What is Data?

- 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 a systematic collection of data.
- They support electronic storage and manipulation of data.
- Databases make data management easy.

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.

What is a Database Management System (DBMS)?

- **Database Management System (DBMS)** is a collection of programs that enable its users to access databases, manipulate data, report, and represent data.
- It also helps to control access to the database.
- Database Management Systems are not a new concept and, as such, had been first implemented in the 1960s.
- Charles Bachman's Integrated Data Store (IDS) is said to be the first DBMS in history.

Database characteristics:

- A modern DBMS has the following characteristics −
- **Real-world entity :**

A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

- **Relation-based tables :**

  DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.

- **Isolation of data and application :**

  A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.

- **Less redundancy :**

  In the database approach, ideally, each data item is stored in only one place in the database. In some cases, data redundancy still exists to improve system performance, but such redundancy is controlled by application programming and kept to minimum by introducing as little redundancy as possible when designing the database.

- **Query Language :**

  DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.

- **ACID Properties :**

  DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID).

  Atomicity

  either the entire transaction takes place at once or doesn't happen at all.

Consistency

This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database.

Isolation

This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of database state. Transactions occur independently without interference.

Durability

This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs.

These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.

- **Multiuser and Concurrent Access :**

  DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.

- **Multiple views :**

  DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

- **Data sharing:**

  The integration of all the data, for an organization, within a database system has many advantages. First, it allows for data sharing among employees and others who have access to the system. Second, it gives users the ability to generate more information from a given amount of data than would be possible without the integration.

- **Security :**

  Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user.

Advantages of DBMS

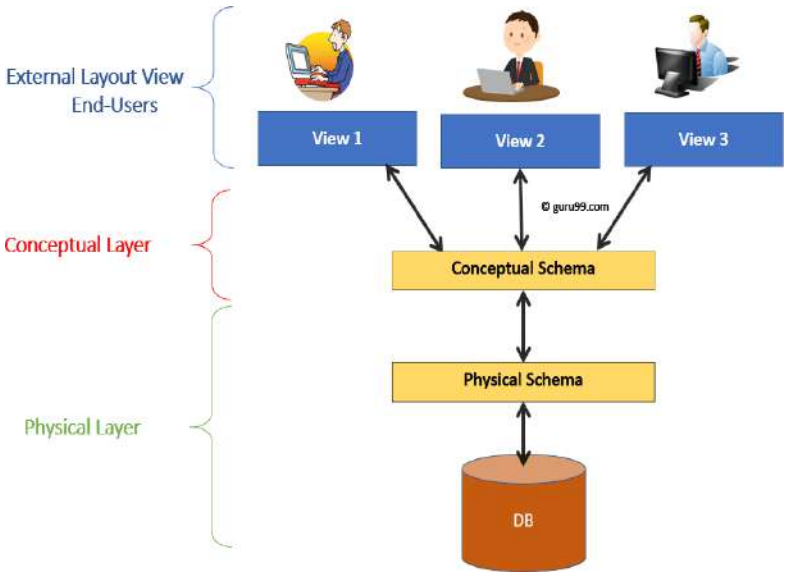
- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.
- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

Disadvantages of DBMS

- **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.
- **Size:** It occupies a large space of disks and large memory to run them efficiently.
- **Complexity:** Database system creates additional complexity and requirements.
- **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

Data Independence

- Data Independence is defined as a property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level. Data independence helps you to keep data separated from all programs that make use of it.



Levels of Database

Before we learn Data Independence, a refresher on Database Levels is important. The database has 3 levels as shown in the diagram below

1. Physical/Internal

2. Conceptual/Logical

3. External/View

1. Physical/Internal level:

- It describes how a record (e.g., customer) is stored.

Features:

- Lowest level of abstraction.
- It describes how data are actually stored.
- It describes low-level complex data structures in detail.
- At this level, efficient algorithms to access data are defined.

2.Conceptual/Logical level:

It describes what data stored in database, and the relationshipsamong the data.

Features*:*

- It is next-higher level of abstraction. Here whole Database is divided into small simple structures.
- Users at this level need not be aware of the physical-level complexity used to implement the simple structures.
- Generally, database administrators (DBAs) work at logical level of abstraction.

3.External/View level:

Application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.

Features:

- It is the highest level of abstraction.
- It describes only a part of the whole Database for particular group of users.
- This view hides all complexity.
- It exists only to simplify user interaction with system.
- The system may provide many views for the whole system.

Types of Data Independence

In DBMS there are two types of data independence

1. Physical data independence
2. Logical data independence.

1. Physical Data Independence

- Physical data independence helps you to separate conceptual levels from the internal/physical levels. It allows you to provide a logical description of the database without the need to specify physical structures. Compared to Logical Independence, it is easy to achieve physical data independence.
- With Physical independence, you can easily change the physical storage structures or devices with an effect on the conceptual schema. Any change done would be absorbed by the mapping between the conceptual and internal levels. Physical data independence is achieved by the presence of the internal level of the database and then the transformation from the conceptual level of the database to the internal level.

Examples of changes under Physical Data Independence

Due to Physical independence, any of the below change will not affect the conceptual layer.

- Using a new storage device like Hard Drive or Magnetic Tapes
- Modifying the file organization technique in the Database
- Switching to different data structures.
- Changing the access method.

- Modifying indexes.
- Changes to compression techniques or hashing algorithms.
- Change of Location of Database from say C drive to D Drive

2. Logical Data Independence

- Logical Data Independence is the ability to change the conceptual scheme without changing

  External views

  External API or programs

- Any change made will be absorbed by the mapping between external and conceptual levels.
- When compared to Physical Data independence, it is challenging to achieve logical data independence.

Examples of changes under Logical Data Independence

Due to Logical independence, any of the below change will not affect the external layer.

- Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs
- Merging two records into one
- Breaking an existing record into two or more records

Difference between Physical and Logical Data Independence

Logical Data Independence	Physical Data Independence
Logical Data Independence is mainly concerned with the structure or changing the data definition.	Mainly concerned with the storage of the data.
It is difficult as the retrieving of data is mainly dependent on the logical structure of data.	It is easy to retrieve.

Compared to Logic Physical independence it is difficult to achieve logical data independence.	Compared to Logical Independence it is easy to achieve physical data independence.
You need to make changes in the Application program if new fields are added or deleted from the database.	A change in the physical level usually does not need change at the Application program level.
Modification at the logical levels is significant whenever the logical structures of the database are changed.	Modifications made at the internal levels may or may not be needed to improve the performance of the structure.
Concerned with conceptual schema	Concerned with internal schema
Example: Add/Modify/Delete a new attribute	Example: change in compression techniques, hashing algorithms, storage devices, etc

Importance of Data Independence

- Helps you to improve the quality of the data
- Database system maintenance becomes affordable
- Enforcement of standards and improvement in database security
- You don't need to alter data structure in application programs
- Permit developers to focus on the general structure of the Database rather than worrying about the internal implementation
- It allows you to improve state which is undamaged or undivided
- Database incongruity is vastly reduced.
- Easily make modifications in the physical level is needed to improve the performance of the system.

Components of Database

There are the following components of database

1.User

2. Data or Database

3. DBMS

4. Database Application

1. User

- User are the one who really uses the database.User can be administrators,developers,or end-users.
- There are three types of user who play different roles in DBMS.

    1. **Application Programmers**

       The user who write the application programs in programming languages(such as Java,C++,or Visual Basic) to interact with database called Application Programmer.

    2. **Database Administrators**

       A person who manage the overall DBMS is called database administrator or simply DBA.

    3. **End-user**

       The end-user are those who interact with the database management system to perform different operations by using the different database commands such as insert,update,retrieve,and delete on the data,etc.

2. Data or Database

- A very huge amount of data will be stored in the database and it forms the main source for all other components to interact with each other.

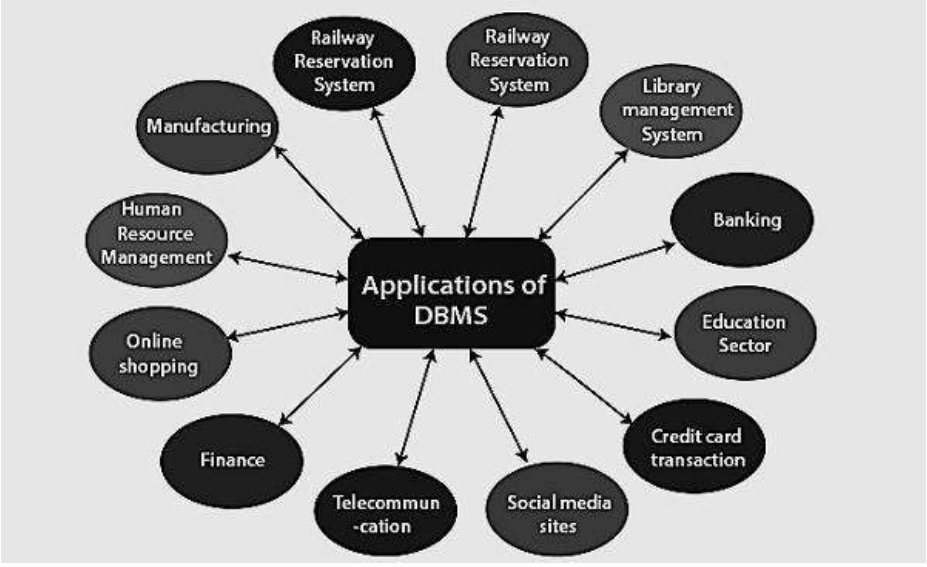
- There are two types of data
  - **One is user Data:**It contain data which is responsible for the database,i.e.;based on the requirement,the data will be stored in the various table of the database in the form of rows and column.
  - **Another data is Metadata**: It is know as 'data about data'i.e,;It store the information like how many tables,their names,how many columns and their names,primary Keys,foreign Keys,etc.Basically these metadata will have information about each table and their constraints in the database.

3. DBMS

- This is the software that helps the user to interact with the database.
- It allows the users to insert,delete,update,or retrieve the data.
- All these operations are handled by query languages like SQL,MYSQL,Oracle,etc.

4. Database Application

- It is the application program which help the users to interact with the database by means of query language.
- The database application will not have any idea about the underlying DBMS.



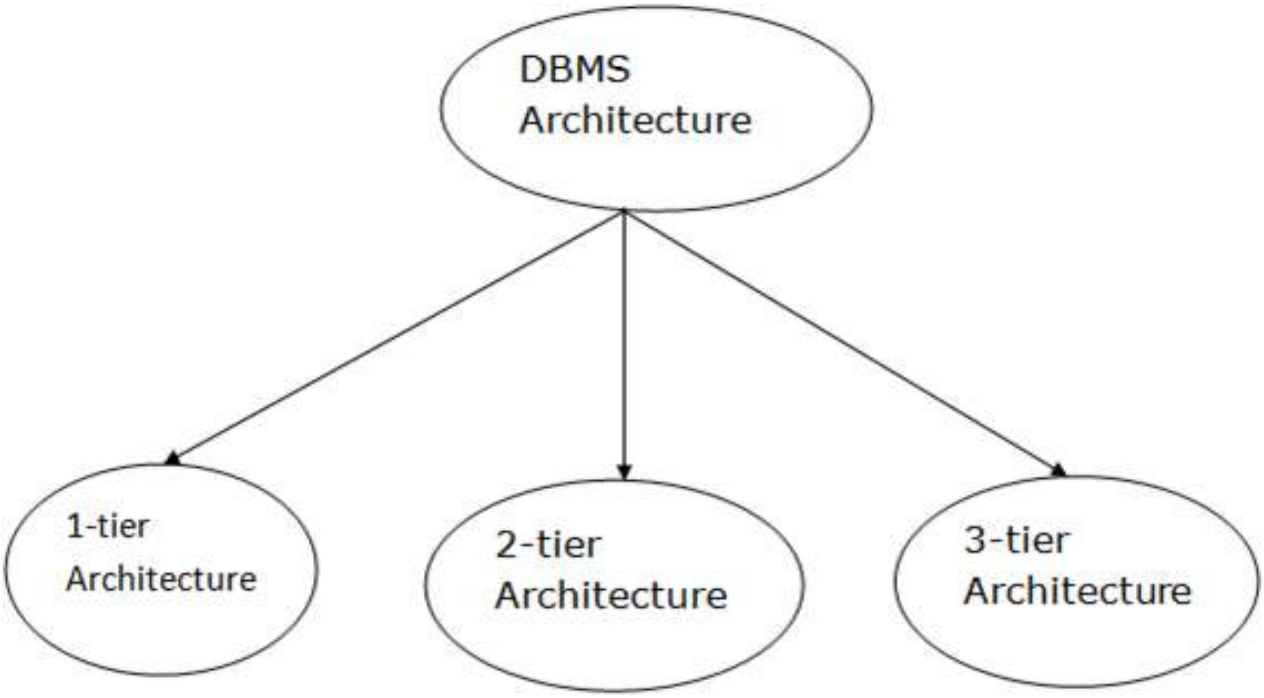
Let's see some of the applications where database management system uses −

- **Railway Reservation System** − The railway reservation system database plays a very important role by keeping record of ticket booking, train's departure time and arrival status and also gives information regarding train late to people through the database.
- **Library Management System** − Now-a-days it's become easy in the Library to track each book and maintain it because of the database. This happens because there are thousands of books in the library. It is very difficult to keep a record of all books in a copy or register. Now DBMS used to maintain all the information related to book issue dates, name of the book, author and availability of the book.
- **Banking** − Banking is one of the main applications of databases. We all know there will be a thousand transactions through banks daily and we are doing this without going to the bank. This is all possible just because of DBMS that manages all the bank transactions.
- **Universities and colleges** − Now-a-days examinations are done online. So, the universities and colleges are maintaining DBMS to store Student's registrations details, results, courses and grade all the information in the database. For example, telecommunications. Without DBMS there is no telecommunication company. DBMS is most useful to these companies to store the call details and monthly postpaid bills.
- **Credit card transactions** − The purchase of items and transactions of credit cards are made possible only by DBMS. A credit card holder has to know the importance of their information that all are secured through DBMS.
- **Social Media Sites** − By filling the required details we are able to access social media platforms. Many users sign up daily on social websites such as Facebook, Pinterest and Instagram. All the information related to the users are stored and maintained with the help of DBMS.
- **Finance** − Now-a-days there are lots of things to do with finance like storing sales, holding information and finance statement management etc. these all can be done with database systems.
- **Telecommunication** –Without DBMS any telecommunication company can't think.The database management system is necessary for these companies to store the details and monthly postpaid bill in the database.
- **Online Shopping** − Now-a-days we all do Online shopping without wasting the time by going shopping with the help of DBMS. The products are added and sold only with the help of DBMS like Purchase information, invoice bills and payment.
- **Human Resource Management** − The management keeps records of each employee's salary, tax and work through DBMS.
- **Manufacturing** − Manufacturing companies make products and sell them on a daily basis. To keep records of all those details DBMS is used.
- **Airline Reservation system** − Just like the railway reservation system, airlines also need DBMS to keep records of flights arrival, departure and delay status.

DBMS Architecture

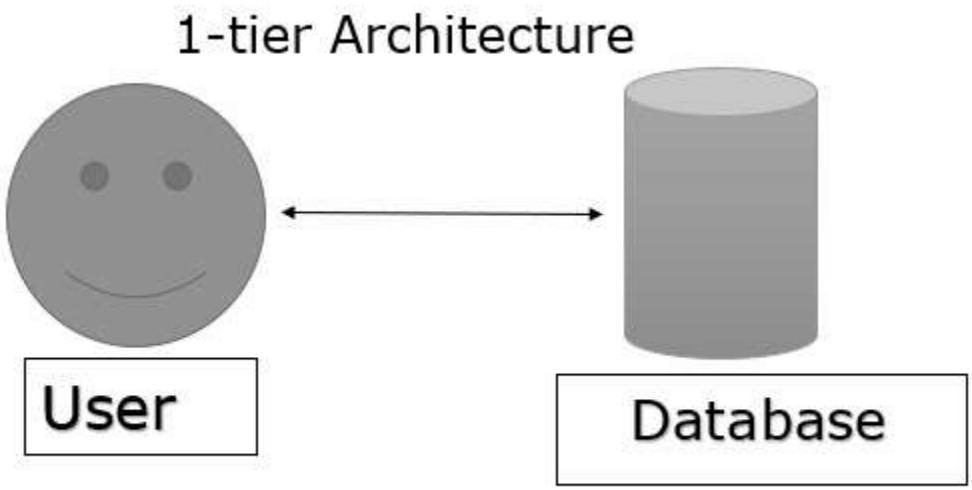
- The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected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

Types of DBMS Architecture



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.



Advantages of 1-tier Architecture

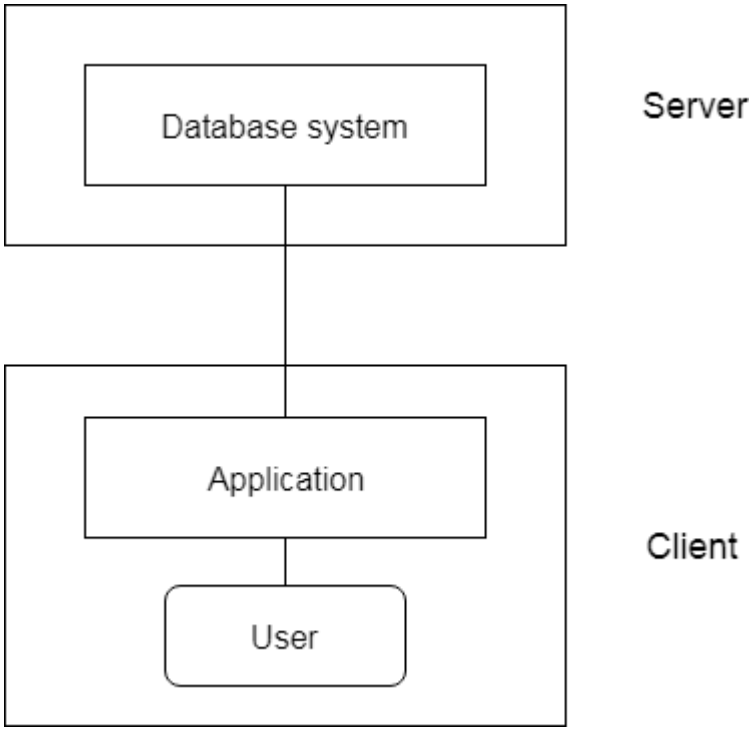
- Easy to implement and optimize performance,
- The cost of development is less.
- It is fast for communication.

Disadvantages of 1-tier Architecture

- Do not support remote/distributed access for data resources.
- The cost of central mainframe is high.
- Completely unscalable[only one user can access the system at given time via the local client].

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.



- Let us consider another example of the two-tier architecture of database. Consider a railway ticket reservation system. How does this work? Imagine a person is reserving the ticket from Delhi to Goa on a particular day. At the same time, another person in some other place of Delhi is also reserving the ticket to Goa on the same day for the same train. Now there is a requirement for two tickets, but for different persons. What will the reservation system do? It takes the request from both of them and queues the requests entered by each of them. Here

the request entered to the application layer and request is sent to the database layer. Once the request is processed in the database, the result is sent back to the application layer for the user.

Advantages of 2-tier Architecture

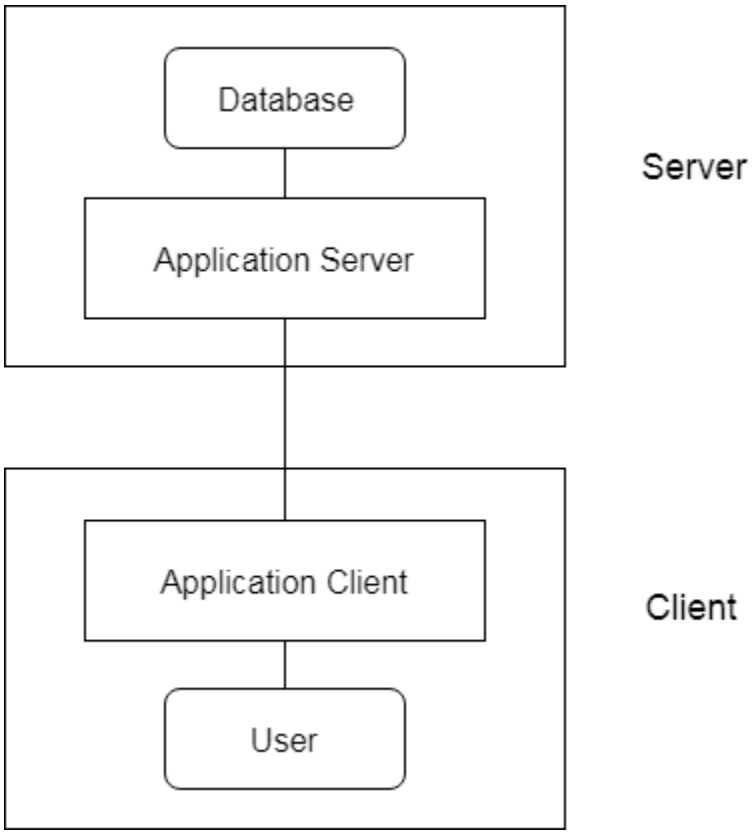
- Easy to understand as it directly communicates with the database.
- Requested data can be retrieved very quickly when there are fewer users.
- Easy to modify – any changes required, directly requests can be sent to the database
- Easy to maintain – When there are multiple requests, it will be handled in a queue and there will not be any chaos.

Disadvantages of 2-tier Architecture

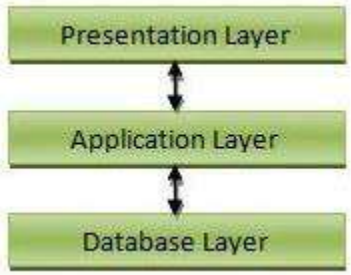
- It would be time-consuming when there is a huge number of users. All the requests will be queued and handed one after another. Hence it will not respond to multiple users at the same time.
- This architecture would little cost-effective.

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.



The 3-tier architecture is the most widely used database architecture. It can be viewed below.



- **The presentation layer / User layer** is the layer where the user uses the database. He does not have any knowledge about the underlying database. He simply interacts with the database as though he has all data in front of him. You can imagine this layer as a registration form where you will be inputting your details. Did you ever guess, after pressing the 'submit' button where the data go? No right? You just know that your details are saved. This is the presentation layer where all the details from the user are taken, sent to the next layer for processing.

- **The application layer** is the underlying program that is responsible for saving the details that you have entered and retrieving your details to show up on the page. This layer has all the business logic like validation, calculations, and manipulations of data, and then sends the requests to the database to get the actual data. If this layer sees that the request is invalid, it sends back the message to the presentation layer. It will not hit the database layer at all.
- **The data layer or Database layer** is the layer where the actual database resides. In this layer, all the tables, their mappings, and the actual data present. When you save you details from the front end, it will be inserted into the respective tables in the database layer, by using the programs in the application layer. When you want to view your details in the web browser, a request is sent to the database layer by the application layer. The database layer fires queries and gets the data. These data are then transferred to the browser (presentation layer) by the programs in the application layer.

Advantages of 3-tier architecture:

- Easy to maintain and modify. Any changes requested will not affect any other data in the database. The application layer will do all the validations.
- Improved security. Since there is no direct access to the database, data security is increased. There is no fear of mishandling the data. The application layer filters out all the malicious actions.
- Good performance. Since this architecture cache, the data once retrieved, there is no need to hit the database for each request. This reduces the time consumed for multiple requests and hence enables the system to respond at the same time.

Disadvantages of 3-tier Architecture

- Disadvantages of 3-tier architecture are that it is a little more complex and little more effort is required in terms of hitting the database.

Difference Between Two-Tier (2-tier) And Three-Tier(3-tier) Database Architecture

S.NO	Two-Tier Database Architecture (2-tier)	Three-Tier Database Architecture (3-tier)
1	It is a Client-Server Architecture.	It is a Web-based application.
2	In two-tier, the application logic is either buried inside the user interface on the client or within the database on the server (or both).	In three-tier, the application logic or process resides in the middle-tier, it is separated from the data and the user interface.
3	Two-tier architecture consists of two layers : Client Tier and Database (Data Tier).	Three-tier architecture consists of three layers : Client Layer, Business Layer and Data Layer.
4	It is easy to build and maintain.	It is complex to build and maintain.
5	Two-tier architecture runs slower.	Three-tier architecture runs faster.
6	It is less secured as client can communicate with database directly.	It is secured as client is not allowed to communicate with database directly.

7	It results in performance loss whenever the users increase rapidly.	It results in performance loss whenever the system is run on Internet but gives more performance than two-tier architecture.

8	Example – Contact Management System created using MS-Access or Railway Reservation System, etc.	Example – Designing registration form which contains text box, label, button or a large website on the Internet, etc.

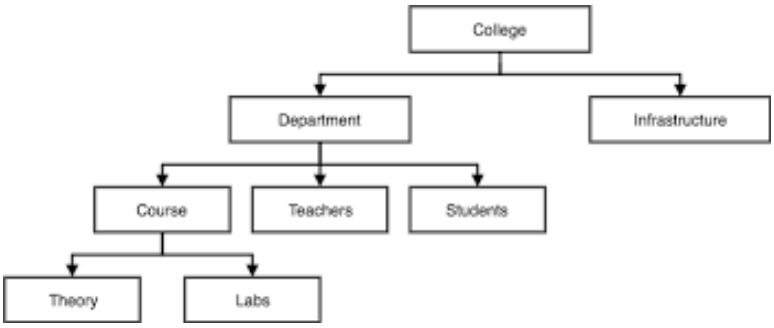
DBMS Database Model

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system.
While the Relational Model is the most widely used database model, there are other models too:

- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

Hierarchical Model

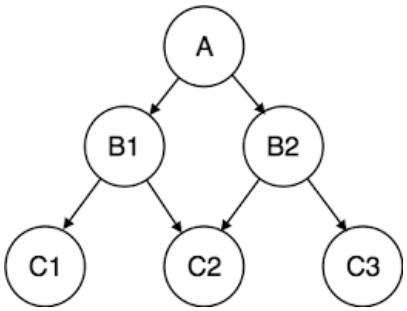
- This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked.
- The hierarchy starts from the Root data, and expands like a tree, adding child nodes to the parent nodes.
- In this model, a child node will only have a single parent node.
- This model efficiently describes many real-world relationships like index of a book, recipes etc.
- In hierarchical model, data is organized 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

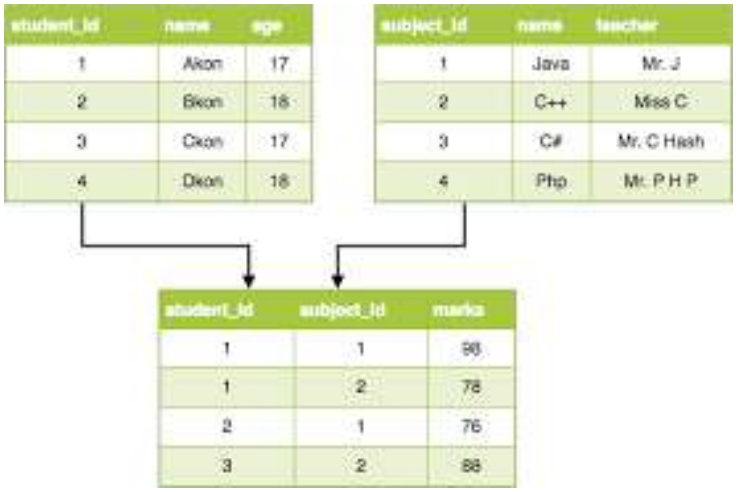
- This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.

- In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast.
- This database model was used to map many-to-many data relationships.
- This was the most widely used database model, before Relational Model was introduced.



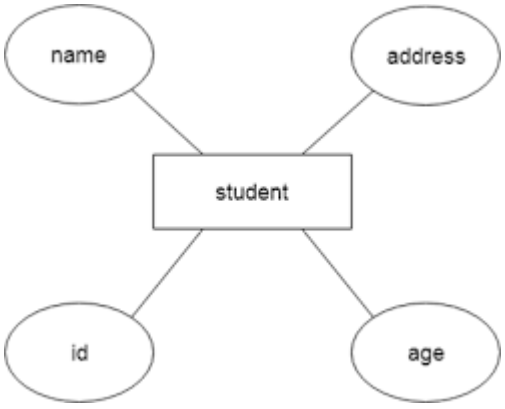
Relational Model

- In this model, data is organized in two-dimensional tables and the relationship is maintained by storing a common field.
- This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.
- The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.
- Hence, tables are also known as relations in relational model.

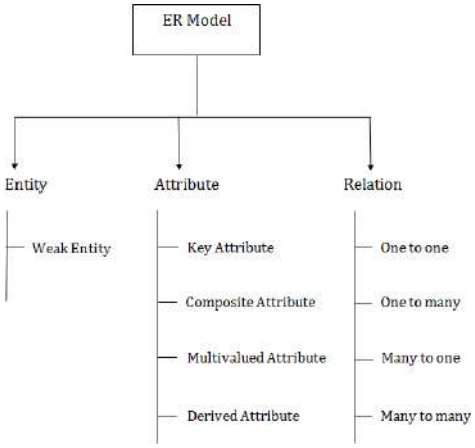


Entity-relationship Model

- In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.
- Different entities are related using relationships.
- E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.
- This model is good to design a database, which can then be turned into tables in relational model(explained below).
- Let's take an example, If we have to design a School Database, then Student will be an entity with attributes name, age, address etc.
- As Address is generally complex, it can be another entity with attributes street name, pincode, city etc, and there will be a relationship between them.
- Relationships can also be of different types.

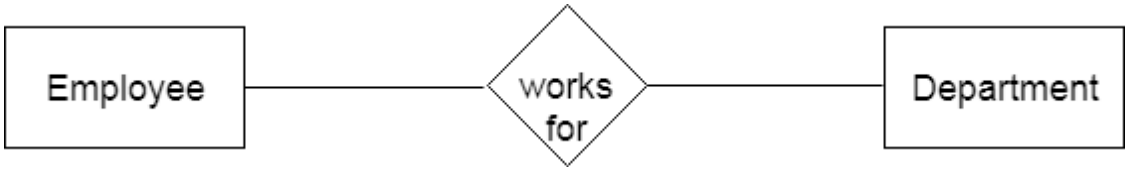


Component of ER Diagram



1. Entity:

- An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
- Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



Strong Entity:

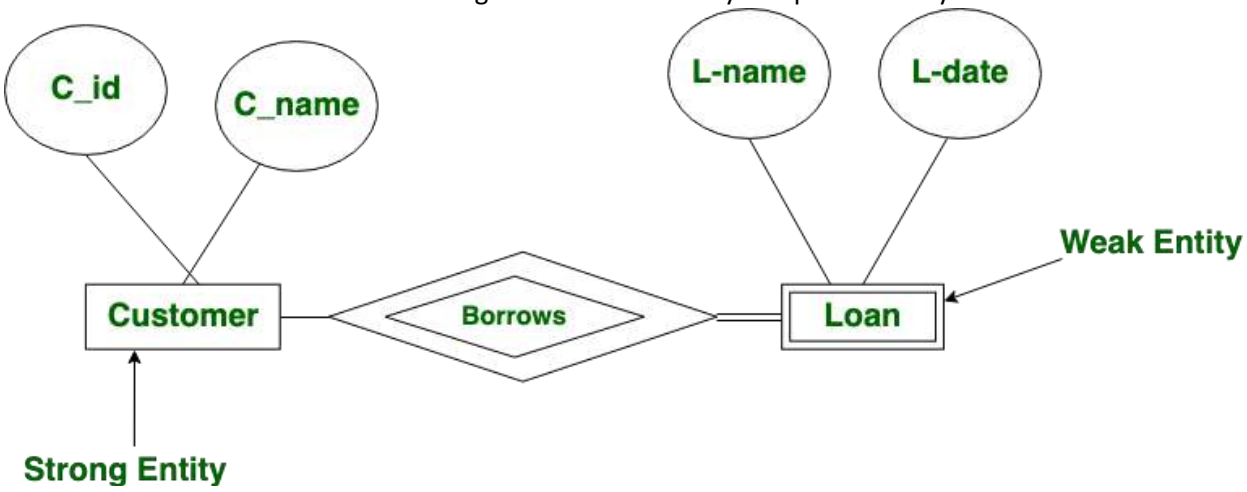
A strong entity is not dependent on any other entity in the schema.

- A strong entity will always have a primary key.
- Strong entities are represented by a single rectangle.
- The relationship of two strong entities is represented by a single diamond.
  Various strong entities, when combined together, create a strong entity set.

Weak Entity:

A weak entity is dependent on a strong entity to ensure its existence.

- Unlike a strong entity, a weak entity does not have any primary key.
- It instead has a partial discriminator key.
- A weak entity is represented by a double rectangle.
  The relation between one strong and one weak entity is represented by a double diamond.



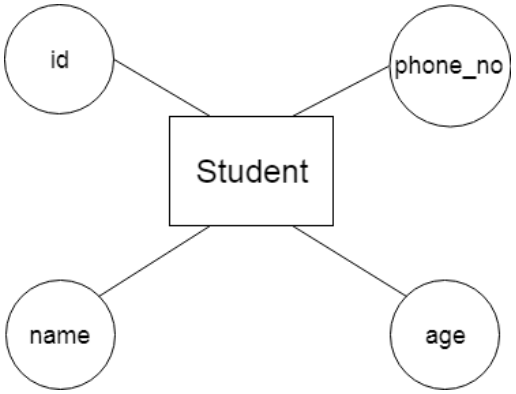
Difference between Strong and Weak Entity:

S.NO	Strong Entity	Weak Entity
1.	Strong entity always has a primary key.	While a weak entity has a partial discriminator key.

2.	Strong entity is not dependent on any other entity.	Weak entity depends on strong entity.
3.	Strong entity is represented by a single rectangle.	Weak entity is represented by a double rectangle.
4.	Two strong entity's relationship is represented by a single diamond.	While the relation between one strong and one weak entity is represented by a double diamond.
5.	Strong entities have either total participation or not.	While weak entity always has total participation.

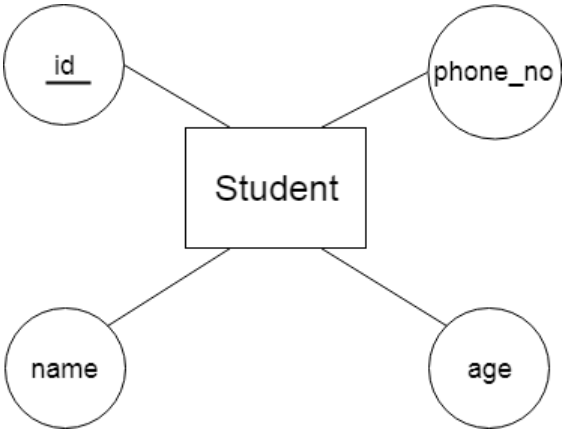
2. Attribute

- The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.
- **For example,** id, age, contact number, name, etc. can be attributes of a student.



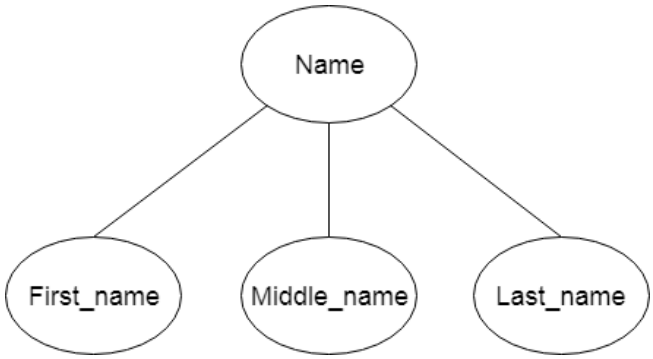
a. Key Attribute

- The key attribute is used to represent the main characteristics of an entity. It represents a primary key.
- The key attribute is represented by an ellipse with the text underlined.



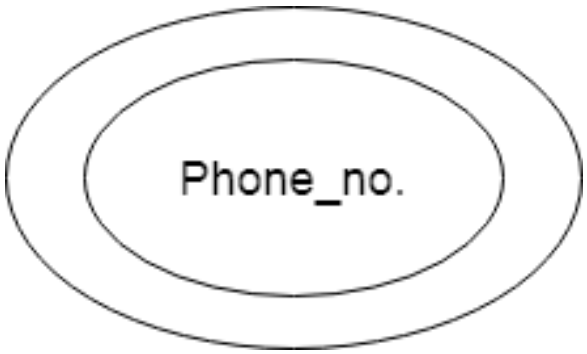
b. Composite Attribute

- An attribute that composed of many other attributes is known as a composite attribute.
- The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



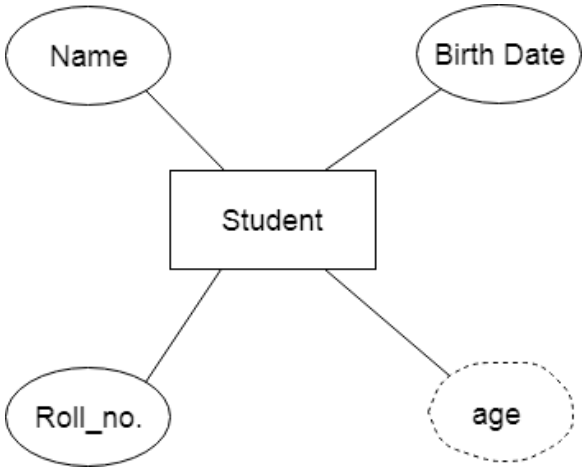
c. Multivalued Attribute

- An attribute can have more than one value.
- These attributes are known as a multivalued attribute.
- The double oval is used to represent multivalued attribute.
- **For example,** a student can have more than one phone number.



d. Derived Attribute

- An attribute that can be derived from other attribute is known as a derived attribute.
- It can be represented by a dashed ellipse.
- **For example,** A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

- A relationship is used to describe the relation between entities.
- Diamond or rhombus is used to represent the relationship.

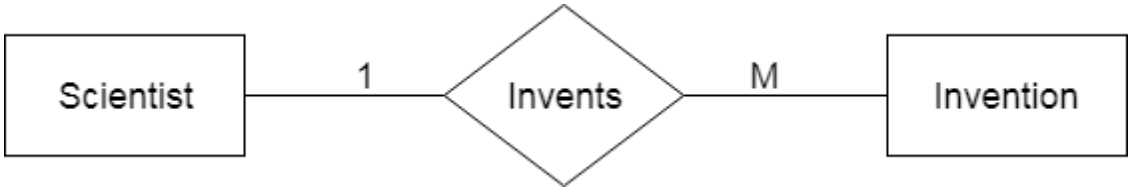
Types of relationship are as follows:

a. One-to-One Relationship

- When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.
- **For example,** A female can marry to one male, and a male can marry to one female.

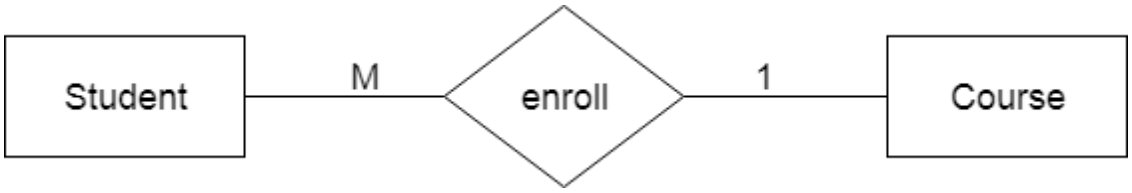
b. One-to-many relationship

- When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.
- **For example,** Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. Many-to-one relationship

- When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.
- **For example,** Student enrolls for only one course, but a course can have many students.

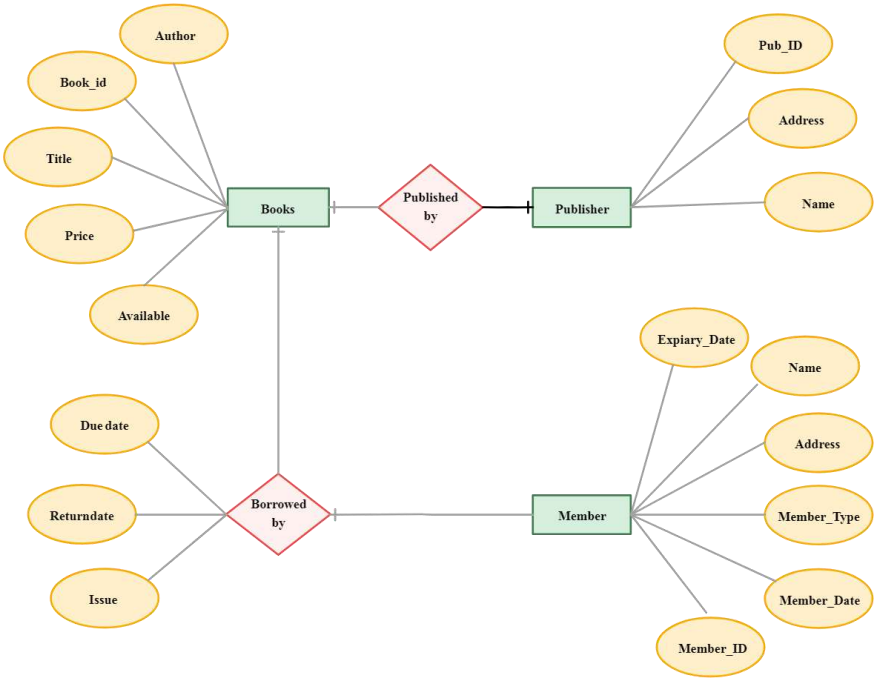


d. Many-to-many relationship

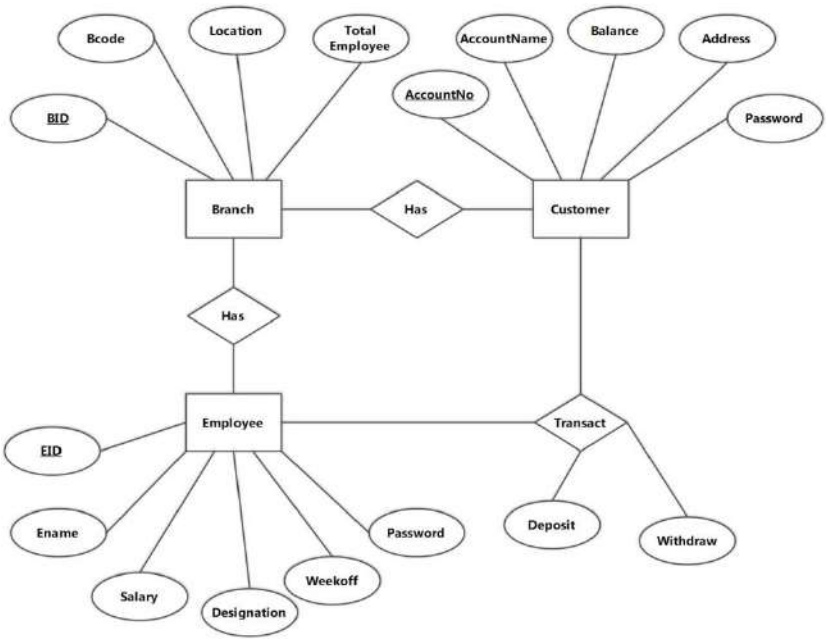
- When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.
- **For example,** Employee can assign by many projects and project can have many employees.

Example of ER Diagram

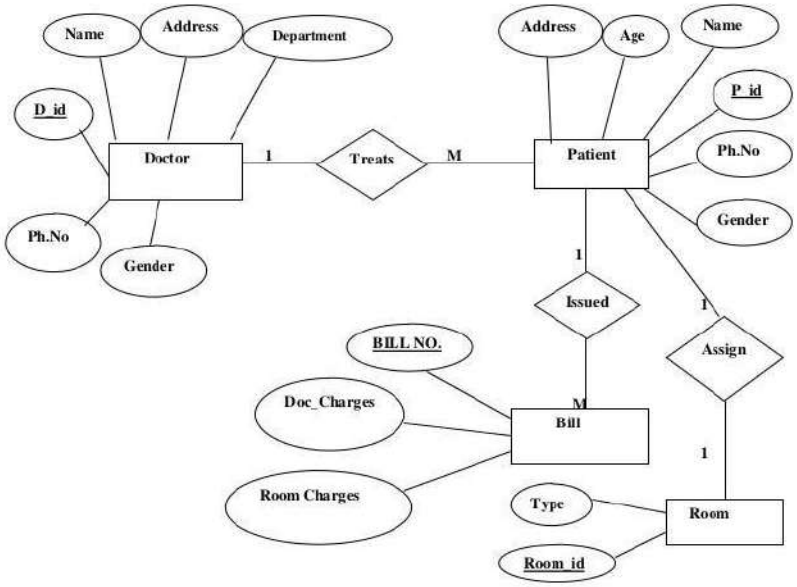
1. Library Management System



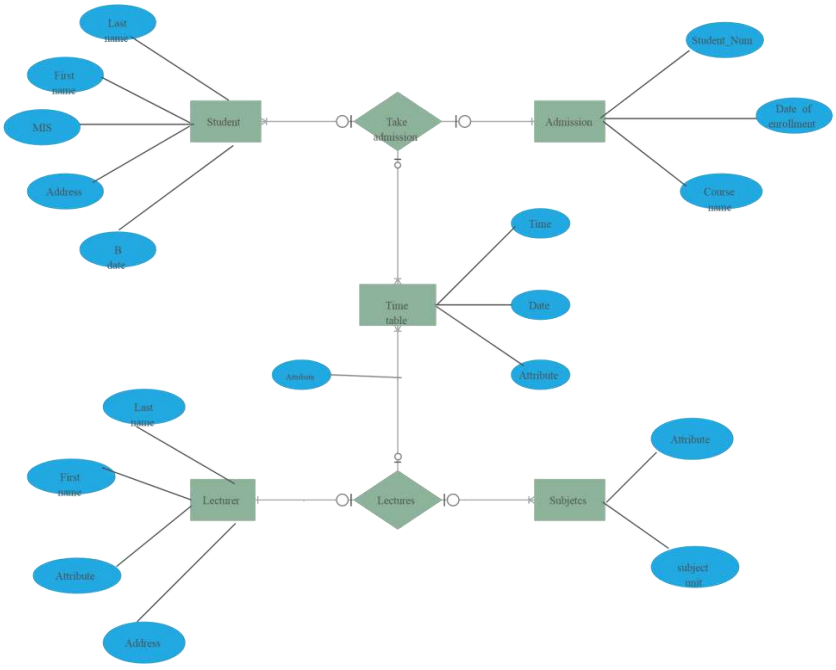
2. Banking Management System



3. Hospital Management System



4. School/Collage Management System



Keys

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.
- **For example,** ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

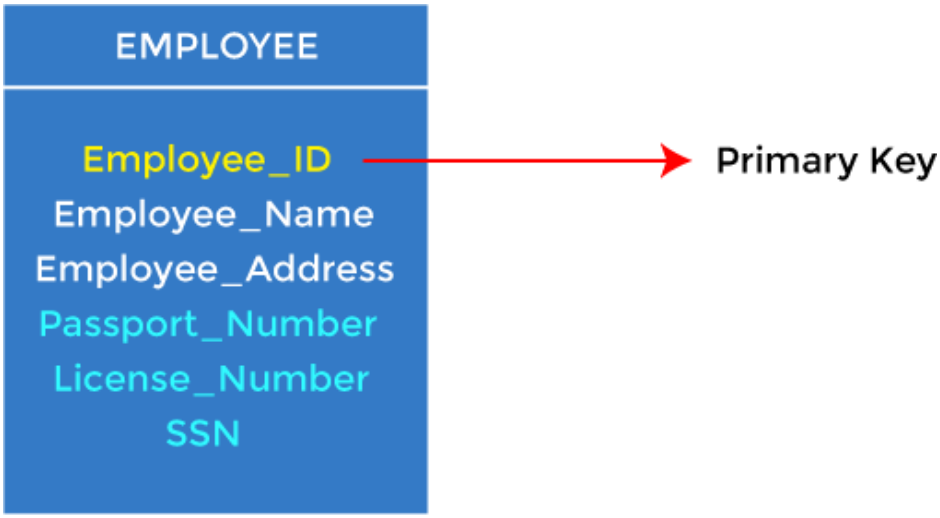
Types of keys:

1. Primary Key
2. Candidate Key
3. Super Key
4. Foreign Key
5. Alternate Key
6. Composite Key
7. Unique Key

1. Primary key

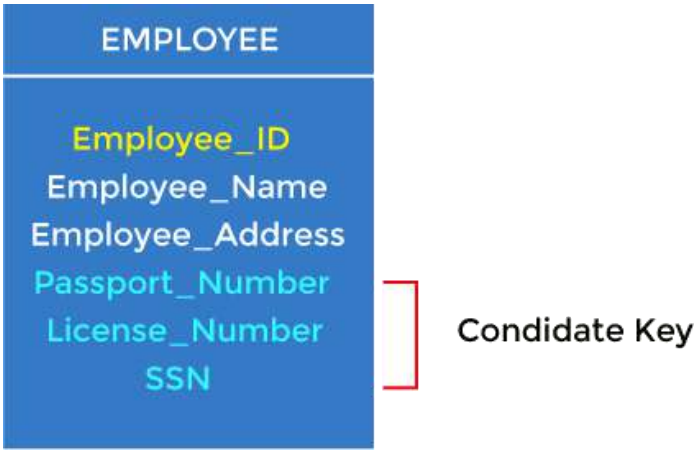
- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.

- In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.



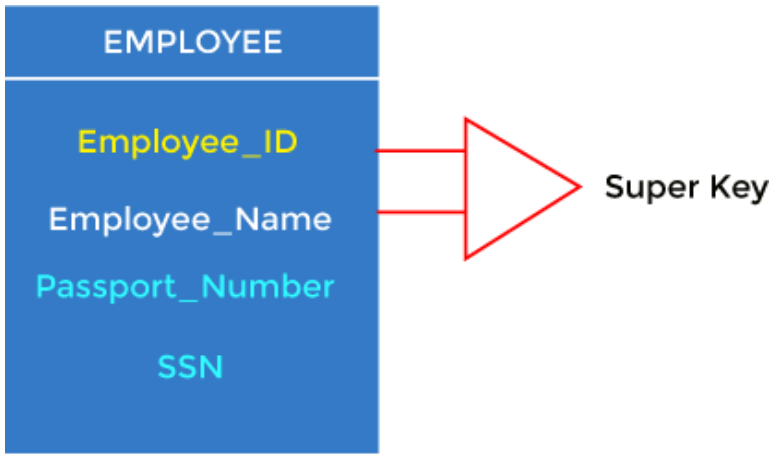
2. Candidate key

- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.
- **For example:** In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



3. Super Key

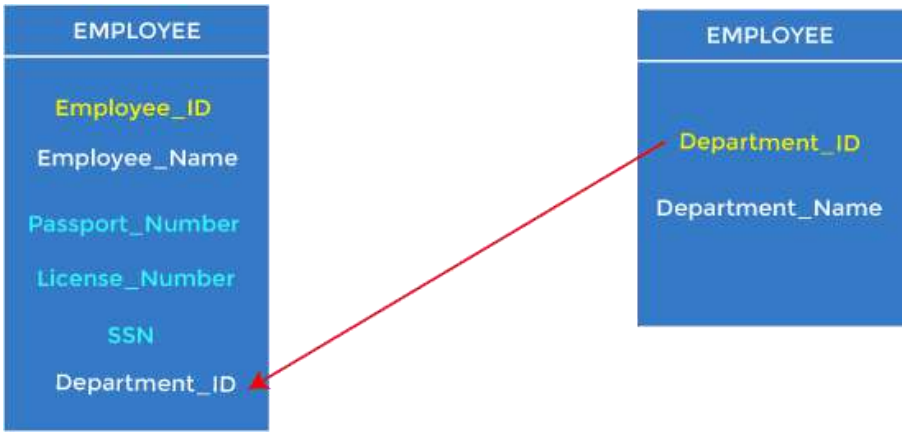
- Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.



- **For example:** In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.
- The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key

- Foreign keys are the column of the table used to point to the primary key of another table.
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.

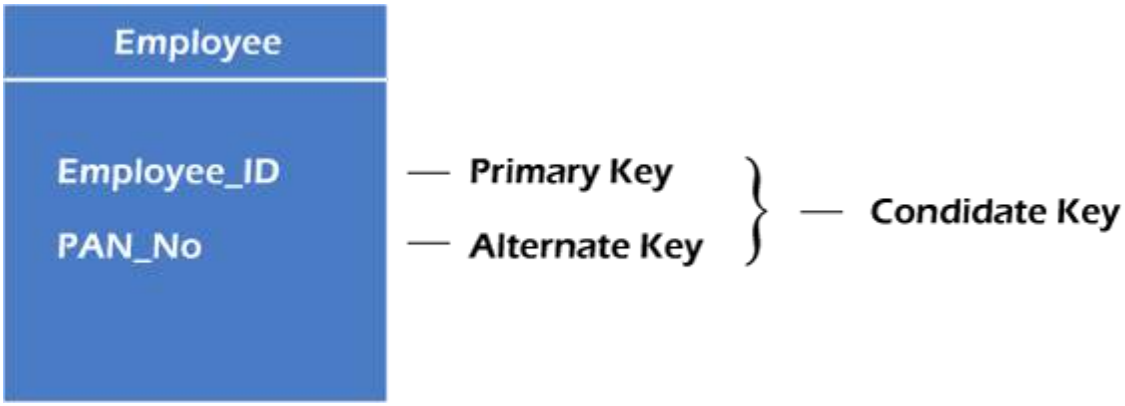


5. Alternate key

- There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining

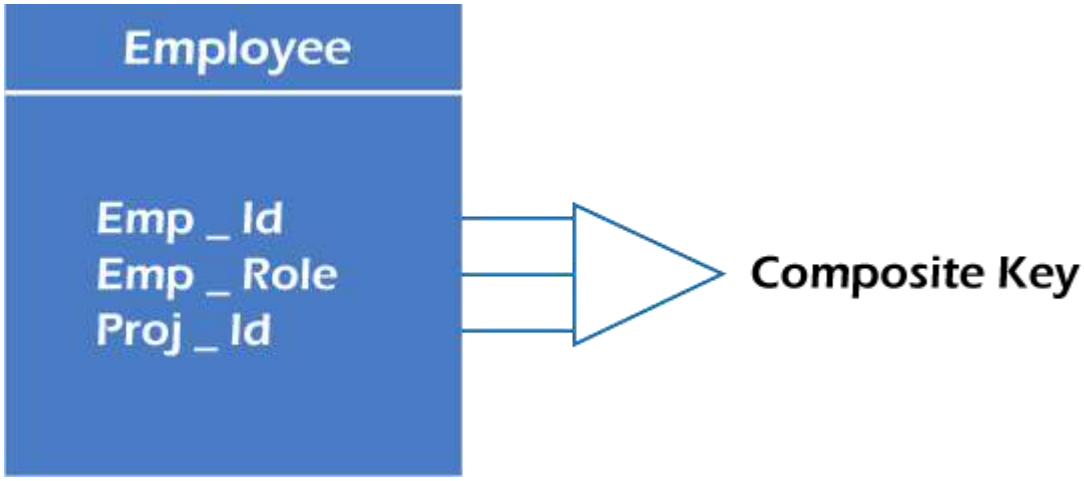
candidate key, if it exists, is termed the alternate key. **In other words,** the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

- **For example,** employee relation has two attributes, Employee_Id and PAN_No, that act as candidate keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.



6. Composite key

- Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.



- **For example,** in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.

7. Unique key

- A unique key is a set of one or more than one fields/columns of a table that uniquely identify a record in a database table.

- You can say that it is little like primary key but it can accept only one null value and it cannot have duplicate values.
- The unique key and primary key both provide a guarantee for uniqueness for a column or a set of columns.
- There is an automatically defined unique key constraint within a primary key constraint.
- There may be many unique key constraints for one table, but only one PRIMARY KEY constraint for one table.