

**CSE/IT  
2025 Batch**

**3rd  
Semester**

**DBMS**

**Polytechnic**

**ONE SHOT**





## Syllabus

1 **Introduction** ✓

2 **Data Model and Keys**

3 **Relational Model**

4 **Relational Database Design**

5 **SQL/MySQL**



## Chapter-1 : Introduction

## Syllabus :

## UNIT 1: Introduction

(6 Periods)

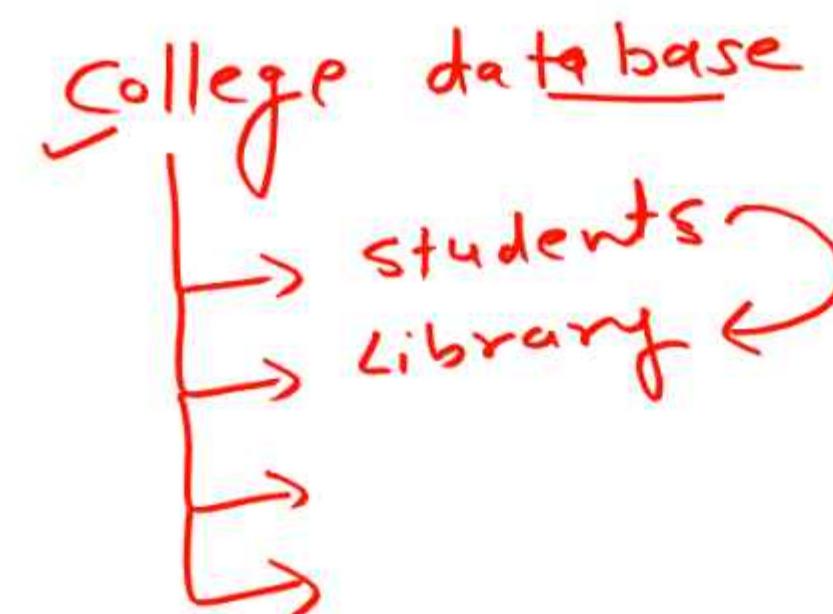
Database System Concepts and Architecture, Components of DBMS, Table Structure, Schema definition,  
Three views of Data (External View, Conceptual View, Internal View).

~~\*\*~~ Client - server  
~~\*\*~~



## डेटाबेस (Database): ✓

- डेटाबेस inter-related डेटा का समूह होता है।
- उदाहरण के लिए, एक यूनिवर्सिटी डेटाबेस।
- Database is a collection of inter-related data.
- For example, a university database.





## → डेटाबेस मैनेजमेन्ट सिस्टम (Database Management System): ✓

- DBMS एक सॉफ्टवेयर है जो यूज़र्स (Users) को डेटाबेस (Database) क्रिएट (Create) एवं मेन्टेन (Maintain) करने की सुविधा प्रदान करता है, साथ ही डेटाबेस (Database) में संग्रहीत डेटा (Data) को रिट्रीव (Retrieve) करने की भी अनुमति प्रदान करता है।
- DBMS is a software that provides the facility to users to create and maintain a database, as well as to retrieve the data stored in the database.
- Example: MySQL, Oracle, Microsoft Access





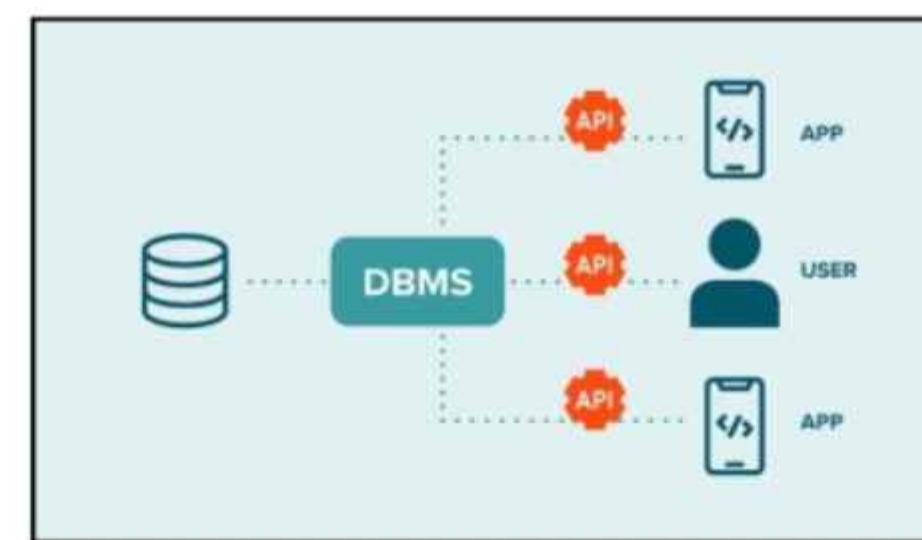
## → DBMS System (DBMS सिस्टम) :

- DBMS सिस्टम हार्डवेयर, सॉफ्टवेयर, डेटा और यूजर्स का समूह है जो डेटाबेस को मैनेज करता है।
- A DBMS system is a set of hardware, software, data, and users that manages a database.

Hardware + S/W + data + user → db system

### उदाहरण (Example):

- हार्डवेयर (Hardware): सर्वर (Server), कंप्यूटर (Computer)।
- सॉफ्टवेयर (Software): DBMS Programs (like MySQL)।
- यूजर्स (Users): शिक्षक (Teacher), डेवलपर्स (Developers), छात्र (Students)।





## → DBMS के लाभ (Advantages of DBMS): ✓

### 1. Data Security: ✓

- DBMS में डेटा को पासवर्ड और एक्सेस कंट्रोल के द्वारा सुरक्षित रखा जाता है, जिससे डेटा सुरक्षित रहता है।
- Data in DBMS is protected by passwords and access control, which keeps the data secure.

### 2. Quick Data Retrieval: ✓

- DBMS में डेटा को आसानी से खोजा जा सकता है। SQL जैसी language से हम जल्दी और सही डेटा प्राप्त कर सकते हैं।
- Data can be easily searched in DBMS. With a language like SQL, we can get data quickly and accurately.

→ Duplicacy (दोषराव)

### 3. Reduction in Data Redundancy:

- DBMS में डेटा एक जगह पर स्टोर होता है, जिससे डेटा का दोहराव कम होता है और जगह की बचत होती है।
- In DBMS, data is stored at one place, which reduces duplication of data and saves space.



#### 4. Data Integration:

- DBMS में अलग-अलग विभागों या लोगों द्वारा डेटा आसानी से शेयर किया जा सकता है।
- In DBMS, data can be easily shared by different departments or people.

#### 5. Data Accuracy and Integrity:

- DBMS में डेटा की सटीकता (accuracy) को सुनिश्चित किया जाता है, जिससे गलत डेटा की संभावना कम होती है।
- Accuracy of data is ensured in DBMS, which reduces the possibility of erroneous data.

#### 6. Backup and Recovery:

- DBMS में डेटा का बैकअप लिया जाता है और किसी समस्या के बाद इसे आसानी से वापस लाया जा सकता है।
- Data in DBMS is backed up and can be easily restored after a problem.



## → DBMS के नुकसान (Disadvantages of DBMS): ✓

### 1. Costly: ✓

- DBMS को सेटअप करने में खर्च ज्यादा होता है। छोटे व्यवसायों के लिए यह महंगा हो सकता है।
- Setting up a DBMS costs more. It can be expensive for small businesses.

### ✓ 2. Complexity:

- DBMS का use और management थोड़ी complex हो सकता है, जिसे समझने के लिए अच्छे technical knowledge की जरूरत होती है। ✓
- The use and management of DBMS can be a bit complex, which requires good technical knowledge to understand.

### 3. Performance Impact: ✓

- अगर डेटाबेस बहुत बड़ा हो, तो डेटा तक पहुंचने की गति धीमी हो सकती है।
- If the database is very large, accessing the data may be slow.



#### 4. Limited Flexibility:

- DBMS में केवल structured डेटा को ही रखा जा सकता है, जबकि unstructured डेटा (जैसे वीडियो, इमेज) को संभालना कठिन हो सकता है।
- Only structured data can be stored in a DBMS, while unstructured data (e.g. videos, images) may be difficult to handle.

#### 5. Dependence on the System:

- DBMS सिस्टम के किसी भी खराबी या क्रैश होने की स्थिति में पूरा डेटाबेस प्रभावित हो सकता है, जिससे organization का operations बाधित (disrupt) हो सकता है।
- In case of any malfunction or crash of the DBMS system, the entire database may get affected, disrupting the operations of the organization.

\* Que: => DBMS क्या होता है? और इसके लाभ लिखा।



→ फाइल प्रोसेसिंग सिस्टम की तुलना में DBMS का लाभ (Advantage of DBMS over File Processing System): ✓

- DBMS, पुराने फाइल सिस्टम की तुलना में बेहतर है।
- DBMS is better than the old file system.

Feature	<u>File Processing System</u>	<u>DBMS</u>
① Data Redundancy	डेटा का <u>दोहराव</u> (Redundancy) हो सकता है।	डेटा एक जगह पर रखा जाता है, जिससे <u>दोहराव</u> <u>नहीं</u> होता।
② Data Security	सुरक्षा <u>कम</u> होती है, कोई भी व्यक्ति <u>डेटा</u> को बदल सकता है।	सुरक्षा <u>उपाय</u> होते हैं, जैसे <u>पासवर्ड</u> और <u>अधिकार</u> <u>नियंत्रण</u> ।
③ Data Integrity	डेटा में <u>बदलाव</u> से <u>अखंडता</u> <u>प्रभावित</u> हो सकती है।	डेटा की <u>अखंडता</u> बनी रहती है, क्योंकि <u>एक</u> <u>जगह</u> पर <u>अपडेट</u> होता है।



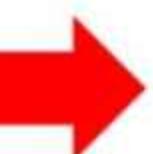
Feature	<u>File Processing System</u>	DBMS
④ Data Management	डेटा का Management <u>मैन्युअल रूप</u> से किया जाता है। <u>Manual</u>	डेटा का Management <u>ऑटोमेटेड</u> होता है <u>Automatic</u> और अधिक आसान होता है।
⑤ Data Access	डेटा एक्सेस करना <u>धीमा</u> और <u>समय लेने वाला</u> होता है। <u>Slow</u>	डेटा को क्वेरी के द्वारा जल्दी से एक्सेस किया <u>Fast</u> जा सकता है।
⑥ Multi-User Support	एक समय में एक ही व्यक्ति <u>डेटा पर</u> काम कर सकता है। <u>single user</u>	कई उपयोगकर्ता एक साथ <u>डेटा पर</u> काम कर <u>multiple user</u> सकते हैं।
⑦ Backup and Recovery	बैकअप और रिकवरी मुश्किल हो सकता है। <u>Hard</u>	ऑटोमैटिक बैकअप और रिकवरी की सुविधा होती है। <u>easy</u>



→ फाइल प्रोसेसिंग सिस्टम की तुलना में DBMS का लाभ (Advantage of DBMS over File Processing System):

- DBMS, पुराने फाइल सिस्टम की तुलना में बेहतर है।
- DBMS is better than the old file system.

Feature	File Processing System	DBMS
Data Redundancy	Data redundancy may occur (duplicate data in multiple files).	Data is stored in one place, eliminating redundancy.
Data Security	Security is low, anyone can modify the data.	Security features like passwords and access control are present.
Data Integrity	Changes in data may affect integrity across files.	Data integrity is maintained as updates happen in one place.



Feature	File Processing System	DBMS
Data Management	Data is managed manually, which can be time-consuming.	Data management is automated, making it easier and faster.
Data Access	Data access is slow and requires manual opening of files.	Data can be accessed quickly using queries (e.g., SQL).
Multi-User Support	Only one person can work on the data at a time.	Multiple users can work on the data simultaneously.
Backup and Recovery	Backup and recovery can be difficult.	Automatic backup and recovery features are available.

~~Ques.~~ file Processing system के अन्य DBMS के advantage

Ques. file Processing system के अन्य DBMS के advantage

20%



### → 1. Data Abstraction: ✓

- Data Abstraction का मतलब है complex details को hide करना और user को सिर्फ जरूरी information दिखाना।
- Data Abstraction means hiding complex details and showing only necessary information to the user.
- User को यह नहीं पता कि डेटा अंदर कैसे store हुआ है, सिर्फ उसे relevant data दिखाता है।
- The user does not know how the data is stored inside, he only sees the relevant data

### → तीन स्तर (Levels of Abstraction):



- Physical Level (Internal View): ✓
- Logical Level (Conceptual View): ✓
- View Level (External View): ✓



→ डाटा एब्स्ट्रैक्शन क्यों जरूरी है? (Why is data abstraction important?): ✓

1. Security: ✓

- डेटा एब्स्ट्रैक्शन डेटा को unauthorized access से बचाता है।
- Data abstraction protects data from unauthorized access.

2. Privacy: ✓

- डेटा एब्स्ट्रैक्शन Users को यह देखने से बचाता है कि उनका डेटा कैसे Store किया जाता है।
- Data abstraction shields users from seeing how their data is stored.

3. Efficiency: ✓

- डेटा एब्स्ट्रैक्शन डेवलपर्स को high-level concepts के साथ काम करने की अनुमति देता है, जिससे सॉफ्टवेयर डिज़ाइन अधिक efficient हो जाता है।
- Data abstraction allows developers to work with high-level concepts, making software design more efficient.



## 2. Data Independence (डेटा इंडिपेंडेंस) ✓

- Data Independence का मतलब है डेटा structure या storage बदलने पर भी application programs पर असर न पड़े।
- Data Independence means that application programs should not be affected even if the data structure or storage changes. Ex APP
- यह database system की flexibility और maintainability के लिए जरूरी है।
- This is necessary for the flexibility and maintainability of the database system.

### ✓ प्रकार (Types of Data Independence):

- ◆ a) Physical Data Independence ✓
- ◆ b) Logical Data Independence ✓

PDf  
↓  
anET

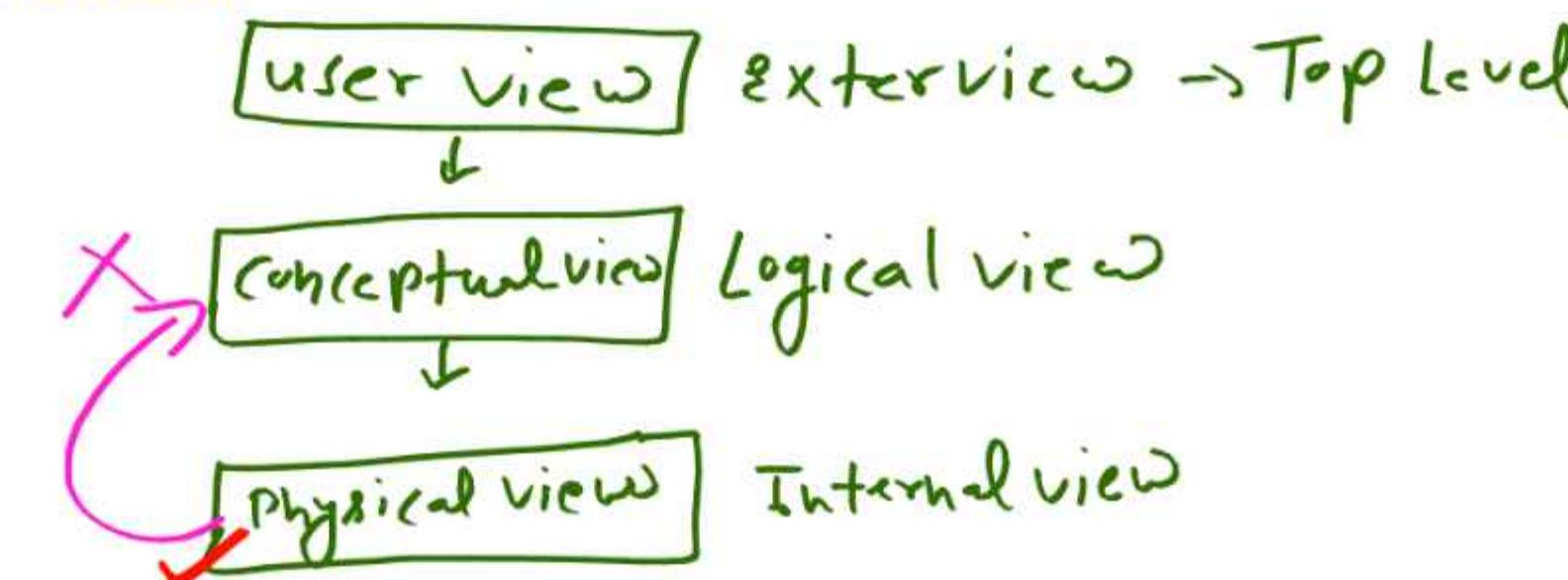
APP → [free PDf]  
↓

3rd Semester 2025



### a) Physical Data Independence:

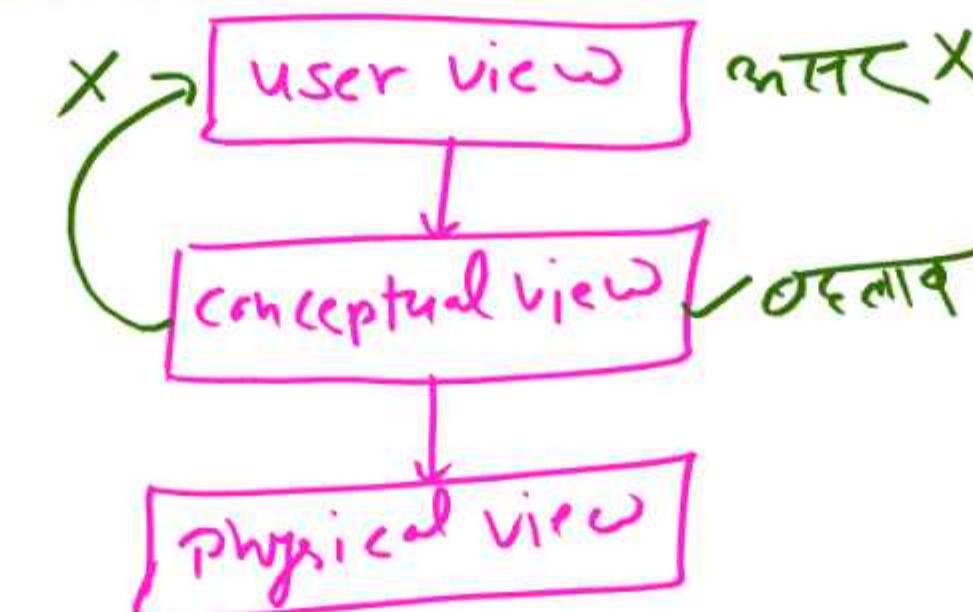
- Internal (Physical) level में बदलाव होने पर भी Conceptual level पर असर न पड़े।
- Even if there is a change at the internal (physical) level, it should not affect the conceptual level.
- Example: अगर storage format बदला (heap file → B-tree), तब भी tables और relationships पर असर नहीं होगा।
- Example: If the storage format changes (heap file → B-tree), then also the tables and relationships will not be affected.





### b) Logical Data Independence ✓

- Conceptual (Logical) level में बदलाव होने पर भी External views (user view) पर असर न पड़े।
- External views (user views) should not be affected even if there is a change at the conceptual (logical) level.
- Example: अगर database में नया column add कर दिया, तो existing user views या application programs break नहीं होंगे।
- Example: If a new column is added to the database, then existing user views or application programs will not break.





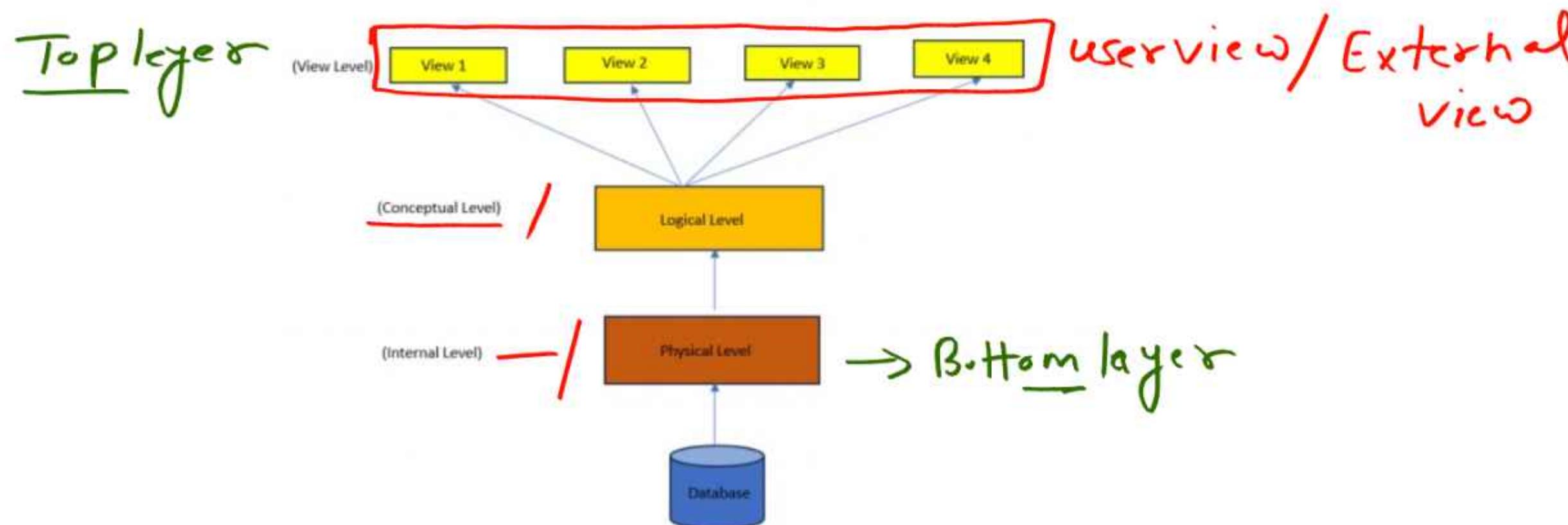
### Relationship Between Both:

- Data Abstraction → Complexity hide करता है। (It hides complexity.)
- Data Independence → Changes के असर से बचाता है। (Protects from the impact of changes.)



## Three Views of Data:

- डेटाबेस सिस्टम में डेटा को **तीन स्तरों (views)** में विभाजित किया गया है ताकि **data abstraction** और **data independence** प्राप्त हो सके। यह **ANSI-SPARC 3-tier architecture** कहलाता है।





### 1. View or External Level :

- View or External Level डेटा एब्स्ट्रैक्शन का सबसे topmost level है।
- View or External Level is the topmost level of data abstraction.
- यह layer एंड-यूजर्स के लिए है, जहां वे अपनी जरूरत के अनुसार डेटा देख और एक्सेस कर सकते हैं।
- This level is for end-users where they can view and access the data as per their need.
- इसे user view या application view भी कहते हैं।
- This is also called user view or application view.

### विशेषताएँ (Features):

- अलग-अलग users के लिए अलग view / Different views for different users
- सुरक्षा और privacy के लिए उपयोगी / Useful for security and privacy
- Physical और logical details hidden / Physical and logical details hidden



## उदाहरण (Example): ✓

- Student portal पर student को सिर्फ उसके marks और profile दिखेंगे
- On the student portal, the student will only see his marks and profile.
- Teacher को सिर्फ attendance और class records दिखेंगे
- The teacher will only see attendance and class records.



## 2. Logical or Conceptual Level :

- यह बीच का layer है। यह define करता है कि डेटाबेस में कौन सा डेटा मौजूद है और उनके बीच क्या संबंध हैं।
- It is intermediate level. It defines what data is present in database and their relationships between them .  
*① what data is present & relation b/w them*
- यह डेटाबेस का logical structure दिखाता है।
- It shows the logical structure of the database.
- इसमें टेबल्स, कॉलम्स, रिलेशनशिप्स, constraints आदि define किए जाते हैं।
- Tables, columns, relationships, constraints etc. are defined in it.
- यह पूरा database का global view होता है, जिसे DBA maintain करता है।
- This is the global view of the entire database, which is maintained by the DBA.

*① Database administrator*



## विशेषताएँ (Features): ✓

- कौन सा data store होगा और उनके relationships
- What data will be stored and their relationships
- Physical details hidden ✓ कृति ✗ X
- Physical details hidden
- Users को सिर्फ schema दिखता है, storage details नहीं
- Users can only see the schema, not the storage details

## उदाहरण (Example): ✓

- ER diagram, tables, attributes, primary keys, foreign keys
- Example: Student (RollNo, Name, Marks)



### 3. Physical or Internal Level:

- यह डेटा एब्स्ट्रैक्शन का सबसे निचला स्तर है जो define करता है कि डेटाबेस में डेटा कैसे store किया जाता है।  
*How data is stored ?*
- It is the lowest level of data abstraction which defines how data is stored in database .
- इसे समझना बहुत जटिल है और इसलिए इसे users से छिपा कर रखा जाता है। डेटाबेस एडमिनिस्ट्रेटर यह decide करता है कि डेटाबेस में डेटा को कैसे और कहाँ store किया जाए।
- It is very complex to understand and hence kept hidden from user. Database administrator decides how and where to store the data in database.
- इसमें data structure, indexing, storage path, compression जैसी details होती हैं।
- It contains details like data structure, indexing, storage path, compression.



### विशेषताएँ (Features):

- Storage structure और access method पर focus
- Focus on storage structure and access method
- Users को यह view नहीं दिखता
- Users do not see this view
- Hardware और OS specific
- Hardware and OS specific

### उदाहरण (Example):

- File organization methods (Heap, Hash, B-Tree)
- Data blocks, pointers, & storage allocation



### Real-Life Example:

Library

Physical level

- **Physical Level:** Book library में किताबें किस shelf पर रखी हैं, यह librarian को पता है, user को नहीं।
- **Physical Level:** The librarian knows on which shelf the books are kept in the book library, not the user.
- **Logical Level:** Library के database में कौन-कौन सी किताबें हैं, उनकी categories  $\Rightarrow$  Logical level
- **Logical Level:** The categories of the books in the library database.
- **External Level:** Student को सिर्फ उसका account और borrowed books दिखती हैं।  $\rightarrow$  User view
- **External Level:** The student can only see his account and borrowed books.

अगर library में किताबों की जगह बदल दी जाए (Physical change), तो student के view पर कोई असर नहीं (Physical Data Independence)।

If the position of books in the library is changed (physical change), then there is no impact on the student's view (Physical Data Independence).

अगर library database में नए fields add किए (जैसे ISBN), तब भी student का account view नहीं बदलेगा (Logical Data Independence)।

Even if new fields are added to the library database (like ISBN), the student account view will not change (Logical Data Independence).

100%  
\* 100%

Que:  $\Rightarrow$  DBMS में डेटा के तीन Level को समझाओ।  
or

Que:  $\Rightarrow$  डेटा के तीन views कोटि-कोटि से होते हैं ? समझाओ।

short Question 2.5 marks

① write the short notes on →

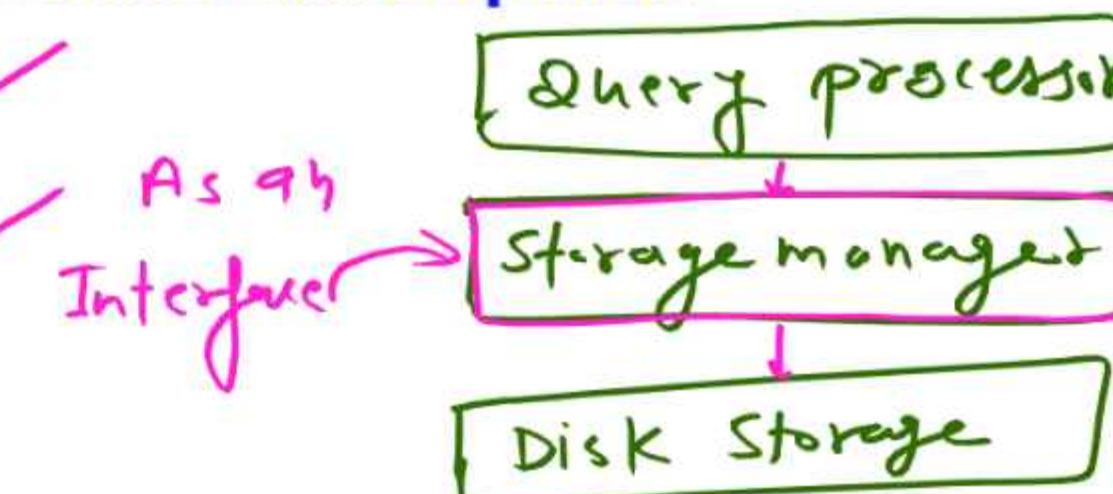
30%  
↓  
① Data Independence    ② Data Abstraction



## → डेटाबेस सिस्टम के घटक (Components of a Database System): ✓

- डेटाबेस सिस्टम मुख्य रूप से तीन भागों में विभाजित होता है:
- The database system is mainly divided into three parts:

- ① Query Processor (क्वेरी प्रोसेसर)
- ② Storage Manager (स्टोरेज मैनेजर)
- ③ Disk Storage (डिस्क स्टोरेज)



### ① Query Processor (क्वेरी प्रोसेसर): ✓

- ◆ यह यूज़र की क्वेरी को समझता है और उसे मशीन-एक्सिक्यूटेबल इंस्ट्रक्शन में बदलता है।
  - ◆ It understands the user's query and converts it into machine-executable instructions.
- ① ◆ DML (Data Manipulation Language) स्टेटमेंट को प्रोसेस करता है और डेटाबेस से रिजल्ट निकालता है।
- ◆ Processes DML (Data Manipulation Language) statements and retrieves results from the database. *Select, Update, Insert, Delete*



→ मुख्य घटक (Main Components):

1. DML Compiler - ✓

- DML स्टेटमेंट को लो-लेवल मशीन लैंग्वेज में ट्रांसलेट करता है।

- Translates DML statements into low-level machine language.

2. DDL Interpreter - ✓

- DDL स्टेटमेंट को मेटाडेटा टेबल में बदलता है।

- Converts DDL statements into metadata tables.

3. Embedded DML Pre-compiler - ✓

- एप्लिकेशन में एम्बेडेड DML स्टेटमेंट्स को प्रोसेस करता है।

- Processes DML statements embedded in the application.

4. Query Optimizer - ✓

- DML Compiler द्वारा जेनरेट किए गए इंस्ट्रक्शंस को एक्सिक्यूट करता है।

- Executes instructions generated by the DML Compiler.

SQL Query

SELECT students FROM  
college where id='1';



## ② Storage Manager (स्टोरेज मैनेजर):

- ◆ यह डेटाबेस और यूज़र क्वेरी के बीच इंटरफ़ेस का काम करता है।
- ◆ डेटाबेस की सुरक्षा, अखंडता (Integrity), और कंसिस्टेंसी बनाए रखता है।
- ◆ यह डेटा को स्टोर, अपडेट, डिलीट और रिट्रीव करने के लिए जिम्मेदार होता है।
- ◆ It acts as an interface between the database and user queries.
- ◆ It maintains the security, integrity, and consistency of the database.
- ◆ It is responsible for storing, updating, deleting, and retrieving data.

## ❖ मुख्य घटक (Main Components):

### 1. Authorization Manager -

- यह चेक करता है कि कौन-सा यूज़र कौन-सा ऑपरेशन कर सकता है।
- It checks which user can perform which operation.



## 2. Integrity Manager - ✓

- यह डेटाबेस में **बदलाव करते समय इंटीग्रिटी कंस्ट्रेंट्स** को लागू करता है।
- It enforces integrity constraints while making changes to the database.

## 3. Transaction Manager - ✓

- यह मल्टीपल **यूज़र्स के बीच ट्रांजेक्शन** को मैनेज करता है, जिससे **डेटा कंसिस्टेंट** बना रहे।
- It manages transactions between multiple users, so that data remains consistent.

## 4. File Manager - ✓

- यह **फाइल स्पेस** और **डेटा स्ट्रक्चर** को मैनेज करता है।
- It manages file space and data structure.

## 5. Buffer Manager - ✓

- यह **डेटा** को **सेकेंडरी स्टोरेज** से **मैन मेमोरी में ट्रांसफर** करने का काम करता है।
- It works to transfer data from secondary storage to main memory.



### ③ Disk Storage (डिस्क स्टोरेज):

- ♦ डेटाबेस डेटा को स्थायी रूप से स्टोर करने के लिए **डिस्क स्टोरेज का** उपयोग करता है।
- ♦ Database uses disk storage to store data permanently.

~~Permanent~~

### मुख्य घटक (Main Components):

#### 1. Data Files - ✓

- डेटाबेस का असली डेटा स्टोर करता है। **actual**
- Stores the actual data of the database.

#### 2. Data Dictionary - ✓

- डेटाबेस की संरचना (Metadata) को स्टोर करता है। **metadata**
- Stores the structure (metadata) of the database.

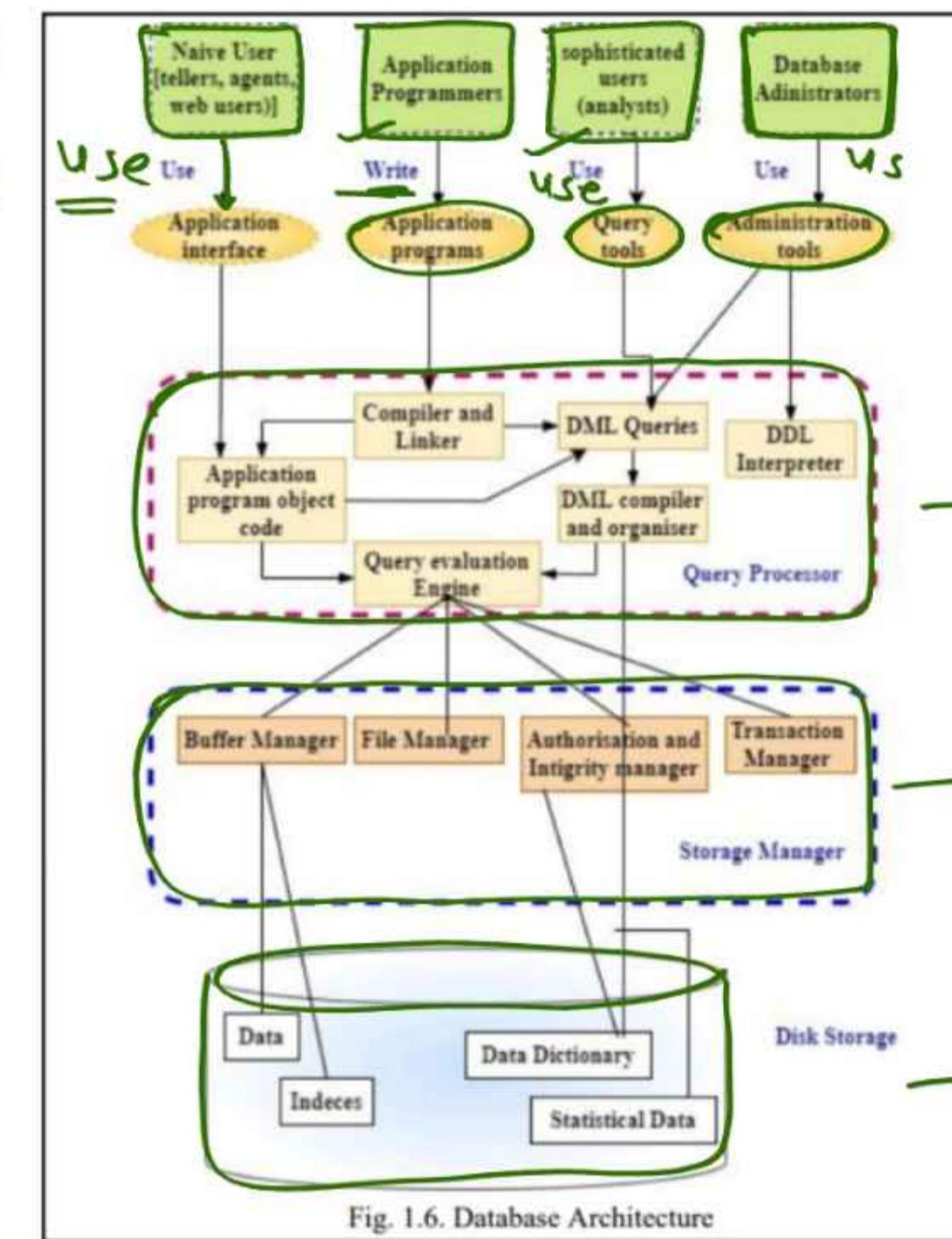
#### 3. Indices - ✓

- डेटा को तेज़ी से एक्सेस करने के लिए **इंडेक्स** बनाए जाते हैं।
- Indexes are created to access data quickly.



## Structure of DBMS:

Native  
teller  
user



Ques: DBMS के component को लिखें। 5marks



### Table Structure:

- DBMS में टेबल (Table) वह स्ट्रक्चर है जिसमें डेटा को rows (पंक्तियों) और columns (स्तंभों) के रूप में व्यवस्थित किया जाता है।
- Table in DBMS is a structure in which data is organized in the form of rows and columns.
- हर टेबल का एक नाम होता है और इसमें रिकॉर्ड्स (rows) और फील्ड्स (columns) होते हैं।
- Each table has a name and contains records (rows) and fields (columns).

### Example Table Structure : Table Name - Students

Column Name (कॉलम नाम)	Data Type (डेटा प्रकार)	Constraint (बाधा)
StudentID ००१	INT INT	PRIMARY KEY
Name Amar	VARCHAR VARCHAR(50)	NOT NULL
Age 27	→ INT	
Course Computer	VARCHAR(30)	
Email	VARCHAR(100)	UNIQUE

Row/  
Tuple/  
Record

$R_1 \rightarrow$   
 $R_2 \rightarrow$   
 $R_3 \rightarrow$   
 $R_4 \rightarrow$

	$C_1$	$C_2$	$C_3$	$C_4$
$R_1 \rightarrow$				
$R_2 \rightarrow$				
$R_3 \rightarrow$				
$R_4 \rightarrow$				

table/  
Relation

Column  
↳ Attribute/  
field



### Components of Table Structure:

#### 1. Table Name (टेबल का नाम)

- यह टेबल को uniquely पहचानता है।
- It uniquely identifies the table.
- Example: Students, Employees

→ Record

Name	Branch	Age	City
------	--------	-----	------

#### 2. Rows / Tuples / Records (पंक्तियाँ या रिकॉर्ड्स)

- टेबल की हर पंक्ति एक डेटा रिकॉर्ड को दिखाती है।
- Each row of the table represents one data record.
- Example: एक छात्र का पूरा डेटा (Roll No, Name, Marks)।
- Example: Complete data of a student (Roll No, Name, Marks).

#### 3. Columns / Attributes / Fields (स्तंभ या गुणधर्म)

- हर कॉलम किसी विशेष डेटा को दर्शाता है।
- Each column represents a particular data.
- Example: RollNo, Name, Course



#### 4. Data Types (डेटा प्रकार)

- हर कॉलम का एक डेटा टाइप होता है जो बताता है कि उस कॉलम में किस प्रकार का डेटा होगा।
- Each column has a data type that indicates what type of data that column will contain.
- Example:
  - INT → संख्या के लिए (for number) ०१, ०२,
  - VARCHAR(50) → टेक्स्ट के लिए (for text) Amar
  - DATE → तारीख के लिए (for date) ०६/११/२०२५

#### 5. Constraints (बाधाएँ)

- Constraints डेटा की शुद्धता और अखंडता बनाए रखते हैं।
- Constraints maintain the correctness and integrity of the data.
- Common Constraints:
  - PRIMARY KEY → हर row को uniquely पहचानता है (Uniquely identifies each row)
  - FOREIGN KEY → दूसरी टेबल से लिंक करता है (Links to another table)
  - NOT NULL → कॉलम खाली नहीं रह सकता (Column cannot be blank)
  - UNIQUE → वैल्यू दोहराई नहीं जा सकती (Value cannot be repeated)



## Example Table: Students ✓

StudentID	Name	Age	Course	Email
101	Ramesh	20	B.Tech	<a href="mailto:ramesh@gmail.com">ramesh@gmail.com</a>
102	Suman	22	BCA	<a href="mailto:suman123@yahoo.com">suman123@yahoo.com</a>
103	Priya	21	MCA	<a href="mailto:priya.mca@gmail.com">priya.mca@gmail.com</a>
104	Ankit	23	B.Sc	<a href="mailto:ankit.science@gmail.com">ankit.science@gmail.com</a>
105	Kavita	20	MBA	<a href="mailto:kavita.mba@gmail.com">kavita.mba@gmail.com</a>

↑

↗

↑

↑

↑

Row X Column  $\Rightarrow$  Table

Record X field  $\Rightarrow$  Table / Relation



### Schema (स्कीमा): ✓

◆ डेटाबेस स्कीमा डेटाबेस का एक डिज़ाइन या ढांचा है जो बताता है कि डेटा कैसे व्यवस्थित किया गया है और आपस में कैसे जुड़ा है।

इसमें टेबल्स, कॉलम्स, उनके डेटा टाइप और टेबल्स के बीच के रिश्तों की जानकारी होती है, लेकिन इसमें असली डेटा शामिल नहीं होता। ✗

◆ A Database Schema is the logical design or structure of a database that defines how data is organized and related.

It describes tables, columns, their data types, and the relationships among them, but does not include the actual data.

Schema is a logical structure that tells how data is arranged in database and how they are related.

↳ actual data ✗

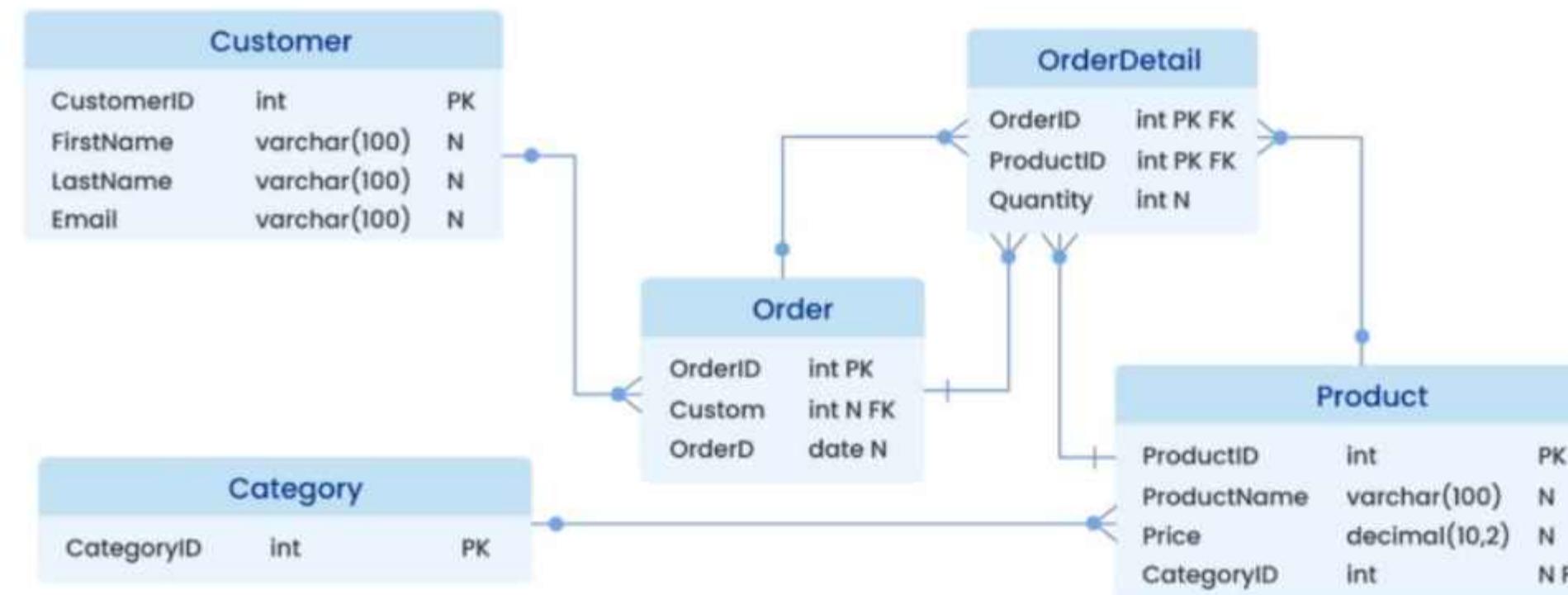


### Key Points:

- ✓ यह Blueprint (नक्शा) की तरह काम करता है।  
✓ It works like a blueprint.
  
- ✓ इसमें Tables, Columns, Constraints और Relationships की जानकारी होती है।  
✓ It contains information about tables, columns, constraints and relationships.
  
- ✓ इसमें Actual Data नहीं होता, सिर्फ़ डिजाइन होता है।  
✓ It does not contain actual data, it only contains design.



## Database Schema Diagram:



### Example:

- Schema Name:** E-Commerce
- Table Name:** Customer, Order, OrderDetail , Product, Category

short question 2.5 marks

Ques ⇒ write a short notes on:-

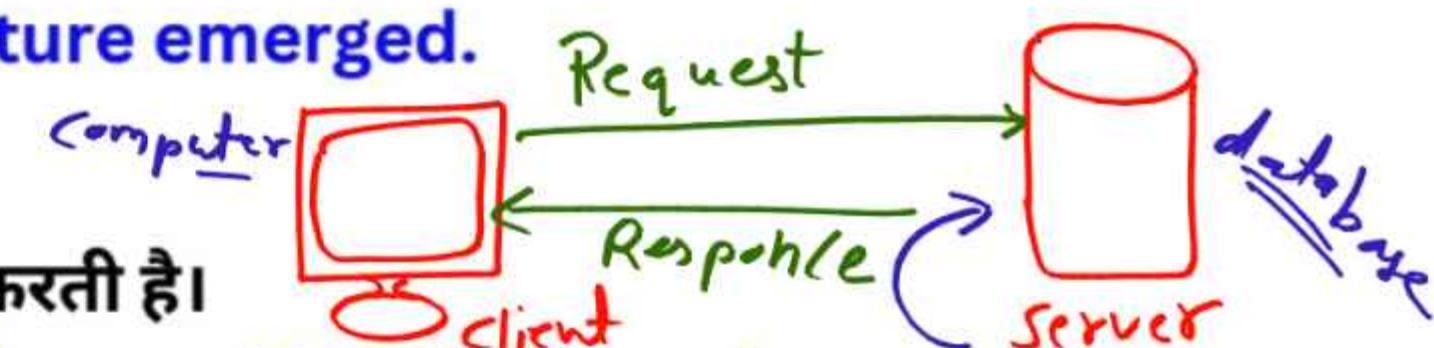
① Table

② Schema



### Client/Server Architecture in DBMS:

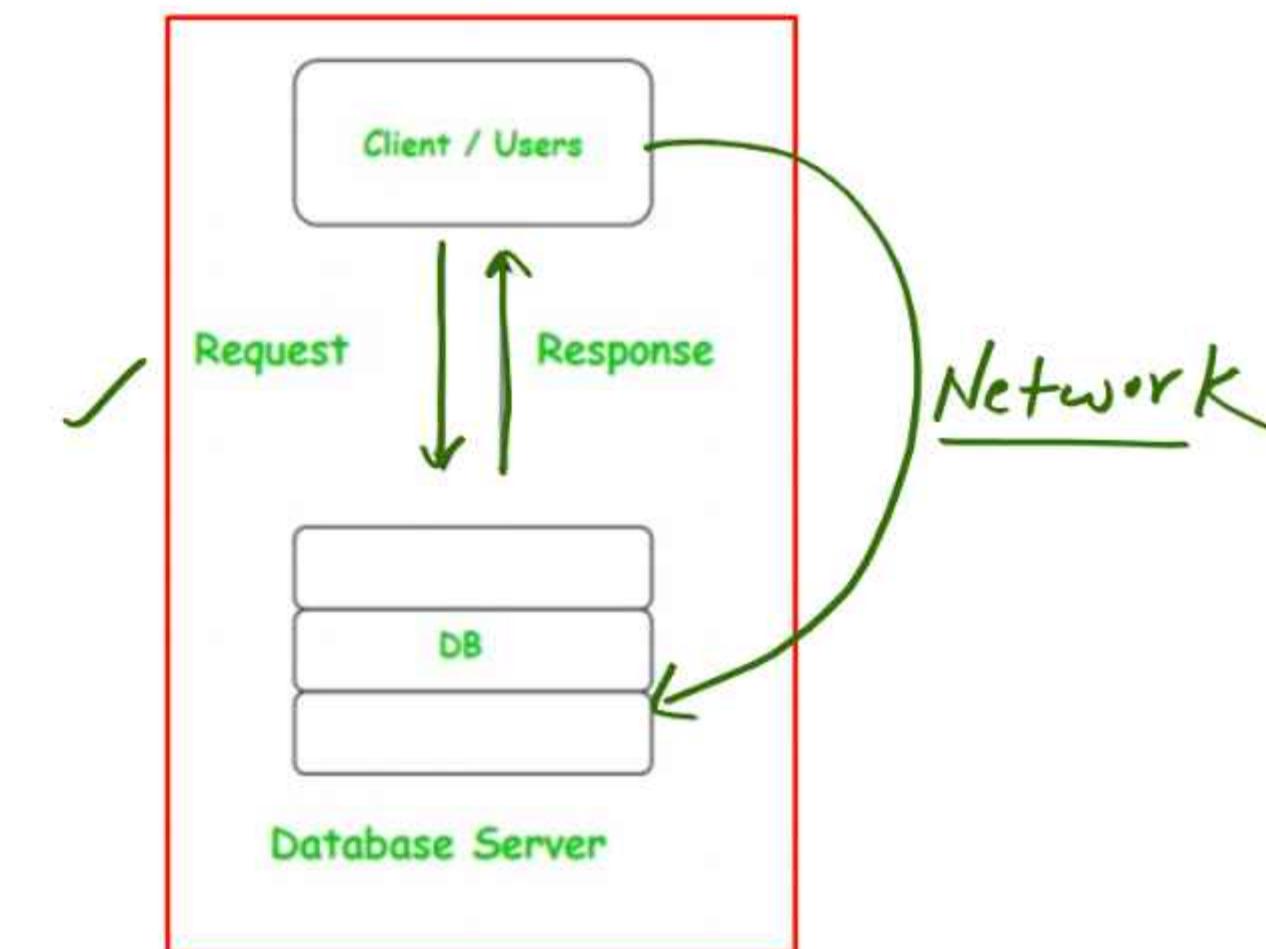
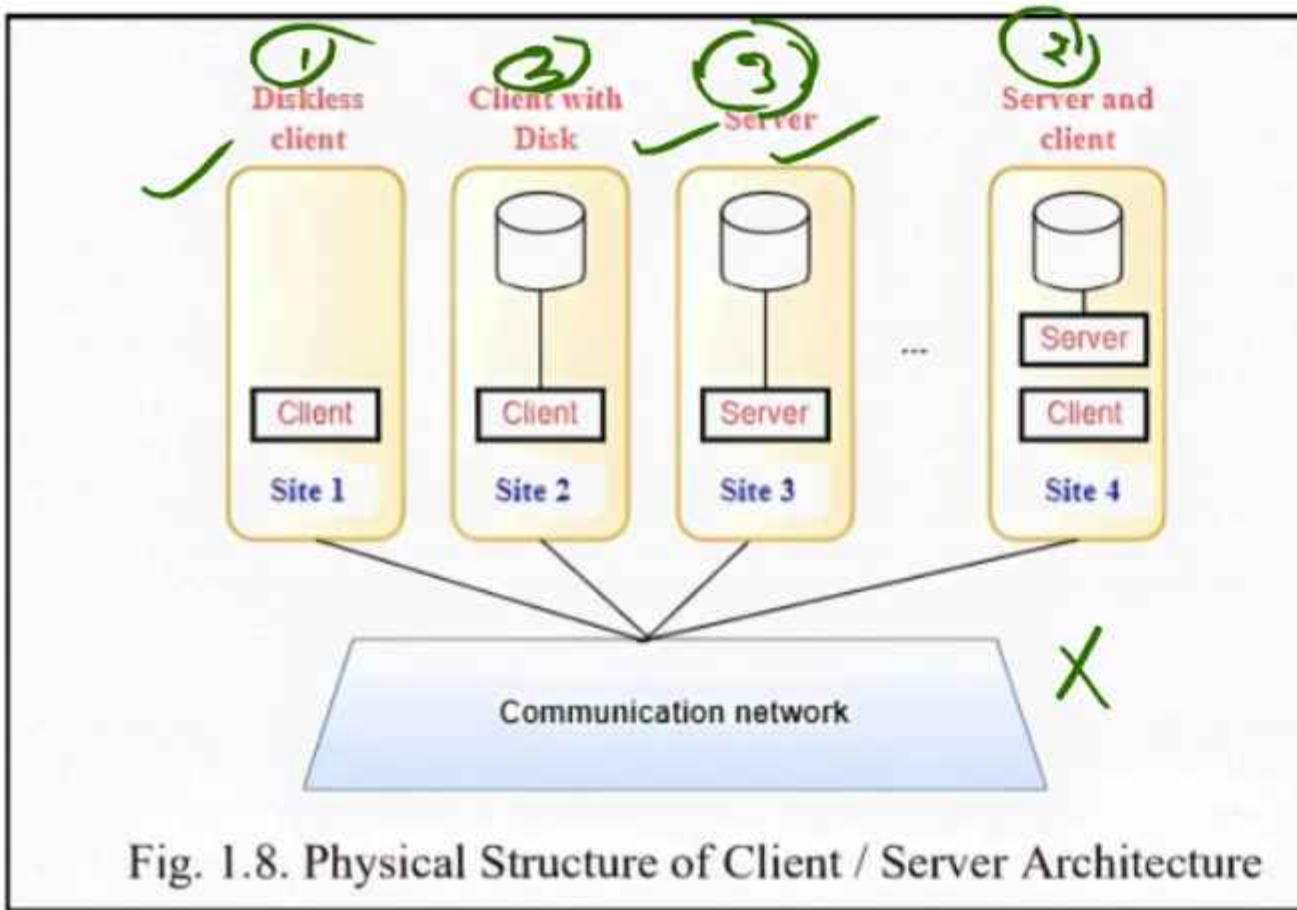
- जैसे-जैसे डेटाबेस मैनेजमेंट सिस्टम (DBMS) ने यूज़र-साइड प्रोसेसिंग कैपेबिलिटी का लाभ उठाना शुरू किया, Client/Server Architecture उभरकर सामने आया।
- As database management systems (DBMS) began to take advantage of user-side processing capabilities, the Client/Server Architecture emerged.
- क्लाइंट (Client) -**
  - यह वह मशीन होती है जो यूज़र इंटरफ़ेस और लोकल प्रोसेसिंग करती है।
  - This is the machine that performs the user interface and local processing.
- सर्वर (Server) -**
  - यह डेटाबेस को स्टोर, मैनेज और प्रोसेस करता है और क्लाइंट से आने वाली रिक्वेस्ट को हैंडल करता है।
  - It stores, manages, and processes the database and handles incoming requests from clients.





## → ↲ क्लाइंट/सर्वर कनेक्शन (Client/Server Connection)

- ✓ यूज़र (Client) नेटवर्क के ज़रिए सर्वर से जुड़ता है।
- ✓ The user (client) connects to the server via the network.
- ✓ क्लाइंट रिक्वेस्ट भेजता है और सर्वर उस रिक्वेस्ट को प्रोसेस करके डेटा भेजता है।
- ✓ The client sends a request and the server processes that request and sends the data.





## Physical Structure of Client/Server Systems:

### ① Client-Only Machines -

- ये सिर्फ क्लाइंट-सॉफ्टवेयर वाले कंप्यूटर होते हैं (जैसे, वर्कस्टेशन या पीसी)।
- These are client-software-only computers (e.g., workstations or PCs).

### ② Client & Server Combined -

- कुछ सिस्टम्स में वर्कस्टेशन क्लाइंट और सर्वर, दोनों का काम करते हैं।
- In some systems, workstations act as both clients and servers.

### ③ Dedicated Server Machines -

- कुछ सिस्टम्स में सर्वर का मुख्य काम केवल डेटाबेस हैंडल करना होता है।
- In some systems the main function of the server is simply to handle the database.



## → Server Services in Client/Server Architecture:

- सर्वर कंप्यूटर हार्डवेयर और सॉफ्टवेयर को मिलाकर बना होता है, जो क्लाइंट को विभिन्न सेवाएँ प्रदान करता है, जैसे:
  - A server is a combination of computer hardware and software that provides various services to clients, such as:
    - ✓ **File Access –**
      - क्लाइंट को डेटा फ़ाइल एक्सेस की सुविधा देता है। / Provides data file access to the client.
    - ✓ **Printing Services –**
      - नेटवर्क पर प्रिंटर को कंट्रोल करता है। / Controls printers over the network.
    - ✓ **Archiving Services –**
      - डेटा बैकअप और स्टोरेज सेवाएँ प्रदान करता है। / Provides data backup and storage services.
    - ✓ **Database Access –**
      - क्लाइंट से आई डेटाबेस क्वेरीज़ को प्रोसेस करता है। / Processes database queries from clients.

~~\* 100%~~

Que: Client - Server Architecture को समझाओ।



## Three Level Architecture of DBMS: ✓

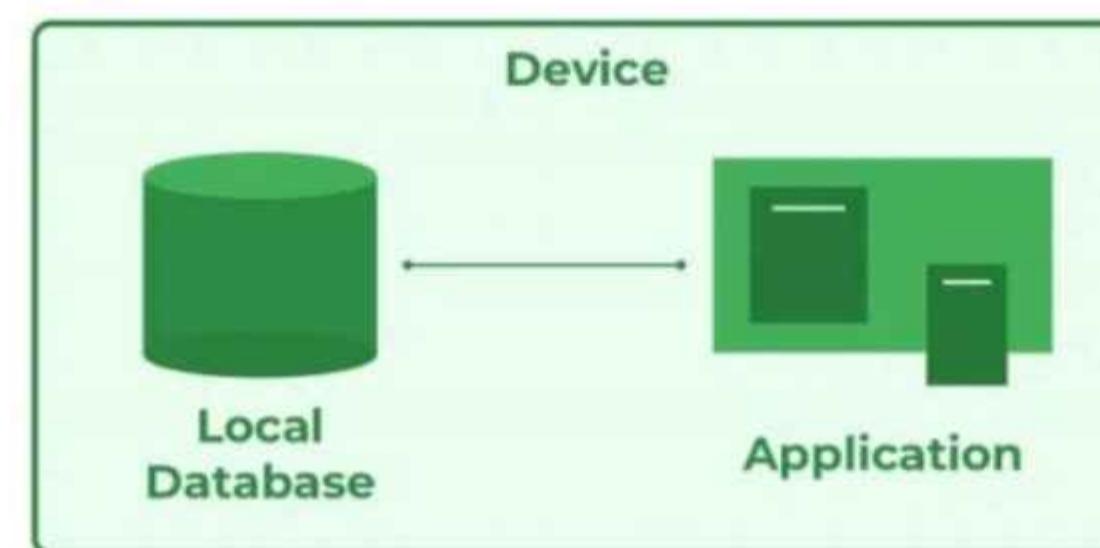
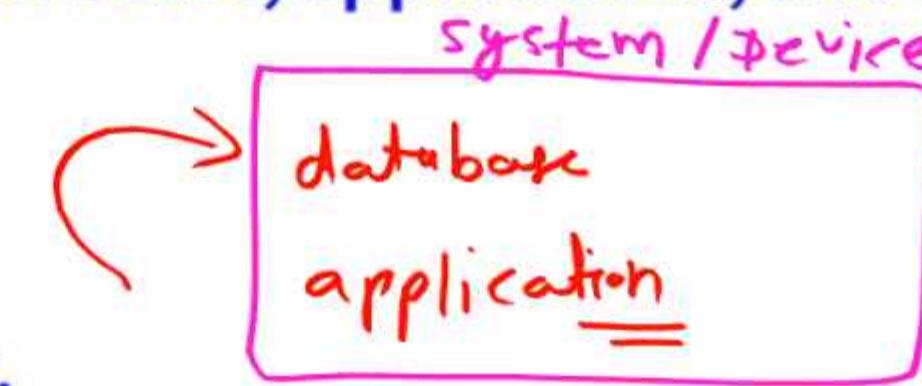
### ① 1-Tier Architecture (Single-Layer Architecture):

#### Structure:

- सब कुछ एक ही सिस्टम पर होता है। यूजर, एप्लिकेशन और DBMS एक ही मशीन पर रन करते हैं।
- **Everything happens on the same system. Users, applications, and the DBMS run on the same machine.**

#### Use Case: ✓

- डेवलपमेंट और टेस्टिंग एनवायरनमेंट।
- **Development and Testing Environment.**





### विशेषताएँ (Features):

- ✓ सीधी इंटरैक्शन - यूज़र सीधे डेटाबेस के साथ काम करता है।
- ✓ कोई नेटवर्क कनेक्शन नहीं - सभी डेटा लोकल मशीन पर स्टोर होते हैं।
- ✓ सरल और सस्ता (Cost-Effective) - कोई अतिरिक्त हार्डवेयर या सर्वर की आवश्यकता नहीं।
- ✓ Direct interaction - User works directly with the database.
- ✓ No network connection - All data is stored on the local machine.
- ✓ Simple and cost-effective - No additional hardware or server required.

### उदाहरण (Examples):

- MS Excel, MS Access, और अन्य Standalone Apps।





### Advantages:

- ✓ सिंपल आर्किटेक्चर - सेटअप आसान है, क्योंकि सिर्फ एक ही मशीन की ज़रूरत होती है।
- ✓ कम लागत (Cost-Effective) - कोई अतिरिक्त हार्डवेयर या नेटवर्किंग की ज़रूरत नहीं।
- ✓ तेज़ और इफेक्टिव - डेटा एक्सेस डायरेक्ट होता है, जिससे परफॉर्मेंस तेज़ होती है।
- ✓ आसान इम्प्लीमेंटेशन - छोटे प्रोजेक्ट्स के लिए बढ़िया ऑप्शन।
- ✓ Simple architecture - Setup is easy as only one machine is required.
- ✓ Cost-effective - No additional hardware or networking required.
- ✓ Fast and efficient - Data access is direct, which leads to faster performance.
- ✓ Easy implementation - Great option for small projects.

### Disadvantages:

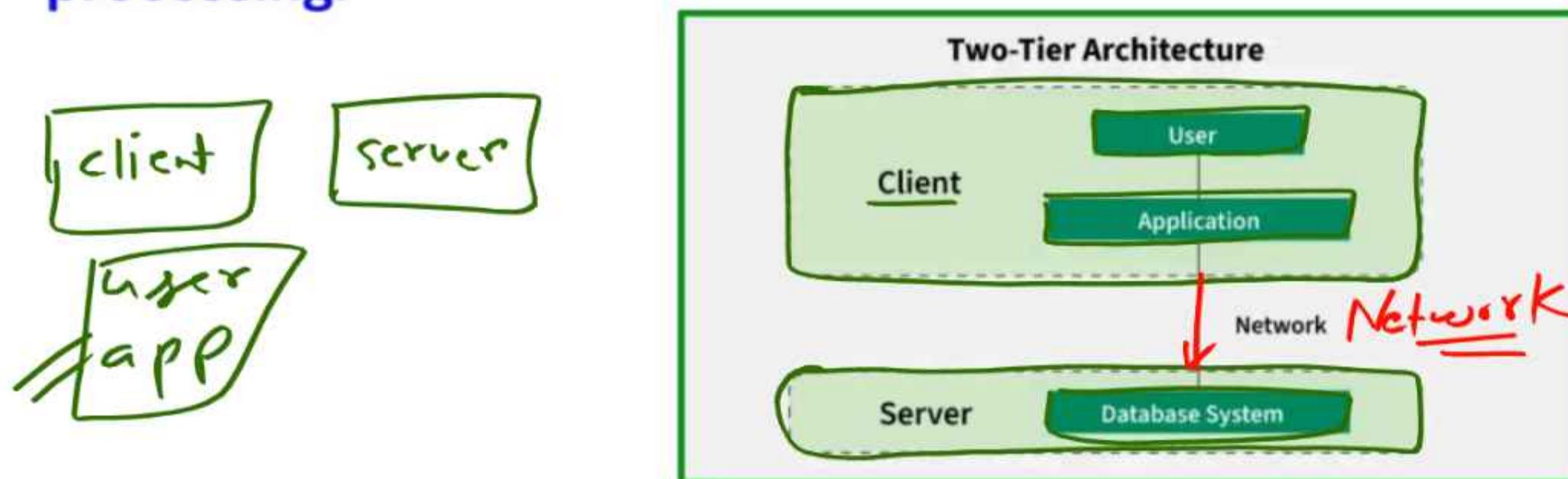
- कम सिक्योरिटी, स्केलेबल नहीं।
- Low security, not scalable.





## ② 2-Tier Architecture (Client-Server Model):

- ✓ क्लाइंट और सर्वर दो अलग-अलग मशीनों पर होते हैं।
- ✓ क्लाइंट-साइड पर GUI और एप्लिकेशन रन होती है, जबकि सर्वर डेटा प्रोसेसिंग करता है।
- ✓ The client and server are on two different machines.
- ✓ The GUI and application run on the client-side, while the server performs data processing.





### विशेषताएँ (Features): ✓

- ✓ डेटाबेस को दूरस्थ (Remote) एक्सेस किया जा सकता है।
- ✓ फास्ट डेटा प्रोसेसिंग - क्लाइंट-साइड से रिक्वेस्ट भेजी जाती है, और सर्वर डेटा प्रोसेस करके रिजल्ट भेजता है।
- ✓ सीधा कनेक्शन - क्लाइंट सीधे सर्वर से बात करता है।
- ✓ The database can be accessed remotely.
- ✓ Fast data processing – Requests are sent from the client-side, and the server processes the data and sends the results.
- ✓ Direct connection – The client talks directly to the server.

### उदाहरण (Examples): ✓

- Library Management System, Online Banking Systems, और छोटे बिजनेस एप्लिकेशन।



### फायदे (Advantages):

- ✓ सस्ता और सरल – 3-Tier की तुलना में आसान और कम महंगा।
- ✓ तेज़ एक्सेस – क्लाइंट सीधे डेटाबेस से जुड़ता है।
- ✓ इंस्टॉलेशन और अपग्रेड आसान – क्योंकि केवल **दो लेयर्स** होती हैं।
- ✓ Cheaper and simpler – Easier and less expensive than 3-Tier.
- ✓ Faster access – Client connects directly to the database.
- ✓ Easy installation and upgrade – as there are only two layers.

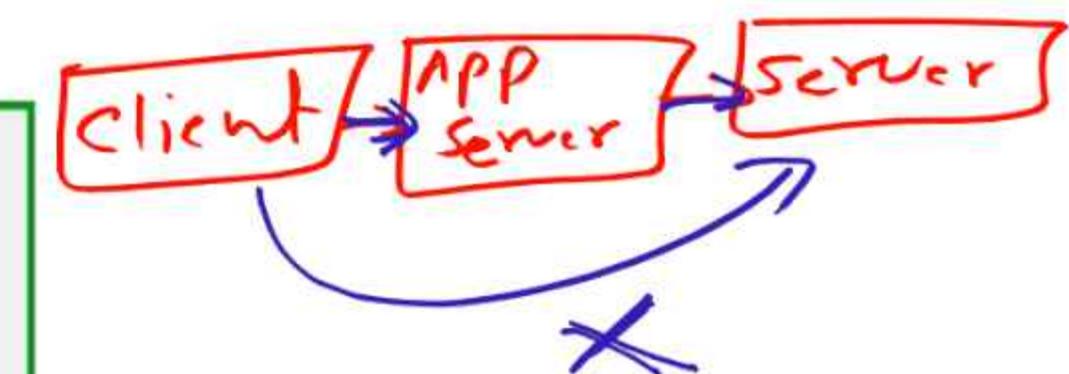
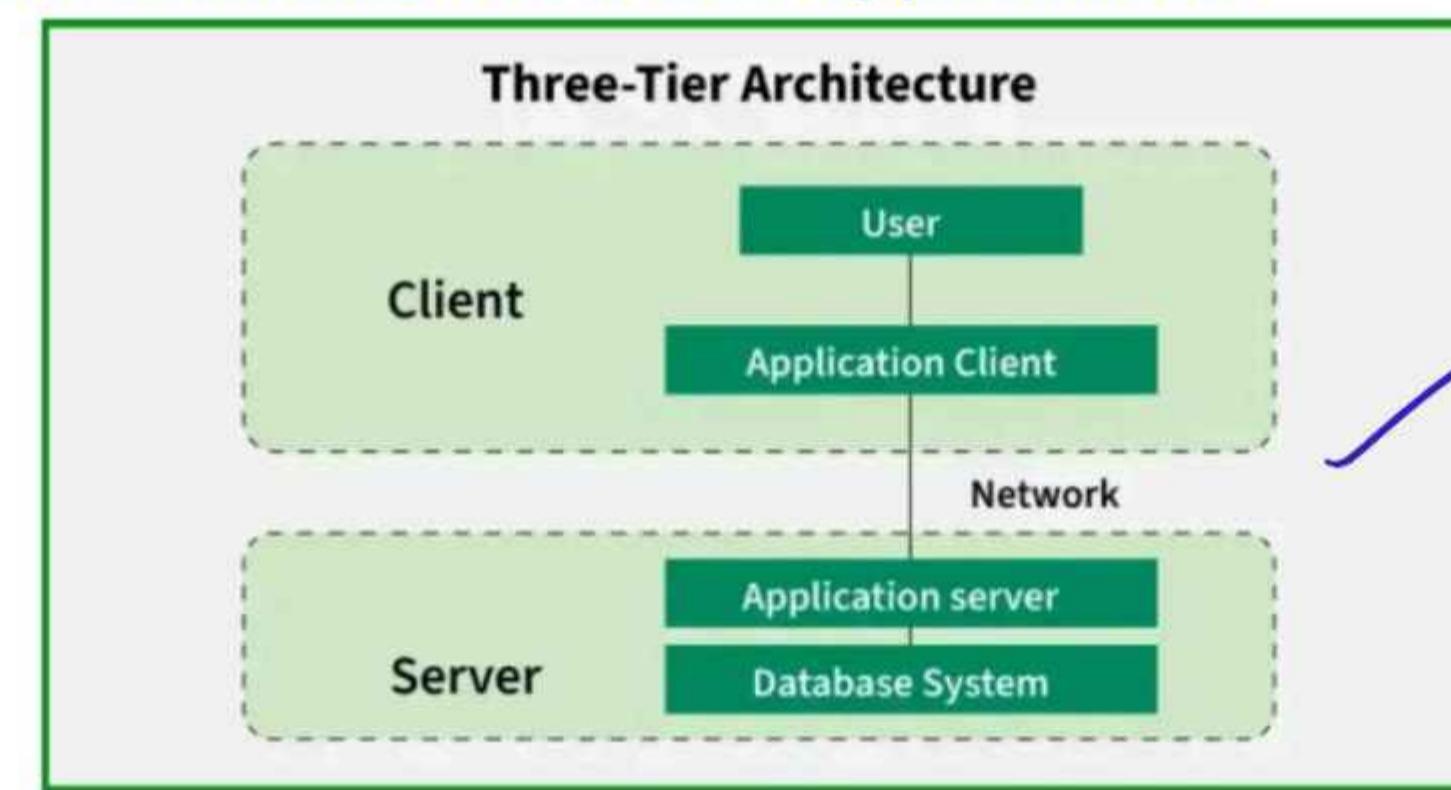
### Disadvantages:

- ज्यादा क्लाइंट होने पर **लोड बढ़ जाता है।**
- The load increases if there are more clients.



### ③ 3-Tier Architecture (Web-Based Architecture):

- ✓ क्लाइंट और सर्वर के बीच एक मिडलवेयर (Application Server) मौजूद होता है।
- ✓ क्लाइंट → Application Server → Database Server के बीच कम्युनिकेशन होता है।
- ✓ बड़े और स्केलेबल वेब-आधारित एप्लिकेशन में उपयोग किया जाता है।
- ✓ A middleware (Application Server) exists between the client and the server
- ✓ Communication takes place between Client → Application Server → Database Server.
- ✓ Used in large and scalable web-based applications.





### विशेषताएँ (Features):

- ✓ बेहतर सिक्योरिटी - क्लाइंट सीधे डेटाबेस से कनेक्ट नहीं होता।
- ✓ स्केलेबल सिस्टम - अधिक क्लाइंट्स को आसानी से सपोर्ट कर सकता है।
- ✓ डेटा इंटीग्रिटी - डेटा के करप्ट होने की संभावना कम होती है।
- ✓ Better security - The client does not connect directly to the database.
- ✓ Scalable system - Can easily support more clients.
- ✓ Data integrity - Data is less likely to be corrupted.

### उदाहरण (Examples):

✓ E-Commerce Websites (Amazon, Flipkart), Social Media Apps (Facebook, Instagram)।



### फायदे (Advantages):

- ✓ सिक्योरिटी बेहतर - क्लाइंट और डेटाबेस के बीच डायरेक्ट कनेक्शन नहीं होता।
- ✓ स्केलेबिलिटी - ज्यादा यूज़र्स को हैंडल कर सकता है।
- ✓ डेटा इंटीग्रिटी - डेटा भ्रष्टाचार (corruption) कम होता है।
- ✓ Better security - There is no direct connection between the client and the database.
- ✓ Scalability - Can handle more users.
- ✓ Data integrity - Data corruption is less.

### Disadvantages:

- सेटअप कॉम्प्लेक्स, कॉस्ट ज्यादा।
- Setup is complex, cost is high.



## Differences:

फीचर (Feature)	1-Tier	2-Tier	3-Tier
लेवल्स (Levels)	केवल एक मशीन पर सबकुछ (Everything on Single Machine)	Client + Database Server	Client + App Server + DB Server
सिक्योरिटी (Security)	बहुत कम Low	मीडियम Medium	बहुत ज्यादा High
स्केलेबिलिटी (Scalability)	नहीं No	Limited लिमिटेड	High हाई
यूज़ (Use Case)	Personal/ testing पर्सनल/टेस्टिंग	small business. छोटे नेटवर्क एप्स	Big Business बड़े वेब/एंटरप्राइज एप्लिकेशन
कॉस्ट (Cost)	Low लो	medium मीडियम	High हाई

Ques: Three tier Architecture त्रितीय तीर का व्यापक विवर

↳ 25%

Thanks For

Watching  
Semester  
Adda



LIKE



SHARE

COMMENT



Subscribe





## Unit-2 : Data Model and Keys

## Syllabus :

## UNIT2: Data Model and Keys:

3 Question → 15 marks

(8 Periods)

Define data model, Data Models : Network Model, Hierarchical Model, E-R Model, Advantage & Disadvantages of each Data Model, Concept of Keys (Primary, Candidate, Super, Foreign), Constraints, Strong Entity Set and Weak Entity Set.

5 marks 200%difference 100%



## → डेटा मॉडल्स (Data Models):

- Data Model एक **framework** है, जो यह बताता है कि database में data को कैसे organize, store और manage किया जाता है।
- A data model is a framework that describes how data is organized, stored, and managed in a database.
- यह एक structure provide करता है, जिसमें data items, उनके relationships, और constraints define किए जाते हैं।
- It provides a structure that defines data items, their relationships, and constraints.
- यह Users और Developers को यह समझने में मदद करता है कि डेटा को डेटाबेस में कैसे organize किया गया है।
- It helps users and developers understand how data is organized in a database.



→ डेटा मॉडल्स के प्रकार (Types of Data Models): ✓

- ① हायरार्किकल डेटा मॉडल (Hierarchical Data Model)
- ② नेटवर्क डेटा मॉडल (Network Data Model)
- ③ रिलेशनल डेटा मॉडल (Relational Data Model) → Unit - 3
- ④ एन्टिटी - रिलेशनशिप मॉडल Entity-Relationship Model (E-R Model)
- ⑤ Object-Oriented Data Model X



## ① हायरार्किकल डेटा मॉडल (Hierarchical Data Model):

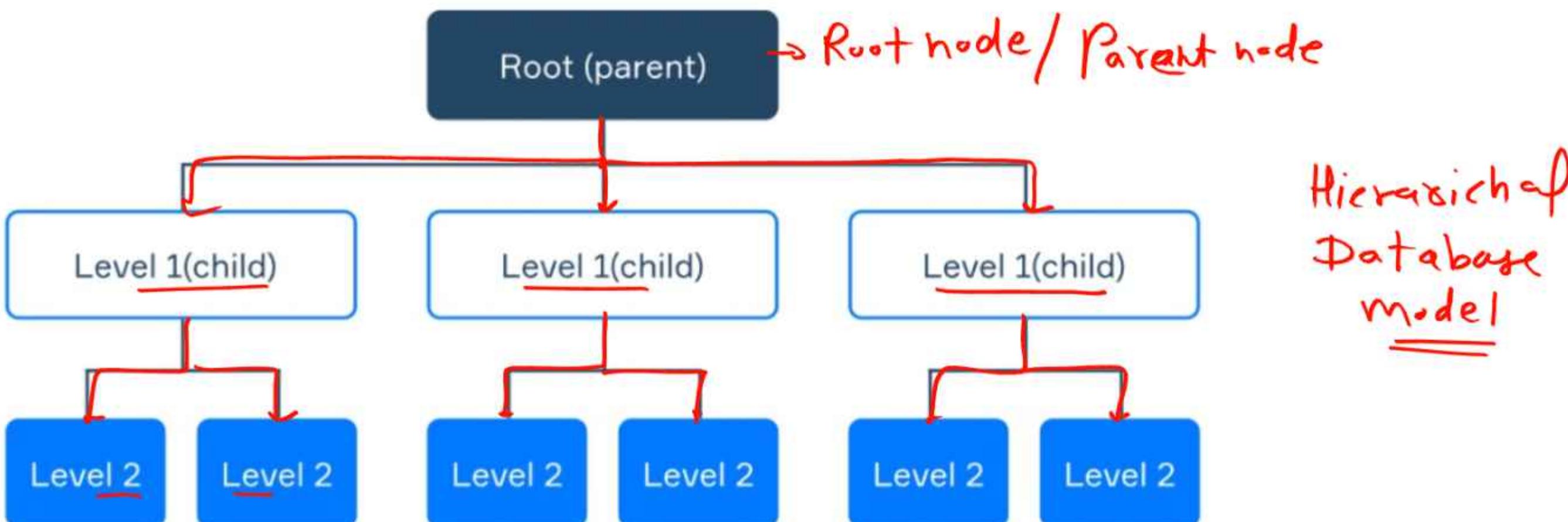
- हायरार्किकल डेटा मॉडल में डेटा को पेड़ (Tree) संरचना में Store किया जाता है, जिसमें पैरेंट-चाइल्ड (Parent-Child) relation होते हैं। इसमें प्रत्येक पैरेंट (Parent) नोड के एक या अधिक चाइल्ड (Child) नोड्स हो सकते हैं, लेकिन प्रत्येक चाइल्ड का केवल एक ही पैरेंट हो सकता है।
- In the hierarchical data model, data is stored in a tree structure, which has a parent-child relationship. In this, each parent node can have one or more child nodes, but each child can have only one parent.

### संरचना (Structure)

- डेटा को Tree-like structure में organize किया जाता है।
- रूट (Root) नोड top पर होता है, और उसके नीचे कई सब-नोड्स (Child Nodes) होते हैं।
- डेटा को टॉप-डाउन (Top-Down) अप्रौच में एक्सेस किया जाता है।
- Data is organized in a tree-like structure.
- The root node is at the top, and there are many child nodes below it.
- Data is accessed in a top-down approach.



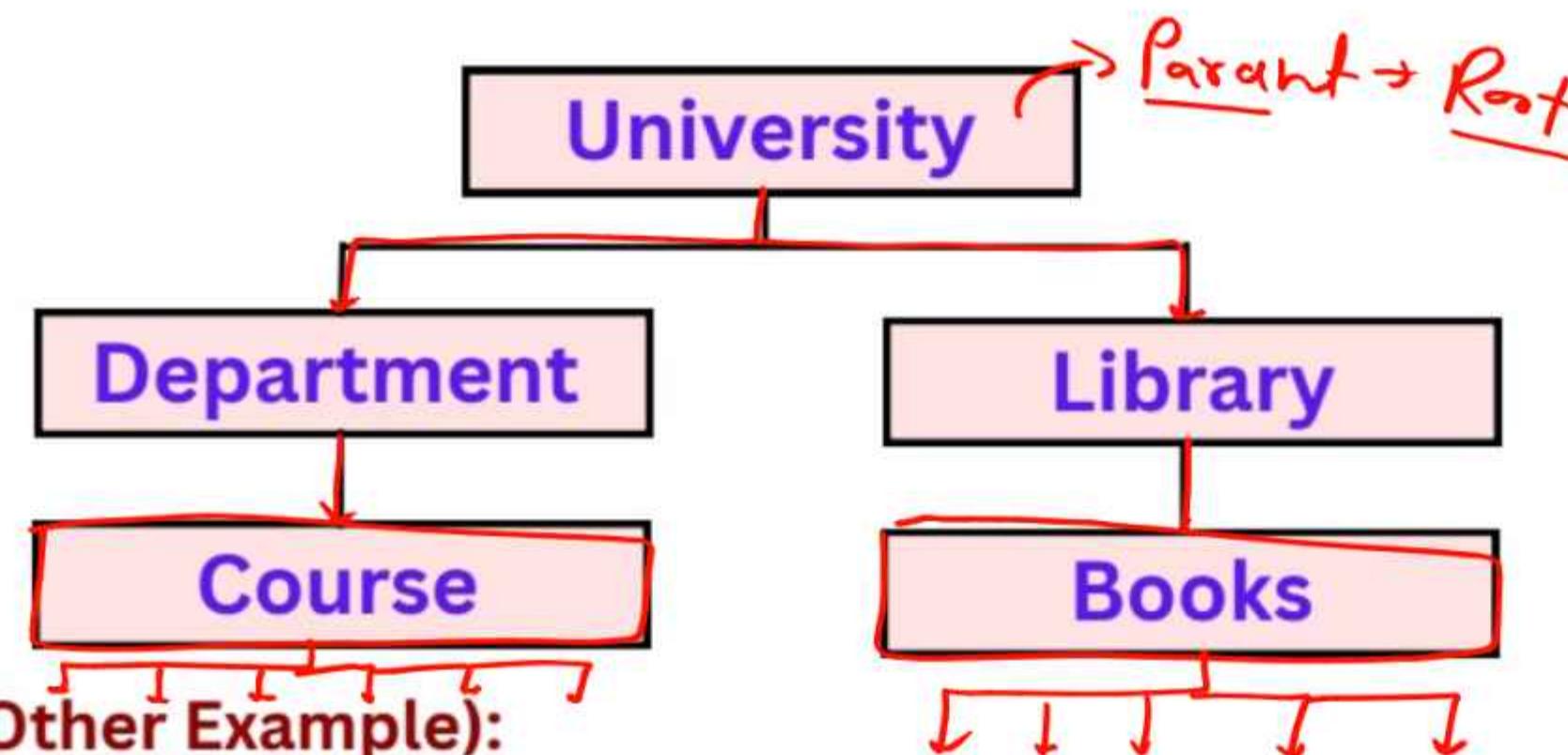
## The Hierarchical Database Model





उदाहरण (Example): ✓

- यूनिवर्सिटी डेटाबेस का हायरार्किकल मॉडल / Hierarchical Model of University Database:



❖ अन्य उदाहरण (Other Example):

✓ फ़ाइल सिस्टम (File System) ✓

✓ IBM Information Management System (IMS)



## लाभ (Advantages) -

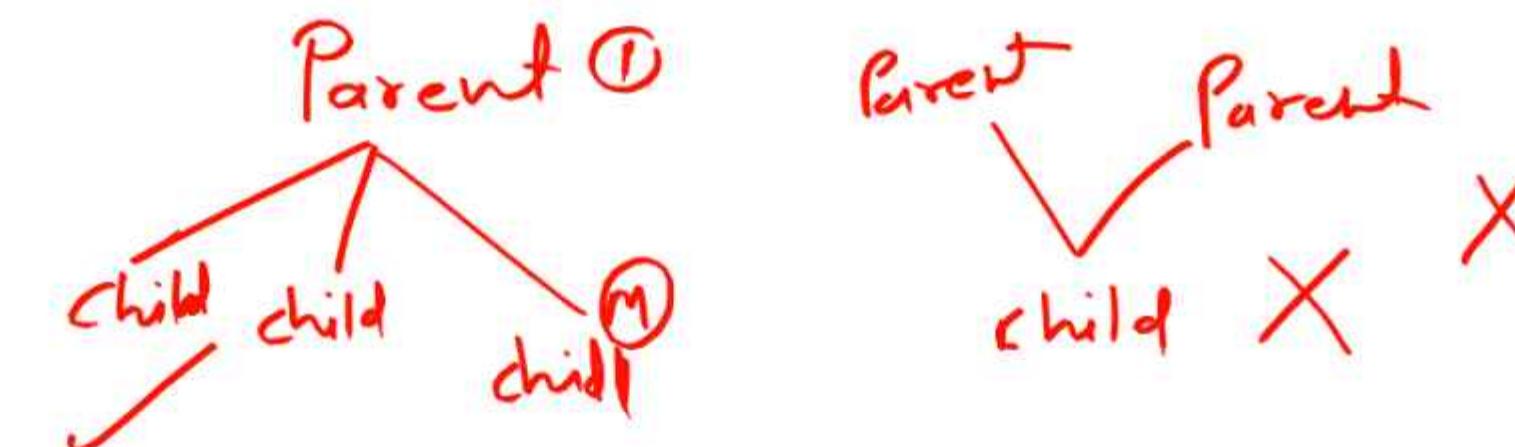
- ✓ तेज़ डेटा एक्सेस: डेटा को pre-defined पथों के माध्यम से आसानी से एक्सेस किया जा सकता है।
  - ✓ स्पष्ट संबंध: पैरेंट-चाइल्ड relationships clear होते हैं, जिससे डेटा को समझना आसान होता है।
  - ✓ डेटा इंटीग्रिटी: चाइल्ड नोड का केवल एक ही पैरेंट होता है, जिससे डेटा की अखंडता (Integrity) बनी रहती है।
- 
- ✓ Faster data access: Data can be easily accessed through pre-defined paths.
  - ✓ Clear relationships: Parent-child relationships are clear, making data easier to understand.
  - ✓ Data integrity: A child node has only one parent, ensuring data integrity.

नहीं X → Relation of  
model



### नुकसान (Disadvantages)

- ✗ कम लचीलापन: यदि डेटाबेस की संरचना बदलनी हो, तो पूरे डेटा मॉडल को फिर से डिज़ाइन करना पड़ता है।
- ✗ जटिलता: बड़े डेटाबेस में डेटा का Management कठिन हो सकता है।
- ✗ एक-से-अनेक संबंध ही संभव: कई-से-कई (Many-to-Many) संबंधों को मैनेज करना मुश्किल होता है।
- ✗ Less flexibility: If the structure of the database needs to be changed, the entire data model has to be redesigned.
- ✗ Complexity: Managing data in a large database can be difficult.
- ✗ Only one-to-many relationships are possible: Many-to-many relationships are difficult to manage.



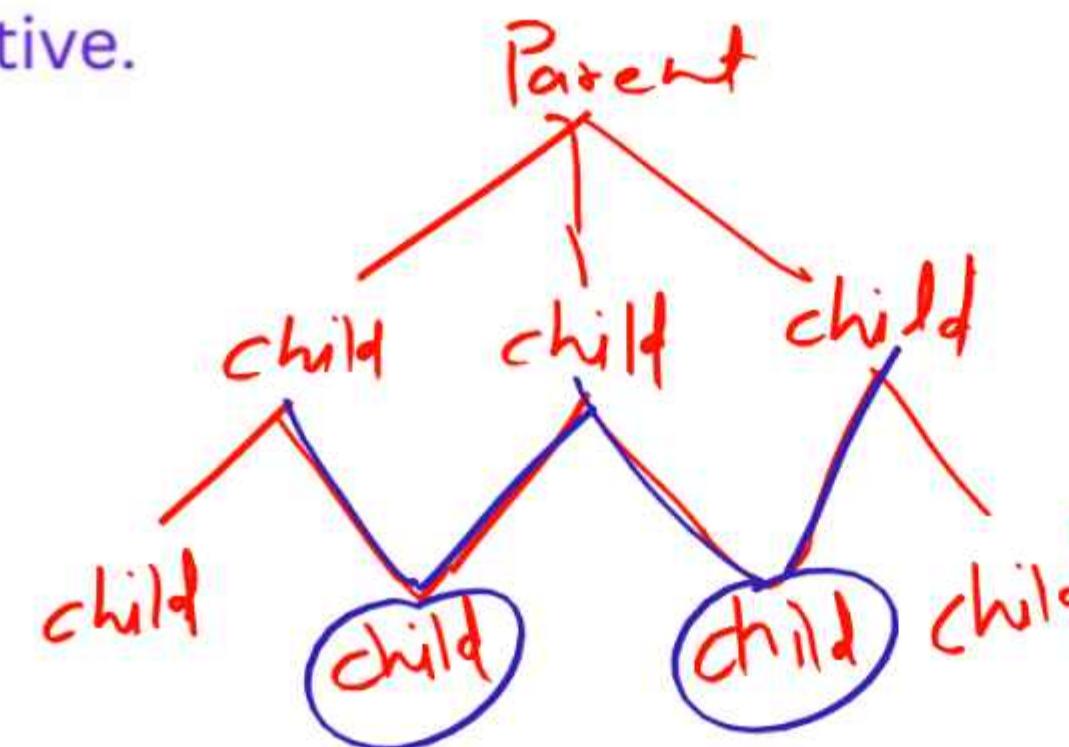
**Note:**

- ▶ हायरार्किकल डेटा मॉडल **छोटे और स्थिर डेटा सेट्स** के लिए उपयुक्त है, लेकिन **बड़े** और **जटिल** डेटाबेस सिस्टम के लिए कम प्रभावी होता है।
- ▶ Hierarchical data model is suitable for small and static data sets, but is less effective for large and complex database systems.
- ▶ आजकल, रिलेशनल डेटा मॉडल (Relational Data Model) अधिक लोकप्रिय है क्योंकि यह अधिक लचीला और उपयोग में आसान है।
- ▶ Nowadays, Relational Data Model is more popular because it is more flexible and easy to use.



## ② नेटवर्क डेटा मॉडल (Network Data Model): ✓

- नेटवर्क डेटा मॉडल में डेटा को ग्राफ (Graph) structure में store किया जाता है, जिसमें कई-से-कई (Many-to-Many) relation संभव होते हैं।
- In the network data model, data is stored in a graph structure, in which many-to-many relationships are possible.
- इसमें डेटा को रिकॉर्ड्स (Records) और लिंक (Links) के रूप में store किया जाता है, जिससे डेटा अधिक flexible और effective हो जाता है।
- In this, data is stored in the form of records and links, which makes data more flexible and effective.



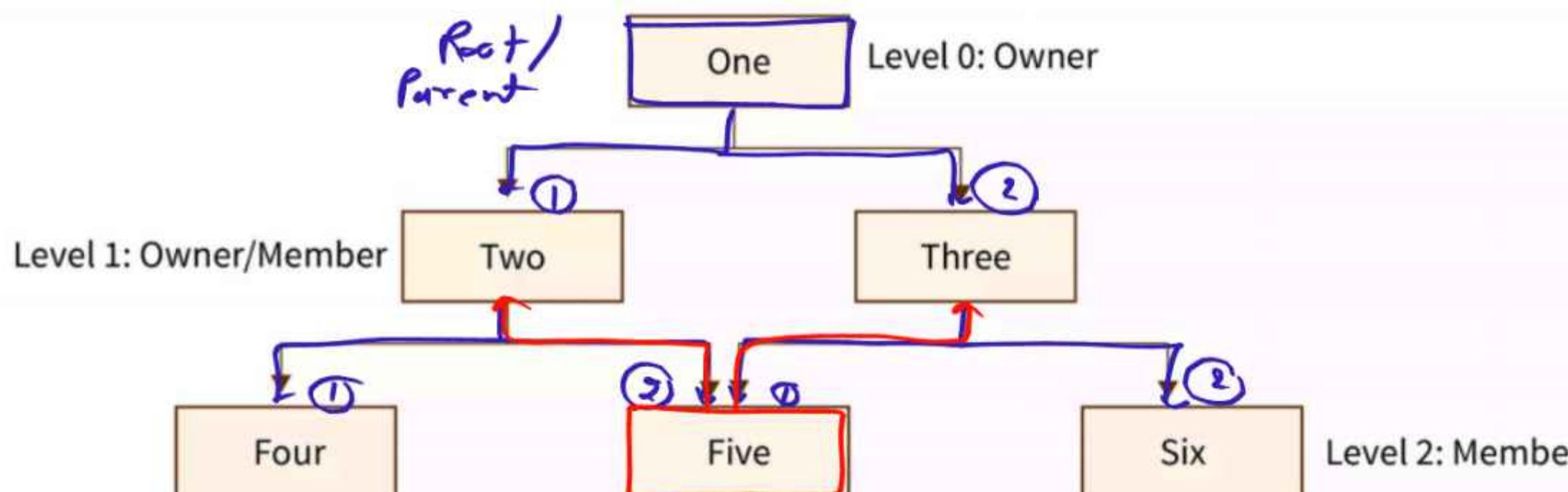


### संरचना (Structure):

- ◆ डेटा को ग्राफ़ based structure में organize किया जाता है।
- ◆ प्रत्येक नोड कई पैरेंट्स और चाइल्ड्स हो सकते हैं।
- ◆ डेटा को रिकॉर्ड्स और लिंक के रूप में संग्रहीत किया जाता है।
- ◆ Data is organized in a graph-based structure.
- ◆ Each node can have multiple parents and children.
- ◆ Data is stored as records and links.



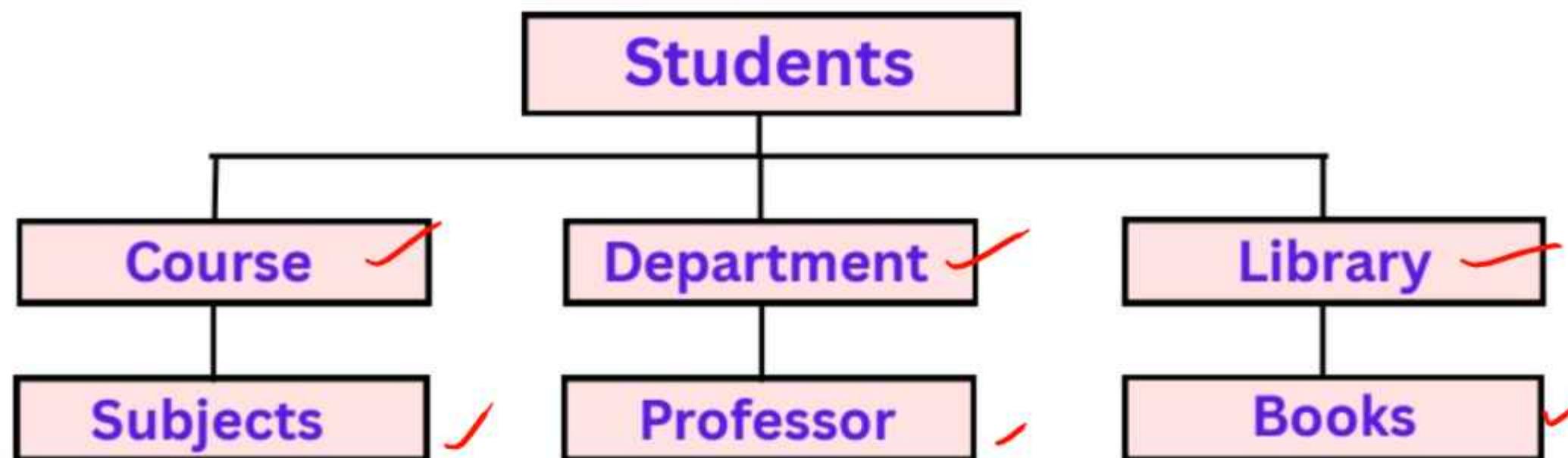
## नेटवर्क डेटा मॉडल (Network Data Model)





उदाहरण (Example): ✓

- यूनिवर्सिटी डेटाबेस का नेटवर्क मॉडल / Network Model of University Database:



❖ अन्य उदाहरण (Other Example): ✓

- ✓ रेलवे आरक्षण प्रणाली (Railway Reservation System) ✓
- ✓ इंटरनेट लिंक संरचना (Internet Link Structure)

Network  
data  
model  
// ==



### लाभ (Advantages):

- ✓ कई-से-कई संबंध संभव: यह मॉडल अधिक जटिल संबंधों को संभाल सकता है।
- ✓ तेज़ डेटा एक्सेस: ग्राफ संरचना के कारण डेटा जल्दी एक्सेस किया जा सकता है।
- ✓ बेहतर प्लेक्सिबिलिटी: इसमें डेटा को विभिन्न तरीकों से एक्सेस और अपडेट किया जा सकता है।
- ✓ Many-to-many relationships possible: This model can handle more complex relationships.
- ✓ Faster data access: Data can be accessed quickly due to the graph structure.
- ✓ Better flexibility: In this, data can be accessed and updated in different ways.

1 to M  
fast / flexible



### नुकसान (Disadvantages): ✓

- ✗ **जटिलता:** हायरार्किकल मॉडल की तुलना में नेटवर्क मॉडल अधिक complex होता है।
- ✗ **मुश्किल डेटा अपडेट:** डेटा को अपडेट करने के लिए कई लिंक को भी अपडेट करना पड़ता है।
- ✗ **अधिक मेंटेनेंस की आवश्यकता:** डेटा relationships को बनाए रखना मुश्किल होता है।
- ✗ **Complexity:** Network model is more complex than hierarchical model.
- ✗ **Difficult data update:** To update data, many links also have to be updated.
- ✗ **More maintenance required:** Data relationships are difficult to maintain.

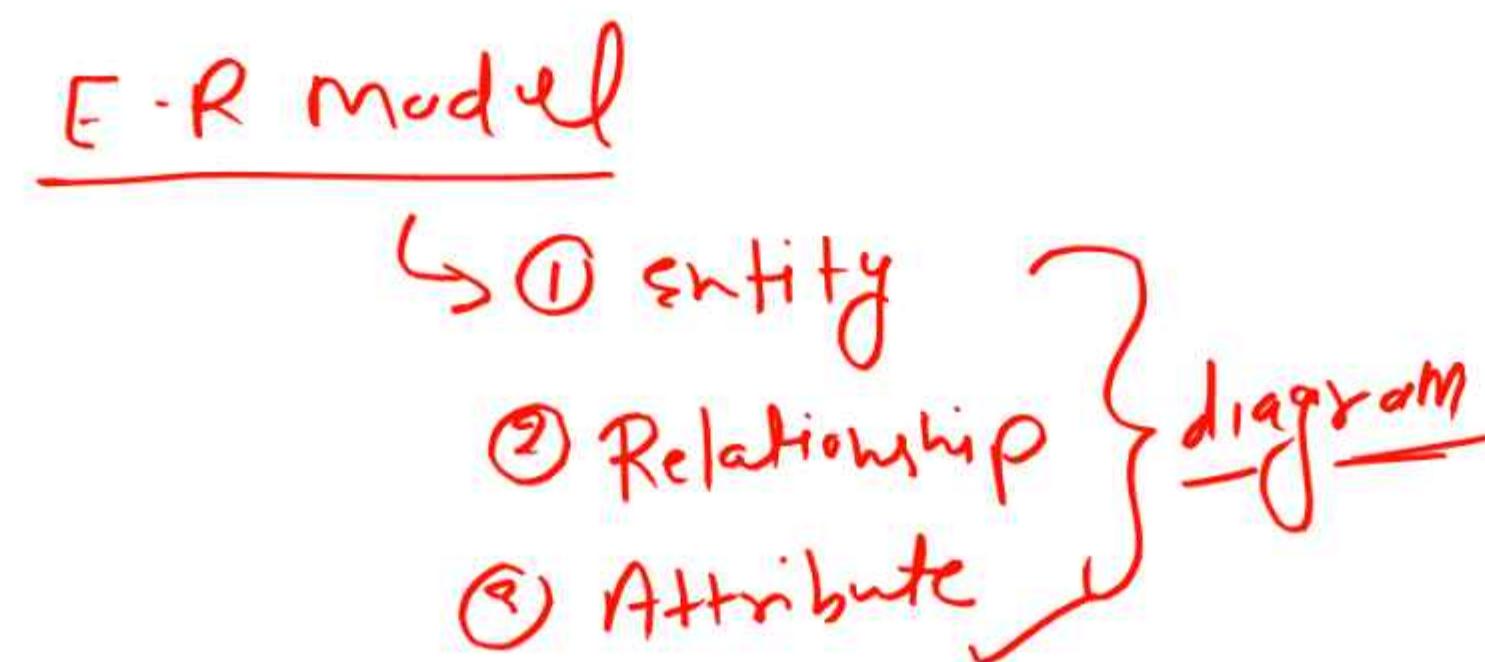
**Note:**

- नेटवर्क डेटा मॉडल **जटिल डेटा संबंधों** वाले सिस्टम के लिए उपयोगी है, जैसे कि बैंकिंग सिस्टम, ट्रांसपोर्टेशन सिस्टम आदि।
- Network data model is useful for systems with complex data relationships, such as banking systems, transportation systems, etc.
- हालांकि, **रिलेशनल डेटा मॉडल (Relational Data Model)** के आने के बाद इसका उपयोग कम हो गया है, क्योंकि रिलेशनल मॉडल अधिक सरल और प्रभावी है।
- However, its use has declined after the advent of Relational Data Model, as the relational model is simpler and more effective.



### ③ E-R मॉडल (Entity-Relationship Model): ✓

- E-R मॉडल (Entity-Relationship Model) एक डायग्रामेटिक तकनीक है, जिसका उपयोग डेटाबेस डिज़ाइन करने के लिए किया जाता है। इसमें डेटा को एंटिटी (Entity), एट्रिब्यूट (Attribute) और रिलेशनशिप (Relationship) के रूप में दर्शाया जाता है।
- E-R model (Entity-Relationship Model) is a diagrammatic technique used to design databases. In this, data is represented in the form of entity, attribute and relationship.





## 1. एंटिटी (Entity): /

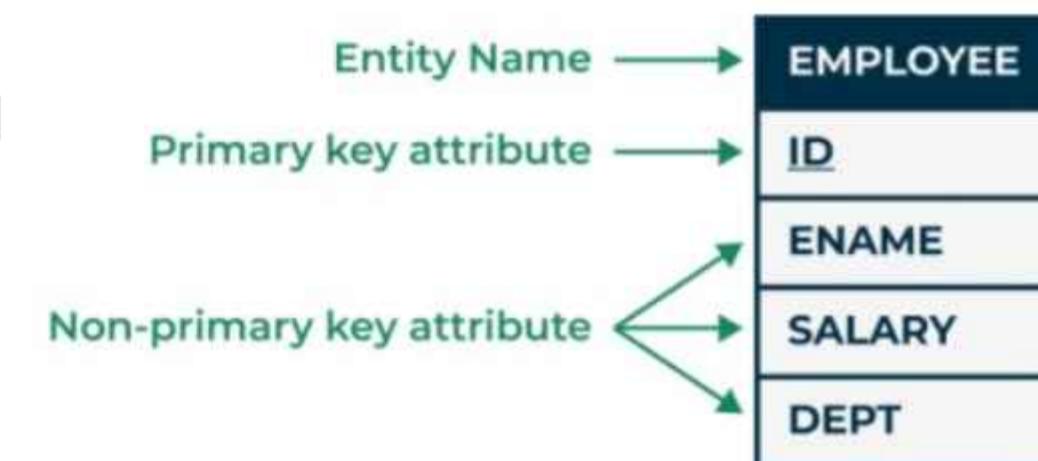
- एंटिटी वह ऑब्जेक्ट या चीज़ होती है, जिसके बारे में डेटा store किया जाता है।
- An entity is an object or thing about which data is stored.

### उदाहरण (Example):

- एक कॉलेज में प्रत्येक छात्र (Student) एक एंटिटी है। ✓
- फैकल्टी (Faculty) को उसके Faculty\_ID द्वारा पहचाना जाता है।
- छात्र (Student) को उसके Roll\_No द्वारा पहचाना जाता है। ✓
- कोर्स (Course) को Course\_ID द्वारा पहचाना जाता है।

### Example:

- Each student in a college is an entity.
- Faculty is identified by its Faculty\_ID.
- Student is identified by its Roll No.
- Course is identified by Course\_ID.





## 2. एट्रिब्यूट (Attribute): ✓

- ऐटिटी की विशेषताएँ या गुण होते हैं। ✓
- It is attributes or properties of Entity.

## प्रकार (Types):

- सिंपल (Simple): जैसे नाम, उम्र। Name, Age
- कम्पोजिट (Composite): जैसे पूरा नाम (First Name + Last Name)।
- मल्टी-वैल्यूड (Multi-valued): जैसे फोन नंबर (एक से अधिक हो सकते हैं)।
- डेराइव्ड (Derived): जैसे उम्र (Date of Birth से निकाला जाता है)। ↳ Mob No, email

Types: ~~Ex:~~ age, percentage



- Simple: e.g. Name, Age.
- Composite: e.g. Full Name (First Name + Last Name).
- Multi-valued: e.g. Phone Number (can be more than one).
- Derived: e.g. Age (derived from Date of Birth).



### 3. रिलेशनशिप (Relationship): ✓

- यह विभिन्न एंटिटीज़ के बीच relation को दिखाता है।
- It shows the relationship between different entities.

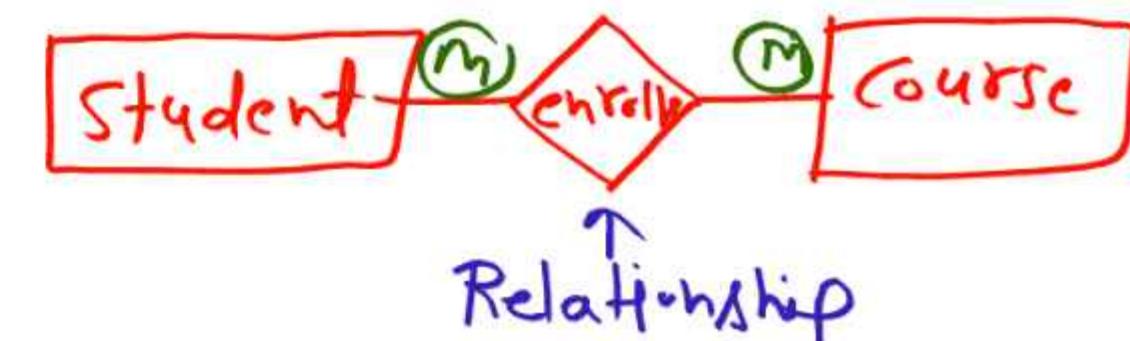
#### प्रकार (Types): ✓

- 1:1 (One-to-One):** एक छात्र के पास एक आईडी कार्ड।
- 1:M (One-to-Many):** एक शिक्षक कई छात्रों को पढ़ाता है।
- M:N (Many-to-Many):** एक छात्र कई कोर्स कर सकता है और एक कोर्स कई छात्रों के लिए उपलब्ध हो सकता है।

1 to 1 , 1 to m , m to 1 , m to m

#### Types:

- 1:1 (One-to-One):** One student has one ID card.
- 1:M (One-to-Many):** One teacher teaches many students.
- M:N (Many-to-Many):** One student can take many courses and one course can be available to many students.





### लाभ (Advantages):

- ✓ डेटाबेस डिज़ाइन को clear करता है। ✓
- ✓ डेटाबेस को विज़ुअल रूप से समझने में मदद करता है। ✓
- ✓ डेटाबेस को बेहतर तरीके से structure करने में मदद करता है। ✓
- ✓ Clarifies database design.
- ✓ Helps to understand database visually.
- ✓ Helps to structure database in a better way.

### नुकसान (Disadvantages):

- ✗ Complex डेटा रिलेशनशिप को संभालना difficult हो सकता है।
- ✗ E-R मॉडल को रिलेशनल स्कीमा में बदलने में अधिक समय लग सकता है।
- ✗ Complex data relationships may be difficult to handle.
- ✗ Converting the E-R model to a relational schema may take more time.



## E-R डायग्राम (E-R Diagram):

- ER डायग्राम (ER Diagram) एक दृश्य प्रतिनिधित्व (Visual Representation) है, जो डेटाबेस में मौजूद एंटीटीज़ (Entities), उनके गुण (Attributes), और उनके बीच के संबंध (Relationships) को दर्शाता है।
- ER Diagram is a visual representation that shows the entities in a database, their attributes, and the relationships between them.

## सरल शब्दों में (In simple words):

- ER डायग्राम डेटाबेस डिज़ाइन को आसान और समझने योग्य बनाता है।
- इसमें Entities , Attributes और Relationships को ग्राफिकल तरीके से दिखाया जाता है।
- ER diagram makes database design easy and understandable.
- It shows Entities , Attributes and Relationships in a graphical way.



### ER डायग्राम के लाभ (Advantages of ER Diagram):

- ✓ डेटाबेस डिज़ाइन को clear रूप से दिखाता है। ✓
- ✓ डेटा के बीच relation को आसानी से समझने में मदद करता है। ✓
- ✓ Large और complex जानकारी को ग्राफिकल रूप में सरल बनाता है।
- ✓ सिस्टम कोडिंग से पहले एक सही मॉडल तैयार करने में सहायता करता है।
- ✓ Shows the database design clearly.
- ✓ Helps to easily understand the relationship between data.
- ✓ Simplifies large and complex information in a graphical form.
- ✓ Helps to prepare a correct model before coding the system.



## नोटेशन (Notation):

- ER डायग्राम को समझने और सही तरीके से प्रस्तुत करने के लिए विभिन्न नोटेशन (Symbols) का उपयोग किया जाता है। ये नोटेशन Entities, Attributes और Relationships को दर्शने के लिए प्रयोग किए जाते हैं।
- To understand and represent ER diagrams correctly, various notations (symbols) are used. These notations are used to represent Entities, Attributes and Relationships.

### ① Entity:

- इसे Rectangle (आयत) से दर्शाया जाता है। यह एक वास्तविक दुनिया की चीज़ या व्यक्ति को दर्शाता है, जैसे Student, Teacher, Car आदि।
- It is represented by a rectangle. It represents a real-world thing or person, like Student, Teacher, Car, etc.

Entity

Rectangle (आयत)



### ② Weak Entity:

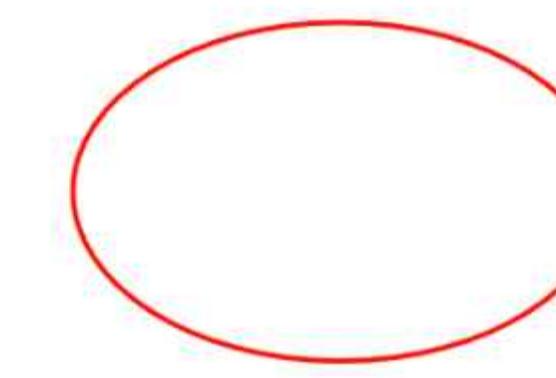
- इसे Double Rectangle (दोहरी आयत) से दर्शाया जाता है। इसे किसी अन्य एंटिटी के प्राइमरी की (Primary Key) की आवश्यकता होती है, जैसे Course\_Offering।
- It is represented by a Double Rectangle. It requires the primary key of another entity, such as Course\_Offering.



double  
Reatangle

### ③ Attribute:

- इसे Ellipse (अंडाकार) से दर्शाया जाता है। यह किसी एंटिटी की विशेषता को दर्शाता है, जैसे Name, Age, Salary।
- It is represented by an Ellipse. It represents the attribute of an entity, such as Name, Age, Salary.



ellips/oval



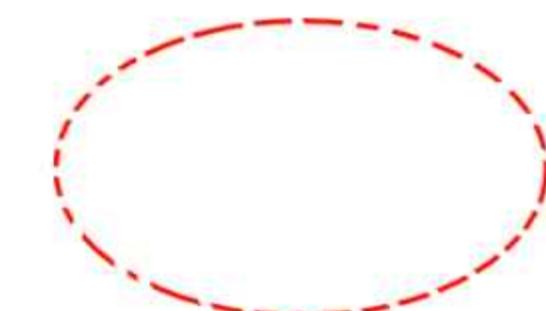
## ④ Primary Key:

- इसे रेखांकित (Underlined) किया जाता है। यह किसी एंटिटी को यूनिक रूप से पहचानता है, जैसे Student\_ID, Roll\_No।
- It is underlined. It uniquely identifies an entity, like Student\_ID, Roll\_No.

Roll No

## ⑤ Derived Attribute:

- इसे Dashed Ellipse (डैश वाली अंडाकार आकृति) से दर्शाया जाता है। इसे किसी अन्य डेटा से निकाला जाता है, जैसे Age (DOB से निकाला गया डेटा)।
- It is represented by a Dashed Ellipse. It is extracted from some other data, such as Age (data extracted from DOB).



e.g. age, percentage  
marks.

Dashed ellipses

Dashed oval



### ⑥ Multivalued Attribute:

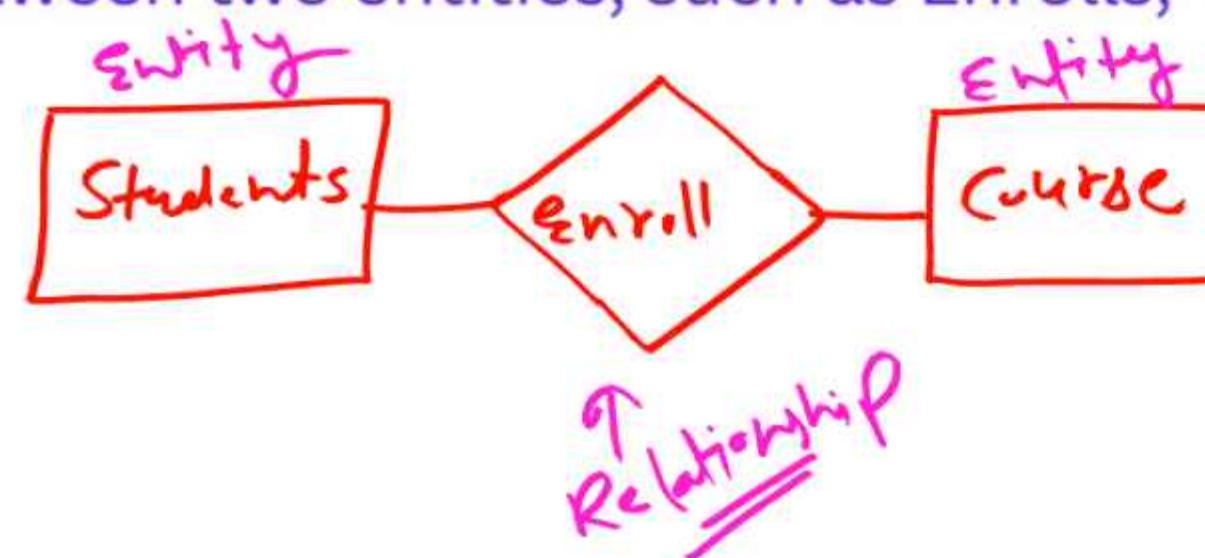
- इसे Double Ellipse (दोहरी अंडाकार आकृति) से दर्शाया जाता है। यह दर्शाता है कि एक एंटिटी के लिए एक से अधिक मूल्य हो सकते हैं, जैसे Phone Numbers।
- It is represented by a Double Ellipse. It indicates that there can be more than one value for an entity, such as Phone Numbers.

Ex: Email

ph. No.  
Multivalued  
Attribute

### ⑦ Relationship:

- इसे Diamond से दर्शाया जाता है। यह दो एंटिटीज़ के बीच संबंध को दिखाता है, जैसे Enrolls, Teaches।
- It is represented by a Diamond. It shows the relationship between two entities, such as Enrolls, Teaches.

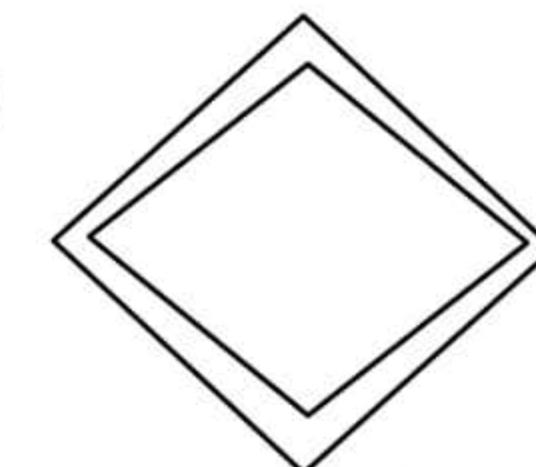


Relationship  
diamond box/  
decision making box



### ⑧ Weak Relationship:

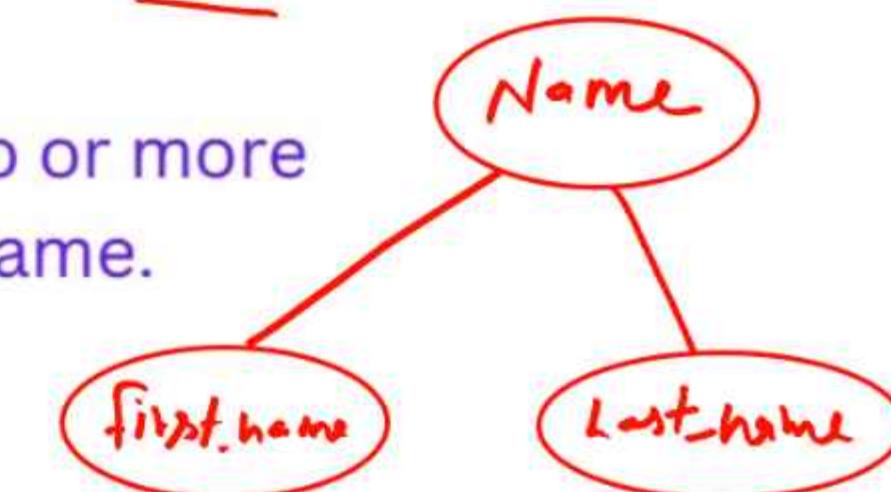
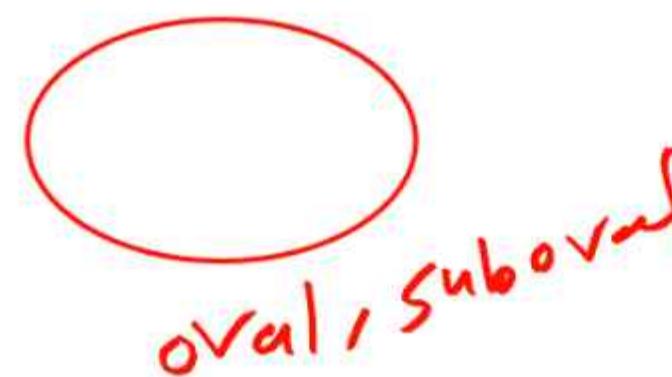
- इसे Double Diamond से दर्शाया जाता है। यह किसी निर्बल एंटिटी को जोड़ने के लिए उपयोग होता है, जैसे Has\_Dependency।
- It is represented by a Double Diamond. It is used to add a weak entity, such as Has\_Dependency.



double  
diamond

### ⑨ Composite Attribute:

- जब कोई attribute दो या अधिक भागों में विभाजित होती है, तो इसे दर्शने के लिए उपयोग होता है। उदाहरण के लिए, Full Name → First Name + Last Name। Ex: full Name, Address, message
- Used to represent when an attribute is split into two or more parts. For example, Full Name → First Name + Last Name.

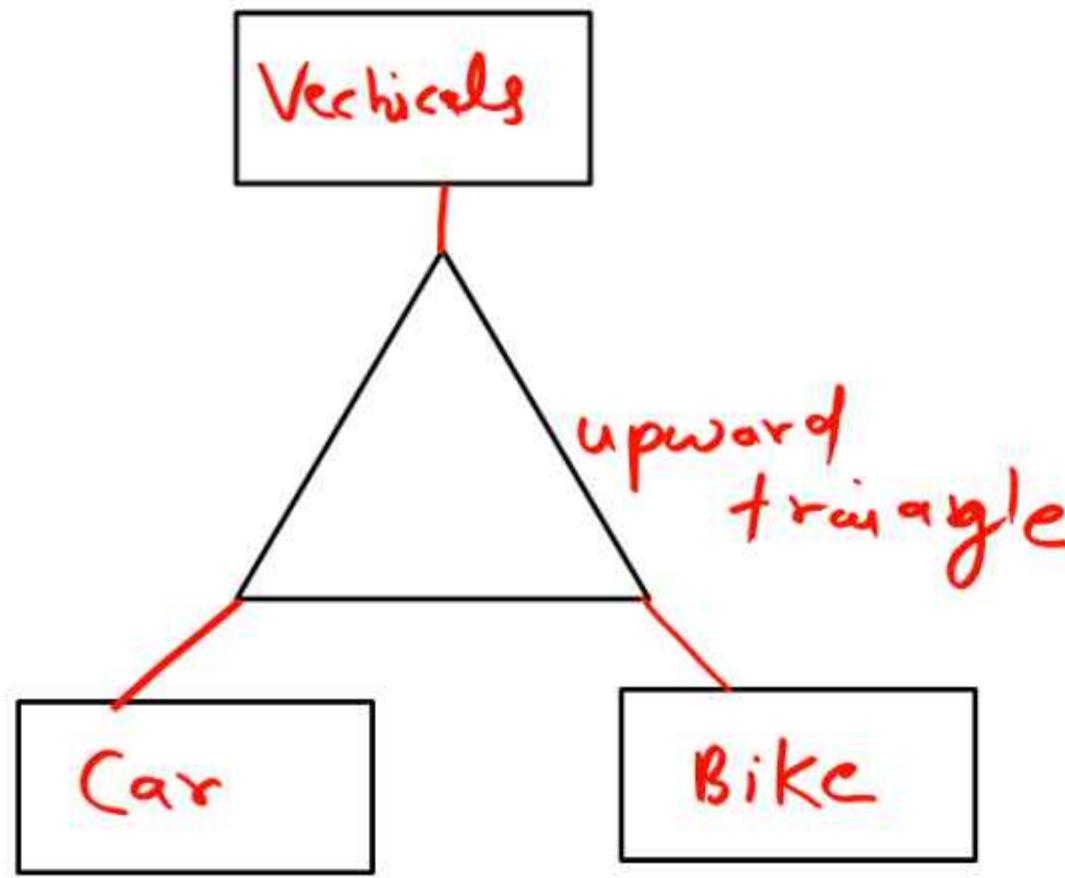




## 10 Generalization:

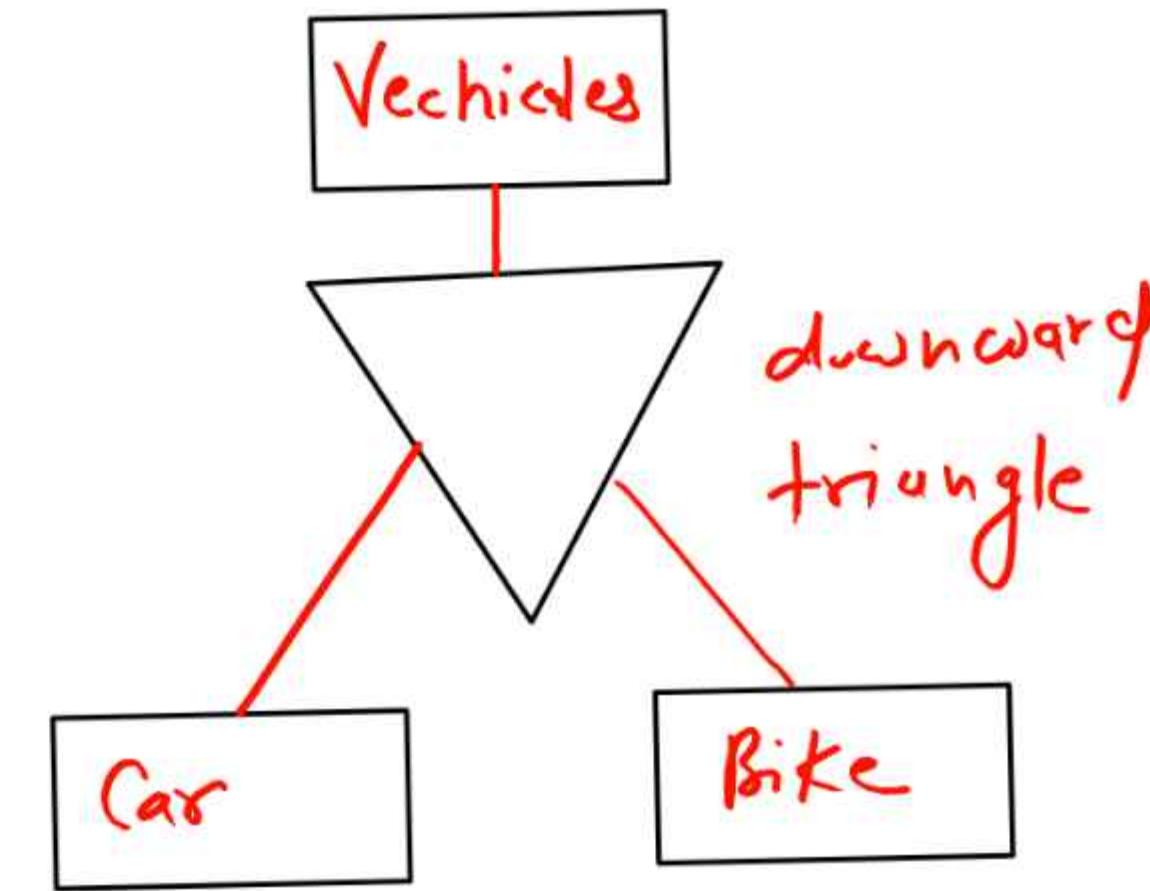
- Generalization वह प्रक्रिया है जिसमें एकाधिक Entities को मिलाकर एक उच्च-स्तरीय (Higher-Level) Entity बनाई जाती है।
- Generalization is the process in which multiple entities are combined to form a higher-level entity.
- यह Bottom-Up Approach (नीचे से ऊपर की ओर) होती है, जिसमें कई विशिष्ट (Specific) Entities को मिलाकर एक सामान्य (General) Entity बनाई जाती है।
- This is a bottom-up approach in which a general entity is created by combining several specific entities.
- इसे ▲ (ऊपर की ओर त्रिभुज) से दर्शाया जाता है। यह दर्शाता है कि विभिन्न एंटीटीज़ को एक सामान्य श्रेणी में समूहित किया जा सकता है, जैसे Vehicle → Car, Bike।
- It is represented by ▲ (upward triangle). It shows that different entities can be grouped into a common category, e.g. Vehicle → Car, Bike.

Ex: Generalization



Bottom-up approach

Ex: specialization



Top-Bottom approach



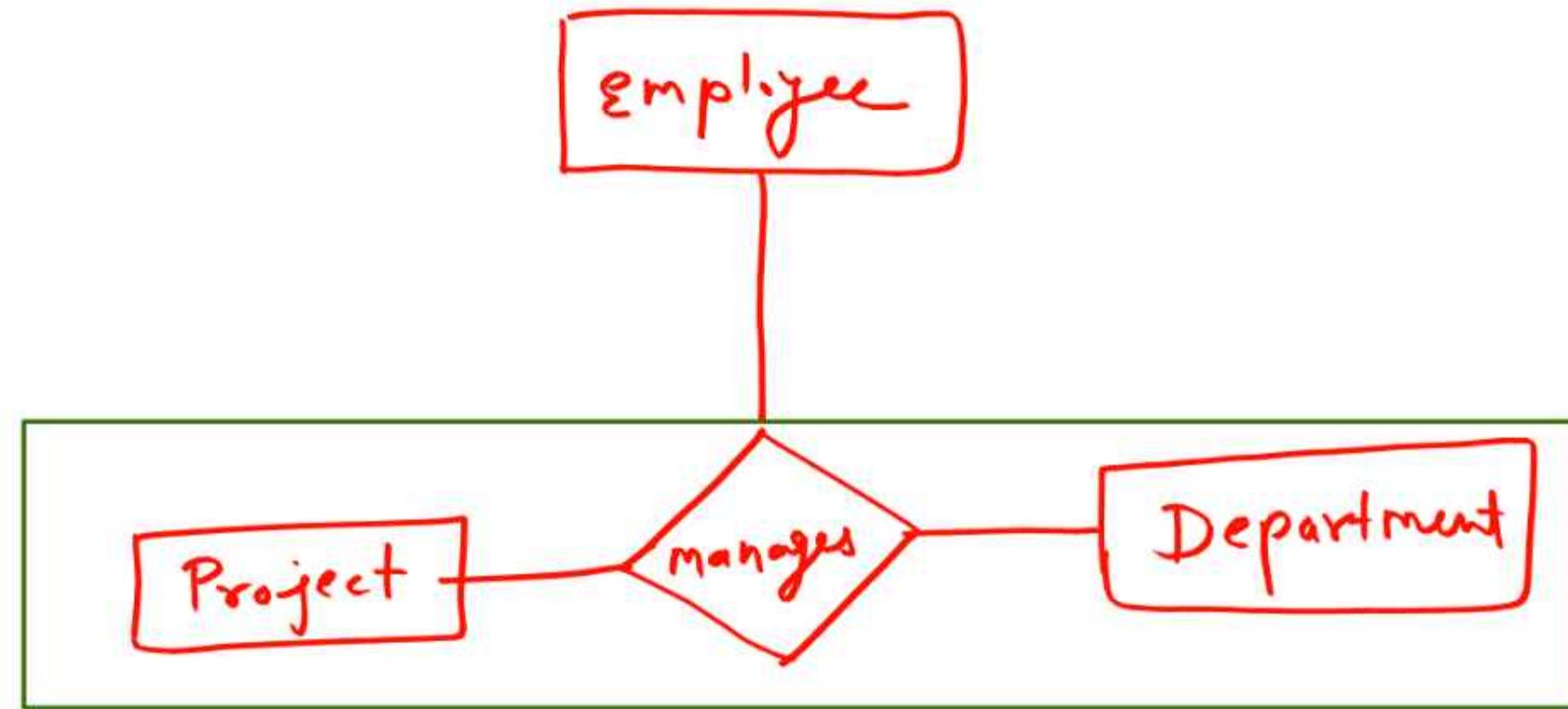
### 11. Specialization:

- Specialization वह प्रक्रिया है जिसमें **एक सामान्य Entity** को **तोड़कर अधिक विशेष Entities** में विभाजित किया जाता है।
- Specialization is the process in which a general entity is broken down into more specific entities.
- यह **Top-Down Approach** (ऊपर से नीचे की ओर) होती है, जिसमें **General Entity** को अधिक **Specific Sub-Entities** में विभाजित किया जाता है।
- This is a top-down approach, in which a general entity is divided into more specific sub-entities.
- इसे **▼ (नीचे की ओर त्रिभुज)** से दर्शाया जाता है। यह सामान्य एंटिटी को और अधिक विशेष प्रकार में विभाजित कرتा है, जैसे Employee → Teacher, Staff।
- It is represented by **▼ (downward triangle)**. It divides general entities into more specific types, such as Employee → Teacher, Staff.



## 12. Aggregation:

- Aggregation एक उन्नत अवधारणा (Advanced Concept) है, जिसका उपयोग तब किया जाता है जब हमें एक संबंध (Relationship) को दूसरी Entity के साथ एक संपूर्ण इकाई (Entity) के रूप में मॉडल करना होता है।
- Aggregation is an advanced concept which is used when we have to model a relationship with another entity as a complete entity.
- यह एक Higher-Level Abstraction प्रदान करता है, जहाँ एक संपूर्ण Relationship Set को दूसरी Entity के साथ जोड़ा जाता है।
- It provides a higher-level abstraction, where an entire relationship set is associated with another entity.
- इसे ER मॉडल में एक आयत (Rectangle) के अंदर एक Diamond (Diamond inside a Rectangle) द्वारा दर्शाया जाता है। जैसे Project-Manages-Department।
- When an entire relationship is treated as a single entity, it is represented as a relationship inside a larger diamond, such as Project-Manages-Department.



Aggregation

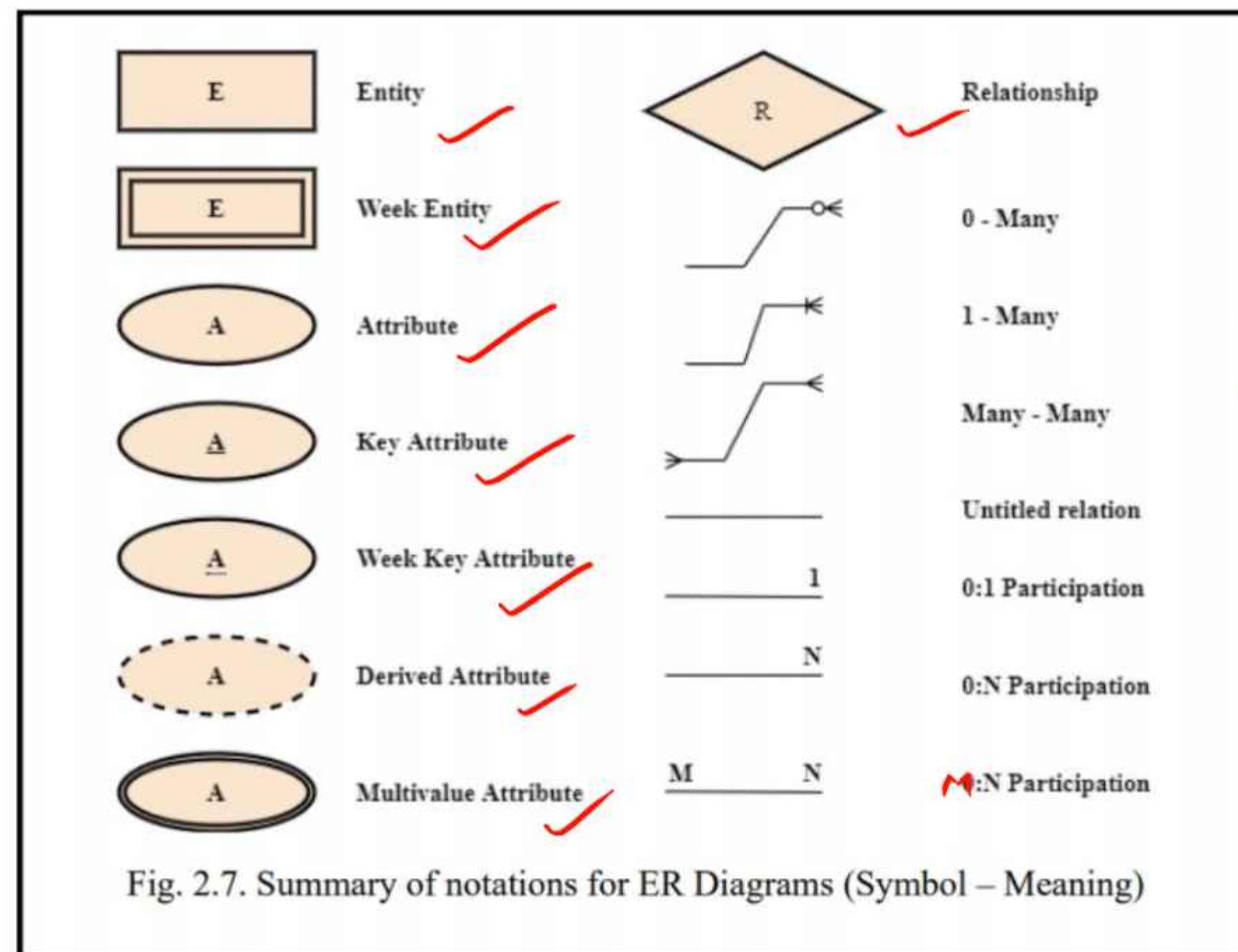


Fig. 2.7. Summary of notations for ER Diagrams (Symbol – Meaning)

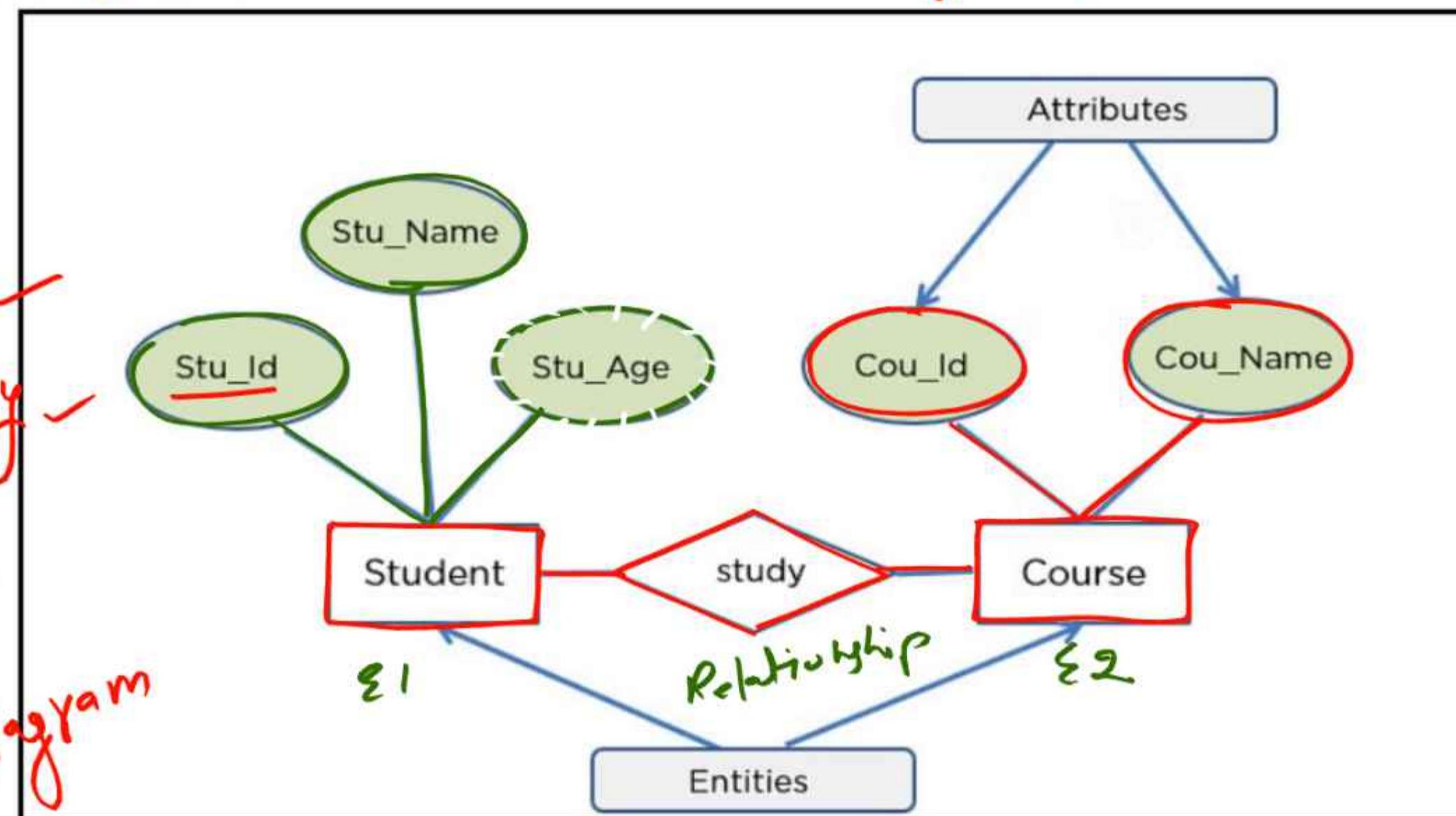




Simple ER Diagram:

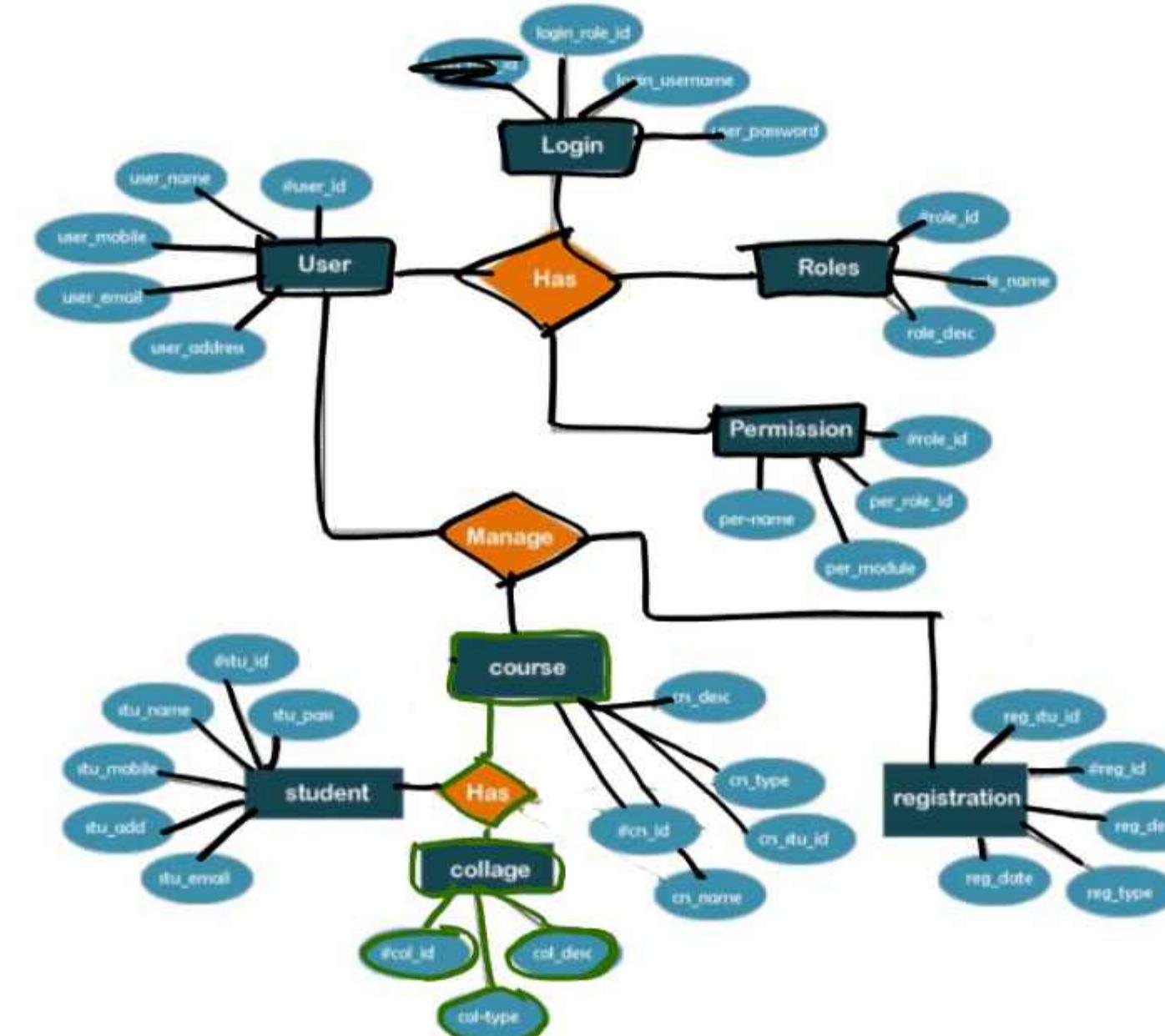
s marker ER diagram

College ✓  
University ✓  
Library ✓  
Management  
System  
on ER diagram



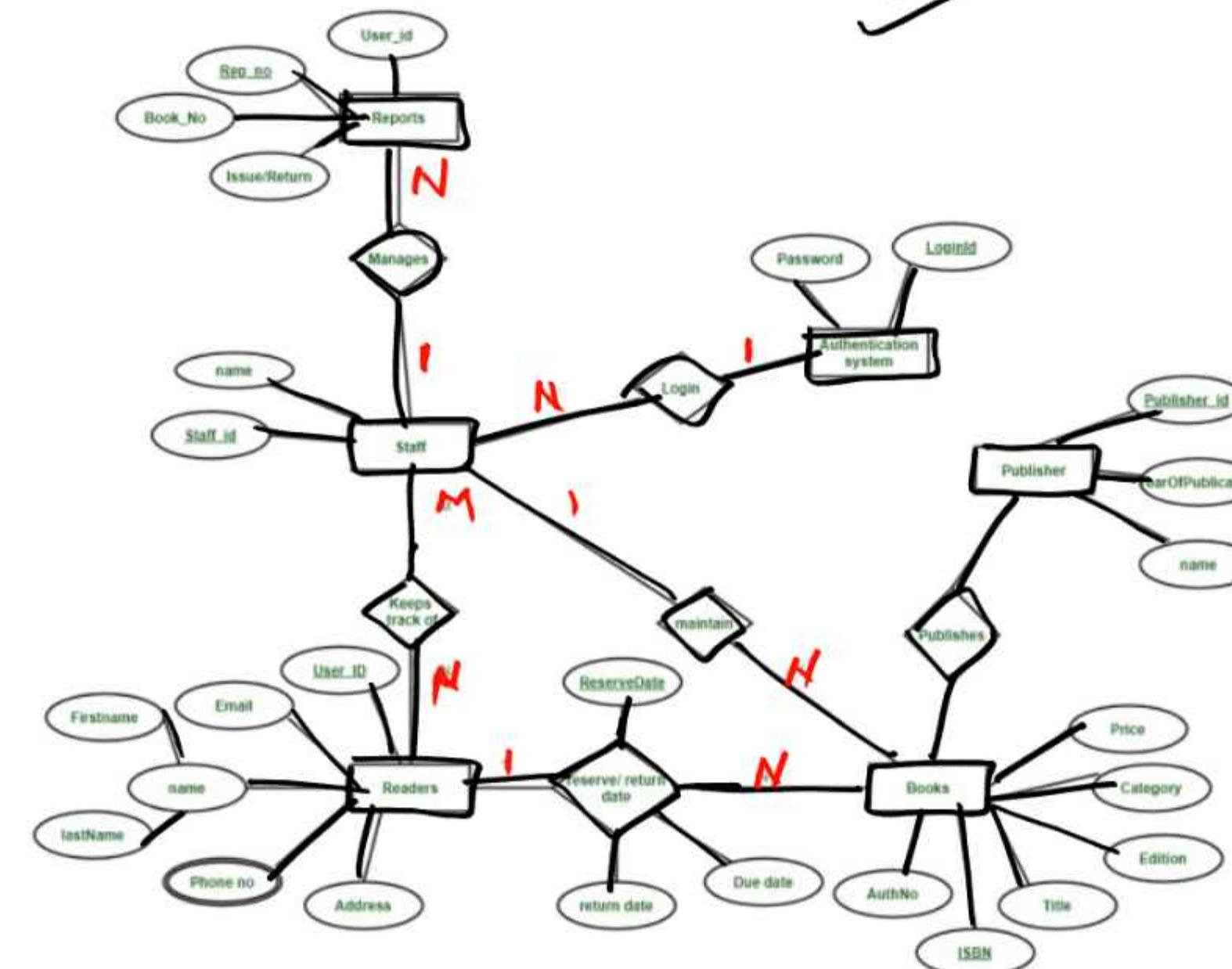


## ER Diagram for University Management System:



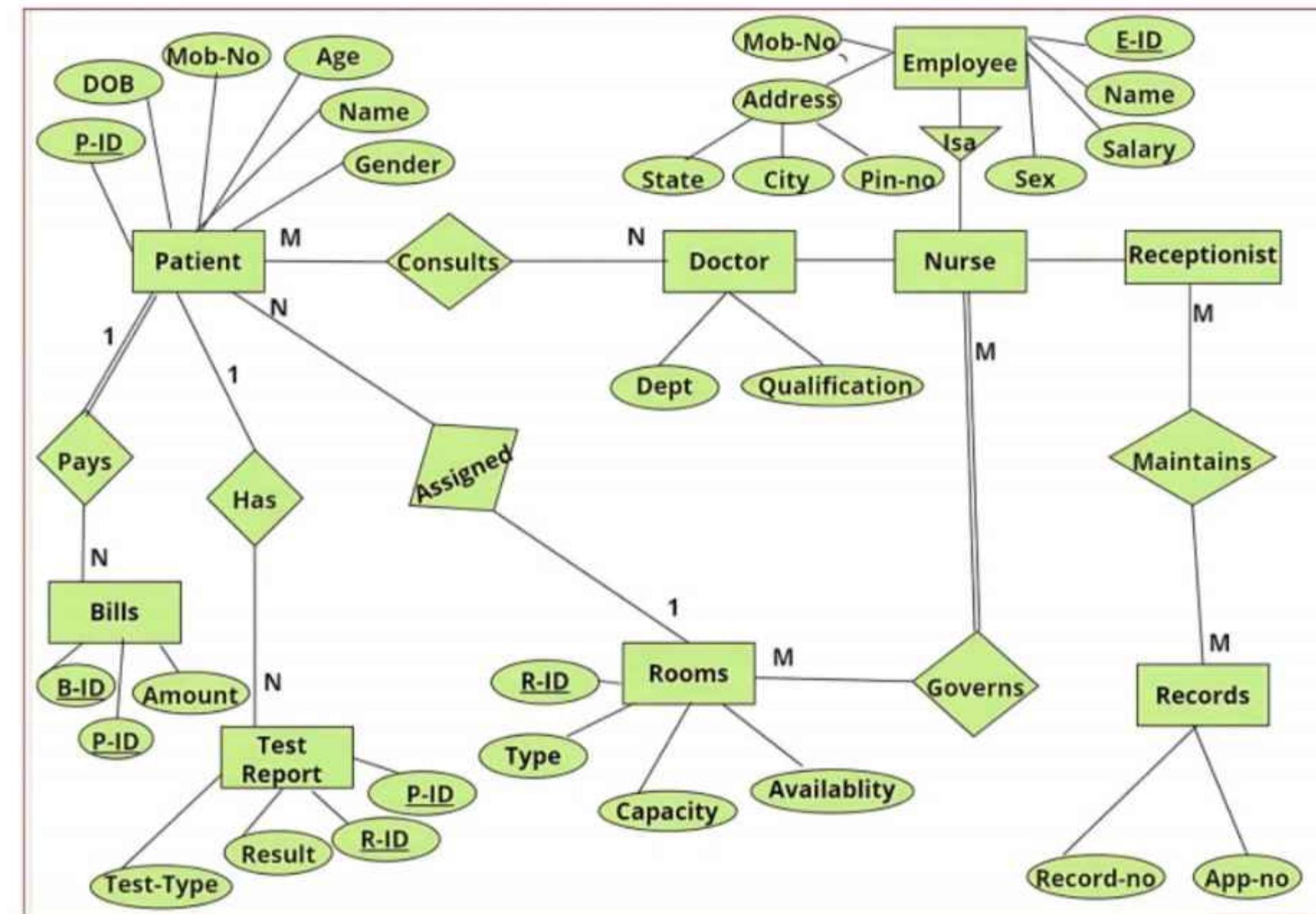


## ER Diagram of library management system:



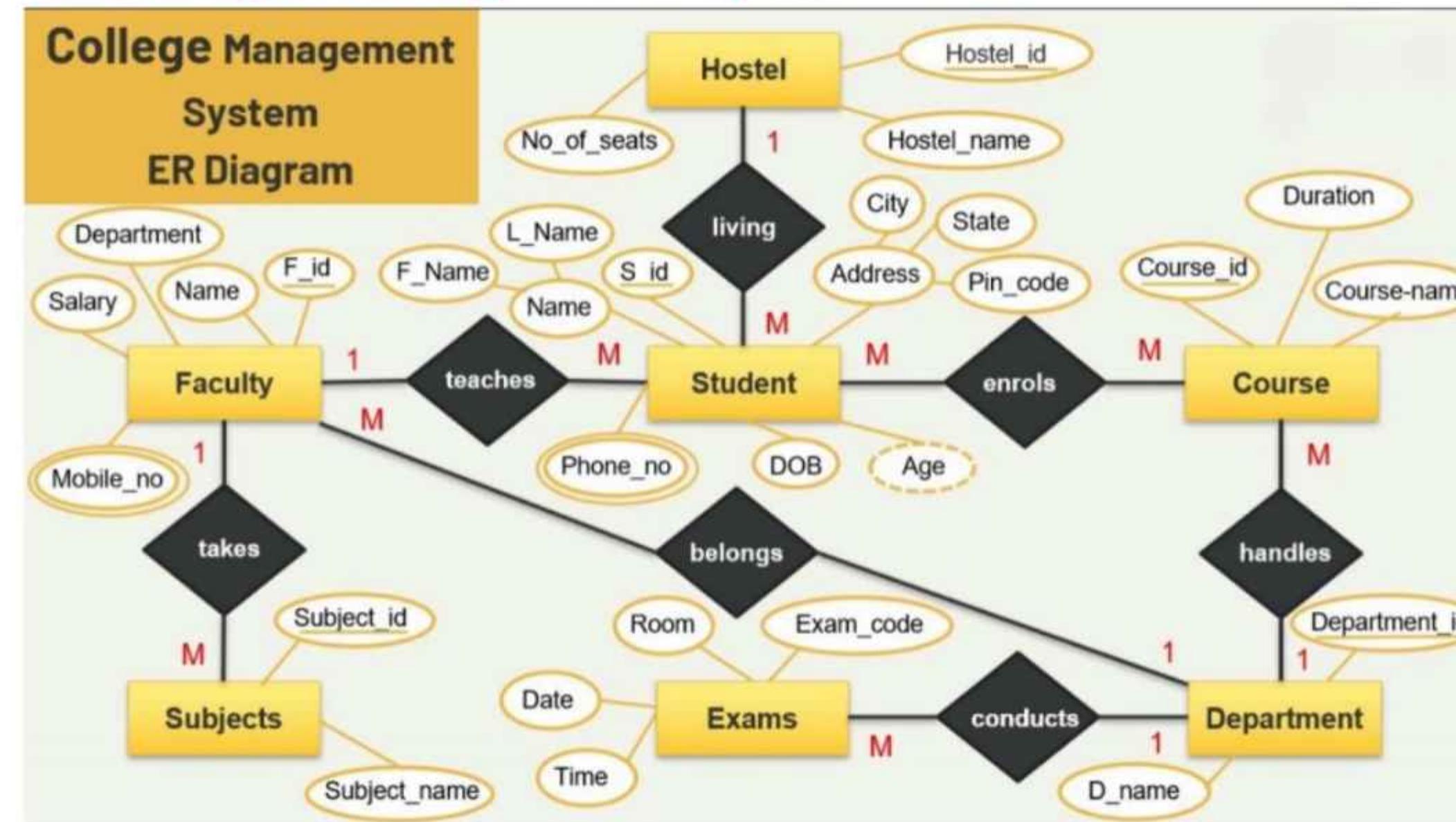


## → ER Diagram of hospital management system:





## ER Diagram of college management system: ✓



Que:  $\Rightarrow$  Different type के Model को समझाओ।  
or

Que:  $\Rightarrow$  Hierarchy & Network model को समझाओ।

Que:  $\Rightarrow$  University database management system का ER diagram बनाओ।

Que:  $\Rightarrow$  Write a short notes on —

- ① Aggregation ② Specialization ③ Generalization



### → Keys in DBMS: ✓

- ◆ Key वह Attribute या Attribute का समूह होता है, जो किसी भी Row (Record) को Uniquely Identify करता है।
- ◆ Key is an attribute or group of attributes that uniquely identifies any row (record).
- ◆ Keys का मुख्य उद्देश्य डेटा को Uniquely Identify करना, Duplication को रोकना और Relationship Maintain करना होता है।
- ◆ The main purpose of keys is to uniquely identify data, prevent duplication and maintain relationship.



## Types of Keys:

### Keys





## 1. Super Key: ✓

✓ Super Key वह Key होती है जो **हर Row को Unique Identify** करती है। ✓

✓ इसमें **एक या एक से ज्यादा Attributes** हो सकते हैं।

✓ Super Key का कोई Minimal Requirement नहीं होता।

✓ एक Table में कई Super Keys हो सकती हैं।

✓ **Example:**

**Super Key**

	Roll No.	Name	Age	Phone
1	Aryan	21	7491901521	
2	Sachin	25	870904365	
3	Prince	20	784600652	
4	Anuj	21	9876534523	

→ super\_key

Student_ID (PK)	Name	Email
101	Rahul	rahul@gmail.com
102	Priya	priya@yahoo.com
103	Aman	aman@gmail.com



## 1. Super Key:

- ✓ Super Key is the key that uniquely identifies each row.
- ✓ It can have one or more attributes.
- ✓ There is no minimum requirement for a Super Key.
- ✓ A table can have many Super Keys.
- ✓ Example:

Super Key

	Roll No.	Name	Age
1	Aryan	21	7491901521
2	Sachin	25	870904365
3	Prince	20	784600652
4	Anuj	21	9876534523

Diagram showing three arrows pointing from the 'Roll No.', 'Name', and 'Age' columns to three separate boxes, each labeled 'Super Key'.

Student_ID (PK)	Name	Email
101	Rahul	rahul@gmail.com
102	Priya	priya@yahoo.com
103	Aman	aman@gmail.com



## 2. Candidate Key:

- ✓ Candidate Key एक Super Key का Minimal Subset होती है।
- ✓ यह Unique होती है और इससे ही Primary Key Select होती है।
- ✓ यह एक या एक से अधिक Columns से मिलकर बन सकती है।
- ✓ एक Table में एक से ज्यादा Candidate Keys हो सकती हैं।

### Example:

Candidate Key			
<p style="text-align: center;">Candidate Key</p> <pre>graph LR; A[Roll No.] --&gt; B[Name]; A --&gt; C[Age]; A --&gt; D[Phone]; B --&gt; E[Student_ID PK]; C --&gt; E; D --&gt; E;</pre>			
1	Aryan	21	7491901521
2	Sachin	25	870904365
3	Prince	20	784600652
4	Anuj	21	9876534523

	superkey	superkey	Candidate key
	Student_ID (PK)	Name	Email
101	Rahul	rahul@gmail.com	
102	Priya	priya@yahoo.com	
103	Aman	aman@gmail.com	



## 2. Candidate Key:

- ✓ Candidate Key is a minimal subset of a super key.
- ✓ It is unique and primary key is selected from it.
- ✓ It can be made up of one or more columns.
- ✓ A table can have more than one candidate keys.
- ✓ Example:

**Candidate Key**

Roll No.	Name	Age	Phone
1	Aryan	21	7491901521
2	Sachin	25	870904365
3	Prince	20	784600652
4	Anuj	21	9876534523

Student_ID (PK)	Name	Email
101	Rahul	rahul@gmail.com
102	Priya	priya@yahoo.com
103	Aman	aman@gmail.com



### 3. Primary Key:

- ✓ Primary Key Super Key का Minimal Subset होती है।
- ✓ यह सबसे छोटी Super Key होती है, जिससे Table के हर Row को Uniquely पहचाना जा सके।
- ✓ Table में सिर्फ एक ही Primary Key हो सकती है।
- ✓ यह Null नहीं हो सकती और Unique होनी चाहिए।
- ✓ Example:

Table:

Primary Key

Roll No.	Name	Age	Gpa
1	Aryan	21	3
2	Sachin	25	4
3	Prince	20	2.5
4	Anuj	21	3.5

Primary Key → Roll No. (R14)

superkey

Student_ID (PK)	PK	Name	Email
101		Rahul	rahul@gmail.com
102		Priya	priya@yahoo.com
103		Aman	aman@gmail.com

Candidate key → एक से अधिक attribute की समूह है।

Primary key → केवल एक Attribute होगा।



### 3. Primary Key:

- ✓ Primary Key is the Minimal Subset of Super Key.
- ✓ It is the smallest Super Key, which can uniquely identify every row of the table.
- ✓ There can be only one Primary Key in a table.
- ✓ It cannot be Null and must be Unique.
- ✓ Example:

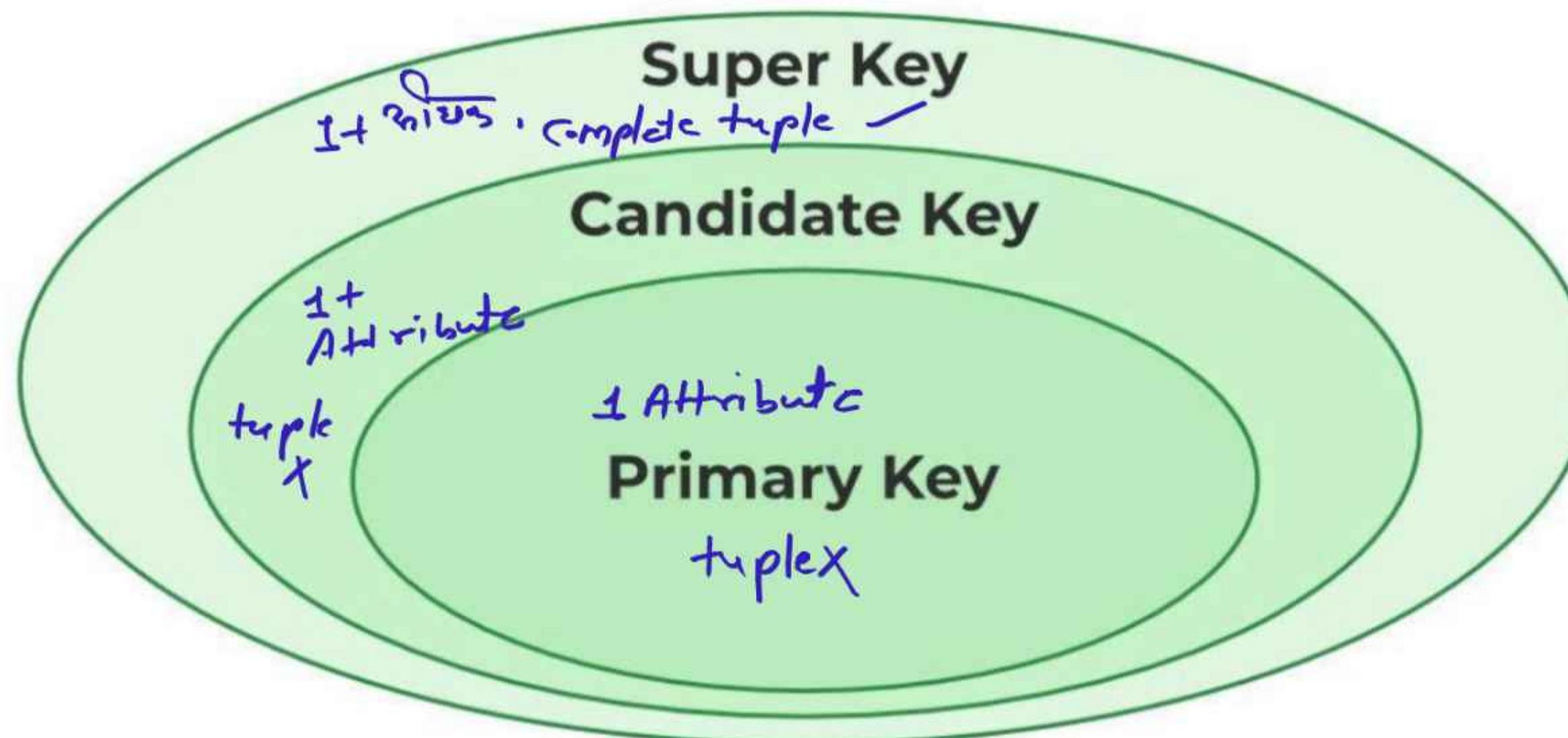
Table:

Primary Key

↑  
Primary Key

Roll No.	Name	Age	Gpa
1	Aryan	21	3
2	Sachin	25	4
3	Prince	20	2.5
4	Anuj	21	3.5

Student_ID (PK)	Name	Email
101	Rahul	rahul@gmail.com
102	Priya	priya@yahoo.com
103	Aman	aman@gmail.com





#### 4. Alternate Key:

- ✓ Candidate Keys में से जो Primary Key नहीं चुनी गई, वह Alternate Key होती है।
- ✓ हमारे पास दो Candidate Keys थीं: {Student\_ID}, {Email}।
- ✓ हमने {Student\_ID} को Primary Key बना लिया।
- ✓ तो {Email} Alternate Key बन गई।
- ✓ Example:

The diagram illustrates the concept of candidate and alternate keys through two tables.

**Left Table:** A student information table with columns: Roll No, Name, Age, and Phone. The primary key is Student\_ID (PK), indicated by a red circle. The column Roll No is labeled "Alternate Key" and "Candidate". The column Phone is labeled "Alternate" and "Candidate".

Roll No	Name	Age	Phone
1	Aryan	21	7491901521
2	Sachin	25	870904365
3	Prince	20	784600652
4	Anuj	21	9876534523

**Right Table:** A student information table with columns: Student\_ID (PK), Name, and Email. The primary key is Student\_ID (PK), indicated by a red circle. The column Email is labeled "Alternate Key" and "Candidate".

Student_ID (PK)	Name	Email
101	Rahul	rahul@gmail.com
102	Priya	priya@yahoo.com
103	Aman	aman@gmail.com



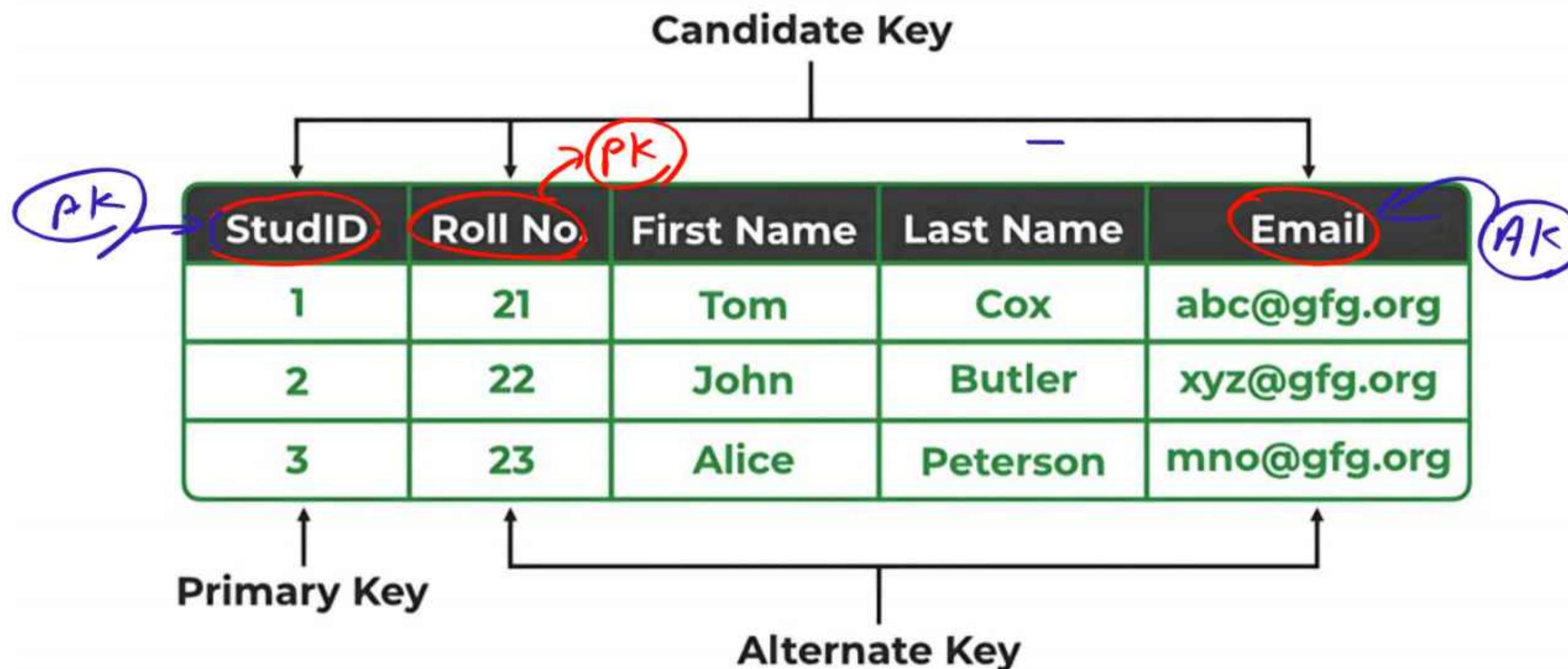
#### 4. Alternate Key:

- ✓ The Candidate Key that is not selected as the Primary Key is called the Alternate Key.
- ✓ We had two Candidate Keys: {Student\_ID}, {Email}.
- ✓ We made {Student\_ID} the Primary Key.
- ✓ So {Email} became the Alternate Key.
- ✓ Example:

**Alternate Key**

Roll No.	Name	Age	Phone
1	Aryan	21	7491901521
2	Sachin	25	870904365
3	Prince	20	784600652
4	Anuj	21	9876534523

Student_ID (PK)	Name	Email
101	Rahul	rahul@gmail.com
102	Priya	priya@yahoo.com
103	Aman	aman@gmail.com





## 5. Composite Key:

- ✓ Composite Key तब बनती है जब एक Column अकेला Unique Identification नहीं कर सकता।
- ✓ इसमें हमेशा दो या अधिक Attributes होते हैं।
- ✓ हर Composite Key एक Candidate Key हो सकती है, लेकिन हर Candidate Key Composite नहीं होती!
- ✓ Example:

Composite Key

Roll No.	Name	Age	Phone
1	Aryan	21	7491901521
2	Sachin	25	870904365
3	Prince	20	784600652
4	Anuj	21	9876534523

→ Composite Key

(Composite Key)

Student_ID	Course_ID	Enrollment_Date	Marks
101	CSE101	2024-01-10	85
101	CSE102	2024-01-12	90
102	CSE101	2024-01-15	80
103	CSE103	2024-01-18	88



## 5. Composite Key:

- ✓ Composite Key is created when a column alone cannot do unique identification.
- ✓ It always has two or more attributes.
- ✓ Every Composite Key can be a Candidate Key, but every Candidate Key is not Composite!
- ✓ Example:

Composite Key

Composite Key			
Roll No.	Name	Age	Phone
1	Aryan	21	7491901521
2	Sachin	25	870904365
3	Prince	20	784600652
4	Anuj	21	9876534523

Student_ID	Course_ID	Enrollment_Date	Marks
101	CSE101	2024-01-10	85
101	CSE102	2024-01-12	90
102	CSE101	2024-01-15	80
103	CSE103	2024-01-18	88



## 6. Foreign Key:

- ✓ वह Key जो एक Table में मौजूद होती है लेकिन किसी दूसरे Table की Primary Key को Refer करती है।
- ✓ Example: यदि हमारे पास Students और Courses दो Tables हैं, तो Course\_ID एक Foreign Key हो सकती है जो Courses Table की Primary Key को Refer करे।
  - ◆ यहाँ पर Course\_ID Students Table में Foreign Key है, जो Courses Table की Primary Key को Refer कर रही है।

**Foreign Key**

Student_Details:		
Roll No.	Name	Course_Id
1	Aryan	101
2	Sachin	102
3	Prince	103

**Student\_Marks:**

Course_Id	Gpa
101	3
102	4
103	2.5

**Students Table**

Student_ID (PK)	Name	Email	Course_ID (FK)
101	Rahul	rahul@gmail.com	C001
102	Priya	priya@yahoo.com	C002
103	Aman	aman@gmail.com	C001

**Courses Table**

Course_ID (PK)	Course_Name
C001	Data Science
C002	Web Development



## 6. Foreign Key:

- ✓ The key which exists in one table but refers to the primary key of another table.
- ✓ Example: If we have two tables Students and Courses, then Course\_ID can be a Foreign Key which refers to the primary key of Courses table.
  - ◆ Here Course\_ID is a Foreign Key in Students table which is referring to the primary key of Courses table.

**Foreign Key**

**Student\_Details:**

Roll No.	Name	Course_Id
1	Aryan	101
2	Sachin	102
3	Prince	103

**Student\_Marks:**

Course_Id	Gpa
101	3
102	4
103	2.5

**Students Table**

Student_ID (PK)	Name	Email	Course_ID (FK)
101	Rahul	rahul@gmail.com	C001
102	Priya	priya@yahoo.com	C002
103	Aman	aman@gmail.com	C001

**Courses Table**

Course_ID (PK)	Course_Name
C001	Data Science
C002	Web Development



### 7. Unique Key: ✓

✓ Unique Key, Primary Key की तरह ही Unique होती है, लेकिन इसमें NULL Value हो सकती है।

✓ Example: Email ✓

✓ Why? ✓

PK → Null ✗

U.K. → null ✓

- Email Unique है, लेकिन कुछ छात्रों का Email हो सकता है और कुछ का नहीं भी।

- अगर किसी छात्र का Email नहीं है, तो इसे NULL रहने दिया जा सकता है, इसलिए यह Unique Key है।

Student_ID (PK)	Email (Unique Key)	Name	Age	Course	Enrollment_ID (FK)
101	rahul@gmail.com	Rahul	21	B.Tech	5001
102	priya@yahoo.com	Priya	22	MBA	5002
103	sohan@outlook.com	Sohan	20	BCA	5003
104	NULL ✓	Aman	23	B.Tech	5004



## 7. Unique Key:

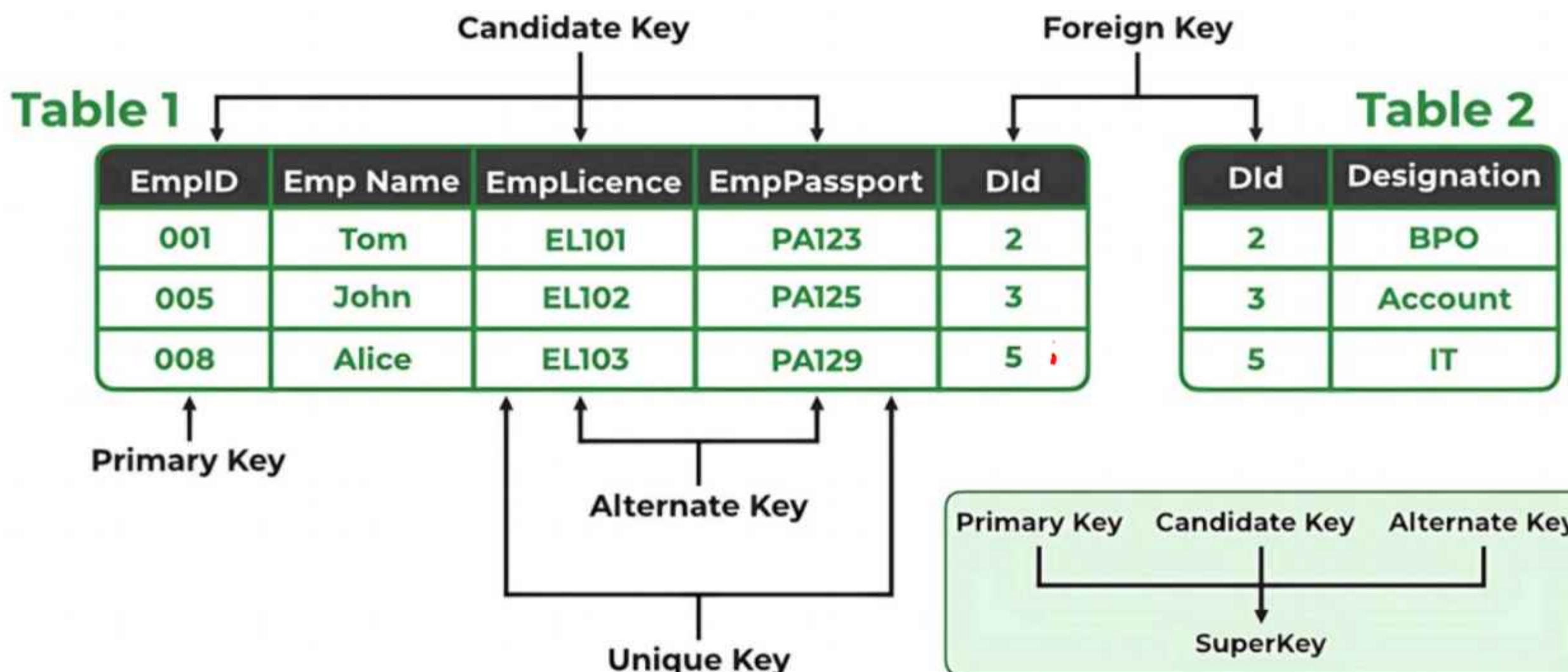
✓ Unique Key is unique like Primary Key but it can have NULL value.

✓ Example: Email

✓ Why?

- Email is unique but some students may have Email and some may not.
- If a student does not have Email then it can be allowed to be NULL so it is a Unique Key.

Student_ID (PK)	Email (Unique Key)	Name	Age	Course	Enrollment_ID (FK)
101	rahul@gmail.com	Rahul	21	B.Tech	5001
102	priya@yahoo.com	Priya	22	MBA	5002
103	sohan@outlook.com	Sohan	20	BCA	5003
104	NULL	Aman	23	B.Tech	5004





## Difference Between Primary Key and Unique Key:

Aspect	Primary Key ✓	Unique Key ✓
Uniqueness ✓	सभी वैल्यूज़ <u>Unique</u> होती हैं।	सभी वैल्यूज़ <u>Unique</u> होती हैं।
Null Values ✓	<u>NULL</u> वैल्यू नहीं हो सकती। ✗	✓ <u>NULL</u> वैल्यू हो सकती है।
<u>Number of Keys</u>	<u>only one</u> एक ही <u>Primary Key</u> हो सकती है।	<u>more than one</u> एक से ज्यादा <u>Unique Keys</u> हो सकती हैं।
Purpose ✓	Row को <u>Uniquely Identify</u> करने के लिए।	<u>Duplicate Values</u> को रोकने के लिए।
Example ✓	<u>Student_ID</u> ( <u>Unique</u> और <u>Not Null</u> )	<u>Email</u> ( <u>Unique</u> लेकिन <u>NULL</u> हो सकता है)



## Difference Between Primary Key and Unique Key:

Aspect	Primary Key	Unique Key
Uniqueness	All values are unique.	All values are unique.
Null Values	Cannot have NULL value.	Can contain NULL values.
Number of Keys	There can be only one Primary Key.	There can be more than one Unique Keys.
Purpose	To uniquely identify the row.	To prevent duplicate values.
Example	Student_ID (Unique और Not Null)	Email (Unique but can be NULL)



## Example Table:

Student_ID (PK)	Name	Email (Unique Key) <i>uk</i>	Phone (Unique Key)
101	Rahul	rahul@gmail.com	9876543210
102	Priya	priya@yahoo.com	8765432109
103	Aman	NULL	7654321098

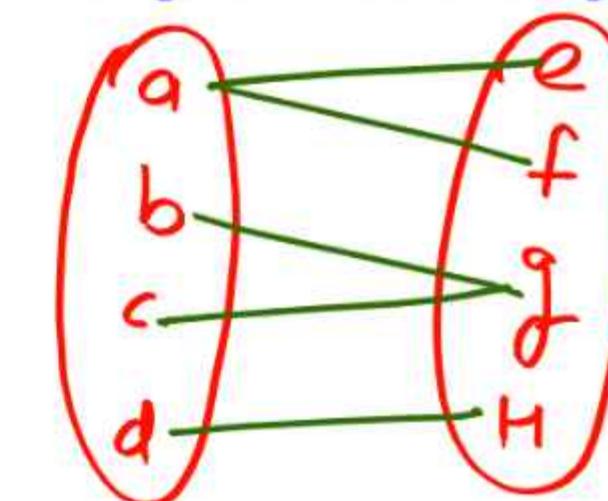


## Mapping Constraint:

- Mapping Constraint यह निर्धारित करता है कि एक Entity Set की कितनी Entities दूसरी Entity Set की कितनी Entities से जुड़ सकती हैं।
- Mapping Constraint determines how many entities of one Entity Set can be joined to how many entities of another Entity Set.

### उदाहरण (Example):

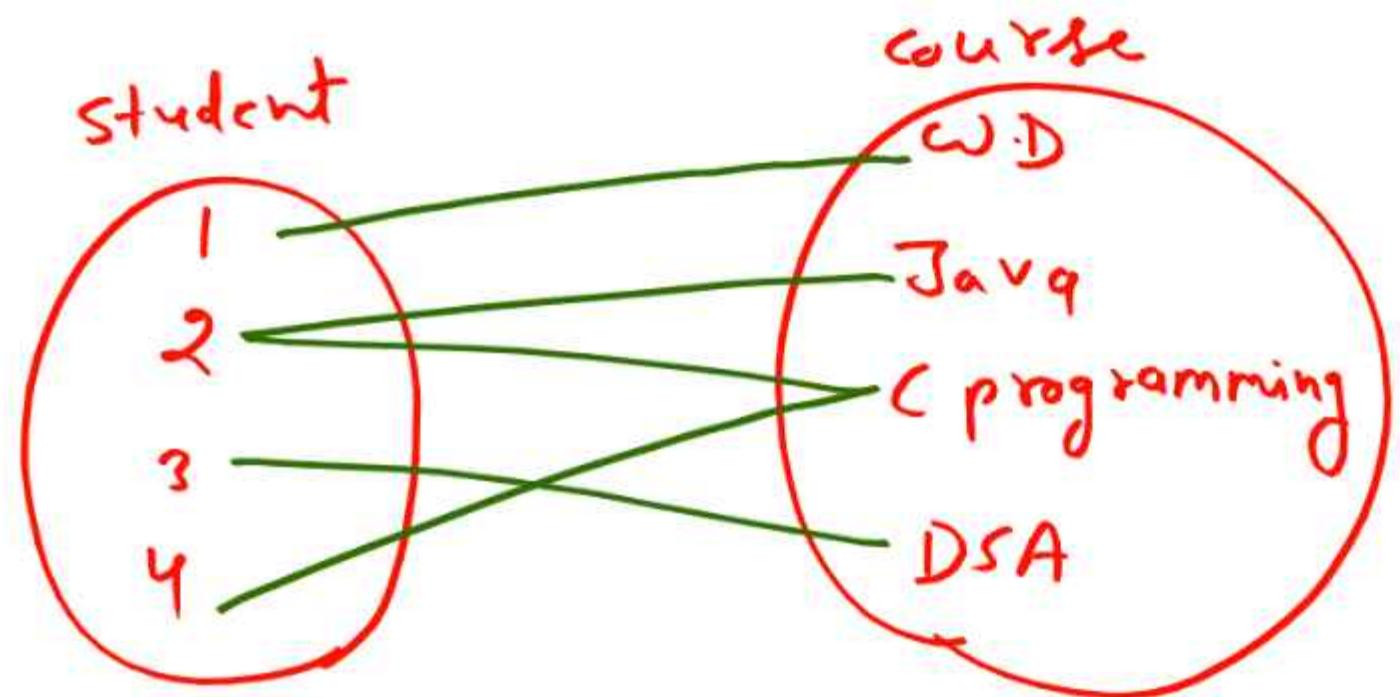
- अगर हमारे पास दो Entity Sets हैं: / If we have two Entity Sets:
- 1. "Student" (छात्र)
- 2. "Course" (पाठ्यक्रम)
- तो हमें यह निर्धारित करना होगा कि एक Student कितने Courses ले सकता है और एक Course कितने Students को पढ़ाया जा सकता है।
- So we have to determine how many courses a student can take and how many students a course can be taught to.



E.S.1

E.S.2

Ex:

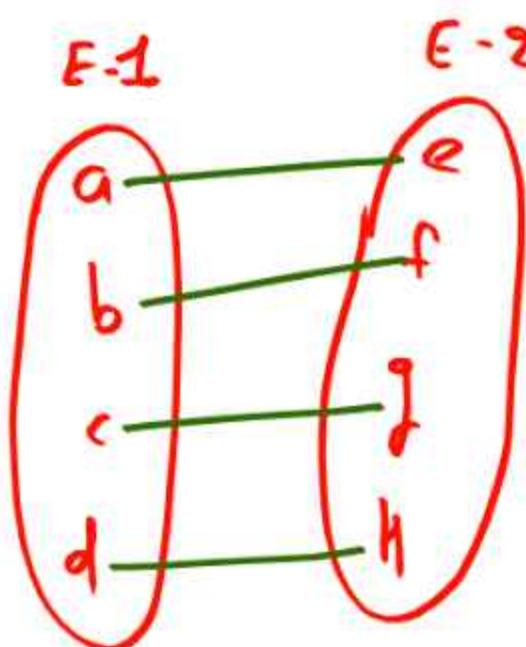


Mapping constraint

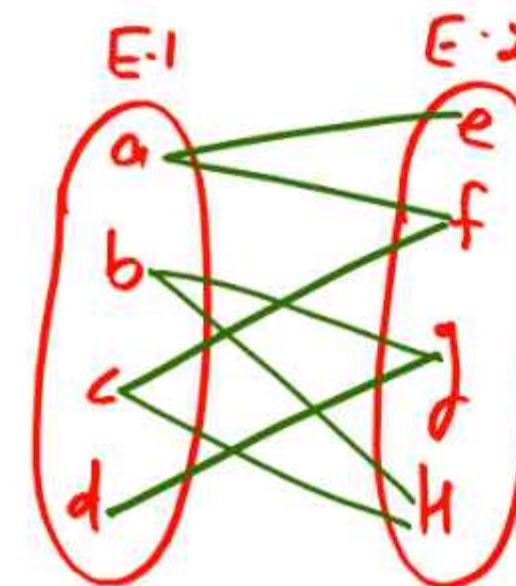


## Types of Mapping Constraints:

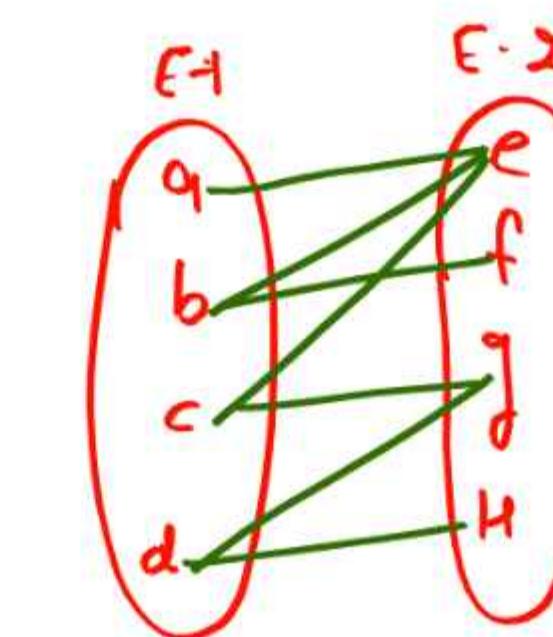
- ① One-to-One (1:1) Mapping
- ② One-to-Many (1:M) Mapping
- ③ Many-to-One (M:1) Mapping
- ④ Many-to-Many (M:N) Mapping



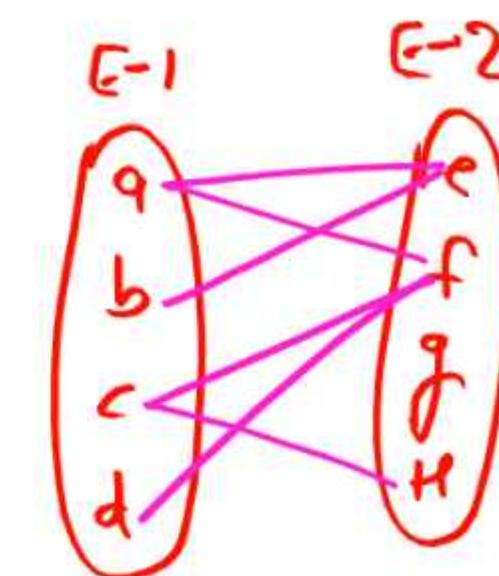
one to one  
mapping



one to many  
mapping  
constraint



many to one  
mapping  
constraint



many to many  
mapping constraint



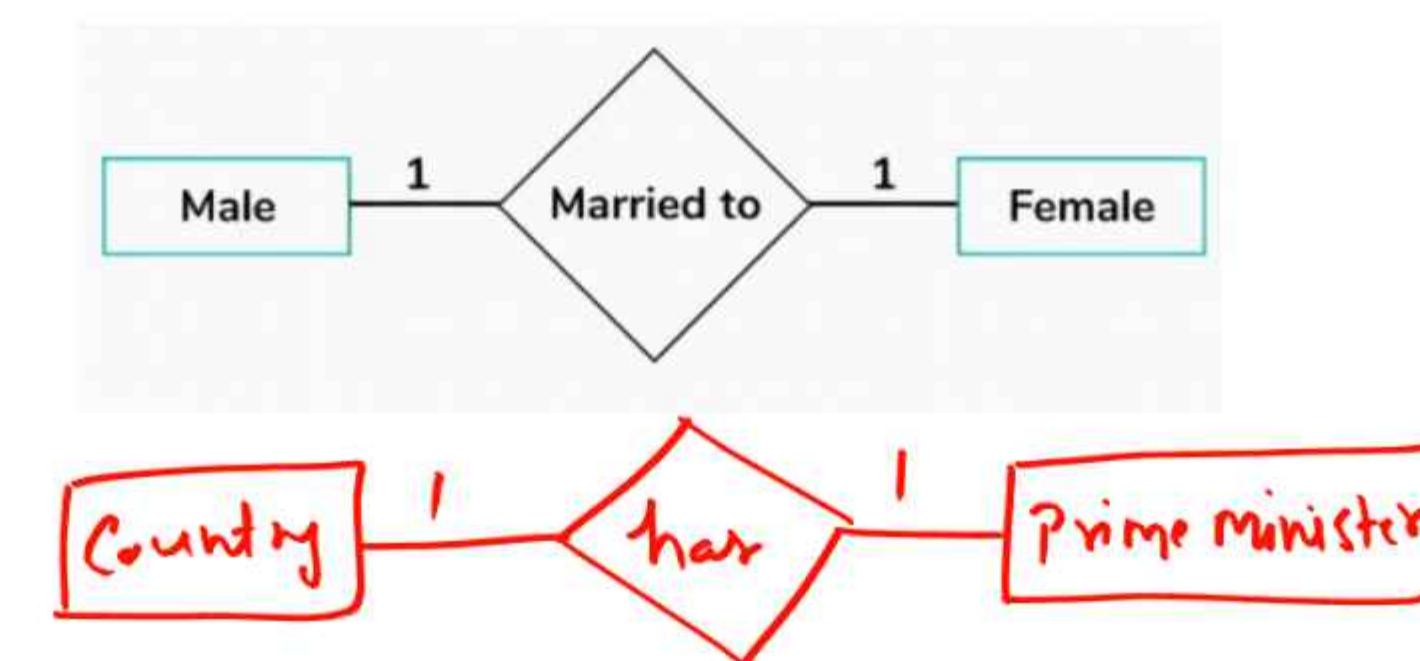
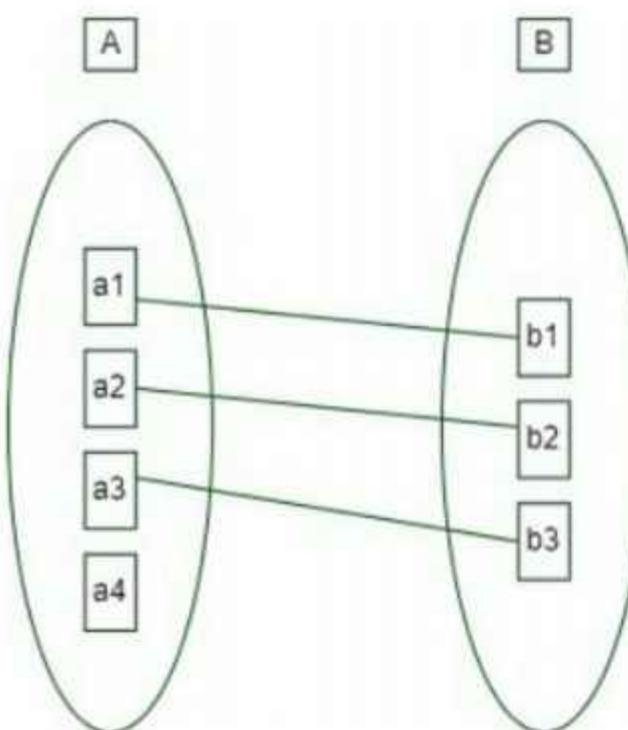
### ① One-to-One (1:1) Mapping:

Ek Entity dusri Entity se keval ek hi baar juudh sakati hai.

An entity can be linked to another entity only once.

#### Example:

- Har pati (Husband) ki keval ek patni (Wife) ho sakati hai, aur har patni ka keval ek pati ho sakta hai.
- Har desh (Country) ka keval ek pradhansamntri (Prime Minister) hota hai.





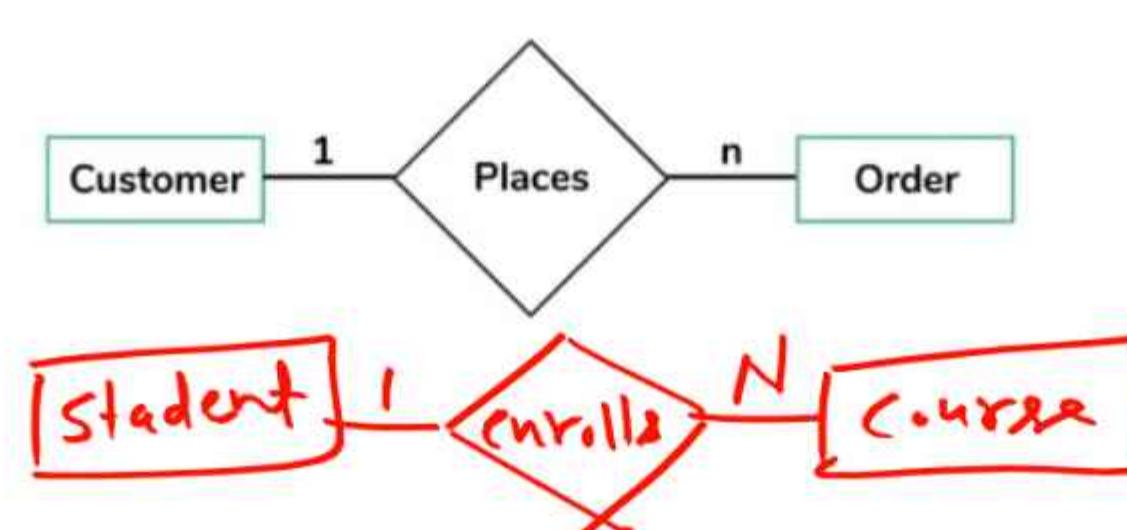
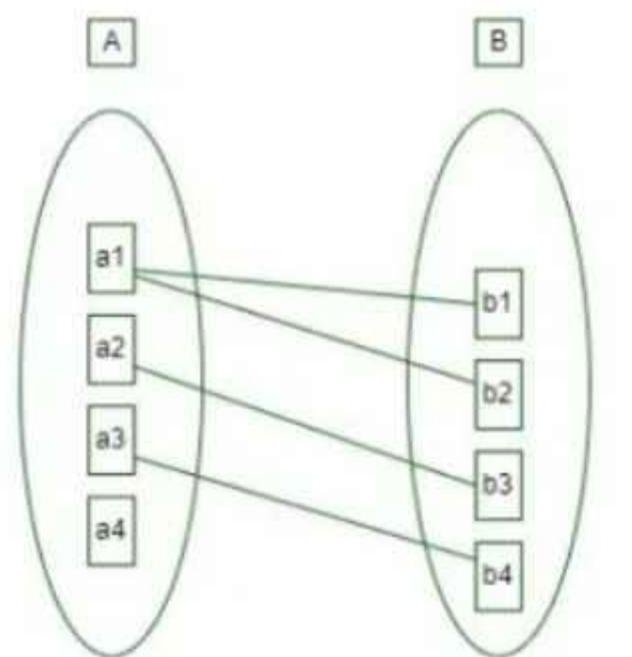
## ② One-to-Many (1:M) Mapping:

👉 एक Entity दूसरी Entity से कई बार जुड़ सकती है, लेकिन दूसरी Entity केवल एक से जुड़ सकती है।

👉 An entity can be linked to another entity multiple times, but another entity can be linked to another entity only once.

### ◆ Example:

- एक Professor कई Students को पढ़ा सकता है, लेकिन एक Student केवल एक ही Professor से पढ़ सकता है।
- एक माता (Mother) के कई बच्चे (Children) हो सकते हैं, लेकिन हर बच्चा केवल एक माता से ही जन्म लेता है।



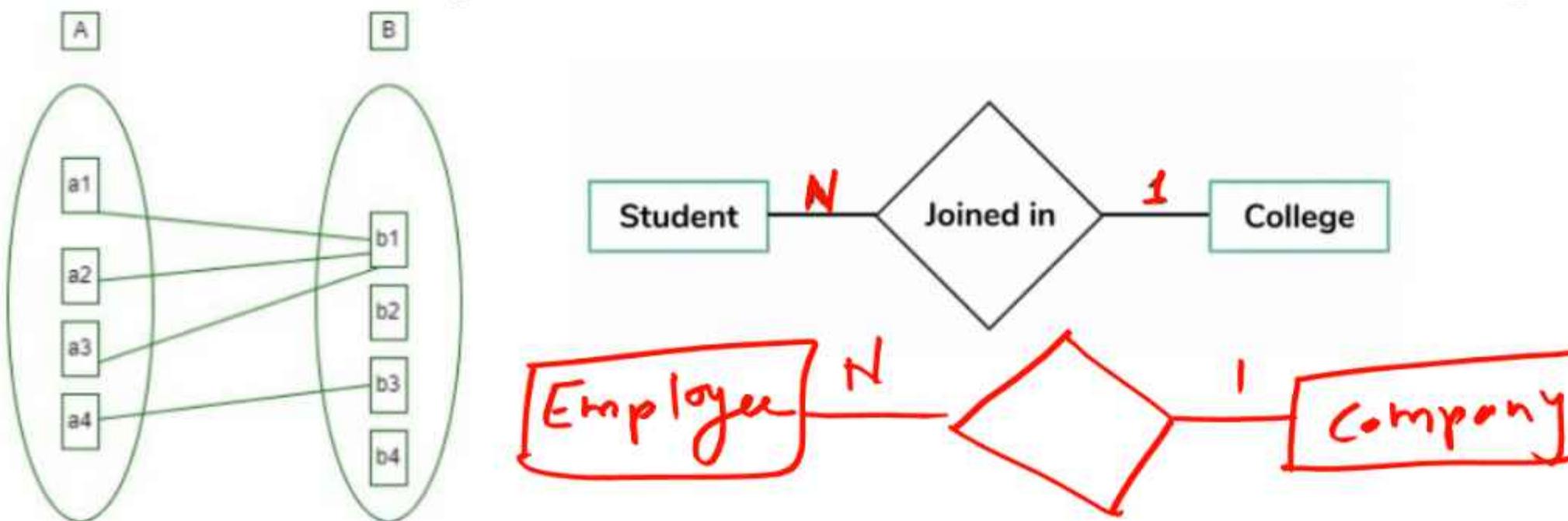


### ③ Many-to-One (M:1) Mapping:

- Many-to-One एक प्रकार से One-to-Many का उल्टा होता है।
- Many-to-One is in a way the opposite of One-to-Many.

#### Example:

- कई Employees एक ही Department में काम कर सकते हैं, लेकिन हर Employee सिर्फ एक ही Department से जुड़ा होगा।
- कई बच्चे एक ही स्कूल में पढ़ सकते हैं, लेकिन हर बच्चा सिर्फ एक ही स्कूल में होगा।





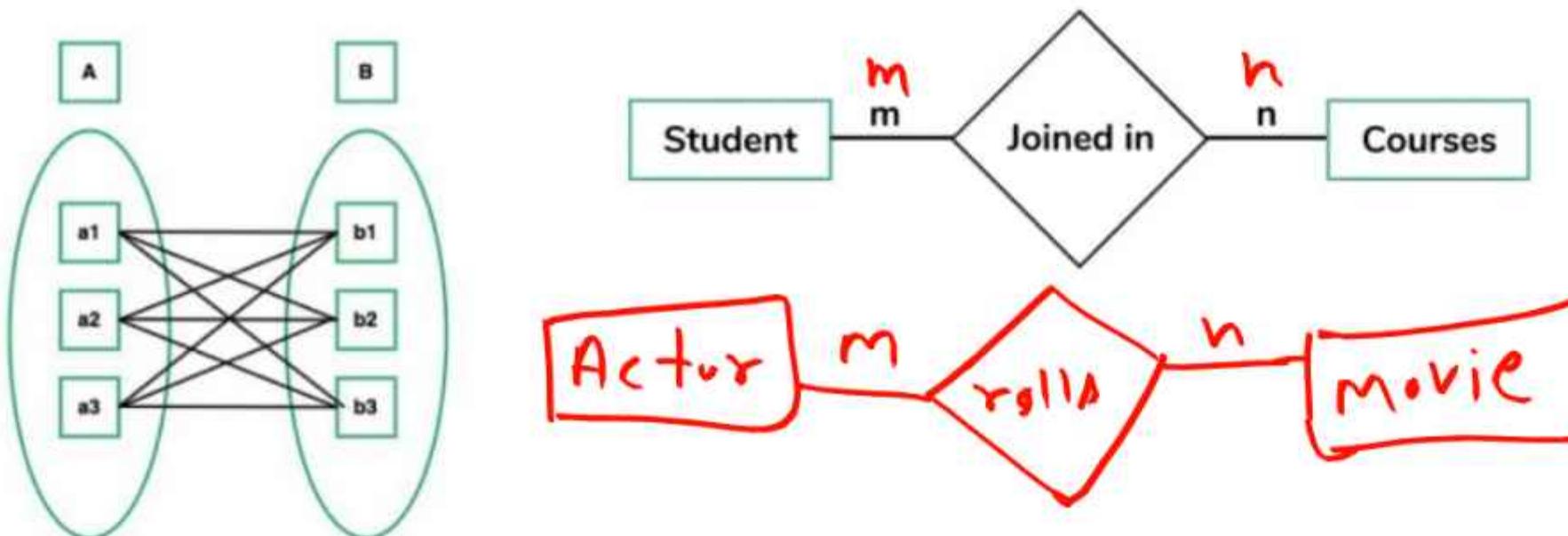
#### ④ Many-to-Many (M:N) Mapping:

एक Entity दूसरी Entity से कई बार जुड़ सकती है, और दूसरी Entity भी पहली Entity से कई बार जुड़ सकती है।

An entity can be connected to another entity multiple times, and the second entity can also be connected to the first entity multiple times.

##### Example:

- एक Student कई Courses ले सकता है, और एक Course कई Students को पढ़ाया जा सकता है।
- एक Actor कई Movies में काम कर सकता है, और एक Movie में कई Actors हो सकते हैं।



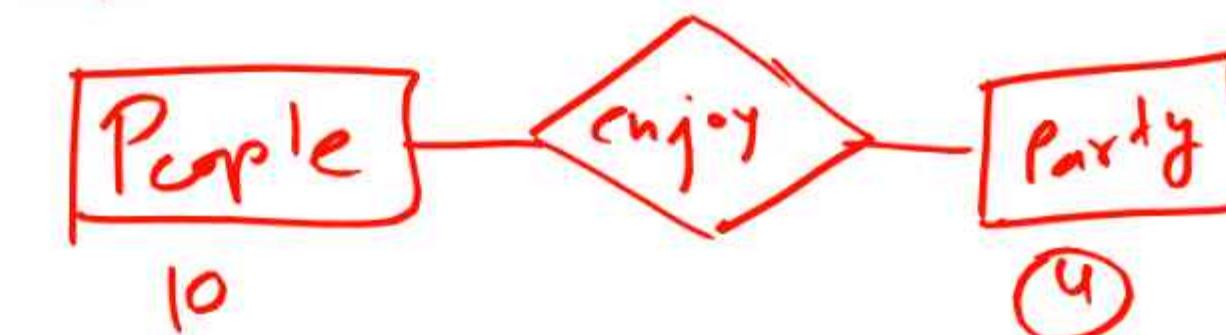


## Participation Constraint: ✓

- Participation Constraint यह निर्धारित करता है कि एक Entity Set किसी Relationship Set में भाग लेता है या नहीं और अगर लेता है तो कितनी मात्रा में।
- Participation Constraint determines whether an Entity Set participates in a Relationship Set and if so, to what extent.

## Types of Participation Constraints:

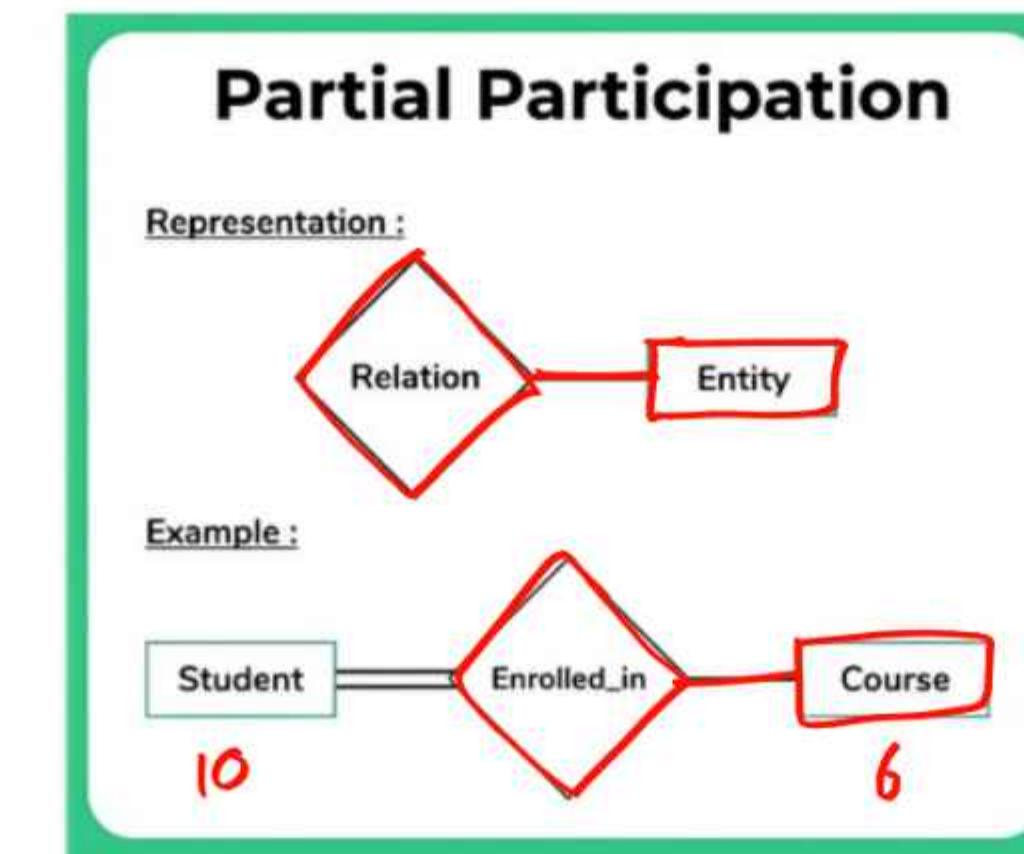
- आंशिक भागीदारी (Partial Participation):
- पूर्ण भागीदारी (Total Participation):





① आंशिक भागीदारी (Partial Participation):

- जब किसी Entity Set की सिर्फ कुछ entities किसी Relationship Set का हिस्सा होती हैं, तो इसे Partial Participation कहते हैं।
- When only some entities of an Entity Set are part of a Relationship Set, it is called Partial Participation.





## उदाहरण (Example): /

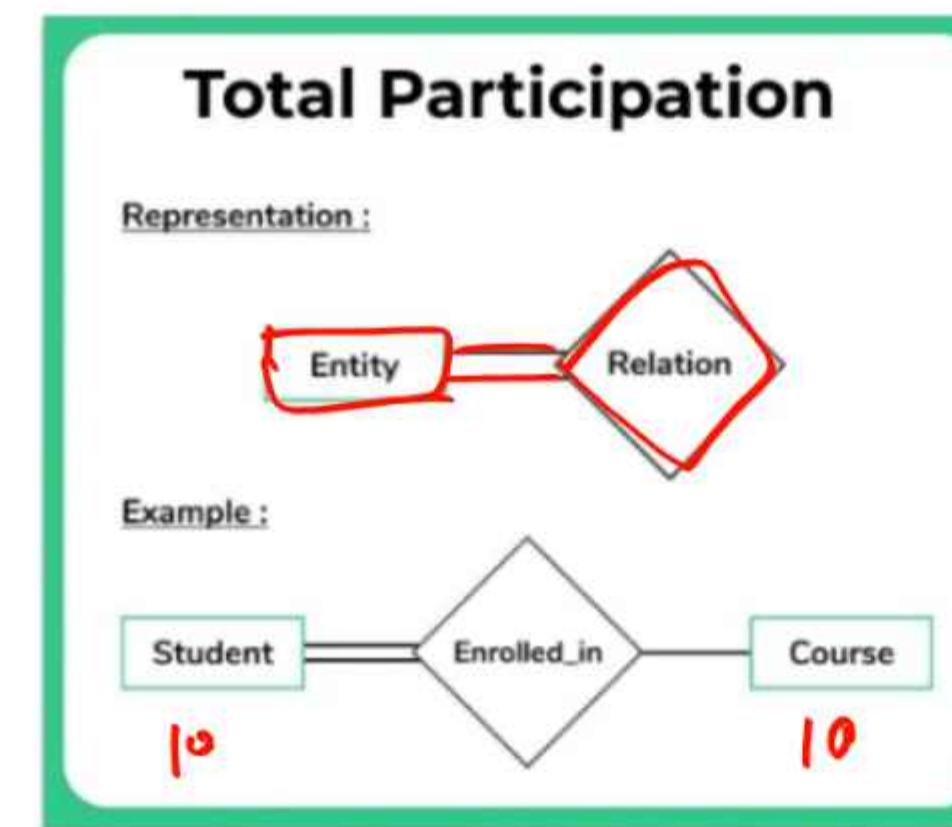
- "Employee - Manages - Department"
  - हर Department को एक Manager की ज़रूरत होती है, लेकिन हर Employee ज़रूरी नहीं कि Manager हो।
  - यहाँ Employee का Partial Participation है क्योंकि कुछ Employees Managers नहीं होते।
- 
- "Employee - Manages - Department"
  - Every Department needs a Manager, but every Employee need not be a Manager.
  - Here there is Partial Participation of Employee because some Employees are not Managers.





## ② पूर्ण भागीदारी (Total Participation):

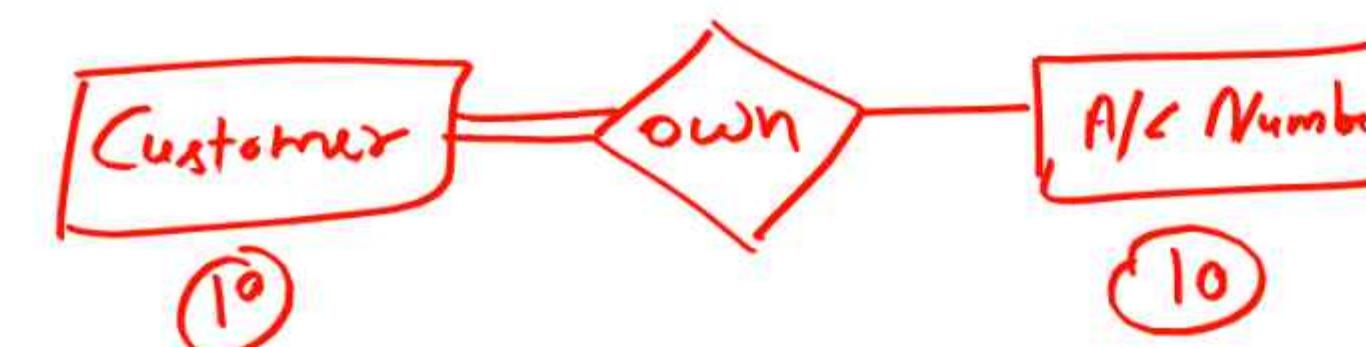
- किसी Entity Set की सभी Entities का किसी Relationship में भाग लेना अनिवार्य होता है। यानी कोई भी Entity इस रिश्ते से बाहर नहीं होती।
- It is mandatory for all entities of an Entity Set to participate in a relationship. That is, no entity is outside this relationship.





## उदाहरण (Example):

- "Customer - Owns - Account" ✓
- हर Account किसी Customer के नाम पर होता है।
- यहाँ Account Entity का Total Participation है क्योंकि हर Account को कम से कम एक Customer से जुड़ना अनिवार्य है।
- "Customer - Owns - Account"
- Every Account is in the name of a Customer.
- Here Account Entity has Total Participation because every Account must be linked to at least one Customer.





→ एंटिटी सेट (Entity Set):

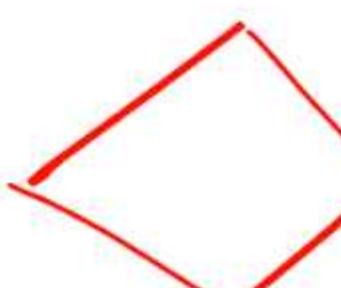
group of  
entity is  
called as  
entity set.

Roll No.	Name	Course	
CS08	Steive	Comp. Sci.	→ Entity
EE54	Jhoson	Electronics	→ Entity
B12	Eva	Biology	→ Entity
F32	Jhoson	Finance	→ Entity
M26	Erica	Maths	→ Entity

Student Table



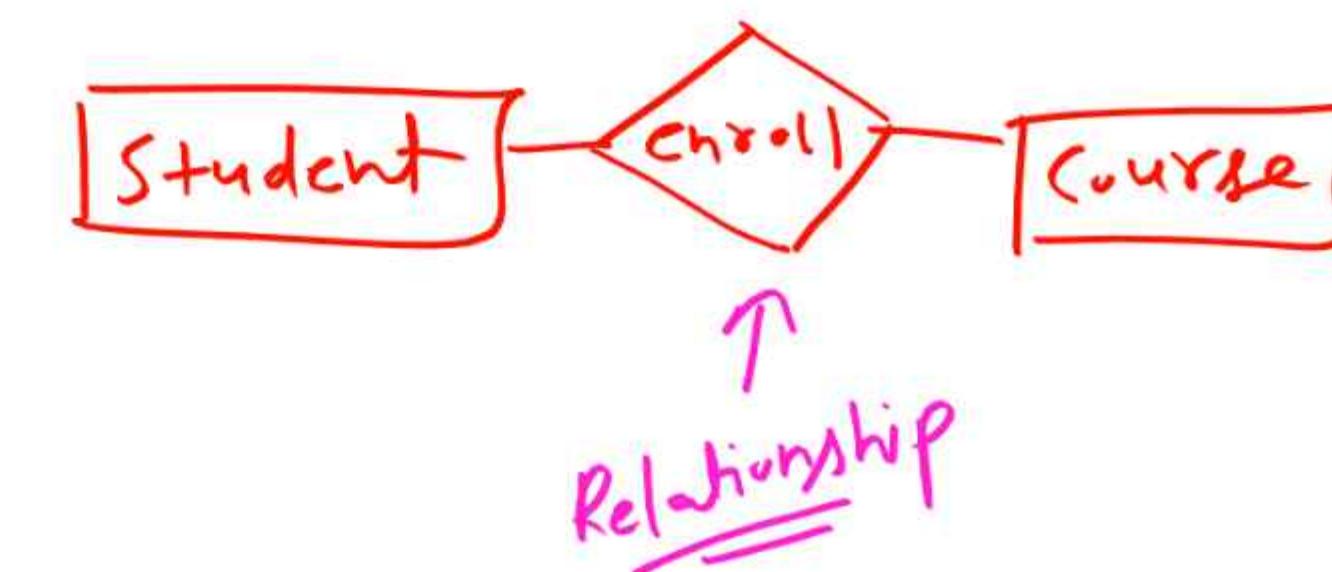
### रिलेशनशिप सेट (Relationship Set): ✓

- जब दो या अधिक Entity Sets आपस में किसी संबंध (Relation) से जुड़े होते हैं, तो उस संबंध को Relationship Set कहा जाता है। ✓
- When two or more Entity Sets are connected with each other by a relationship, then that relationship is called a Relationship Set.
  
- "Entities के बीच आपसी संबंध (Association) को Relationship Set कहा जाता है।
- "The mutual relationship (association) between entities is called Relationship Set."
  
- ◆ नोटेशन (Noation): ✓
  - ♦ Diamond (हीरा) आकृति में Relationship लिखा जाता है।
  - ♦ Relationship is written in diamond shape. ✓



## उदाहरण (Example):

- अगर एक Student किसी Course में Admission लेता है, तो Student और Course के बीच Enrolls In नाम का Relationship होगा।
- If a student takes admission in a course, then there will be a relationship named Enrolls In between the student and the course.
- Entities: Student, Course
- Relationship Set: Enrolls In





### Strong Entity: Set: =>

- Strong Entity Set वह Entity Set होता है जो स्वतंत्र रूप से (Independently) अपना अस्तित्व बनाए रख सकता है।
- इसमें प्रत्येक Entity को एक Unique Primary Key द्वारा पहचाना जा सकता है।
- इसे किसी दूसरी Entity की आवश्यकता नहीं होती।
- Strong Entity Set is that entity set which can maintain its existence independently.
- In this, each entity can be identified by a Unique Primary Key.
- It does not require any other entity.



### विशेषताएँ (features):

- ✓ इसमें एक Primary Key होती है। / It has a primary key.
- ✓ किसी अन्य Entity पर निर्भर नहीं होती। / It does not depend on any other entity.
- ✓ ER Diagram में Rectangle से दर्शाया जाता है। / It is represented by a rectangle in the ER diagram.



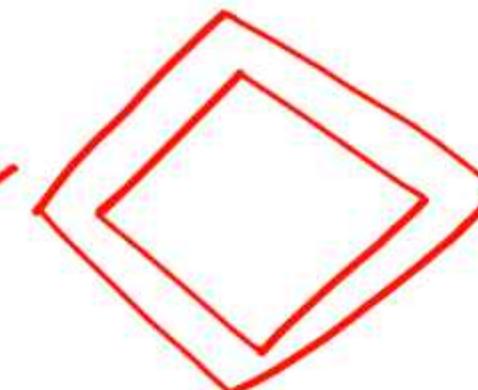


### Weak Entity: Set! →

- Weak Entity Set वह Entity Set होता है, जिसे अपनी पहचान के लिए किसी Strong Entity पर निर्भर  
रहना पड़ता है। ✓ Dependent.
  - इसमें कोई Primary Key नहीं होती! ✗
  - इसे पहचानने के लिए एक और Entity की Primary Key + Partial Key (Discriminator) की  
आवश्यकता होती है। ✓
- Weak Entity Set is that entity set which has to depend on a Strong Entity for its identification.
- It does not have any Primary Key.
- To identify it, Primary Key + Partial Key (Discriminator) of another entity is required.

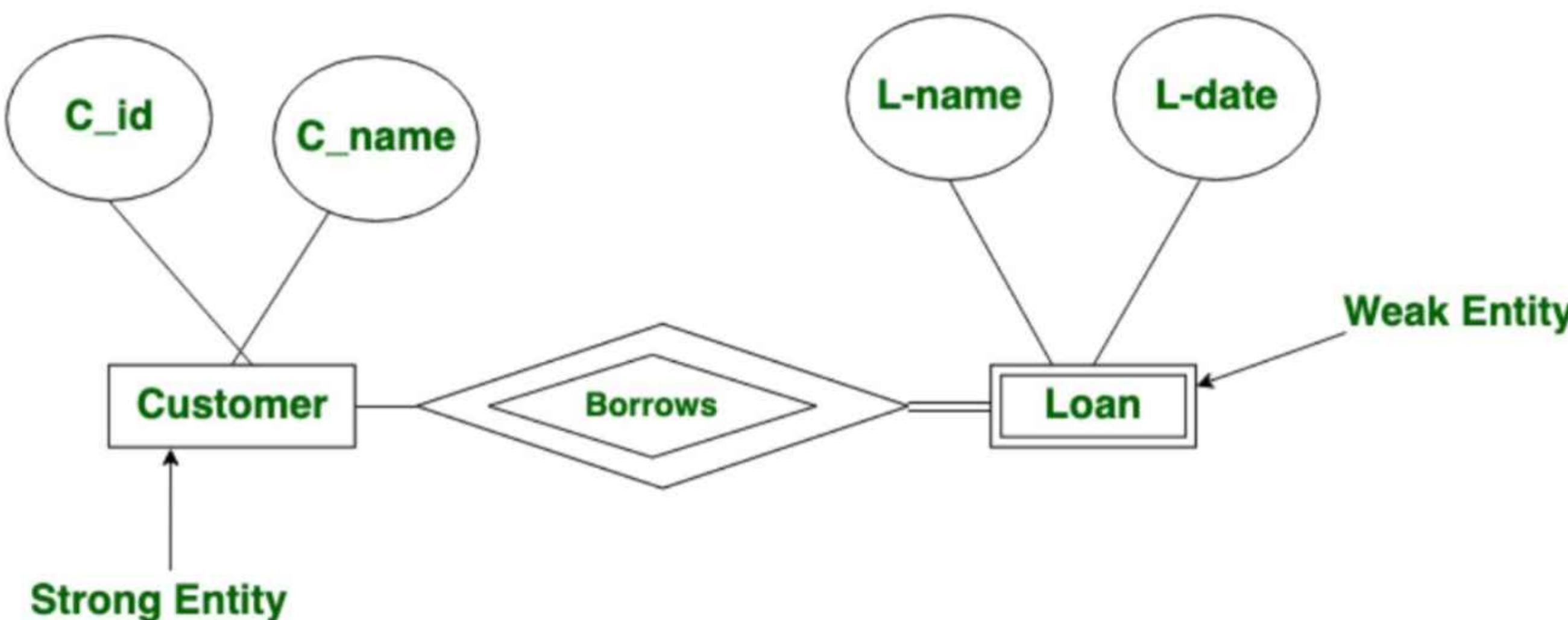


### विशेषताएँ (features):

- ✓ इसमें कोई Primary Key नहीं होती। ✗
- ✓ यह किसी दूसरी Strong Entity पर निर्भर करता है। *dependent*
- ✓ इसे पहचानने के लिए Discriminator (Partial Key) और Owner Entity की Primary Key का उपयोग किया जाता है।
- ✓ ER Diagram में Double Rectangle से दर्शाया जाता है। 
- ✓ इसका Identifying Relationship (Double Diamond) होता है। 
- ✓ It does not have any Primary Key.
- ✓ It depends on some other Strong Entity.
- ✓ Discriminator (Partial Key) and Primary Key of Owner Entity are used to identify it.
- ✓ It is represented as Double Rectangle in ER Diagram.
- ✓ It has an Identifying Relationship (Double Diamond).



Example:





### Example of Strong Entity:

- Customer (ग्राहक) एक Strong Entity है।
- इसमें C\_id (Primary Key) और C\_name Attribute शामिल हैं।
- इसे एकल आयत (Single Rectangle) से दर्शाया गया है।

### Example of Weak Entity:

- Loan (ऋण) एक Weak Entity है।
- इसमें L-name और L-date जैसे गुण (Attributes) शामिल हैं।
- इसमें कोई Primary Key नहीं है, बल्कि इसे Customer की Primary Key पर निर्भर रहना पड़ता है।
- इसे दोहरी आयत (Double Rectangle) से दर्शाया गया है।

### Relationship:

- Borrows (उधार लेना) एक संबंध (Relationship) है जो Customer और Loan को जोड़ता है।
- इसे Double Diamond से दर्शाया गया है क्योंकि यह एक Identifying Relationship है।
- यह दिखाता है कि Loan को Customer के बिना पहचाना नहीं जा सकता।



### Example of Strong Entity:

- Customer is a Strong Entity.
- It contains C\_id (Primary Key) and C\_name Attribute.
- It is represented by a Single Rectangle.

### Example of Weak Entity:

- Loan is a Weak Entity.
- It contains attributes like L-name and L-date.
- It does not have any Primary Key but it has to depend on the Primary Key of Customer.
- It is represented by a Double Rectangle.

### Relationship:

- Borrows is a relationship that connects Customer and Loan.
- It is represented by Double Diamond because it is an Identifying Relationship.
- It shows that Loan cannot be identified without Customer.



## Difference Between Strong Entity Set & Weak Entity Set:

5 marks

	Strong Entity Set ✓	Weak Entity Set ✗
1	इसमें Primary Key होती है।	इसमें Primary Key नहीं होती।
2	यह किसी दूसरी Entity पर निर्भर नहीं करता।	यह किसी Strong Entity पर निर्भर करता है।
3	Single Rectangle से दर्शाया जाता है।	Double Rectangle से दर्शाया जाता है।
4	दो Strong Entities के बीच संबंध Single Diamond से दर्शाया जाता है।	एक Strong और Weak Entity के बीच संबंध Double Diamond ((Identifying Relationship)) से दर्शाया जाता है।
5	Total या Partial Participation हो सकता है।	हमेशा Total Participation Constraint होता है।



10





## Difference Between Strong Entity Set & Weak Entity Set:

Strong Entity Set	Weak Entity Set
It contains the Primary Key.	There is no primary key in it.
It does not depend on any other entity.	It depends on some Strong Entity.
It is represented by a single rectangle.	It is represented by a double rectangle.
The relationship between two Strong Entities is represented by a Single Diamond.	The relationship between a Strong and Weak Entity is represented by a Double Diamond ((Identifying Relationship)).
There can be total or partial participation.	There is always a Total Participation Constraint.



weak entity set & strong entity set के बारे में।

Thank  
You



## Unit-3 : Relational Model

## Syllabus :

UNIT 3: Relational Model:

(10 Periods)

The Relational Data Model and Relational Database Constraints; ER/EER to Relational Model mapping; Relational Algebra and Relational Calculus, Relations algebra (Basic operation: Union intersection difference and Cartesian product), Additional Relational Algebraic Operations (Projection, Selection rows, Division, rename and join )

Que: ⇒ Relation data Model का detail तकीे explain करें। | 5 marks | 100%.

Que: ⇒ Different type का Relational Algebra का समाजो। | 5 marks

Que: ⇒ write a short notes on - Projection Selection



## रिलेशनल मॉडल (Relational Model): ✓

- Relational Model एक ऐसा तरीका है जिससे हम **डेटा को टेबल (Table)** के रूप में स्टोर करते हैं।
- Relational Model is a way we store data in the form of tables. ✓✓
- ✓ • हर टेबल में कई **rows (records)** और **columns (attributes)** होते हैं।
- Every table has many rows (records) and columns (attributes). ↴
  - ↳ ○ इसे E. F. Codd ने 1970 में IBM में introduce किया था।
    - It was introduced by E. F. Codd in IBM in 1970.
  - ✓ ○ यही model आज के सभी **RDBMS** (Relational Database Management System) जैसे MySQL, Oracle, SQL Server, PostgreSQL आदि का आधार है।
    - This model is the basis of all today's RDBMS (Relational Database Management System) like MySQL, Oracle, SQL Server, PostgreSQL etc.

Relation  
↓  
Table



**Column Attribute**

Roll_No	Name	Class	Age	
101	Ramesh	10th	15	$\leftarrow R_1$
102	Sita	9th	14	$\leftarrow R_2$
103	Amit	10th	15	$\leftarrow R_3$

**R-ω/tuple**

$\uparrow c_1 \quad \uparrow c_2 \quad \uparrow c_3 \quad \uparrow c_3$

- इसमें हर row एक student का data है - जिसे Tuple कहते हैं।
  - हर column एक property (जैसे नाम, उम्र) है - जिसे Attribute कहते हैं।
  - पूरी table को हम Relation कहते हैं।
- Relation = Table*
- In this, every row is data of one student - which is called Tuple.
  - Each column is a property (like name, age) - which is called Attribute.
  - We call the whole table Relation.



### Advantages (फायदे): ✓

#### 1. Simple Structure - ✓

- टेबल format बहुत आसान होता है, कोई भी समझ सकता है।
- Table format is very easy, anyone can understand it.

#### 2. SQL Language Support - ✓

- SQL एक powerful language है जो relational model पर काम करती है।
- SQL is a powerful language that works on the relational model.

#### 3. Data Integrity - ✓

- Primary key, foreign key, constraints से data safe और consistent रहता है।
- Primary key, foreign key, constraints keep data safe and consistent.

#### 4. Data Independence - ✓

- Logical और physical storage अलग रहता है।
- Logical and physical storage is separate.



## 5. Normalization - ✓

- डेटा को efficiently store करने के लिए redundancy हटाई जा सकती है।  
*redundancy* → *duplicacy*
- Redundancy can be removed to store data efficiently.

## 6. Multi-user Access - ✓

- एक साथ कई लोग database को access कर सकते हैं।
- Many people can access the database simultaneously.



## → Disadvantages (कमियाँ): ✓

### 1. Complex Joins - ✓

- अलग-अलग टेबल को जोड़ने के लिए complex joins लिखने पड़ते हैं।
- Complex joins have to be written to join different tables.

### 2. Performance Issue - ✓

- बहुत बड़े database पर performance degrade हो सकती है।
- Performance can degrade on very large databases.

### 3. Rigid Schema - ✓ स्थिर संरचना ✓

- एक बार schema fix हो जाए तो उसे बदलना मुश्किल हो जाता है।
- Once the schema is fixed, it becomes difficult to change it.

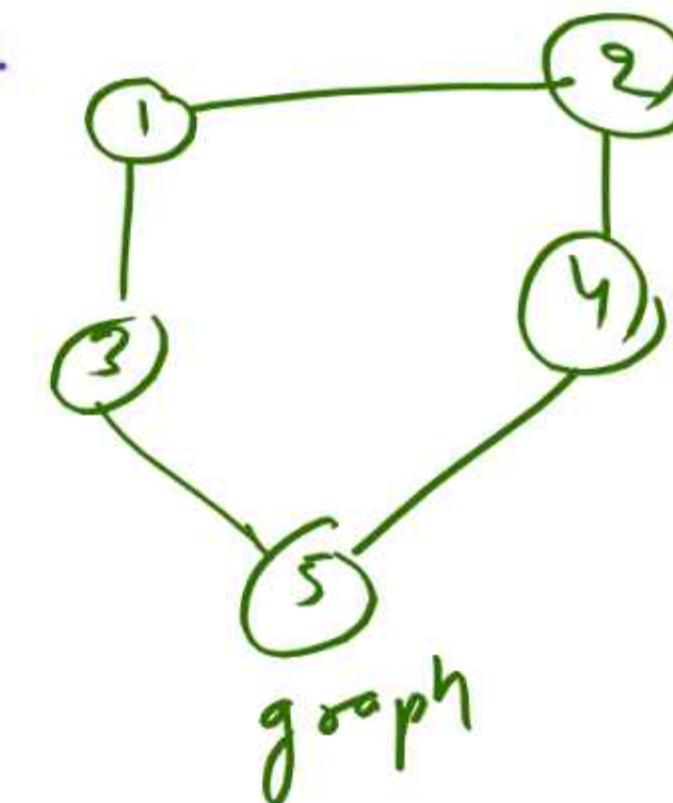
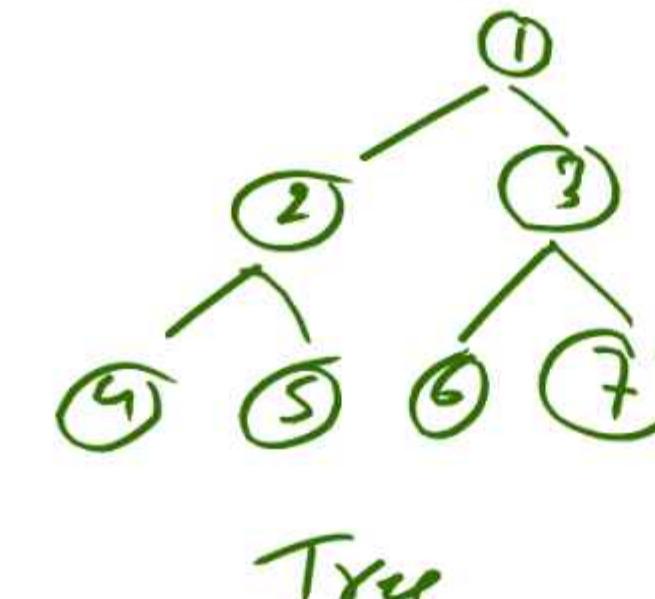


#### 4. Not Ideal for Non-Tabular Data - ✓

- Graphs, tree structures के लिए अच्छा नहीं।
- Not good for Graphs, tree structures.

#### 5. Memory और Hardware Load - ✓

- बड़ी queries को run करने के लिए अच्छा hardware चाहिए।
- Good hardware is required to run large queries.





### Application (उपयोग):

- ✓ • Banking System
- ✓ • School/College Database
- ✓ • Hospital Management
- ✓ • Railway Reservation System
- ✓ • E-commerce Websites
- ✓ • ERP Systems

100%

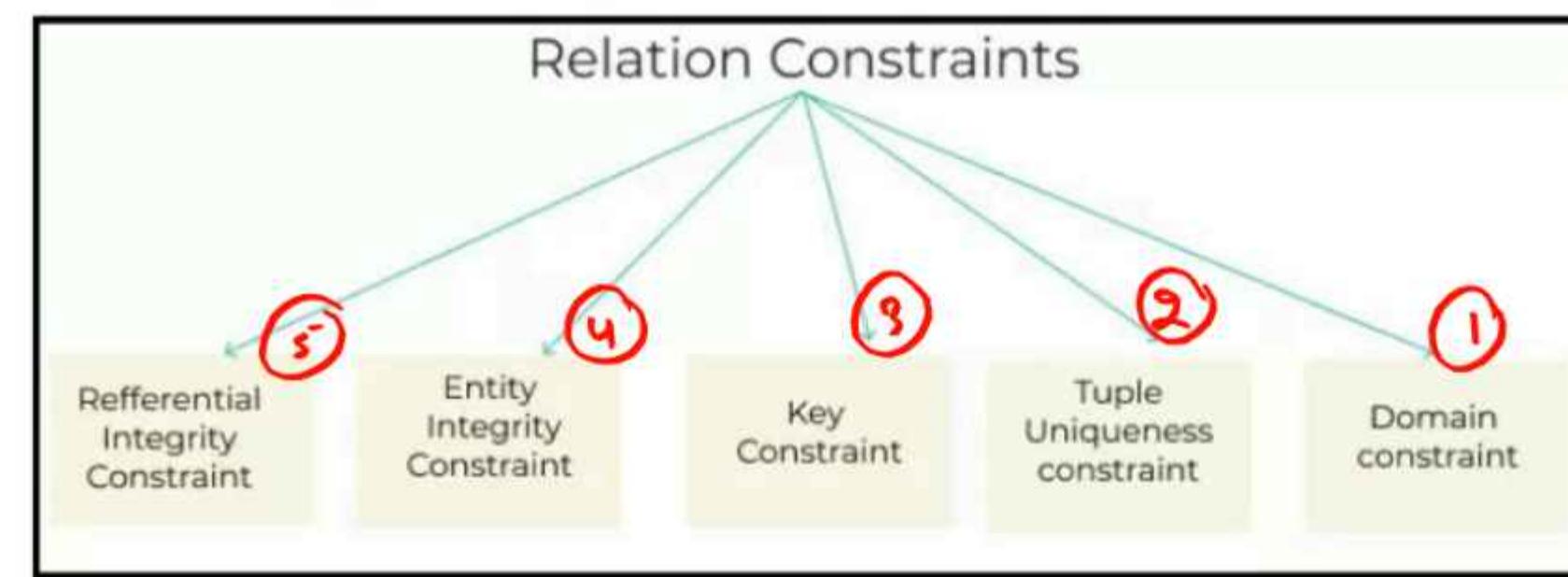
Ques: Relational data model का शाला है?

उत्तर: advantage लिया।



## रिलेशनल मॉडल प्रतिबंध (Relational model constraints): ✓ / प्रीति वादा

- रिलेशनल मॉडल constraints ऐसे नियम हैं जो डेटाबेस में डेटा की accuracy और integrity को बनाए रखने में मदद करते हैं।
- Relational model constraints are rules that help maintain data accuracy and integrity in a database.
- Relational Model** में निम्न पाँच तरह के **constraints** होते हैं।
- There are the following five types of constraints in the relational model.





## 1. Domain Constraint: ✓ [data type] check ✓

- यह constraint यह सुनिश्चित करता है कि प्रत्येक attribute (column) में केवल उसकी निर्धारित type की values ही हों।
- This constraint ensures that each attribute (column) can only have values of its specified type.

<u>StudentID</u> <u>(Integer)</u> INT	<u>Name</u> <u>(Varchar)</u> text	<u>Age (Integer)</u> INT
101 ✓	Rahul ✓	20 ✓
102 ✓	Priya ✓	Nineteen ✗ Int ✗ string ✓

*-ive value*  
→ -10, -5 ✗

→ logical value  
→ 19 ✓



### गलती(Mistake): ✓

- Age में "Nineteen" एक string है, जबकि age को integer होना चाहिए।
- "Nineteen" in Age is a string, while age must be an integer.

→ Domain Constraint का उल्लंघन तब होता है जब datatype mismatch हो जाए, या value logical limit से बाहर हो (जैसे negative age)।

→ Domain Constraint violation occurs when the datatype is mismatched, or the value is outside the logical range (such as negative age).



## 2. Tuple Uniqueness Constraint: ✓

- यह नियम कहता है कि एक टेबल में दो एक जैसे rows (tuple) नहीं होने चाहिए।
- This rule says that a table should not have two identical rows (tuples).

	StudentID	Name	Age
$R_1 \rightarrow$	101	Rahul	20
$R_2 \rightarrow$	102	Priya	19
$R_3 \rightarrow$	101	Rahul	20

A pink bracket on the right side of the table groups the last two rows (R3 and R1). A pink arrow points from the word "Same" written next to the bracket to the grouped rows, indicating that R3 is a duplicate of R1.



गलती (Mistake): ✓

- तीसरी row पहली row से पूरी तरह समान है – यानी पूरा duplicate ट्यूपल।
- The third row is exactly identical to the first row – a complete duplicate tuple.

► Relational model के अनुसार, हर ट्यूपल को यूनिक होना चाहिए।

► According to the relational model, every tuple must be unique.



### 3. Key Constraint: ✓

- Key constraints यह तय करते हैं कि कोई विशेष कॉलम या कॉलम का समूह (Key) टेबल में सभी rows को uniquely identify कर सके।
- Key constraints determine whether a particular column or group of columns (Key) can uniquely identify all rows in a table.
- Primary Key Constraint: किसी भी टेबल की Primary Key में दोहराव (duplicate) और NULL value नहीं हो सकती।
- Primary Key Constraint: The primary key of any table cannot contain duplicates or NULL values.  
    uniqueness ✓  
    null X



StudentID (Primary Key)	Name	Age
101 ✓	Rahul	20
102 ✓	Priya	19
101	Amit	22

*Duplication*

A pink bracket on the left side of the table groups the first two rows, both containing the value '101' in the 'StudentID' column. A pink circle highlights the value '101' in the third row's 'StudentID' column. The word 'Duplication' is written in pink to the left of the table, with an arrow pointing towards the grouped rows.

### गलती (Mistake):

- StudentID = 101 दो बार आ गया।
  - StudentID = 101 occurred twice.
- Primary key violation का मतलब है कि uniqueness टूट गई।
- Primary key violation means that the uniqueness is broken.



#### 4. Entity Integrity Constraint: ✓

- Entity integrity कहती है कि Primary Key किसी भी row में NULL नहीं हो सकती।
- Entity integrity says that the Primary Key cannot be NULL in any row.

StudentID <b>(Primary Key)</b>	Name	Age
101 ✓	Rahul	20
NULL ✗	Priya	19



### गलती (Mistake): ✓

- Primary Key NULL है, जिससे उस row को uniquely identify नहीं किया जा सकता।
  - Primary Key is NULL, so that row cannot be uniquely identified.
- Primary Key का मूल उद्देश्य ही **identification** है, इसीलिए NULL अलाउड नहीं है।
- The basic purpose of Primary Key is identification, hence NULL is not allowed.



## 5. Referential Integrity Constraint: ✓

- यह constraint Foreign Key पर लागू होती है और सुनिश्चित करती है कि referencing टेबल (child) में दी गई foreign key values, referenced टेबल (parent) की Primary Key में मौजूद होनी चाहिए।
- This constraint applies to Foreign Key and ensures that the foreign key values in the referencing table (child) must be present in the primary key of the referenced table (parent).

Parent Table: Student

StudentID	Name
101	Rahul
102	Priya

Child Table: Enrollment

StudentID (FK)	CourseID
101	C101
103	C102



### गलती (Mistake):

- 103 ऐसा StudentID है जो Student टेबल में नहीं है।
  - 103 is a StudentID that does not exist in the Student table.
- इसका मतलब है कि child टेबल ऐसी entity को refer कर रही है जो असल में मौजूद ही नहीं है – जो referential integrity का उल्लंघन है।
- This means that the child table is referring to an entity that does not exist – a violation of referential integrity.

Ques.  $\Rightarrow$  Relational Model Constraint का दृष्टा है, तथा  $\overline{UE}$

$\Downarrow$

किसी उकार का दृष्टा है। Explain  $\overline{an(1)}$

Sol.

$\Downarrow$



## Relational Algebra:

- Relational Algebra एक **procedural query language** है जो relational database पर operations perform करने के लिए use होता है।
- Relational Algebra is a procedural query language used to perform operations on a relational database.
- इसका **output** भी हमेशा **relation (table)** होता है।
- Its output is also always a relation (table).
- इसमें कई **operations** होते हैं, जिनमें से चार मुख्य **basic operations** हैं: Union, Intersection, Difference और Cartesian Product.
- It has many operations, of which four main basic operations are: Union, Intersection, Difference, and Cartesian Product.

PQL

→ step by step बताना पड़ता है  
इसका क्या है, कौने operation perform करना है

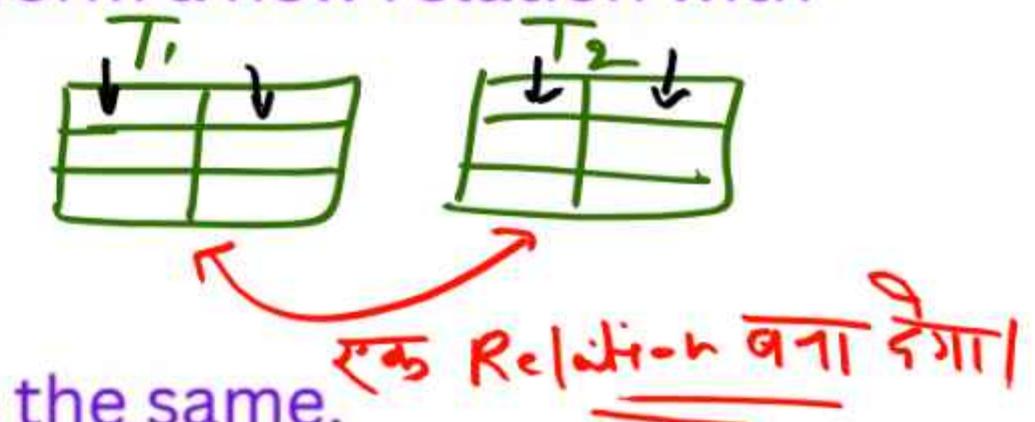


## 1. Union (U)

- Union दो relations (tables) के सभी tuples को मिलाकर एक नया relation बनाता है जिसमें duplicate tuples को हटा दिया जाता है।
- Union combines all the tuples of two relations (tables) to form a new relation with the duplicate tuples removed.

## Rule:

- दोनों relations की degree (columns की संख्या) समान होनी चाहिए।  
The degree (number of columns) of both relations must be the same. *एक Relation बना देगा।*
- Corresponding attributes के domains (Valid data type & value range) एक जैसे होने चाहिए।  
*Domain constraint को follow करता पाइए।*
- The domains (valid data type & value range) of corresponding attributes must be same.





## Example:

*Relation A* $C \Rightarrow 2$ 

ID	Name
1	Ravi
2	Priya

*Relation B* $C = 2$ 

ID	Name
2	Priya
3	Aman

 *$A \cup B$  (Union Result)*

ID	Name
1	Ravi
2	Priya
3	Aman



## 2. Intersection ( $\cap$ ):

- Intersection उन tuples को return करता है जो दोनों relations में common होते हैं।
- Intersection returns the tuples that are common in both relations.

### Rule:

- Degree और domains समान होने चाहिए।
- Degree and domains must be same.

Relation A ✓  $c=2$

ID	Name
1	Ravi
2	Priya

Relation B ✓  $c=2$

ID	Name
2	Priya
3	Aman

$A \cap B$  (Intersection Result)

ID	Name
2	Priya



### 3. Set Difference (-):

- Difference वह tuples return करता है जो पहले relation में तो हैं लेकिन दूसरे में नहीं।
- Difference returns the tuples that are in the first relation but not in the second.

**Rule:**

- दोनों relations की degree और domain समान होनी चाहिए।
- The degree and domain of both the relations should be the same.

*Relation A*

ID	Name
1	Ravi
2	Priya

✗

*Relation B*

ID	Name
2	Priya
3	Aman

*A - B (Difference Result)*

ID	Name
1	Ravi

✓

(B-A)

ID	Name
3	Aman



#### → 4. Cartesian Product (×): ✓

- Cartesian product दो relations की सभी possible combinations बनाता है।
- The Cartesian product creates all possible combinations of two relations.

#### Rule:

- डिग्री या डोमेन पर कोई प्रतिबंध नहीं होता है। ✗
  - No restriction on degree or domain.
  - Result की tuples = (rows in R1) × (rows in R2)
  - Result tuples = (rows in R1) × (rows in R2)
- Cartesian Product को Join operations बनाने के लिए बेस के रूप में इस्तेमाल किया जाता है।
- Cartesian Product is used as the base to create Join operations.

***Relation A***

ID	Name
1	Ravi
2	Priya

***Relation B***

Course
Math
English

***A × B (Cartesian Product Result)***

ID	Name	Course
1	Ravi	Math
1	Ravi	English
2	Priya	Math
2	Priya	English



## Additional Relational Algebraic Operations:

### 1. Projection ( $\pi$ ): (Pic) ✓

- Projection ऑपरेशन का उपयोग किसी रिलेशन (टेबल) से केवल चुने हुए कॉलम (Attributes) निकालने के लिए किया जाता है।
- Projection operation is used to extract only selected columns (attributes) from a relation (table).

Syntax: ✓

- $\pi<\text{attribute}_1, \text{attribute}_2, \dots>(\text{Relation})$

$\pi \text{ attribute}_1, \text{attribute}_2 (\text{tablename})$



Example: यदि एक relation **Student** है:

Roll	Name	Age	Dept
101	Ravi	20	CS
102	Neha	21	IT
103	Amit	20	CS

### Syntax:

- $\pi \text{ Name, Dept}(\text{Student})$

Name	Dept
Ravi	CS
Neha	IT
Amit	CS



Example: यदि एक relation Student है:

Roll	Name	Age	Dept
101	Ravi	20	CS
102	Neha	21	IT
103	Amit	20	CS

Syntax:

- $\pi \text{Name, Dept}(\text{Student})$

Column ✓

Row X

Name	Dept
Ravi	CS
Neha	IT
Amit	CS

 Features:

- यह डुप्लिकेट रिकॉर्ड्स को हटा देता है (set semantics)।
- This removes duplicate records (set semantics).
- सिर्फ़ कॉलम्स (attributes) को select करता है, न कि rows।
- Selects only columns (attributes), not rows.

2. Selection ( $\sigma$ ):

(Sigma)

- Selection ऑपरेशन rows को चुनता है जो किसी विशेष condition को satisfy करती हैं। इसे row filter भी कह सकते हैं।
- The selection operation selects the rows that satisfy a particular condition. It can also be called a row filter.

Row ✓      Column X

## Syntax:

- $\sigma<\text{condition}>(\text{Relation})$

$\sigma \text{Condition}(\text{TableName})$



Example: यदि एक relation **Student** है:

Roll	Name	Age	Dept
101	Ravi	20	CS
102	Neha	21	IT
103	Amit	20	CS

### Syntax:

- $\sigma \text{Age} > 20 \text{ (Student)}$

$\sigma \text{Age} > 20 \text{ (Student)}$  ✓

Roll	Name	Age	Dept
102	Neha	21	IT



### Features: ✓

- यह **rows** को फिल्टर करता है।
- This filters the rows.
- यह **vertical** नहीं horizontal slicing करता है।
- It does horizontal slicing, not vertical.

Horizontal ✓

|

Vertical X



### 3. Rename ( $\rho$ ): $(Rho)$

- Rename ऑपरेशन किसी relation या उसके attributes का नाम अस्थायी रूप से बदलने के लिए उपयोग किया जाता है। इसका प्रयोग तब जरूरी होता है जब self-join या complex expressions होते हैं।
- The Rename operation is used to temporarily rename a relation or its attributes. It is important to use it when there are self-joins or complex expressions.

#### Syntax:

- $\rho(\text{new\_relation\_name}, (\text{A}_1, \text{A}_2, \dots, \text{A}_n))(\text{Relation})$

$\rho(\text{new\_relation\_name}, (\text{new Attribute name. . . . . n}))(\text{table name})$



Example: यदि एक relation Employee है:  $\Rightarrow \text{Emp}$

<u>EID</u> EmpID	Name EName	<u>Dept</u> D
1	Ravi	CS
2	Neha	IT

### Syntax:

- $p(\text{Emp}, (\text{EmpID}, \text{EName}, \text{D}))(\text{Employee})$

$p(\text{Emp}, (\text{EmpID}, \text{EName}, \text{D}))(\text{Employee})$

<u>Emp</u>		
EmpID	EName	D
1	Ravi	CS
2	Neha	IT



### Features:

- इससे clarity और aliasing मिलती है।
- This provides clarity and aliasing.
- Complex queries में naming conflict से बचाता है।
- It prevents naming conflict in complex queries.



#### 4. Division (÷): ✓

- Division ऑपरेशन का उपयोग उन tuples को निकालने के लिए किया जाता है जो दूसरे relation के सभी tuples से जुड़ते हैं। ✓
- Division operation is used to extract tuples that join all the tuples of another relation.

#### Syntax:

- $R \div S$

$$R \div S$$

↓              ↓  
table      table  
name      name  
|              2



Example: यदि दो relation R और S हैं तो:

Relation R:

Student	Course
Aman	Math
Aman	Physics
Riya	Math
Riya	Physics
Soham	Math X

$R \div S$

Relation S:

Course
Math
Physics

out

Aman  
Riya



### Result:

Student
Aman
Riya



## Features:

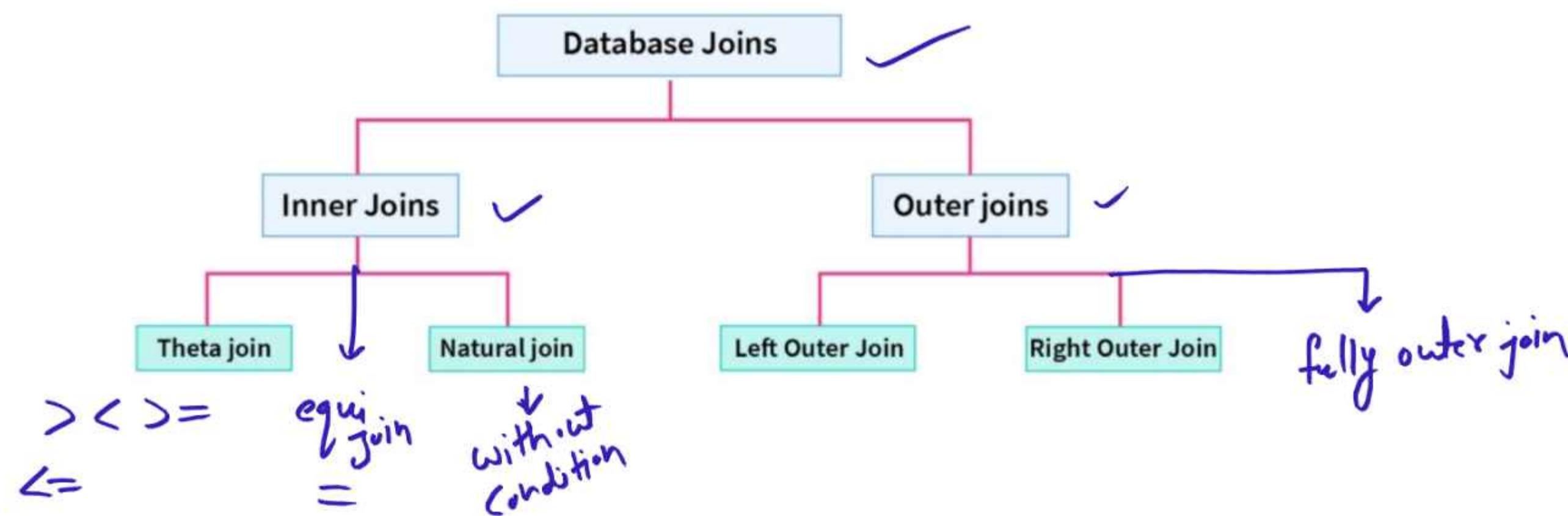
- यह "for all" condition को दर्शाता है।
- It represents the "for all" condition.
- आमतौर पर ऐसे सवालों में उपयोग होता है: ✓
- Usually used in questions like:
- "Find students who have taken all courses."



## 5. Join ( $\bowtie$ ): ✓

- Join दो relations को जोड़ता है, उन attributes के आधार पर जो common होते हैं। इसका उपयोग तब होता है जब हम दो टेबल्स से संबंधित जानकारी एक साथ देखना चाहते हैं।
- Join connects two relations based on the attributes that are common. It is used when we want to see related information from two tables together.

### Types of Joins:

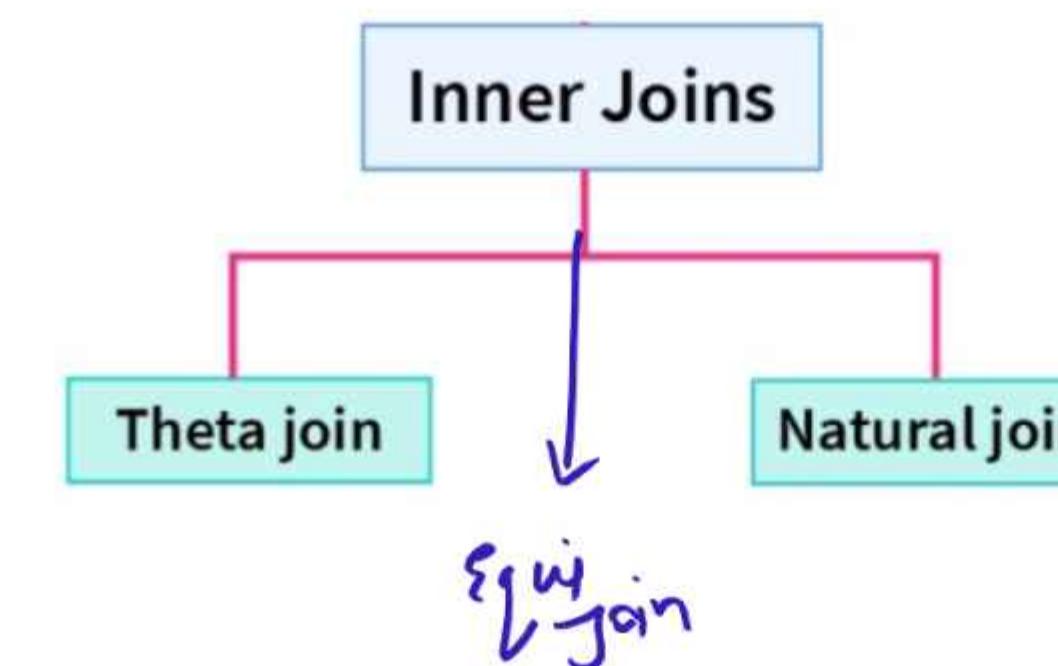




## 1. INNER JOIN:

- Inner Join दो relations के बीच केवल सिर्फ वही rows दिखाता है जिनमें दोनों tables में matching values हों। ✓
- Inner Join between two relations shows only those rows that have matching values in both the tables.

Types of Inner Join: ✓



a. Theta Join ( $\theta$ -Join): $\Theta$  $\geq, \leq$ 

- दो टेबल्स को किसी भी शर्त (comparison operator:  $=, >, <$ , etc.) के आधार पर जोड़ा जाता है।

- Inner Join between two relations shows only those rows that have matching values in both the tables.

Syntax:

Join  $\rightarrow$   $\bowtie$   
Symbol

 $R \bowtie \theta S$  $R \bowtie \theta S$ 

- जहाँ  $\theta$  एक condition है – जैसे  $R.A < S.B$



Example: यदि दो relation हैं-

Relation Employee:

EmplD	Name	Age
1	Ravi	25
2	Neha	30
3	Amit	22

Relation Retirement:

MinAge
28
35



Query: Find all employees whose age is less than the retirement age.

- Employee  $\bowtie$  (Employee.Age < Retirement.MinAge)

Result:

RMS

Employee  $\bowtie$  (Employee.Age < Retirement.MinAge) ✓

EmpID	Name	Age	MinAge
1	Ravi	25	28
2	Neha	30	35
3	Amit	22	28
3	Amit	22	35
1	Ravi	25	35



Example: यदि दो relation हैं-

Relation Student:

SID	Name	Marks	
1	Ravi	75	Python
2	Neha	60	
3	Amit	85	Python, Java

Relation Course:

CID	CourseName	MinMarks
C1	Python	70
C2	Java	80



Query: Find all students eligible for the courses based on marks (i.e., student marks  $\geq$  course MinMarks)

- Student  $\bowtie$  (Student.Marks  $\geq$  Course.MinMarks)

Result:

SID	Name	Marks	CID	CourseName	MinMarks
1	Ravi	75	C1	Python	70
3	Amit	85	C1	Python	70
3	Amit	85	C2	Java	80



### b. Equi Join (Equality Join):

- Equi Join दो relations को उस स्थिति में जोड़ता है जब उनके attributes के values बराबर (equal) हों। इसमें सिर्फ = (equality operator) का उपयोग होता है।
- Equi Join joins two relations when the values of their attributes are equal. It uses only = (equality operator).
- यह Theta Join का special case है, जिसमें condition सिर्फ equality (=) होती है।
- This is a special case of Theta Join, in which the condition is only equality (=).

Syntax:

$$R \bowtie (R.A = S.B)$$

$$R \bowtie (R.A = S.B)$$



Example: यदि दो relation हैं-

Relation Employee:

EmplID	Name	DeptID
1	Ravi	101
2	Neha	102
3	Amit	103

Relation Department:

DeptID	DeptName
101	CS
102	IT



Query: Find each employee with their department name.

- Employee  $\bowtie$  (Employee.DeptID = Department.DeptID)

Result:

Employee  $\bowtie$  (Employee.DeptID = Department.DeptID)

EmplID	Name	DeptID	DeptID	DeptName
1	Ravi	101	101	CS
2	Neha	102	102	IT

Employee  
Department



## Equi Join vs Theta Join: ✓

Feature	Theta Join	Equi Join
Condition Type	कोई भी ( <u>&gt;</u> , <u>&lt;</u> , <u>=</u> , etc)	सिर्फ = (Equality only)
General/Special	General Form ✓	✓ Special case of Theta Join
Output	Condition-based match ✓	Equal values based match ✓



### c. Natural Join:

- Natural Join दो relations को समान नाम और समान values वाले attributes के आधार पर जोड़ता है बिना duplications के। इसमें condition manually specify नहीं करनी पड़ती।
- Natural Join joins two relations based on attributes with same name and same values without duplications. In this, condition does not have to be manually specified.
- ये auto-match करता है common column को, और उन पर equality apply करता है।
- It auto-matches common columns, and applies equality on them.

Syntax:

$R \bowtie S$

$R \bowtie S$



Example: यदि दो relation हैं-

Relation Employee:

EmplID	Name	DeptID
1	Ravi	101
2	Neha	102
3	Amit	103

Relation Department:

DeptID	DeptName
101	CS
102	IT



Query: कोई condition लिखने की जरूरत नहीं होती.

- Employee ✕ Department

✓

### Result:

EmplID	Name	DeptID	DeptName
1	Ravi	101	CS
2	Neha	102	IT

✓



## Equi Join vs Natural Join:

X

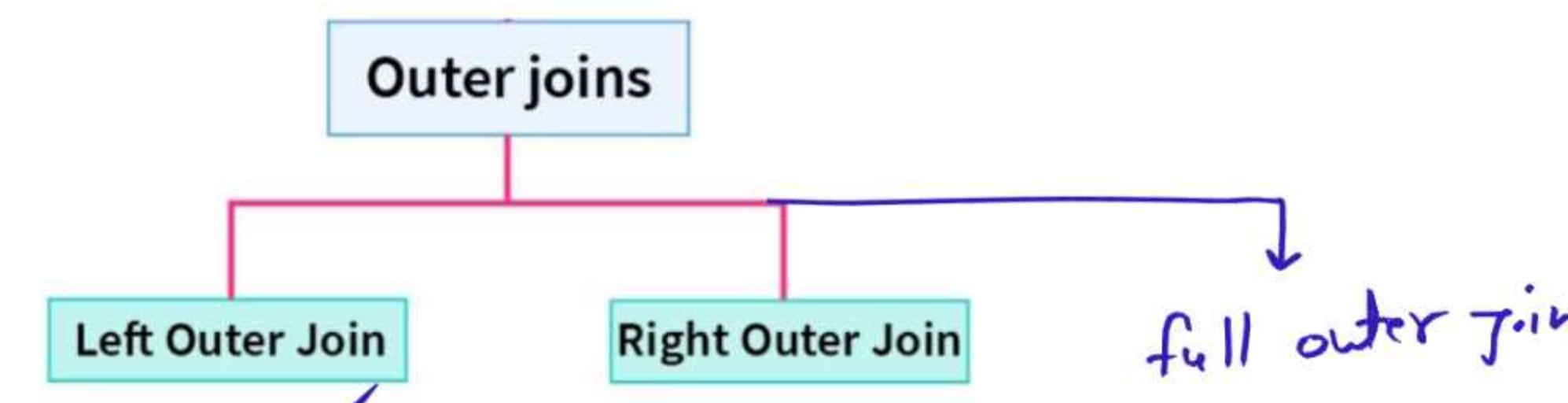
Feature	Equi Join	Natural Join
Condition Required	हों ( R.A = S.B )	नहीं
Duplicate Columns	हो सकते हैं	नहीं (Auto remove common columns)
Column Names	कुछ भी हो सकते हैं	Same name required for common columns



## 2. OUTER JOIN:

- Outer joins वो rows भी शामिल करते हैं जो एक table में हैं, लेकिन दूसरी में नहीं। और उनके लिए missing values की जगह NULL भर दी जाती है।
- Outer joins also include rows that are present in one table but not in the other. And for them, missing values are filled with NULL.

### Types of Outer Join:





### → a. Left Outer Join: ↗

- Left table की सभी rows दिखाता है – अगर right table में match नहीं है, तो NULL डाल देता है।
- Shows all rows in the left table - if there is no match in the right table, returns NULL.

Syntax:

$R \bowtie S$

$R \bowtie S$



Example: यदि दो relation हैं-

Relation Student :

SID	Name	Class
1	Ravi	10
2	Neha	11
3	Amit	12

Relation Class :

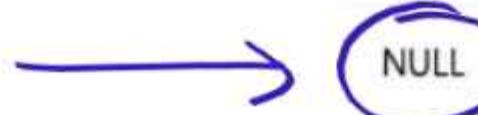
Class	Teacher
10	Mr. A
11	Ms. B



## Query:

- Student  $\bowtie$  Class

## Result:

SID	Name	Class	Teacher
1	Ravi	10	Mr. A
2	Neha	11	Ms. B
3	Amit	12	 NULL



### b. Right Outer Join: ✓

- Right table की सभी rows दिखाता है – unmatched left rows के लिए **NULL** देता है।
- Shows all rows of the right table – returns NULL for unmatched left rows.

Syntax:

$R \bowtie S$

$R \bowtie S$



Example: यदि दो relation हैं-

Relation **Student**:

SID	Name	Class
1	Ravi	10
2	Neha	11
3	Amit	12

Relation **Class**:

Class	Teacher
10	Mr. A
11	Ms. B
13	Mr. C

Right



## Query:

- Student ✕ Class

## Result:

SID	Name	Class	Teacher
1	Ravi	10	Mr. A
2	Neha	11	Ms. B
NULL	NULL	13	Mr. C

X



### c. Full Outer Join ✓

- दोनों tables की सभी rows दिखाता है – matched + unmatched दोनों।
- Shows all rows of both tables – both matched + unmatched.

Syntax:

$R \bowtie S$

$R \bowtie S$



Example: यदि दो relation हैं-

Relation Student :

SID	Name	Class
1	Ravi	10
2	Neha	11
3	Amit	12

Relation Class :

Class	Teacher
10	Mr. A
11	Ms. B
13	Mr. C



## Query:

- Student ✕ Class

## Result:

SID	Name	Class	Teacher
1	Ravi	10	Mr. A
2	Neha	11	Ms. B
3	Amit	12	NULL
NULL	NULL	13	Mr. C



## Comparision Between Inner & Outer Join

Join Type	Includes Matched	Includes Unmatched Left	Includes Unmatched Right
Inner Join	✓	✗	✗
Left Outer Join	✓	✓	✗
Right Outer Join	✓	✗	✓
Full Outer Join	✓	✓	✓

Ques:  $\Rightarrow$  Join की समझाओ।  
Explain Join. ✓

①  A diagram showing two tables side-by-side. The first table has three columns labeled A, B, and C. The second table also has three columns labeled A, B, and C. The 'A' column is underlined in both tables, and the 'B' column is also underlined in both tables, indicating they are common columns used for joining.



## Relational Calculus (RC): ✓

→ Step by step X

- Relational Calculus एक **non-procedural query language** है जो Relational Database से data निकालने के लिए use होती है। ✓

- Relational Calculus is a non-procedural query language which is used to extract data from Relational Database.

- इसमें हम क्या चाहिए (What to retrieve) बताते हैं, कैसे निकालना है (How to retrieve) ये नहीं बताते।
- In this we tell what we want (What to retrieve) but do not tell how to retrieve (How to retrieve).  
What ? How ? X

👉 यानी यह Declarative language है (जैसे SQL का SELECT).

👉 That means it is a Declarative language (like SELECT of SQL).

**"It focuses on what to retrieve, not on how to retrieve it"**



## 1. Types of Relational Calculus

- Relational Calculus के 2 प्रकार होते हैं: ✓
- There are 2 types of Relational Calculus:

### (a) Tuple Relational Calculus (TRC): ✓

- TRC में हम tuple variables का इस्तेमाल करते हैं और एक condition लिखते हैं।
- In TRC we use tuple variables and write a condition.

**Syntax:** { T | P(T) } → "T tuple ऐसा Select करो जहाँ P(T) True हो"

जहाँ (Where)

- T = tuple variable ✓
- P(T) = predicate (condition) on T



Example with Table:

Student Table

RollNo	Name	Course	Marks
1	Riya	BCA	85
2	Aman	B.Tech	76
3	Priya	BCA	92
4	Rahul	BBA	65

t  
tuple  
tuple

Result:

Name
Riya
Priya

→ [and] A

Example-1: Show the names of all the students who are in BCA course.

TRC Expression: { t.Name | t ∈ Student  $\wedge$  t.Course = 'BCA' }

मतलब: "ऐसे tuples T के Name निकालो, जो STUDENT relation में हैं और जिनका Course = 'BCA' है।"

Meaning: "Find the names of tuples T that are in the STUDENT relation and whose Course = 'BCA'."



Example 2: Find the Roll No and Name of the students whose marks are more than 70.

TRC Expression: {t.RollNo, t.Name | t ∈ Student ∧ t.Marks > 70}

Marks > 70

मतलब: "ऐसे tuples के RollNo और Name निकालो जिनके Marks 70 से ज्यादा हैं।"

Meaning: "Retrieve the RollNo and Name tuples whose marks are more than 70."

Student Table

RollNo	Name	Course	Marks
1 ✓	Riya ✓	BCA	85
2 ✗	Aman ✓	B.Tech	76
3 ✓	Priya ✗	BCA	92
4	Rahul	BBA	65

{t.RollNo, t.Name | t ∈ Student ∧ t.Marks > 70}

Result:

RollNo	Name
1	Riya
2	Aman
3	Priya



Example 3: Find the names of those students whose marks are less than 90 and course is BCA.

TRC Expression: {t.Name | t ∈ Student ∧ t.Course = 'BCA' ∧ t.Marks < 90}

{t.Name | t ∈ Student ∧ t.Mark < 90 ∧ Course = 'BCA'}

Student Table

RollNo	Name	Course	Marks
1	Riya	BCA	85
2	Aman	B.Tech	76
3	Priya	BCA	92
4	Rahul	BBA	65

Result:

Name
Riya



Example 4: Find the names of students whose course is 'BCA' or whose marks are more than 80.

TRC Expression: {t.Name | t ∈ Student ∧ (t.Course = 'BCA' ∨ t.Marks > 80)}

Student Table

RollNo	Name	Course	Marks
1	Riya	BCA	85 ✓
2	Aman	B.Tech	76
3	Priya	BCA	92 ✓
4	Rahul	BBA	65

and →  $\wedge$   
or →  $\vee$

$\{t.Name | t \in \text{Student} \wedge (t.Course = 'BCA' \vee t.Marks > 80)\}$

Result:

Name
Riya
Priya

Course = BCA  
or  
Mark > 80



→ TRC के महत्वपूर्ण बिंदु (Important points of TRC): ✓

- TRC में tuple variable (जैसे T) का उपयोग किया जाता है।
- Tuple variables (such as T) are used in TRC.
- TRC non-procedural है – इसमें केवल शर्त लिखी जाती है, तरीका नहीं।
- TRC is non-procedural—only conditions are written in it, not methods.
- Output relation में केवल वही attributes आते हैं जिन्हें {} के अंदर लिखा जाता है।
- The output relation only includes attributes enclosed within {}.
- Logical operators जैसे AND ( $\wedge$ ), OR ( $\vee$ ), NOT ( $\neg$ ) का उपयोग किया जा सकता है।
- Logical operators such as AND ( $\wedge$ ), OR ( $\vee$ ), and NOT ( $\neg$ ) can be used.

AND  $\wedge$       NOT  $\neg$   
OR  $\vee$       EXIST  $\exists$



## (b) Domain Relational Calculus (DRC):

- इसमें query को **attributes (columns)** के आधार पर लिखा जाता है –
- In this, the query is written on the basis of attributes (columns) –
- यानि TRC में जहाँ **tuple (row) variable** का प्रयोग होता है, वहीं DRC में **domain (column values) variables** का उपयोग होता है।
- That is, while tuple (row) variables are used in TRC, domain (column values) variables are used in DRC.

**Syntax:** {  $x_1, x_2, \dots, x_n$  |  $P(x_1, x_2, \dots, x_n)$  }

**जहाँ (Where)**

- $x_1, x_2 \dots x_n$  = **domain variables (attributes)**
- $P(\dots)$  = **predicate / शर्त (condition)** है जो पूरी होनी चाहिए



Example with Table:

Student Table

RollNo	Name	Age	Dept
1	Ram	18	CSE
2	Sita	19	IT
3	Mohan	18	CSE
4	Rohan	20	ECE

$\exists$  = Exist

Condition

Query: Find the Roll No and Name of those students whose Age = 18.

DRC Expression:

{  $r, n | \exists a, d ( \text{Student}(r, n, a, d) \wedge a = 18 )$  }

Result:

RollNo	Name
1	Ram
3	Mohan

{  $r, n | \exists a, d ( \text{Student}(r, n, a, d) \wedge a = 18 )$  }



Example with Table:

### Student Table

RollNo	Name	Course	Marks
1	Riya	BCA	85
2	Aman	B.Tech	76
3	Priya	BCA	92
4	Rahul	BBA	65

Result:

Name
Riya
Priya

Course = 'BCA'

Example-1: Show the names of all the students who are in BCA course.

DRC Expression: {<N> |  $\exists R,C,M \text{ (Student}(R,N,C,M) \wedge C = \text{'BCA'})\}$ }

मतलब: ऐसे domain variable N (Name) को निकालो जहाँ COURSE = 'BCA' हो।

Meaning: Find the domain variable N (Name) where COURSE = 'BCA'.

{<N> |  $\exists r,c,m \text{ (Student}(r,n,c,m) \wedge c = \text{'BCA'})\}$ }



Example 2: Find the names and marks of those students whose marks are more than 80.

DRC Expression: { $\langle N, M \rangle \mid \exists R, C (\text{Student}(R, N, C, M) \wedge M > 80)$ }

$M > 80$

मतलब: ऐसे Name (N) और Marks (M) निकालो जिनके Marks 80 से ज़्यादा हैं।

Meaning: Find the Names (N) and Marks (M) whose marks are more than 80.

Student Table

RollNo	Name	Course	Marks
1	Riya	BCA	85
2	Aman	B.Tech	76
3	Priya	BCA	92
4	Rahul	BBA	65

{ $\langle n, m \rangle \mid \exists r, c (\text{Student}(r, n, c, m) \wedge m > 80)$ }

Result:

Name	Marks
Riya	85
Priya	92



Example 3: Find the names of those students whose course is 'BCA' and marks are less than 90.

DRC Expression: {<N> |  $\exists R,C,M (\text{Student}(R,N,C,M) \wedge C = 'BCA' \wedge M < 90 )$ }

Student Table

$\{<n> | \exists r,c,m (\text{student}(r,n,c,m) \wedge c = 'BCA' \wedge m < 90)\}$

Result:

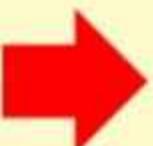
RollNo	Name	Course	Marks
1	Riya	BCA	85
2	Aman	B.Tech	76
3	Priya	BCA	92
4	Rahul	BBA	65

Name
Riya

*Thank  
you*



## Unit-4 : Relational Database Design



## Syllabus :

## UNIT 4: Relational Database Design :

(100):

5 marks



10 Marks

(10 Periods)

Functional dependencies and normalization for relational databases, Types of Normalization (1NF, 2NF, 3NF, multivalued dependencies and BCNF, Forth Normal Form, Fifth Normal Form).

- ✓ Ques → Different types के Normal form को साझा करो। Ans
- ✓ Ques → INF, 2NF & 3NF को साझा करो। Ans
- ✓ Ques → BCNF, 4NF & 5th NF को साझा करो। Ans
- 1) Ques → Write a short note — Ans 2.5 marks
- ① functional dependency

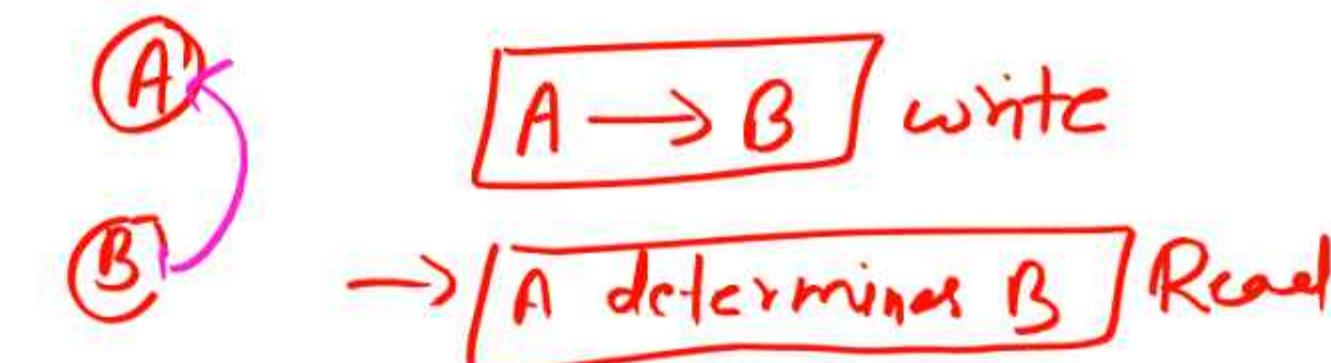


## Functional Dependency (FD): ✓

- Functional Dependency यह बताता है कि **किसी कॉलम का मान (value)** **किसी दूसरे कॉलम पर निर्भर है।**
- Functional Dependency indicates that the value of a column is dependent on another column.
- Attribute B, attribute A पर functionally dependent है अगर हर A के लिए एक unique B होता है।
- Attribute B is functionally dependent on attribute A if for every A there is a unique B.

Symbol:  $A \rightarrow B$

Pronounce: A determines B





उदाहरण (Example): मान लीजिए एक **table** है

Roll_No	Name	Dept
101	Ravi	CS
102	Seema	IT
103	Aman	CS

यहाँ पर (Here):

- $\text{Roll\_No} \rightarrow \text{Name}$  क्योंकि  $\text{Roll\_No}$  uniquely student का नाम बताता है।
- $\text{Roll\_No} \rightarrow \text{Dept}$  क्योंकि  $\text{Roll\_No}$  से हम उसका department भी जान सकते हैं।

यानी Roll\_No के बदलने से Name और Dept दोनों बदल सकते हैं।

That means by changing the Roll\_No, both Name and Dept can change.



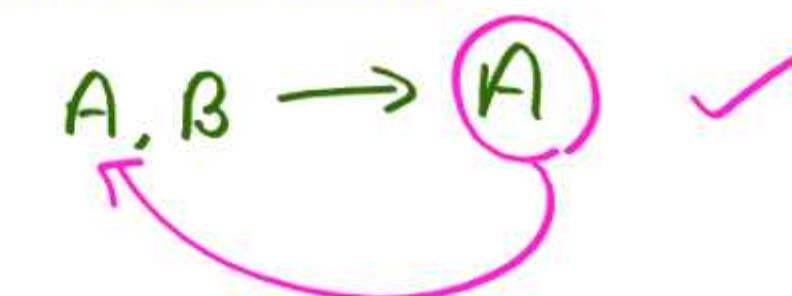
## Types of Functional Dependency:

### 1. Trivial Dependency:

- जब right-side वाला column, left-side में पहले से ही मौजूद हो।
- When the right-side column already exists on the left-side.

#### Example:

- $\text{Roll\_No}, \text{Name} \rightarrow \text{Name}$

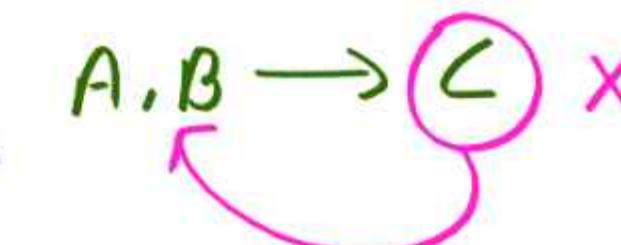


### 2. Non-Trivial Dependency:

- जब right-side वाला column, left-side का हिस्सा नहीं होता।
- When the right-side column is not part of the left-side.

#### Example:

- $\text{Roll\_No} \rightarrow \text{Name}$





## Normalization:

~~Table~~

- Normalization डेटाबेस डिजाइन की एक तकनीक है जिसका उपयोग टेबल को इस तरह से विभाजित करने और रिलेशनशिप बनाने में किया जाता है जिससे डाटा का दोहराव (redundancy) कम हो जाए और डाटा हमेशा सही (consistent) बना रहे।
- Normalization is a database design technique used to divide tables and create relationships in such a way that redundancy is reduced and the data remains consistent.
- Normalization का मतलब है एक बड़े टेबल को छोटे-छोटे टेबल्स में तोड़ना और उनके बीच सही संबंध (relation) बनाना, ताकि डाटा को बार-बार न दोहराना पड़े और अपडेट, डिलीट या इंसर्ट करते समय कोई गलती न हो।
- Normalization means breaking a large table into smaller tables and creating proper relationships between them, so that the data is not repeated again and again and there is no mistake while updating, deleting or inserting.



उदाहरण (Example): मान लीजिए एक टेबल है (Suppose there is a table)-

Student_ID	Name	Dept_Name	Dept_Location
101	Ravi	Computer	Block A
102	Seema	Computer	Block A
103	Mohan	Electronics	Block B

- यहां "Computer" और "Block A" दो बार आ रहा है – यह **redundancy** है।
- Here "Computer" and "Block A" appear twice - this is redundancy.
- Normalization में इसे दो टेबल्स में बाँटा जाएगा:
- Normalization will split this into two tables:



## Student Table: ✓

Student_ID	Name	Dept_ID
101	Ravi	1
102	Seema	1
103	Mohan	2

## Department Table:

Dept_ID	Dept_Name	Dept_Location
1	Computer	Block A
2	Electronics	Block B

- अब "Computer" और "Block A" एक ही बार स्टोर हुआ है – यही है Normalization!
- Now "Computer" and "Block A" are stored only once - this is Normalization.



## → Purpose of Normalization: ✓

- Normalization का मुख्य उद्देश्य database को इस तरह design करना होता है जिससे:
- The main purpose of normalization is to design the database in such a way that:

### 1. Data redundancy को कम किया जाए (Data redundancy should be reduced)

- एक ही data को बार-बार store करने से बचने के लिए।
- To avoid storing the same data again and again.
- Example: अगर हर बार employee की department details दोहराई जाती हैं, तो redundancy है।
- Example: If employee's department details are repeated every time, then there is redundancy.



## 2. Data inconsistency को रोकना (Preventing data inconsistency)

- अगर एक ही जानकारी कई जगह store है, और उनमें बदलाव अलग-अलग तरीके से होता है, तो inconsistency हो जाती है।
- If the same information is stored in multiple places and changes are made to them in different ways, then inconsistency occurs.
- **Normalization** से data केवल एक जगह रहता है → consistency बनी रहती है।
- Normalization stores data in only one place → consistency is maintained.

## 3. Data integrity बनाए रखना (Maintaining data integrity) ✓

- **Referential integrity** और other constraints से data हमेशा valid और सही रहता है।
- Referential integrity and other constraints ensure that data always remains valid and correct.



#### 4. Efficient data organization ✓

- Tables logically structured होती हैं → maintenance, updates और queries आसान होती हैं।
- Tables are logically structured → maintenance, updates and queries are easy.

#### 5. Storage space को optimize करना (Optimizing Storage Space) ✓

- Duplicate data को हटा कर space बचाई जाती है।
- Space is saved by removing duplicate data.

#### 6. Scalability और flexibility बढ़ाना (Increasing scalability and flexibility)

- अगर future में database को modify करना हो, तो normalized database को manage करना आसान होता है। ✓
- If the database needs to be modified in the future, a normalized database is easier to manage.



## 1NF (First Normal Form): ✓

1NF (First Normal Form) कहता है कि किसी भी table में:

⇒ 1NF (First Normal Form) states that in any table:

1. सभी attributes (columns) के values atomic (सिंगल वैल्यू) होने चाहिए।

The values of all attributes (columns) must be atomic (single value).

✓ 2. कोई repeating group (duplicate या multi-valued column) नहीं होना चाहिए।

There should be no repeating groups (duplicate or multi-valued columns).



### Step-by-Step Process to Normalize a Table into 1NF:

- ◆ Step 1: Identify the Unnormalized Table (UNF)

- मान लो हमारे पास एक table है / Suppose we have a table:

RollNo	Name	Subjects
101	Rahul	Math, Physics
102	Priya	Chemistry
103	Aman	Physics, Chemistry

👉 यहाँ पर Subjects column में multiple values हैं – ये table 1NF में नहीं है।

👉 Here the Subjects column has multiple values – this table is not in 1NF.



## ◆ Step 2: Remove Multivalued Attributes

- अब हर subject को अलग row में रखो / Now put each subject in a separate row:

RollNo	Name	Subject
101	Rahul	Math
101	Rahul	Physics
102	Priya	Chemistry
103	Aman	Physics
103	Aman	Chemistry

👉 अब हर attribute atomic है – अब यह **table 1NF में है।**

👉 Now every attribute is atomic – this table is now in 1NF.



### Conditions for a Table to be in 1NF

**1. Single-valued attributes:** हर column में केवल एक ही value होनी चाहिए।

1. Single-valued attributes: Each column should have only one value.

**2. No repeating groups:** एक ही टाइप का डेटा बार-बार ना हो (जैसे Subject1, Subject2, Subject3 etc.)

2. No repeating groups: Same type of data should not be repeated (e.g. Subject1, Subject2, Subject3 etc.)

**3. Unique rows:** हर row अलग होनी चाहिए (primary key का उपयोग करके)।

3. Unique rows: Each row should be unique (using primary key).

Before NF

EmplID	Name	Skills
1	Ramesh	Java, Python
2	Suresh	SQL

Before NF

EmplID	Name	Skills
1	Ramesh	Java
1	Ramesh	python
2	Suresh	SQL



## Partial Dependency:

- Partial Dependency तब होती है जब कोई non-prime attribute (जो primary key का हिस्सा नहीं है), composite key के सिर्फ एक हिस्से पर depend करता हो, ना कि पूरे primary key पर।
- Partial Dependency occurs when a non-prime attribute (which is not part of the primary key) depends on only a part of the composite key, not the entire primary key.

**Example: मान लीजिए एक टेबल है - StudentCourse Table**

	StudentID	CourseID	StudentName	CourseName
Composite key	101	C1	Ravi	DBMS
	102	C2	Seema	CN

2 minute  
Break

यहाँ Primary Key = (StudentID, CourseID) → Composite है



**Note:**

- StudentName **सिर्फ StudentID पर depend करता है**
- StudentName depends only on StudentID
- CourseName **सिर्फ CourseID पर depend करता है**
- CourseName depends only on CourseID

👉 यानी दोनों attributes पूरे **primary key (StudentID, CourseID)** पर depend नहीं कर रहे हैं - बल्कि केवल एक **हिस्से** पर कर रहे हैं।

👉 That means both the attributes are not dependent on the entire primary key (StudentID, CourseID) - but only on a part of it.

✓ यही है **Partial Dependency / This is Partial Dependency**



### Why is Partial Dependency Bad?

- इससे Redundancy (बार-बार डाटा दोहराना) होती है
- This leads to redundancy (repeating data again and again)
- अगर हम एक जगह बदलाव करें और दूसरी जगह भूल जाएं → Inconsistency हो जाती है
- If we make a change in one place and forget about it in another place → Inconsistency occurs



→ Note: How to Fix Partial Dependency? (2NF में)

Table को 3 parts में तोड़ो / Break the table into 3 parts:

1. Student Table ✓

► ( $\text{StudentID} \rightarrow \text{StudentName}$ )

2. Course Table ✓

► ( $\text{CourseID} \rightarrow \text{CourseName}$ )

3. StudentCourse Table ✓

► ( $\text{StudentID}, \text{CourseID}$ )



## 2NF (Second Normal Form): ✓

Second Normal Form (2NF) कहता है कि: / Second Normal Form (2NF) states that:

1. Table को पहले 1NF में होना ज़रूरी है ✓

The table must first be in 1NF

2. और उसके बाद Table में कोई भी **partial dependency** नहीं होनी चाहिए।

and that the table must not have any partial dependencies.

✓ 2NF में हर non-prime attribute पूरे primary key पर depend होना चाहिए, ना कि उसके हिस्से पर।

✓ In 2NF, every non-prime attribute must depend on the entire primary key, not on a part of it.



## Step-by-Step Process for 2NF Normalization:

### ◆ Step 1: Table should be in 1NF

- पहले table को 1NF में लाओ – यानी हर column में atomic values हों और कोई repeating group ना हो।
- First bring the table to 1NF – that is, every column should have atomic values and there should be no repeating groups.

### ◆ Step 2: Check for Composite Primary Key

- अगर table में composite primary key है, तो check करो कि कोई non-key attribute सिर्फ primary key के हिस्से पर तो depend नहीं है।
- If the table has a composite primary key, then check that no non-key attribute depends only on a part of the primary key.



◆ Step 3: Remove Partial Dependencies ✓

- ऐसे attributes को अलग table में निकालो जो सिर्फ primary key के हिस्से पर depend कर रहे हों।
- Pull out attributes that depend only on part of the primary key into a separate table.

Example: *Composite key*

StudentID	CourseID	StudentName	CourseName	Instructor
1	C101	Aman	Math	Mr. Roy
2	C102	Nisha	Physics	Ms. Neha

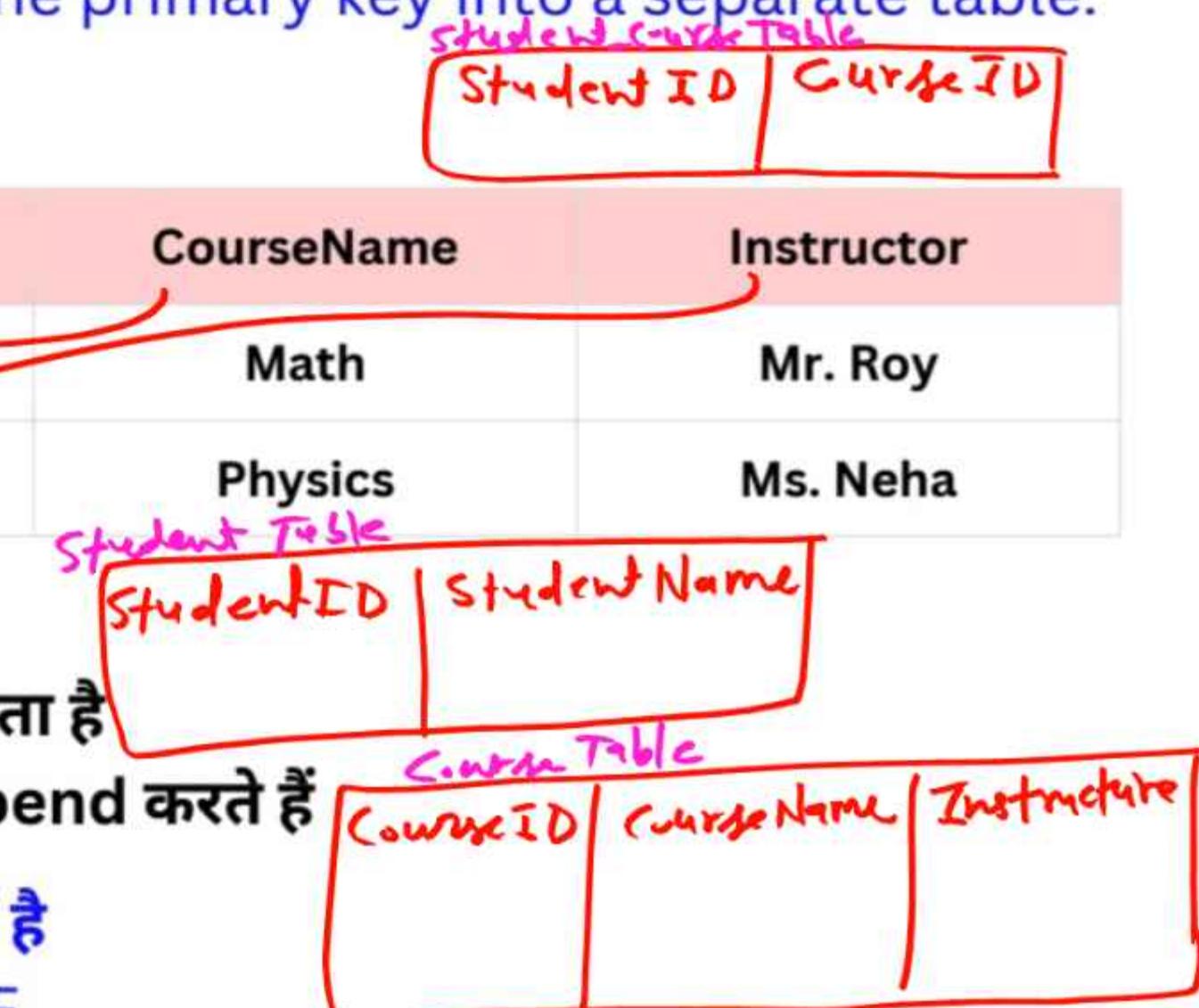
◆ Primary Key = (StudentID, CourseID) ✓

◆ लेकिन StudentName सिर्फ StudentID पर depend करता है

◆ और CourseName, Instructor सिर्फ CourseID पर depend करते हैं

👉 इसलिए यह partial dependency है ⇒ Table 2NF में नहीं है

👉 So this is partial dependency ⇒ Table is not in 2NF





✓ After Applying 2NF ✓

- अब table को 3 parts में divide करते हैं / Now let us divide the table into 3 parts:

### 1. Student Table ✓

StudentID	StudentName
1	Aman
2	Nisha

### 2. Course Table ✓

CourseID	CourseName	Instructor
C101	Math	Mr. Roy
C102	Physics	Ms. Neha

### 3. Enrollment Table (Relationship) ✓

StudentID	CourseID
1	C101
2	C102

2NF



### Transitive Dependency: ✓

- जब एक non-key attribute किसी primary key पर directly नहीं, बल्कि किसी दूसरे non-key attribute के ज़रिए depend करता है, तो उसे Transitive Dependency कहते हैं।
- When a non-key attribute depends on a primary key not directly but through another non-key attribute, it is called Transitive Dependency.

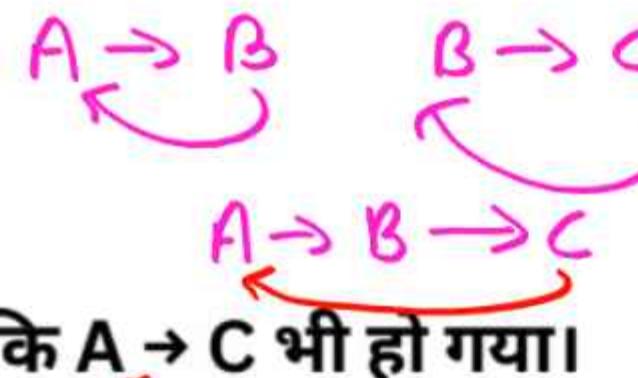
अगर / If

$A \rightarrow B$  ✓

&  $B \rightarrow C$

तो इसका मतलब है कि  $A \rightarrow C$  भी हो गया।

So this means  $A \rightarrow C$  also happened.



3NF →

① 2NF ✓

② Transitive dependency ✗

यहाँ A primary key है, लेकिन C directly A पर नहीं, बल्कि B के ज़रिए depend कर रहा है।

Here A is the primary key, but C is not directly dependent on A but through B.

👉 यही Transitive Dependency है। / This is Transitive Dependency.



Example: मान लीजिए एक टेबल है / Suppose there is a table- ✓

StudentID	StudentName	DeptID	DeptName
101	Aman	D1	CSE
102	Neha	D2	ECE

StudentID → DeptName

StudentID → Dep ID

DeptID → DeptName

- Primary Key: StudentID

- StudentName और DeptID – दोनों StudentID पर directly depend करते हैं (👍 सही है)

- StudentName and DeptID – both depend directly on StudentID (👍 is correct)

- लेकिन DeptName StudentID पर directly depend नहीं करता

- But DeptName does not depend directly on StudentID.

- वो तो DeptID पर depend करता है

- It depends on DeptID.



अब Relation देखो / Now look at the relation:

- $\text{StudentID} \rightarrow \text{DeptID}$
- $\text{DeptID} \rightarrow \text{DeptName}$

तो हम कह सकते हैं / So we can say:

👉  $\text{StudentID} \rightarrow \text{DeptName}$  (Transitive Dependency)





→ Note: क्यों Transitive Dependency गलत है? / Why Transitive Dependency is wrong?

- अगर Department का नाम बदलता है, तो हर student row में उसे बार-बार बदलना पड़ेगा
- इससे redundancy और update anomalies होती हैं
- इसलिए इसे 3NF में हटाया जाता है
- If the department name changes, it will have to be changed again and again in every student row
- This leads to redundancy and update anomalies
- So it is eliminated in 3NF



## 3NF (Third Normal Form): ✓

कोई भी table तब 3NF में होता है जब:

1. वो पहले 2NF में हो
2. और उसमें **transitive dependency** ना हो

A table is in 3NF if:

1. it is previously in 2NF
2. and it does not contain transitive dependencies

3NF कहता है कि - कोई भी **non-prime attribute (non-key attribute)** किसी **non-key attribute** पर **depend** नहीं होना चाहिए।

3NF states that – No non-prime attribute (non-key attribute) should depend on any non-key attribute.



### Step-by-Step Process for 3NF Normalization

- ◆ Step 1: Table should be in 2NF
  - यानी no partial dependency होनी चाहिए।
  - That means there should be no partial dependency.
- ◆ Step 2: Identify Transitive Dependencies
  - देखो कि कोई non-key attribute किसी और non-key attribute पर तो depend नहीं कर रहा।
  - Check that no non-key attribute is dependent on any other non-key attribute.
- ◆ Step 3: Remove Transitive Dependencies
  - ऐसे attributes को एक नया table बना कर अलग कर दो।
  - Separate such attributes by creating a new table.



Example: 2NF Table:

StudentID	StudentName	DeptID	DeptName	HOD
1	Aman	D1	CSE	Dr. Verma
2	Nisha	D2	ECE	Dr. Ahuja

◆ Primary Key = StudentID

### Analysis:

- StudentName directly depends on StudentID ✓
- लेकिन DeptName और HOD depend करते हैं DeptID पर
- और DeptID depend करता है StudentID पर
- ➡ यानि DeptName और HOD → indirectly StudentID पर depend कर रहे हैं
- => ये transitive dependency है ✓



→ 3NF Apply करने के बाद Tables (Tables after applying 3NF):

### 1. Student Table

StudentID	StudentName	DeptID
1	Aman	D1
2	Nisha	D2

### 2. Department Table:

DeptID	DeptName	HOD
D1	CSE	Dr. Verma
D2	ECE	Dr. Ahuja



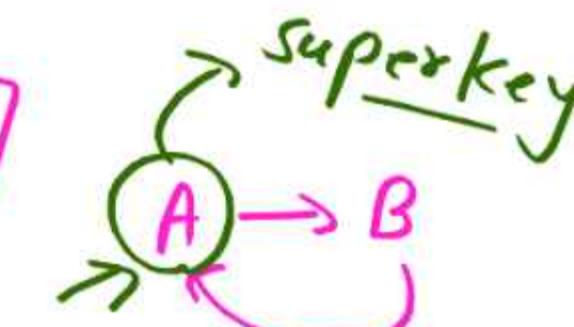
## → BCNF (Boyce-Codd Normal Form):

कोई table BCNF में होता है जब:

- वो 3NF में हो  $\rightarrow$  3NF ✓
- और हर functional dependency  $A \rightarrow B$  में  $A$  एक super key हो

A table is in BCNF when:

- It is in 3NF
- And A is a super key in every functional dependency  $A \rightarrow B$



मतलब: हर determinant (जो attribute किसी और attribute को uniquely पहचानता है)  $\rightarrow$  super key होना चाहिए।

Meaning: Every determinant (attribute that uniquely identifies another attribute)  $\rightarrow$  must be a super key.



→ BCNF क्यों ज़रूरी है? (Why is BCNF important?) ✓

- कई बार table 3NF में होता है, लेकिन उसमें candidate key के अलावा भी कोई दूसरा attribute determinant बन जाता है।
- ऐसे case में 3NF तो true होता है, लेकिन design सही नहीं होता।

इसलिए हम BCNF apply करते हैं।

- Sometimes a table is in 3NF, but an attribute other than the candidate key becomes a determinant.
- In such cases, 3NF is true, but the design is not correct.

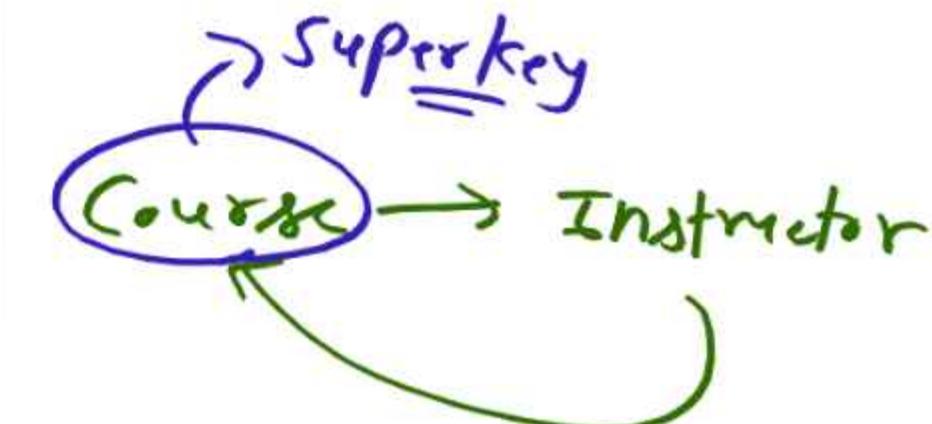
Therefore, we apply BCNF.



Example: मान लीजिए एक टेबल है- 3NF में है, लेकिन BCNF में नहीं (Suppose there is a table which is in 3NF but not in BCNF.)

StudentID	Course	Instructor
101	Math	Prof. A
101	Physics	Prof. B
102	Math	Prof. A
102	Physics	Prof. B

Candidate key / superkey



### Functional Dependencies:

1. StudentID + Course → Instructor (Candidate Key)
2. Course → Instructor

👉 यहाँ Course से हम Instructor को uniquely पहचान सकते हैं, लेकिन Course super key नहीं है, फिर भी वो determinant है।

👉 Here we can uniquely identify the Instructor through Course, but Course is not a super key, still it is determinant.

=> This violates BCNF. ✓



BCNF Apply करने के बाद (After applying BCNF): ✓

Table 1: Course\_Instructor

Course	Instructor
Math	Prof. A
Physics	Prof. B

Table 2: Student\_Course ✓

StudentID	Course
101	Math
101	Physics
102	Math
102	Physics

अब कोई non-superkey attribute किसी को determine नहीं कर रहा।

Now no non-superkey attribute is defining anything.

✓ अब ये table BCNF में हैं। / This table is now in BCNF.



### Multivalued Dependency (MVD): ✓

- जब एक attribute पर एक से ज्यादा independent values depend करती हैं, और वो values किसी और attribute से independent होती हैं, तो उसे Multivalued Dependency कहते हैं।
- When more than one independent value depends on an attribute, and those values are independent from any other attribute, then it is called Multivalued Dependency.

Example: मान लीजिए एक टेबल है-

Student (StudentID, Hobby, Phone)

- एक Student के कई Hobby हो सकते हैं
- एक Student के कई Phone number भी हो सकते हैं
- लेकिन Hobby और Phone आपस में related नहीं हैं X

तो यहां पर:

- StudentID  $\rightarrow\!\!\! \rightarrow$  Hobby } X
- StudentID  $\rightarrow\!\!\! \rightarrow$  Phone } X

4NF →

① BCNF ✓

② Multivalued dependency X



- ये दोनों multivalued dependencies हैं, क्योंकि: ✓
- StudentID से Hobby की list निकलती है
- StudentID से Phone की list निकलती है
- लेकिन Hobby और Phone एक-दूसरे से कोई लेना-देना नहीं रखते
- These two are multivalued dependencies because:
- StudentID yields a list of Hobbies
- StudentID yields a list of Phones
- But Hobbies and Phones have nothing to do with each other.



### Problem: ✓

- अगर हम इनको एक ही table में रखें तो redundancy बढ़ेगी और anomalies आ सकती हैं।
- If we keep them in the same table, redundancy will increase and anomalies may arise.

### ✓ Solution:

- इनको अलग-अलग tables में तोड़ देना चाहिए ताकि डेटा clean और normalized हो।
- These should be broken into separate tables so that the data is cleaned and normalized.



## Fourth Normal Form (4NF):

→ कोई relation (table) 4NF में होता है अगर:

1. वो पहले से Boyce-Codd Normal Form (BCNF) में हो ✓
2. उसमें multi-valued dependency (MVD) नहीं हो ✓

A relation (table) is in 4NF if:

- It is already in Boyce-Codd Normal Form (BCNF)
- It does not contain multi-valued dependencies (MVD)



### Step-by-Step Process for 5NF:

→ ◆ Step 1: Table should be in BCNF

- पहले check करो कि table BCNF में है या नहीं। अगर नहीं, तो पहले उसे BCNF में लाओ।
- First, check whether the table is in BCNF. If not, bring it into BCNF first.

✓ ◆ Step 2: Identify Multi-Valued Dependencies

- देखो कि एक primary key से जुड़ी दो attributes independent तरीके से multiple values तो नहीं रख रहीं।
- Check if two attributes associated with a primary key are independently holding multiple values.

✓ ◆ Step 3: Break the Table to Remove MVD

- हर multi-valued attribute को अलग table में shift कर दो।
- Shift each multi-valued attribute into a separate table.



Example: मान लो हमारे पास एक table है (Suppose we have a table):

Student	Course	Hobby
Raj	Math	Cricket
Raj	Math	Chess
Raj	Science	Cricket
Raj	Science	Chess

यहाँ हम देख सकते हैं (Here we can see):

- एक student Raj दो course कर रहा है (A student named Raj is pursuing two courses):  
**Math, Science**
- और उसकी दो hobbies हैं (And he has two hobbies): **Cricket, Chess**

लेकिन ये hobbies और courses एक-दूसरे से related नहीं हैं। मतलब:

But these hobbies and courses are not related. Meaning:

- Student → Course
- Student → Hobby (Multivalued dependency)



यह table 4NF में नहीं है (This table is not in 4NF):

- क्योंकि multivalued dependency है और Student candidate key है – फिर भी दो independent MVD हैं। इसे तोड़ना पड़ेगा।
- because there is a multivalued dependency and Student is the candidate key—yet there are two independent MVDs. This needs to be broken.

After Applying 4NF:

Table 1: Student\_Course

Student	Course
Raj	Math
Raj	Science

4NF

Table 2: Student\_Hobby

Student	Hobby
Raj	Cricket
Raj	Chess



अब (Now): ✓

- दोनों tables में कोई भी multivalued dependency नहीं है।
- Both tables have no multivalued dependencies.
- दोनों Student पर based हैं। ✓
- Both are based on Student.
- अब ये दोनों tables 4NF में हैं। ✓
- Both tables are now in 4NF.



## Join Dependency:

- जब कोई table को **तीन या उससे ज्यादा tables में तोड़कर** (decompose) फिर से join करने पर **original table lossless** तरीके से वापस मिलती है, तो हम कहते हैं कि **Join Dependency** है।
- When a table is decomposed into three or more tables and then joined again to get the original table back in a lossless manner, then we say that there is Join Dependency.

### Note:

- Join Dependency **सिर्फ तब होती है जब table को 3 या उससे ज्यादा parts में तोड़ा जाए।**
  - Join Dependency occurs only when a table is divided into 3 or more parts.
- अगर **सिर्फ 2 tables** में तोड़ रहे हैं, तो वो **functional dependency** या **BCNF** के **case** में आता है।
  - If it is divided into only 2 tables, then it falls under the case of functional dependency or BCNF.

5NF → ① 4NF ✓

② **Join dependency** **सिर्फ तब होती है जब table को 3 या उससे ज्यादा parts में तोड़ा जाए।**



Example: मान लो हमारे पास एक table है (Suppose we have a table): Project\_Assignments

Employee	Project	Role
Ram	AI	Leader
Ram	AI	Coder
Sita	Web	Designer
Sita	Web	Tester

- अब हम इसे 3 हिस्सों में तोड़ सकते हैं (Now we can break this down into 3 parts):

### 1. Employee\_Project

Employee	Project
Ram	AI
Sita	Web

### 2. Employee\_Role

Employee	Role
Ram	Leader
Ram	Coder
Sita	Designer
Sita	Tester

### 3. Project\_Role

Project	Role
AI	Leader
AI	Coder
Web	Designer
Web	Tester



- अब अगर हम इन तीनों tables को join करें, तो हमें original Project\_Assignments table वापस मिल जाएगा, बिना किसी data loss के। ✓
- Now if we join these three tables, we will get the original Project\_Assignments table back, without any data loss.

👉 इसका मतलब है (This means):

- इस table में एक join dependency है: ✓✓
- This table has a join dependency: ✓
- $(Employee, Project, Role) \rightarrow (Employee, Project), (Employee, Role), (Project, Role)$



## Fifth Normal Form (5NF) :

कोई table Fifth Normal Form में होता है, जब:

- वो पहले 4NF में होता है ✓
- और उसमें कोई **join dependency** न हो, जो कि lossless ना हो

A table is in Fifth Normal Form when:

- it is in 4NF and
- it has no join dependencies that are not lossless

मतलब: अगर कोई table को छोटे-छोटे हिस्सों में तोड़कर join करने पर कोई भी data loss नहीं होता, तो वो 5NF में है। ✓

Meaning: If there is no data loss when a table is broken into smaller parts and joined, then it is in 5NF.



### Step-by-Step Process for 5NF:

#### ◆ Step 1: Table should be in 4NF

- यानि multi-valued dependency नहीं होनी चाहिए।
- That means there should not be multi-valued dependency.

#### ◆ Step 2: Check for Join Dependencies

देखो क्या कोई ऐसा condition है जहाँ (See if there's a condition where):

- **Table** को तीन या ज्यादा **parts** में तोड़ना पड़े
- The table needs to be broken into three or more parts
- और फिर उन **parts** को **join** करके **original table lossless** तरीके से मिल जाए
- and then those **parts** can be joined to form the **original table losslessly**.



## → Step 3: Decompose Table ✓

- अगर ऐसा join dependency मिलती है, तो table को उस dependency के आधार पर तीन या उससे ज्यादा parts में तोड़ दो।
- If such a join dependency is found, then break the table into three or more parts based on that dependency.

Example: मान लो हमारे पास एक table है (Suppose we have a table): Course\_Offering

Teacher	Course	Subject
A	B.Tech	Physics
A	B.Tech	Math
A	M.Tech	Physics
B	B.Tech	Physics

इस table को तीन हिस्सों में बांटा जा सकता है (This table can be divided into three parts):



## 1. Teacher\_Course ✓

Teacher	Course
A	B.Tech
A	M.Tech
B	B.Tech

## 2. Teacher\_Subject ✓

Teacher	Subject
A	Physics
A	Math
B	Physics

## 3. Course\_Subject ✓

Course	Subject
B.Tech	Physics
B.Tech	Math
M.Tech	Physics

- जब इन तीनों टेबल को आपस में join करते हैं, तो हमें फिर से original table Course\_Offering मिल जाता है, बिना किसी data loss के
- When we join these three tables, we get the original table, Course\_Offering, back, without any data loss.

⇒ इसलिए यह table **5NF** में है। (Hence this table is in **5NF**.)



## Conclusion:

Normal Form	क्या Problem हटाता है? (What problem removes?)
→ 1NF	Repeating groups
→ 2NF	Partial dependency
→ 3NF	→ Transitive dependency
→ BCNF	→ Non-superkey dependency
→ 4NF	→ Multivalued dependency
→ 5NF	→ Join Dependency

*Thank  
you*



## Unit-5: SQL/MySQL

## Syllabus:

UNIT 5: SQL/MySQL:

1 Question → 2 Question

(8 Periods)

MySQL data types – Data Definition Commands – Data Manipulation Commands – Data Retrieval Commands, Types of operators – Arithmetic, Comparison and Logical Operators

Ques => Different type के command को समझाओ। 100% DDL, DML, DQL

Ques => MySQL में Operator का पर्याप्त है। Arithmetic & logical operator को 100%

Ques => SQL में data type किसी भी वाक्य के द्वारा है। Explain कर।



## SQL Data Types (डेटा टाइप्स): ✓

- SQL में डेटा स्टोर करने के लिए विभिन्न प्रकार के डेटा टाइप्स उपलब्ध होते हैं। ये डेटा टाइप्स यह निर्धारित करते हैं कि किसी कॉलम में कौन सा प्रकार का डेटा स्टोर किया जा सकता है। ✓
- There are different types of data types available to store data in SQL. These data types determine what type of data can be stored in a column.

1. String Data Types (Character/String डेटा टाइप्स) ✓
2. Numeric Data Types (संख्यात्मक डेटा टाइप्स) ✓
3. Date and Time Data Types (तारीख और समय के डेटा टाइप्स) ✓
4. Boolean Data Type (बूलियन डेटा टाइप) ✓
5. Binary Data Types (बाइनरी डेटा टाइप्स - इमेज, फाइल्स के लिए) ✓
6. Special Data Types (अन्य विशेष डेटा टाइप्स)



## 1. String Data Types (Character/String डेटा टाइप्स):

S.No	Data Type	Description
①	CHAR(n) CHAR(10)	Fixed-length string type, जिसमें n character होते हैं। अगर string की length कम होती है, तो बाकी की जगह spaces से भर दी जाती है। (AMAR- - - -) (AMAR-) (RAM--)
②	VARCHAR(n) VARCHAR(5)	Variable-length string type, जिसमें maximum n characters होते हैं। यहाँ केवल उतनी ही जगह ली जाती है जितनी string की असल length होती है। (AMAR) (RAM)
③	TEXT Large string	बहुत बड़ी strings को store करने के लिए, बिना किसी fixed length के। यह text के लिए इस्तेमाल होता है (जैसे: articles, descriptions)।
④	NCHAR(n)	National Character Set के लिए fixed-length string type, जिसमें n characters होते हैं।
⑤	NVARCHAR(n)	National Character Set के लिए variable-length string type, जिसमें maximum n characters होते हैं।



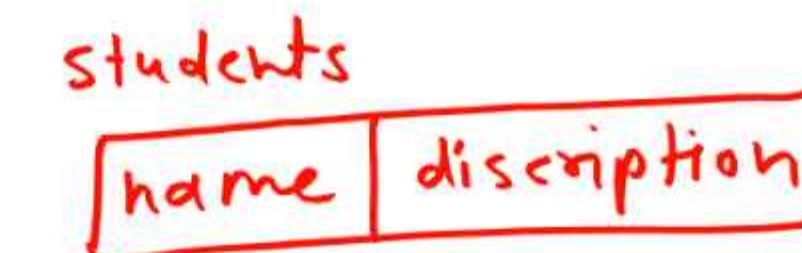
## 1. String Data Types (Character/String डेटा टाइप्स):

S.No	Data Type	Description
	<b>CHAR(n)</b>	Fixed-length string type, which holds n characters. If the string length is less, the remaining space is filled with spaces.
	<b>VARCHAR(n)</b>	A variable-length string type with a maximum of n characters. It only takes up as much space as the actual length of the string.
	<b>TEXT</b>	To store very large strings, without any fixed length. It is used for text (e.g. articles, descriptions).
	<b>NCHAR(n)</b>	A fixed-length string type for the National Character Set, consisting of n characters.
	<b>NVARCHAR(n)</b>	A variable-length string type for the National Character Set, containing a maximum of n characters.



Example: SQL Query

```
CREATE TABLE students (
    name VARCHAR(50),
    description TEXT
);
```





## 2. Numeric Data Types (संख्यात्मक डेटा टाइप्स): ✓

S.No	Data Type	Description
①	INT / INTEGER ✓	पूर्णांक (integer) प्रकार के मान को store करता है। यह -2,147,483,648 से लेकर 2,147,483,647 तक के मान ले सकता है। 1, 2, 3, 10, 50, 111, - . --
②	SMALLINT ✓	छोटे पूर्णांक (small integer) प्रकार के मान को store करता है, जो -32,768 से लेकर 32,767 तक के होते हैं।
③	BIGINT ✓	बड़े पूर्णांक (large integer) प्रकार के मान को store करता है। इसका रेंज बहुत बड़ा होता है: -9,223,372,036,854,775,808 से लेकर 9,223,372,036,854,775,807 तक।
④	DECIMAL(p, s)/ NUMERIC(p, s)	एक fixed-point number है जहाँ p precision (कुल अंक) और s scale (दशमलव के बाद के अंक) को दर्शाता है। (जैसे DECIMAL(5, 2) = 999.99)
⑤	FLOAT ✓	फ्लोटिंग-point number, जो 15 डिजिट तक की precision support करता है। यह बहुत बड़े या छोटे decimal मानों को represent कर सकता है।

DECIMAL(p,s)      999.999  
 DECIMAL(4,2)      999.9



## 2. Numeric Data Types (संख्यात्मक डेटा टाइप्स):

S.No	Data Type	Description
	<b>INT / INTEGER</b>	Stores a value of type integer. It can take values from -2,147,483,648 to 2,147,483,647.
	<b>SMALLINT</b>	Stores a value of the small integer type, ranging from -32,768 to 32,767.
	<b>BIGINT</b>	Stores a value of the large integer type. Its range is very large: from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
	<b>DECIMAL(p, s)/ NUMERIC(p, s)</b>	is a fixed-point number where p represents the precision (total digits) and s represents the scale (digits after the decimal point). (e.g. DECIMAL(5, 2) = 999.99)
	<b>FLOAT</b>	A floating-point number, which supports precision up to 15 digits. It can represent very large or small decimal values.



Example:

```
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    price DECIMAL(10,2)
);
```

Products

product_id(PK)	price
1	10.00
2	20.00
3	30.00
4	40.00
5	50.00
6	60.00
7	70.00
8	80.00
9	90.00
10	100.00

9999999999



### 3. Date and Time Data Types (तारीख और समय के डेटा टाइप्स): ✓

S.No	Data Type	Description
1	DATE ✓ <span style="border: 1px solid red; padding: 2px;">07/11/2025</span>	केवल तारीख (Date) को store करता है – format: 'YYYY-MM-DD' (जैसे: 2024-12-25)
2	TIME ✓ <span style="border: 1px solid red; padding: 2px;">01:31:10</span>	केवल समय (Time) को store करता है – format: 'HH:MI:SS' (जैसे: 14:30:00)
3	<u>DATETIME</u>	तारीख + समय दोनों को store करता है – format: 'YYYY-MM-DD HH:MI:SS'
4	<u>TIMESTAMP</u> ✓	DATETIME जैसा होता है, पर यह <u>ऑटो-अपडेट</u> या <u>tracking</u> के लिए ज्यादा उपयोगी होता है
5	YEAR ✓ <span style="border: 1px solid red; padding: 2px;">2025</span> ✓	केवल साल को store करता है – format: 'YYYY' (जैसे: 2025)
6	TIME WITH <u>TIME ZONE</u>	समय के साथ टाइम ज़ोन भी store करता है <span style="color: pink;">01:32:15 +5:30 kolkata</span>
7	<u>TIMESTAMP WITH TIME ZONE</u> ✓	टाइमस्टैम्प के साथ टाइम ज़ोन की जानकारी भी रखता है <span style="color: pink;">07/11/2025 01:33:20 +5:30 Mumbai</span>



### 3. Date and Time Data Types (तारीख और समय के डेटा टाइप्स):

S.No	⌚ Data Type	Description
	DATE	Stores only dates – format: 'YYYY-MM-DD' (e.g., 2024-12-25)
	TIME	Stores only time – format: 'HH:MI:SS' (e.g. 14:30:00)
	DATETIME	Stores both date and time – format: 'YYYY-MM-DD HH:MI:SS'
	TIMESTAMP	Similar to DATETIME, but more useful for auto-updates or tracking
	YEAR	Stores only the year – format: 'YYYY' (e.g.: 2025)
	TIME WITH TIME ZONE	Stores the time zone along with the time.
	TIMESTAMP WITH TIME ZONE	Keeps time zones and timestamps



Example:

```
CREATE TABLE events(  
    event_id INT PRIMARY KEY,  
    event_date DATE,  
    event_time TIME,  
    created_at TIMESTAMP DEFAULT  
CURRENT_TIMESTAMP  
);
```

events

event_id (PK)	event_date	event_time	created_at
---------------	------------	------------	------------



## 4. Boolean Data Type (बूलियन डेटा टाइप): ✓

S.No	Data Type	Description
①	BOOLEAN	Boolean मान, जो TRUE या FALSE हो सकते हैं। (कुछ डेटाबेस में BOOLEAN को TINYINT या BIT के रूप में store किया जाता है) ✓ Boolean values, which can be either TRUE or FALSE. (Some databases store BOOLEAN as TINYINT or BIT)

True / false

### Example:

```
CREATE TABLE users (
    user_id INT PRIMARY KEY,
    is_active BOOLEAN DEFAULT TRUE
);
```

users	
user_id (PK)	is_active
	True



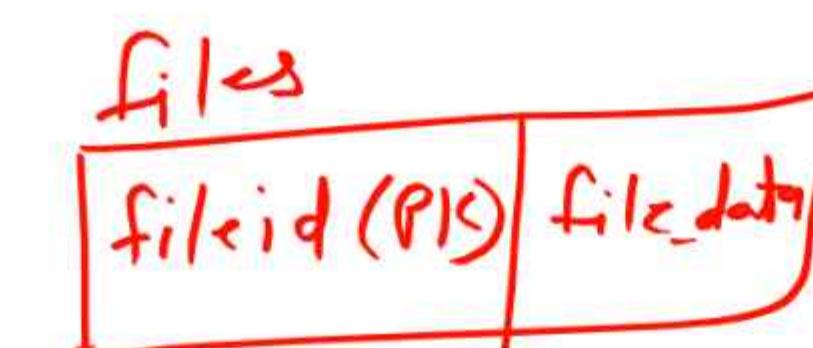
## 5. Binary Data Types (बाइनरी डेटा टाइप्स - इमेज, फाइल्स के लिए): ✓

S.No	Data Type	Description
①	BINARY(n) BINARY(5)	Fixed-length binary data को store करता है, जहाँ n bytes तक की length होती है। Stores fixed-length binary data, where the length is up to n bytes.
②	VARBINARY(n) BINARY(5)	Variable-length binary data को store करता है, जहाँ maximum length n bytes होती है। Stores variable-length binary data, where the maximum length is n bytes.
③	BLOB Binary Large object	Binary Large Object – बड़ी binary data (जैसे images, audio, video) को store करता है (up to several GBs)। Binary Large Object – Stores large binary data (e.g. images, audio, video) (up to several GBs).



Example:

```
CREATE TABLE files (
    file_id INT PRIMARY KEY,
    file_data BLOB
);
```





## 6. Special Data Types (अन्य विशेष डेटा टाइप्स):

S.No	Data Type	Description
1	XML	XML डेटा को स्टोर करने और क्वेरी करने के लिए प्रयोग किया जाता है। XML is used to store and query data.
2	ENUM (MySQL) <i>chymable</i>	Predefined set of values में से एक ही चुनने की अनुमति देता है। Allows you to choose only one of the predefined set of values.
3	SET (MySQL)	एक या एक से अधिक values को predefined list से चुनने की अनुमति देता है। Allows one or more values to be selected from a predefined list.

Enum gender (male, female, others)

Set Hobby (Reading, Travelling, playing chess)



## Example:

```
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    order_status ENUM('pending', 'shipped', 'delivered', 'cancelled')
);
```

orders

orderid (PK)	order status
1	shipped
2	Pending
3	Cancelled

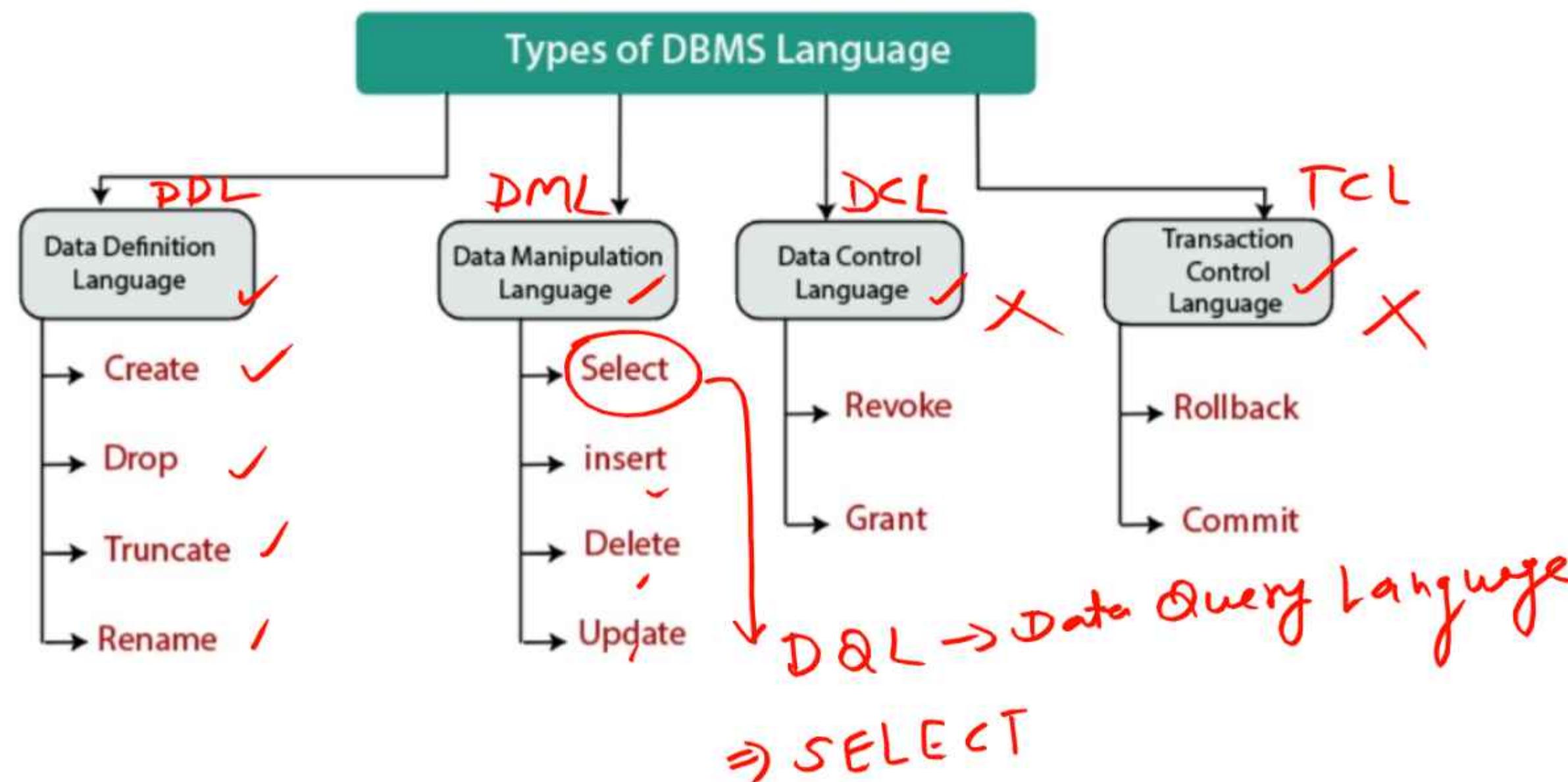


### Database Command :

- DBMS में ऐसे Commands और Interfaces का उपयोग होता है जो **डेटाबेस से क्वेरी चलाने** और **डेटा अपडेट करने** के लिए बनाए गए हैं। ✓
- DBMS uses languages and interfaces designed to run queries and update data from the database.
- **डेटाबेस Command** का use **डेटा को read, स्टोर करने और अपडेट करने** के लिए किया जाता है।
- Database language is used to read, store, and update data.



## Types of Database Command :





## 1. Data Definition Commands (DDL Commands)

- DDL का पूरा नाम Data Definition Language है। ✓
  - DDL stands for Data Definition Language.
  - यह database की **structure** या **schema** को define, modify या delete करने के लिए उपयोग होती है।
  - It is used to define, modify or delete the structure or schema of the database.
- ◆ Key Point: ✓
- DDL commands permanent changes करती हैं (**Auto Commit होती हैं**)।
  - DDL commands make permanent changes (auto commit).

Database Create—  
↳ Drop—  
→ rename✓



## Common DDL Commands:

Command	उपयोग (Purpose)	उदाहरण (Example)
CREATE ✓	नई table या database बनाना	<u>CREATE TABLE</u> <u>student</u> ( <u>id INT</u> , <u>name VARCHAR(50)</u> );
ALTER ✓	किसी existing table की <u>structure</u> बदलना	<u>ALTER TABLE</u> <u>student</u> <u>ADD age INT</u> ;
DROP ✓	किसी table या database को पूरी तरह हटाना	<u>DROP TABLE</u> <u>student</u> ;
TRUNCATE ✓	Table का सारा <u>data</u> हटाना ( <u>structure</u> बना रहता है)	<u>TRUNCATE TABLE</u> <u>student</u> ;
RENAME ✓	Table का नाम <u>बदलना</u>	<u>RENAME TABLE</u> <u>student</u> <u>TO learners</u> ; old name + name new name + name



## Common DDL Commands:

Command	उपयोग (Purpose)	उदाहरण (Example)
CREATE	Create a new table or database	<code>CREATE TABLE student (id INT, name VARCHAR(50));</code>
ALTER	Changing the structure of an existing table	<code>ALTER TABLE student ADD age INT;</code>
DROP	Completely deleting a table or database	<code>DROP TABLE student;</code>
TRUNCATE	Deleting all data from the table (retaining the structure)	<code>TRUNCATE TABLE student;</code>
RENAME	Renaming Table	<code>RENAME TABLE student TO learners;</code>



## 2. Data Manipulation Commands (DML Commands)

- DML का पूरा नाम Data Manipulation Language है।
  - DML stands for Data Manipulation Language.
  - यह database की tables के अंदर मौजूद data को बदलने (insert, update, delete) के लिए उपयोग होती है।  
*Table → अंदर data*
  - It is used to change (insert, update, delete) data within database tables.
- ◆ Key Point:
- DML commands data को modify करती हैं लेकिन structure को नहीं।
  - DML commands modify data but not its structure.
  - Rollback और Commit जैसे commands से इनका control किया जा सकता है।
  - They can be controlled by commands like Rollback and Commit.



## Common DML Commands:

student

<u>id</u>	<u>name</u>	<u>Age</u>
1	Bulbul	22

Command	उपयोग (Purpose)	उदाहरण (Example)
INSERT ✓	नए रिकॉर्ड (row) को जोड़ना	<u>INSERT INTO</u> <u>student</u> <u>VALUES (1, 'Bulbul', 22);</u>
UPDATE ✓	किसी existing रिकॉर्ड को <u>बदलना</u>	<u>UPDATE</u> <u>student</u> <u>SET age = 23 WHERE id = 1;</u>
DELETE ✓	<u>किसी रिकॉर्ड को हटाना</u>	<u>DELETE FROM</u> <u>student WHERE id = 1;</u>

Command	उपयोग (Purpose)	उदाहरण (Example)
INSERT	Adding a new record (row)	<u>INSERT INTO</u> <u>student</u> <u>VALUES (1, 'Bulbul', 22);</u>
UPDATE	Replacing an existing record	<u>UPDATE</u> <u>student</u> <u>SET age = 23 WHERE id = 1;</u>
DELETE	delete a record	<u>DELETE FROM</u> <u>student WHERE id = 1;</u>



### 3. Data Retrieval Commands (DQL Commands)

- DQL का पूरा नाम Data Query Language है।  
• DQL stands for Data Query Language.
  - यह database से data निकालने या देखने (Retrieve/Query) के लिए उपयोग होती है।  
• It is used to retrieve or query data from a database.
- ◆ **Key Point:**
- DQL में केवल SELECT command होता है। / DQL only has the **SELECT command**.
  - इसका उपयोग रिपोर्ट या जानकारी निकालने के लिए किया जाता है। / It is used to generate **reports** or extract information.
  - यह data को केवल read करता है, बदलता नहीं है। / It only reads data, not changes it.



## Common DQL Command:

Command	उपयोग (Purpose)	उदाहरण (Example)
SELECT ✓	Data को <u>fetch (retrieve)</u> करने के लिए (To retrieve data) ✓	<pre>SELECT * FROM student;</pre> <pre>SELECT name, age FROM student WHERE age &gt; 20;</pre>

- ① `SELECT * FROM students;`
- ② `SELECT name, age FROM students;`
- ③ `SELECT id, name, Age FROM students  
WHERE Age = 18;`

\* = all

<u>id</u>	<u>Name</u>	<u>Age</u>	<u>Branch</u>	<u>city</u>
1	Amit	18	CSE	Mumbai
2	Shivam	20	ECE	Pune
3	Rahul	22	EEE	Nagpur
4	Pratik	20	IT	Bangalore
5	Yash	18	CSE	Mumbai
6	Abhishek	19	EEE	Nagpur
7	Vishal	21	IT	Bangalore



## Operators (ऑपरेटर)

- Operator (ऑपरेटर) वे symbols या चिन्ह होते हैं जिनका उपयोग किसी expression या statement में गणना (calculation), तुलना (comparison) या शर्तों (conditions) को जाँचने के लिए किया जाता है।
- Operators are symbols that are used to perform calculations, comparisons, or check conditions in an expression or statement.

Symbol → Calculation ✓  
↓  
operator

Comparison ✓  
Condition ✓

$a > b$   
↑  
operator

$A + B$   
↑  
operator



## Types of Operators (ऑपरेटर के प्रकार) ✓

मुख्य रूप से MySQL में 3 प्रकार के Operators होते हैं:

There are mainly 3 types of operators in MySQL:

- Arithmetic Operators (गणितीय ऑपरेटर) ✓
- Comparison (Relational) Operators (तुलनात्मक ऑपरेटर) ✓
- Logical Operators (तार्किक ऑपरेटर) ✓



## 1. Arithmetic Operators (गणितीय ऑपरेटर)

- ये operators संख्यात्मक गणना (Mathematical calculations) के लिए उपयोग किए जाते हैं – जैसे जोड़, घटाव, गुणा, भाग आदि।
- These operators are used for numerical calculations – such as addition, subtraction, multiplication, division, etc.

Operator	अर्थ (Meaning)	उदाहरण (Example)	परिणाम (Result)
+	जोड़ना (Addition)	10 + 5	15
-	घटाना (Subtraction)	10 - 5	5
*	गुणा (Multiplication)	10 * 5	50
/	भाग देना (Division)	10 / 2	5
%	शेषफल (Modulus / Remainder)	10 % 3 3)10(3 1 2 1 0	1 ✓

1  
2  
1  
0  
Remainder



Example:

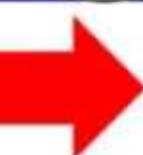
- `SELECT 15 + 5 AS Addition, 15 - 5 AS Subtraction;`



## 2. Comparison (Relational) Operators (तुलनात्मक ऑपरेटर) ✓

- इनका उपयोग दो मानों की तुलना (comparison) करने के लिए किया जाता है।
- These are used to compare two values.
- ये **TRUE (सत्य)** या **FALSE (असत्य)** परिणाम लौटाते हैं।
- They return either TRUE or FALSE.

Operator	अर्थ (Meaning)	उदाहरण (Example)	परिणाम (Result)
=	बराबर (Equal to)	5 = 5	✓ TRUE
!= या <>	बराबर नहीं (Not equal to)	5 != 3 ✓	✓ TRUE
>	बड़ा (Greater than)	7 > 5    6 < 2	✓ TRUE    false
<	छोटा (Less than)	3 < 8	✓ TRUE ✓



Operator	अर्थ (Meaning)	उदाहरण (Example)	परिणाम (Result)
$\geq$	बड़ा या बराबर (Greater than <u>or equal to</u> )	$5 \geq 5$	TRUE
$\leq$	छोटा या बराबर (Less than or equal to)	$4 \leq 6$	TRUE
BETWEEN ... AND	किसी रेंज के बीच (Within range)	10 BETWEEN 5 AND 15	TRUE
LIKE	पैटर्न मिलाना (Pattern match)	'Ram' LIKE 'R%'	TRUE
IN	सूची में <u>मौजूद</u> (In a list)	5 IN (3,5,7)	TRUE



Example:

```
SELECT * FROM students WHERE age >= 18;
```



### 3. Logical Operators (तार्किक ऑपरेटर)

- इनका उपयोग **एक से अधिक शर्तों (conditions)** को जोड़ने या जांचने के लिए किया जाता है।
- These are used to combine or check more than one condition.

Operator	अर्थ (Meaning)	उदाहरण (Example)	परिणाम (Result)
AND	दोनों शर्तें सत्य हों (both conditions are true)	(age > 18 AND city = 'Delhi')	TRUE जब दोनों सही हों/ when both are true
OR	कोई एक शर्त सत्य हो (one of the conditions is true)	(age > 18 OR city = 'Delhi')	TRUE जब कोई एक सही हो / When one is right
NOT	शर्त को उल्टा करता है (reverses the condition)	NOT (age > 18)	TRUE जब age $\leq$ 18 हो / When age $\leq$ 18
XOR	केवल एक शर्त सही हो (Only one condition is true)	(TRUE XOR FALSE)	✓ TRUE



Example:

```
SELECT * FROM student  
WHERE age > 18 AND gender = 'Male';
```

# Thank you

DBMS  
↓  
1 to 5 unit  
↓  
Complete  
==