UNIT I

1. DBMS Concepts

What is Data?

Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.

In simple words, data can be facts related to any object in consideration. For example, your name, age, height, weight, etc. are some data related to you. A picture, image, file, pdf, etc. can also be considered data.

What is Database?

A **database** is an organized collection of data, so that it can be easily accessed and managed.

You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

Database handlers create a database in such a way that only one set of software program provides access of data to all the users.

The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data.

There are many **databases available** like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

SQL or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

a database example: An online telephone directory uses a database to store data of people, phone numbers, and other contact details. Your electricity service provider uses a database to manage billing, client-related issues, handle fault data, etc.

Let us also consider Facebook. It needs to store, manipulate, and present data related to members, their friends, member activities, messages, advertisements, and a lot more. We can provide a countless number of examples for the usage of databases.

DBMS (Data Base Management System)

Database management System is software which is used to store and retrieve the database. For example, Oracle, MySQL, etc.; these are some popular DBMS tools.

- DBMS provides the interface to perform the various operations like creation, deletion, modification, etc.
- DBMS allows the user to create their databases as per their requirement.
- DBMS accepts the request from the application and provides specific data through the operating system.
- DBMS contains the group of programs which acts according to the user instruction.
- It provides security to the database.

Database Management Systems (DBMS) are software systems used to store, retrieve, and run queries on data. A DBMS serves as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database.

DBMS manage the data, the database engine, and the database schema, allowing for data to be manipulated or extracted by users and other programs. This helps provide data security, data integrity, concurrency, and uniform data administration procedures.

Let us see a simple example of a university database. This database is maintaining information concerning students, courses, and grades in a university environment. The database is organized as five files:

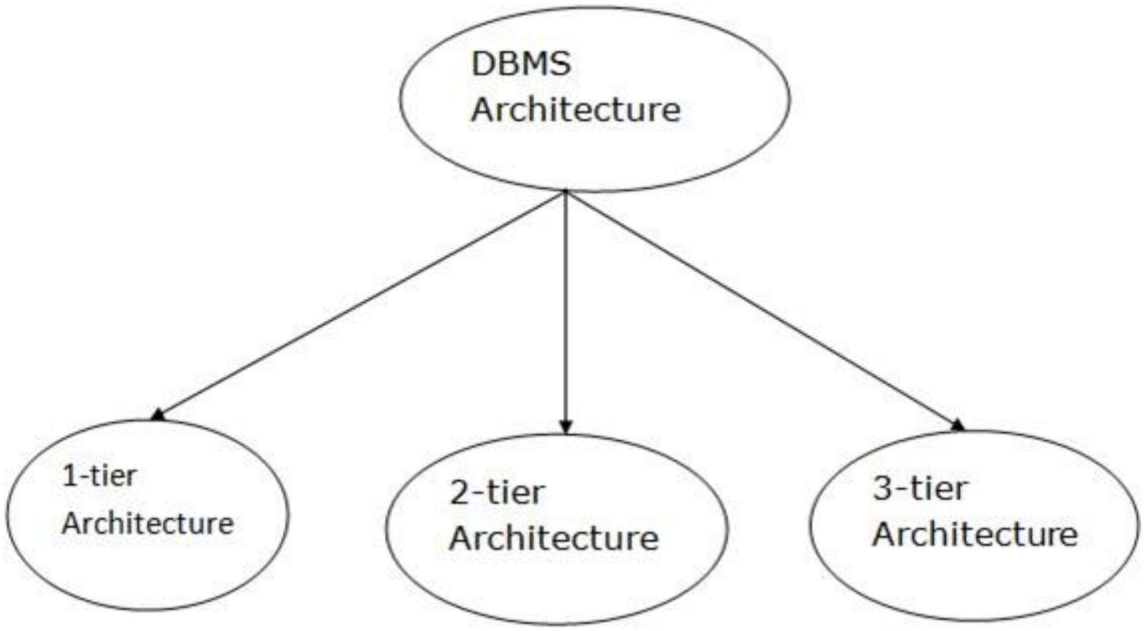
- The STUDENT file stores data of each student
- The COURSE file stores contain data on each course.

- The SECTION stores the information about sections in a particular course.
- The GRADE file stores the grades which students receive in the various sections

DBMS Architecture

- The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected via the network.
- DBMS architecture depends upon how users are connected to the database to get their request done.

Types of DBMS Architecture



Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: **2-tier architecture** and **3-tier architecture**.

1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

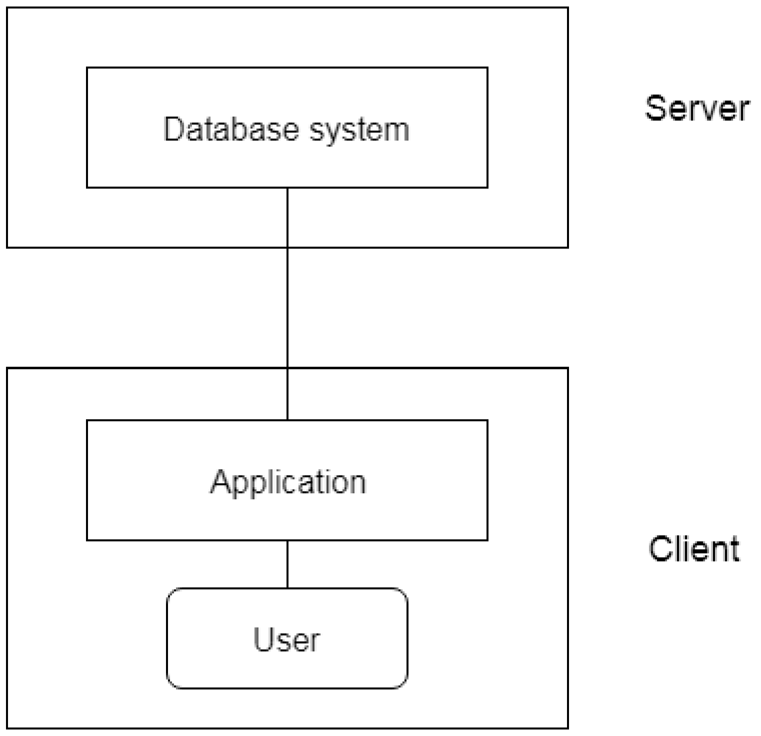


Fig: 2-tier Architecture

3-Tier Architecture

- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

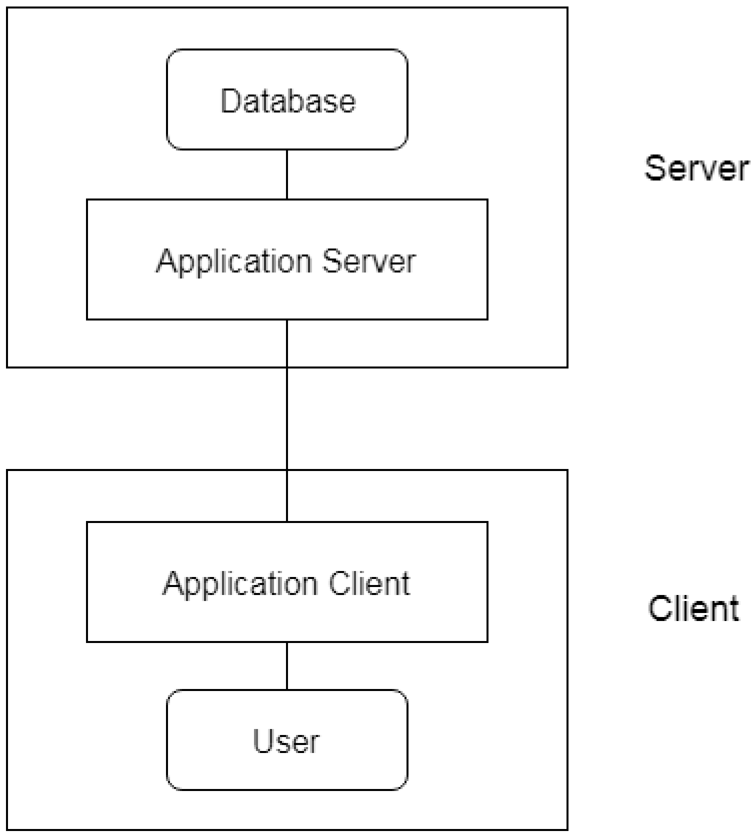
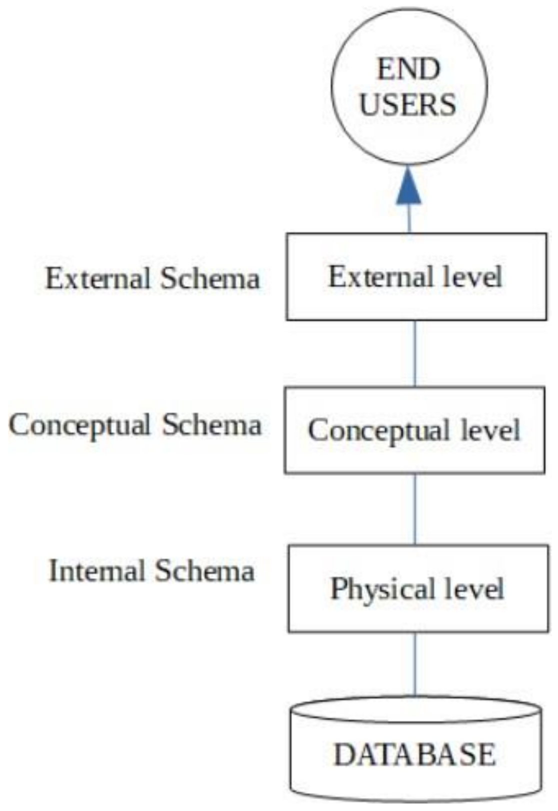


Fig: 3-tier Architecture

Three schema Architecture

- database architecture is the basis of most of the modern databases.
- The three levels present in this architecture are Physical level, Conceptual level and External 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 in the form of bits and at a little high level,
- it can be said that the data is stored in files and folders. The physical level also discusses compression and encryption techniques.
- The internal level has an internal schema which describes the physical storage structure of the database.
- The internal schema is also known as a physical schema.
- It uses the physical data model. It is used to define that how the data will be stored in a block.

- **Conceptual Level**
- The conceptual level is at a higher level than the physical level. It is also known as the logical level.
- The conceptual level does not care for how the data in the database is actually stored.
- The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
- The conceptual schema describes the structure of the whole database.
- The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
- In the conceptual level, internal details such as an implementation of the data structure are hidden.
- Programmers and database administrators work at this level.
-
- **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.

Data Models

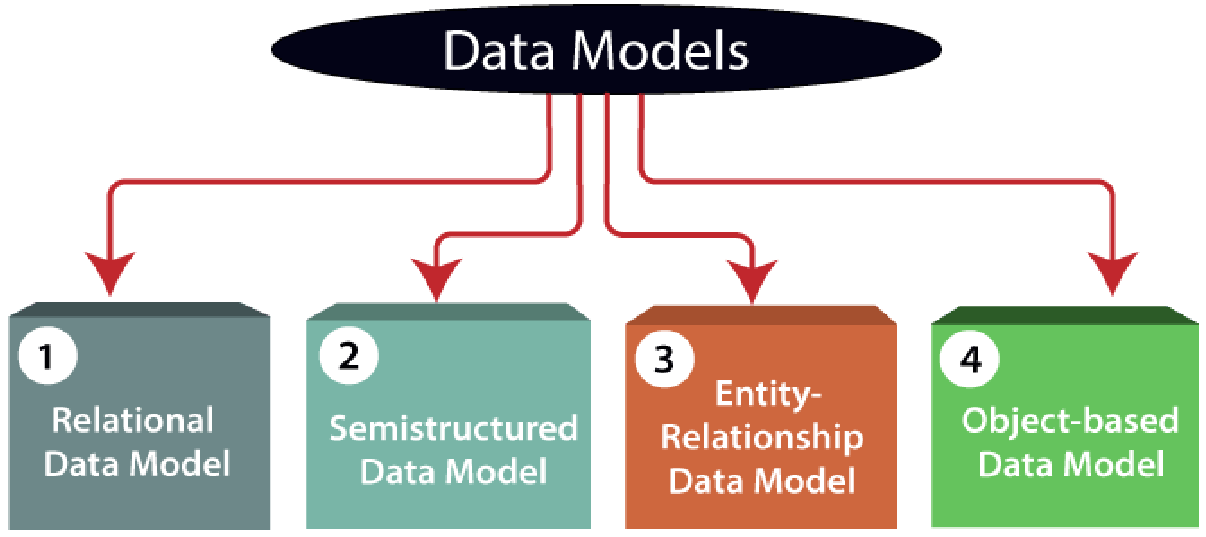
Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction.

Data Model gives us an idea that how the final system will look like after its complete implementation.

It defines the data elements and the relationships between the data elements. Data Models are used to show how data is stored, connected, accessed and updated in the database management system.

Here, we use a set of symbols and text to represent the information so that members of the organisation can communicate and understand it.

Therefore, there are following four data models used for understanding the structure of the database:


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.

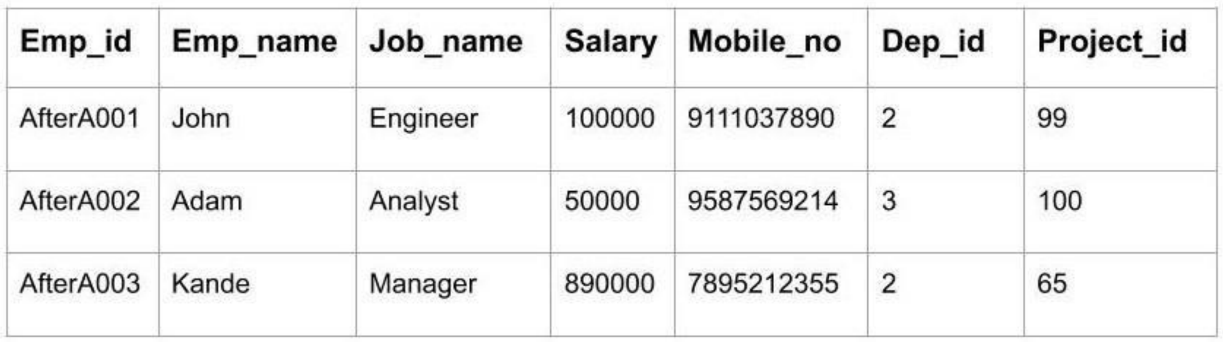
Thus, a relational model uses tables for representing data and in-between relationships. Tables are also called relations. This model was initially described by Edgar F. Codd, in 1969.

The relational data model is the widely used model which is primarily used by commercial data processing applications.

Relational Model is the most widely used model. In this model, the data is maintained in the form of a two-dimensional table.

All the information is stored in the form of row and columns.

The basic structure of a relational model is tables. So, the tables are also called relations in the relational model. Example: In this example, we have an Employee table.



Emp_id	Emp_name	Job_name	Salary	Mobile_no	Dep_id	Project_id
AfterA001	John	Engineer	100000	9111037890	2	99
AfterA002	Adam	Analyst	50000	9587569214	3	100
AfterA003	Kande	Manager	890000	7895212355	2	65

EMPLOYEE TABLE

Features of Relational Model

- **Tuples:** Each row in the table is called tuple. A row contains all the information about any instance of the object. In the above example, each row has all the information about any specific individual like the first row has information about John.
- **Attribute or field:** Attributes are the property which defines the table or relation. The values of the attribute should be from the same domain. In the above example, we have different attributes of the employee like Salary, Mobile_no, etc.

2) Entity-Relationship Data Model: An ER model is the logical representation of data as objects and relationships among them.

These objects are known as entities, and relationship is an association among these entities. This model was designed by Peter Chen and published in 1976 papers.

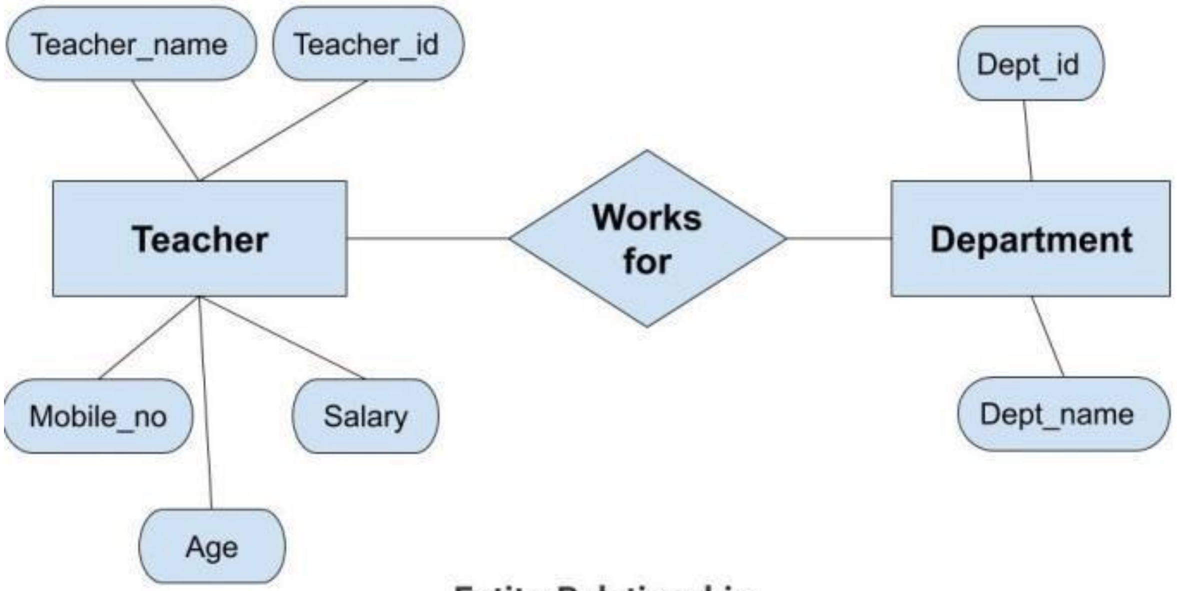
It was widely used in database designing. A set of attributes describe the entities.

For example, student_name, student_id describes the 'student' entity. A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as 'relationship set'.

- **Entities:** Entity is a real-world thing. It can be a person, place, or even a concept. Example: Teachers, Students, Course, Building, Department, etc are some of the entities of a School Management System.

- **Attributes:** An entity contains a real-world property called attribute. This is the characteristics of that attribute. Example: The entity teacher has the property like teacher id, salary, age, etc.
- **Relationship:** Relationship tells how two attributes are related. Example: Teacher works for a department.

Example:



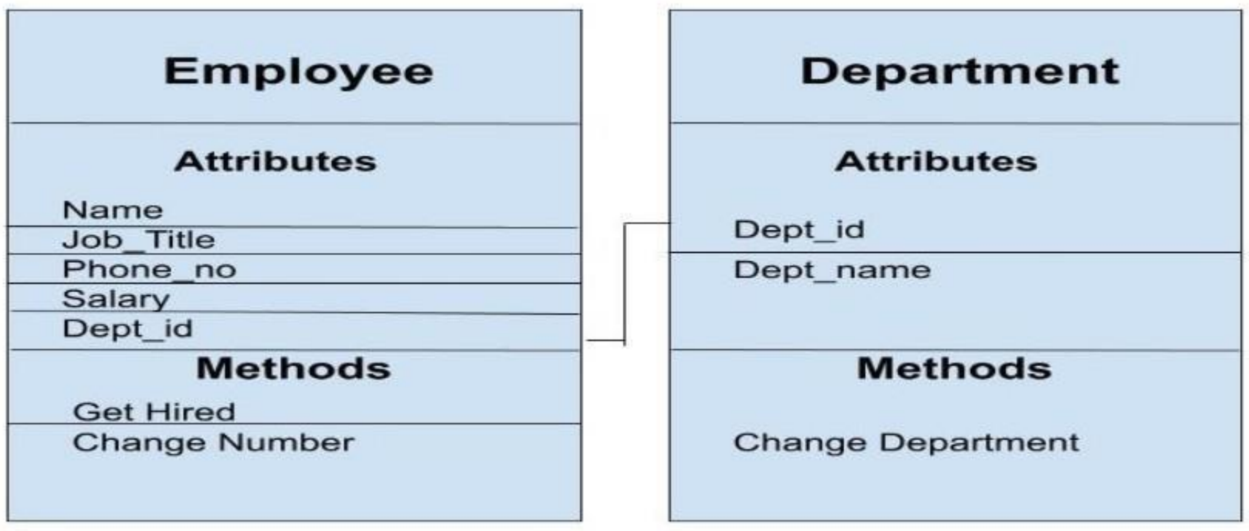
Entity-Relationship Model

In the above diagram, the entities are Teacher and Department. The attributes of Teacher entity are Teacher_Name, Teacher_id, Age, Salary, Mobile_Number. The attributes of entity Department entity are Dept_id, Dept_name. The two entities are connected using the relationship. Here, each teacher works for a department.

3) Object-based Data Model: An extension of the ER model with notions of functions, encapsulation, and object identity, as well. 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.

In this model, two are more objects are connected through links. We use this link to relate one object to other objects. This can be understood by the example given below.



Object_Oriented_Model

The two objects are connected through a common attribute i.e the Department_id and the communication between these two will be done with the help of this common id.

4) Semistructured Data Model:

This type of data model is different from the other three data models (explained above).

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.

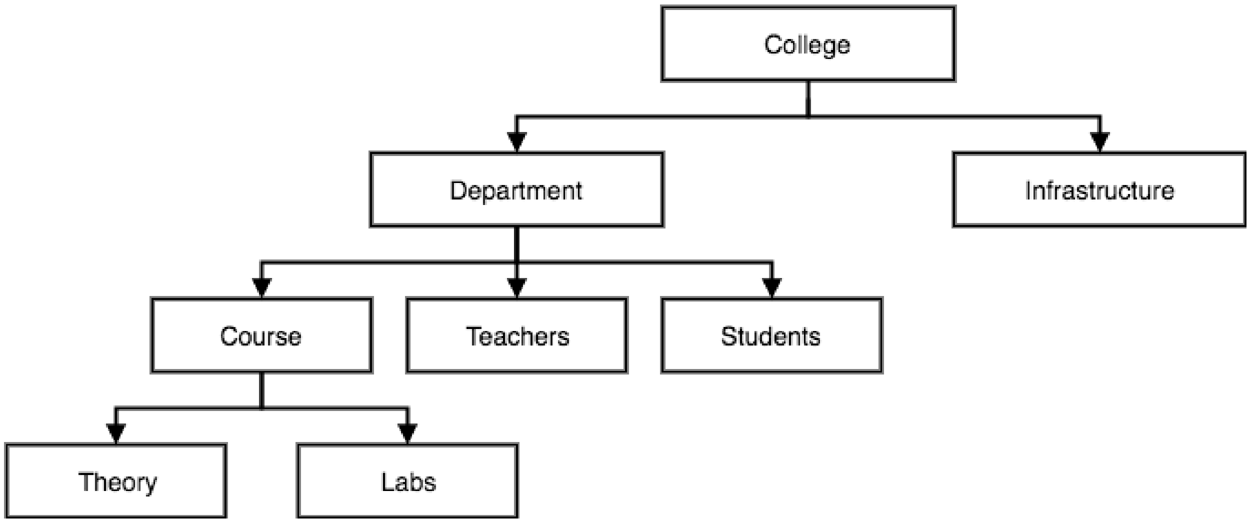
Hierarchical Model

This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked. The heirarchy starts from the Root data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

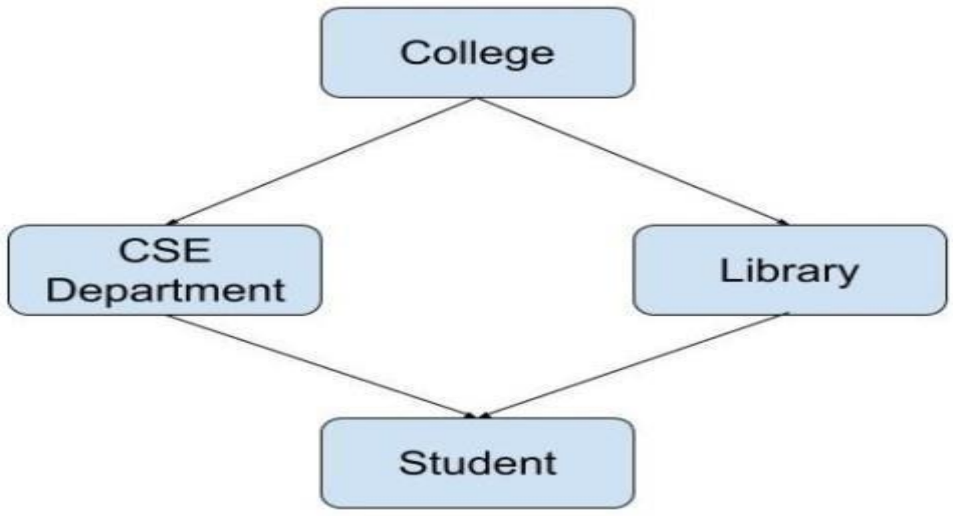
This model efficiently describes many real-world relationships like index of a book, recipes etc.

In hierarchical model, data is organised into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.



Network Model

This model is an extension of the hierarchical model. It was the most popular model before the relational model. This model is the same as the hierarchical model, the only difference is that a record can have more than one parent. It replaces the hierarchical tree with a graph. Example: In the example below we can see that node student has two parents i.e. CSE Department and Library. This was earlier not possible in the hierarchical model.



Network Model

Structure of a DBMS

A database system is partitioned into modules that deal with each of the responsibilities of the overall system.

The functional components of a database system can be broadly divided into the **storage manager** and the **query processor** components.

The storage manager is important because databases typically require a large amount of storage space.

The query processor is important because it helps the database system simplify and facilitate access to data.

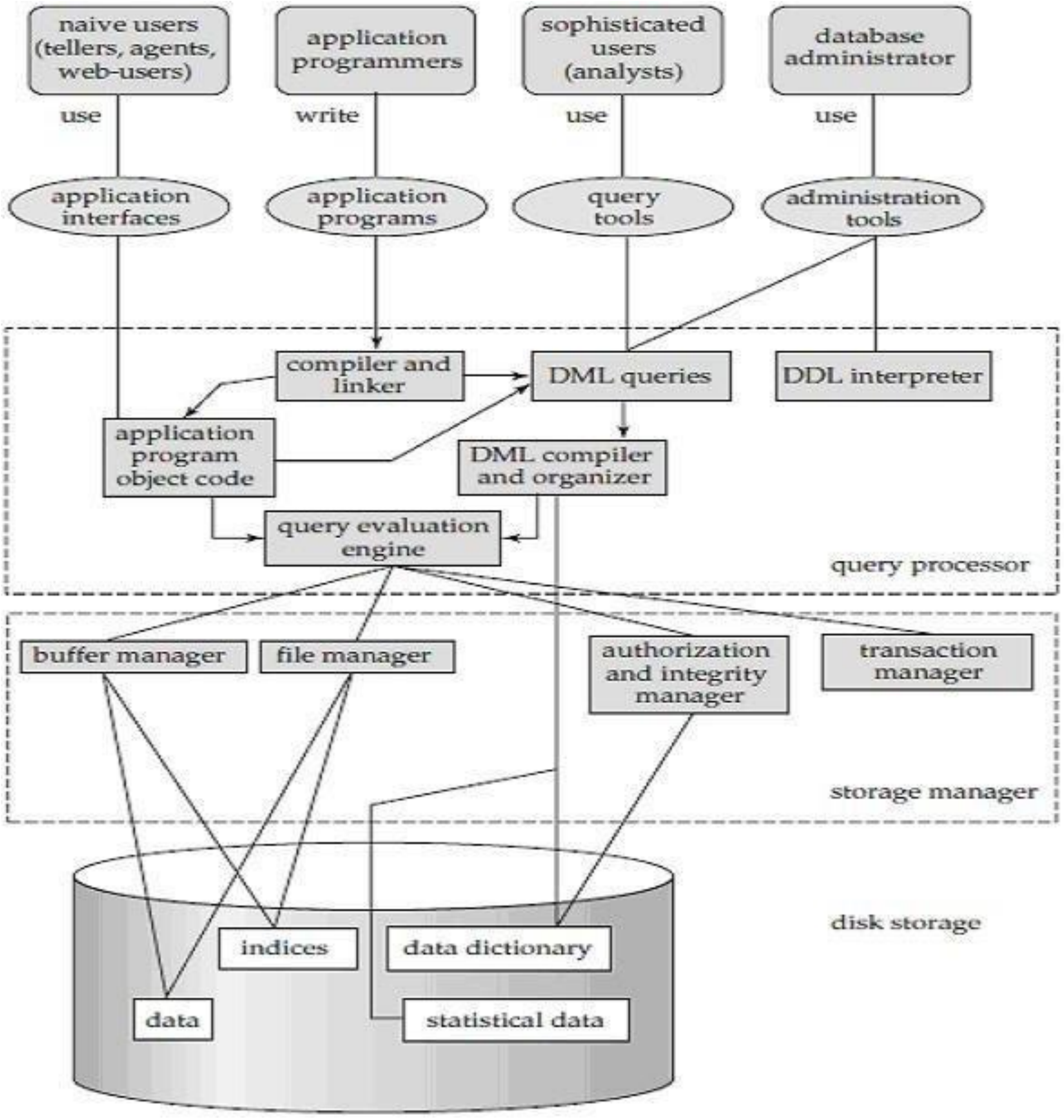
A database system is partitioned into modules that deal with each of the responsibilities of the overall system.

The functional components of a database system can be broadly divided into the storage manager and the query processor components.

The storage manager is important because databases typically require a large amount of storage space.

The query processor is important because it helps the database system to simplify and facilitate access to data.

It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level



It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.

Query Processor

The query processor components include

- **DDL interpreter,** which interprets DDL statements and records the definitions in the data dictionary.

- **DML compiler,** which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs **query optimization**, that is, it picks the lowest cost evaluation plan from among the alternatives.

- **Query evaluation engine,** which executes low-level instructions generated by the DML compiler.

Storage Manager

A *storage manager* is a program module that provides the interface between the lowlevel data stored in the database and the application programs and queries submitted to the system.

The storage manager is responsible for the interaction with the file manager. The raw data are stored on the disk using the file system, which is usually provided by a conventional operating system.

The storage manager translates the various DML statements into low-level file-system commands. Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

The storage manager components include:

- **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.

- **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.

- **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

- **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

Transaction Manager

A **transaction** is a collection of operations that performs a single logical function in a database application.

Each transaction is a unit of both atomicity and consistency. Thus, we require that transactions do not violate any database-consistency constraints.

That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates.

Transaction - manager ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

About Disk Storage:

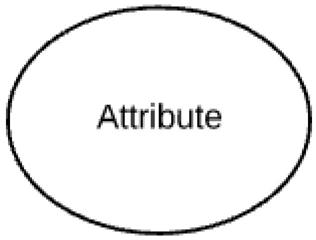
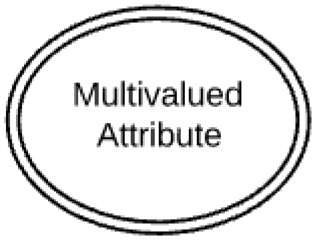
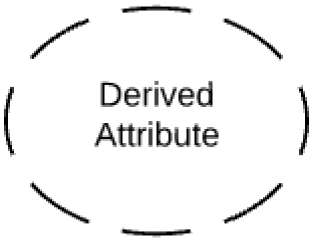
1. **Data Files:** It stores the Data.
2. **Data Dictionary**: It contains information about the structure of database object.
3. **Indices**: It provide faster retrieval of data item.

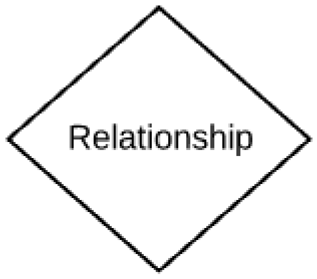
What is an Entity Relationship Diagram (ER Diagram)?

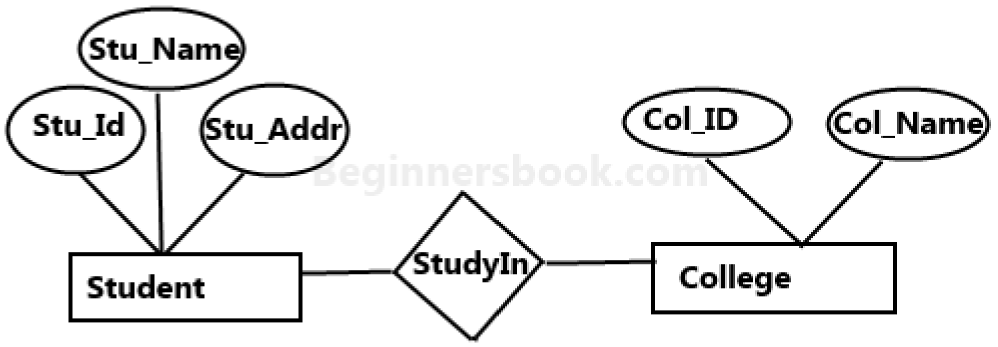
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes.

In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

Lets have a look at a simple ER diagram to understand this concept.

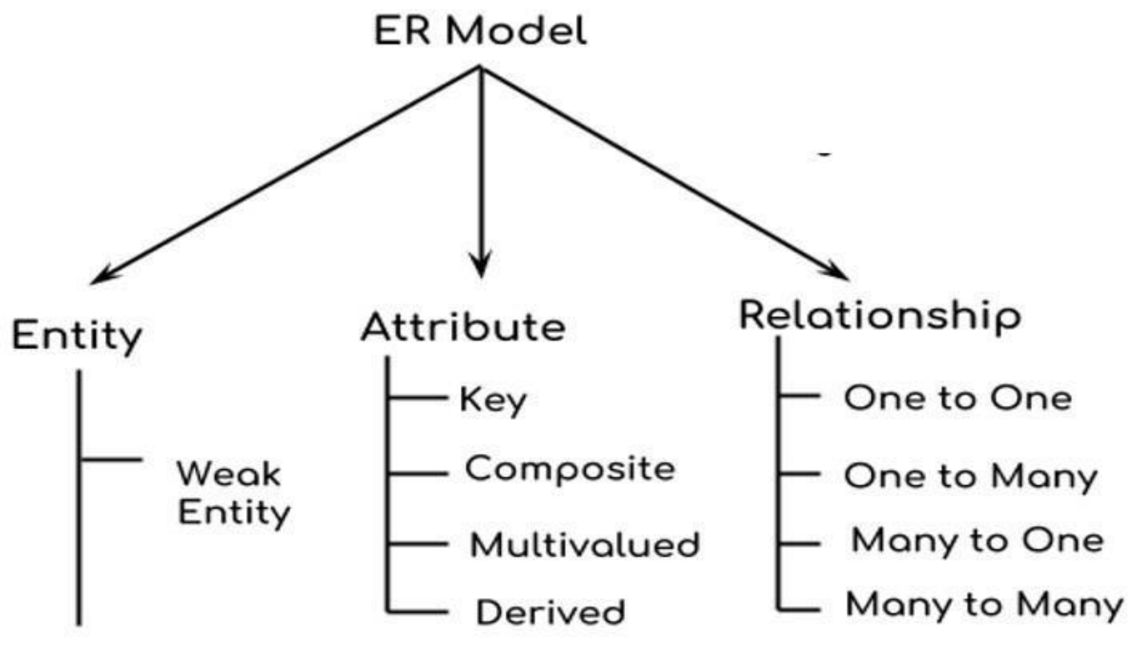
Attribute Symbol	Name	Description
	Attribute	Attributes are characteristics of an entity, a many-to-many relationship, or a one-to-one relationship.
	Multivalued attribute	Multivalued attributes are those that are can take on more than one value.
	Derived attribute	Derived attributes are attributes whose value can be calculated from related attribute values.

Attribute Symbol	Name	Description
	Relationship	Relationships are associations between or among entities.



Sample E-R Diagram

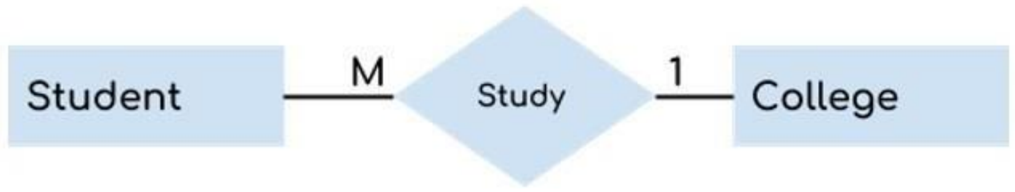
In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name.



Components of ER Diagram

As shown in the above diagram, an ER diagram has three main components:
1. Entity
2. Attribute
3. Relationship

1. Entity
An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.
For example: In the following ER diagram we have two entities Student and College and these two entities have many to one relationship as many students study in a single college. We will read more about relationships later, for now focus on entities.



Weak Entity:
An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle.

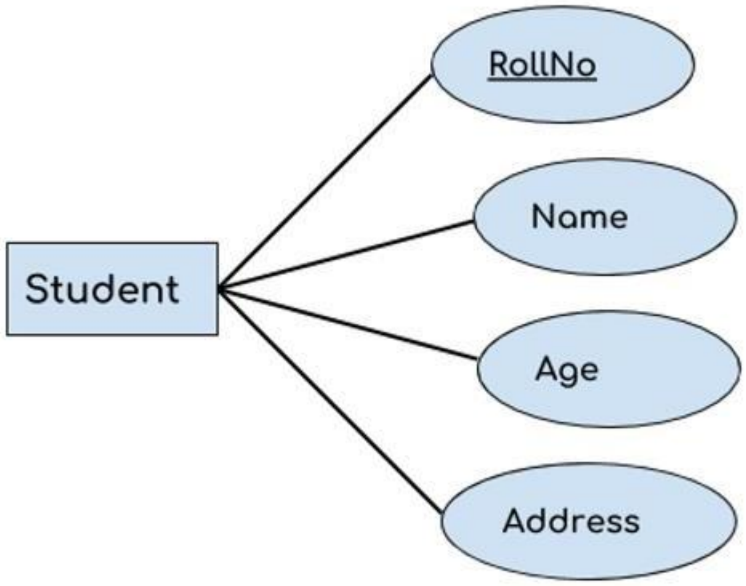
For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.

2. Attribute

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

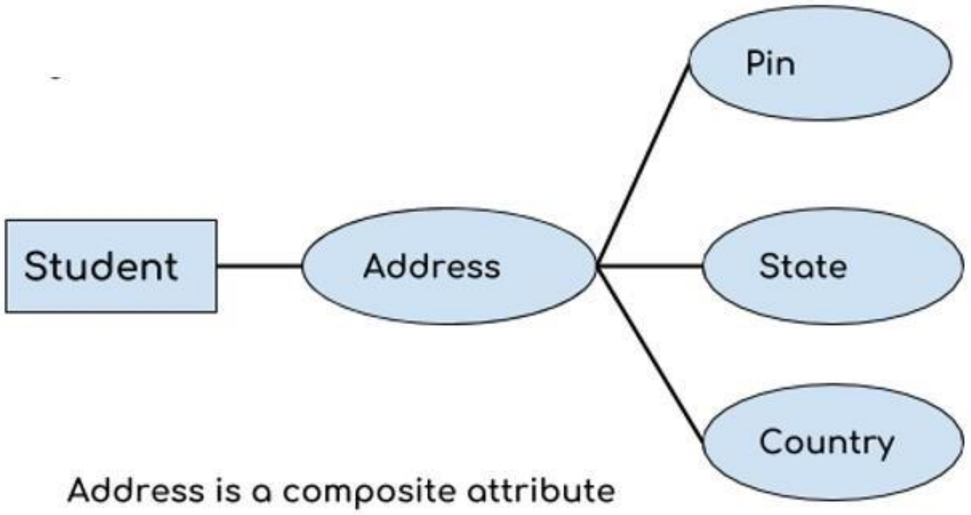
1. Key attribute
2. Composite attribute
3. Multivalued attribute
4. Derived attribute

1. Key attribute:



A key attribute can uniquely identify an entity from an entity set. For example, student roll number can uniquely identify a student from a set of students. Key attribute is represented by oval same as other attributes however the **text of key attribute is underlined**.

2. Composite attribute:



Address is a composite attribute

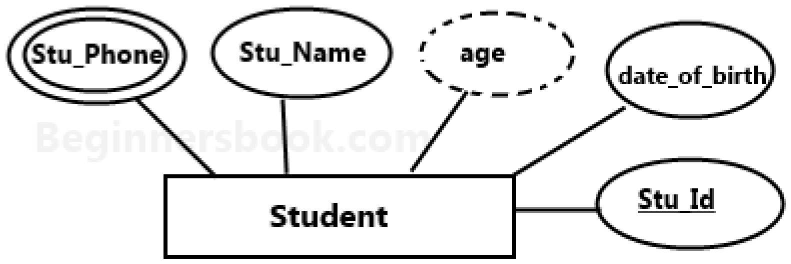
An attribute that is a combination of other attributes is known as composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.

3. Multivalued attribute:

An attribute that can hold multiple values is known as multivalued attribute. It is represented with **double ovals** in an ER Diagram. For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

4. Derived attribute:

A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by **dashed oval** in an ER Diagram. For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).



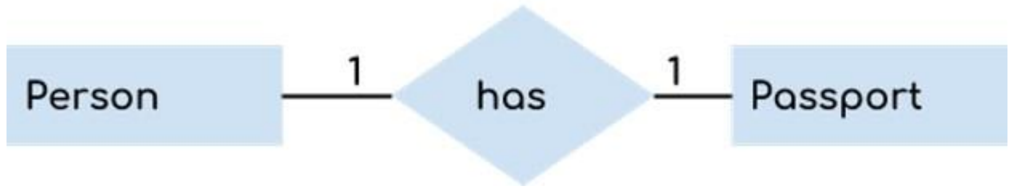
E-R diagram with multivalued and derived attributes:

3. Relationship

A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

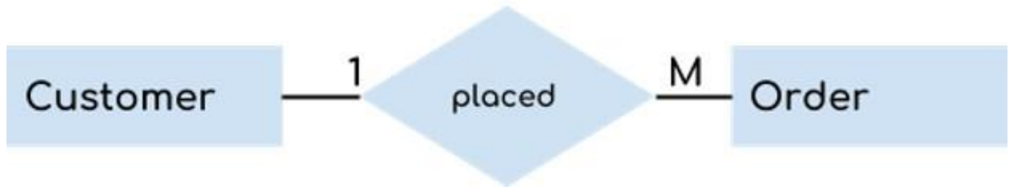
1. One to One Relationship



When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship.

For example, a person has only one passport and a passport is given to one person.

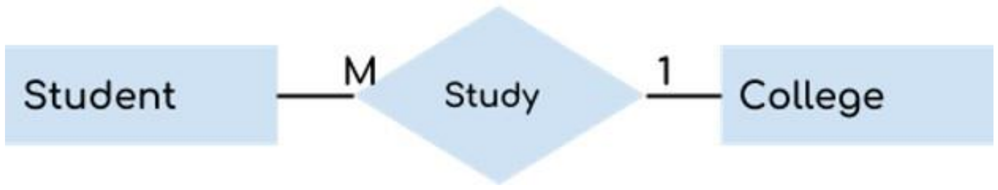
2. One to Many Relationship



When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship.

For example – a customer can place many orders but a order cannot be placed by many customers.

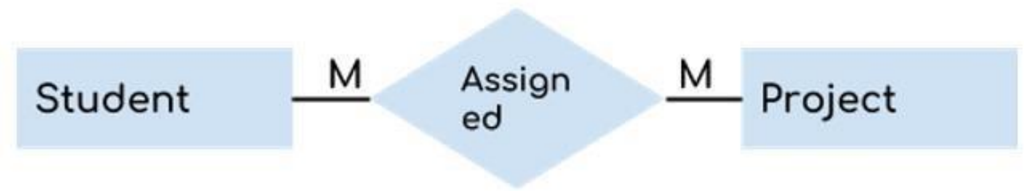
3. Many to One Relationship



When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship.

For example – many students can study in a single college but a student cannot study in many colleges at the same time.

4. Many to Many Relationship



When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship.

For example, a can be assigned to many projects and a project can be assigned to many students.

Different types of Database Users

1. **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 data base.
   DBA is also responsible for providing security to the data base and he allows only the authorized users to access/modify the data base.
   - DBA also monitors the recovery and back up and provide technical support.
   - The DBA has a DBA account in the DBMS which called a system or superuser account.
   - DBA repairs damage caused due to hardware and/or software failures.

2. **Naive / Parametric End Users :**
   Parametric End Users are the unsophisticated who don't have any DBMS

knowledge but they frequently use the data base 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.

3. **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.

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

5. **Data Base Designers :**
   Data Base Designers are the users who design the structure of data base which includes tables, indexes, views, constraints, triggers, stored procedures. He/she controls what data must be stored and how the data items to be related.

6. **Application Program :**
   Application Program 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.

Database users are categorized based up on their interaction with the data base.

These are seven types of data base users in DBMS.

Advantages of DBMS

- **Controls database redundancy**: All the data is stored in one place, and that recorded in the database and hence controls the redundancy in the database.
- **Data sharing:** DBMS allows users with authority to share the data in the database with multiple users.
- **Easy Maintenance:** The centralized nature of the database helps in the easy maintenance of the data.
- **Reduce time:** It reduces the maintenance need and development time.
- **Backup:** It automatically backs up data to maintain its integrity in case of failure.
- **Multiple user interfaces**: It offers a number of user interface to multiple users.

Disadvantages of DBMS

- **Cost of software and hardware**: It requires a number of high powered processors and large size memory to run DBMS.
- **Size:** a Large amount of storage size is required to run DBMS efficiently.
- **Complexity:** DBMS adds an additional layer of complexity to the data.
- **Higher impact of failure**: DBMS faces a higher risk of losing the data since all the data is stored at a single location and a catastrophic failure can wipe it all.

DBMS Facilities:

It provides facilities for defining the structure in which data is to be store on physical devices.

Which consist on those command that create the objects like table , index ,view etc. in the data base Create, Drop and Alter are few commands of Data Definition Language.

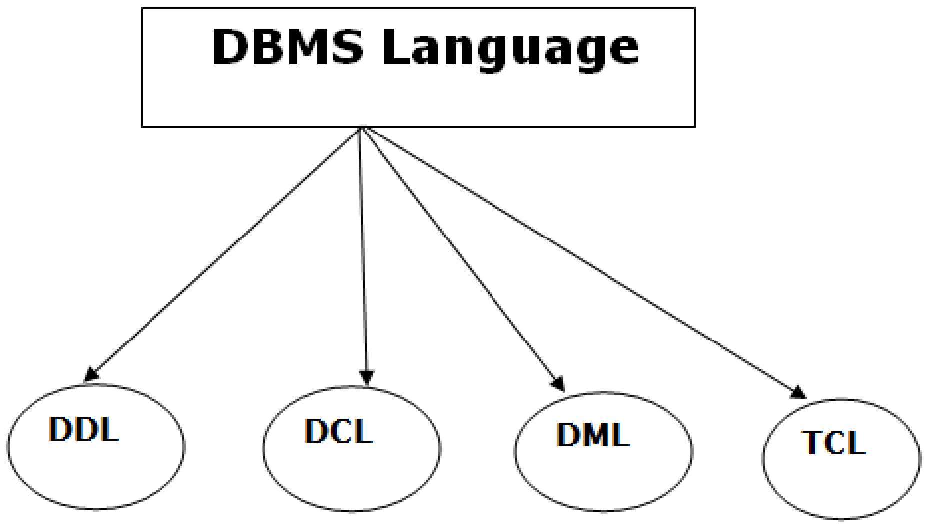
It provides facilities for entering , changing and extracting data from the data base. Data Manipulation Operation is also called Query and DML is also called a Query Language. Select, Delete , Insert ,Update are few commands of Data Manipulation Language.

DCL is another facilities of DBMS that provides security mechanism or scheme for protecting data from an authorized access. Grant , Revoke are few commands of Data Control Language.

Database Language

- 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 Language



1. Data Definition Language

- **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.

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

2. Data Manipulation Language

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.

3. Data Control Language

- **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 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.