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 Abstracⁿ (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.		Quesⁿ 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 Relaⁿship (Cardinality)		28-30.
17.		One to one Relaⁿship		31-32
18.		Many to many Relnship.		32-33.
19.		Quesⁿ 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).

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

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

EFcodd → Father of DBMS

(1.) Database Management System (DBMS) :→

→ Basic Introduc^n → 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
  - Structured
    - IRCTC.
    - University.
  - Unstructured
    - Webpages
- DBMS.
  - SQL Server
  - Oracle 9i, 11, 12c, etc.
  - My SQL
  - DBL.

⇒ Database :→ Collec^n of Related Data.

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

⇒ Structured :→
RDBMS → Rela^al Database Management Sys^m.

⇒ DBMS :→ collection of operations

operations
1. Insertion
2. Deletion
3. Updation

It provide easyness to perform opera's.

⇒ Diff. Companies have made diff. DBMS.

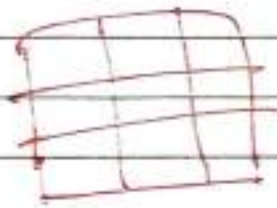
Ex :-
Microsoft Co. ⇒ SQL Server
(Sequal Server).

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

⇒ IBM ⇒. DBL.

(#) Structured Data! → RDBMS
↳ Relation.

Mean,
We stored in the table form.



(Table).

Now, Table is technically called as Relⁿ.

→ Relⁿ is most usuable form.

Now
→ Store Relⁿ's & access Relⁿ's is done by the Management System i.e. RBMS.

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

Hence,
→ We perform Insert, Delete & updation operⁿ's on Relⁿ's.

(#) Unstructured Data! → There is not predefined structure. Ex! →
A webpage is a collecⁿ 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.e. There is more technologies on unstructured data.
Ex:→ Bigdata, Hadoop, etc.

---

(3) File System Vs DBMS :→

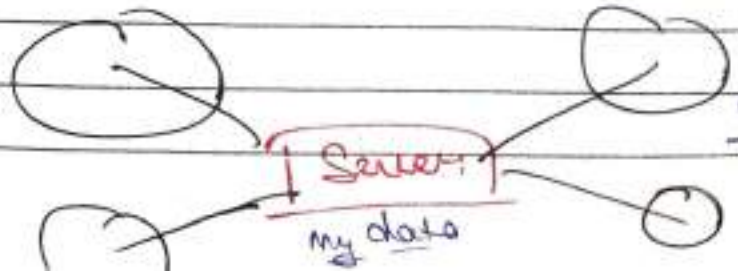
⇒ File System is used before DBMS.

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

Ex:→ CIFS, NFS File Systems.

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

→ Why we use DBMS?
Bcz, 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).
↓,
More memory usage. Now,
DBMS :→ gives us only of 1KB data from the server.

(Searching is fast & Memory utilizeⁿ is Efficient).

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

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

In DBMS, no locaⁿ, 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

Data

Student | faculty | Dean

(Role based Access Control.)

If we are users, we only get users data.
—— " — faculty, —— " — faculty —

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

5.) Data Redundancy! → (Duplicacy).

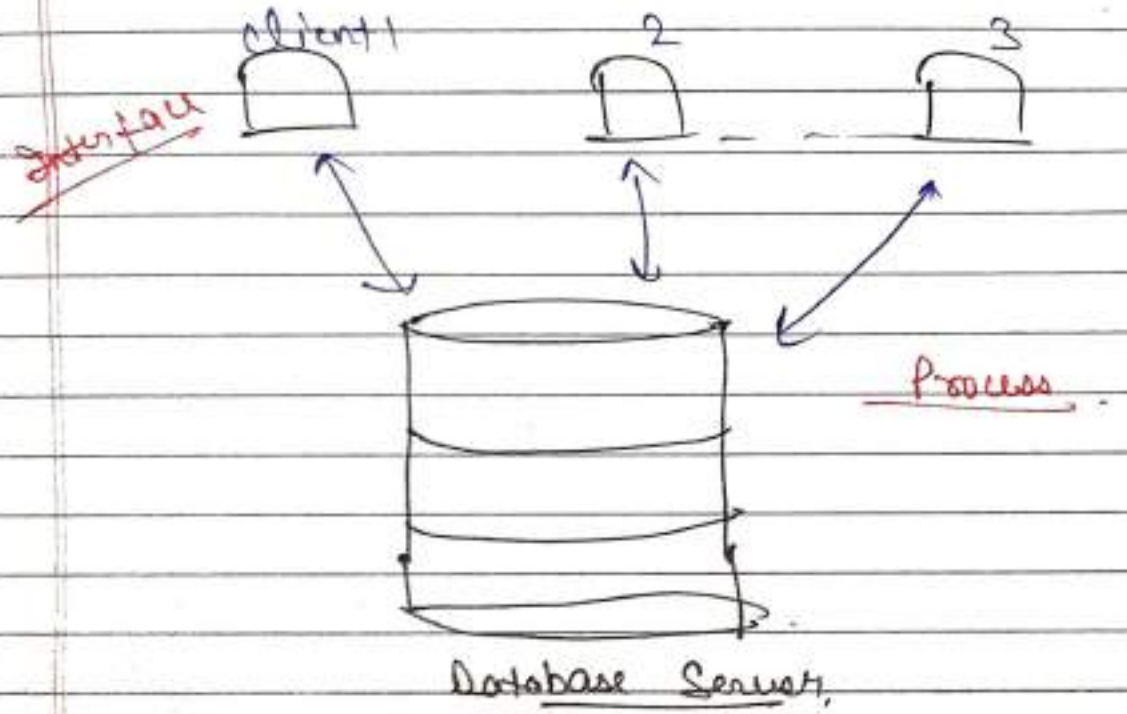
In File System, we can save same content by diff file names.
In DBMS, has many constraints to ensure unique data. Stops redundancy of data.

→ DBMS is at back end of every client server & web applicn.

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

(H) 2 tier :- means 2 layer.

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 sta^n 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)

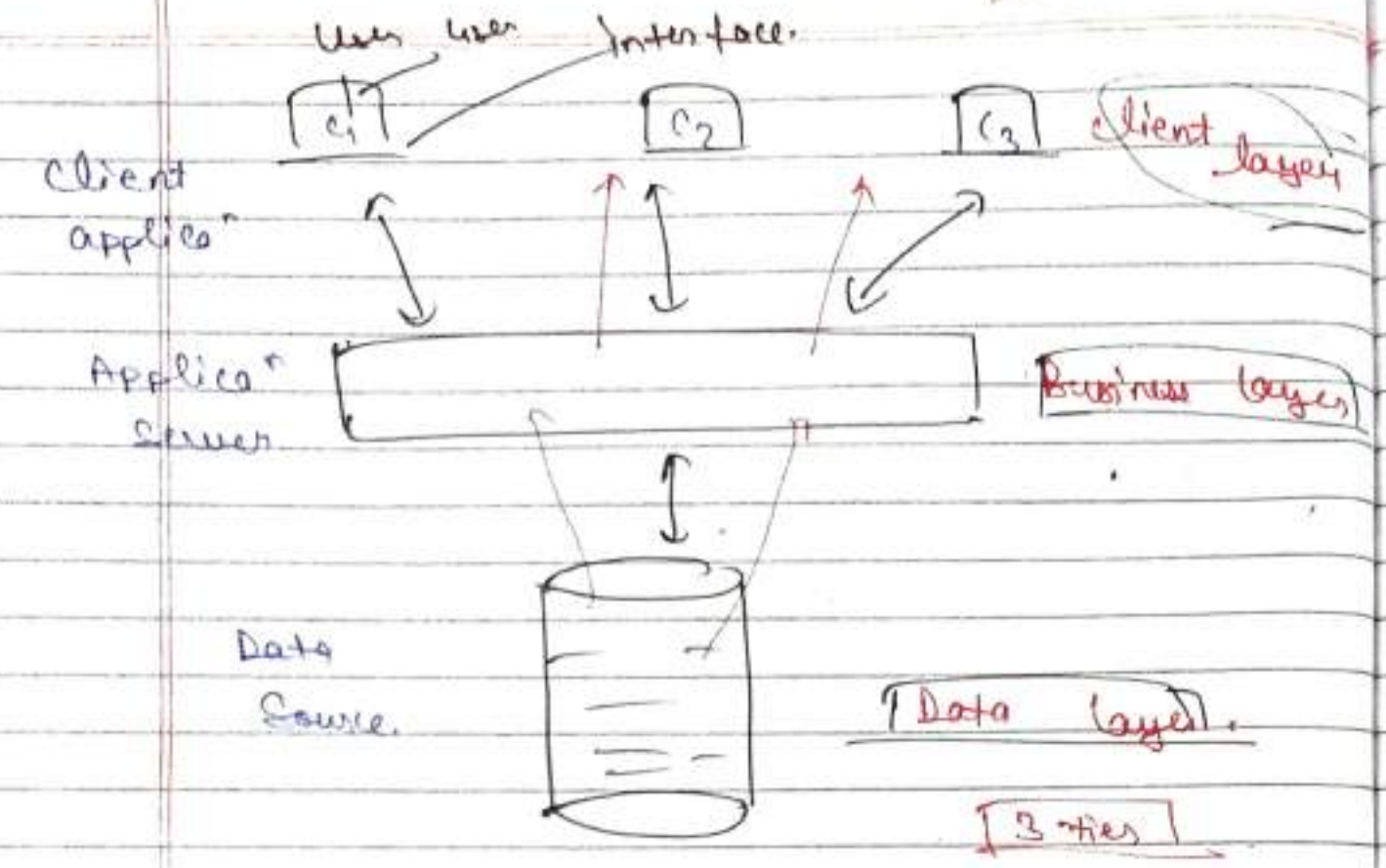
Hence,

We don't give load to Database Server here.

Hence, Business layer acts as Intermediate.

Ex:- IRCTC & banking app & G mail app

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



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

Ex:- Web Applica^ns (APPS) are kind of 3-tier Architecture.

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

---

(5) Schema → Logical Representa^n of a database.

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

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

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

Roll. No.	name	addresses

✓ Relationship.

(ii) Course :- (Schema, Entity).

C.. ID	name	Durat^n

✓ (Relat^nship).

↳ Logical Representat^n (structure).

→ But, we implement it by SQL. (Sequel).
(integer, character ...).
Data Defin^n language (DDL) → to implement Schema. (or to design)

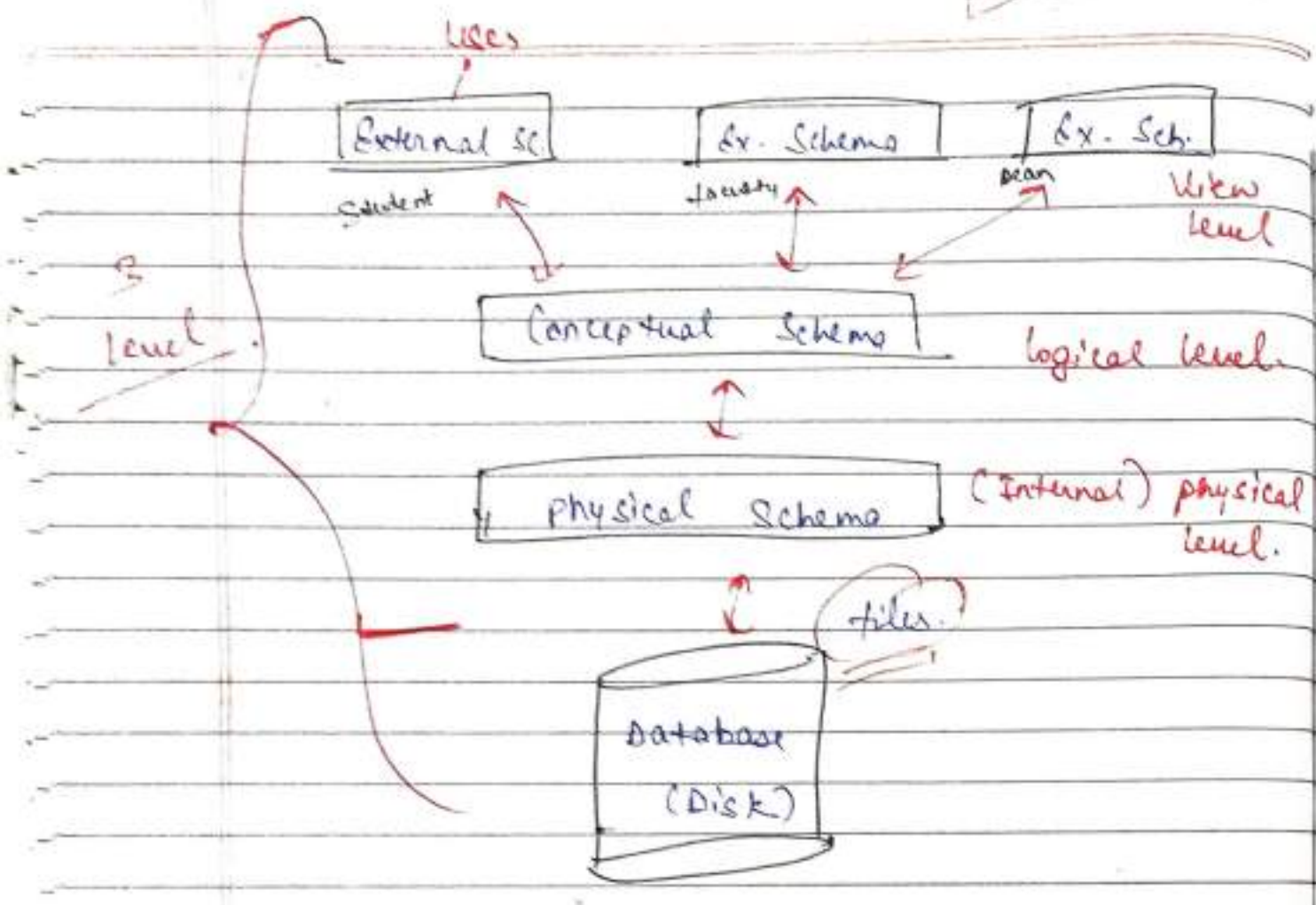
→ [Schema is simply a structure (table form).]

---

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

→ (we hide the loc^n of where the data is stored, from the user)

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



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

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

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

Conceptual Sch: E-R Model

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

→ Information of all tables that we use. & their Relationship. A type of Blueprint is this.

physical Schema :→ where the data is actually physically present. The loca^n 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.

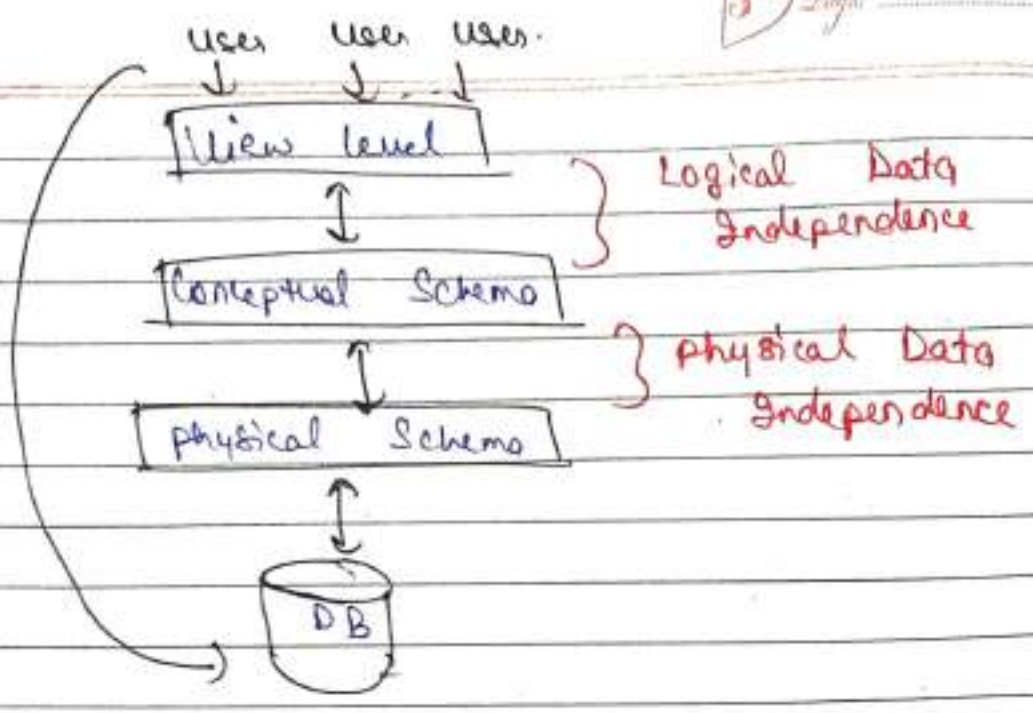
There are 3 layers b/w the user & the data

---

7. Data INDEPENDENCE :→

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

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



(#) Logical Data Independence! →

let,

→ Student Table. → add by user 1.

Name	Age	Mob. No

↓

It will not affect the Applicⁿ program.

→ We don't need to write the Appⁿ 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. colⁿ is only ∀ user 1.

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

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

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

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

Hence,
It is logical Data Independence.

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

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

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

---

8. Candidate key & Primary key :-

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

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

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

4 To identify this, we must have a key.

Ex:) Student Table

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

These all attributes can uniquely identify any 2 rows.

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

{Aadhar card, Roll No, Reg No ——— Email-Id}.

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

---

9. PRIMARY KEY :→

(Every lock has its unique key.)
i.e.
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.

Not NULL

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

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

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

→ PRimary key = { unique + NOT NULL }

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

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

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

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

---

(10) Foreign key in DBMS :→

- Foreign key (F.k.) :→ 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 →

Roll no	name	Add.
1	A	Delhi
2	B	Chandi
3	A	Mumbai

P.K (primary key).

Course →

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

(F.k.)

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

→ In Table 2, Roll No. col^n take^ 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. or Base Table.
(with p.k.).

```
* 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)

→ 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:-

P.K.

Roll No.	name	add.
1	A	Delhi
2	B	Mumbai
3	A	Chd.
4	C	Chd.

Student (Base Table or Referenced Table)

Deletion

F.K.

C. Id	C. name	Roll. No
C_1	DBMS	1
C_2	networks	2
C_3	Cat	7

Course (Referencing table).

(#) Referenced Table :→

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

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

Soln :→ 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 colⁿ is also a primary key in that other table. Then, we can't use this solⁿ. bcz,

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

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

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

(#). Referencing table: →

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

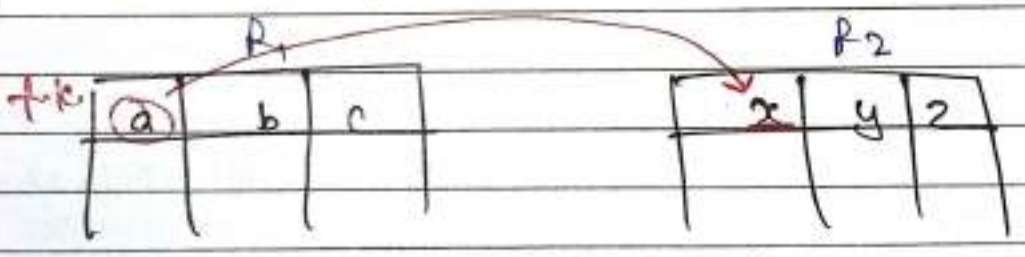
a) Insert into R_1 ✗
b) Insert into R_2 ✓
c) Delete from R_1 ✓
d) Delete from R_2 ✗

Which is correct regarding referential Integrity ?

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

Let, x be p.k. in R_2.

Sol^n:-



Referencing Table. Base Table (or) Referenced Table.

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

(13.) SUPER KEY in DBMS :→

⇒ 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, वो 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

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

Compulsory.

$1 \times \underbrace{2 \times 2 \times \ldots 2}$.

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

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

⇒

→ 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,

$\Rightarrow \boxed{2^{n-1} + 2^{n-1} - 2^{n-2}}$ ✓

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

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

then,

2^{n-2} ✓

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

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

Now,

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

So, To remove these common elements.

$\rightarrow A_1 A_2 A_3 A_4 - \underbrace{\quad\quad}_{n-4} A_n$

$\rightarrow 2^{n-4}$

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

06. $\boxed{2^{n-1} - 2^{n-4}}$

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

→ used for logical representation.

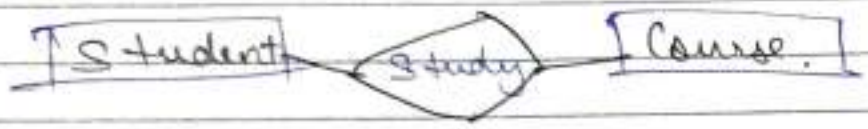
→ To see logical structure before implementation (Design).

→ It does the job of database design.

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

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

→ Relationship → Relationship b/w 2 or more Entities



Relationship b/w student & course is of study.

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

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

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

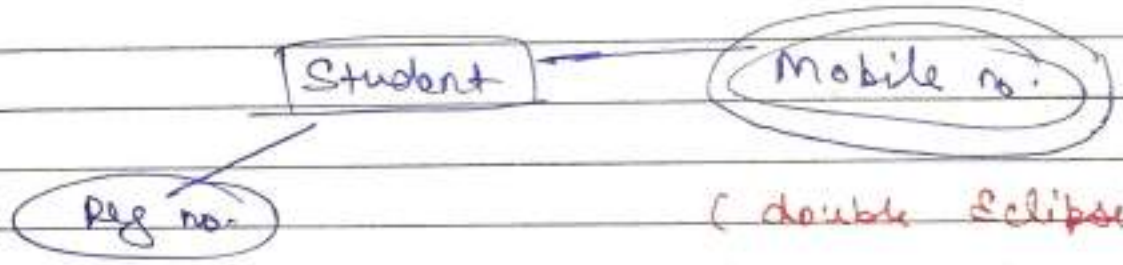
Entity → Student → represented by rectangle
Attribute →
Relationship →

15. Types of Attributes in ER Model :-

[Student]

1.) Single vs Multivalued attributes :-

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



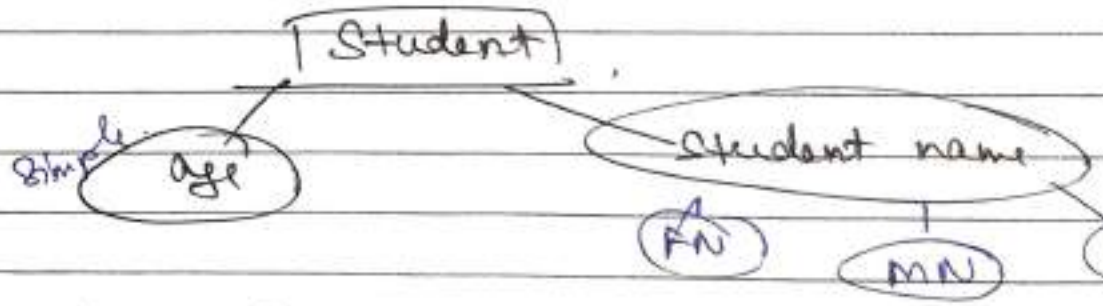
(double Eclipse).

2.) Simple vs Composite Attributes.

Simple - can't be further divided.

Composed - composed of more than 1 value.

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

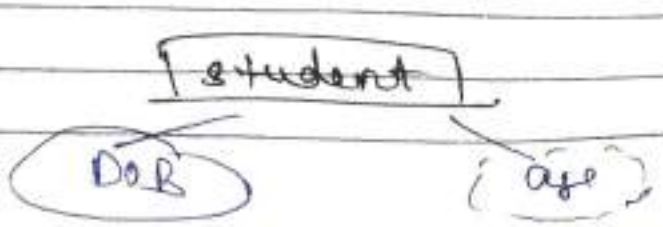


3.) Stored & Derived Attributes :-

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

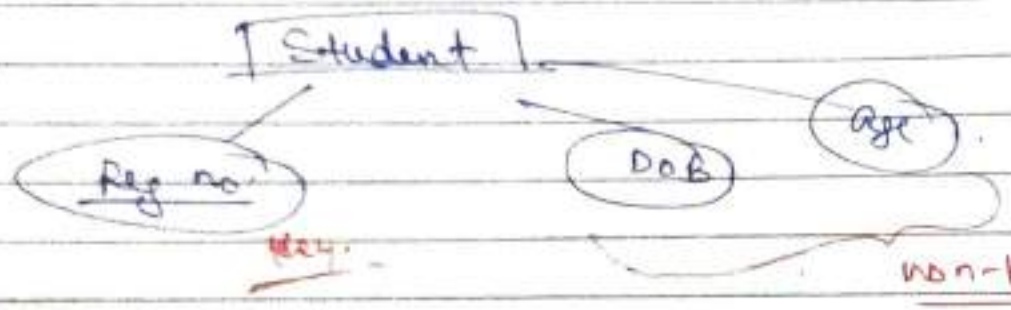
Derived → Ex - Age. (derived from D.O.B.)

dotted Eclipse.



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

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

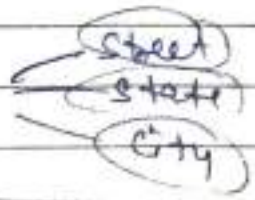


5) Required vs optional Attributes :→

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 

---

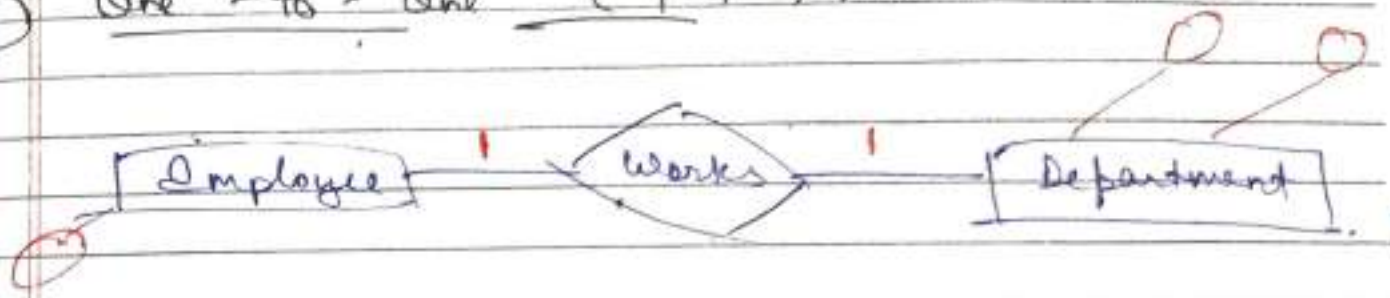
16. Degree of Relⁿship :→ (Cardinality).

~ 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 :→

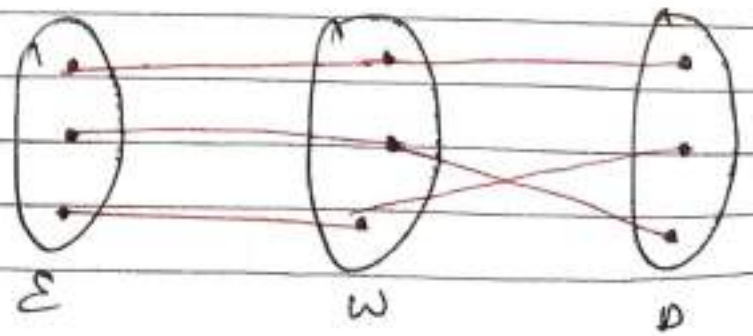
(Relo'ship का भी Table बनेगा, तो हमें Employee & Department का Rel^n बनाना है)

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^n ship table, then we have to see Rel'ship (1-1, 1-M, ---).



1-1

→ 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

E_ID (f.k.)	D_ID (f.k.)
E_1	D_1
E_3	D_2
E_2	D_3

D_ID	Dname	loc
D_1	I.T.	Bang.
D_2	Fed.	Delhi
D_3	HR	Delhi

Table 1

PK = E_ID

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

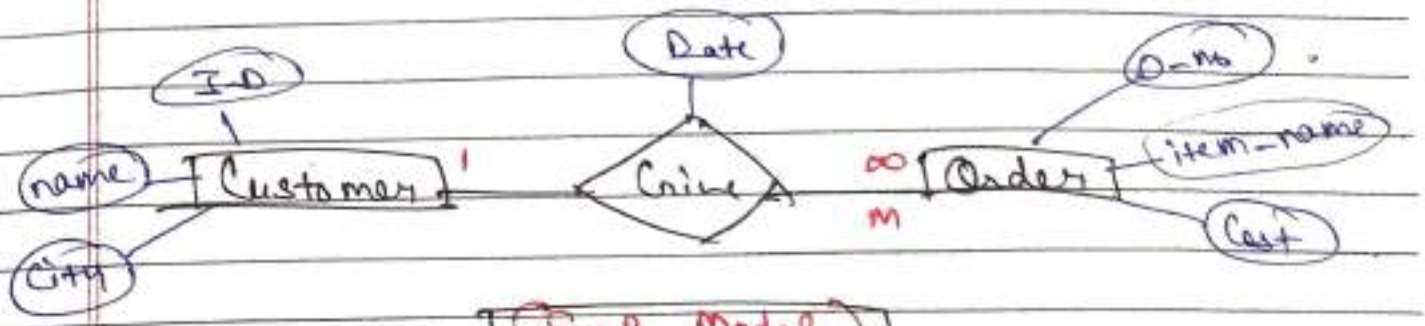
E_ID	E name	age	D_ID
E_1	A	20	D_1
E_2	B	25	D_3
E_3	C	28	D_2
E_4	A	24	—
E_5	B	25	—

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

— x — x —

Every Table must have its primary key.

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

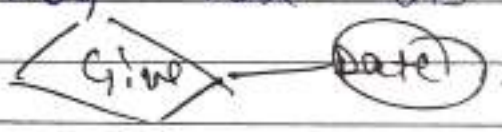


(E-R Model)

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

⇒ 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	Mumb
C_4	A	Mumb

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.
so,

P.K. = (O_no).

Note:- Always P.K. of the (many side) in (1-M) is also the P.K. of the relo^n ship Table.

Can we Merge Tables? (many side merge)

yes,

By Relo^n ship & Order Table.
(bcz, both have same P.K.).

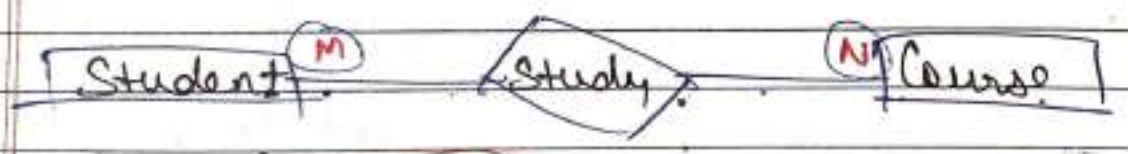
ID	O_no	item_name	Cost	Date
~	~	~	~	~

Now,

2 Tables. (M-1) is also same like this.

---

(18) Many-to-Many Relo^n ship :→
(M-N)



Roll_no	name	age
1	A	16
2	B	17
3	A	16
4	C	17
5	D	15

Base Table

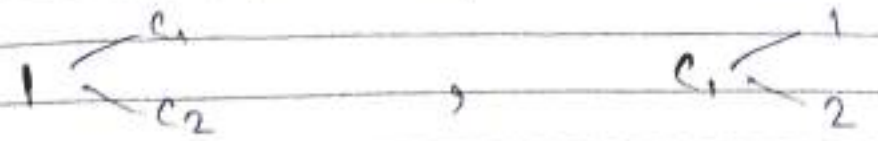
Roll_no (f.k.)	C_id (f.k.)
1	C_1
2	C_2
1	C_2
2	C_4
3	C_3

Referencing Table

C_id	name	Credit
C_1	Maths	4
C_2	phy.	4
C_3	Chem.	4
C_4	Hindi	4

Base Table

Many - Many



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

Roll. No seperates &
C-id also seperates

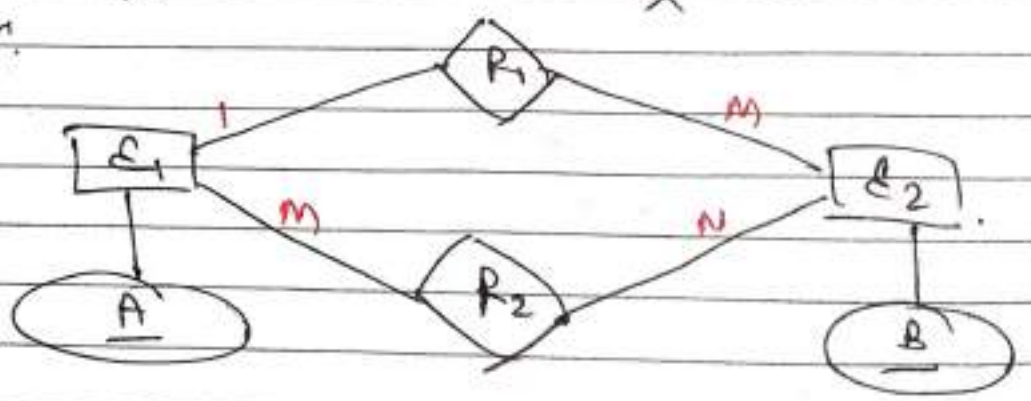
So, Roll no. & C-id both make P.K. combinely.
i.e,

$\boxed{\text{Composite key} = \text{Roll no. C-id}}$

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).

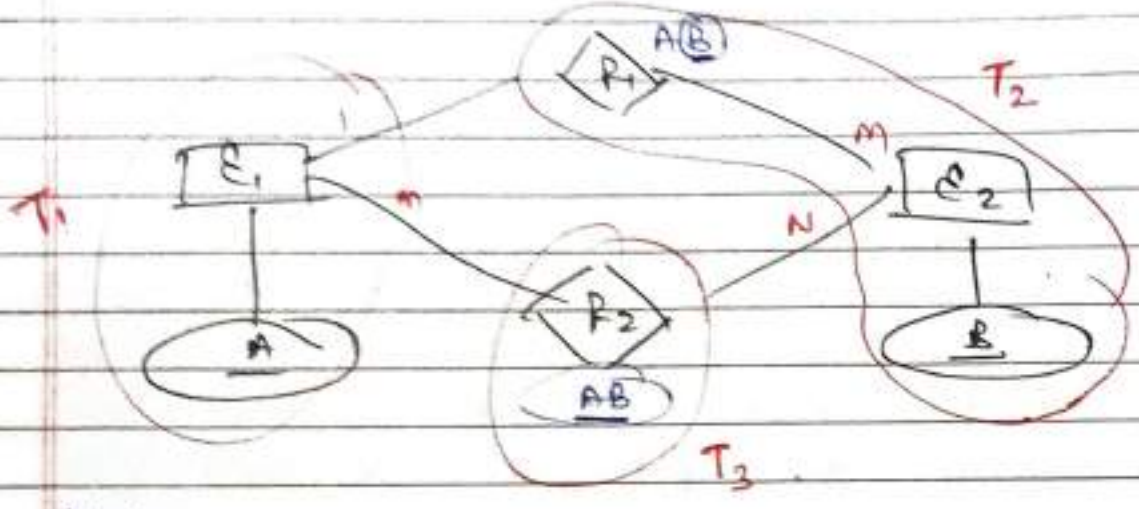
Q:→



→ 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 सारी attributes $R_1 E_2$ के Combined Table में आ जाएगा। So Now, we also don't need separate E_2 Table. $\boxed{Min^m = 3}$ ✓

Normalization :→

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

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

(#) Row level :-

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

Same. (Duplicate) =

Row level

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

Primary key (Unique + Not Null).

P.K. will take care of this duplicacy.

(#) Column-level :-

	student		course		faculty		
P.K.	S-Id	S-name	C-Id	C-name	F-Id	F-name	Salary
	1	Ram	C_1	DBMS	F_1	John	30,000
	2	Rani	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 times.

→ ~~Normal~~

→ Insertion Anomaly
→ Deletion "
→ updation "

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

Now,

S.Id	S.name	Cid	Cname	Fid	fname	Salary
~~1~~	~~Ram~~	~~C_1~~	~~DBMS~~	~~F_1~~	~~John~~	~~30k~~ 40K
2	Ravi	C_2	Java	F_2	Bob	40k
3	Nitin	C_1	DBMS	F_1	John	~~30k~~ 40K
4	Amitpal	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 problem here.

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

| Delete from student where S-id = 2 | → Row 2 is deleted fully from database. |

Now,
Row 2 is blank there.
Now,
tell us who is teaching to Roll No. 2
& what was the Course name of Roll. No 2.

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

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

Course Info — lost &
Faculty Info — lost.

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

3) <u>updation Anomaly</u> :→

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, Normalizⁿ removes Redundancy.

How ?

→ a simple solⁿ 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 solⁿ.

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

---

21. First Normal Form :→ (1 NF)

Ef Codd → father of D.B.M.S

→ Table should not contain any multivalued attribute.

student

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

→ Not in 1st NF

[Null 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 $A \to B$, $B \to C$, $C \to D$ }.

(Functional Dependency).

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

Here, A is determing B (from FD ①)

closure sym. ← $A^+ = B$

↳ $A^+ = BCDA$.

$A \to B$
$B \to C$
Transitive.

(A can determine itself also)
Ex:- Roll no. can determine itself.

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.e. 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.

bcz

C.K. is always minimal.

&

In AB only A is C.K.

& XtoA is super key (S.K.)

AB

super key

(A is Saath adds any thing, becomes S.K.)

bcz

AB is a super key.

Ex:

R(ABCD).

FD = {A → B, B → C, C → D, D → A}.

$A^+ = ABCD$

$B^+ = BCDA$

$C^+ = CDAB$

$D^+ = DABC$

∴ (C.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}. {bcz, all ↑ are C.K.}

&

Non-prime att. = {φ}. → NULL.

Q:- R(ABCDE).

FD = {A → B, BC → D, E → C, D → A}.

⇒ Now, we have to check that which attribu are coming on the Right side. bcz, Attributes on the Right side, ही वो determine तो करते हैं।

= BDCA. (E नहीं आया)

E = BDCAE

[Note :→ Each & Every candidate key must contai. E. bcz, E if present on left side, then only it is written in right side also.]

जो Right side में नहीं आ रहा। यानि 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.

$BE^+ = BECDA$
$CE^+ = CE$ ×
$DE^+ = DEABC$

$\boxed{C.K. = \{AE, BE, DE\}}$ Ans.

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

F.D = {A → B, BC → D, E → C, D → A}.

So,
directly DE is also becomes the C.K.
now check ∀ D, it is depend on 2. So, check ~~them both~~ (used in making C.K.).

$\boxed{\text{prime Att.} = \{A, B, D, E\}}$

$\boxed{\text{non-prime att.} = \{C\}}$ 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
2 → Ranjit } ∴ These 2 are diff.

Ex:-
1 → Ranjit
1 → Ranjit } Same Student

Valid case.

Ex:- 1 → Ranjit
2 → Karan } Valid.

Ex:- 1 → Ranjit
1 → Karan } 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 ∩ RHS ≠ ϕ (Never NULL).

Ex:- Sid Sname → Sid.
∩ Sid. ✓ Valid.

2.) Non-Trivial F.D.:→
If $x \rightarrow y$
then,
y is not a subset of x.

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

Ex:- Sid → Sname
Sid → phone no.
Sid → 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 Trivial

$X \rightarrow Y$. (Sid → Sid).

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

(Sid → Sname
Sid phone → Sname phone).

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

Sid → Sname & Sname → City
Sid → City.

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

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

$XZ \to YW$

× ——— ×

24. 2nd Normal Form :→ (2nd NF).

2 Rules

→ Table or Relⁿ must be in 1st NF.

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

(There should be no partial dependency in the Relⁿ).

Ex:

part of CK ← non prime

AB

C.K.

A → C

(a part) (non-prime)

It is partial depend

not 2nd NF

Ex: →

Customer

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

C.K. : Customer -Id Store Id

Prime attributes: C - Id
Store - Id.

Non prime : Loca^n

Here, Loca^n is only depend on Store - Id.
i.e. partial dependency.

(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	Bangalor
3	Mumbai

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

2nd NF.

Q:- R (ABCDEF)

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 some of the values.)

So,
On LHS, there must be CE.
(C.K. जो भी बनेगी, उसमें CE तो होगा ही)

CE = FADB

Now,

EC^+ = ECFADB (all 6 are present)

So, 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

Step 1 is comp. here,
After finding C.K. = {EC}

Proof:- check
$A^+ = AB$
$B^+ = B$
$C^+ = CF$
∴, 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 }.

Rules
(Either proper subset of C.K.)
for partial dependency i.e. not in 2nd NF
Check on LHS → Either 'E' or 'C' [redacted]).
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	{ PD }	partial
E → A	{ PD }	.
EC → D	{ FD }	fully.
A → B	{ FD }	.

→ 1 भी P.D. मिलते ही, Table is not in 2nd NF.

2S. 3rd NF :→

→ Table or Rel' must be in 2nd NF.

&

→ There should be no transitive dependency in table.

Non prime or Non-unique
prime or unique.

not sufficient condⁿ.

(NPA)

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

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

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}.

(NPA → NPA).

Here Roll → State and state → city.

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

So,
It is not in 3rd NF.

Ex:→ R (ABCD)
FD: {AB → C, C → D}

⇒ C.K. = {AB}
PA = {A, B}
NPA = {C, D}

AB^+ = ABCD.

Transitive

C → D
NPA NPA

It is not in 3rd NF.

C.K. + any thing = S.K.
Super Key.

Ex:→ R (ABCD).
FD: (AB → CD, D → A).

Sol^n: 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: Now, for each F.D.
LHS → C.K. or S.K
[OR]
RHS → is a 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.

~~Here,~~
~~[LHS → C.K. or S.K]~~
~~Table only [OR] C.K.~~

Student

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

Table is in 3rd NF already.

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

F.D. :=
- Roll no → name
- Roll no → voterid
- voterid → age
- voterid → Roll no.

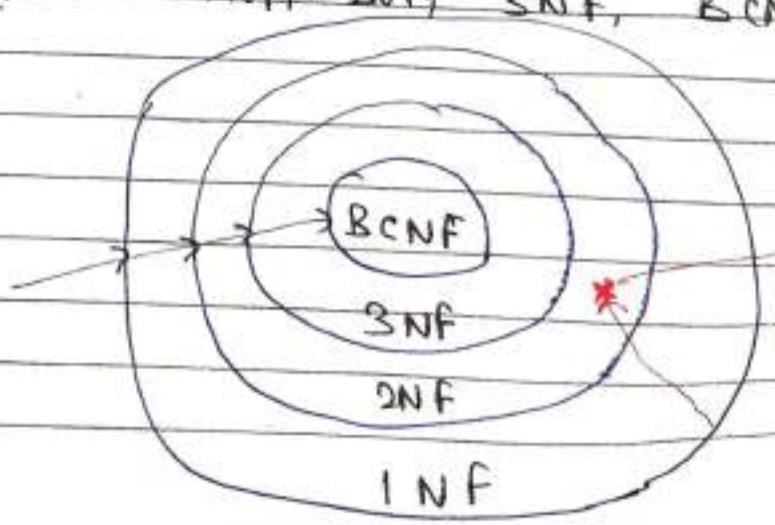
Note:→ LHS of each FD should be C.K. or S.K.

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

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

Sol^n:
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.]

(#) Compare:- 1NF, 2NF, 3NF, BCNF :→



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

ie,

1NF.

2NF = 1NF + condi^n

3NF = 2NF + condi^n

BCNF = 3NF + condi^n

---

27 Lossless & Lossy Join Decomposi^n :→

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

R

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

→ R_1 (AB)

→ R_2 (BC)

We divide this table into R_1 & R_2

Q B is common in both the Table.

R_1

A	B
1	2
2	2
3	3

R_2

B	C
2	1
2	2
3	2

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

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

So,

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

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

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

condi^n :- Common ~~element~~ col^n 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 (त्रुटि). 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 Lossy? → (2 tuples more)

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

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

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

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.) $R_1 \cup R_2 = R$

$AB \cup AC = ABC$.

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

$AB \cap AC$

$A \neq \phi$

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

To take common attribute →

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

5 forms here

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

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

Varun → 3 Phone no.
→ 3 Mail Id

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

table

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

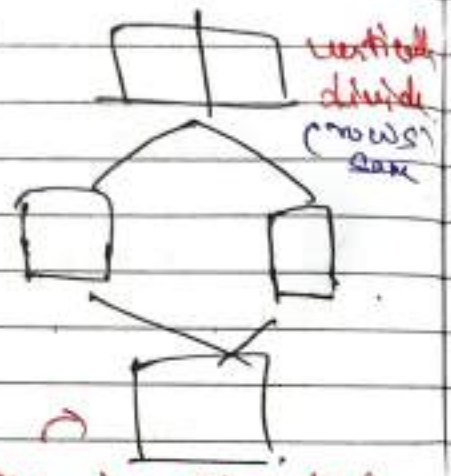
→ Very long
→ We also don't able to form a key.

So, Make 2 table.

Varun	phone

Varun	Mail

(Now, no multivalued dependency)



May be xtra 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

Ans:- 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. :-

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

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

$A^+ = A$ (not all ✓).

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

→ Same check this for every F.D. :-

$D \to B$ is redundant.

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

i.e.,

[remove, $D \to B$]

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

Now, for $AC \to D$

$AC^+ = ACB$

So, also include $AC \to D$ ✓

Now, we get

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

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

Here, $AC \to D$

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

$C^+ = CB$

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

$AC \to D$ ही रहेगा।

Now check for A, by removing C.

$A^+ = AB$.

So, can't remove C.

So, $AC \to D$ can't be reduced.

So;

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

So,

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

(30) Question on Normalization :→

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

F.D. : {AB→C, C→DE, E→F, F→A}.

Solⁿ:- 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 +)

So, [(AB is C.K.)]

Now, check ∀ A on RHS.

we get, F → A

So, [(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.

$AB \to C$

AB is already an C.K.
So, we get all C.K.

$\boxed{C.K. = \{AB, FB, EB, CB\}}$ ∴ 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 → DE, E → F, F → A}.

Now,
check 1-by-1.

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

Redundancy decrease.

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

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

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

$AB \to C$	$C \to DE$	$E \to F$	$F \to A$
✓	×	×	×

↓

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.

	$AB \to C$	$C \to DE$	$E \to F$	$F \to A$
BCNF	✓	×	×	×
3NF	✓	×	✓	✓

∴ not in 3NF :→

→ Now, 2NF :→

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

Check! →

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

↓

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

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

∴ Ans is 1st NF.

bcz,

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

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 मिलती है।
(Assume already).

∴ 1st NF.

proper Subset is always less than a set.

(31) Find out Normal Form of a Relⁿ :→
(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}

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

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

Condⁿ ⇒ { 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, T & T, T & F, T & F, T & F
FD, PD, FD, FD, FD

(Full dependency).

∴ not in 2nd NF.

to make it in 2nd NF :→
We do it by decompose the Table.
(Divide into 2 parts).

Date :
Page :

Condⁿ :-

Common attribute must be C.K.

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

Now, ~~ist~~ problem ते जो, $C \to D$ अलग अलग हो, Alag Table बना दो।

Now.

$$R(ABCDEF)$$

R_1 $(ABEF)$ $\qquad$ R_2 (CD)

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

Criteria ∀ common :- can be C.K. of any R_1 & R_2.

So, In R_2 (CD)

$C \to D$ $\qquad$ $C^+ = CD$ (all 2).

$\therefore$ $\boxed{C \text{ is C.K.}}$

So,

Make 'C' common in both R_1 & R_2.

Now, again

$$R(ABCDEF).$$

R_1 $(ABCEF)$ $\qquad$ R_2 (CD)

R_1 $\{AB \to C, C \to E, E \to F, F \to A\}$

2nd NF

R_2 $\{C \to D\}$

& F.D. $\boxed{C.K., SC}$

2nd NF.

C itself is not its proper subset

Now!

→ Check ∀ 3NF! →

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

So,

R_1 (ABCEF)

{AB→C, C→E, E→F, F→A}

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

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

→ It is 3rd NF.

R_2 (CD)

{C→D}

C.K. = {C}.

PA = {C}.

It is also 3rd NF.

Now!

→ Check ∀ BCNF! →

Condi^n → L.H.S. must be a C.K.

R_1 {AB→C, C→E, E→F, F→A}

not in BCNF term.

→ There is Redundancy.

R_2 {C→D}

It is BCNF term.

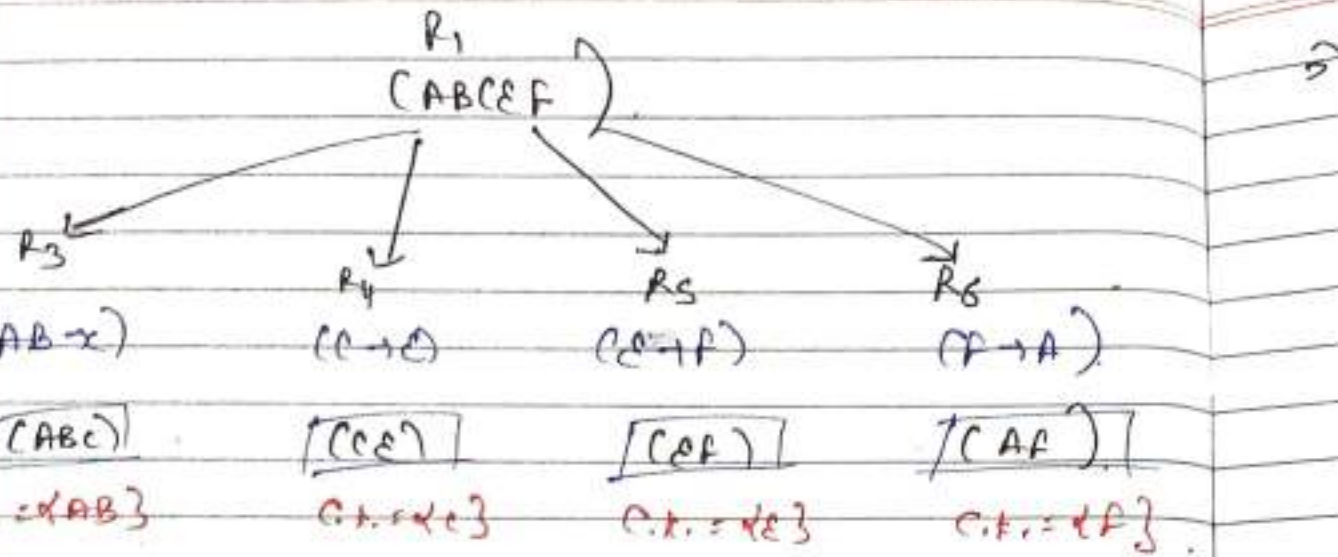
Redundancy – 0%.

Now,

Again Decompose R_1 (ABCEF).

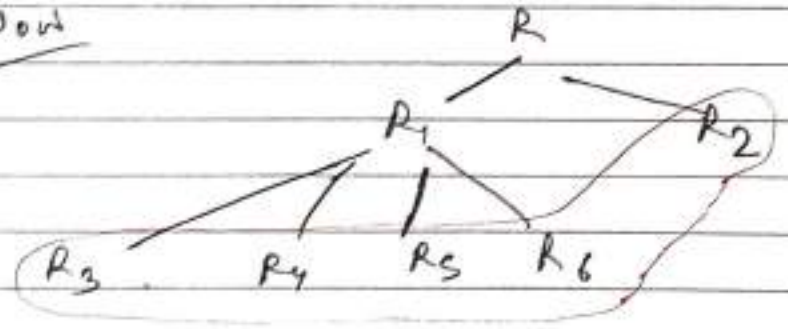
(जो problem करे दे, उसको Alag करदो।)

Normalisation aim → To make redundancy 0%. (5M) → use decomposition to normalise.



Now, these all 4 tables have C.K.
So,
these all 4 are in BCNF now.
& (कोई - भी - कोई common भी नहीं है सब में)

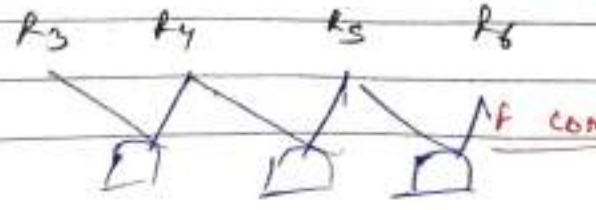
Now



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

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

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



⇒ But, Join R_3 & R_5.
(ABC) (EF).

(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^n 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:
F P.A.
T.

∴ (T)

It is in 3NF.

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

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

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

→ check if BCNF:

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

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

→ check ∀ 2nd NF :→

condⁿ :- LHS must be a proper subset of CK.
[AND]
RHS must NPA.

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

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

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

Scheme 4 :→ Registⁿ (roll no, C-Id, Credit).

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

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

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

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

So, not in 3rd NF.

→ Check ∀ 2NF ! →

1st FD → ✓
2nd FD → C-Id → Credit
T T
(→ T)

∴ It is Partial Dependency (P.D.)
so,
It is not in 2nd NF. ≠

∴
→ Table is in 1st NF. ids.
(we already assume it).

(34.) Cover & Equivalence of F.D. : →

If X covers Y $Y \subseteq X$
& If Y covers X $X \subseteq Y$
& if both are True.
then, $\boxed{X \equiv Y}$ → Equivalent.

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

i) X covers Y : → $(Y \subseteq X)$
(इसमें Y की FD. में LHS का closure लेंगे लेकिन X वाले में से।).

so,

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

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

⇒ all 3 covers in their 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.

$X \not\subseteq Y$ $\qquad$ $Y \not\subseteq X$

$\boxed{X \equiv Y}$ 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 :→

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

$AB^+ = ABCD$, $\qquad$ $C^+ = CD$ → 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.

∴ $\boxed{X \not\equiv Y}$ ok.

& $\boxed{X \supseteq Y}$ ✓, True & $\boxed{Y \supseteq X}$ ✗, false ✓.

---

35. Dependency Preserving Decomposi^n :→

Def^n :→ Let, any table $\boxed{R(ABCD)}$
& also some F.D., $\boxed{FD\{A \to B, B \to C\}}$

Now,
we find closure & then find c.k., & then we also find hidden dependencies :→
like.

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

and,

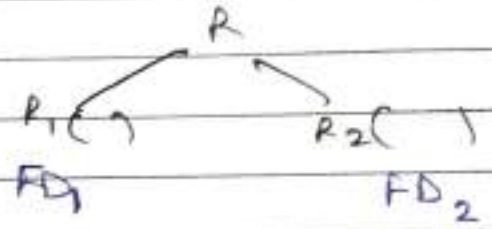
→ In normalisa^n, we do decompos^n, then, we divide

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

→ We distribute attr. of R in R_1 & R_2

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

Now, F.D. also divides, like



Now,

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

i.e. $\boxed{FD_1 \cup FD_2 = FD^+}$ (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 → A~~	B → C ✓	B → D ✓
~~B → B~~	C → B ✓	D → B ✓
A → B ✓		
B → A. ✗		

B^+ = BCD
(don't have A).

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

Now

→ Check Dependency preservation :→

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

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

We don't want Trivial F.D. :→

→ Trivial F.D.: in which intersec^n ($\cap$) 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. वो होती ही हैं like

$\boxed{A \to A}$ is definitely true.

So,

→ Only take non trivial F.D.: →

($\cap = \phi$)

→ Now:

F.D :→ { A→B, B→C, C→D, D→B }.

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

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

Now
F.D. which we get from Table :→

A→B ✓
B→C ✓
C→B
B→D
D→B. ✓

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

So, F.D : { A→B, B→C, C→D, D→B }.

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

C^+ = CBD ∴ C→D also preserved.

All F.D. are preserved, True Ans.

(36.) Dependency Preserving Decompos^n Example

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

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

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

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

Now,

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

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

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

$BC^+ = BC$,

$BD^+ = BDAC$

$CD^+ = CDA$.

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't get

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

Hence,

$\boxed{\text{Dependency preserved not}}$.

37 Joins & Its Types :→

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

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

'Employee'

Dep.No	Name	E.no
D_1	HR	1
D_2	IT	2
D_3	MRKT	4
D_4	Finance	5

'Department'.

↓

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

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

Now :-

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

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

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

Its Types :→

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

---

(38) "Natural Join" ! →

→ Join = Cross product + Condi^n.

(m x n)

m

E.No	E name	Add.
1	Ram	Delhi
2	Varun	chd.
3	Ravi	Chd.
4	Amrit	Delhi

Employee

n

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

Department

→ first find :- Table name

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

(Emp, Dept Mean. Cross product)

- Here, we need both, - Table Employee
  - Department Table.

We do it by "Natural Join".

- Common Att - 'E_No'.

So,

Whenever we have to equalise (ढालना) 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	Ram	D_1	1
3	"	D_2	2
3	"	D_3	4
4	Amrit	D_1	1
4	"	D_2	2
4	"	D_3	4

So, finally we get 3 rows.

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

{ Ram, Varun, Amit } → Ans.

Now, how to write Actual Query :→

→ Select E_name from Emp Natural Join Dept;

---

39. Self Join :→

→ in which the Table is joined to itself.

S-id

Reference Cid Table.

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

student course

Study

Q: find student id who is enrolled in not least 2 courses

→ Now, we do Self Join.

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

Query :→

Select (from study as T_1, study as T_2;)

';' → means Cross product.

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

S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

T_2 (m)

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

(n) T_1

$m \times n$

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

Condi^n :→

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

< > → different (not equal to).

Now, Complete Query :-

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

So, final output Table :-

S_1	C_1	S_1	C_2
S_1	C_2	S_1	C_1

→ Now, we want S-id from this Table.

But T_1 & T_2 both have S-id.

So,

we can use any $\pi_{T_1.Sid}$ or $T_2.Sid$

[Ans - S_1]

---

(40) EQUI - Join :- → (=).

(कोई भी attr के बीच (=) लगा सकते हैं)

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

Employee.

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

Department.
Dept.

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

Q: Find the Emp name who worked in a department having locaⁿ 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.Locaⁿ ;
```

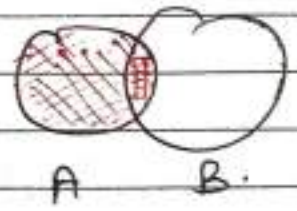
Emp.			Dept.		
Delhi	1	Ram	D_1	Delhi	1
"	1	"	D_2	pune	2
"	1	"	D_3	patna	4
Chd	2	Varun	D_1	Delhi	1
"	2	"	D_2	pune	2
"	2	"	D_3	patna	4
"	3	Rani	D_1	Delhi	1
"	3	"	D_2	Pune	2
"	3	"	D_3	patna	4
Delhi	4	Amrit	D_1	Delhi	1
"	4	"	D_2	pune	2
"	4	"	D_3	patna	4

→ Ram

Note: दो Table के common att. के बीच में हमेशा (=) लगाना है & किसी भी 2 att के बीच हम (=) लगा सकते हैं।

(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	Deptno.
E_1	Varun	D_1
E_2	Amrit	D_2
E_3	Rani	D_1
E_4	Nitin	—

Dept.

Deptno.	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

→ Condi'n of Natural Join. (Common att. को 1 & 2 को दिया।)

Relational Database always shows output in Table form.

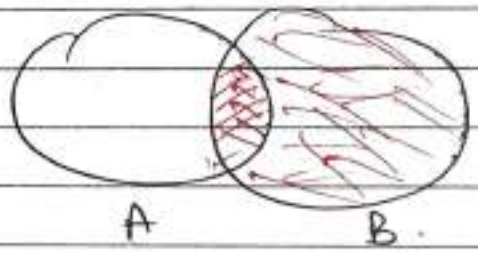
output Table :-

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 सारी Row आती ही है)

(92) Right Outer Join :→

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



(Emp)

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

(Dept)

Dept-No	D-name	Loc^n
D_1	IT	Delhi
D_2	HR	Hyd
D_3	Finance	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	Amit	HR	Hyd.
E_3	Rani	Finance	Pune.
—	—	Testing	Noida.

Right Table all Rows.

Full-Outer Join :- Take the union of Left outer Join & Right outer Join Rows.

((Left O.J.) ∪ (Right O.J.))

(43.) Relational Algebra :→ (1970).
(procedural lang. or Formal Query lang.) also.

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

⇒ SQL language base is Rela^al Algebra.
('collec^n of mathematical Expres^n').

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

"operators"

Basic operator	Derived operator.
→ projection (π)	→ Join ($\bowtie$)
→ Selection (σ)	→ intersect ($\cap$)
→ Cross product ($\times$)	$(x \cap y) = x - (x - y)$.
→ Union ($\cup$)	
→ Rename (ρ)	→ Division ($/$, $\div$)
→ Set Difference ($-$)	

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

(44.) Projection in Rela^n^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)

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

Roll no.
1
2
3

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

We fetch the 2 colm (roll no. & name) from the student (Table).

Output →

Roll no	Name
1	A
2	B
3	A

Note: It only gives unique Answer.

Q2: # Query:- π_{name} (Student).

Name
A
B

→ (only name here).

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

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

σ (Sigma).

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

→ It works on Tuples (rows).

→ First, it found rows.

π operator.

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

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

↓

that.

⇒ [π_{name} ($\sigma_{rollno='2'}$ (Student))] → (B)

π always last में operate होगा।

* We can also find, 2 at a time.

Ex:-

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

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

(46) Cross / Cartesian product in Rela'al algebra

⇒

A	B	C
1	2	3
2	1	4

(R1)

C	D	E
3	4	5
2	1	2

(R2)

To join, we must have to cross product.

→ $3 + 3 = 6$ $(m+n)$.

$2 \times 2 = 4$ $(x \times y)$.

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:

$\{1, 2, 3\}$ $\{3, 4\}$

S_1 S_2

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

→ It is not Commutative :→

i.e.

$A - B \neq B - A$

Condi^n :-

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

Ex :-

Rollno	Name
1	A
2	B
3	C

(Student)

Emp-No.	Name
7	E
1	A.

(Employee)

⇒ (Student) - (Employee) [: A हटा गया]

⇒

Rollno	Name
2	B
3	C.

⇒ (By default, पहले Table के ही 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 Operⁿ in R.A. :→

→ Same as in Set. (∪).

(1,2,3) S_1 (3,4,5) S_2.

↓

(1,2,3,4,5)

Ex:→

Roll no	Name
1	A
2	B
3	C

(Student)

Emp. no.	Name
7	E
1	A.

(Employee)

Condⁿ :→

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

(Student) ∪ (Employee)

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

→ (left side Table ⁿ और ही).

Easy Trick! -

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

Name.
A
B
C
E.

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

Note: 1)

Roll no	name
1	A
2	B
3	C

Numeric

Name	Emp No.
E	2
A	1.

alphabet

not Same domain.

(ye, By Default order pick होती!).

so, Here, not Union operⁿ applies. (NULL).

(49.) Division Operaⁿ in R. A. :→

⇒ Divisⁿ operator is actually a Derived operator.

(ele derived them, from normal operators).

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

→ ele use these

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

Date :
Page :

⇒ every (या) all देखते ही Division 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'

Notation of Division Method :→

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

Here,

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

ie,
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).

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

S_1 − S_2
~~S_2~~ S_3
~~S_3~~

$\Rightarrow S_1$ ok.

(So-) Tuple Calculus in DBMS :→

Relational Calculus (RC)
- TRC (Tuple or rows)
- DRC (Domain or column) Att.

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

Date :
Page:

⇒ In Tuple Calculus, a query is expressed as

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

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

* Operations :→

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

→ Atomic func^ns :- (We use one variable here. Ex - (Supply Ids. from a supply table) → only 1 condi^n.

→ It also uses quantifiers :

- $\exists t \in r (Q(t))$ = "there exists" a tuple in t in rela^n r such that predicate Q(t) is true.

- $\forall t \in r (Q(t))$ = Q(t) is true "for all" tuples in rela^n 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 में चला जाएगा।).

↳ unsafe Expresⁿ.

(These are not in the Relⁿal algebra).

- Both TRC & RA have same expressive powers.

unsafe exp^n → $\{ U.E(x) \}$.
नहीं होता
चाहिए।

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

But, SQL has more powers than these.
SQL has more xtra funⁿ.

* 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')

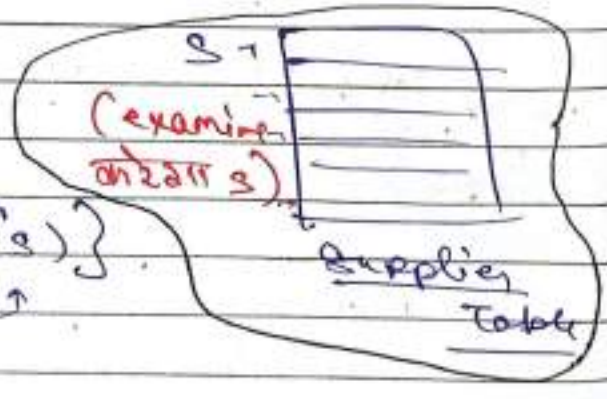
Q.3 Write a query in SQL to display SID of suppliers whose name is 'Varun' & address is 'chandigarh'.

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

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

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

(1)



{ S.Sname | Suppliers (S) }

(2)

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

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

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

{ C.SID | Catalog (C) }

Sid तो Catalog Table में भी मिल जाता।
But, Sname only in Supplier Table.

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

&

final ans. from supplier Table.
(Tuple Exists).

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

Here,

S → free variable
c → bounded variable

(जिसके साथ हम ∃ या ∀ लगाते।)

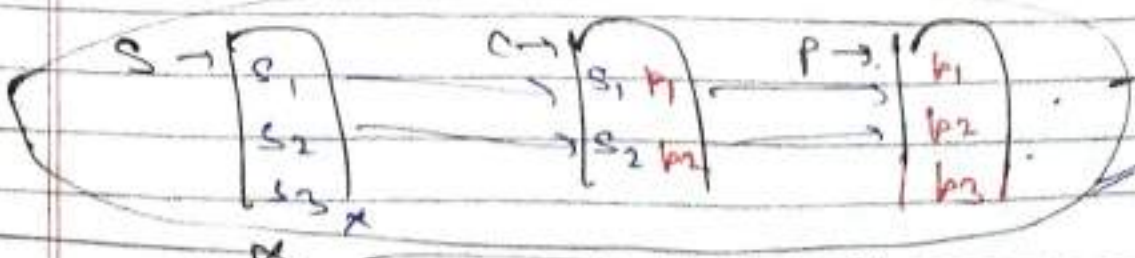
(6.) Here, we need 3 Table — Supplier, Catalog, parts.

But,
ans from Supplier Table.

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

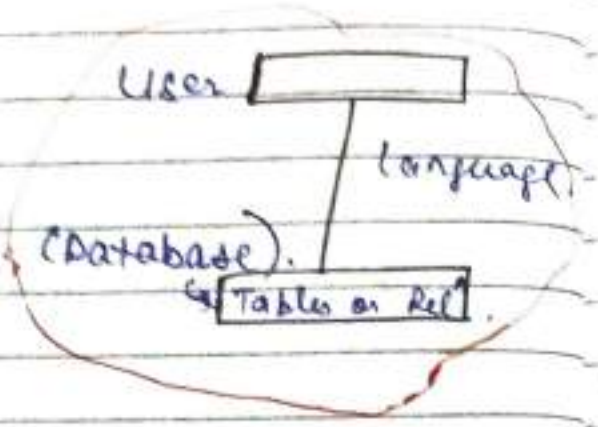
S variable for Supplier Table.
c — " — Catalog —.
p — " — parts —.

S → free variable
c, p → bounded "



Intro to SQL :→ (Structured Query language).

(1970)



→ EF codd gave theory of store & fetching of data.
Relational Model:

By the help of, Relational algebra & TRC (Tuple Relational calculus)

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

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

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

→ Now, No SQL is come into market.

→ 10% data → Structure
90% data → Unstructure (spark, mogodb)

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 → समूह, family, ग्रुप, ग्रुपिंग करना.

Exist / Not Exist.

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

→ We use these operators intergally to use Query.

Ex:- IRCTC :- We query on this app, by APIs (Applicⁿ 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, procedure)
same as in Java.

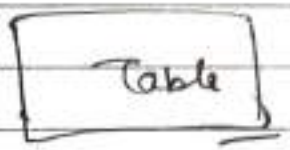
(H) What to do :-
In libraries of oracle as like SQL Series, it is already defined that what select do, & what other commands 'do', so, we don't need to give procedure that how to do.

52 All Types of SQL Commands :-

* DDL Commands :- (Data Definition language).

→ Deals with Schema (structure | Table).

- → Create
- → Alter (change Table specifica^n).
- → Drop. (Remove Table)
- → Truncate. (Remove Data).
- → Rename.



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

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

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

* DCL Commands :- (Data Control language)

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

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

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

→ Mainly used in Transac^ns.

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 if it starts from '0' again as from 90%). This is save point.

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

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

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

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

S3- Create Table in SQL with execution :→

like SQL

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

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

→
```
Create table <table-name>
(
 Column1 name datatype ,
 Column 2 name datatype ,
 Column 3 name datatype
);
desc table-name ;
```
// no space
Ex: abc_xyz.

// describe.

Ex:
```
Create table emp
(
id int,
name varchar2 (20),
Salary number (10)
);
desc emp;
```
→ fixed type of datatype (2 digit fixed)

→ variable — // — (can be increased)

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 है

→ Add or Remove colmns/s.
→ Modify datatype.
→ Modify datatype length.
→ Add Constraints
→ Remove Constraints
→ Rename colmn / table.

Id	name	Address

Student.

Alter table student Add address varchar(30);

(,) ;

* multiple colmns.

⇒ 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 seg_no ;
"  "  rename to emp1 ;
alter table emp1 add primary key (seg_no);
```

(55) DB Alter & Update : →

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

Ex!

Emp

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

} → can only change in this struc.

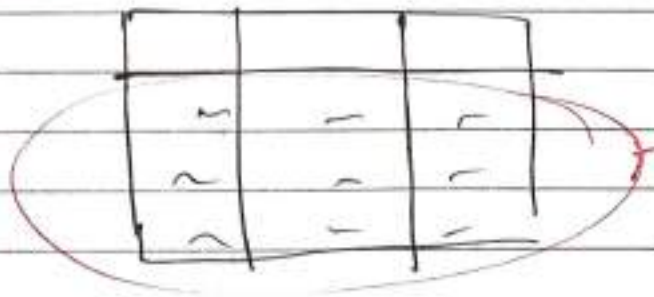
→ Id → varchar

→ name varchar (10) → 20

→ ID → Eid. (colm name change)

→ Emp → Emp_detail (table name -)

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



→ can only change in data.

```
update Emp
Set Salary = Salary * 2;
```

→ for all students

```
update Emp
Set Salary = Salary * 2
Where id = 1;
```

→ for only student of id = 1.

S6 D/B Delete, Drop & Truncate :-

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 condi^n for delete some particular rows.

Ex: Delete from Student
Where id = 1;

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

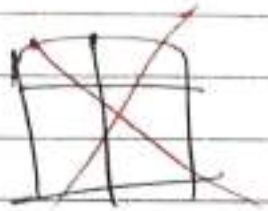
> log :- Emp file Completely delete होने से पहले, Computer एक temp. file बनाता है, to restore data, इसे log कहते हैं।

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) restric^n.

Ex: In gmail:→ (2 account Ids can't be same)
anurag@gmail.com ✗ (not present)
anurag13@gmail.com ✓ (present). ✓

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

Ex → x (-10)

check (age>18) → 10 x
→ 20 ✓

i.e. (एक particular Domain में से ही User Id बनेगी)

Domain - fixed.

5.). Foreign key :-

6.). Default :-

[Salary int default 10k].

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

→ We can use multiple constraint in a colm also.

58. SQL Queries & Sub-Queries :→

(from Beginning to end).

Q:-

Emp.

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

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

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

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

<> → not equal to, = !, ≠

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

Output → 50000

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

→ Select E_name from Emp where Salary = (Select max (Salary) from Emp);

Outer Query. Inner Query

Output → Karan

⇒ Inner Query execute first 1 time

पहले

Ex-

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

(59) (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

max ← 10k - (50k)

20k

30k

40k

50k ✗ <

↳ Inner

40k Ans

(iii) Query:-

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

Sal.

[40k] ✓

(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 40k)

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] clause.

(terms up of dept.).

HR	
HR	2
MRKT	
MRKT	2
IT.	1

Note:-1 Condⁿ of group by → जो Att. 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 att. in Select other than the att. of group by func, then we have to use aggregate funcⁿ.

- 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;
aggregate func.

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

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

HR 2
MRKT 2
IT 1

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

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

Output → IT.

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

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

In
HR IT
MRKT
IT

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

Note: A SQL query always starts from 'from'

Select ___ from (table name)

Note: Select max (y) from ___ group by x;

Here, It is Right, bcz
We use aggregate func (max) with y
i.e. max(y) & if we don't use max,
then we can only use x, bcz group by x.
(Same).

Query:-

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

Output → Rani, Nitin, Varun.

Steps →

			Outer	Inner
HR	HR - 30,000	30k	10k ✗	30k
HR			20k ✗	40k
MRKT	MRKT - 40,000	40k	30k ✓	50k
MRKT	IT - 50,000	50k	40k ✓	
IT			50k ✓	

* means All attributes.

(63.) Use of IN and Not IN in Query.

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

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

Emp

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

Project

E_id	P_id	P_name	Loca^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 :- (Mean not included)

Query :-

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

Output :-

4	Robin	Bangalore

Chd ×
Delhi ×
Pune ×
Bangalore ✓ ok
Chd. ×

---

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

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

⇒ Suggestion :- First make Dummy Tables.

⇒ Here, we need both 2 Tables.

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

⇒ Nested → Bottom up → नीचे से ऊपर

Means पहले अंदर 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.

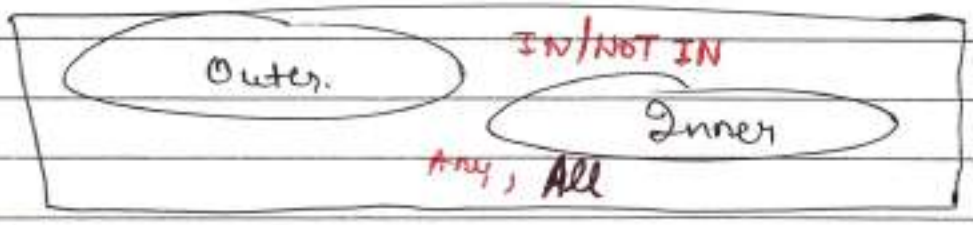
Output :- Rani
Nitin
Robin
Amrry.

(1, 2, 3, 4, 5) (1, 5, 3, 4)

---

65. Exist & NOT Exist Subqueries :-

→ We use these in Corelated Nested Query.



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

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

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

Null → means value is not available.
Empty

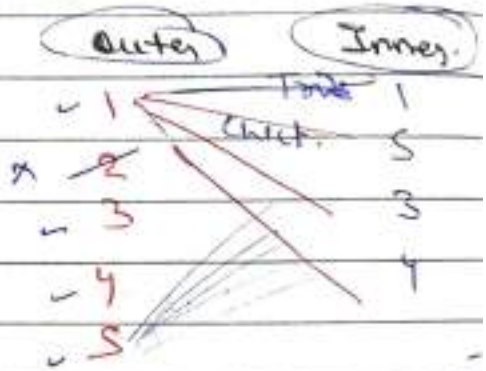
Q Query :→

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

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

Output:-

1	Ram	Chd
3	Nitin	pune
4	Robin	Bang.
5	Ammy	Chd.



66. Aggregate Funcs in SQL :→

Max, Min, Count, Avg, Sum.

(Total no. of values in a Table).

Emp

E_id	E_name	Dept	Salary
1	Ram	HR	10k
2	Amrit	MRKT	20k
3	Ram	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.

→ Same for Min.

→ Select Min (Salary) from Emp;

output → 10k.

* COUNT! -

Count (*) means no' of rows in the Table.

→ Select Count (*) from Emp;

Output → 6 . 6 rows.

→ Select Count (Salary) from Emp;

Output → 5 (bcz one value is Null)

* Distinct! -

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

Output → 4. (bcz 30k is 2 times).

* SUM! →

Select Sum (Salary) from Emp;

Output → 140 k. (Sum of all salary)

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

Output → 110 k (30k repeats).

AVG :→

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

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

$$= 28k$$

→ Select Avg (Salary) from Emp;

Output :→ 28k.

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

Output :→ 27,500.

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

$$= \frac{110k}{4}$$

$$= 27,500$$

---

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 is compare with every row of inner Table.

→ Returns True / False.

Emp

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

Dept

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

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

→ True 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 2 3 5
2 1 2 3 5
3 1 2 3 5
4 1 2 3 5
False

68 DIB Joins, Nested Subquery & Correlated Subquery :-

→ Nested → Bottom up. (अंदर से बाहर)
→ Correlated → Top down approach (बाहर से अंदर)
→ Joins → Cross product + Condi^n

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	MRKT	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	Inner
1 = 1 → True	1 = 5
2 = 1	2 = 5
3 = 1	3 = 5
	False

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

then, [Total Comparison → m × n]

→ JOINS :→
Cross product + Condi^n.

It creates a new Table with (m×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×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. Δ But, Joins take more space. (bcz big Table of (m×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 Correlated Subquery ?
जहाँ पर outer Query का कोई ना कोई value हमने Inner use किया है।

How to find N^{th} Highest Salary :→

Query :-

Best method → learn it.

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

∴ Co-related Subquery

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

Ex: on 1st case :→

E_1

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

E_2

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

⇒ why we make 2 (E_1 & E_2) :→
bcz employee we use 2 times. (E_1 & E_2)

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

↳ It means, $\boxed{\text{employee salary > employee salary}}$
↳ no mean, ✗.

& E_1 & E_2.

Ex:

	10k	> 20k	f
	20k	> 20	f
2nd row	20k	> 20	f
	30k	> 20	count = 1
	40k	> 20	count = 2
	50k.	> 20.	count = 3

count = 3.

Ex:→ lets 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

i.e.

Output:→

ID	Salary
2	20k

&

20k is our 4th highest Salary.

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

Ex:- for 3rd highest Salary:-
N = 3, N-1 = 2

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

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

4th Row is our output.

Output: 4, 30k.

&

30k is our 3rd highest Salary.

3 Imp. Questions on SQL basic concepts :

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

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

b) — " — last name = '*A%' ×

c) — " — name like '%A%'

d) — " — like '*A%'

Note:- Questions like, 2nd letter same, Salary be of only 5 nos, like that, based on 'Like' command.

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

Ex.) i) %A% anything AABA ठीक नहीं है

ii) 2nd letter → '_A%' (one pos^n is fixed)

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

iv) 2nd last letter must be A → '%A_'

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

A) Delete table < table name >.
B) Drop table < ——— > 3. Ans
C) Erase table < ——— >
D) Alter table < ——— >.

→ DDL commands deal with Scheme (Table structure).
Create, drop, Alter → DDL Commands.

→ Alter table → to change anything in table.

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

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

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

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

Ex:

T

1
2
3
3

Q1 | Q3 | Q2

Q1	Q3	Q2
1	1	1
2	2	2
3	3	3
3	3	3

Q2 ∩ Q3 → 1, 2, 3

Q2 & Q3

[Dummy Table बनाके Data लो]

Ies type की टेबल बनानी है (with same data)

71. PL - SQL :→

(procedural - SQL).

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

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

(programming flaws सही किया).

PL - SQL

- Function
- * procedure
- * Triggers
- Cursor

→ 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;
```

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

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

in C++

(CPU always performs in RAM.) CPU → fast.
(CPU never works on Hard Disk) Hard Disk → slow

(77.) TRANSACTION CONCURRENCY :→

Transac^n :- It is a set of operations used to perform a logical unit of work.

(कुछ भी हम change करे रहे, Database के अंदर, कुछ भी हम Database को Read करे रहे, that is also a part of transac^n.)

Ex:-
→ When we withdraw our money from ATM, then we have to perform a ~~task~~ set of opera^n, these set of opera^n'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.]

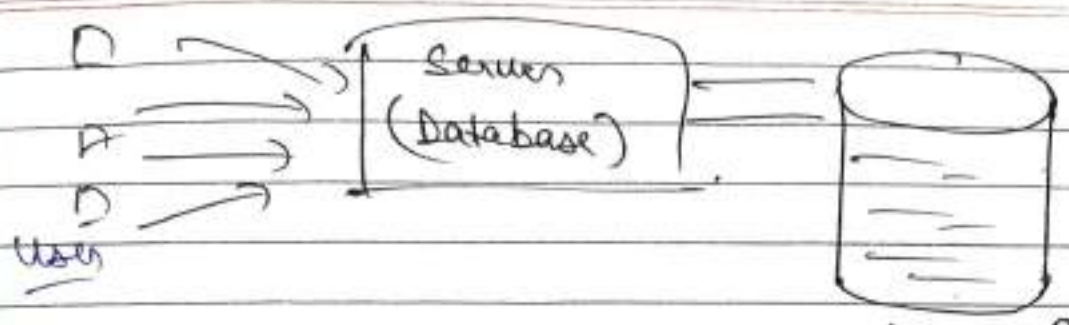
→ When we read or access any data from HDD (hard drives), then it comes to RAM (where we perform opera^n's).

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

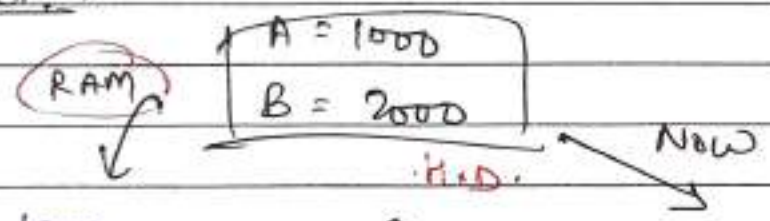
In C++ for assignment, =
In PL SQL , :=

C++, =
PLSQL, =

(137)



Transfer :-



R(A) → 1000
A = A - 500
W(A) → 500
R(B) → 2000
B = B + 500
W(B) → 2500
Commit

A = 500
B = 2500.
Saved in H.D.

---

(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^n)

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

Roll back

Commit

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

या तो सभी operⁿ execute होंगे, commit तक।
वरना शुरू से Roll back होगा।

Note:- A failed Transaction cannot be Resumed.
A failed Transacⁿ 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 ——→ B
1000

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 Transacⁿs.

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

CPU speed is in MIPS (Million instruc^n per second).
& Hard Disk → 10/20 instruc^n per second.
∴ CPU never compat'ble 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).

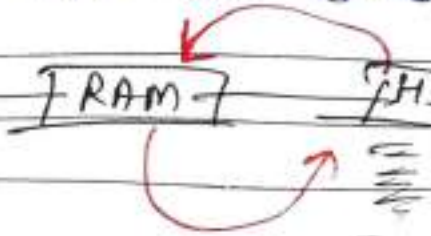
⇒ That's why, we save data in HDD for Durability.

---

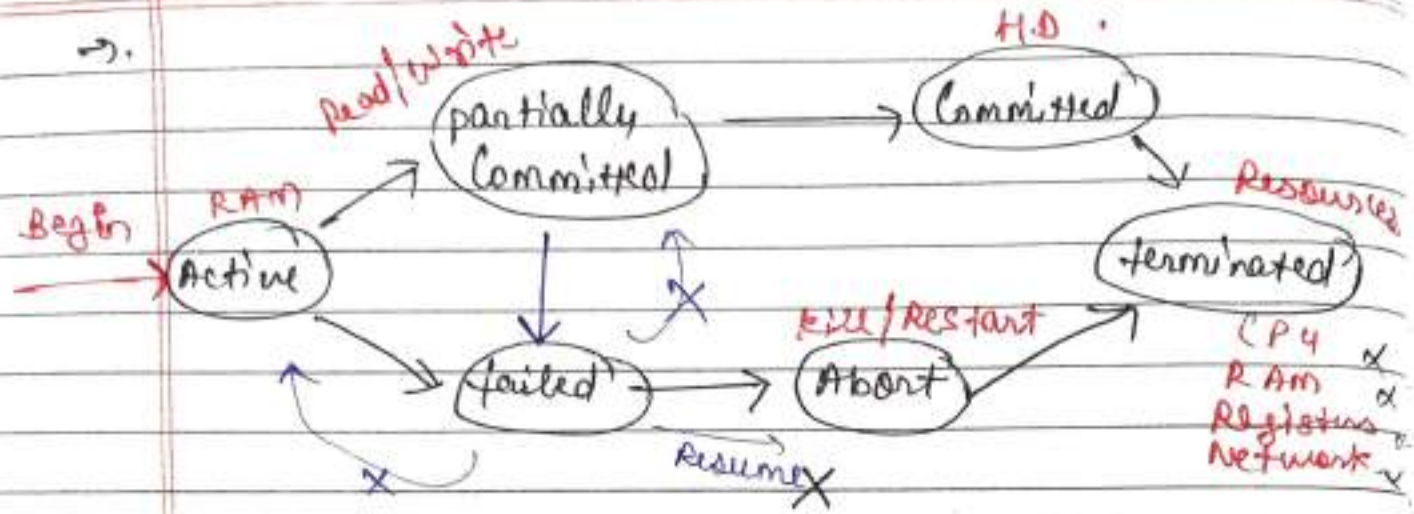
74. Transaction States :→

⇒ At now, Transac^n 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 opera's are stored in ram/local memory/shared memory until now.

→ Committed! -
Changes are now saved to Hard Disk.

→ terminated! →
Here, we deallocate our Resources
ie. free all Resources. bcz,
(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^n & Restart the opera^n.

---

(75) SCHEDULE! → (Serial vs parallel Schedule).

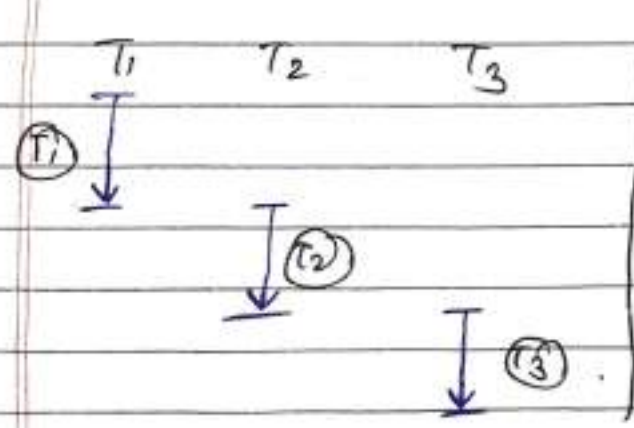
→ Schedule! → It is chronological execution sequence of multiple transactions.

T_1 T_2 T_3 - - - T_n

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

Serial Schedule! → Until a transac^n gets completed, no other trans^n can interfere.

All transa^n's executed by a serial sequence.



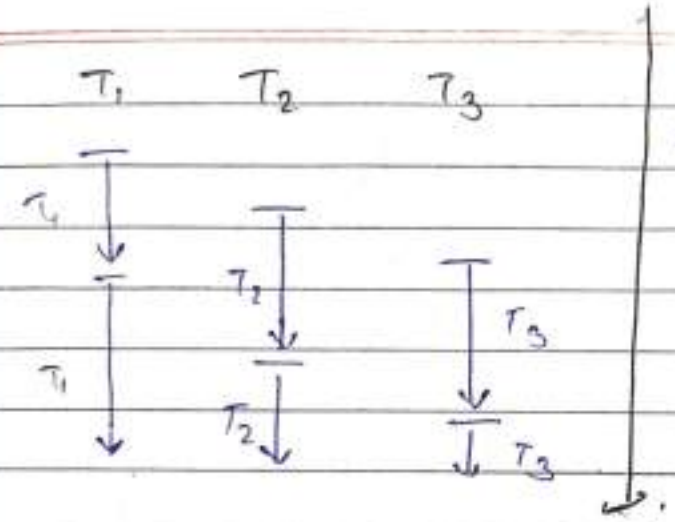
Advantage →
→ Consistent (bcz, no one can interfere).

Disadvantage →
→ (When T_1 gets executed, T_2 & T_3 are in waiting.)
→ Performance degrade.

Parallel Schedule! →
→ We can switch to multiple transac^n's at a time. i.e.
→ Multiple transac^n can execute at a same time.

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

Throughput → No. of transac^ns executed / time.



Throughput ∝ performance.

* Advantage :→
→ performance Increased. i.e, (Throughput is high.).

* Disadvantage :-
→ problem may occur. & (Inconsistent)

(*) Nowadays,
(Parallel Schedule is more performed.)
↳ bcz good performance in less time.

—x— —x—

(7b) Types of problems in Concurrency :→

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

→ (mean, No problem is in Serial Schedule. There are problems 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).

T_1	T_2
100 → R(A)	
50 ← W(A)	
	(50) R(A)
	W(A)
	Commit.
Roll back. Failure	

// Dirty Read.

Hence, when T_1 gets failed. Then, how T_2 can use the A – 50. So, Dirty Read.

2.) Incorrect Summary problem ! → (T_1)
It occurs mostly when a transac^n start & T_2 comes & start performing its aggregate (sum, average) funcn^s.
then,
we get incorrect value of sum, average etc.

3.) Lost update! →

⇒ जो पहले (T_1) ने changes किये। उन changes को (T_2) ने और update कर दिया। & पुराने वाले changes हमारे Lost हो गए। i.e, [update किया हुआ Lost हो गया।]

Ex! →

(id-1) T_1	T_2 (id-1)
5 Likes	
	5 Likes

⇒ & we finally get 5 Likes on the same product instead of 10 Likes.

Phantom – glimpse, shadows.

4.) Unrepeatable Read :→

T_1	T_2
R(x) – 100	
	R(x) – 100
50 – W(x)	
	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
10 ← R(x)	
	R(x) → 10
Delete (x) (0) ×	→ Read (x). (It also tries to Delete '0')

---

(7) Read-Write (Problem) or Conflict (OR) Unrepeatable Read Problem :→

4 cases are there →

Same Data.

R (A)	R (A)	
R (A)	W (A)	Problem.
W (A)	R (A)	
W (A)	W (A)	

Ex-(1)

User 1 T_1	User 2 T_2
2 R(A)	
	R(A) 2
	A = A-2
	W(A) 0
	Commit.
0 R(A)	
W(A)	
Commit.	

$\boxed{A = 2}$ 0.

problem occurs due to this.

(*) $\boxed{\text{Ex- IRCTC}}$

⇒ Let, user 2 reserve both the seats. Then,

$A = A - 2 \quad = 0$

→ & user 1 remained in thinking to reserve or not. & when he sees again.

Now,

$\boxed{\text{Seats becomes '0'}}$

So,

Now, user 1 have to be Roll back. (Abort)

Ex-(2)

T_1	T_2
10 R(A)	
9 A = A-1	
	R(A) 10
	A = A-1
	W(A) 9
	Commit
9 W(A).	
Commit	

$\boxed{A = \cancel{10}\ \cancel{9}\ 9}$

⇒ We issued 2 books, but value still is 9. (instead of 8)

(78.) Irrecoverable Vs Recoverable Schedules in Transac's :→

(3) Schedule

T_1	T_2
10 R(A)	
5 A = A - 5	
5 W(A)	
	R(A) 5
	A = A - 2
	W(A) 3
	Commit.
20 R(B).	
fail.	

A = ~~10~~ 5 B = 20 — 10.

Roll back.

(due to any reason, let, T_1 fails)

↳ Now, (T_1 Rolls back due to Atomicity property).

(Either all or None)

∴ On Roll back, everything happens in T_1 is gone. So,

Again Now, [A is 10]

∴ but, here T_2 also done something & that is lost Now.

⇒ T_2 Change is lost. we can't recover it now. ⇒ It is Irrecoverable Schedule.

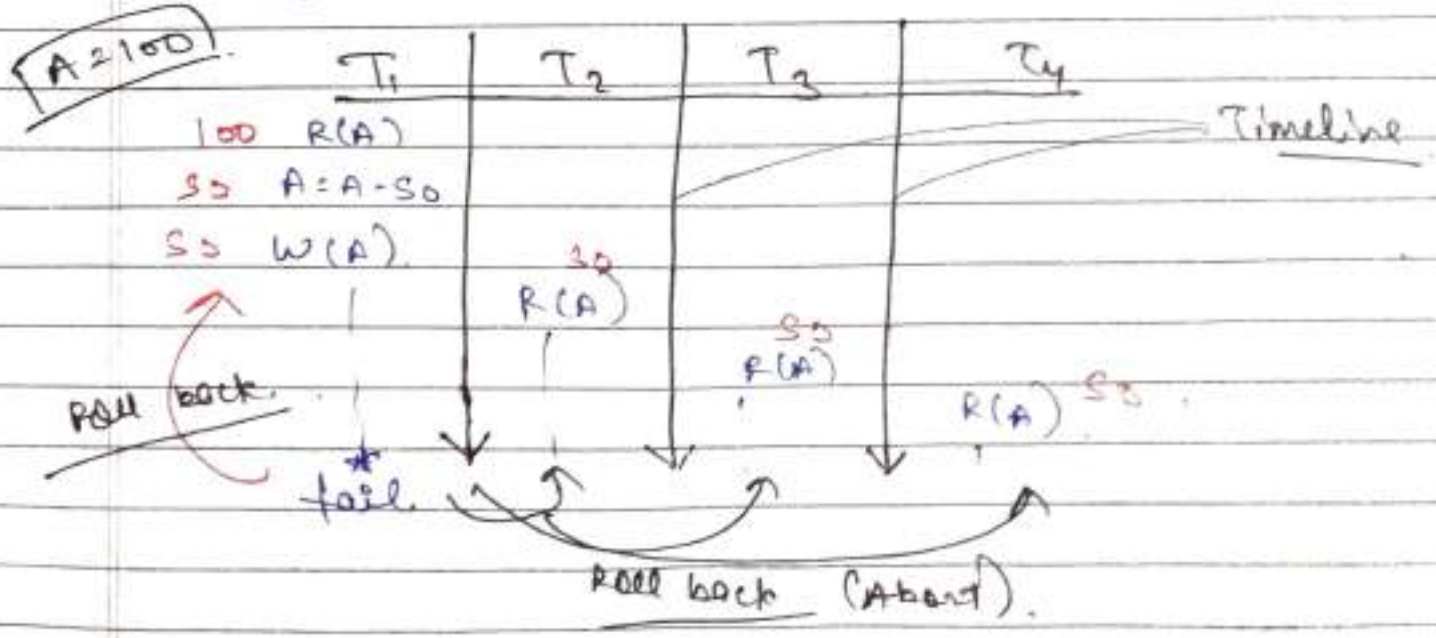
(Cascade → झरना)

(79.) Cascading (Vs) Cascadeless Schedule :-

→ In Recoverability, these 5 are Important :-

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

⇒ Cascading :- means Due to occurence of one event, multiple events are automatically occuring.



→ Here, 1st 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 utilisaⁿ 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?

→ Solⁿ:- T_2 & T_3 can't Read the (A) value from T_1 until A value gets Committed or roll back in T_1.
& (Commit → होने के बाद तो कोई दिक्कत ही नहीं है।)

A=100 50

T_1	T_2	T_3
100 R(A)		
50 W(A)		
Commit	R(A) ✗	
		R(A) ✗

→ i.e, Don't allow Read in T_2 & T_3.
So,
It automatically becomes Cascadeless.

Ex,

R(A) W(A) R(A)	R(A) W(A) Commit R(A)
Cascading	Cascadeless

(write के बाद Read नहीं कर सकता)

→ But, there is still W-W (Write-Write) problem in Cascadeless, bcz

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

A = 100

T_1	T_2
100 R(A)	100
90 A = 10	R(A)
90 W(A)	A = A - 20 . 80
	W(A) 80
Roll back	Commit
fails	

Lost

→ Now, when T_1 fails. (A becomes 100 again)
&
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 होगी ही नहीं। ∴ & finally A = 100

⇒ Strict Recoverable tells that we also can't write along with Read.

(80) SERIALIZABILITY :→

→ (Serializability means that a schedule has ability to become a serializable.)

Mean,

Schedule - Collec^n of transac^n. ($T_1, T_2, \ldots T_n$)

* S

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)

$T_1 \rightarrow T_2$

S

T_1	T_2
	R(A)
	W(A)
R(A)	
W(A)	

$T_1 \leftarrow T_2$.

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

तो, Serial Schedule तो ये already है, इसको कुछ नहीं कर सकते।)

→ We want, Parallel Schedule →

*

T_1	T_2
R(A)	
	R(A)
	W(A)
W(A).	

$T_1 \rightleftarrows T_2$.

→ Convert it to Serial Schedule →

2 ways :-

(1) $T_1 \rightarrow T_2$

(OR)

(2) $T_2 \rightarrow T_1$

→ To check Serializable ? Check that if there Exists an Serial Schedule Equivalent to Parallel 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
(clone of ∥ schedule)

(#) Let, a Schedule has 3 transac^ns.

S

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

parallel Schedule →

⇒ Serial Schedule

[6 ways]

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

⇒ If we able to convert that parallel schedule in any of the 6 forms of serial schedule, then we can say that it is a Serializable.

⇒ Why we got 6 forms! →

because we have 3 values (T_1, T_2 & T_3).

So,

$3! = \boxed{6}$ ways.

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)	

Here, A & B are 2 diff. op, rep.

How to Convert :→

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

Ex:

T_1	T_2
	R(A)
R(B)	

⇒

T_1	T_2
R(B)	
	R(A)

Q: To check Conflict Equivalent :→

$\boxed{S \equiv S'}$ → 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^n:- So, In S, we have adjacent non-conflict pair, so swap them.

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(A)
R(B)	
	W(A)

Again.

S

T_1	T_2
R(A)	
W(A)	
R(B)	
	R(A)
	W(A)

Now Serial Schedule ←

= S'

Hence

$\boxed{S \equiv S'}$

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.