Data Base Management System.

Data :

Data refers to a collection of raw facts or figures that can be processed to derive meaning or knowledge.
eg: XYZ, 12.

Information :

Processed data is called information.

Ex: Your name, Temperature, etc.

Database :

Collection of interrelated data is called a database

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

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.

major disadvantage

- Data redundancy
- Poor memory utilisation
- Data inconsistency
- Data security.

Database Management System (DBMS)

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.

Types of Databases.

1. Relational Databases (RDBMS)
2. NoSQL Databases
3. Object - Oriented Databases
4. Time-series Datases.
5. In Memory Databases
6. Spatial Databases
7. Multimedia Databases
8. Columnar Databases.

9. XML Databases

10. NewSQL Databases

11. Blockchain Databases.

Relational Databases (RDBMS):

These databases structure data into organised 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.

Other Query Language :- Cypher – for querying graph databases like Neo4j, GraphQL -- For API's and Graph based data.

NoSQL Databases : (key-value pairs)

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

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.

EX :- Object DB

Object = stores both data and behavior (methods)

Class = blueprint for objects (like table schema)

Attributes = fields of the object (like columns)

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.

Advantages of DBMS.

=> Data security
=> Data Redundancy and Inconsistency
=> Data Integrity.
=> Data scalability
=> Data Abstraction.

Disadvantages of DBMS.

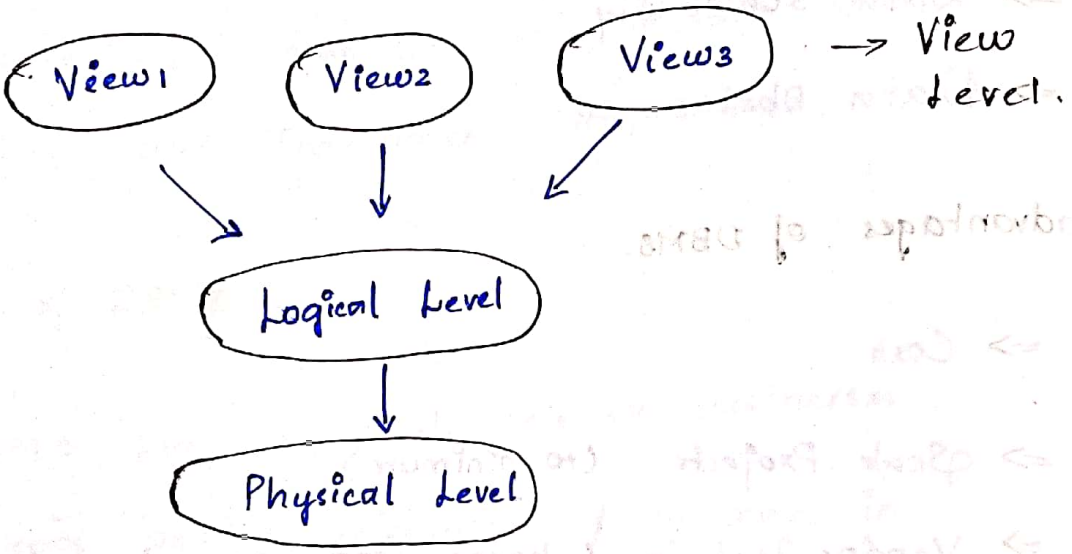
=> Cost
=> Scale Projects. (to minimum)
=> Vendor lock_in (change from one format to another format)

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 dont need to worry about. This hiding of unnecessary things from users is called data abstraction.

~~There~~

There are three levels of Abstraction.



Types of Levels.

Physical level. (hashmap. b trees etc).

This is the lowest level of data abstraction it describes how data is actuatually stored in database. You can get the complex data structu at this level.

Logical level. (relation between data) (conceptual level)

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

View Level (External level)

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

Attributes → Columns.

Tuples → Rows.

DBMS Architecture.

DBMS architecture defines how a database system is structured.

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

Types of DBMS Architecture.

There are several types of DBMS Architecture.

Choice of architecture depends on factors such as the type of database (eg. relational, NoSQL) and the specific need of an application.

- 1-Tier Architecture
- 2-Tier Architecture.
- 3-Tier Architecture.

1- Tier Architecture.

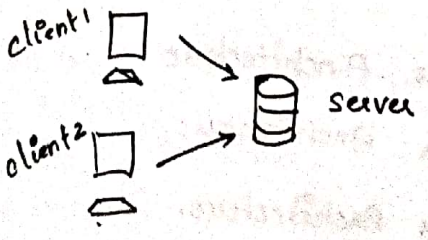
In 1 Tier Architecture the entire database application, including the user interface, application logic and data storage resides on a single machine or computer.

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.

2- Tier Architecture.

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

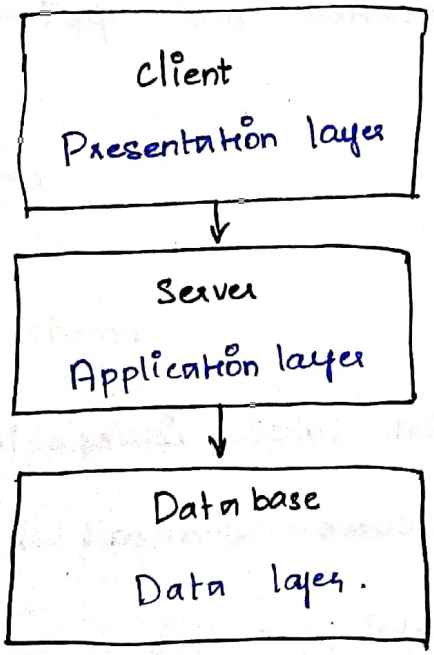
Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.



3. Hex 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.



Advantages.

- 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.

Disadvantages.

Complexity, potential latency issues, longer development time, resource overhead, and the possibility of bottlenecks.

Schema.

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

It defines how data is organised 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 retrival.

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.

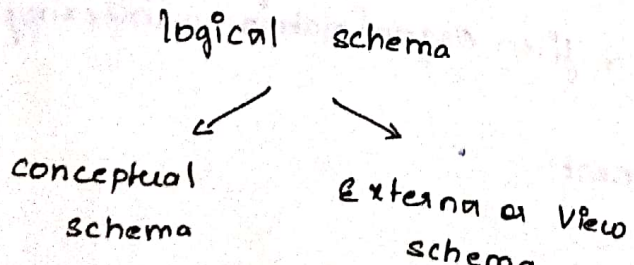
Characteristics of physical schema.

- 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 retrival.

Designing the physical schema involves , selecting on which column the indexing need to be done .

2. Logical schema.

A logical schema defines the database's structure from a logical or conceptual prespective without considering how the data is physically stored.



Logical schema is the **structure of the database** at a high level.

It defines:

* **Tables**
* **Columns**
* **Data types**
* **Relationships** (like foreign keys)

It's **independent of physical storage** and focuses on **what data is stored and how it's related**.

Types

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.

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.

Instance

The Information residing within a database at a specific point in time is refered 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.

Data Model.

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

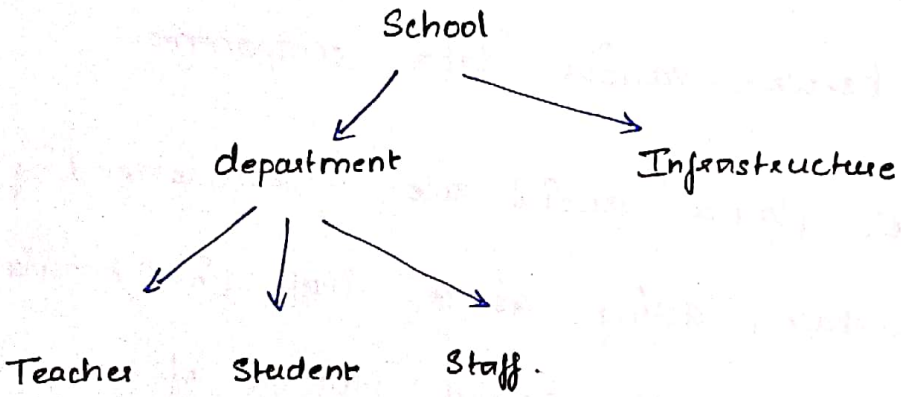
Data model.	schema
conceptual framework	Implementation of
high level view of DB.	Data model

Types of Data Model.

- Hierarchical Data model.
- Network Data model.
- Relational Data model.
- Entity Relationship Model (ER model)
- Object-Oriented Data Model
- No SQL Data models.

Hierarchical Data Model.

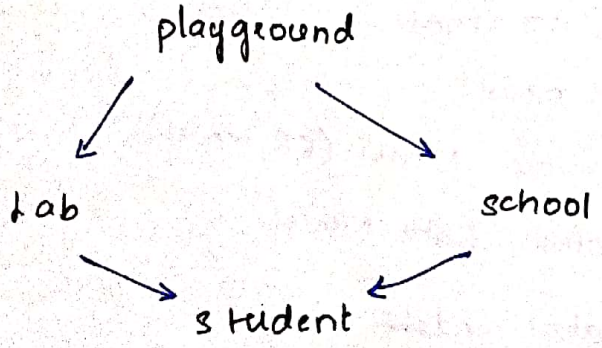
This model portrays data in a manner resembling a tree structure, where each record maintains a parent-child relationship. Its primary application lies in older database systems.



here every child had only one parent.

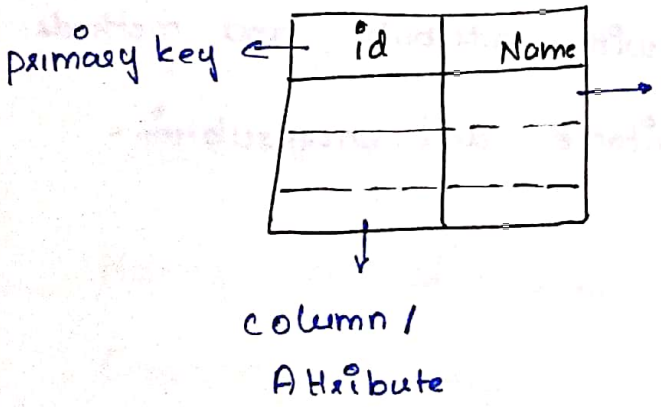
Network Data model.

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.



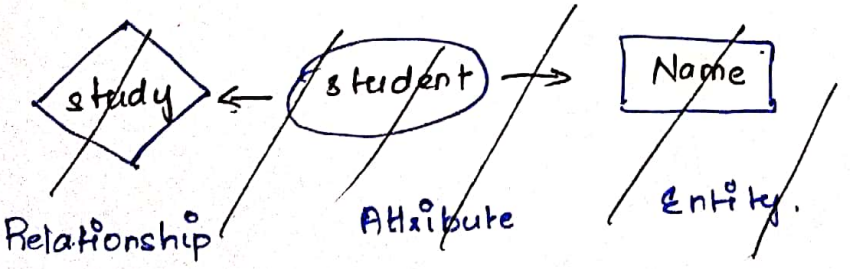
Relational Data models.

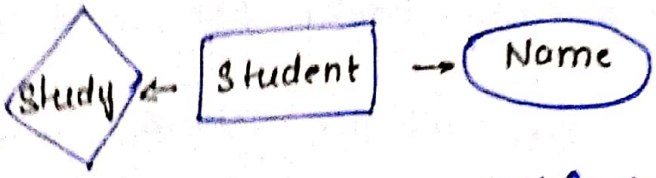
Organizing data into tables (known as relations) consisting of rows and columns characterizes the relational model. It stands as the most prevalent data model, rooted in the principles of set theory and relies on Structured Query Language (SQL) for data manipulation.



Entity - Relationship Model (ER Model):

Utilized for crafting relational database, the ER model represents data through entities (objects) attribute (entity properties) and relationships connecting these entities.





Relation - ship Entity Attribute

Object Oriented Data model.

Extending the principles of object-oriented programming into the database domain, this model depicts data as objects complete with attribute and methods fostering support for inheritance and encapsulation.

NoSQL Data models.

NoSQL database encompass a diverse array of data models, such as document - oriented (eg. mongo-DB), key-value (eg: Redis), column-family (eg: Cassandra) and graph (eg: Neo4j). These models are designed to offer scalability and flexibility when handling extensive volumes of unstructured or semi-structured data.

Introduction to ER model.

Entity. Things / Object Ex - person.

Attributes Properties of entity Ex - name, age.

Relationship association among entities Ex - works for.

- 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.

Entity

→ Strong Entity
→ Weak Entity

Attribute

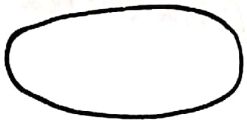
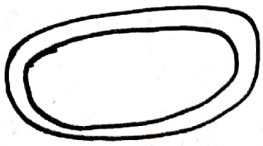
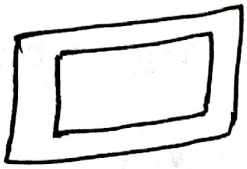
→ Simple Attribute
→ Composite Attribute
→ Single valued Attribute
→ Multivalued Attribute
→ Stored Attribute
→ Derived Attribute

Compositer Attr

Relationship

→ One to One
→ one to many
→ many to one
→ many to many.

Symbols Used in ER model.

Entity	Rectangle	
Attribute	Ellipse	
Relationship	Diamond	
Attribute to entity relationship	Line	
Multivalued attributes	Double ellipse	
Weak Entity.	Double rectangle	

ER model in DBMS.

Entity. represented by 

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

Types of Entity.

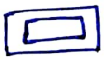
=> Strong Entity → person Entity

=> Weak Entity. → dependent 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.

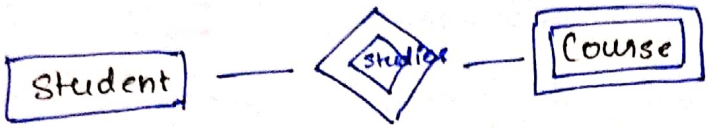
Ex- person.

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.

Example.



Relationship in ER model.  diamond.

It is the connection between entities (tables) based on related data.

Types of Relationship.

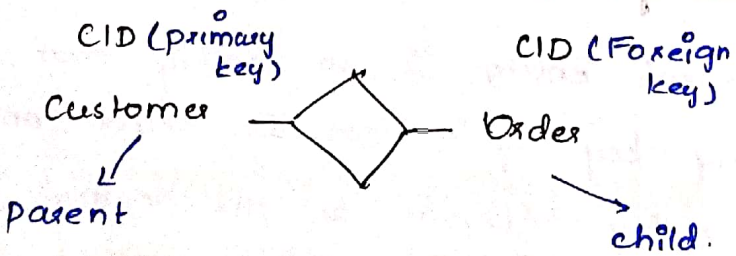
=> Strong Relationship

=> Weak Relationship.

Strong Relationship.

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

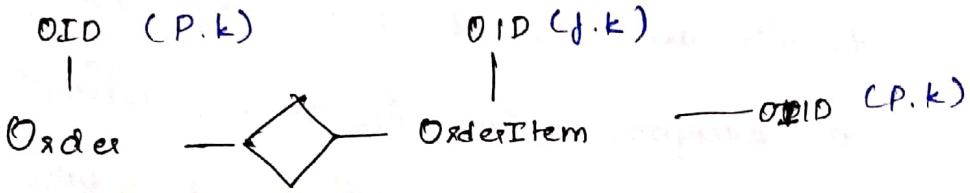
Ex:



Weak Relationship.

On the other hand, exists when two entities are related, but one entity can exist without the other.

Eg:



Degrees.

The degree in DBMS refers to the number of attribute / columns that a relation / table has.

Types of degree.

1. Unary degree.

   A relation with a single attribute.

2. Binary degree.

   A relation with two attribute.

3. Ternary degree.

   A relation with three attribute.

4. n-ary degree.

   A relation with more than ~~one~~ n attribute.

   ($n \geq 3$)

Null value

In database a null value can occur for various reasons.

Not needed Informations : 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.

Dont know the Answer:

Every now and then, were asked a question, but we don't have an answer yet.

Forget to Fill in:

Like when you're filling out a form, and you accidentally miss putting in some important information.

Types of Attributes.

Simple Attribute.

A simple Attribute is atomic and cannot be divided any further

Eg- First-Name.

Composite Attribute.

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

Ex- Name (First Name, Last Name)

Single Valued Attribute

It is an attribute that holds a single value for each entity.

Ex - age.

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).

Stored Attribute.

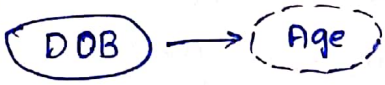
Attribute that is stored as a part of a database record.

Ex - Date of Birth.

Derived Attribute.

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

Ex- Age derived from DOB.



Complex attribute.

It is an attribute that is made up of multiple smaller attribute.

Ex: Name (Composite) → FirstName, last Name, Middle Name (Simple attribute)

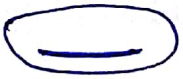
key Attribute.

The Attribute which can be uniquely Identified.

Ex- s.id.

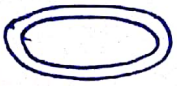
simple attribute

key



derived

Multivalued.



(Based on Degree)

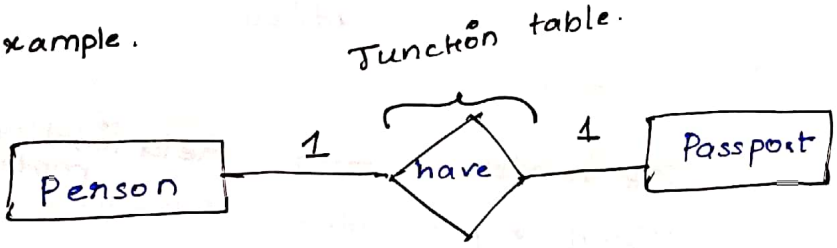
There are 4 types of relationship:

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

1 to 1 relationship :

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

Example.



one person has only one passport

One passport belongs to only one person

Junction Table.

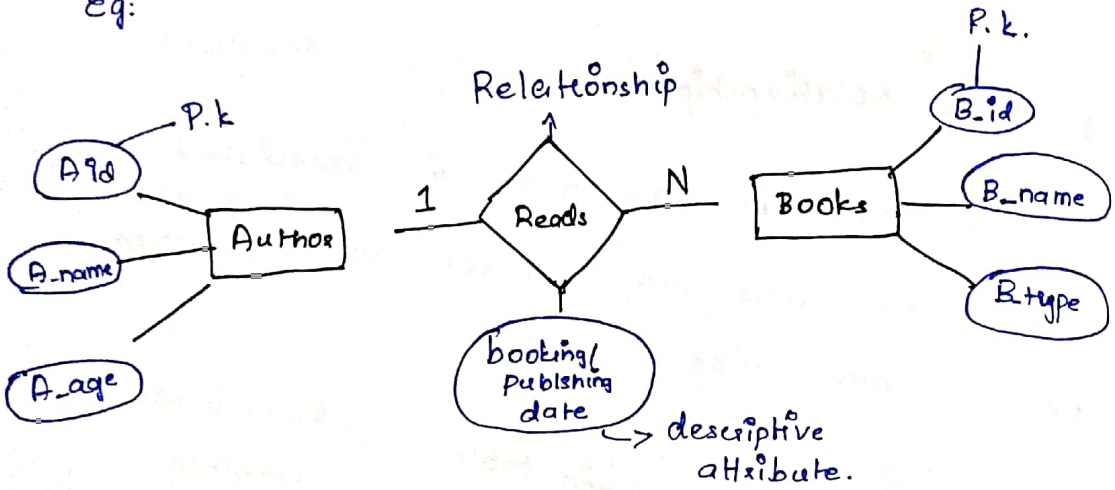
A Junction Table (also known as Association Table or Linking Table) is used to model relationships between two entities.

1 to 1 can be reduced to two table considering any one as P.k.

1 to many Relationship

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

Eg:



Schema → ER model → Relational model

Visual representation

Relational model.

It is a method of structuring data using tables (relations). Each table consists of rows and columns.

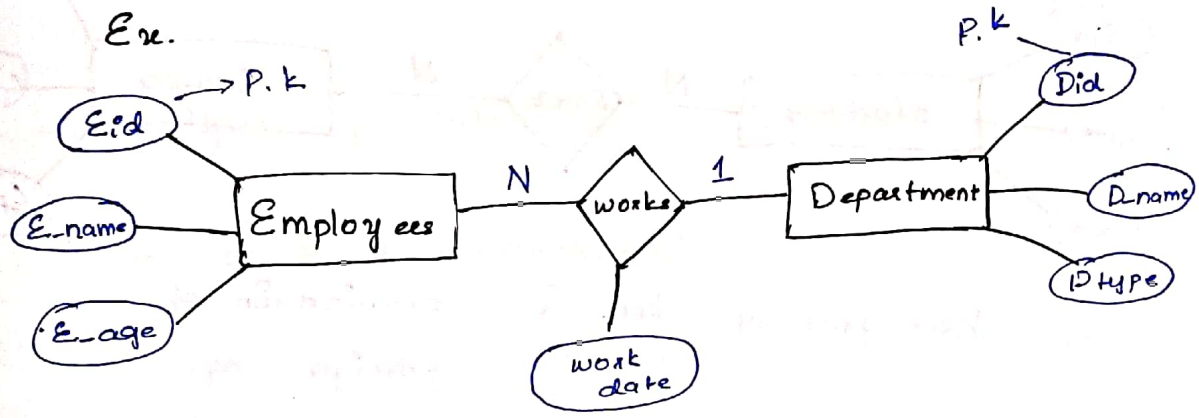
(tuples) (attributes)

In Relationship table the primary key always lies on many side.

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.

Ex.



Both 1 to many and many to 1 can be reduced where P.k in the many side that is "B_id and E_id."

Cardianality.

It refers to the number of occurences of one entity relate to the number of occurences in another entity. It's about how two entity are related in terms of count.

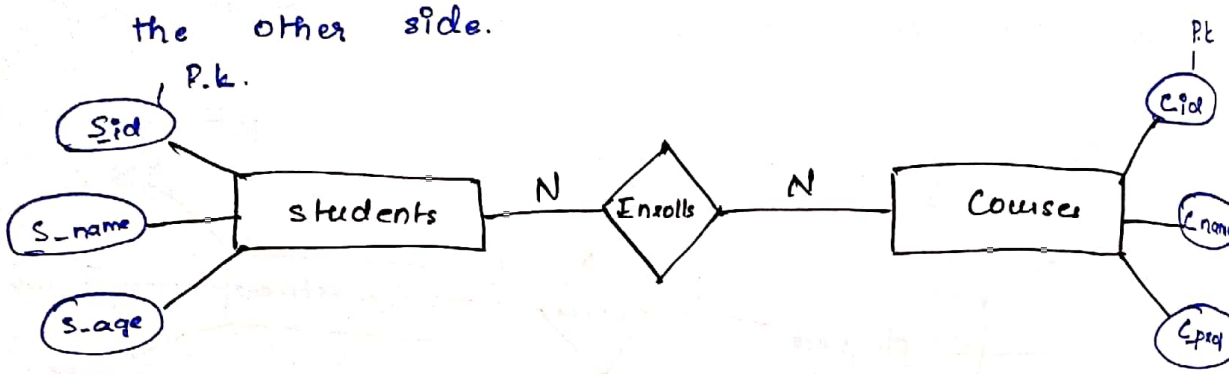
Types. => 1:1 , N:1, 1:N, N:N

Mapping Constraints.

This are rules that defines how entity sets participate in a relationship, specifically in terms of cardinality.

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.

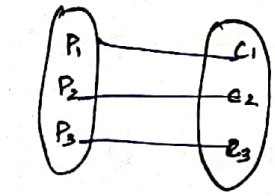


here primary key is combination of foreign key.

cannot reduce the table (No of table 3)

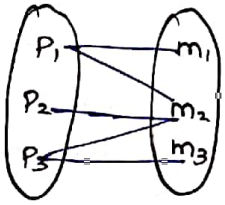
In Ven diagram.

1:1



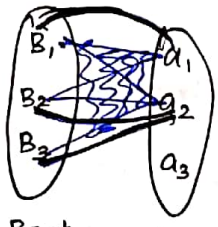
person pancard

1 : N



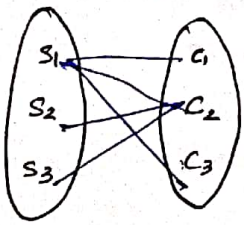
person mobile

N : 1



Book Author

N : N



Studen course

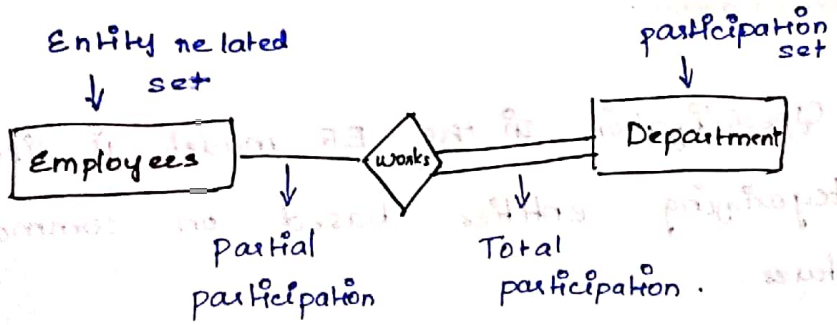
Participation Constraints.

It is Define whether every entity in one group must be connected with at least one entity in another group or if the connection is optional.

Types of participation constraints.

Total participation (Mandatory)

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



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.

Extended ER features.

Why do we need.

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

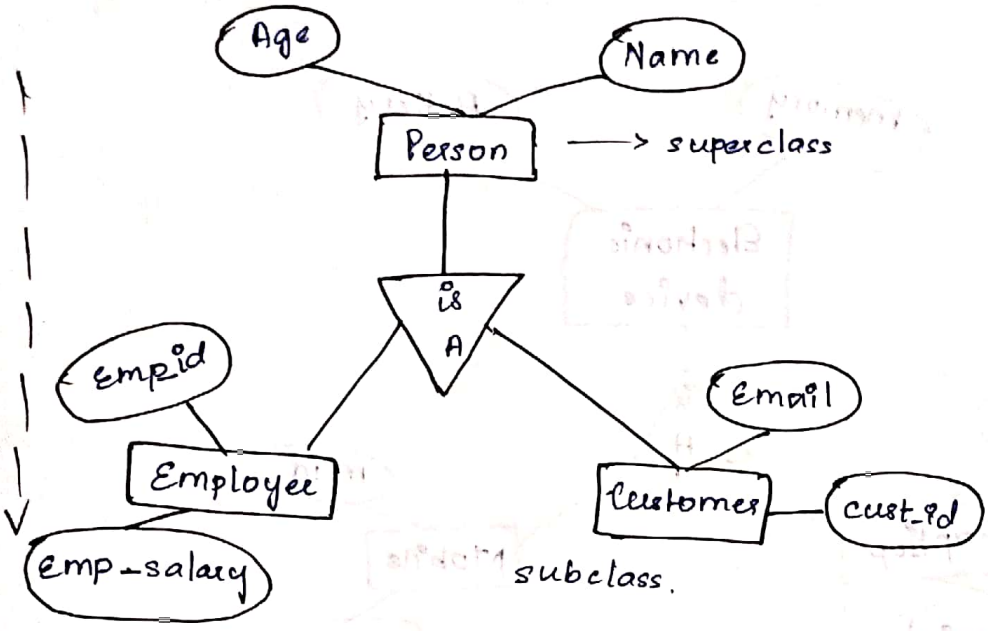
1. Specialization
2. Aggregation
3. Generalization.

Specialization.

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

A "Superetype" groups entities with shared attributes and relationships while "subtypes" have their own unique attributes and relationships. Its a way to organize data efficiently. It is a Top-Down approach.

We have is-a relationship between superclass and subclass.

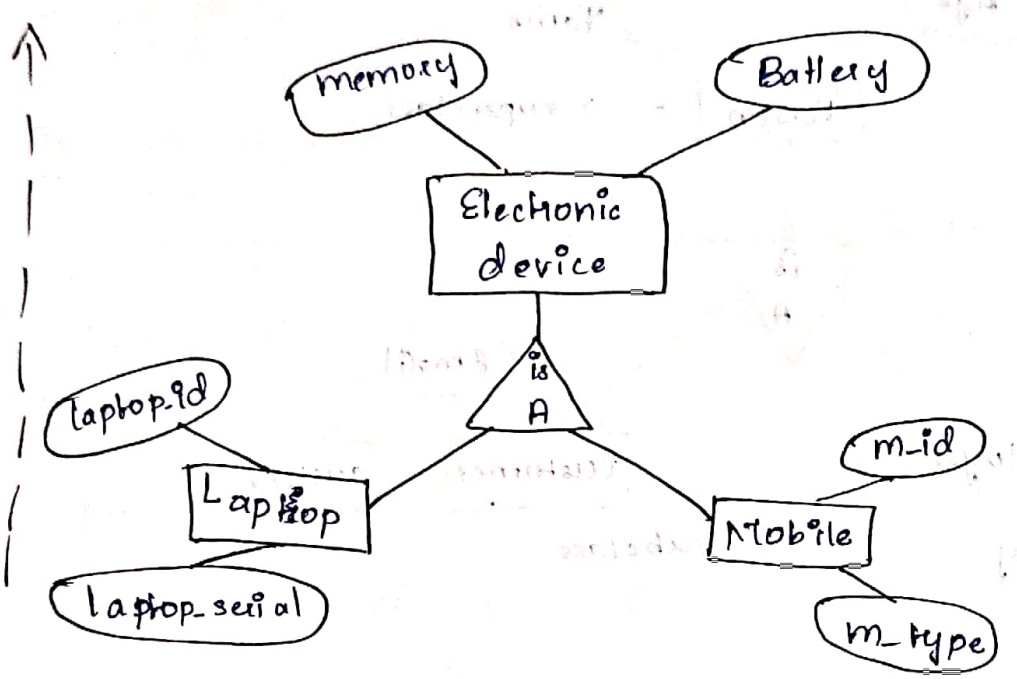


Generalization.

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.



Inheritance.

Attribute

Both specialization & Generalization exhibit attribute

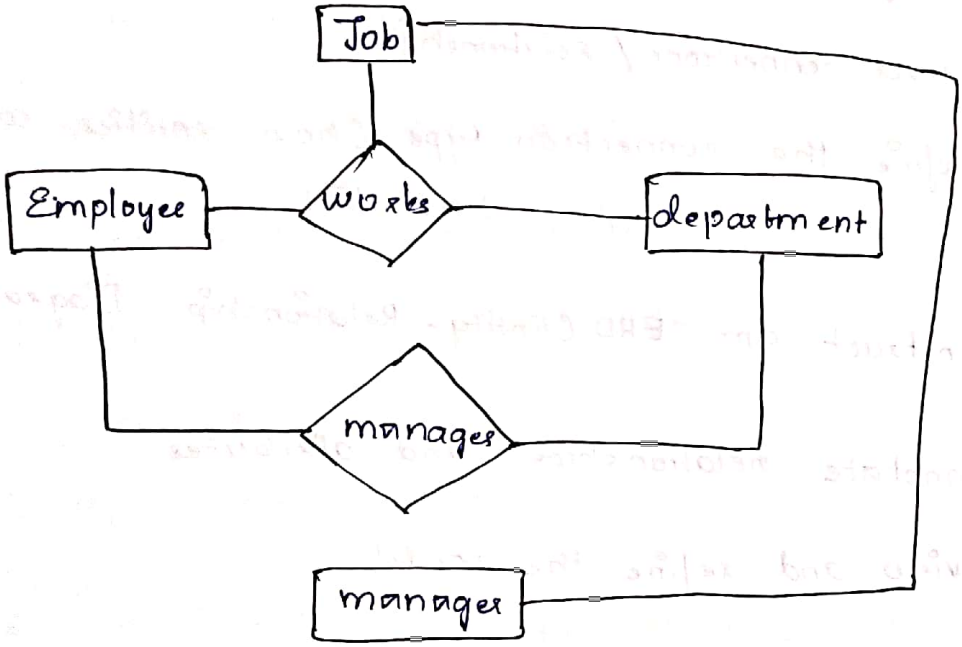
Participation.

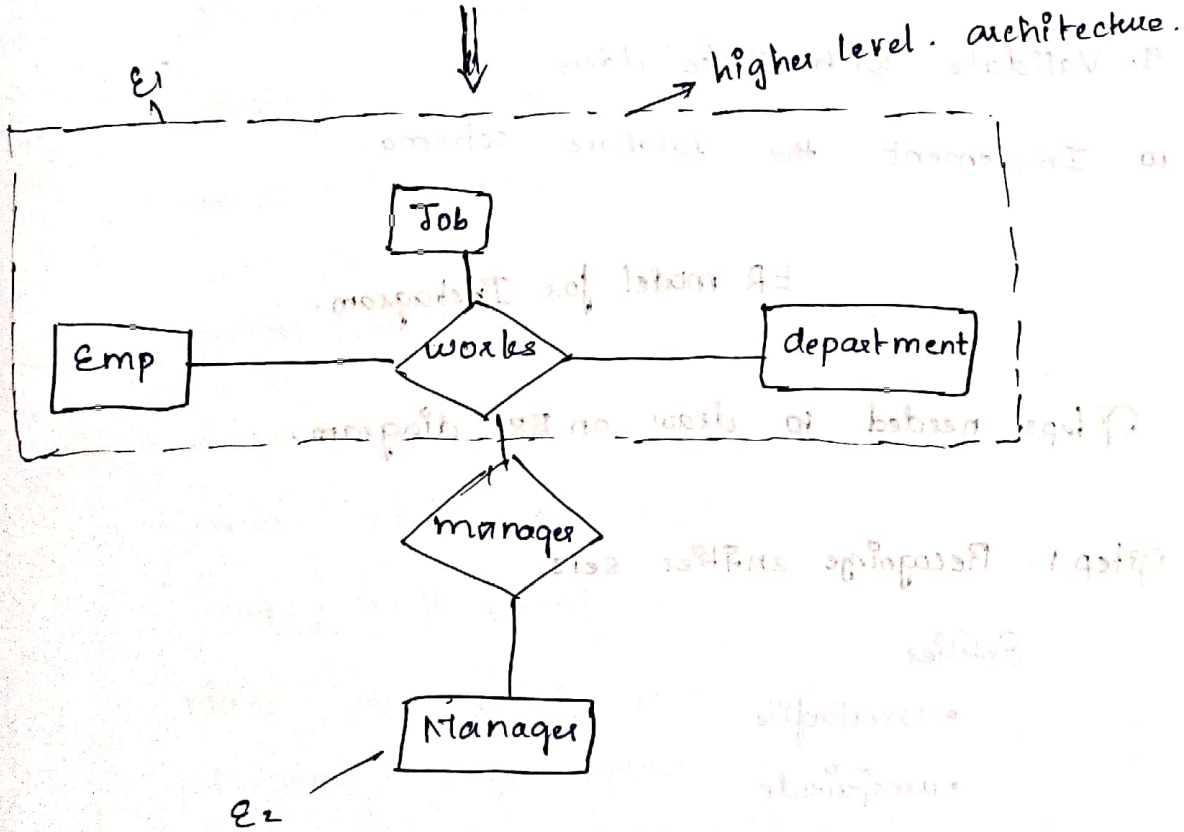
If parent → participate in a relation the child also participate in same relation.

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 prespective focusing on a higher-level entity.





E_1 — has relationship with — E_2

Steps to draw an ER model.

1. Recognize entities.
2. Specify entity characteristics / attributes.
3. Discover connections / relationships.
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 for Instagram.

Steps needed to draw an ER diagram.

Step 1: Recognize entities sets.

Entities.

- UserProfile
- userFriends
- userPost
- userLogin
- userLikes.

Step 2: Specify entity characteristics / attributes.

Attributes

1. userProfile (userID, username, email, profile pic)

   userID - primary key

   userName - composite attribute

   email - single valued attribute

   profile pic - single valued attribute

   dob - stored attribute

   age - derived attribute.

2. userFriends (followerID, followerName, userID)

   followerID - primary key

   followerName - single valued attribute

   userID - single valued attribute.

3. userPost (postID, caption, image, video, likesCount, timestamp)

   postID - primary key

   caption - single valued attribute

   image - multi valued attribute

   video - multi valued attribute

   likesCount - single valued attribute

   timestamp - single valued attribute

4 userLogin (LoginID, LoginUserName, LoginPassword)

LoginID - primary key

LoginUserName - single valued attribute.

LoginPassword - multi valued attribute.

5. userLikes (postID, userID)

postID - primary key

userID - single valued attribute.

Step 3: Discover connections / relationships (also constraints 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.

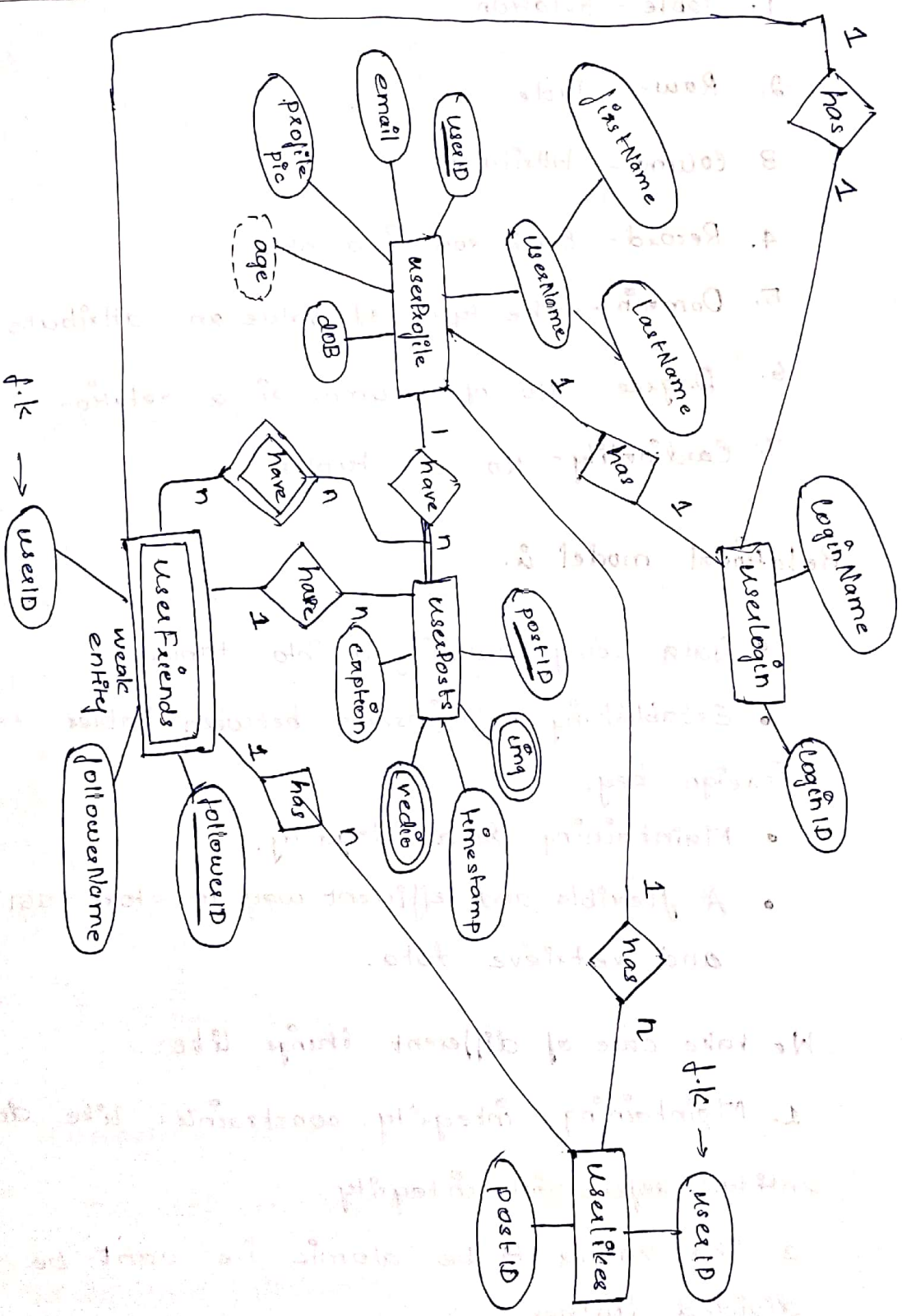
3. userProfile has userLogin (1:1)

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

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

b. userFriends has userLogin (1:1)

7. userFriends has userLoginLikes (1:n) userLikes will always be associated to a ~~userFriends~~ Profile. Therefore total participation.



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 is.

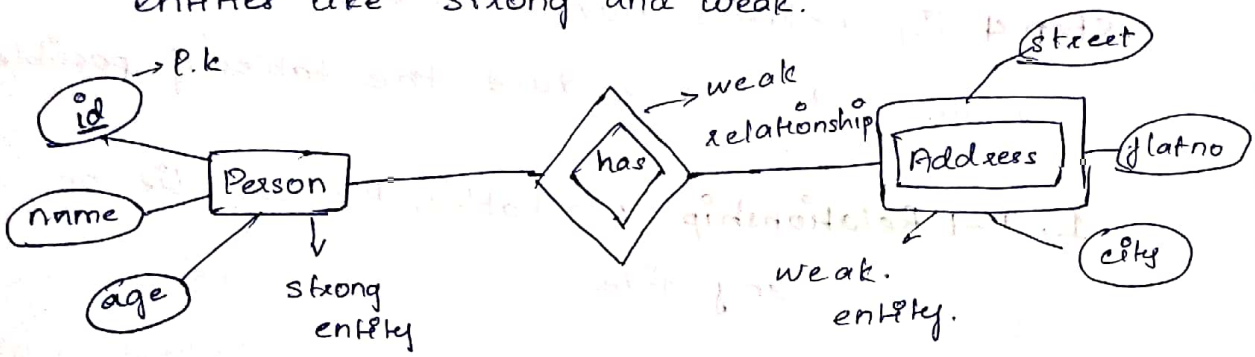
- Data being organized into tables.
- Establishing Relationships between tables using Foreign key.
- Maintaining data integrity.
- A flexible and efficient way to store (SQL) and retrieve data.

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 in to picture i.e candiate super

Converting an Entity-Relationship (ER) model to a relational model.

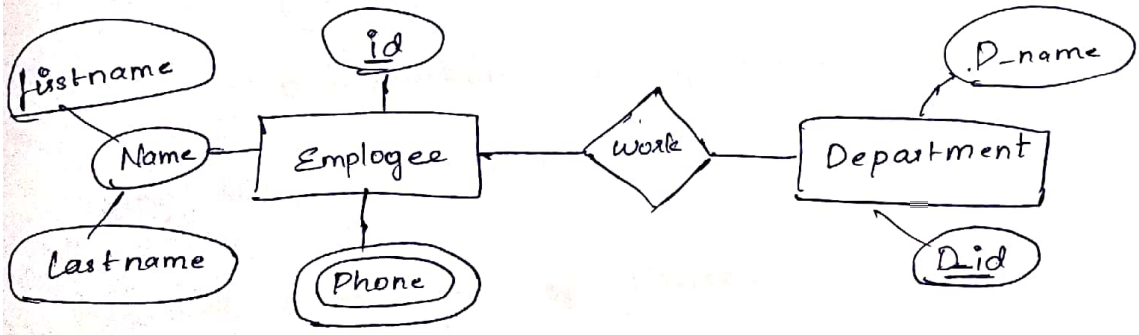
Step 1: Identify the entities - List down all the entities like strong and weak.



1. Person (id, name, age) → id (P.k)

2. Address (id, flatno, street, city) → id + flatno (P.k), id (f.k) (composite P.k)

Step 2: Identify the attributes. - For each entity, identify its attributes which becomes a column in the tables.



Composite attribute

Employee → (id (P.k), firstName, lastName)

Multivalued attribute

Employee → (id (P.k), phoneNumber)

New table with duplicates id.

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 both the table acting as fk

RDBMS.

Database : Collection of data is called as database.

DBMS : A software application to manage our data.

Relational
(Use tables to store data)

- MySQL
- Oracle
- MariaDB

Non - Relational
(Data is not stored in tables)

- MongoDB.

RDBMS (Eg. MySQL)

The databases structure data into organnized tables that have predefined connections between them.

Data manipulation and querying and performed using SQL (Structure Query Language)

Essential components of Tables.

Row/Tuple - Row 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 data being stored and are named to describe the information they hold (eg : "ID", "Name", "Age")

Degree - No. of columns in a table

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 : Ensure 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.

Intension in Database:

The Intension defines what kind of data can be stored and the relationships between them. This is basically the blueprint or definition of the database structure. It doesn't change frequently and is the permanent definition of the database structure.

It includes:

- Table definitions (name of tables, their columns, and the data types allowed in each column)
- Constraints (Rules that govern the data, such as primary keys, foreign keys data validation rules)
- Relationships between tables (how tables are connected through shared columns).

Example:

```
customers (
    Id INT PRIMARY KEY,
    name VARCHAR(50))
```

Extension in Database.

The extension is the actual data stored in the database at a given instance in time Basically the data which is stored in tuples/ rows at a given instance of time. when there are more tuples added the data can change.

Eg.

Employee.

id	name
1	Rahul
2	Afsara

Data at instance t_1

Employee

id	name
1	Rahul
2	Afsara
3	prabha

at instance t_2

Employee

id	name
1	Rahul
2	Afsara
3	prabha
4	magesh

at instance t_3

Normalisation and its types.

Normalisation

Normalisation is a process in which we organise data to reduce redundancy (duplicacy) and improve data consistency. It involves dividing a database into two or more tables.

What is data redundancy and consistency and why its important.

When there is same set of data repeated each and every time it results in duplicacy of data (either in row or column)
Now row level duplicacy can be remove by using primary key for unique values.

Now when we have same data for some set of columns, it leads to different anomalies (inconsistencies or errors that occur when manipulating or querying data in a database)

1. Insertion Anomaly
2. Updation Anomaly
3. Deletion Anomaly.

Also it increases the size of database with the same data.

Insertion Anomaly.

It occcus when it is difficult to insert data into the database due to the absence of other required data.

Consider you want to add a new dept but there is no employee in that dept yet.

id	name	dept	Salary
1	Rahul	'IT'	1500
2	Afsana	'HR'	2000
3	Raj	'IT'	1500

want to add 'AIDS'

Deletion Anomaly;

It occurs when deleting data removes other valuable data.

Consider If you delete all the record in the table, you will loose the track of dept, their manager and salaries.

Updation Anomaly.

It occurs when changes to data require multiple updates.

Consider you want to change the salary for people working in HR department, you need to update it at 3 place.

Using Normalisation we can divide the employee table in two tables

1. Employee
2. Department.

id	name	dept
1	Rahul	IT
2	Afsaan	HR
3	Raj	IT

Dept	Salary
IT	1500
HR	2000

Types of Normalisation

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF).
- Boyce-Codd Normal Form (BCNF)

Denormalization.

This is the opposite of normalisation. It involve intentionally introducing some redundancy into a well-normalized database schema to improve query.

Consider If you wish to find the salary of Rahul.

So first you have to make a query in employee table to find department of Rahul and then in department table to find the salary of Rahul.

Eg. Employee.

id	name
1	Rahul
2	Afsara
3	Prabha

Department

dept	Manage	Salary
IT	Raj	1500
HR	Avinash	1000

Employee.

id	name	dept	Manage	Salary
1	Afsara	HR	Avinash	1000
2	Rahul	IT	Raj	1500
3	Prabha	IT	"	"

Benifits.

- Faster Queries.
- Simpler Queries

allows to execute on a single table instead of requiring joins across multiple tables.

Disadvantages.

- Increased data Redundancy
- Less Data Consistency
- Denormalization can make the database schema less flexible for future adding/ modifying new data elements.