#computer #college-studies IT

By: Yashraj Maher

Syllabus:

Unit 1: Introduction to DBMS, Database System Application Purpose of Database System

- Advantages & Disadvantages of DBMS
- Schema
- Instance & Database Site
- Database System Utilities
- Database Architecture (1, 2, 3 Tier Architecture)

Unit 2: Database Concepts Database Users & Admin Responsibilities, Structure of DBMS

- Types of Data Model: Bottom ER, Object-Based Overview of Hierarchical/Hierarchical & Network Data Models
- Phases of Database Design
- ER Model Entity
- Entity Set
- Entity Types
- Attributes & Types of Attributes
- Representation of ER Symbol
- Data Association Attributes, Association & Mapping Cardinality
- Constraints: Key Constraints Not Null Constraints Stringent Constraints Integrity Constraints, Referential Constraints, Unique Constraints, Domain Check Constraints, Mapping Constraints
- Difference Between DBMS & RDBMS

Unit 3: Data Association:

- Normalization Techniques:
  - 1NF
  - 2NF
  - 3NF
  - BCNF
- Introduction to DDL, DML, Creating Tables
- Insert Data in Tables
- Retrieving Data from Tables using (SELECT) Statement & Sorting Data
- Conversion Function & Conditional Expressions
- Aggregate Data Using Group Functions
- Displaying Data from Multiple Tables
- Sub Queries
- Set Operations

Q.1) Define Data

- Data refers to raw facts, figures, or information collected from various sources, which can be qualitative or quantitative. It serves as the foundation for analysis, decision-making, and deriving meaningful insights.

Q.2) Real-life Example of a Database

- An example of a database is a hospital's patient management system. It stores and organizes patient records, including personal details, medical history, prescriptions, lab reports, and appointment schedules, allowing healthcare professionals to access and update information efficiently.

Introduction

Database

- DBMS is software used to manage a database (DB).
- A database is a collection of interrelated data used to retrieve, insert, and delete information.
- DBMS was created by Charles Bachman.
- A DBMS is a collection of programs that enables users to create and maintain databases.
- The DBMS is a general-purpose system facilitating the manipulation and sharing of databases among several users and applications.
- **Example**:
  - A company database organizes data about admins, employees, managers, clients, and janitors.
- Data is a group of measurements, observations, and descriptions used to convey information.
- DBMS stands for Database Management System.
- Database definition or descriptive information is stored by the DBMS in the form of a catalog or dictionary, called Metadata.
- DBMS is software used to manage databases.
  - Examples: Oracle, MySQL, XAMPP, etc.
- Data refers to raw facts that must be processed by humans or machines to derive meaning.
- Data is the plural form of "datum," derived from Latin.
- DBMS organizes data in the form of tables, schemas, views, and reports.
- **Syntax**:

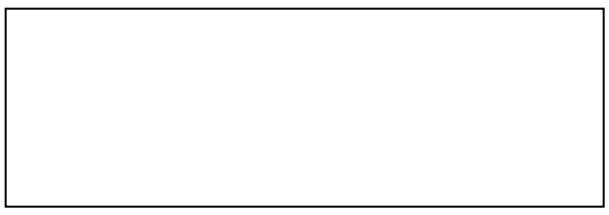
```
CREATE DATABASE db_name;
```

- **Example**:

```
CREATE DATABASE hospital;
```

Entity

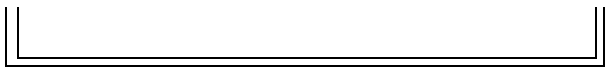
- An entity is represented



- In DBMS, an entity represents any real-world object, while an entity set refers to a group of similar entities.
- An entity is a thing or object in the real world recorded in the database and must be distinguishable to recognize it easily.
  - **Examples**: A student, employee, etc.
- All are Entities:

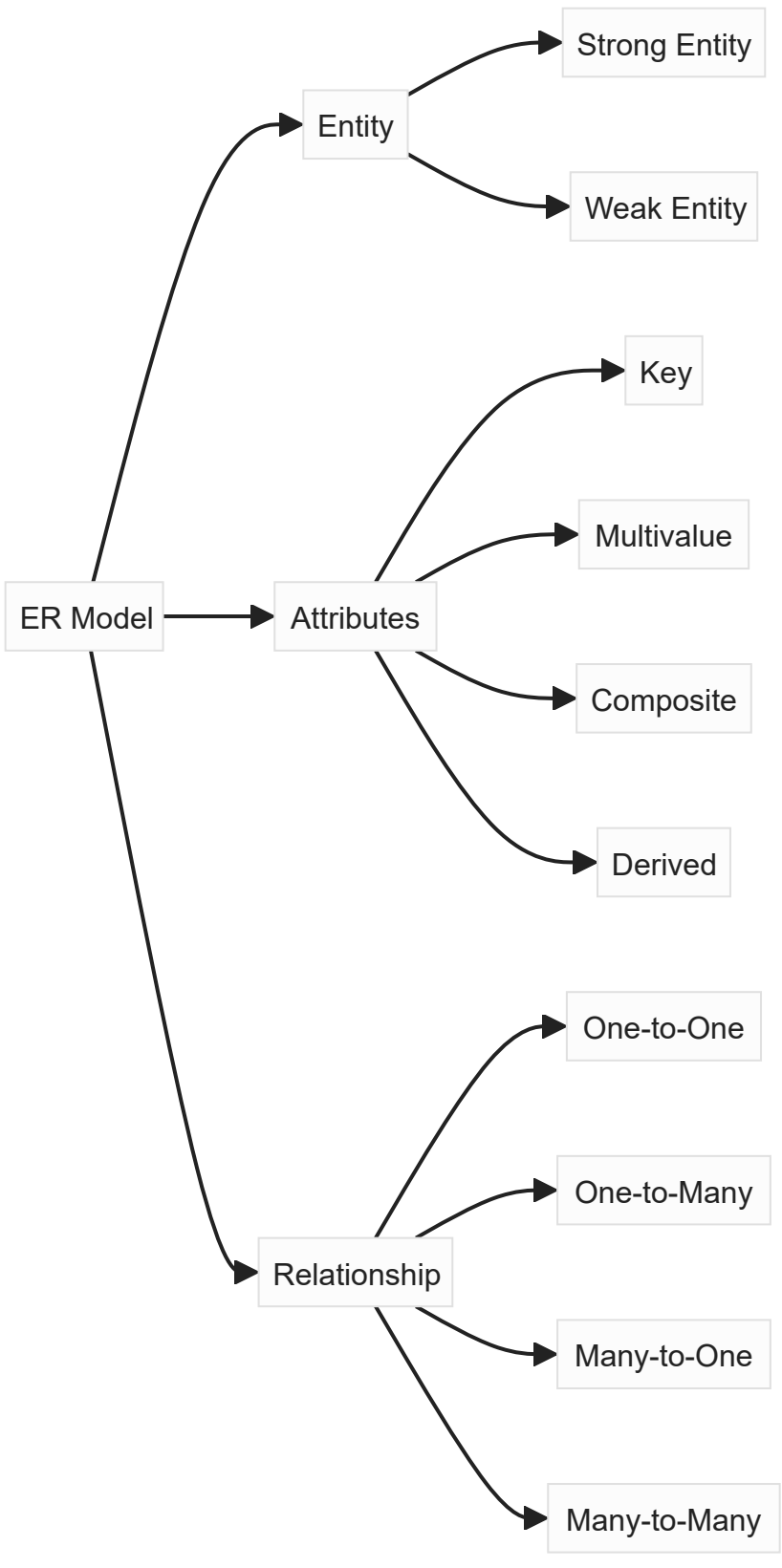
employee
employee_id
name
contact no.

- A category or class of entities that share the same attributes is referred to as an entity kind.
- **Types of Entities**:
  1. **Strong Entity**:
     - Exists independently.
     - Each instance has a unique primary key.
  2. **Weak Entity**:
     - Cannot be uniquely identified by its attributes and relies on relationships with other entities.
     - **Visual Representation**:

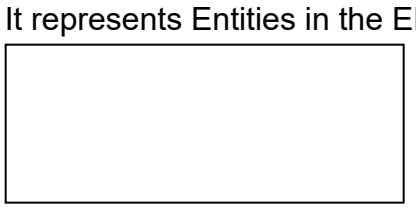


- **Entity-Relationship Model (ER Model)**:
  - Describes the structure of a database with the help of a diagram.
  - Entities can have:
    - **Physical Existence**: e.g., A particular person, car, house, or employee.
    - **Conceptual Existence**: e.g., A company, job, or university course.

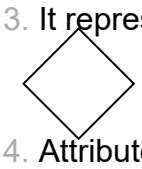
ER Model

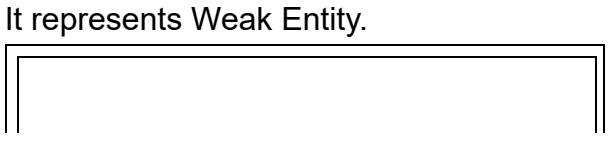


- ER Model is used to Model the logical view of the system from a Data perspective which consists of as followed symbols:
  1. It represents Entities in the ER Model.

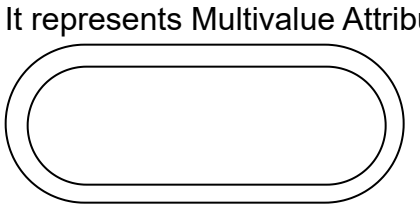
     
  2. It represents Attributes in the ER Model.

     
  3. It represents Relationship among entities.

     
  4. Attributes to Entities and Entity sets with other relationship types.
  5. It represents Weak Entity.

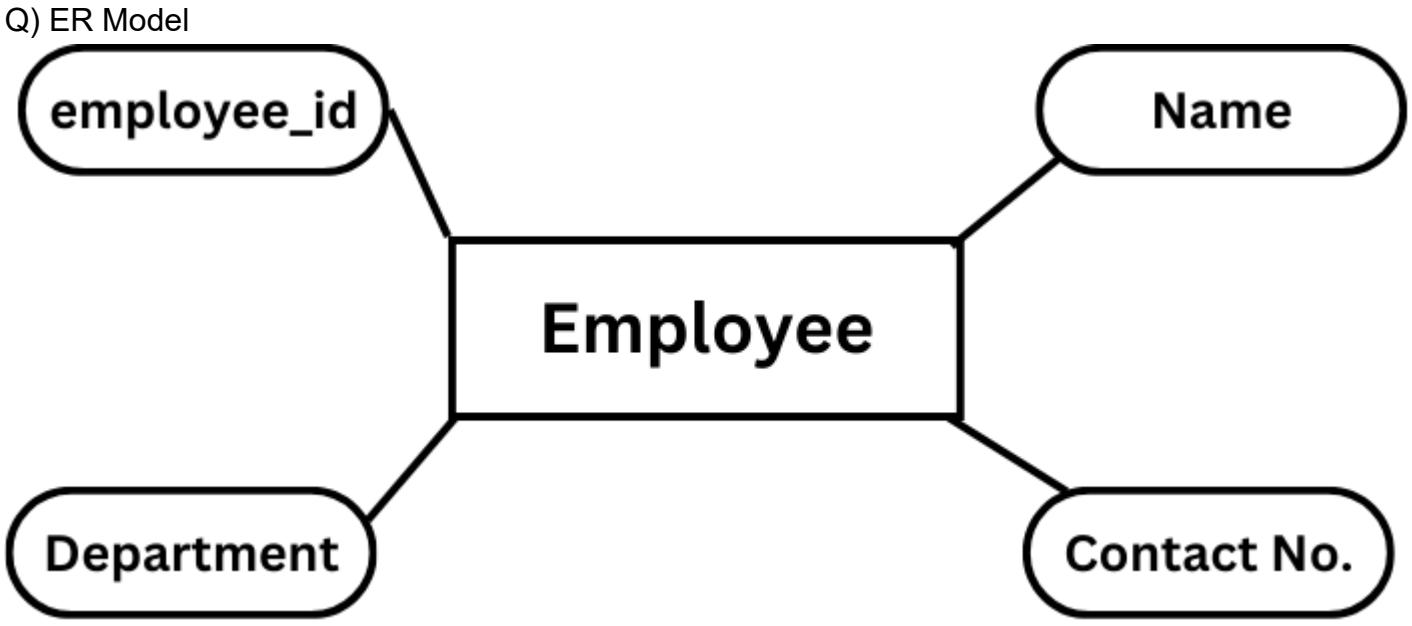
6. It represents Multivalue Attributes.



Q) What is entity and attributes with examples ?

- An Entity is a thing or object in real world they are recorded in the DB and must be distinguish it easy recognize from the group.
  - For Example: Student, Employee etc.
- An Attribute is a property or characteristic of an entity that describes its features or quantity.
- For Example: Student is Entity in College Database and Student has name, roll no. etc as attributes.

Q) ER Model



Q) Blueprint meaning in DBMS.

- In Database management system, a blueprint is a file that define the structure of a database, including its tables, columns, forms and procedure.

Schema

- **Syntax**:

```
CREATE SCHEMA schema_name;
```

  - **Example**:

```
CREATE SCHEMA hospital;
```

- Database schema is a logical representation of data that shows how a data, in a database should be stored logically.
- A Schema in DBMS in a blueprint that defines how data is organized, structured & related in Database the description of a Database is called DB schema which not specified during DB design & is not changed frequently most data models have certain convention for displaying schema as a diagram. A Displayed schema is called schema diagram.
- A schema is a collection of data object like tables, triggers etc.
- The Schema is connected with a user which is know as the schema owner.
- To create schema in SQL server use that 'CREATE SCHEMA' statement.

Advantages and Disadvantages of DBMS

Advantages	Disadvantages
i. Simplicity	Maintenance Problem
ii. Structural Independence	The Maintenance of Relational DB becomes difficult over time due to the increase in the data.
iii. Ease of Use	Cost

Advantages	Disadvantages
iv. Query Capability	Physical Storage
v. Few Relational DB has limits fields, length which cannot be exited.	Complexity in Structure, Decrease in Performance overtime.

Database Schema

- The skeleton of the DB is created by the attributes and this skeleton is named schema.
- The schema mentions the logical constraints like table, primary_id etc. it doesn't represent data type of the attributes.

Applications of DBMS

- Enterprise Information
  - Sales, accounting, human resources, online retails, manufacturing.
- Airlines
  - Reservation and Schedules
- Telecommunication
  - Postpaid, Prepaid, Bill Maintenance
- University
  - It maintains information about student course, loans, banking, transactions, enroll, student grades, stuff roles.
- Banking and Finance Sector
  - Banks maintaining the customer details, accounts, banking transactions, credit card transactions.
  - Finance: Storing information about sales and holding, purchasing of finance stock and bonds.
- Social Media Sites
  - By filling the required details we are able to access social media platform, many users signup daily or websites such as facebook, discord , reddit, X, bereal etc. All the information related to the users are stored and maintained with the help of DBMS.
- Manufacturing
  - Manufacturing companies make products and sell them on a daily basis.
  - To keep records of all of those details DBMS is used.
  - The DBMS is playing a very important role in each and every field.

Instance

Example:
Rollno int
name varchar(20)
DOB
Pin
Address

- The Collection of information is stored at a particular moment is called as an Instance.
- Instance changes when addition, delete, updating but if you searching then it cannot change instances.

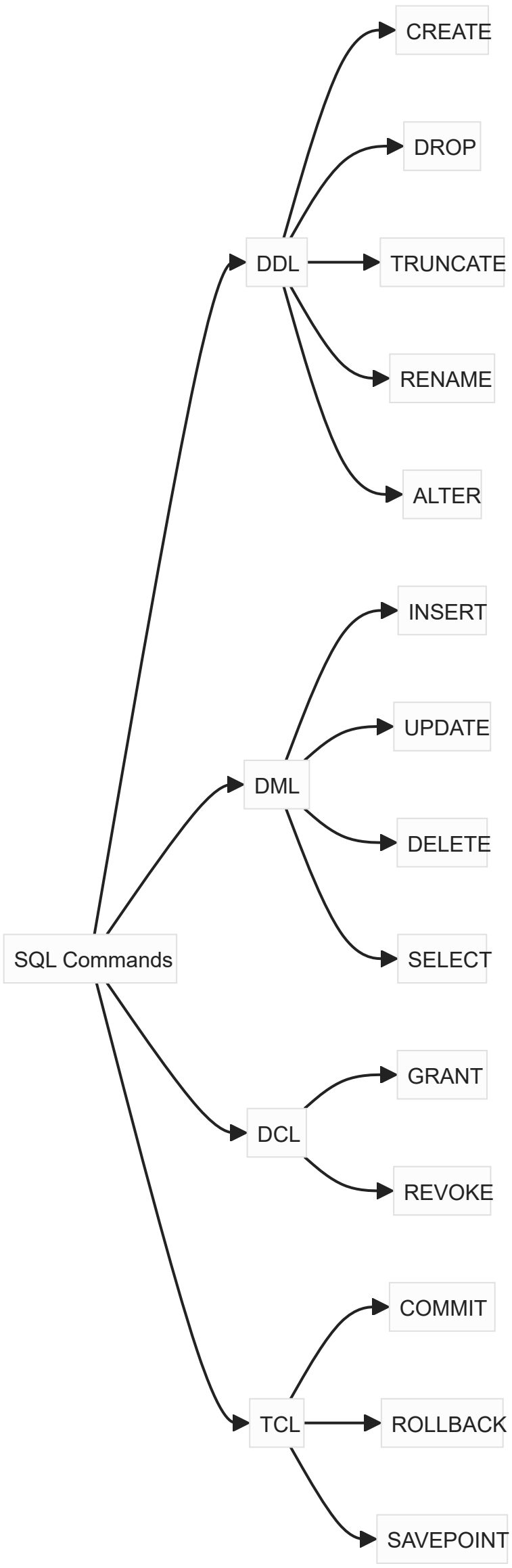
Utilities

- Utilities are programs designed to help the DBA with various administration tasks.
- Some Utility programs operate at the external level of the system, and thus are effectively nothing more that special-purpose applications: some might not even be provided by the DBMS vendor, but rather by some third party.
- Other Utilities, however operate directly at the internal level (in other words, they are really part of the server), and hence must be provided by the DBMS vendor.
- Here are some examples of the kind of utilities that are typically needed in practice:
  - Load routines, to create the initial version of the database form regular data files.
  - Unload/reload (or dump/restore) routines, to unload the database or portions thereof to backup storage and to reload data from such backup copies (of course the "reload utility" is basically identical to the load utility just mentioned).
  - Reorganization routines, to rearrange the data in the stored database for various reasons (usually having todo with performance) -- for example, to cluster data in some particular way on the disk, or to reclaim space occupied by logically deleted data.
    - Statistical routines, to compute various performance statistics such as file sizes, value distributions, I/O counts, and so on.
    - Analytics routines, to analyze the statistics just mentioned.
- Of course, this list represents jus a small sample of the range of functions that utilities typically provide; numerous other possibilities exist.

Introduction to SQL Commands

- The SQL commands are instructions used to communicate with database, its also used to perform specific task, functions and Queries of data.
- There are four types of SQL commands

SQL Commands Flowchart



- It can perform several task like create table, delete, update, truncate table, modify the table, set permissions for users.

DDL (Data Definition Language)

Create

- The create command is used to create new table in the database also it can create database.
- Syntax for creating database and table

```
CREATE DATABASE db_name;
CREATE TABLE tb_name(
        column1_name datatype,
        column2_name datatype
);
```

- For Example

```
CREATE TABLE student_data(
                Rno int,
                Name varchar(20),
                DOB date,
                Pin int,
                Address text,
                email text
);
```

Drop

- It is used to delete both the structure and record stored in the table.
- To drop a table permanently from memory.
- Syntax

```
DROP TABLE tb_name;
```

Alter

- It is used to alter the structure of the database this change could be either to modify the characteristic of an existing attributes of probably to add a new attribute.
- By using alter command we can add or drop one or more columns from the existing tables, also we can add new columns in existing table.
- Syntax

```
ALTER TABLE tb_name ADD column_name column_definition;
```

- For example

```
ALTER TABLE student ADD Rln int;
```

Truncate

- It is used to delete all the rows from the table and free the space containing the table.
- Syntax

```
TRUNCATE TABLE tb_name;
```

- For Example

```
truncate table student;
```

Rename

- It is used to rename the table.
- Syntax

```
rename old_tb_name TO new_tb_name;
```

- For Example

```
rename student TO student_data;
```

DML (Data Manipulation Language)

- It is a class of statements that are used to query, edit, modify and add or delete row level data from the database views.
- The Main DML commands or statements are:
  - INSERT, SELECT, UPDATE, DELETE

SELECT

- To access the data in database object use SELECT statement.
- Syntax

```
SELECT * FROM tb_name;
```

- Example

```
SELECT * FROM student;
```

INSERT

- Using this command we can add row to a table.
- Syntax

```
INSERT INTO tb_name VALUES();
```

- Example

```
INSERT INTO student VALUES(102,'abc','L-*O232S');
```

DELETE

- The delete command will remove rows in existing table in your DB.
- Using usually specify WHERE CLAUSE with your DELETE statement to only remove specific rows from a table but you should not really ever delete rows from the tables.

- Instead you should apply filters on queries themself' to remove rows from your modeling or analysis.
- Syntax

```
DELETE FROM tb_name WHERE conditions;
```

- Example

```
DELETE FROM student WHERE Rno = 101 AND Name = "Abc";
```

UPDATE

- It allows users to update or modify the existing data in database table's.
- Syntax

```
UPDATE tb_name SET [column1_name = Value, column2_name = Value] WHERE condition;
```

- Example

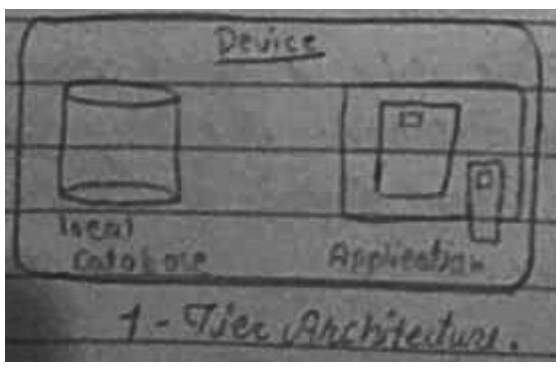
```
UPDATE student SET [Marks = 90] WHERE PRN = 110;
```

Database Architecture

---

1 Tier Architecture / Client Tier Architecture

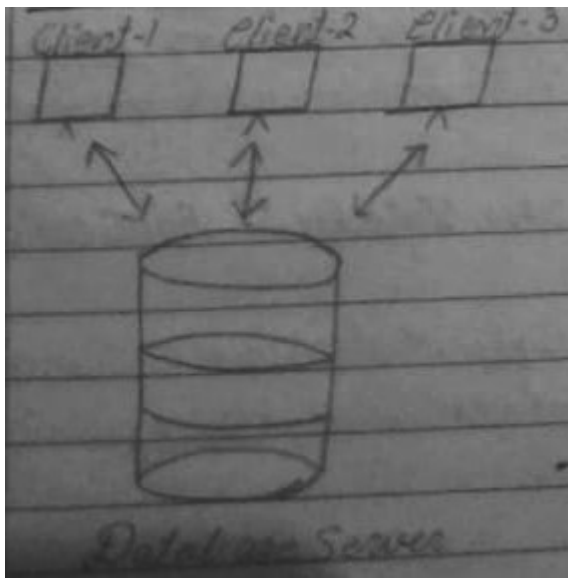
- All the Application & Data are present on one Computer.
- Even presentation will be also done on The Same Computer.
  - **Ex:** Microsoft Excel, Word, Even Games
- The whole Application (consider one of them) expressed on one device & all The Presentation & View is also on Same device the application is installed & All Data related To it will be stored on the Same Computer.
- There is No other layer For This Type of Architecture. Everything is present on a Single Machine.
  (Geeks for Geeks)
- In 1-Tier Architecture, The Database is Directly available To User. The user Can directly sit on The DBMS & use it. That is, The client, Server & Database all are present on Same Machine.
  - **Ex:** Microsoft Access - A user opens Microsoft Access on Their Computer.
- The application Causes 1 directly Access The Local Database Like 1 Performs operation like Querying, Inserting, Updating or Deleting records.
- There is **No Separation Between The Application & The Database** Since Both resides on a Same Machine.



Advantages of 1-Tier Architecture	Disadvantages of 1-Tier Architecture
Simple Architecture: 1-Tier Architecture is the most simple architecture to set up as only a single	**Scalable:** Only one user can access the system at

Advantages of 1-Tier Architecture	Disadvantages of 1-Tier Architecture
machine is required to maintain it.	a time.
Cost Effective: No extra hardware is required for implementing 1-Tier Architecture, which makes it cost effective.	**Cannot Share Data:** Data cannot be shared on the client machine.
Easy To Implement: 1-Tier Architecture can be easily deployed (moved) & hence it is mostly used in small projects.	**Application May Not Work:** It may not work if changes are made in the machine.

2-Tier Architecture



- Two Tiers Mean 2 Layers.
- Here are Two Layers: one **Client Layer** & second **Database Layer**.
- Here, Client is a pc Machine in which a Interface is running & This interface is helping us To fetch The data from The database Server.
- It is for Interaction with database using **JDBC - ODBC**.

Here 1st Client & Database Source will form Connection

Then a Query will be written on the Interface & Then this Query will come To Database Source. Here it will get Processed (Because our written Programme can be in the high level language) So To convert it into low level language Processing is done & after This whatever would be the demand of the client will be given back to it.

- This is how 2-Tier Architecture works.
- Limited Client & Database have Maintenance is Easy.

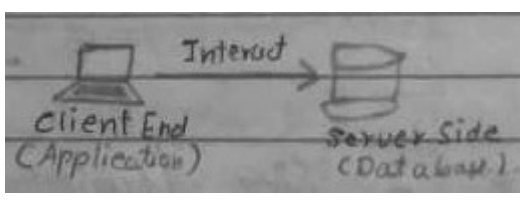
Problems

- **Scalability**
- **Security** - Client is directly Interacting with Database

---

(Geeks for Geeks)

- The 2-Tier Architecture is Similar to a Basic Client-Server Model.
- The Application at the Client End Directly Communicates with Database on Server Side.
- API's Like JDBC & ODBC are Used for this Interaction



- The Server Side is Responsible for Providing Query Processing (Solving Given Program or Question) & Transaction Management functionalities.
- On the Client Side, the User Interface & Application Program are run.
- The Application on Client Side Establishes a Connection with Server Side To Communicate with DBMS.
- An Advantage of *This Tape* is that maintenance & understanding are easier & compatible with existing systems.
- However, this model gives poor performance when there are large numbers of users.

(Application Client) <-> (Application Server)

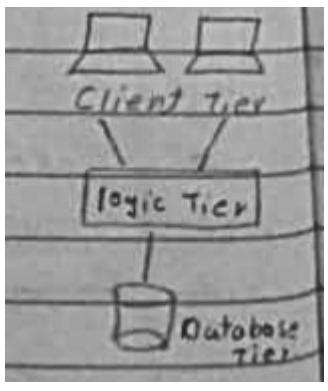
Advantages of 2-Tier Architecture	Disadvantages
Easy To Access - 2 Tier Architecture makes easy access to Database which makes fast retrieval.	**Security** - Client directly interacts with the Database which can expose the sensitive data.
Scalability - We can scale the Database Easily by including clients or upgrading hardware.	**Scalability** - It is harder to protect system from security threats.
Low Cost - 2 Tier Architecture is cheaper than 3 Tier Arch. & Multi - Tier Arch.	- As more users connect, the system or Database gets overloaded, slowing things down.
Easy Deployment - 2 Tier Arch. is easier to deploy than 3 - Tier Architecture.	- It's harder to expand system to handle more users.
Simple - 2 Tier Architecture is easily understandable & well as simple because of only 2 components.	**Single Point of Failure** - If server or Database goes down, the whole system stops working because there is no backup layer to help things running.

---

3 Tier Architecture

- In a Three Tier Architecture for DBMS, The System is divided in 3 Distinct layers - Each with Specific Role.
- These structures improve Scalability, Scalability, maintainability & Security by separating the different responsibilities.

1. **Presentation Tier (Client Tier)**:
   - **Role**: This is the Topmost layer that Interacts with the user. Its Consist of User Interface, where users can input Data, view results & Interact with System.
   - **Ex**: A Web Browser, Mobile App, Desktop Application.
   - **Functions**: It sends user requests to Middle layer & receives the Processed Data from Middle layer & Presents the result to user in a readable form (like a web page or a application).



2. Logic Tier (Application / Business Logic Tier)

- **Role**: This layer acts as an intermediary between the Client Tier & the Database Tier.
  - This layer handles the business logic, Processing of Data & Performs any Memory Calculations or Transaction on behalf of the Client Tier & Database or Back to the User.
- **Ex**: Web Servers, Application Servers or Business Logic Engines.
- **Functions**: It receives user input, Interacts with the Database to retrieve or Modify data, applies business rules & sends results back to the (Presentation Tier/Client Tier).

3. Database Tier

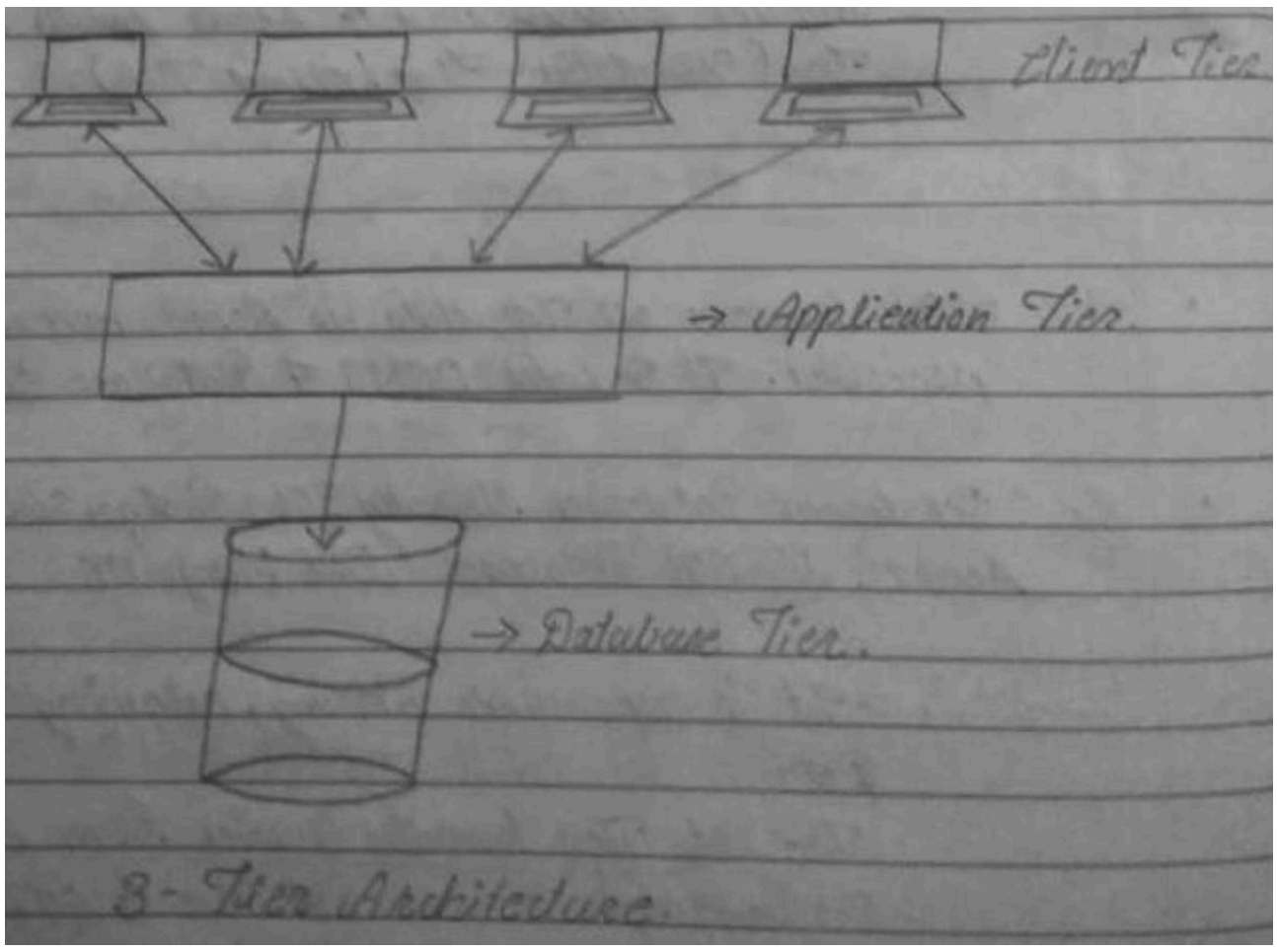
- **Role**: This is where all the data is stored, managed & processed. It involves a DBMS & Database itself.
- **Ex**: Relational Databases like MySQL, PostgreSQL or SQL Server, NoSQL Databases like MongoDB.
- **Functions**: It is responsible for storing, retrieving & updating Data.
  - The Data Tier handles Queries from Logic Tier.

- Perform operations on Database & returns the results.

Workflow in a 3-Tier Architecture

---

1. **The user interacts with the Presentation (Web Interface).**
2. **The request is sent to the Application Tier, which processes the request and communicates with the Database Tier.**
3. **The Database Tier executes queries, retrieves data, and sends it back to the Application Tier.**
4. **The Application Tier then utilizes the processed data in the Presentation Tier, which displays it to users.**



Advantages	Disadvantages
1. **Each layer is separate, which makes it easier to manage & update each part without affecting others.**	1. **Complexity** - Managing & maintaining 3 different layers can be more complicated than a simple architecture.
2. **You can scale each tier independently.** E.g. you can include more servers to the Database Tier without affecting Presentation layers.	2. **Performance Overhead** - Communication between tiers can lead to delays or latency & reduce overall system performance.
3. **By separating the data & business logic layer from Presentation layer, sensitive data can be better protected & managed.**	3. **Cost** - More resources (servers, infrastructure) may be needed to manage each separate layer, increasing the overall cost.

Data Models

- **Data Models** are mainly useful in order to design the **Database**.
- Data Model gives complete idea about how final system would look like after its implementation.
- A **Data Model** in DBMS is a **conceptual framework** that defines **structure**, **relationships**, **constraints** of data stored in **Database**.
- It serves as **blueprint** for designing database, describing how data is **organized**, how it can be **accessed** & **manipulated**.
- Data Model provides a way **to describe the logical structure** of data, **independent** of actual implementation in **DBMS**.
- **E.g.**
  - **MySQL** - A widely used open-source relational DBMS. It stores data in **Tables** & is used by many web applications.

Example Use - A website storing **Users Information**, **Names**, **Email**, **Password** in Database.

1. **Hierarchical DM**
2. **Network DM**

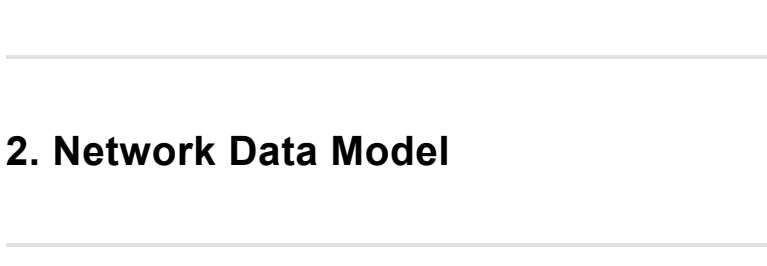
3. **Entity-Relationship DM**
4. **Relational Model**
5. **Object Based Data Model**

---

1. Hierarchical Data Model (Developed by IBM 1950's)

---

1. **It is mainly used to store the information in a hierarchical level by level manner.**
    - **Grand Parent level**, **Parent level**, **Children level**
    - **It forms a tree like structure**



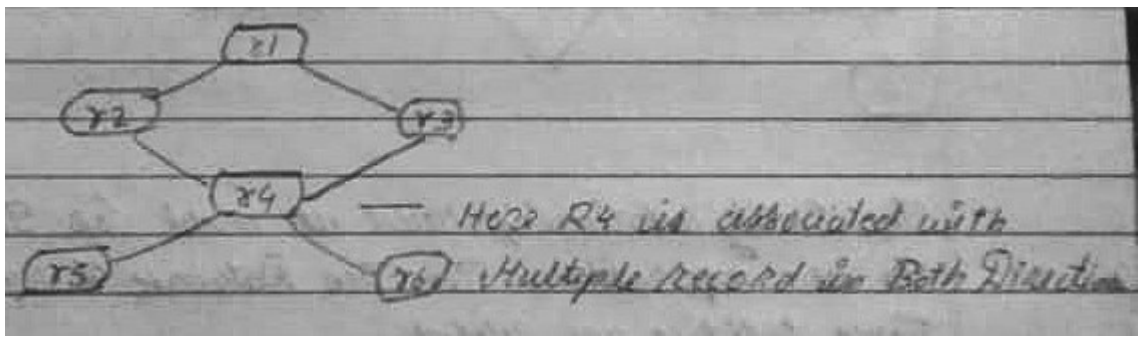
2. **It is mostly used to store one to many relationships.**
    - Each node will have only one Parent Node & Many Children.
3. **In this Data is organized as a Tree like structure where each record consists of one Parent record & Many Children.**
    - In Hierarchical Model, Segments Pointed to by the logical navigation are called the **child segment** & other segment is called **Parent segment**.
    - If there is a segment without Parent, it is called **Root Segment** which has no children & are called **Leaves**.

Advantages	Disadvantages
1. **It has very simple hierarchical DB structure.**	1. **Lacks flexibility. Deletion of one segment can lead to deletion of all segments under it.**
2. **They have Data sharing, as all data are held in common DB.**	2. **It has no standard.**
3. **It offers Data Security.**	3. **It is also limited as many of common relationships do not conform to 1 to N format as required by Hierarchical Model.**

---

2. Network Data Model

- It follows the **graph structure**.
- It follows **many-to-many relationships** where **each node can have many children & parents**.
- It is an **extension of Hierarchical Data Model**.
- The **Network Data Model** is one of the **oldest data models** that was designed to handle **complex data relationships more effectively than the Hierarchical Model**.
- In **Network Model**, data is organized in a **graph structure** which allows for **more flexibility & complex relationships between different types of data**. This model is **particularly useful for representing many-to-many relationships where a single record can be associated with multiple other records in both directions**.



- Here, **R2 is associated with R4 & R5 in multiple records in both directions**.

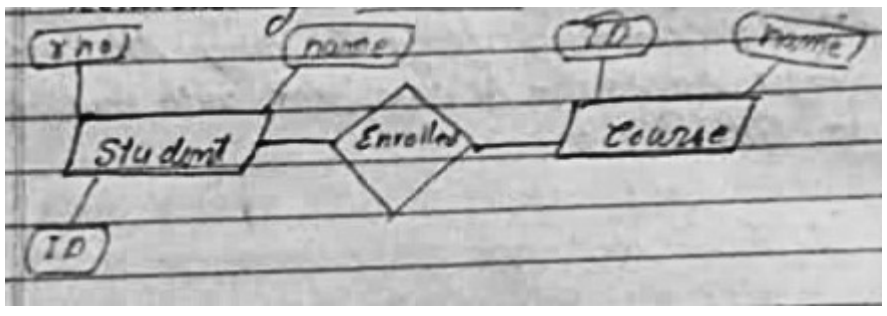
Advantages
1. **The Network Model is flexible**. It allows to represent **complex real-world relations**.

Advantages
2. **The Network Model allows many-to-many**. This is useful in cases where an entity might have **multiple relations with others**.
3. **Data Integrity**.
4. **Supports Multiple Parent Records**.

---

3. Entity-Relationship Model (ER-Model)

- Contains **3 things**:
    1. **Entity**
    2. **Attributes**
    3. **Relationship**
- **Entity**: Anything that has **real physical existence** is called an **entity**. It is **distinguished**.
- **Attributes**: Properties of entities. It tells more clearly about an entity.
- **Relation**: It is used to **relate two entities**.
    - Entities are represented by **rectangle**.
    - Attributes by **ellipse**.
    - Relations by **Diamond**.



- The **Entity Relationship Model** is a model for **identifying entities to be represented in the Database & representation of how those entities are related**.
- The ER Data Model specifies **clearly the Enterprise (project) Schema that represents the overall logical structure of a DB graphically**.
- Peter Chen developed **ER Model** in 1976.
- The ER Model was **created to provide a simple & understandable model for representing the structures & logic of databases**.
- The ER Diagram explains the **relationship among different entities present in the DB**. ER Model is used to **model real-world objects like a person, car, company & the relations between these real-world objects**.

---

4. Relational Model

- A **Relational Database** is defined as a group of **Interrelated Tables** which are **linked** to each other using some common fields of each related table.
- This Model can be represented as **Model with rows & columns**.
- Each row is known as **Tuple**.
- Each field of a column has a **name or attribute**.
- Column - **Table**
- It is well known as **DB Technology** because it is widely used to represent **real world objects & relations between them**.
- E.g.: **Oracle**, **Sybase**, **MySQL Server** etc. - **Relational Models**.

---

5. Object Based Data Model