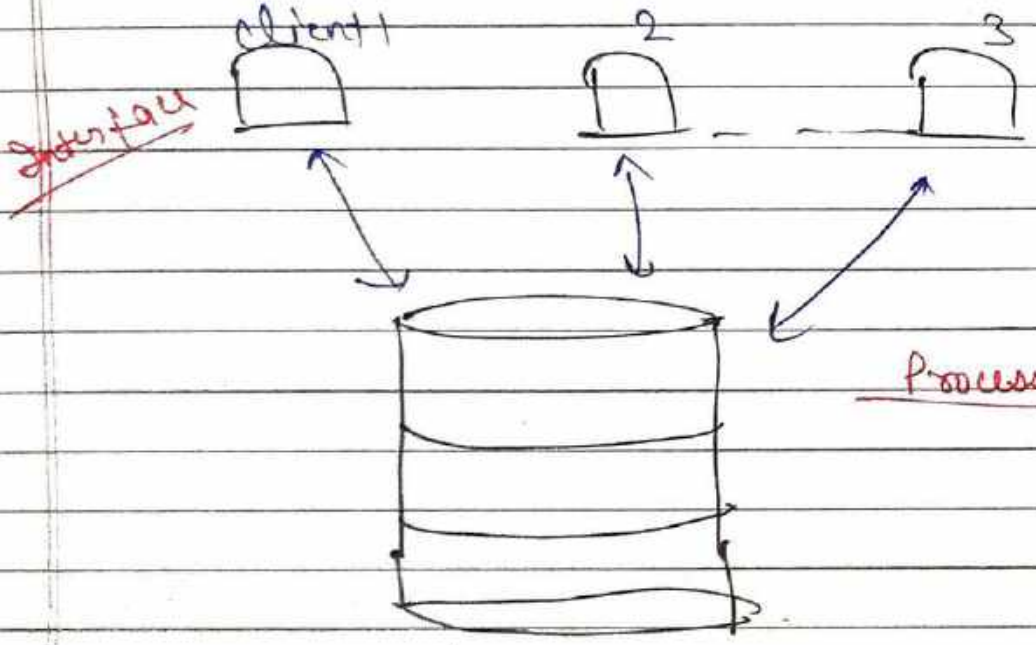
In file system, we can save same content by diff file names.
In DBMS, has many constraints to ensure unique data. Stops redundancy of data.

→ DBMS is at back End of Every client Server & web Applica".

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

(I) 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 :- If we reserve ticket by going to railway Stn at ticket window.

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

(client → Request → Database Server → gives info)

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

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

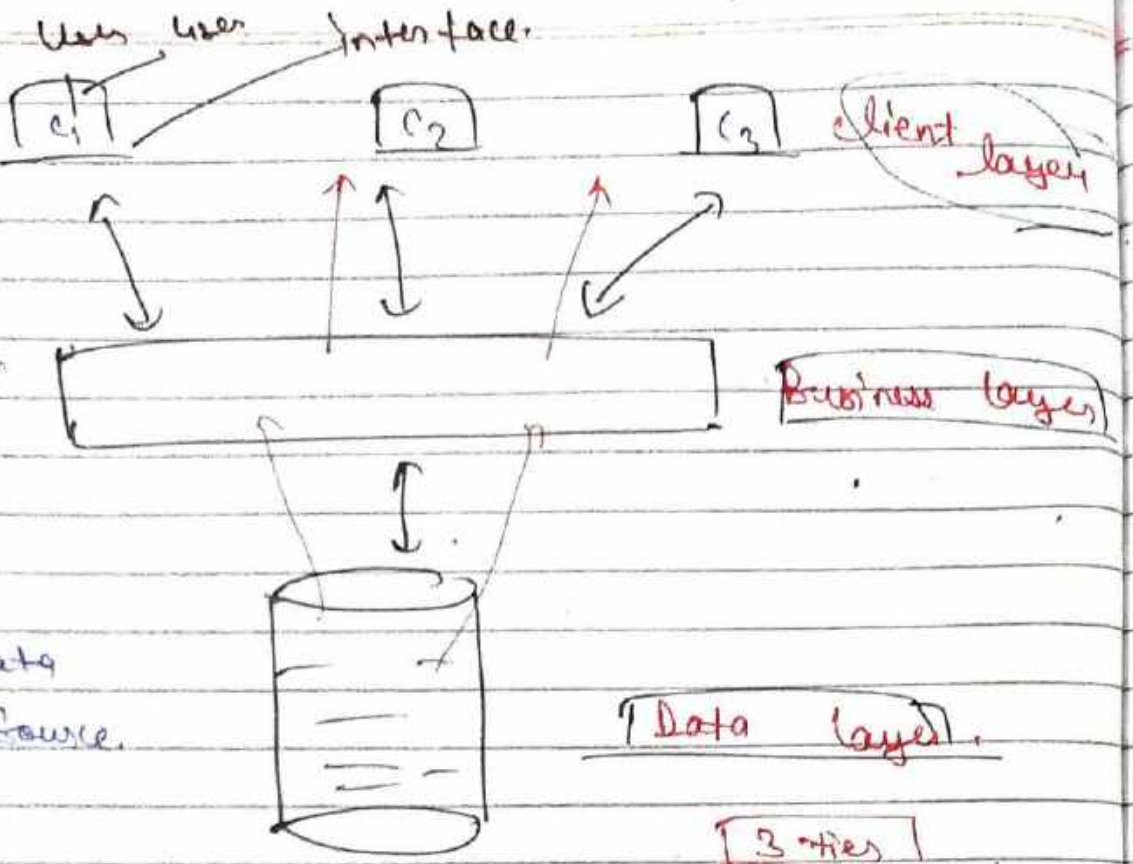
Advantage :- Maintenance is Easy.

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

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

Ex:- IRCTC & banking app. & Gmail app

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



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

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

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

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

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

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

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

Roll. No.	name	addresses.

✓ Relationships.

ii) Course :- (Schema, Entity).

C.. ID	name	Dura"

✓ (Relationship)

↳ Logical Representa" (structure).

↳ But, we implement it by SQL.
(Integers, Characters - - -). (Sequeal).
Data Defi" language (DDL) → to implement Schema. (Or to design)

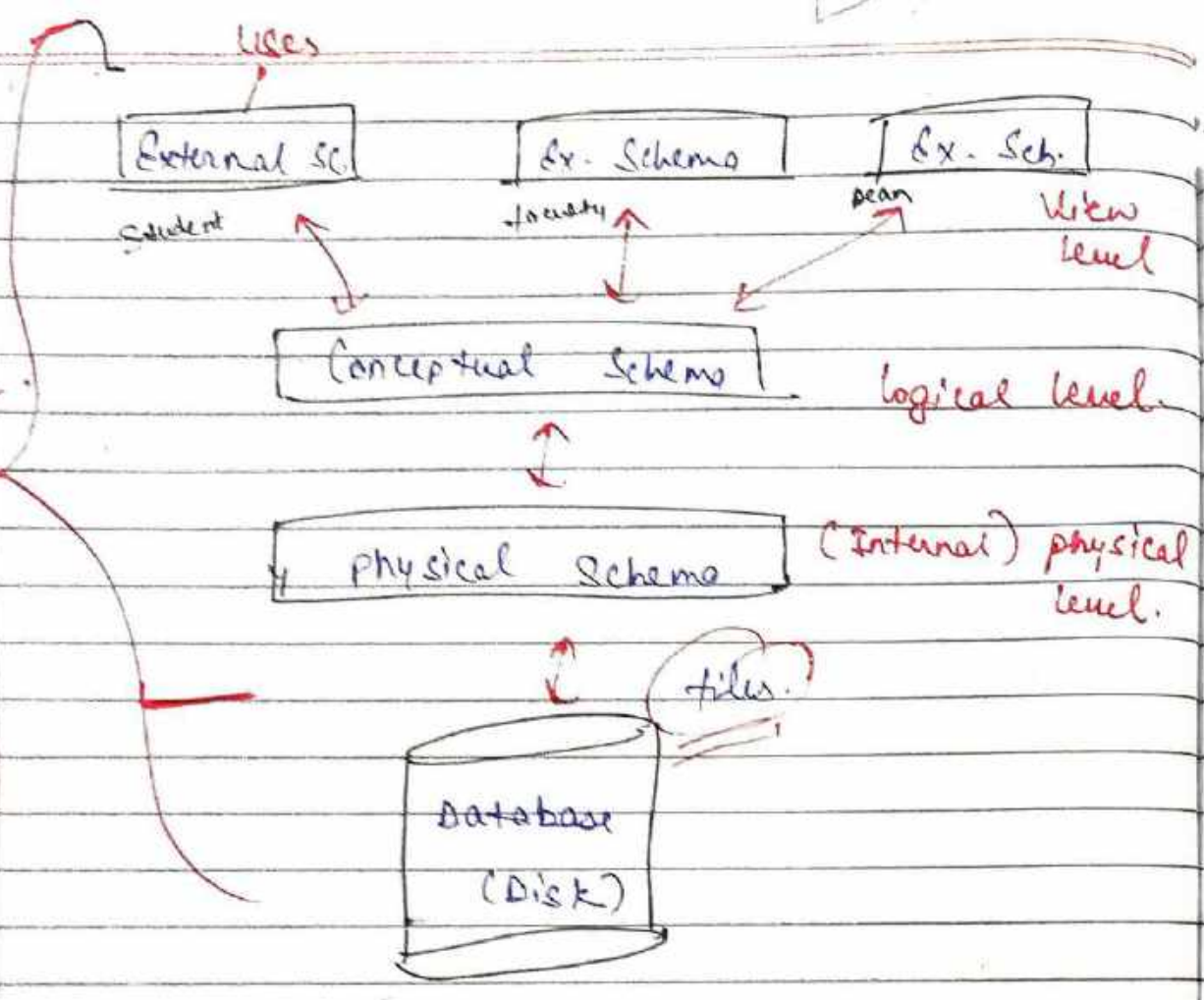
Schema is simply a structure (table form).

---

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

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

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



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

Students have their View.
Faculty have —"— .

Ex:→ After login, which view comes to us is external Schema.

Conceptual Sch: E-R Model

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

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

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

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

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

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

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

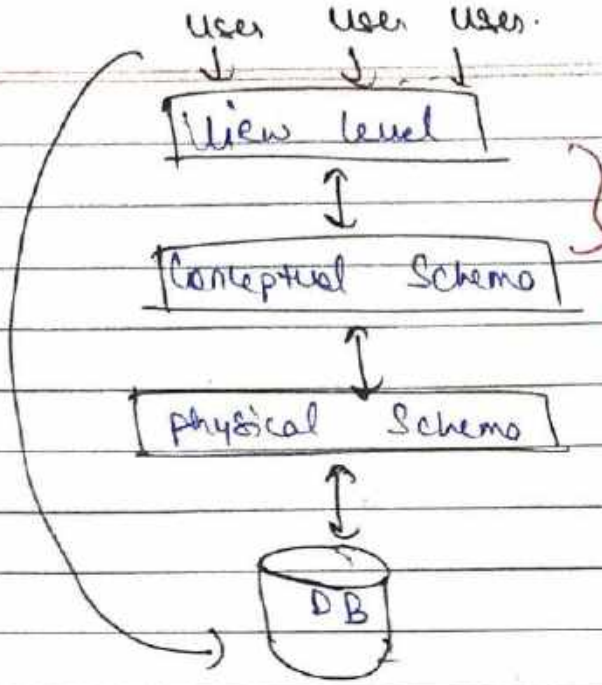
Note:) When we see the data as a user, then we see it in Table form. But, Actually in hard disk, data is stored as files. and we apply the layer of DBMS on it.

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

7. Data INDEPENDENCE :→

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

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



(#) Logical Data Independence: →

Ex.

Student Table. → add by user 1.

Name	Age	Mob. No.

It will not affect the Applica^n program.

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

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

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

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

→ Hence, View level don't change by these changes by users in Conceptual Schema.

Ex:- UMS (University Management System), Shopping website → They can add or delete any col's

Hence,
It is logical Data Independence.

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

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

Any change in Back End, 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	Reddy	Shamli	20
2	Anurag	Kanpur	21
3	Reddy	Shamli	20

1) Same 1 student repeats

2) 2 diff. students with same attri. or

4 To identify this, we must have a key.

Ex:) Student Table

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

These all attributes can uniquely identify any 2 rows.

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

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

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

---

9. PRIMARY KEY :→

(Every lock has its unique key.)

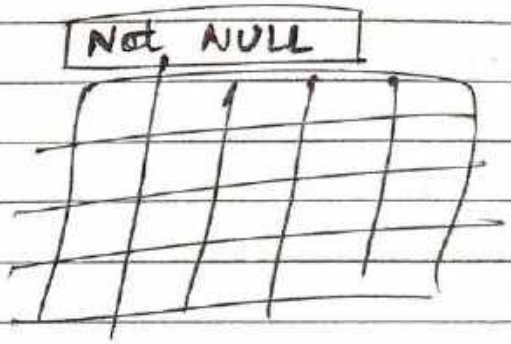
ie,
We uniquely identify 2 things.

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

Candidate keys! → These are Unique.

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

} Candidate keys.



→ phone no, aadhar card vagera को NULL नहीं रख सकते। करना ही पड़ेगा।

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

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

→ PRimary key = (unique + NOT NULL)

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

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

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

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

(10) Foreign key in DBMS :→

(F.k.)

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

→ It Maintains referential Integrity.

Ex:- Student →

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

Course →

(F.k.)

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

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

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

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

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

(with f.k.)

⇒ Table 2 is called Referencing Table.

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

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

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

⇒ Code.

Ex.

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

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

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

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

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

⇒ Integrity means same value for the database.

Ex 1 (1)

→ In mobile phones, diff. price at Flipkart, Amazon & store so, not Integrity.

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

Ex 1 -

Student (Base Table or Referenced Table)	Roll No.	name	add.
	1	A	Delhi
	2	B	Mumbai
	3	A	Chd.
	4	D	Ctd.

P.K. ✓ — Deletion

Course (Referencing table)	C. Id	C. name	Roll. No
	C_1	DBMS	1
	C_2	networks	2
	C_3	C++	7

F.K.

(#) Referenced Table →

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

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

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

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

~	~	Roll. No NULL

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

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

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

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

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

(#). Referencing table! →

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

2.) Delete! → No violation

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

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

(12) (Q.7)

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

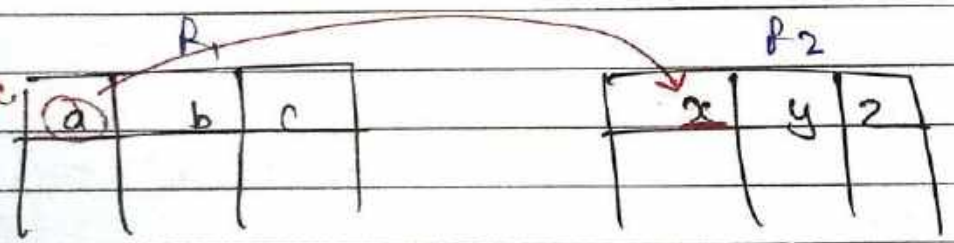
a) Insert into R_1 — ✗
b) Insert into R_2 — ✓
c) Delete from R_1 — ✓
d) Delete from R_2 — ✗

Which is correct regarding referential Integrity ?

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

Let x be p.k. in R_2.

Soln :-



Referencing Table. Base Table (or) Referenced Table.

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

(13) SUPER KEY in DBMS :→

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

⇒ Candidate key is minimal.

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

Roll no.	name	age

- Roll no, name
- Roll no, age
- Roll no, name, age

all are super key.

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

⇒ name, age × (not super key).

⇒ Super set of any candidate key is super key.

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

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

Ex:- ⇒ A1 A2 A3 (Either take or not take A1, 2 possibility)

2 × 2 × 2 = 8

⇒ 2^n.

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

i) (इसमें A_1 तो आएगा ही).

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

ii) $R(A_1, A_2, A_3, \ldots A_n)$.
Compulsory. $n-1$
$1 \times 2 \times 2 \times \ldots 2$.

So, $[2^{n-1}]$ Ans.

iii) $R(A_1, A_2, A_3, \ldots A_n)$.
$n-2$

→ A_1, A_2 both C.K.

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

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

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

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

Now,

$2^{n-1} + 2^{n-1} - 2^{n-2}$ ✓

or

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

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

then,

2^{n-2} ✓

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

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

Now,

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

To remove these common elements.

$A_1 A_2 A_3 A_4 - - - A_n$

2^{n-4}

$2^{n-2} + 2^{n-2} - 2^{n-4}$ ✓

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

06.

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

→ used for logical representation.

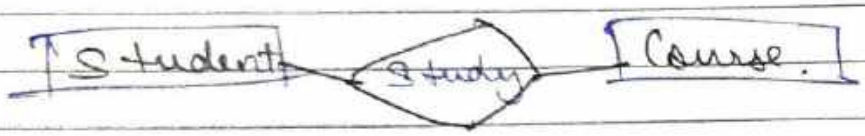
→ To see logical structure before implementation (Design).

→ It does the job of database design.

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

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

→ Relationship → Relationship b/w 2 or more Entities



Relationship b/w Student & Course is of Study.

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

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

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

Entity → Student → represented by rectangle
Attribute →
Relationship →

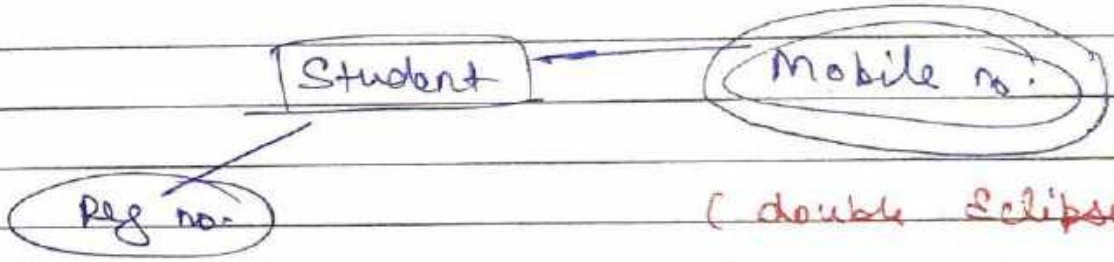
Q5. Types of Attributes in ER Model :-

[Student]

1.) Single vs Multivalued attributes :→

↓ Reg no.

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



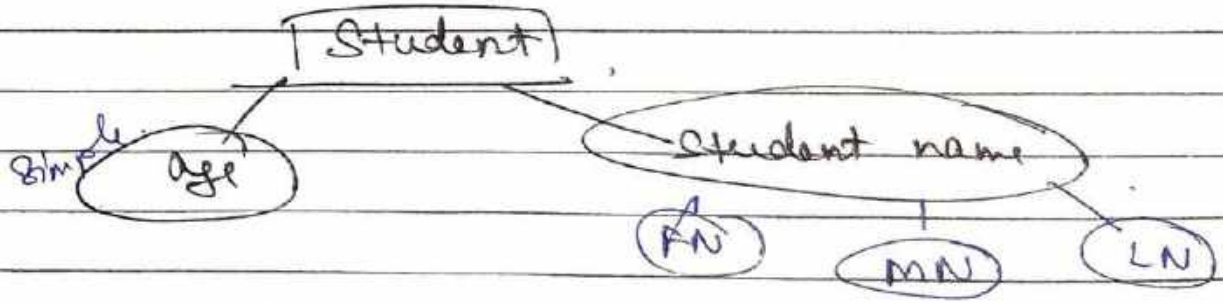
(double Eclipse).

2.) Simple vs Composite Attributes.

Simple - Can't be further divided.

Composed - Composed of more than 1 value.

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

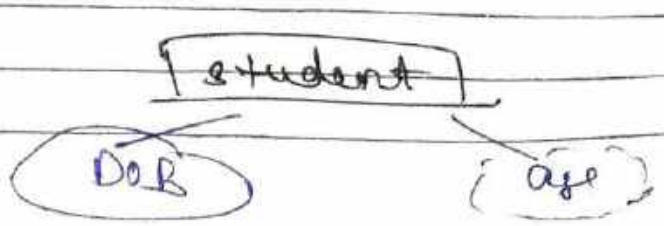


3.) Stored & Derived Attributes :→

Stored → These are stored & can't be derived.
Ex - DOB.

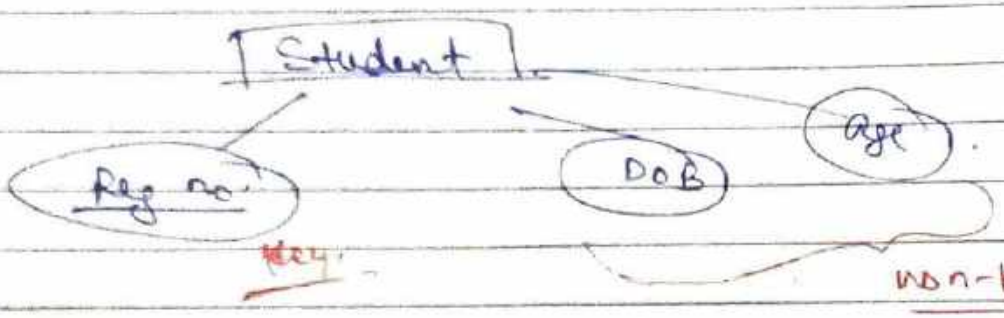
Derived → Ex - Age. (derived from D.O.B.)

dotted Eclipse.



4.) Key vs Non-key Attributes :- →
Key - used to uniquely identified.
Unique (No Repitition)

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



5.) Required vs optional Attributes :- →
Required → These are Mandatory (*) Name
optional → can also be leave D.OB.,
Add.,

6.) Complex Attribute :- →
(Composite + Multivalued)

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

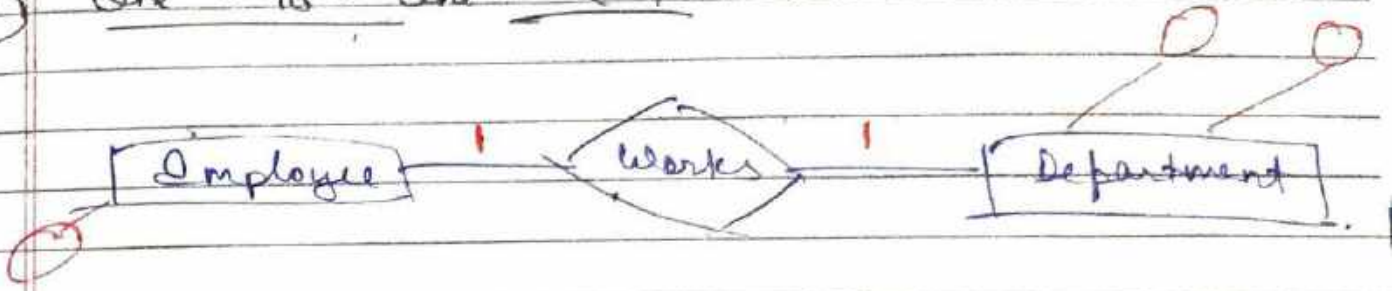
Add. is Composite → Street, State, City

16. Degree of Relationship :- → (Cardinality).
→ how the Entities are connected with Each other.

4 types! →

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

(1) One-to-One (1-1) :→



Convert Entity into Table :→

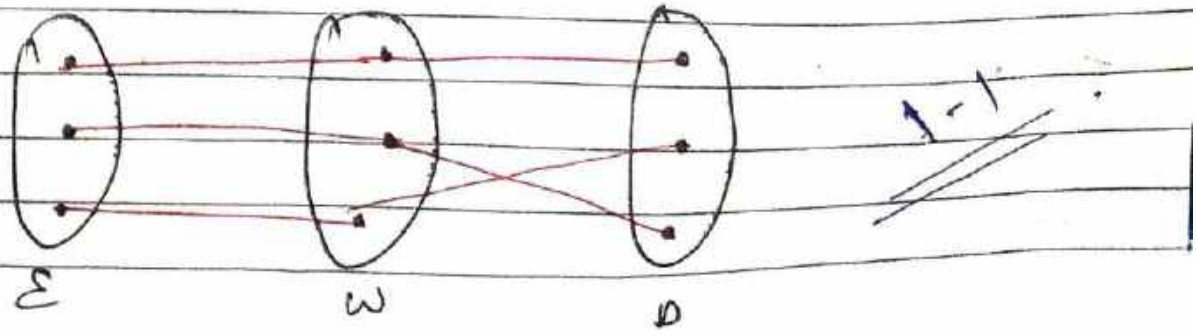
(Relo'ship का भी Table बनेगा, पर इसमें Employee & Department का Rel" बनाएंगे।)

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

(E.ID) & (D.ID)

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

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



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

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

EID	D_ID
E_1	D_1
E_3	D_2
E_2	D_3

D_ID	Dname	loc
D_1	IT	Bang.
D_2	Prod.	Delhi
D_3	HR	Delhi

Both
PK = E_ID

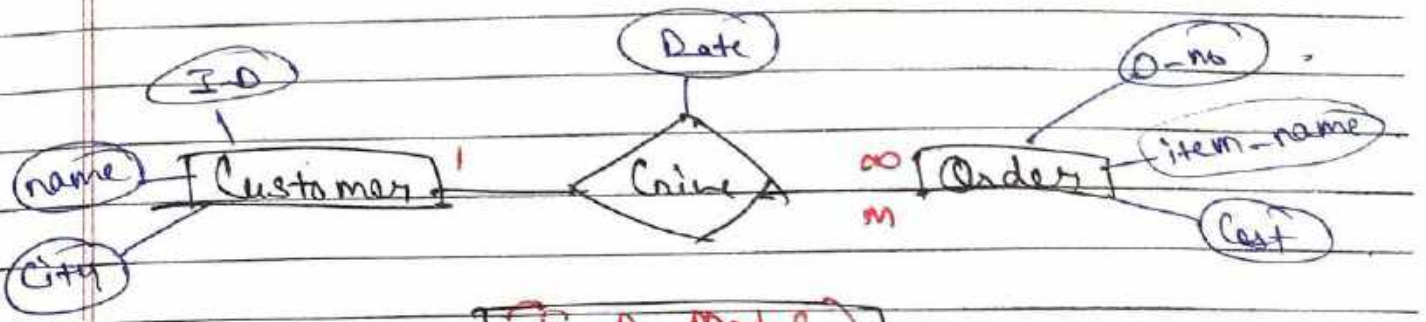
* Can we Merge?
Now, In Table 1 & Table 2, EID is the primary key.
Hence, we can merge Table 1 & 2.

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

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

Every Table must have its primary key.

(17) One to Many Relation ship : → (1-M).



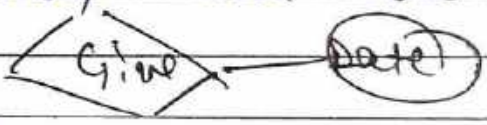
(E-R Model)

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

↓

a) Rela'ship may have its attribute.

Ex:-



& we call it Descriptive Attribute.

f.k. f.k.

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

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

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

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

P.K. = (O_no).

Note:- Always P.K. of the (many side) in (1-m) is also the p.k. of the relo"ship Table.

Can we Merge Tables? (many diff side merge).

Yes,

By Relo"ship & Order Table.
(bcz both have same P.K.).

=>

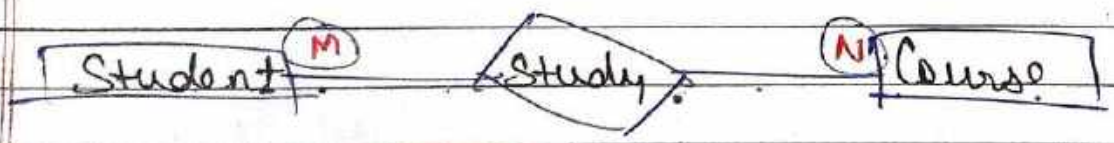
ID	O_no	item name	Cost	Date
~	~	~	~	~

Now,

2 Tables.

(M-1) is also same like this.

18. Many-to-Many Relo"ship :- (M-N).



f.k. f.k.

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

base Table

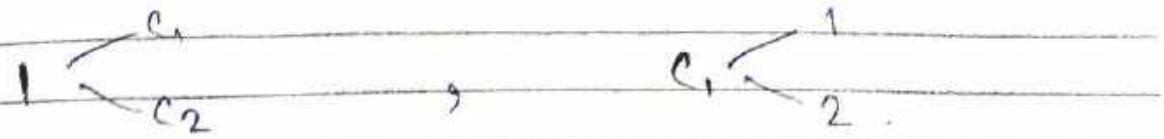
Rollno	C-id
1	C1
2	C2
1	C2
2	C4
3	C3

Referencing Table

C-id	name	Credit
C1	Maths	4
C2	phy.	4
C3	Chem.	4
C4	Hindi	4

Base Table

Many - Many.



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

Roll. No repeats &
C-id also repeats

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

i.e,

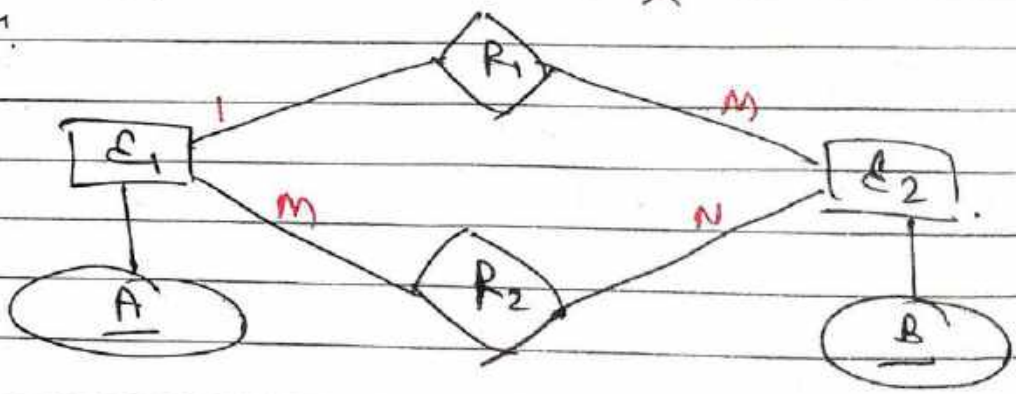
Composite key = Roll no. C-id

Can we Reduce Tables ?

→ No : bcz, p.k. is combined.

Note:- P.K. in Relationship Table depends on Relationship (1-1, 1-M, ... M-M).

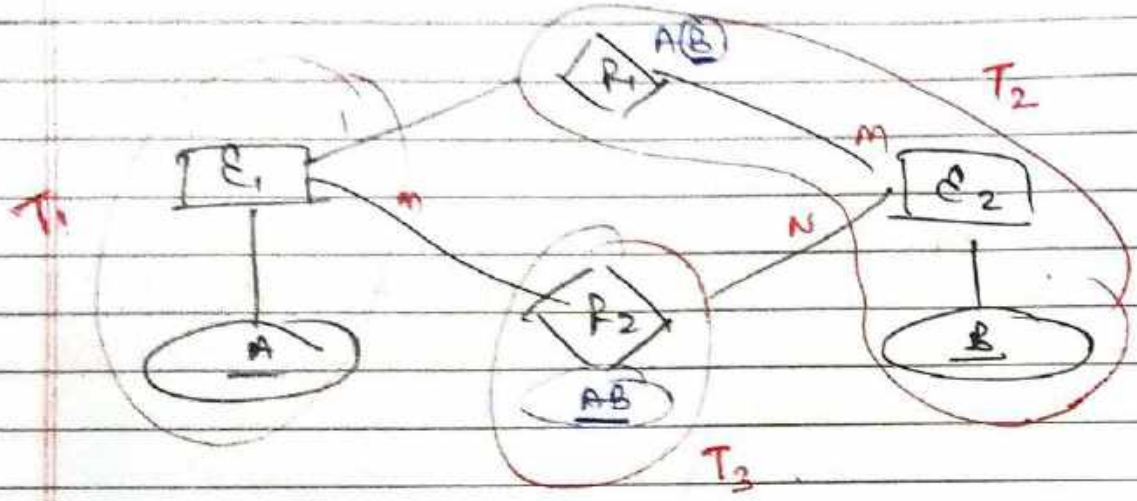
Q:-



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

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

E_1
E_2
R_1
R_2
} 4 Tables . But Minimum ?



Hence,

$T_1 = E_1$
$T_2 = R_1 E_2$
$T_3 = R_2$

3 Tables ✓.

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

---

20. **Normalization** :→

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

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

(1) Row level! →

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

Same (Duplicate) =

Row level.

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

primary key (unique + Not Null).

→ P.K. will take care of the duplicacy.

(2) Column-level! →

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

→ 4 columns are same in many rows.

→ ~~Anomaly~~

→ Insertion Anomaly

→ Deletion "

→ updation "

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

Now,

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

1) Insertion Anomaly :→

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

It is the our Insertion Anomaly.

2.) Deletion Anomaly :→

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

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

∴ It will delete the whole row.

We don't face any problm here.

Now,

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

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

→ Row 2 is deleted fully from database.

Now,

Row 2 is blank there.

Now,

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

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

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

Course Info — lost
faculty Info — lost.

i.e, data info is removed here & we can't recover it later.

3.) updation Anomaly :→

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

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

No problem here.

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

Change Salary of F1 from 30k to 40k.

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

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

* Now, Normalize removes Redundancy.

How?

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

P.K.	S-id	S-name

P.K.	C-id	C-name

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

This can be 1 of the rel^n.

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

---

21. First Normal Form :→ (1 NF)

Ef Codd → father of R.B.M.S

→ Table should not contain any multivalued attribute.

student

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

→ Not in 1st NF

{ Null means not available

Now, how to convert in 1st NF :-

1st way

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

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

(Combined, it is composite P.K.)

2nd Way

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

P.K. = Roll no.

3rd way

Roll no	Name
1	Sai
2	Anurag
3	Onkar

Base Table

P.K. = Roll no

f.k.

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

Referencing Table

P.K. = Roll no Course

f.K. = Roll no.

22. Closure Method :→

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

Ex:- Candidate key (C.K.).

R(ABCD).

FD of { A→B, B→C, C→D }.

(Functional Dependency).

Meaning of closure is that what 'A' can determine.

Here, A is determing B (from FD ①).

closure sym. ← $A^+ = B$

$A^+ = BCDA$.

(A can determine itself also)

Ex:- Roll no. can determine itself.

A → B
B → C
Transitive.

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

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

Now, $B^+ = BCD$

Here, B not determine A.

i, B cannot be a C.K.

Q $C^+ = CD$

$D^+ = D$.

Prime att. $= \{A\}$

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

→ Only A is C.K.

$\boxed{CK = \{A\}}$ Ans.

Note:- $(AB)^+ = ABCD$ (AB - itself, B → C, C → D.)

Here,

AB can be a C.K.

But

It is not a C.K.

bc→ C.K. is always minimal.

Q In AB only A is C.K.

∴ XtoA is Super key (S.K.)

AB → Super key (A ke saath add any thing, becomes S.K.)

→ AB is a Super key.

Ex:- R(ABCD).

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

$A^+ = ABCD$

$B^+ = BCDA$

$C^+ = CDAB$

$D^+ = DABC$

∴ {C.K. $= \{A, B, C, D\}$} all ✓.

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

Eg, prime att. = {A, B, C, D} (क्योंकि all ↑ are C.K.)

Non-prime att. = {ϕ} → NULL.

Q! R(ABCDE).

FD = {A → B, BC → D, E → C, D → A}.

⇒ Now, we have to check that which attribute are coming on the Right side. क्योंकि Attributes on the Right side, वो ↑ ϕ determine हो रहे हैं।

E = BDCA (E वाली आया)
= BDCAE.

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

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

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

Now, Start! → With A!

AE → ABECD ∴ A is C.K.

$BE^+ = BECDA$

$CE^+ = CE$ ×

$DE^+ = DEABC$

∴ C.K. = { AE, BE, DE } Ans.

→ Trick!- First, we got AE as C.K. So, Check either (A or E) on the right side of FD.

FD = { A → B, BC → D, E → C, D → A }.

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

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

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

× ×

(23) Functional Dependency :→ (F.D.)

is the method which describes the relationship b/w the attributes.

X → Y → Dependent Att.

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

Ex!- S-id → Sname → valid.

1 → Ranjit } ∴ These 2 are diff.

2 → Ranjit

Ex!-

1 → Ranjit } Same Student

1 → Ranjit

Valid case.

Ex:- 1 → Ranjit
2 → Raman
} Valid.

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

(A) F.D. are of 2 types:→

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

1.) Trivial F.D.:→

If $x \rightarrow y$

then,

y is subset of x.

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

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

NOTE:→ $x \rightarrow y$

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

Ex:- $Sid\ Sname \rightarrow Sid.$

$\cap\ Sid.$ ✓ Valid.

2.) Non-Trivial F.D.:→

If $x \rightarrow y$

then,

y is not a subset of x.

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

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

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

* Properties of F.D :→

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

2.) Augmentation: If $X \to Y$, then
$XZ \to YZ$
Sid → Sname
Sid phone → Sname phone

3.) Transitive: If
$X \to Y$ & $Y \to Z$
then, $X \to Z$.
Sid → Sname & Sname → City
Sid → City.

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

5.) Decomposition: If $X \to YZ$ then

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

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

6.) Pseudo Transitive:

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

$WX \to Z$

7.) Composition:-

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

then

$XZ \to YW$

2nd Normal Form :- (2nd NF).

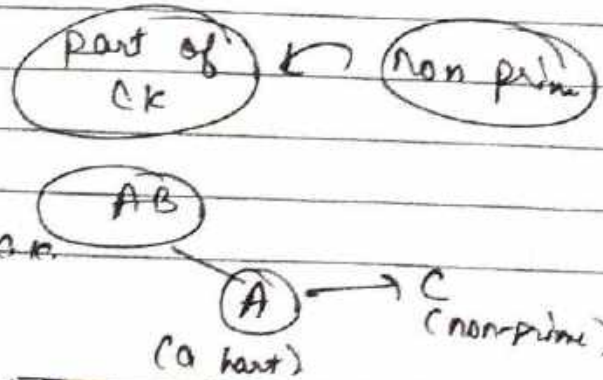
2 Rules

→ Table or Rel° must be in 1st NF.

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

(There should be no partial dependency in the Rel°).

Eg:



Ex! →

Customer

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

C.K. : Customer -Id Store Id

Prime attributes: C - Id
Store - Id.

Non prime : Loca^n

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

Now →

Convert it into 2nd NF :→

Here, we make 2 Table.

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

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

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

2nd NF.

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

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

Soln:- [C.K.]

First, check Right hand side.

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

So, on LHS, there must be CE.

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

CE = FADB

Now,

EC^+ = ECFADB (all 6 are present)

∴, EC is C.K.

Now, Use Trick

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

but

not, neither E nor C, noone is present.

So,

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

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

Proof:- check

Step 1 is comp. here.

After finding C.K. = {EC}

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

∴, proved.

proper subset is always less than a set.

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

Step 2 :→ prime Attributes : { E, C }

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

Step 3 :→ C.K. = { EC }

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

Now check :-

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

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

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

We get,

C → F

↳ partial dependency.

So,

Table is not in 2nd NF.

C → F	α PD }	partial
E → A	α PD }	fully.
EC → D	α FD }	
A → B	α FD }	

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

25. 3rd NF :→

→ Table or Rel' must be in 2nd NF.

&

→ There should be no transitive dependency in table.

Non prime ya Non-unique prime ya unique.

Date:
Page:

not sufficient condⁿ.

(NPA).

* Mean, Non-prime attr. kaa koi Non-prime attr. determine nahi kar sakta.

(C.k. & Prime attr. (P.A.) to kar sakte h determine NPA ko).

Ex:-

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

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

⇒ PA = { Roll no. }

⇒ NPA = { state, city }.

Here, Roll → State and State → City. (NPA → NPA).

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

So,

It is not in 3rd NF.

Ex:- R (ABCD)

FD: { AB → C, C → D }

⇒ C.k. = { AB }

PA = { A, B }

NPA = { C, D }

AB → ABCD. (Transitive)

A → D
NPA NPA

It is not in 3rd NF.

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

Ex:- R (ABCD).
FD: (AB → CD, D → A).

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

(B not on RHS)
B⁺ = B
AB⁺ = ABCD
Now, A on RHS.
so,
DB⁺ = DBAC
is also C.K.

Step 2:- Now, for each F.D.
LHS → C.K. or S.K.
[OR]
RHS → belong P.A.
→ check only 1 bcz [OR].

Then, Table is in 3rd NF.

FD: (AB → CD, D → A)
✓ ✓

∴ Table is in 3rd NF.

bcz, (NPA → NPA) is not present here.

---

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

⇒ also called as special case of 3rd NF.

[illegible]

Student

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

Table is in 3rd NF already.

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

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

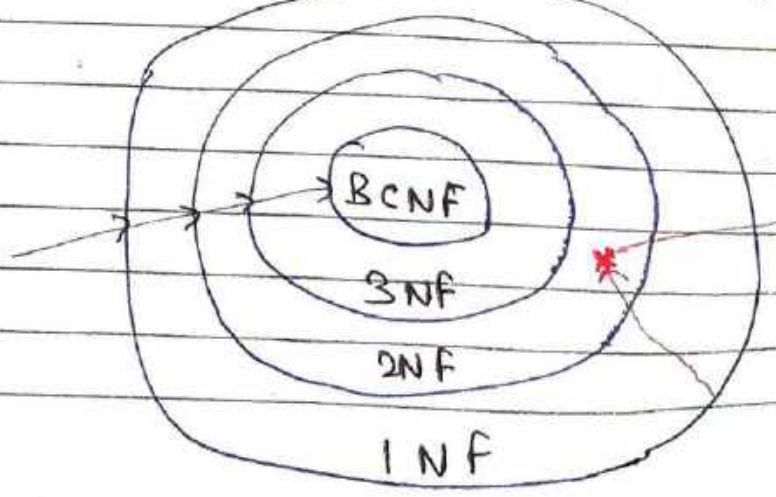
Note:- LHS of each FD should be C.K. or S.K.

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

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

Sol":- So, check all the F.D. one-by-one.
In all the 4 F.D, the LHS is C.K.
So,
It is in BCNF form. Ans.

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



It is outside → 3NF & in 2NF & 1NF both.

1NF.

2NF = 1NF + condi"

3NF = 2NF + condi"

BCNF = 3NF + condi"

27 Lossless & Lossy - Decomposi" :→

Join

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

R

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

→ R_1 (AB)

→ R_2 (BC)

We divide this table into R_1 & R_2

Q. B is common in both the Table.

R_1

A	B
1	2
2	2
3	3

R_2

B	C
2	1
2	2
3	2

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

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

So,

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

(1st row को Multiply all rows of Table 2 में होगी).

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

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

Natural Join = Cross product + Condi'n.

Now,

R_1 R_2

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

Now.

R'

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

→ table after Joining.

(fake) Spurious tuples.

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

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

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

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

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

→ Bcz, we take B as common in both Table.
but

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

3) ∴ B has duplicacy in Table, We have to choose attri. 'A' for Fifth Anc.

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

i.e.,

R_1 (AB)
R_2 (AC) → We get 3 tuples also in joining table.

Condⁿ for lossless Joining Decomposⁿ :→

1.) $R_1 \cup R_2 = R$

AB ∪ AC = ABC.

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

AB ∩ AC

A ≠ ϕ

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

To take Common attribute →

(28.) All normal forms with Real life Examples →

6 forms here

1st NF	2nd NF	3rd NF.
→ No multivalued attribute.	⇒ In 1st NF +	→ In 2nd NF. +
→ only single valued.	→ No partial Dependency. * only full dependency. (AB) → C (If A & B are 2 Head of a Co., then both will use the Emp. Id)	→ No Transitive dependency → No, NPA. should determine N.P.A.

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

Varun → 3 Phone no. → 3 Mail Id.

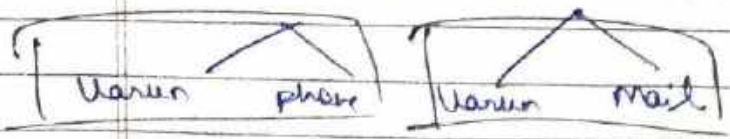
(Varun depends on multiple att i.e. phone & mail.

table

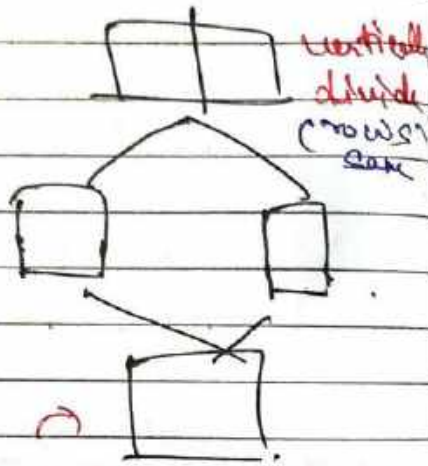
Varun	M_1	E_1
Varun	M_1	E_2
↓	M_1	E_3
	M_2	E_1
	M_2	E_2
	⋮	

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

So, make 2 table.



(Now, no multivalued dependency)



May be x^{100} tuples come. So, make C.K. as common attribute in both tables.

---

29. Minimal Cover : → (Irreducable) :

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

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

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

Sol^{n}:- Our RHS in F.D. must be single.

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

By decomposition prop, separate them.

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

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

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

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

$A^+ = A$ (not all ✗).

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

→ Same check this for every F.D. :-

$D \to B$ is redudant.

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

∴,

remove, $D \to B$.

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

Now, for $AC \to D$

$AC^+ = ACB$

∴ Also include $AC \to D$ ✓

Now, we get

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

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

Here, $AC \to D$

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

$C^+ = CB$

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

$AC \to D$ ही रहेगा।

Same check for A, by removing C

$A^+ = AB$.

∴ Can't remove C.

∴ $AC \to D$ can't be reduced.

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

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

30) Question on Normalization :→

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

F.D. : { AB→C, C→DE, E→F, F→A }.

Sol^n: Step 1: Find all CKs in Relⁿ :→

By closure Method :→

(B is not on RHS.) So, B compulsⁿ.

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

$AB^+ = ABCDEF$ (all ✓)

↳ [AB is C.K.]

Now, Check ∀ A on RHS.

we get F → A.

↳ [FB is also CK.]

Now, check F on RHS :→

E → F.

So, [EB is also C.K.]

Now, Check E on RHS.

C → DE.

So, [CB is also C.K.]

Now check ∀ C on RHS.

AB → C.

AB is already our C.K.

So, we get all C.K.

C.K. = { AB, FB, EB, CB } ∴ 4 CK.

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

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

NPA = { D }.

Step 3 :- Now, check FD :-

{ AB → C, C → DC, E → F, F → A }.

Now,

Check 1-by-1.

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

Redundancy decreases.

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

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

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

{ AB → C , C → DE , E → F , F → A }

× × ×

↓

It is not in BCNF form.

→ Now, 3NF :→

Check :→ transitive dependency (NPA → NPA) ×

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

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

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

↓ not in 3NF :→

→ Now, 2NF :→

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

Check! 1

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

↓

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

	AB → C	C → DE	E → F	F → A.
BCNF	✓	✗	✗	✗
3NF	✓	✗	✓	✓
2NF	✓	✗ (bcz both cond^n true). ↳ not in 2NF.	✓	✓
1st NF	✓	✓	✓	✓

↳ Ans is 1st NF. Ans.

bcz

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

& in

Table → R (ABCDEF)

all are general att.,

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

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

1st NF. Ans

proper subset is always less than a set.

31. Find Out Normal Form of a Rela" :→ (from 1NF - BCNF).

Q:- R(ABCDEF),

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

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

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

NPA = { D }

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

→ Check ⁴ 2nd NF :→ (Partial dependency) P.D.

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

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

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

(Full dependency).

∴ not in 2nd NF.

⁴ Make it in 2nd NF :→

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

Date: Page:

Condⁿ! →

1.) Lossless Decomposⁿ
2.) Dependency should be preserved.

Common attribute must be C.K.

Now, जो problem दे रखी, C → D. उसको Alg कर दो, Alg Table बना दो।.

Now.

R (A B C D E F)

R1 (A B • E F) R2 (C D)

Now, we have to make a common attr. b/w these 2 table! –

Criteria # common! – can be C.K. of any R1 & R2.

So, in R2 (C D)

C → D C+ = C D (all 2).

∴ C is C.K.

So,

Make 'C' common in both R1 & R2.

now, again

R (A B C D E F).

R1 (A B C E F) R2 (C D)

R1 { AB → C, C → E, E → F, F → A }

↓

2nd NF.

R2 { C → D }

& F T | C, F, ∅

(F.D.) | ∅ itself is not its proper subset

↓

2nd NF.

Now:-

Check ∀ 3NF:- →

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

So,

R_1 (A B C E F)

{AB → C, C → E, E → F, F → A}

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

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

→ It is 3rd NF.

R_2 (C D)

{C → D}

C.K. = {C}.

PA = {C}.

It is also 3rd NF.

Now:-

Check ∀ BCNF:- →

Condi" → L.H.S. must be a C.K.

R_1 {AB → C, C → E, E → F, F → A}

× × ×

not in BCNF form.

→ There is Redundancy.

R_2 {C → D}

It is BCNF. form.

↓

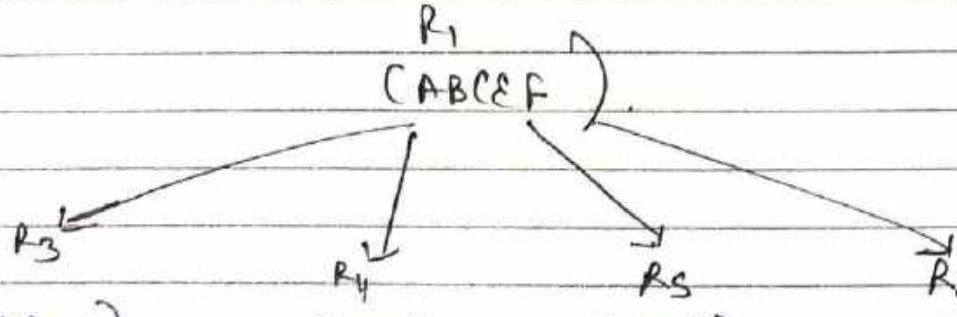
Redundancy = 0%.

Now,

Again Decompose R_1 (A B C E F).

(Till problem che 2 ke, Solution Aaya chahiye!)

Normalisation aim → To make redundancy 0%. SM We decompose, to normalise.



(AB→C) (C→E) (E→F) (F→A)

(ABC) (CE) (EF) (AF)

C.K. = {AB} C.K. = {C} C.K. = {E} C.K. = {F}.

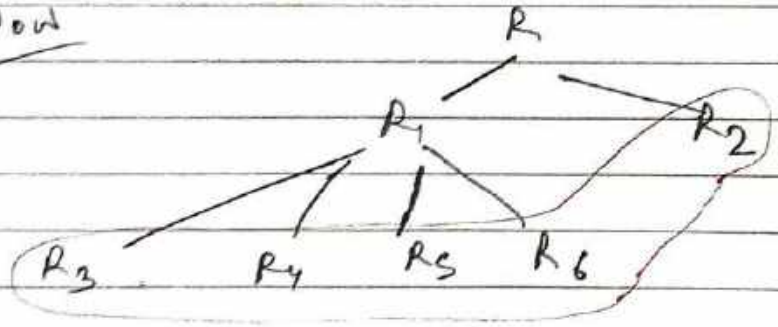
Now, these all 4 tables have C.K.

So,

These all 4 are in BCNF now.

(क्योंकि - सभी - सभी common भी हैं हैं साथ में)

Now

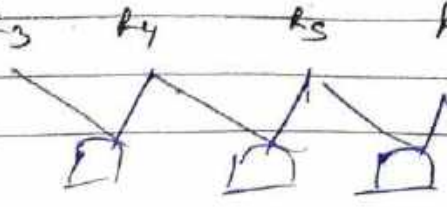


Now, In all these 5 tables,

Redundancy is 0%.

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

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



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

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

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

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

$R_3R_4R_5$ (ABCEF).
↳ E is C.K. now.

(32.) Normalisation Questions :→

→ A relⁿ is in BCNF, then it is in 2NF also

Q: A Relⁿ R has 8 att. (ABCDEFGH)

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

how many candidate keys in R.

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

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

D → So, now it comes with every.

$D^+ = D$

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

Now, check A on RHS,

So,

ED^+ also a C.K.

Now, check E on RHS.

So,

FD^+ also a C.K.

Now, check F on RHS.

So,

BD^+ also a C.K.

Now, check B on RHS.

Now, completed.

So,

AD, ED, FD, BD ↳ C.K.

33. Ques" Explained on Normalisation! →

Schema is a structure of a Table.

→ So, Scheme (or) Table → same.

→ Non trivial F.D. means in which.

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

→ Trivial F.D. are always valid.

Schema 1 : Registration (Roll. No, Courses)

Non trivial F.D of roll no → Courses }

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

→ So, as always, start from higher form.

→ Check for BCNF! -

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

and,

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

So,

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

So,

→ Table is in BCNF form.

∴ also in 1st NF, 2nd NF & 3NF.

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

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

Ans → C.K. = { roll no Course id }.

PA = { roll no, Co. Id }.

NPA = { email }.

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

So, not in BCNF form.

→ Check for 3NF:-

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

Now, 1st FD is already valid,
&
2nd FD: email → roll no.
P.A.
F T.
∴ (T)

It is in 3NF.

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

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

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

→ check for BCNF:-

1st FD → valid.
2nd FD → not valid.
∴ Not in BCNF.

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

→ check ∀ 2nd NF ! →

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

↳ for P.D. → ie, not in 2nd NF.
if both cond'n are True.

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

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

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

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

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

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

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

∴ not in 3rd NF.

Check ∀ 2NF ! →

1st FD → ✓

2nd FD → C-Id → Credit
T T
(+ T)

∴ It is Partial Dependency (P.D.)

So,

It is not in 2nd NF. ∴

∴

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

(34.) Cover & Equivalence of F.D. : →

If X covers Y Y ⊆ X

If Y covers X X ⊆ Y

If both are True.

then, [X ≡ Y] → Equivalent.

Q:- X = {A → B, B → C} | Y = {A → B, B → C, A → C}.

If X covers Y : → (Y ⊆ X)

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

So,

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

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

∴ all 3 covers in these closure forms.

∴ X covers Y. ✓

(ii) Y covers X :→

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

(X का closure Y से लेंगे)

$A^+ = ABC$

$B^+ = BC$

∴ Y also covers X. ✓

∴ both symbols cancels each other & become Equivalent.

$X \not\subseteq Y \quad Y \not\subseteq X$

$\boxed{X \equiv Y}$ Ans. ✓

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

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

(i) X Covers Y :→

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

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

∴ True.

ii) Y Covers X! →

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

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

∴ Y not Covers X.

It becomes false.

∴

$[X \neq Y]$ ok.

&

$[X \supseteq Y]$ & $[Y \supseteq X]$.

✓, true. ✗, false ✓.

---

35. Dependency Preserving Decomposⁿ! →

Defⁿ! → Let, any table $[R(ABCD)]$

& also some F.D., $[FD\{A \to B, B \to C\}]$

Now,

We find closure & then find C.K., & then we also find hidden dependencies! → like.

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

and,

→ In normalisaⁿ, we do decompⁿ, then, we divide

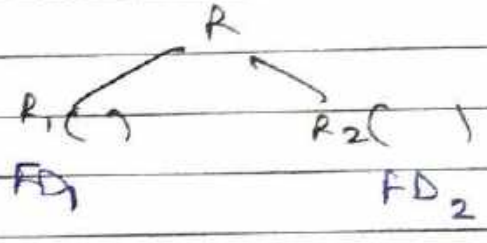
R

$R_1(\ )$ $R_2(\ )$

Use distributive attr. of R in R_1 & R_2

Union of attr. of R_1 & R_2 must be equal to the attr. of R.

Now, F.D. also divides, like



Now,

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

ie, $FD_1 \cup FD_2 = FD^+$ (original F.D. के Equal)

Q! Let $R(ABCD)$ with F.D.

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

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

⇒

$R_1(AB)$	$R_2(BC)$	$R_3(BD)$
~~$A \to A$~~	$B \to C$ ✓	$B \to D$ ✓
~~$B \to B$~~	$C \to B$ ✓	$D \to B$ ✓
$A \to B$ ✓		
$B \to A$ ✗		

$A^+ = ABCD$
(don't have A).

union of att' R_1, R_2 & R_3 are Equal to R. So, True. (Right decomposition)

Now

→ Check Dependency preservation :→

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

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

We don't want Trivial F.D. :→

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

i.e.

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

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

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

$A \to A$ is definitely true.

So,

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

→ Now:

F.D :- { $A \to B$, $B \to C$, $C \to D$, $D \to B$ }.

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

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

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

$A \to B$ ✓
$B \to C$ ✓
$C \to B$
$B \to D$
$D \to B$. ✓

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

So, F.D : { $A \to B$, $B \to C$, $C \to D$, $D \to B$ }.

Now,

$C \to D$ is not direct.
So, now take 'C' closure.

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

∴ all F.D. are preserved, True. Ans'.

(36.) Dependency Preserving Decomposition Example

Q.2:- Let R(ABCD) with F.D.
{ $AB \to CD$, $D \to A$ }.

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

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

attributes of $(R_1 \cup R_2)$ make R.

Now,

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

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

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

So,

F.D. we get from table! →

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

Now, check ∀ original F.D.! →

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

✗ ✓

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

We, 'can' get

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

Hence,

Dependency preserved not. Ans

37. Joins & Its Types :→

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

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

"Employee"

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

"Department"

↓

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

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

Now :-

Q. Find Ename of Emp whose working in HR

Here, HR is in department Table
& Ename is in Employee Table.

So,

Here, we need Joins.

Note :- There must be same common att. in Tables to use Join. Here, E_no is common.

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

Its Types! →

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

38 "Natural Join"! →

Join = Cross product + Condi°.

(m x n)

Employee (m)

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

Department (n)

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

→ first find! - Table name

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

(Emp, Dept Tables. Cross product.)

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

We do it by "Natural Join".

→ Common Att - 'E-No.'

Eg,

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

→ Now, Write Query :→

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

Output :→ Ram, Varun, Amrit.

'E-No' - must be same in both the Tables.
i.e (E-No / Emp-No.)

Emp		Dept	
E-no	E name	Dep No.	E no.
1	Ram	D_1	1
1	"	D_2	2
1	"	D_3	4
2	Varun	D_1	1
2	"	D_2	2
2	"	D_3	4
3	Ravi	D_1	1
3	"	D_2	2
3	"	D_3	4
4	Amrit	D_1	1
4	"	D_2	2
4	"	D_3	4

So, finally we get 3 rows.

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

Ram
Varun
Amit

→ Ans:

Now, how to write Actual Query! →

→ Select E_name from Emp Natural Join Dept;

39. Self Join ! →

→ in which the Table is joined to itself.

S-id

Different C-id Table.

S-id	C-id	Since
S_1	C_1	2016
S_2	C_2	2014
S_1	C_2	2017

Student Course

Study

Q: Find student id who is enrolled

→ Now, we do Self Join.

- SQL query always start from - 'from' (sequal)

Query :→

```
Select  from Study as T1, Study as T2;
```

',' → means Cross product.

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

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

(n) T_1

S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

T_2 (m)

m x n

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

Condⁿ :-

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

< > → different (not equal to).

Now, Complete Query :-

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

So, final output Table :-

S_1	C_1	S_1	C_2
S_1	C_2	S_1	C_1

→ Now, we want S-id from this Table.
But T1 & T2 both have S-id.
So,
we can use any $T_1.Sid$ or $T_2.Sid$.

Ans - S_1

(Q.) EQUI - Join : → (=).
(कोई भी attribute के बीच (=) लगा सकते हैं)

Ans

Employee.

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

Department / Dept.

Dept_No	Loc^n	E_no
D_1	Delhi	1
D_2	Pune	2
D_3	Patna	4

T ∧ T	= T
T ∧ F	= F
F ∧ F	= F

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

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

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

(A) Query :→

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

Emp.

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

Dept.

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

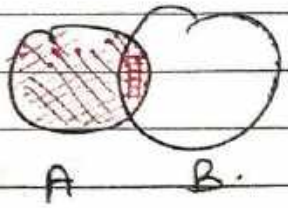
→ Ram

Ans

Note:- दो table के common attr. के बीच में हमेशा (=) लगाना है & किसी भी 2 Attr के बीच हम (=) लगा सकते है।

(41) Left Outer Join :→

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



Natural Join + Something else

Emp

Emp-no	E-name	Dept-no.
E_1	Varun	D_1
E_2	Amrit	D_2
E_3	Rani	D_1
E_4	Nitin	—

Dept.

Dept-no.	Dname	Loc.
D_1	IT	Delhi
D_2	HR	Hyd.
D_3	Finance	Pune

Query :→

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

left → Condⁿ of Natural Join. (Common attr. वाले col. 1/2 बार नहीं।)

Rela'al database always shows output in Table form.

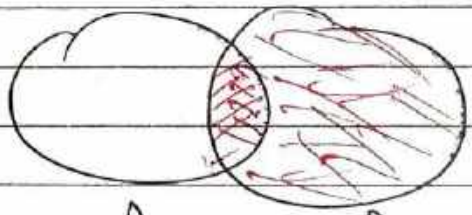
output Table :-

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

→ (Left की तो सभी Row आनी ही है।)

(12.) Right Outer Join :→

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



(Emp)

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

(Dept)

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

Query :-

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

Natural join.

⇒ Output Table : →

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

Right Table all Rows.

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

(Left O.J.) ∪ (Right O.J.)

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

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

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

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

"operators"

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

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

(44.) Projection in Rela"al algebra :→

→ Projection (π) : →

π func. is to Retrieve the data.

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

← (Student)

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

Date:
Page:

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

Roll no.
1
2
3

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

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

Output →

Roll no	Name
1	A
2	B
3	A

Note: It only gives unique Answers.

Q.) # Query:- $\Pi_{\text{name}}(\text{Student})$.

Name
A
B

(only name here).

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

---

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

σ (Name).

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

⇒ It works on Tuples (rows).

→ First, it found rows.

π operator.

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

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

↓ Final.

$\pi_{name}(\sigma_{rollno='2'}(Student))$ → B

π always last में operate होता है।

* We can also find 2 at a time.

Ex1.

$\pi_{name, Age}(\sigma_{rollno='2'}(student))$ → B, 21

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

---

46. Cross / Cartesian product in Rel'al algebra

⇒

A	B	C
1	2	3
2	1	4

R_1

C	D	E
3	4	5
2	1	2

R_2

To join, we MUST have to cross product.

→ $3 + 3 = 6$ (m+n).

$2 \times 2 = 4$ (x×y).

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

→ $(R_1 \times R_2)$

(R_1) (R_2) =

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

---

(47) **Set Difference in R.A :→**

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

$= A \cap B'$

Ex:-

(1,2,3) S_1 (3,4) S_2

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

→ **It is not Commutative :→**

i.e.

$A - B \neq B - A$

Condⁿ :-

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

Ex :-

Rollno	Name
1	A
2	B
3	C

(Student)

Emp-No	Name
7	E
1	A

(Employee)

⇒ (Student) - (Employee)

Rollno	Name
2	B
3	C.

⇒ By default, पहले table की column लेता है।

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

⇒ (Student - Employee).

⇒ How to Write ?

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

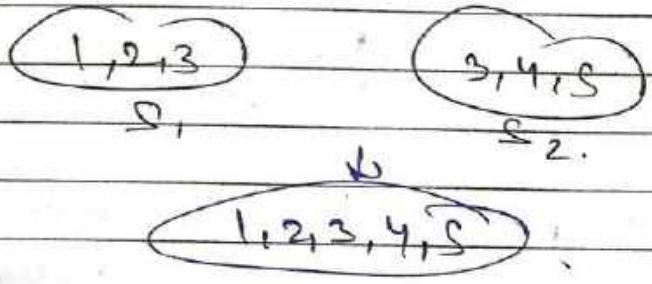
⇒ Output :-

Name
B
C

((A(B,C) - (E,A))

(48) Union Opera" in R.A. :→

⇒ Same as in Set. (∪).



Ex:→

Roll no.	Name
1	A
2	B
3	C

(Student)

Emp. No.	Name
7	E
1	A

(Employee)

Condⁿ :→

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

(student) ∪ (Employee)

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

→ (Left side" Table. ātā hī).

Easy Trick! -

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

Name.
A
B
C
E.

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

Note: 1)

Roll no	Name
1	A
2	B
3	C

Numeric

Name	Emp No.
E	2
A	1.

alphabet

X

not Same domain.

(By Default order pick ch2rT !).

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

× ×

(49.) Division Opera" in R. A. :→

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

÷ (×, −, Π, σ).

→ We use these

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

Date:
Page:

⇒ Every (या) all दिखाने की Divis" method.

Sid	Cid
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

'Enrolled'

Cid
C_1
C_2

'Course'

(#) Notaⁿ of Divis" Method :→

$A(x, y) \,/\, B(y)$ = it results 'x' values for that there should be tuple $<x, y>$ for every y value of Relⁿ B.

Here,

$$E(Sid, Cid) \,/\, C(Cid) = S_1$$

i.e,
S_1 enrolled in every course (C_1 & C_2).

How it happens ?

We let, all students are enrolled in every course.
(for this, we do the cross product),

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

$S_1 \times C_1$
$S_2 \quad C_2$
S_3

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

—

S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

(Enrolled)

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

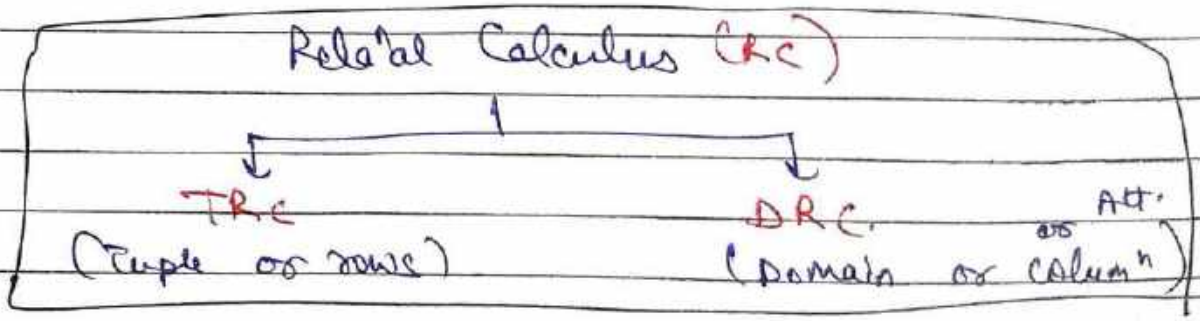
(*) Now, final ! → (Query).

$$\Pi_{Sid}(Enrolled) - (\Pi_{sid}(((\Pi_{sid}(Enrolled)) * \Pi_{cid}(course)) - (Enrolled))$$

S_1 — S_2
~~S_2~~ S_3
~~S_3~~

∴ S_1

So- Tuple Calculus in DBMS :→



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

⇒ 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 t.

* Operations :→

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

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

→ It also uses quantifiers :

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

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

* Unsafe Expression :- → (U.E.)

Supplier (Table).

- $\{ s.name \mid \neg Supplier(s) \}$

(not sign)

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

So, here, ans is ∞. (So, ∞ loop ~~it~~ ~~that output~~).

↳ Unsafe Expression.

(These are not in the Relational algebra)

- Both TRC & RA have same expressive powers.

↓

unsafe Exp → $\{ U.E(x) \}$.

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

But, SQL has more powers than these. SQL has more extra funcs

* 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?

('Catalog' = सूची)

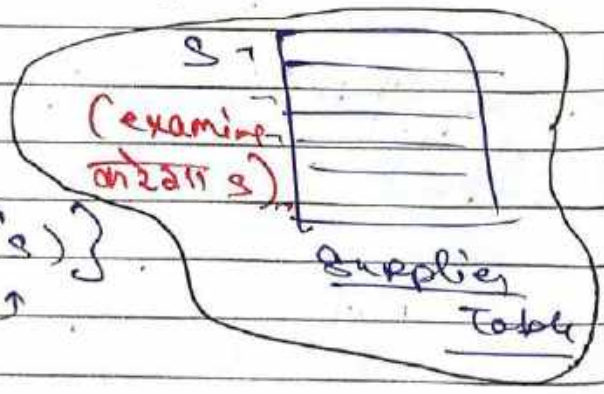
Q.3 Write a query in SQL to display SID of suppliers whose name is 'Varun' & address is 'chandigarh'?

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

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

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

(1)



{ S.Sname | Supplier (S) }

(2) { p.Pname | Parts (p) ∧ p.color = 'Red' }

(3) { S.SID | Supplier (S) ∧ S.name = 'Varun' ∧ S.add = 'chandigarh' }

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

{ C.SID | Catalog (C) }

Sid ही Catalog Table मे ही Field आहे.
But, Sname Only in Supplier table.

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

&

final ans. from Supplier Table.
(There Exists).

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

Here,

S [redacted] → free variable
c → bounded variable.

(जिथे कोणता दोन्ही ∃ आणि ∀ लावावे लागतात).

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

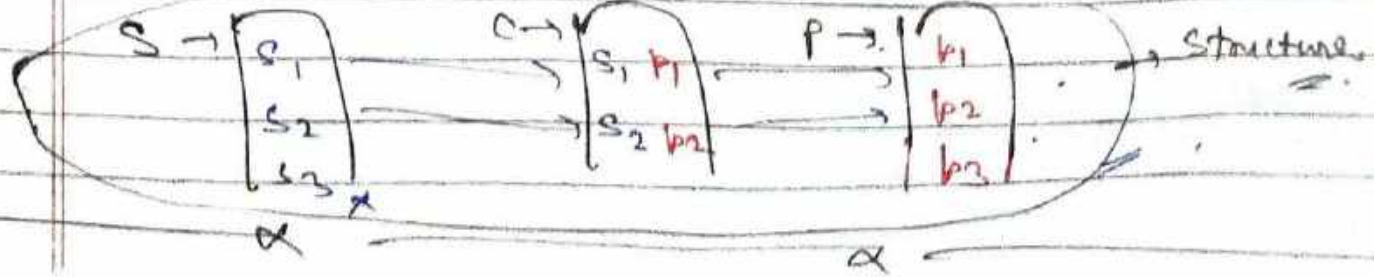
But,

ans from Supplier Table.

$$\{ S.Sname \mid Supplier(s) \wedge \exists c (Catalog(c)) \wedge \exists p (parts(p)) \wedge S.Sid = C.Sid \wedge C.pid = p.pid \wedge p.color = 'red' \}$$

S variable for Supplier Table.
C —— " —— Catalog ——.
P —— " —— parts ——.

S → free variable
C, P → bounded "



SM

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

(1970)

→ EF codd give theory of store & fetching of data.
Rel'al Model:

By the help of Relational algebra & TR: (Tuple Rel'l calculus) 1/2

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

→ Before,
SEQUEL → Simple English Query language
then,
SQL → Structure Query lang.

→ then, Oracle & Microsoft, etc. comes into this field of databases. 3

→ Now, No SQL is come into market.

10% data → Structure
90% data → Unstructure (Spark, Mongodb)

2) Properties :-

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

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

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

later on,
PL SQL → procedural language.

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

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

aggregate → मतलब, family, ग्रुप आदि बनाना.

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 (Applica' programming interface).

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

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

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

8.) Joins & Nested Query :-

9.) PL SQL (Triggers, func., Cursor, procedures) same as in 'C' lang.

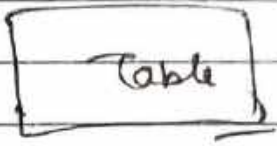
(#) What to do :-
In libraries of oracle as like SQL series, it is already defined that what select do, & what other commands 'do', so, we don't need to give procedure, that how to do.

(57) All Types of SQL Commands :-

* DDL Commands :- (Data Definition language).

→ Deals with Schema (structure | Table).

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



* DML Commands :- (Data manipula" language).

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

- → Select
- → Insert.
- → Update.
- → Delete.

* DCL Commands :- (Data Control language)

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

- → Grant (give permis")
- → Revoke. (take back that permis").

* TCL commands :- (Transac" control lang)

→ Mainly used in Transac"s.

Ex:- Shopping, Bill payment - online

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

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

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

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

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

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

S3:- Create Table in SQL with execution :- →

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

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

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

Ex:-

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

ex- (2 bit fixed)

54. Alter Command (DDL) in SQL :→

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

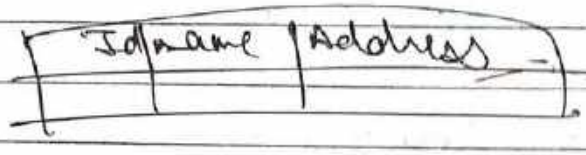
Ex:

Student

int	ID	name	varchar (20)

→ already हैं

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



student.

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

(,);

* multiple colu^ms.

⇒ Modify datatype:-

Ex:- from int to varchar.

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

X X

(55.) D/B Alter & Update : →

(*) **Alter!** → only Change in structure, (DDL) & not in data.

Ex!

Emp

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

} → can only change in this struc.

→ Int → varchar

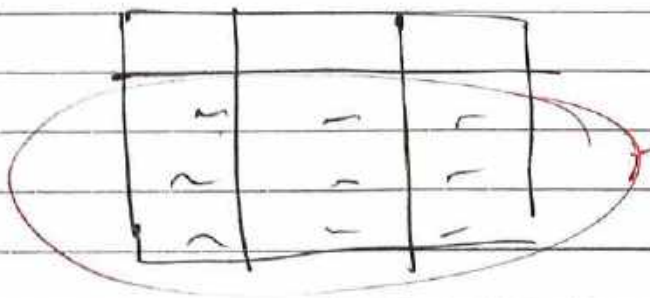
→ name varchar (10) → 20

→ ID → Eid.

→ Emp → Emp_detail

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

(*) **Update!** → can only change in data, (DML) & not in Structure.



→ can only change in data.

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

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

Date:
Page:

56. 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

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

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

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

> log :- कोई file completely delete होने से पहले, Computer उसकी temp. file बनाता है, to restore data, इसे log कहते हैं।

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

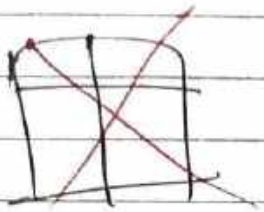
Rollback ;

(possible, but it creates log).

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

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

→ no object found.
(all delete).



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) restrict.

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

anurag@gmail.com ✗ (not present)

anurag13@gmail.com ✓ (present).

Date: Page:

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

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

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

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

2.) Not Null :→ *

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

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

* 3.) Primary key = Unique + Not Null

Ex:- Reg. no.

4.) Check :→

age int

✗ (-10)

Check (age > 18)

→ 10 ✗

→ 20 ✓

i.e, (एक particular Domain में ही ही User Id बनानी है)

Domain - fixed.

5.) Foreign key :-

6.) Default :-

```
Salary int default 10k
```

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

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

---

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

Emp.

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

Q:-

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

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

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

<> → not equal to : = ≠

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

Output → 50000

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

```
Select ename from Emp where
Salary = (Select max(Salary) from Emp);
```

Outer Query. Inner Query.

Output → Varun

→ Inner Query execute first 1 time

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

Q9. (iii) Write a SQL query to display 2nd highest Salary from Emp table ?

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

(iii)

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

Outer
10k -
20k -
30k -
40k -
(50k)
↳ Inner

max ← 40k

50k. x <

(iii) Query :-

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

Sol.

Ok ✓

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

Ex :-

2 = 2 ✓

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

Query :- (only add name in (iii))

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

(Output will be.)

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

Output :-

HR	2
MRKT.	2
I T.	1

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

AI → 101
CSE → 100

Note-1 Here, we use a special func. [group by] (Comp.)

→ (forms set of, dept.).

HR
HR
MRKT
MRKT
IT.

HR — 2
MRKT — 2
IT — 1

Note-1 Cond^{n} of group by → जो Attr. का नाम हम group by के साथ use कर रहे हैं, सिर्फ वही Select ____ के साथ लिख सकते हैं

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

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

→ Output → HR, HR, MRKT, MRKT, IT

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

→ Query:-
Select dept, Count(*) from Emp group by dept;

Output:-
HR — 2
MRKT — 2
IT — 1

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

HR 2
MRKT 2
IT 1 ✓

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

Query:-

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

Output → IT.

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

Query →

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

↳ Varun.

In
* HR IT
* MRKT
IT ✓

62) (viii) 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 → Ravi, Nitin, Varun ... dt.

Steps →

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

Outer: 10k x, 20k x, 30k ✓, 40k ✓, 50k ✓

Inner: 30k, 40k, 50k

* means All attributes.

Simple in Query.

63. Use of IN and Not IN

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

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

Emp

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

Project

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

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

* Query:-

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

Output:- | 2 | Varun | Delhi |

⇒ But, we have 3 cities. So,

* Query:-

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

Output:

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

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

Nested Query = Query (Query)

NOW, NOT IN :- (rows not included)

* Query :-

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

Output :-

4	Robin	Bangalore

Chd ×
Delhi ×
Pune ×
Bangalore ✓
Chd. ×

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

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

⇒ Suggestion :- first make Dummy Tables.

⇒ Here, we need 2 Tables

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

⇒ Nested → Bottom up → नीचे से ऊपर

Means पहले ऊपर वाली query & then बाहर वाली

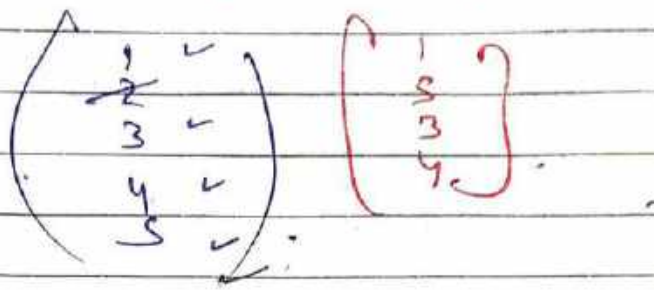
Inner Query in Nested Query - runs 1 time.

* Query :-

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

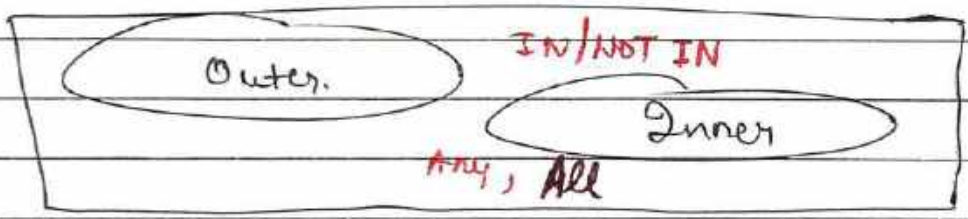
Inner Query.

Output :- Ravi
Nitin
Robin
Anuj.



65. EXIST & NOT EXIST Subqueries :-

↳ We use these in Correlated Nested Query.



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

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

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

Null → means value is not available.
Empty

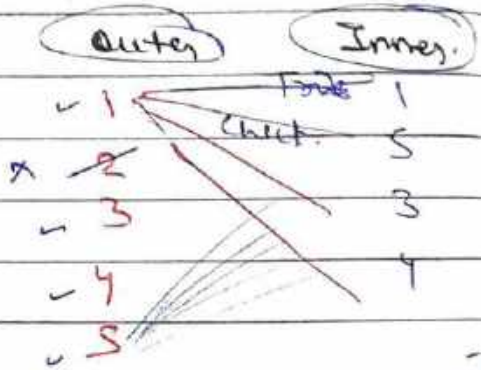
3 Query :-

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

Note:- Exist / not Exist gives True or False.

Output:-

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



66. Aggregate Funcs in SQL :- →

Max, Min, Count, Avg, Sum.

(Total no of values in a Table).

Emp

E_id	Ename	Dept	Salary
1	Ram	HR	10K
2	Amrit	MRKT	20K
3	Ravi	HR	30K
4	Nitin	MRKT	30K
5	Varun	IT	50K
6	Sandy	Testing	Null

- Query for Max. Salary :-

→

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

Output → 50K

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

4 Same for Min.

→ Select Min (Salary) from Emp;

Output → 10k.

* COUNT! -

Count (*) Means no. of rows in the Table

→ Select Count (*) from Emp;

Output → 6 — 6 rows.

→ Select Count (Salary) from Emp;

Output → 5 (bcz one value is Null)

* Distinct! -

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

Output → 4. (bcz 30k is 2 times)

(*) SUM! -

Slect Sum (Salary) from Emp;

Output → 140 k. (Sum of all salary)

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

Output → 110 k (30k repeats)

(*) AVG :-

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

= 140k / 5

= 28k

Select Avg (Salary) from Emp;

Output :- 28k.

Select Distinct (Avg (Salary)) from Emp;

Output :- 27,500.

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

= 110k / 4

= 27,500

---

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

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

→ Returns True / False.

Emp

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

Dept

D-id	D-name	E-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 होते ही हमें Employee Detail दे देगा।

→ Query :-

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

Output :→

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

Inner

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

False

68. DB Joins, Nested Subquery & Corelated Subquery :→

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

Emp

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

Dept

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

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

(Here, we need both 2 Tables).

→ Nested Subquery →

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

Output →

1	A
2	B
3	C

→ Correlated Subquery :→

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

E_id is common in both Table

Output :-

1	A
2	B
3	C

Inner

1 = 1 True
2 = 1
3 = 1

Outer

1 = 5
2 = 5
3 = 5

False

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

then, Total Comparison → m x n

→ JOINS :→

Cross product + Condi^n.

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

- then,
check Condi^n → emp.eid = dept.eid

Note:- Joins is faster than Correlated Subquery bcz, it made a table of rows (m x 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 x n) Rows).

69. Find N^th Highest Salary using SQL :→

Emp:

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

→ Highest Salary :-

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

Output → 50k.

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

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

Output :- 40k.

Now

How to recognise Corelated Subquery ?
जहाँ पर outer Query का कोई ना कोई value इसने Inner use किया है।

How to find N^th Highest Salary ! →

Query :- Best method → learn it.

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

→ i.e. Co-related Subquery

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

(Dummy)

* Ex:- on leet :-

E

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

E2

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

⇒ Why we make 2 (E1 & E2) :-
bcz employee we use 2 times. (E1 & E2).

If we don't create E1 & E2, then
e2.salary > e1.salary

↳ It means, employee salary > employee salary
↳ E1 & E2. ↳ no mean, X.

Ex:- 10k > 20k f
1st row 20k > 20 f
2nd row 30k > 20 count=1
40k > 20 count=2
50k. > 20 count=3 } count=3

Ex:- Let we have to find 4th highest, then
N-1 ⇒ 4-1 = 3

&

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

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

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

Output:→

ID	Salary
2	20k.

20k is our 4th highest Salary.

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

Ex:1 for 3rd highest Salary!?

N=3, N-1 = 2

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

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

4th Row is our output

Output: 4, 30k.

& 30k is our 3rd highest Salary.

70. 3 Imp. Questions on SQL basic concepts :-

Q.(1) 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%' ×
c) — " — name like '%A%'
d) — " — like '*A%'

Note :- Questions like, 2nd letter same, Salary be of only 5 no's, like that,

based on 'Like' command.

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

Ex:- i) %A% anything AABA कहीं भी आ सकती है

ii) 2nd letter → '_A%' — (one position is fixed).

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

iv) 2nd last letter must be A → '%A_'

(2.) A command to remove redo from SQL database → (Table).

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

→ DDL Commands deal with Schema (table structure).

Create, drop, Alter → DDL Commands.

→ Alter table → to change anything in table.

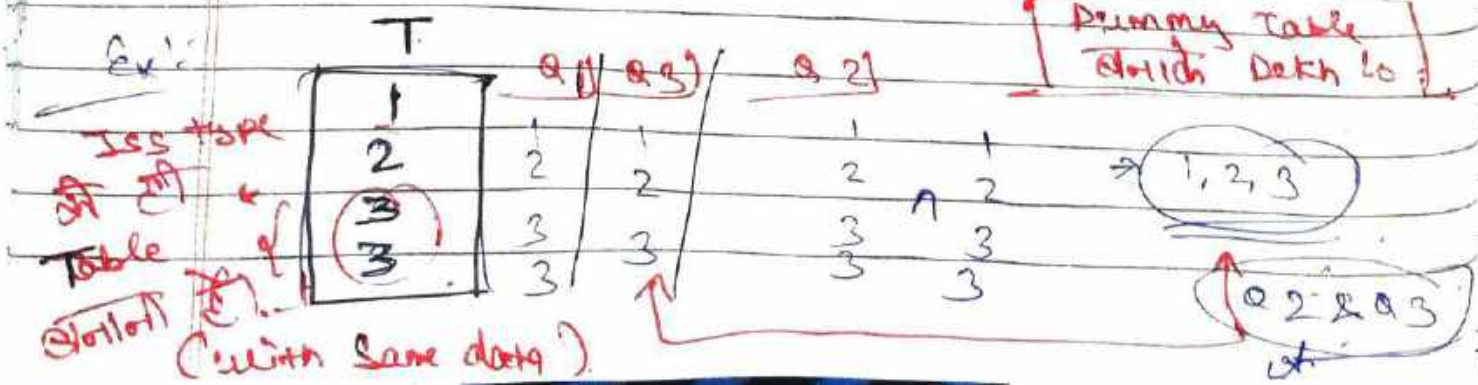
[Delete table, Erase table.] ≠ (Even not a command)

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

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

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

Ex:



Iss type ki Table (with same data)

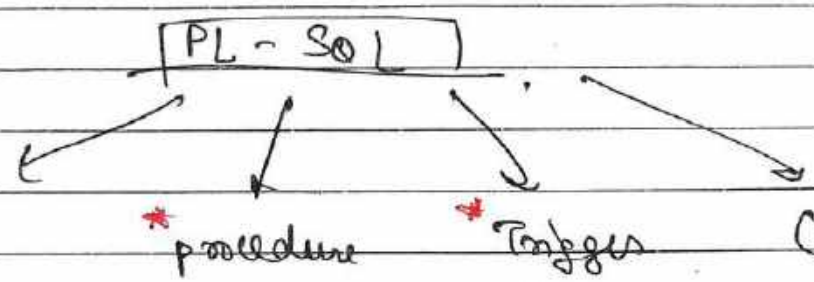
11. PL-SQL

(procedural - SQL).

⇒ SQL → Declarative in nature. (Only 'What to Do').

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

(programming flavour ही है).



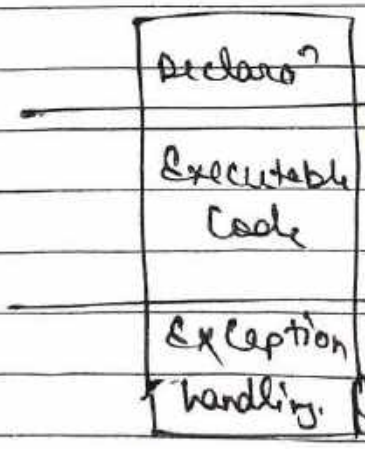
→ In SQL → (write a query).

→ In PL-SQL → (write a program, or write a PL-SQL Code

(program write करना है).

⇒ Its Block Code has 3 parts! →

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



↳ 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 never works on Hard Disk

CPU - fast.
Hard Disk → slow

---

72. TRANSACTION CONCURRENCY :→

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

(कुछ भी हम change करते, Database के अंदर, कुछ भी हम Database को Read करते, that is also a part of transac°.)

Ex:-

→ When we withdraw our money from ATM, then we have to perform a ~~test~~ set of opera°, these set of opera's called as Transaction.

→ [Work - Withdrew Money.]

A transaction generally represent change in database.

Database Transactions has 2 operations :→ Read & Write. (Commit) - we also use this.

[Read is the access of Database.
Write is change in database, we made.]

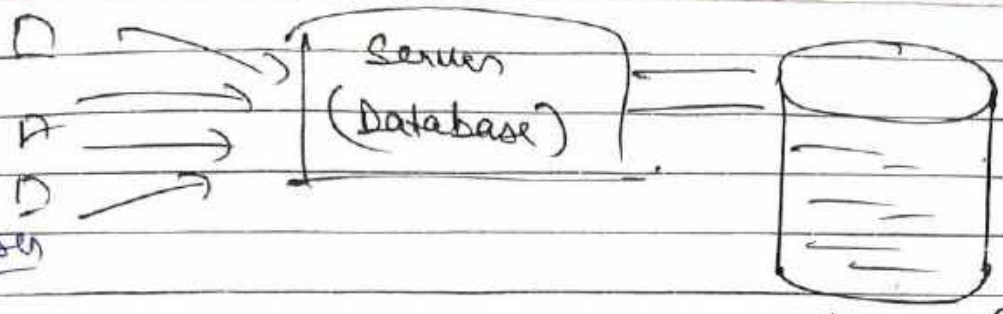
→ When we read or access any data from HDD (hard drives), then it comes to RAM (where we perform opera's).

Commit → Whatever change we made save that permanent in Hard Disk.
(Update data in HDD)

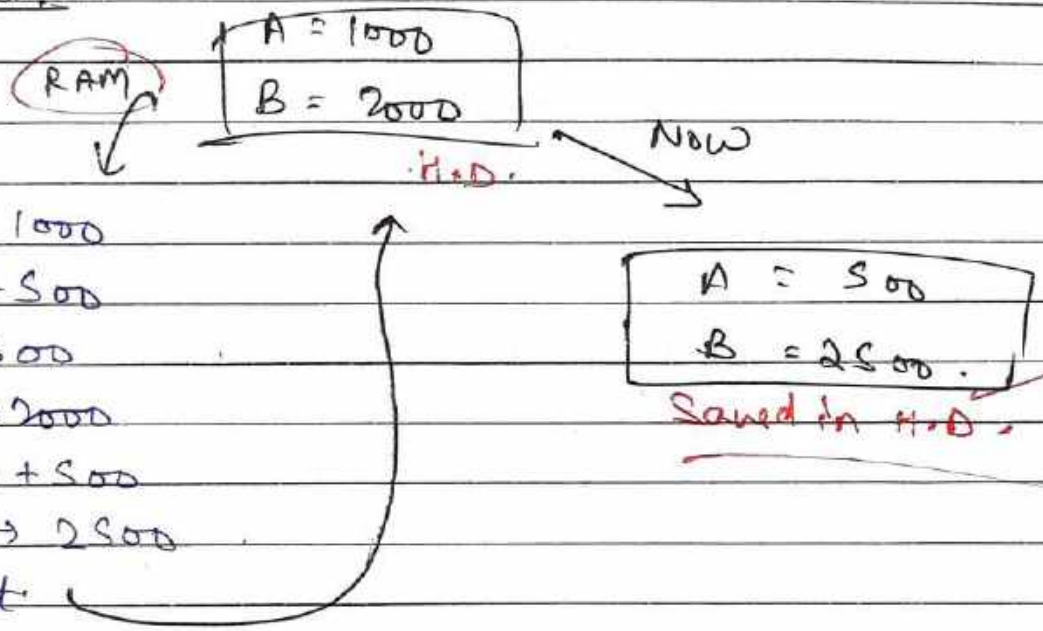
In C++ for assignment, =
In PLSQL , : =

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.

Ex:- T_1 (Transac")

R(A)
A = A - 50
W(A)
R(B)

Rollback

Commit

(बीच में commit से पहले कहीं भी fail हो गया, then Roll Back)

या तो सभी oper° execute होते, commit वरना
वरना कुछ न हो Roll back होता।

Note:- A failed Transaction cannot be Resumed.
A failed Transac° will always restart.

Consistency :→

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

Sum of money should be same.

Ex:

A = 2000
B = 3000
5000

A ——1000——→ B

T_1

R(A) 2000
A = A - 1000
W(A) 1000
R(B) 3000
B = B + 1000
W(B) 4000
Commit.

A = 1000
B = 4000
5000

Sum is same

Isolation :→

T_1	T_2	T_3
R(A)		
	R(B)	
		R(A)

parallel Transac's.

→ We try to Convert a parallel schedule into a Serial Schedule. (Conceptually)

CPU Speed is in MIPS (Million instruc" per second).
Q Hard Disk → 10/20 instruc" per second.
∴ CPU never compatible with HD for execution.

⇒ Then,
Serial Schedule is always Consistent

# Parallel	# Schedule
$T_1 \rightleftarrows T_2$	$T_1 \rightarrow T_2$
	$T_2 \rightarrow T_1$

* Durability ! →
Whatever changes we made, they must be permanent. i.e.
(Update for lifetime until we again update the data).

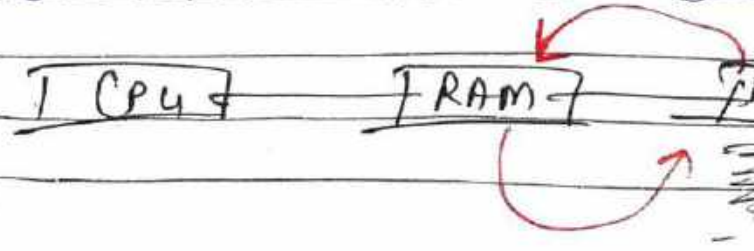
⇒ That's why, we save data in HDD for Durability.

---

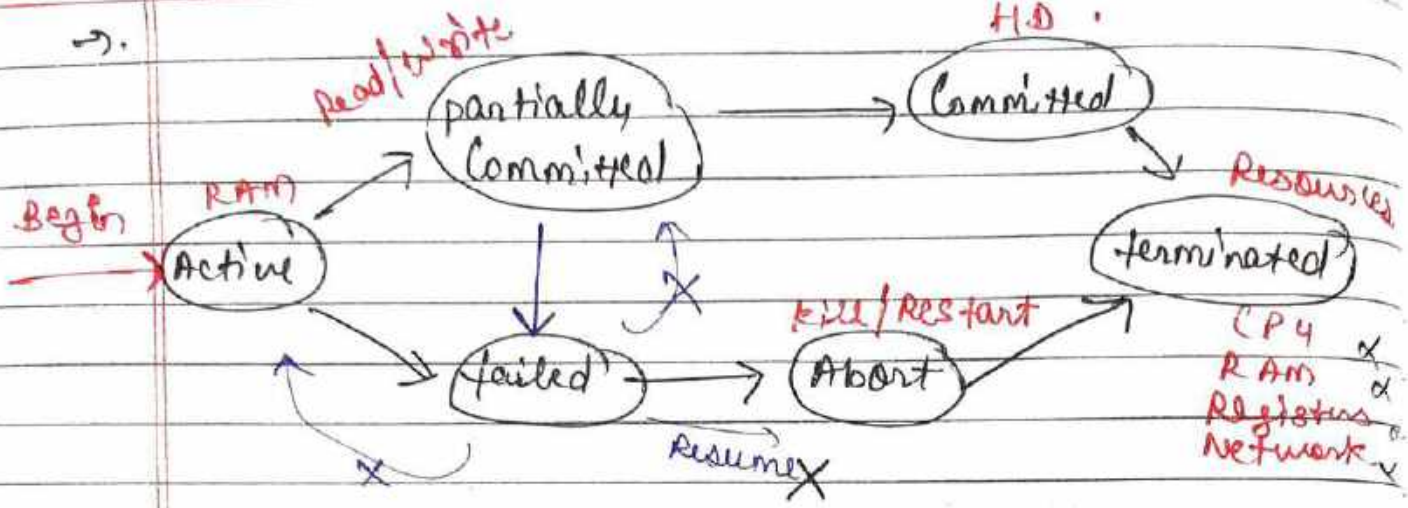
74. Transaction States ! →

⇒ At now, Transac" is in passive state. and As we start Executing it, it comes into Active state.

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



→.



→ partially Committed! →
All oper's are done except Commit.
ie. If N operation.
then
(N-1) is done.

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

→ Committed! -
Changes are now saved to Hard Disk.

→ terminated! →
Here we deallocate our Resources
ie. free all Resources. (Resources are limited).
Now, they move to anyone else.

→ failed: → power failure, switch damage, etc.
(failed either from Active or partially committed state)

→ Abort! → Rollback the opera's & Restart the opera's.

(#5) SCHEDULE! → (Serial vs parallel Schedule)

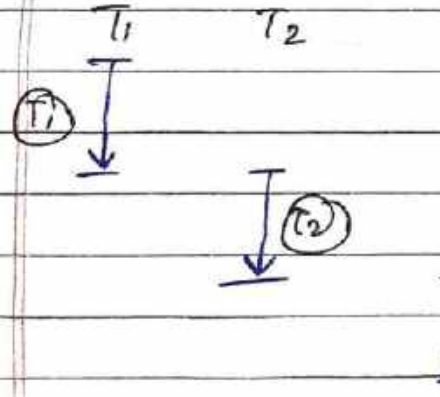
→ Schedule! → It is chronological execution sequence of multiple transactions.

T_1 T_2 T_3 — — — T_n

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

Serial Schedule! → Until a transac" gets completed, no other trans. can interfere.

All transa's executed by a serial sequence.



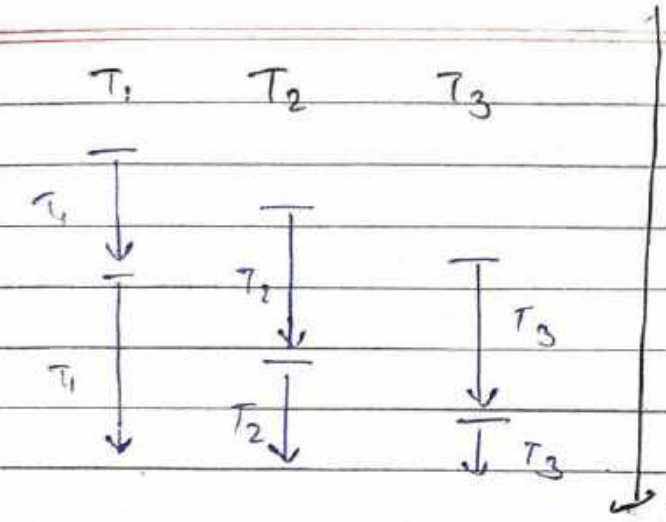
Advantage →
→ Consistent (bcz, no one can interfere).

Disadvantage →
→ When T_1 gets executed, T_2 & T_3 are in waiting.
→ Performance degrade.

Parallel Schedule! →
→ We can switch to multiple transac"s at a time. i.e.
→ Multiple transac" can execute at a same time.

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

Throughput → No. of transc's executed / time.



Throughput ∝ performance

* Advantage :→
→ performance Increased i.e, (Throughput is high.).

* Disadvantage :→
→ problm may occur . & (Inconsistent)

(*) Nowadays,
(Parallel Schedule is more performed.)
↳ bcz good performance in less time.

(Q6) Types of problems in Concurrency :→

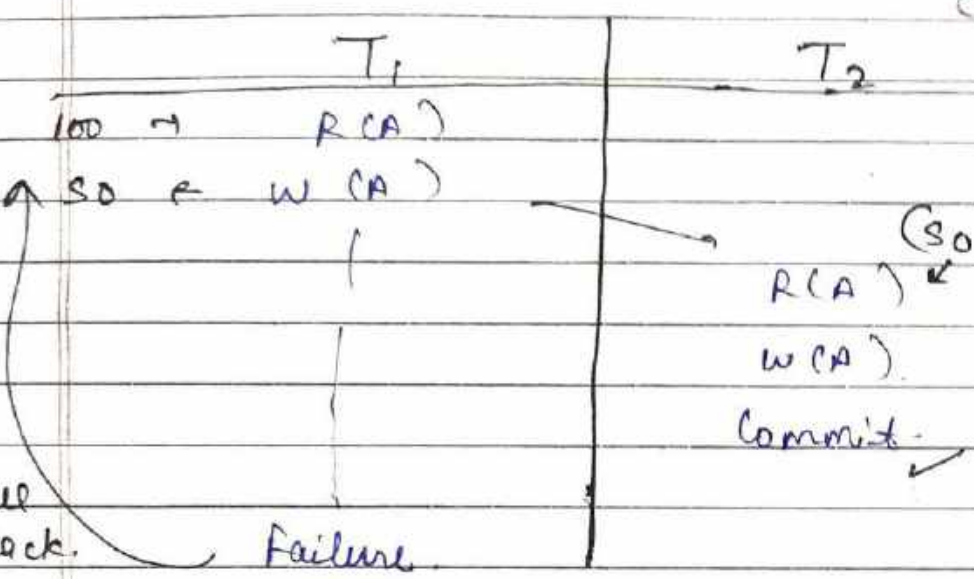
→ Concurrency, means when multiple trans. executed at a same time i.e, Parallel schedule.

→ (Mean, No problm is in Serial Schedule. There are Prblms only in Parallel Schedule)

1) Dirty Read
2) Incorrect Summary

3.] Lost update
4.] Unrepeatable Read
5.] phantom Read.

1.] Dirty Read :→ Or Uncommited Read or RAW.
(Read after write)



Hence, when T_1 gets failed. Then, how T_2 can use the A = 50. So, Dirty Read.

2.] Incorrect Summary problem :→ (T_1, T_2)
It occurs mostly when a transac° start & T_2 comes & start performing its aggregate func°s. (sum, average)
then,
we get incorrect value of sum, average etc.

3.] Lost Update :→

⇒ जो पहले (T_1) ने changes किए / उन्हें changes को (T_2) ने ओर update कर दिया / & पुराने वाले changes हमारे lost हो गए / i.e, [update किया हुआ lost हो गया]

Ex:→ Cid-1 T_1 | T_2 Cid-1
5 Likes | 5 Likes

⇒ & we finally get 5 Likes on the same product instead of 10 Likes.

Phantom - Simpdl, Shadow.

4.) * Unrepeatable Read :→

T_1	T_2
R(x) - 100	
	R(x) - 100.
W(x) - 50	
	R(x) - 50.

// Unrepeated Read.

⇒ T_2 got diff values on diff. Read. (100 & 50).

So, It is also a problem.

5.) * phantom Read :→

T_1	T_2
R(x) → 10	
	R(x) → 10.
Delete (x) [0] ✗	→ Read (x). ("It also tries to Delete '0'")

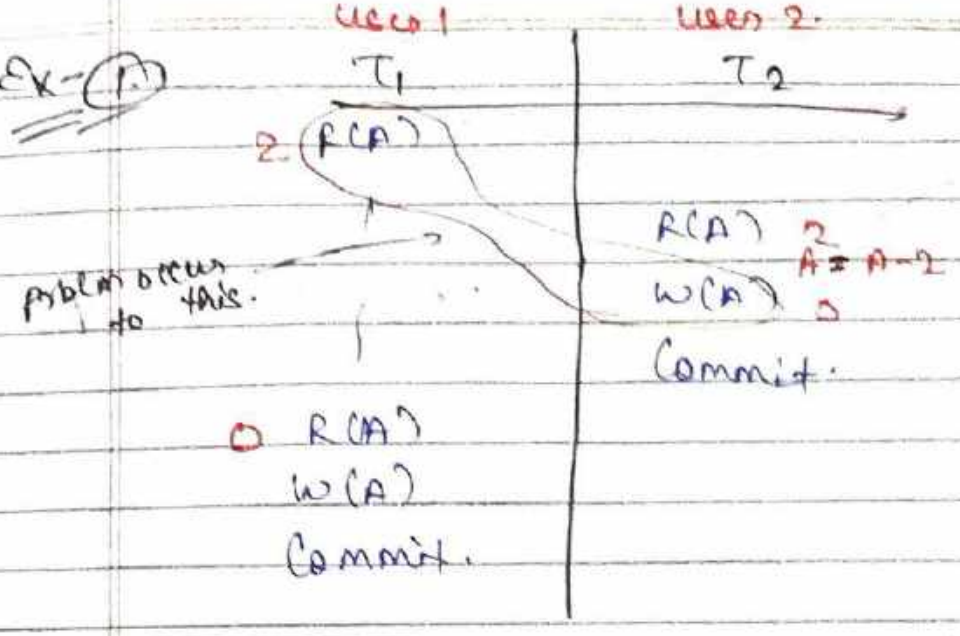
(7) Read - Write (Problem) conflict (or) Unrepeatable Read Problem :→

⇒ 4 cases we have →

Same Data.

R (A)	R (A)	✓
R (A)	W (A)	Problem.
W (A)	R (A)	
W (A)	W (A)	

Ex-(1)



(#) Ex - IRCTC

Let user 2 reserve both the seats. Then,

$A = A - 2 \quad = 0$.

→ If user 1 remained in waiting to reserve or not. & when he sees again.

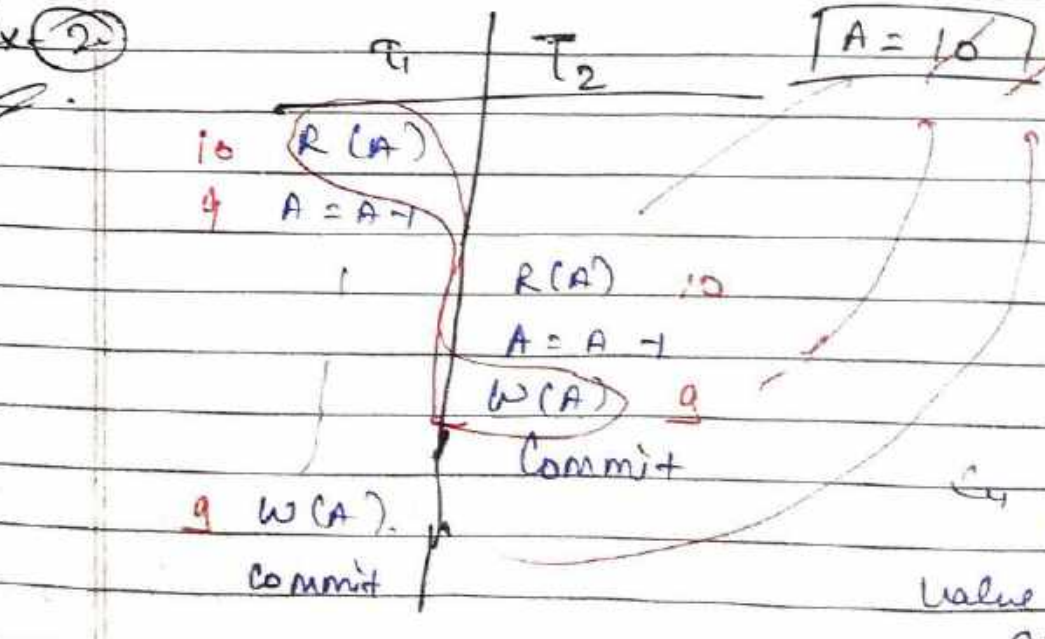
Now,

Seats becomes '0'.

So,

Now, user 1 have to be Roll back. (Abort)

Ex (2)

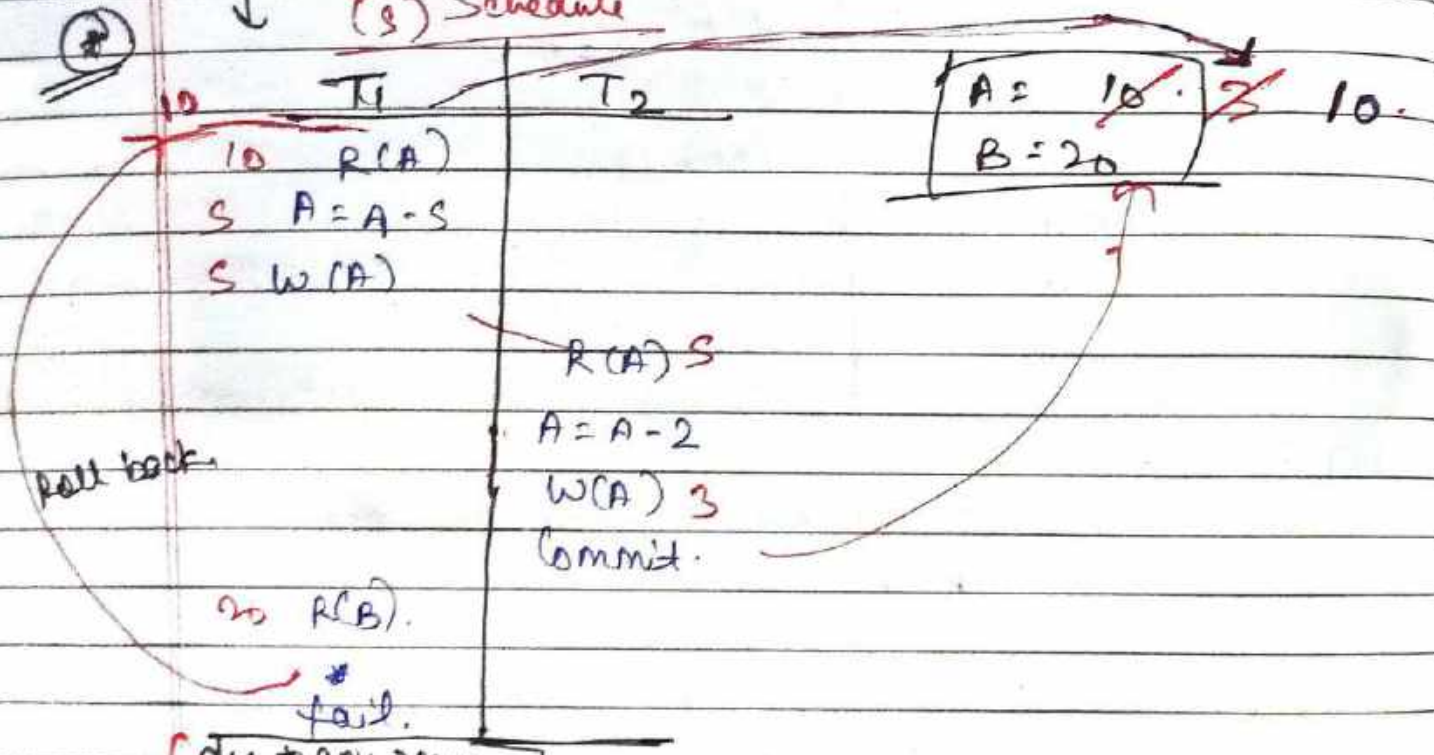


We issued 2 books but value still is 9. (instead of 8)

Date:
Page:

18. Irrecoverable Vs Recoverable Schedules in Transactions :-

↓ (3) Schedule



↳ Now, (T_1 Rolls back due to Atomicity property. ↓ (Either all or None))

∴ On Roll back, everything happens in T_1 is gone. So, Again Now, [A is 10].

But, here T_2 also done something & that is lost Now.

⇒ T_2 Change is lost. We can't recover it now. ↳ It is Irrecoverable Schedule.

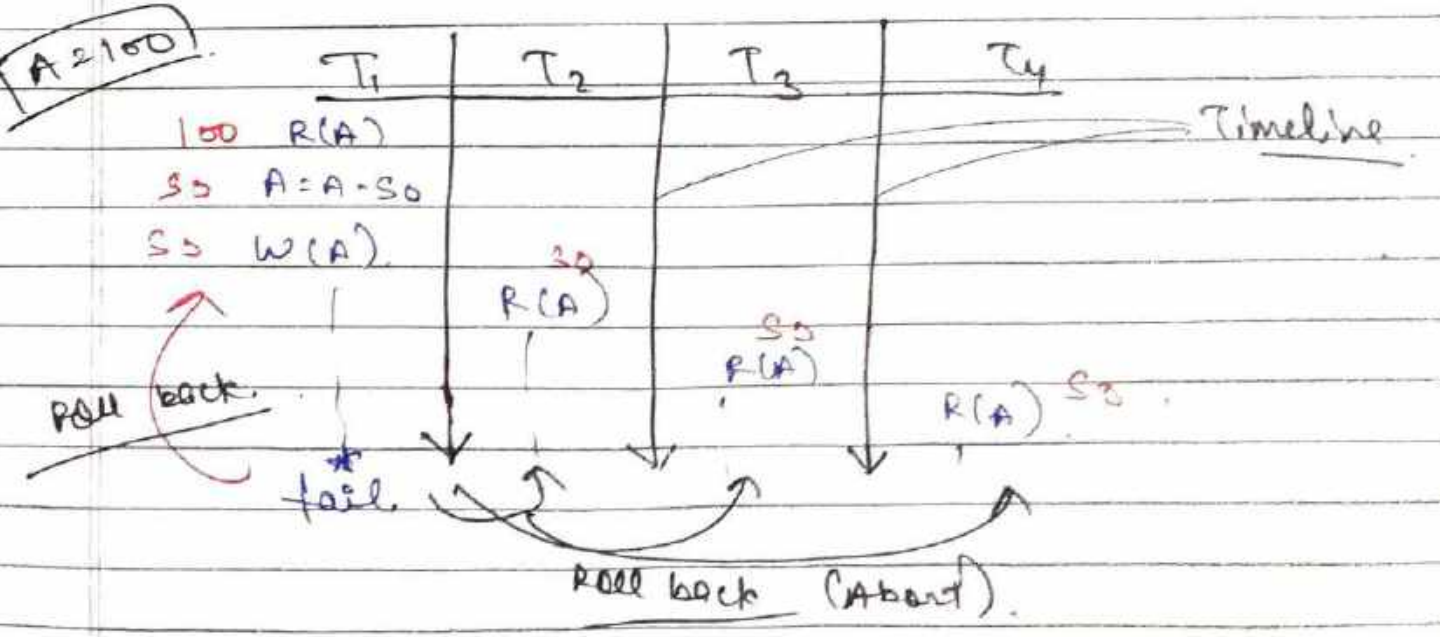
Cascade → ಕ್ಯಾಸ್ಕೇಡ್

(79.) Cascading (Vs) Cascadeless Schedule :-

↳ In Recoverability, these 5 are Important :-

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

⇒. **Cascading** :- Means Due to occurence of one event, multiple events are automatically occuring.



↳ Here, let T_1 fails due to any reason. Then, it roll backs automatically due to Atomicity & again (A is 100). But, now T_2, T_3, T_4 has (A → 50). So, they are working on wrong data. So. Now, We also forcefully Roll back (Abort) the T_2, T_3 & T_4.

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

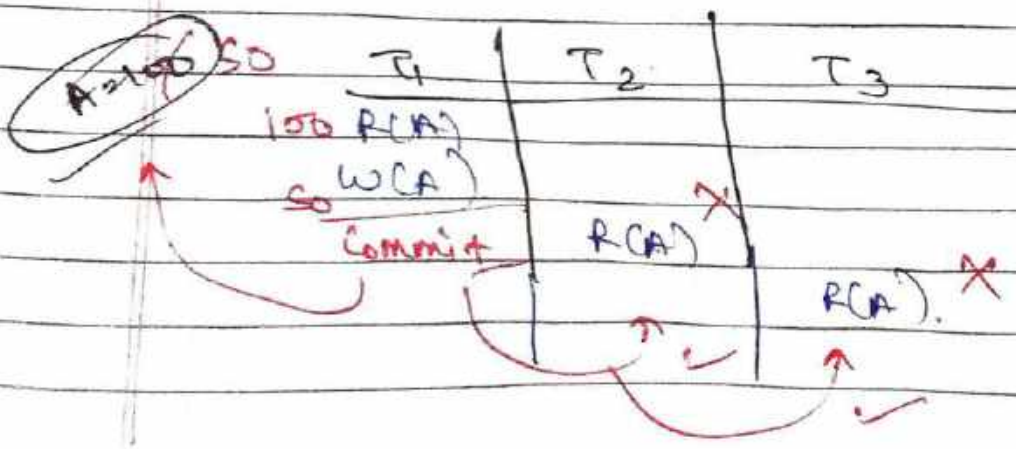
→ Here,
CPU utilisation gone waste by (T_2, T_3 & T_4).
∴
performance is bad/poor.

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

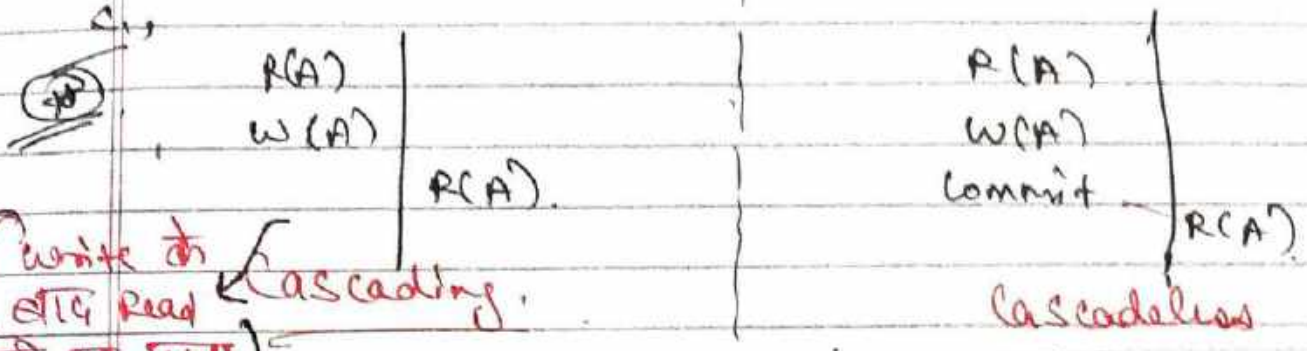
* Cascadeless :→

→ How to remove the problem of cascading?

→ Sol^n :- T_2 & T_3 can't Read the (A) value from T_1 until A value gets Committed or Roll back in T_1.
(Commit → होने के बाद तो कोई Problem ही नहीं है।)

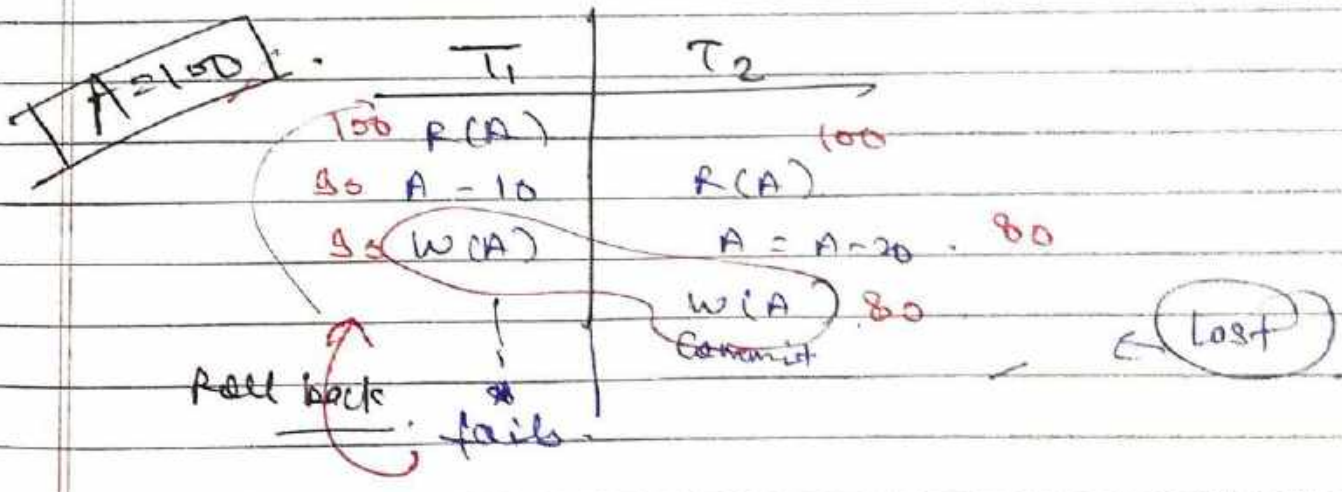


→ i, Don't allow Read in T_2 & T_3.
So,
It automatically becomes Cascadeless.



→ But, there is still W-W (Write - write) problem in Cascadeless. bcz,

(Read allow नहीं है, write तो कर सकता है।)



→ Now, when T_1 fails. (A becomes 100 again)
So
The work of T_2 automatically gets lost.
(i.e, Write - Write problem (or) lost updation problem
bcz T_2 value of W(A) → 80 तो कहीं पे Reflect होना ही नहीं। So finally A = 100

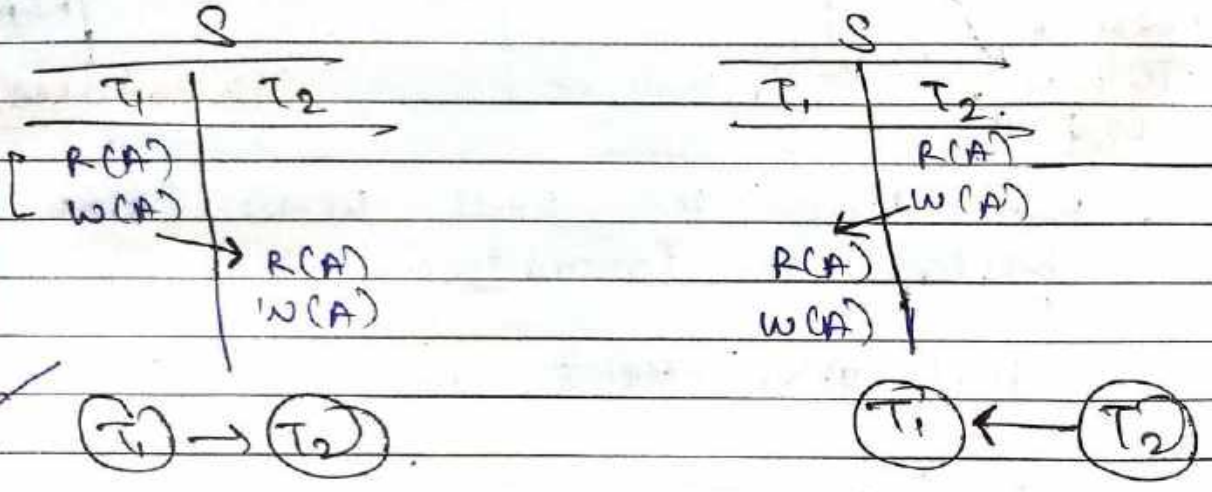
⇒ Strict Recoverable tells that we also can't write along with Read.

80. SERIALIZABILITY :→

→ (Serializability means that a Schedule has ability to become a Serializable.)

Mean,

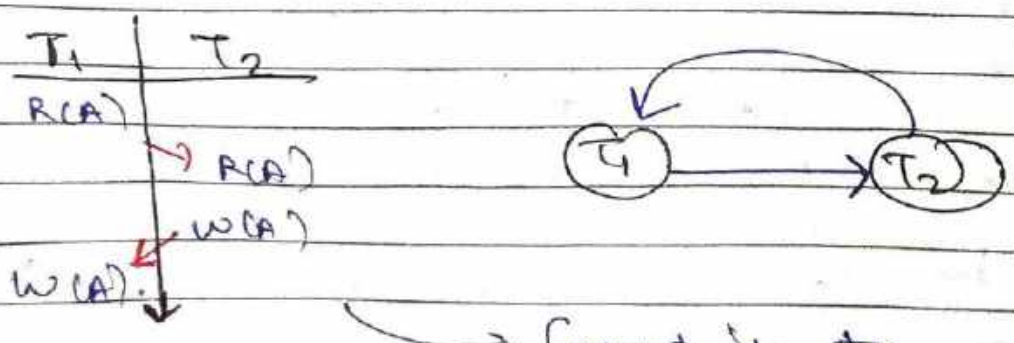
Schedule - Collection of transaction $(T_1, T_2, \ldots T_n)$



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

(तो, Serial Schedule तो ये already है, इसको कुछ नहीं कर सकते।)

→ We want, Parallel Schedule →



→ Convert it to Serial Schedule →

2 ways :-

1. $T_1 \longrightarrow T_2$

(OR)

2. $T_2 \longrightarrow T_1$

⇒ To check Serializable? Check that if there exists an Serial Schedule Equivalent to Parallel Schedule, or not. This Concept is known as Serializability.

(#) Serializability (2 method).

- Conflict Serializable
- View Serializable

→ We check that a parallel Schedule can convert to a Serial Schedule or not. → Serializable (Class of 11 Schedule).

(#) Let, a Schedule has 3 transac^n's.

S

parallel Schedule →

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

⇒ Serial Schedule

16 ways!

$T_1 \to T_2 \to T_3$
$T_1 \to T_3 \to T_2$
$T_2 \to T_3 \to T_1$
$T_2 \to T_1 \to T_3$
$T_3 \to T_1 \to T_2$
$T_3 \to T_2 \to T_1$

⇒ If we able to convert that parallel schedule in any of the 6 forms of serial schedule, then we can say that it is a Serializable.

⇒ Why we got 6 forms! →

we have 3 values (T_1, T_2 & T_3).

∴ 3! = 6 ways.

---

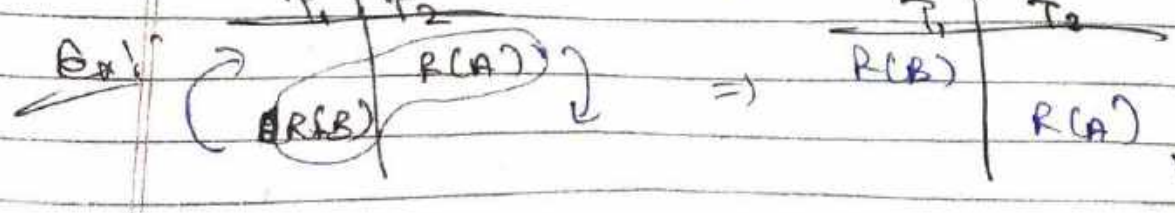
81. Conflict Equivalent Schedules! →

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

(A × B = 2 diff. ∞, np.)

How to Convert : →

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



Q: To check Conflict Equivalent :→

$S = S'$ → check

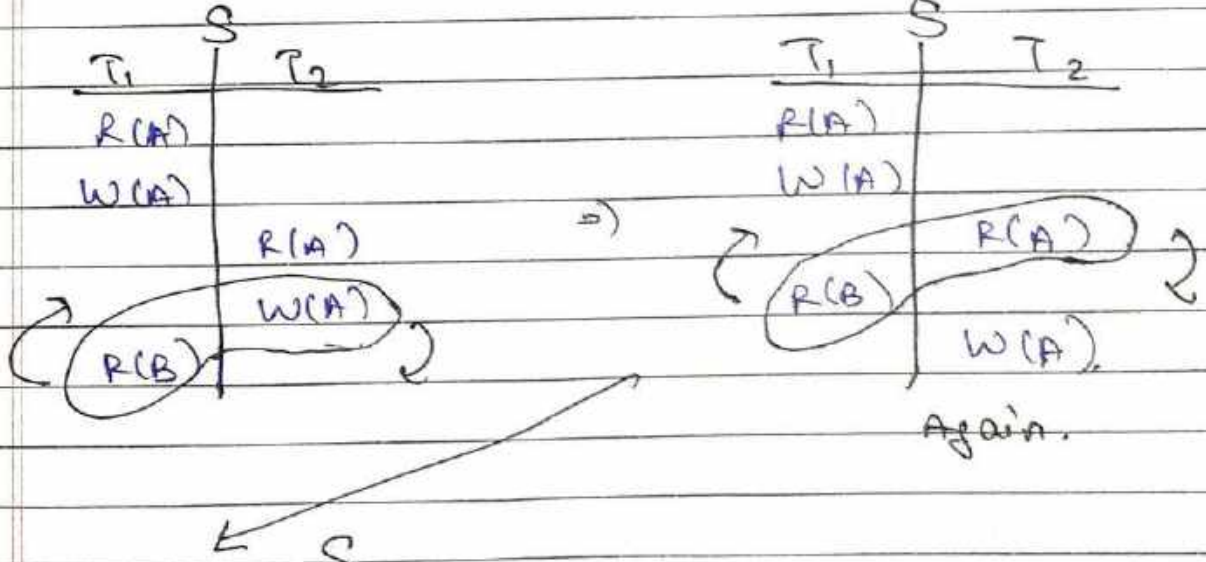
S

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	

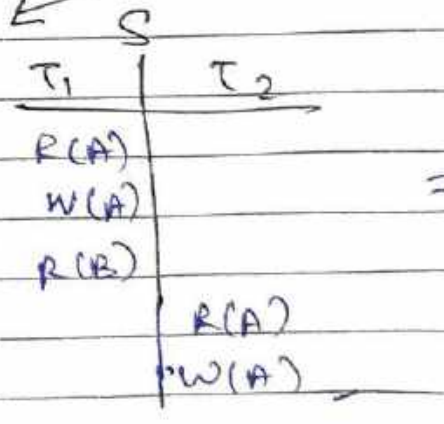
S'

T_1	T_2
R(A)	
W(A)	
R(B)	
	R(A)
	W(A)

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



Again.



Now Serial Schedule.

$= S'$

Hence

$S \equiv S'$

Hence,

S & S' are conflict Equivalent Schedule.

Ex:-

T_1	T_2
R(A)	
W(A)	
	R(A)
W(A)	
R(B).	

↓

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

Note:-> $S \xrightarrow{CE} S' \longrightarrow$ Serializable i.e, (Serial Schedule)

(CE: conflict equivalent)

(1 - 81) videos End here.