

SYLLABUS

UNIT I	INTRODUCTION
	Database System, Its Applications, Purpose of Database Systems, View of Data – Data Abstraction – Instances and Schemas – data Models – the ER Model – Relational Model – Other Models – Database Languages – DDL – DML – database Access for applications Programs – database Users and Administrator – Transaction Management – database Architecture – Storage Manager – the Query Processor Database design and ER diagrams – ER Model - Entities, Attributes and Entity sets – Relationships and Relationship sets – ER Design Issues – Concept Design – Conceptual Design for University Enterprise. Introduction to the Relational Model – Structure – Database Schema, Keys – Schema Diagrams
UNIT II	RELATIONAL ALGEBRA AND CALCULUS
	Relational Query Languages, Relational Operations. Relational Algebra – Selection and projection set operations – renaming – Joins – Division – Examples of Algebra overviews – Relational calculus – Tuple relational Calculus – Domain relational calculus.
	Normalization – Introduction, non-loss decomposition and functional dependencies, First, Second, and Third normal forms – dependency preservation, Boyce/Codd normal form. Higher Normal Forms - Introduction, Multi-valued dependencies and Fourth normal form, Join dependencies and Fifth normal form
UNIT III	SQL QUERIES
	SQL: data definition, aggregate function, Null Values, nested sub queries, Joined relations.
	What are constraints, types of constraints, Integrity constraints
	Views: Introduction to views, data independence, security, updates on views, comparison between tables and views
UNIT IV	TRANSACTION AND CONCURRENCY CONTROL
	Transaction Concept- Transaction State- Implementation of Atomicity and Durability – Concurrency control – Executions – Serializability- Recoverability – Implementation of Isolation – Testing for serializability- Lock –Based Protocols – Timestamp Based Protocols- Validation- Based Protocols – Multiple Granularity. Recovery and Atomicity – Log – Based Recovery – Recovery with Concurrent Transactions – Buffer Management – Failure with loss of nonvolatile storage-Advance Recovery systems- Remote Backup systems.

UNIT V	PL/SQL
	Beginning with PL / SQL, Identifiers and Keywords, Operators, Expressions, Sequences, Control Structures, Cursors and Transaction, Collections and composite data types, Procedures and Functions, Exceptions Handling, Packages and Triggers

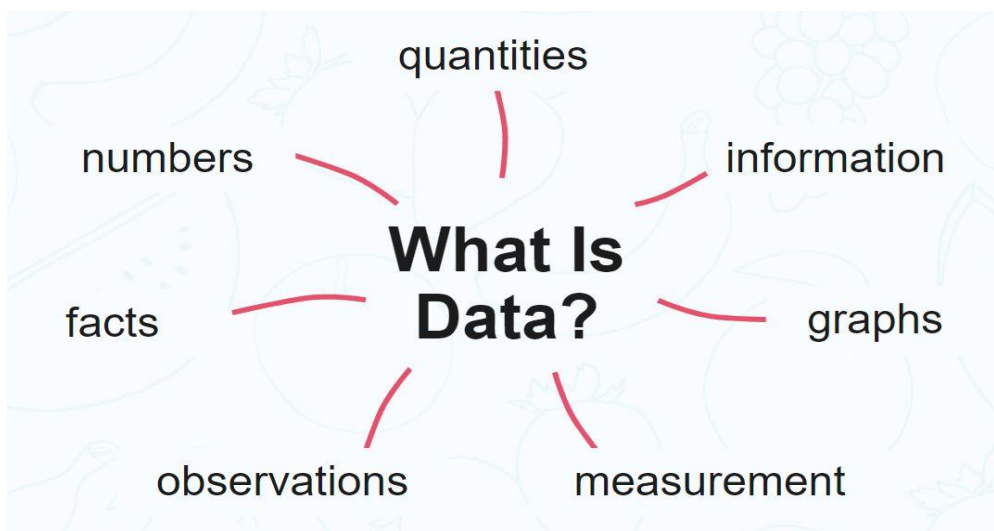
1.1 Introduction:

SQL is a database computer language designed for the retrieval and management of data in relational databases like MySQL, MS Access, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres etc. SQL stands for Structured Query Language. SQL was developed in the 1970s by IBM Computer Scientists.

SQL is not a database management system, but it is a query language which is used to store and retrieve the data from a database or in simple words SQL is a language that communicates with databases.

Data:

Data is a collection of discrete or continuous values that convey information, describing the quantity, quality, fact, statistics, other basic units of meaning, or simply sequences of symbols that may be further interpreted formally.



Database:

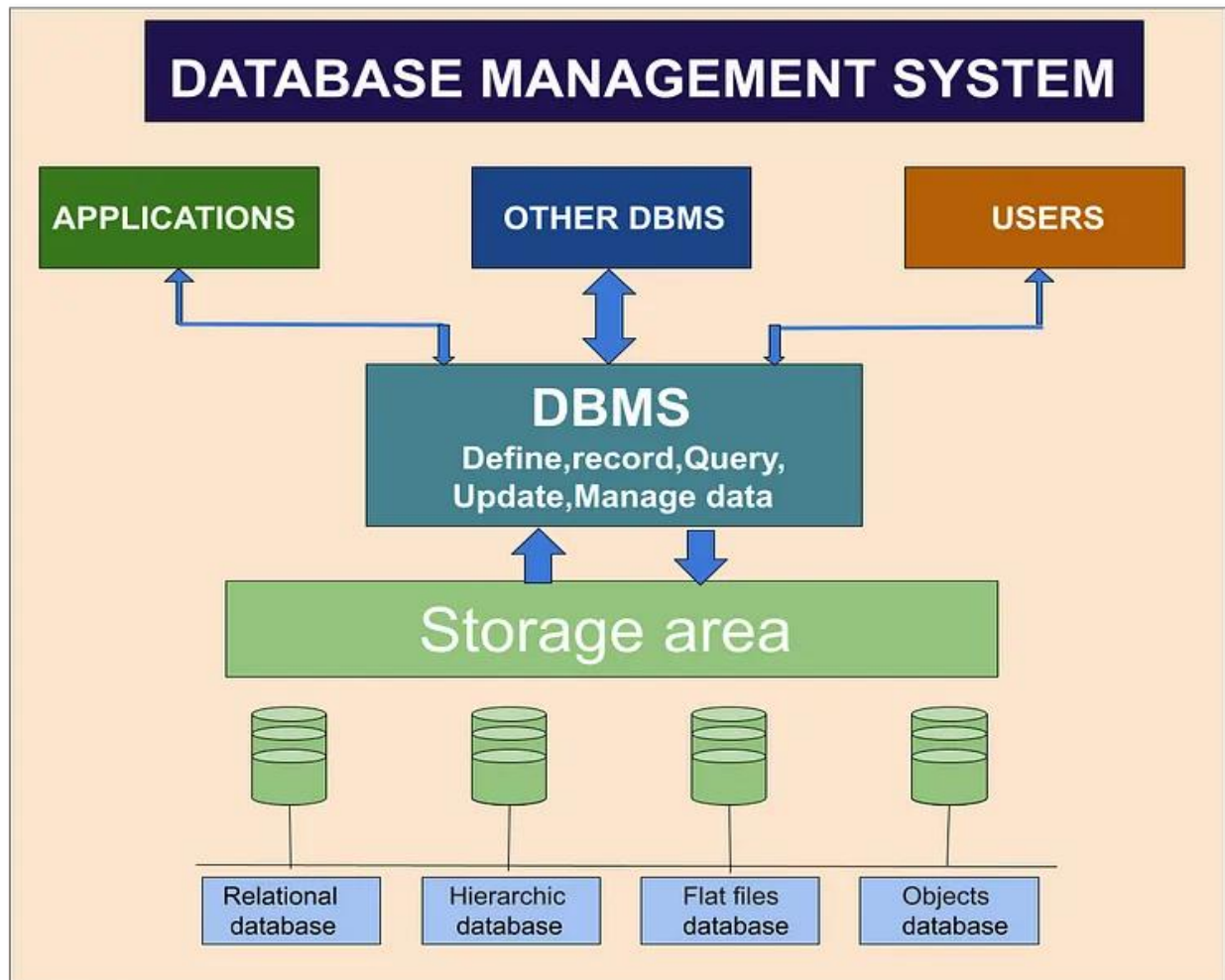
A database is an organized collection of structured information, or data, typically stored electronically in a computer system.



DBMS :

DBMS stands for **Database Management System**.

- We can break it like this DBMS = Database + Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can define DBMS like this: DBMS is a collection of interrelated data and a set of programs to store & access those data in an easy and effective manner.



- **Database Management System (DBMS)** is software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs that manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software store and retrieve data.
- DBMS allows users to create their own databases as per their requirements. The term "DBMS" includes the user of the database and other application programs. It provides an interface between the data and the software application.

To define DBMS:

- We need to specify the structure of the records of each file by defining the different types of data elements to be stored in each record.
- We can also use a coding scheme to represent the values of a data item.
- Basically, your Database will have 5 tables with a foreign key defined amongst the various tables.

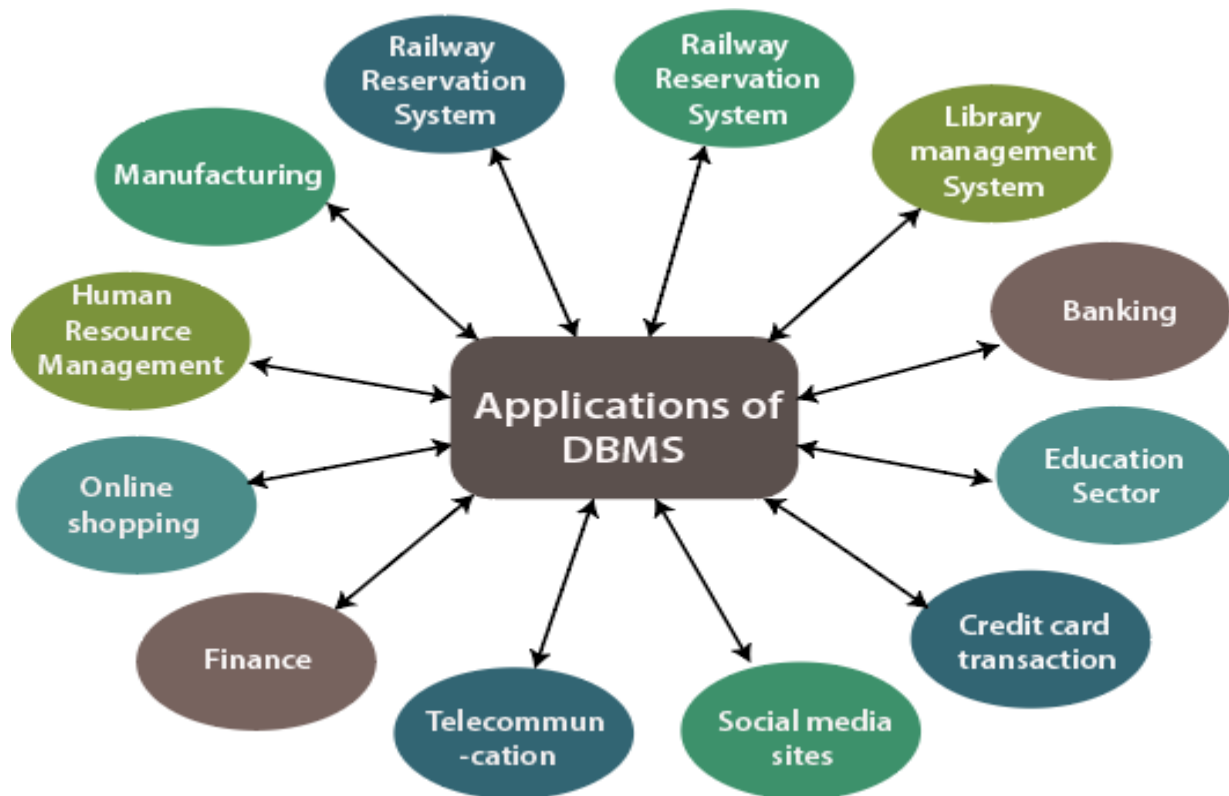
History:

Here, are the important landmarks from the history of DBMS:

- 1960 – Charles Bachman designed the first DBMS system
- 1970 – Codd introduced IBM'S Information Management System (IMS)
- 1976- Peter Chen coined and defined the Entity-relationship model, also known as the ER model
- 1980 – Relational Model becomes a widely accepted database component
- 1985- Object-oriented DBMS develops.
- 1990s- Incorporation of object-orientation in relational DBMS.
- 1991- Microsoft ships MS access, a personal DBMS, and that displaces all other personal DBMS products.
- 1995: First Internet database applications
- 1997: XML applied to database processing. Many vendors begin to integrate XML into DBMS products.

1.2 Applications of DBMS:

There are different fields where a database management system is utilized.



Following are a few applications which utilize the information base administration framework:

1. **Railway Reservation System –**

In the rail route reservation framework, the information base is needed to store

the record or information of ticket appointments, status about train's appearance, and flight. Additionally, if trains get late, individuals become acquainted with it through the information base update.

2. Library Management System –

There are lots of books in the library so; it is difficult to store the record of the relative multitude of books in a register or duplicate. Along these lines, the data set administration framework (DBMS) is utilized to keep up all the data identified with the name of the book, issue date, accessibility of the book, and its writer.

3. Banking –

Database the executive's framework is utilized to store the exchange data of the client in the information base.

4. Education Sector –

Presently, assessments are led online by numerous schools and colleges. They deal with all assessment information through the data set administration framework (DBMS). In spite of that understudy's enlistments, subtleties, grades, courses, expense, participation, results, and so forth all the data is put away in the information base.

5. Credit card exchanges –

The database Management framework is utilized for buying on charge cards and age of month to month proclamations.

6. Social Media Sites –

We all utilize online media sites to associate with companions and to impart our perspectives to the world. Every day, many people pursue these online media accounts like Pinterest, Facebook, Twitter, and Google in addition to. By the utilization of the data set administration framework, all the data of clients are put away in the information base and, we become ready to interface with others.

7. Broadcast communications –

Without DBMS any media transmission organization can't think. The Database the executive's framework is fundamental for these organizations to store the call subtleties and month to month postpaid bills in the information base.

8. Account –

The information base administration framework is utilized for putting away data about deals, holding and acquisition of monetary instruments, for example, stocks and bonds in a data set.

9. Online Shopping –

These days, web-based shopping has become a major pattern. Nobody needs to visit the shop and burn through their time. Everybody needs to shop through web based shopping sites, (for example, Amazon, Flip kart, Snap deal) from home. So all the items are sold and added uniquely with the assistance of the information base administration framework (DBMS). Receipt charges, installments, buy data these are finished with the assistance of DBMS.

10. Human Resource Management –

Big firms or organizations have numerous specialists or representatives working

under them. They store data about worker's compensation, assessment, and work with the assistance of an information base administration framework (DBMS).

11. **Manufacturing –**

Manufacturing organizations make various kinds of items and deal with them consistently. To keep the data about their items like bills, acquisition of the item, amount, inventory network the executives, information base administration framework (DBMS) is utilized.

12. **Airline Reservation System –**

This framework is equivalent to the railroad reservation framework. This framework additionally utilizes an information base administration framework to store the records of flight takeoff, appearance, and defer status.

13. **Healthcare:** DBMS is used in healthcare to manage patient data, medical records, and billing information.

14. **Data retrieval:** DBMS provides a way to retrieve data quickly and easily using search queries.

15. **Data manipulation:** DBMS provides tools to manipulate data, such as sorting, filtering, and aggregating data.

16. **Security:** DBMS provides security features to ensure that only authorized users have access to the data.

17. **Data backup and recovery:** DBMS provides tools to back up data and recover it in case of system failures or data loss.

18. **Multi-user access:** DBMS allows multiple users to access and modify data simultaneously.

19. **Reporting and analysis:** DBMS provides tools to generate reports and analyze data to gain insights and make informed decisions.

1.3 Purpose:

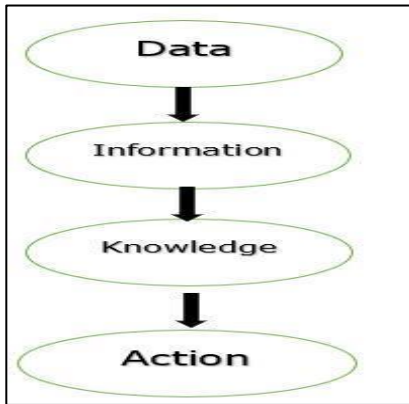
The Database Management System (DBMS) is defined as a software system that allows the user to define, create and maintain the database and provide control access to the data.

It is a collection of programs used for managing data and simultaneously it supports different types of users to create, manage, retrieve, update and store information.

The purpose of DBMS is to transform the following –

1. Data into information.
2. Information into knowledge.
3. Knowledge of the action.

The diagram given below explains the process as to how the transformation of data to information to knowledge to action happens respectively in the DBMS –



Previously, the database applications were built directly on top of the file system.

Drawbacks in File System:

There are so many drawbacks in using the file system. These are mentioned below –

1. **Data redundancy and inconsistency:** Different file formats, duplication of information in different files.
2. **Difficulty in accessing data:** To carry out new task we need to write a new program.
3. **Data Isolation** – Different files and formats.
4. **Integrity** problems.
5. **Atomicity of updates** – Failures leave the database in an inconsistent state. For example, the fund transfer from one account to another may be incomplete.
6. **Concurrent** access by multiple users.
7. **Security** problems.

Database system offers so many solutions to all these problems.

Uses of DBMS:

The main uses of DBMS are as follows –

- Data independence and efficient access of data.
- Application Development time reduces.
- Security and data integrity.
- Uniform data administration.
- Concurrent access and recovery from crashes.

1.4 View of Data in DBMS:

View of data - DBMS narrates how the data is visualized at each level of data abstraction.

Data abstraction allows developers to keep complex data structures away from the users.

The developers achieve this by hiding the complex data structures through **levels of abstraction**.

There is one more feature that should be kept in mind i.e. the **data independence**. While changing the data schema at one level of the database must not modify the data schema at the next level. In this section, we will discuss the view of data in DBMS with data abstraction, data independence and data schema in detail.

- a. Data Abstraction
- b. Data Independence
- c. Instance and Schema

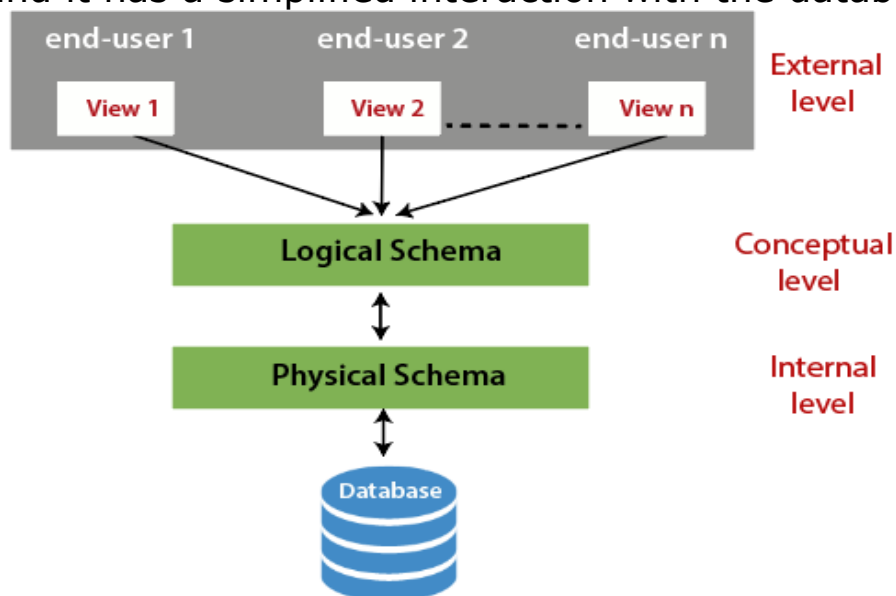
a. Data Abstraction:

Data abstraction is **hiding the complex data structure** in order to **simplify the user's interface** of the system. It is done because many of the users interacting with the database system are not that much computer trained to understand the complex data structures of the database system.

To achieve data abstraction, we will discuss a **Three-Schema architecture** which abstracts the database at three levels discussed below:

i. Three-Schema Architecture:

The main objective of this architecture is to have an effective separation between the **user interface** and the **physical database**. So, the user never has to be concerned regarding the internal storage of the database and it has a simplified interaction with the database system.



The three-schema architecture defines the view of data at three levels:

1. Physical level (internal level)
2. Logical level (conceptual level)
3. View level (external level)

1. Physical Level/ Internal Level

The physical or the internal level schema describes **how the data is stored in the hardware**. It also describes how the data can be accessed. The physical level shows the data abstraction at the lowest level and it has **complex data structures**. Only the database administrator operates at this level.

2. Logical Level/ Conceptual Level

It is a level above the physical level. Here, the data is stored in the form of the **entity set, entities, their data types, the relationship** among the entity sets, **user operations** performed to retrieve or modify the data and certain **constraints on the data**. Well adding constraints to the view of data adds the security. As users are restricted to access some particular parts of the database. It is the developer and database administrator who operates at the logical or the conceptual level.

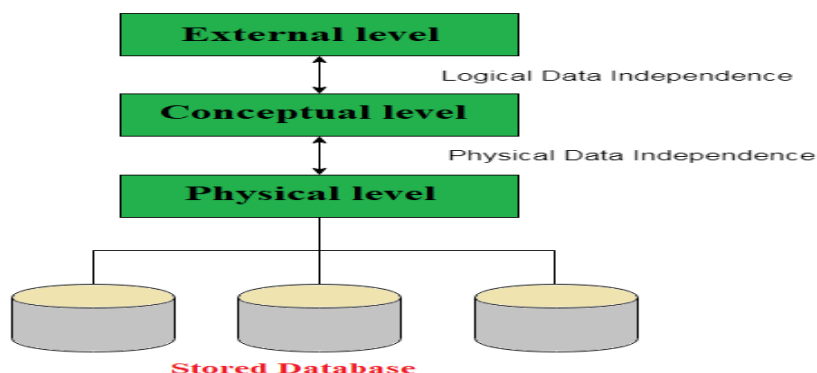
4. View Level/ User level/ External level

It is the highest level of data abstraction and exhibits only a part of the whole database. It exhibits the data in which the user is interested. The view level can describe many views of the same data. Here, the user retrieves the information using different applications from the database.

Data Independence

Data independence defines the extent to which the data schema can be changed at one level without modifying the data schema at the next level. Data independence can be classified as shown below:

Data Independence in DBMS



a. Logical Data Independence:

Logical data independence describes the degree up to which the logical or conceptual schema can be changed without modifying the external schema.

Well, the changes to data schema at the logical level are made either to **enlarge** or **reduce** the database by adding or deleting more entities, entity sets, or changing the constraints on data.

b. Physical Data Independence:

Physical data independence defines the extent up to which the data schema can be changed at the physical or internal level without modifying the data schema at logical and view level. Well, the physical schema is changed if we add additional storage to the system or we reorganize some files to enhance the retrieval speed of the records.

Instances and Schemas:

Instances:

We can define an instance as the information stored in the database at a particular point of time. As we discussed above the database comprises several entity sets and the relationship between them. Now, the data in the database keeps on changing with time. As we keep inserting or deleting the data to and from the database. Now, at a particular time if we retrieve any information from the database then that corresponds to an instance.

Schema:

Whenever we talk about the database the developers have to deal with the definition of database and the data in the database. The definition of a database comprises the description of what data it would contain and what would be the relationship between the data. This definition is the database schema.

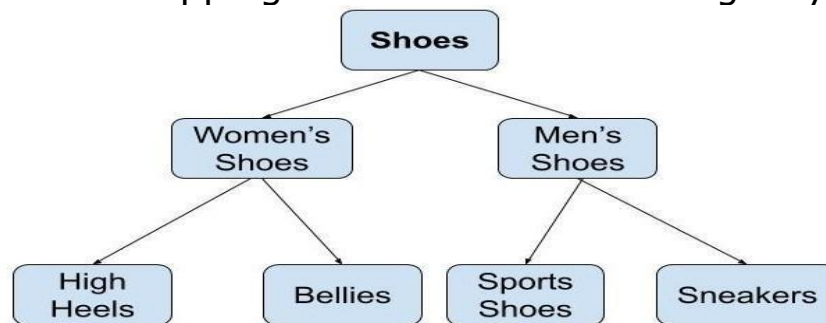
1.5 Data Model:

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 organization can communicate and understand it. Though there are many data models being used nowadays, the Relational model is the most widely used model. Some of the Data Models in DBMS are:

1. Hierarchical Model
2. Network Model
3. Entity-Relationship Model
4. Relational Model
5. Object-Oriented Data Model
6. Object-Relational Data Model
7. Flat Data Model
8. Semi-Structured Data Model
9. Associative Data Model
10. Context Data Model

1. Hierarchical Model:

The Hierarchical Model was the first DBMS model. This model organizes the data in the hierarchical tree structure. The hierarchy starts from the root which has root data and then it expands in the form of a tree adding a child node to the parent node. This model easily represents some of the real-world relationships like food recipes, sitemap of a website etc. **Example:** We can represent the relationship between the shoes present on a shopping website in the following way:



Hierarchical Model

Features of a Hierarchical Model:

One-to-many relationship: The data here is organized in a tree-like structure where the one-to-many relationship is between the data types. Also, there can be only one path from parent to any node.

Example: In the above example, if we want to go to the node *sneakers* we only have one path to reach there i.e. through the men's shoes node.

1. **Parent-Child Relationship:** Each child node has a parent node but a parent node can have more than one child node. Multiple parents are not allowed.
2. **Deletion Problem:** If a parent node is deleted then the child node is automatically deleted.
3. **Pointers:** Pointers are used to link the parent node with the child node and are used to navigate between the stored data. *Example:* In the above

example the ' *shoes* ' node points to the two other nodes ' *women shoes* ' node and ' *men's shoes* ' node.

Advantages of Hierarchical Model:

- It is very simple and fast to traverse through a tree-like structure.
- Any change in the parent node is automatically reflected in the child node so the integrity of data is maintained.

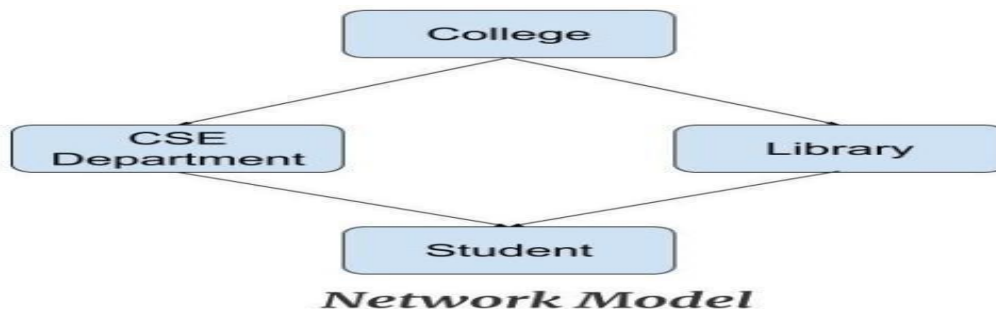
Disadvantages of Hierarchical Model:

- Complex relationships are not supported.
- As it does not support more than one parent of the child node so if we have some complex relationship where a child node needs to have two parent nodes then that can't be represented using this model.
- If a parent node is deleted, then the child node is automatically deleted.

2. 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.



Features of a Network Model:

1. **Ability to merge more Relationships:** In this model, as there are more relationships so data is more related. This model has the ability to manage one-to-one relationships as well as many-to-many relationships.
2. **Many paths:** As there are more relationships so there can be more than one path to the same record. This makes data access fast and simple.
3. **Circular Linked List:** The operations on the network model are done with the help of the circular linked list. The current position is maintained with the help of a program and this position navigates through the records according to the relationship.

Advantages of Network Model:

- The data can be accessed faster as compared to the hierarchical model. This is because the data is more related in the network model and there can be more than one path to reach a particular node. So the data can be accessed in many ways.
- As there is a parent-child relationship so data integrity is present. Any change in parent record is reflected in the child record.

Disadvantages of Network Model:

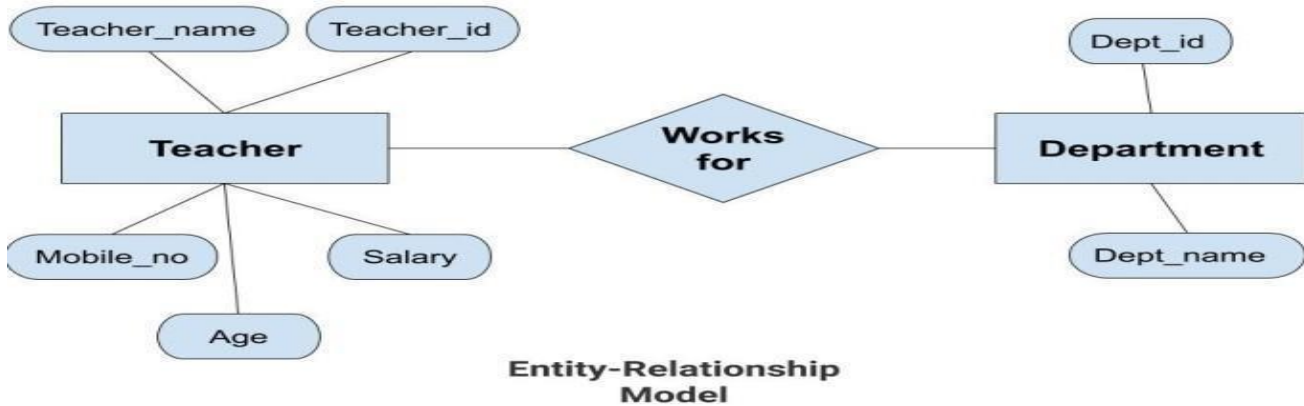
- As more and more relationships need to be handled the system might get complex. So, a user must have detailed knowledge of the model to work with the model.
- Any change like updation, deletion, insertion is very complex.

3. Entity-Relationship Model:

Entity-Relationship Model or simply ER Model is a high-level data model diagram. In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand. It is also very easy for the developers to understand the system by just looking at the ER diagram. We use the ER diagram as a visual tool to represent an ER Model. ER diagram has the following three components:

- **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:



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.

Features of ER Model

- **Graphical Representation for Better Understanding:** It is very easy and simple to understand so it can be used by the developers to communicate with the stakeholders.
- **ER Diagram:** ER diagram is used as a visual tool for representing the model.
- **Database Design:** This model helps the database designers to build the database and is widely used in database design.

Advantages of ER Model

- **Simple:** Conceptually ER Model is very easy to build. If we know the relationship between the attributes and the entities we can easily build the ER Diagram for the model.
- **Effective Communication Tool:** This model is used widely by the database designers for communicating their ideas.
- **Easy Conversion to any Model:** This model maps well to the relational model and can be easily converted relational model by converting the ER model to the table. This model can also be converted to any other model like network model, hierarchical model etc.

Disadvantages of ER Model

- **No industry standard for notation:** there is an industry standard for developing an ER model. So one developer might use notations which are not understood by other developers.
- **Hidden information:** Some information might be lost or hidden in the ER model. As it is a high-level view, there are chances that some details of information might be hidden.

4. Relational Model

The 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.

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

- **Tuple:** 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.

Advantages of Relational Model

- **Simple:** This model is simpler as compared to the network and hierarchical model.
- **Scalable:** This model can be easily scaled as we can add as many rows and columns we want.
- **Structural Independence:** We can make changes in database structure without changing the way to access the data. When we can make changes to the database structure without affecting the

capability of DBMS to access the data we can say that structural independence has been achieved.

Disadvantages of Relational Model

- **Hardware Overheads:** For hiding the complexities and making things easier for the user this model requires more powerful hardware computers and data storage devices.
- **Bad Design:** As the relational model is very easy to design and use. So the users don't need to know how the data is stored in order to access it. This ease of design can lead to the development of a poor database which would slow down if the database grows. But all these disadvantages are minor as compared to the advantages of the relational model. These problems can be avoided with the help of proper implementation and organization.

1.16 Relationship Set In DBMS:

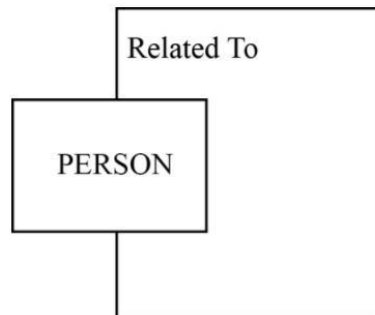
The number of entity types which took part in the entity relationship is called the degree of relationships.

There are three different types of degree of relationships, they are as follows –

- Unary relationship
- Binary relationship
- Ternary relationship

Unary relationship:

It is the relationship between the instances of a single entity type. It is also called a recursive relationship.



Example

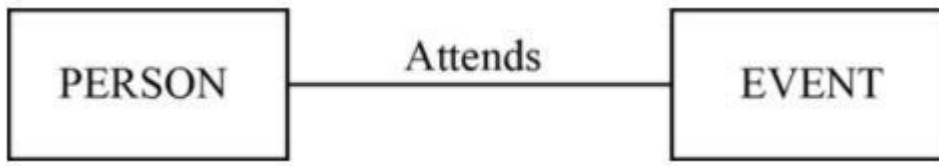
A person is married is a one-to-one relationship between the instances of Person entity type of unary relationship.

Binary relationship

It is the relationship between the instances of two different entity types. Two entities will participate in the relationship.

For Example

Person and events are two different entity types which are related by using the relationship called "attends".



Ternary relationship

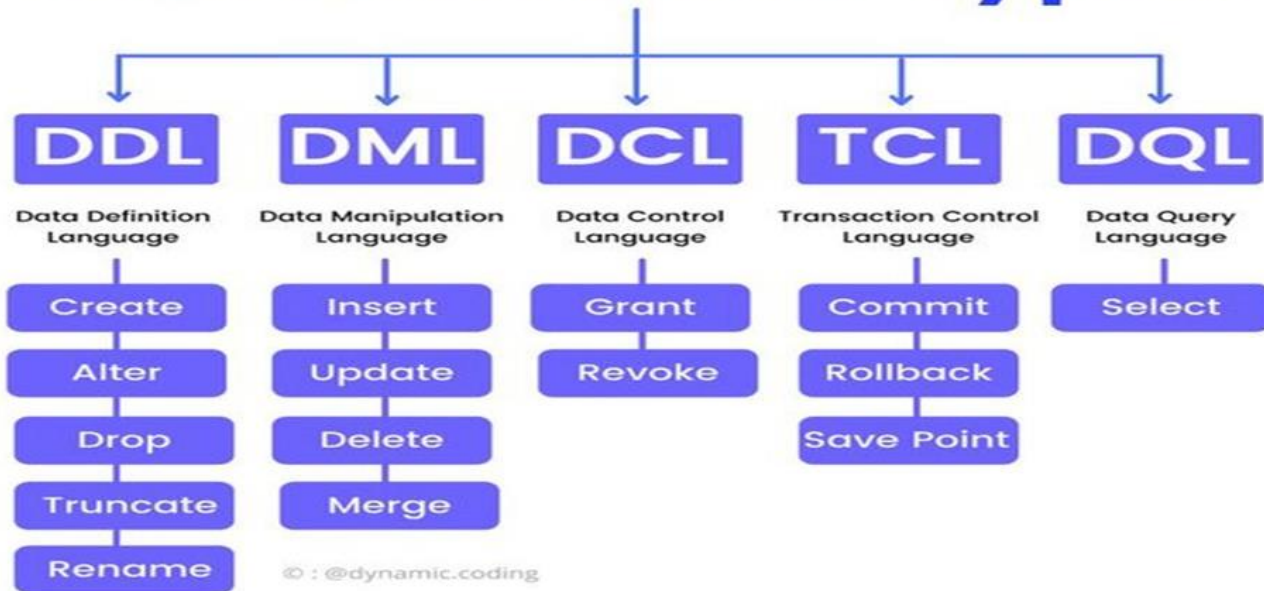
A simultaneous relationship between the instances of three entity types with unique attributes is called a ternary relationship.

Example

Consultant, client and contract are three different entities with different attributes. These three entities are related with a single relationship called "signs".

1.6 Database Languages:

SQL Command Types



1. Data Definition Language (DDL)

- **DDL** stands for **Data Definition Language**. 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 **Data Manipulation Language**. 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 **Data Control Language**. 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 originality since the last Commit.

1.7 Database Access for applications Programs:

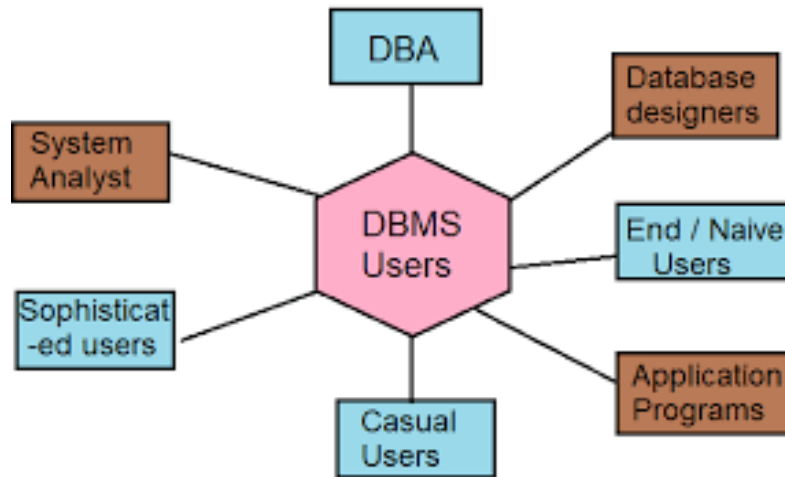
The database management system is a bridge between the application program, (that determines what data are needed and how they are processed), and the operating system of the computer, which is responsible for placing data on the magnetic storage devices.

To retrieve data from the database, the following operations are performed internally:

- A user issues an access request, using some application program or data manipulation language.
- The application program determines what data are needed and communicates the need to the database management system.
- The DBMS intercepts the request and interprets it .
- . The DBMS inspects, in turn, the external schema, the external/conceptual mapping, the conceptual schema, the conceptual internal mapping, and storage structure definition.
- The database management system instructs the operating system to locate and retrieve the data from the specific location on the magnetic disk (or whatever device it is stored on).
- A copy of the data is given to the application program for processing.

1.8 database Users and Administrator:

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



1. **Database Administrator (DBA) :** Database Administrator (DBA) is a person/team who defines the schema and also controls the 3 levels of the database. The DBA will then create a new account id and password for the user if he/she needs 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 database. DBA is responsible for the problems such as security breaches and poor system response time.
 - DBA also monitors the recovery and backup and provides technical support.
 - The DBA has a DBA account in the DBMS which is called a system or super user account.
 - DBA repairs damage caused due to hardware and/or software failures.
 - DBA is the one having privileges to perform DCL (Data Control Language) operations such as GRANT and REVOKE, to allow/restrict a particular user from accessing the database.
2. **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.
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 analysts, who are familiar with the database. They can develop their own database applications according to their

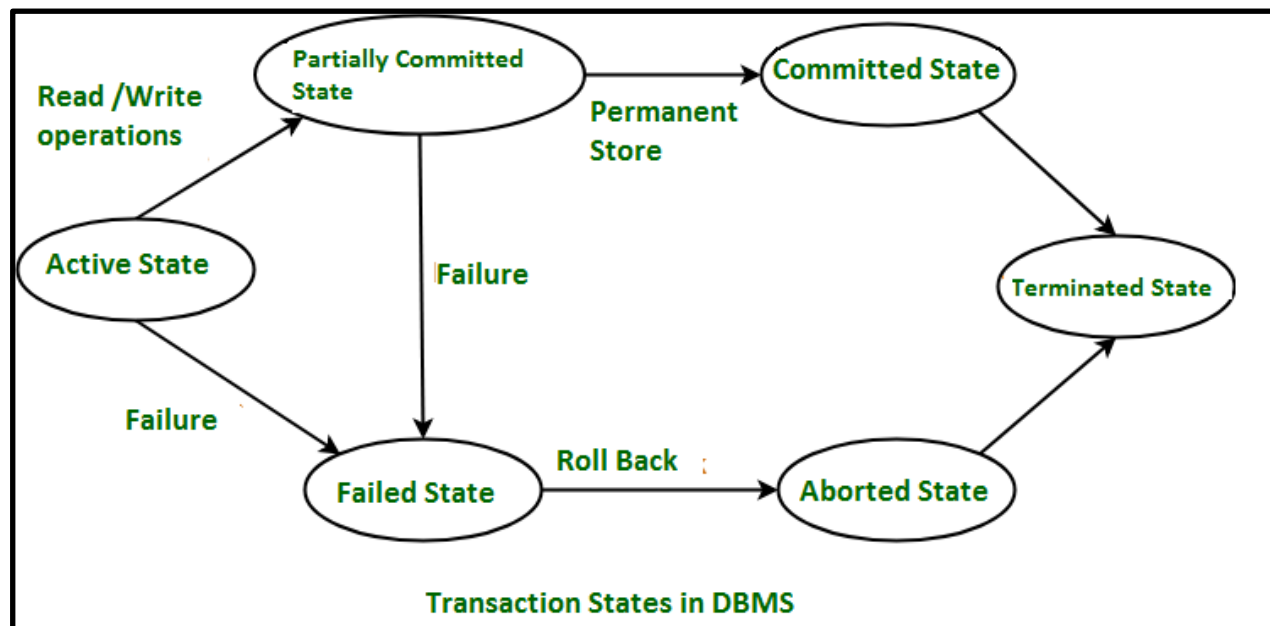
requirements. They don't write the program code but they interact with the database by writing SQL queries directly through the query processor.

5. **Database Designers :** Database Designers are the users who design the structure of a 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 the responsibility of Database Designers to understand the requirements of different user groups and then create a design which satisfies the needs of all the user groups.
6. **Application Programmers :** Application Programmers also referred as System Analysts or simply Software Engineers, are the back-end programmers who write the code for the application programs. They are 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 a set of programs called "canned transactions" for the Naive (parametric) users in order to interact with the database.
7. **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.
8. **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.

1.10 Transaction Management :

Transaction States :

Transactions can be implemented using SQL queries and Servers. In the below-given diagram, you can see how transaction states work.



Transaction States

The transaction has four properties. These are used to maintain consistency in a database, before and after the transaction.

Property of Transaction:

1. Atomicity
2. Consistency
3. Isolation
4. Durability

Atomicity:

- it states that all operations of the transaction take place at once if not, the transactions aborted.
- There is no midway, i.e., the transaction cannot occur partially. Each transaction is treated as one unit and either run to completion or is not executed at all.
- Atomicity involves the following two operations:
- **Abort:** If a transaction aborts, then all the changes made are not visible.
- **Commit:** If a transaction commits then all the changes made are visible.

Consistency:

- The integrity constraints are maintained so that the database is consistent before and after the transaction.
- The execution of a transaction will leave a database in either its prior stable state or anew stable state.
- The consistent property of the database states that every transaction sees a consistent database instance.
- The transaction is used to transform the database from one consistent state to another consistent state.

Isolation:

- It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.
- In isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends.
- The concurrency control subsystem of the DBMS enforced the isolation property

Durability:

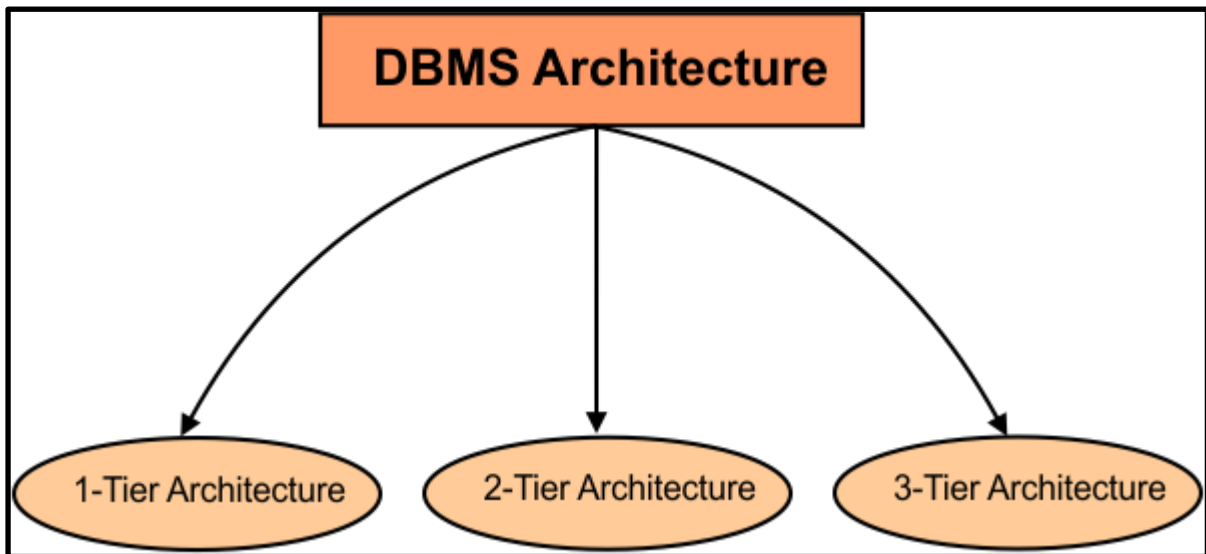
- The durability property is used to indicate the performance of the database's consistent state. It states that the transaction made the permanent changes.
- They cannot be lost by the erroneous operation of a faulty transaction or by the system failure. When a transaction is completed, then the database reaches a state known as the consistent state. That consistent state cannot be lost, even in the event of a system's failure.
- The recovery subsystem of the DBMS has the responsibility of Durability property.

1.11 Database Architecture :

A Database stores a lot of critical information to access data quickly and securely. Hence it is important to select the correct architecture for efficient data management. DBMS Architecture helps users to get their requests done while connecting to the database. We choose database architecture depending on several factors like the size of the database, number of users, and relationships between the users. There are two types of database models that we generally use, logical model and physical model. Several types of architecture are there in the database which we will deal with in the next section.

Types of DBMS Architecture:

There are several types of DBMS Architecture that we use according to the usage requirements.

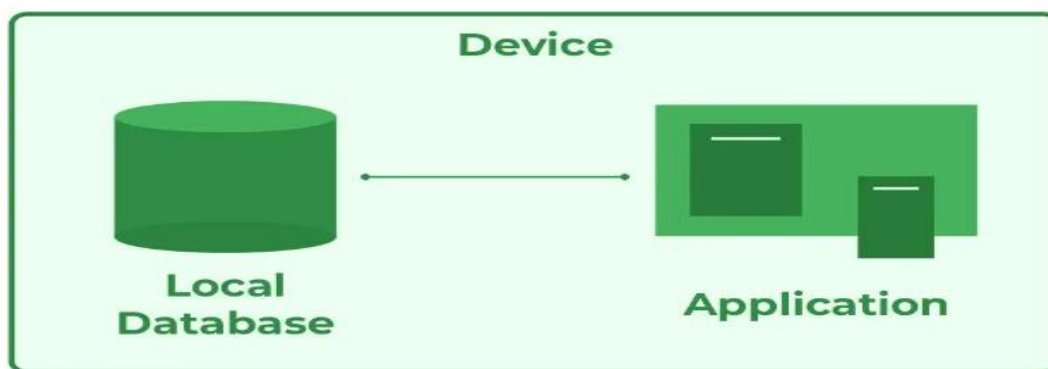


Types of DBMS Architecture are discussed here.

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

1-Tier Architecture :

In 1-Tier Architecture the database is directly available to the user, the user can directly sit on the DBMS and use it, that is, the client, server, and Database are all present on the same machine. For Example: to learn SQL we set up an SQL server and the database on the local system. This enables us to directly interact with the relational database and execute operations. The industry won't use this architecture; they logically go for 2-Tier and 3-Tier Architecture.



DBMS 1-Tier Architecture

Advantages of 1-Tier Architecture

Below mentioned are the advantages of 1-Tier Architecture.

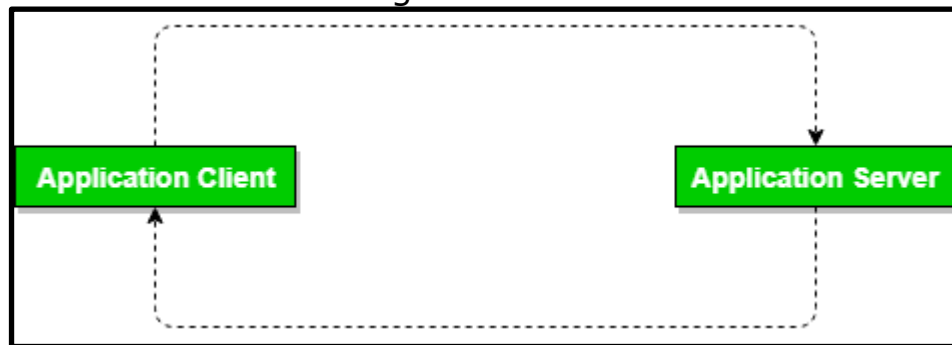
- **Simple Architecture:** 1-Tier Architecture is the most simple architecture to set up, as only a single machine is required to maintain it.
- **Cost-Effective:** No additional hardware is required for implementing 1-Tier Architecture, which makes it cost-effective.

- **Easy to Implement:** 1-Tier Architecture can be easily deployed, and hence it is mostly used in small projects.

2-Tier Architecture:

The 2-tier architecture is similar to a basic client-server model. The application at the client end directly communicates with the database on the server side. APIs like ODBC and JDBC are used for this interaction. The server side is responsible for providing query processing and transaction management functionalities. On the client side, the user interfaces and application programs are run. The application on the client side establishes a connection with the server side in order to communicate with the DBMS.

An advantage of this type is that maintenance and understanding are easier, and compatible with existing systems. However, this model gives poor performance when there are a large number of users.



DBMS 2-Tier Architecture

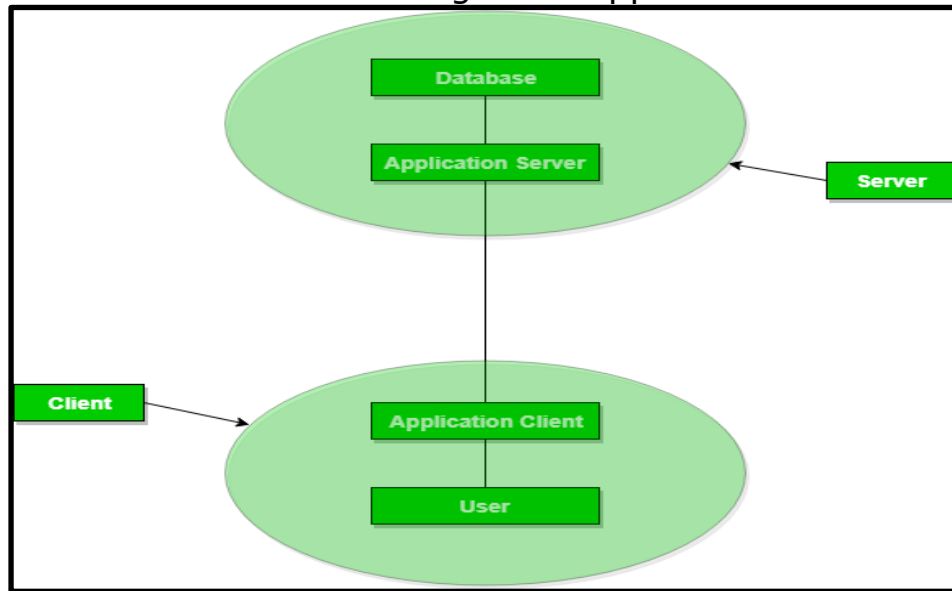
Advantages of 2-Tier Architecture :

- **Easy to Access:** 2-Tier Architecture makes easy access to the database, which makes fast retrieval.
- **Scalable:** We can scale the database easily, by adding clients or by upgrading hardware.
- **Low Cost:** 2-Tier Architecture is cheaper than 3-Tier Architecture and Multi-Tier Architecture.
- **Easy Deployment:** 2-Tier Architecture is easy to deploy than 3-Tier Architecture.
- **Simple:** 2-Tier Architecture is easily understandable as well as simple because of only two components.

3-Tier Architecture :

In 3-Tier Architecture, there is another layer between the client and the server. The client does not directly communicate with the server. Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place. This intermediate layer acts as a medium for the exchange of

partially processed data between the server and the client. This type of architecture is used in the case of large web applications.



DBMS 3-Tier Architecture

Advantages of 3-Tier Architecture :

- **Enhanced scalability:** Scalability is enhanced due to distributed deployment of application servers. Now, individual connections need not be made between the client and server.
- **Data Integrity:** 3-Tier Architecture maintains Data Integrity. Since there is a middle layer between the client and the server, data corruption can be avoided/removed.
- **Security:** 3-Tier Architecture Improves Security. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.

Disadvantages of 3-Tier Architecture

- **More Complex:** 3-Tier Architecture is more complex in comparison to 2-Tier Architecture. Communication Points are also doubled in 3-Tier Architecture.
- **Difficult to Interact:** It becomes difficult for this sort of interaction to take place due to the presence of middle layers.

1.12 Storage Manager:

Storage Manager in a Database Management System in DBMS :

The Storage Manager is a crucial component of a Database Management System (DBMS) that handles the storage, retrieval, and management of data on secondary storage devices. It acts as a bridge between the higher-level components of the DBMS, such as the query processor and transaction manager, and the physical storage media, typically disk drives.

At its core, the Storage Manager is responsible for efficiently organizing and managing the persistent storage of data. It provides an interface between the logical representation of data within the DBMS and the physical storage devices where the data resides. This involves translating high-level data operations and requests from the DBMS into low-level disk operations, ensuring that data is stored, retrieved, and managed reliably.

Components of Storage Manager in DBMS :

1. Buffer Manager

The buffer manager is responsible for caching data in main memory to optimize disk I/O operations. It manages a buffer pool, which is a portion of memory used to store frequently accessed data pages from the disk. Key functions of the buffer manager include:

- **Buffer Allocation:** Allocating and deallocating memory buffers to hold data pages.
- **Page Replacement:** Implementing policies to evict pages from the buffer pool when space is needed for new pages.
- **Disk I/O Operations:** Reading data pages from the disk into the buffer pool and writing modified pages back to the disk.

2. File Manager

The file manager is responsible for managing the physical storage of data on the disk. It provides operations to create, delete, open, close, read, and write data files. Key functions of the file manager include:

- **File Organization:** Defining the structure and organization of data files on the disk, such as sequential, indexed, or hashed organization.
- **File Allocation:** Allocating space on the disk to store data files efficiently.
- **Record Management:** Managing the storage and retrieval of individual records within data files.
- **Access Control:** Enforcing access control mechanisms to ensure data integrity and security.

3. Disk Space Manager

The disk space manager handles the allocation and deallocation of disk space for data storage. It keeps track of free space and manages the space allocation structures on the disk. Key functions of the disk space manager include:

- **Space Allocation:** Allocating disk blocks to store data files and managing the data structures, such as linked lists or bitmaps, that track the allocated and free blocks.
- **Space Reclamation:** Reclaiming disk space when data is deleted or updated, ensuring efficient space utilization.
- **Space Management Policies:** Implementing policies to optimize disk space allocation and fragmentation.

4. Transaction Manager

The transaction manager ensures the atomicity, consistency, isolation, and durability (ACID) properties of database transactions. It handles concurrent access and recovery in a multi-user environment. Key functions of the transaction manager include:

- **Transaction Control:** Initiating, executing, and coordinating the execution of database transactions.
- **Concurrency Control:** Managing concurrent access to data to ensure data consistency and prevent conflicts.
- **Transaction Recovery:** Providing mechanisms to recover the database to a consistent state in the event of failures or system crashes.

5. Disk Manager

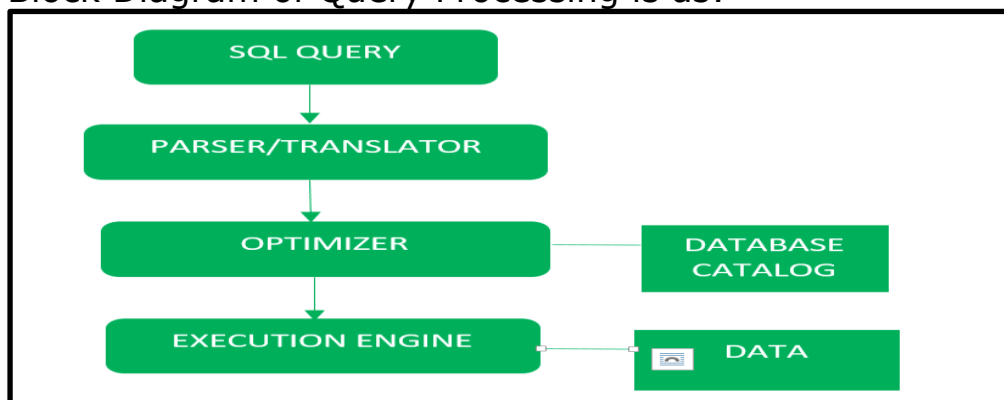
The disk manager interacts directly with the operating system and hardware to perform low-level disk operations. It provides an interface to read and write disk blocks and manages disk-level I/O operations efficiently. Key functions of the disk manager include:

- **Disk I/O Operations:** Reading and writing disk blocks based on the requests from the buffer manager and file manager.
- **Caching and Prefetching:** Implementing techniques to optimize disk I/O performance, such as caching frequently accessed blocks and prefetching data.
- **Error Handling:** Handling disk-related errors, detecting and recovering from disk failures.

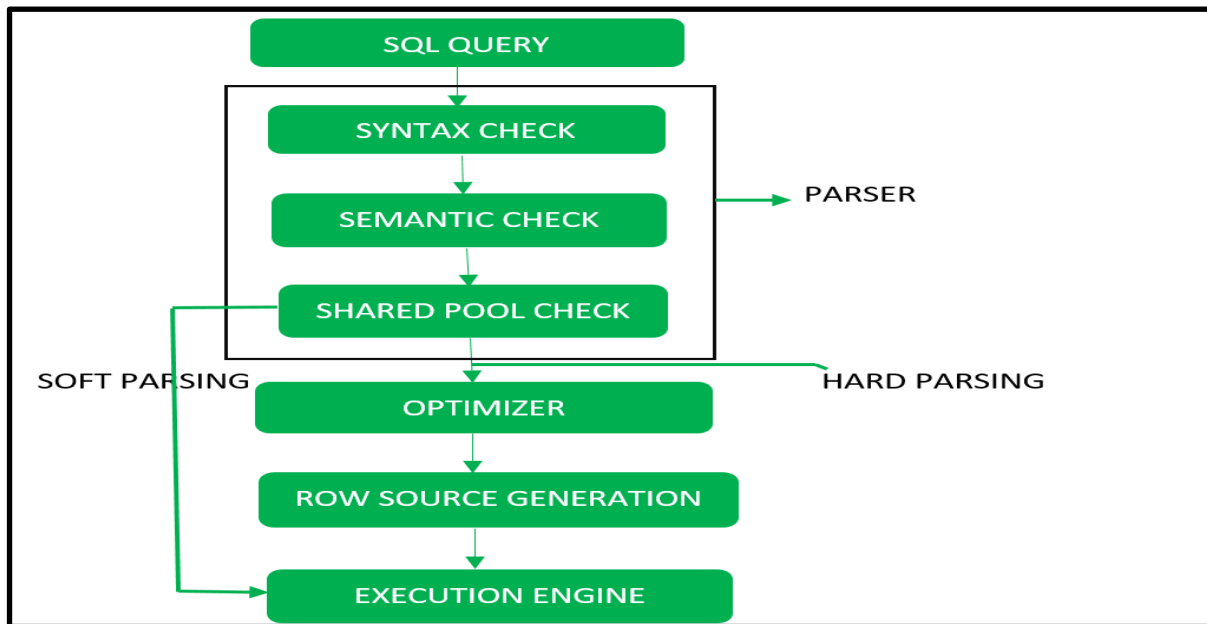
1.13 Query Processor:

Query Processing includes translations on high level Queries into low level expressions that can be used at physical level of file system, query optimization and actual execution of query to get the actual result.

Block Diagram of Query Processing is as:



Detailed Diagram is drawn as:



1.14 Database Design:

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Properly designed databases are easy to maintain, improve data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored.

The main objectives of database design in DBMS are to produce logical and physical designs models of the proposed database system.

The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.

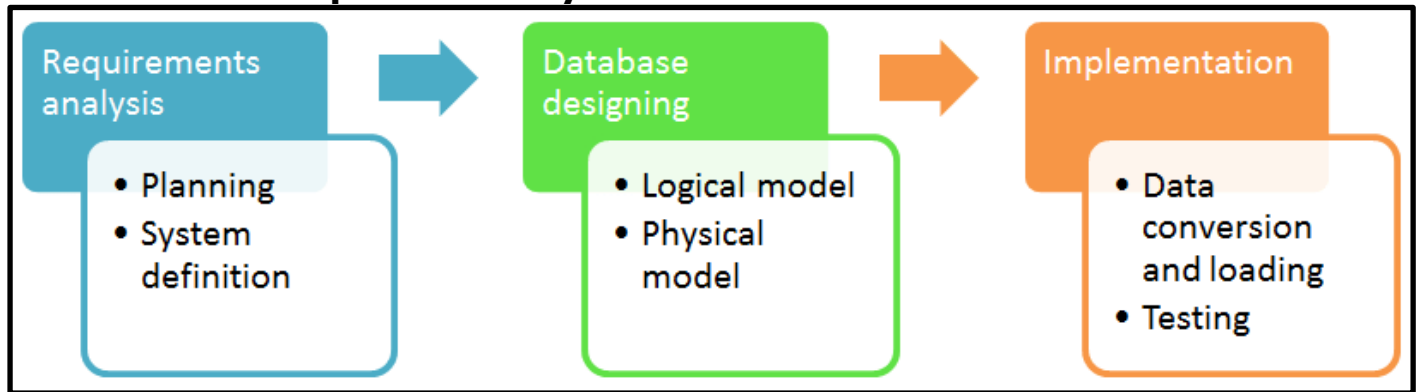
The physical data design model involves translating the logical DB design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

It helps produce database systems

1. That meet the requirements of the users
2. Have high performance.

Database design process in DBMS is crucial for **high performance** database system.

Database development life cycle



The database development life cycle has a number of stages that are followed when developing database systems. The steps in the development life cycle do not necessarily have to be followed religiously in a sequential manner. On small database systems, the process of database design is usually very simple and does not involve a lot of steps.

In order to fully appreciate the above diagram, let's look at the individual components listed in each step for an overview of the design process in DBMS.

1. Requirements analysis

- **Planning** – This stage of database design concepts are concerned with planning of entire Database Development Life Cycle. It takes into consideration the Information Systems strategy of the organization.
- **System definition** – This stage defines the scope and boundaries of the proposed database system.

2. Database designing

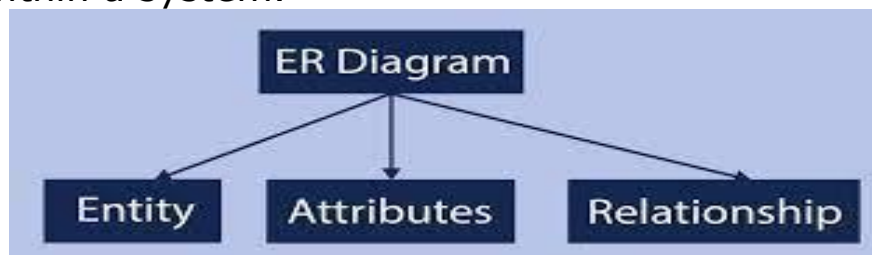
- **Logical model** – This stage is concerned with developing a database model based on requirements. The entire design is on paper without any physical implementations or specific DBMS considerations.
- **Physical model** – This stage implements the logical model of the database taking into account the DBMS and physical implementation factors.

3. Implementation

- **Data conversion and loading** – this stage of relational databases design is concerned with importing and converting data from the old system into the new database.
- **Testing** – this stage is concerned with the identification of errors in the newly implemented system. It checks the database against requirement specifications.

1.15 ER diagram and ER Models:

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.



The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies an enterprise schema that represents the overall logical structure of a database graphically.

The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, ER Diagram is the structural format of the database.






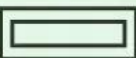
Why Use ER Diagrams In DBMS?

- ER diagrams are used to represent the E-R model in a database, which makes them easy to be converted into relations (tables).
- ER diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
- ER diagrams require no technical knowledge and no hardware support.
- These diagrams are very easy to understand and easy to create even for a naive user.
- It gives a standard solution for visualizing the data logically.

Symbols Used in ER Model

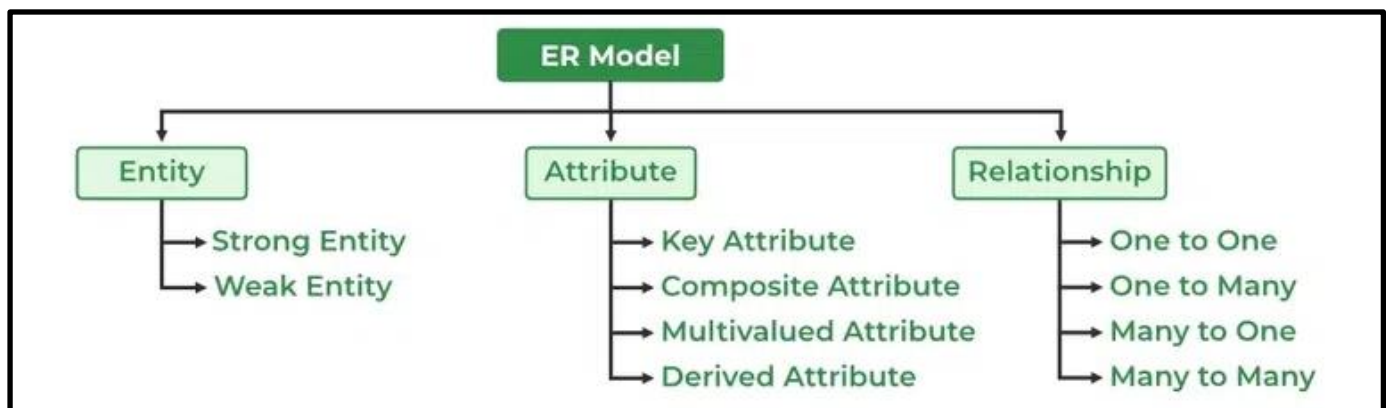
ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:

- Rectangles: Rectangles represent Entities in ER Model.
- Ellipses: Ellipses represent Attributes in ER Model.
- Diamond: Diamonds represent Relationships among Entities.
- Lines: Lines represent attributes to entities and entity sets with other relationship types.
- Double Ellipse: Double Ellipses represent [Multi-Valued Attributes](#).
- Double Rectangle: Double Rectangle represents a Weak Entity.

Figures	Symbols	Represents
Rectangle		Entities in ER Model
Ellipse		Attributes in ER Model
Diamond		Relationships among Entities
Line		Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse		Multi-Valued Attributes
Double Rectangle		Weak Entity

Components of ER Diagram

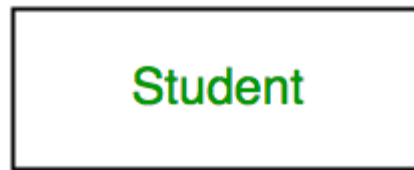
ER Model consists of Entities, Attributes, and Relationships among Entities in a Database System.



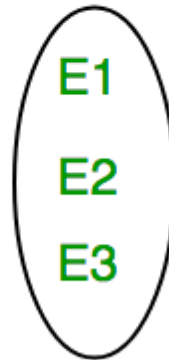
Entity:

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

Entity Set: An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



Entity Type



Entity Set

Entity Set

1. Strong Entity

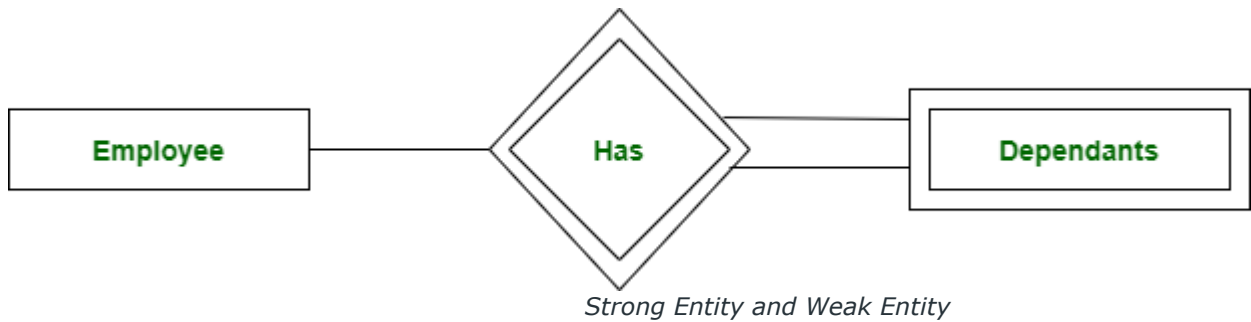
A [Strong Entity](#) is a type of entity that has a key Attribute. Strong Entity does not depend on other entities in the Schema. It has a primary key that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

2. Weak Entity

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can't be defined. These are called [Weak Entity types](#).

For Example, A company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents don't exist without the employee. So Dependent will be a Weak Entity Type and Employee will be Identifying Entity type for Dependent, which means it is Strong Entity Type.

A weak entity type is represented by a Double Rectangle. The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



Attributes :

[Attributes](#) are the properties that define the entity type. For example, Roll_No, Name, DOB, Age, Address, and Mobile_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.



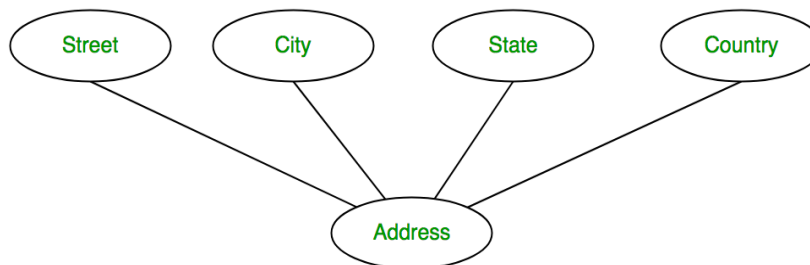
1. Key Attribute

The attribute which uniquely identifies each entity in the entity set is called the key attribute. For example, Roll_No will be unique for each student. In the ER diagram, the key attribute is represented by an oval with underlying lines.



2. Composite Attribute

An attribute composed of many other attributes is called a composite attribute. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country. In ER diagram, the composite attribute is represented by an oval comprising of ovals.



3. Multivalued Attribute :

An attribute consisting of more than one value for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.



Multivalued Attribute

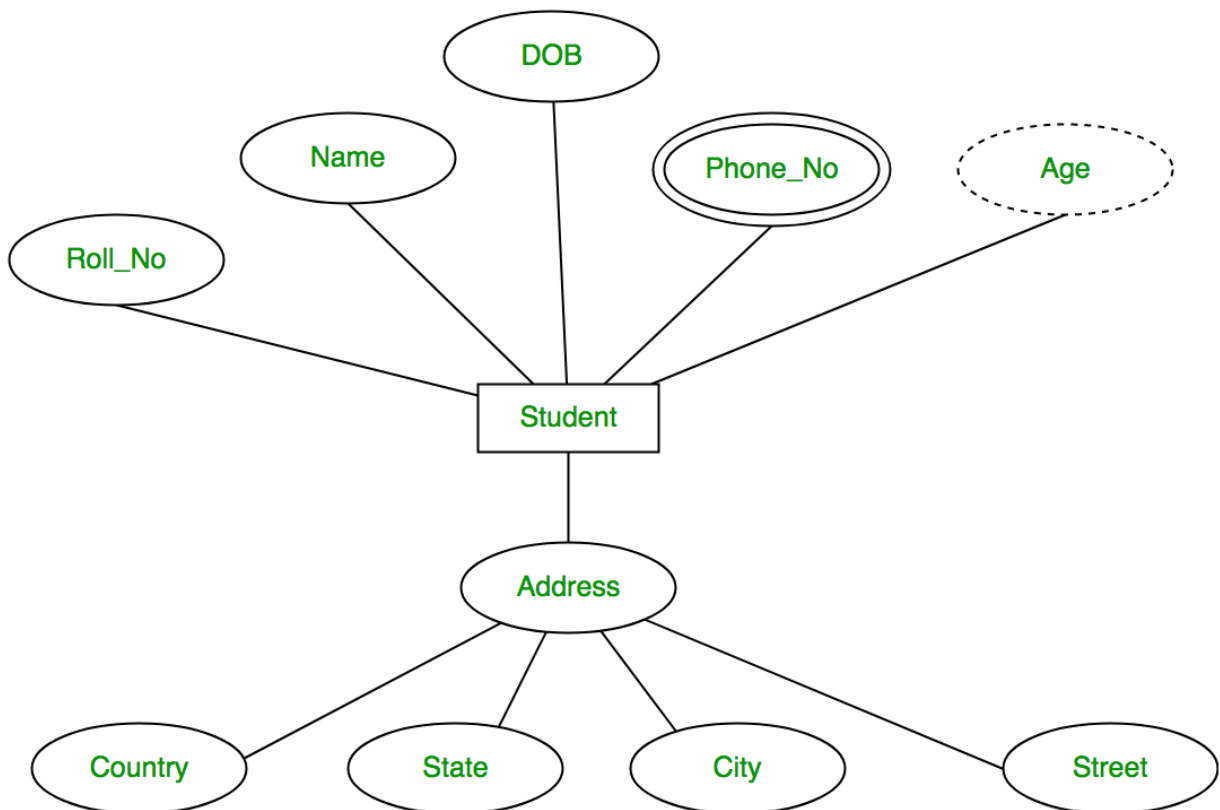
4. Derived Attribute

An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.



Derived Attribute

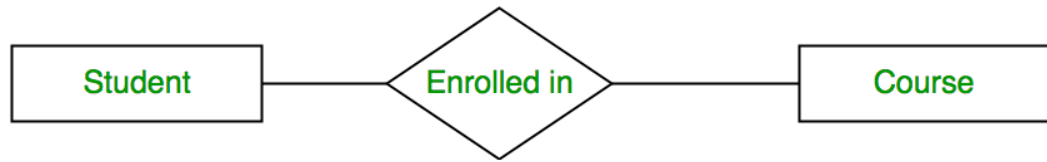
The Complete Entity Type Student with its Attributes can be represented as:



Entity and Attributes

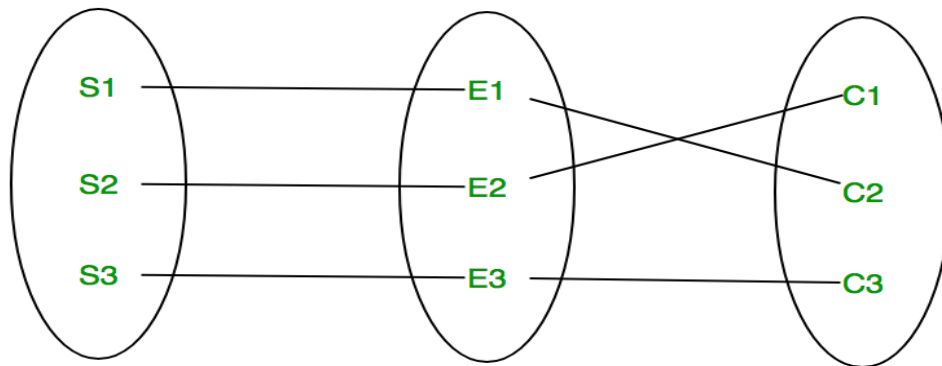
Relationship Type and Relationship Set :

A Relationship Type represents the association between entity types. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course. In the ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



Entity-Relationship Set

A set of relationships of the same type is known as a relationship set. The following relationship set depicts S1 as enrolled in C2, S2 as enrolled in C1, and S3 as registered in C3.

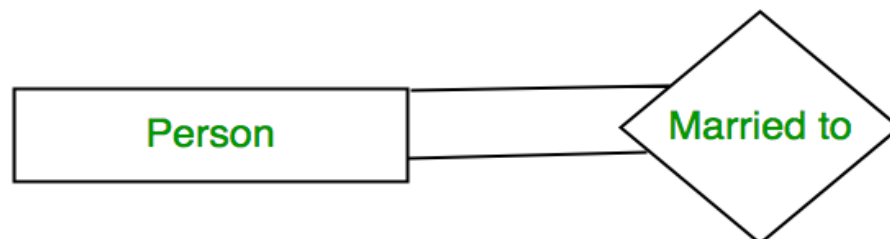


Relationship Set

Degree of a Relationship Set:

The number of different entity sets participating in a relationship set is called the degree of a relationship set.

1. Unary Relationship: When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



Unary Relationship

2. Binary Relationship: When there are TWO entities participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.



Binary Relationship

3. n-ary Relationship: When there are n entities set participating in a relation, the relationship is called an n-ary relationship.

Cardinality

The number of times an entity of an entity set participates in a relationship set is known as **cardinality**. Cardinality can be of different types:

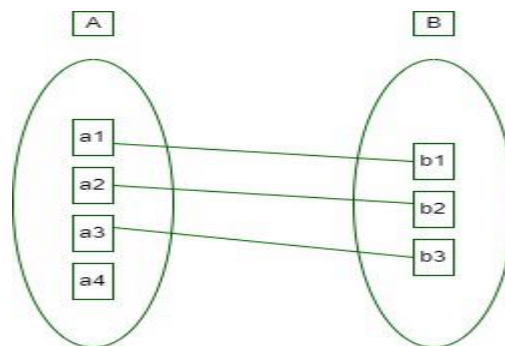
1. One-to-One: When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.

The total number of tables that can be used in this is 2.



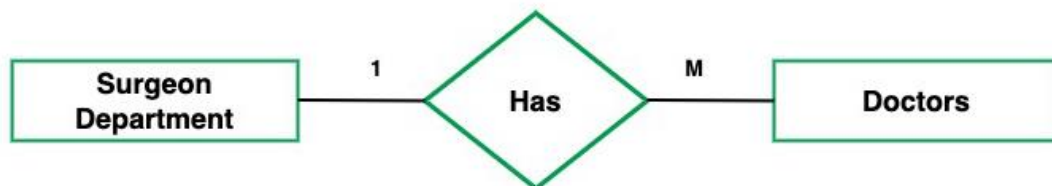
One-to-One Cardinality

Using Sets, it can be represented as:



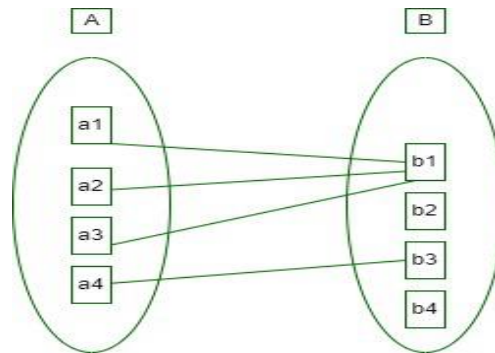
Set Representation of One-to-One

2. One-to-Many: In one-to-many mapping as well where each entity can be related to more than one relationship and the total number of tables that can be used in this is 2.



One to Many

Using sets, one-to-many cardinality can be represented as:



Set Representation of One-to-Many

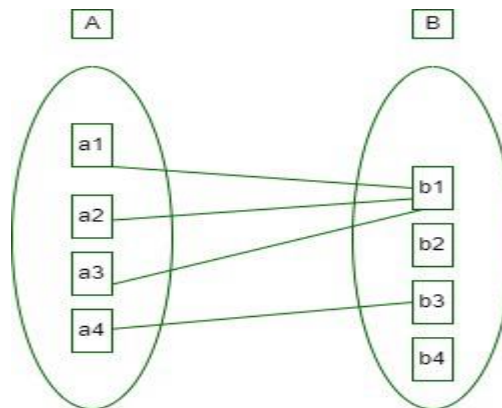
3. Many-to-One: When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1 . It means that for one course there can be n students but for one student, there will be only one course.

The total number of tables that can be used in this is 3.



Many-to-One Relationship

Using Sets, it can be represented as:



Set Representation of Many-to-One

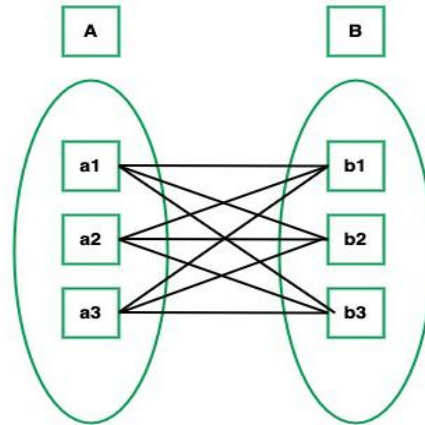
In this case, each student is taking only 1 course but 1 course has been taken by many students.

4. Many-to-Many: When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many. The total number of tables that can be used in this is 3.



Many-to-Many

Using Sets, it can be represented as:

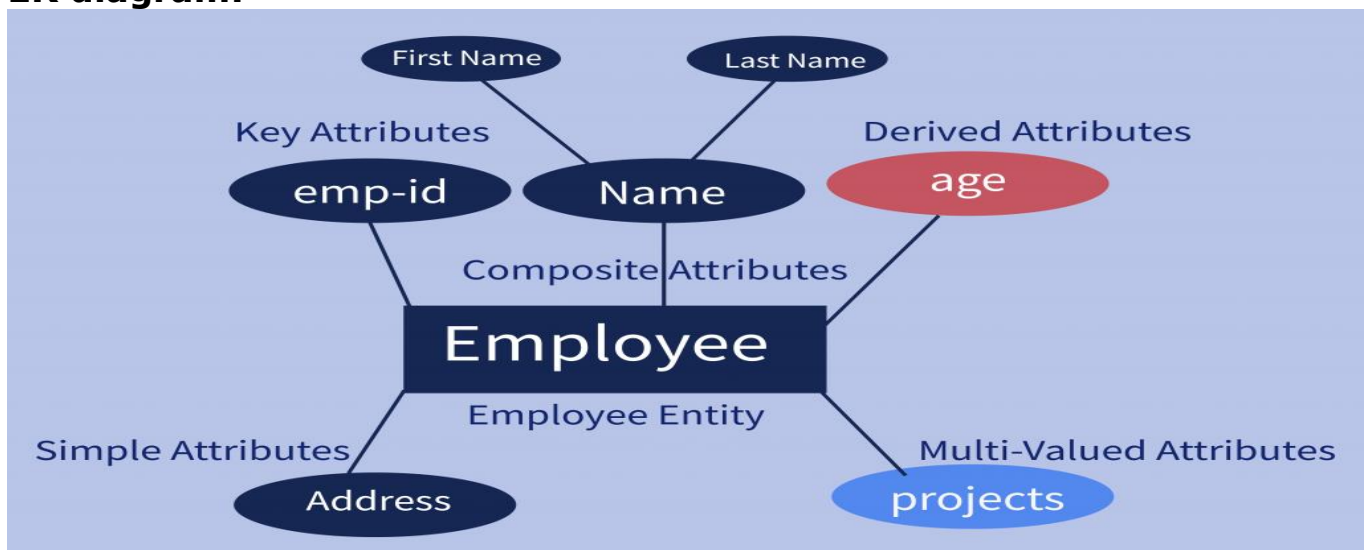


Many-to-Many Set Representation

How to Draw an ER Diagram?

- The very first step is Identifying all the Entities, and place them in a Rectangle, and labeling them accordingly.
- The next step is to identify the relationship between them and place them accordingly using the Diamond, and make sure that, Relationships are not connected to each other.
- Attach attributes to the entities properly.
- Remove redundant entities and relationships.
- Add proper colors to highlight the data present in the database.

Here is the figure is given below represents all the attributes in the ER diagram:



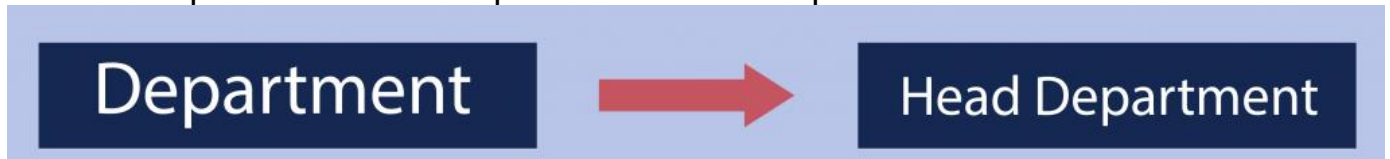
Relationship:

A relationship in a DBMS is primarily the way two or more data sets are linked. Relationships allow the datasets to share and store data in separate tables. They also help link disparate data with each other. A relationship, in the context of databases, is a situation that exists between two relational database tables when one table has a foreign key that references the primary key of the other table. Relationships allow relational databases to split and store data in different tables while linking disparate data items. Relationships are of three types and the next segment talks about the same.

Types of relationships:

- One to One
- One to Many
- Many to Many

One to One – It is used to create a relationship between two tables in which a single row of the first table can only be related to one and only one record of a second table. This relationship tells us that a single record in Table A is related to a single record in Table B. And vice versa. Example – In a university, each department has only one head of the department. And one HOD can take only one department. This shows a one-to-one (1:1) relationship between the department and the person as a head.

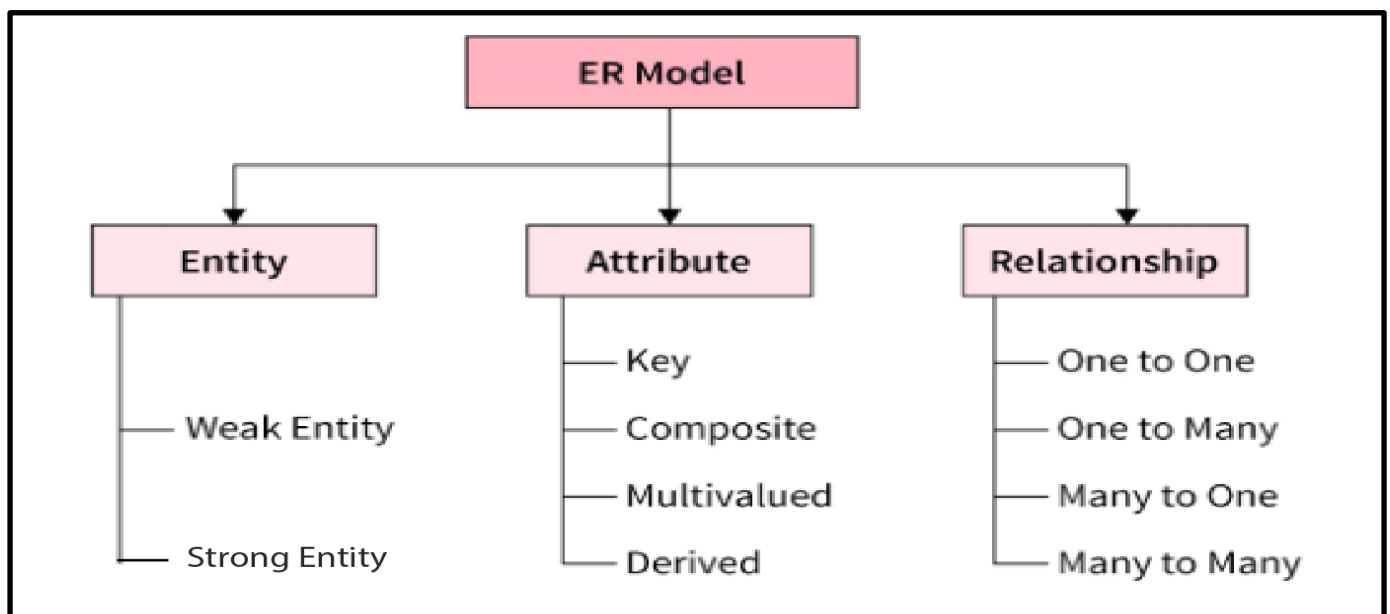
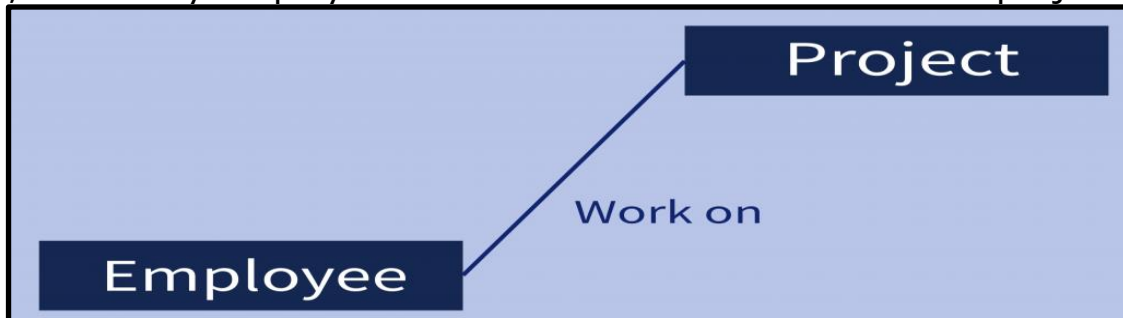


One to Many – It is used to create a relationship between two tables. Any single row of the first table can be related to one or more rows of the second table, but the rows of the second table can only relate to the only row in the first table. It is also known as a many-to-one relationship.

Example: of a 1:M relationship is A department that has many employees, Each employee is assigned to one department.



Many to Many – Many to many relationships that create a relationship between two tables. Each record of the first table can relate to any records (or no records) in the second table. Similarly, each record of the second table can also relate to more than one record of the first table. It also represented an N:N relationship. Example: there are many employees involved in each project, and every employee can be involved in more than one project.



1.18 ER Design Issues

In the previous sections of the data modeling, we learned to design an ER diagram. We also discussed different ways of defining entity sets and relationships among them. We also understood the various designing shapes that represent a relationship, an entity, and its attributes. However, users often mislead the concept of the elements and the design process of the ER diagram. Thus, it leads to a complex structure of the ER diagram and certain issues that does not meet the characteristics of the real-world enterprise model. Here, we will discuss the basic design issues of an ER database schema in the following points:

1) Use of Entity Set vs Attributes

The use of an entity set or attribute depends on the structure of the real-world enterprise that is being modeled and the semantics associated with its attributes. It leads to a mistake when the user uses the primary key of an entity set as an attribute of another entity set. Instead, he should use the relationship to do so. Also, the primary key attributes are implicit in the relationship set, but we designate it in the relationship sets.

2) Use of Entity Set vs. Relationship Sets

It is difficult to examine if an object can be best expressed by an entity set or relationship set. To understand and determine the right use, the user needs to designate a relationship set for describing an action that occurs in-between the entities. If there is a requirement of representing the object as a relationship set, then it's better not to mix it with the entity set.

3) Use of Binary vs n-ary Relationship Sets

Generally, the relationships described in the databases are binary relationships. However, non-binary relationships can be represented by several binary relationships. For example, we can create and represent a ternary relationship 'parent' that may relate to a child, his father, as well as his mother. Such relationships can also be represented by two binary relationships i.e, mother and father, that may relate to their child. Thus, it is possible to represent a non-binary relationship by a set of distinct binary relationships.

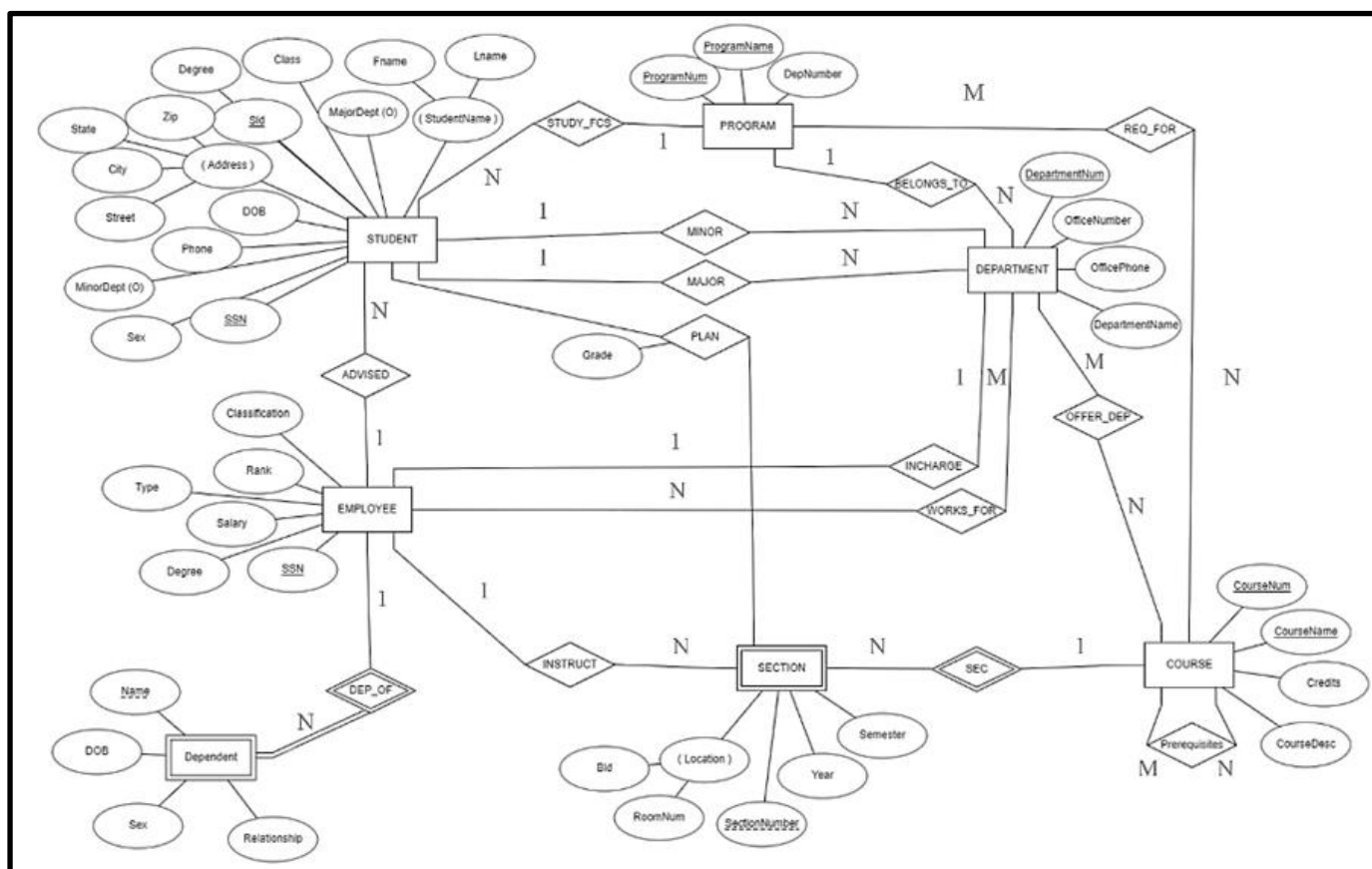
4) Placing Relationship Attributes

The cardinality ratios can become an effective measure in the placement of the relationship attributes. So, it is better to associate the attributes of one-to-one or one-to-many relationship sets with any participating entity sets, instead of any relationship set. The decision of placing the specified attribute as a relationship or entity attribute should possess the characteristics of the real world enterprise that is being modeled.

For example, if there is an entity which can be determined by the combination of participating entity sets, instead of determining it as a separate entity. Such type of attribute must be associated with the many-to-many relationship sets.

Thus, it requires the overall knowledge of each part that is involved in designing and modeling an ER diagram. The basic requirement is to analyze the real-world enterprise and the connectivity of one entity or attribute with another.

1.19 Conceptual Design for University Enterprise :



Conceptual design is the first stage in the database design process. The goal at this stage is to design a database that is independent of database software and physical details. The output of this process is a conceptual data model that describes the main data entities, attributes, relationships, and constraints of a given problem domain. This design is descriptive and narrative in form.

All data elements required by the database transactions must be defined in the model, and all data elements defined in the model must be used by at least one database transaction. The conceptual design has four steps, which are as follows.

1. Data analysis and requirements
2. Entity relationship modeling and normalization
3. Data model verification
4. Distributed database design

Data Analysis and Requirements:

The first step in conceptual design is to discover the characteristics of the data elements. Appropriate data element characteristics are those that can be

transformed into appropriate information. Therefore, the designer's efforts are focused on:

- Information needs
- Information users
- Information sources
- Information constitution
- Directly observing the current system
- Interfacing with the systems design group
- Developing and gathering end-user data views

2. Entity Relationship Modeling and Normalization:

Before creating the ER model, the designer must communicate and enforce appropriate standards to be used in the documentation of the design. The process of defining business rules and developing the conceptual model using ER diagrams can be described using the following steps.

1. Identify, analyze, and refine the business rules.
2. Identify the main entities, using the results of Step 1.
3. Define the relationships among the entities, using the results of Steps 1 and 2.
4. Define the attributes, primary keys, and foreign keys for each of the entities.
5. Normalize the entities. (Remember that entities are implemented as tables in an RDBMS.)
6. Complete the initial ER diagram.
7. Validate the ER model against the end users' information and processing requirements.
8. Modify the ER model, using the results of Step 7.

3. Data Model Verification:

The data model verification step is one of the last steps in the conceptual design stage, and it is also one of the most critical ones. In this step, the ER model must be verified against the proposed system processes in order to corroborate that the intended processes can be supported by the database model. Verification requires that the model be run through a series of tests against:

- End-user data views.
- All required transactions: SELECT, INSERT, UPDATE, and DELETE operations.
- Access rights and security.
- Business-imposed data requirements and constraints.

4. Distributed Database Design:

Although not a requirement for most databases, sometimes a database may need to be distributed among multiple geographically disperse locations. Processes that access the database may also vary from one location to another. For example, a retail process and a warehouse storage process are likely to be found in different physical locations. If the database data and processes are to be distributed across the system, portions of a database, known as database fragments, may reside in several physical locations.

1.20 Relational Model in DBMS

Relational model can be represented as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute.

Domain: It contains a set of atomic values that an attribute can take.

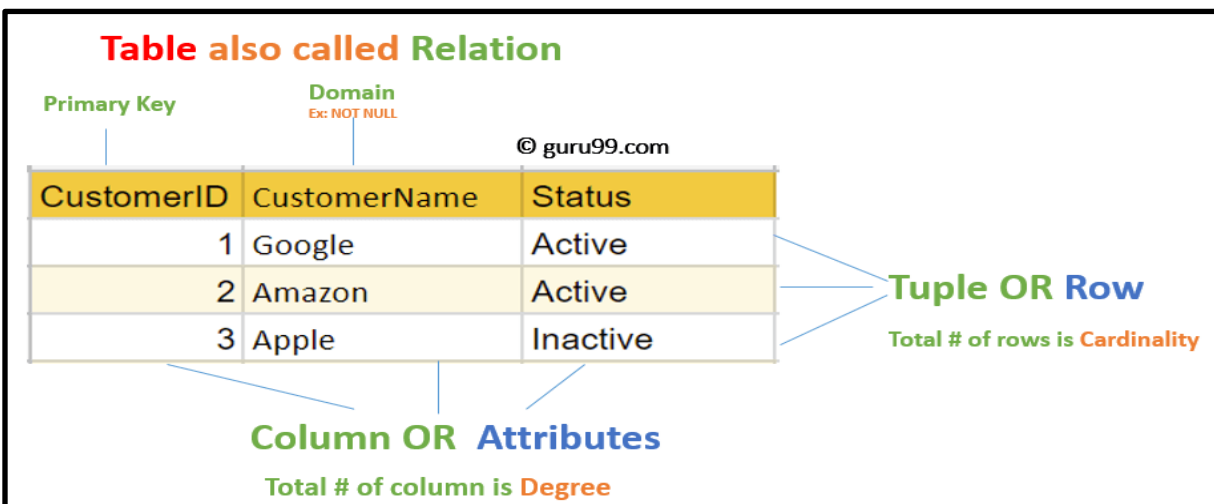
Attribute: It contains the name of a column in a particular table. Each attribute A_i must have a domain, $dom(A_i)$

Relational instance: In the relational database system, the relational instance is represented by a finite set of tuples. Relation instances do not have duplicate tuples.

Relational schema: A relational schema contains the name of the relation and name of all columns or attributes.

Relational key: In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

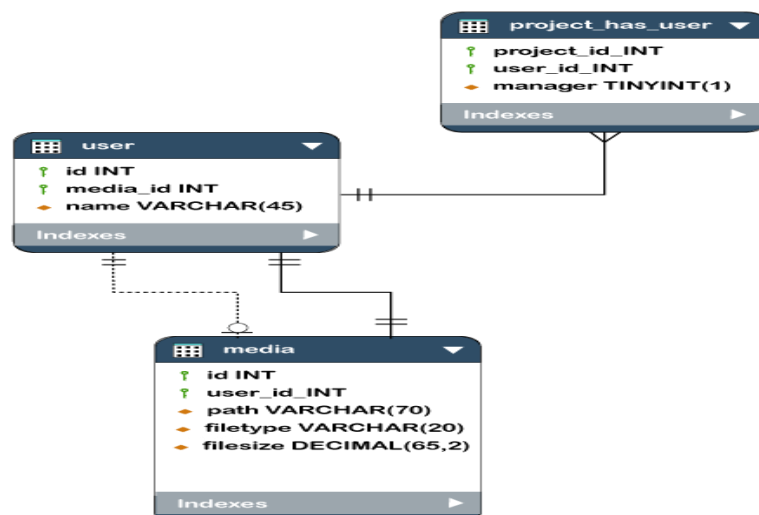
1.21 Structure



1.22 Database Schemas

Database Schema:

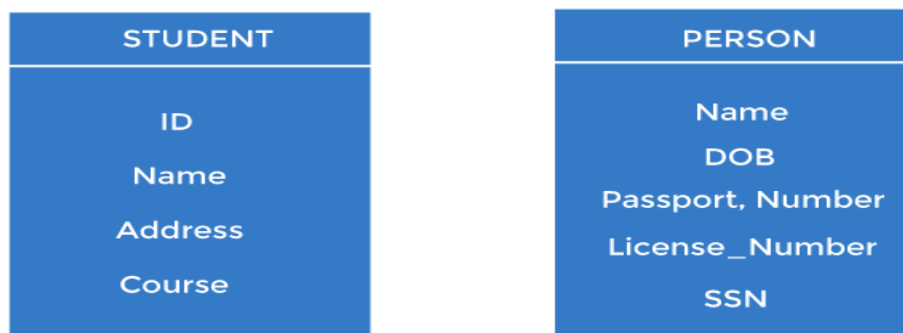
- A database schema is the logical representation of a database, which shows how the data is stored logically in the entire database. It contains a list of attributes and instructions that informs the database engine how the data is organized and how the elements are related to each other.
- A database schema contains schema objects that may include **tables, fields, packages, views, relationships, primary key, foreign key,**
- In actuality, the data is physically stored in files that may be in unstructured form, but to retrieve it and use it, we need to put it in a structured form. To do this, a database schema is used. It provides knowledge about how the data is organized in a database and how it is associated with other data.
- **The schema does not physically contain the data itself; instead, it gives information about the shape of data and how it can be related to other tables or models.**
- A database schema object includes the following:
 - Consistent formatting for all data entries.
 - Database objects and unique keys for all data entries.
 - Tables with multiple columns and each column contain its name and data type.
- The complexity & the size of the schema vary as per the size of the project. It helps developers to easily manage and structure the database before coding it.
- The given diagram is an example of a database schema. It contains three tables, their data types. This also represents the relationships between the tables and primary keys as well as foreign keys.



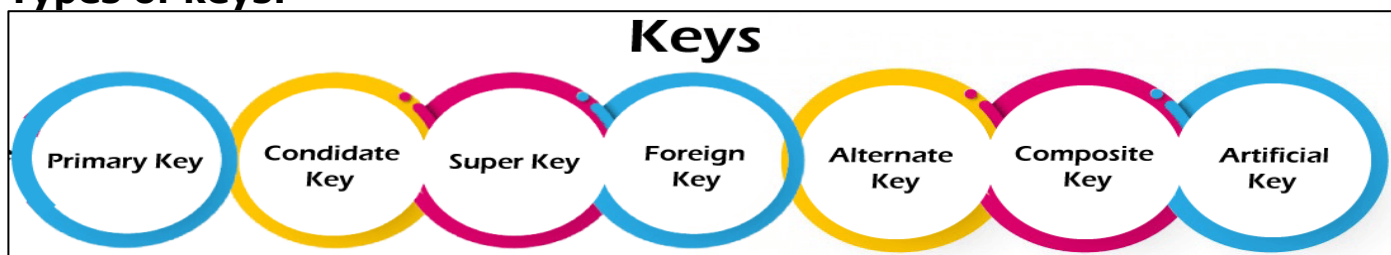
1.23 Keys:

- 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.

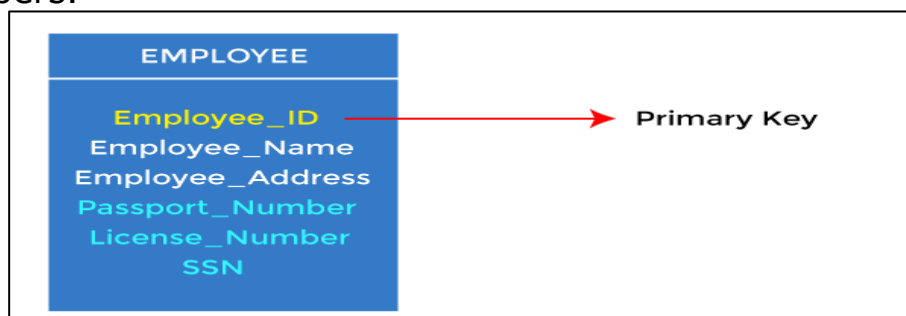


Types of keys:



1. Primary key

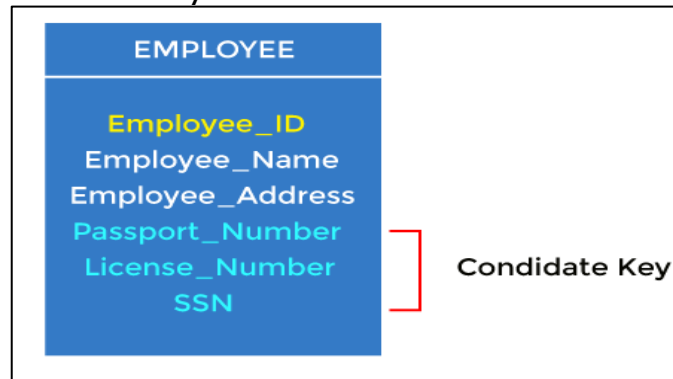
- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
- In the EMPLOYEE table, ID can be the primary key since it is unique for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

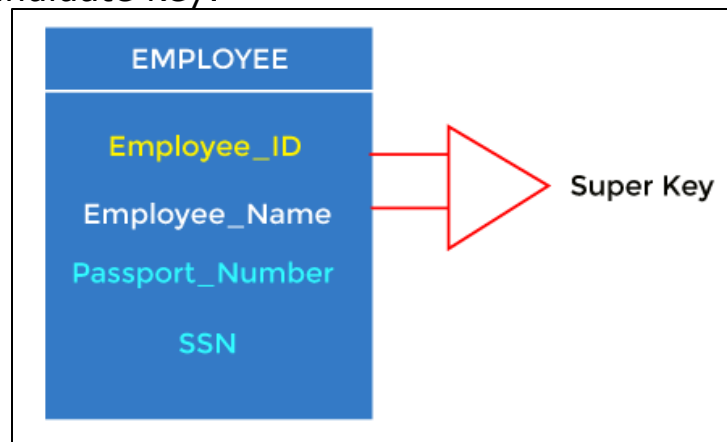
- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

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.

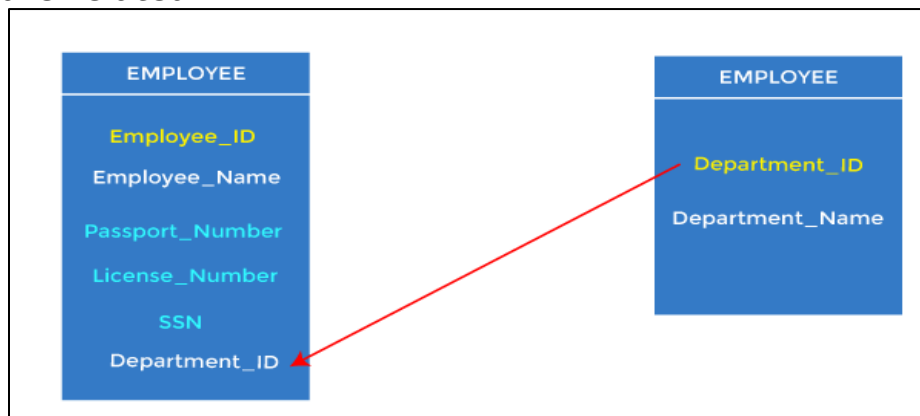


For example: In the above EMPLOYEE table, for (EMPLOYEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLOYEE_ID can't be the same. Hence, this combination can also be a key. The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

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.

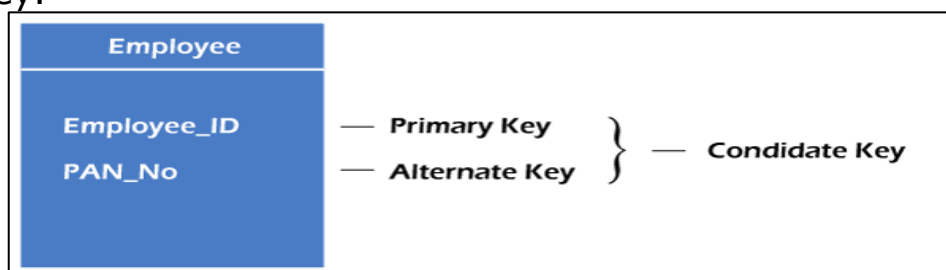
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



5. Alternate key

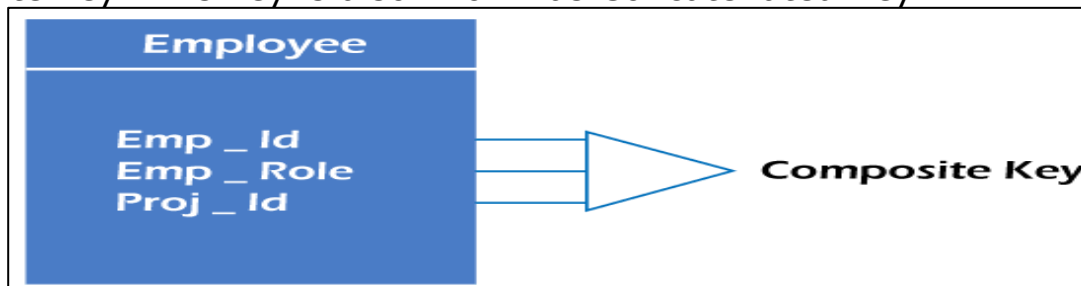
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it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

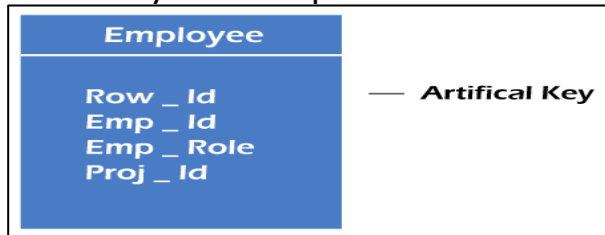


For example, in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite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

7. Artificial key

The keys created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are usually numbered in a serial order.

For example, the primary key, which is composed of Emp_ID, Emp_role, and Proj_ID, is large in employee relations. So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.



Primary key	Candidate key	Super key	Foreign key
<p>A column or set of columns in a database table that serves as a unique identifier.</p> <p>Examples include customer or employee numbers, email addresses and telephone numbers.</p>	<p>A column or set of columns in a table that can potentially be used as a primary key. To qualify, it must be able to function as a unique identifier to sort all of the table's data records.</p>	<p>A set of data attributes from different columns in a table that can be used as an identifier. For example, columns containing employee numbers and email addresses could be combined.</p>	<p>A column in one database table that is linked to the primary key in another table. Foreign keys are used to make data available in different tables without having to create redundant data sets.</p>
