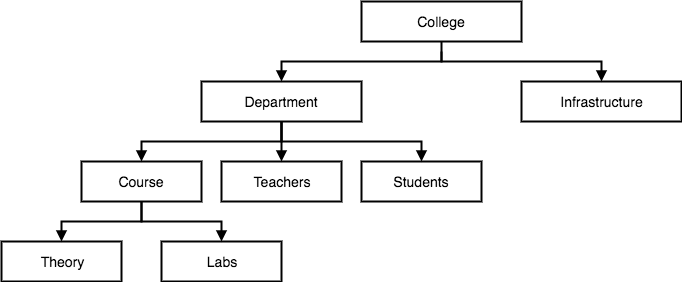
**Chapter 2 Database Models**

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:

1. Hierarchical Model
2. Network Model
3. [Entity-relationship Model](https://www.studytonight.com/dbms/er-model-concepts.php)
4. [Relational Model](https://www.studytonight.com/dbms/rdbms-concept.php)

**Hierarchical Model**

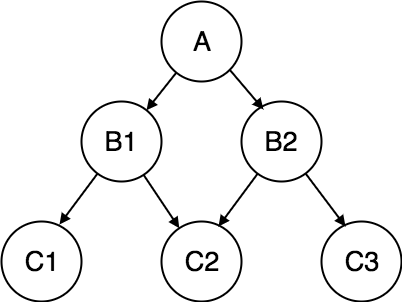
* This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.
* In this model, a child node will only have a single parent node.
* This model efficiently describes many real-world relationships like index of a book, recipes etc.
* In hierarchical model, data is organized into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.



**Advantages:**  
  
• Simple based on Hierarchal structure, the relationships between the layers (parents or child nodes).  
• Data Security: the first database model that offered the data security that is provided by the DBMS.  
• Efficiency: It is very efficient because when the database contains a large number of 1:n relationship and when the user require large number of transaction.  
• It’s very fast to access data at the top  
  
**Disadvantages:**  
  
• Implementation complexity: While it is simple and easy to design, it is quite difficult to implement.  
• Database Management Problem: If you make any changes in the database structure, then you need to make changes in the entire application program that access the database.  
• Lack of Structural Independence: there is lack of structural independence because when we change the structure then it becomes compulsory to change the application too.  
• Operational Anomalies: This model has irregularities from the insert, delete and update, also retrieval operation can be difficult.  
• Duplicate data.  
• Data can be very slow when information on the lower entities.

**Network Model**

* This is an extension of the Hierarchical model. In this model data is organised 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.



**ADVANTAGES OF NETWORK MODEL-**

The major advantage of network model are-

**1.) Conceptual simplicity-**Just like the hierarchical model, the network model is also conceptually simple and easy to design.

**2.) Capability to handle more relationship types-**The network model can handle the one to many and many to many relationships which is real help in modeling the real life situations.

**3.) Ease of data access-**The data access is easier and flexible than the hierarchical model.

**4.) Data integrity-**The network model does not allow a member to exist without an owner.

**5.) Data independence-**The network model is better than the hierarchical model in isolating the programs from the complex physical storage details.

**DIS-ADVANTAGE OF NETWORK MODEL-**

**1.) System complexity-**All the records are maintained using pointers and hence the whole database structure becomes very complex.

**2.) Operational Anomalies-** The insertion, deletion and updating operations of any record require large number of pointers adjustments.

## Entity-Relationship Model

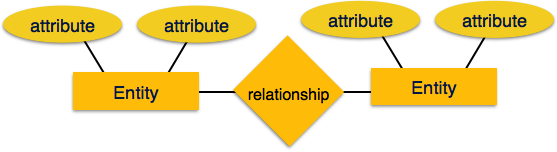
Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database.

ER Model is based on −

* **Entities** and their *attributes.*
* **Relationships** among entities.

These concepts are explained below.



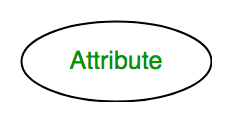
* **Entity** − An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.

Types of Entities

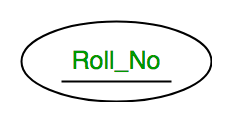
**Strong Entity:**   
A strong entity is not dependent on any other entity in the schema. A strong entity will always have a primary key. [Strong entities](https://practice.geeksforgeeks.org/problems/what-is-the-difference-between-strong-and-weak-entity) are represented by a single rectangle. The relationship of two strong entities is represented by a single diamond.   
Various strong entities, when combined together, create a strong entity set.

[**Weak Entity**](https://practice.geeksforgeeks.org/problems/explain-weak-entity-types)**:**   
A weak entity is dependent on a strong entity to ensure its existence. Unlike a strong entity, a weak entity does not have any primary key. It instead has a partial discriminator key. A weak entity is represented by a double rectangle.   
The relation between one strong and one weak entity is represented by a double diamond.

**Types of Attributes**  
Attributes are the **properties which define the entity type**. For example, Roll\_No, Name, DOB, Age, Address, Mobile\_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.



**1. Key Attribute –**   
The attribute which **uniquely identifies each entity** in the entity set is called key attribute.For example, Roll\_No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.

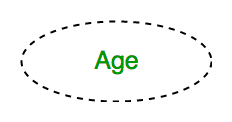


**2. Composite Attribute –**   
An attribute **composed of many other attribute** is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.

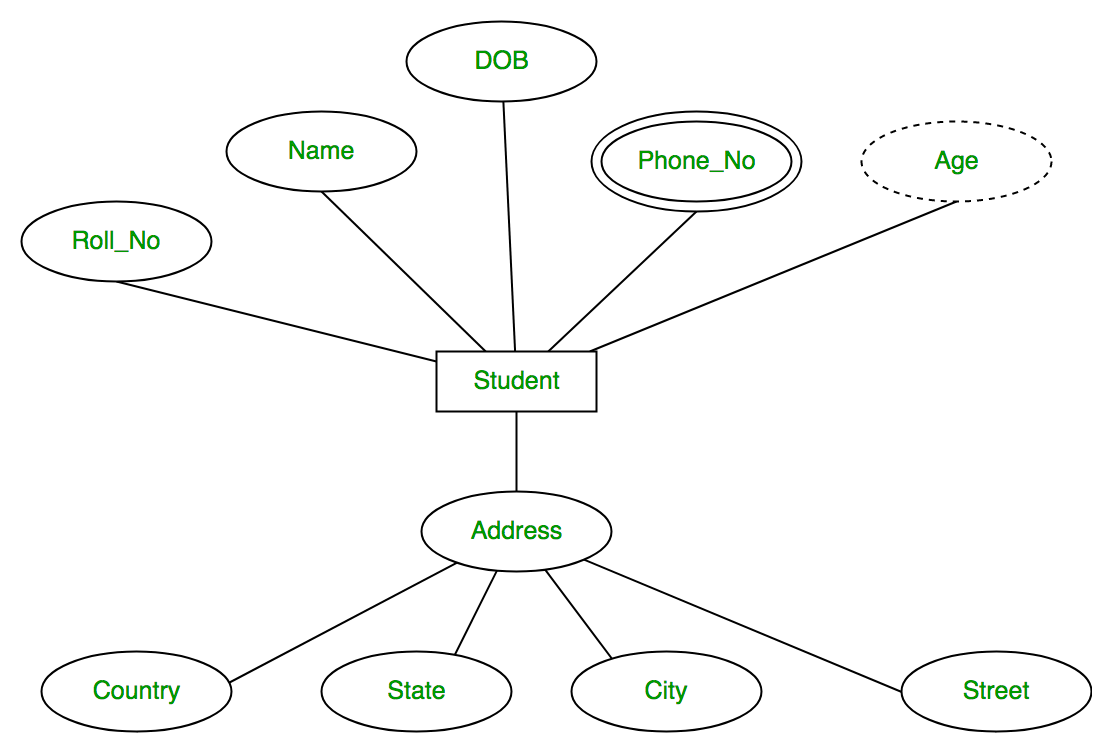
**3.Multivalued Attribute –**   
An attribute consisting **more than one value** for a given entity. For example, Phone\_No (can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.



**4. Derived Attribute –**   
An attribute which can be **derived from other attributes** of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, derived attribute is represented by dashed oval.



* The complete entity type**Student** with its attributes can be represented as:



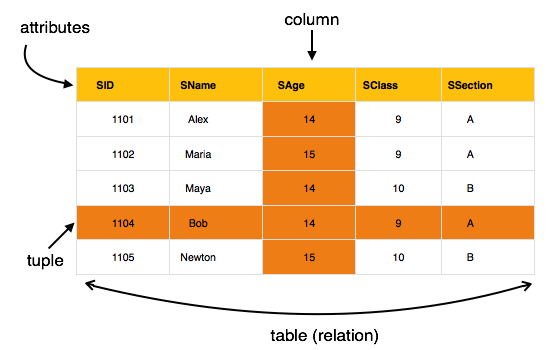
* **Relationship** − The logical association among entities is called ***relationship***. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.

Mapping cardinalities −

* + one to one
  + one to many
  + many to one
  + many to many

## Relational Model

The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model is based on first-order predicate logic and defines a table as an **n-ary relation**.



The main highlights of this model are −

* Data is stored in tables called **relations**.
* Relations can be normalized.
* In normalized relations, values saved are atomic values.
* Each row in a relation contains a unique value.
* Each column in a relation contains values from a same domain.

**Database Language**

Database languages, also known as query languages or data query languages, are a classification of programming languages that developers use to define and access databases, which are collections of organized data that users can access electronically. These languages allow users to complete tasks such as controlling access to data, defining and updating data and searching for information within the database management system (DBMS). A DBMS is a piece of technology that interacts with users, applications and the database to record and analyze data while also manipulating the database to offer a way to store, access and retrieve data.

1. Data definition language (DDL)

Data definition language (DDL) creates the framework of the database by specifying the database schema, which is the structure that represents the organization of data. Its common uses include the creation and alteration of tables, files, indexes and columns within the database. This language also allows users to rename or drop the existing database or its components. Here's a list of DDL statements:

CREATE: Creates a new database or object, such as a table, index or column

ALTER: Changes the structure of the database or object

DROP: Deletes the database or existing objects

RENAME: Renames the database or existing objects

2. Data manipulation language (DML)

Data manipulation language (DML) provides operations that handle user requests, offering a way to access and manipulate the data that users store within a database. Its common functions include inserting, updating and retrieving data from the database. Here's a list of DML statements:

INSERT: Adds new data to the existing database table

UPDATE: Changes or updates values in the table

DELETE: Removes records or rows from the table

SELECT: Retrieves data from the table or multiple tables

3. Data control language (DCL)

Data control language (DCL) controls access to the data that users store within a database. Essentially, this language controls the rights and permissions of the database system. It allows users to grant or revoke privileges to the database. Here's a list of DCL statements:

GRANT: Gives a user access to the database

REVOKE: Removes a user's access to the database

**ER To Relational model**

**Step 1: Strong Entities**

* + Create a table that includes all the simple attributes.
  + Include only the simple composed attributes.
  + Choose a primary key

**Step 2: Weak Entiies**

* + Create a table that includes all the simple attributes.
  + Include the primary key from the owner entity this will become a foreign key

**Step 3: 1:1**

* + Copy the primary key from the first table into the second table, it becomes a foreign key in the second table.
  + Any attributes tied to the relationship goes to the table with the foreign key.

**Step 4: 1:N**

* + Choose the entity on the n side and include a copy of the primary key from the one side, it becomes a foreign key in the n table.
  + Any attributes tied to the relationship goes to the table with the foreign key.

**Step 5 N:N**

* + Create a new table and include the primary key from the two participating entities.
  + Both of these fields become a foreign keys in the new table.
  + Any attributes tied to the relationship goes to the table with the foreign key.

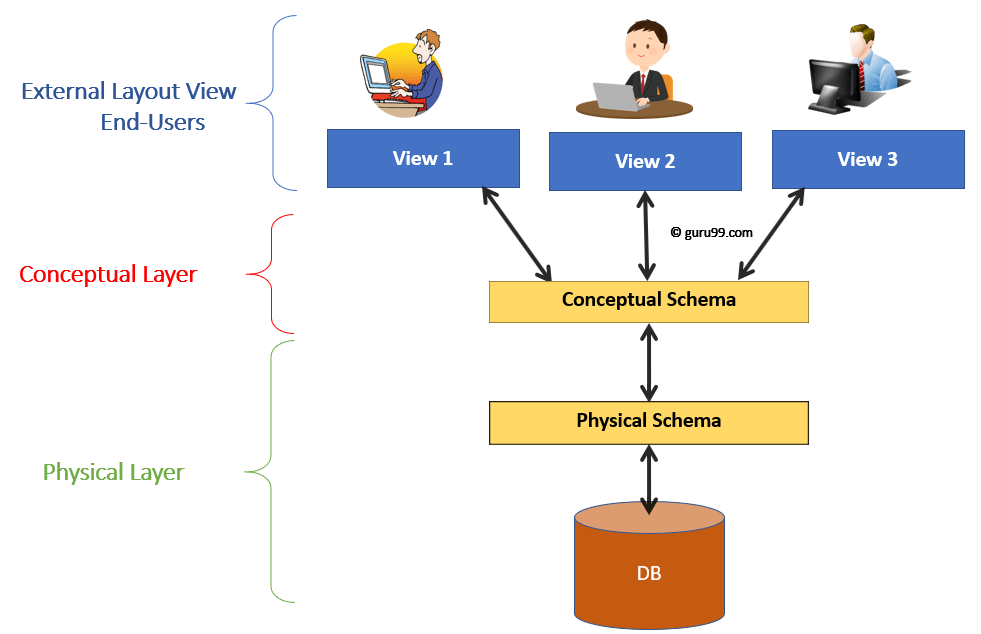
**Step 6: Multivalued Attribute**

* + For each multivalued attributed (A)
  + Create a new table and move the attribute (A) into the table.
  + Include the primary key from the entity that originally had (A) as an attribute.
  + The combination of those attributes will be the primary key of the new table.

**Levels of Database**

Before we learn Data Independence, a refresher on Database Levels is important. The database has 3 levels as shown in the diagram below

1. Physical/Internal
2. Conceptual
3. External

Levels of DBMS Architecture Diagram

## Data Independence

Data Independence is defined as a property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level. Data independence helps you to keep data separated from all programs that make use of it.

## Physical Data Independence

Physical data independence helps you to separate conceptual levels from the internal/physical levels. It allows you to provide a logical description of the database without the need to specify physical structures. Compared to Logical Independence, it is easy to achieve physical data independence.

With Physical independence, you can easily change the physical storage structures or devices with an effect on the conceptual schema. Any change done would be absorbed by the mapping between the conceptual and internal levels. Physical data independence is achieved by the presence of the internal level of the database and then the transformation from the conceptual level of the database to the internal level.

### Examples of changes under Physical Data Independence

Due to Physical independence, any of the below change will not affect the conceptual layer.

* Using a new storage device like Hard Drive or Magnetic Tapes
* Modifying the file organization technique in the Database
* Switching to different data structures.
* Changing the access method.
* Modifying indexes.
* Changes to compression techniques or hashing algorithms.
* Change of Location of Database from say C drive to D Drive

## Logical Data Independence

Logical Data Independence is the ability to change the conceptual scheme without changing

1. External views
2. External API or programs

Any change made will be absorbed by the mapping between external and conceptual levels.

When compared to Physical Data independence, it is challenging to achieve logical data independence.

### Examples of changes under Logical Data Independence

Due to Logical independence, any of the below change will not affect the external layer.

1. Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs
2. Merging two records into one
3. Breaking an existing record into two or more records

**DBA Roles and Responsibilities**

A Database Administrator (DBA) plays a crucial role in managing and maintaining an organization's databases. Their responsibilities include ensuring the availability, security, and performance of databases. Here are the typical roles and responsibilities of a DBA:

**Database Design:**

Collaborate with developers and system architects to design and implement efficient and scalable database structures.

Define data integrity constraints, data models, and relationships between tables.

**Installation and Configuration:**

Install and configure database management systems (DBMS) software.

Set up and configure database instances based on organizational needs.

**Performance Monitoring and Tuning:**

Monitor database performance regularly and identify opportunities for optimization.

Tune database parameters to ensure optimal performance.

**Backup and Recovery:**

Implement and maintain database backup and recovery procedures.

Regularly test backup and recovery processes to ensure data integrity and availability.

**Security Management:**

Implement and enforce security measures to protect the database from unauthorized access and data breaches.

Manage user access and permissions, including role assignments.

**Database Upgrades and Patching:**

Plan and execute database software upgrades.

Apply patches and updates to keep the database software secure and up to date.

**Capacity Planning:**

Forecast future database growth and plan for additional storage, memory, and processing power as needed.

Optimize resource utilization to meet current and future demands.

**Monitoring and Alerting:**

Set up monitoring tools to track database performance, availability, and security.

Respond to alerts promptly and address any issues that arise.

**Data Migration:**

Plan and execute data migration tasks, ensuring minimal downtime and data loss.

Validate data integrity after migration.

**Documentation:**

Maintain documentation for database configurations, processes, and procedures.

Keep documentation up to date with any changes made to the database environment.

**Collaboration and Communication:**

Collaborate with developers, system administrators, and other IT professionals to address database-related issues.

Communicate effectively with stakeholders about database performance, security, and related matters.

**Disaster Recovery Planning:**

Develop and implement disaster recovery plans to minimize data loss and downtime in case of emergencies.

Test and update disaster recovery plans regularly.

**Automation:**

Implement automation scripts for routine database tasks to improve efficiency and reduce manual errors.

**Troubleshooting:**

Identify and resolve database-related issues promptly.

Perform root cause analysis for recurring problems and implement preventive measures.

**Stay Informed:**

Stay updated on industry trends, new technologies, and best practices in database administration.

These responsibilities may vary slightly depending on the organization's size, industry, and specific database management system in use (e.g., Oracle, Microsoft SQL Server, MySQL, PostgreSQL). DBAs often need a combination of technical skills, problem-solving abilities, and effective communication to excel in their roles.