# Unit 1: DBMS CONCEPTS

## 1.1 What is DATABASE

- A database is like a highly organized collection of information.
- It is like a library houses books or a grocery store organizes different kinds of food.
- these collections are stored electronically in computer systems and allow for efficient access, management, and analysis of the data.
- Database handlers create a database in such a way that only one set of software program provides access of data to all the users.

Here are some key things to understand about databases:

**Structure:**

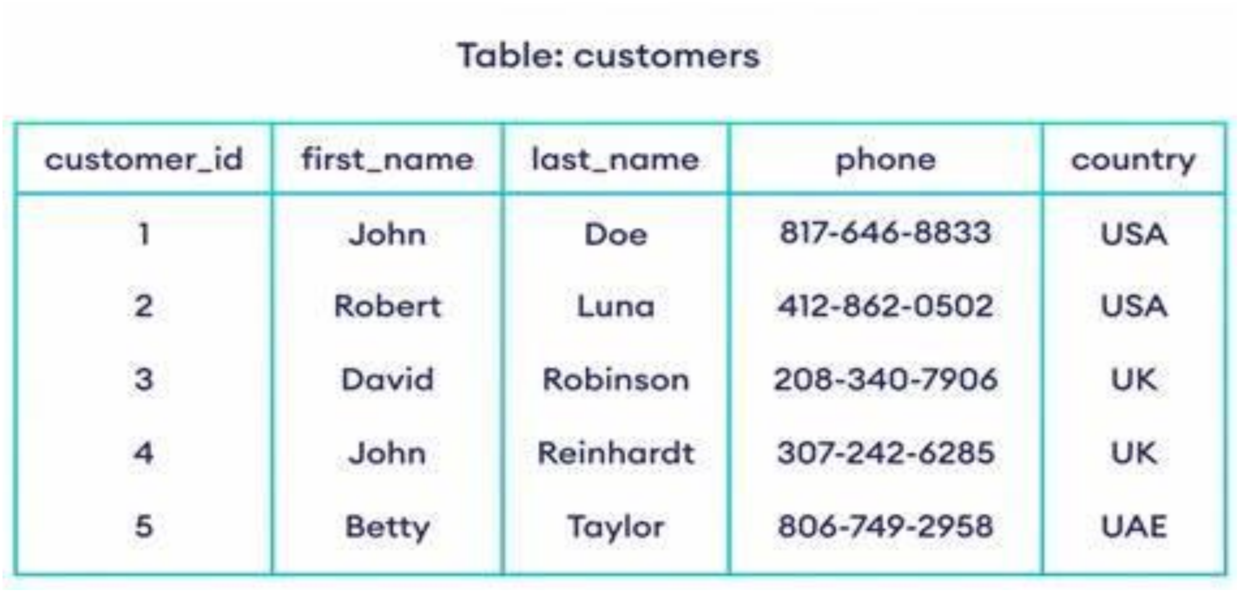
- Data in a database is typically organized in tables, which are similar to spreadsheets.
- Each table has rows and columns, where rows represent individual records and columns hold specific data points about those records.
- Tables are related to each other through relationships defined by shared fields. This allows for efficient and accurate retrieval of data across multiple tables.
- 

Table: customers

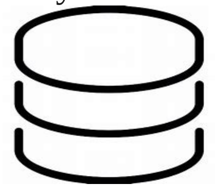
| customer_id | first_name | last_name | phone | country |
|---|---|---|---|---|
| 1 | John | Doe | 817-646-8833 | USA |
| 2 | Robert | Luna | 412-862-0502 | USA |
| 3 | David | Robinson | 208-340-7906 | UK |
| 4 | John | Reinhardt | 307-242-6285 | UK |
| 5 | Betty | Taylor | 806-749-2958 | UAE |

**Types of data:**

- Databases can store different types of data, including numbers, text, dates, images, and even multimedia files.
- Choosing the right data type for each field is crucial for optimizing storage and ensuring data integrity.

**Accessing and manipulating data:**

- We use special languages called query languages like SQL (Structured Query Language) to interact with databases.
- SQL allows us to retrieve, insert, update, and delete data in specific ways.
- Database management systems (DBMS) are software programs that handle the overall functionality of a database, providing access control, security, and performance optimization.
- A cylindrical structure is used to display the image of a database.



**Uses of databases:**

- Databases are used in almost every aspect of modern life, from online shopping and social media to banking, healthcare, and government services.
- They allow organizations to store large amounts of data, analyze trends, make informed decisions, and ultimately improve their services and operations.

**Here are some real-world examples of databases:**

- Online store: Product information, customer order history, and inventory levels are stored in a database for efficient management and analysis.
- Social media platform: User profiles, posts, comments, and connections are stored in a database to personalize user experience and facilitate interactions.
- Hospital: Patient medical records, diagnoses, medications, and treatment plans are stored in a database for secure access and coordinated care.

## Evolution of Databases

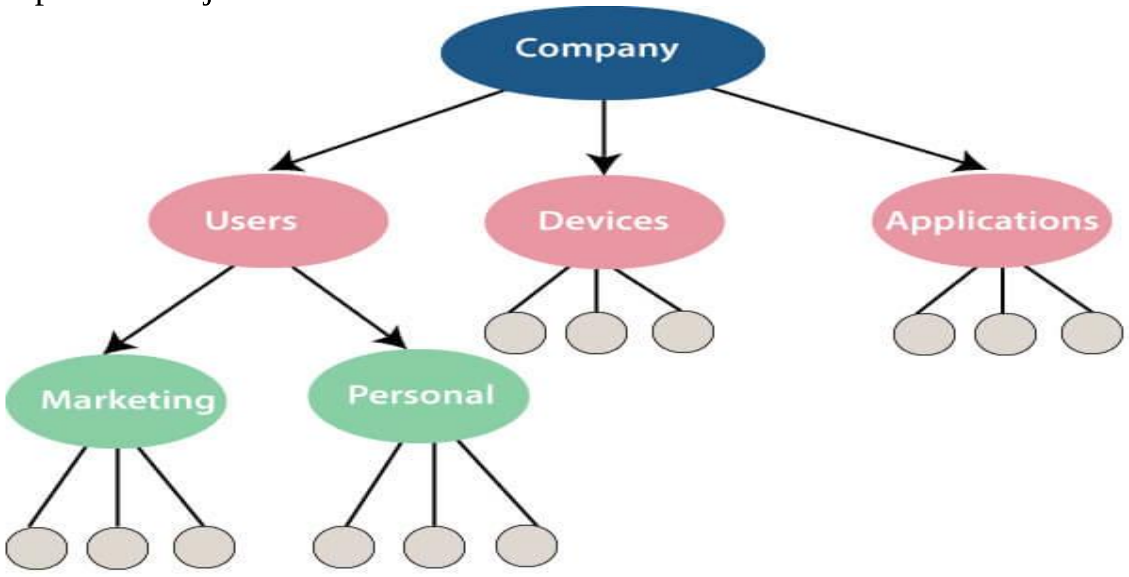
- its evolution from flat-file system to relational and objects relational systems. It has gone through several generations.

### File-Based

- 1968 was the year when File-Based database were introduced. In file-based databases, data was maintained in a flat file. Though files have many advantages, there are several limitations.
- One of the major advantages is that the file system has various access methods, e.g., sequential, indexed, and random.
- It requires extensive programming in a third-generation language such as COBOL, BASIC.

### Hierarchical Data Model

- 1968-1980 was the era of the Hierarchical Database. Prominent hierarchical database model was IBM's first DBMS. It was called IMS (Information Management System).
  In this model, files are related in a parent/child manner.
- Below diagram represents Hierarchical Data Model. Small circle represents objects



Like file system, this model also had some limitations like complex implementation, lack structural independence, can't easily handle a many-many relationship, etc.

## Relational Database

- **1970 - Present:** It is the era of Relational Database and Database Management. In 1970, the relational model was proposed by E.F. Codd.
- Relational database model has two main terminologies called instance and schema.
- The instance is a table with rows or columns

Schema specifies the structure like name of the relation, type of each column and name.

**A relational database contains the following components:**

- Table
- Record/ Tuple
- Field/Column name /Attribute
- Instance
- Schema
- Keys

An RDBMS is a tabular DBMS that maintains the security, integrity, accuracy, and consistency of the data.

## Cloud database

Cloud database facilitates you to store, manage, and retrieve their structured, unstructured data via a cloud platform. This data is accessible over the Internet. Cloud databases are also called a database as service (DBaaS) because they are offered as a managed service.

**Some best cloud options are:**

- AWS (Amazon Web Services)

- Snowflake Computing
- Oracle Database Cloud Services
- Microsoft SQL server
- Google cloud spanner

**Advantages of cloud database**

- **Lower costs -** Generally, company provider does not have to invest in databases. It can maintain and support one or more data centers.
- **Automated** - Cloud databases are enriched with a variety of automated processes such as recovery, failover, and auto-scaling.
- **Increased accessibility** - You can access your cloud-based database from any location, anytime. All you need is just an internet connection.

**NoSQL Database**

- A NoSQL database is an approach to design such databases that can accommodate a wide variety of data models.
- NoSQL stands for "**not only SQL**." It is an alternative to traditional relational databases in which data is placed in tables, and data schema is perfectly designed before the database is built.

NoSQL databases are useful for a large set of distributed data.

Some examples of NoSQL database system with their category are:

- MongoDB, CouchDB, Cloudant **(Document-based)**
- Memcached,
- Redis, Coherence **(key-value store)**
- HBase, Big Table, Accumulo **(Tabular)**

**Advantage of NoSQL**

- **High Scalability-** NoSQL can handle an extensive amount of data because of scalability. If the data grows, NoSQL database scale it to handle that data in an efficient manner.
- **High Availability -** NoSQL supports auto replication. Auto replication makes it highly available because, in case of any failure, data replicates itself to the previous consistent state.

**Disadvantage of NoSQL**

- **Open source -** NoSQL is an open-source database, so there is no reliable standard for NoSQL yet.
- **Management challenge -** Data management in NoSQL is much more complicated than relational databases. It is very challenging to install and even more hectic to manage daily.
- **GUI is not available -** GUI tools for NoSQL database are not easily available in the market.
- **Backup -** Backup is a great weak point for NoSQL databases. Some databases, like MongoDB, have no powerful approaches for data backup.

## 1.2 DBMS

**Database Management System (DBMS)** is a software system that allows for the management and organization of data in a database. Here are some important points about DBMS:

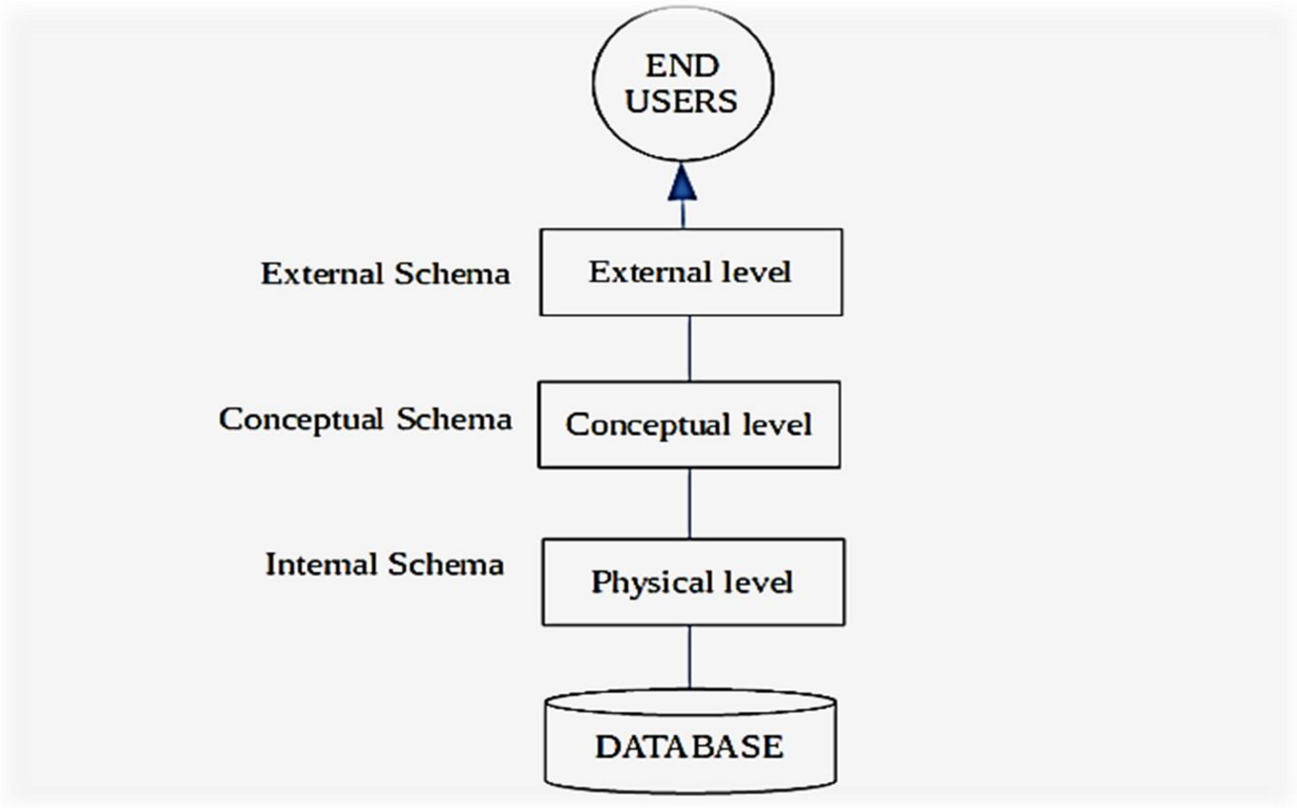
common types of Database Management Systems (DBMS) include:

- **Relational Database Management Systems (RDBMS):**
  - These systems store data in tables that have rows and columns, and use a standardized query language (SQL) to access and manipulate the data. Examples include Oracle Database, Microsoft SQL Server, and MySQL.

- **Hierarchical Database Management Systems:** In these systems, data is organized in a tree-like structure, with each record having only one parent and potentially multiple children. An example of a hierarchical DBMS is IBM's Information Management System (IMS).

- **NoSQL Database Management Systems:** These systems are designed to handle large volumes of unstructured or semi-structured data, and often use alternative data models such as key-value, document, or graph databases. Examples include MongoDB, Cassandra, and Neo4j.

### Architecture of DBMS – Three Level:



The details of these levels are as follows −

**Physical Level**

- This is the lowest level in the three level architecture. It is also known as the internal level.
- The physical level describes how data is actually stored in the database. In the lowest level, this data is stored in the **external hard drives**
- The physical level also discusses compression and **encryption techniques**.

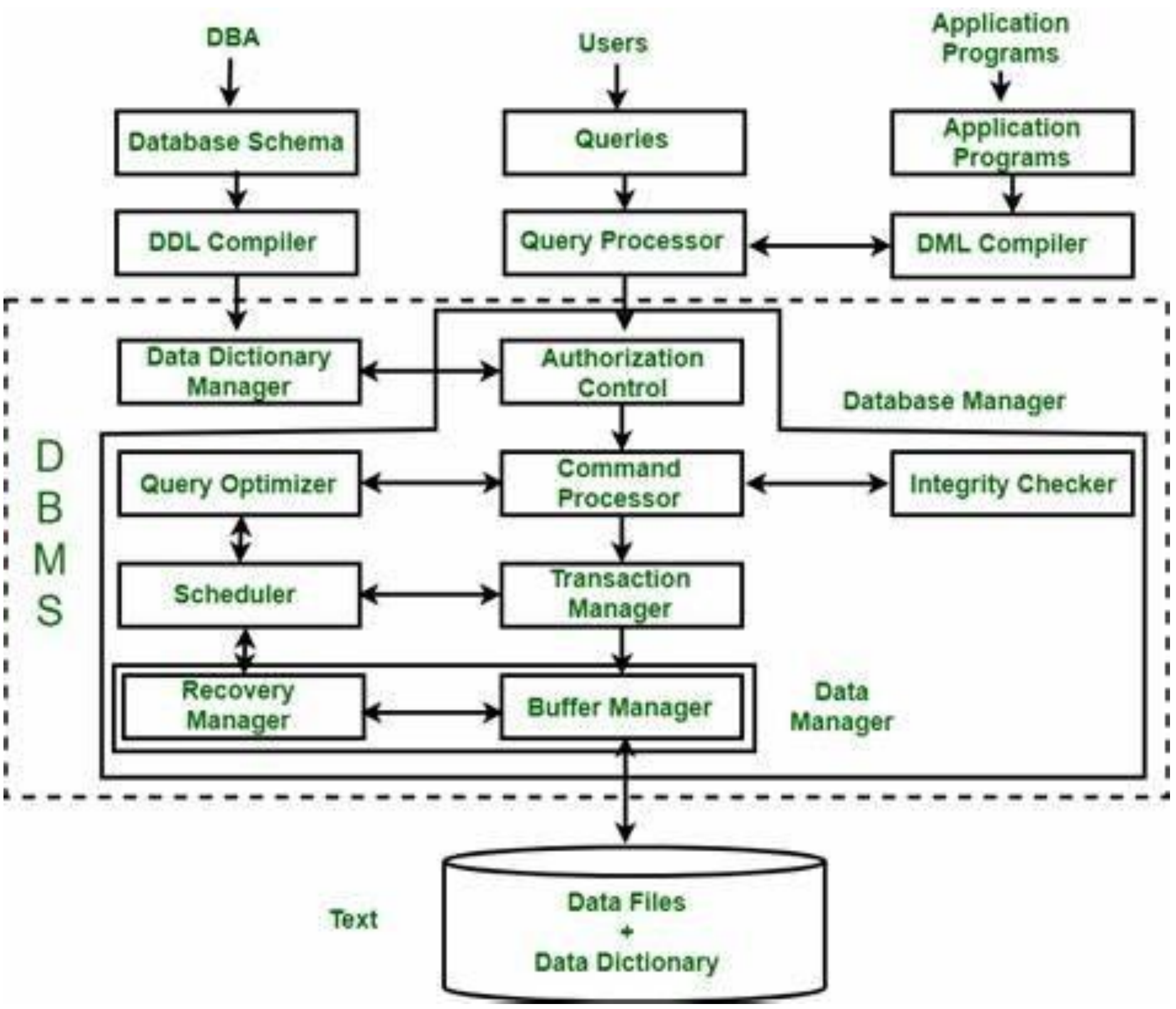
**Conceptual Level**

- The conceptual level is at a higher level than the physical level. It is also known as the logical level.
- It describes how the database appears to the users conceptually and the relationships between various **data tables**.
- The conceptual level does not care for how the data in the **database** is actually stored.

**External Level**

- This is the highest level in the three level architecture and closest to the user. It is also known as the view level.
- The external level only shows the relevant database content to the users in the form of views and hides the rest of the data. So different users can see the database as a different view as per their individual requirements.

## 1.4 Structure of DBMS:



**Structure of DBMS is Classified under**

- **Query Processor**
- **Storage Manager**
- **Disk Storage**

### 1. Query Processor

- The query processing is handled by the query processor, as the name implies. It executes the user's query, to put it simply.

- The query processor's primary duty is to successfully execute the query.

- The Query Processor transforms (or interprets) the user's application program-provided requests into instructions that a computer can understand.

**Components of the Query Processor**

- **DDL Interpreter:**

  - the DDL Interpreter interprets DDL statements like those used in schema definitions (such as create, remove, etc.).

  Note: An interpreter is a program that directly executes the instructions in a high-level language, without converting it into machine code

- **DML Compiler:**
  - the DML Compiler converts DML statements like select, update, and delete into low-level instructions or simply machine-readable object code, to enable execution.
  - This process, known as query optimization, is exclusively carried out by the DML compiler. Simply put, query optimization determines the most effective technique to carry out a query.

- **Query Optimizer:**

  - It starts by taking the evaluation plan for the question, runs it, and then returns the result. Simply said, the query evaluation engine evaluates the SQL commands used to access the database's contents before returning the result of the query.

**2. Storage Manager:**

- An application called Storage Manager acts as a conduit between the queries made and the data kept in the database. Another name for it is Database Control System.

- By applying the restrictions and running the DCL instructions, it keeps the database's consistency and integrity.

- It is in charge of retrieving, storing, updating, and removing data from the database.

**Components of Storage Manager**

- **Integrity Manager:**

  Whenever there is any change in the database, the Integrity manager will manage the integrity constraints.

- **Authorization Manager:**

  Authorization manager verifies the user that he is valid and authenticated for the specific query or request.

- **File Manager:**

  All the files and data structure of the database are managed by this component.

- **Transaction Manager:**

  It is responsible for making the database consistent before and after the transactions. Concurrent processes are generally controlled by this component.

- **Buffer Manager:**

  The transfer of data between primary and main memory and managing the cache memory is done by the buffer manager.

**3. Disk Storage**

A DBMS can use various kinds of Data Structures as a part of physical system implementation in the form of disk storage.

**Components of Disk Storage**

- **Data Dictionary:**

It contains the metadata (data of data), which means each object of the database has some information about its structure. So, it contains the details about the structure of the database object.

- **Data Files:**

  Stores the data in file

- **Indices:**

  These indices are used to access and retrieve the data in a very fast and efficient way.

## 1.5 Entity Attributes and Types of Relationship:

### Entity

- An **entity** in DBMS is used to represent any real-world object,
- an entity is a piece of data that is stored in the database. An entity can be a person, place, thing, or even an event.
  There are two types of entities in DBMS: strong and weak.

### 1. Strong Entity Type with Example

- A strong entity in DBMS is an independent table that doesn't rely on any other tables for its existence. A primary key uniquely identifies each row in the table, and foreign keys are used to relate this table to other tables.

- A simple example of a strong entity type would be "customer" in a customer relational database table. The customerID attribute would be the primary key (and it can't have duplicate values or be NULL), and other information about the customer would be stored in separate attributes such as "firstname," "last name," etc.

### 2. Weak Entity Type with Example

- A weak entity type is a dependent table that relies on another table for its existence; it has no meaningful attributes of its own except for the foreign key from the parent table.

- For a weak entity type to exist, it must have some relationship with another (parent) table; otherwise, it wouldn't appear in the database!

- A typical example of a weak entity type is an "order." It has no meaning by itself–it must be placed by some customer so it has a foreign key relation to the "customer" table–but it has several attributes of its own such as orderID, productID, etc.

**Attributes**

- each entity contains some property about their behavior which is also called the attribute.

- In relational databases, we have tables, and each column contains some attributes, so all the entries for that column should strictly follow the attribute of the entity.

**Types of Attributes**

**Simple Attribute:**

- It is also known as atomic attributes. When an attribute cannot be divided further, then it is called a simple attribute.
- For example, in a student table, the branch attribute cannot be further divided.

**Composite Attribute:**

- Composite attributes are those that are made up of the composition of more than one attribute. When any attribute can be divided further into more sub-attributes, then that attribute is called a composite attribute.
- For example, in a student table, we have attributes of student names that can be further broken down into first name, middle name, and last name. So the student name will be a composite attribute.
- Another example from a personal detail table would be the attribute of address. The address can be divided into a street, area, district, and state.

**Single-valued Attribute:**

- Those attributes which can have exactly one value are known as single valued attributes. They contain singular values, so more than one value is not allowed.
- For example, the DOB of a student can be a single valued attribute. Another example is gender because one person can have only one gender.

**Multi-valued Attribute:**

- Those attributes which can have more than one entry or which contain more than one value are called multi valued attributes
- In the Entity Relationship (ER) diagram, we represent the multi valued attribute by double oval representation.
- For example, one person can have more than one phone number, so that it would be a multi valued attribute. Another example is the hobbies of a person because one can have more than one hobby.

**Derived Attribute:**

- Derived attributes are also called stored attributes. When one attribute can be derived from the other attribute, then it is called a derived attribute. We can do some calculations on normal attributes and create derived attributes.
- For example, the age of a student can be a derived attribute because we can get it by the DOB of the student.
- Another example can be of working experience, which can be obtained by the date of joining of an employee.
- In the ER diagram, we represent the derived attributes by a dotted oval shape.

**Complex Attribute:**

- If any attribute has the combining property of multi values and composite attributes, then it is called a complex attribute.
- It means if one attribute is made up of more than one attribute and each attribute can have more than one value, then it is called a complex attribute.
- For example, if a person has more than one office and each office has an address made from a street number and city. So the address is a composite attribute, and offices are multi valued attributes, So combing them is called complex attributes.

**Key Attribute:**

Those attributes which can be identified uniquely in the relational table are called key attributes.

- For example, a student_ID, is a unique attribute.

## Types of Relationships:

A relationship represents the association between two are more entities

### 1. Unary relationship

When there is only one entity set participating in a relationship then such type of relationship is called unary relationship

**Example of unary relationship**

For example, a person has **only one passport and only one passport is given to only one person** and hence unary relationship is observed

### 2. Binary relationship

When there are exactly two entity sets participating in a relationship then such type of relationship is called binary relationship

**Example of binary relationship**

For example, a **teacher teaches a subject** here 2 entities are teacher and subject for the relationship teacher teaches subject

### 3. N-ary relationship

When a large number of entity sets are participating in a relationship, then such type of relationship is called an n-ary relationship

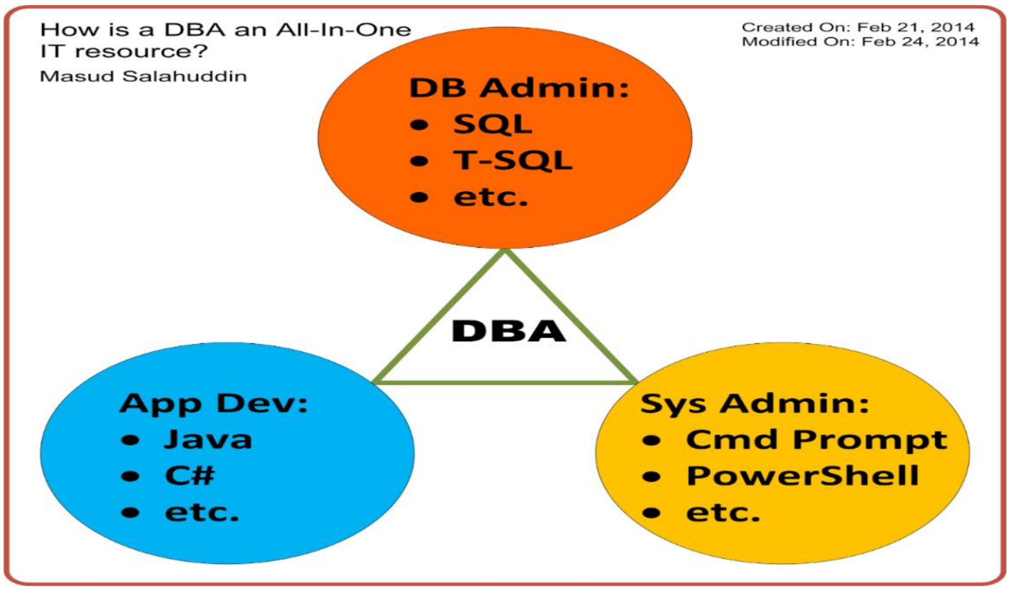
**Example of n-ary Relationship**

In the real world, a patient goes to a doctor and doctor prescribes the medicine and diagnosis to the patient, **four entities Doctor, patient and medicine, diagnostics are involved in the relationship "prescribes".**

## 1.6 DBMS USERS:

Database users are categorized based up on their interaction with the database. These are seven types of database users in DBMS.

**Database Administrator (DBA):**



- Database Administrator (DBA) is a person/team who defines the schema and also controls the 3 levels of database.
- The DBA will then create a new account id and password for the user if he/she need to access the database.
- DBA is also responsible for providing security to the database and he allows only the authorized users to access/modify the data base. DBA is responsible for the problems such as security breaches and poor system response time.
- DBA also monitors the recovery and backup and provide technical support.
- DBA repairs damage caused due to hardware and/or software failures.

**Naive / Parametric End Users:**

- Parametric End Users are the unsophisticated who don't have any DBMS knowledge but they frequently use the database applications in their daily life to get the desired results. For examples, Railway's ticket booking users are naive users.

- Clerks in any bank is a naive user because they don’t have any DBMS knowledge but they still use the database and perform their given task.

**System Analyst:**

- System Analyst is a user who analyzes the requirements of parametric end users. They check whether all the requirements of end users are satisfied.

**Sophisticated Users:**

- Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database.
- They can develop their own database applications according to their requirement.
- They don’t write the program code but they interact the database by writing SQL queries directly through the query processor.

**Database Designers:**

- Data Base Designers are the users who design the structure of database which includes tables, indexes, views, triggers, stored procedures and constraints which are usually enforced before the database is created or populated with data. He/she controls what data must be stored and how the data items to be related.
- It is responsibility of Database Designers to understand the requirements of different user groups and then create a design which satisfies the need of all the user groups.

**Application Programmers:**

- Application Programmers also referred as System Analysts or simply Software Engineers, are the back-end programmers who writes the code for the application programs.
- They are the computer professionals. These programs could be written in Programming languages such as Visual Basic, Developer, C, FORTRAN, COBOL etc. Application programmers design, debug, test, and maintain set of programs called “canned transactions” for the Naive (parametric) users in order to interact with database.

**Casual Users / Temporary Users:**

- Casual Users are the users who occasionally use/access the database but each time when they access the database they require the new information, for example, Middle or higher level manager.

**Specialized users:**

- Specialized users are sophisticated users who write specialized database application that does not fit into the traditional data-processing framework.
- Among these applications are computer aided-design systems, knowledge-base and expert systems etc.

## 1.7 <u>Features Provided by DBMS</u>:

- **Data Organization:** DBMS organizes data in a structured way, such as tables, schemas and records making it easier to access and manage.
- **Data Security:** DBMS provides features for data security, such as user authentication and access control, to protect the data from unauthorized access.
- **Data Consistency:**
  - Data consistency refers to the usefulness of the data. Data Consistency in DBMS is the state of data that is consistent across all systems.
  - DBMS ensures data consistency by enforcing rules and constraints, such as primary keys and foreign keys, to maintain the integrity of the data.
- **Data Concurrency:** DBMS allows multiple users to access and manipulate the data concurrently(at same time), while ensuring data consistency and isolation.
- **Data Backup and Recovery:** DBMS provides features for data backup and recovery, such as creating backups and restoring data in case of failures.
- **Data Independence:** DBMS provides data independence, which separates the physical storage of data from the logical view of the data, making it easier to change the physical storage without affecting the applications that use the data.
- **Data Retrieval and Manipulation:** DBMS provides a standardized query language, such as SQL, to retrieve and manipulate data in the database.
- **Data Integration:**
  **-** Data integrity refers to the quality of the data. This is the state of accurate and complete data.
  - DBMS allows for the integration of data from multiple sources, making it easier to manage and analyze the data.

## 1.8 Advantage of DBMS:

- **Data Redundancy:** DBMS eliminates data redundancy by organizing and structuring the data in a manner that ensures easy retrieval.

- **Data Modification:** DBMS simplifies data modification by allowing the addition, removal, or modification of data elements without affecting the existing data structure.
- **Data Integrity:** DBMS maintains data integrity by applying constraints and ensuring data accuracy, consistency, and reliability.

- **User-Friendly Interface**: DBMS provides user-friendly interfaces for accessing and managing data, which allows users to interact with the data without requiring in-depth technical knowledge.
- **Access Control**: DBMS provides access control features that enable the creation and enforcement of rules for controlling who can access what data.
- **Database Recovery:** DBMS supports efficient database recovery, allowing for the recovery of data even in case of system crashes or power failures.
- **Performance Optimization:** DBMS offers features for optimizing the performance of the database system by identifying and resolving performance issues.

### <u>Disadvantage of DBMS</u>:

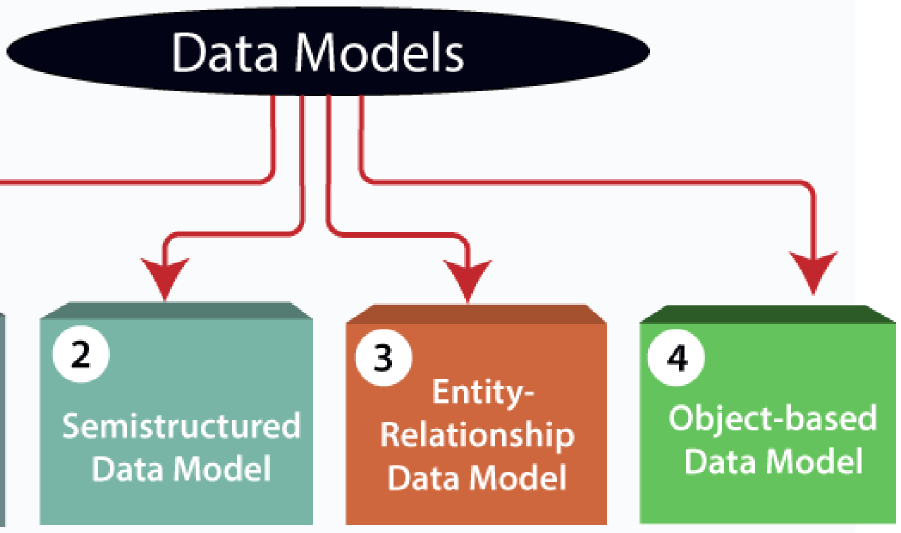
**Limitations of DBMS:**

a) **Cost:** Implementing a DBMS can be expensive, particularly for small-scale operations or organizations with limited budgets.
b) **Complexity*:*** DBMS can be complex and may require specialized knowledge to implement, maintain, and use effectively**.**
c) **Compatibility Issues*:*** Different DBMS systems may not be fully compatible with each other, which can limit the flexibility and scalability of a system.
d) **Slower Data Retrieval:** DBMS can sometimes introduce overhead that may slow down data retrieval, particularly for large databases or complex queries.

## 1.9 Data Models in DBMS

- **Data models** in DBMS help to understand the **design** at the conceptual, physical, and logical levels as <u>it provides a clear picture of the data making it easier</u> for developers to create a physical database.

- Data models are used to describe how the data is stored, accessed, and updated in a DBMS.
- A set of symbols and text is used to represent them so that all the members of an organization can understand how the data is organized.


Data Models
1 Relational Data Model
2 Semistructured Data Model
3 Entity-Relationship Data Model
4 Object-based Data Model

-

**1)Relational Data Model:**

- This type of model designs the data in the form of **rows** and **columns** within a table. Thus, a relational model uses **tables** for representing data and in-between relationships.
- **Tables are also called relations**. This model was initially described by Edgar F. Codd, in 1969.
- The relational data model is the widely used model which is primarily used by commercial data processing applications.

| Stu. Id | Name | Branch |
|---|---|---|
| 101 | Naman | CSE |
| 102 | Saloni | ECE |
| 103 | Rishabh | IT |
| 104 | Pulkit | ME |

**2) Entity-Relationship Data Model:**

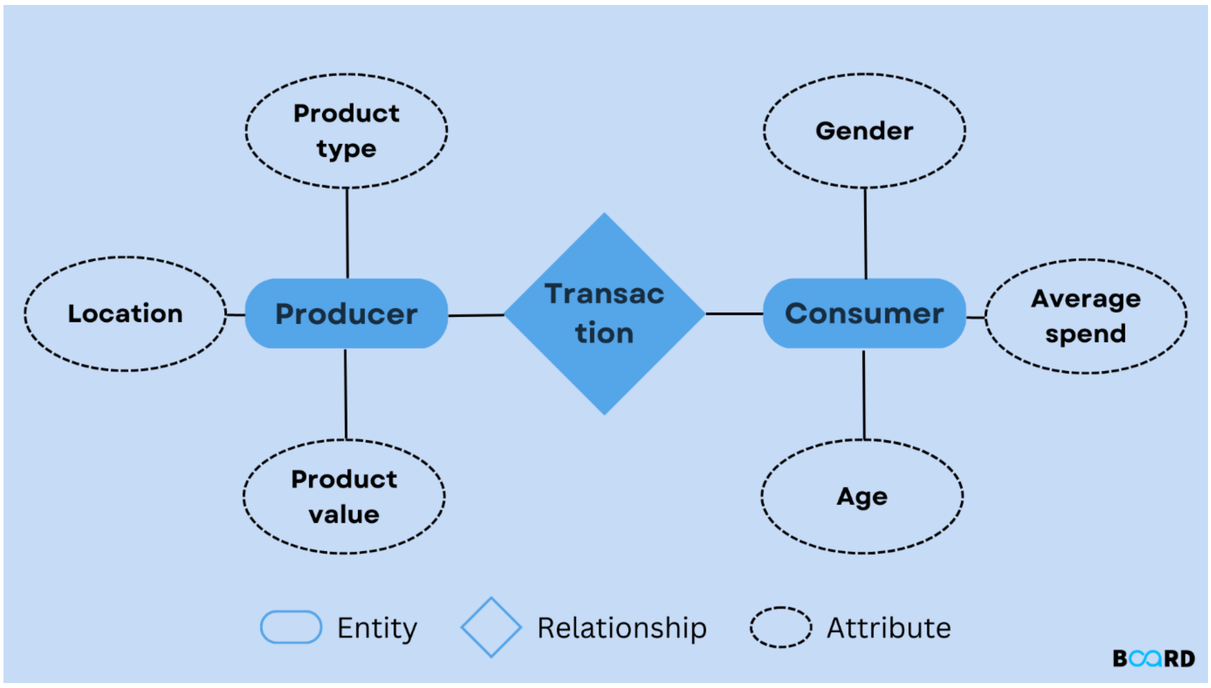
- **An Entity-Relationship model is a high-level data model that describes the <u>structure of the database</u> in a pictorial form which is known as ER-diagram.**
- In simple words, an ER diagram is used to represent logical structure of the database easily.

- ER model develops a conceptual view of the data hence it can be used as a blueprint to implement the database in the future.
- Developers can easily understand the system just by looking at ER diagram. Let's first have a look at the components of an ER diagram.

- **Entity -** Anything that has an independent existence about which we collect the data. To learn more about **Entity in DBMS** click here.

They are represented as rectangles in the ER diagram. For example - Car, house, employee.

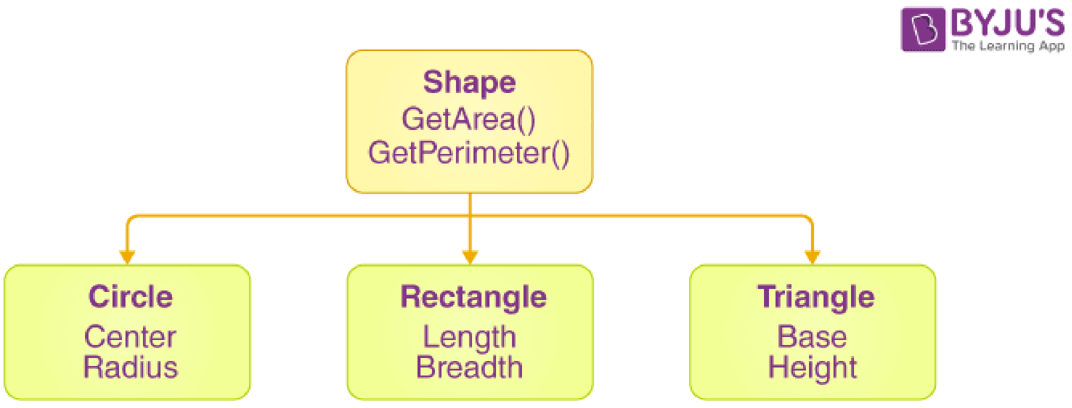
- **Entity Set -** A set of the same type of entities is known as an entity set. For example - Set of students studying in a college.
- **Attributes -** Properties that define entities are called attributes. They are represented by an ellipse shape.
- **Relationships -** A relationship in DBMS is used to describe the association between entities. They are represented as diamond or rhombus shapes in the ER diagram.



**3) Object-based Data Model:**

- The data and data relationship is stored together in a single entity known as an object in the Object Oriented Model.
- This model supports a rich type system that includes structured and collection types. Thus, in 1980s, various database systems

following the object-oriented approach were developed. Here, the objects are nothing but the data carrying its properties.



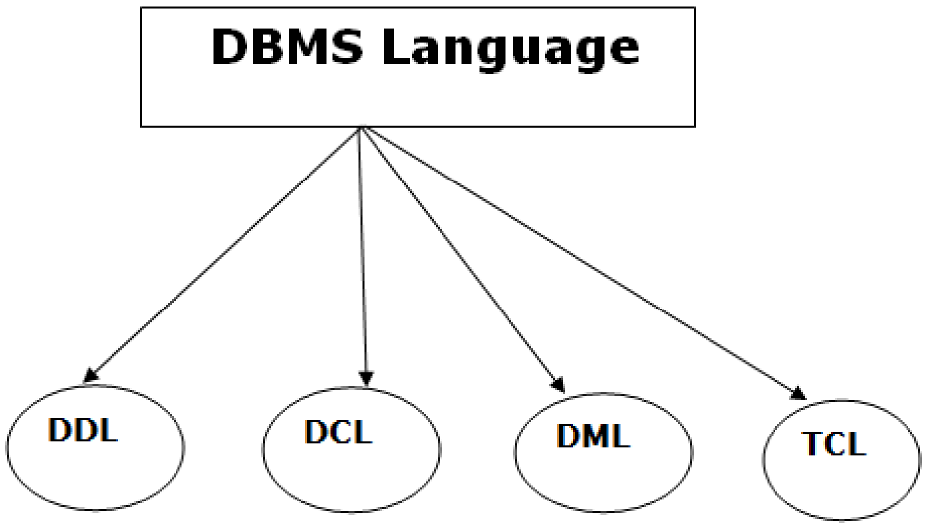
**4) Semi Structured Data Model:**

- This type of data model is different from the other three data models. The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets.
- The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data.
- In this model some entities may have missing attributes while others may have an extra attributes.

## 1.10 Database Languages

- A DBMS has appropriate languages and interfaces to express database queries and updates.
- Database languages can be used to read, store and update the data in the database.

**Types of Database Languages**



## 1. Data Definition Language (DDL)

- **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

- **Create:** It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

## 2. Data Manipulation Language (DML)

**DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

**3. Data Control Language (DCL)**

- **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.

  (But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

**4. Transaction Control Language (TCL)**

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

- **Commit:** It is used to save the transaction on the database.

- **Rollback:** It is used to restore the database to original since the last Commit.

**Assignment 1:**

**Q1.**