Database Management System (DBMS)

1) **Data:**
   Data is the raw material that can be processed for any computing machine.
   For example − Employee name, Product name, Name of the student, Marks of the student, Mobile number, Image etc.
2) **Information:**
   Information is the data that has been converted into more useful or intelligent form.
   For example: Report card sheet.
3) **Field:**
   A database field is a set of data values, of the same data type, in a table. It is also referred to as a column or an attribute**.**
4) **Record:**
   The term record in DBMS refers to a collection of items or data organized within a table within a set of fields related to a particular topic or theme.
5) **File:**
   A file is a collection of records, each of which contains information about one person or thing.
6) **Database:**

A database can be defined as a collection of consistent, **meaningful** data. The phrase collection of logical data needs to have a point of reference to be understood.

7) **What Is DBMS?**

The system that would **help in managing data** in such a database, called Data Base Management System. DBMS is a system that allows inserting, updating, deleting and processing of data.

The three-schema architecture for a Database Management System(DBMS)

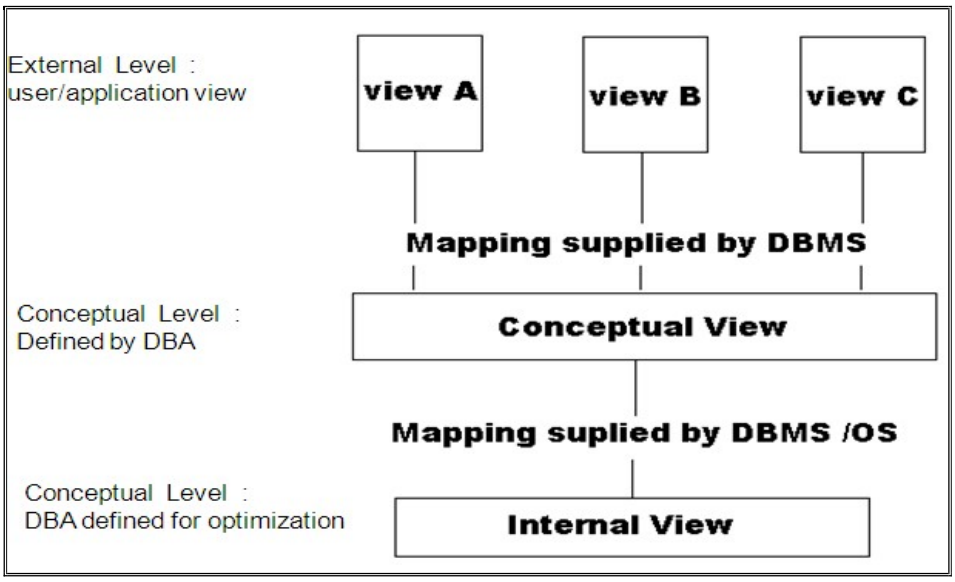
The Architecture, is divided into three levels :

1. External level
2. Conceptual level And
3. Internal level

The view at each of the level is described by a **SCHEMA**.

A **Schema** is an outline or a plan that describes the records and relationships existing in the view.

1. INTERNAL VIEW
2. CONCEPTUAL VIEW
3. EXTERNAL VIEW



1. **INTERNAL VIEW :**

   In this view at the lowest level of abstraction, ***closet to the physical storage methods used***. It indicates ***how data will be stored*** and describes the data structures and access method to be used by the database. The internal view is expressed by the internal view schema, which contains the definition of the stored record, the method of representing the data fields, and the access of it.

2. CONCEPTUAL OR GLOBAL VIEW:

At this level, all the database entities and the relationships among them are included. One conceptual view represents the ***entire DATABASE***. And this conceptual view is defined by conceptual schema. It describes all the records and relationships includes in the conceptual view. There is only one conceptual schema per database. The schema also contains the methods of deriving the object in the conceptual view from the object in the internal view.

3. EXTERNAL OR USER VIEW:

The external or user view is at the highest level of database abstraction (concept) ***where only those portions of the database of concern to a user or application program included***. Any number of user views may exist for a given global or conceptual view. Each external view is described by means of a scheme called an external schema. The external schema consists of the definition of the logical records and the relationships in the external view. The external view also contains the method of deriving the objects in the external view from the objects in the conceptual view.

INTRODUCTION TO DATA MODELS

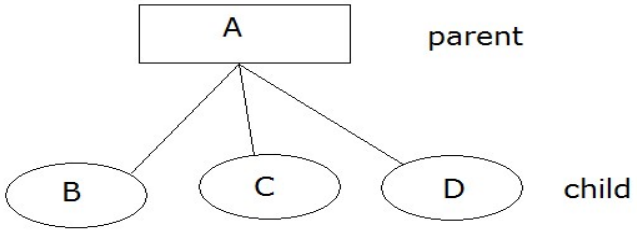
- A data model is a picture or description which depicts **how data is to be arranged** to serve a specific purpose.
- The data model depicts what that data items are required, and how that data must look.

- **DATA MODELS can be classify….**

1. Hierarchical
2. Network
3. Relational

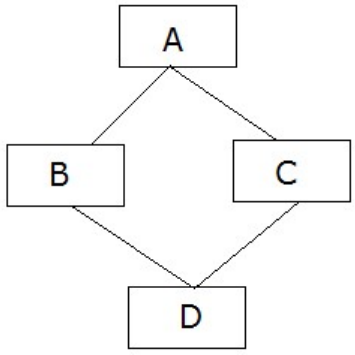
1. Hierarchical model

- The hierarchical data model organizes data in a **tree structure**.
- There is a hierarchy of **parent and child data segments**.
- This structure implies that a record can have repeating information, generally in the child data segments.
- Data in a series of records, which have a set of field values attached to it. It collects all the instances of a specific record together as a record type.
- These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows.
- To create links between these record types, the hierarchical model uses Parent Child Relationships.
- These are a **1:N** mapping between record types.
- i.e. it represent a **one-to-many relationship** between two entities where the two are respectively parent and child
- **For example,** an organization might store information about an employee, such as name, employee number, department, salary.
- The organization might also store information about an employee's children, such as name and date of birth. The employee and children data forms a hierarchy, where the employee data represents the parent segment and the children data represents the child segment.
- If an employee has three children, then there would be three child segments associated with one employee segment.
- In a hierarchical database the parent-child relationship is one to many. This restricts a child segment to having only one parent segment.
- Hierarchical DBMSs were popular from the late 1960s, with the introduction of IBM's Information Management System (IMS) DBMS, through the 1970s.



2. Network Model

- The popularity of the network data model coincided with the popularity of the hierarchical data model.
- Some data were more naturally modeled with **more than one parent per child**. So, the network model permitted the modeling of many-to-many relationships in data.
- In 1971, the Database Task Group of the Conference on Data Systems Languages **(DBTG / CODASYL)** formally defined the network model. The basic data modeling construct in the network model is the set construct.
- A set consists of an owner record type, a set name, and a member record type.
- A member record type can have that role in more than one set, hence the multi parent concept is supported.
- An owner record type can also be a member or owner in another set.
- The data model is a simple network, and link and intersection record types (called junction records by IDMS) may exist, as well as sets between them.
- Thus, the complete network of relationships is represented by several pair wise sets; in each set some (one) record type is owner (at the tail of the network arrow) and one or more record types are members (at the head of the relationship arrow).
- Usually, a set defines a **1:M relationship**, although 1:1 is permitted. The CODASYL network model is based on mathematical set theory.



In above diagram we can clearly seen that entity d having multiple parents which will display one child having multiple parents.

Relational Model

- The relational data management model eliminated all **patent-child relationship** and instead represented all data in the database as simple row/column tables of data values.
- A relation is similar to a table with rows/columns of data values. The **rows of a table are referred to as Tuples** and **the columns are referred to as Attributes.** Several tupels of equal length placed one below the other create a table.
- Each table is an independent entity and there is no physical relationship between tables.
- Most data management systems based on the relational model have a built-in support for query languages like ANSI SQL.
- Relational model of data management is based on **set theory**. Built-in query language is designed in the RDBMS, so that it can manipulate sets of data (one or more tuples).

Properties of Relational Tables:

- Values Are Atomic(tiny)
- Each Row is Unique
- Column Values Are of the Same Kind
- The Sequence of Columns is Insignificant
- The Sequence of Rows is Insignificant
- Each Column Has a Unique Name

- **CONCEPT: -** A relational database matches data by using common characteristics found within the data set.

  The resulting groups of data are organized and are much easier for many people to understand.

  For example, a data set containing all the real-estate transactions in a town can be grouped by the year each transaction occurred, the sale price, a buyer's last name and so on.

  Such a grouping uses the relational model (a technical term for this is schema).

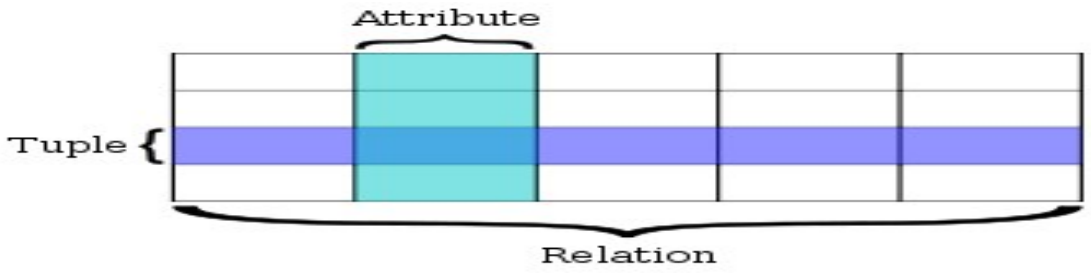
  Hence, such a database is called a "relational database."

  The software used to do this grouping is called a relational database management system (RDBMS). The term "relational database" often refers to this type of software

- **TERMINOLOGY :-** The term relational database was originally defined by and is attributed to Edgar Codd at IBM Almaden Research Center in 1970.

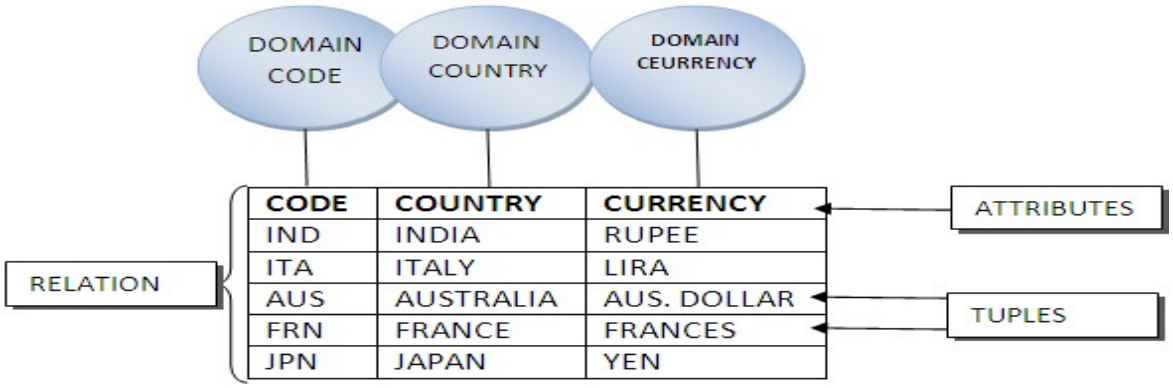
Relational database theory uses a set of mathematical terms, which are roughly equivalent to SQL database terminology.

The table below summarizes some of the most important relational database terms and their SQL database equivalents.



A relation is defined as a set of tuples that have the same attributes.

A tuples usually represents an object and information about that object. Objects are typically physical objects or concepts.



- **DOMAIN :** A Domain is a pool of values from where one or more attributes (column/field) can draw their actual values.

  Ex. For above table domain name for this field is COUNTRY.if we takes values from country field.

- **TUPLE :** According to the relational model , every relation or table is made of many Tuples.

  They are also called row/records.

- **ATTRIBUTES :** The term Attributes refers to characteristics.

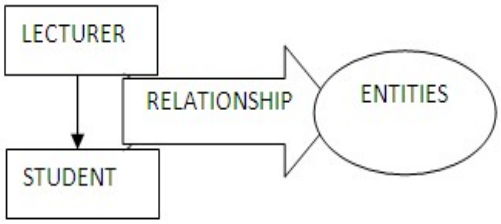
  The characteristics of the tuple are reflected by its attributes or fields.

This simply means that the column contains will be defined by the attributes of that column.

RELATIONSHIP AND RELATIONSHIP TYPES:

- A relationship can be defined as an association among ENTITIES.
- An association of several entities in a r-y model is called relationship.
- The relationship represents the fact that the LECTURER teaches several STUDENTS And STUDENT is taught by several LECTURERS.

ENTITIES AND RELATIONSHIP



➢ **Three Types of Relationship exists among entities….**

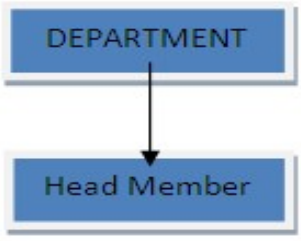
1. **One-to-One (1:1)**
2. **One-to-Many (1:M)**
3. **Many-to-Many (M:M)**

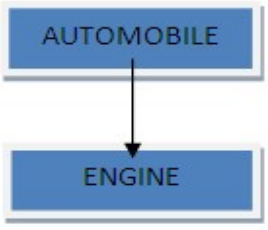
1. **One-to-One (1:1)**

- A One-to-One relationship is an association between two entities.

  For Ex…

- In University each Department have one head of department.
- More Over, One member cannot be a head of more than one department.

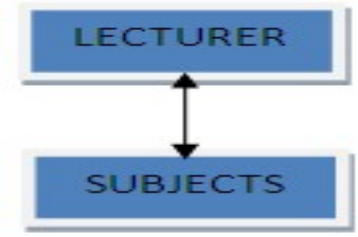


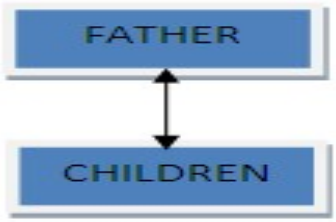


2. **One-to-Many**

- A One-to-Many relationship exists when one entity is related to more than one entity.

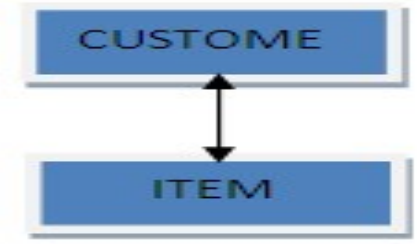
- A father may have many children but, a child has only one father.
- More than one subject is taught by one lecturer only.

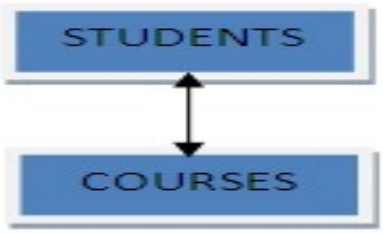




3. Many-to-Many Relationship

- A many-to-many relationship describes entities that may have many relations in both the directions.
- One customer may buy many items and one item may be bought by many customer.
- Another Ex..A students can take many courses in university, and many students can register for given course.





Key Constraints

A Key Constraint is a statement that a certain minimal subset of the fields of a relation is a **unique identifier** for a **tuple**.

- **Super Key**: It is a set of one or more attributes which put together enable us **to identify uniquely an entity in the entity set**.
- **Candidate Key:** A set of fields that **uniquely identifies a tuple** according to a key constraint is called a candidate key. A superkey for which no subset is called a candidate key. (A superkey that is minimal is a candidate key) **Eg. In students relation, sid is candidate key.**
- **Primary Key:** It is a candidate key (there may be more than one) chosen by the database designer to identify entities in an entity set.

Dr. E.F.Codd's 12 Rules For RDBMS:- (RIL RCD VIP LID Non)

Dr. E.F. Codd is an IBM researcher who first developed the relational data model in 1970. In 1985 Dr. E.F. Codd published a list of 12 Rules that define an ideal relational database and has provided guidelines for the design of all relational database systems.

Rule 0: Relational Database Management System:-

A relational database management system uses only its **relational capabilities** to manage the information **stored** in the database.

Rule 1: The Information Rule:-

All information stored in a relational database is represented only by data item values. All data should be presented in **table form**.

Rule 2: Logical Access Rule (Guaranteed access Rules):-

All data should be **accessible** without ambiguity. This can be accomplished through combination of the table name, primary key and column name.

Rule 3: Representation of NULL values:-

A field should be allowed to **remain empty**. This involves the support of null value, which is distinct from an empty string or a number with a value of zero. This can't apply to primary key.

Rule 4: Online Catalog Facilities Based on Relational Model:-

A relational database must provide **access to its structure** through the same tool that are used to access the data.

Rule 5: Data Sublanguage Rule:-

The database must support at least one clearly defined language that includes functionality for data definition, data manipulation, data integrity and database transaction control. All commercial relational database use forms of standard **SQL**.

Rule 6: View Updating Rule:-

Data can be presented in different logical combination called views. Each view should support the same full range of **data manipulation** that has direct access to a table available.

Rule 7: High Level Insert, Update, Delete:-

Data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. The rule states that insert, update and delete operations should be **supported for any retrievable set rather than just for a single row in single table**.

Rule 8: Physical Data Independence:-

The user is isolated from the **physical method of storing and retrieving information** from the database. Changes can be made to the underlying architecture (hardware, disk storage) without affecting how the user accesses it.

Rule 9: Logical Data Independence:-

How data is viewed should not be changed when the logical structure of the database changes. This rule is particularly difficult to satisfy.

Rule 10: Integrity Constraints:-

The database language (like SQL) should support constraints on user input that maintain database integrity. This rule is not fully implemented. At a minimum, all databases do preserve two constraints through SQL. No component of a primary key can have a null value. If a foreign key is defined in one table, any value in it must exist as a primary key in another table.

Rule 11: Database Distribution :-

A user should be totally unaware of whether or not that database is distributed (whether parts of the database exist in multiple locations). A variety of reasons make this rule difficulty to implement.

Rule 12: Non Subversion Rule:-

There should be no way to modify the database structure other than through the multiple row database language (**like SQL**).

DBMS V/S RDBMS

DBMS	RDBMS
❖ In **DBMS** relationship between two tables or file are maintained programmatically.	❖ In **RDBMS** relationship between two tables or files can be specified at the time of table creation.
❖ DBMS does not support client/server architecture.	❖ Most of the RDBMS support client/server architecture.
❖ DBMS does not support distributed database.	❖ Most of the RDBMS support distributed database.
❖ In DBMS there is no security of data.	❖ IN RDBMS there are multiple levels of security ✓ Logging in O/S level ✓ Command level ✓ Object level
❖ Each table is given an extension in DBMS	❖ Many tables are grouped in one database in RDBMS
❖ DBMS may satisfy less than 7 to 8 rules of Dr.E.F.codd	❖ RDBMS may satisfy more than 7 to 8 rules of Dr.E.F.codd
❖ Naming Conventions: Field Record File	❖ Naming Conventions: Column, Attributes Row, Tuple, Entity Table, Relation, Entity Class

Consequences of Poor Database Design:

- Poor Design and Planning.
- Poor Naming Standard.
- Lack of Documentation.
- Lack of Testing.
- One table to hold all domain values.
- Ignoring Normalization.
- No using SQL facility to protect data integrity.
- Not using stored procedure to access data.

Anomaly:

Anomalies are problem that can occur in poorly planned, un normalized database where all the data in stored in one table.

Redundancy: -

When some data is stored multiple times unnecessarily in database.

Disadvantages:

1. Insert, update, delete anomalies
2. Incontinency (Data)
3. Increase in database size and increase in time to access data.

Types of Anomalies:

- **Insert**
- **Update**
- **Delete**

Insert Anomaly:

- **An insert anomaly occur when certain data or attributes cannot be inserted into database without the presence of other data or attributes.**

S_ID	Sname	BCode	BName	HOD Nm
1	Ankit	101	CS	XYZ
2	Sachin	101	CS	XYZ
3	Miral	101	CS	XYZ
4	Divyesh	102	ME	ABC
5	Rahul	102	ME	ABC
6	Priya	103	Civil	PQR

- Here we can not add new branch unless we have at least one student enrolled in the particular Branch.

Delete Anomaly:

- An delete Anomalies exists when certain attributes are lost because of the deletion of other attributes.

S_ID	Sname	BCode	BName	HOD Nm
1	Ankit	101	CS	XYZ
2	Sachin	101	CS	XYZ
3	Miral	101	CS	XYZ
4	Divyesh	102	ME	ABC
5	Rahul	102	ME	ABC
6	Priya	103	Civil	PQR

- Here we can not add new branch unless we have at least one student enrolled in the particular Branch

Update Anomaly:

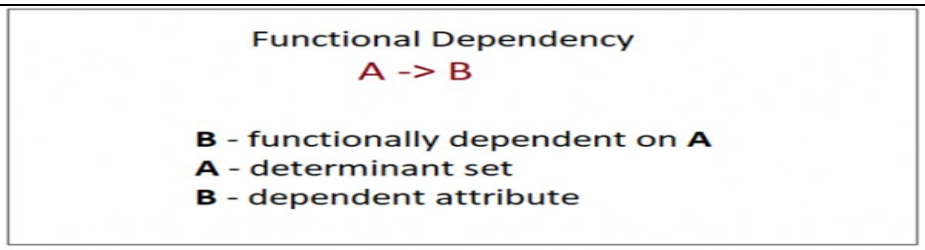
- An Update anomaly exists when one or more instances of duplicate data is updated, but not all.

S_ID	Sname	BCode	BName	HOD Nm
1	Ankit	101	CS	XYZ
2	Sachin	101	CS	XYZ
3	Miral	101	CS	XYZ
4	Divyesh	102	ME	ABC
5	Rahul	102	ME	ABC
6	Priya	103	Civil	PQR

- Here, In above table HOD name has been changed , it is now ABC instead of XYZ. So for updating a single entity we have to update all the columns value where branch name is CS.

Functional Dependency:-

- Functional dependency is the relationship that exists when one attribute in a relation uniquely determine another attribute.
- Consider If R is the relation with attributes x and y, so functional dependency between the attribute is represented as x->y which specifies y is functionally dependent on x.
- Functional Dependency is represented by -> (arrow sign)
- It is also known as database dependency.



Example:

- Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.
- Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.
- Functional dependency can be written as:

  Emp_Id → Emp_Name

  We can say that Emp_Name is functionally dependent on Emp_Id.

Types of Functional Dependency:-

- There are main two types of Functional Dependency.
  - Trivial Functional Dependency
  - Non-Trivial Functional Dependency

Trivial Functional Dependency:-

- The dependency of an attribute or a set of attributes is known as trivial functional dependency if the set of attributes includes that attribute.
- Symbolically: A->B is trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial: A->A & B->B, AB->A, AB->B, AB->AB
- **For example**: Consider a table with two columns Student_id and Student_Name.
- {Student_Id, Student_Name} -> Student_Id is a trivial functional dependency as Student_Id is a subset of {Student_Id, Student_Name}.
- That makes sense because if we know the values of Student_Id and Student_Name then the value of Student_Id can be uniquely determined.

  Also, Student_Id -> Student_Id & Student_Name -> Student_Name are trivial dependencies too.

Non-Trivial Functional Dependency:-

- If a functional dependency X->Y holds true where Y is not a subset of X then this dependency is called non trivial Functional dependency.

- **For example**:
  An employee table with three attributes: emp_id, emp_name,emp_address.
  The following functional dependencies are non-trivial:
  emp_id -> emp_name (emp_name is not a subset of emp_id)
  emp_id -> emp_address (emp_address is not a subset of emp_id)

Normalization:-

Normalization is a process that helps analyst or database designers to design table structures for an application. The focus of normalization is to attempt to reduce redundant table data to the very minimum. The normalization process, the collection of data in a single table is replaced, by the same data being distributed over multiple tables with a specific relationship.

- **Normalization is techniques that :**
  - Decompose data into 2-dimension tables.
  - Eliminates any relationship in which table data does fully depend upon the primary key of a record.
  - Eliminates any relationship that contains transitive dependencies.
- **Need for Normalization:**
  i. To structure the data between tables that data maintenance is simplified.
  ii. To allow data retrieval at optimal speed.
  iii. To simplify data maintenance through updates, inserts and deletes.
  iv. To reduce the need to restructure tables as new application requirements arise.
  v. To improve the quality of design for an application by rationalization of table data.
- **Benefits of Normalization:**

  - To remove redundancy.
  - To achieve consistency.
  - To increase accessing speed of data.
  - To improve the performance of the server.
  - To decrease accessing speed of data from the database.

Normal Forms of Normalization:-

- **First Normal Form:- (1 NF)**

When a table is decomposed into 2-dimensional tables with all repeating groups of data eliminated, the table data is said to be in its first normal form.

The reparative portion of data belonging to the record is termed as repeating groups.

Example:- Table – EmpProj

Field	Key
Proj_no	-
Proj_nm	-
Emp_no	-

Proj_no	Proj_nm	Emp No	EmpName	Rate Cate	Hour Rate
P001	Using My SQL	E001	Ashish	A	700
P001	**Using My SL**	E002	Bhavesh	B	600
P001	Using My SQL	E006	Manoj	C	450

Emp_nm	-
Rate_cat	-
Hour_rate	-

P002	Using MSOffice	E001	Vaishali	A	500
P002	Using MSOffice	E002	Dhaval	B	590

In Above data there are few problems.

- The project name in the second record is misspelled. This can be solved by removing duplicates.
- Data is repeated and thus occupies more spaces.

A table is in 1NF if:

- There are no repeating groups.
- All the key attributes are defined.
- All attributes are dependent on a primary key.

So far, there are no keys and there are repeating groups. So remove repeating groups and define the primary key.

To convert a table to its 1 NF:

Table1: EmpProj

Field	Key
Proj_no	**Primary Key**
Proj_nm	-
Emp_no	**Primary Key**
Emp_nm	-
Rate_cat	-
Hour_rate	-

The table is now 1st Normal form.

- **Second Normal Form:- (2NF)**

A table is said to be in its second normal form when each record in the table is in the first normal form and each column in the record is fully dependent on its primary key.

A table is in 2NF if:

- It is 1st Normal form.
- It includes no partial dependencies(when an attributes is dependent on only part of primary key)

The steps to convert a table to its Second Normal Form:

1. Find and remove fields that are related to the only part of the key.
2. Group the removed items in another table.
3. Assign the new table with the key. (i.e. Part of a whole composite key)

Going through all of the fields reveals the following.

- Proj_nm is only dependent on proj_no.
- Emp_nm, Rate_cat and Hour_Rate dependent only Emp_no

To convert a table to its 2 NF:

Table1: EmpProj **Table2: Emp:** **Table3: Proj**

Field	Key
Proj_no	**Primary Key**
Emp_no	**Primary Key**

Field	Key
Emp_no	**Primary Key**
Emp_nm	-
Rate_cat	
Hour_rate	-

Field	Key
Proj_no	**Primary Key**
Proj_nm	-

The table is now 2nd Normal form.

- **Third Normal Form:- (3 NF)**

Table data is said to be in 3 Normal format when all transitive dependencies are removed from the data.

A table is in 3NF if:

- It is in 2nd Normal Form.
- It contain no transitive dependencies (where a non-key attributes is dependent on another non-key attributes)

Going through all of the fields reveals the following.

- Emp table is the only one with more than one non-key attributes.
- Emp_nm is not dependent on either rate_cat or hour_rate.
- Hour_rate is dependent on rate_cat.

This leads to the following 4 tables.

To convert a table to its 3 NF:

Table1: EmpProj

Field	Key
Proj_no	**Primary Key**
Emp_no	**Primary Key**

Table2 : Emp

Field	Key
Emp_no	**Primary Key**
Emp_nm	-
Rate_cat	-

Table3: Rate

Field	Key
Rate_cat	**Primary Key**
Hour_rate	-

Table4: Proj

Field	Key
Proj_no	**Primary Key**
Proj_nm	-

The table is now 3rd Normal form.