Database Management System (DBMS)

A database management system (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure. It also defines rules to validate and manipulate this data.

Some of the advantages are given below −

a. Reducing Data Redundancy
b. Sharing of Data
c. Data Integrity
d. Data Security
e. Privacy
f. Backup and Recovery
g. Data Consistency

Database Models

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system.

While the **Relational Model** is the most widely used database model, there are other models too:

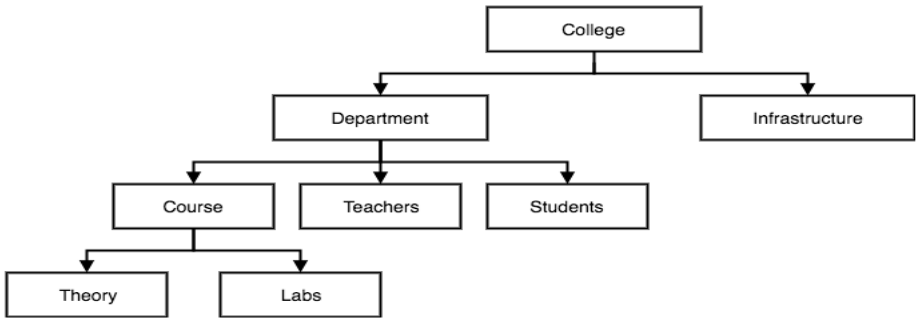
- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

1. Hierarchical Model

This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked.

The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.

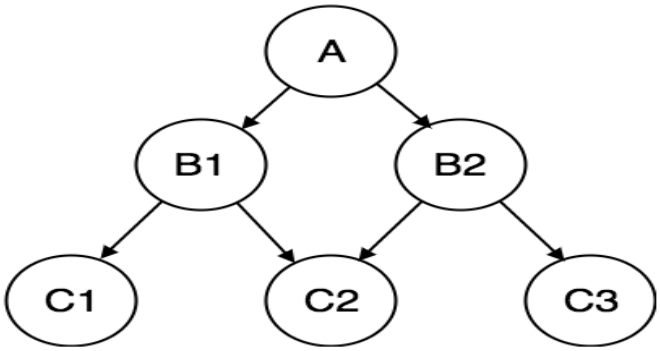
In this model, a child node will only have a single parent node.



- **Advantages of Hierarchical model**
  1. Simplicity
  2. Security
  3. Database Integrity
  4. Efficiency
- **Disadvantages**
  1. Complexity of Implementation:
  2. Difficulty in Management:
  3. Complexity of Programming:
  4. Poor Portability:
  5. Database Management Problems

2. Network Model

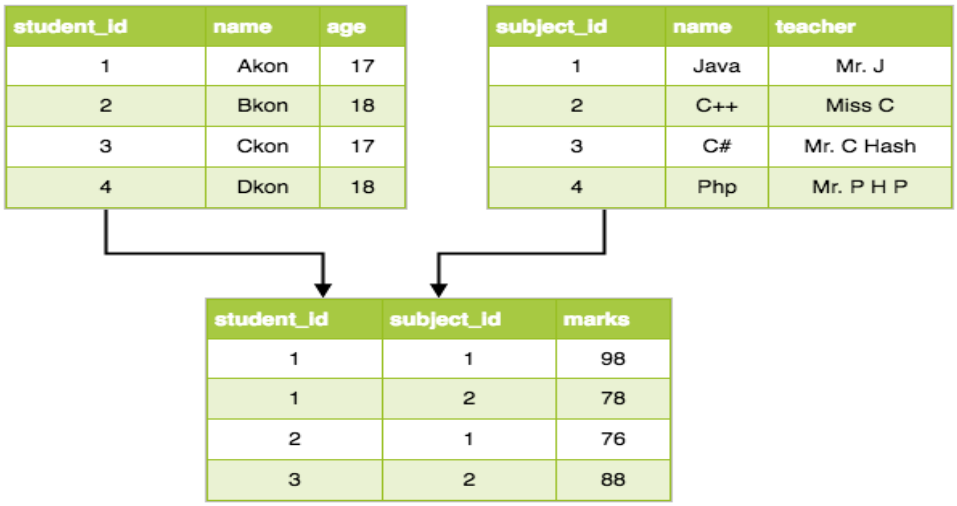
This is an extension of the Hierarchical model. In this model data is organized more like a graph, and are allowed to have more than one parent node.
In this database model data is more related as more relationships are established in this database model.
Also, as the data is more related, hence accessing the data is also easier and fast.
This database model was used to map many-to-many data relationships.
This was the most widely used database model, before Relational Model was introduced.



3. Relational Model

In this model, data is organized in two-dimensional **tables** and the relationship is maintained by storing a common field.
This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.
The basic structure of data in the relational model is tables.

All the information related to a particular type is stored in rows of that table.
Hence, tables are also known as **relations** in relational model.



student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

4. ER Model

ER Model stands for Entity Relationship Model is a high-level conceptual data model diagram.
ER model helps to systematically analyze data requirements to produce a well-designed database.
The ER Model represents real-world entities and the relationships between them.

Entities

- A real-world thing either living or non-living that is easily recognizable and non recognizable.
- It is anything in the enterprise that is to be represented in our database.
- It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.

Examples of entities:

- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

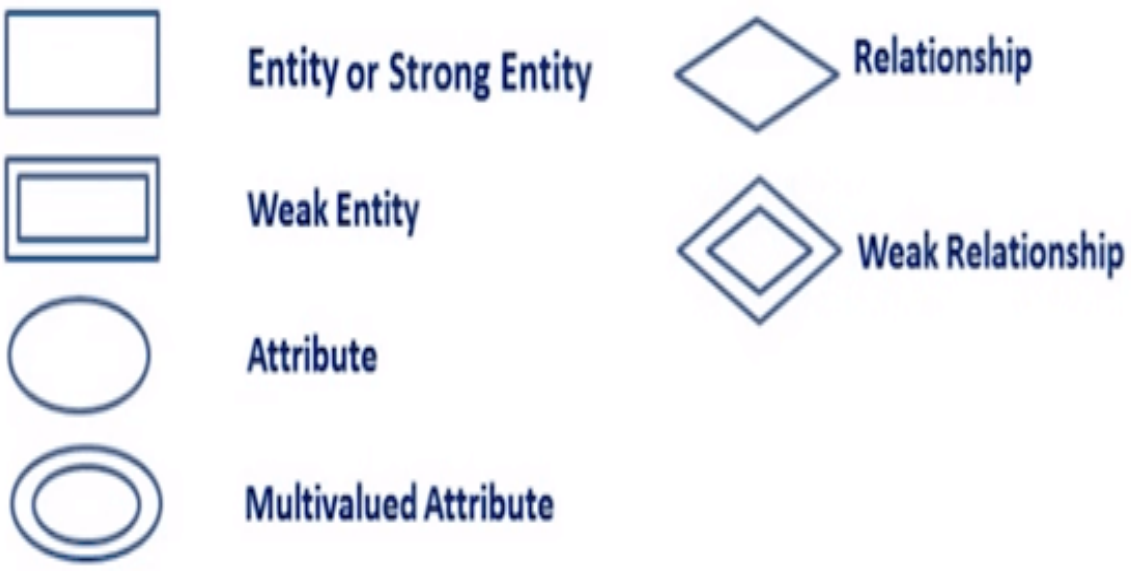
Attributes

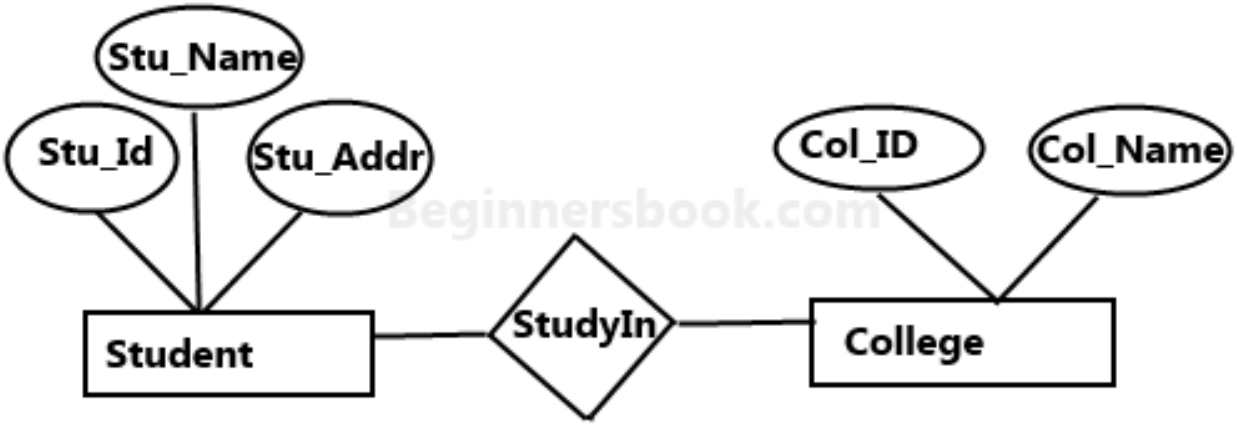
- It is a single-valued property of either an entity-type or a relationship-type.
- For example, a lecture might have attributes: time, date, duration, place, etc.
- An attribute in ER Diagram examples, is represented by an Ellipse

ER Diagrams Symbols & Notations

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes

Types of Attributes	Description
Simple attribute	Simple attributes can't be divided any further. For example, a student's contact number. It is also called an atomic value.
Composite attribute	It is possible to break down composite attribute. For example, a student's full name may be further divided into first name, second name, and last name.
Derived attribute	This type of attribute does not include in the physical database. However, their values are derived from other attributes present in the database. For example, age should not be stored directly. Instead, it should be derived from the DOB of that employee.
Multivalued attribute	Multivalued attributes can have more than one values. For example, a student can have more than one mobile number, email address, etc.





Sample E-R Diagram

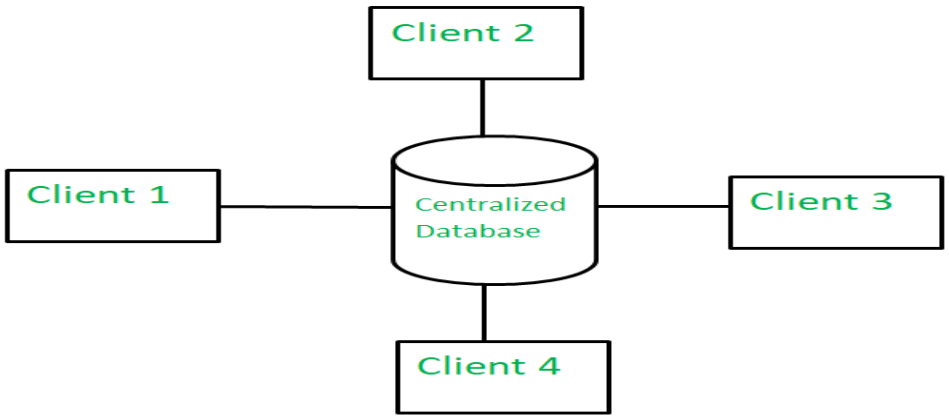
Centralized Database

A centralized database is basically a type of database that is stored, located as well as maintained at a single location only.

This type of database is modified and managed from that location itself.

The centralized location is accessed via an internet connection (LAN, WAN, etc).

This centralized database is mainly used by institutions or organizations.



Advantages –

- Since all data is stored at a single location only thus it is easier to access and co-ordinate data.
- The centralized database has very minimal data redundancy since all data is stored at a single place.
- It is cheaper in comparison to all other databases available.

Disadvantages –

- The data traffic in case of centralized database is more.
- If any kind of system failure occurs at centralized system then entire data will be destroyed.

Distributed Database :

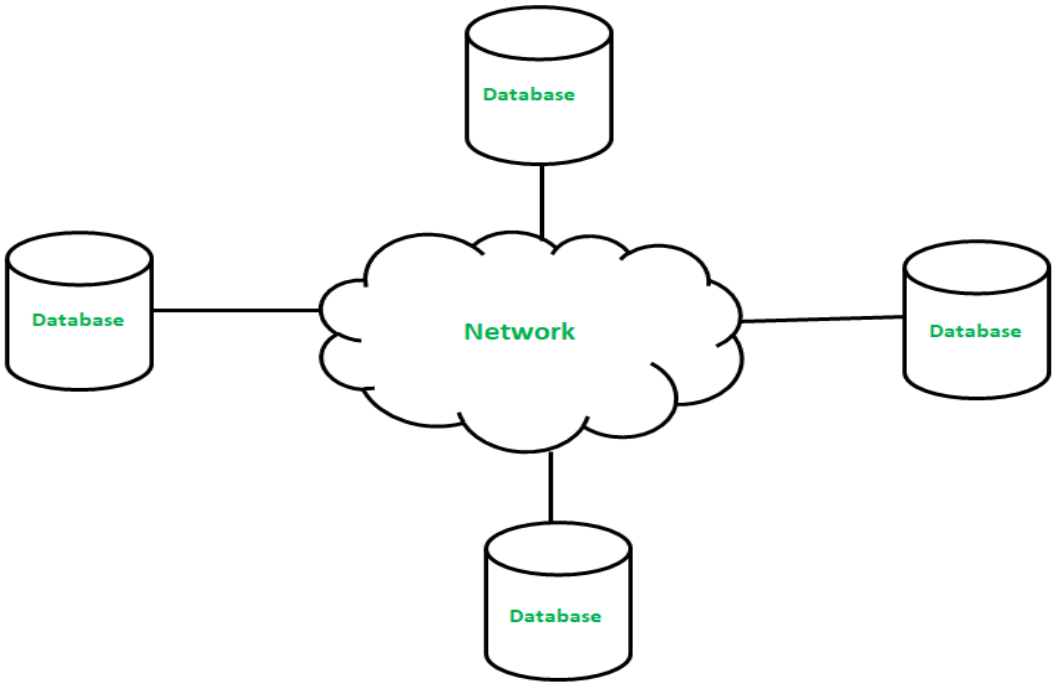
- A distributed database is basically a type of database which consists of multiple databases that are connected with each other and are spread across different physical locations.
- The data that is stored on various physical locations can thus be managed independently of other physical locations.
- The communication between databases at different physical locations is thus done by a computer network.

Advantages –

- This database can be easily expanded as data is already spread across different physical locations.
- The distributed database can easily be accessed from different networks.
- This database is more secure in comparison to centralized database.

Disadvantages –

- This database is very costly and it is difficult to maintain because of its complexity.
- In this database, it is difficult to provide a uniform view to user since it is spread across different physical locations.



Database Normalization

- Database Normalization is a technique of organizing the data in the database.
- Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies.
- It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.
- Normalization is used for mainly two purposes,
  - Eliminating redundant(useless) data.
  - Ensuring data dependencies make sense i.e data is logically stored.

Problems Without Normalization

rollno	name	branch	hod	office_tel
401	Akon	CSE	Mr. X	53337
402	Bkon	CSE	Mr. X	53337
403	Ckon	CSE	Mr. X	53337
404	Dkon	CSE	Mr. X	53337

- In the table above, we have data of 4 Computer Sci. students.
- As we can see, data for the fields branch, hod(Head of Department) and office_tel is repeated for the students who are in the same branch in the college, this is Data Redundancy.

Insertion Anomaly

- Suppose for a new admission, until and unless a student opts for a branch, data of the student cannot be inserted, or else we will have to set the branch information as NULL.
- Also, if we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students.
- These scenarios are nothing but Insertion anomalies.

Updation Anomaly

- What if Mr. X leaves the college? or is no longer the HOD of computer science department?
- In that case all the student records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency.
- This is Updation anomaly.

Deletion Anomaly

- In our **Student** table, two different information's are kept together, Student information and Branch information.
- Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information.
- This is Deletion anomaly.

Normalization rules

- Normalization rules are divided into the following normal forms:
  - First Normal Form
  - Second Normal Form
  - Third Normal Form

First Normal Form

For a table to be in the First Normal Form, it should follow the following 4 rules:

- It should only have single(atomic) valued attributes/columns.
- Values stored in a column should be of the same domain
- All the columns in a table should have unique names.
- And the order in which data is stored, does not matter.

roll_no	name	subject
101	Akon	OS, CN
103	Ckon	Java
102	Bkon	C, C++

Fig: Table which is not in first normal form

roll_no	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C++

Fig: Table in First normal form

Second Normal Form

For a table to be in the Second Normal Form, it must satisfy two conditions:

- The table should be in the First Normal Form.
- There should be no Partial Dependency.

What is Dependency?

- In this table, student_id is the primary key and will be unique for every row, hence we can use student_id to fetch any row of data from this table
- Even for a case, where student names are same, if we know the student_id we can easily fetch the correct record.

student_id	name	reg_no	branch	address

- In the above table all non prime attribute depends on prime attribute.
- This is **Dependency** and we also call it **Functional Dependency**.

student_id	name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat

- Let's create another table for **Subject**, which will have subject_id and subject_name fields and subject_id will be the primary key.
- Now we have a **Student** table with student information and another table **Subject** for storing subject information.

subject_id	subject_name
1	Java
2	C++
3	Php

- Let's create another table **Score**, to store the **marks** obtained by students in the respective subjects.
- We will also be saving **name of the teacher** who teaches that subject along with marks.

score_id	student_id	subject_id	marks	teacher
1	10	1	70	Java Teacher
2	10	2	75	C++ Teacher
3	11	1	80	Java Teacher

- In the score table we are saving the **student_id** to know which student's marks are these and **subject_id** to know for which subject the marks are for.
- Together, student_id + subject_id forms a **Candidate Key** for this table, which can be the **Primary key**.
- See, if I ask you to get me marks of student with student_id 10, can you get it from this table?
- No, because you don't know for which subject.
- And if I give you subject_id, you would not know for which student.
- Hence we need student_id + subject_id to uniquely identify any row.

But where is Partial Dependency?

- Now if you look at the **Score** table, we have a column names teacher which is only dependent on the subject, for Java it's Java Teacher and for C++ it's C++ Teacher & so on.
- Now as we just discussed that the primary key for this table is a composition of two columns which is student_id & subject_id but the teacher's name only depends on subject, hence the subject_id, and has nothing to do with student_id.
- This is **Partial Dependency**, where an attribute in a table depends on only a part of the primary key and not on the whole key.

How to remove Partial Dependency?

- The simplest solution is to remove columns teacher from Score table and add it to the Subject table. Hence, the Subject table will become:

subject_id	subject_name	teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

- And our Score table is now in the second normal form, with no partial dependency.

score_id	student_id	subject_id	marks
1	10	1	70
2	10	2	75
3	11	1	80

3rd Normal Form

- A table is said to be in the Third Normal Form when,
  - It is in the Second Normal form.
  - And, it doesn't have Transitive Dependency.

Database Administrator (DBA)

- A Database Administrator (DBA) in Database Management System (DBMS) is an IT professional who works on creating, maintaining, querying, and tuning the database of the organization.
- They are also responsible for maintaining data security and integrity.
- This role requires the professionals to have good knowledge and experience in the particular RDBMS that the company uses.

Roles and responsibilities of DBA

1. Selection of hardware and software

- Keep up with current technological trends
- Predict future changes

2. Managing data security and privacy

- Protection of data against accidental or intentional loss, destruction, or misuse

- Firewalls
- Establishment of user privileges
- Complicated by use of distributed systems such as internet access and client/ server technology.

3. Managing Data Integrity
   - Integrity controls protects data from unauthorized use
   - Data consistency
   - Maintaining data relationship
   - Domains- sets allowable values
   - Assertions- enforce database conditions
4. Data backup
   - We must assume that a database will eventually fail
   - Establishment procedures
     - how often should the data be back-up?
     - what data should be backed up more frequently?
     - Who is responsible for the back ups?
5. Tuning database performance
   - Set installation parameters/ upgrade DBMS
   - Monitor memory and CPU usage

SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database