Jisy raju
Assistant Professor, CE Cherthala

Module 1

Introduction: Data: structured, semi-structured and unstructured data, Concept & Overview of DBMS, Data Models, Database Languages, Database Administrator, Database Users, Three Schema architecture of DBMS. Database architectures and classification. Entity-Relationship Model:Basic concepts, Design Issues, Mapping Constraints, Keys, Entity-Relationship Diagram, Weak Entity Sets, Relationships of degree greater than 2

1.1 Data

The information collected together for reference or analysis. Big Data can be divided into following three categories.

1. Structured Data
2. Unstructured Data
3. Semi-structured Data

Structured Data

The data that has a structure and is well organized either in the form of tables or in some other way and can be easily operated is known as structured data. Searching and accessing information from such type of data is very easy.
For example, data stored in the relational database, the spreadsheet is an another good example of structured data.

Unstructured Data

The data that is unstructured or unorganized Operating such type of data becomes difficult and requires advance tools and softwares to access information.
For Example, images and graphics, pdf files, word document, audio, video, emails, powerpoint presentations, webpages and web contents, wikis, streaming data, location coordinates etc.

Semi-structured Data

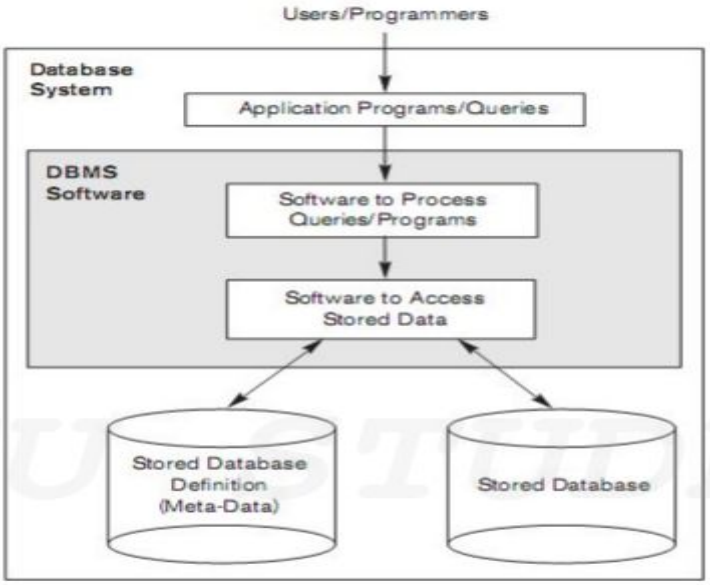
Semi-structured data is basically a structured data that is unorganised. Web data such JSON(JavaScript Object Notation) files, BibTex files, .csv files, tab-delimited text files, XML and other markup languages are the examples of Semi-structured data found on the web.
Due to unorganized information, the semi-structured is difficult to retrieve, analyze and store as compared to structured data. It requires software framework like Apache Hadoop to perform all this.

1.2 Concept & Overview of DBMS

DBMS stands for Database Management System. We can break it like this DBMS = Database + Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can define DBMS like this: DBMS is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner. The DBMS is a general-purpose software system that facilitates the processes of **defining, constructing, manipulating, and sharing.**

Defining a DB includes specifying data types, structure, constraints, metadata etc. Metadata is data that describes other data. **Constructing** the database is the process of storing the data on some storage medium that is controlled by the DBMS. **Manipulating** a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data. **Sharing** a database allows multiple users and programs to access the database simultaneously. An application program accesses the database by sending queries to the DBMS.

Database System Environment



1.2.1 An Example

Let us consider a simple example that most readers may be familiar with: a UNIVERSITY database for maintaining information concerning students, courses, and grades in a university environment. Database contain three relations.

Relation Name	No. of columns
STUDENTS	4
COURSES	3
GRADES	3

The STUDENT file stores data on each student like Roll no., Name, Course, Address.

STUDENTS

Roll no	Name	Course	Address
1	Jis	CS	ABC
2	vish	CS	XYZ

1.2.2 Characteristics of the Database Approach

In the database approach, a single repository maintains data that is defined once and then accessed by various users. In file systems, each application is free to name data elements independently.. The main characteristics of the database approach versus the file-processing approach are the following:

- Self-describing nature of a database system
- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing

Self-describing nature of a database system

A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints. This definition is stored in the DBMS catalog, the information stored in the catalog is called meta-data.

Insulation between programs and data, and data abstraction

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called **data abstraction.**

The main purpose of data abstraction is achieving data independence in order to save time and cost required when the database is modified or altered. The structure of data files is stored in the DBMS catalog separately from the access programs. We call this property **program-data independence**. An operation (also called a function or method) is specified in two parts.

- Interface
  - ✓ The interface (or signature) of an operation includes the operation name and the data types of its arguments (or parameters).
- Implementation
  - ✓ The implementation (or method) of the operation is specified separately and can be changed without affecting the interface.

User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented. This is termed as **program-operation independence.**

The characteristic that allows program-data independence and program operation independence is called data abstraction. A data model is a type of data abstraction that is used to provide conceptual representation.

Support of multiple views of the data

A database has many users, each user may require a different perspective or view of the database. A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.

Sharing of Data and Multiuser Transaction Processing

A multiuser DBMS, must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently.

The DBMS must enforce several transaction properties.

1. Isolation property

■ The isolation property ensures that each transaction appears to execute in isolation from other transactions, even though hundreds of transactions may be executing concurrently.

2. Atomicity property

■ The atomicity property ensures that either all the database operations in a transaction are executed or none are.

1.2.3 Actors on the Scene

The people whose jobs involve the day-to-day use of a large database are called as the actors on the scene .

Database Administrators: In an organization where many people use the same resources, there is a need of chief administrator to oversee and manage these resources. In database environment, the primary resource is the database itself and the secondary resource is the DBMS and related software. The DBA is responsible for authorizing acess to the database, coordinating and monitoring its use and acquiring software and hardware resources as needed.

Database Designers: These are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meet these requirements.

End Users: These are the people whose jobs require access to the database for querying, updating, and generating reports, the database primarily exist for their use. Various categories of end users are:

- Casual End users: Occasionaly access the database, but they may need different information each time.
- Parametric Or Naive Users**:** These are the users who communicate with the database for a regular period. Their main job is to constantly querying and updating the database using standard queries , this is called Canned Transaction.
- Sophisticated End Users: Include engineers, scientists, business analyst and others who thoroughly familarize themselves with the facilities of DBMS.
- Stand alone users: These are the users who maintain their personal database using ready made software which is available in the market easily and provide the menu based interface through which they can easily used the database.

System Analysts and Application Programmers: System analysts find out the requirements of parametric end users and develop all the analysis for canned transaction through which they can

meet all the requirements of parametric users. Application programmers implement these analysis as programs. These are especially Software engineers.

1.2.4 Workers behind the Scene

These persons are typically not interested in the database content itself. We call them the workers behind the scene, and they include the following categories:

- DBMS system designers and implementers design and implement the DBMS modules and interfaces as a software package.
- Tool developers design and implement tools—the software packages that facilitate database modeling and design, database system design, and improved performance.
- Operators and maintenance personnel (system administration personnel) are responsible for the actual running and maintenance of the hardware and software environment for the database system.

1.2.5 Advantages of Using the DBMS Approach

1. Controlling Redundancy

➢ Redundancy in storing the same data multiple times leads to several problems.

a) Duplication of effort

b) Storage space is wasted

c) Files that represent the same data may become inconsistent .

2. Restricting Unauthorized Access

A DBMS should provide a security and authorization subsystem , which the DBA uses to create accounts and to specify account restrictions.

3. Providing Persistent Storage for Program Objects

Databases can be used to provide persistent storage for program objects and data structures.

4. Providing Storage Structures and Search Techniques for Efficient Query Processing

Database systems must provide capabilities for efficiently executing queries and updates. The database is typically stored on disk, the DBMS must provide specialized data structures and search techniques to speed up .

5. Providing Backup and Recovery

A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing.

6. Providing Multiple User Interfaces

Users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces. These include query languages for casual users, programming language interfaces for application programmers, forms and command codes for parametric users, and menu-driven interfaces and natural language interfaces for standalone users.

7. Enforcing Integrity Constraints

Integrity constraints are use to ensure accuracy and consistency of data in DB.

8. Representing Complex Relationships among Data

A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently.

9. Permitting Inferencing and Actions Using Rules

Some database systems provide capabilities for defining deduction rules for inferencing new information from the stored database facts. Such systems are called deductive database systems .

10. Additional Implications of Using the Database Approach

a) Potential for Enforcing Standards.
b) Reduced Application Development Time
c) Flexibility.
d) Availability of Up-to-Date Information
e) Economies of Scale.

1.2.6 DATABASE SYSTEM APPLICATIONS

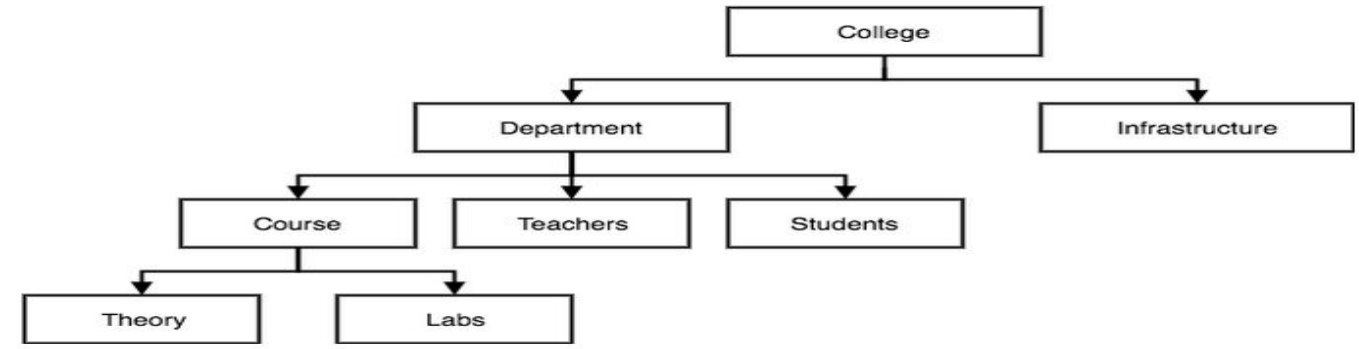
- Banking: for storing customer information, accounts, loans and banking transactions.
- Airlines: For reservations and schedule information.
- Universities: For student information, course registration and grade.
- Credit card transactions: For purchase on credit card and generation of monthly statements.
- Tele communications: For keeping records for calls made, generating monthly bills, maintaining balances on prepaid calling cards etc.
- Finance: For storing information about holdings, sales and purchase of financial instruments such as stokes, bonds and for storing real time market data.

1.3 DATA MODELS, SCHEMAS AND INSTANCE

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

Hierarchical model: The hierarchical model organizes data into a tree-like structure, where each record has a single parent or root. In hierarchical model, data is organised into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses.



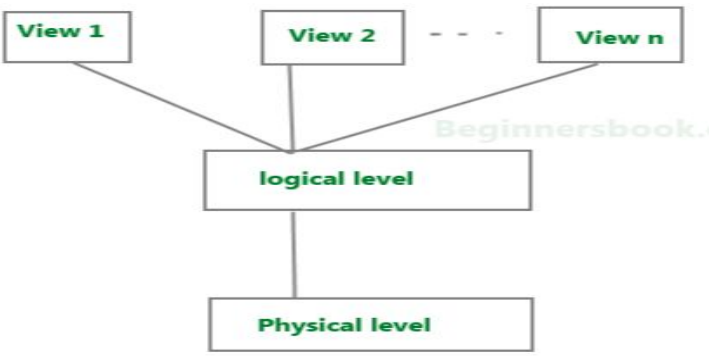
Network Model: This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node. The network model builds on the hierarchical model by allowing many-to-many relationships between linked records.

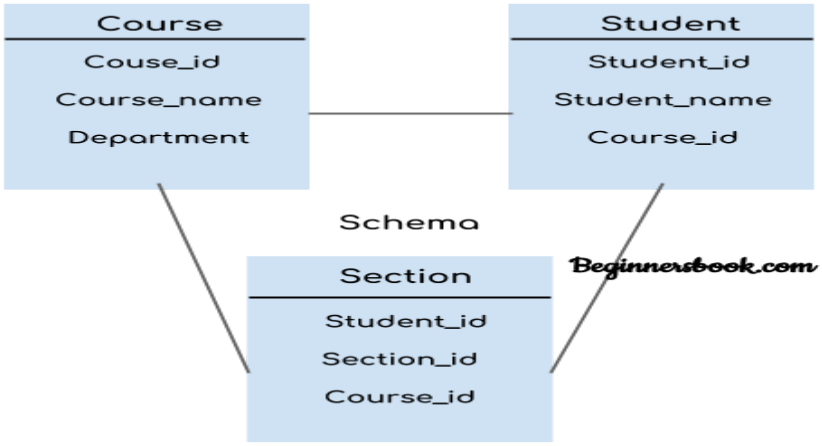
Entity-relationship Model: In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.Different entities are related using relationships. E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand. Let's take an example, If we have to design a School Database, then **Student** will be an **entity (real world object)** with **attributes** (properties of entity) name, age, address etc.

Relational model: The most common model, the relational model sorts data into tables, also known as relations, each of which consists of columns and rows. Each column lists an attribute of the entity.

DATA SCHEMA: Design of a database is called the schema. Schema is of three types

- **Physical schema**:The design of a database at physical level is called physical schema. Physical level is the lowest level. It describes how data is actually stored in database.
- **Logical schema**:Design of database at logical level is called logical schema, programmers and database administrators work at this level. Logical level is the middle level. It describes what data is stored in database.
- **View schema:**Design of database at view level is called view schema. View level is the highest level. This level describes the user interaction with database system.





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

Example: Let's say we are storing customer information in a customer table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.
At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.
At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

DBMS Instance: The data stored in database at a particular moment of time is called **instance** of database or **database state**. It is also called the current set of **occurrences** or **snapshot**.

For example, lets say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. Lets say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table.

Schema Vs Instance

Schema	Instance
Skeleton of database	Snapshot of database
Change occurrences in DB is rare	Change occurrences in DB is frequent
Initial state is empty	Initial state is empty

1.4 Database Languages and Interfaces

Database Languages are used to create and maintain database. There are large numbers of database languages like Oracle, MYSQL, MS Access, dBase, FoxPro etc. SQL statements commonly used in Oracle and MS Access can be categorized as,

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

- To create the database instance -**Create**
- To alter the structure of database – **ALTER**
- To drop database instances - **Drop**
- To delete tables in a database instance – **TRUNCATE**
- To rename database instances – **RENAME**
- To drop objects from database such as tables – **DROP**
- To Comment – **Comment**

Data manipulation language (DML): DML is used for accessing and manipulating data in a database. The following operations on database comes under DML:

- To read records from table(s) –**Select**
- To insert record(s) into the table(s) – **INSERT**
- Update the data in table(s) – **Update**
- Delete all the records from the table –**Delete**

DML is divided into two,

- **Procedural DML** :Procedural DML is used to tell the system what data is needed and how to take the data. Procedural DML is embedded into a high-level programming language. Example of Data Manipulation Language Using Java:

```
ResultSet rs = st.executeQuery("SELECT * FROM students");
```

Resultset declare what data is needed, which are included in the line of the SQL query SELECT * FROM students. While the **while** line states the way to retrieve the data.

- **Non-procedural DML:** Non-procedural DML is used to declare what data is needed-not how the data is retrieved. Non procedural also called declarative programming. The examples of non-procedural DMLs are SQL and QBE (Query-By-Example) that are used by relational database systems.

Data control language (DCL) : DCL is used for granting and revoking user access on a database

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

Transaction Control Language(TCL): The changes in the database that we made using DML commands are either performed or rollbacked using TCL.

- To persist the changes made by DML commands in database – **COMMIT**
- To rollback the changes made to the database – **ROLLBACK**
- To identifies a point in a transaction to which you can later roll back-**SAVEPOINT**

DBMS Interfaces: User-friendly interfaces provided by a DBMS may include the following:

1. Menu-Based Interfaces for Web Clients or Browsing.

2. Forms-Based Interfaces.

3. Graphical User Interfaces (GUI)

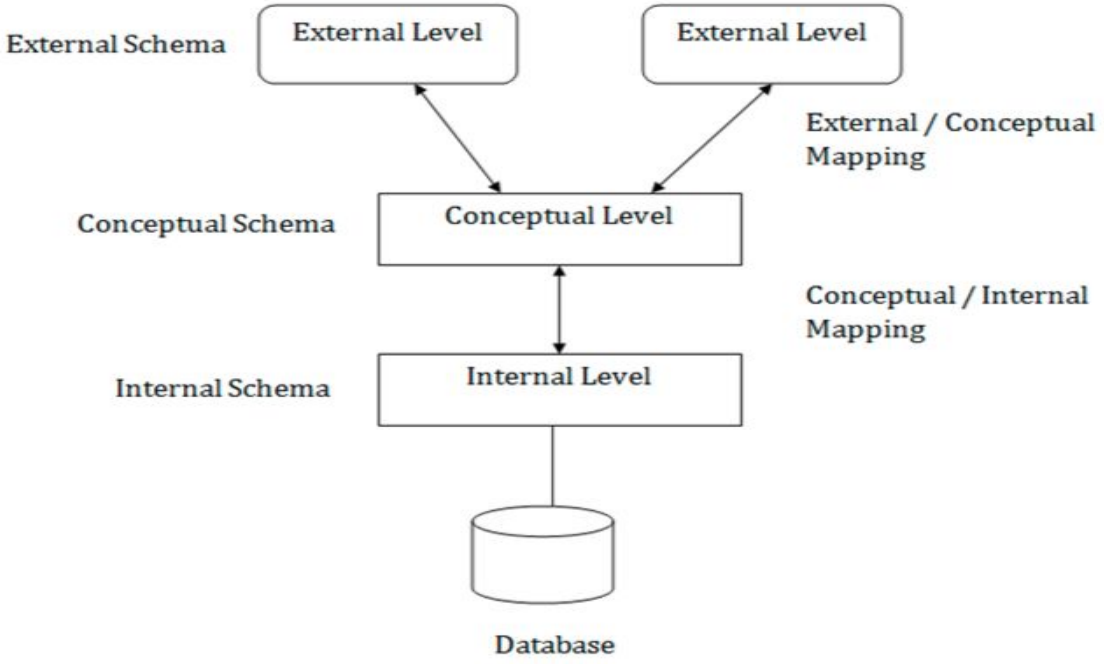
4. Natural Language Interfaces

5. Speech Input and Output

6. Interfaces for Parametric Users.

1.5 Three Schema architecture and Data Independence

The goal of the three-schema architecture is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:



Mapping is used to transform the request and response between various database levels of architecture. Mapping is not good for small DBMS because it takes more time.

- In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
- In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

Internal Level / Physical level

- The internal level has an internal schema which describes the physical storage structure of the database.
- The internal schema is also known as a physical schema.
- It uses the physical data model. It is used to define that how the data will be stored in a block.

- At lowest level, it is stored in the form of bits with the physical addresses on the secondary storage device. At highest level, it can be viewed in the form of files.

Conceptual Level /Logical Level

- The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
- The conceptual schema describes the structure of the whole database.
- The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
- In the conceptual level, internal details such as an implementation of the data structure are hidden.
- Programmers and database administrators work at this level.

External Level /View Level

- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- An external schema is also known as view schema.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The view schema describes the end user interaction with database systems.

1.5.1 Data Independence

Data independence can be explained using the three-schema architecture. Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

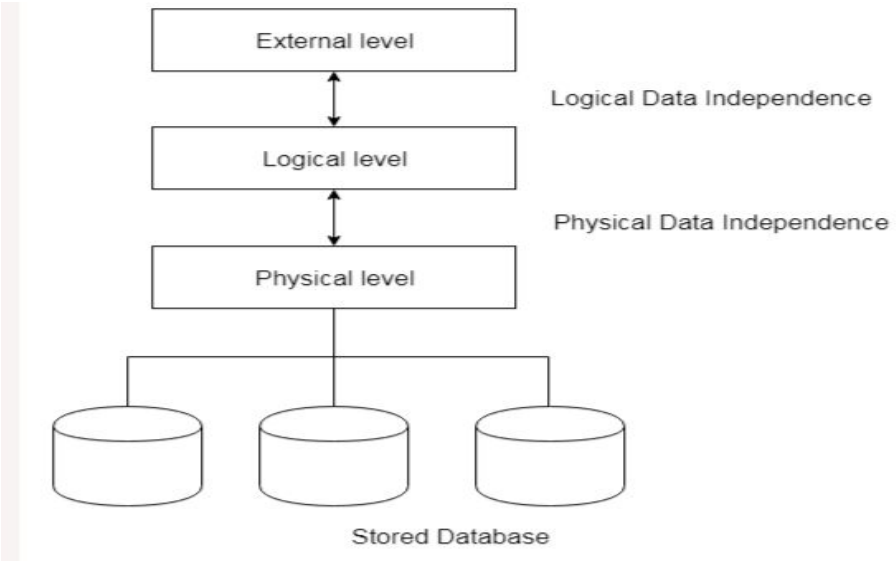
There are two types of data independence:

Physical Data Independence

- Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
- If we do any changes in the storage size of the database system server, then the Conceptual structure of the database will not be affected.
- Physical data independence is used to separate conceptual levels from the internal levels.
- Physical data independence occurs at the logical interface level.

Logical Data Independence

- Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
- Logical data independence is used to separate the external level from the conceptual view.
- If we do any changes in the conceptual view of the data, then the user view of the data would not be affected.
- Logical data independence occurs at the user interface level.



1.6 DBMS Architecture

A Database Management system is not always directly available for users and applications to access and store data in it. The design of a DBMS depends on its architecture. It can be **centralised**(all the data stored at one location), **decentralised**(multiple copies of database at different locations) or **hierarchical**, depending upon its architecture.

Database Architecture is logically of three types:

- **1-tier DBMS** architecture
- **2-tier DBMS** architecture
- **3-tier DBMS** architecture

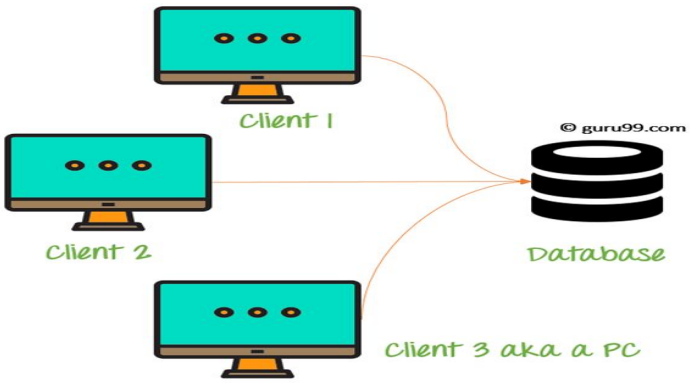
1-tier DBMS architecture is when the database is directly available to the user for using it to store data. It is the simplest of Database Architecture where the Client, Server, and Database all reside on the same machine. Anytime you install a DB in your system and access it to practise SQL queries it is 1 tier architecture. But such architecture is rarely used in production.

2-tier DBMS Architecture is like client server application.It includes,

- Presentation layer(client side)
- Database Server

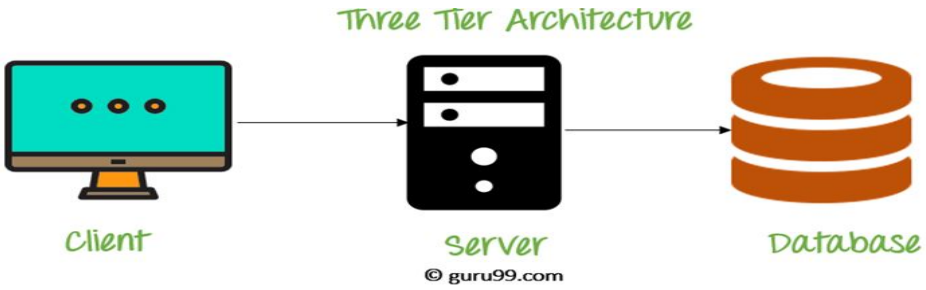
Application layer between the user and the DBMS, which is responsible to communicate the user's request to the database management system and then send the response from the DBMS to the

user. An application interface known as ODBC(Open Database Connectivity) provides an API that allow client side program to call the DBMS. Most DBMS vendors provide ODBC drivers for their DBMS.



3-tier DBMS architecture is an extension of the 2-tier architecture. 3-tier architecture has following layers

1. Presentation layer (client)
2. Application layer (server)
3. Database Server



1.7 Entity-Relationship Model

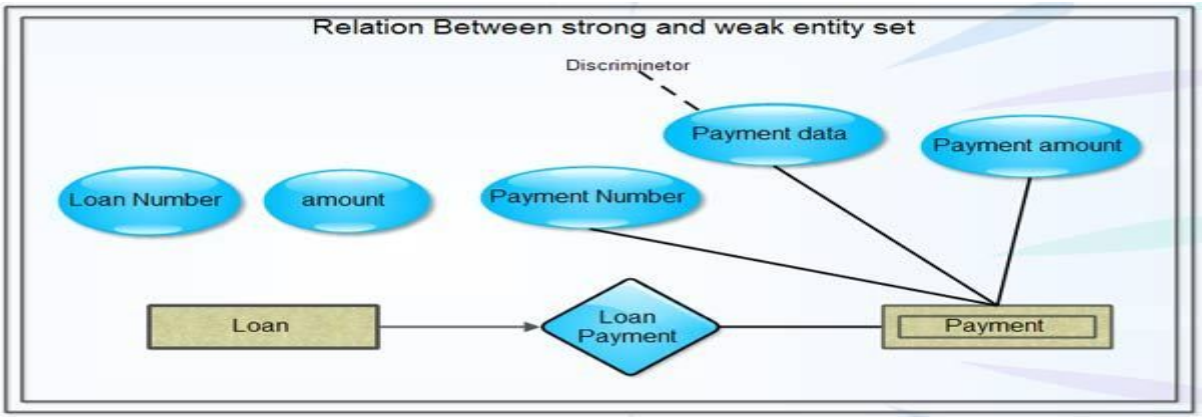
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity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

Entity: An entity can be a real-world object that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity. An entity set is a collection of similar types of entities. Represents in rectangle box.

Two type of entities

- **Strong entity:** An entity that can be uniquely identified by its own attribute.
- **Weak Entity:**An entity that cannot be uniquely identified by its own attributes is called weak entity. The *entity* set which does not have sufficient primary key is called as *Weak entity* set. The weak entity is represented by a double rectangle. For example – a bank account cannot

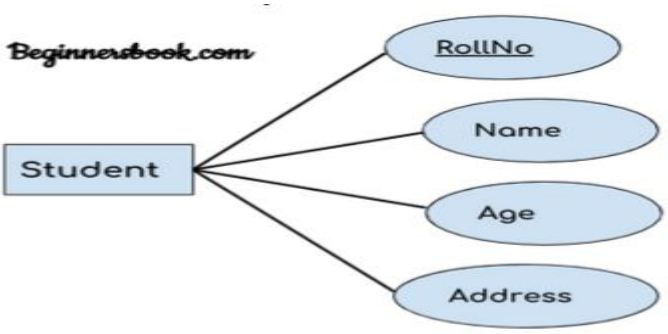
be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.



Strong vs Weak entity

Strong Entity	Weak Entity
The Strong entity has a primary key.	The weak entity has a partial key
Strong entity is denoted by a single rectangle.	Weak entity is denoted with the double rectangle.
The relation between two strong entities is denoted by a single diamond simply called relationship.	The relationship between a weak and a strong entity is denoted by Identifying Relationship denoted with double diamond.
Strong entity may or may not have total participation in the relationship.	Weak entity always has total participation in the identifying relationship shown by double line.

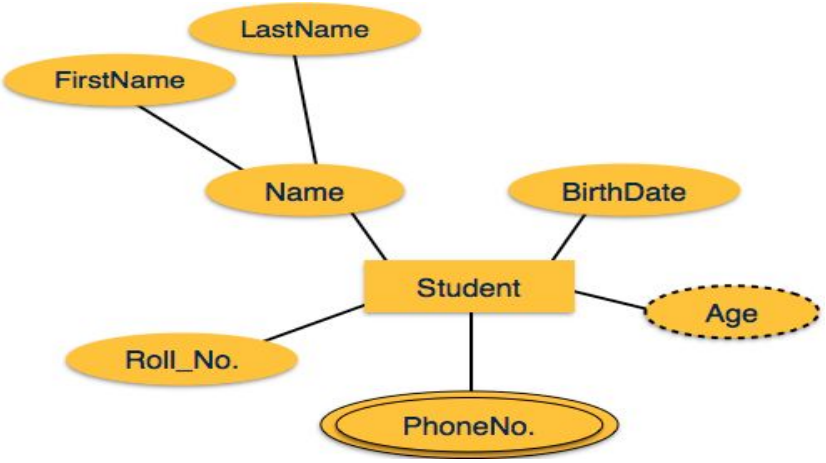
Attributes: Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes. There exists a **domain** or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc. Represents in oval shape.



Types of Attributes

- **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.

- **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
- **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived from stored attribute salary. For another example, age can be derived from data_of_birth. It is represented by **dashed oval**.
- **Single-value attribute** − Single-value attributes contain single value. For example − Social_Security_Number.
- **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc. t is represented with double ovals in an ER Diagram.
- **Complex Attributes:** Composite and multivalued attributes can be nested.



KEYS

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set. For example, the roll_number of a student makes him/her identifiable among students.

- **Super Key:** A set of attributes (one or more) that collectively identifies a .
- **Candidate Key:** A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key:** A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.
- **Foreign Key:** Key used to link two tables together. A FOREIGN KEY is a field in one table that refers to the PRIMARY KEY in another table.

Look at the following two tables:

"Persons" table:

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

"Orders" table:

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3

Example: Consider a Relation or Table R1. Let A,B,C,D,E are the attributes of this relation.

R(A,B,C,D,E)

A→BCDE This means the attribute 'A' uniquely determines the other attributes B,C,D,E.

BC→ADE This means the attributes 'BC' jointly determines all the other attributes A,D,E in the relation.

Primary Key :A

Candidate Keys :A, BC

Super Keys : A,BC,ABC,AD

RELATIONSHIP: The association among entities is called a relationship.
Relationship Set: A set of relationships of similar type is called a relationship set. Represents using Diamonds shape.

Degree of Relationship (also known as cardinality): The number of participating entities in a relationship defines the degree
of the relationship. Above figure degree is two.

- Binary relationship= degree 2
- Ternary relationship= degree 3
- Unary relationship = 1 degree

Two main types of (binary)relationship constraints:

- cardinality ratio
- participation

Mapping Cardinality /Cardinality ratio

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

- **One to Many Relationship (1: M)**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.

- **Many to One Relationship(M:1):**When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship. For example – many students can study in a single college but a student cannot study in many colleges at the same time.

- **Many to Many Relationship(M:N):**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.

Participation Constraints

Participation constraints define the l relationship instances in which an entity must participate.

- **Total Participation** − Each entity is involved in the relationship. It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.Total participation is represented by double lines.
- **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.

Example: Double line between the entity set “Student” and relationship set “Enrolled in” signifies total participation. It specifies that each student must be enrolled in at least one course. Single line between the entity set “Course” and relationship set “Enrolled in” signifies partial participation. It specifies that there might exist some courses for which no enrollments are made.

Relationship between Cardinality and Participation Constraints-

Minimum cardinality tells whether the participation is partial or total.

- If minimum cardinality = 0, then it signifies partial participation.
- If minimum cardinality = 1, then it signifies total participation.

Maximum cardinality tells the maximum number of entities that participates in a relationship set.

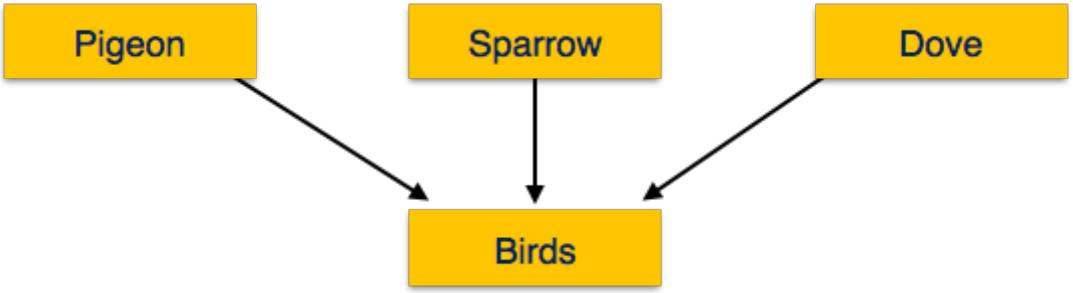
Role Name And Recursive Relationship

The role name signifies the role which helps to explain what the relationship means. A relationship has always been between occurrences in two different entities. However, it is possible for the same entity to participate in the relationship. This is termed a recursive relationship.
Let us suppose that we have an employee table. There are 10 employees in this employee table. One employee may be the supervisor of other 9 employees (subordinates). Supervisor and Subordinate are called Role Names.

Enhanced Entity Model (EER):

- Extension of ER model, for more complex DB.
- Includes all concepts of ER model
- Additionally concept of Specialization and Generalization

Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. Sub-classes are combined to form a super-class. For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.



Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. Top-down approach. Super class are splitted to sub-classes.