Program : **B.Tech**

Subject Name: **Database Management System**

Subject Code: **IT-405**

Semester: **4th**

UNIT -I

Introduction

The database is a collection of similar data. The database management system is software designed to assist the maintenance and utilisation of large-scale collection of data. DBMS came into existence in 1960 by Charles. Integrated data store which is also called as the first general-purpose DBMS. Again in 1960 IBM brought IMS-Information management system. In 1970 Edgar Codd at IBM came with a new database called RDBMS. In 1980 then came SQL Architecture- Structure Query Language. From 1980 to 1990 there were advances in DBMS, e.g. DB2, ORACLE.

Data

• Data is raw fact or figures or entity.
• When activities in the organisation take place, the effect of these activities needs to be recorded which is known as Data.

Information

Processed data is called information
The purpose of data processing is to generate the information required for carrying out business activities.

We can further define different functions of DBMS in details like.
- **Data capture**: Collection of data from the place of origination like taking attendance in the class using at PDA.
- **Data classification:** Categorization of captured data based on different dimensions like size, type like segregation of data images, audio, video.
- **Data storage:** The categorised data must be stored, and it must maintain data persistence and integrity of it like we are storing details of students and it should remain the same for years.
- **Data arranging:** Arrangement of the data in the database in a proper manner that will help the user to understand how we are going to use it.
- **Data Retrieval:** Data required for retrieval so there must be mechanics through which they can be retrieved efficiently. All the DBMS software provider that will support a familiar language SQL used for the retrieval of data.
- **Data maintenance:** It is the database to keep it up-to-date. One of the critical feature of the DBMS through which data can be updated and remain useful for the user.
- **Data Verification:** Before storing the data, it must be checked based on the syntax and semantics to avoid errors. To keep data persistence, it is mandatory to verify it before the storage and must store in the database the way it supports.
- **Data Editing:** Providing tools that facilitate to perform change and update operation on DBMS.
- **Data transcription:** Providing a facility to transfer one form to another. User requirement changes very frequently and in the digital world expectations from the users are also very high. So, DBMS must provide facility to change it from one form to other.

Database

Let us understand first about the what is a database. So, a database is a set of data, and it contains interrelated data. The database contains a set of algorithm and rules through which data can be stored in it systematically. So, all the data fields must be related to each other. For example, where it is used suppose an organisation from attendance, salary calculation and different allowances are given to them can be done through the Database. To keep a record nowadays from billing an item to take attendance is the classroom database is required.

Evolution of DBMS: -
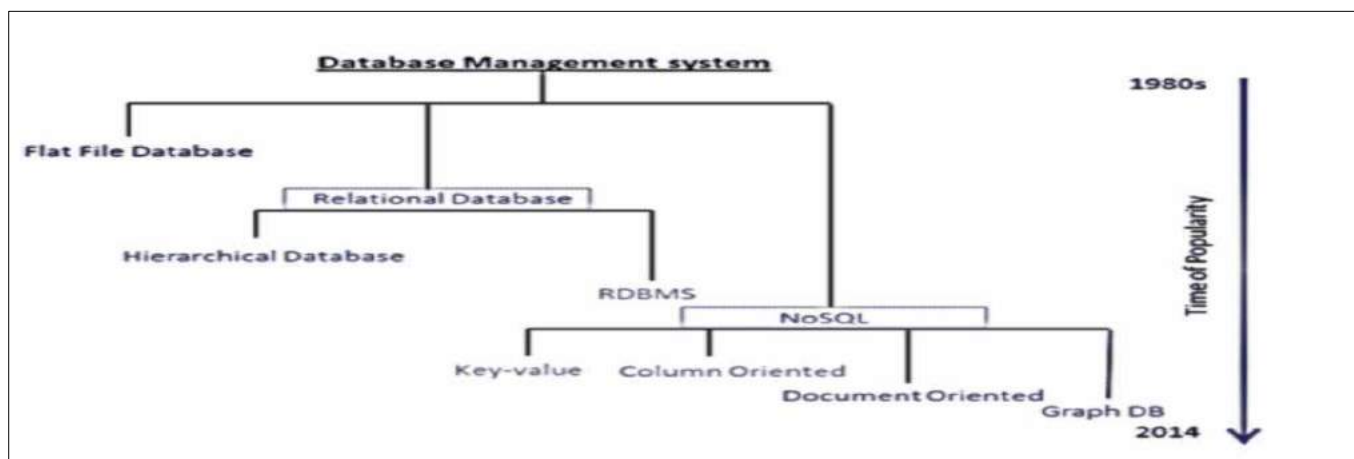
File Oriented Approach: -

Computer System comes into the picture to store the records and after performing computation producing information. As compared to the manual work done by the human being, they are more accurate and faster and secure also. Such a system stores a group of records like department wise or class wise record and each group has separate files, or for a category. Such arrangements called file processing system and in which each branch or department is having its file and a dedicated application designed to operate it. This is mainly to keep a record of the students like fees details, result details, and contact details and whenever some information is required then through an application program, it can be accessed. Still, file processing approach of data storage and access is used, but it has some drawbacks.

Database Management System

A Database Management System (DBMS) is a collection of programs that enables the user to create and maintain a database.
The DBMS is hence a general-purpose software system that facilitates the process of defining constructing and manipulating database for various applications.

Comparison Between File System & DBMS

DBMS	File System
1. DBMS is a collection of data and user is not required to write the procedures for managing the database.	1. The file system is a collection of data. Any management with the file system, the user must write the procedures
2. DBMS provides an abstract view of data that hides the details.	2. File system gives the details of the data representation and Storage of data.
3. DBMS is efficient to use since there are wide varieties of sophisticated techniques to store and retrieve the data.	3. In File system storing and retrieving of data cannot be done efficiently.
4. DBMS takes care of Concurrent access using some form of locking.	4. Concurrent access to the data in the file system has many problems like
5. DBMS has crash recovery mechanism DBMS protects the user from the effects of system failures.	a. Reading the file while other deleting some information, updating some information
6. DBMS has a good protection mechanism.	5. The file system does not provide a crash recovery mechanism.
	E.g. While we are entering some data into the file if System crashes then the content of the file is lost.
	6. Protecting a file under file system is very difficult.

A database management system is, well, a system used to manage databases.

RDBMS

A relational database management system is a database management system used to manage relational databases. A relational database is one where tables of data can have relationships based on primary and foreign keys.

Advantages of DBMS

Due to its centralised nature, the database system can overcome the disadvantages of the file system-based system

● **Concurrent Use** a database system provides facility to access database several users concurrently. Let we understand it by taking an example that a movie online booking system database employee of different branches concurrently access the database. Each employee can handle customers at their desk, Able to see the seats available for the booking from the interface provided to them.

● **Structured data & details** one of the essential features of the database system is not only to store the data and to provide access to the database. However, it also provides details about the data and how to access and use it. Taking an example when you are going for an online form of exam then details about every field is given, what type of data it accepts and format details also.

● **Data Independence**, another essential characteristic of the DBMS, is data independence in which application through which user can interact with the user is not dependent on the physical data storage. So, if there is any change in the application program, it will not affect the data stored in the database.

● **The integrity of data** This characteristic of the DBMS deals with one of the fundamental properties of the data called integrity, in which if some data is saved in the database and later retrieved from the database it must be same. This also covers restriction on the unauthorised access of the data that can make changes in the data sets.

● **Transaction of Data,** a transaction is a set of actions that are done in a database to transfer it from one consistent state to another. If it is not handled correctly, it may result in inconsistent state and loss of data, so DBMS has certain constraints to maintain the fundamental properties of transaction like atomicity, integrity, isolation, and durability.

● **Data Persistence,** this is one of the essential characteristics of the database because data can be retained in the database for years and it should be in the same condition. In the banking system, a user will open his account and lifelong maintain it, or a user has opted an LIC policy, or media claim policy.

A modern DBMS has the following characteristics –

Real-world entity – A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behaviour and attributes too. For example, a school database may use students as an entity and their age as an attribute.

Relation-based tables – DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.

Isolation of data and application – A database system is entirely different from its data. A database is an active entity, whereas data is said to be passive, on which the database works and organises. DBMS also stores metadata, which is data about data, to ease its process.

Less redundancy – DBMS follows the rules of normalisation, which splits a relation when any of its attributes is having redundancy in values. Normalisation is a mathematically rich and scientific process that reduces data redundancy.

Consistency – is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect an attempt of leaving the database in the inconsistent state.

Query Language – DBMS is equipped with a query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where the file-processing system was used.

ACID Properties – DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (commonly

shortened as ACID). These concepts are applied to transactions, which manipulate data in a database.

Multiuser and Concurrent Access – DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, users are always unaware of them.

Multiple views – DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of a database than a person working in the Production department. This feature enables the users to have a concentrated view of the database according to their requirements.

Security – Features like multiple views offer security to some extent where users are unable to access data from other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage.

DBMS Architecture: -

The design of a DBMS depends on its architecture. It can be centralised or decentralised or hierarchical. The architecture of a DBMS be either single tier or multi-tier. An n-tier architecture divides the entire system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers usually prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use a 2-tier architecture where they access the DBMS using an application. Here the application tier is entirely independent of the database regarding operation, design, and programming.

3-Tier Architecture

Database (Data) Tier – At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

Application (Middle) Tier – At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

User (Presentation) Tier – End-users operate on this tier, and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.
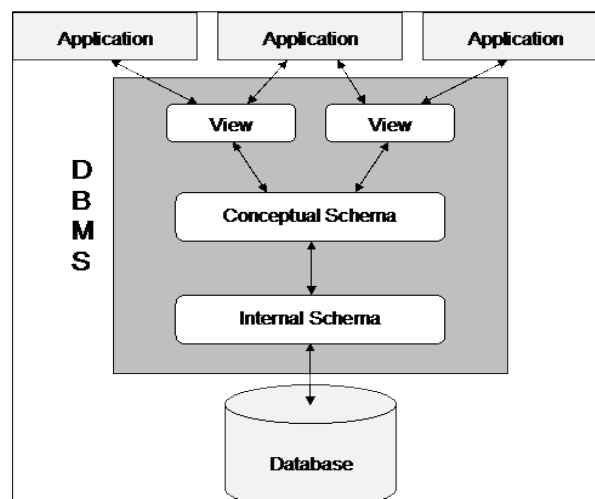
DBMS Three-Tier Architecture

DBMS Architecture Component: -
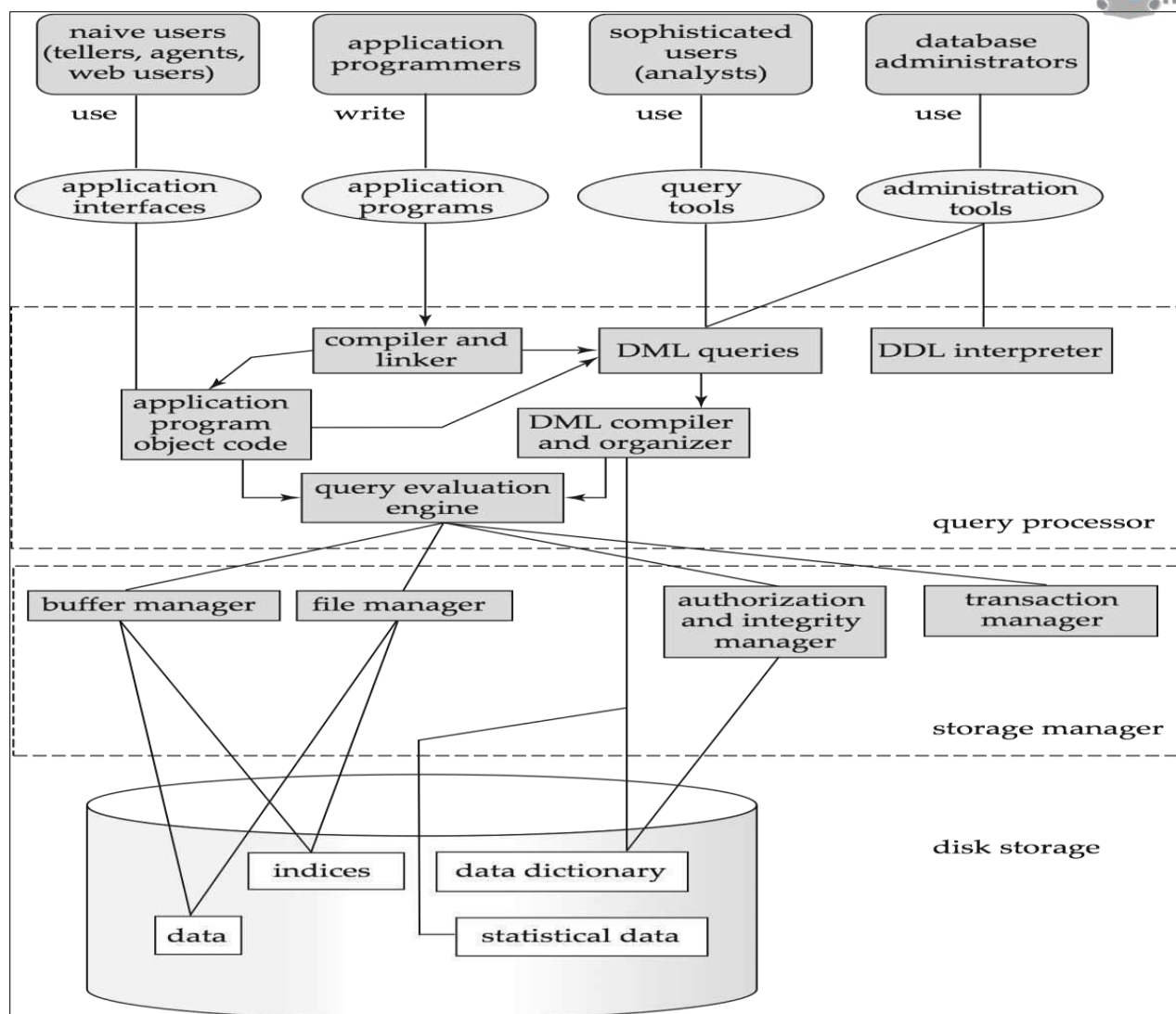
DBMS Architecture Component

Database Users:

Users are differentiated by the way they expect to interact with the system:

Application programmers:
Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.
Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.

Sophisticated users:
Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language.

Specialised users:
Specialised users are sophisticated users who write specialised database applications that do not fit into the traditional data-processing framework.

Naïve users:

Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

Database Administrator:
Coordinates all the activities of the database system. The database administrator has a good understanding of the enterprise's information resources and needs.

Query Processor:

The query processor will accept a query from a user and solves it by accessing the database.

Parts of Query processor:

DDL interpreter

This will interpret DDL statements and fetch the definitions in the data dictionary.

DML compiler

a. This will translates DML statements in a query language into low-level instructions that the query evaluation engine understands.

b. A query can usually be translated into any of some alternative evaluation plans for same query result DML compiler will select the best plan for query optimisation.

Query evaluation engine

This engine will execute low-level instructions generated by the DML compiler on DBMS.

Storage Manager/Storage Management:

A storage manager is a program module which acts as an interface between the data stored in a database and the application programs and queries submitted to the system.

The storage manager components include:

Authorisation and integrity manager: Checks for integrity constraints and authority of users to access data.

Transaction manager: Ensures that the database remains in a consistent state although there are system failures.

File manager: Manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

Buffer manager: It is responsible for retrieving data from disk storage into main memory. It enables the database to handle data sizes that are much larger than the size of main memory.

Data files: Stored in the database itself.

Data dictionary: Stores metadata about the structure of the database.

Indices: Provide fast access to data items.

Data Models

Data models define how the logical structure of a database is modelled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected and how they are processed and stored inside the system.

The very first data model could be flat data models, where all the data used are to be kept in the same plane. Earlier data models were not so scientific; hence they were prone to introduce lots of duplication and update anomalies.

There are many kinds of data models. Some of the most common ones include:
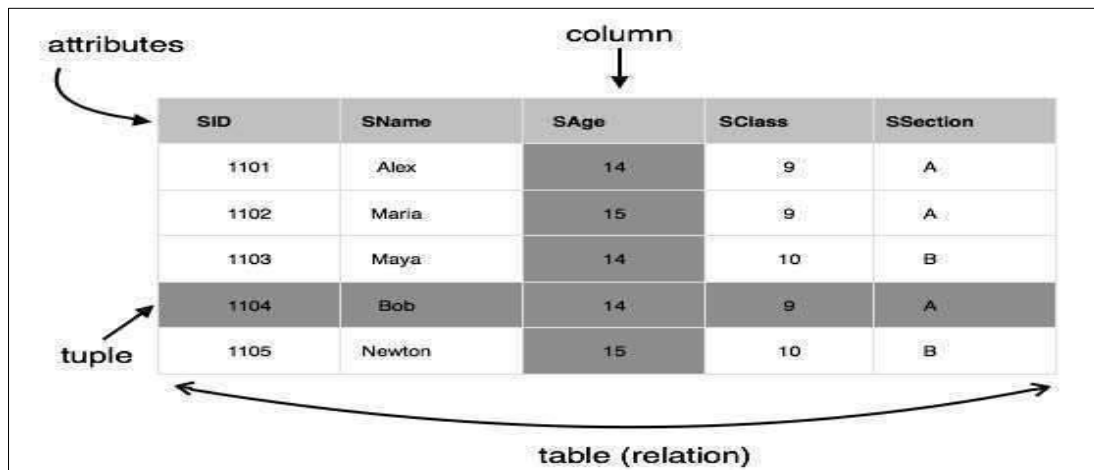
- Hierarchical database model
- Relational model
- Network model
- Object-oriented database model
- Entity-relationship model
- Document model
- Entity-attribute-value model
- Star Schema
- The object-relational model, which combines the two that make up its name.

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 in question, such as price, zip code, or birth date. Together, the attributes in a relation are called a domain. An attribute or combination of attributes is chosen as a primary key that can be referred to in other tables when it is called a foreign key.

Each row also called a tuple, includes data about a specific instance of the entity in question, such as an employee.

The model also accounts for the types of relationships between those tables, including one-to-one, one-to-many, and many-to-many relationships. Here's an example:

Relational model: -

Within the database, tables can be normalised, or brought to comply with normalisation rules that make the database flexible, adaptable, and scalable. When normalised, each piece of data is atomic or broken into the smallest useful pieces.

Relational databases are typically written in Structured Query Language (SQL). E.F. Codd introduced the model in 1970.

The main highlights of this model are –
- Data is stored in tables called relations.
- Relations can be normalised.
- In normalised relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from the same domain.

Hierarchical model
The hierarchical model organises data into a tree-like structure, where each record has a single parent or root. Sibling records are sorted in an order. That order is used as the physical order for storing the database. This model is useful for describing many real-world relationships.
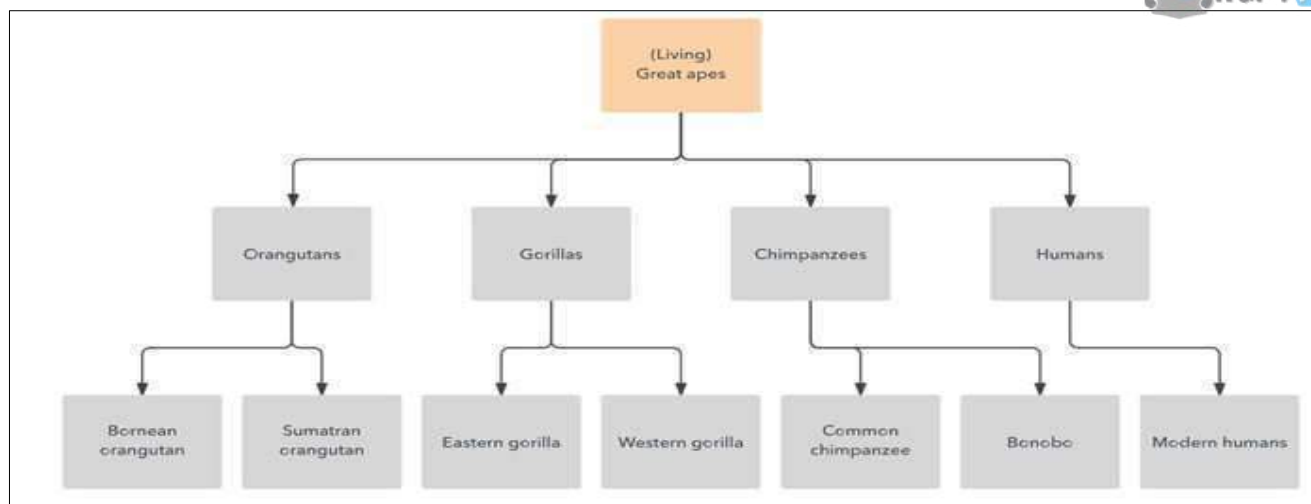
Network model

The network model builds on the hierarchical model by allowing many-to-many relationships between linked records, implying multiple parent records. Based on mathematical set theory, the model is constructed with sets of related records. Each set consists of one owner or parent record and one or more member or child records. A record can be a member or child in multiple sets, allowing this model to convey complex relationships.
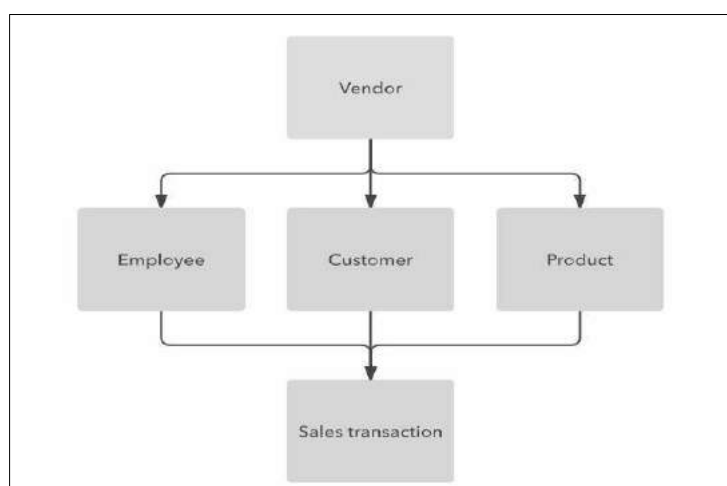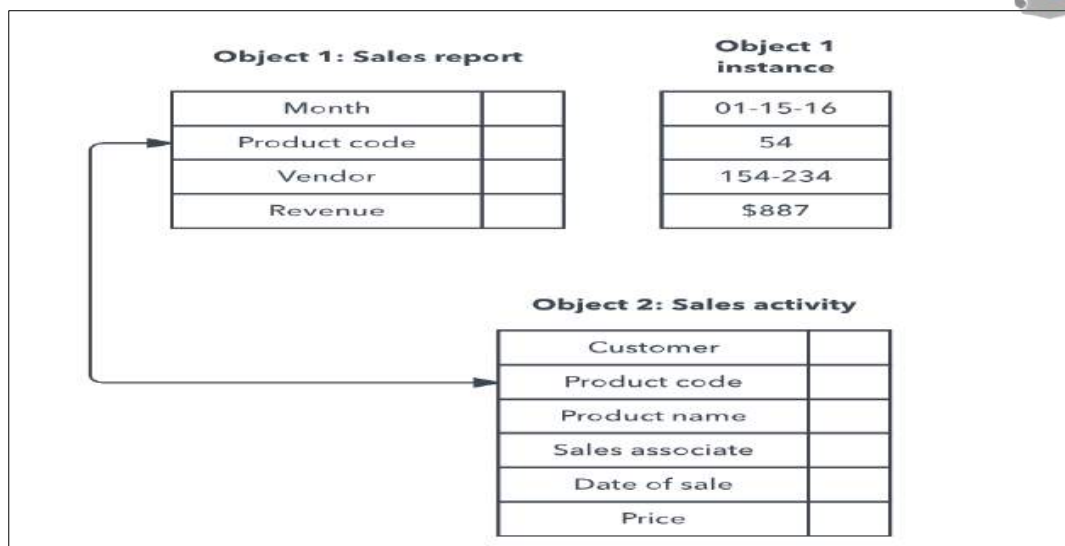

Object-oriented database model

This model defines a database as a collection of objects, or reusable software elements, with associated features and methods. There are several kinds of object-oriented databases:
A multimedia database incorporates media, such as images, that could not be stored in a relational database.
A hypertext database allows any object to link to any other object. It is useful for organising lots of disparate data, but it is not ideal for numerical analysis.
The object-oriented database model is the best known post-relational database model, since it incorporates tables, but isn't limited to tables. Such models are also known as hybrid database models.
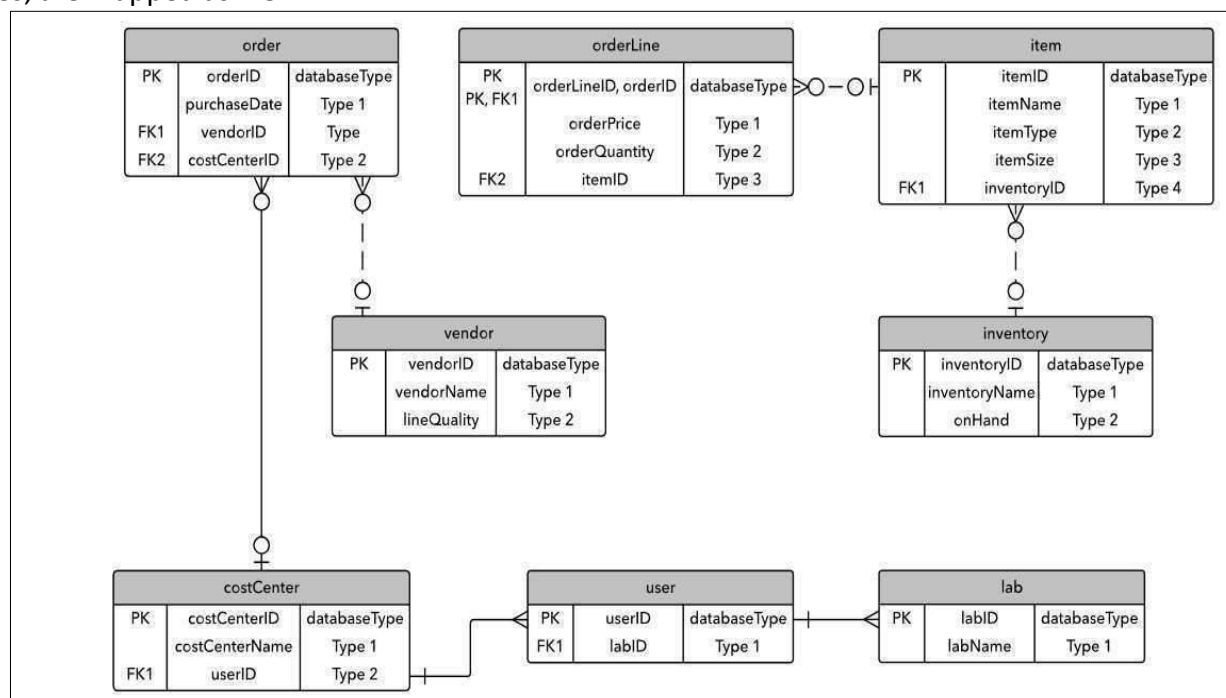
Object-relational model

This hybrid database model combines the simplicity of the relational model with some of the advanced functionality of the object-oriented database model. It allows designers to incorporate objects into the simple table structure.
Languages and call interfaces include SQL3, vendor languages, ODBC, JDBC, and proprietary call interfaces that are extensions of the languages and interfaces used by the relational model.

Entity-relationship model

This model captures the relationships between real-world entities much like the network model, but it is not as directly tied to the physical structure of the database. Instead, it is often used for designing a database conceptually.

Here, the people, places, and things about which data points are stored are referred to as entities, each of which has specific attributes that together make up their domain. The cardinality, or relationships between entities, are mapped as well.

NoSQL database models

In addition to the object database model, other non-SQL models have emerged in contrast to the relational model: The graph database model, which is even more flexible than a network model, allowing any node to connect with any other — the multi-valued model, which breaks from the relational model by allowing attributes to contain a list of data rather than a single data point.

Schema and Instance in DBMS
Design of a database is called the schema. The schema is of three types:

- **Physical schema:** - The design of a database at the physical level is called physical schema, how the data stored in blocks of storage is described at this level.

- **Logical schema:** - Design of database at a logical level is called logical schema, programmers and database administrators work at this level, at this level data can be described as certain types of data records gets stored in data structures, however the internal details such as implementation of data structure is hidden at this level (available at physical level).

- **View schema:** - Design of database at view level is called view schema. This generally describes end-user interaction with database systems.

Instances: - The data stored in a database at a moment of time is called an instance of the database. The database schema defines the variable declarations in tables that belong to a database the value of these variables at a moment of time is called the instance of that database.

Data Independence
A database system contains typically many data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data quickly. It is rather difficult to modify or update a set of metadata once it is stored in the database. However, as a DBMS expands, it needs to change over time to satisfy the requirements of the users.

Logical Data Independence
Logical data is data about the database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied to that relation.
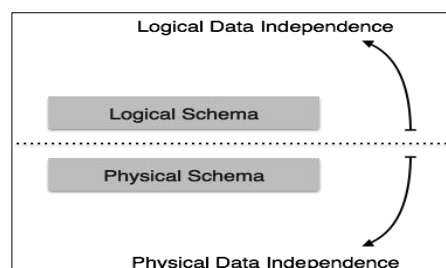

Logical data independence is a kind of mechanism, which liberalises itself from actual data stored on the disk. If we do some changes in table format, it should not change the data residing on the disk.

Physical Data Independence
All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

For example, in case we want to change or upgrade the storage system itself – suppose we want to replace hard-disks with SSD – it should not have any impact on the logical data or schemas. Requirements of the users. If the entire data is dependent, it will become a tedious and highly complex job.

DBA
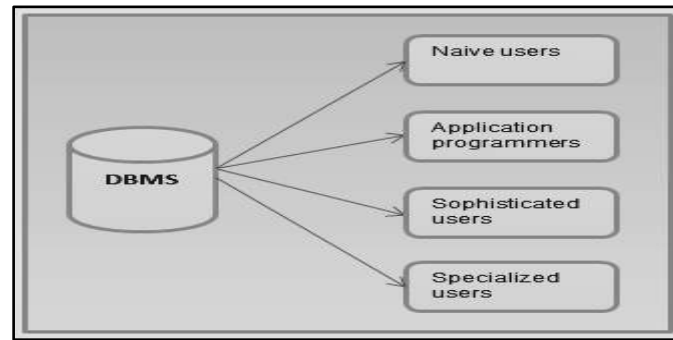
A data administration (also known as a database administration manager, data architect, or information centre manager) is a high-level function responsible for the overall management of data resources in an organisation. In order to perform its duties, the DA must know a good deal of system analysis and programming.

These are the functions of a data administrator (not to be confused with database administrator functions):

1. Data policies, procedures, standards
2. Planning- development of an organization's IT strategy, enterprise model, cost/benefit model, the design of the database environment, and administration plan.
3. Data conflict (ownership) resolution
4. Data analysis- Define and model data requirements, business rules, operational requirements, and maintain corporate data dictionary
5. Internal marketing of DA concepts
6. Managing the data repository

E-R Diagram
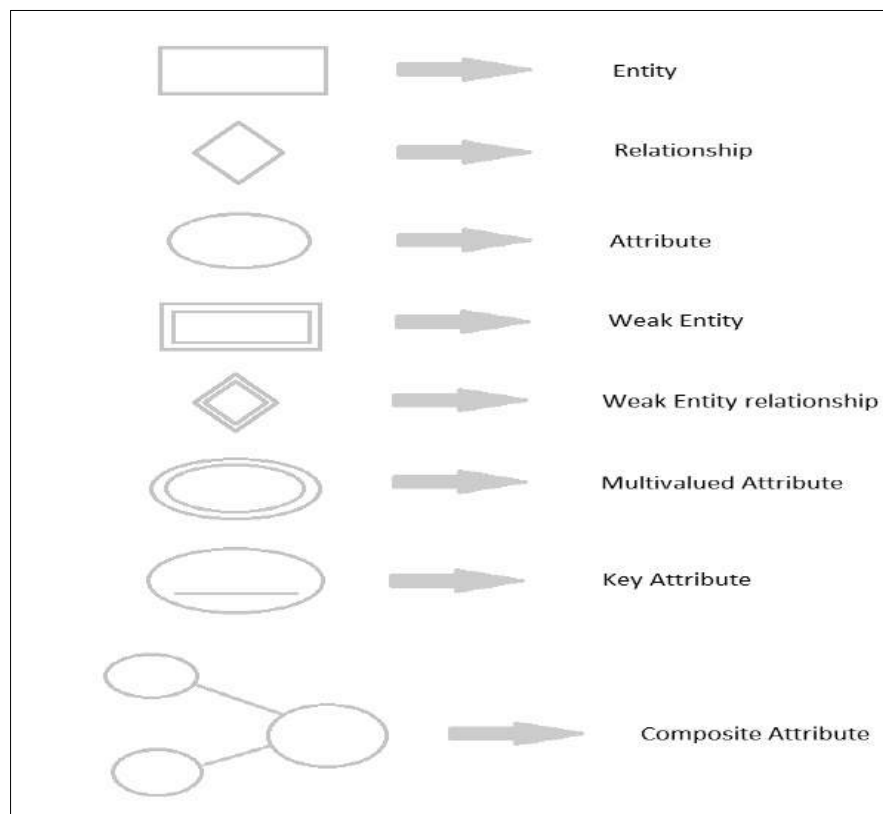
ER Model is represented using an ER diagram. Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram.

Entity
Entities are represented using rectangles. Rectangles are named with the entity set they represent.
EX. Student, Faculty, Course
- **Strong Entity: -** This type of entities has key attributes set that are used to create a primary key. Like student entity has Roll no as a critical attribute.
- **Weak Entity: -** This type of Entities does not have any primary key attribute, and they are dependent on some other entity have the crucial prime attribute. For example, Employee child is a weak entity some other entity has the crucial prime attribute. For example, Employee child is a weak entity which is dependent on the Employee Entity.
- **Composite Entity:** - This type of entities is used to replace many to many relationships, and Entity replaces that relationship. It is of three types one-to-one cardinality, one-to-many cardinality, many-to-many cardinality.

Attributes
Attributes are the properties of entities. Attributes are represented using ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).
Derived attributes: - This is the attributes that can be derived from other attributes. Like Age can be derived from the present date and Date of Birth.
Composite Attribute: - This is the attributes that can be created with the combination of two other attributes. Like a combination of first name and Last name will give the full name.
Single-value attribute: - This is the attributes which have single value when they are converted in the form of a table. Like DOB of a person.
Multi-Valued Attribute: - This is the attributes which have multiple values for the instance When an Entity is converted to a table, and that attribute allows multiple values for it. Like a person have two contact numbers.

3) Relationship
A Relationship describes relations between **entities**. The relationship is represented using diamonds.

There are three types of relationship that exist between Entities.
- Binary Relationship
- Recursive Relationship
- Ternary Relationship

Binary Relationship
Binary Relationship means the relation between two Entities. This is further divided into three types.
- **One to One:** This type of relationship is rarely seen in the real world.

The above example describes that one student can enrol only for one course and a course will also have only one Student. This is not what you will usually see in the relationship.
- **One to Many:** It reflects business rule that one entity is associated with any numbers of the same entity.

The example for this relation might sound a little weird, but this means that one student can enrol too many courses, but one course will have one Student.
The arrows in the diagram describe that one student can enrol for only one course.
- **Many to One:** It reflects a business rule that many entities can be associated with just one entity. For example, Student enrols for only one Course, but a Course can have many Students.
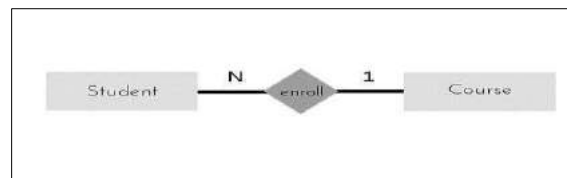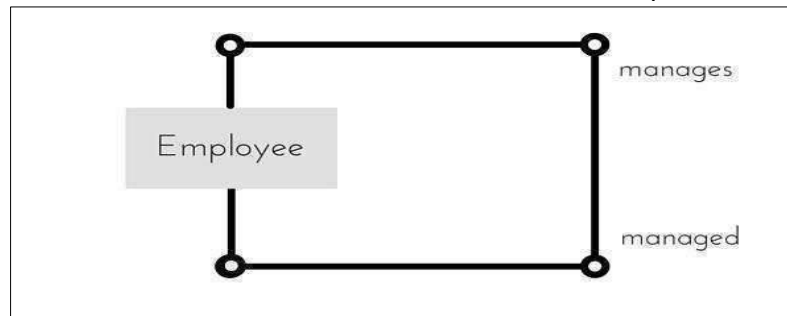
- **Many to Many:**

The above diagram represents that many students can enrol for more than one courses.

Recursive Relationship

When an Entity is related with itself, it is known as **Recursive** Relationship.

Ternary Relationship

Relationship of degree three is called Ternary relationship.

Here we are going to design an Entity Relationship (ER) model for a college database.
We have the following statements.

- A college contains many departments
- Each department can offer any number of courses
- Many instructors can work in a department
- An instructor can work only in one department
- For each department, there is a Head
- An instructor can be head of only one department
- Each instructor can take any number of courses
- Only one instructor can take a course
- A student can enrol for any number of courses
- Each course can have any number of students

Step 1: Identify the Entities

What are the entities here?
From the statements given, the entities are
Department
Course
Instructor
Student

Step 2: Identify the relationships

- One department offers many courses. However, one course can be offered by only one department. hence the cardinality between department and course is One to Many (1: N)
- One department has multiple instructors. However, instructor belongs to only one department. Hence the cardinality between department and instructor is One to Many (1: N)
- One department has only one head, and one head can be the head of only one department. Hence the cardinality is one to one. (1:1)
- One course can be enrolled by many students, and one student can enrol in many courses. Hence the cardinality between the course and the student is Many to Many (M: N)

- Only one instructor teaches one course. However, one instructor teaches many courses. Hence the cardinality between the course and the instructor is Many to One (N: 1)

Step 3: Identify the key attributes

- "Departmen_Name" can identify a department uniquely. Hence Department Name is the key attribute for the Entity "Department".
- Course_ID is the critical attribute for "Course" Entity.
- Student_ID is the critical attribute for "Student" Entity.
- Instructor_ID is the critical attribute for "Instructor" Entity.

Step 4: Identify other relevant attributes

- For the department entity, other attributes are the location
- For course entity, other attributes are course_name, duration
- For instructor entity, other attributes are first_name, last name, phone
- For student entity, first_name, last_name, phone

Step 5: Draw a complete ER diagram

By connecting all these details, we can now draw an ER diagram as given below.
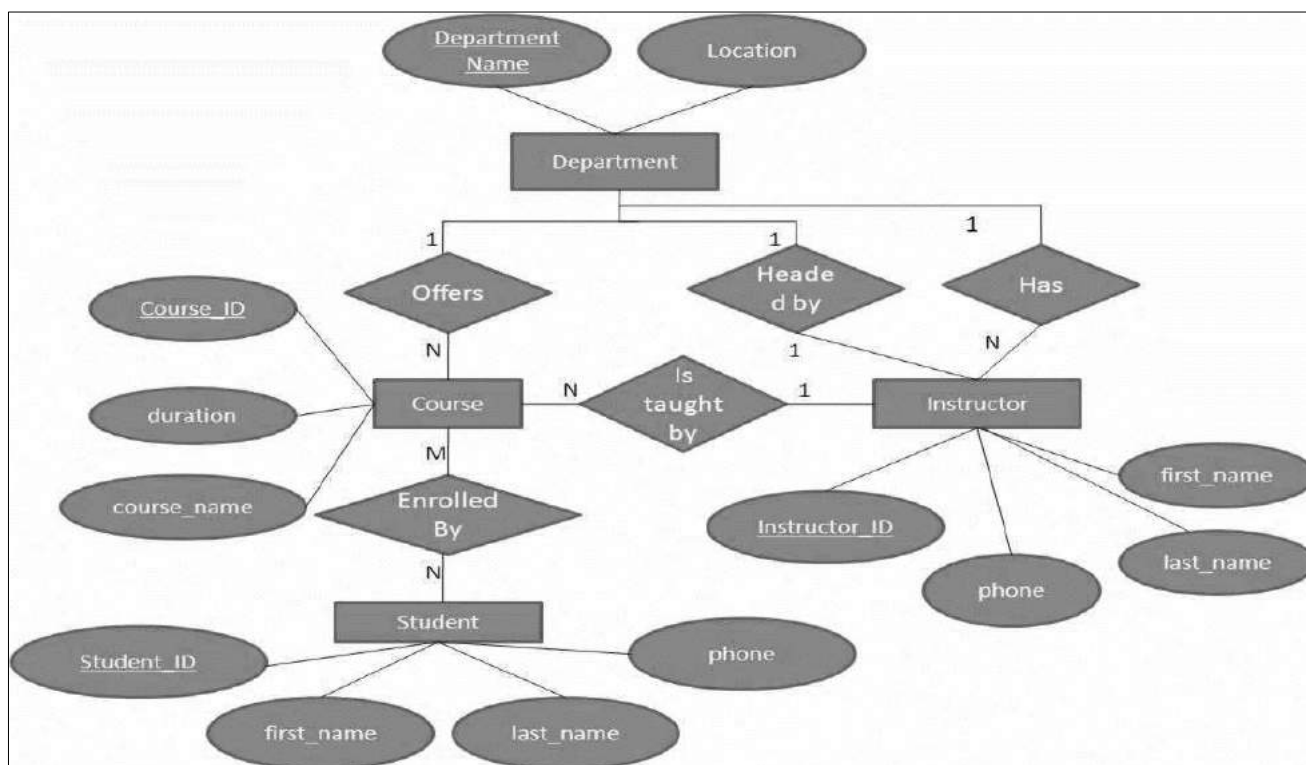
We hope you find these notes useful.

You can get previous year question papers at
https://qp.rgpvnotes.in .

If you have any queries or you want to submit your
study notes please write us at
rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in