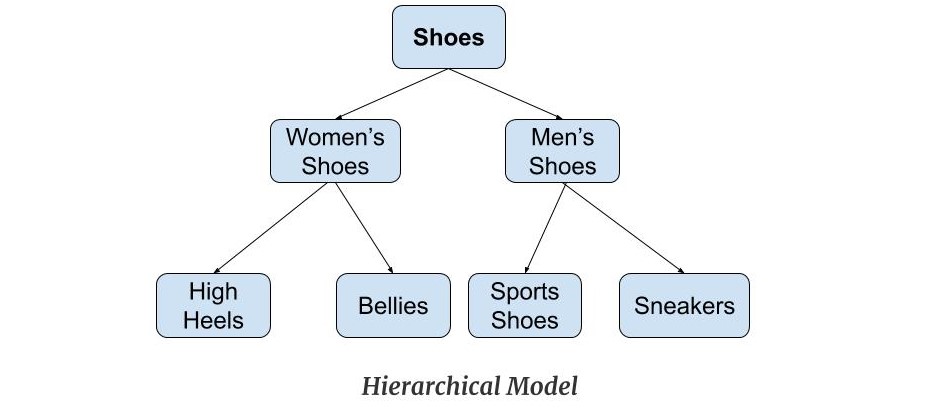
**Data Model**

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 but the Relational model is the most widely used model. Apart from the Relational model, there are many other types of data models about which we will study in details in this blog. 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

**Hierarchical Model**

Hierarchical Model was the first DBMS model. This model organises 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 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:***



*Features of a Hierarchical Model*

1. ***One-to-many relationship:*** The data here is organised in a tree-like structure where the one-to-many relationship is between the datatypes. 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 men's shoes node.
2. ***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.
3. ***Deletion Problem:*** If a parent node is deleted then the child node is automatically deleted.
4. ***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*

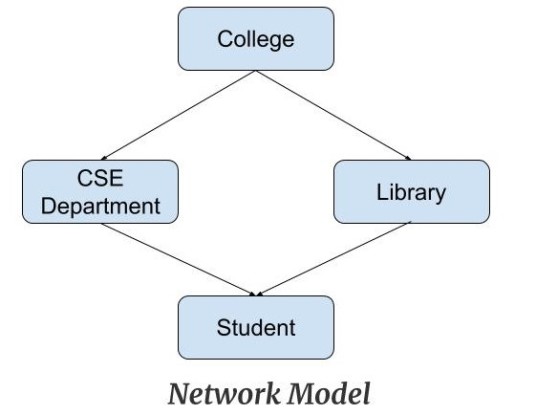
* It is very simple and fast to traverse through a tree-like structure.
* Any change in the parent node is automatically reflected in the child node so, the integrity of data is maintained.

*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 node then that can't be represented using this model.
* If a parent node is deleted then the child node is automatically deleted.

**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.

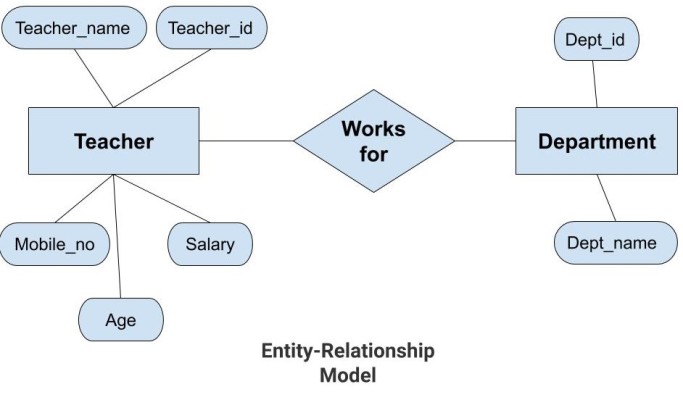
*Disadvantages of Network Model*

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

**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.



*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.

*Disadvatages of ER Model*

* ***No industry standard for notation:*** There is no 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 so there are chances that some details of information might be hidden.

**Relational Model**

Relational Model is the most widely used model. In this model, the data is maintained in the form of a two-dimensional table. All the information is stored in the form of row and columns. The basic structure of a relational model is tables. So, the tables are also called *relations* in the relational model. Example



*Features of Relational Model*

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

*Advnatages of Relational Model*

* ***Simple:*** This model is more simple 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 to DBMS to access the data we can say that structural independence has been achieved.

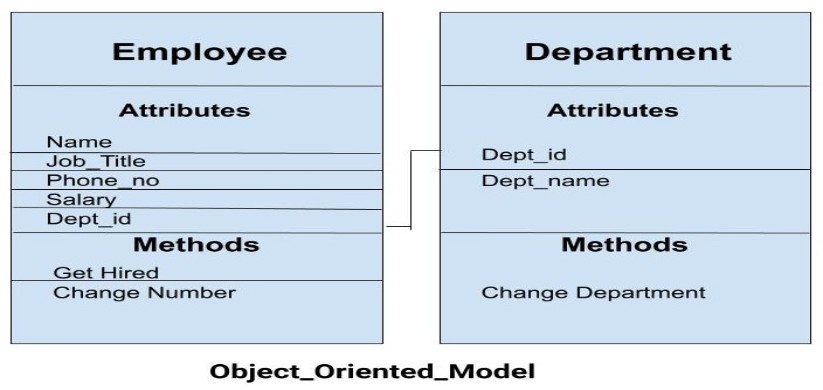
*Disadvantages of Relatinal Model*

* ***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 organisation.

**Object-Oriented Data Model**

The real-world problems are more closely represented through the object-oriented data model. In this model, both the data and relationship are present in a single structure known as an object. We can store audio, video, images, etc in the database which was not possible in the relational model(although you can store audio and video in relational database, it is adviced not to store in the relational database). In this model, two are more objects are connected through links. We use this link to relate one object to other objects. Example



In the above example, we have two objects Employee and Department. All the data and relationships of each object are contained as a single unit. The attributes like Name, Job\_title of the employee and the methods which will be performed by that object are stored as a single object. The two objects are connected through a common attribute i.e the Department\_id and the communication between these two will be done with the help of this common id.

**Object-Relational Model**

As the name suggests it is a combination of both the relational model and the object-oriented model. This model was built to fill the gap between object-oriented model and the relational model. We can have many advanced features like we can make complex data types according to our requirements using the existing data types. The problem with this model is that this can get complex and difficult to handle. So, proper understanding of this model is required.

**Flat Data Model**

It is a simple model in which the database is represented as a table consisting of rows and columns. To access any data, the computer has to read the entire table. This makes the modes slow and inefficient.

**Semi-Structured Model**

Semi-structured model is an evolved form of the relational model. We cannot differentiate between data and schema in this model. ***Example:*** Web-Based data sources which we can't differentiate between the schema and data of the website. In this model, some entities may have missing attributes while others may have an extra attribute. This model gives flexibility in storing the data. It also gives flexibility to the attributes. ***Example:*** If we are storing any value in any attribute then that value can be either atomic value or a collection of values.

**Associative Data Model**

Associative Data Model is a model in which the data is divided into two parts. Everything which has independent existence is called as an *entity* and the relationship among these entities are called *association*. The data divided into two parts are called items and links.

* ***Item***: Items contain the name and the identifier(some numeric value).
* ***Links:*** Links contain the identifier, source, verb and subject.

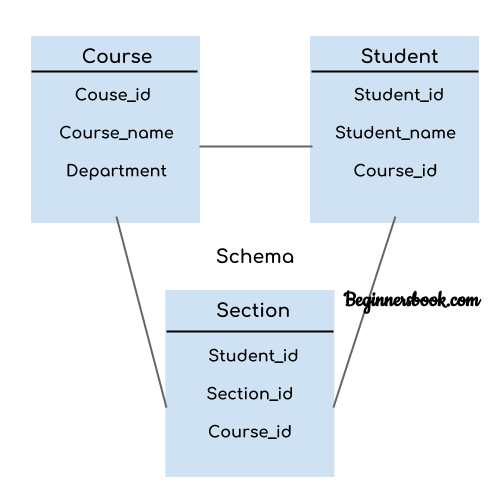
#### Context Data Model

Context Data Model is a collection of several models. This consists of models like network model, relational models etc. Using this model we can do various types of tasks which are not possible using any model alone.

## \*\*\*\*DBMS Schema:-

Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

or example: In the following diagram, we have a schema that shows the relationship between three tables: Course, Student and Section. The diagram only shows the design of the database, it doesn’t show the data present in those tables. Schema is only a structural view(design) of a database as shown in the diagram below.



The design of a database at physical level is called **physical schema**, how the data stored in blocks of storage is described at this level.

Design of database at logical level is called **logical schema**, programmers and database administrators work at this level, at this level data can be described as certain types of data records gets stored in data structures, however the internal details such as implementation of data structure is hidden at this level (available at physical level).

Design of database at view level is called **view schema**. This generally describes end user interaction with database systems.

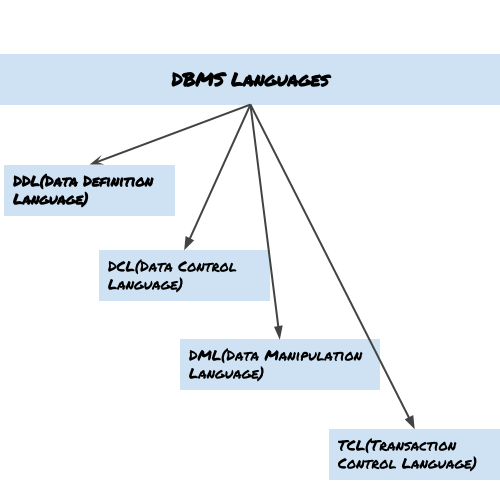
## DBMS Instance

**Definition of instance**: The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

# DBMS languages

Database languages are used to read, update and store data in a database. There are several such languages that can be used for this purpose; one of them is SQL (Structured Query Language).

## Types of DBMS languages:



### Data Definition Language (DDL)

DDL is used for specifying the database schema. It is used for creating tables, schema, indexes, constraints etc. in database. Lets see the operations that we can perform on database using DDL:

* To create the database instance – [CREATE](https://beginnersbook.com/2014/05/sql-create-database-statement/)
* To alter the structure of database – **ALTER**
* To drop database instances – [DROP](https://beginnersbook.com/2014/05/sql-drop-database-statement/)
* To delete tables in a database instance – **TRUNCATE**
* To rename database instances – **RENAME**
* To drop objects from database such as tables – **DROP**
* To Comment – **Comment**

All of these commands either defines or update the database schema that’s why they come under Data Definition language.

### Data Manipulation Language (DML)

DML is used for accessing and manipulating data in a database. The following operations on database comes under DML:

* To read records from table(s) – [SELECT](https://beginnersbook.com/2014/05/sql-select-query/)
* To insert record(s) into the table(s) – **INSERT**
* Update the data in table(s) – [UPDATE](https://beginnersbook.com/2014/05/update-query-in-sql/)
* Delete all the records from the table – [DELETE](https://beginnersbook.com/2014/05/delete-query-in-sql/)

### Data Control language (DCL)

DCL is used for granting and revoking user access on a database –

* To grant access to user – GRANT
* To revoke access from user – REVOKE

**In practical data definition language, data manipulation language and data control languages are not separate language, rather they are the parts of a single database language such as SQL.**

### Transaction Control Language(TCL)

The changes in the database that we made using DML commands are either performed or rollbacked using TCL.

* To persist the changes made by DML commands in database – COMMIT
* To rollback the changes made to the database – ROLLBACK

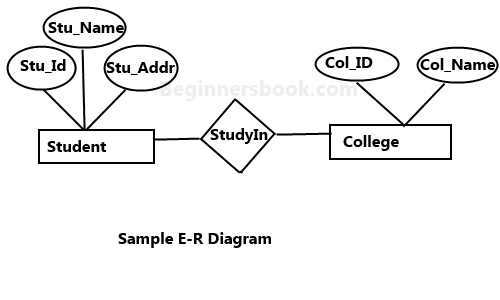
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

## What is an Entity Relationship Diagram (ER Diagram)?

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Lets have a look at a simple ER diagram to understand this concept.

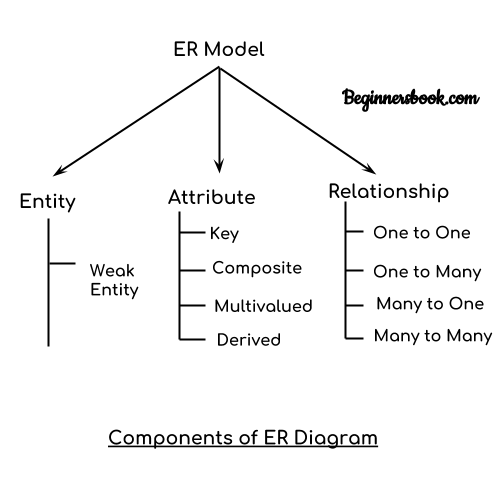
## A simple ER Diagram:

In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu\_Id, Stu\_Name & Stu\_Addr and College entity has attributes such as Col\_ID & Col\_Name.



**Rectangle**: Represents Entity sets.  
**Ellipses**: Attributes  
**Diamonds**: Relationship Set  
**Lines**: They link attributes to Entity Sets and Entity sets to Relationship Set  
**Double Ellipses:** Multivalued Attributes  
**Dashed Ellipses**: Derived Attributes  
**Double Rectangles**: Weak Entity Sets  
**Double Lines**: Total participation of an entity in a relationship set

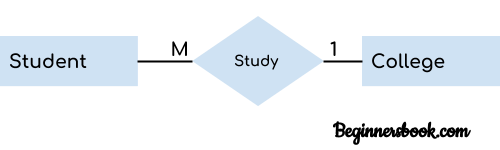
## Components of a ER Diagram



ER diagram has three main components:  
1. Entity  
2. Attribute  
3. Relationship

### Entity

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



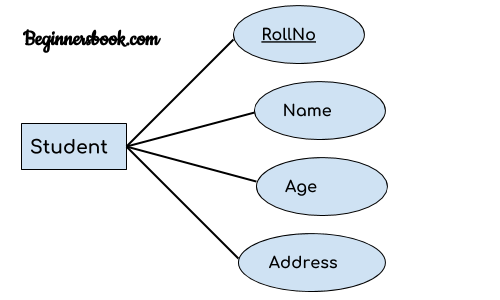
### Attribute

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

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

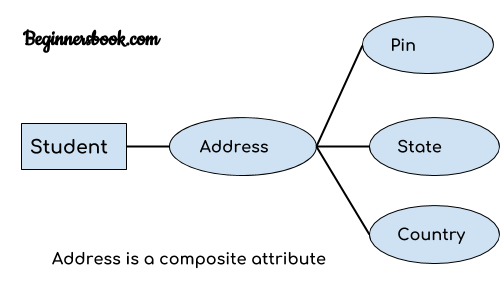
#### 1. Key attribute:

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



#### Composite attribute:

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

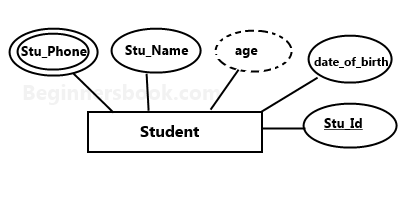


#### Multivalued attribute:

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

#### Derived attribute:

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

**E-R diagram with multivalued and derived attributes**:  


### Relationship

A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:  
1. One to One  
2. One to Many  
3. Many to One  
4. Many to Many

#### 1. One to One Relationship

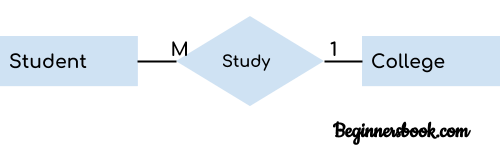
When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship.

#### 2. One to Many Relationship

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

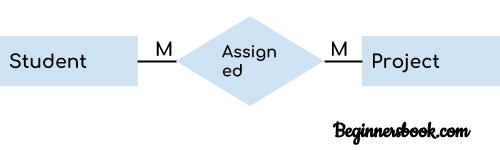
#### Many to One Relationship

When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship. For example – many students can study in a single college but a student cannot study in many colleges at the same time.



#### Many to Many Relationship

When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship. For example, a can be assigned to many projects and a project can be assigned to many students.



Relational Data Models :-

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

## Concepts

**Tables** − In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.

**Tuple** − A single row of a table, which contains a single record for that relation is called a tuple.

**Relation instance** − A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

**Relation schema** − A relation schema describes the relation name (table name), attributes, and their names.

**Relation key** − Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.

**Attribute domain** − Every attribute has some pre-defined value scope, known as attribute domain.

## Constraints

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called **Relational Integrity Constraints**. There are three main integrity constraints −

* Key constraints
* Domain constraints
* Referential integrity constraints

### Key Constraints

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called **key** for that relation. If there are more than one such minimal subsets, these are called ***candidate keys***.

Key constraints force that −

* in a relation with a key attribute, no two tuples can have identical values for key attributes.
* a key attribute can not have NULL values.

Key constraints are also referred to as Entity Constraints.

### Domain Constraints

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

### Referential integrity Constraints

Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

## Advantages of using Relational model

* **Simplicity**: A relational data model is simpler than the hierarchical and network model.
* **Structural Independence**: The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
* **Easy to use**: The relational model is easy as tables consisting of rows and columns is quite natural and simple to understand
* **Query capability**: It makes possible for a high-level query language like SQL to avoid complex database navigation.
* **Data independence**: The structure of a database can be changed without having to change any application.
* **Scalable**: Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

## Disadvantages of using Relational model

* Few relational databases have limits on field lengths which can't be exceeded.
* Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
* Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.

Transaction Processing Concepts:-

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Let’s take an example of a simple transaction. Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

**A’s Account**

Open\_Account(A)

Old\_Balance = A.balance

New\_Balance = Old\_Balance - 500

A.balance = New\_Balance

Close\_Account(A)

**B’s Account**

Open\_Account(B)

Old\_Balance = B.balance

New\_Balance = Old\_Balance + 500

B.balance = New\_Balance

Close\_Account(B)

## ACID Properties

A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain **A**tomicity, **C**onsistency, **I**solation, and **D**urability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

* **Atomicity** − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
* **Consistency** − The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
* **Durability** − The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.
* **Isolation** − In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

## Serializability

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.

* **Schedule** − A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.
* **Serial Schedule** − It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

In a multi-transaction environment, serial schedules are considered as a benchmark. The execution sequence of an instruction in a transaction cannot be changed, but two transactions can have their instructions executed in a random fashion. This execution does no harm if two transactions are mutually independent and working on different segments of data; but in case these two transactions are working on the same data, then the results may vary. This ever-varying result may bring the database to an inconsistent state.

To resolve this problem, we allow parallel execution of a transaction schedule, if its transactions are either serializable or have some equivalence relation among them.

## Equivalence Schedules

An equivalence schedule can be of the following types −

### Result Equivalence

If two schedules produce the same result after execution, they are said to be result equivalent. They may yield the same result for some value and different results for another set of values. That's why this equivalence is not generally considered significant.

### View Equivalence

Two schedules would be view equivalence if the transactions in both the schedules perform similar actions in a similar manner.

For example −

* If T reads the initial data in S1, then it also reads the initial data in S2.
* If T reads the value written by J in S1, then it also reads the value written by J in S2.
* If T performs the final write on the data value in S1, then it also performs the final write on the data value in S2.

### Conflict Equivalence

Two schedules would be conflicting if they have the following properties −

* Both belong to separate transactions.
* Both accesses the same data item.
* At least one of them is "write" operation.

Two schedules having multiple transactions with conflicting operations are said to be conflict equivalent if and only if −

* Both the schedules contain the same set of Transactions.
* The order of conflicting pairs of operation is maintained in both the schedules.

**Note** − View equivalent schedules are view serializable and conflict equivalent schedules are conflict serializable. All conflict serializable schedules are view serializable too.

## States of Transactions

A transaction in a database can be in one of the following states −

* **Active** − In this state, the transaction is being executed. This is the initial state of every transaction.
* **Partially Committed** − When a transaction executes its final operation, it is said to be in a partially committed state.
* **Failed** − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
* **Aborted** − If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts −
  + Re-start the transaction
  + Kill the transaction
* **Committed** − If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.