

C. Isolation \Rightarrow

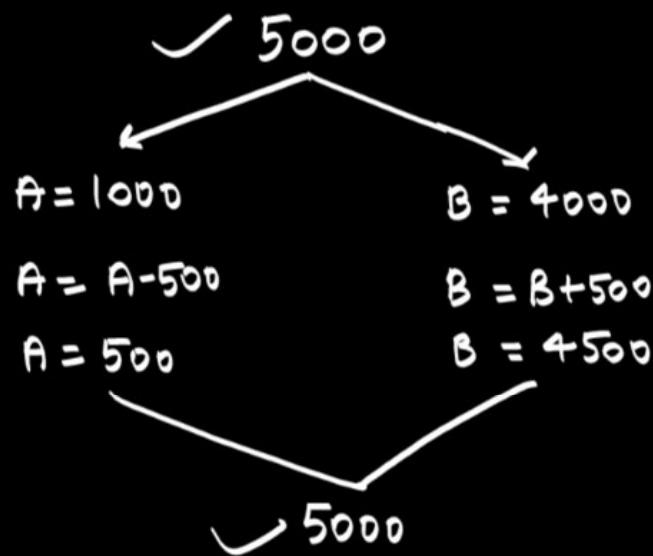
→ Parallel Transactions को एक sequence में Convert करके Perform करना चाहिए,
जिससे Data में inconsistency नहीं हो।

D. Durability \Rightarrow

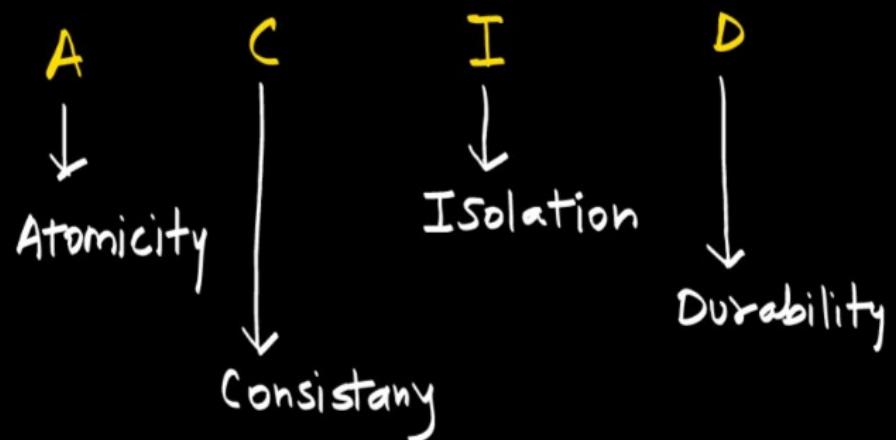
→ DataBase में होने काले सोई changes Permanent होने चाहिए ताकि उसके बाद कोई भी
Hardware/ Software Failure की condition में Data Loss नहीं हो।

B. Consistency \Rightarrow

→ Transaction Start होने से पहले वह End होने के बाद Amount का sum same होना चाहिए।



ACID Property ⇒



A. Atomicity ⇒

- Either All or None
- कोई भी Transaction या तो पूरा Execute होगा या उसका कोई भी Part Execute नहीं होगा।
- जैसे - $T_1 \{ S_Id, S_Name, S_Class \}$

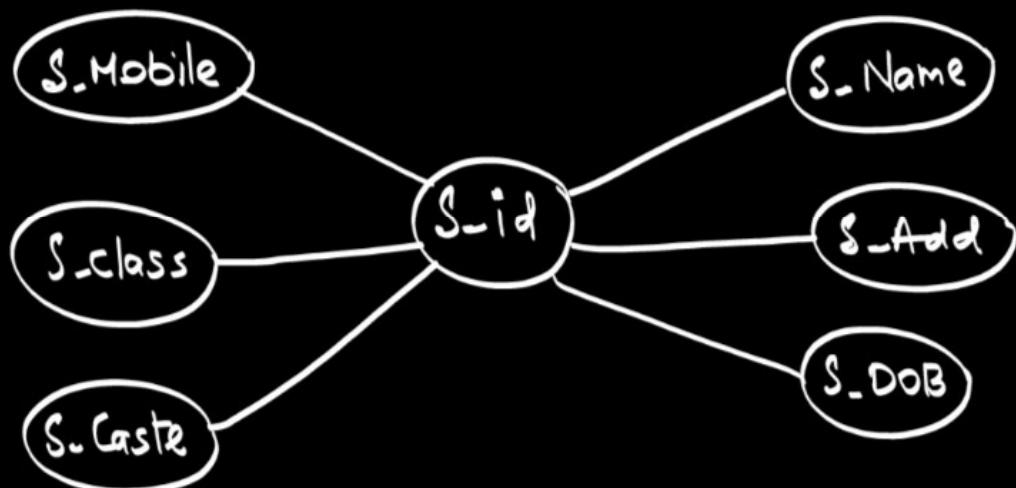
Differences →

Feature	Document Based	Key-Value Base	Column Based	Graph Based
1. Data Model	JSON	Key-Value Pairs	Columns instead of Rows	Nodes & Edges
2. Best Use	Semi Structured Data	Fast Lookup & Caching	Analytics & Big Data	Relationship Heavy Data
3. Query Performance	Moderate	Fast	High for Analytics	Optimized for Relationships
4. Schema	Flexible	Dynamic	Semi Structured	Schema Less
5. Scalability	Horizontal	High Horizontal	Highly Scalable	Scales with Relationship

→ इस प्रकार के DataBase का उपयोग Netflix, Instagram & Facebook के द्वारा किया जाता है, क्षेत्रिक इनमें उजारों users का Data प्रति second process करता रहता है।

D. Graph Based NOSQL →

→ Data को Nodes & Edges के Pattern में store किया जाता है।



उदाहरण - Amazon Neptune,
ArangoDB, Neo4J.

C. Wide Column Based NOSQL =

- यह Database RDBMS जैसा ही दोता है, लेकिन इसमें Data को Row की बजाए Columns में store करते हैं।
- इसमें Column, Dynamic दोता है, इसलिए इसकी size अधिक हो सकती है।
- इसमें Data को Multiple Columns में store किया जाता है।

101	102	103	Ram	Mohan	Hari	10	12	11
-----	-----	-----	-----	-------	------	----	----	----

या

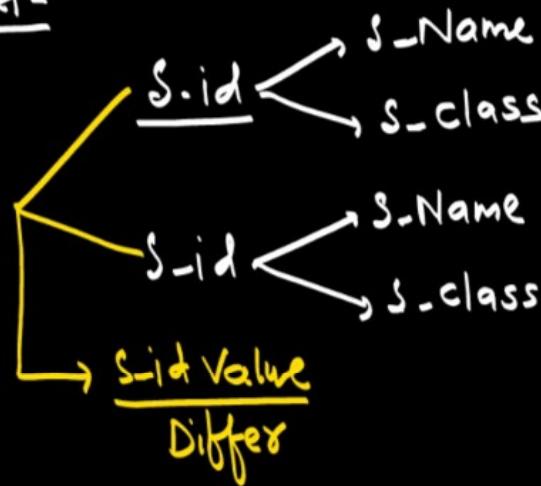
101	Ram	10	102	Mohan	12	103	Hari	11
-----	-----	----	-----	-------	----	-----	------	----

जैसे - Apache Cassandra, Hbase, Google BigTable.

b. KeyBased NoSQL

- इस DataBase में Data को Key-Value के Pair में store किया जाता है।
- इन Keys का Data मल्टी-दो सकता है।

जैसे-



- जैसे- Amazon DynamoDB, Redis DB, Memcached.

* Types of NoSQL DataBase ⇒

- A. Document Based
- B. Key Based
- C. Wide Column Based
- D. Graph Based

A. Document Based ⇒

→ इस प्रकार के DataBase में Data को JSON (JavaScript Objects) के रूप store किया जाता है।
जैसे - MongoDB, CouchDB, Firebase.

→ NoSQL में Data को Scaling कर Store किया जाता है।

→ 2 प्रकार

- A. Horizontal Scaling
- B. Vertical Scaling

A. Horizontal Scaling ⇒

→ Existing Server के द्वारा नई Server Machine को Attach करना Horizontal Scaling कहलाता है।

B. Vertical Scaling ⇒

→ Existing server Machine पर नया Hardware / Storage (RAM, HardDisk) Add करना Vertical scaling कहलाता है।

* NOSQL →

→ Not in SQL

→ Non Relational DataBase

→ NOSQL में Data को बिना किसी Structure के store किया जाता है।

S-id	Name	Class
101	Ram	10
102	Kari	12
103	Mohan	11

RDBMS - Student

S-id	Name	Class
101	Ram	10
102	Kari	12
103	Mohan	11

NoSQL = Student

120 t
 $\frac{110 - 120}{200} : 60\% = 18.40 \text{ t}$
 R: 35 40 t
 I 40% II 40%] 40%
 60% 60%
 $80/200 = \text{पात्र}$
 $10/100$
 5900 t

* Timestamp Based Concurrency Control ⇒

- Transaction Ordering Method
- इसमें जब भी Multiple Transactions Execution के लिए available होते हैं, तो इन्हें
इक TimeStamp (T) Allocate कर दी जाती है, जिससे प्रत्येक Transaction का Order
Decide कर दिया जाता है।
- इससे Transaction के Execution की Priority decide हो जाती है।

* Variations of 2 Phase Locking ⇒

A. Strict 2 Phase Locking ⇒

→ इसमें Transactions अपने Exclusive Locks (X) को केवल उनके रबत्तम की condition में ही Release कर सकते हैं।

B. Rigorous 2 Phase Locking ⇒

→ इसमें Transactions अपने सभी Locks (S-standard & X-Exclusive) को केवल उनके रबत्तम होने की condition में ही Release कर सकते हैं।

C. Conservative 2 Phase Locking ⇒

→ इसमें Transaction सभी Locks को Execution से पहले ही Acquire कर लेता है, ताकि Deadlock की संभावना पूर्ण रूप से खत्तम हो जाये।

* Advantages of 2 Phase Locking ⇒

A. Deadlock Avoidance ⇒

→ 2 Phase Locking को सदी तरीके से Apply करते हो Deadlock को रुक बेता Level तक Avoid किया जा सकता है।

B. Ensuring Serializability ⇒

→ 2 Phase Locking में Ensure करता है कि Multiple Transactions को Allocated Resources रुक ही serial Order में ही Execute हो ताकि Database में Data की Consistency challenge नहीं हो।

b. Shrinking Phase \Rightarrow

- इस Phase में Transaction को Allot किए गये Locks को Release किया जाता है।
- एक बार जब कोई Transaction Shrinking Phase में Enter करता है, तो उसे नया Lock Allot नहीं किया जा सकता है।

C. Two Phase Locking \Rightarrow

- 2 Phase locking एक Concurrency Control Protocol है, जो Ensure करता है कि Database Transactions के Execution के समय Consistency हो।
- 2 Phase
 - a. Growing Phase
 - b. Shrinking Phase

a. Growing Phase \Rightarrow

- इस Phase में Transaction, Required Resources पर Lock प्राप्त करता है।
- Transaction को जिन्हें भी locks की जहरत है, वह इसी Phase में Acquire किये जा सकते हैं तथा Acquired locks को leave नहीं किया जा सकता है।

* Types of Serializability =

A. Conflict Serializability →

→ जब एक साथ चल रहे Transactions को Re-Arrange किया जाता है, तो उनका Execution Sequence नहीं serial Order में हो, तो इसे Conflict serializability कहा जाता है।

B. View Serializability →

→ जब एक साथ Execute हो रहे Transactions एक serial Order में हो तथा उन्हें Re-Arrange करने की Need नहीं हो, तो इसे View Serializability कहते हैं।

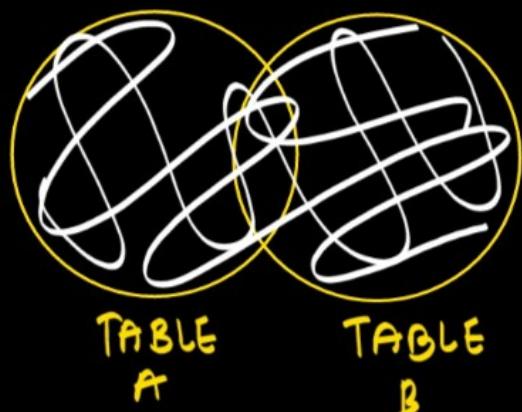
* Serializability →

- यह एक Property है, जो DataBase Transactions की Execution Scheme को बताता है।
- यह Ensure (सुनिश्चित) करता है कि Transactions एक-दूसरे के साथ उस प्रकार से Interleave / Interrelated हो, कि उनका Result एक Serial क्रम में हो जाए एक समय में केवल एक ही Transaction Execute हो सकता है।
- Serializability यह सुनिश्चित करती है, कि एक साथ यह रहे Multiple Transactions एक-दूसरे की Consistency को Affect नहीं करेंगे।
- Execution के दौरान Multiple Transaction के बीच होने वाले Conflict को Standard define करती है।

E. NATURAL JOIN →

- पृष्ठ JOIN 2 Tables को Common Columns के आधार पर जोड़ता है।
- पृष्ठ Columns से Values को match करके Rows return करता है।
- प्रत्येक DataTable में Data को Insert करते समय कुछ Business Rules Follow किए जाते हैं, इन्हें Data Constraints कहते हैं।

D. FULL JOIN ↳

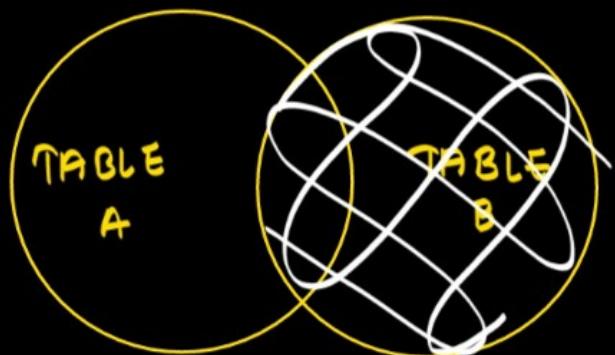


जटि- - SELECT columns
FROM TABLE_1
FULL JOIN TABLE_2
ON
TABLE1.Col = TABLE2.Col ;

→ यह LEFT तथा RIGHT JOIN से मिलकर बनता है तो दोनों Tables की सभी Rows को JOIN करता है।

C. RIGHT JOIN =>

→ इस Join में Right Table की सारी Rows तथा Left Table की केवल Matching Rows की दी लिया जाता है।

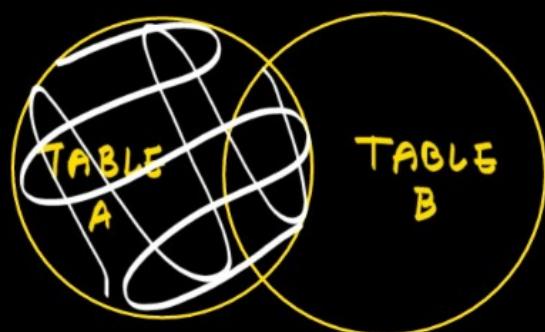


जैसे -

```
SELECT Columns  
FROM TABLE -1  
RIGHT JOIN TABLE -2  
ON TABLE -1 . Col = TABLE -2 . Col ;
```

B. LEFT JOIN

→ इस Join में Left Table की सारी Rows तथा Right Table की केवल Matching Rows को ही लिया जाता है।



ज्ञान = SELECT Column-Names
FROM Table_1
LEFT JOIN Table_2
On Table_1.Column = Table_2.Column;

P-ID	P-Cat	P-Code
101	Drink	1001
102	Food	1002
103	Dessert	1003

PRODUCT

P-Code	P-Description
1001	Coke, Pepsi
1002	Paneer, Dal
1003	Gulab Jamun, Ice Cream

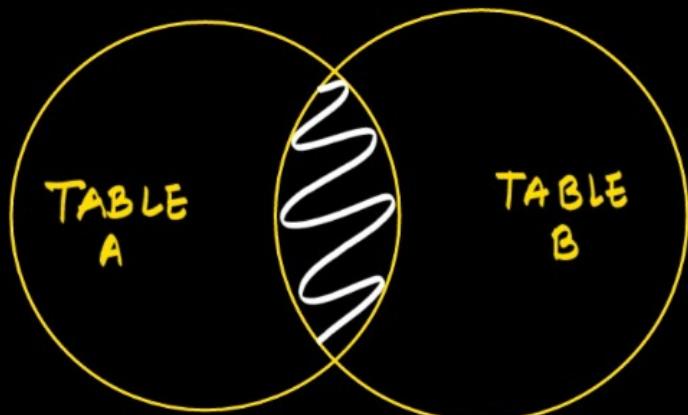
CATEGORY

```

SELECT PRODUCT.P-ID,
PRODUCT.P-Cat, CATEGORY.P-Description
INNER JOIN CATEGORY ON
PRODUCT.P-Code = CATEGORY.P-Code;
    
```

P-id	P-Cat	P-Description
101	Drink	Coke, Pepsi
102	Food	Paneer, Dal
103	Dessert	GJ, IC

A. INNER JOIN →



→ 2 अलग-2 Tables की Same Rows को Select करना INNER JOIN कहलाता है।
Common

जैसे- SELECT Table1.Column1, Table1.Column2
FROM Table1
INNER JOIN Table2
ON Table1.matchingColumn = Table2.matchingColumn;

* SQL Join =>

- 2 या अधिक Tables को मिलाकर Data को Fetch करता।
- 2 या अधिक Tables की Rows को किसी एक Column के आधार पर जोड़ना Join कहलाता है।
- 2 या अधिक Tables के Data को combine करने में USE.
- किसी एक Table से Related Data को दुसरी Table से लेने में JOIN का USE.
- Types =>
 - A. InnerJoin
 - B. Left Join
 - C. Right Join
 - D. FULL Join
 - E. Natural Join

NULLIF
 ↓
 → 2 Value को
 Compare करता है।
 → Same होने पर NULL
 Return करेगा।

Count(*) = Counts NULL
 Count(exp) = Don't count
 NULL

30. What will be the result of the following SQL query?

SELECT COUNT (NULLIF (salary, 5000)) FROM employees;
 (Assume there are 10 employees, 3 of whom have a salary of 5000.)
 (★)

(A) 3

(B) 7

(C) 10

(D) More Than One of The Above

(E) None of The Above

5000 NULL
 10 5,5 NULL 10
 Count(*)
 10

29. What will be the result of the following SQL query?

SELECT POWER (2, 3) + SQRT (16);

(A) 8

(B) 12

(C) 10

$$\begin{array}{r} \cancel{2 \times 2 \times 2} \\ \hline 8 + 4 \\ \hline \textcircled{12} \end{array}$$

(D) More Than One of The Above

(E) None of The Above

Cost-id	Amt
101	1300
102	200
103	600
101	1300
103	600
102	200

28. Consider the table orders:

order_id	customer_id	order_date	amount
1	101	2024-01-01	500
2	102	2024-01-02	200
3	101	2024-01-03	800
4	103	2024-01-04	600

What will be the result of the following SQL query?

`SELECT customer_id, SUM(amount) FROM orders GROUP BY customer_id ORDER BY SUM(amount) DESC LIMIT 1;`

(A) (101, 1300)

(B) Error in query

(C) (102, 200)

(D) More Than One of The Above

(E) None of The Above

⇒ NULLIF()

→ 2 Values को
Compare करता
है।
→ Same होने पर
NULL

⇒ COALESCE()

→ Used to handle NULL Values
→ If Function Extra Value के साथ declare होता है, तो उसे ignore करता है।
function में NULL Value दे, तो इसके साथ वाली Value O/P में display होती है।

27. What is the output of this query?

SELECT COALESCE (NULLIF (5, 5), 10);

- (A) 5
- (B) 10
- (C) 5, 5, 10
- (D) More Than One of The Above
- (E) None of The Above



6. Table Constraint

- किसी एक Column पर लगाया जाने वाला constraint, Column Constraint कहलाता है।
Whereas
Group of Columns पर लगाया जाने वाला Constraint, Table Constraint कहलाता है।
- Column Constraints को Column के साथ ही define किया जाता है जबकि Table Constraint की सभी Columns की Definition के बाद Last में define किया जाता है।

जैसे - CREATE TABLE Product (
 PID Number(4) Not NULL,
 PName Varchar(25) Not NULL,
 Company Varchar(25),
 Price Number(6,3)
 PRIMARY KEY (PID, PName));

TABLE CONSTRAINTS

COLUMN CONSTRAINTS

F. Check Constraint

- किसी Column पर TRUE/FALSE Condition को Apply होगा।
- मर्दात् किसी Column में Value नहीं Insert होगी जब वह Condition को satisfy करेगी।

उदाहरण = CREATE TABLE Product (

PID Number(3) PRIMARY KEY,

PName Varchar(25) CHECK (PName LIKE 'P%.'!),

Company Varchar(25),

Price Number(6,3));

PID	PName	Company	Price
PK	वही Values Insert होंगी, जिनका पहला Char = P होगा।	NULL	NULL

E. Default Constraint

- इस constraint का प्रयोग कर किसी भी column के लिए default value set की जा सकती है।
- INSERT INTO Command के खाल column की value नहीं देने पर वह default value ले लेता है।

जैसे - CREATE TABLE Product (

PID Number(4) PRIMARY KEY,

PName Varchar(25) NOT NULL,

Company Varchar(25),

Price Number(6,3) DEFAULT 100000.000);

PID	PName	Company	Price
PK	NOT NULL	NULL	Default

PID = Product Table की PK

Company

Product Table में PK

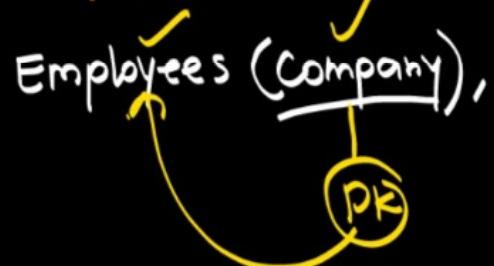
Employee Table में PK

D. Foreign Key Constraint ↗

- A foreign key is used to provide Relationship between two tables, that is why it is referred as Reference key.
- Foreign Key is a primary key in another table.

जैसे ↗ CREATE TABLE product (
 PID Number(4) PRIMARY KEY,
 PName varchar(25) Not NULL,
 Company Varchar(25) REFERENCES Employees(Company),
 Price Number(6,3));

PID	PName	Company	Price
PK	Not NULL	FK PK = Emp Table Company	NULL



C. Primary Key Constraint

- एक Primary Key DataBase Table का ऐसा Column होती है, उन्हें Record/Row को Uniquely Identify करती है।
- By default this column is Not NULL & Unique.

जैसे → CREATE TABLE Product (

PID Number(4) PRIMARY KEY,
PName Varchar(25) Not NULL,
Company Varchar(25),
Price Number(6,3));

PID	PName	Company	Price
(PK)	Not NULL	NULL	NULL

B. Unique Key Constraint

- This constraint ensures that no rows have the same values in the specified column.
- एक Good Database की Tables में Multiple Unique Key Constraints होते जाते हैं।

जैसे → CREATE TABLE Product(

PID Number(4) Not NULL UNIQUE,

PName Varchar(25) Not NULL,

Company Varchar(25),

Price Number(6,3));

PID	PName	Company	Price
Not NULL & Unique	Not NULL	NULL	NULL

A. Not NULL Constraint ↗

- By default column can hold NULL Values.
- When a column is defined as Not NULL then it becomes mandatory.
- Not NULL का अर्थ Column Empty नहीं रह सकता है।

जैसे ↗ CREATE TABLE Product (

PID Number(4) Not NULL,
PName Varchar(25) Not NULL,
Company Varchar(25),
Price Number(6,3));

PID	PName	Company	Price
Not NULL	Not NULL	NULL	NULL

* SQL Constraints →

- These are the rules enforced on Data Columns on Table.
- इनका प्रयोग Table में Data insertion की Quantity को Limit करते के लिए किया जाता है।
- Constraints ensure the accuracy & reliability of Data in Table.
- 2 Types
 - Column Level
 - Table Level
- Generally we use following types of Constraints -
 1. NOT NULL
 2. UNIQUE Key
 3. PRIMARY Key
 4. FOREIGN Key
 5. DEFAULT
 6. CHECK

* MAX() =

→ Find the maximum value in a column.

Ex- SELECT MAX(Phy) FROM Exam;

100

* MIN() =

→ Find the minimum value in a column.

Ex- SELECT MIN(Phy) FROM Exam;

33

CHANGE (CONT'D)
31/10/14

- ✓ 7-8 AM = MS Office
8-9 AM = YT 9-5:30
6-8 PM = BPSC ✓
✓ 8-9:30 YT

* AVG() ⇒

→ Find out the Average of a Numerical Column.

Ex- SELECT AVG(cs) FROM Exam;

74.7

* SUM() ⇒

→ Find out the Total of a Numerical Column.

Ex- SELECT SUM(cs) FROM Exam;

747

* COUNT(*)

→ Counts total values in a column.

Ex-1 ⇒ SELECT COUNT(*) from Exam;

10

Ex-2 ⇒ SELECT COUNT(RegNo) from Exam WHERE CC='C3';

4

* SCALAR Functions ⇒

→ ऐसे Functions जो एक बार में केवल एक ही Value पर काम करते हैं।

जटिल →

Numeric Functions = ABS(), SQRT(), POWER(), ROUND()

String Functions = UPPER(), LOWER(), LENGTH(), LTRIM(), RTRIM(), SUBSTR()

Date Functions = LAST_DAY(), NEXT_DAY(), MONTHS-BETWEEN()

Conversion Functions = TO-NUMBER(), TO-CHAR(), TO-DATE

* SQL Functions ⇒

- Available Data Items को Manipulate कर Result generate करता।
- There are many built-in functions categorised in GROUP functions & SCALAR functions.

* GROUP Functions ⇒

- Functions that act on a set of values.
- A group function can take entire column of data as its arguments and produces a single value as result.

जैसे - AVG(), COUNT(), SUM(), MIN(), MAX()

EXAM ✓

RegNo	Name	CC	Phy	Che	Mat	Cs	Total	City
101	Ajay	C1	98	100	97	99	394	Hassan
102	Banu	C2	38 ✓	50	37	49	174	Belur
103	Chandan	C2	100✓	100	97	99	396	Mysuru
104	John	C3	78	80	67	79	304	Alur
105	Kaleem	C1	88	80	91	79	338	Hassan
106	Raheem	C2	100✓	98	97	79	374	Hassan
107	Sanjay	C3	47 ✓	60	56	78	241	Alur
108	Tarun	C3	33 ✓	34	77	28	172	Arasikere
109	Uday	C2	100✓	98	97	79	374	Hassan
110	Venki	C3	47	60	56	78	241	Belur

747

Query 7

→ DISTINCT ⇒ यह Table के Duplicate Records को Eliminate कर Unique Records Display करेगा।

Syntax ⇒ SELECT DISTINCT Col1, Col2, Col3
FROM Table-Name
WHERE Condition;

जैसे ⇒ SELECT DISTINCT Name, Age, Fees
FROM student
ORDER BY Name;

DISTINCT
↑
Name, Age, Fees
↓
MATCH
पहला ✓

* SELECT DISTINCT Age
FROM student
ORDER BY Age;

Age
18
19
20

Name	Age	Fees
Hari	20	10000
Krishna	18	15000
Mohan	19	15000
Ram	19	10000

* GROUP BY + HAVING ⇒

→ SELECT Col-name Function_Name() AS New-Col-name
FROM Table-name

GROUP BY Col-name

HAVING Function-name() \geq Value;

→ SELECT Age sum(fees) As Amount
FROM student

GROUP BY Age

HAVING sum(fees) > 20000;

AGE	AMOUNT
19	25000
20	25000

उत्तर = SELECT Age Sum(Fees) AS Amount
 FROM Student
 GROUP BY Age ;

Sum = 61171

Age	Amount
18	15000
19	25000
20	25000

Query 6. ⇒

→ GROUP BY ⇒ Group rows based on column values.

⇒ इसमें एक नया Column Virtually Create किया जाता है, जिसके पहले AS Keyword use होता है।

Syntax ⇒ SELECT Col-1, Col-2 COUNT(*) AS New-Col-Name
 FROM table-name
 GROUP BY Column-Name;

उदाहरण ⇒ SELECT Age Count(*) AS Number
 FROM student
 GROUP BY Age;

Age	Number
18	1
19	2
20	2

* SELECT * FROM student
ORDER BY Class ASC, Age DESC;

ID	Name	Class	Age	Fees
101	Hari	10	20	10000
103	Ram	10	19	10000
105	Ram	11	20	15000
102	Mohan	11	19	15000
104	Krishna	11	18	15000

18 ✓ 1
19 ✓ 2
20 ✓ 2

ID	Name	Class	Age	Fees
101	Hari	10	20 ✓	10000
102	Mohan	11	19	15000 ✓
103	Ram	10	19	10000
104	Krishna	11	18	15000 ✓
105	Ram	11	20.	15000 ✓

DESC

ID Number(3)

Name Varchar2(25)

Class Number(2)

Age Number(3)

Fees Number(5,2)

Query 5. →

→ ORDER BY -

SELECT Column-list FROM table-name
WHERE Condition
ORDER BY Column1 ASC, Column2 DESC;

Ans = SELECT ID, Name, Age FROM Student

WHERE Class = 11

ORDER BY Name;



By Default Ascending

ID	Name	Age
104	Krishna	18
102	Mohan	19
105	Ram	20

AND

X	Y	$O = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

Query 3. → SELECT Name, Age, Fees FROM student
WHERE Age > 19 AND Fees > 10000;

COND-1

COND-2

दोनों Conditions
Follow करता है।

AND } Conjunctions
&
OR } Operators

Name	Age	Fees
Ram	20	15000

Query 4. → SELECT Name, Age, Fees FROM Student
WHERE Age > 19 OR Fees > 10000;

Name	Age	Fees
Mari	20	10000
Mohan	19	15000
Krishna	18	15000
Ram	20	15000

Query 1. ⇒ SELECT Id, Name from student;

ID	Name
101	Hari
102	Mohan
103	Ram
104	Krishna
105	Ram

Query 2. ⇒ SELECT Id, Name, fees FROM student
WHERE fees > 10000;

ID	Name	Fees
102	Mohan	15000
104	Krishna	15000
105	Ram	15000

ID	Name	Class	Age	Fees
101	Hari	10	20 ✓	10000
102	Mohan	11	19	15000 ✓
103	Ram	10	19	10000
104	Krishna	11	18	15000 ✓
105	Ram	11	20.	15000 ✓

DESC

ID Number(3)

Name Varchar2(25)

Class Number(2)

Age Number(3)

fees Number(5,2)

8. Select ⇒

→ Database से Data को Retrieve करता।

Syntax ⇒

A. SELECT * FROM tablename;

⇒ Table के सारे Records Display करता है।

B.

SELECT Col-1, Col-2... Col-N FROM tablename;	Mandatory PART
WHERE conditions; GROUP BY Column-List; HAVING Conditions; ORDER BY Column-Name; └─ Asc (Default) DESC	Optional PART

7. Delete ⇒

→ Existing Row को Delete करता |

→ इसके साथ WHERE Clause के साथ Condition की जारी जरूरी है अन्यथा All Records Delete हो जाएंगे |

Syntax ⇒ DELETE table-name
 WHERE Condition;

जैसे DELETE student where Id = 101;

 DELETE student where Name = 'Malti';

⇒ Select * from Student;

ID	Name	Age	Phone
102	Raju	24	2657432
104	Shree	21	2498675

⇒ Select * from student;

Id	Name	Age	Phone
101	Ram	40	2567421
102	Raju	24	2657432
103	Manti	19	2694327
104	Shree	21	2498675

6. Update ⇒

- Table का Data update करना।
- Existing Row को Update करना।
- इसमें New Value को SET Keyword के साथ लिखते हैं।
- इसमें update के लिए WHERE के साथ Condition दी जाती है, यदि WHERE clause नहीं हो तो पुरी table update हो जाती है।

Syntax ⇒ UPDATE table-name SET Col-Name = New-Value
WHERE Condition;

जैसे - UPDATE student SET Name = 'Malti' WHERE Id = 103;
* यह Command SEETA को MALTI में change कर देगी।

UPDATE student SET Age = 40 WHERE Name = 'Ram';
* यह Command Ram की Age 40 कर देगी।

* Method - 2 ⇒

→ Multiple Records | Rows at a time.

Syntax ⇒ `INSERT INTO table-name (col-1, col-2 col-N) Values
(Value1, Value2 valuen), (Value1, Value2 valuen);`

Ex = `INSERT INTO student (Id, Name, Age, Phone) Values
(103, 'Seeta', 19, 2694327), (104, 'Shree', 21, 2498675);`

⇒ Select * from student;

Id	Name	Age	Phone
101	Ram	26	2567421
102	Rajiv	24	2657432
103	Seeta	19	2694327
104	Shree	21	2498675

* Methods ↳

Syntax ⇒ INSERT INTO table-name values (value1,
value2 . . . value N);

⇒ INSERT INTO student Values(101, 'Ram',
26, 2567421);

⇒ INSERT INTO student Values(102, 'Raju',
24, 2657432);

Id	Name	Age	Phone
Number	Varchar	Number	Num.

* Select * from student;

Id	Name	Age	Phone
101	Ram	26	2567421
102	Raju	24	2657432

4. Drop =

→ Table की structure सहित Delete करना।

Syntax =

DROP table table-name;

DROP table student;

5. Insert =

→ INSERT INTO Statement का Use Table में New Row Insert करते में किया जाता है।

→ इसे 2 Methods से Use में किया जा सकता है।

3. Alter ⇒

→ इस Command का प्रयोग कर Table को Change/Modify कर सकते हैं।

Syntax ⇒

A. Alter table table-name ADD (Column-name data-type);

⇒ Alter table student ADD (Add varchar2(25), phone Number(10));

B. Alter table table-name MODIFY (Column-name data-type);

⇒ Alter table Student MODIFY (Phone Number(7));

C. Alter table table-name DROP (Column-name data-type);

⇒ Alter table Student DROP (Add varchar2(25));

DROP (Add);

} Column को Table के
Structure से Delete

2. Describe ↗

→ Table के सारे Columns including datatypes & constraints display करता।

जैसे- Describe student;

Name	NULL?	Data Type
Regno	Not NULL	Number(6)
Name		Varchar2(25)
Class		Number(3)
DOB		Date
Fees		Number(6,3)

→ Also known as DESC.

- इस Command का use करके Table बनाए के लिए इसके प्रत्येक column का नाम datatype के साथ लिखा जाता है।
- Multiple columns को comma से separate किया जाता है।

जैसे `Create table table-name (CN-1 DT-1, CN-2 DT-2 ... , CN-N DT-N)`

RegNo	Name	Class	DOB	Fees	[RegNo=Primary Key]
Number(6)	Varchar2(25)	Number(3)	Date	Number (6,3)	

⇒ `Create table Student (Regno Number(6), Name Varchar2(25),
class Number(3), DOB Date,
Fees Number(6,3), PRIMARY KEY (Regno));`

★ Commands ↳

1. Create ↳

→ नया DataBase या Table बनाना।

जैसे =

```
create database database-name;  
create database kautilya;
```

↓
DataBase को Create कर उसमें Table बनाने के लिए उसे Open करना जरूरी होता है।

↓

```
use database-name;  
use kautilya;
```

5. EXISTS

This operator is used to search for the presence of a row in specified Table.

6. IN

It is used to compare a value in a list of literal values.

7. LIKE

This operator is used to compare a value in similar values using wildcard character (%).

8. NOT

NOT EXISTS, NOT IN, NOT BETWEEN

9. OR

It is used to combine multiple conditions.

* SQL Logical Operators →

Operator	Description
1. ALL	It is used to compare a value to all values in another value set.
2. AND	This operator allows the existence of multiple conditions in an SQL's WHERE clause.
3. ANY	This operator is used to compare a value to any applicable value in the list according to the condition.
4. BETWEEN	This operator is used to search for values that are within a set of values in between minimum & maximum value.

→ Keywords = Select , From & Where

→ Identifiers = Name , Student & Roll No

↓
Refers any object in Database

→ Clauses = From Student & Roll No = 109

→ Name & Roll No = Columns

→ Student = Table Name

↳ From के साथ TableName आता है।

→ = → Operator

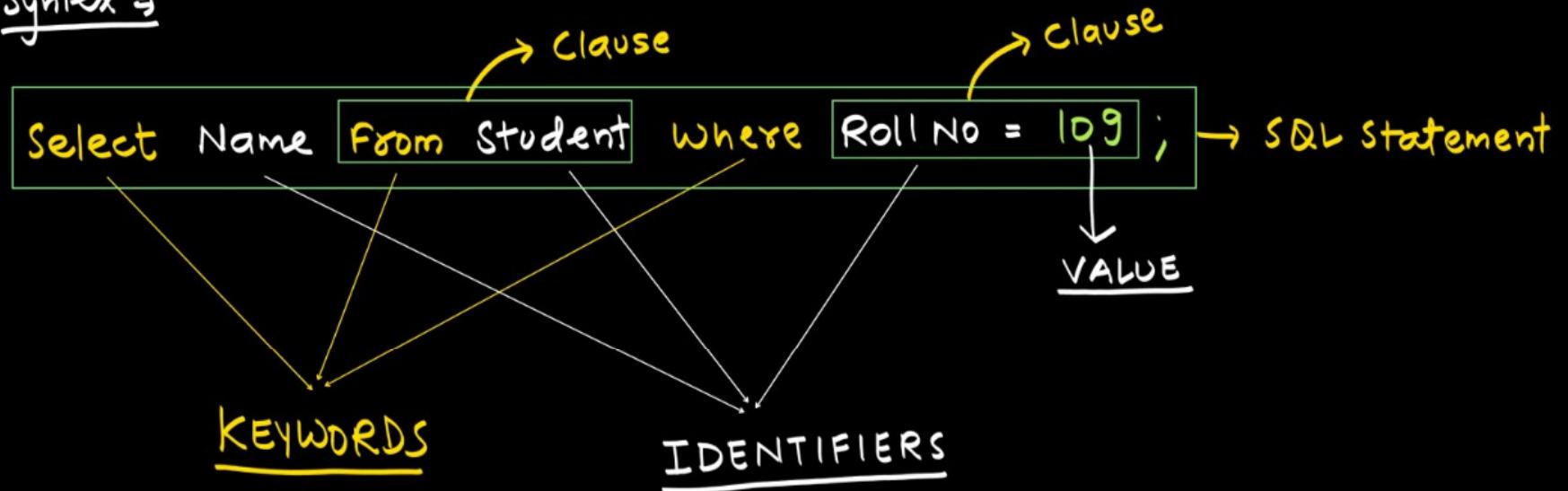
→ 109 = Numerical Constant

* Select & From
Mandatory

* Where
Optional

* Structure of SQL Command =>

- यहाँ SQL Commands Keywords, Identifiers & Clauses का Combination होती है।
- प्रत्येक SQL Command Keyword के start होती है।
- Syntax :-



7. Raw ⇒

→ 2000 Bytes तक की Binary file को store करना।

8. Long Row ⇒

→ 2GB तक की Binary File को store करना।

4. Date ⇒

- Used to store Date in field.
- Various Date formats.
- Syntax ⇒ DOB Date

5. Time ⇒

- Used to store Time in field.

6. Long ⇒

- used to store variable length string upto 2 GB.
- Syntax ⇒ Description Long

2. Char

→ किसी Field में character Data Insert करने में उपयोगी।

→ Max Size = 255 char

→ Syntax ⇒ Name Char(15)


अगर Name की Value = "Rani" हो तो यह field size 15 ही होगा।

3. Varchar ⇒ Varchar2

→ किसी field में Variable Length (परिवर्तनीय लंबाई) का character Data Insert करने में उपयोगी।

→ Max Size = 2000 Char

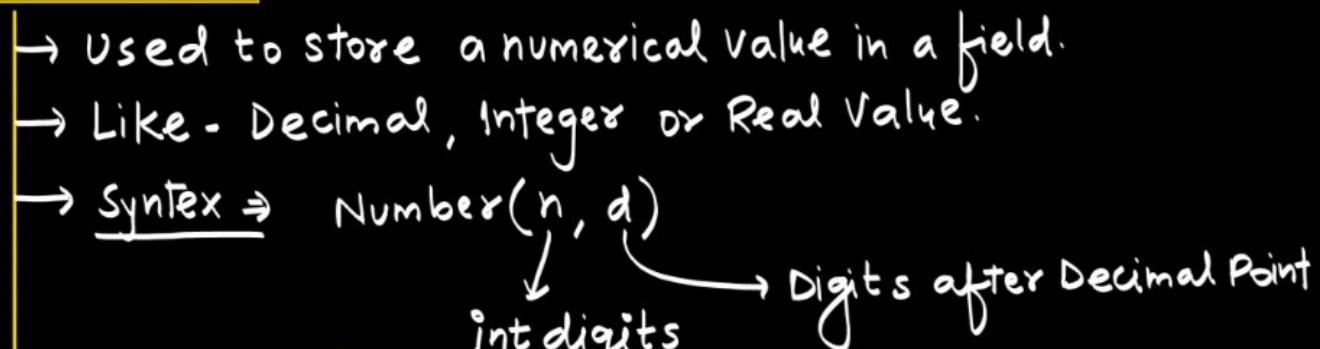
→ Syntax ⇒ Name Varchar(50)

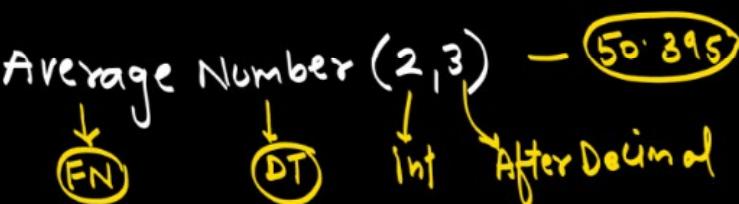

अगर Name की Value = "Rani" हो तो field size 4 होगा।

* Data Types in SQL =

1. Number
2. Char
3. Varchar
4. Date / Time
5. Long
6. Raw
7. Long Raw

1. Number =>

→ Used to store a numerical value in a field.
→ Like - Decimal, Integer or Real Value.
→ Syntax => Number(n, d)

Like => marks Number(2) — 50

Average Number(2,3) — 50.395


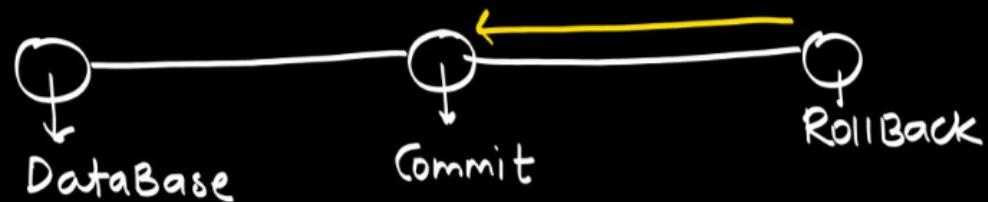
13. Save Point ⇒

→ इस Command का प्रयोग किसी Particular Transaction को Mark करने में किया जाता है,
ताकि उस Selected Transactions को ही RollBack कर सकते हैं।

11. Commit ↗

→ इस Command के द्वारा User के द्वारा किये गये Transactions/ Queries को Permanent Store किया जाता है।

12. RollBack ↘



→ इस Command के द्वारा Last Committed Transactions को Revert कर दिया जाता है।

8. Select ⇒

→ Standard Query by which we retrieve table's data.

9. Grant ⇒

→ When we allocate some permission to the user of the Database.

10. Revoke ⇒

→ If want to deny the permission or deallocate the permission of the user of the database.

6. Insert ⇒

- किसी Table में New Record Insert करने में उपयोगी।
- इस Command द्वारा Table में Record Insert करते हमें Table के सभी Columns की Value देनी पड़ती है।
- NULL Value can be given.

7. Update ⇒

- Table में Available Data को Modify करना।
- Update के साथ WHERE Clause भीजोने से Selected Data की Update होता है।
- यदि WHERE Clause का use नहीं किया जाए, तो Table के सब Records Update हो जायेगे।

* * DROP V/S TERMINATE V/S DELETE →

DROP	TERMINATE	DELETE
<p>1. Table के Structure & Complete Data को स्टार्फ हप से Delete करता है।</p> <p>2. No Recovery of Deleted Data.</p>	<p>1. Table के Complete Data को एक साथ स्टार्फ हप से Delete करता है But Table के Structure को Delete नहीं करता है।</p> <p>2. Faster Command than Delete.</p> <p>3. No Recovery of Deleted Data</p>	<p>1. Table के Records को One by One Delete करता है।</p> <p>2. Deleted Data can be Recovered.</p> <p>3. Slower Command than Terminate.</p>

5. Delete ↳

- इस कमाण्ड का प्रयोग कर Table के Records को Delete किया जाता है।
- Delete Command के साथ WHERE Clause का use होता है, जिसके Selected Data ही Delete होता है।
- यदि Delete Command के साथ WHERE Clause का use नहीं किया जाएगा, तो Table का Complete Data Delete हो जाएगा।

3. Drop ⇒

→ Table का structure + Data दोनों को एक साथ Delete करने में उपयोगी।

→ **Complete Table Delete** करता।

→ Drop के द्वारा Delete की गई Table कभी भी Recover नहीं कर सकते हैं।

4. Terminate ⇒ TRUNCATE

→ यह Command Table का सारा Data वह साथ Delete कर देती है।

→ यह Command Table का structure Delete नहीं करती।

→ Deleted Data Recovery नहीं किया जा सकता है।

S-id	Name
101	ABC
102	XYZ

DROP

S-id	Name
101	ABC
102	XYZ

TERMINATE

* Working of Commands *

1. Create ↳

- Creation of DataBase or Table
- Table Name में Space is not allowed but we can use underscore (-).

2. Alter ↳

- किसी Table में नया Column Add करता |
- किसी Available column को Drop | Delete करता |
- Column का DataType Change करता |
- Column का Name & Size Change करता |
- Table का नाम Change करता |
- Table के किसी Column पर Constraints Add करता |
- Alter के द्वारा Table का structure update किया जा सकता है

E. Constraints ⇒

- a. Primary Key Constraints
- b. Foreign " "
- c. Unique " "
- d. check constraints
- e. Default Constraints
- f. Not NULL constraints

* SQL Commands =>

A. DDL (Data Definition Language) =>

- a. Create
- b. Alter
- c. Drop
- d. Terminate / Truncate

B. DML (Data Manipulation Language) =>

- a. Select
- b. Insert
- c. Update
- d. Delete

C. DCL (Data Control Language) =>

- a. Grant
- b. Revoke

D. TCL (Transaction Control Language) =>

- a. Commit
- b. Roll Back
- c. Savepoint

- Case Insensitive language.
- वर्ष 1970 में IBM Company ने Father of DBMS = Dr E F Codd के Relational Algebra के Research Papers का use करके SEQUEL को बनाया।
- SEQUEL was renamed as SQL.
- SQL में TableName 64 characters का हो सकता है, तथा यह case sensitive होता।

~~SQL~~ →

- Structured Query Language
- Old Name = SEQUEL
 - ↓
Structured English Query Language
- Developed by - IBM
- Declarative language.
- Domain Specific language.
 - ↳ It works on RDBMS only.

B. NonTrivial FD \Rightarrow

→ if x, y are the attributes of relation R, then it would be Nontrivial FD , if

$$x \cap y \neq \emptyset$$

$$S_id \rightarrow S_Name$$

→ Non-Trivial FD जो Determinent होनेशा नये Attribute को Determine करता है

C. SemiTrivial FD \Rightarrow

→ if x, y are the attributes of R , then there would be semiTrivial FD if-

दोनों side Same Variables नहीं बल्कि भिन्न

$$\frac{S_id + S_Name}{y} = \frac{S_id + S_Age}{y} \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} y \text{ is not the subset of } x.$$
$$S_Name = S_Age$$

Q Let $E\text{-id} + E\text{-name} = E\text{-id} + E\text{-Age}$ in a relation employee.
will this be a trivial FD?

A

$$\underbrace{E\text{-id} + E\text{-Name}}_x = \underbrace{E\text{-id} + E\text{-Age}}_y$$

$$x \geq y$$

$$\boxed{x \cap y}$$

$$\cancel{E\text{-id} + E\text{-Name}} = \cancel{E\text{-id} + E\text{-Age}}$$

$$\boxed{E\text{-Name} = E\text{-Age}}$$

$$\hookrightarrow x \neq y$$

$\boxed{\text{i.e. No Trivial FD}}$

★ ★
★ ★ $x \geq y = x \cap y \neq 0$

दोनों Side के Attributes
Same होने पर Trivial FD
होती है

$$\underbrace{S\text{-id} + \text{Name}}_{x} = \underbrace{\text{Name}}_y \quad \left. \right\} \quad y \text{ is the subset of } x = \checkmark$$

x ≥ y

- x side Columns की संख्या, y side Columns की संख्या से अधिक होना प्राप्ति, then Trivial FD होगी।
- दोनों side के attributes same होने पर Trivial FD होगी।

$\underbrace{s_id + Name}_{x} \rightarrow \underbrace{Name}_{y}$

$$x \geq y$$

→ x side columns की संख्या, y side
columns की संख्या

A. Trivial FD \Rightarrow

S-id	Name	Age
101	XYZ	22

→ if x, y are the attributes of Relation R, then $x \rightarrow y$ will be in Trivial FD when-

y, x का subset होगा

$$x \geq y$$

→ कोई ग्री Attribute शब्द को देखा Determine करता है, यह statement देखा True होगा।

$$S_id \rightarrow S_id$$

* \rightarrow Determinent की Same value पर Determiner दमेरा Same होगा, But Determiner की Same value पर Determinent का Same होना नहीं है।

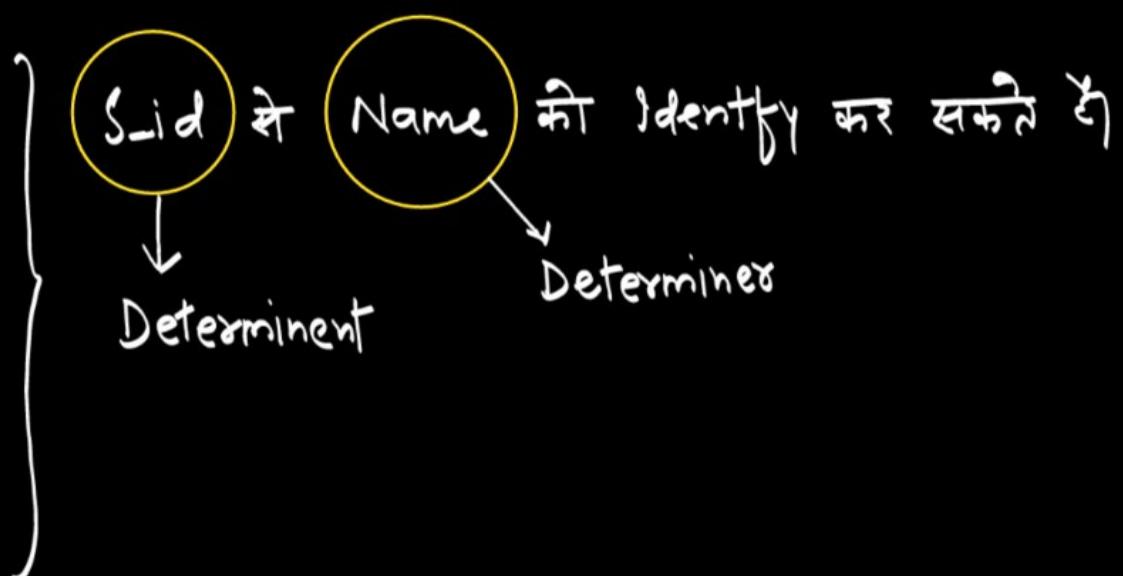
* Types of Functional Dependancy \Rightarrow

- A. Trivial FD
- B. Semi Trivial FD
- C. Non Trivial FD

* Functional Dependency \Rightarrow

→ किसी भी Relation के Attributes के मध्य Relationship को Functional Dependency कहते हैं।

S-id	Name	Age
101	N1	22
102	N2	26
103	N3	30
104	N4	27



→ S-id से Name को Identify किया जाता है, इसलिए S-id = Determinant & Name = Dependent / Dependant होते हैं।

6. 5NF

- कोई नी Table जो 4NF में हो तथा उसमें Join Dependencies नहीं है।
- इसे PJNF (Project Join Normal Form) कहते हैं।
- Non loss Decomposition

जब हम Data में Redundancy को Remove करने के लिए Table को Split करते हैं तथा बाद में जब Normalized करने के लिए Merge किया जाता है तो कोई नी Data loss नहीं होगा यादि।

⇒ यदि इन दोनों Tables में Cross Product किया जाए तो -

$R_1 \times R_2$

SID	SName	Cid	CName
S1	A	C1	C
S1	A	C2	D
S2	B	C1	C
S2	B	C2	D

Multivalued Dependency
↓

$SID \rightarrow\!> CID$

$SID \rightarrow\!> Cname$

$SName \rightarrow\!> Cname$

4NF (X)

⇒ इस Table में Multivalued Dependancy है, अतः इस 4NF में नहीं होगा।

⇒ इसे 4NF में लाने के लिए 2 अलग-² Tables में split किया जाता है।

Q ⇒ Consider the Database table of a class that has two relations R1 contains Student ID (SID) and studentName (SName) and Relation R2 contains Course ID (CID) and Course Name (CName).

SID	SName
S1	A
S2	B

CID	CName
C1	C
C2	D

} 4NF ✓

★ Properties ⇒

1. Relation must be in BCNF.
2. Table में कोई भी MultiValued Dependancy नहीं होनी चाहिए।

→ ↗
→

Ex ⇒

M-Model	Mfg-Year	Colour
C-7	2013	White
C-7	2013	Gray
C-11	2014	White
C-11	2014	Gray

} यहाँ Mfg-Year & Colour से दुसरे से Independent हैं तथा इन दोनों M-Model पर Dependant हैं।

M-Model → Mfg-Year
M-Model → Colour

5. Fourth Normal Form

- 4th & 5th are the highest form of Normalization.
- 4NF Multivalued Dependancy को control करता है।
- Multivalued Dependancy नब होती है, जब किसी Table में एक से प्रधिक Independent Multivalued Attributes होती हैं।
- इसमें कम से कम तीन Attributes की Need होती है, जिनमें से दो Independent Attributes तीसरे Attribute पर depend रहते हैं।
- Denote = →>

REVISION

1NF	2NF	3NF	BCNF
Each Attribute of Relation must contain atomic values.	<ul style="list-style-type: none"> → Must be in 1NF → No Partial Dependancy <ul style="list-style-type: none"> ↓ Prime → NonPrime को determine नहीं करता चाहिए। 	<ul style="list-style-type: none"> → Must be in 2NF → No Transitive Dependancy <ul style="list-style-type: none"> ↓ NP → NP को determine नहीं करता चाहिए। 	<ul style="list-style-type: none"> → Must be in 3NF → left hand side of each FD must be either a SK or a CK.

Q \Rightarrow Relation $R = (A, B, C, D, E, F)$

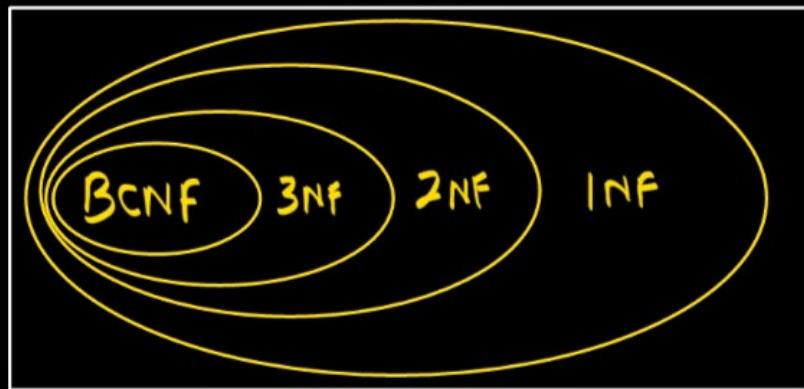
$CK = (A, C, D, \underline{F}, E)$ \textcircled{B} $\overset{NP}{\cancel{N}}$

$FD = \{ \underset{\text{CE}}{C} \rightarrow D, \underset{\text{CE}}{F} \rightarrow E, \underset{\text{CK}}{A} \rightarrow E, \underset{\text{CK}}{D} \rightarrow F \}$

यहां पर Relation BCNF में नहीं.

A \Rightarrow \textcircled{YES}

$\cancel{D} = CK$



|



4. BCNF \Rightarrow

- Boyce Codd Normal Form
- Relation 3NF में होना चाहिए |
 - Atomic Value
 - No Partial Dependancy
 - No Transitive Dependancy
- Left hand side of each functional dependancy is superkey or candidate key.

1NF = Atomic Value
2NF = PDX
3NF = TDX

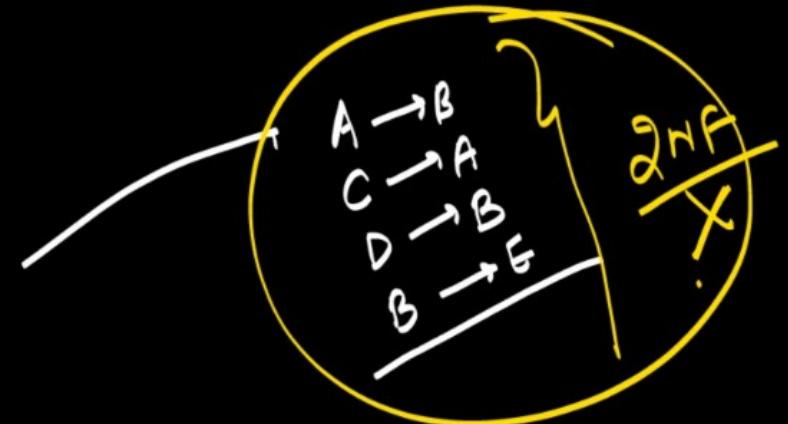
R \Rightarrow

Relation $R_1 = (A, B, C, D, E, F)$

CK = (A, C, BD)

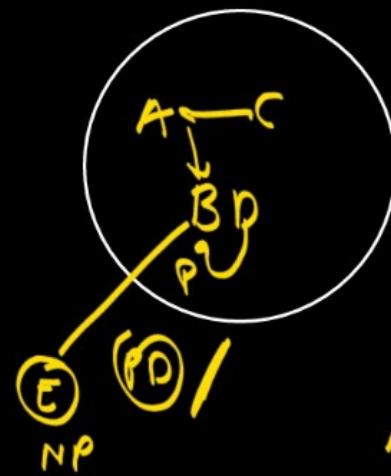
FD = (A \rightarrow B, C \rightarrow A, D \rightarrow B, E \rightarrow F)

तथा ये Relation 3NF नहीं हैं।



A \Rightarrow

NO



$$\frac{E \xrightarrow{NP} F}{\text{Transitive}}$$

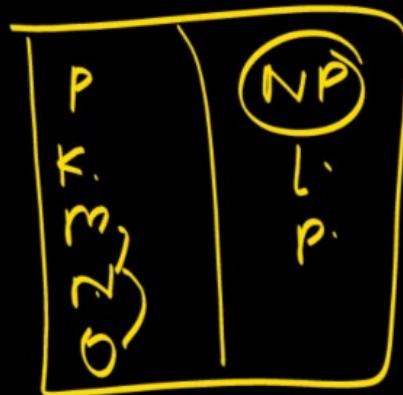
Q \Rightarrow Relation $R1 = (K, L, M, N, O, P)$

FD = $\{ K \rightarrow M, MN \rightarrow O, N \rightarrow K, P \rightarrow L \}$

CK = { K, M, ON }

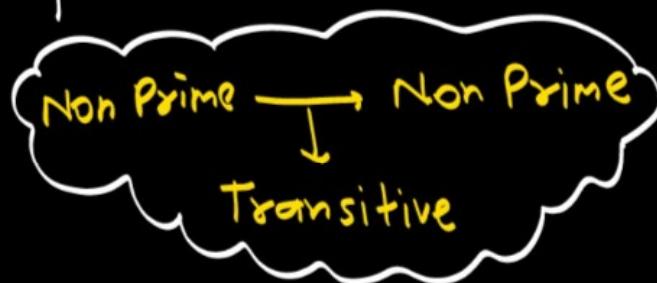
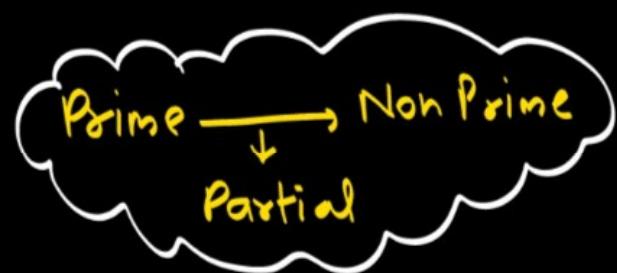
का ये Relation 3NF नहीं है।

A \Rightarrow $\text{NO} \checkmark$



3. Third Normal Form \Rightarrow

- Transitive Dependancy Based.
- Relation 2NF में होगा -पार्टिए |
- Relation में Transitive dependancy नहीं होनी चाहिए |
- जब कोई Non-prime Attribute किसी दूसरे Non-prime Attribute को determine करे, तो इसे Transitive Dependancy कहते हैं।



- यदि Relation में Transitive Dependancy है, तो वह Relation 3NF में नहीं होगा।

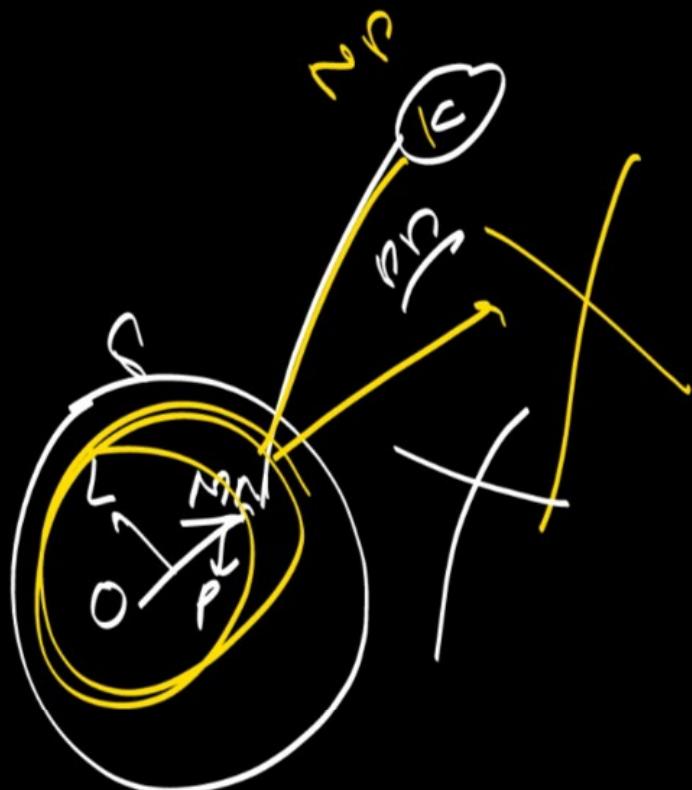
Q \Rightarrow Relation $R1 = (K, L, M, N, O, P)$

$FD = (MN \rightarrow P, P \rightarrow L, O \rightarrow N)$

$CK = (L, MN, O, P)$

इस Relation 2NF नहीं होगा।

A \Rightarrow YES



$\nexists \nexists FD = (N \rightarrow K)$

Q ⇒ Relation $R_1 = (A, B, C, D, E, F)$

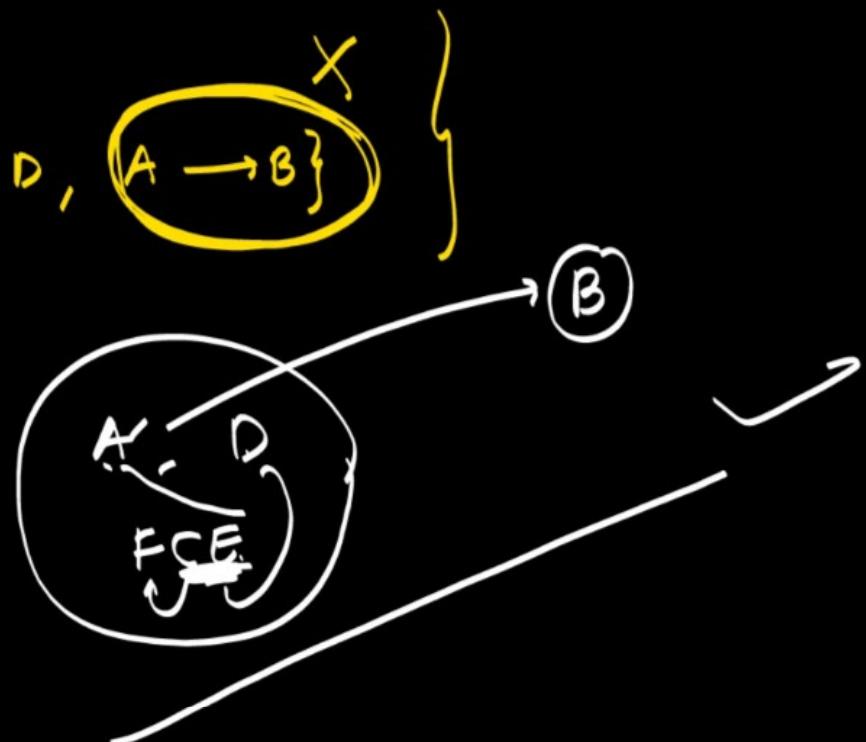
$$FD = \{C \rightarrow F, E \rightarrow A, EC \rightarrow D\}$$

$$CK = \{A, D, FCE\}$$

यहां से Relation 2NF में है।

A ⇒ **No**

$$\begin{array}{ccc} A & \rightarrow & B \\ \downarrow & & \downarrow \\ \text{Prime} & & \text{Non Prime} \end{array}$$



→ जब एक Prime Member, किसी NonPrime Member को denote करना है, तो इसे Partial Dependancy कहते हैं।

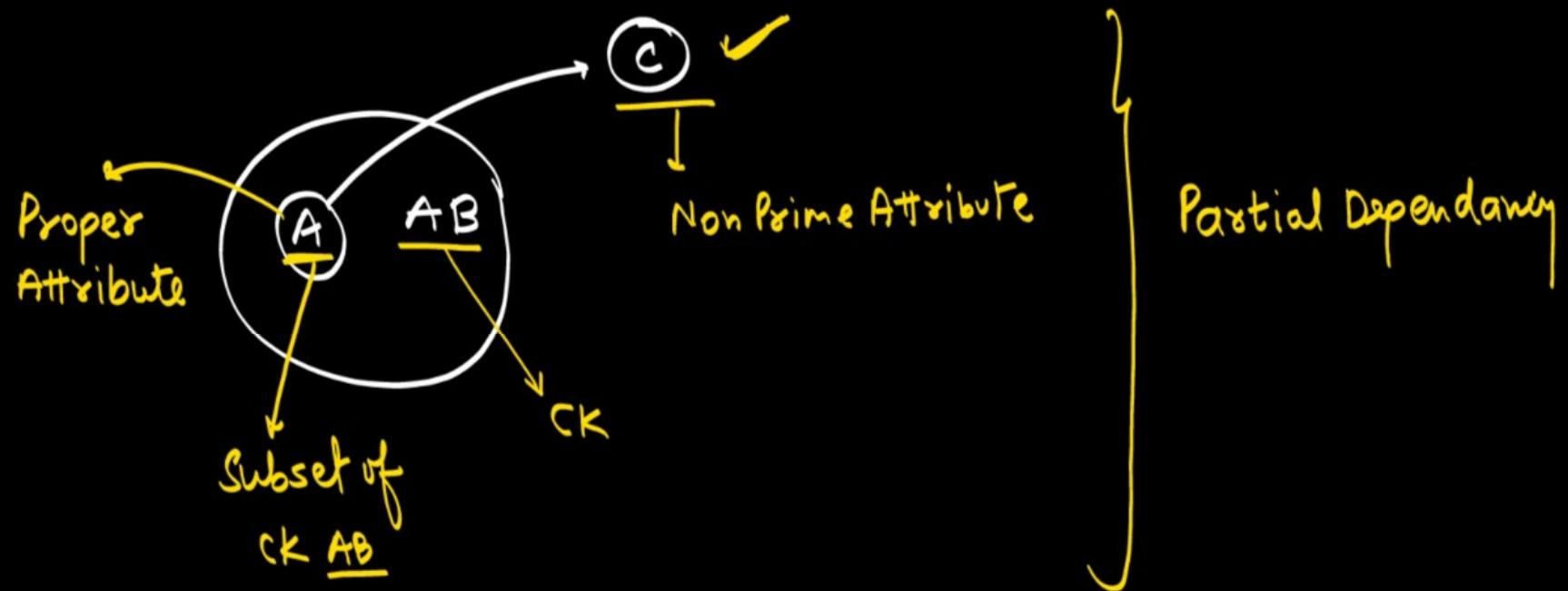
* Rules of 2NF →

→ Table हमेशा 1NF में होनी चाहिए।

→ Relation में कोई भी Partial Dependancy नहीं होनी चाहिए।

कुछ सभी Non Prime Attributes, Candidate Key पर Fully Functionally Depend होने चाहिए।

→ इस Table में Roll No + C-id Candidate Key है और Roll No , Candidate Key का Proper Subset है, जो एक Non-Prime Attribute Name को Determine करता है, इसे Partial Dependancy कहते हैं।



2. Second Normal Form \Rightarrow

→ Partial Dependancy नहीं होती है।

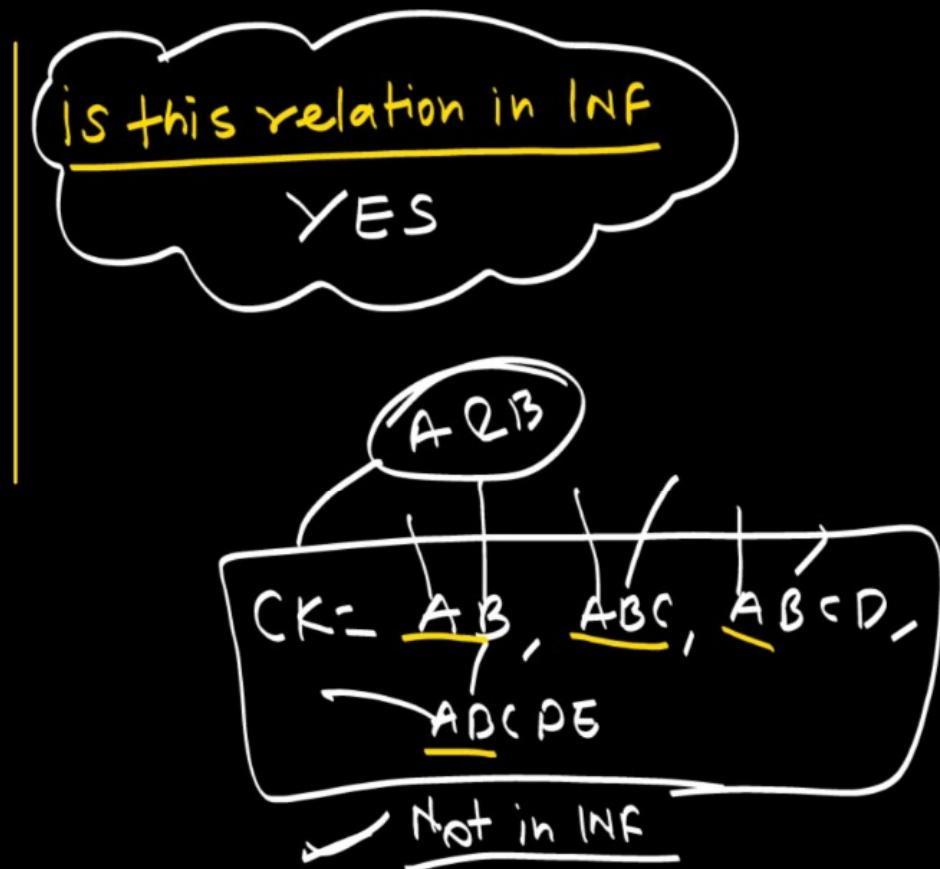
Roll No	Name	C-id
101	A	C
101	A	C++
101	A	Python
102	B	C++
102	B	Java
103	C	C
103	C	Python

Candidate Key = Roll No + C-id

Functional
Dependancy = Roll No \rightarrow Name ✓
Partial Dependancy
Subset
of
Candidate Key

Q Let A relation $Z_1(P, Q, R, S, T)$ & candidate keys are PR, PQ, RQ, ST, PST .
How many multivalued attributes are there?

- A. 0
- B. 1
- C. 2
- D. MOTA
- E. NOTA



Q3 Let A Relation P (U, V, W, X, Y), in which candidate keys are \underline{UV} , \underline{VWX} , \underline{VX} , \underline{VY} .
find out the following -

- A. is there any multivalued attribute = YES (V)
- B. is this relation in 1NF = NO
Because 'V' is
a multivalued attribute.

→ 1NF की Table में highest Degree of Redundancy होती है।

Que ⇒ यदि एक Relation A (P, Q, R, S, T) है, जिसके Candidate Key = $\underline{(P, PR, RST)}$
है। तो इस Relation 1NF में होगा?

Ans ⇒ 1NF में होगा।

* * * यदि सभी Candidate Keys में कोई Attribute common हो, तो वह Multivalued
Attribute होगा और Relation 1NF में नहीं होगा।

B. Repeat Columns ⇒

Roll No.	Name	Age	C-Name	C-Name	C-Name
101	A	15 8 47	C	C++	Java
102	B	26 1 50	C	Python	-
103	C	-	C++	Python	VB

C. Split Table ⇒

(PK) →

Roll No	Name	Age
101	A	15 8 47
102	B	26 1 50
103	C	-

(FK) ✓

Roll No.	C-Name
101	C
101	C++
101	Java
102	C
102	Python
103	C++
103	Python
103	VB

A. Repeat Roll No. =

Roll No.	Name	Age	C. Name
101	A	15 08 47	C
101	A	15 08 47	C++
101	A	15 08 47	Java
102	B	26 01 50	C
102	B	26 01 50	Python
103	C	-	C++
103	C	-	Python
103	C	-	VB

J. INF \Rightarrow

- First Normal Form
- RDBMS की guidelines के अनुसार प्रत्येक Table को INF में होना चाहिए।
- * → INF के अनुसार Table के प्रत्येक Attribute के लिए Single Value ही Allot की जाती है।

R-No	Name	DOB	C-Name
101	A	15/08/47	C, C++, Java
102	B	25/01/50	C, Python
103	C	-	C++, Python, VB

→ इस Table में C-Name में Multiple Values हैं, अब: इस INF में ले जाने के लिए 3 Methods Use में ले सकते हैं।

- A. Repeat Roll No.
- B. Repeat Columns
- C. split table

* Normal Form ⇒

- E-R Model का प्रयोग करते हुए जब Table create की जाती है, तो उसमें Multivalued attributes रह जाते हैं, जिससे Data Duplicacy/Redundancy आ जाती है।
- Normal form के द्वारा Table की Redundancy को दूर किया जाता है।
- Simplified form of Table is known as Normal Form.

* * Solution of Anomaly \Rightarrow

→ Decomposition of Table

→ Relation/Table में Anomaly होने से उसे multiple Table में Decompose कर दिया जाता है।

* Types of Normalization \Rightarrow

- 1. 1NF
- 2. 2NF
- 3. 3NF
- 4. BCNF

- 5. 4NF
- 6. 5NF
- 7. 6NF

B. Delete Anomalies ⇒

→ जब Database से किसी Record को Delete किया जाता है, तो उसके Related अन्य Data भी Delete हो जाता है, इसे Delete Anomaly कहा जाता है।

C. Update Anomalies ⇒

→ जब Database के किसी Column को Update किया जाता है, तो इसके अधिक Fields automatically update हो जाते हैं, इसे Update Anomaly कहते हैं।

S-id	Name	Address	C-id	C-Name	Teacher	Fees
S1	N1	Jalpa	C1	CN	T1	NULL
S2	N2	Jpr	NULL	C	T2	10k
S3	N3	Patna	C2	C++	T3	15k
S4	N4	Agra	C3	DS	T4	NULL
S4	N5	Ranchi	NULL	NULL	T3	NULL

A. Insert Anomalies

→ Given table में जब नया record insert किया जाता है, तो C-id, C-Name, Teacher, Fees में या नो NULL value या Dummy Values enter करनी पड़ती है, इससे DataBase में NULL Values / Dummy Values की संख्या बढ़ती है. इसे ही Insert anomaly कहा जाता है।

* Normalization ⇒

- इसके द्वारा हम Table में Data Duplicacy / Redundancy को Minimize करते हैं।
- जब Table में Data Redundancy होती है, तो इसके कारण Data Inconsistent हो जाता है, जिससे Table में 3 प्रकार की Anomalies / Problems Create होती हैं -

- A. Insert Anomalies
- B. Delete Anomalies
- C. Update Anomalies

13. Rule Twelve ⇒

- Non Subversion Rule
- इस Rule के अनुसार किसी Lower Level Language का प्रयोग करके , DataBase language के द्वारा लगाए गए Integrity Rules को Ignore नहीं किया जा सकता है।
- DataBase language के द्वारा लगाए गए Constraints को bypass नहीं किया जा सकता है।

12. Rule Eleven →

→ Distribution Independence Rule

→ इस Rule के अनुसार DataBase के विभिन्न locations पर Distributed Data के parts DataBase के users के Invisible होते चाहिए।

अर्थात् किसी भी DataBase का Data user को Seamlessly deliver होता चाहिए,
विना मह बनाये कि Data किसी servers पर Distributed हो।

11. Rule Ten \Rightarrow

पूर्ण (Complete)

→ Integrity Independence Rule.

→ इस Rule के प्रत्येक integrity constraints को Application Program से मालगा कर Catalogue में store किया जाना चाहिए।

उदास = Student Table

✓ Phone = Not Null

Record Phone = Not Null

9. Rule Eight ⇒

- Physical Data Independence Rule
- इस Rule के जब कोई storage/Disk change की जाती है, तो इसके Database की Tables & उनकी Relationship के मध्य कोई Effect नहीं पड़ता -जाहिर।

10. Rule Nine ⇒

- Logical Data Independence
- इस Rule के अनुसार जब Database के Logical Level पर कुछ Modification किए जाएं तो End user के View पर उसका Effect नहीं पड़ता है।
- Logical Data Independence प्राप्त करना is tough.

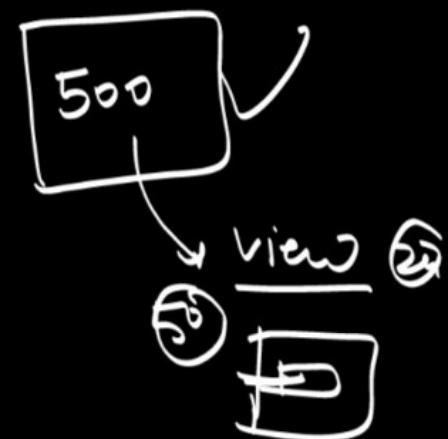
8. Rule Seven →

- High level Insert, Delete & Update Rule
- इस Rule के अनुसार Systems द्वारा Insert, Delete & Update Operators को एक साथ Set करने का Support किया जाना चाहिए।
- यह Rule कहता है कि Insert, Delete & Update का प्रयोग किसी एक Table के एक Row पर करने की बजाए Retrievable Set पर दोनों चाहिए।



6. Rule Five \Rightarrow

- Data SubLanguage Rule
- इस Rule के अनुसार System में से Data Access करने के लिए कम से कम एक Relational Language होनी चाहिए।
- जो Linear Syntax को Support करे।
- जैसे - Data Definition & Data Manipulation Language.



7. Rule Six \Rightarrow

- View Updating Rule
- इस Rule के अनुसार Views, System द्वारा भी update किये जाने चाहिए।

→ किसी user को Query का प्रयोग करके DataBase Catalogue / Structure को Access करने में सक्षम होता है। याहाँ, ताकि user Database के Data का प्रयोग कर सके।

4. Rule Three ⇒

- Systematic Treatment of NULL Value Rule
- इस Rule के अनुसार RDBMS के इत्येक Field को NULL / Empty रखने की Permission होनी चाहिए।
- NULL value का प्रयोग Data Missing, Data Unknown या Data Not Applicable के सन्दर्भ में किया जाता है।

5. Rule FOUR ⇒

- Active Online Catalogue Based on the Relational Model Rule
- इस विषय के अनुसार इत्येक System को एक Online Relational Catalogue, जो कि Authorised Users को उनकी Regular Query language के माध्यम से Accessible होना चाहिए।

2. Rule One ⇒

→ Information Rule

→ इस नियम के अनुसार किसी DataBase में All Information को Only & Only एक दी तरीके i.e. Table में Row & Column के रूप में स्थित किया जाना चाहिए।

3. Rule Two ⇒

→ Guaranteed Access Rule

→ किसी DataBase का पूरा Data Accessible होना चाहिए।

→ इस Rule के अनुसार प्रत्येक Table में एक Primary Key होनी चाहिए।

→ यह Rule कहता है कि Database के प्रत्येक प्रत्येक Individual Data के Primary Key के प्रयोग से logically पहुंचा जा सकता है।

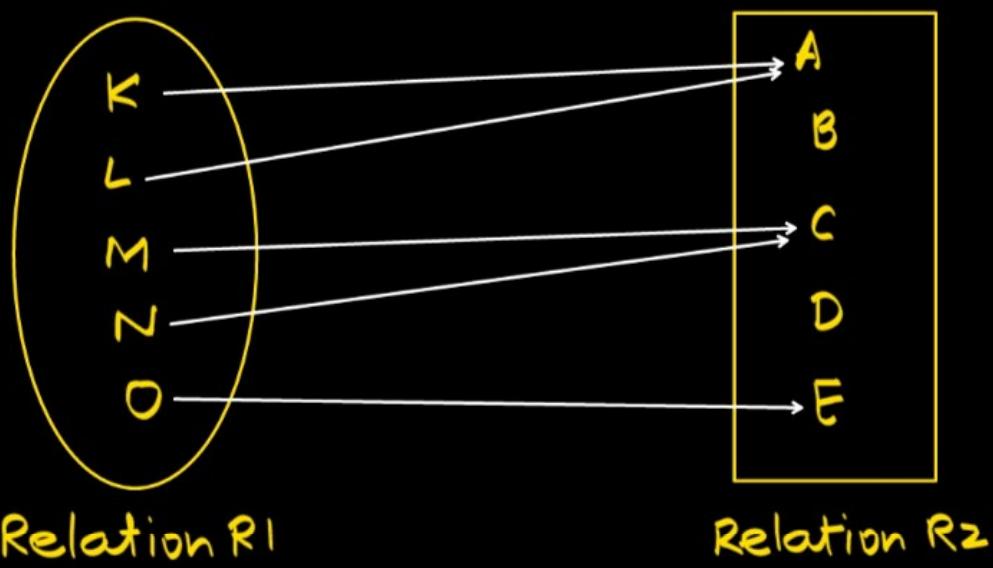
* Codd's Rules *

- Dr Edger Frank Codd
- Codd ने 0-12 (कुल 13) Rules दिये थे, जिन्हें Codd's 12 Rules कहते हैं
- यदि कोई भी DBMS इन Rules को follow करता है, तो उसे Relational DBMS कहते हैं।

1. Rule Zero ⇒ (Foundation Rule / Base Rule)

- किसी System को Relation के रूप में, एक DataBase के रूप में व एक Management System के रूप में Qualify करना पड़ता है।
- किसी System को अपनी Relational Facilities का प्रयोग करके RDBMS के लिए Qualify करना पड़ता है।

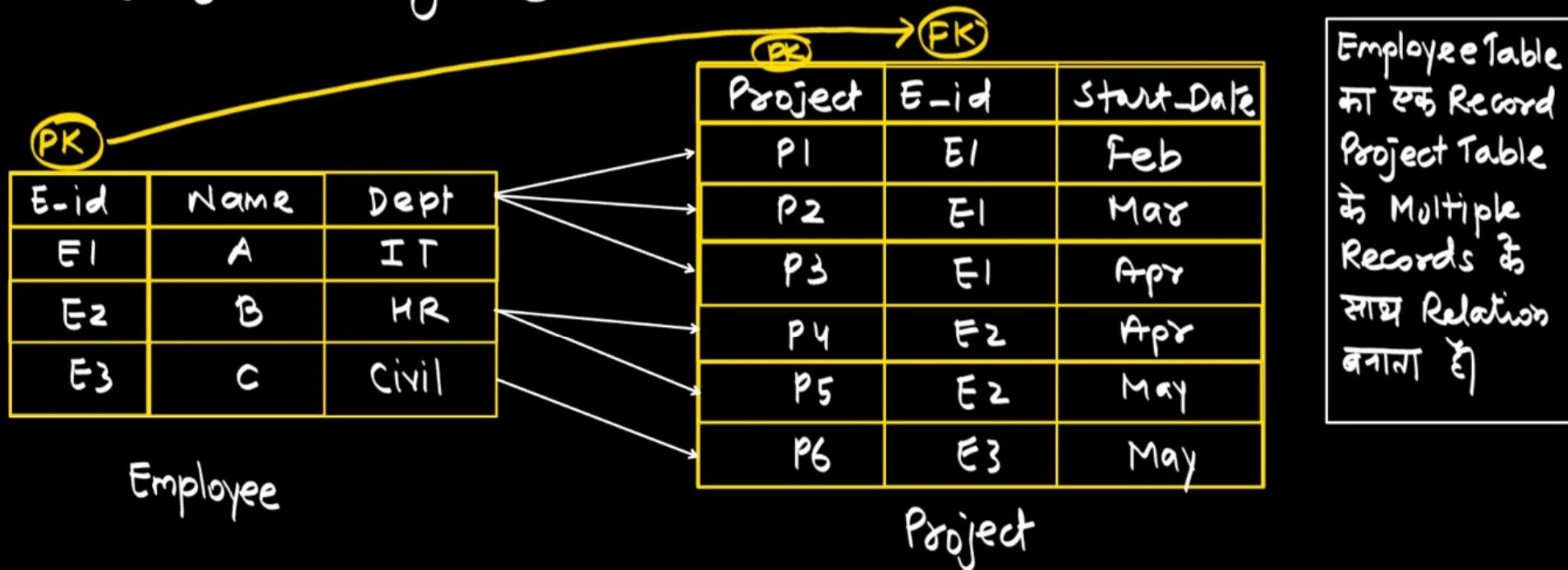
3. Many To One Relationship \Rightarrow



→ इस Relationship में Relation R1 के एक से अधिक Records, Relation R2 के केवल एक Record से Relation बनाते हैं।

2. One to Many Relationship →

→ इस प्रकार की Relationship में एको बाली Relationship Table में दोनों Tables की Primary Key को Foreign key के रूप में इस्तोग में लिया जाना है।



Employee Table का एक Record Project Table के Multiple Records के साथ Relation बनाना है।

- One to One Relationship ਵਿੱਚ ਹਮ ਦੋ ਕਮ 2 tables ਬਣਾਉ ਜਾ ਸਕਣੀ ਹੈ
- Max = No Limit

* Mapping Cardinality / Degree of Relationship ⇒

A. One To One ⇒

- इस Relationship में Relationship की Table में Minimum 2 Attributes होते हैं, जो दोनों Tables की Primary key होती हैं।
- Relationship Table में किसी भी Attribute को Primary key बनाया जा सकता है।

(PK)

E-id	Name	Age
E1	N1	22
E2	N2	20
E3	N3	18
E4	N4	24
E5	N5	27

EMPLOYEE

E-id	D-id
E1	D1
E2	D2
E3	D4
E4	D5
E5	D6

WORK

(PK)

D-id	D.Name	Location
D1	DN1	L1
D2	DN2	L2
D3	DN3	L3
D4	DN4	L4
D5	DN5	L5
D6	DN6	L6

DEPARTMENT

C. Update \Rightarrow May Cause Violance

→ Referencing Table में किसी भी Record को update करने पर Problems हो सकती हैं, जब वह Updated Record Referenced Table में नहीं हो।

6. Composite Key \Rightarrow

→ जब Table के Records को Uniquely Identify करने के लिए एक से अधिक Columns का use करा जाता है, तो इसे Composite Key कहते हैं।

* Referencing Table \Rightarrow
Child Table

A. Insert \Rightarrow May cause violation

\hookrightarrow Solution \Rightarrow Referencing Table में कोई भी Record केवल नए ही insert होगा,
जब Referenced Table में वह same field हो।

B. Delete \Rightarrow No violation

\hookrightarrow Referencing Table में से किसी भी column को Delete किया जा सकता है।

b. On Delete Cascade ⇒ Referenced Table में से कोई भी Record Delete करने पर Referencing Table में से भी Data Automatically Delete हो जाएगा।

c. On delete No action ⇒ Referenced Table से कोई भी Record delete होने पर Referencing Table पर कोई Effect नहीं पड़ेगा।

C. Update ⇒ May Cause Violence

 └→ Solutions⇒

- a. On Update Set Null
- b. On Update Cascade
- c. On update No Action

* Referenced Table \Rightarrow
Parent Table

A. Insert \Rightarrow No violence

\Rightarrow Referenced Table में नया Record insert करने पर कोई Problem नहीं होती है।

B. Delete \Rightarrow May cause violence

\hookrightarrow Solutions \Rightarrow

a. On Delete set NULL \Rightarrow Referenced Table में से किसी Record को Delete होने पर Referencing Table के Column में NULL Value set हो जाती है।

* Referential Integrity & Constraints ⇒

Student

S-id	Name	Class
101	Ram	X
102	Raxi	XI

Referenced Table

Course

C-id	C-Name	S-id
1001	Java	101
1002	C++	101

Referencing Table

⇒ Primary Key वाली Table को Referenced Table कहते हैं तथा यह Primary Key जिस Table में Foreign Key के रूप में प्रयोग में भी जाती है, उसे Referencing Table कहते हैं।

Student

S-id	Name	Phone
101	Ram	54321
102	Ravi	42351

↑
Primary key
(Unique & NotNULL)

Course

C-id	C-Name	C-Expiry	S-id
1001	C	12/2025	101
1002	C++	12/2025	101
1003	C#	12/2025	—
1004	Java	12/2025	102

] ✓ Repeat
→ NULL

↑
Foreign key
(Not Unique & Null)

5. Foreign Key / Reference Key ⇒

- किसी एक Table की Primary key किसी अन्य Table में यदि एक column के स्पैस में प्रयोग में ली जाती हो, तो इसे Foreign key कहते हैं।
- इसे Reference key भी कहते हैं, क्योंकि यह 2 Tables के बीच Relation Generate करती है।
- Foreign Key की Value Not Unique & NULL हो सकती है।
- Primary Key पर Uniqueness Constraint लगता है जबकि Foreign Key पर नहीं।
Inputted Value हमेशा
unique होती।

Q यदि एक Relation R₁ (A, B, C, D) में AB एक Candidate Key हो, तो Super Keys की संख्या ज्ञात कीजिए ?

A \Rightarrow No. of Super Keys = 2^{n-2} (n=4)

$$= 2^{4-2}$$

$$= 2^2 = \boxed{4 \text{ Super Keys}}$$

Q यदि एक Relation R₁ (A, B, C, D) में ABCD एक Candidate Key हो, तो Super Keys की संख्या ज्ञात कीजिए।

Ans \Rightarrow } क्योंकि कोई भी प्रत्येक Attribute बना ही नहीं है,
जिसके साथ Primary key बनाया जा सके।

$$\begin{aligned} &= 2^{n-n} \quad (n=4) \\ &= 2^{4-4} \\ &= 2^0 \\ &= \boxed{1} \end{aligned}$$

Q ⇒ If A Relation R₁ (A, B, C, D) has a candidate key 'A', Then how many Super keys are there -

A ⇒ No. of super keys = 2^{n-1} (n=4)
= 2^{4-1}
= 2^3 = 8 Super Keys

Q ⇒ यदि एक Relation R₁ (A, B, C, D) में A तथा B Candidate Key हैं, तो कुल Superkeys की संख्या ज्ञात कीजिए ?

A ⇒ No. of Super Keys = $2^{n-1} + 2^{n-1} - 2^{n-2}$ (n=4)
= $2^{4-1} + 2^{4-1} - 2^{4-2}$
= $2^3 + 2^3 - 2^2$
= $8 + 8 - 4$ = 12 Super Keys

* * MASTER CAPSULE ⇒

1. If candidate is one = Number of Super keys
 A



$$2^{n-1} \quad n = \text{Number of Attributes in a relation}$$

2. if candidate keys are Two = Number of Super Keys
 A, B

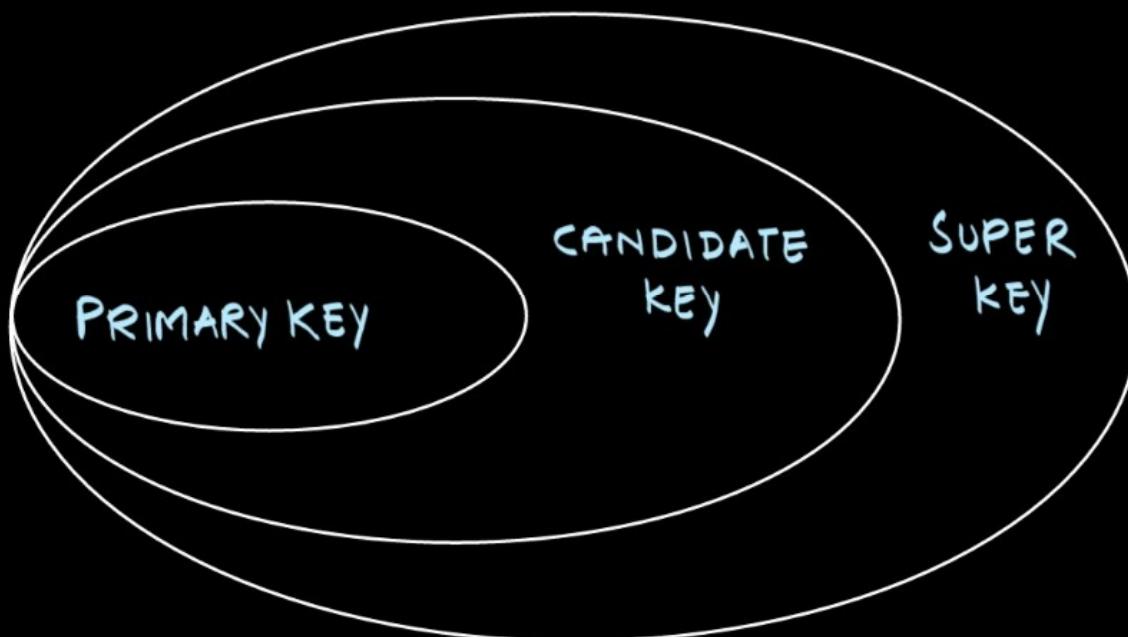


$$2^{n-1} + 2^{n-1} - 2^{n-2}$$

3. if candidate Key is AB = Number of Super Keys

$$\downarrow \\ 2^{n-2}$$

- एक Super Key में कम से कम एक Attribute Candidate Key का होना चाहिए, लेकिन शारे दी Attributes Candidate Key के हो, पर जरूरी नहीं हैं।
- * → किसी भी Table में Minimum Super keys को Candidate Key कहते हैं।



4. Super Key \Rightarrow

→ A set of attributes जो किसी Table के एक Tuple को uniquely identify करते हैं,
उन्हें Super Key कहते हैं।

S-id	Name	F-Name	Class

} Relation Student (S-id, Name, F-Name, Class)



S-id, Name
Name, F-Name
S-id, Class } SUPER KEYS

→ A candidate key is a superkey, but A superkey is not a candidate key always.

Codd अर्फ्या

1st Rule = Data Must be In Row & Columns

2nd Rule = Keys & Their Uses

3rd Rule = प्रत्येक Table में कम से कम एक Candidate Key ऐसी होनी चाहिए,
जिसकी Values unique & Not NULL होनी चाहिए।

* Candidate Key (A, BC, E)

A, E = Simple Candidate Key

BC = Composite Candidate Key

- } → ऐसी Candidate Key जो Single Attribute से बनती है - SIMPLE CANDIDATE KEY
- ऐसी Candidate Key जो Multiple Attributes से मिलकर बनती है - COMPOSITE CANDIDATE KEY

* Prime / Key Attributes \Rightarrow

Relation $R_1 (A, B, C, D, E)$
 \downarrow
Candidate Key (A, BC, E)
 \downarrow
Prime Attributes (A, B, C, E)

ऐसे Attributes जो किसी Table की Candidate Key से Belong करते हैं,
उन्हें Prime/Key Attributes कहते हैं

* Non Prime/Key attributes \Rightarrow

→ ऐसे Attributes जो किसी Table की Candidate Key से Belong नहीं करते हैं,
उन्हें Non-Prime/Key Attributes कहते हैं
जैसे - Relation R_1 में D Non Prime Attribute है।

- Primary key हमेशा एक candidate key होनी चाहिए।
- एक Table में Primary key जैवल एक ही होती।
- Given table में S-id Primary key है।

3. Alternate Key ⇒

- किसी Table में Primary key को छोड़कर अच्छी तरह Keys Alternate Keys कहलाती हैं।
- Alternate Keys कास्तब में Candidate key ही होती है, लेकिन उसकी Field Value NULL हो सकती है।
- इसे Secondary key भी कहते हैं।
- Given Table में S-id के अलावा सब Alternate Keys कहलाती हैं।

→ Candidate Key होशा Not NULL हो, यदि Possible नहीं हो, तो इसे Blank या NULL रखा जा सकता है।

2. Primary Key ⇒

→ Attribute value must be unique & Not NULL.

S-id	Name	Class	Phone

} ⇒ Student(S-id, Name, Class, Phone)

S-id → Candidate Key
Name → Primary Key
Class → Alternate Key

→ Primary Key एवं Candidate Key ही होती हैं, जिसकी Value होशा Unique & Not NULL होती है।

1. Candidate Key

→ The minimum set of attributes that can uniquely identify a tuple/record is known as candidate key.

Ex ⇒

S_id	Name	Add.

} S_id एक Record को Uniquely Identify कर सकती है, इसलिए यह Candidate Key बन सकती है।

→ किसी Table में Multiple Unique Keys हो सकती है, जिनमें से एक Primary Key Select होती है तभा अन्य Candidate Keys वर्ती हैं।

→ Candidate Key की Value हमेशा Unique & Not NULL होती।

→ Candidate Key can be Simple or Composite.

* Keys in DBMS =>

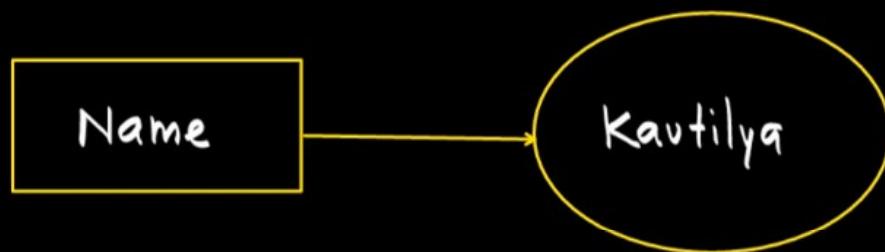
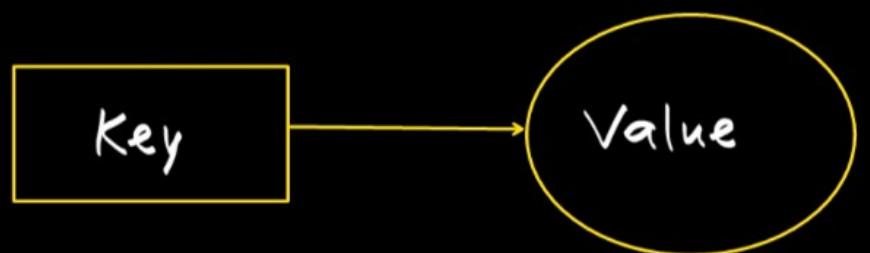
- Key किसी Table का एक particular Column होता है, जो Table के 2 Records / Tuples को Uniquely Identify करता है
- Dr. E F Codd ने Rule No. - 2 में Keys & उनके Uses को Define किया है।
- Codd's Rule 2 ⇒ किसी Table के 2 Records same नहीं हो सकते हैं
- Key किसी Table के Single या multiple Attributes को मिलकर बनती है।

* Types of Keys =>

- | | |
|------------------|----------------------------------|
| 1. Candidate Key | 4. Super Key |
| 2. Primary Key | 5. Foreign Key / Referential Key |
| 3. Alternate Key | 6. Composite Key |

7. Key - Value Data Model →

→ NOSQL → use



Key → Value

→ Simple & Fast

5. Object Oriented Data Model ↗

- इस Data Model में Data/Information को object के रूप में store किया जाता है।
- में objects Instance variable में value को store करते हैं।
- High level languages में use.
- Complex Datatype & Inheritance को support करता है।

6. Document Data Model ↗

- Superset of other Data Models
- NoSQL में use.
- Programming Languages में object को Map करने में use.

6. Required Attribute

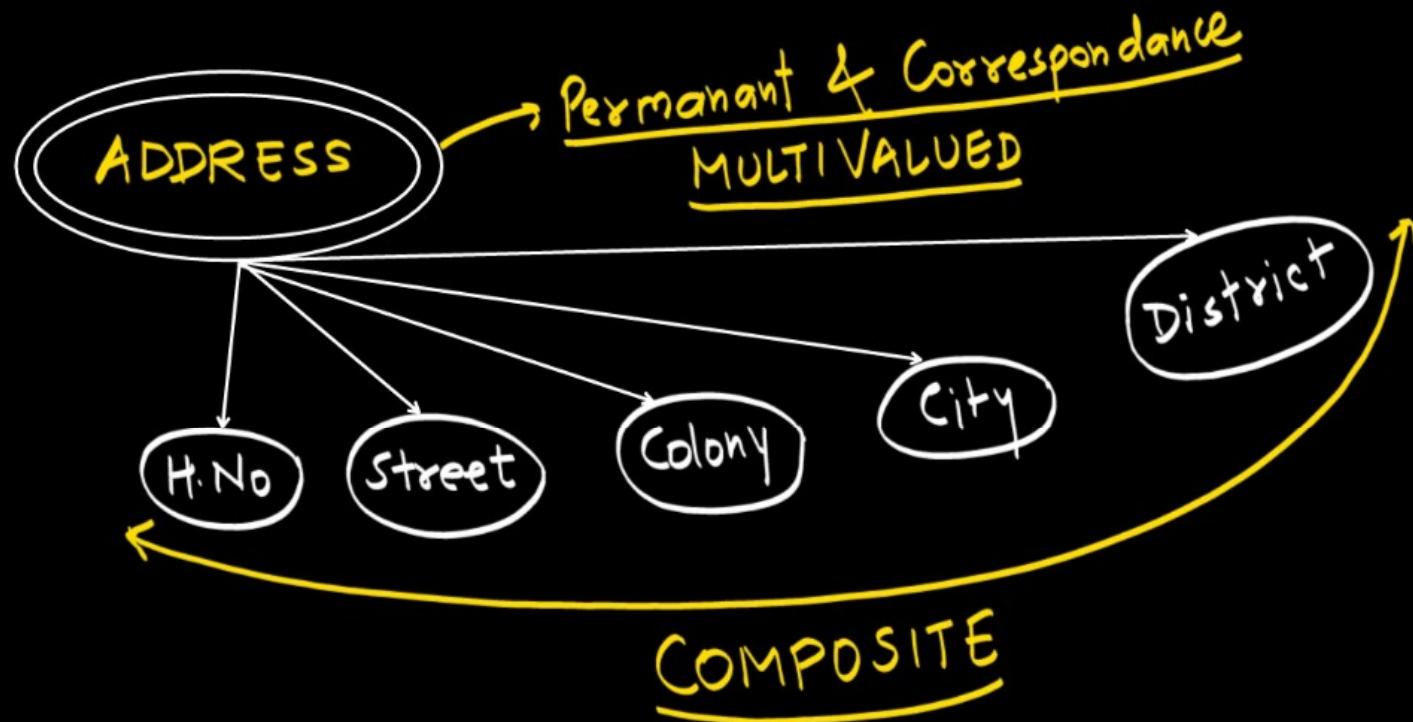
- किसी Table या User Application को Complete करने के लिए Compulsory Attribute को Required Attribute कहते हैं।
- It can't be NULL.

7. Optional Attribute

- ऐसा Attribute जिसे fill करना User पर निर्भर करता है, जिसके दोनों ओर न होने से Database पर कोई Effect नहीं पड़ता है।

5. Complex Attribute \Rightarrow

→ Multivalued + Composite



4. Key & Non-Key Attributes →

- ऐसा Attribute जिसका प्रयोग करके Table के Records को Uniquely Identify किया जा सकता है, उसे Key Attribute कहते हैं (Primary Key)
जैसे - Roll No.
- ऐसा Attribute जिसका प्रयोग करके Table के Records को Uniquely Identify नहीं किया जा सकता है, उसे Non Key Attribute कहते हैं
जैसे - Name, DOB.
- Table का Key Attribute हमेशा Unique & Not NULL होता है
↓
Not Empty

3. Stored & Derived Attribute

→ ऐसा Attribute जो Table में दिया जाता है, उसे Stored Attribute कहा जाता है।

Whereas

ऐसा Attribute जिसे Table के Multiple Attributes का प्रयोग कर जात किया जाता है, उसे Derived Attribute कहते हैं।

→ Derived Attribute is not a part of the table.

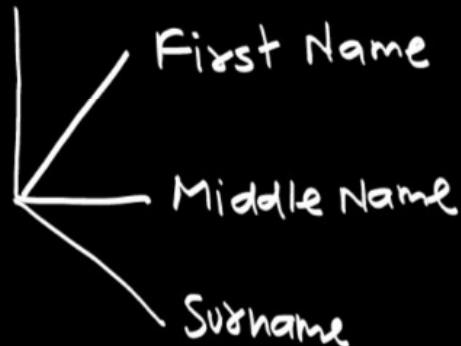
Id	Name	DOB

DOB → Stored Attribute, Single / Simple Attribute

Age → Derived Attribute

2. Simple & Composite Attribute

- ऐसा Attribute जो Single Value Carry करता है, उसे Simple Attribute कहते हैं।
जैसे - Registration No.
- ऐसा Attribute जो Multiple Values से मिलकर बनता है, उसे Composite Attribute कहते हैं।
जैसे - Name



* Types of Attributes ⇒

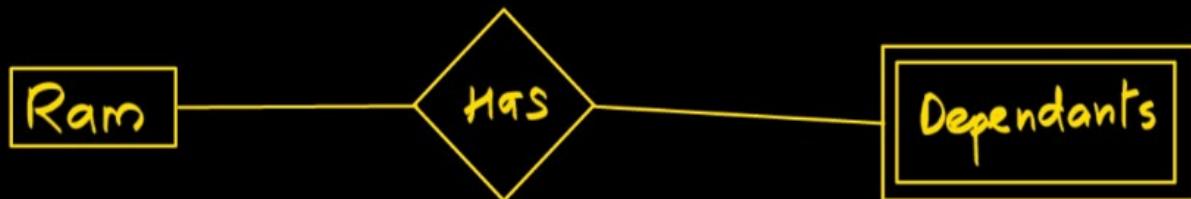
1. Single & Multivalued Attribute ⇒

→ ये सा Attribute जो Single Value Carry करता है, उसे Single Valued Attribute कहा जाता है।

जैसे - Roll No.

→ ये सा Attribute जो Multiple Values Carry करता है, उसे MultiValued Attribute कहा जाता है।

जैसे - Mobile.



$\frac{9\text{pm}}{\text{Yr}}$

$\frac{\text{EMRS}}{38000}$

CS < PGT
TGT

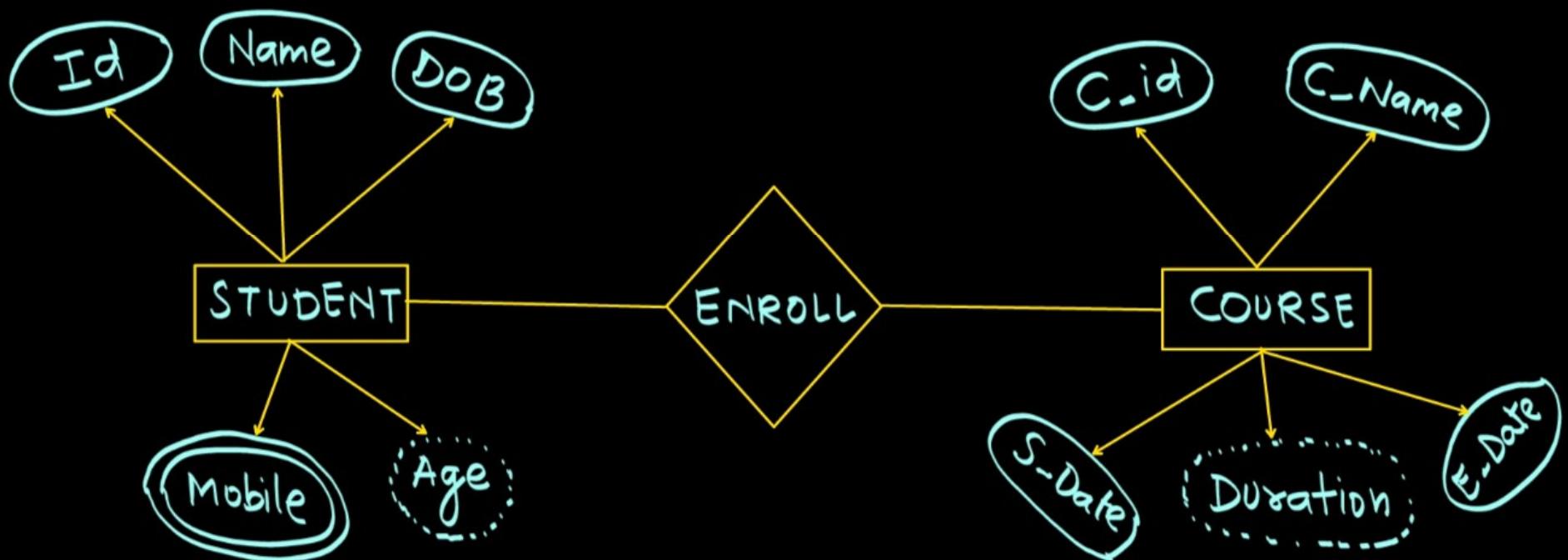
- पहले Student & Course 2 Entities हैं, जो DataBase Design के सभी Table का नाम बनती हैं।
- Student & Course के बीच Relationship = Enroll है।
- Id, Name, DOB, Mobile उन्हीं student Entity /Table के Attributes हैं, जो DataBase Design के सभी Table के Columns बनते हैं।
- Mobile एक Multivalued Attribute है, जिसके DataBase Design के सभी Table में इसके लिए स्पेशल Columns बनते हैं।
- Age & Duration Derived Attribute है, जिसके लिए एक या अधिक Columns का use करके बनाया जाता है।
- Derived Attribute DataBase Design के सभी Table का Column नहीं बनता है।
- ऐसी Entity जो खुद के Attributes द्वारा Uniquely identify नहीं की जा सकती है, उसे Weak Entity कहते हैं।

STUDENT

ID	Name	DOB	M1	M2

COURSE

C-id	C-Name	S-Date	E-Date



5.

Double Ellipse



6.

Dashed Ellipse



7.

Double Rectangle



8.

Ellipse with Line



9.

Double Diamond



Multivalued Attribute

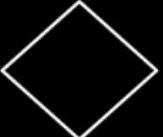
Derived Attribute

Weak Entity

Key Attribute

Weak Entity Relationship

* Symbols used in E-R Data Model →

S.No.	Name	Symbol	Work
1.	Rectangle		Entity
2.	Ellipse (दीर्घवृत)		Attributes
3.	Diamond		Relationship
4.	Lines	 	Entity to Relationship Entity to Attributes

B. Relation \Rightarrow

\rightarrow 2 Entities के सद्वय Connection.

C. Attributes \Rightarrow

\rightarrow किसी Entity की characteristics (विशेषताएँ)

* Data Dictionary \Rightarrow

- Data about Data
- METADATA
- Data के बारे में Additional Information
- इसमें storage & Physical Details Store की जाही है।
- Managed by DBMS.
- यह बताता है कि Tables कितने हैं, Columns कौन-कौनसे हैं, Data क्या है व उनके मध्य Relationship क्या है।
- Data Dictionary को Database के Tablespace में रखा जाता है
 - ↓
 - जहाँ सभी Tables Store होती है।
 - Database का Internal Storage

* → Logical representation of Data is called Schema.

→ RDBMS में एक Table = Student (Id, Name, Class, Phone) ↳
Relation

Id	Name	Class	Phone
}			

SCHEMA

→ किसी Database का overall logical Representation Schema कहलाता है।

→ Schema में Tables, Primary Key, Foreign Key, Views को display किया जाता है।

* → Database के किसी विशेष समय पर Available Data के Screenshot / Snapshot को INSTANCE कहते हैं।

↓
Schema at a Particular Time.

4. E-R Data Model

→ Entity - Relationship Model

A. Entity →

→ Any Object that have physical appearance or exist physically is called Entity.

उदाहरण - Student Table में

Roll No., Name, FName, MName

Entity SET

→ इस Data Model में DataBase को graphically represent किया जाता है, जिससे DataBase का Complete Logical Structure Display होता है।

→ इस Model के द्वारा DataBase के Schema को Theoretically Represent किया जाता है।

Student Table
(RELATION)

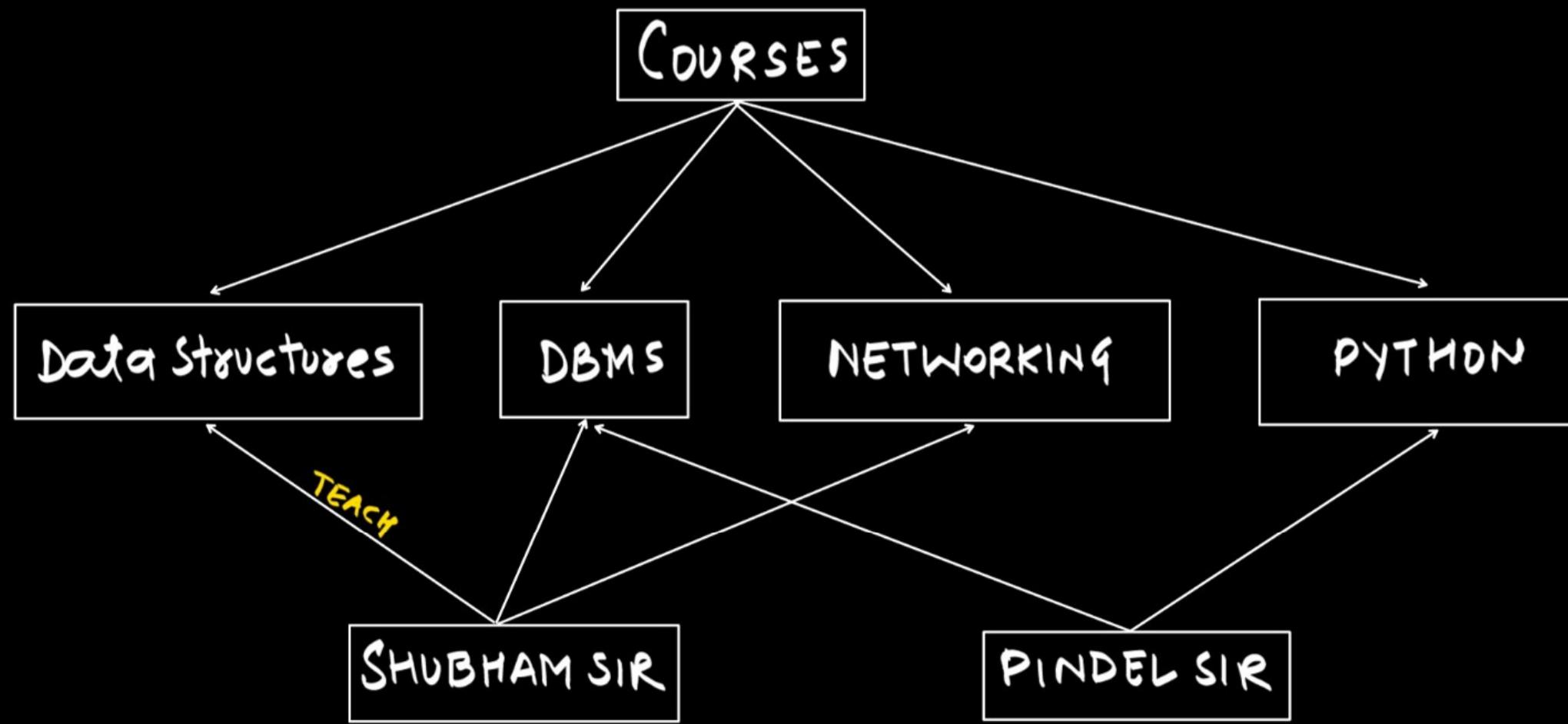
↓
Column
Domain
Attribute

→
Row
Tupple
Record

<u>ID</u>	<u>Name</u>	<u>Class</u>	<u>Phone</u>
1001	Shweta	10	9875643210
1002	Swati	12	9785643201
1003	Suchita	11	9576854321
1004	Sneha	9	9213456708

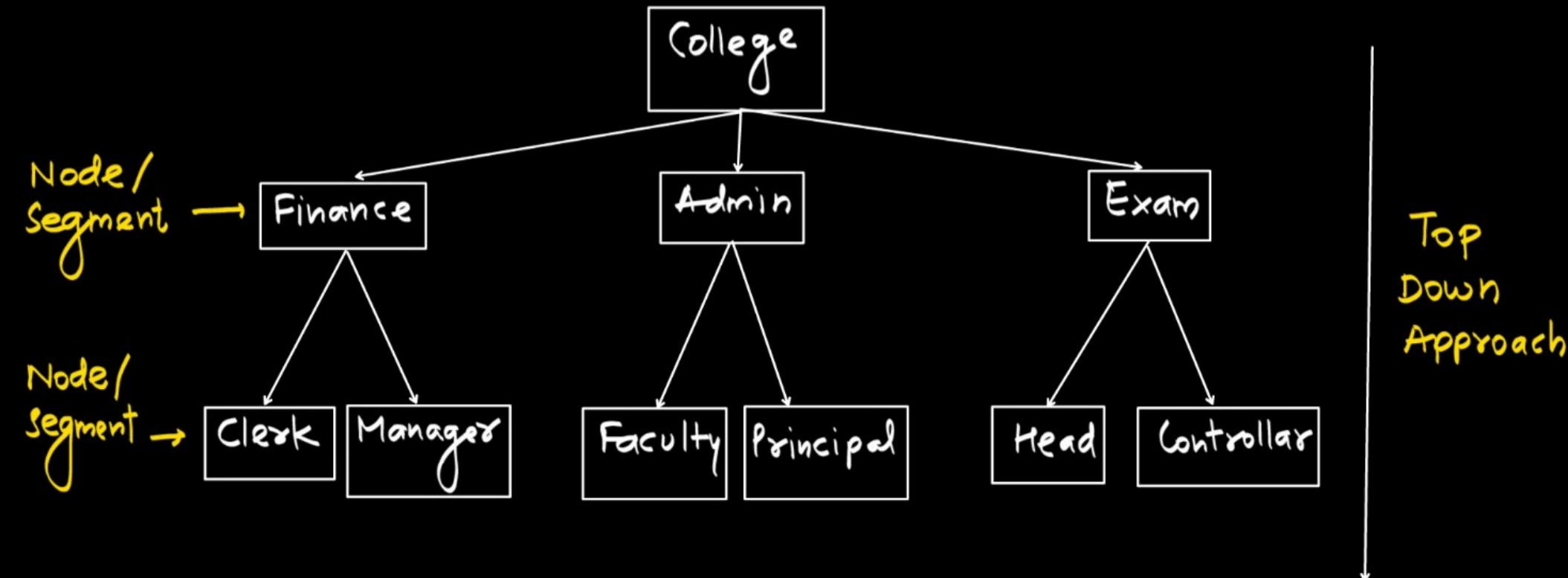
3. Relational Data Model ⇒

- सबसे Best Data Model क्योंकि इसमें Data को Table के format में Represent किया जाता है।
- Table = Row & Column से बनती है।
- Table is known as Relation.
- Table में Column / Attribute / Field / Domain एक ही Entity को Represent करते हैं।
- Table में Present Records की Entry एक से अधिक बार नहीं दोनी है।
- Table में Row / Record / Tuple एक ही Entity को Represent करते हैं।
- यह Data Model E.F.CODD का 1st Rule follow करता है अर्थात् Data एक Table में Row तथा Column के format में arranged होना चाहिए।
- E.F.CODD ने 12 Rules दिये थे।



2. Network Data Model ⇒

- Hierarchical Data Model का Extension.
- Parent - Child Relationship
- इसमें एक child के Multiple Parents हो सकते हैं।
- इस Data Model में Relationship को represent करने के लिए Graph Data Structure का प्रयोग किया जाता है।
- Many to Many Relationship
- इसमें Data Element की Nodes तथा Relationship को Edges द्वारा Represent किया जाता है।



I. Hierarchical Data Model :-

- Developed By = IBM
1968
- Developed for = IMS (Information Management System)
- इसमें Parent-Child / One to Many Relationship बनता है, इसलिए इसे Tree Data Model भी कहते हैं।
- इस Model में किसी Parent का एक child दुसरे child की Information को नहीं देख सकता है, इसलिए इसे Single Parent Model भी कहते हैं।
- इसमें प्रत्येक Record को Node/Segment कहा जाता है।
- Top-Down Approach

* Data Models \Rightarrow

- Data को Represent करने के लिए प्रोग्राम में लिए जाने वाले structures को Data Model कहते हैं।
- प्रकार :-

1. Hierarchical Data Model
2. Network Data Model
3. Relational Data Model
4. E-R Data Model
5. Object Oriented Data Model
6. Document Data Model
7. Key Value Data Model

COLMNS / ATTRIBUTES / DOMAIN

The diagram illustrates a database table structure with the following components:

- Columns / Attributes / Domain:** An arrow points from the top right towards the table headers.
- Rows / Tuple:** An arrow points from the middle right towards the table rows.
- FILE TABLE:** A bracket on the right side groups the three rows under this label.
- RECORD-1:** An arrow points to the first row (Id 101).
- RECORD-2:** An arrow points to the second row (Id 102).
- RECORD-3:** An arrow points to the third row (Id 103).
- Field:** Two arrows point to the first two cells of the first row, labeled "Field".
- Field value:** Two arrows point to the first two cells of the first row, labeled "Field value".

Id	Name	Address
101	RAM	JODHPUR
102	SHYAM	JAIPUR
103	HARI	KANPUR

* Representation/Hierarchy of Data ⇒

1. Bit ⇒ Smallest Unit to represent the Data. (0/1)

2. Character ⇒ Collection of Bits. (Byte)

3. Field ⇒ Collection of Characters

Ex - Shubham Swarnkar

whoshubhamSir

4. Record ⇒ Collection of Fields.

5. File ⇒ Collection of Records.

6. DataBase ⇒ Collection of Files /Tables related with each other.

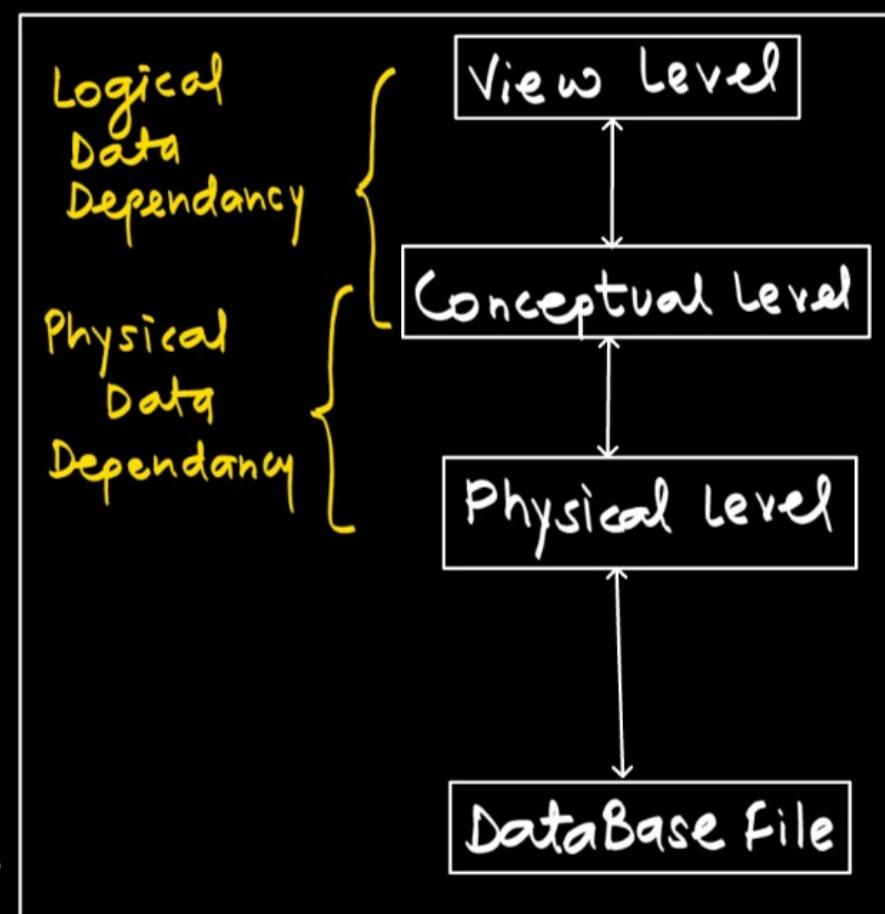
* Data Independency ⇒

A. logical Data Dependancy (Table) ⇒

→ Table में किसी नये column को Add करने, Tables के बीच Relationship को change करने से End user के Interface पर कोई effect नहीं पड़ता है, इसे Logical Data Dependancy कहते हैं।

B. Physical Data Dependancy ⇒

→ जब हम DataBase में कोई नया storage/Disk add करते हैं या किसी Disk की Location Change करते हैं, तो DataBase की Tables तथा उनके मध्य Relationship पर कोई असर नहीं पड़ता है, इसे Physical Data Dependancy कहा जाता है।



Shots

Physical Schema = Deals in storage of Database

Conceptual Schema = Deals in Tables & Their Relationships

External Schema = Deals in Requirements of the user

CRUX ✓

→ जॉसे - DataBase का Logical Structure (Tables, Columns), Data Models, Relationship, Rules & constraints etc.

3. Physical Schema / Internal Schema / physical Level ↗

- DataBase की Actual DataBase file के storage को इस Schema के द्वारा display किया जाता है।
- इसे DBA (DataBase Administrator) द्वारा Manage किया जाता है।
- इस Schema द्वारा हम Actual Disk पर कार्य करते हैं अर्थात् Database में present Disk को इस Schema के द्वारा ही Manage किया जाता है।
- इस Schema द्वारा ही Data Dictionary को maintain किया जाता है।
- यही Schema DataBase को Control करता है।

1. External Schema / View Level ↗

- इस Schema में users Connected रहते हैं।
- प्रत्येक User के लिए उसकी Need के अनुसार अलग Schema होता है।
- External Schema, Conceptual & Physical Schema से Hidden होता है, अर्थात् प्रद
User को display नहीं किया जाता है।
- User का Data कौनसी Table में stored है, उस Table की अन्य Tables के साथ
Relationship करता है तथा Data कौनसी Disk में है, पर user से hidden रखता है।

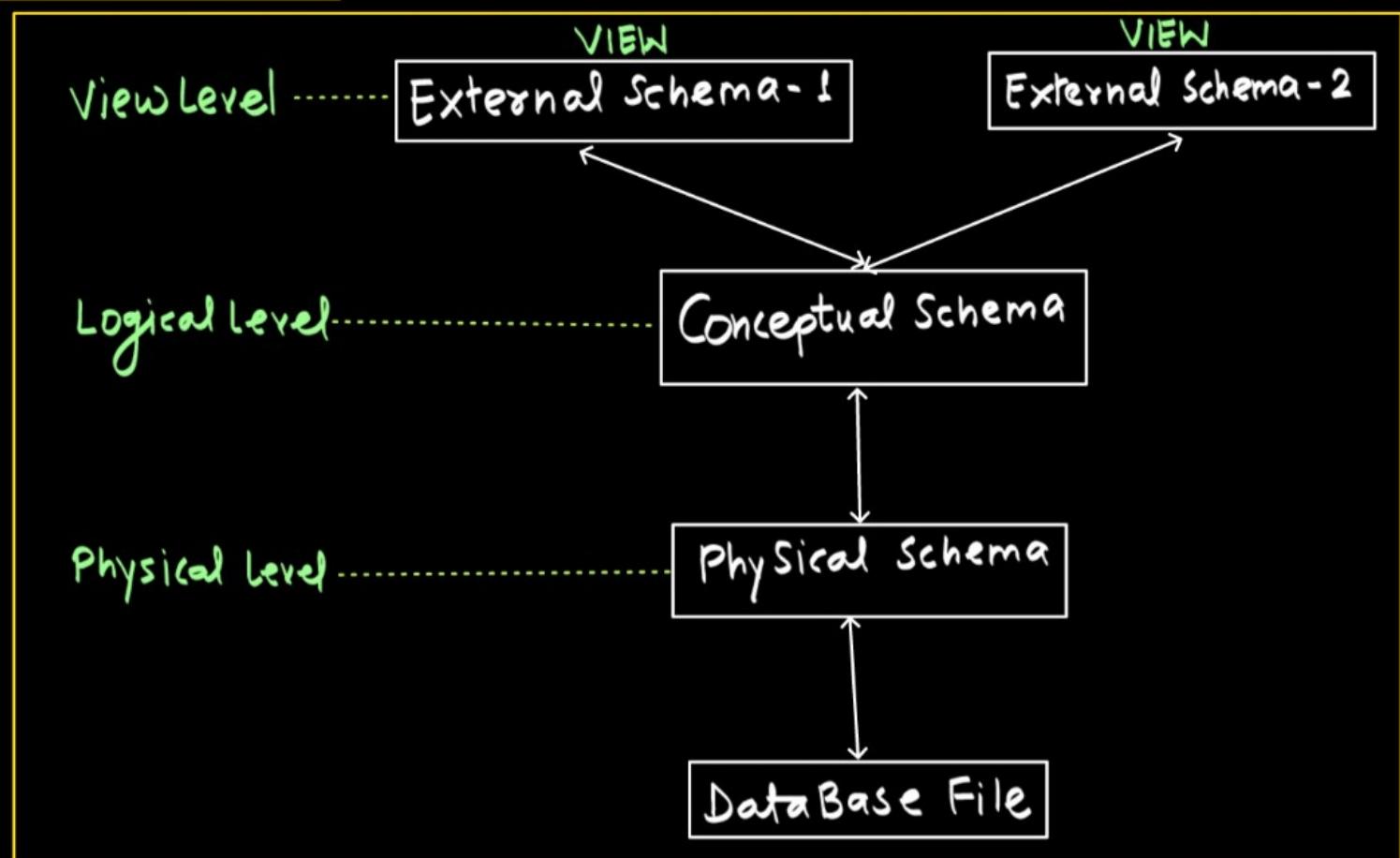
2. Conceptual Schema / Logical Level ↗

- इसमें प्रद बताया जाता है कि DataBase में कौन-कौनसा Data stored है।
- इस Schema के द्वारा ही DataBase को Design किया जाता है।
- इसमें Data के आधार पर Tables Create की जाती है तथा Tables के बीच Relationship
को define / create किया जाता है।

* * 3 Level Abstraction / Abstraction Levels / DataBase Abstractions / 3 Schema

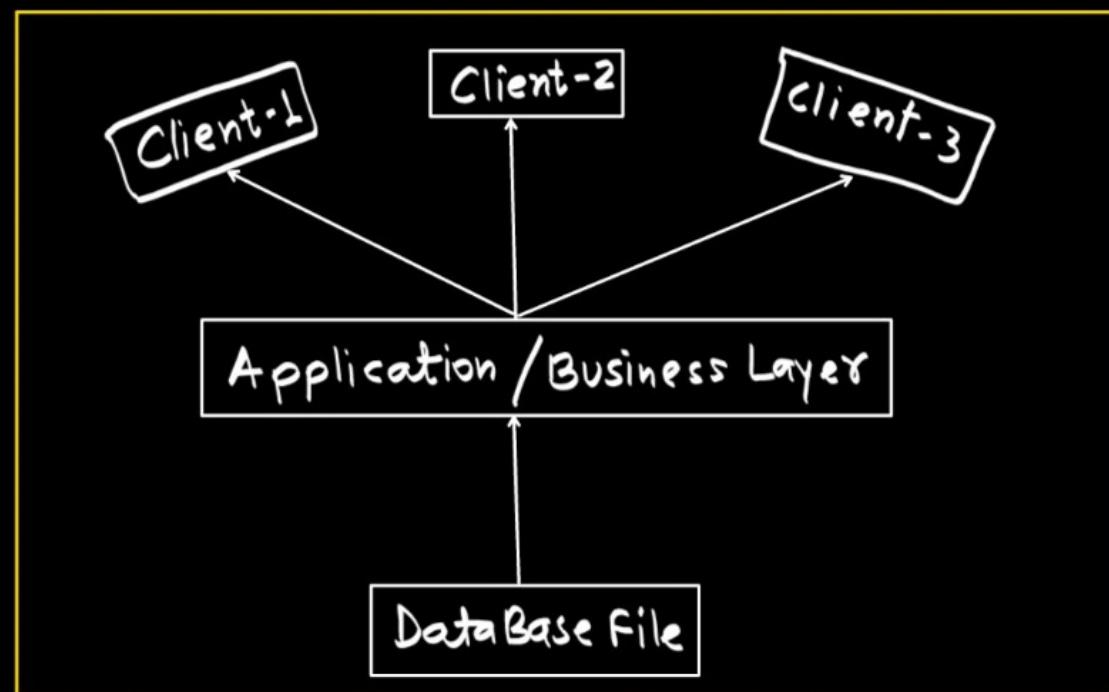
Abstraction / 3 Schema Architecture ⇒

1. External Schema / View Level
2. Conceptual Schema / Logical Level
3. Physical / Internal Schema / Physical Level



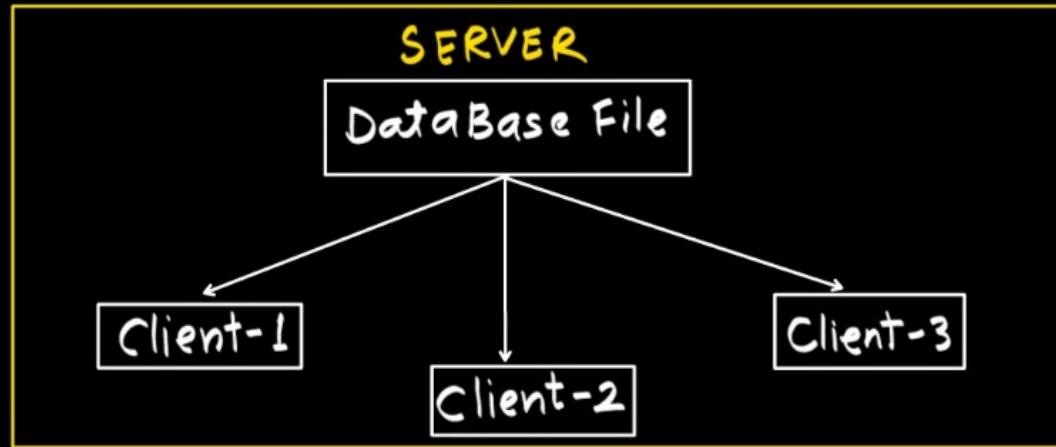
* 3 Tier / Level Architecture →

- इस Architecture में user/client Directly DataBase Server से Connect नहीं होता है।
- इसमें client की Request को Application Layer द्वारा Process किया जाता है तथा Application Layer द्वारा इस Request को Low Level Language में Convert करके Database Server के साथ Interact किया जाता है।
- Application Layer Client तथा Server के बीच Interface के रूप में काम करती है।
- इस Architecture में Users/Clients Direct Server से Connect नहीं होते हैं, इसलिए Security ज्यादा होती है।
- सभी Web Applications Facebook, Instagram, Google, Amazon इसी का use करते हैं।



→ Client की Request को DataBase Server पहले Process करता है और उसे Query में Convert करके Machine Language में DataBase file पर Operations करता है।

* Two Tier/Level Architecture →



→ इसमें प्रोसेस को Query में बदला जाता है, तो इसे Low Level Language कहते हैं।

- इस Architecture में Client Directly DataBase Server से Connect होता है।
- Also known as Client-Server Architecture.
- Limited users के लिए Best.
- यदि Number of users increase हो जाते हैं, तो इसके Fail होने के chances भी बढ़ जाते हैं।
- Less Secure.
- Maintenance Easy.

4. File System में अलग-² Levels पर Access करने के लिए अलग-² Physical files की Need होती है।

Whereas

DBMS में Data को अलग-² Levels पर Access करने के लिए अलग-² Views Create करने की Need होती है।

* View → Virtual Table

Master Table में ही Operations Perform करती है।

- Physically Exist नहीं करती है।
- Looks like original Physical Table

3. File System में Concurrency Level = File Level / Resource Level का होता है अर्थात् जब कोई User किसी File का use कर रहा होता है, तो दूसरा User उस File का प्रयोग नहीं कर सकता है।

Whereas

DBMS में Concurrency Level = Row Level / Record Level का होता है अर्थात् जब कोई User किसी Row का प्रयोग कर रहा होता है तो केवल वही Row अन्य users के लिए Locked रहती है, अन्य सभी Rows सभी users के लिए Available रहती है।

1. DBMS में users विंग Details की जोने Data को Access कर सकता है। इसमें users को केवल वही Information ही जाती है, जो उसे चाहिए।
(Table, Row, Column)
2. किसी File System में Data की Secondary Memory से Primary Memory में लोने (Data Access) की cost अधिक होती है, और कि पुरी File Open करनी पड़ती है।
Whereas
DBMS में Indexing का प्रयोग किया जाता है, इसलिए users को जिस Data की Requirement होती है, केवल उसे ही Access करता है।

$$\text{Input / Output Cost} = \text{Filesystem} > \text{DBMS}$$

* FileSystem v/s DBMS ⇒

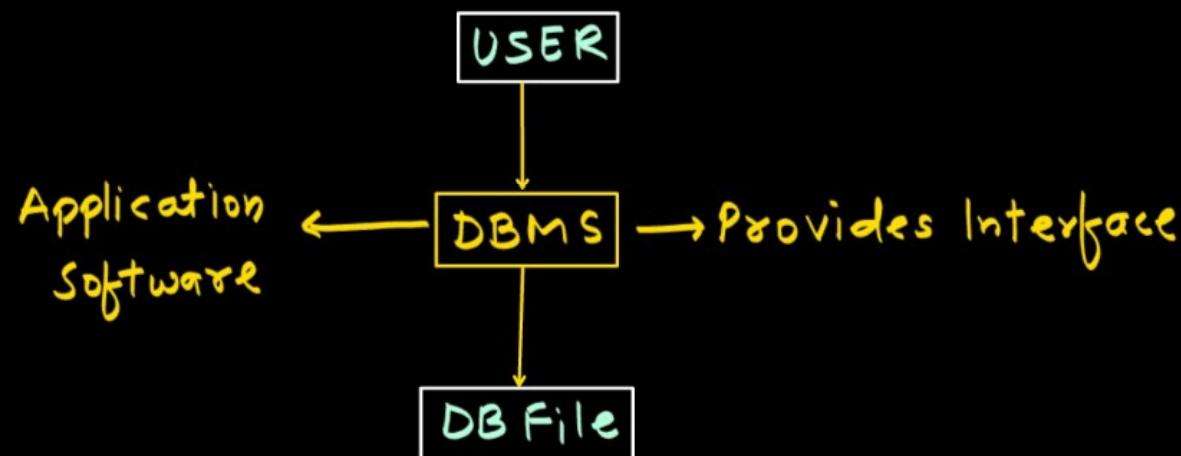
Difference	File Systems	DBMS
1. Data Redundancy	1. Same Data Multiple Files में Repeat / Duplicate हो सकता है	1. No Duplicacy
2. Data Inconsistency	2. इसमें किसी एक File का Data update करने पर अन्य Files पर Effect नहीं पड़ता है, इसलिए Data Inconsistent रह जाता है	2. Data Consistency Accurate होती है
3. Security	3. इसमें Multiple Users को Multiple Type के Access नहीं दे सकते हैं	3. Advanced Security के कारण Multiple Users को Multiple Access दे सकते हैं
4. Concurrency	4. Same Time पर Multiple Users एक File को Access नहीं कर सकते हैं	4. Same Time पर Multiple Users एक File को Access कर सकते हैं

* File System :-

- Computer में विभिन्न Files को Storage पर Manage करने का कार्य File System करता है।
- File System is managed by Operating System.
- 2 Types
 - A. FAT (File Allocation Table) - BASIC
 - B. NTFS (New Technology File System)
 - 1. इसकी Main File MFS (Master File System) कहलाती है।
 - 2. इसे 1993 में Windows NT Kernel के लिए बनाया गया था।
- File System is used for small files.
- DBMS is used to manage large file System.

* DBMS ⇒

- यह किसी user तथा DataBase File (DB) के बीच का Interface होता है, जो किसी user की Request को DataBase Query के रूप में छव्वलकर DataBase File तक पहुँचाता है और DB File से Request fetch कर प्राप्त Information को User तक पहुँचाता है।
- यह Application Software का Collection होता है, जो user तथा DB File के बीच Communication करवाता है।



Database System

Database

Collection of Data

Structured

Banking,
Railways

(RDBMS)

TABLE

Unstructured

Websites

DBMS

A Software used to manage
Database Files

Performs Operations on Database
Files

Ex - Oracle, MySQL, Access DB2,
SQL Server etc.

* Database ⇒

- Connection of interrelated Data.
- आपस में related Data का Collection.
- 2 प्रकार ⇒
 - A. Structured DataBase
 - B. Unstructured DataBase

Data



Process



Information



Knowledge



Wisdom

"DBMS"

"Data Base Management Systems"

* Data →

- Raw facts या करचे अॉकडे जो एक unorganized manner में होते हैं और जिनका कोई एक Proper Meaning नहीं विकलता है, उसे Data कहते हैं।
जैसे - Text, Number, Audio, Video, Image etc.

* Information →

- Data पर विभिन्न Operations करने से प्राप्त Meaningfull Data / Processed Data.
→ इसे Table के अवस्थित किया जाता है।
जैसे = 123, Shubham, Jodhpur (Data)

ID	NAME	ADD	Information
123	Shubham	Jodhpur	