Advance Database Management System

Data :

- Data is a distinct piece of information that is gathered and translated for some purpose.
- Data is the known facts, figures and statistics, concepts or instruction in a formal manner, having no particular meaning. it's raw and unprocessed, which is suitable for understanding and processing. **Example -** 001, Ram, 28 etc.
- Computers represent data, including video, images, sounds and text, as binary values using patterns of just two numbers 1 and 0.

Information

- Data becomes information when it is processed, turning it into something meaningful.
- Information is the processed data on which decisions and actions are based.

  Example - "The age of Karan is 20"

Difference Between Data and Information

DATA	INFORMATION
1.Raw facts	1.Processed data
2. It is in unorganized form	2. It is in organized form
3. Data doesn't help in decision making process	3. Information helps in decision making process

Database

- The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently.
- It is also used to organize the data in the form of a table, schema, views, and reports, etc.
- Using the database, you can easily retrieve, insert, and delete the information.

Example: The college Database organizes the data about the students, faculty, admin, staff, etc.

Database Management System (DBMS)

A **database-management system (DBMS)** is a collection of interrelated data and a set of programs to access those data. A **DBMS** is a software that allows creation, definition and manipulation of database. DBMS is actually a tool used to perform any kind of operation on data in database. DBMS also provides protection and security to database. It maintains data consistency in case of multiple users. **Examples of popular DBMS -** MySql, Oracle, Sybase, Microsoft Access and IBM DB2 etc.

The collection of data, usually referred to as the **database**, contains information relevant to an enterprise. A **Database** is a collection of related data organized in a way that data can be easily accessed, managed and updated. Any piece of information can be a data. Database is actually a place where related piece of information is stored and various operations can be performed on it.

Goals of DBMS

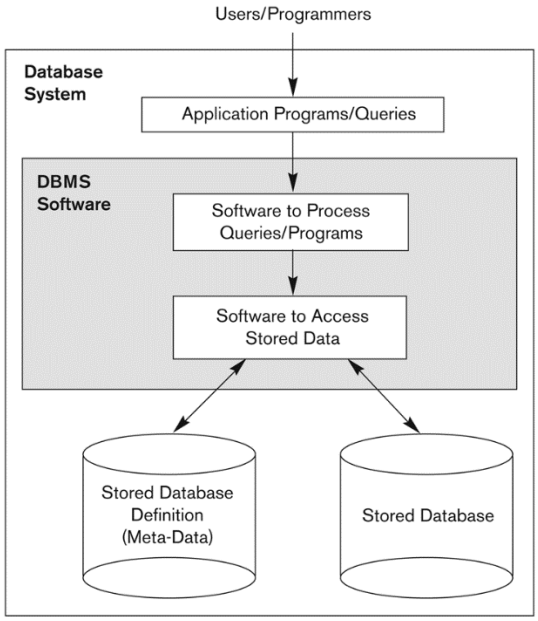
The primary goal of a DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient*.

- Database systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.
- The database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results.
- A DBMS is software that allows creation, definition and manipulation of database, allowing users to store, process and analyze data easily.
- DBMS provides us with an interface or a tool, to perform various operations like creating database, storing data in it, updating data, creating tables in the database and a lot more.
- DBMS also provides protection and security to the databases.
- It also maintains data consistency in case of multiple users.

Popular DBMS used these days:

1) Oracle
2) MySQL
3) MS SQL Server
4) MS Access
5) PostgreSQL
6) IBM DB2
7) MariaDB
8) MongoDB

Architecture of Database System : The database system can be divided into four components.



- **Users :** Users may be of various type such as DB administrator, System developer and End users.
- **Database Application :** Database application may be Personal, Departmental, Enterprise and Internal
- **DBMS :** Software that allow users to define, create and manages database access, Ex: MySql, Oracle etc.
- **Database :** Collection of logical data.

Database System Applications

Databases are widely used. Here are some representative applications:

- **Banking:** For customer information, accounts, and loans, and banking transactions.
- **Airlines:** For reservations and schedule information.
- **Universities:** For student information, course registrations, and grades.
- **Credit card transactions:** For purchases on credit cards and generation of monthly statements.
- **Human resources:** For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.
- **Telecommunication:** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
- **Finance:** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
- **Sales:** For customer, product, and purchase information.
- **Manufacturing:** For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.

Purpose of Database Systems

The main purpose of database systems is to manage the data. Consider a university that keeps the data of students, teachers, courses, books etc. To manage this data we need to store this data somewhere where we can add new data, delete unused data, update outdated data, retrieve data, to perform these operations on data we need a Database management system that allows us to store the data in such a way so that all these operations can be performed on the data efficiently.

Characteristics of DBMS

- **Data stored into Tables:** Data is never directly stored into the database. Data is stored into tables, created inside the database.
- **Reduced Redundancy:** In the modern world hard drives are very cheap, but earlier when hard drives were too expensive, unnecessary repetition of data in database was a big problem. But DBMS follows Normalization which divides the data in such a way that repetition is minimum.
- **Data Consistency:** On Live data, i.e. data that is being continuously updated and added, maintaining the consistency of data can become a challenge. But DBMS handles it all by itself.
- **Support Multiple user and Concurrent Access:** DBMS allows multiple users to work on it (update, insert, delete data) at the same time and still manages to maintain the data consistency.
- **Query Language:** DBMS provides users with a simple Query language, using which data can be easily fetched, inserted, deleted and updated in a database.

Advantages of DBMS

- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.
- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **Multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

Disadvantages of DBMS

- **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.
- **Size:** It occupies a large space of disks and large memory to run them efficiently.
- **Complexity:** Database system creates additional complexity and requirements.
- **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

Data Models

Underlying the structure of a database is the **data model**: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

There are two data models, the entity-relationship model and the relational model. Both provide a way to describe the design of a database at the logical level.

Entity-Relationship Model

The entity-relationship (E-R) data model is based on a perception of a real world that consists of a collection of basic objects, called *entities*, and of *relationships* among these objects.

Entity- An entity is a “thing” or “object” in the real world that is distinguishable from other objects. For example, each person is an entity, and bank accounts can be considered as entities. Entities are described in a database by a set of **attributes**.

Example - The attributes *account-number* and *balance* may describe one particular account in a bank, and they form attributes of the *account* entity set.

Similarly, attributes *customer-name*, *customer-street* address and *customer-city* may describe a *customer* entity. customer-id is used to uniquely identify customers.

A unique customer identifier must be assigned to each customer.

Relationship- A **relationship** is an association among several entities.

Example - A *depositor* relationship associates a customer with each account that he/she has.

The set of all entities of the same type and the set of all relationships of the same type are termed an **entity set** and **relationship set**, respectively.

An **entity set** is a set of entities of the same type that share the same properties, or attributes.

The set of all persons who are customers at a given bank, for example, can be defined as the entity set *customer*.

An entity is represented by a set of **attributes**. Attributes are descriptive properties possessed by each member of an entity set. Possible attributes of the *customer* entity set are *customer-id*, *customer-name*, *customer-street*, and *customer-city*. Each entity has a **value** for each of its attributes.

For each attribute, there is a set of permitted values, called the **domain**, or **value set**, of that attribute. The domain of attribute *customer-name* might be the set of all text strings of a certain length.

Entity-Relationship Diagram

The overall logical structure (schema) of a database can be expressed graphically by an *E-R diagram*, which is built up from the following components:

- **Rectangles**, which represent entity sets
- **Ellipses**, which represent attributes
- **Diamonds**, which represent relationships among entity sets
- **Lines**, which link attributes to entity sets and entity sets to relationship sets
- **Double ellipses**, which represent multivalued attributes
- **Dashed ellipses**, which denote derived attributes
- **Double lines**, which indicate total participation of an entity in a relationship set
- **Double rectangles**, which represent weak entity sets.

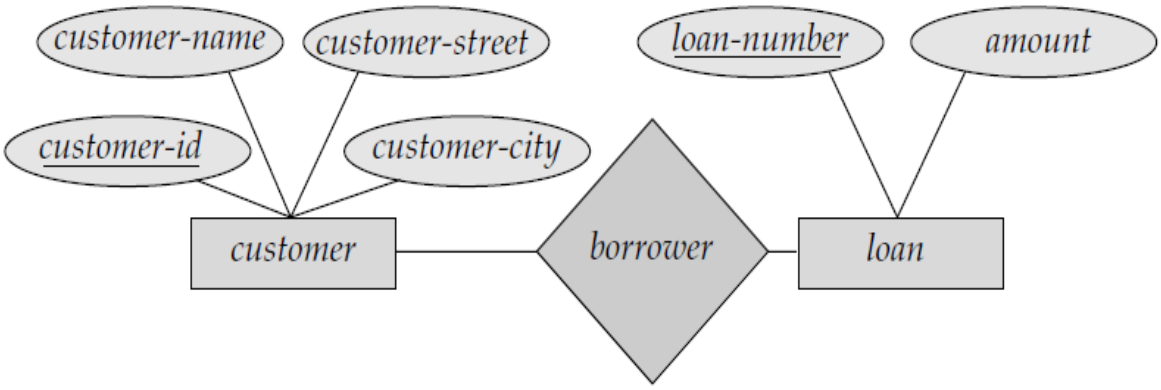


Figure - A sample E-R diagram

Attributes - Entity sets have associated *attributes,* which are properties of the entities in that set. For instance, the entity set *customer have the attributes customer-id, customer-name, customer-street, customer-city. In* E/R model, attributes are atomic values, such as strings, integers, or real's.

Types of Attributes –

Simple Attributes - The attributes have been simple; that is, they are not divided into subparts.

Composite Attributes - Composite attributes can be divided into subparts.

For example, an attribute *name* could be structured as a composite attribute consisting of *first-name*, *middle-initial*, and *last-name*.

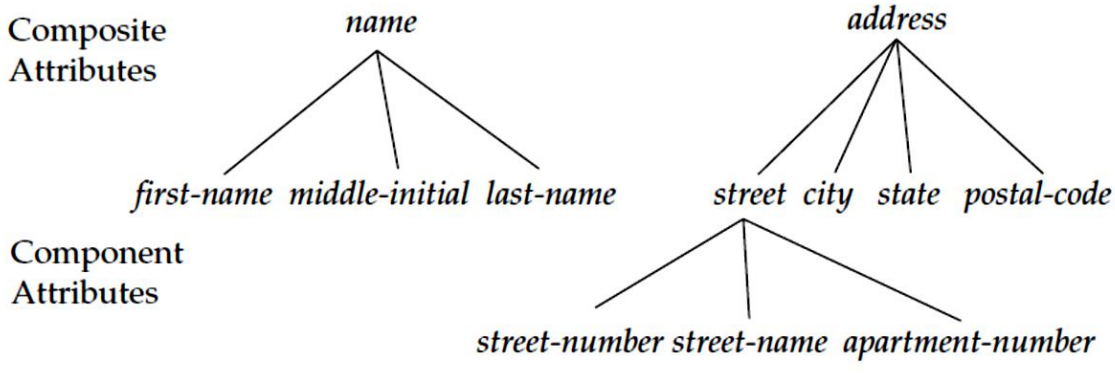


Figure : Composite attributes *customer-name* and *customer-address*

Single-valued Attributes - The attributes that have a single value for a particular entity.

For instance, the *loan-number* attribute for a specific loan entity refers to only one loan number. Such attributes are said to be **single valued**.

Multi-valued Attributes - The attributes that have multiple values for a particular entity.

There may be instances where an attribute has a set of values for a specific entity. Consider an *employee* entity set with the attribute *phone-number*. An employee may have zero, one, or several phone numbers, and different employees may have different numbers of phones. This type of attribute is said to be **multi-valued**.

Derived Attribute –

The value for this type of attribute can be derived from the values of other related attributes or entities.

Example - The *customer* entity set has an attribute *age*, which indicates the customer's age. If the *customer* entity set also has an attribute *date-of-birth*, we can calculate *age* from *date-of-birth* and the current date. Thus, *age* is a derived attribute.

In this case, *date-of-birth* may be referred to as a *base* attribute, or a *stored* attribute. The value of a derived attribute is not stored, but is computed when required.

Types of Attributes:

Simple attribute – attribute that consist of a single atomic value.
e.g. – Roll, Marks, Salary, etc.

Composite Attribute – attributes which can be decomposed further
e.g. – Name (First Name, Middle Name, Last Name)
Address (House No., Street, City, State)

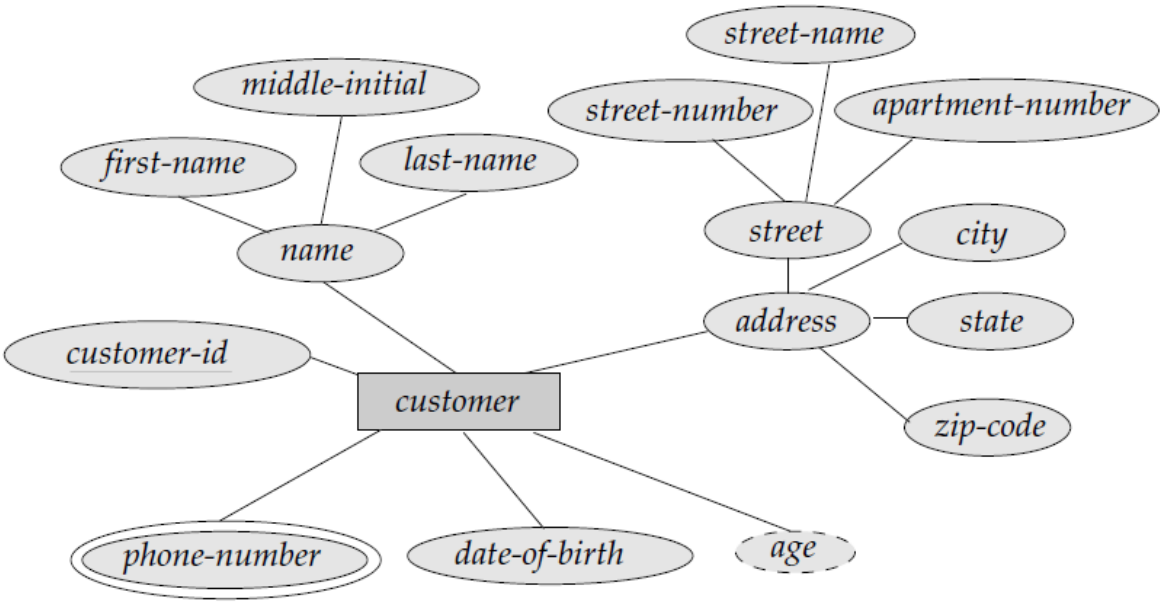
Single-valued atribute – attribute that hold a single value
e.g. – City, EmpId, Salary, etc.

Multi-valued attribute – attribute that hold multiple values
e.g. – Phone no., EmailID

Derived attribute – an attribute that can be calculated or derived from another attribute
e.g. – Age (derived from DOB)

Null attribute – an attribute whose value is missing for a record.
e.g. - if a record does not contain an assignment for the Price attribute

Example-



Relational Model

The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as **relations**. The relational model is an example of a **record-based model**.

Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type. The relational data model is the most widely used data model.

customer-id	*customer-name*	*customer-street*	*customer-city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

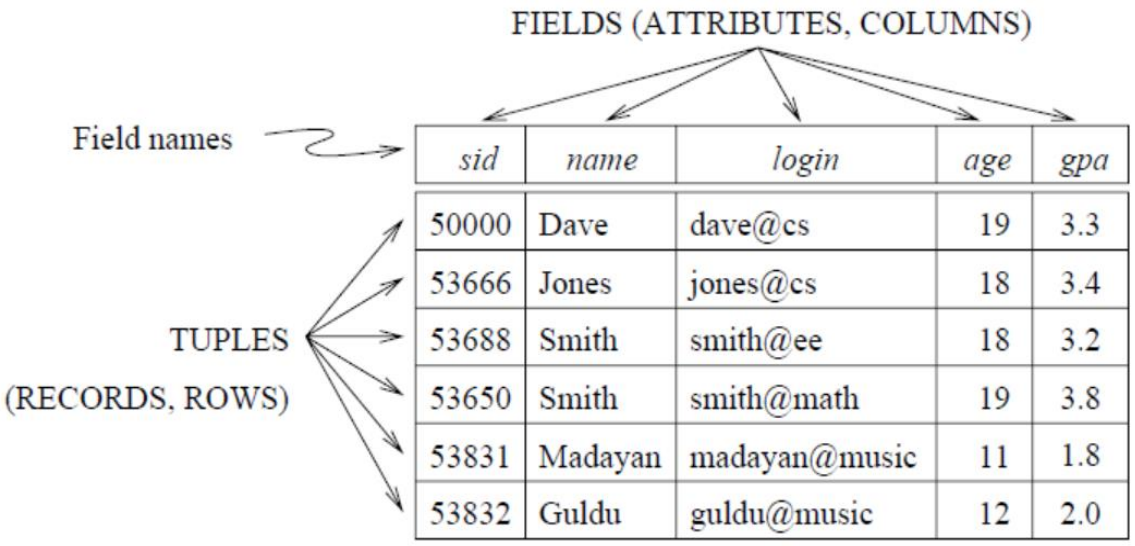
(b) The *account* table

customer-id	*account-number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

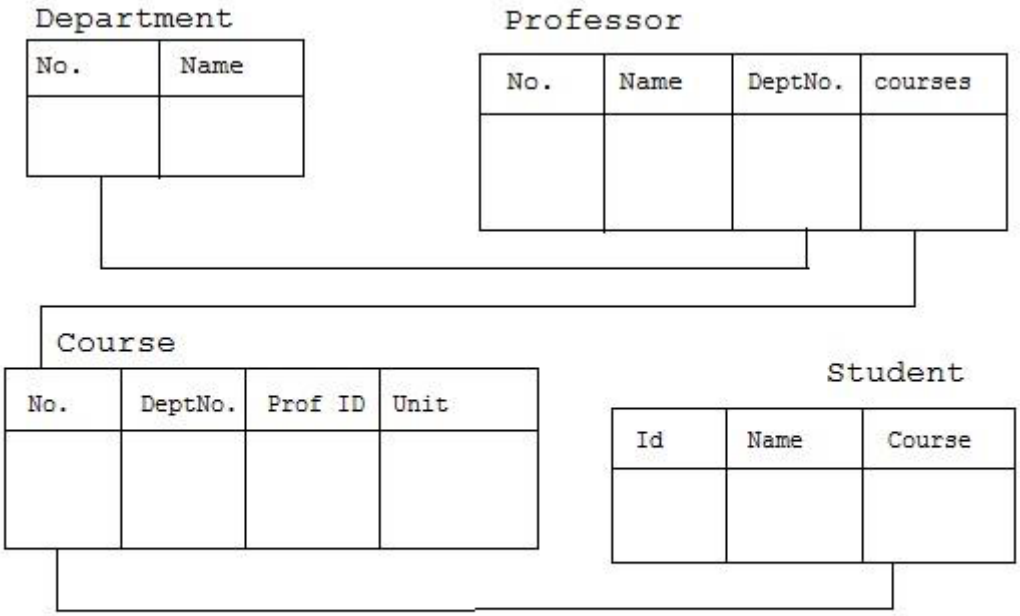
(c) The *depositor* table

Figure - A sample relational database

Example-



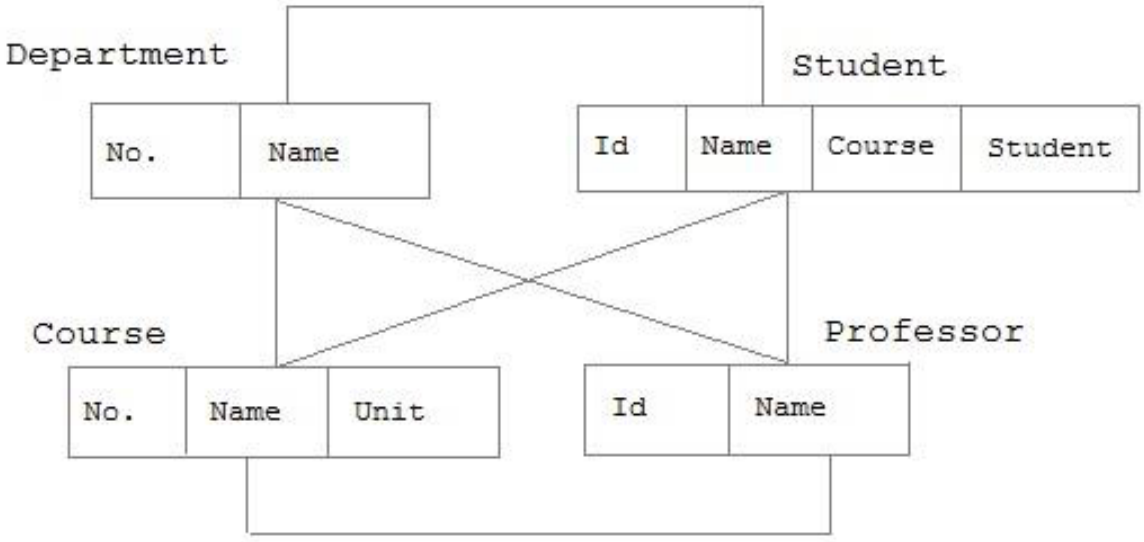
sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0



Network Data Model

Data is represented as a collection of records and relationship between data is represented by links which can be viewed as pointers. The record in the database are organized as collection of arbitrary graphs.

Example-

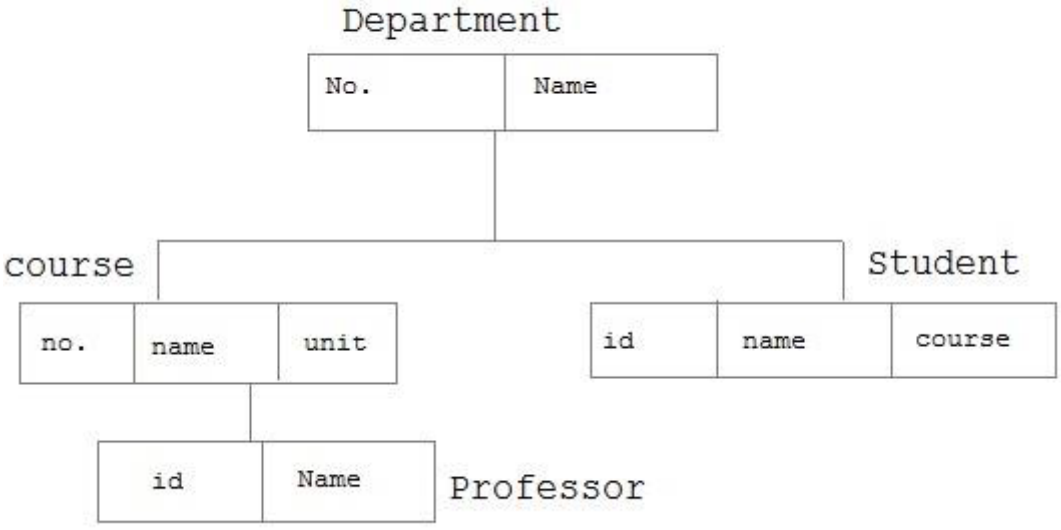


Hierarchical Data Model

It is different from network model in the way that records are organized into a tree like structure.

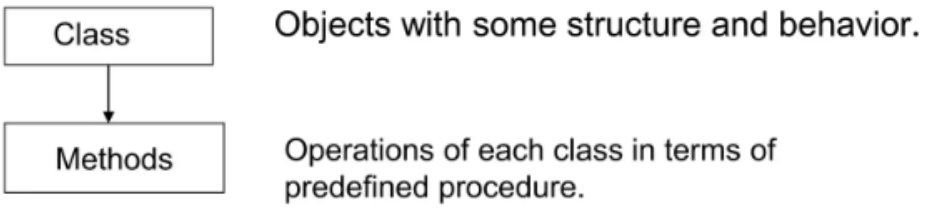
For eg. An organization might store information about an employee, such as name, dep, sal. The organization might also store information about employee's family. The employee and the family data forms hierarchy.

Example-



Object Oriented Data Model

It defines the database in terms of objects, their properties and their operations.



- Support the basic elements of the object approach used in object oriented programming languages like inheritance, use of methods, and encapsulation.
- Some object-oriented databases are designed to work well with object oriented programming languages such as Java, C++, C# etc.
- OODBMS use exactly the same model as object-oriented programming languages.

Database Languages :-

A database system provides a **data-definition language** to specify the database schema and a **data-manipulation language** to express database queries and updates.

Data-Definition Language

We specify a database schema by a set of definitions expressed by a special language called a **data-definition language** (**DDL**). The DDL is also used to specify additional properties of the data. We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a **data storage and definition** language.

Data-Manipulation Language

A **data-manipulation language (DML)** is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:

• Retrieval of information stored in the database

• Insertion of new information into the database

• Deletion of information from the database

• Modification of information stored in the database

There are basically two types:

Procedural DML require a user to specify *what* data are needed and *how* to get those data.

Non-Procedural DML require a user to specify *what* data are needed *without* specifying how to get those data.

Data Control Language (DCL) Commands:

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

Transaction Control Language (TCL) Commands:

Command	Description
commit	Save work done
Save point	Identify a point in a transaction to which we can later roll back.
Roll backs	Restore database to original since the last Commit

Query :- A **query** is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a **query language**.

Domain Constraints :- A domain of possible values must be associated with every attribute (for example, integer types, character types, date/time types).
Declaring an attribute to be of a particular domain act as a constraint on the values that it can take. Domain constraints are the most elementary form of integrity constraint. They are tested easily by the system whenever a new data item is entered into the database.

Database Administrator :-

One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data. A person who has such central control over the system is called a **database administrator (DBA)**. The functions of a DBA include:

• **Schema definition**. The DBA creates the original database schema by executing a set of data definition statements in the DDL.

• **Storage structure and access-method definition**.

• **Schema and Physical-Organization Modification**.

The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.

• **Granting of Authorization for Data Access**.

By granting different types of authorization, the database administrator can regulate which parts of the database various users can access.

The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.

Routine maintenance.

- Periodically **backing up the database**, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.
- Ensuring that **enough free disk space is available** for normal operations, and upgrading disk space as required.
- **Monitoring jobs running on the database** and ensuring that performance is not degraded by very expensive tasks submitted by some users.

View of Data

- A database system is a collection of interrelated files and a set of programs that allow users to access and modify these files.
- A major purpose of a database system is to provide users with an *abstract* view of the data. That is, the system hides certain details of how the data are stored and maintained.

Data Abstraction

- For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database.
- Since many database-systems users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:

Physical level - The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.

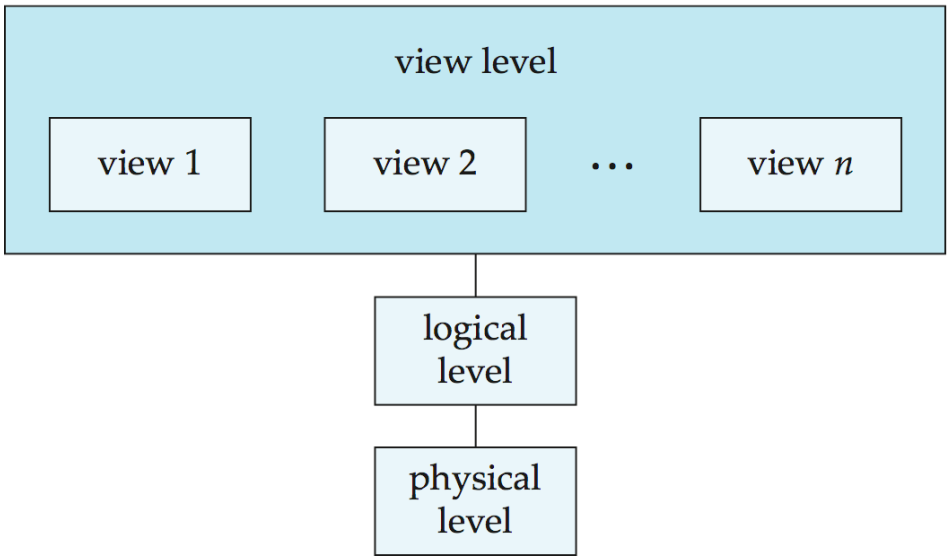
Logical level - The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data.

Database Administrators use the logical level of abstraction.

View level - The highest level of abstraction describes only part of the entire database.

Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database.

Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.



The three levels of data abstraction

Instances and Schemas

Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an **instance** of the database.

The overall design of the database is called the database **schema**. Schemas are changed infrequently, if at all.

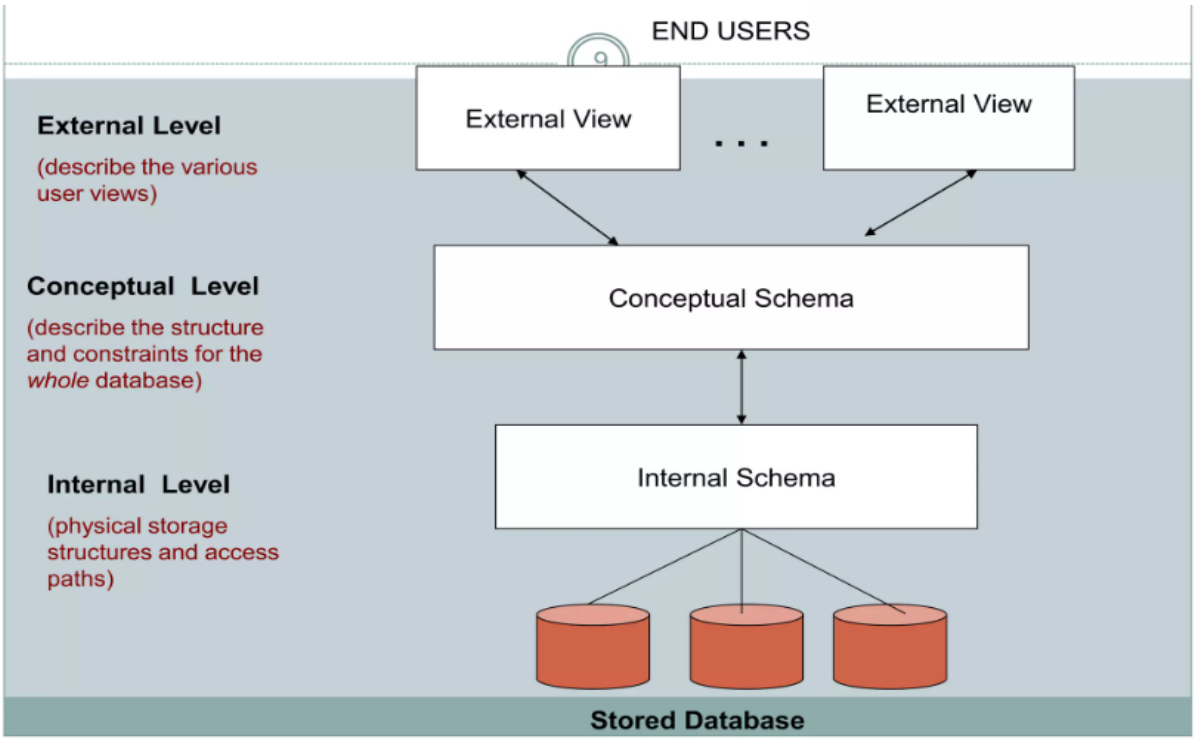
The concept of database schemas and instances can be understood by analogy to a program written in a programming language. A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Each variable has a particular value at a given instant. The values of the variables in a program at a point in time correspond to an *instance* of a database schema.

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

1. The **internal level** has an **internal schema,** which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

2. The **conceptual level** has a **conceptual schema,** which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. A high-level data model or an implementation data model can be used at this level.

3. The **external** or **view level** includes a number of **external schemas** or **user views.** Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. A high-level data model or an implementation data model can be used at this level.



DATA INDEPENDENCE:

- A very important advantage of using DBMS is that it offers Data Independence.
- The ability to modify a scheme definition in one level without affecting a scheme definition in a higher level is called **data independence**.
- There are two kinds:

1. Physical Data Independence
2. Logical Data Independence

Physical Data Independence:

- The ability to modify the physical schema without causing application programs to be rewritten
- Modifications at this level are usually to improve performance.

Logical Data Independence:

- The ability to modify the conceptual schema without causing application programs to be rewritten
- Usually done when logical structure of database is altered
- Logical data independence is harder to achieve as the application programs are usually heavily dependent on the logical structure of the data.

Data Independence

- Ability to modify a schema definition in one level without affecting a schema definition in the other levels.
- Logical data independence: Protection from changes in *logical* structure of data.
- Physical data independence: Protection from changes in *physical* structure of data.