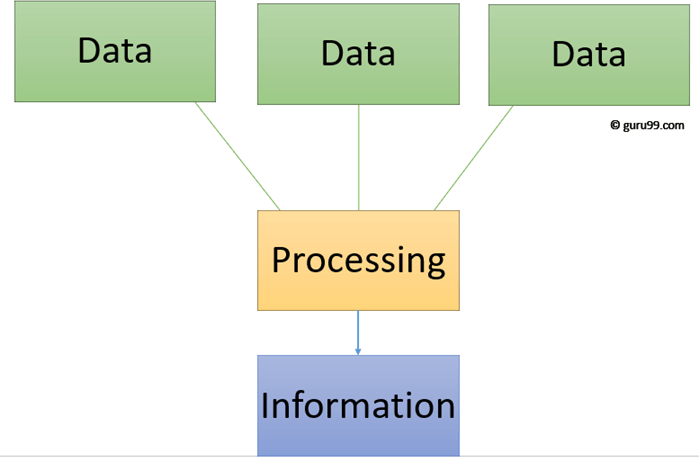
**Data:**

* Data is collection of raw facts and figures, such as numbers, words, measurements, observations or just descriptions of things.
* Alone it tells you nothing.
* The real goal is turn the data into information.
* Data is not in meaningful way, it is unorganized format.
* Once the data is processed, organized, structured or presented in given context, it can become useful. EX: your name, age, height etc.

Information:

* Information is credited from the data
* Information is nothing but refined data that have been put into a meaningful and useful context.
* Information is the back bone of any otganization.it consists of images, text, document and voice etc.
* The 3 key attributes of information are ,
  + Accuracy(correctness)
  + Timelines
  + Relevancy



Data Base:

* The data is an organised form of the word “DATUM” that means “single piece of information”.
* Data base is organised collection of data, so that it can be easily stored, managed and accessed electrically from a computer system.
* It is also used to organise the data in the form of rows and columns and index etc. To make it easier to find relevant information.
* The main purpose of database is to operate large amount of information by storing, retrieving and managing the data.
* There are many databases are available , such as MySQL, Sybase, Oracle, SQL server etc.,
* A cylindrical structure is used to represent the data.

DBMS:

* DBMS is software which is used to manage the database.
* It is a collection of programs that enable to store, modify, and extract information from the database.
* It is a piece of software that provides services for accessing a database, while maintaining all the required features of data
* It provides security and protection to the database; in case of multiple users it also maintains data consistency.

Data Models

***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 organisation 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

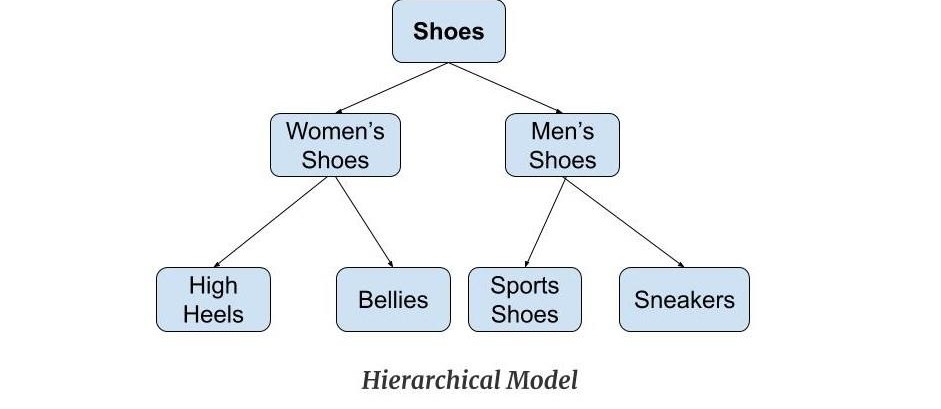
**Hierarchical Model**

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:** We can represent the relationship between the shoes present on a shopping website in the following way:



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.

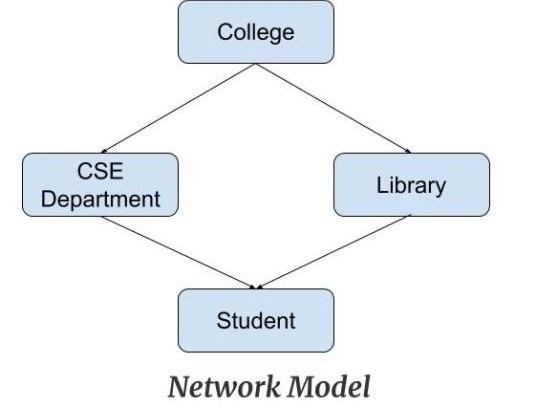
**Network Model**

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**

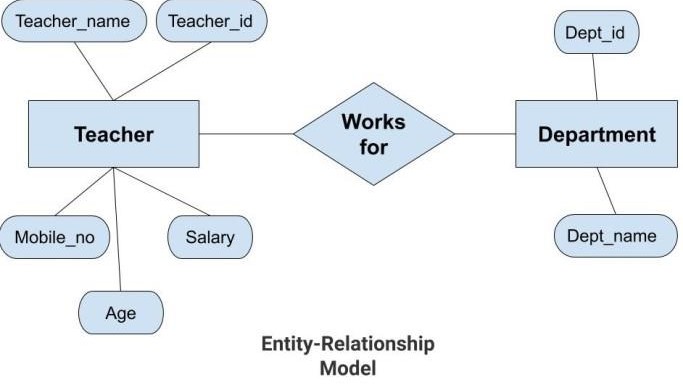
* As more and more relationships need to be handled the system might get complex. So, a user must be having detailed knowledge of the model to work with the model.
* Any change like updation, deletion, insertion is very complex.

**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.

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.

**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:** In this example, we have an Employee table.

*Features of Relational Model*

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

*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.

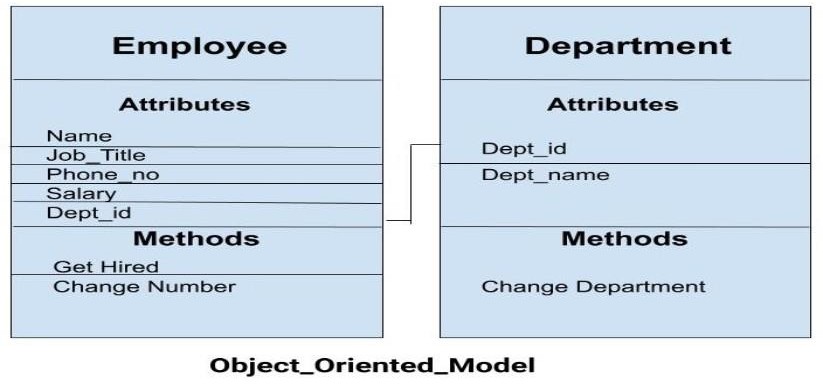
But all these disadvantages are minor as compared to the advantages of the relational model. These problems can be avoided with the help of proper implementation and organisation.

**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. This can be understood by the example given below.



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.

**What is ER Diagram? Entity Relationship Diagram – ER Diagram in DBMS (or) ER Model**

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.

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.

Rectangle**: Represents Entity sets.**

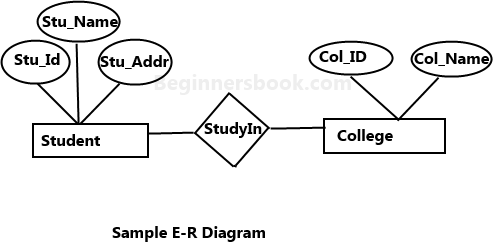
Ellipses**: Attributes**

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**

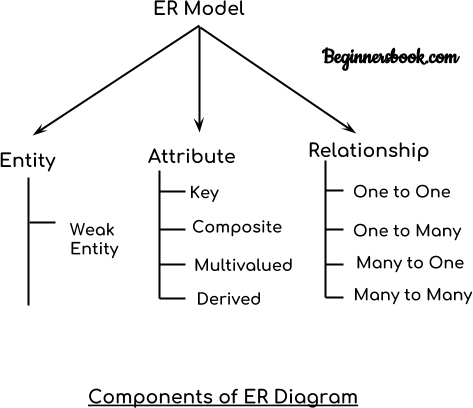
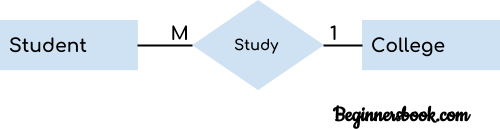
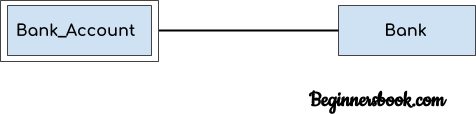
Double Lines: Total participation of an entity in a relationship set Components of a ER Diagram



**As shown in the above diagram, an ER diagram has three main components:**

1. Entity
2. **Attribute**
3. Relationship
   1. **Entity**

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



Weak Entity:

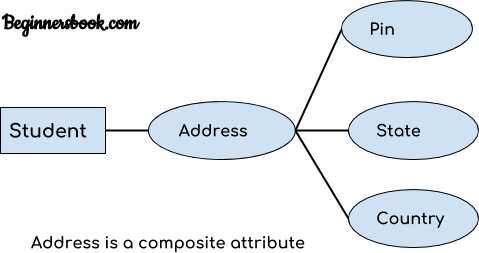
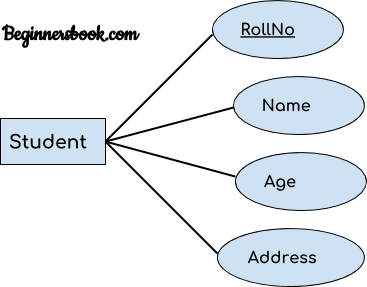
An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle. For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.

* 1. **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 attributes:



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**.

* 1. Composite attribute*:*

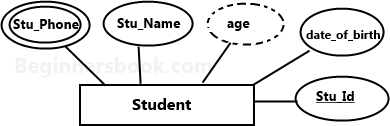
An attribute that is a combination of other attributes is known as composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.

* 1. 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.

* 1. Derived attribute:

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

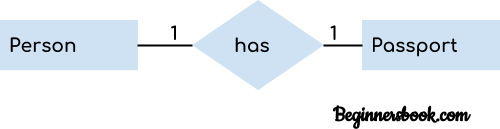


E-R diagram with multivalued and derived attributes:

**3. Relationship**

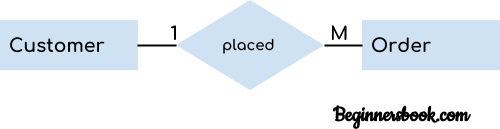
A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:

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



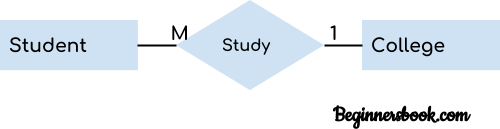
When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship. For example, a person has only one passport and a passport is given to one person.

* 1. One to many Relationship



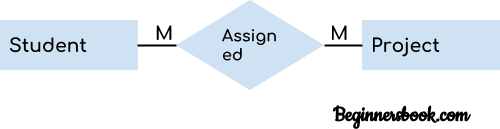
When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship. For example – a customer can place many orders but a order cannot be placed by many customers.

* 1. 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.

* 1. 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.

What is Relational Algebra in DBMS?

Relational algebra is a **procedural** query language that works on relational model. The purpose of a query language is to retrieve data from database or perform various operations such as insert, update, delete on the data. When I say that relational algebra is a procedural query language, it means that it tells what data to be retrieved and how to be retrieved.

On the other hand relational calculus is a non-procedural query language, which means it tells what data to be retrieved but doesn’t tell how to retrieve it. We will discuss relational calculus in a separate tutorial.

Types of operations in relational algebra

We have divided these operations in two categories:

1. Basic Operations
2. Derived Operations

Basic/Fundamental Operations:

* 1. Select (σ)
  2. Project (∏)
  3. Union (𝖴)
  4. Set Difference (-)
  5. Cartesian product (X)
  6. Rename (ρ)

Derived Operations:

* + 1. Natural Join (⋈)
    2. Left, Right, Full outer join (⟕, ⟖, ⟗)
    3. Intersection (∩)
    4. Division (÷)

Let’s discuss these operations one by one with the help of examples.

Select Operator (σ)

Select Operator is denoted by sigma (σ) and it is used to find the tuples (or rows) in a relation (or table) which satisfy the given condition.

If you understand little bit of SQL then you can think of it as a [where clause in SQL](https://beginnersbook.com/2014/05/where-clause-in-sql/), which is used for the same purpose.

Syntax of Select Operator (σ)

σ Condition/Predicate(Relation/Table name)

Select Operator (σ) Example

|  |  |  |  |
| --- | --- | --- | --- |
| Table: CUSTOMER | | | |
| Customer\_Id | | Customer\_Name Customer\_City | |
| C10100 | Steve | | Agra |
| C10111 | Raghu | | Agra |
| C10115 | Chaitanya | | Noida |
| C10117 | Ajeet | | Delhi |
| C10118 | Carl | | Delhi |

**Query:**

σ Customer\_City="Agra" (CUSTOMER)

Output:

|  |  |  |  |
| --- | --- | --- | --- |
| Customer\_Id Customer\_Name | | | Customer\_City |
| C10100 | Steve | Agra | |
| C10111 | Raghu | Agra | |

**Project Operator (∏)**

Project operator is denoted by ∏ symbol and it is used to select desired columns (or attributes) from a table (or relation).

Project operator in relational algebra is similar to the [Select statement in SQL](https://beginnersbook.com/2018/11/sql-select/).

Syntax of Project Operator (∏)

∏ column\_name1, column\_name2, , column\_nameN(table\_name)

Project Operator (∏) Example

In this example, we have a table CUSTOMER with three columns, we want to fetch only two columns of the table, which we can do with the help of Project Operator ∏.

Table: CUSTOMER

Customer\_Id Customer\_Name Customer\_City

|  |  |  |  |
| --- | --- | --- | --- |
| C10100 |  | Steve | Agra |
| C10111 |  | Raghu | Agra |
| C10115 |  | Chaitanya | Noida |
| C10117 |  | Ajeet | Delhi |
| C10118 |  | Carl | Delhi |

Query:

∏ Customer\_Name, Customer\_City (CUSTOMER)

Output:

Customer\_Name Customer\_City

Steve Agra

Raghu Agra

Chaitanya Noida Ajeet Delhi

Carl Delhi

**Union Operator (**𝖴**)**

Union operator is denoted by 𝖴 symbol and it is used to select all the rows (tuples) from two tables (relations).

Lets discuss union operator a bit more. Lets say we have two relations R1 and R2 both have same columns and we want to select all the tuples(rows) from these relations then we can apply the union operator on these relations.

**Note:** The rows (tuples) that are present in both the tables will only appear once in the union set. In short you can say that there are no duplicates present after the union operation.

Syntax of Union Operator (𝖴)

table\_name1 𝖴 table\_name2

Union Operator (𝖴) Example

Table 1: COURSE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Course\_Id | | Student\_Name | | Student\_Id |
| C101 | Aditya | | S901 | |
| C104 | Aditya | | S901 | |
| C106 | Steve | | S911 | |
| C109 | Paul | | S921 | |
| C115 | Lucy | | S931 | |

Table 2: STUDENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student\_Id | | Student\_Name | | Student\_Age |
| S901 | Aditya | | 19 | |
| S911 | Steve | | 18 | |
| S921 | Paul | | 19 | |
| S931 | Lucy | | 17 | |
| S941 | Carl | | 16 | |
| S951 | Rick | | 18 | |

Query:

∏ Student\_Name (COURSE) 𝖴 ∏ Student\_Name (STUDENT)

Output:

Student\_Name

Aditya Carl Paul Lucy Rick Steve

**Note:** As you can see there are no duplicate names present in the output even though we had few

common names in both the tables, also in the COURSE table we had the duplicate name itself.

Intersection Operator (∩)

Intersection operator is denoted by ∩ symbol and it is used to select common rows (tuples) from two tables (relations).

Lets say we have two relations R1 and R2 both have same columns and we want to select all those tuples(rows) that are present in both the relations, then in that case we can apply intersection operation on these two relations R1 ∩ R2.

**Note:** Only those rows that are present in both the tables will appear in the result set.

Syntax of Intersection Operator (∩)

table\_name1 ∩ table\_name2

Intersection Operator (∩) Example

Lets take the same example that we have taken above. Table 1: COURSE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Course\_Id | | Student\_Name | | Student\_Id |
| C101 | Aditya | | S901 | |
| C104 | Aditya | | S901 | |

C106 C109 C115

Steve Paul Lucy

S911 S921 S931

Table 2: STUDENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student\_Id | | Student\_Name | | Student\_Age |
| S901 | Aditya | | 19 | |
| S911 | Steve | | 18 | |
| S921 | Paul | | 19 | |
| S931 | Lucy | | 17 | |
| S941 | Carl | | 16 | |
| S951 | Rick | | 18 | |

Query:

∏ Student\_Name (COURSE) ∩ ∏ Student\_Name (STUDENT)

**Output:**

Student\_Name

Aditya Steve Paul Lucy

Set Difference (-)

Set Difference is denoted by – symbol. Lets say we have two relations R1 and R2 and we want to select all those tuples(rows) that are present in Relation R1 but **not** present in Relation R2, this can be done using Set difference R1 – R2.

Syntax of Set Difference (-)

table\_name1 - table\_name2

**Set Difference (-) Example**

Lets take the same tables COURSE and STUDENT that we have seen above.

Query:

Lets write a query to select those student names that are present in STUDENT table but not present in COURSE table.

∏ Student\_Name (STUDENT) - ∏ Student\_Name (COURSE)

Output:

Student\_Name

Carl Rick

**Cartesian product (X)**

Cartesian Product is denoted by X symbol. Lets say we have two relations R1 and R2 then the cartesian product of these two relations (R1 X R2) would combine each tuple of first relation R1 with the each tuple of second relation R2. I know it sounds confusing but once we take an example of this, you will be able to understand this.

Syntax of Cartesian product (X)

R1 X R2

Cartesian product (X) Example

Table 1: R

|  |  |
| --- | --- |
| Col\_A | Col\_B |
| AA | 100 |
| BB | 200 |
| CC | 300 |
| Table 2: S | |
| Col\_X | Col\_Y |
| XX | 99 |
| YY | 11 |
| ZZ | 101 |

Query:

Lets find the cartesian product of table R and S.

R X S

Output:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Col\_A | | Col\_B | | Col\_X | | Col\_Y |
| AA |  | 100 | XX | | 99 | |
| AA |  | 100 | YY | | 11 | |
| AA |  | 100 | ZZ | | 101 | |
| BB |  | 200 | XX | | 99 | |
| BB |  | 200 | YY | | 11 | |
| BB |  | 200 | ZZ | | 101 | |
| CC |  | 300 | XX | | 99 | |
| CC |  | 300 | YY | | 11 | |
| CC |  | 300 | ZZ | | 101 | |

**Note:** The number of rows in the output will always be the cross product of number of rows in each table. In our example table 1 has 3 rows and table 2 has 3 rows so the output has 3×3 = 9 rows.

Rename (ρ)

Rename (ρ) operation can be used to rename a relation or an attribute of a relation.

Rename (ρ)

**Syntax:** ρ(new\_relation\_name, old\_relation\_name)

Rename (ρ) Example

Lets say we have a table customer, we are fetching customer names and we are renaming the resulted relation to CUST\_NAMES.

Table: CUSTOMER

|  |  |  |  |
| --- | --- | --- | --- |
| Customer\_Id | | Customer\_Name Customer\_City | |
| C10100 | Steve | | Agra |
| C10111 | Raghu | | Agra |
| C10115 | Chaitanya | | Noida |
| C10117 | Ajeet | | Delhi |
| C10118 | Carl | | Delhi |

Query:

ρ(CUST\_NAMES, ∏(Customer\_Name)(CUSTOMER))

**Output:**

CUST\_NAMES

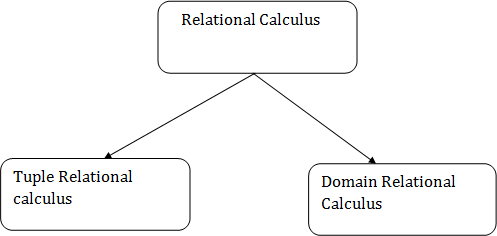
Steve Raghu Chaitanya Ajeet Carl

##### What is Relational Calculus?

Relational calculus is a non-procedural query language that tells the system what data to be retrieved but doesn’t tell how to retrieve it. the user is concerned with the details of how to obtain the end results.

* The relational calculus tells what to do but never explains how to do.

Types of Relational calculus:



**1.Tuple Relational Calculus (TRC):**

* The tuple relational calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation.
* The result of the relation can have one or more tuples.

Notation:

{T | P (T)} or {T | Condition (T)} Where **T** is the resulting tuples

**P(T)** is the condition used to fetch T.

For example:

{ T.name | Author(T) AND T.article = 'database' }

**OUTPUT:** This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.

TRC (tuple relation calculus) can be quantified. In TRC, we can use Existential (∃) and Universal Quantifiers (∀).

For example:

{ R| ∃T ∈ Authors(T.article='database' AND R.name=T.name)}

**Output:** This query will yield the same result as the previous one.

**. Tuple Relational Calculus (TRC)**

Tuple relational calculus is used for selecting those tuples that satisfy the given condition.

Table: Student

|  |  |  |
| --- | --- | --- |
| First\_Name | Last\_Name | Age |
| Ajeet | Singh | 30 |
| Chaitanya | Singh | 31 |
| Rajeev | Bhatia | 27 |
| Carl | Pratap | 28 |

Lets write relational calculus queries.

Query to display the last name of those students where age is greater than 30

{ t.Last\_Name | Student(t) AND t.age > 30 }

In the above query you can see two parts separated by | symbol. The second part is

where we define the condition and in the first part we specify the fields which we want to display for the selected tuples.

The result of the above query would be:

Last\_Name

Singh

Query to display all the details of students where Last name is ‘Singh’

{ t | Student(t) AND t.Last\_Name = 'Singh' }

**Output:**

|  |  |  |
| --- | --- | --- |
| First\_Name | Last\_Name | Age |
| Ajeet | Singh | 30 |
| Chaitanya | Singh | 31 |

1. **Domain Relational Calculus (DRC):**

* The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes.
* Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives 𝖠

(and), ∨ (or) and ┓ (not).

* It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable.

Notation:

{ a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)} Where

For example:

**a1, a2** are attributes

**P** stands for formula built by inner attributes

{< article, page, subject > | ∈ javatpoint 𝖠 subject = 'database'}

**Output:** This query will yield the article, page, and subject from the relational javatpoint, where the subject is a database.

In domain relational calculus the records are filtered based on the domains. Again we take the same table to understand how DRC works.

Table: Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Ajeet |  | Singh |  | 30 |
| Chaitanya |  | Singh |  | 31 |
| Rajeev |  | Bhatia |  | 27 |
| Carl |  | Pratap |  | 28 |

First\_Name Last\_Name Age

Query to find the first name and age of students where student age is greater than 27

{< First\_Name, Age > | ∈ Student 𝖠 Age > 27}

Note:

The symbols used for logical operators are: 𝖠 for AND, ∨ for OR and ┓ for NOT.

|  |  |  |
| --- | --- | --- |
| **Output:** |  | |
| First\_Name |  | Age |
| Ajeet |  | 30 |
| Chaitanya |  | 31 |
| Carl |  | 28 |

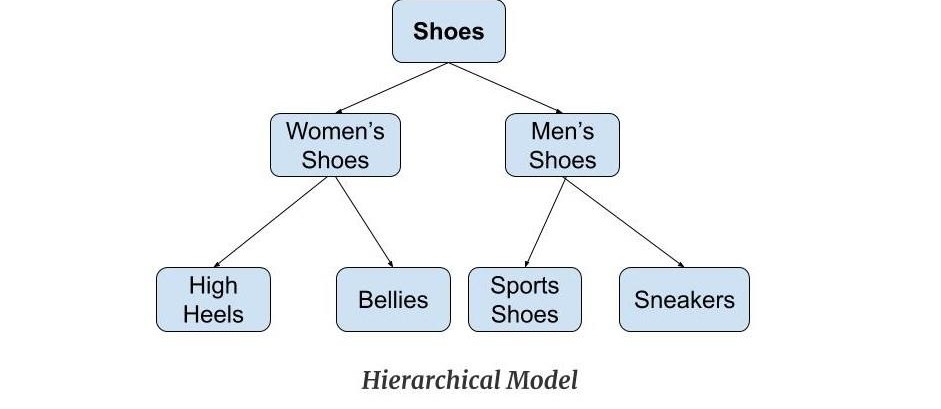
**Hierarchical Model**

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:** We can represent the relationship between the shoes present on a shopping website in the following way:



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.

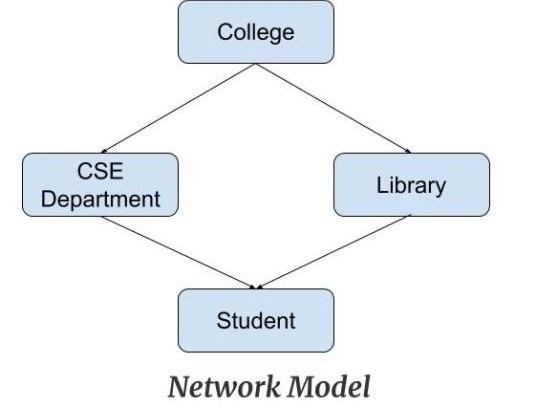
**Network Model**

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

# **Normalization**

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

* Making relations very large.
* It isn't easy to maintain and update data as it would involve searching many records in relation.
* Wastage and poor utilization of disk space and resources.
* The likelihood of errors and inconsistencies increases.

So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

## What is Normalization?

* Normalization is the process of organizing the data in the database.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
* Normalization divides the larger table into smaller and links them using relationships.
* The normal form is used to reduce redundancy from the database table.

Why do we need Normalization?

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

**Data modification anomalies can be categorized into three types:**

* **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
* **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
* **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

## Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

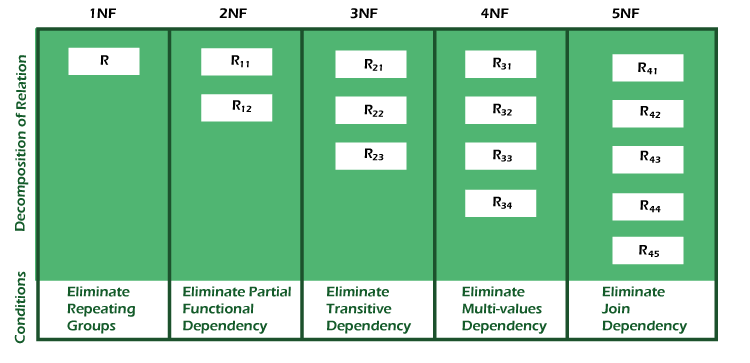
|  |  |
| --- | --- |
| **Normal Form** | **Description** |
| [1NF](https://www.javatpoint.com/dbms-first-normal-form) | A relation is in 1NF if it contains an atomic value. |
| [2NF](https://www.javatpoint.com/dbms-second-normal-form) | A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key. |
| [3NF](https://www.javatpoint.com/dbms-third-normal-form) | A relation will be in 3NF if it is in 2NF and no transition dependency exists. |
| BCNF | A stronger definition of 3NF is known as Boyce Codd's normal form. |
| [4NF](https://www.javatpoint.com/dbms-forth-normal-form) | A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency. |
| [5NF](https://www.javatpoint.com/dbms-fifth-normal-form) | A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless. |

## Advantages of Normalization

* Normalization helps to minimize data redundancy.
* Greater overall database organization.
* Data consistency within the database.
* Much more flexible database design.
* Enforces the concept of relational integrity.

## Disadvantages of Normalization

* You cannot start building the database before knowing what the user needs.
* The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
* It is very time-consuming and difficult to normalize relations of a higher degree.
* Careless decomposition may lead to a bad database design, leading to serious problems.



# **First Normal Form (1NF)**

* A relation will be 1NF if it contains an atomic value.
* It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
* First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

**Example:** Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP\_PHONE.

**EMPLOYEE table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_PHONE** | **EMP\_STATE** |
| 14 | John | 7272826385, 9064738238 | UP |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389, 8589830302 | Punjab |

The decomposition of the EMPLOYEE table into 1NF has been shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_PHONE** | **EMP\_STATE** |
| 14 | John | 7272826385 | UP |
| 14 | John | 9064738238 | UP |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389 | Punjab |
| 12 | Sam | 8589830302 | Punjab |

# **Second Normal Form (2NF)**

* In the 2NF, relational must be in 1NF.
* In the second normal form, all non-key attributes are fully functional dependent on the primary key

**Example:** Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

**TEACHER table**

|  |  |  |
| --- | --- | --- |
| **TEACHER\_ID** | **SUBJECT** | **TEACHER\_AGE** |
| 25 | Chemistry | 30 |
| 25 | Biology | 30 |
| 47 | English | 35 |
| 83 | Math | 38 |
| 83 | Computer | 38 |

In the given table, non-prime attribute TEACHER\_AGE is dependent on TEACHER\_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

**TEACHER\_DETAIL table:**

|  |  |
| --- | --- |
| **TEACHER\_ID** | **TEACHER\_AGE** |
| 25 | 30 |
| 47 | 35 |
| 83 | 38 |

**TEACHER\_SUBJECT table:**

|  |  |
| --- | --- |
| **TEACHER\_ID** | **SUBJECT** |
| 25 | Chemistry |
| 25 | Biology |
| 47 | English |
| 83 | Math |
| 83 | Computer |

# **Third Normal Form (3NF)**

* A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
* 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
* If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → Y.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

**Example:**

**EMPLOYEE\_DETAIL table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_ZIP** | **EMP\_STATE** | **EMP\_CITY** |
| 222 | Harry | 201010 | UP | Noida |
| 333 | Stephan | 02228 | US | Boston |
| 444 | Lan | 60007 | US | Chicago |
| 555 | Katharine | 06389 | UK | Norwich |
| 666 | John | 462007 | MP | Bhopal |

**Super key in the table above:**

* 1. {EMP\_ID}, {EMP\_ID, EMP\_NAME}, {EMP\_ID, EMP\_NAME, EMP\_ZIP}....so on

**Candidate key:** {EMP\_ID}

**Non-prime attributes:** In the given table, all attributes except EMP\_ID are non-prime.

Here, EMP\_STATE & EMP\_CITY dependent on EMP\_ZIP and EMP\_ZIP dependent on EMP\_ID. The non-prime attributes (EMP\_STATE, EMP\_CITY) transitively dependent on super key(EMP\_ID). It violates the rule of third normal form.

That's why we need to move the EMP\_CITY and EMP\_STATE to the new <EMPLOYEE\_ZIP> table, with EMP\_ZIP as a Primary key.

**EMPLOYEE table:**

|  |  |  |
| --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **EMP\_ZIP** |
| 222 | Harry | 201010 |
| 333 | Stephan | 02228 |
| 444 | Lan | 60007 |
| 555 | Katharine | 06389 |
| 666 | John | 462007 |

**EMPLOYEE\_ZIP table:**

|  |  |  |
| --- | --- | --- |
| **EMP\_ZIP** | **EMP\_STATE** | **EMP\_CITY** |
| 201010 | UP | Noida |
| 02228 | US | Boston |
| 60007 | US | Chicago |
| 06389 | UK | Norwich |
| 462007 | MP | Bhopal |

# **Boyce Codd normal form (BCNF)**

* BCNF is the advance version of 3NF. It is stricter than 3NF.
* A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
* For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

**Example:** Let's assume there is a company where employees work in more than one department.

**EMPLOYEE table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_COUNTRY** | **EMP\_DEPT** | **DEPT\_TYPE** | **EMP\_DEPT\_NO** |
| 264 | India | Designing | D394 | 283 |
| 264 | India | Testing | D394 | 300 |
| 364 | UK | Stores | D283 | 232 |
| 364 | UK | Developing | D283 | 549 |

**In the above table Functional dependencies are as follows:**

1. EMP\_ID  →  EMP\_COUNTRY
2. EMP\_DEPT  →   {DEPT\_TYPE, EMP\_DEPT\_NO}

|  |  |
| --- | --- |
| **EMP\_ID** | **EMP\_COUNTRY** |
| 264 | India |
| 264 | India |

**EMP\_DEPT table:**

|  |  |  |
| --- | --- | --- |
| **EMP\_DEPT** | **DEPT\_TYPE** | **EMP\_DEPT\_NO** |
| Designing | D394 | 283 |
| Testing | D394 | 300 |
| Stores | D283 | 232 |
| Developing | D283 | 549 |

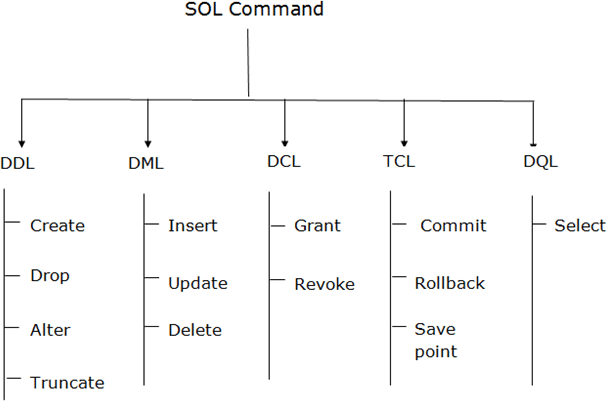
**EMP\_DEPT\_MAPPING table:**

|  |  |
| --- | --- |
| **EMP\_ID** | **EMP\_DEPT** |
| D394 | 283 |
| D394 | 300 |
| D283 | 232 |
| D283 | 549 |

**MYSQL COMMAND SYNTAX**

SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.

SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.



1. Data Definition Language (DDL)
   * DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
   * All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

* + CREATE
  + ALTER
  + DROP
  + TRUNCATE

**CREATE** It is used to create a new table in the database.

Syntax:

CREATE TABLE TABLE\_NAME (COLUMN\_NAME DATATYPES[,. ]);

Example:

CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

**DROP:** It is used to delete both the structure and record stored in the table.

Syntax

DROP TABLE table\_name;

Example

DROP TABLE EMPLOYEE;

**ALTER:** It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

ALTER TABLE table\_name ADD column\_name COLUMN-definition; To modify existing column in the table:

ALTER TABLE table\_name MODIFY(column\_definitions. );

EXAMPLE

EXAMPLE

ALTER TABLE STU\_DETAILS ADD(ADDRESS VARCHAR2(20)); ALTER TABLE STU\_DETAILS MODIFY (NAME VARCHAR2(20));

**d. TRUNCATE:** It is used to delete all the rows from the table and free the space containing the table.

Syntax:

TRUNCATE TABLE table\_name;

Example:

TRUNCATE TABLE EMPLOYEE;

1. Data Manipulation Language
   * DML commands are used to modify the database. It is responsible for all form of changes in the database.
   * The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML

* + INSERT
  + UPDATE
  + DELETE

**INSERT:** The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

INSERT INTO TABLE\_NAME (col1, col2, …. col N) VALUES (value1, value2, , valueN);

Or

INSERT INTO TABLE\_NAME VALUES (value1, value2, value3, valueN);

For example:

INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");

**UPDATE:** This command is used to update or modify the value of a column in the table.

Syntax:

UPDATE table\_name SET [column\_name1= value1,...column\_nameN = valueN]

[WHERE CONDITION]

For example:

UPDATE students SET User\_Name = 'Sonoo' WHERE Student\_Id = '3'

**DELETE:** It is used to remove one or more row from a table.

Syntax:

DELETE FROM table\_name [WHERE condition];

For example:

DELETE FROM javatpoint WHERE Author="Sonoo";

;

Data Control Language

DCL commands are used to grant and take back authority from any database user. Here are some commands that come under DCL:

* + Grant
  + Revoke

**Grant:** It is used to give user access privileges to a database.

Example

GRANT SELECT, UPDATE ON MY\_TABLE TO SOME\_USER, ANOTHER\_USER;

**Revoke:** It is used to take back permissions from the user.

Example

REVOKE SELECT, UPDATE ON MY\_TABLE FROM USER1, USER2;

Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

* + COMMIT
  + ROLLBACK
  + SAVEPOINT

**Commit:** Commit command is used to save all the transactions to the database.

Syntax:

COMMIT;

Example:

DELETE FROM CUSTOMERS WHERE AGE = 25; COMMIT;

**b. Rollback:** Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

ROLLBACK;

Example:

DELETE FROM CUSTOMERS WHERE AGE = 25; ROLLBACK;

**SAVEPOINT:** It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

SAVEPOINT SAVEPOINT\_NAME;

Data Query Language

DQL is used to fetch the data from the database. It uses only one command:

SELECT

**SELECT:** This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

SELECT expressions FROM TABLES

WHERE conditions;

For example:

SELECT emp\_name FROM employee WHERE age > 20;