LET'S LEARN

DATABASE MANAGEMENT SYSTEM (DBMS)

-By Riti Kumari

TOPICS TO BE COVERED

1. DBMS Introduction
2. DBMS Architecture
3. Data Abstraction
4. Types of data model
5. ER Model
6. Relational Model
7. Type of keys
8. Normalisation
9. Denormalization
10. Transactions & Concurrency control
11. Indexing(B ,B+ trees)
12. SQL

DATA & INFORMATION

Data : Data refers to a collection of raw facts or figures that can be processed to derive meaning or knowledge. In easier terms we can say any fact that can be stored. *Ex- XYZ, 12*

Information : Processed data is called information.
Ex- Your name, temparature, etc.

DATABASE

Collection of interrelated data is called a database.

- It can be stored in the form of table
- It can be any size

Multimedia database

College database

EXAMPLE

ID	Name	subject
1	Rahul	PHE
2	Raj	ECO
3	Riti	IT

ID	Name	Place
1	Rahul	DELHI
2	Raj	KOLKATA
3	Riti	MUMBAI

Collection of related data

FILE SYSTEM

An operating system's approach for organising and storing data on storage units like hard drives is called a file system.
In a file system, data is organised into files.

The major disadvantage of file system is

- Data redundancy
- Poor Memory utilisation
- Data inconsistency
- Data security

DATABASE MANAGEMENT SYSTEM

The acronym DBMS stands for **"Database Management System."**

Users can access databases, save data, retrieve it, update it, and manage it safely and effectively with the use of a software program or combination of programs.

The presence of rules and regulations in the management system is crucial as they are necessary to uphold and maintain the database effectively.

APPLICATION OF DBMS

- Schools and Colleges

- Banks

- Airlines

APPLICATION OF DBMS

- **Schools and Colleges** – DBMS is used to create and maintain a student information system that stores student records, including personal details, academic performance, attendance, and extracurricular activities.

- **Banks** – DBMS is used to maintain a centralised and secure database of customer information, including personal details, account numbers, contact information, and transaction history.

TYPES OF DATABASES

1. Relational Databases (RDBMS)
2. NoSQL Databases
3. Object-Oriented Databases
4. In-Memory Databases
5. Time-Series Databases
6. Spatial Databases
7. Multimedia Databases
8. Columnar Databases
9. XML Databases
10. NewSQL Databases
11. Blockchain Databases

SQL

TYPES OF DATABASES

- **Relational Databases (RDBMS)**- These databases structure data into organized tables that have predefined connections between them. Data manipulation and querying are performed using SQL (Structured Query Language). Well-known instances encompass MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.

- **NoSQL Databases**- NoSQL databases are created to handle data that doesn't fit neatly into the strict setup of traditional relational databases. Ex- MongoDB(Document Oriented DB)

TYPES OF DATABASES

- **Object-Oriented Databases**- These databases hold objects (data and actions) utilized in object-oriented programming. They work well for applications with intricate data designs, like scientific simulations or multimedia software.

- **In-Memory Databases**- In these databases, data is kept in the primary memory (RAM) rather than on a disk, leading to quicker data retrieval. They're employed in applications that demand instant data processing and top-notch performance.

NEED OF DBMS

DBMS plays a vital role for businesses, institutions, and organizations of all scales in effectively managing their data, ensuring data accuracy and security, and supporting essential decision-making processes.

It serves as the core of contemporary information systems, facilitating efficient data management and serving as a basis for a wide range of applications and services.

ADVANTAGE OF DBMS

- **Data Security**- DBMS implements security mechanisms that regulate access to sensitive information, safeguarding it from unauthorized access and potential data breaches.

- **Data Redundancy and Inconsistency** - DBMS removes data redundancy, minimizing storage needs and ensuring consistency through the maintenance of a unified version of the data.

- **Data Integrity** - DBMS guarantees data integrity by enforcing rules and constraints that prohibit the entry of incorrect or inconsistent data into the database.

ADVANTAGE OF DBMS

- **Data Scalability**- DBMS can handle large datasets and scale to accommodate increasing amounts of data as an organization grows.

- **Data Abstraction**- DBMS offers data abstraction, allowing users and applications to interact with the database without needing to understand its underlying complexities.

DISADVANTAGE OF DBMS

- **Cost**- Acquiring, deploying, and sustaining DBMS software can incur significant costs. Furthermore, the hardware essential for the proficient operation of a DBMS can also lead to substantial expenses.

- **Scale Projects**- When dealing with modest applications and minimal data storage requirements, adopting a comprehensive DBMS could introduce avoidable intricacies and additional burdens. In these instances, more streamlined data storage alternatives could be better suited.

DISADVANTAGE OF DBMS

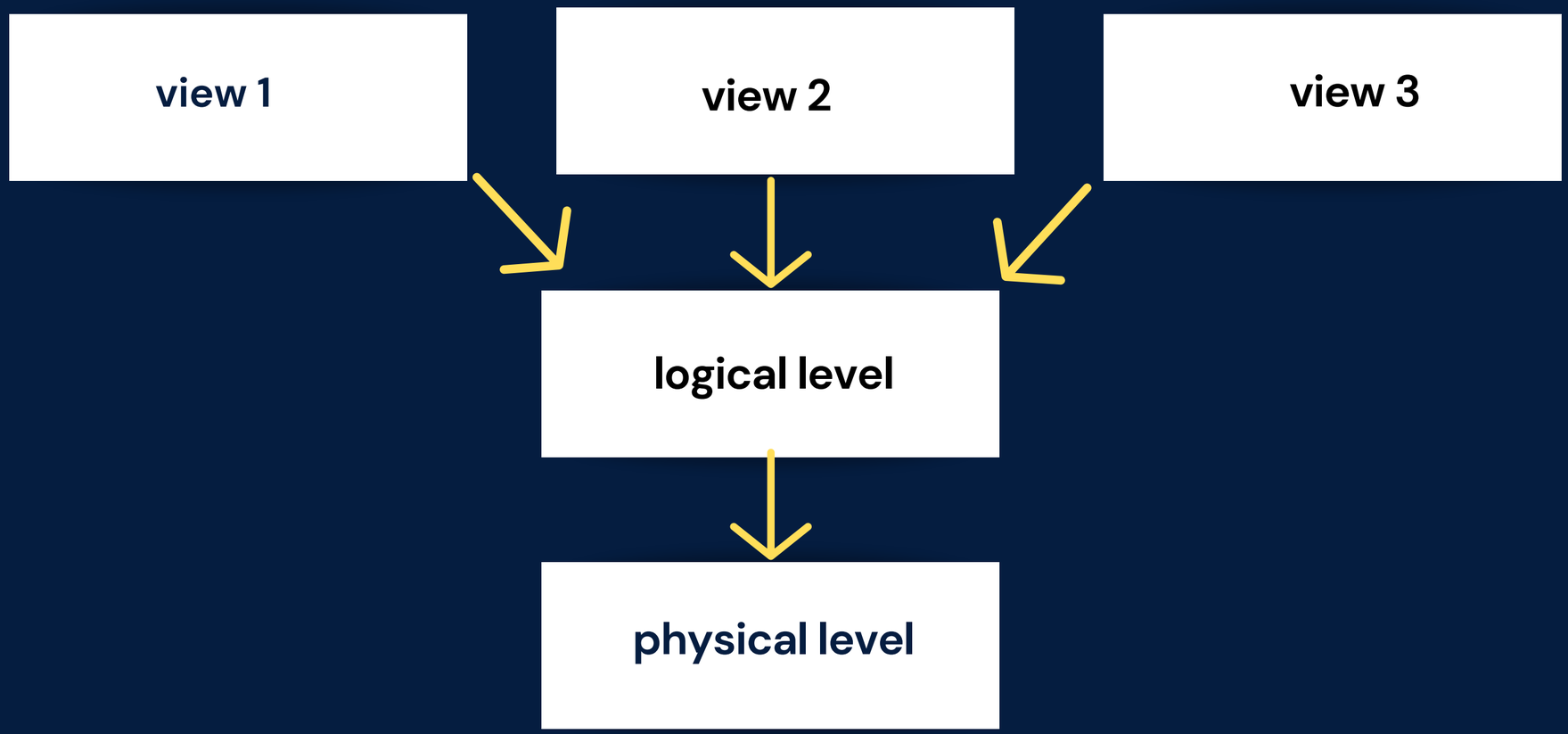
- Vendor Lock-In- Once you've chosen a specific DBMS, it can be challenging to switch to a different one due to differences in data formats, query languages, and other technical aspects. This can lead to vendor lock-in, where you are dependent on a particular vendor's technology and pricing.

DATA ABSTRACTION

Database systems are built with complex ways of organizing data. To make it easier for people to use the database, the creators hide the complicated stuff that users don't need to worry about. This hiding of unnecessary things from users is called data abstraction.

DATA ABSTRACTION

There are three levels of Abstraction



TYPES OF LEVEL

- **Physical level-** This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.

- **Logical level-** This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

- **View level-** Highest level of data abstraction. This level describes the user interaction with database system.

SCHEMA & INSTANCE

Let us first learn about some basic concepts:

Schema- A schema is a logical container or structure that organizes and defines the structure of a database.

It defines how data is organized, what data types are used, what constraints are applied, and the relationships between different pieces of data. A schema acts as a blueprint for the database, ensuring data integrity, consistency, and efficient data retrieval.

SCHEMA & INSTANCE

Types of Schema :

1. **Physical Schema–** A physical schema defines how data is stored on the underlying hardware, including details such as storage format, file organization, indexing methods, and data placement.

SCHEMA & INSTANCE

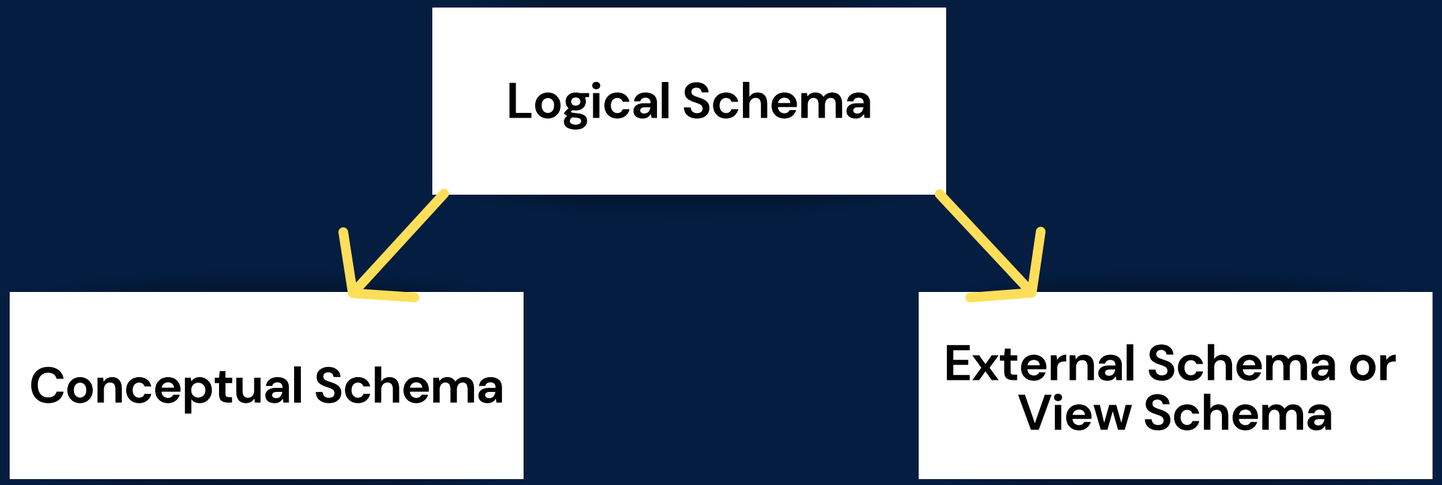
Characteristics of **Physical Schema** :

- Its primary focus lies in enhancing the storage and retrieval of data to boost performance.
- Modifications made to the physical schema demand meticulous planning and can potentially affect the overall performance of the database.
- Example: Deciding to use clustered indexes on specific columns for faster retrieval.

SCHEMA & INSTANCE

Types of Schema :

2. **Logical Schema-** A logical schema defines the database's structure from a logical or conceptual perspective, without considering how the data is physically stored.



SCHEMA & INSTANCE

Types of Logical Schema :

- **Conceptual Schema:** The conceptual schema represents the overall view of the entire database. It defines the high-level structure and relationships between all data elements.

- **External/View Schema:** An external schema defines the user-specific views of the database. It focuses on the portions of the database that are relevant to specific user roles or applications.

SCHEMA & INSTANCE

Characteristics of **Logical Schema** :

- It delineates how data is structured into tables, the interconnections between these tables, and the restrictions placed on the data.
- Logical schemas prioritize data modeling and database design over considerations related to hardware or storage specifics.
- Example: Defining tables, specifying primary and foreign keys, and creating views for data access.

SCHEMA & INSTANCE

Instance – The information residing within a database at a specific point in time is referred to as the database's "instance."

Within a given database schema, the declarations of variables within its tables pertain to that specific database. The term "instance" in this context denotes the current values of these variables at a particular moment in time for that database.

DBMS ARCHITECTURE

Database Management System architecture, refers to the structural framework and organization of a database management system. It defines how the various components of the system work together to store, manage, and retrieve data efficiently.

DBMS ARCHITECTURE

Types of DBMS ARCHITECTURE :

There are several types of DBMS Architecture.
Choice of architecture depends on factors such as the type of database (e.g., relational, NoSQL) and the specific needs of an application.

- **1-Tier Architecture**
- **2-Tier Architecture**
- **3-Tier Architecture**

DBMS ARCHITECTURE

1-Tier Architecture - In 1 tier architecture the entire database application, including the user interface, application logic, and data storage, resides on a single machine or computer.

ex- An illustration of a straightforward single-tier architecture can be seen when you install a database on your system and use it to practice SQL queries.

DBMS ARCHITECTURE

2-Tier Architecture - In 2 Tier Architecture the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server.

Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.

DBMS ARCHITECTURE

2-Tier Architecture

Client (User/Application)

Server(db)

DBMS ARCHITECTURE

- **3-Tier Architecture** - It separates the application into three logically distinct layers presentation, application, and data layer

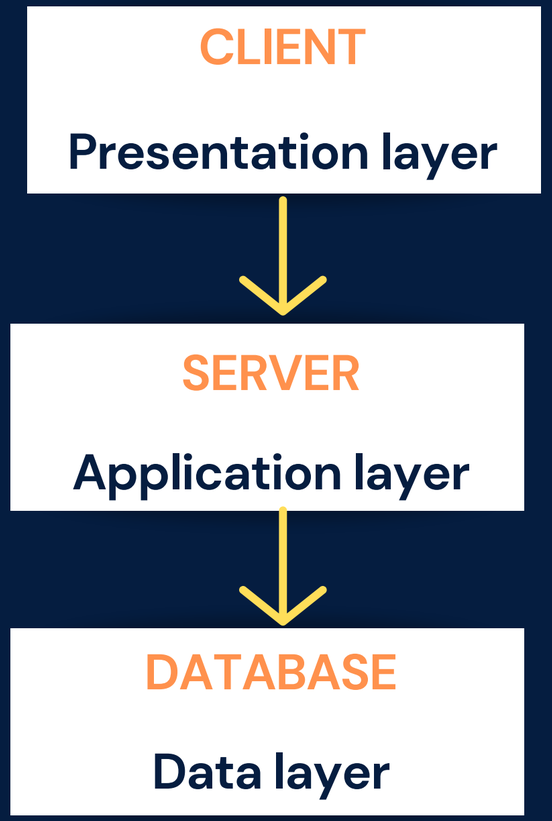
- **Presentation layer**- It handles the user interface.
  ex- your PC, Tablet, Mobile, etc

- **Application layer** - It manages business logic
  ex- server

- **Data layer**- It manages data storage and processing.
  ex- Database Server

DBMS ARCHITECTURE

- 3-Tier Architecture



DBMS ARCHITECTURE

Advantages of 3-tier-architecture

- Scalability: Easily adjust each tier to handle changing user demands.
- Modularity and Maintainability: Simplify maintenance by separating responsibilities.
- Security: Protect sensitive data with an additional layer.
- Performance: Optimize presentation and application tiers for better performance.

DBMS ARCHITECTURE

Disadvantages of 3-tier-architecture

- The disadvantages of 3-Tier Architecture include increased complexity, potential latency issues, longer development time, resource overhead, and the possibility of bottlenecks.

DATA MODEL

A data model within a Database Management System (DBMS) serves as an abstract representation of how data gets structured and organized within a database.

It outlines the logical arrangement of data and the connections between various data components.

Data models play a crucial role in comprehending and shaping databases, acting as a vital link between real-world entities and the actual storage of data within the database.

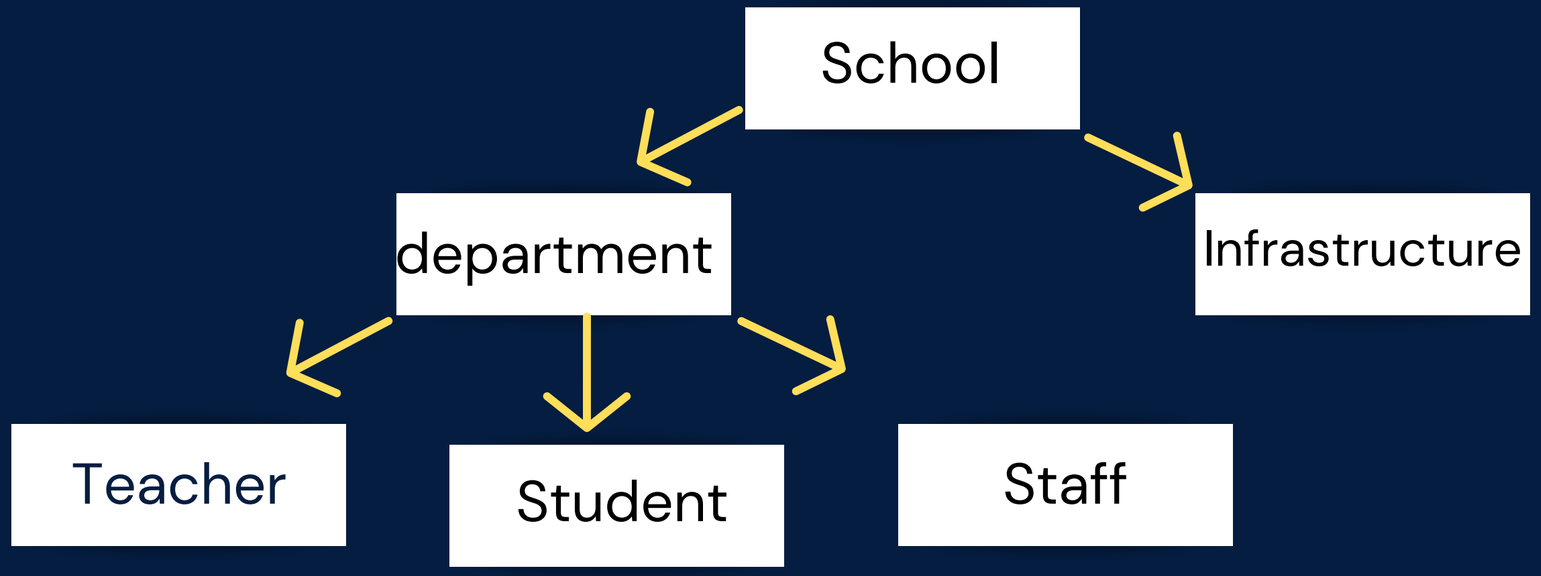
DATA MODEL

Types of Data Model

- Hierarchical Data Model
- Network Data Model
- Relational Data Model
- Entity-Relationship Model (ER Model)
- Object-Oriented Data Model
- NoSQL Data Models

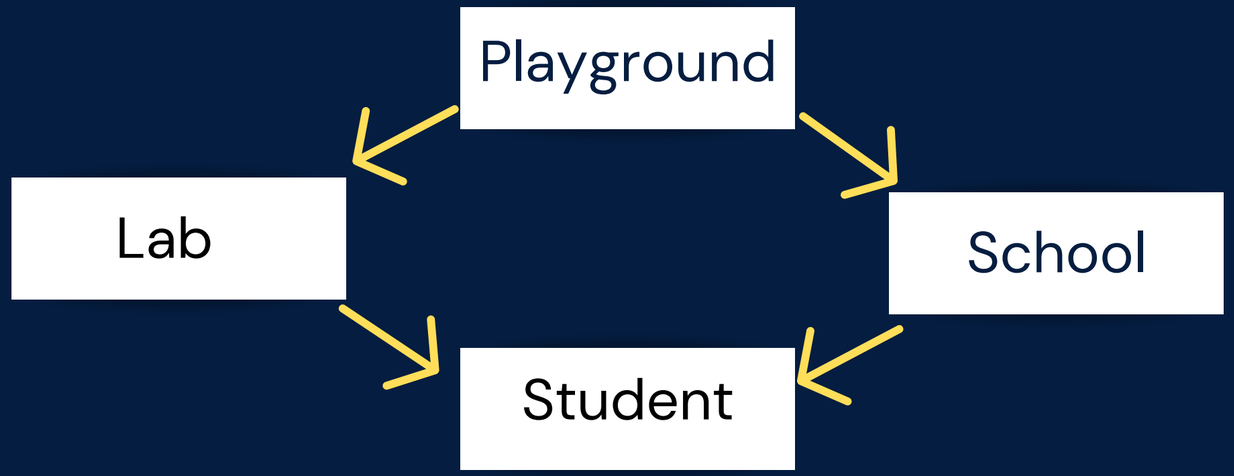
TYPES OF DATA MODEL

- **Hierarchical Data Model:** This model portrays data in a manner resembling a tree structure, where each record maintains a parent-child relationship. Its primary application lies in older database systems.



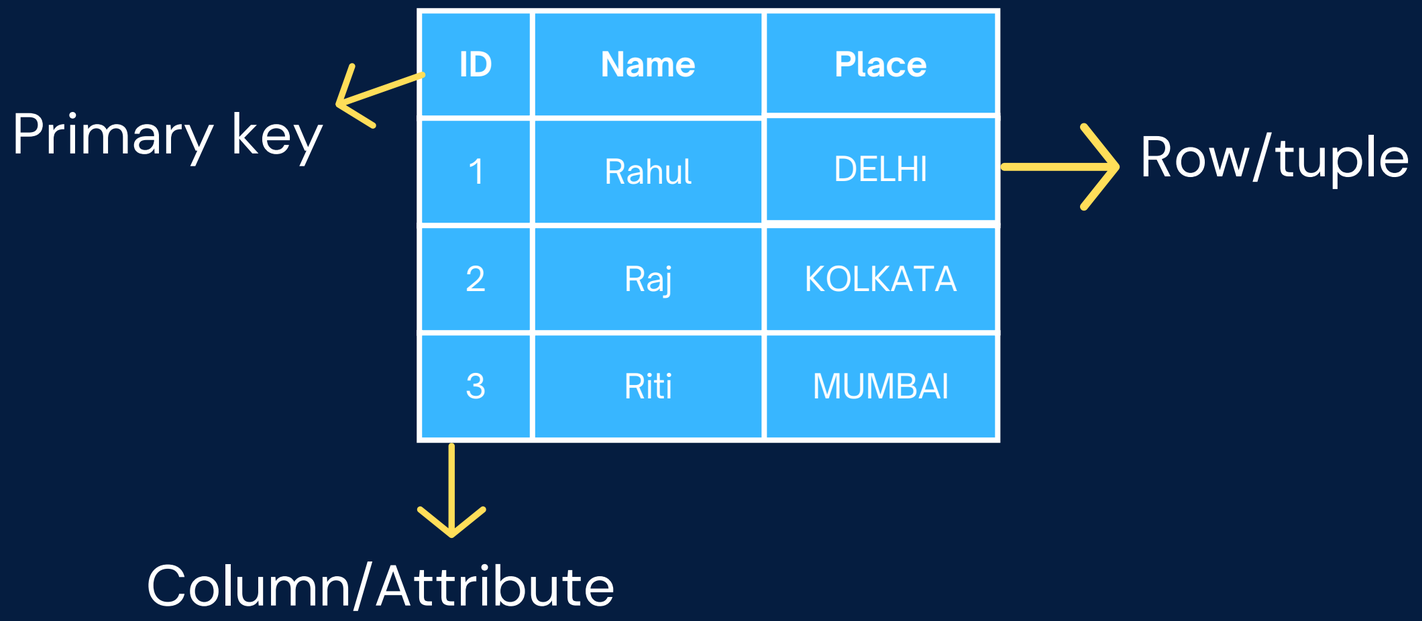
TYPES OF DATA MODEL

- **Network Data Model**: This model shares similarities with the hierarchical approach, permitting records to hold multiple parent-child relationships. It adopts a structure akin to a graph, offering more flexibility compared to the hierarchical model.



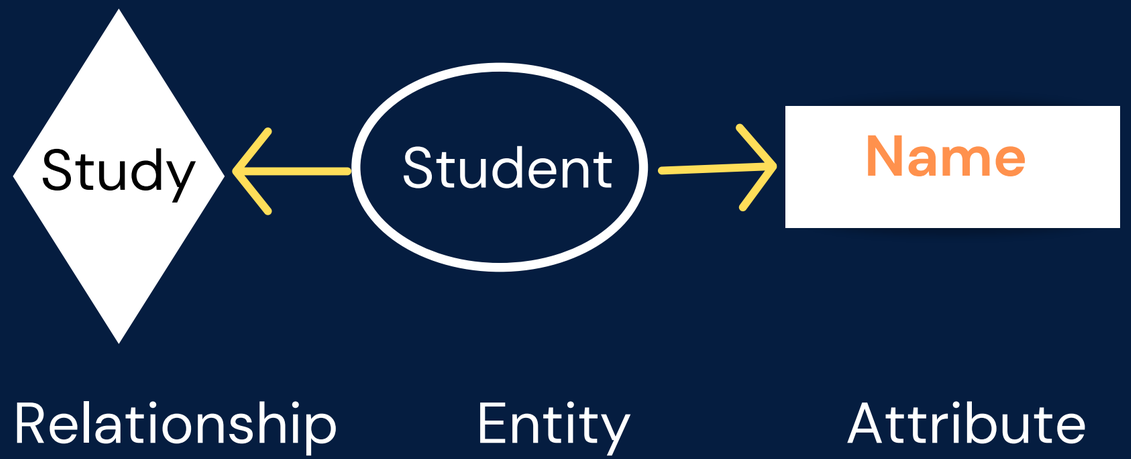
TYPES OF DATA MODEL

- **Relational Data Model:** Organizing data into tables (known as relations) consisting of rows and columns characterizes the relational model. It stands as the most prevalent data model, rooted in the principles of set theory, and relies on Structured Query Language (SQL) for data manipulation.



TYPES OF DATA MODEL

- **Entity-Relationship Model (ER Model):** Utilized for crafting relational databases, the ER model represents data through entities (objects), attributes (entity properties), and relationships connecting these entities.



TYPES OF DATA MODEL

- **Object-Oriented Data Model:** Extending the principles of object-oriented programming into the database domain, this model depicts data as objects complete with attributes and methods, fostering support for inheritance and encapsulation.

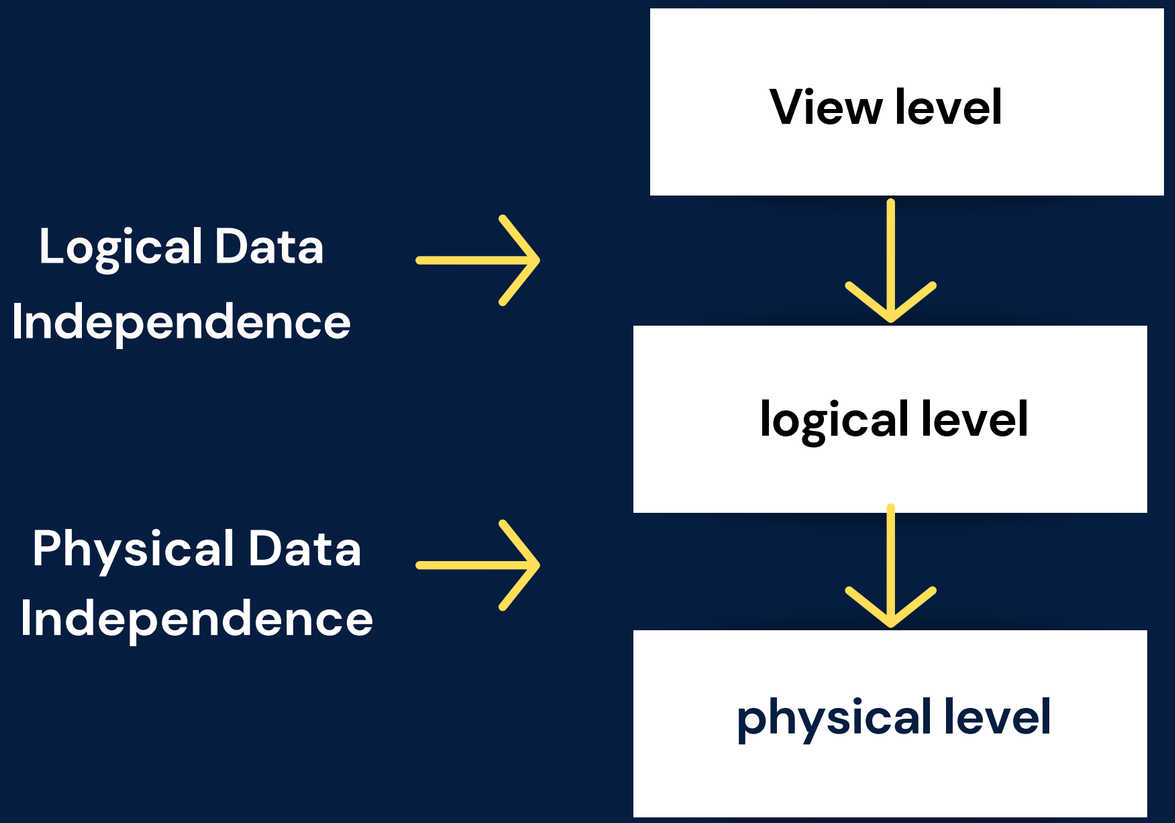
- **NoSQL Data Models:** NoSQL databases encompass a diverse array of data models, such as document-oriented (e.g., MongoDB), key-value (e.g., Redis), column-family (e.g., Cassandra), and graph (e.g., Neo4j). These models are designed to offer scalability and flexibility when handling extensive volumes of unstructured or semi-structured data.

DATA INDEPENDENCE

Data independence is a fundamental concept within database design and management, emphasizing the distinction between the logical and physical dimensions of data storage and administration in a database management system (DBMS). This principle yields various benefits, such as enhanced flexibility, heightened security, and simplified maintenance.

DATA INDEPENDENCE

There are three levels of Abstraction



ESSENTIAL COMPONENTS OF TABLES

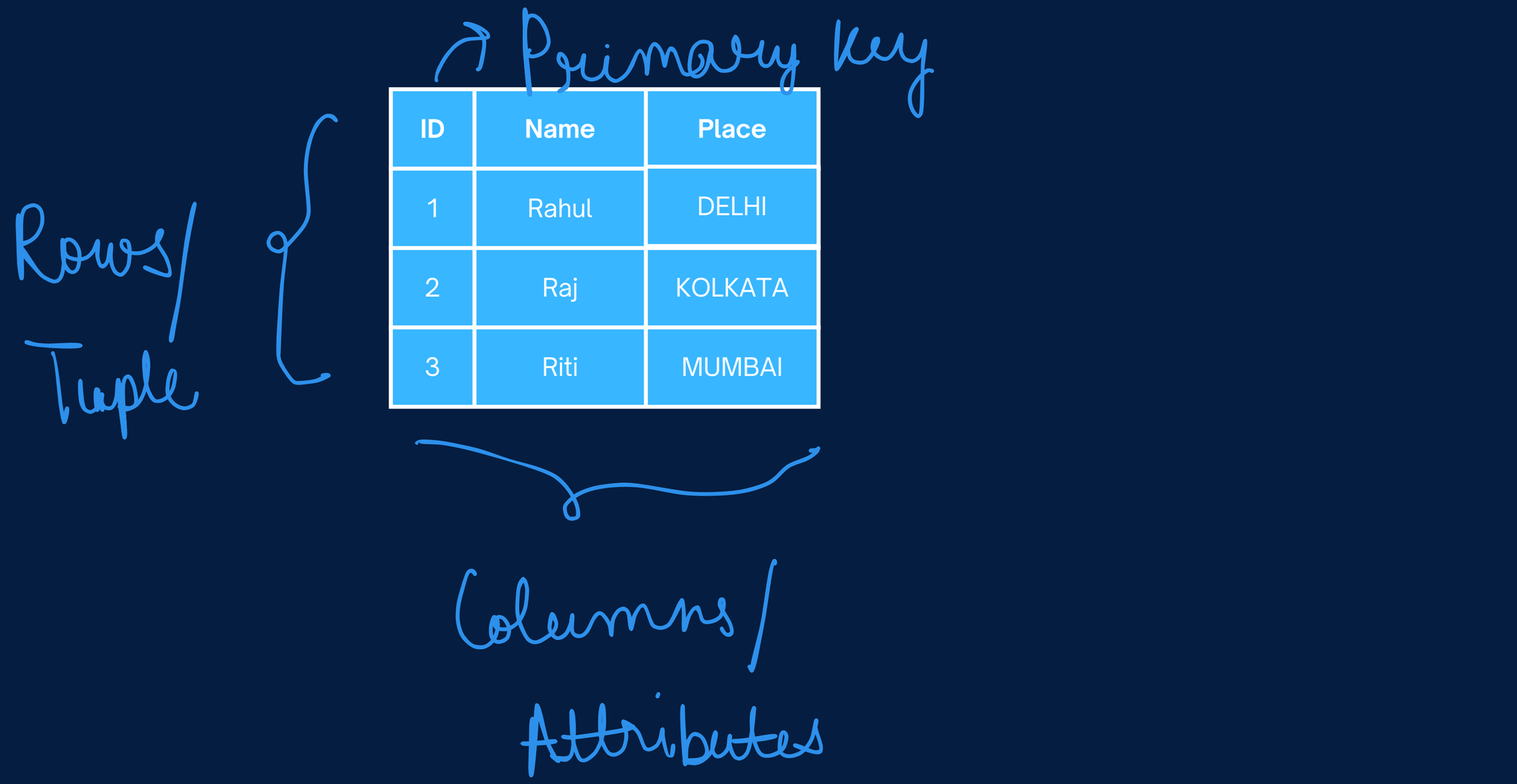
Row/Tuple – Rows, also known as records or tuples, represent individual entries or instances of data within the table.

Cardinality – No of rows in a table

Column/Attribute – Columns represent the attributes of the data being stored and are named to describe the information they hold (e.g., "ID," "Name," "Age").

Degree – No of Columns in a ta in a tableble

ESSENTIAL COMPONENTS OF TABLES



ESSENTIAL COMPONENTS OF TABLES

Constraints - Constraints define rules or conditions that must be satisfied by the data in the table.
Common constraints include uniqueness, nullability, default values, etc.

- Unique constraint: Ensures values in a column are unique across the table.
- Not null constraint: Ensures a column cannot have a null value.
- Check constraint: Enforces a condition to be true for each row.
- Default constraint: Provides a default value for a column if no value is specified.

Keys - A primary key is a unique identifier for each record in the table. It ensures that each row can be uniquely identified and accessed within the table.

A foreign key is a field in a table that refers to the primary key of another table. It establishes relationships between tables.

VIEWS IN DBMS

View is a virtual table that is derived from one or more underlying tables. This means that it doesn't physically store data but rather provides a logical representation of data.

Customer DB

ID.	NAME	**phn**	**Address**	**Pin**	**Age**
1	Raj	456	blr	123	18
2	Ravi	123	delhi	124	21
3	Ram	789	hyd	345	22

KEYS IN DBMS

Keys in DBMS make sure of data integrity, uniqueness, and the quick retrieval of information. Key is a attribute in table

Types of keys :

- **Candidate Key**
- **Primary Key**
- **Foreign Key**
- **Super Key**

KEYS IN DBMS

Candidate Key : A candidate key refers to a group of attributes capable of uniquely identifying a record within a table. Among these, one is selected to serve as the primary key.
Ex- For student possible attributes for candidate key could be

Student<ID, Roll no , Aadhar Card>

Age	Name	Hometown
20	Rahul	KOLKATA
21	Raj	KOLKATA
20	Riti	DELHI

KEYS IN DBMS

Primary Key : A primary key is a key which uniquely identifies each record in a table.
It ensures that each tuple or record can be uniquely identified within the table.
It is always **Unique+ Not null**

ID	Name	Hometown
123	Rahul	KOLKATA
245	Raj	KOLKATA
434	Riti	DELHI

KEYS IN DBMS

Foreign Key : A foreign key is a field in a table that refers to the primary key in another table. It establishes a relationship between two tables.

Student
(Base/referenced table)

Roll no	Name	Hometown
1	Rahul	KOLKATA
2	Raj	KOLKATA
3	Riti	DELHI

Primary key

Subject
(referencing table)

Roll no	Name	subject
1	Rahul	Maths
2	Raj	SST
3	Riti	Science

Foreign key

KEYS IN DBMS

Foreign Key : A foreign key is a field in a table that refers to the primary key in another table. It establishes a relationship between two tables.

Student
(Base/referenced table)

Roll no	Name	Hometown
1	Rahul	KOLKATA
2	Raj	KOLKATA
3	Riti	DELHI

Primary key

Subject
(referencing table)

Roll no	Name	subject
1	Rahul	Maths
2	Raj	SST
3	Riti	Science

Foreign key

KEYS IN DBMS

Referenced table - Table having primary key (pk)
Referencing table- Table having foreign key(fk)

Student
(Base/referenced table)

Roll no	Name	Hometown
1	Rahul	KOLKATA
2	Raj	KOLKATA
3	Riti	DELHI

Primary key

Subject
(referencing table)

Roll no	subject id	subject
1	s1	Maths
2	s2	SST
3	s3	Science

Foreign key

KEYS IN DBMS

Referential Integrity in Foreign key

Referential integrity is an important concept in foreign key. We always say foreign key maintains referential integrity.

Referential integrity ensures that the relationships between tables remain accurate, consistent, and meaningful within a relational database.

KEYS IN DBMS

Referential Integrity in Foreign key

Now consider there are two tables one is refrencing and other is refrenced table .
Lets see how some operations like insert, update and delete works here.

KEYS IN DBMS

Refrential Integrity in Foreign key

- Insertion in Referenced/base table

No violation

KEYS IN DBMS

Refrential Integrity in Foreign key

- Deletion in Referenced/base table

May cause violation if the coressponding data is present in refrencing table.

KEYS IN DBMS

Refrential Integrity in Foreign key

If a record in referenced table is deleted or updated , the corresponding records in the referencing table should be deleted or updated to maintain the integrity of the relationship.

We using action like "CASCADE DELETE" for the same. Also we can set null for the values deleted.

KEYS IN DBMS

Refrential Integrity in Foreign key

- Updation in Referenced/base table

May cause violation if the coressponding data is present in refrencing table. We can using action like "CASCADE UPDATE”.

KEYS IN DBMS

Refrential Integrity in Foreign key

- Insertion in Referencing table

May cause violation

KEYS IN DBMS

Refrential Integrity in Foreign key

- Deletion in Referencing table

No violation

KEYS IN DBMS

Refrential Integrity in Foreign key

- Updation in Referencing table

No issues untill we are updating foreign key attribute
Voilation would be caused on updating

KEYS IN DBMS

Referential Integrity in Foreign key

Referential integrity is an important concept in foreign key. We always say foreign key maintains referential integrity.

Referential integrity ensures that the relationships between tables remain accurate, consistent, and meaningful within a relational database.

KEYS IN DBMS

Referential Integrity in Foreign key

Now consider there are two tables one is refrencing and other is refrenced table .
Lets see how some operations like insert, update and delete works here.

KEYS IN DBMS

Refrential Integrity in Foreign key

- Insertion in Referenced/base table

No violation

KEYS IN DBMS

Refrential Integrity in Foreign key

- Deletion in Referenced/base table

May cause violation if the coressponding data is present in refrencing table.

KEYS IN DBMS

Refrential Integrity in Foreign key

If a record in referenced table is deleted or updated , the corresponding records in the referencing table should be deleted or updated to maintain the integrity of the relationship.

We using action like "CASCADE DELETE" for the same. Also we can set null for the values deleted.

KEYS IN DBMS

Refrential Integrity in Foreign key

- Insertion in Referencing table

May cause violation

KEYS IN DBMS

Refrential Integrity in Foreign key

- Deletion in Referencing table

No violation

KEYS IN DBMS

Refrential Integrity in Foreign key

- Updation in Referencing table

No issues untill we are updating foreign key attribute
Voilation would be caused on updating

INTEGRITY CONSTRAINT IN DBMS

Integrity constraints help to ensure that data remains reliable and meaningful throughout its lifecycle.

Types of Integrity Constraint:

- **Domain Integrity Constraint**
- **Entity Integrity Constraint**
- **Referential Integrity Constraint**
- **Key Constraint**
- **Check Constraint**
- **Null Constraint**
- **Unique Constraint**
- **Default Constraint**

INTEGRITY CONSTRAINT IN DBMS

Domain Integrity Constraint

It ensures the validity and appropriateness of data values
(i.e valid data types, ranges, and formats for columns)
within a specific column or attribute of a table.

Ex-> Check for date column so that it contains valid date values

INTEGRITY CONSTRAINT IN DBMS

Entity Integrity Constraint

It ensures that each row/record in a table is uniquely identified by a primary key.

It also helps in preventing duplicate or null values in the primary key.

INTEGRITY CONSTRAINT IN DBMS

Referential Integrity Constraint

It ensures that values in a foreign key column match with the values in the corresponding primary key column in another table.

INTEGRITY CONSTRAINT IN DBMS

Key Constraint

It ensures uniqueness for the primary key.

INTEGRITY CONSTRAINT IN DBMS

Check Constraint

It checks for a condition that each row in a table must satisfy.

If the condition is not met, the insertion or update of the row is rejected.

INTEGRITY CONSTRAINT IN DBMS

Null Constraint

It determines whether a column in a table can have null (i.e., missing or unknown) values or not.

INTEGRITY CONSTRAINT IN DBMS

Unique Constraint

It ensures that values in a specified column or combination of columns are unique across a table.

This constraint prevents duplicate values from being inserted into the specified column(s), maintaining data consistency and integrity.

INTEGRITY CONSTRAINT IN DBMS

Default Constraint

It ensures a default value for a column, which is used if no other value is provided

SUPER KEY IN DBMS

It is a set of one or more attributes (columns) that can uniquely identify a tuple (a row) in a relation (table).

Superset of any candidate key.

A super key becomes a candidate key if it is minimal (i.e. no proper subset of it can uniquely identify a tuple).

ER MODEL IN DBMS

Introduction to ER Model

- **Entity**
  Things/Object
  Ex-person
- **Attributes**
  Properties of entity
  Ex-name,age
- **Relationship**
  association among entities
  Ex-Works for

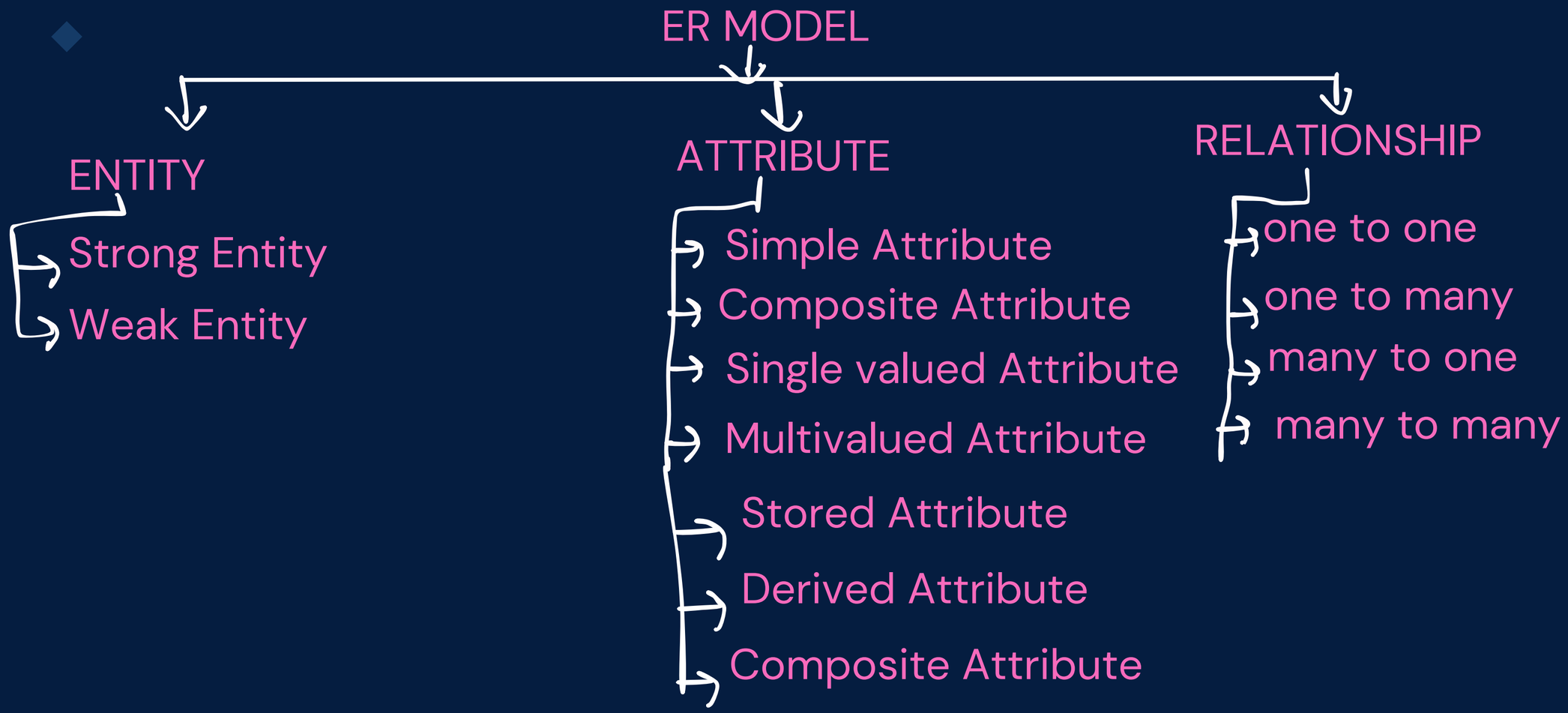
ER MODEL IN DBMS

- The Entity-Relationship (ER) model stands as a prevalent conceptual modeling approach within the realm of database design.

- Its primary role is to offer a visual representation of a database's architecture by illustrating the entities, their respective attributes, and the interconnections between them.

- In the process of database design, the ER model holds significant importance, aiding in the development of an efficient and systematically structured database schema.

ER MODEL IN DBMS

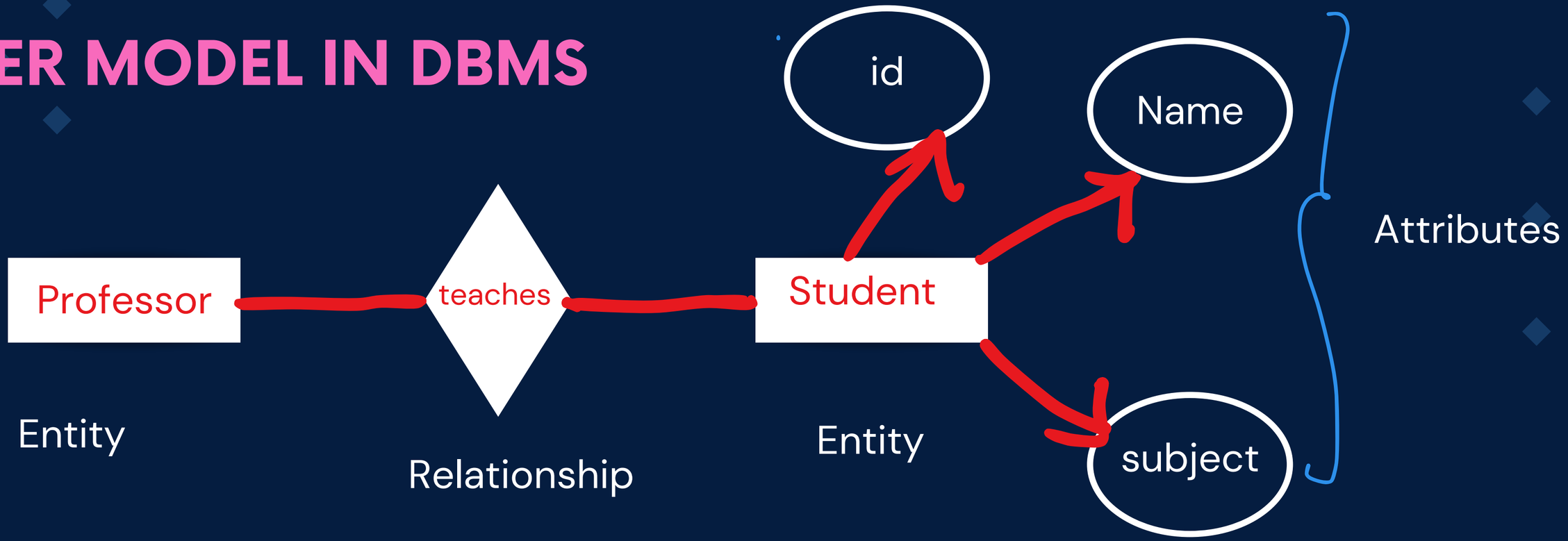


ER MODEL IN DBMS

Symbols used in ER Model

Figures	Symbols	For what
Rectangle		Entity
Ellipse		Attribute
Diamond		Relationship
Line		**Attribute to entity relationship**
Double ellipse		Multivalued attributes
Double rectangle		Weak Entity

ER MODEL IN DBMS

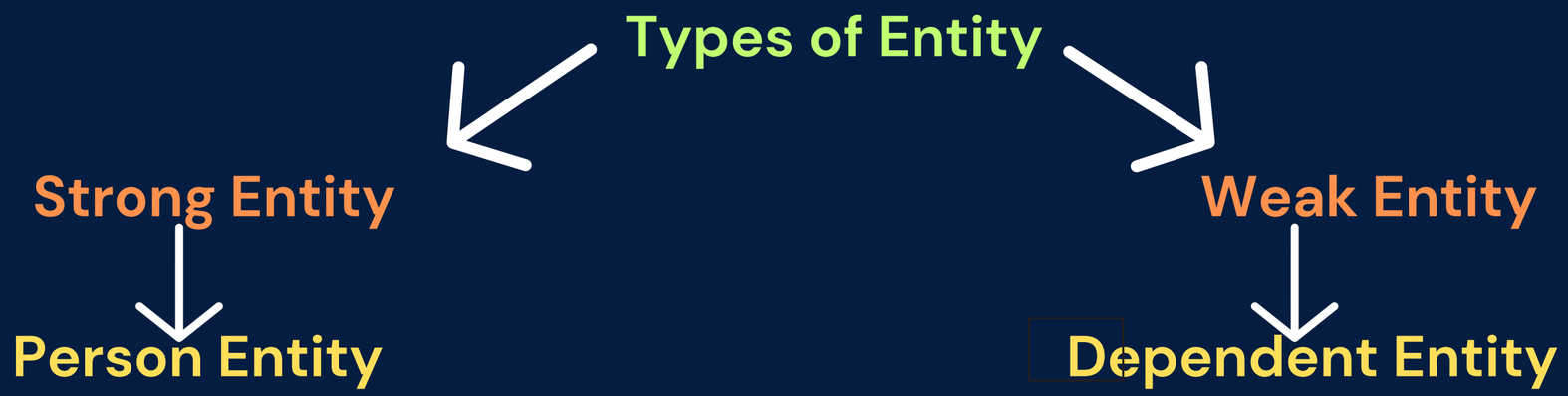


ER MODEL IN DBMS

- Entity

An entity is something from the real world, like a person, place, event, or idea. Each entity has specific features or traits that describe it.

ER MODEL IN DBMS



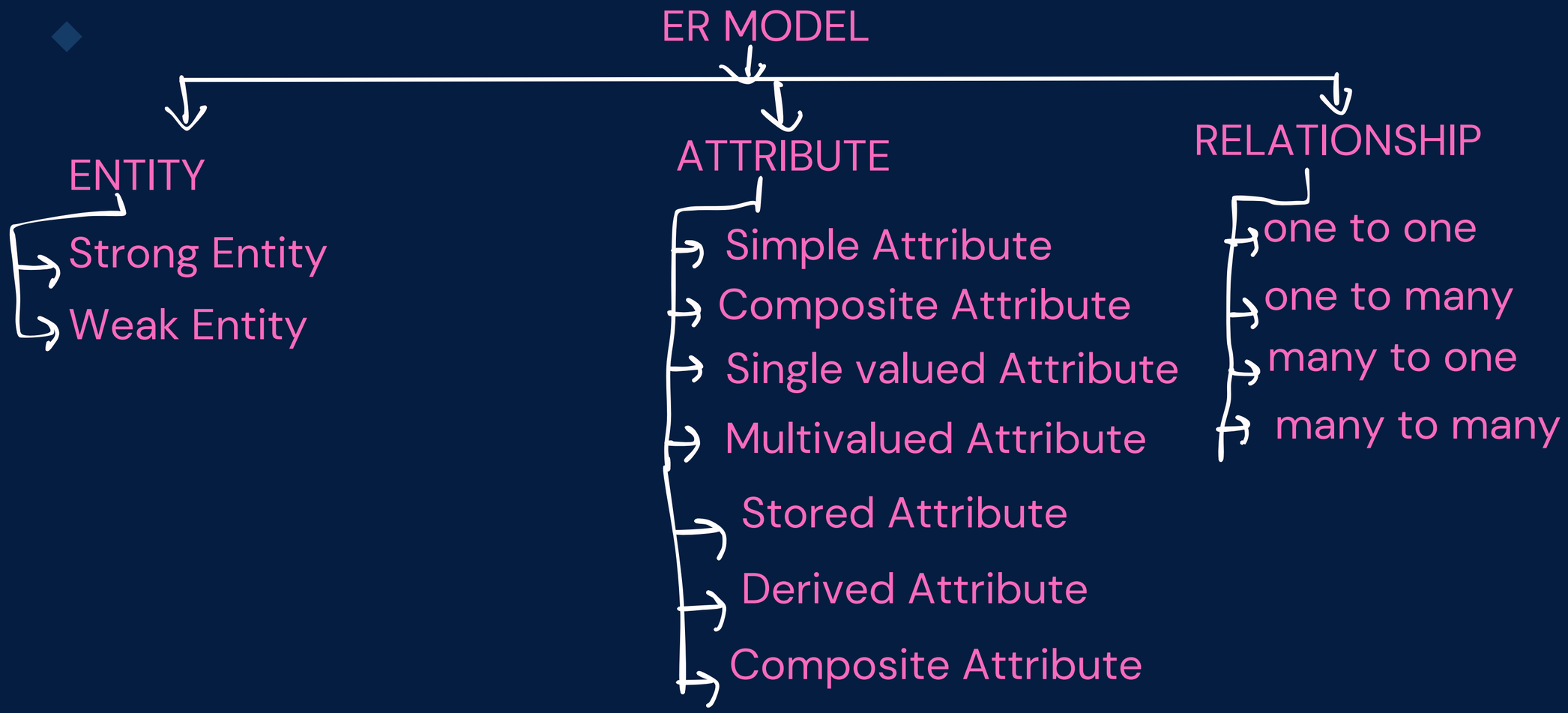
ER MODEL IN DBMS

- **Types of Entity**

Strong Entity: A strong entity is an entity that has its own unique identifier (primary key) and is not dependent on any other entity for its existence within the database. Strong entities stand alone and have their own set of attributes.
Ex-Person

Weak Entity: A weak entity is an entity that doesn't have a primary key of its own. It relies on a related strong entity (known as the "owner" entity) for its identity. The weak entity's existence is defined by being related to the owner entity.
ex- dependent

ER MODEL IN DBMS



ER MODEL IN DBMS

Attribute

Attributes represent properties or characteristics of an entity or relationship.

They provide information about the entities and relationships in the database.

ER MODEL IN DBMS

- **Types of Attributes**

Simple Attribute

A simple attribute is atomic and cannot be divided any further.
Ex- First Name

ER MODEL IN DBMS

- **Types of Attributes**

Composite Attribute

A composite attribute is made up of several smaller parts, where each part represents a piece of the whole attribute. In simpler terms it is composed of attributes which can be divided further.

Ex- Name(First Name, lastName)

ER MODEL IN DBMS

- **Types of Attributes**

Single Valued Attribute

A single-value attribute is an attribute that holds a single value for each entity

Ex- Age

ER MODEL IN DBMS

- **Types of Attributes**

Multivalued Attribute

A multi-valued attribute in a database is an attribute that can hold multiple values for a single entity.

Ex- Address (permanent, residential)

ER MODEL IN DBMS

- **Types of Attributes**

Stored Attribute

Attribute that is stored as a part of a database record.

Ex- Date of birth

ER MODEL IN DBMS

- **Types of Attributes**

Derived Attribute

A derived attribute is derived from other attributes within the database.

Ex- Age derived from dob

ER MODEL IN DBMS

- **Types of Attributes**

Complex Attribute

A complex attribute is an attribute that is made up of multiple smaller attributes

Ex- Name(Composite) → FirstName, Middle Name, lastName (Simple attribute)

ER MODEL IN DBMS

Relationship in ER Model

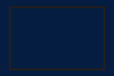
Relationship in ER MODEL is the connection between entities (tables) based on related data.

ER MODEL IN DBMS

Types of Relationship

Strong Relationship

Weak Relationship

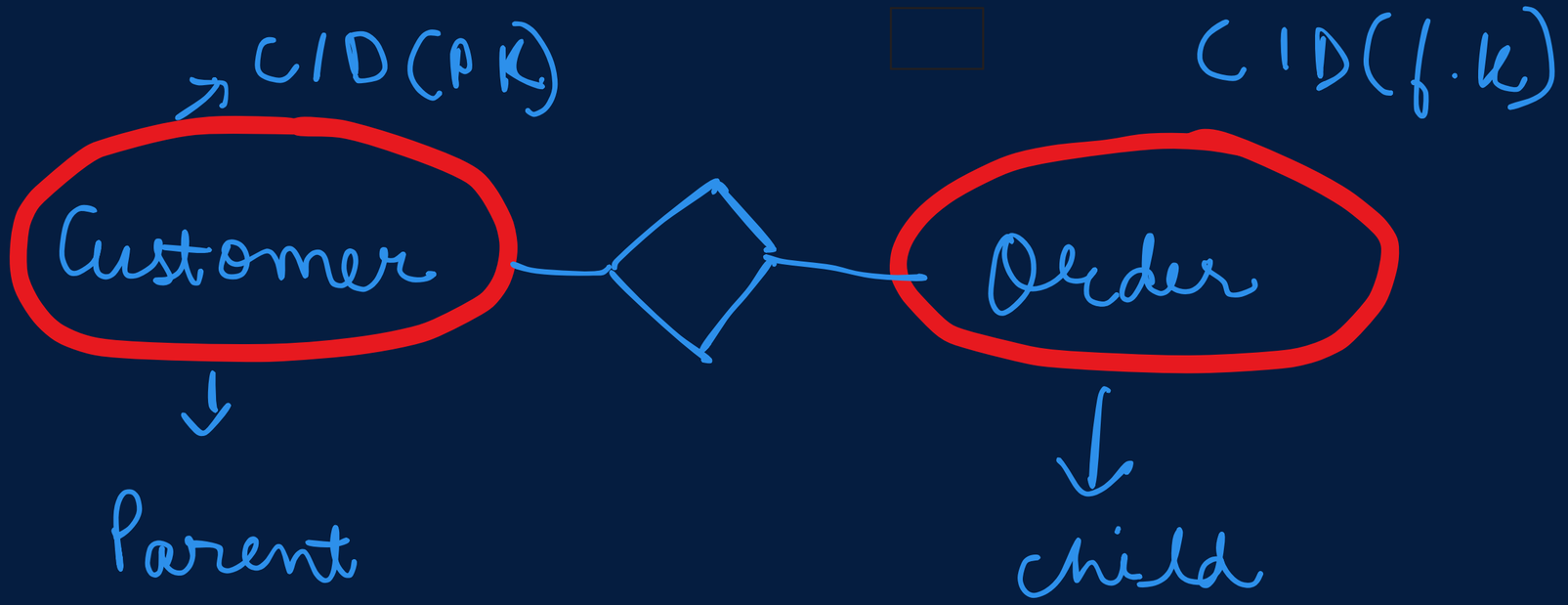


ER MODEL IN DBMS

Strong Relationship

A strong relationship exists when two entities are highly dependent on each other, and one entity cannot exist without the other.

Ex-

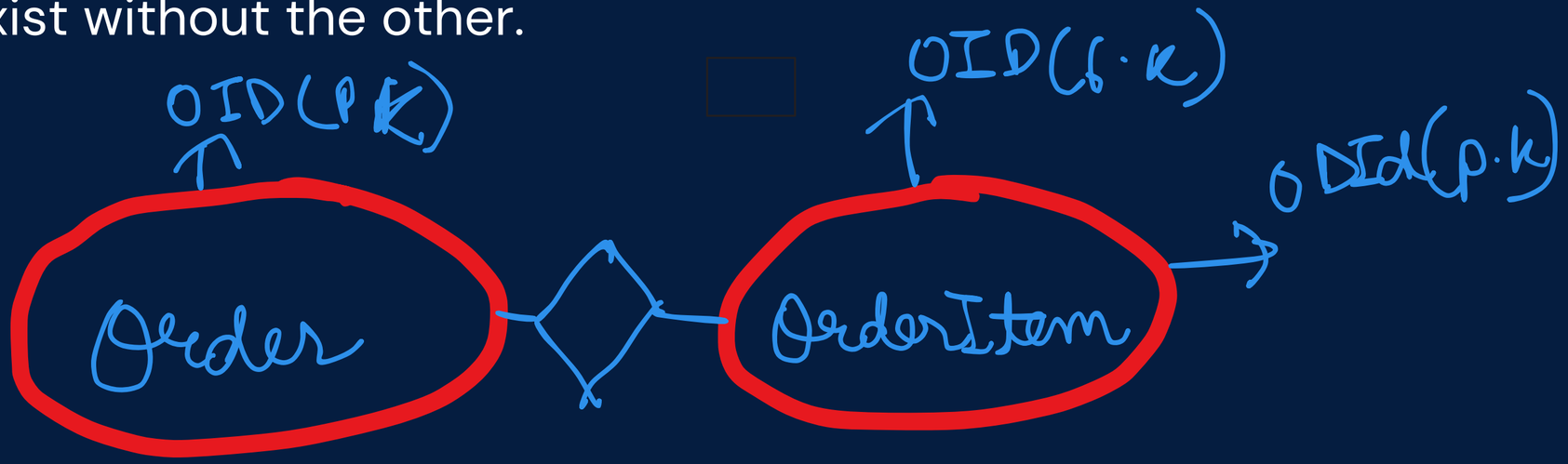


ER MODEL IN DBMS

Weak Relationship

A weak relationship, on the other hand, exists when two entities are related, but one entity can exist without the other.

Ex-



ER MODEL IN DBMS

Degree in DBMS

A degree in dbms refers to the number of attributes / columns that a relation/table has.

ER MODEL IN DBMS

Types of Degree

Degree	Name	Defination
1	**Unary Degree**	A relation with a single attribute
2	**Binary Degree**	A relation with two attributes
3	**Ternary Degree**	A relation with three attributes
n	n-ary Degree	A relation with more than three attributes n>3

ER MODEL IN DBMS

Null value : In databases, a null value can occur for various reasons

Not Needed Information: Sometimes, some details are asked, but they don't apply to everyone. For instance, asking for a "Spouse Name" from someone who isn't married.

Don't Know the Answer: Every now and then, we're asked a question, but we don't have an answer yet.

Forgot to Fill In: Like when you're filling out a form, and you accidentally miss putting in some important information.

ER MODEL IN DBMS

Types of relationship in dbms (Based on degree)

There are 4 types of relationship:

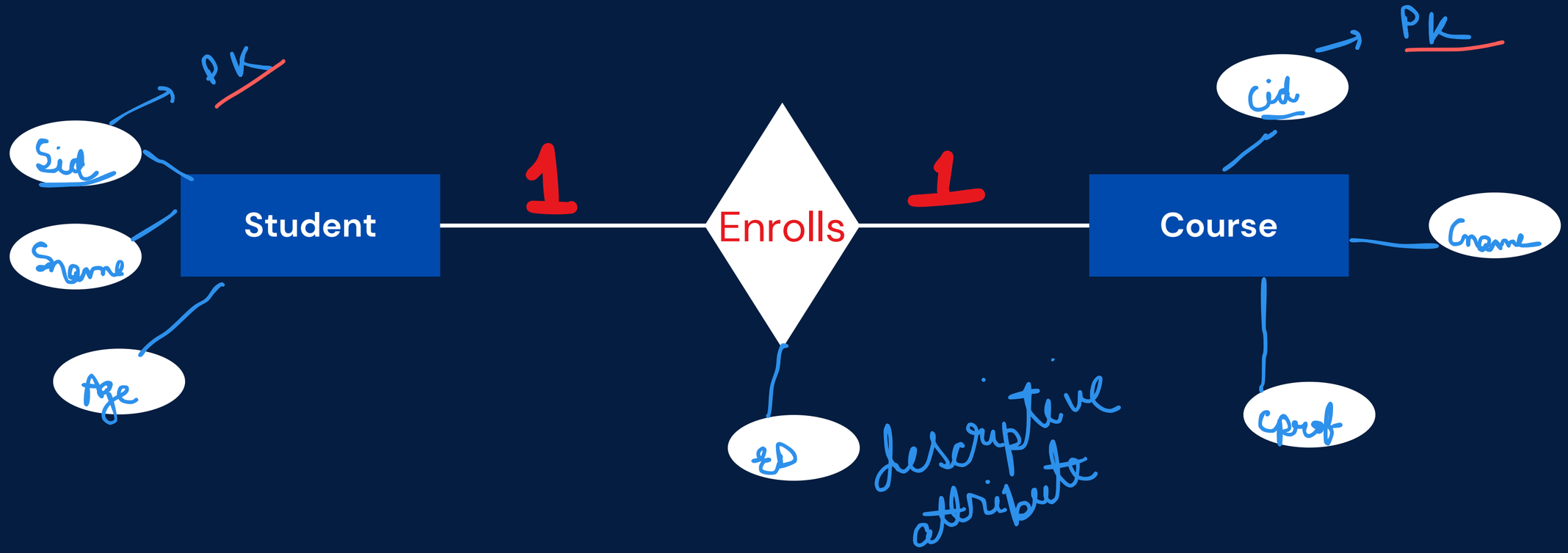
- one to one (1-1)
- one to many (1-N)
- many to one (N-1)
- many to many (N-N)

ER MODEL IN DBMS

- ## Types of Relationship(Cardinality)

1 to 1 Relationship(1:1)

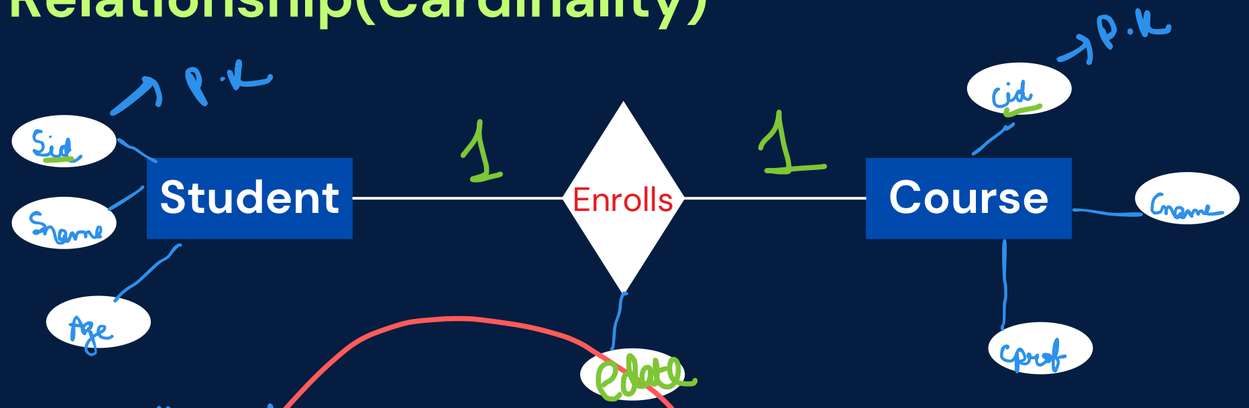
Each row in one table is associated with one and only one row in the other table, and vice versa.



ER MODEL IN DBMS

- Types of Relationship(Cardinality)

1 to 1 Relationship(1:1)

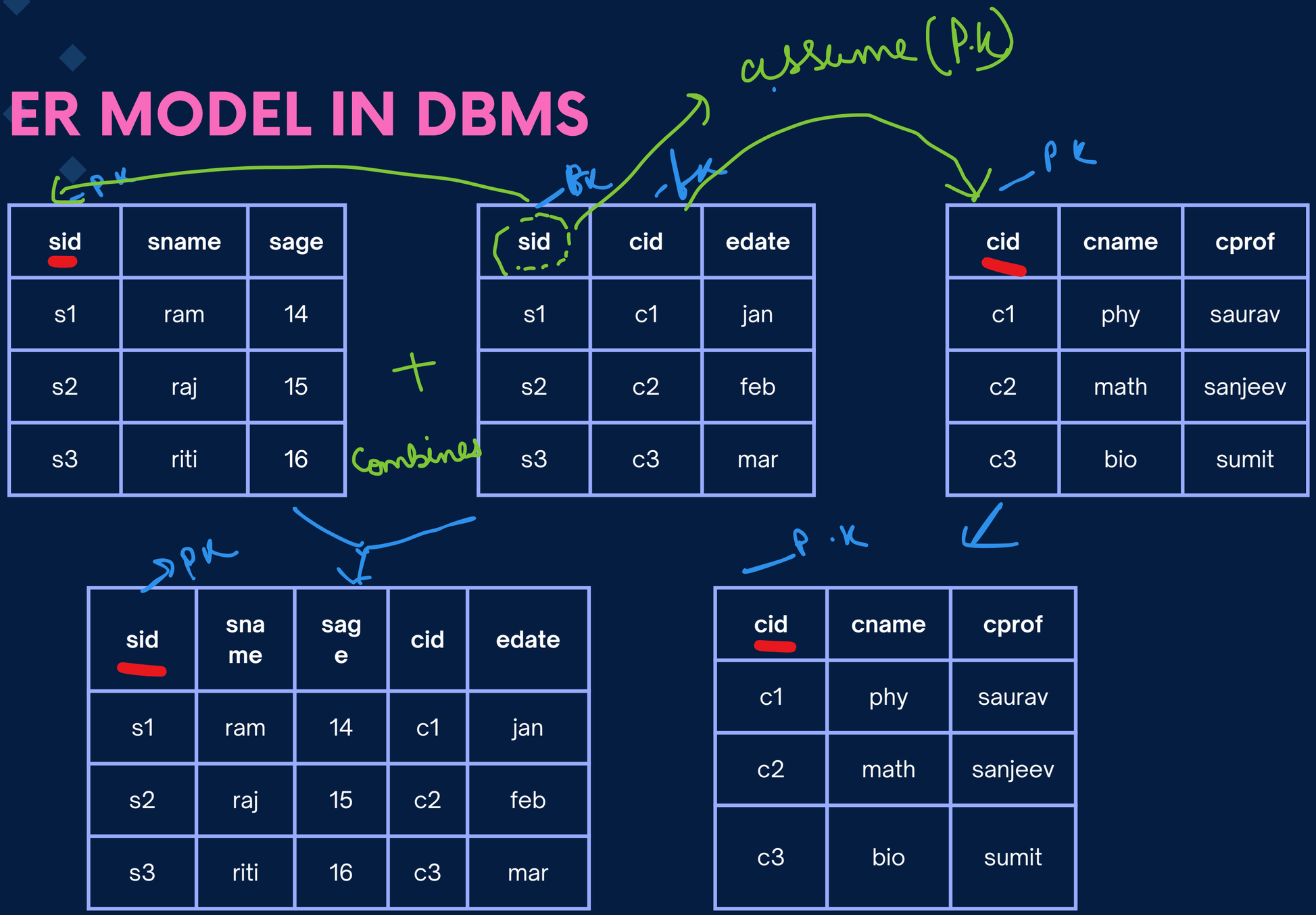


sid	sname	sage
s1	ram	14
s2	raj	15
s3	riti	16

sid	cid	edate
s1	c1	jan
s2	c2	feb
s3	c3	mar

cid	cname	cprof
c1	phy	saurav
c2	math	sanjeev
c3	bio	sumit

ER MODEL IN DBMS



sid	sname	sage
s1	ram	14
s2	raj	15
s3	riti	16

sid	cid	edate
s1	c1	jan
s2	c2	feb
s3	c3	mar

cid	cname	cprof
c1	phy	saurav
c2	math	sanjeev
c3	bio	sumit

sid	sname	sage	cid	edate
s1	ram	14	c1	jan
s2	raj	15	c2	feb
s3	riti	16	c3	mar

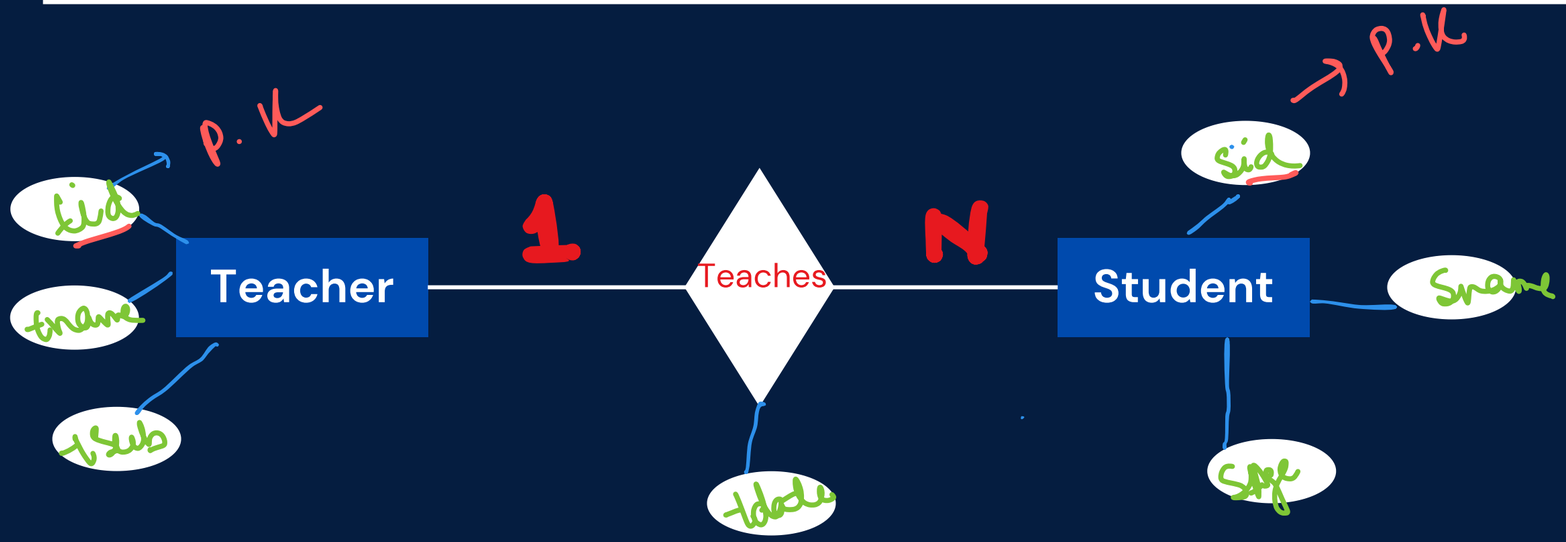
cid	cname	cprof
c1	phy	saurav
c2	math	sanjeev
c3	bio	sumit

ER MODEL IN DBMS

- Types of Relationship

1 to Many Relationship(1:N)

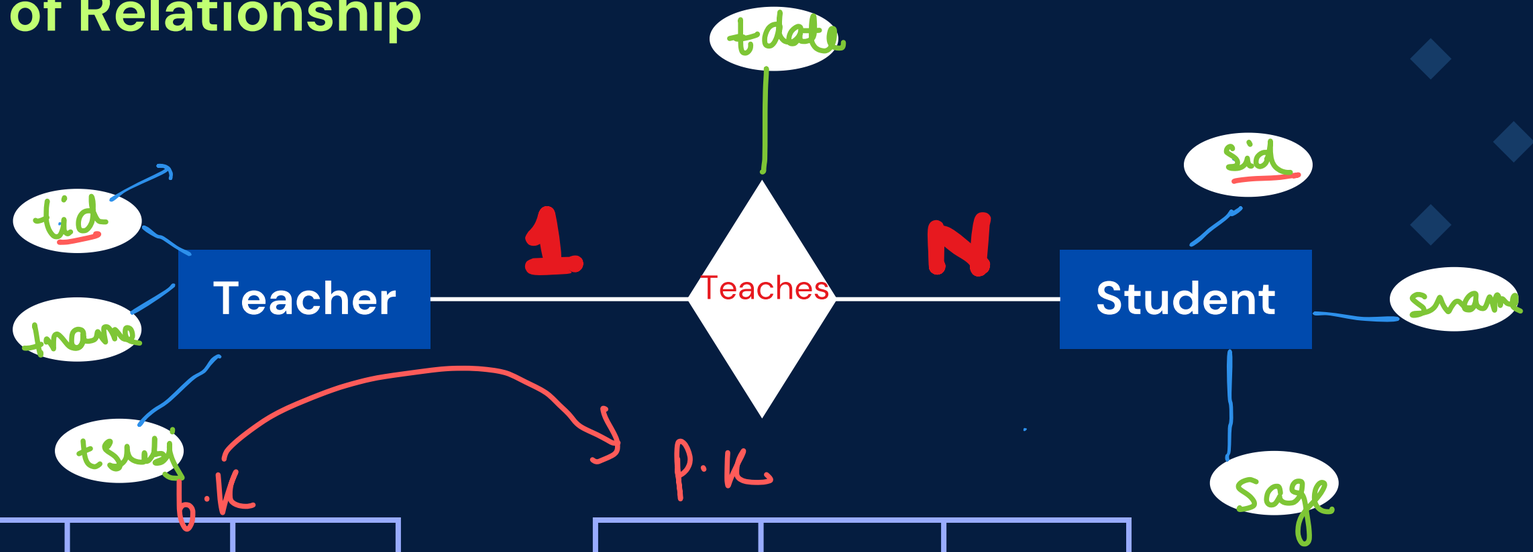
A single record in one table may have connections to multiple records in another table, while each record in the second table is linked to only one record in the first table.



ER MODEL IN DBMS

- Types of Relationship

1 to Many Relationship(1:N)



P.K FK P.K FK P.K

sid	sname	sage
s1	ram	14
s2	raj	15
s3	riti	16

teacher

sid	tid	tdate
s1	c1	jan
s2	c2	feb
s3	c3	mar

teaches

tid	tsubj	tname
c1	phy	saurav
c2	math	sanjeev
c3	bio	sumit

Subj

ER MODEL IN DBMS

sid	sname	sage
s1	ram	14
s2	raj	15
s3	riti	16

sid	tid	tdate
s1	c1	jan
s2	c2	feb
s2	c3	mar

tid	tname	tsub
c1	phy	saurav
c2	math	sanjeev
c3	bio	sumit

combines

sid	sname	sage
s1	ram	14
s2	raj	15
s3	riti	16

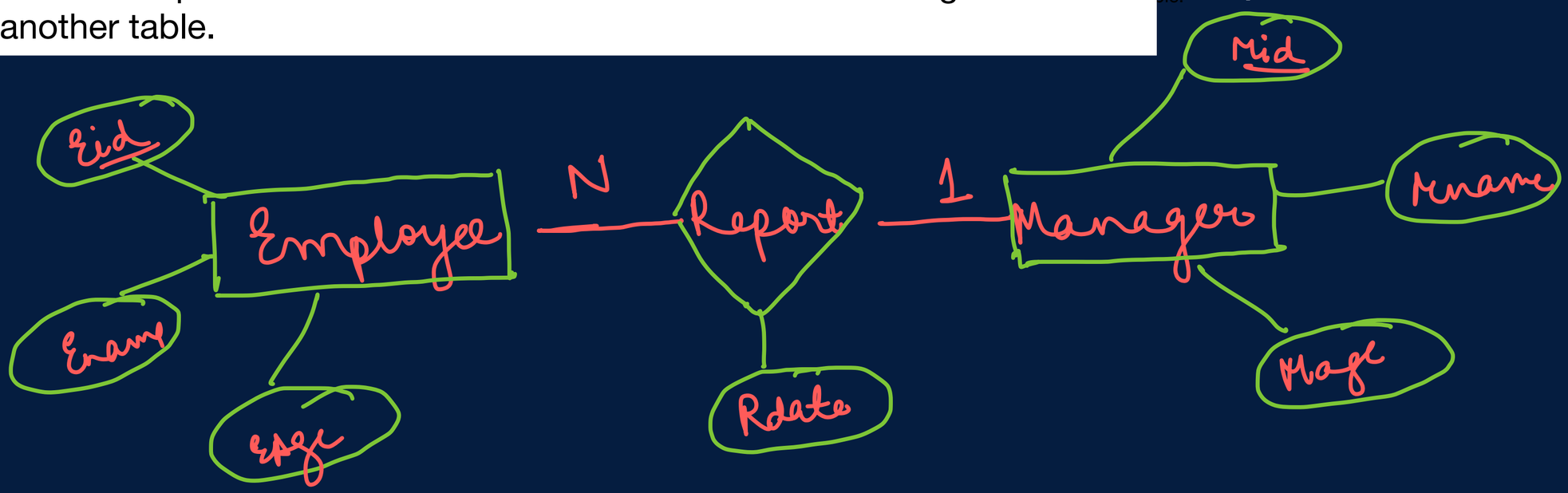
tid	tsub	tname	sid
c1	phy	saurav	s1
c2	math	sanjeev	s2
c3	bio	sumit	s3

ER MODEL IN DBMS

- Types of Relationship

Many to 1 Relationship(N:1)

When multiple records in one table are associated with a single record in another table.



ER MODEL IN DBMS

- Types of Relationship

Many to 1 Relationship(N:1)

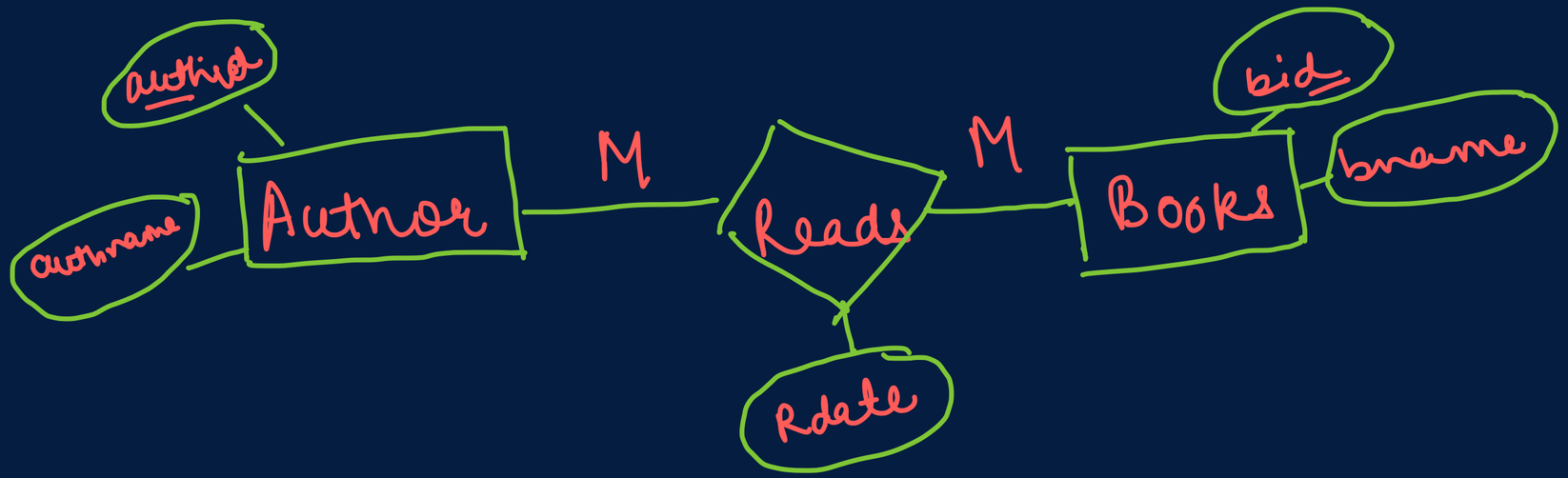
1. Primary Key → the one at the many side (End)
2. Reduction → Combine the many + Relationship table.
   (Emp + Report)

ER MODEL IN DBMS

- Types of Relationship

Many to many Relationship(N:N)

When multiple records in one table can be associated with multiple records in another table

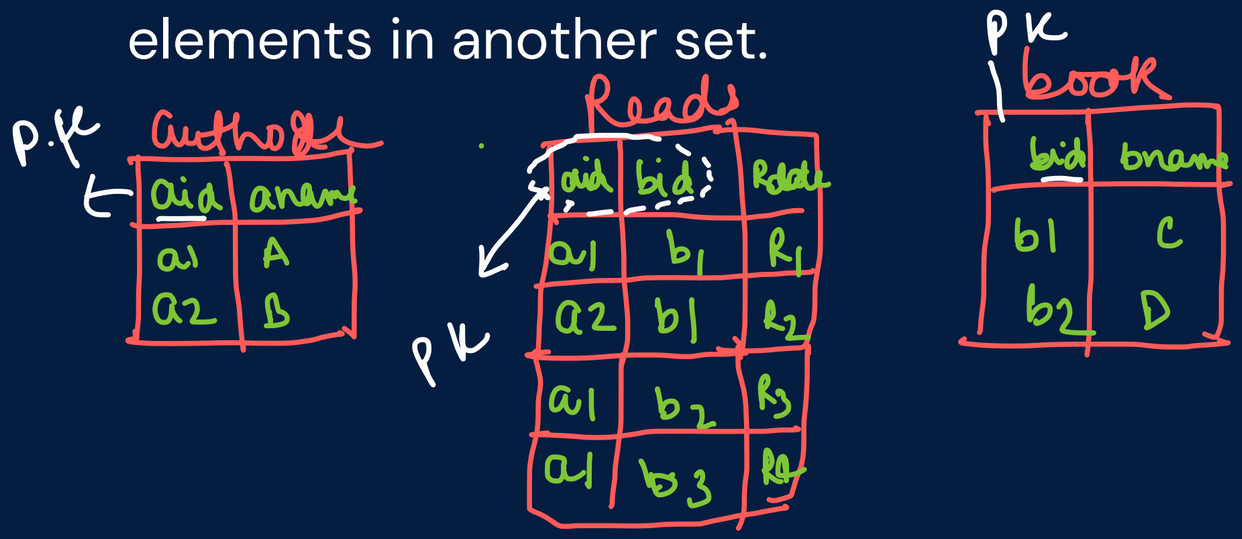


ER MODEL IN DBMS

- **Types of Relationship**

Many to many Relationship(N:N)

In this relationship multiple elements from one set are related to multiple elements in another set.

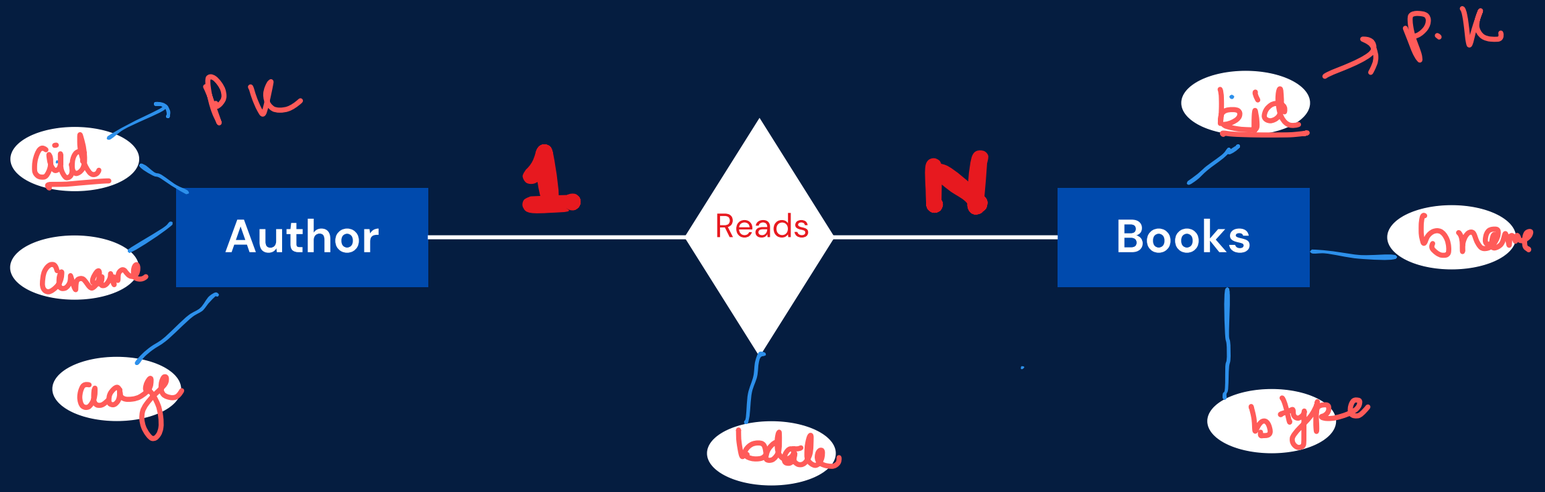


ER MODEL IN DBMS

- **Types of Relationship**

1 to Many Relationship(1:N)

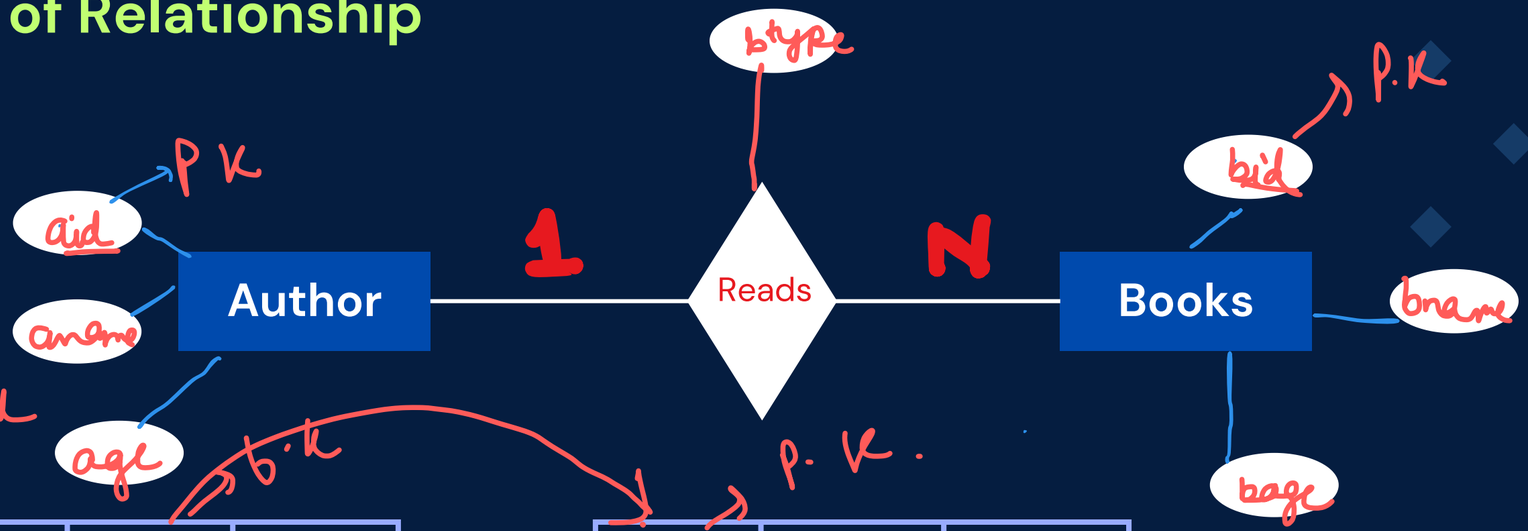
A database model where one entity (record) on one side of the relationship is associated with multiple entities (records) on the other side



ER MODEL IN DBMS

- Types of Relationship

1 to Many Relationship(1:N)



aid	aname	aage
a1	ram	14
a2	raj	15
a3	riti	16

aid	bid	bdate
a1	b1	jan
a2	b2	feb
a1	b3	mar

bid	bname	btype
b1	ab	fiction
b2	cd	thrill
b3	ef	drama

ER MODEL IN DBMS

P.K → always on many side

aid	aname	aage
a1	ram	14
a2	raj	15
a3	riti	16

aid	bid	bdate
a1	b1	jan
a2	b2	feb
a1	b3	mar

+

bid	bname	btype
b1	ab	fiction
b2	cd	thrill
b3	ef	drama

aid	aname	aage
a1	ram	14
a2	raj	15
a3	riti	16

bid	bname	btype	aid	bdate
b1	ab	fiction	a1	jan
b2	cd	thrill	a2	feb
b3	ef	drama	a1	mar

ER MODEL IN DBMS

- **Types of Relationship**

Many to 1 Relationship(N:1)

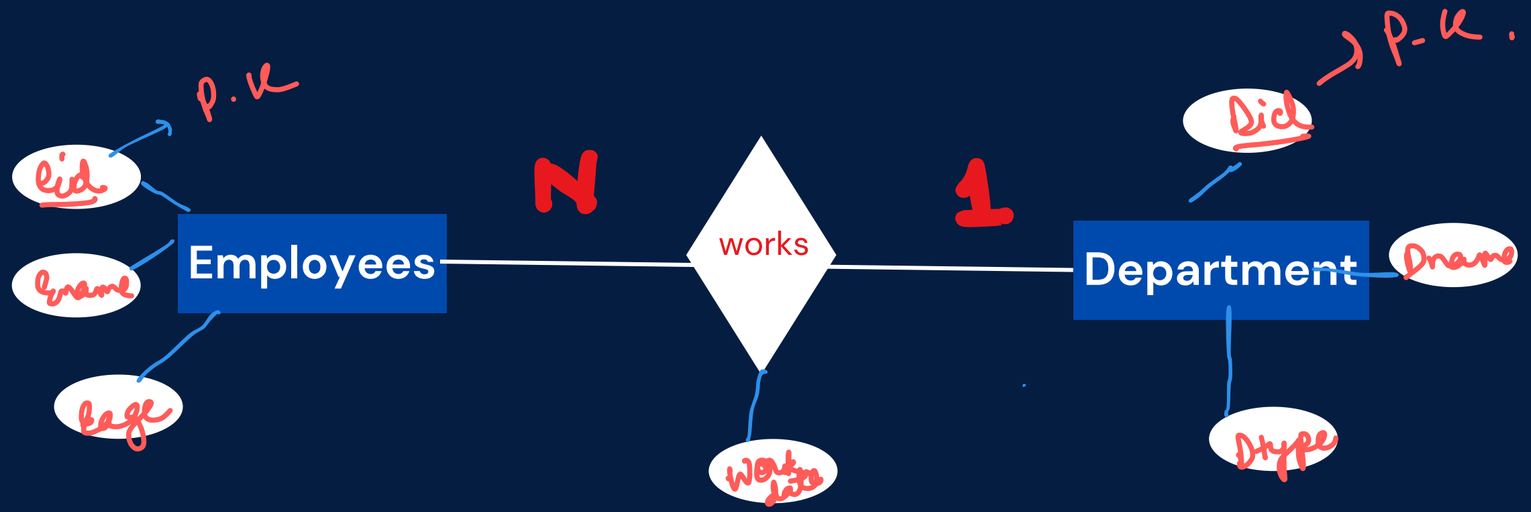
A database model where multiple entities (records) on one side of the relationship are associated with a single entity (record) on the other side.

ER MODEL IN DBMS

1. P.K → In the many side i.e Eid
2) Can it be reduced → Yes.
Emp + works, Dept

- Types of Relationship

Many to 1 Relationship(N:1)

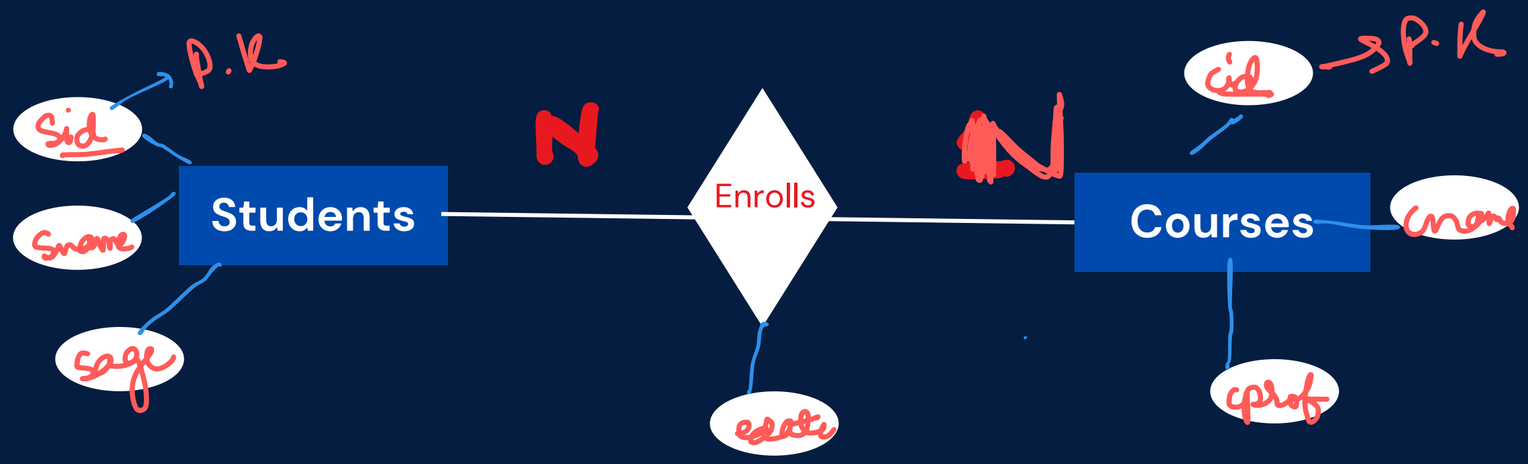


ER MODEL IN DBMS

- **Types of Relationship**

Many to many Relationship(N:N)

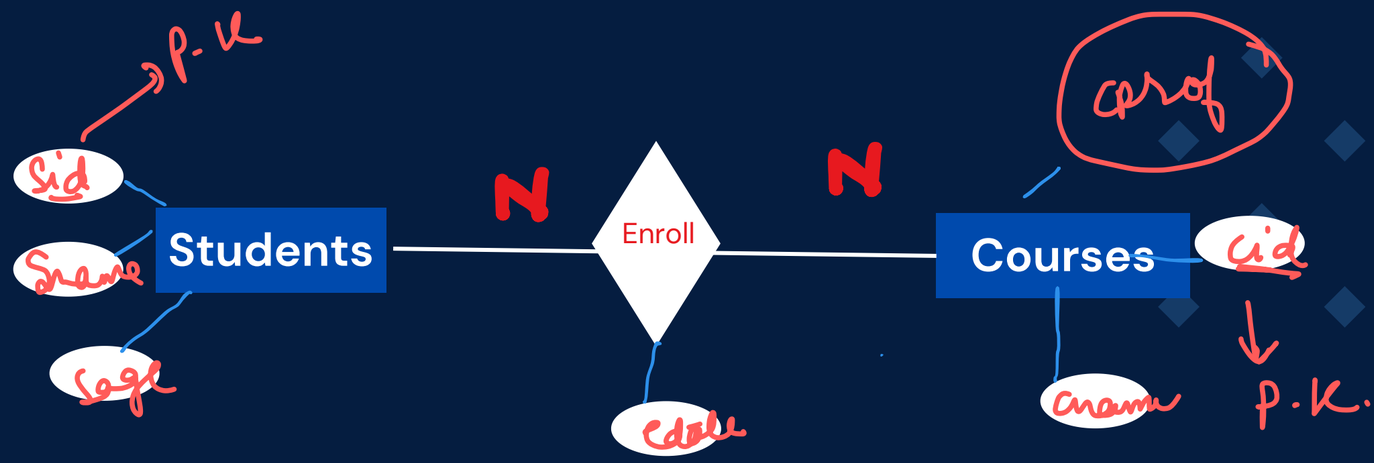
A database model where multiple entities (records) on one side of the relationship are associated with multiple entities on the other side.



ER MODEL IN DBMS

- Types of Relationship

Many to many Relationship(N:N)



P.K

sid	sname	sage
s1	ram	14
s2	raj	15
s3	riti	16

Student (Base table)

f.K f.K

sid	cid	edate
s1	c1	jan
s2	c2	feb
s3	c3	mar

Enroll (Refrencing table)

P.K

cid	cname	cprof
c1	phy	saurav
c2	math	sanjeev
c3	bio	sumit

course (base table)

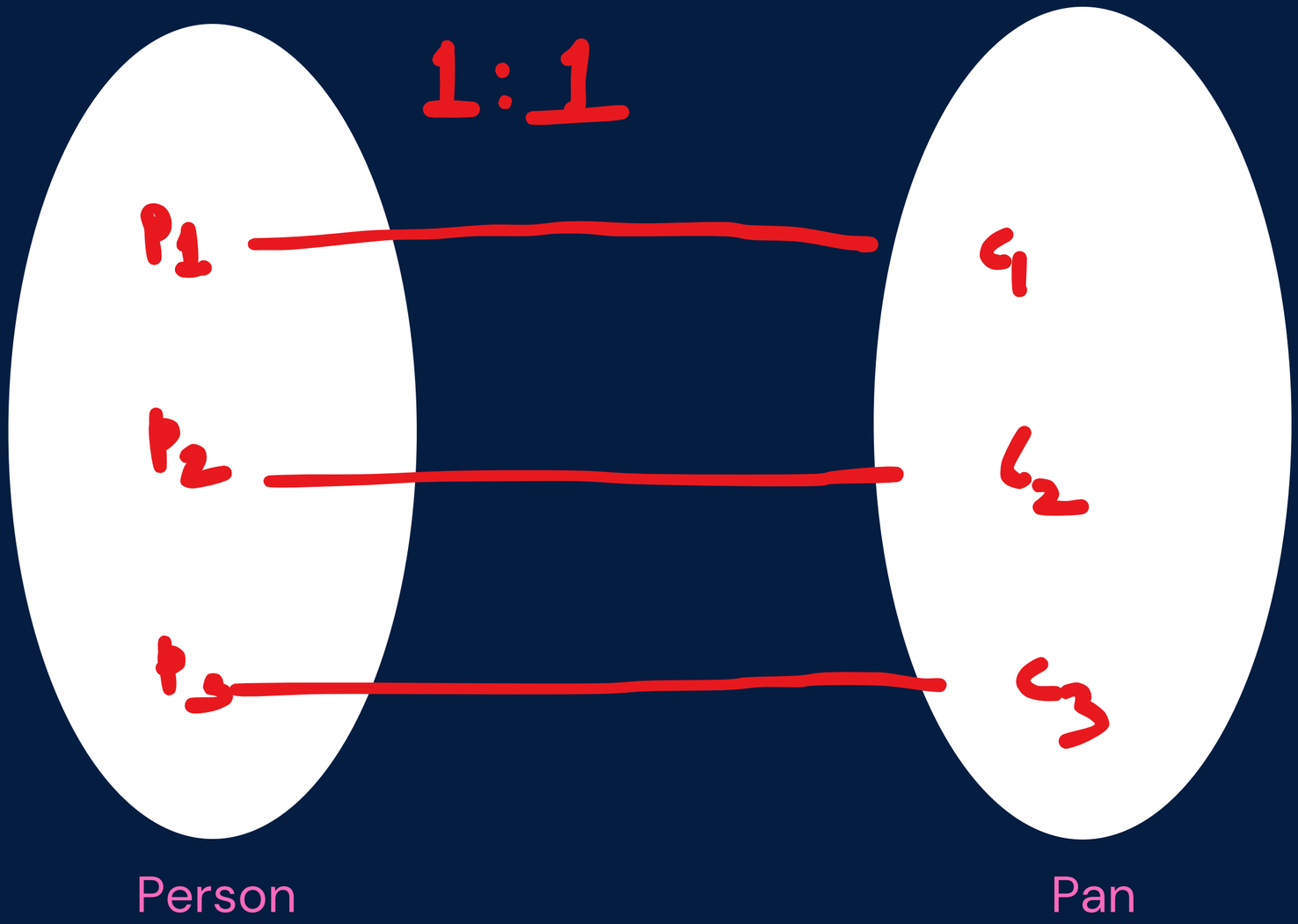
ER MODEL IN DBMS

sid	sname	sage
s1	ram	14
s2	raj	15
s3	riti	16

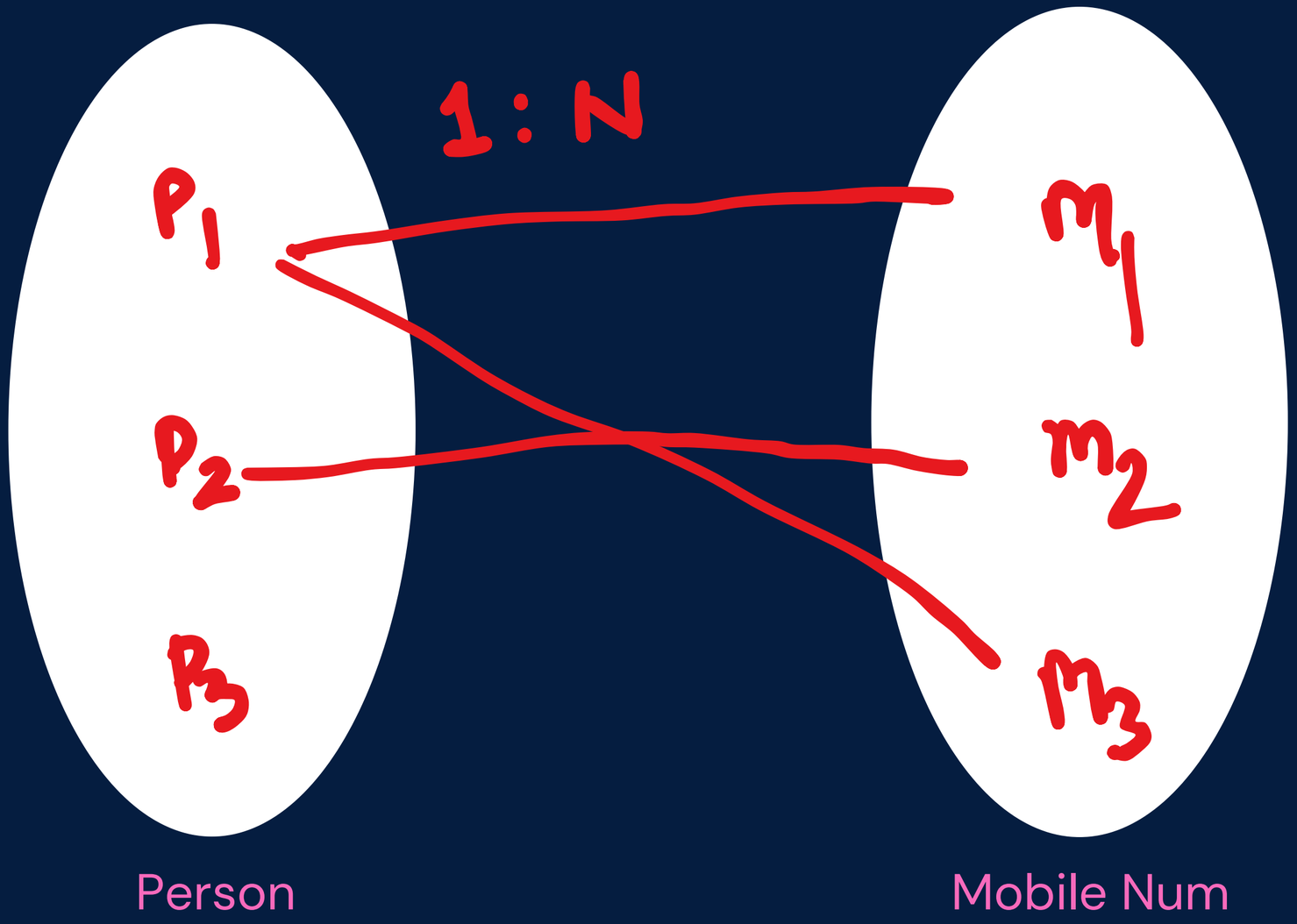
sid	bid	edate
s1	c1	jan
s2	c2	feb
s3	c3	mar

cid	cname	cprof
c1	phy	saurav
c2	math	sanjeev
c3	bio	sumit

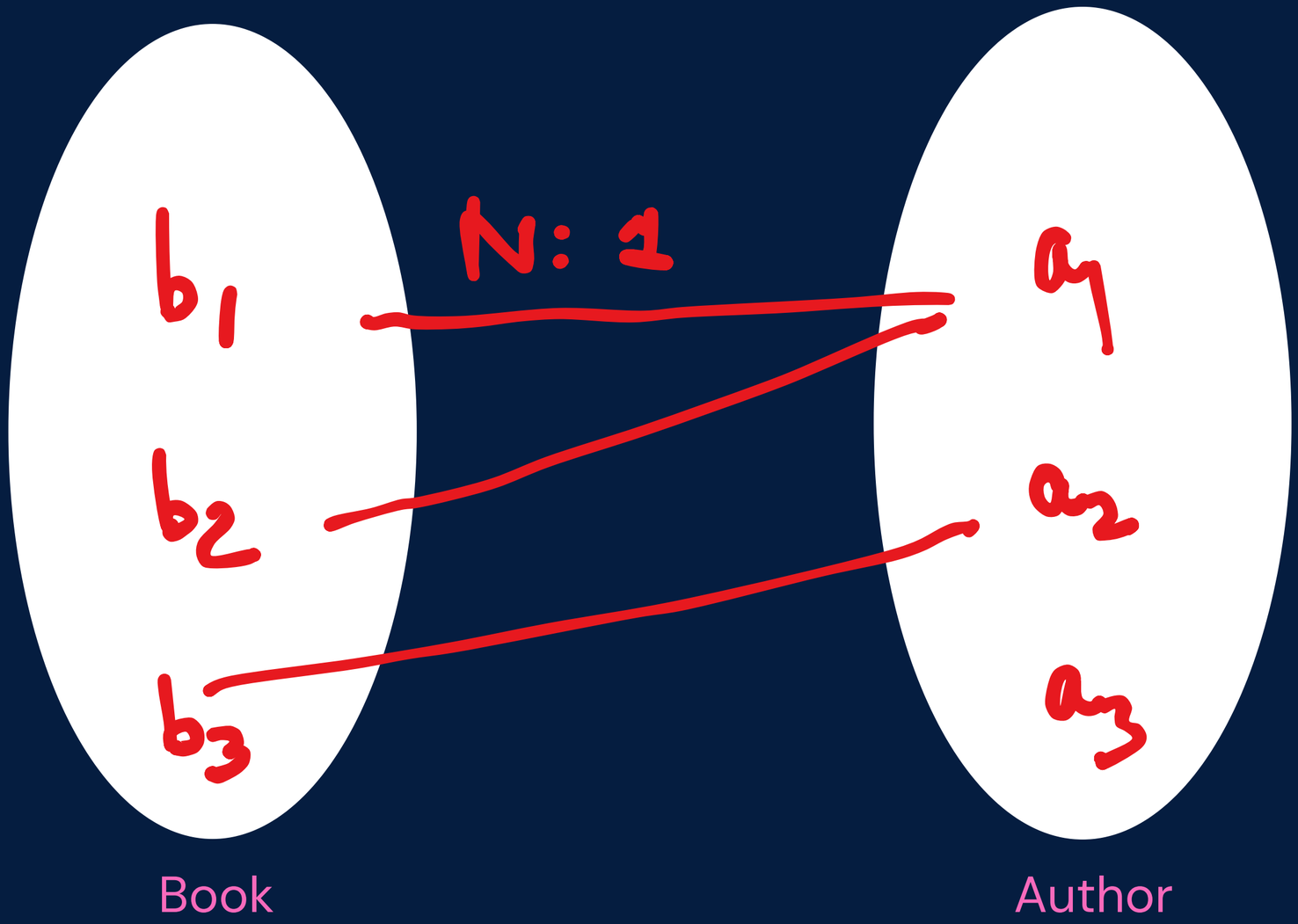
ER MODEL IN DBMS



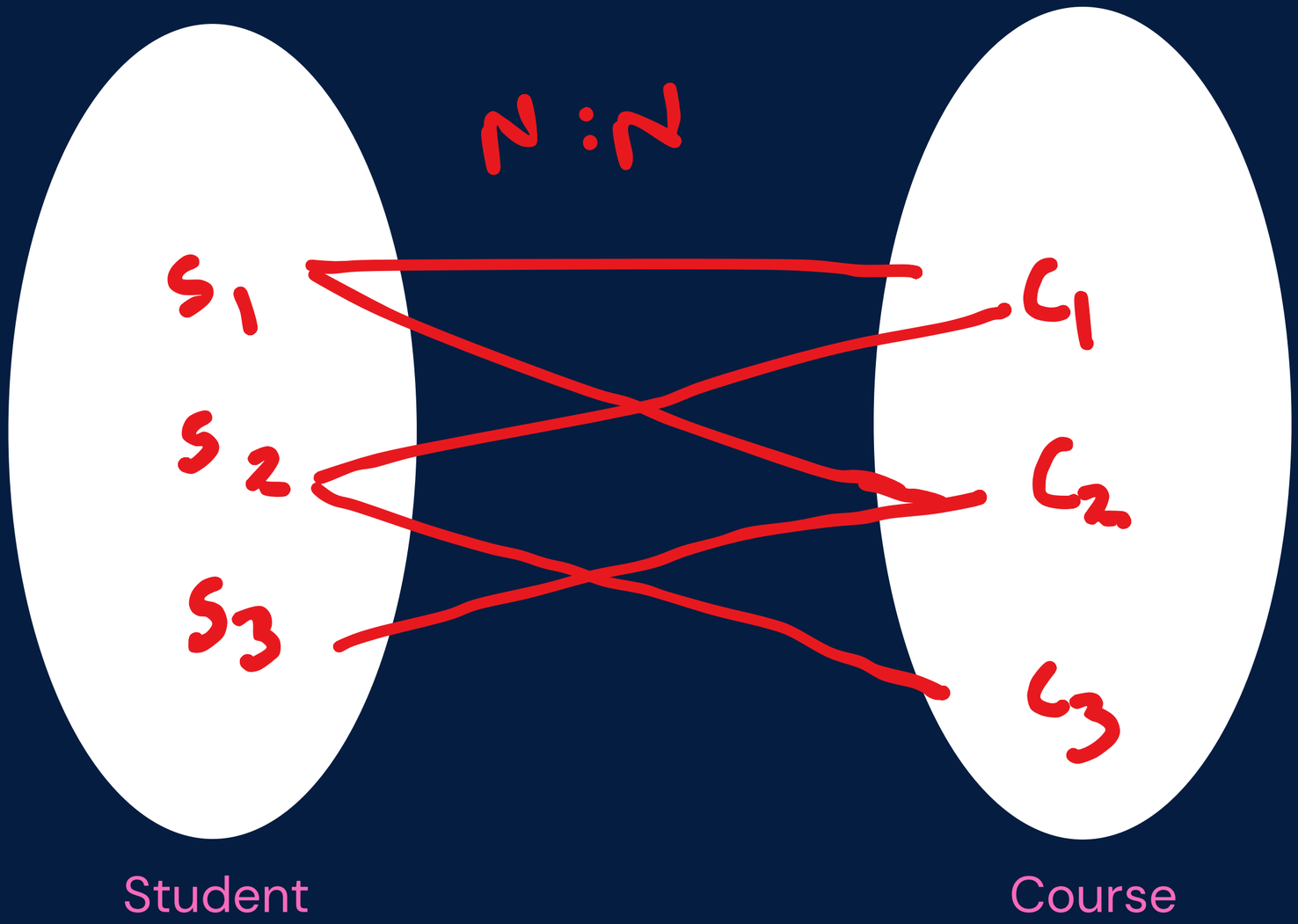
ER MODEL IN DBMS



ER MODEL IN DBMS



ER MODEL IN DBMS



ER MODEL IN DBMS

Participation Constraints

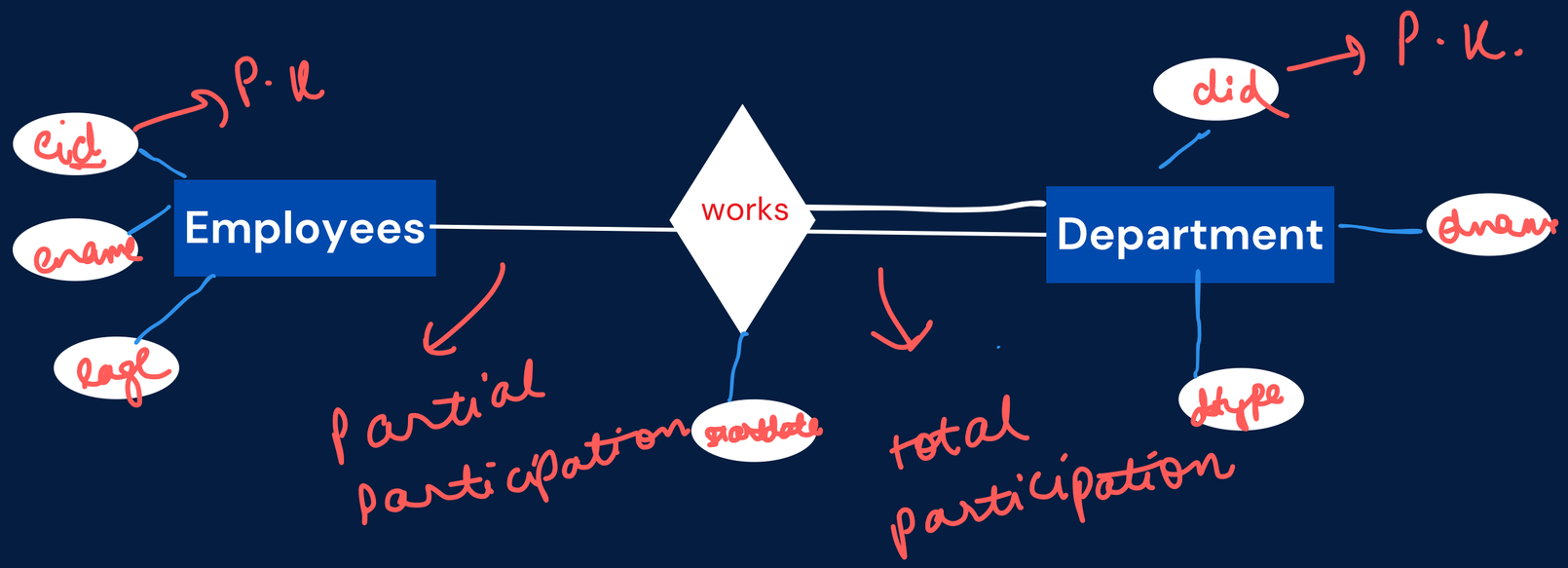
Participation Constraints in an ER model define whether every entity in one group must be connected with at least one entity in another group or if the connection is optional.

ER MODEL IN DBMS

- ## Types of Participation Constraints

Total Participation(Mandatory)

In a total participation constraint, each entity in a participation set must be associated with at least one entity in the related entity set.

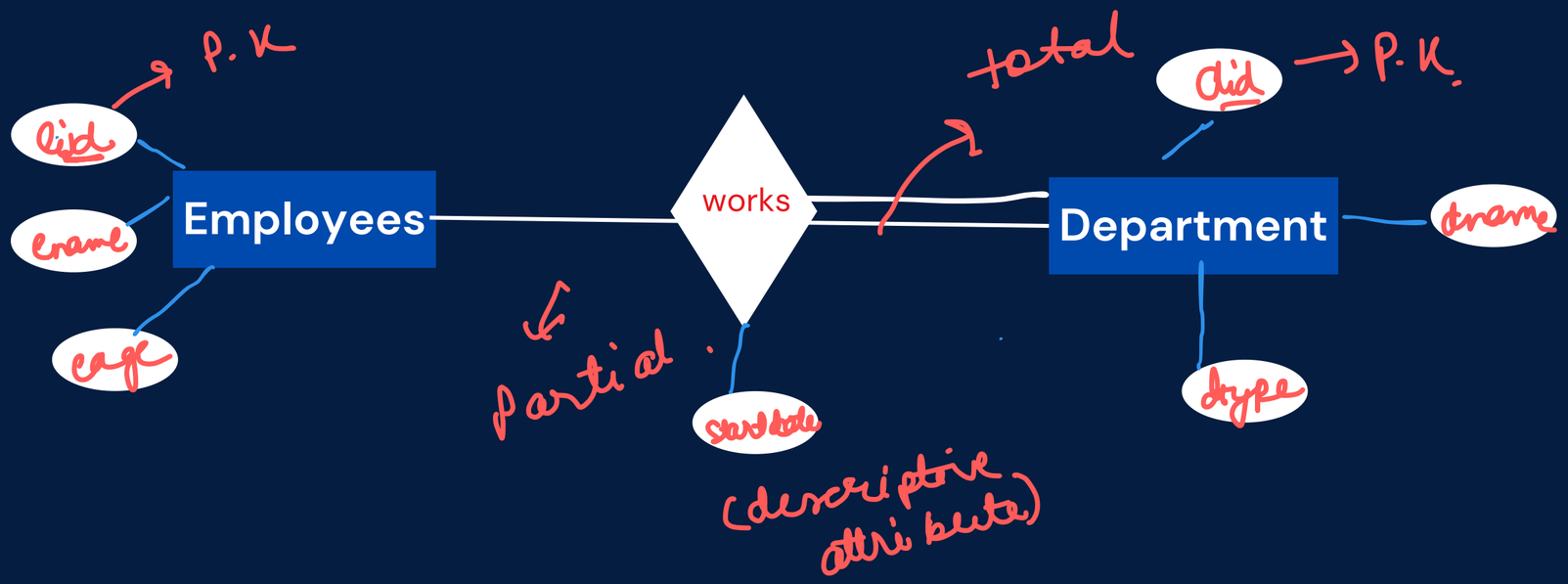


ER MODEL IN DBMS

- ## Types of Participation Constraints

Partial Participation(Optional)

In a partial participation constraint, entities in the participating entity set may or may not be associated with entities in the related entity set.



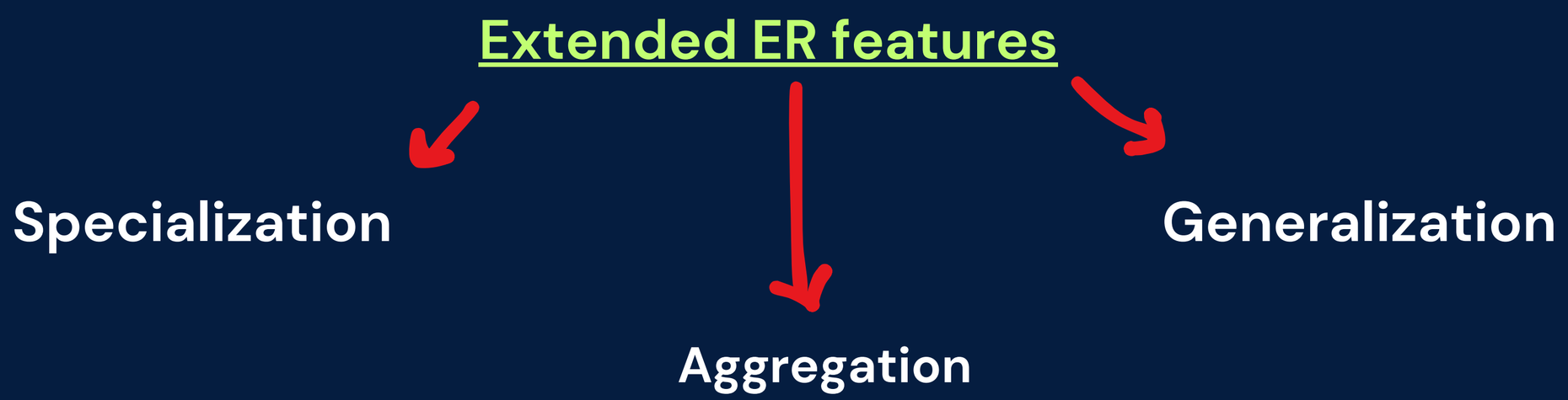
ER MODEL IN DBMS

Extended ER features

Why do we need?

We design ER model for relationship betwn entities
In real-world the data may exhibit some hierarchical relationships, and the EER model provides mechanisms to represent these relationships accurately which helps in code reusabilitilty, ensuring data integrity and consistency and lower the complexity.

ER MODEL IN DBMS



ER MODEL IN DBMS

Extended ER features

Specialization

Specialization in the ER model is like categorizing entities based on common features.

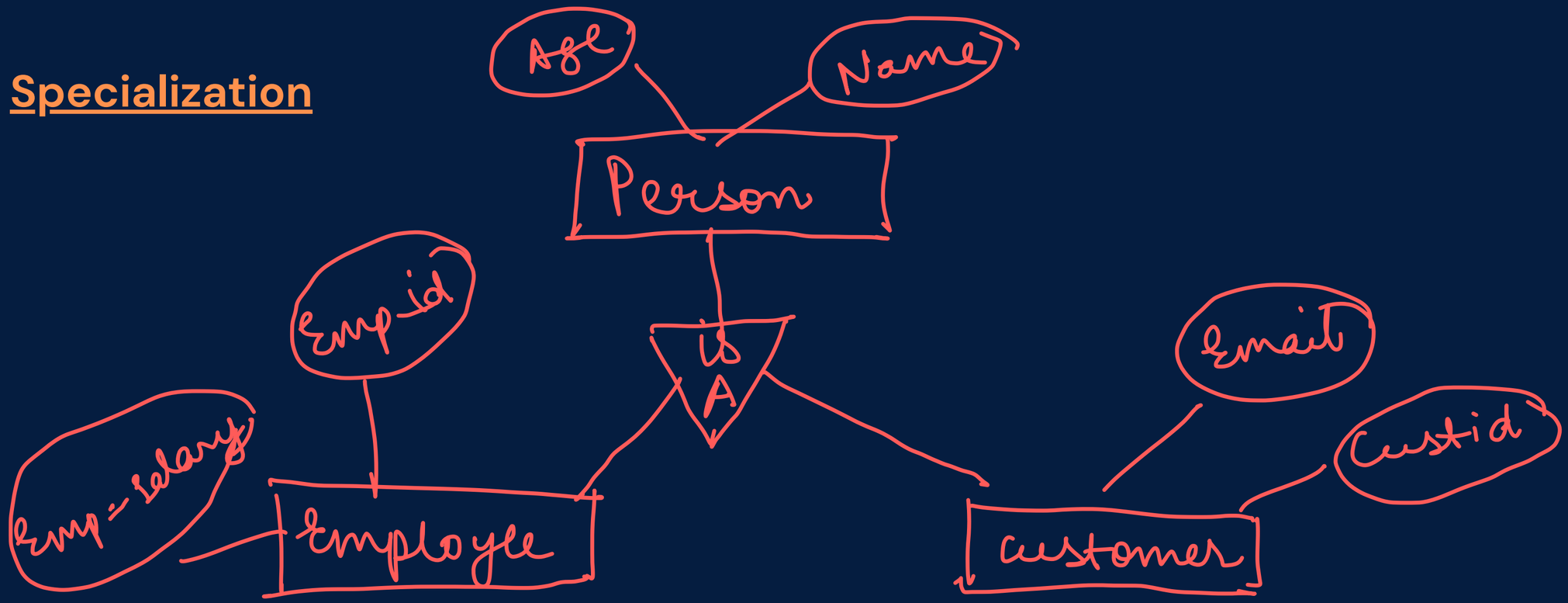
A "Supertype" groups entities with shared attributes and relationships, while "Subtypes" have their own unique attributes and relationships. It's a way to organize data efficiently. It is a **Top-Down approach.**

We have **is-a** relationship between superclass and subclass.

ER MODEL IN DBMS

Extended ER features

Specialization



ER MODEL IN DBMS

Extended ER features

Generalization

Generalization is like finding things that are alike and putting them into a big group to represent what they have in common. It helps make things simpler and organized.

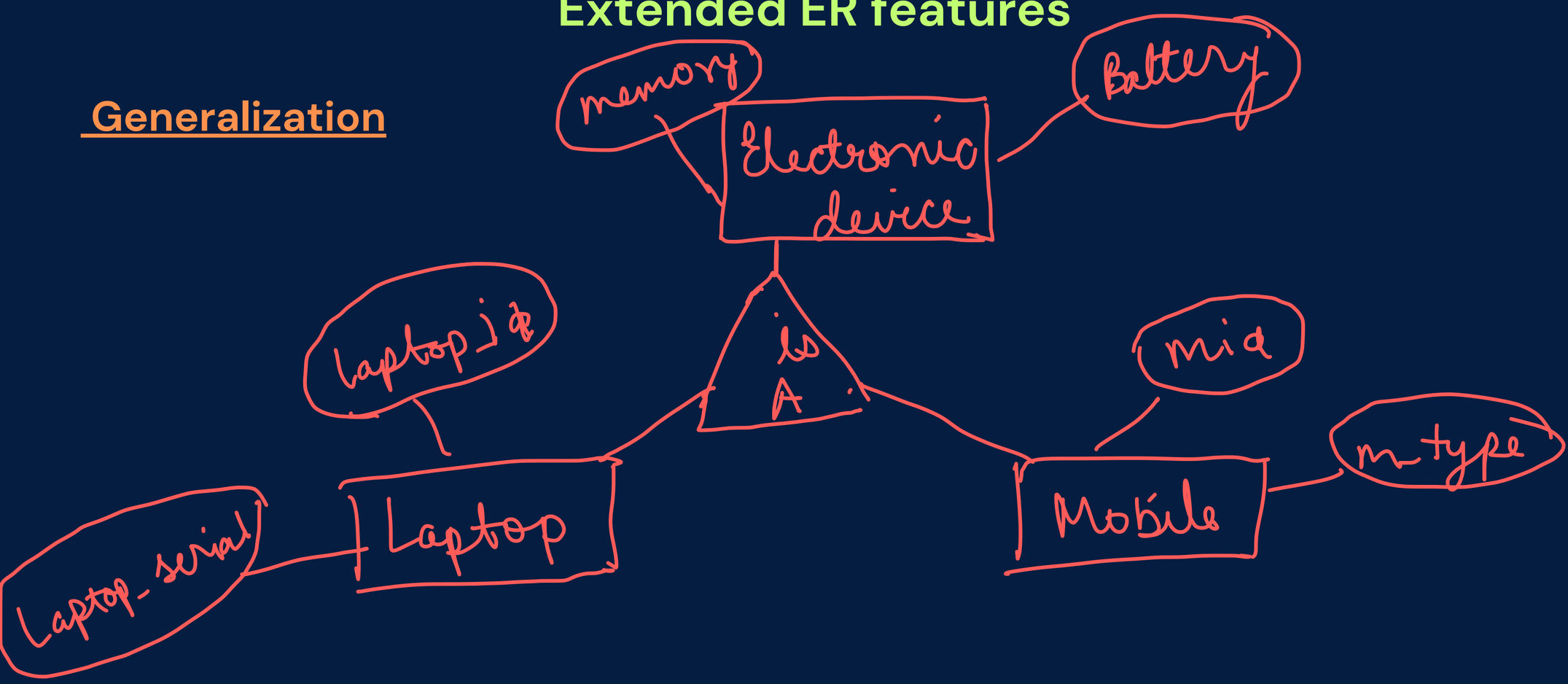
It is a **Bottom-Up approach.**

We have **is-a** relationship between subclass and superclass.

ER MODEL IN DBMS

Extended ER features

Generalization



ER MODEL IN DBMS

Extended ER features

Inheritance

Attribute

Both specialization + generalization exhibit attribute inheritance

Participation

If parent → participate → relation

child → participate

ER MODEL IN DBMS

Extended ER features

Specialization

Specialization in the ER model is like categorizing entities based on common features.

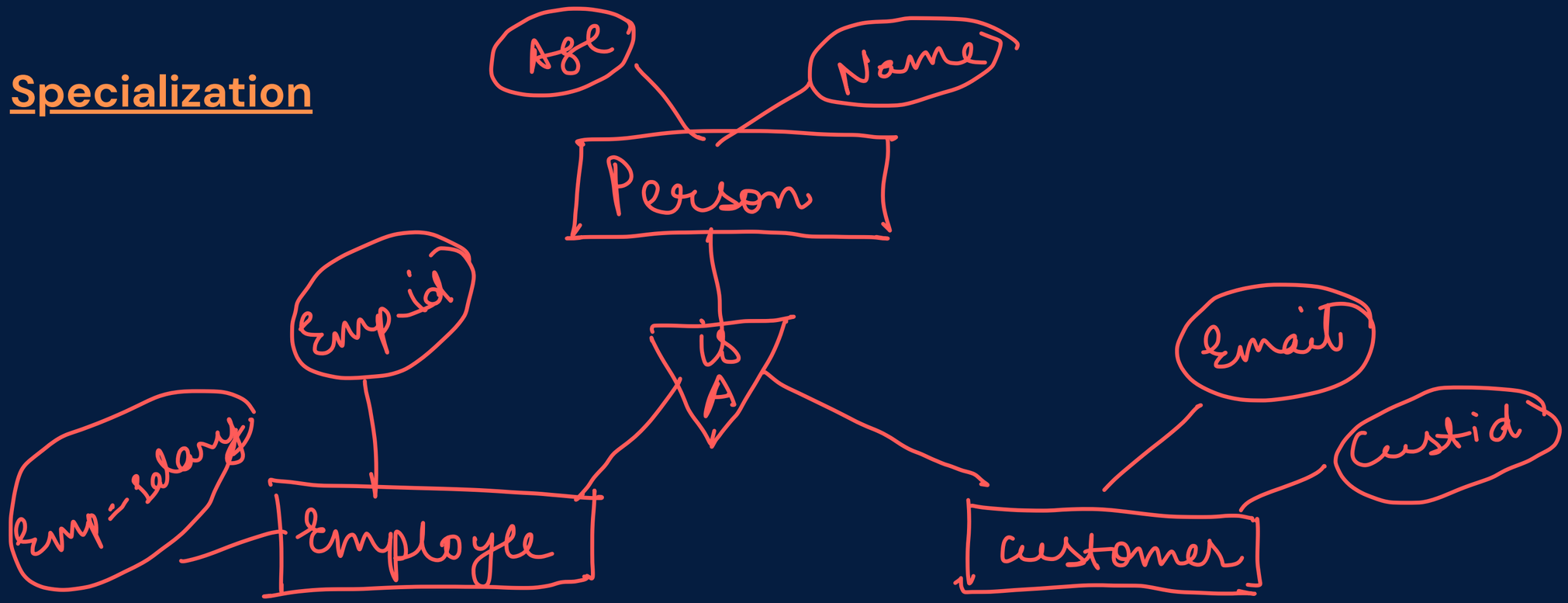
A "Supertype" groups entities with shared attributes and relationships, while "Subtypes" have their own unique attributes and relationships. It's a way to organize data efficiently. It is a **Top-Down approach.**

We have **is-a** relationship between superclass and subclass.

ER MODEL IN DBMS

Extended ER features

Specialization



ER MODEL IN DBMS

Extended ER features

Generalization

Generalization is like finding things that are alike and putting them into a big group to represent what they have in common. It helps make things simpler and organized.

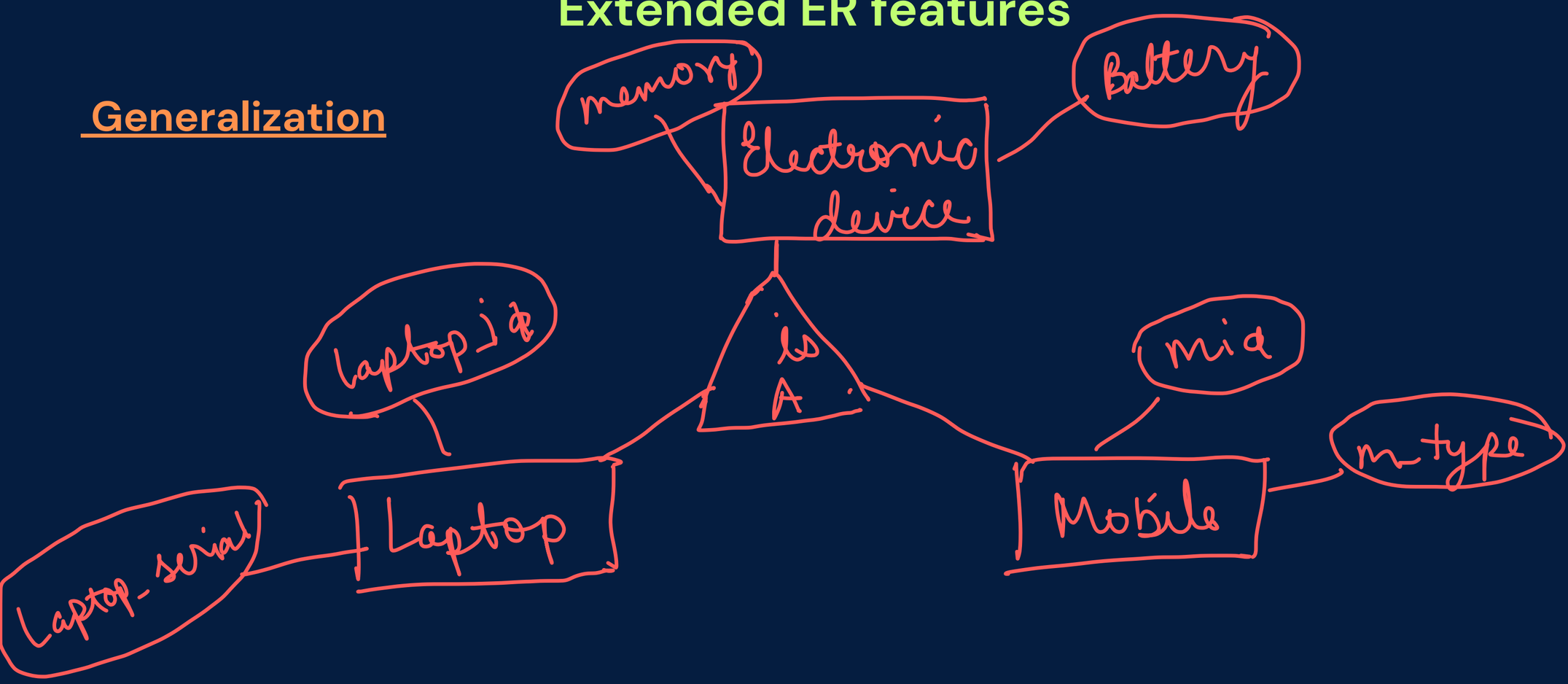
It is a **Bottom-Up approach.**

We have **is-a** relationship between subclass and superclass.

ER MODEL IN DBMS

Extended ER features

Generalization



ER MODEL IN DBMS

Extended ER features

Inheritance

Attribute

Both specialization + generalization exhibit attribute inheritance

Participation

If parent → participate → relation

child → participate

ER MODEL IN DBMS

Extended ER features

Aggregation

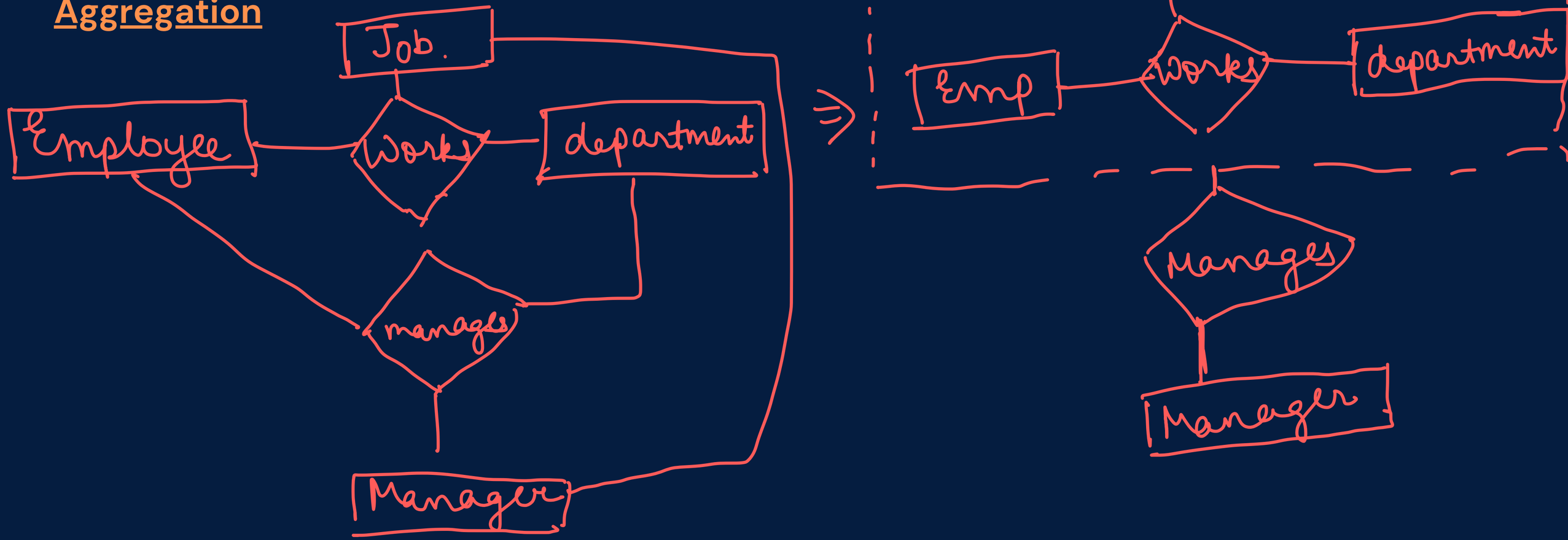
Aggregation is like stacking things on top of each other to create a structure.
It is used to create a hierarchical structure in data modeling, showing how a higher-level entity is composed of lower-level entities.

Abstraction is employed to view relationships from a more general perspective, focusing on a higher-level entity.

ER MODEL IN DBMS

Extended ER features

Aggregation



ER MODEL IN DBMS

Steps to draw an ER model

1. Recognize entities.
2. Specify entity characteristics/attributes.
3. Discover connections/relationships(also contraints like mapping/participation)
4. Define the connection type (how entities connect)/cardinality.
5. Construct an ERD (Entity-Relationship Diagram).
6. Annotate relationships and attributes.
7. Review and refine the model.
8. Document the model.
9. Validate with stakeholders.
10. Implement the database schema.

ER MODEL IN DBMS

ER Model of Instagram

Lets start with what is instagram?

Instagram is a social media platform that allows users to share photos and videos.

ER MODEL IN DBMS

ER Model of Instagram

Now what all things we can do on instagram?

- Create our profile
- Add profile picture and details
- Connect with friends
- Upload a post
- Like and comment on post
- Share stories

  and much more

ER MODEL IN DBMS

ER Model of Instagram

Lets start with all the steps needs to draw an ER diagram.

Step-1 : Recognize entities sets

Entities

- userProfile
- userFriends
- userPost
- userLogin
- userLikes

ER MODEL IN DBMS

ER Model of Instagram

Step-2 : Specify entity characteristics/attributes

Attributes

1. userProfile (user ID, username, email, profile pic)

    user ID- primary key
    userName- composite attribute
    email - single valued attribute
    profile pic - single valued attribute
    dob- stored attribute
    age- derived attribute

ER MODEL IN DBMS

ER Model of Instagram

Step-2 : Specify entity characteristics/attributes

Attributes

2. userFriends (followerID, followerName, userID)

followerID- primary key
followerName - single valued attribute
userID - single valued attribute

ER MODEL IN DBMS

ER Model of Instagram

Step-2 : Specify entity characteristics/attributes

Attributes

3. userPost (post ID, caption, image, video, likesCount, timestamp)

post ID- primary key
caption - single valued attribute
image - multi valued attribute
video - multi valued attribute
likesCount - single valued attribute
timestamp - single valued attribute

ER MODEL IN DBMS

ER Model of Instagram

Step-2 : Specify entity characteristics/attributes

Attributes

4. userLogin (login ID,loginUserName,loginPassword)

login ID- primary key
loginUserName - single valued attribute
loginPassword - multi valued attribute

ER MODEL IN DBMS

ER Model of Instagram

Step-2 : Specify entity characteristics/attributes

Attributes

4. userLikes (postID, userID)

postID- primary key
userID - single valued attribute

ER MODEL IN DBMS

ER Model of Instagram

Step-2 : Discover connections/relationships(also contraints like mapping/participation)

1.userProfile have userFriends (n:n)

2. userProfile have userPost (1:n) userPost will always be associated to a userProfile therefore total participation

3. userProfile has userLogin (1:1)

4. userProfile has userLikes (1:n) userLikes will always be associated to a userProfile therefore total participation

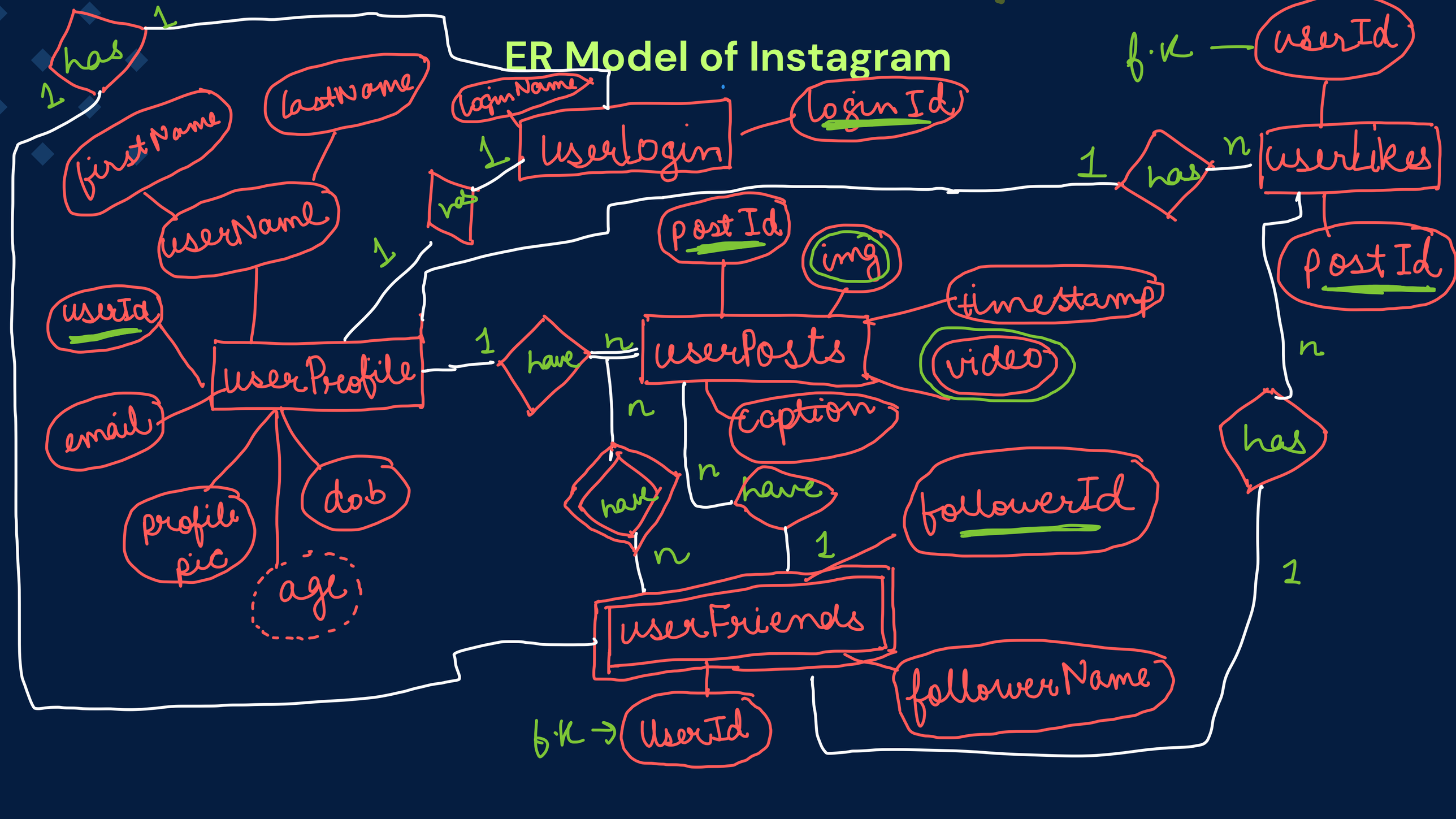
ER MODEL IN DBMS

ER Model of Instagram

Step-2 : Discover connections/relationships(also contraints like mapping/participation)

5. userFriends have userPost (1:n) userPost will always be associated to a userProfile therefore total participation
6. userFriends has userLogin (1:1)
7. userFriends has userLikes (1:n) userLikes will always be associated to a userProfile therefore total participation

ER Model of Instagram



RELATIONAL MODEL

It is a way of organizing data in tables.

Some terms used in relational model

1. **Table** - Relation
2. **Row** - Tuple
3. **Column** - Attribute
4. **Record** - Each row in a table
5. **Domain** - The type of value an attribute can hold
6. **Degree** - No. of columns in a relation
7. **Cardinality** - No of tuples

RELATIONAL MODEL

Relational model is all about:

- **Data being organized into tables**
- **Establishing Relationships between tables using Foreign key**
- **Maintaining data Integrity**
- **A flexible and efficient way to store(SQL) and retrieve data**

RELATIONAL MODEL

In relational model we take care of different things like:

1. Maintaining integrity constraints like domain, entity, referential integrity.

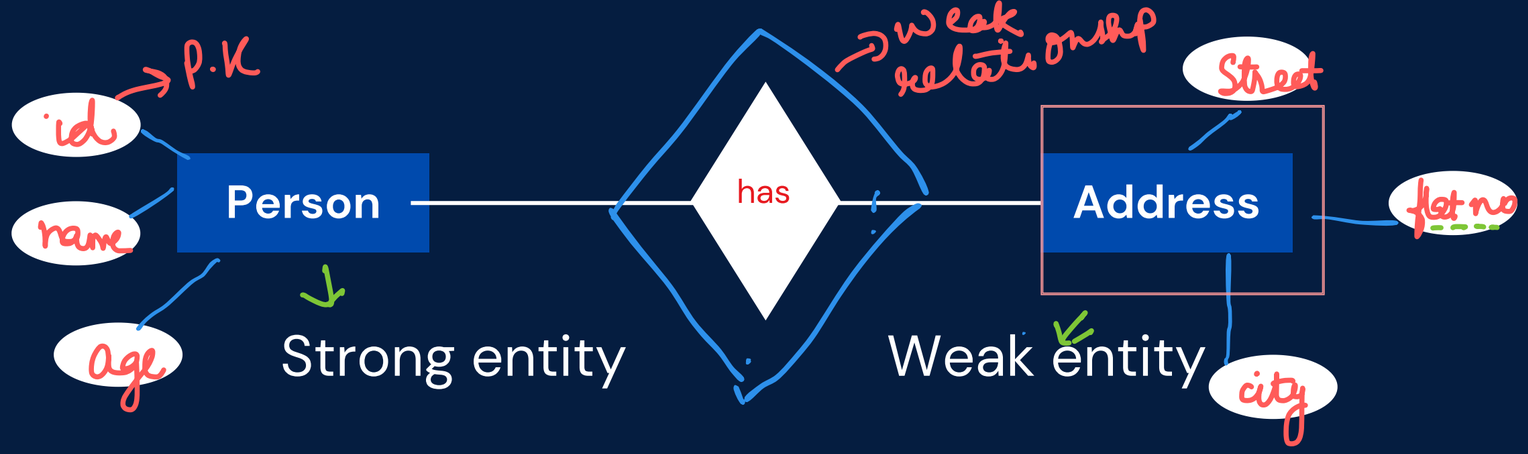
2. The values to be atomic i.e can’t be divided further.

3. Each row must be unique, here keys comes into picture i.e candiate, super, primary etc

CONVERT AN ER MODEL TO RELATIONAL MODEL

Converting an Entity-Relationship (ER) model to a relational model involves several steps:

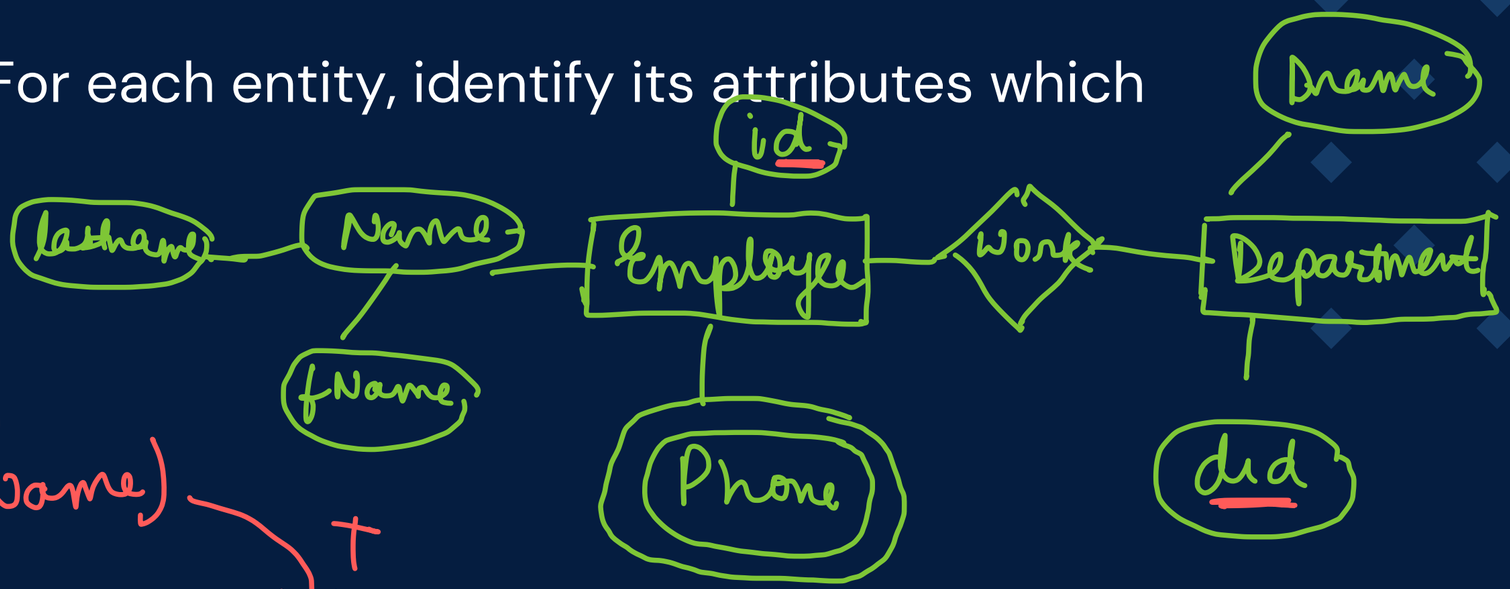
Step 1: **Identify the entities** - List down all the entities like strong and weak.



1. Person (id, name , age) -> id (p.k)
2. Address (id , flatno, street, city)->id+flatno (p.k) , id (f.k)

CONVERT AN ER MODEL TO RELATIONAL MODEL

Step 2: **Identify the attributes** – For each entity, identify its attributes which becomes a column in the table.



Multivalued attribute

1. Employee → (id(P.K), EfName, ElastName)

Composite attribute

1. Employee → (id(P.K), PhoneNbrs)

new table

CONVERT AN ER MODEL TO RELATIONAL MODEL

Step 3: **Key selection** - Choose the primary key for each table, for some it can be in form of composite key (Weak entity)

Step 4: **If entities have relationship break it down and the reduce the tables if possible.**

1. 1-1 Relationship : 2 tables , P.K can lie on any side
2. 1-Many Relationship : 2 tables , P.K can lie on many side
3. Many -1 relationship : 2 tables , P.K can lie on many side
4. Many-Many relationship : 3 tables , P.K lie in the relation table having pk from both the table acting as fk

CONVERT AN ER MODEL TO RELATIONAL MODEL

Step 3: Key selection – Choose the primary key for each table, for some it can be in form of composite key (Weak entity)