
RELATIONAL DATABASE MANAGEMENT

UNIT-I

SYLLABUS

BASIC CONCEPTS: Database Management System - File based system - Advantages of DBMS over file-based system - Database Approach - Logical DBMS Architecture - Three level architecture of DBMS or logical DBMS architecture - Need for three level architecture - Physical DBMS Architecture - Database Administrator (DBA) Functions & Role - Data files indices and Data Dictionary - Types of Database.

Relational and ER Models: Data Models - Relational Model – Domains - Tuple and Relation - Super keys - Candidate keys - Primary keys and foreign key for the Relations - Relational Constraints - Domain Constraint - Key Constraint - Integrity Constraint - Update Operations and Dealing with Constraint Violations - Relational Operations - Entity Relationship (ER) Model – Entities – Attributes – Relationships - More about Entities and Relationships - Defining Relationship for College Database - E-R Diagram - Conversion of E-R Diagram to Relational Database.

1) WHAT IS DATA AND DATABASE?

A) **Data** is nothing but facts and statistics stored or free flowing over a network, generally it's raw and unprocessed. This “data” is used in a variety of forms of text, numbers, media and many more. Data becomes **information** when it is processed, turning it into something meaningful.

A **database** is an organized collection of data, so that it can be easily accessed and managed. You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data. There are many **databases available** like MySQL, Sybase, Oracle, MongoDB, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

SQL or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

The major components of the Database are:

- **Hardware** : This consists of a set of physical electronic devices such as I/O devices, storage devices and many more. It also provides an interface between computers and real-world systems.
- **Software** : This is the set of programs that are used to control and manage the overall Database. It also includes the DBMS software itself. The Operating System, the network software being used to share the data among the users, the application programs used to access data in the DBMS.
- **Data** : Database Management System collects, stores, processes, and accesses data. The Database holds both the actual or operational data and the metadata.

2) What is DBMS?

A) **DBMS** is a 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.

Here are some examples of popular DBMS used these days:

- MySql
- Oracle
- SQL Server
- IBM DB2
- PostgreSQL
- Amazon SimpleDB (cloud based) etc.

Characteristics of Database Management System (DBMS)

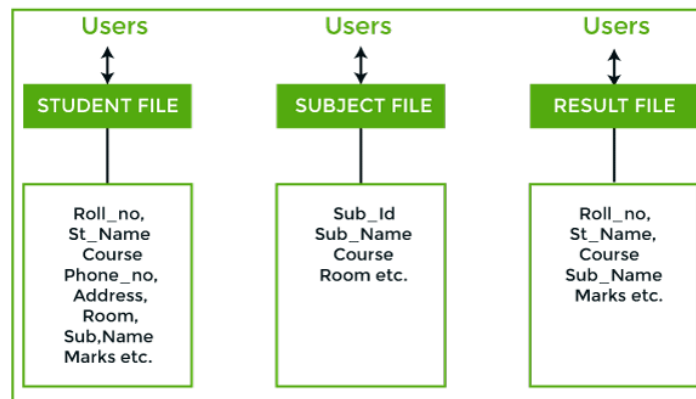
A database management system has following characteristics:

1. **Data stored into Tables:** Data is stored into tables, created inside the database. You can easily understand what type of data is stored where by looking at all the tables created in a database.
2. **Reduced Redundancy:** Unnecessary repetition of data in database was a big problem. But DBMS follows **Normalisation** which divides the data in such a way that repetition is minimum.
3. **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.
4. **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.
5. **Query Language:** DBMS provides users with a simple Query language, using which data can be easily fetched, inserted, deleted and updated in a database.
6. **Security:** The DBMS also takes care of the security of data, protecting the data from unauthorised access.
7. DBMS supports **transactions**, which allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used.

Q:2 WHAT IS FILE BASED SYSTEM OR FILE SYSTEM?

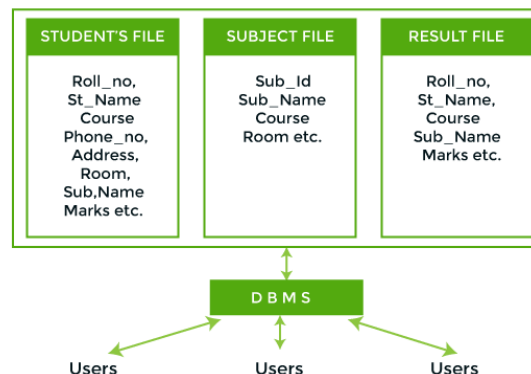
A) File based systems is also called a traditional based approach in which a decentralized approach A file system is a technique of arranging the files in a storage medium like a hard disk, pen drive, DVD, etc. It helps you to organize the data and allows easy retrieval of files when they are required. It mostly consists of different types of files like mp3, mp4, txt, doc, etc. that are grouped into directories.

A file system enables you to handle the way of reading and writing data to the storage medium. It is directly installed into the computer with the Operating systems such as Windows and Linux.



In the above figure: Consider an example of a student's file system. The student file will contain information regarding the student (i.e. roll no, student name, course etc.). Similarly, we have a subject file that contains information about the subject and the result file which contains the information regarding the result. Some fields are duplicated in more than one file, which leads to data redundancy. So to overcome this problem, we need to create a centralized system, i.e. DBMS approach.

DBMS: A database approach is a well-organized collection of data that are related in a meaningful way which can be accessed by different users but stored only once in a system. The various operations performed by the DBMS system are: Insertion, deletion, selection, sorting etc.



In the above figure, duplication of data is reduced due to centralization of data.

3) What is DBMS? EXPLAIN ADVANTAGES OF DBMS OVER FILE SYSTEM? and disadvantages of DBMS.

A) **DBMS** is a 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.

Advantages of Database Management System (DBMS)

- 1. Improved data sharing:** The DBMS helps to create an environment in which end users have better access to more and better-managed data.
- 2. Improved data security:** The more users access the data, the greater the risks of data security breaches. A DBMS provides a framework for better enforcement of data privacy and security policies.
- 3. Better data integration:** Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture
- 4. Minimized data inconsistency:** Data inconsistency exists when different versions of the same data appear in different places. The probability of data inconsistency is greatly reduced in a properly designed database.
- 5. Improved data access:** The DBMS makes it possible to produce quick answers to queries.
- 6. Improved decision making:** Better-managed data and improved data access make it possible to generate better-quality information, on which better decisions are based.
- 7. Increased end-user productivity:** The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.
- 8. Minimized Data redundancy:** DBMS follows **Normalization** which divides the data in such a way that repetition is minimum.

Disadvantages of Database Management System (DBMS):

Although the database system yields considerable advantages over previous data management approaches, database systems do carry significant disadvantages. For example:

- 1. Increased costs:** The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial. Training, licensing, and regulation compliance costs are often overlooked when database systems are implemented.
- 2. Management complexity:** Database systems interface with many different technologies and have a significant impact on a company's resources and culture. Given the fact that database systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.
- 3. Maintaining currency:** To maximize the efficiency of the database system, you must keep your system current. Therefore, you must perform frequent updates and apply the latest patches and security measures to all components.
- 4. Frequent upgrade/replacement cycles:** DBMS vendors frequently upgrade their products by adding new functionality. the upgrades themselves cost money, but it also costs trained database users and administrators to properly use and manage the new features.

Q:4 Difference between DBMS and File systems:

Basis	DBMS Approach	File System Approach
Meaning	DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	The file system is a collection of data. In this system, the user has to write the procedures for managing the database.
Sharing of data	Due to the centralized approach, data sharing is easy.	Data is distributed in many files, and it may be of different formats, so it isn't easy to share data.
Security and Protection	DBMS provides a good protection mechanism.	It isn't easy to protect a file under the file system.
Recovery Mechanism	DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from system failure.	The file system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will be lost.
Concurrency Problems	DBMS takes care of Concurrent access of data using some form of locking.	In the File system, concurrent access has many problems like redirecting the file while deleting some information or updating some information.
Cost	The database system is expensive to design.	The file system approach is cheaper to design.
Data Redundancy and Inconsistency	Redundancy is the concept of repetition of data. Due to the centralization of the database, the problems of data redundancy and inconsistency are controlled.	In this, the files and application programs are created by different programmers so that there exists a lot of duplication of data which may lead to inconsistency.
Structure	The database structure is complex to design.	The file system approach has a simple structure.
Integrity Constraints	Integrity Constraints are easy to apply.	Integrity Constraints are difficult to implement in file system.

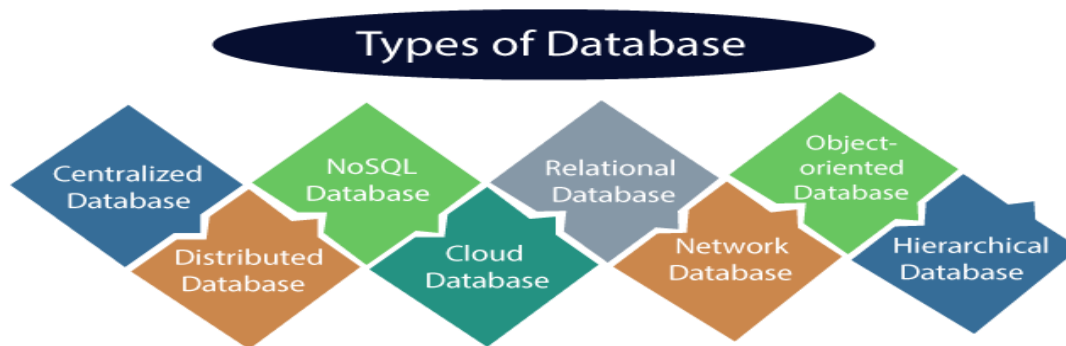
Data Models	In the database approach, there is concept of data models exists.	In the file system approach, there is no concept of data models exists.
Flexibility	changes are done more easily with a database approach.	The flexibility of the system is less as compared to the DBMS approach.
Examples	Oracle, SQL Server, Sybase etc.	Cobol, C++ etc.

Q:5) WHA IS DATA BASE APPROACH? EXPLAIN TYPES OF DATA BASES?

A) Database is an organized collection of data, so that it can be easily accessed and managed. You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

Types of Databases

There are various types of databases used for storing different varieties of data:



1) Centralized Database: It is the type of database that stores data at a centralized database system. It comforts the users to access the stored data from different locations through several applications. An example of a Centralized database can be Central Library that carries a central database of each library in a college/university.

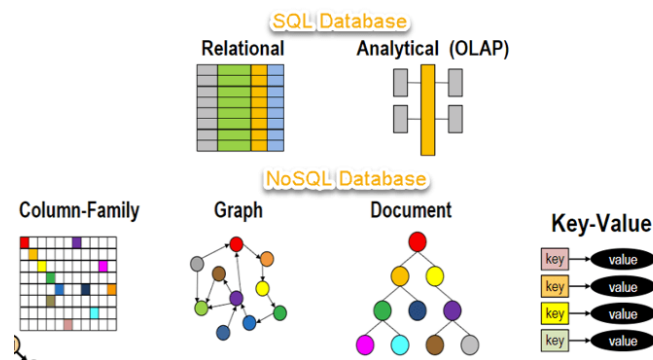
2) Distributed Database

Unlike a centralized database system, in distributed systems, data is distributed among different database systems of an organization. These database systems are connected via communication links. Such links help the end-users to access the data easily. We can further divide a distributed database system into:

- **Homogeneous DDB:** Those database systems which execute on the same operating system and use the same application process and carry the same hardware devices.
- **Heterogeneous DDB:** Those database systems which execute on different operating systems under different application procedures, and carries different hardware devices.

3) Relational Database: This database is based on the relational data model, which stores data in the form of rows(tuple) and columns(attributes), and together forms a table(relation). A relational database uses SQL for storing, manipulating, as well as maintaining the data.

4) NoSQL Database : NoSQL databases (Not Only SQL") are non-tabular databases and store data differently than relational tables.



- Key-value storage:** It is the simplest type of database storage where it stores every single item as a key (or attribute name) holding its value, together.
- Document-oriented Database:** A type of database used to store data as JSON-like document. It helps developers in storing data by using the same document-model format as used in the application code.
- Graph Databases:** It is used for storing vast amounts of data in a graph-like structure. Most commonly, social networking websites use the graph database.
- Column-Family:** It is similar to the data represented in relational databases. Here, data is stored in large columns together, instead of storing in rows.

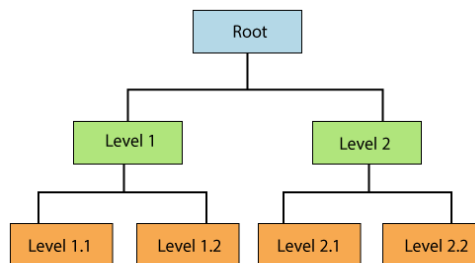
5) Cloud Database: A type of database where data is stored in a virtual environment and executes over the cloud computing platform. It provides users with various cloud computing services (SaaS, PaaS, IaaS, etc.) for accessing the database.

6) Object-oriented Databases

The type of database that uses the object-based data model approach for storing data in the database system. The data is represented and stored as objects which are similar to the objects used in the object-oriented programming language.

7) Hierarchical Databases

It is the type of database that stores data in the form of parent-children relationship nodes. Here, it organizes data in a tree-like structure. Each child record in the tree will contain only one parent. On the other hand, each parent record can have multiple child records.



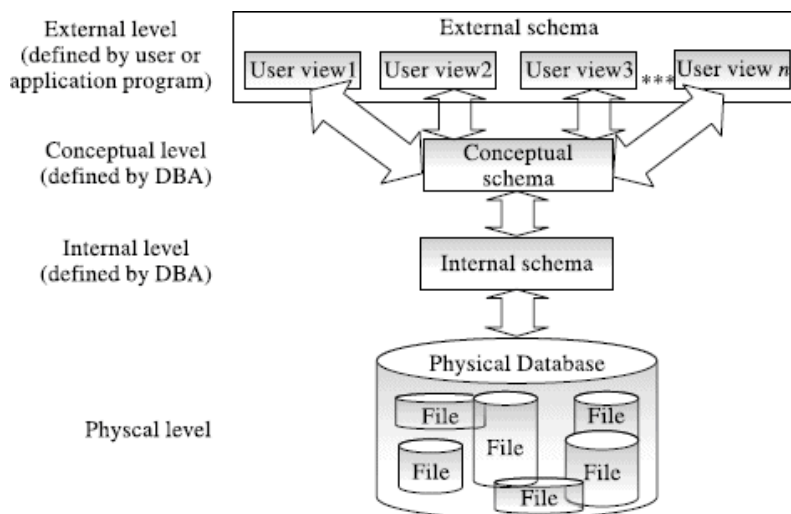
Hierarchical Database

8) Network Databases

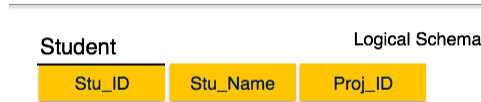
It is the database that typically follows the network data model. Here, the representation of data is in the form of nodes connected via links between them. Unlike the hierarchical database, it allows each record to have multiple children and parent nodes to form a generalized graph structure.

6) EXPLAIN THREE LEVEL ARCHITECHTURE OF DBMS or LOGICAL ARCHITECHTURE

A) DBMS Three Level Architecture Diagram



Schema: A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. A database schema defines its entities and the relationship among them.



This architecture has three levels:

1. External level
2. Conceptual level
3. Internal level

1. External level

It is also called **view level**. An external schema describes the part of the database which specific user is interested in. It hides the unrelated details of the database from the user. There may be "n" number of external views for each database.

An external view is just the content of the database as it is seen by some specific particular user. For example, a user from the sales department will see only sales related data.

2. Conceptual level

It is also called **logical level**. The conceptual schema describes the Database structure of the whole database for the community of users. This schema hides information about the physical storage structures and focuses on describing data types, entities, attributes, and their relationships etc.

This logical level comes between the user level and physical storage view. However, there is only single conceptual view of a single database.

3. Internal level

The internal schema defines the physical storage structure of the database. The internal schema is a very low-level representation of the entire database. It contains multiple occurrences of multiple types of internal record. In the ANSI term, it is also called "stored record".

Q:7) Explain physical DBMS architecture.**A) PHYSICAL DBMS ARCHITECTURE**

The physical architecture defines the software components used to process and enter data, and how these software components are related and interconnected. Though it is not possible to simplify the component structure of a DBMS, it is possible to recognize a number of key functions which are similar to most database management systems. The components that normally execute these functions are shown in Figure, which depicts the physical architecture of a typical DBMS.

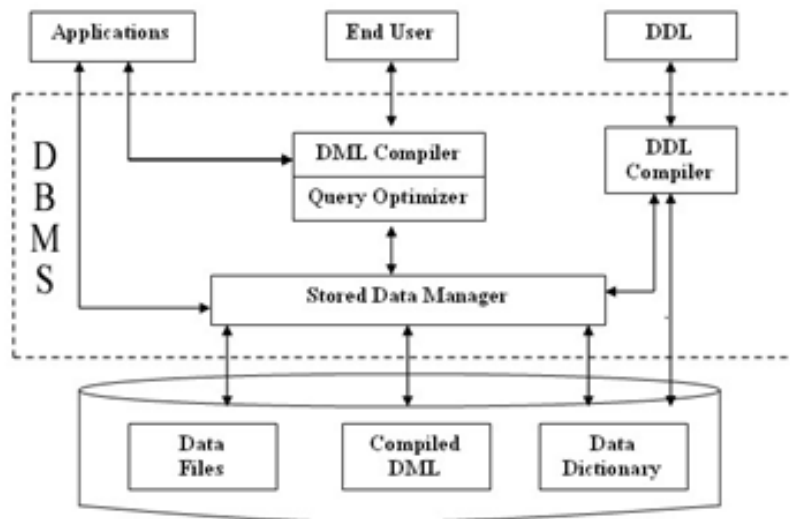


Figure: DBMS Structure

Based on many functions, the database system may be partitioned. Some functions (for example, file systems) may be given by the operating system.

- Data Definition Language(DDL) - Set of commands required to define the format of data.
- Data Manipulation Language(DML) - Set of commands that modify, process data.
- DML Precompiler - It converts DML statements embedded in an application program to normal procedural calls in the host language. It interacts with the query processor in order to generate the appropriate code.
- DDL Compiler - It converts DDL statements into a set of tables containing metadata tables – which are in a form that can be used by other components of the DBMS. These are stored in system catalog or data dictionary.
- Data Manager - Manages the allocation of space on disk storage.
- Query Processor - Responsible for receiving query language statements and changing to a form the DBMS can understand. It has two parts : parser and query optimizer.

Q8) Who is A DBA (Database Administrator)? Explain role and duties of DBA.

A) A Database Administrator is a person or a group of people who are responsible for managing all the activities related to database system. This job requires a high level of expertise by a person or group of people. There are very rare chances that only a single person can manage all the database system activities so companies always have a group of people who take care of database system.

Role, Duties and Responsibilities of database Administrator (DBA)

1. Installing and Configuration of database: DBA is responsible for installing the database software. He configures the software of database and then upgrades it if needed. There is many database software like oracle, Microsoft SQL and MySQL in the industry so DBA decides how the installing and configuring of this database software will take place.

2. Deciding the hardware device

Depending upon the cost, performance and efficiency of the hardware, it is DBA who have the duty of deciding which hardware device will suit the company requirement. It is hardware that is an interface between end users and database so it needed to be of best quality.

2. Managing Data Integrity

Data integrity should be managed accurately because it protects the data from unauthorized use. DBA manages relationship between the data to maintain data consistency.

3. Decides Data Recovery and Back up method

If any company is having a big database, then it is likely to happen that database may fail at any instance. It is requiring that a DBA takes backup of entire database in regular time span. DBA has to decide that how much data should be backed up and how frequently the back should be taken. Also, the recovery of data base is done by DBA if they have lost the database.

4. Tuning Database Performance

Database performance plays an important role for any business. If user is not able to fetch data speedily then it may loss company business. So, by tuning a modifying SQLl commands a DBA can improves the performance of database.

5. Capacity Issues

All the databases have their limits of storing data in it and the physical memory also has some limitations. DBA has to decide the limit and capacity of database and all the issues related to it.

6. Database design

The logical design of the database is designed by the DBA. Also, a DBA is responsible for physical design, external model design, and integrity control.

7. Database accessibility

DBA writes subschema to decide the accessibility of database. He decides the users of the database and also which data is to be used by which user. No user has to power to access the entire database without the permission of DBA.

8. Decides validation checks on data

DBA has to decide which data should be used and what kind of data is accurate for the company. So he always puts validation checks on data to make it more accurate and consistence.

9. Monitoring performance

If database is working properly then it doesn't mean that there is no task for the DBA. Yes f course, he has to monitor the performance of the database. A DBA monitors the CPU and memory usage.

10. Decides content of the database

A database system has many kinds of content information in it. DBA decides fields, types of fields, and range of values of the content in the database system. One can say that DBA decides the structure of database files.

11. Provides help and support to user

If any user needs help at any time then it is the duty of DBA to help him. Complete support is given to the users who are new to database by the DBA.

12. Database implementation

Database has to be implemented before anyone can start using it. So, DBA implements the database system. DBA has to supervise the database loading at the time of its implementation.

13. Improve query processing performance

Queries made by the users should be performed speedily. As we have discussed that users need fast retrieval of answers so DBA improves query processing by improving their performance. So, these were the Role, Duties and Responsibilities of database Administrator (DBA).

Q9) What is data dictionary?

A) Data Dictionary consists of database metadata. It has records about objects in the database.

Metadata is simply defined as data about data.

Meta data in Relational database:

Relational databases store and provide access not only data but also metadata in a structure called data dictionary or system catalog. It holds information about:

- tables,
- columns,
- data types,
- table relationship,
- constraints etc.

Data Dictionary consists of

Data Dictionary consists of the following information –

- Name of the tables in the database
- Constraints of a table i.e. keys, relationships, etc.
- Columns of the tables that related to each other
- Owner of the table
- Last accessed information of the object
- Last updated information of the object

An example of Data Dictionary can be personal details of a student –

Example**<StudentPersonalDetails>**

Student_ID	Student_Name	Student_Address	Student_City
-------------------	---------------------	------------------------	---------------------

The following is the data dictionary for the above fields –

Field Name	Datatype	Field Length	Constraint	Description
Student_ID	Number	5	Primary Key	Student id
Student_Name	Varchar	20	Not Null	Name of the student
Student_Address	Varchar	30	Not Null	Address of the student
Student_City	Varchar	20	Not Null	City of the student

Q10) What is Data file indices in DBMS ?

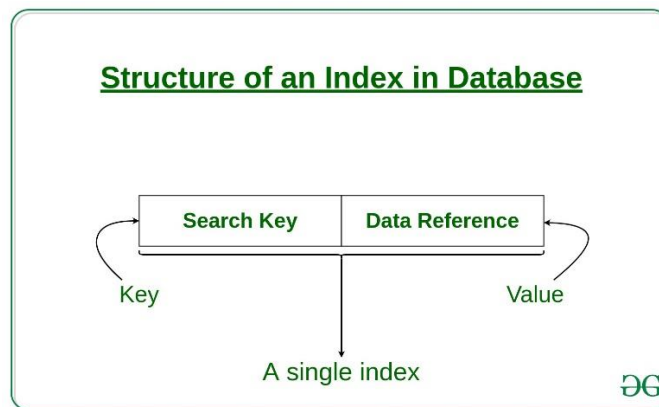
A) Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database.

Indexes are created using a few database columns.

- The first column is the **Search key** that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly.

Note: The data may or may not be stored in sorted order.

- The second column is the **Data Reference** or **Pointer** which contains a set of pointers holding the address of the disk block where that particular key value can be found.



The indexing has various attributes:

- Access Types:** This refers to the type of access such as value based search, range access, etc.
- Access Time:** It refers to the time needed to find particular data element or set of elements.
- Insertion Time:** It refers to the time taken to find the appropriate space and insert a new data.
- Deletion Time:** Time taken to find an item and delete it as well as update the index structure.
- Space Overhead:** It refers to the additional space required by the index.

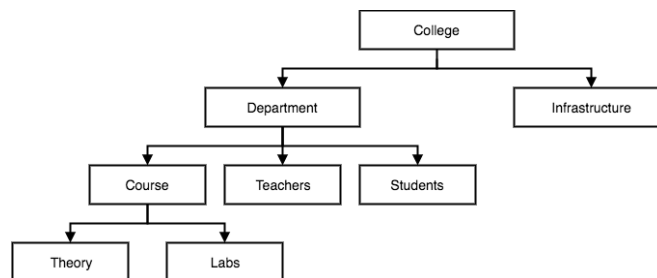
Q11) What is data model in DBMS?

A) 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

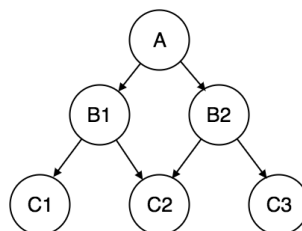
In hierarchical model, data is organized into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students. In this model, a child node will only have a single parent node.

**Network Model**

This is an extension of the Hierarchical model. In this model data is organized more like a graph, and **are allowed to have more than one parent node.**

In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.

This was the most widely used database model, before Relational Model was introduced.

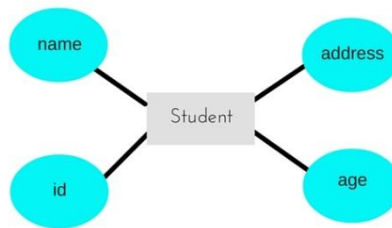
**Entity-relationship Model**

E-R Models are defined to represent the relationships into pictorial form to make it easier for different users to understand.

This model is good to design a database, which can then be turned into tables in relational model(explained below).

Let's take an example, If we have to design a School Database, then **Student** will be an **entity** with **attributes** name, age, address etc. As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.

Relationships can also be of different types. To learn about [E-R Diagrams](#) in details, click on the link.



Relational Model

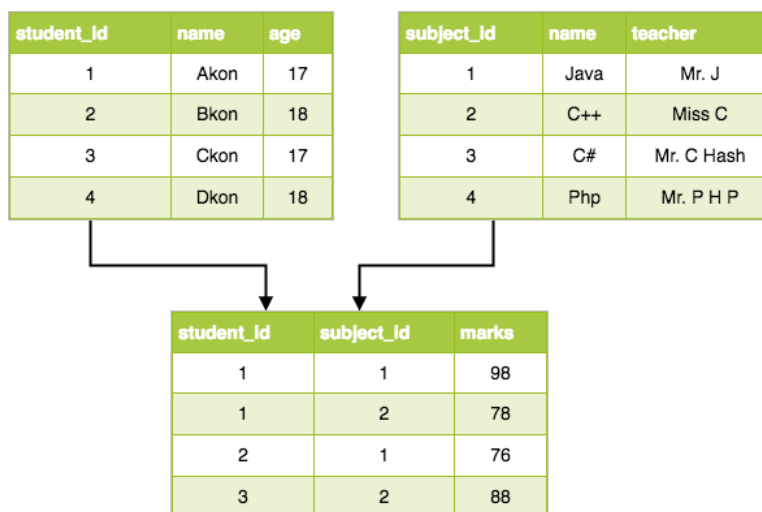
In this model, data is organized in two-dimensional **tables** and the relationship is maintained by storing a common field.

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.

The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.

Hence, tables are also known as **relations** in relational model.

In the coming tutorials we will learn how to design tables, normalize them to reduce data redundancy and how to use Structured Query language to access data from tables.



Relational Data Model

Q12) What is Relational Data Model? Briefly explain about Relational Data Model.

A) The relational data model was introduced by **C. F. Codd in 1970**. Currently, it is the most widely used data model. The relational data model describes the world as “a collection of inter-related relations (or tables).”

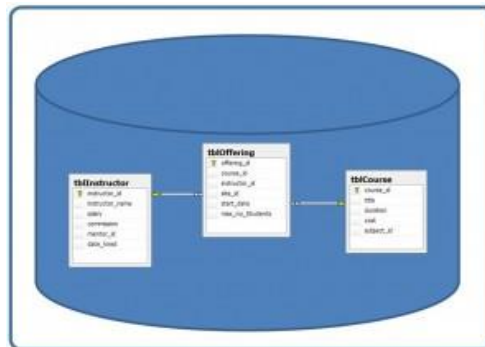
Relation

A **relation**, also known as a table or *file*, is a subset of the Cartesian product of a list of domains characterized by a name. And within a table, each row represents a group of related data values. A *row*, or record, is also known as a **tuple**. The columns in a table is a field and is also referred to as an **attribute**. You can also think of it this way: an attribute is used to define the record and a record contains a set of attributes.

Sno	Sname	course
001	Abc	comp

Table

A database is composed of multiple tables and each table holds the data. Following Figure shows a database that contains three tables.



Column

A database stores pieces of information or facts in an organized way. Understanding how to use and get the most out of databases requires us to understand that method of organization.

The principal storage units are called columns or fields *or* attributes. This is how the basic components of data into which your content can be broken down. When deciding which fields to create, you need to think generically about your information, for example, drawing out the common components of the information that you will store in the database and avoiding the specifics that distinguish one item from another.

Look at the example of an ID card in Figure 7.2 to see the relationship between fields and their data.

Field Name	Data
First Name	Isabelle
Family Name	Whelan
Nationality	British
Salary	109,900
Date of Birth	15 September 1983
Marital Status	Single
Shift	Mon, Wed
Place of issue	Addis Ababa
Valid until	17 December 2003

Domain

A *domain* is the original sets of atomic values used to model data. By *atomic value*, we mean that each value in the domain is indivisible as far as the relational model is concerned. For example:

- The domain of Marital Status has a set of possibilities: Married, Single, Divorced.
- The domain of Shift has the set of all possible days: {Mon, Tue, Wed...}.
- The domain of Salary is the set of all floating-point numbers greater than 0 and less than 200,000.
- The domain of First Name is the set of character strings that represents names of people.

In summary, a domain is a set of acceptable values that a column is allowed to contain. This is based on various properties and the data type for the column

Records

Just as the content of any one document or item needs to be broken down into its constituent bits of data for storage in the fields, the link between them also needs to be available so that they can be reconstituted into their whole form. Records allow us to do this. *Records* contain fields that are related, such as a customer or an employee. As noted earlier, a tuple is another term used for record.

Records and fields form the basis of all databases. A simple table gives us the clearest picture of how records and fields work together in a database storage project.

Record ID	PubDate	Author	Title
1	26/07/1968	B. Pitt	Rights and Wrongs online
2	3/5/2000	A. Jolie	Networking for Change
3	27/02/1971	J. Carter	The Myth of Cyber Crimes
4	15/09/1983	I. Wheaton	Connecting the disconnected

Figure. Example of a simple table by A. Watt.

Degree

The *degree* is the number of attributes in a table. In our example in Figure, the degree is 4.

Cardinality

The number of tuples in a relation is known as cardinality. The In our example defined above has cardinality 4.

Properties of a Table

- A table has a name that is distinct from all other tables in the database.
- There are no duplicate rows; each row is distinct.
- Entries in columns are atomic. The table does not contain repeating groups or multivalued attributes.
- Entries from columns are from the same domain based on their data type including:
 - number (numeric, integer, float, smallint,...)
 - character (string)
 - date
 - logical (true or false)
- Operations combining different data types are disallowed.
- Each attribute has a distinct name.
- The sequence of columns is insignificant.
- The sequence of rows is insignificant.

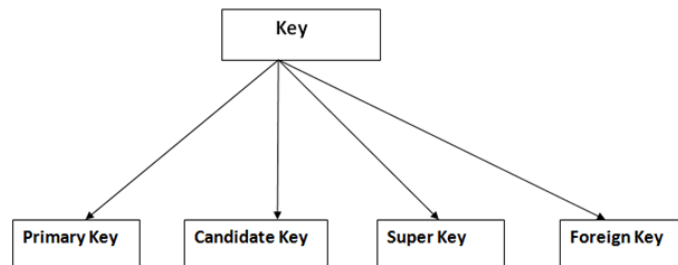
Q13) What is a Key? Explain types of keys.**A) Keys**

- Keys play an important role in the relational database.
- It is used to identify any record or row of data uniquely from the table. It is also used to establish and identify relationships between tables.

For example: In Student table, ID is used as a key because it is unique for each student. In PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

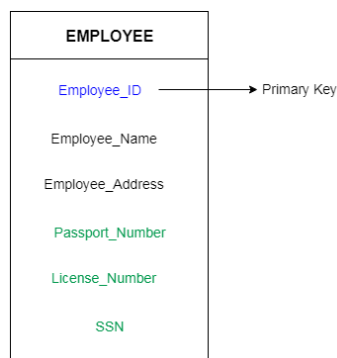
STUDENT	PERSON
ID	Name
Name	DOB
Address	Passport_Number
Course	License_Number
	SSN

Types of key:



1. Primary key

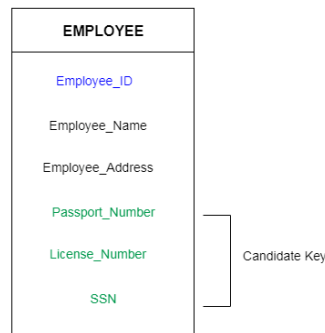
- It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.
- In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary key since they are also unique.
- For each entity, selection of the primary key is based on requirement and developers.



2. Candidate key

- A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
- The remaining attributes except for primary key are considered as a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. Rest of the attributes like SSN, Passport_Number, and License_Number, etc. are considered as a candidate key.



3. Super Key

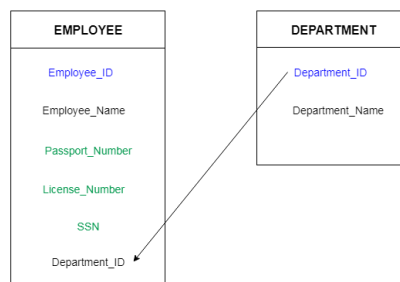
Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

For example: In the above EMPLOYEE table, for (EMPLOYEE_ID, EMPLOYEE_NAME) the name of two employees can be the same, but their EMPLOYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID, (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key

- Foreign keys are the column of the table which is used to point to the primary key of another table.
- In a company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id as a new attribute in the EMPLOYEE table.
- Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



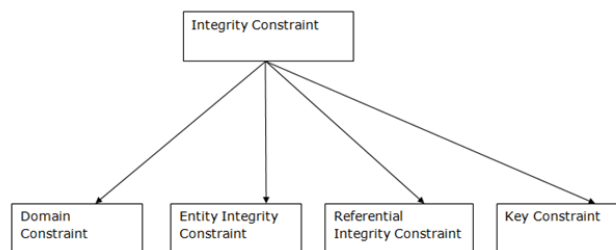
Types of constraints

Q14) Explain types of Integrity Constraints in DBMS.

A) Integrity Constraints

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

Types of Integrity Constraint



1. Domain constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1 st	17
1001	Johnson	2 nd	24
1002	Leonardo	5 th	21
1003	Kate	3 rd	19
1004	Morgan	8 th	A

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:

EMPLOYEE

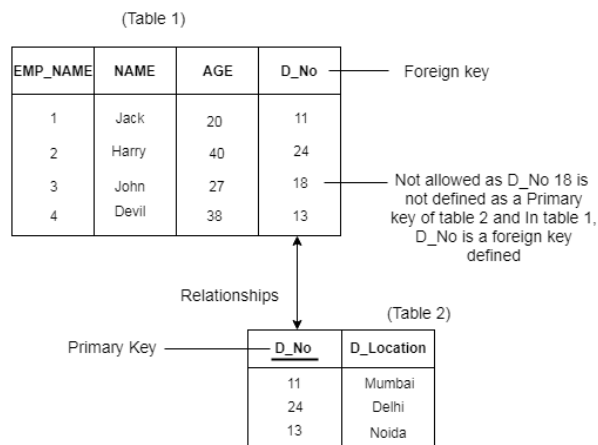
EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:



4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example:

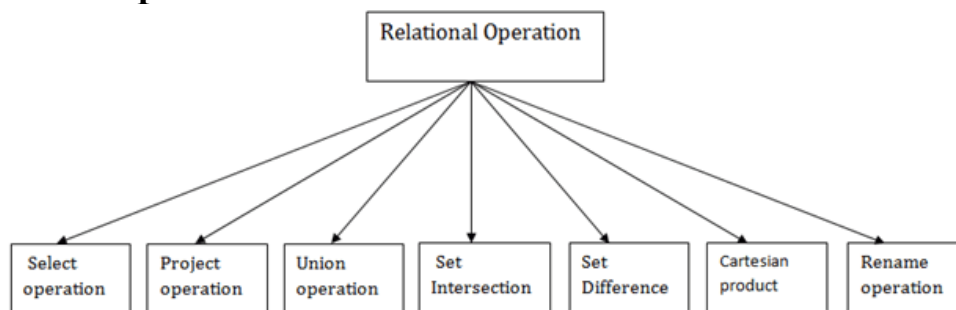
ID	NAME	SEMENSTER	AGE
1000	Tom	1 st	17
1001	Johnson	2 nd	24
1002	Leonardo	5 th	21
1003	Kate	3 rd	19
1002	Morgan	8 th	22

Not allowed. Because all row must be unique

Q 15) Explain Relational Algebra in detail. / List out Relational Operators

A) Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

Types of Relational operation



1. Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).

1. Notation: $\sigma p(r)$

Where:

σ is used for selection prediction

r is used for relation

p is used as a propositional logic formula which may use connectors like: AND OR and NOT.

These relational can use as relational operators like $=, \neq, \geq, <, >, \leq$.

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

1. $\sigma \text{ BRANCH_NAME} = \text{"perryride"} (\text{LOAN})$

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.
- It is denoted by Π .

1. Notation: $\Pi A_1, A_2, A_n (r)$

Where

A₁, A₂, A₃ is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

1. $\Pi \text{NAME, CITY (CUSTOMER)}$

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn

Brooks	Brooklyn
--------	----------

3. Union Operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by $\cup$.

1. Notation: $R \cup S$

A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

Example:**DEPOSITOR RELATION**

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

Input:

- Π CUSTOMER_NAME (BORROW) $\cup$ Π CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection $\cap$.

1. Notation: $R \cap S$

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. $\Pi \text{ CUSTOMER_NAME (BORROW)} \cap \Pi \text{ CUSTOMER_NAME (DEPOSITOR)}$

Output:

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).

1. Notation: R - S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. Π CUSTOMER_NAME (BORROW) - Π CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.

1. Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

1. EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

7. Rename Operation:

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

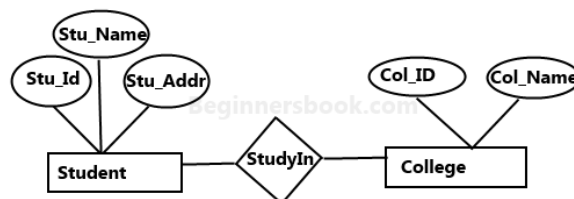
1. $\rho(\text{STUDENT1}, \text{STUDENT})$

Q16) What is an Entity Relationship Diagram (ER Diagram)?

A) An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.

A simple ER Diagram:

In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name.



Sample E-R Diagram

Here are the geometric shapes and their meaning in an E-R Diagram. We will discuss these terms in detail in the next section(Components of a ER Diagram) of this guide so don't worry too much about these terms now, just go through them once.

Rectangle: Represents Entity sets.

Ellipses: Attributes

Diamonds: Relationship Set

Lines: They link attributes to Entity Sets and Entity sets to Relationship Set

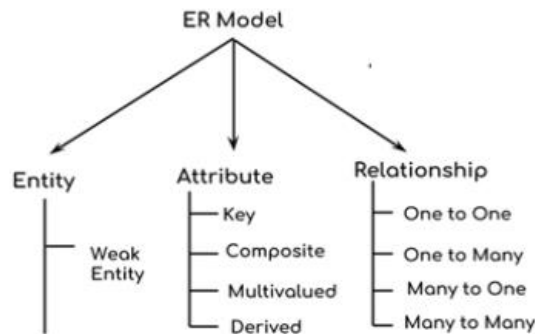
Double Ellipses: Multivalued Attributes

Dashed Ellipses: Derived Attributes

Double Rectangles: Weak Entity Sets

Double Lines: Total participation of an entity in a relationship set

Components of a ER Diagram



Components of ER Diagram

As shown in the above diagram, an ER diagram has three main components:

1. Entity
2. Attribute
3. Relationship

1. Entity

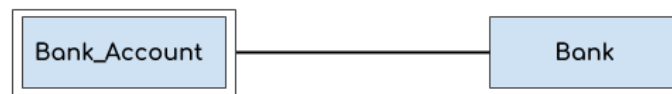
An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.



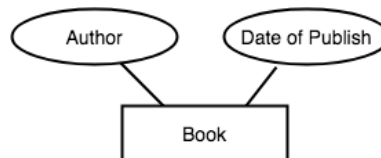
For example: In the following ER diagram we have two entities Student and College and these two entities have many to one relationship as many student's study in a single college. We will read more about relationships later, for now focus on entities.

**Weak Entity:**

An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. 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.

**2. Attribute****Q) What is Attribute in ER model? Explain types of Attribute.**

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram.



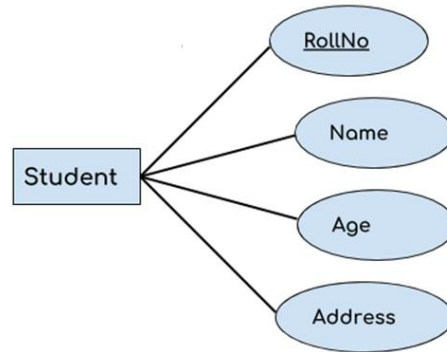
There are four types of attributes:

1. Key attribute
2. Composite attribute
3. Multivalued attribute
4. Derived attribute

1. Key attribute:

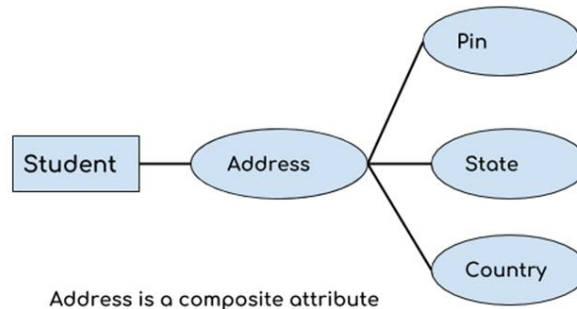
A key attribute can uniquely identify an entity from an entity set. For example, student roll number can uniquely identify a student from a set of students. Key attribute is represented by oval same as other attributes however the **text of key attribute is underlined**.





2. Composite attribute:

An attribute that is a combination of other attributes is known as composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.



3. Multivalued attribute:

An attribute that can hold multiple values is known as multivalued attribute. It is represented with **double ovals** in an ER Diagram.



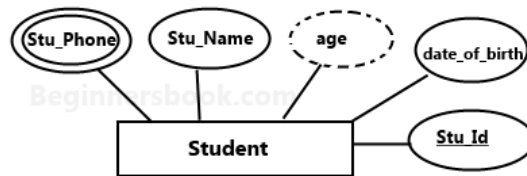
For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

4. Derived attribute:

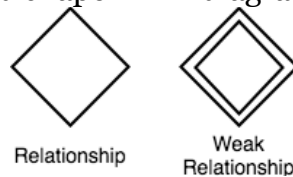
A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by **dashed oval** in an ER Diagram.



For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).

E-R diagram with multivalued and derived attributes:**3. Relationship****Q) What is Relationship in ER model? Explain types of relationship.**

A relationship is represented by diamond shape in ER diagram,



It shows the relationship among entities. There are four types of relationships:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

1. One to One Relationship

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.

**2. One to Many Relationship**

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.

**3. Many to One Relationship**

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.



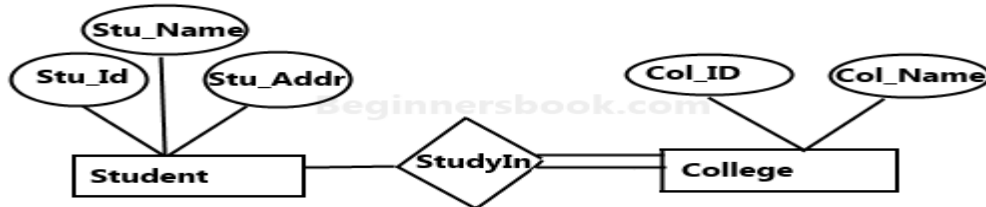
4. Many to Many Relationship

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.



Q:17) Explain Total Participation of an Entity set

A Total participation of an entity set represents that each entity in entity set must have at least one relationship in a relationship set. For example: In the below diagram each college must have at-least one associated Student.



E-R Digram with total participation of College entity set in StudyIn relationship Set - This indicates that each college must have atleast one associated Student.

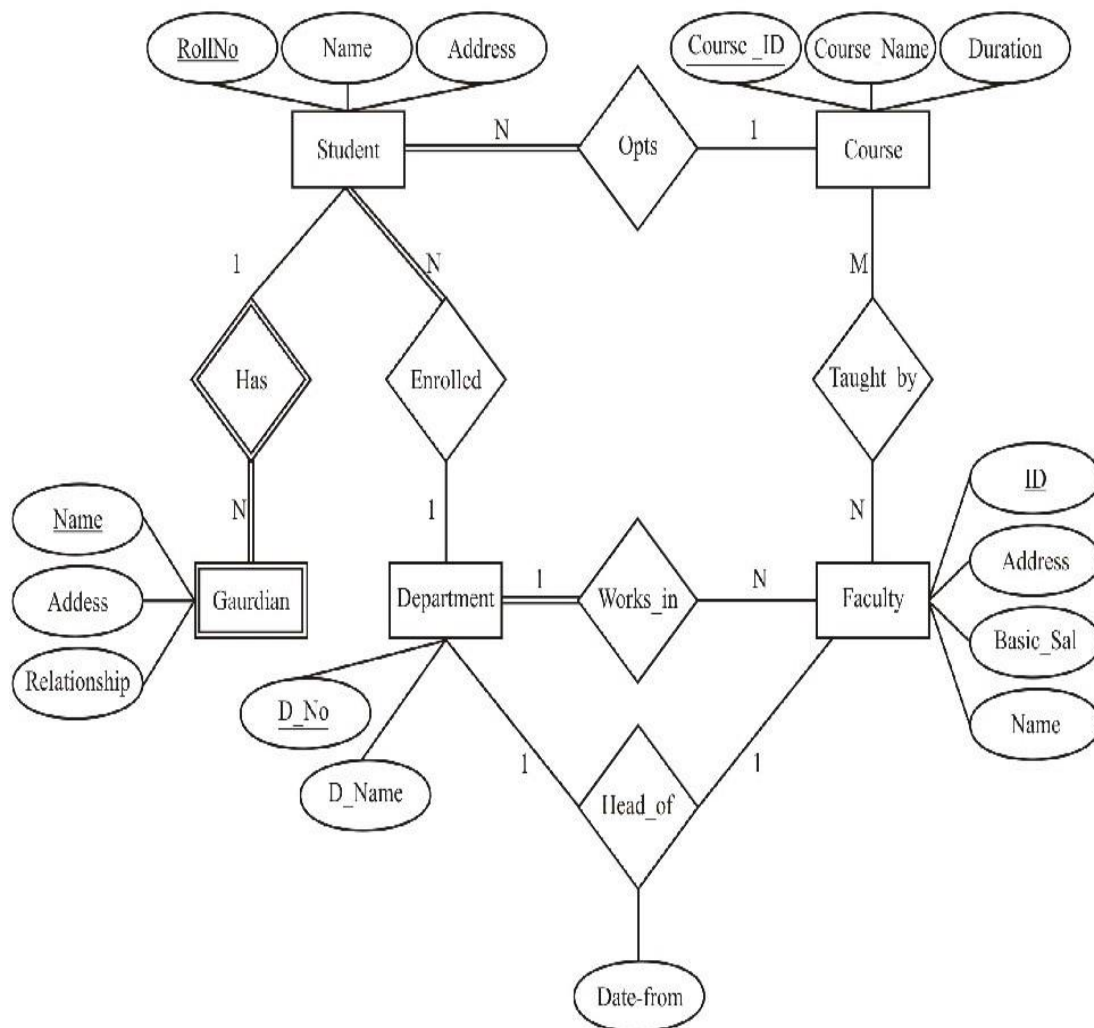
Q 18) Defining Relationship for College Database

Using the concepts defined earlier, we have identified that strong entities in COLLEGE database are STUDENT, FACULTY, COURSE and DEPARTMENT. This database also has one weak entity called GUARDIAN. We can specify the following relationships:

- **Head-of**, is a 1:1 relationship between FACULTY and DEPARTMENT. Participation of the entity FACULTY is partial since not all the faculty members participate in this relationship, while the participation from department side is total, since every department has one head.
- **Works_in**, is a 1:N relationship between DEPARTMENT and FACULTY. Participation from both side is total.

- **Opts**, is a 1:N relationship between COURSE and STUDENT. Participation from student side is total because we are assuming that each student enrolled opts for a course. But the participation from the course side is partial, since there can be courses that no student has opted for.
 - **Taught_by**, is a M: N relationship between FACULTY and COURSE, as a faculty can teach many courses and a course can be taught by many faculty members.
 - **Enrolled**, is a 1:N relationship between STUDENT and DEPARTMENT as a student is allowed to enroll for only one department at a time.
 - **Has**, is a 1:N relationship between STUDENT and GUARDIAN as a student can have more than one local guardian and one local guardian is assumed to be related to one student only. The weak entity Guardian has total participation in the relation “Has”.
- So now, let us make an E-R diagram for the college database in the next section.

E-R DIAGRAM



Q19) Illustrate Converting ER Diagrams to Tables with examples.

A) After designing an **ER Diagram**,

- ER diagram is converted into the tables in relational model.
- This is because relational models can be easily implemented by RDBMS like MySQL , Oracle etc.

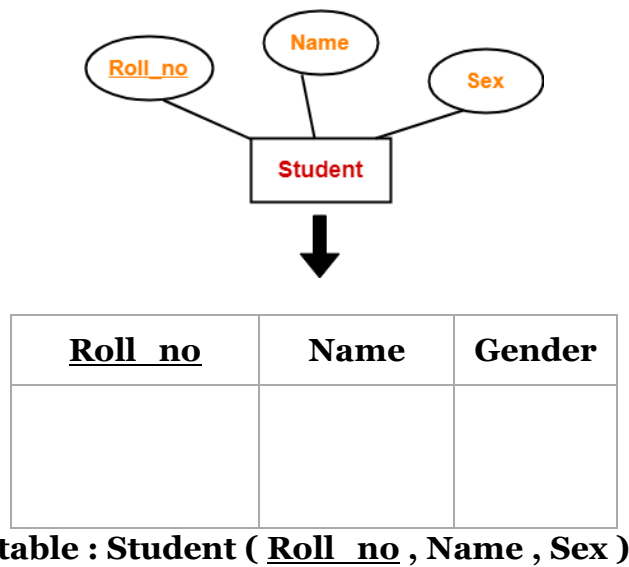
Following rules are used for converting an ER diagram into the tables-

Rule-01: For Strong Entity Set with Only Simple Attributes-

A strong entity set with only simple attributes will require only one table in relational model.

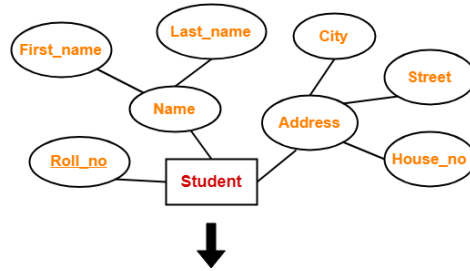
- Attributes of the table will be the attributes of the entity set.
- The primary key of the table will be the key attribute of the entity set.

Example-

**Rule-02: For Strong Entity Set with Composite Attributes-**

- A strong entity set with any number of composite attributes will require only one table in relational model.
- While conversion, simple attributes of the composite attributes are considered and not the composite attribute itself.

Example-



<u>Roll_no</u>	First_name	Last_name	House_no	Street	City

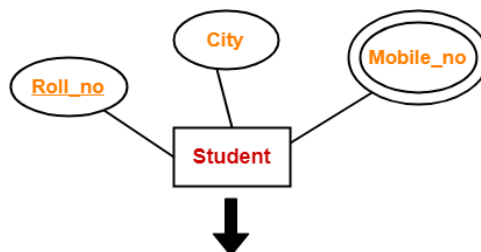
Schema : Student (Roll_no , First_name , Last_name , House_no , Street , City)

Rule-03: For Strong Entity Set With Multi Valued Attributes-

A strong entity set with any number of multi valued attributes will require two tables in relational model.

- One table will contain all the simple attributes with the primary key.
- Other table will contain the primary key and all the multi valued attributes.

Example-



<u>Roll_no</u>	City

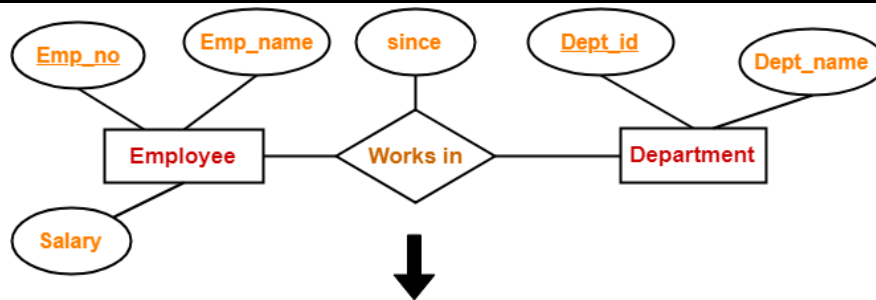
<u>Roll_no</u>	Mobile_no

Rule-04: Translating Relationship Set into a Table-

A relationship set will require one table in the relational model. Attributes of the table are-

- Primary key attributes of the participating entity sets
- Its own descriptive attributes if any. Set of non-descriptive attributes will be the primary key.

Example-



<u>Emp no</u>	<u>Dept id</u>	since

table: Works in (Emp no , Dept id , since)

NOTE-

If we consider the overall ER diagram, three tables will be required in relational model-

- One table for the entity set "Employee"
- One table for the entity set "Department"
- One table for the relationship set "Works in"

Rule-05: For Binary Relationships With Cardinality Ratios-

The following four cases are possible-

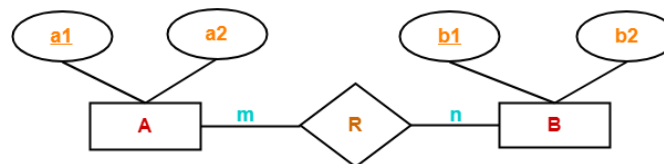
Case-01: Binary relationship with cardinality ratio m:n

Case-02: Binary relationship with cardinality ratio 1:n

Case-03: Binary relationship with cardinality ratio m:1

Case-04: Binary relationship with cardinality ratio 1:1

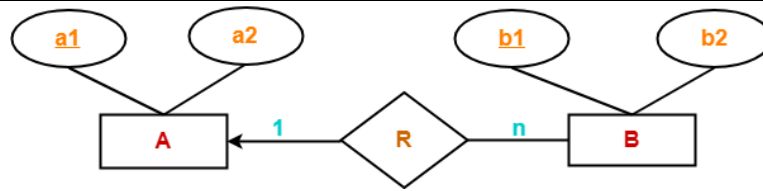
Case-01: For Binary Relationship With Cardinality Ratio m:n



Here, three tables will be required-

1. A (a1 , a2)
2. R (a1 , b1)
3. B (b1 , b2)

Case-02: For Binary Relationship With Cardinality Ratio 1:n

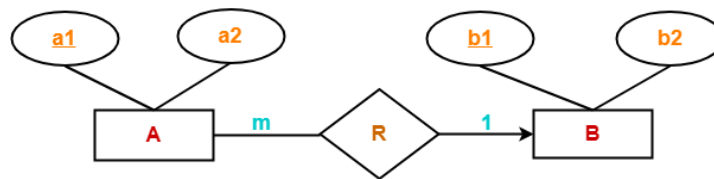


Here, two tables will be required-

1. A (a1 , a2)
2. BR (a1 , b1 , b2)

NOTE- Here, combined table will be drawn for the entity set B and relationship set R.

Case-03: For Binary Relationship With Cardinality Ratio m:1

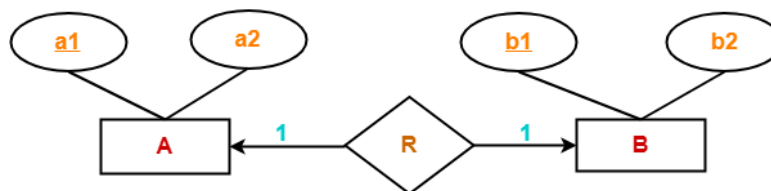


Here, two tables will be required-

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

NOTE- Here, combined table will be drawn for the entity set A and relationship set R.

Case-04: For Binary Relationship With Cardinality Ratio 1:1



Here, two tables will be required. Either combine 'R' with 'A' or 'B'

Way-01:

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

Way-02:

1. A (a1 , a2)
2. BR (a1 , b1 , b2)

UNIT -II

**DATABASE INTEGRITY AND
NORMALISATION**

SYLLABUS

Relational Database Integrity - The Keys - Referential Integrity - Entity Integrity - Redundancy and Associated Problems – Single Valued Dependencies – Normalisation - Rules of Data Normalisation - The First Normal Form - The Second Normal Form - The Third Normal Form - Boyce Codd Normal Form - Attribute Preservation - Lossless-join Decomposition - Dependency Preservation.

File Organisation : Physical Database Design Issues - Storage of Database on Hard Disks - File Organisation and Its Types - Heap files (Unordered files) - Sequential File Organisation - Indexed (Indexed Sequential) File Organisation - Hashed File Organisation - Types of Indexes - Index and Tree Structure - Multi-key File Organisation - Need for Multiple Access Paths - Multi-list File Organisation - Inverted File Organisation.

Q1) What is Integrity Constraints? Explain types of Integrity Constraints?

A) Integrity Constraints

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

Types of Integrity Constraint

i) Entity Integrity Constraints

ii) Referential Integrity Constraints

i) Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

ii) Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:

(Table 1)

EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

Foreign key

Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined

Relationships

(Table 2)

Primary Key	D_No	D_Location
	11	Mumbai
	24	Delhi
	13	Noida

Q 2) Explain the Problem of redundancy in Database in detail.

A) Redundancy means having multiple copies of same data in the database. This problem arises when a database is not normalized. Suppose a table of student details attributes are: student Id, student name, college name, college rank, course opted.

Student_ID	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU	Btech	1
101	Ankit	7900734858	GEU	Btech	1
102	Aysuh	7300936759	GEU	Btech	1
103	Ravi	7300901556	GEU	Btech	1

As it can be observed that values of attribute college name, college rank, course is being repeated which can lead to problems. Problems caused due to redundancy are: **Insertion anomaly**, **Deletion anomaly**, and **Updation anomaly**.

InsertionAnomaly –

An Insert Anomaly occurs when certain attributes (column) cannot be inserted into the database without the presence of the other attributes (columns).

Table

CourseNo	Course Name	Room No	Room Capacity	Teacher Name
401	General	5	50	Raj
402	Computers	6	60	Ram
405	Comp.Applications	7	70	Anthony
407	Honors	8	80	Ravi

Example:

College has built a new room (e.g. 9) but it has not yet been scheduled for any course (or) members of staff. We can't insert New Room No 9, if it is not allotted to a Course, and Faculty.

Delete anomaly

A Delete Anomaly exists when certain attributes are lost because of the deletion of other attributes.

CourseNo	Course Name	Room No	Room Capacity	Teacher Name
401	General	5	50	Raj
402	Computers	6	60	Ram
405	Comp.Applications	7	70	Anthony
407	Honors	8	80	Ravi

Example:

If we remove the course no: 402 from the above table, the details of Room No 6 get deleted. Which implies the corresponding course will also get deleted.

Update Anomaly

An Update Anomaly exists when one or more places duplicated data is updated, but not all.

CourseNo	Course Name	Room No	Room Capacity	Teacher Name
401	General	5	50	Raj
402	Computers	6	60	Ram
405	Comp.Applications	7	70	Anthony
407	Honors	8	80	Ravi

Example:

Room 7 has been improved, it is now of Room Capacity 90. For updating a single entity (Table), we have to update all other columns where Room No 7.

Q3) What is Functional Dependency? Explain types of functional dependency?

A) A functional dependency is a constraint that specifies the relationship between two sets of attributes where one set can accurately determine the value of other sets. It is denoted as $X \rightarrow Y$, where X is a set of attributes that is capable of determining the value of Y . The attribute set on the left side of the arrow, X is called **Determinant**, while on the right side, Y is called the **Dependent**.

Types of Functional dependencies in DBMS:

1. Trivial functional dependency
2. Non-Trivial functional dependency
3. Multivalued functional dependency
4. Transitive functional dependency

1. Trivial Functional Dependency

In **Trivial Functional Dependency**, a dependent is always a subset of the determinant. i.e. If $X \rightarrow Y$ and Y is the subset of X , then it is called trivial functional dependency

For example,

Roll no.	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, $\{\text{roll_no, name}\} \rightarrow \text{name}$ is a trivial functional dependency, since the dependent **name** is a subset of determinant set $\{\text{roll_no, name}\}$. Similarly, $\text{roll_no} \rightarrow \text{roll_no}$ is also an example of trivial functional dependency.

2. Non-trivial Functional Dependency

In **Non-trivial functional dependency**, the dependent is strictly not a subset of the determinant.

i.e. If $X \rightarrow Y$ and Y is not a subset of X , then it is called Non-trivial functional dependency.

For example,

Roll no	name	age
42	abc	17
43	pqr	18
44	Xyz	18

Here, $\text{roll_no} \rightarrow \text{name}$ is a non-trivial functional dependency, since the dependent **name** is not a subset of determinant **roll_no**. Similarly, $\{\text{roll_no, name}\} \rightarrow \text{age}$ is also a non-trivial functional dependency, since **age** is not a subset of $\{\text{roll_no, name}\}$.

3. Multivalued Functional Dependency

In **Multivalued functional dependency**, entities of the dependent set are **not dependent on each other**. i.e. If $a \rightarrow \{b, c\}$ and there exists **no functional dependency** between **b** and **c**, then it is called a **multivalued functional dependency**.

For example,

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18
45	abc	19

Here, $\text{roll_no} \rightarrow \{\text{name}, \text{age}\}$ is a multivalued functional dependency, since the dependents **name** & **age** are **not dependent** on each other (i.e. $\text{name} \rightarrow \text{age}$ or $\text{age} \rightarrow \text{name}$ doesn't exist!)

4. Transitive Functional Dependency

In transitive functional dependency, dependent is indirectly dependent on determinant. i.e. If $a \rightarrow b$ & $b \rightarrow c$, then according to axiom of transitivity, $a \rightarrow c$. This is a **transitive functional dependency**

For example,

Rollno.	name	dept	building_no
42	abc	CO	4
43	pqr	EC	2
44	xyz	IT	1
45	abc	EC	2

Here, $\text{enrol_no} \rightarrow \text{dept}$ and $\text{dept} \rightarrow \text{building_no}$,

Hence, according to the axiom of transitivity, $\text{enrol_no} \rightarrow \text{building_no}$ is a valid functional dependency. This is an indirect functional dependency, hence called Transitive functional dependency.

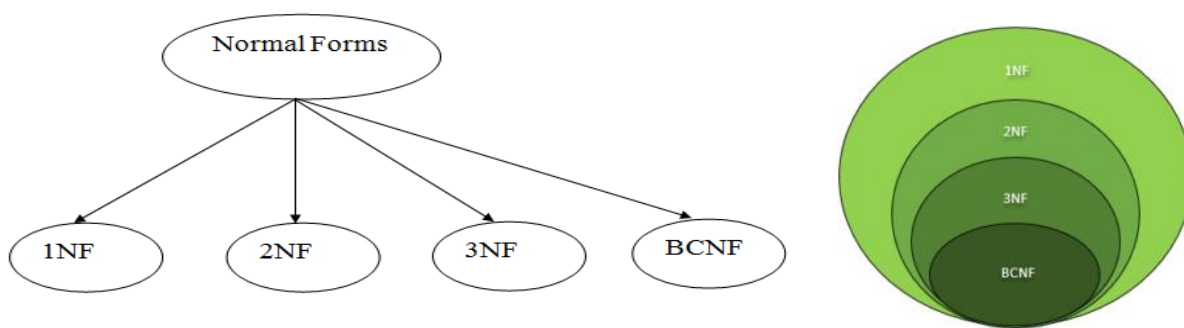
Q4)What is Normalization? Write about types of Normalization.

Normalization is the process of minimizing **redundancy** from a relation or set of relations. Redundancy in relation may cause **insertion, deletion, and update anomalies**. So, it helps to minimize the redundancy in relations.

Or

“Normalization is a process of **decomposing** a relation (Table) into two (or) more relations with specific relationships being set up between the tables, so that all **insertion, deletion, and update anomalies** are removed.

There are the four types of normal forms:



1. First Normal Form –

If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is **singled valued attribute**.

Example 1 – Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD_PHONE. Its decomposition into 1NF has been shown in table 2.

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	INDIA
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

2. Second Normal Form (2NF)

A relation is in second normal form if it satisfies the following conditions:

- A relation must be in first normal form (1NF)
- Every non key attribute must be FULLY functionally dependent on the primary key attributes.

EMPPROJECT

EMPID	EMPNAME	PROJID	PROJNAME
1	Mohan	101	Highway Road
2	Deepthi	102	Shopping
3	Deepthi	103	Metro
4	Kranthi	104	College Building
5	Shakthi	101	Highway Road
6	Shakthi	103	Metro

- WE see here in EMPPROJECT table the prime key attributes are EmpID and ProjID.
- According to the rule, non-key attributes, i.e, EmpName and ProjName must be dependent upon both prime key attribute together.
- But we find that EmpName can be recognized by EmpID and ProjName can be recognized by ProjID independently. This is called partial dependency, which is not allowed in Second Normal Form, we need fully functional Dependency.
- So we are going to decompose (split) the EMPROJ table into two tables as EMPLOYEE and PROJECT.

PROJECT

EMPLOYEE

PROJID	PROJNAME
101	Highway Road
102	Shopping
103	Metro
104	College Building
101	Highway Road
103	Metro

EMPNAME	PROJID	PROJNAME
Mohan	101	Highway
Deepthi	102	Shopping
Deepthi	103	Metro
Kranthi	104	College
Shakthi	101	Highway
Shakthi	103	Metro

We broke the relation in 2 as described in the above picture. So there exists no partial dependency i.e it is now fully functional dependent.

3. Third Normal Form (3NF)

A Relation in the third normal form (3NF) if it satisfies the following conditions:

- A relation must be in second normal form (2NF)
- There should not be any transitive dependencies.

“A transitive dependency is a type of functional dependency in which the value in a non-key field is determined by the value in another field.

- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

Table Name:-STUDENT

StuID	studName	City	Pincode
1	Ali	Chennai	50001
2	John	Bangalore	50002
3	Ram	Hyderabad	50003

We find that in the above STUDENT relation (Table), StuID is the key and only prime key attribute.

We find that City can be identified by StuID as well as Pin itself.

StuID → Pincode and Pincode → City, so there exists transitive dependency.

To bring this relation into third normal form, we break the relation into two relations as follows

Table name: Student**table name :Pincode**

StuID	studName	Pincode
1	Ali	50001
2	John	50002
3	Ram	50003

Pincode	City
50001	Chennai
50002	Bangalore
50003	Hyderabad

Boyce Codd normal form (BCNF)

BCNF (Boyce Codd Normal Form) is the advanced version of 3NF. A table is in BCNF if every functional dependency $X \rightarrow Y$, X is the super key of the table.

For BCNF,

- The table should be in 3NF, and
- For every Functional Dependency LHS is super key.

Example Consider a relation R with attributes (student, subject, teacher).

Student	Teacher	Subject
Jhansi	P.Naresh	Database
Jhansi	K.Das	C
Subbu	P.Naresh	Database
Subbu	R.Prasad	C

F: { (student, Teacher) -> subject

(student, subject) -> Teacher

Teacher -> subject}

Here Candidate keys are (student, teacher) and (student, subject).

The above relation is not in BCNF, because in the FD (teacher->subject), teacher is not a key. This relation suffers with anomalies –

For example, if we try to delete the student Subbu, we will lose the information that R. Prasad teaches C. These difficulties are caused by the fact the teacher is determinant but not a candidate key.

So make Decomposition of above table

Teacher-> subject violates BCNF [since teacher is not a candidate key].

If X->Y violates BCNF then divide R into R1(X, Y) and R2(R-X).

So R is divided into two relations R1(Teacher, subject) and R2(student, Teacher).

R1

Teacher	Subject
P.Naresh	Database
K.DAS	C
R.Prasad	C

R2

Student	Teacher
Jhansi	P.Naresh
Jhansi	K.Das
Subbu	P.Naresh
Subbu	R.Prasad

All the anomalies which were present in R, now removed in the above two relations.

Note :

BCNF decomposition does not always satisfy dependency preserving property. After BCNF decomposition if dependency is not preserved then we have to decide whether we want to remain in BCNF or rollback to 3NF. This process of rollback is called denormalization.

Q 5) Write about physical database design issues.

A) The database design involves the process of logical design with the help of E-R diagram, normalization, etc., followed by the physical design.

The Key issues in the Physical Database Design are:

- The purpose of physical database design is to translate the logical description of data into the technical specifications for storing and retrieving data for the DBMS.
- The goal is to create a design for storing data that will provide adequate performance and ensure database integrity, security and recoverability.

Some of the basic inputs required for Physical Database Design are:

- Normalised relations
- Attribute definitions
- Data usage: entered, retrieved, deleted, updated

- Requirements for security, backup, recovery, retention, integrity
- DBMS characteristics.
- Performance criteria such as response time requirement with respect to volume estimates.

The issues relating to the Design of the Physical Database Files

Physical File is a file as stored on the disk. The main issues relating to physical files are:

- Constructs to link two pieces of data:
 - Sequential storage.
 - Pointers.
- File Organisation: How the files are arranged on the disk?
- Access Method: How the data can be retrieved based on the file Organisation?

Attribute Preservation – Lossless Decomposition**Attribute Preservation**

For every decomposition, it is necessary that all the attributes belonging to the relation to be decomposed must correspond to the attributes of the fragments.

This condition is used to ensure that no attribute gets skipped or lost.

Lossless-join Decomposition

Lossless-join decomposition is one of the properties of decompositions. This dependency is also called as nonadditive or nonloss join dependency. In a more specific way, lossless-join dependency can be defined as the one which generates no additional tuples when the natural "join" operation is performed on the decomposed relation schemes.

Example

Consider the "STUDENT" relation.

STUDENT (Stdid, Name, Location)

This relation can be broken-down into two relations as follows,

- Location (Std id, Location) and
- Name (Std id, Name)

When a natural join operation performed on these two schemes, the original 'STUDENT' relation is sustained.

Location $\bowtie$ Name $\rightarrow$ Student

Student

location

Std id	Location	Name
012	Hyderabad	Radha
013	Secunderabad	Meena
014	Hyderabad	Pinky
015	Secunderabad	Rani

Std id	Location	Std id	Name
012	Hyderabad	012	Radha
013	Secunderabad	013	Meena
014	Hyderabad	014	Pinky
015	Secunderabad	015	Rani

Location $\bowtie$ Name $\rightarrow$ Student

Std id	Location	Name
012	Hyderabad	Radha
013	Secunderabad	Meena
014	Hyderabad	Pinky
015	Secunderabad	Rani

No additional tuples are generated and neither data is duplicated nor data is lost. Therefore the relation STUDENT is lossless.

Lossless Decomposition

1. If the information is not lost from the relation that is decomposed, then the decomposed relation will be lossless.
2. The lossless decomposition guarantees that the join of relations will result in the same relation as it was decomposed.
3. The relation is said to be lossless decomposition if natural joins of all the decomposed relations give the original relation.

DEPENDENCY PRESERVING

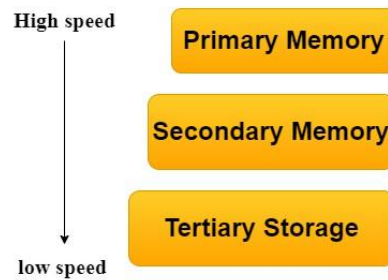
It is an important constraint of the database. In the dependency preservation at least one decomposed table must satisfy every dependency.

If a relation R is decomposed into relations R1 and R2, then the dependencies of R either must be derivable from the combination of functional dependencies of R1 and R2.

For Example suppose there is a relation R(A,B,C,D) with functional dependency set $(A \rightarrow BC)$. The relation R is decomposed into R1(ABC) and R2(AD) which is dependency preserving because functional dependency $A \rightarrow BC$ is a part of relation R1(ABC).

,Q 6) Explain database storage methods on hard disk?

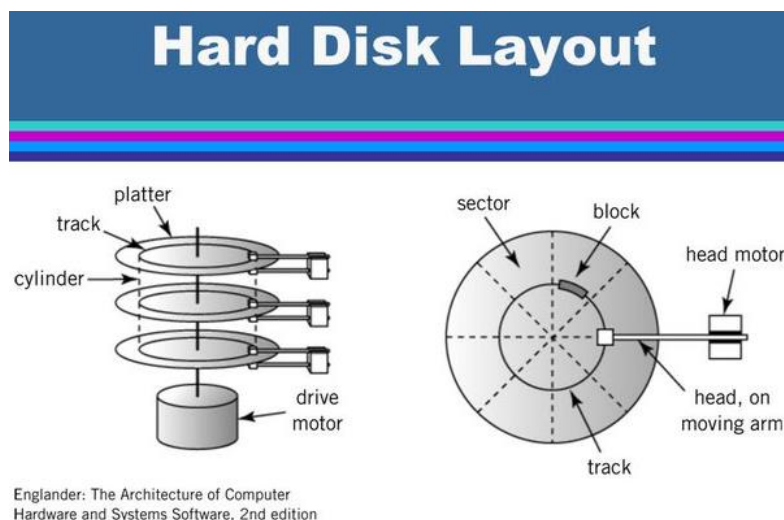
The records in databases are stored in file formats. Physically, the data is stored in electromagnetic format on a device. The electromagnetic devices used in database systems for data storage are *classified* as follows :



- **Primary Memory:** The primary memory of a server is the type of data storage that is directly accessible by the central processing unit, meaning that it doesn't require any other devices to read from it
- **Secondary Memory:** Data storage devices known as secondary storage, Secondary storage systems nowadays consist of magnetic disks and optical discs such as DVD or CD, which were used in earlier times. In addition to portable hard disks and reusable flash drives, peripheral devices are equipped with USB ports so that they can be used as secondary storage devices through plug-and-play.
- **Tertiary Memory:** Tertiary Memory refers to devices that can hold a large amount of data without being constantly connected to the server or the peripherals. A device of this type is connected either to a server or to a device where the database is stored from the outside. The tertiary storage provides more space than other types of device memory but is most slowly performing, the cost of tertiary storage is lower than primary and secondary. As a means to make a backup of data, this type of storage is commonly used for making copies from servers and databases.

Mostly the databases are stored persistently on magnetic disks for the reasons given below:

- The databases being very large may not fit completely in the main memory.
- Storing the data permanently using the non-volatile storage and provide access to the users with the help of front end applications.
- Primary storage is considered to be very expensive and in order to cut short the cost of the storage per unit of data to substantially less.



- Each hard drive is usually composed of a set of disk platters. Each disk platter has a layer of magnetic material deposited on its surface. The entire disk can contain a large amount of data, which is organized into smaller packages called BLOCKS (or pages). On most computers, one block is equivalent to 1 KB of data (= 1024 Bytes).
- A block is the smallest unit of data transfer between the hard disk and the processor of the computer. Each block therefore has a fixed, assigned, address.
- Typically, the computer processor will submit a read/write request, which includes the address of the block, and the address of RAM in the computer memory area called a buffer (or cache) where the data must be stored / taken from.
- The processor then reads and modifies the buffer data as required, and, if required, writes the block back to the disk. Let us see how the tables of the database are stored on the hard disk.

File Organisation

A database consist of a huge amount of data. The data is grouped within a table in RDBMS, and each table have related records. A user can see that the data is stored in form of tables, but in actual this huge amount of data is stored in physical memory in form of files.

File – A file is named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.

Q7) What is file organisation? Explain types of file organization in detail?

In the database management system, the file organization describes the logical relationship among the various stored records. In simple words, we can say that this technique defines how the file records are mapped onto disk blocks. File Organization is also defined as storing the files in a specific order.

Types of File Organization

Following are the various methods which are introduced to organize the files in the database management system are:

1. Sequential File Organization
2. Indexed file Organization
3. Hash File Organization
4. Heap File Organization

1. Sequential File Organization: It is a method in which the files are stored and sorted one after another on disk. This method is so simple for file organization. This file organization arranged the records in either descending or ascending order of the key column. As the files are sorted in a specific order, so the **binary search technique** can be used to reduce the time in searching a file.

Following are the two different ways of implementing this method:

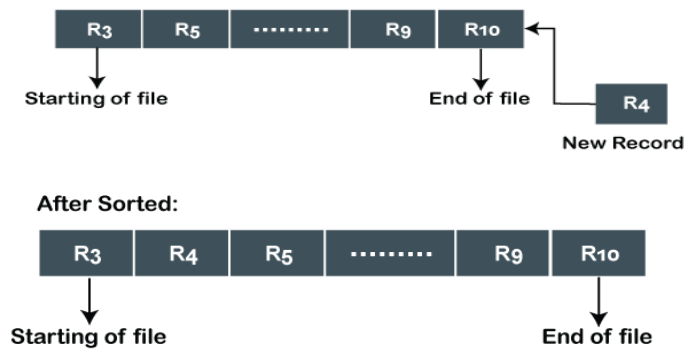
a. Pile File Method: In this sequential file organization method, the files are entered in the same sequence in which they are inserted into the database tables. When any user inserts the new record, the record is then placed at the end of that file. If we delete or update the record, then the record is searched in the blocks of memory. Once it is found, then that founded record is marked for deleting. And, the new block of record will be entered.

Insertion of New record using Pile File Method: Suppose the four records are already stored in the sequence. And, we want to insert the new record (R4) in the sequence, then the R4 record will be placed at the end of the sequence.



b. Sorted File Method: In this sequential organization method, the records are sorted based on the key attribute or another key when they are entered into the database system.

Insertion of New record using Sorted File Method: Suppose the five records are already stored in a sorted manner. And, you want to enter the new record (R4) between the existing records, firstly it will be placed at the end of the file, and then it will sort the specified sequence.



Advantages of Sequential File Organisation

Following are the benefits or advantages of sequential file organization:

- It is a fast and efficient method for the huge amount of data.
- This method does not require so much effort to store the data in the database.
- It is basically used for generating reports and calculating the statistical data.
- Storing the files in this method is cheaper

Disadvantages of Sequential File Organisation.

Following are the two limitations or disadvantages of sequential file organization:

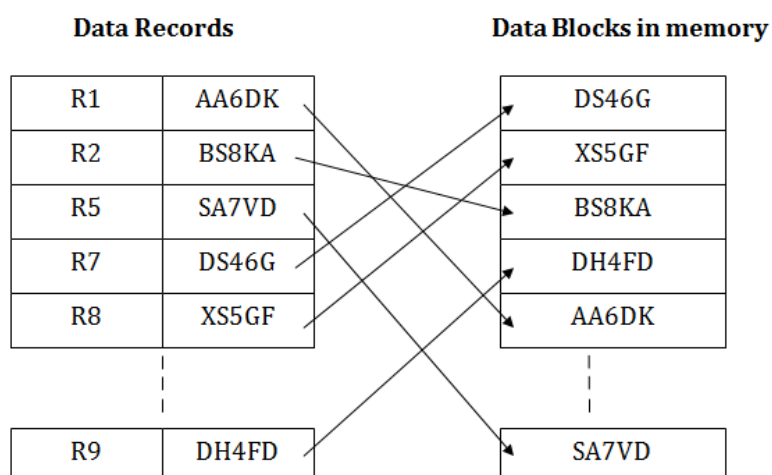
- The sorted file method of sequential file organization is inefficient because it takes more space and time for sorting the records.
- It is a time-consuming process.

2. Indexed Sequential Access Method (ISAM):

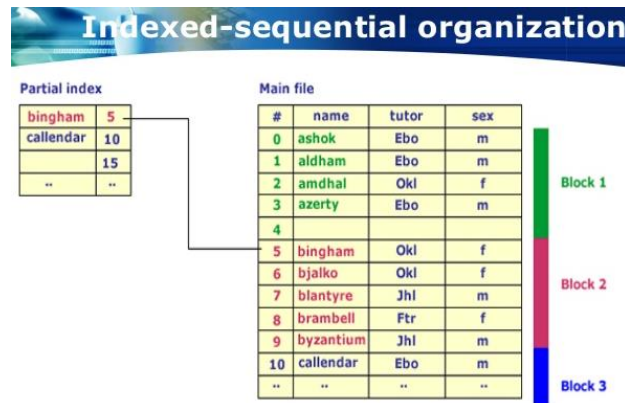
ISAM method is an advanced sequential file organization. In this method, records are stored in the file using the primary key.

In indexed sequential access file, records are stored randomly on a direct access device such as magnetic disk by a primary key. An index value is generated for each primary key and mapped with the record.

This index contains the address of the record in the file.



If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.



Advantages of ISAM:

- In this method, each record has the address of its data block, searching a record in a huge database is quick and easy.
- This method supports range retrieval and partial retrieval of records. Since the index is based on the primary key values, we can retrieve the data for the given range of value. In the same way, the partial value can also be easily searched, i.e., the student name starting with 'JA' can be easily searched.

Disadvantage of ISAM

- This method requires extra space in the disk to store the index value.
- When the new records are inserted, then these files have to be reconstructed to maintain the sequence.
- When the record is deleted, then the space used by it needs to be released. Otherwise, the performance of the database will slow down.

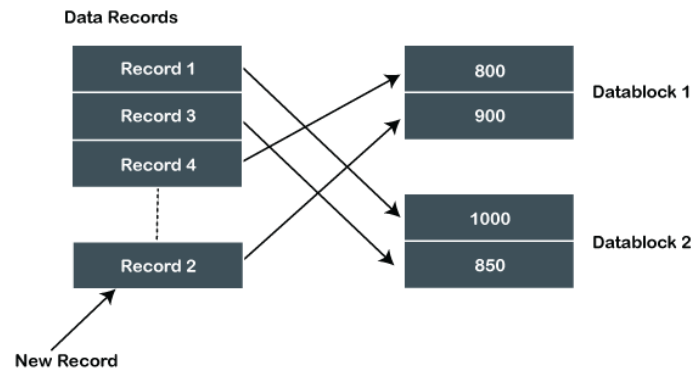
3. Heap File Organisation

- Heap file is also called the unordered file. This type of organization works with the blocks of data, and the new record is inserted at the last page of the file. This type of file organization does not require any sorting for sorting the records.
- If there is insufficient space in the last data block, then the new data block is added to the file. And, then we can easily insert the record in that data block. This makes the insertion of records very efficient.
- As there is no particular ordering to the field values, so the linear search must be performed for accessing the records from the file.
- In the heap file organization, each record has an ID which is unique, and every page or every data block of the file is of the same size.
- If we want to delete the record from the file, then the required record has to be accessed, and then the marked record to be deleted, and after then the block is written back to the disk. The block which contains the deleted record cannot be used as again.

Insert New Record using Heap File Organization: Suppose the three records are already stored in a heap, and we want to insert the new record **Record2** in that heap. Let's

suppose that **Data Block 2** is full. Then the **Record2** will be added in any one of the data blocks selected by the database system.

Let's say **Data Block 1**.



If we want to update, search, or delete the record from the heap file, then we have to read the file from starting until the required record is not found. Suppose, if the database contains a huge amount of data, then the operations take a lot of time for performing on the record because the records are not sorted or not specified in some order.

Advantages of Heap File Organization

- For the small database systems, users can access the records fastly than the sequential file organization.
- It is a simple file organization method.
- It is the best method for loading a large amount of data in the database at a time.

Disadvantages of Heap file Organization

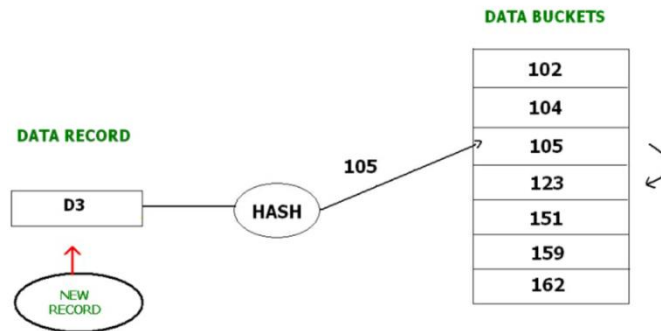
- It is not efficient for large database systems because this method takes more time for performing the operations on the data.
- The main disadvantage of this file organization is that there is a problem with an unused block of memory.

4. Hash File Organization

- In DBMS, hashing is a technique to directly search the location of desired data on the disk without using index structure.
- Hashing method is used to index and retrieve items in a database as it is faster to search that specific item using the shorter hashed key instead of using its original value.
- Data is stored in the form of data blocks whose address is generated by applying a hash function in the memory location where these records are stored known as a **data block or data bucket**.

This file organization uses the *hash function* for calculating the block addresses.

The output of the hash function provides the disk location where the data is actually stored.



The non-key field on which the hash function is generated is called the **hash column**, and the key column on which the hash function is generated is called a **hash key**.

Advantages of Hash File Organization

- Users can access the record at a fast speed because the address of the block is known by the hash function.
- It is the best method for online transactions like ticket booking, online banking, etc.

Disadvantages of Hash File Organization

- There is more chance of losing the data. For Example, In the employee table, when the hash field is on the Employee Name, and there are two same names – 'Areena', then the same address is generated for both. In such a case, the record which is older will be overwritten by newer.
- This method of file organization is not correct in that situation when we are searching for a given range of data because each record in the database file will be stored at a random address. So, in this condition, searching for records is not efficient..

Q) What is Indexing? Explain types of indexes.

Video : <https://www.youtube.com/watch?v=krrSzX7q3oc>

A) Indexing is a data structure technique which allows you to quickly retrieve records from a database file. An Index is a small table having only two columns. The first column comprises a copy of the primary or candidate key of a table. Its second column contains a set of pointers for holding the address of the disk block where that specific key value stored.

An index -

- Takes a search key as input
- Efficiently returns a collection of matching records(data reference)

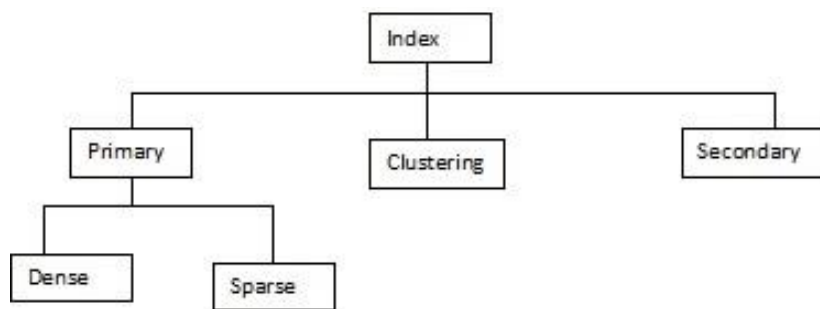
Index structure:

Indexes can be created using some database columns.

Structure of an Index in Database

- The first column of the database is the search key that contains a copy of the primary key or candidate key of the table. The values of the primary key are stored in sorted order so that the corresponding data can be accessed easily.
- The second column of the database is the data reference. It contains a set of pointers holding the address of the disk block where the value of the particular key can be found.

Indexing is defined based on its indexing attributes. **Indexing can be of the following types –**



- **Primary Index** – Primary index is defined on an ordered data file. The data file is ordered on a **primarykey field**. The key field is generally the primary key of the relation.

Primary Index

Data File

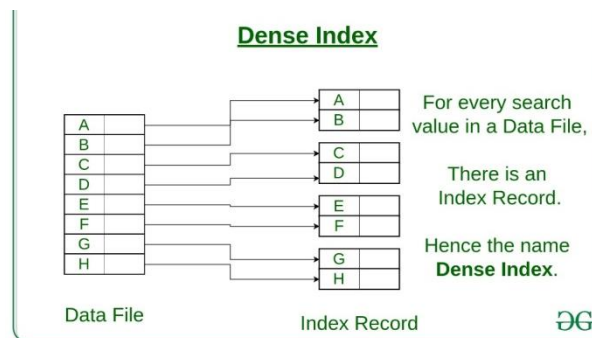
Search key	Address	Address	RegNo	Name	Program	Age
14MS01	0x1F00	0x1F00	14MS01	Ram	MS CS	21
14MS10	0x2000	0x2000	14MS10	Vishal	MS CS	22
14MS29	0x2100	0x2100	14MS29	Priya	MS CS	21
14MS30	0x2200	0x2200	14MS30	Tom	MS CS	21
14MT59	0x2300	0x2300	14MT59	Azad	MTech	23

Ordered Indexing or primary index is of two types –

- Dense Index
- Sparse Index

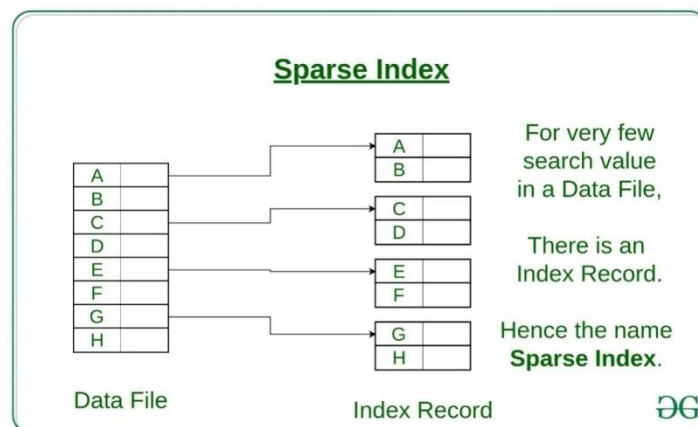
Dense Index

In dense index, there is an index record for every search key value in the database. This makes searching faster but requires more space to store index records itself. Index records contain search key value and a pointer to the actual record on the disk.



Sparse Index

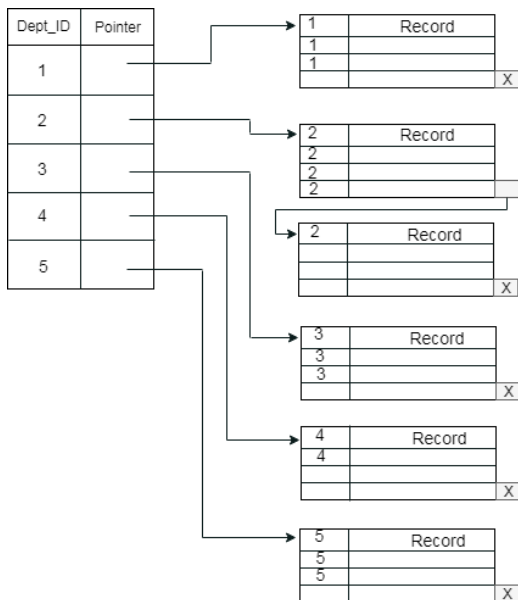
In sparse index, index records are not created for every search key. An index record here contains a search key and an actual pointer to the data on the disk. To search a record, we first proceed by index record and reach at the actual location of the data. If the data we are looking for is not where we directly reach by following the index, then the system starts sequential search until the desired data is found.



- **Secondary Index** – Secondary index may be generated from a field which is a **candidate key** and has a unique value in every record,

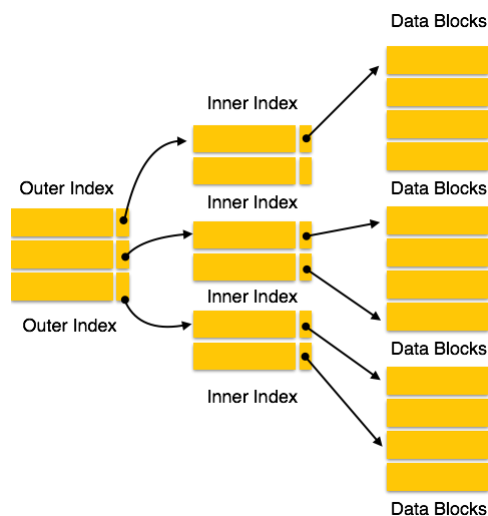
Secondary Index		Data File				
Search key	Address	Address	RegNo	Name	Program	Age
Azad	0x2300	0x1F00	14MS01	Ram	MS CS	21
Priya	0x2100	0x2000	14MS10	Vishal	MS CS	22
Ram	0x1F00	0x2100	14MS29	Priya	MS CS	21
Tom	0x2200	0x2200	14MS30	Tom	MS CS	21
Vishal	0x2000	0x2300	14MT59	Azad	MTech	23

- **Clustering Index** – Clustering index is defined on an ordered data file. The data file is ordered on a non-key field. In some cases, the index is created on **non-primary key columns** which may not be unique for each record. In such cases, in order to identify the records faster, we will group two or more columns together to get the unique values and create index out of them. This method is known as the clustering index.



Multilevel Index

Index records comprise search-key values and data pointers. Multilevel index is stored on the disk along with the actual database files. As the size of the database grows, so does the size of the indices. There is an immense need to keep the index records in the main memory so as to speed up the search operations. If single-level index is used, then a large size index cannot be kept in memory which leads to multiple disk accesses.



Multi-level Index helps in breaking down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block, which can easily be accommodated anywhere in the main memory.

Q Explain BTREE and B+ tree:

A B-tree index stands for “balanced tree” and is a type of index that can be created in relational databases.

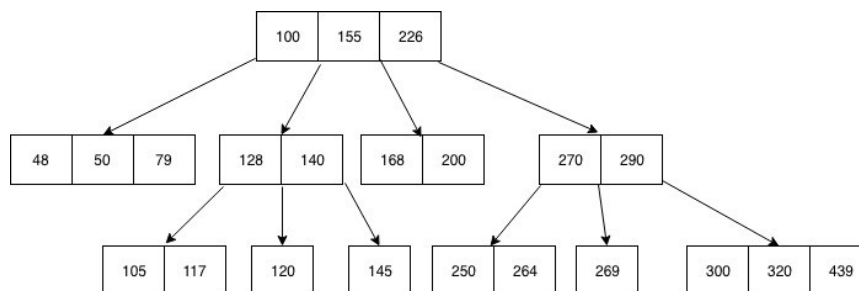
A b-tree index works by creating a series of nodes in a hierarchy. It’s often compared to a tree, which has a root, several branches, and many leaves.

The steps to find this record would be:

1. Start at the root node and go to the first level.

2. Find the node that covers the range of values that span the value of 109 (for example, a range of 100 to 200).
3. Move to the second level from this node.
4. Find the node that covers the range of values that span the value of 109 on the second level (for example, 100 to 120).
5. Move to the third level from this node.
6. Find the record that has an ID of 109.

B-tree stores data such that each node contains keys in ascending order. Each of these keys has two references to another two child nodes. To left side child node keys are less than the current keys and the right side child node keys are more than the current keys. If a single node has “n” number of keys, then it can have maximum “n+1” child nodes.

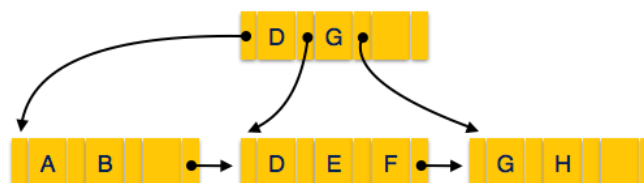


B + TREE:

A B⁺ tree is a balanced binary search tree that follows a multi-level index format. The leaf nodes of a B⁺ tree denote actual data pointers. B⁺ tree ensures that all leaf nodes remain at the same height, thus balanced. Additionally, the leaf nodes are linked using a link list; therefore, a B⁺ tree can support random access as well as sequential access.

Structure of B⁺ Tree

Every leaf node is at equal distance from the root node. A B⁺ tree is of the order **n** where **n** is fixed for every B⁺ tree.



Internal nodes –

- Internal (non-leaf) nodes contain at least $\lceil n/2 \rceil$ pointers, except the root node.
- At most, an internal node can contain **n** pointers.

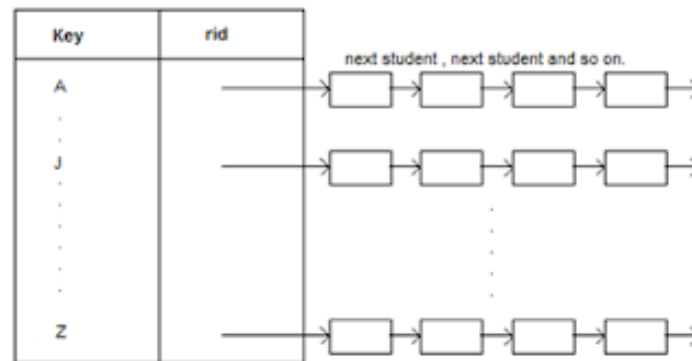
Leaf nodes –

- Leaf nodes contain at least $\lceil n/2 \rceil$ record pointers and $\lceil n/2 \rceil$ key values.
- At most, a leaf node can contain **n** record pointers and **n** key values.
- Every leaf node contains one block pointer **P** to point to next leaf node and forms a linked list.

Q) What is Multi key File Organization?

A) This technique is used to sort a file based on multiple key values. Multi key file organization allows access to a data file by several different key fields. Example: Library file that requires access by author and by subject matter and title.

Scenario 1, records are sorted based on Roll numbers; now requirement is to get names of students starting with 'J'. Therefore create inverted index on names where names starting with each alphabet will be stored as linked list. Now go through the linked list of 'J' to get all the records.



STRUCTURES QUERY LANGUAGE (SQL): Meaning – SQL commands - Data Definition Language - Data Manipulation Language - Data Control Language - Transaction Control Language - Queries using Order by – Where - Group by - Nested Queries. Joins – Views – Sequences - Indexes and Synonyms - Table Handling

Introduction to SQL

SQL is a language which is used to interact with relational database management system.

Q)What is a database

Database is a organized collection of data. For example a database of a college would be having a collection of data such as –

1. Personal records of Students
2. Student's performance history
3. Teacher's data
4. Financial department data etc.

Database Management System (DBMS)

A database management system is a software application which is used for managing different databases. It helps us to create and manage database. With the help of DBMS we take care following tasks –

1. Data Security
2. Data Backup
3. Manages huge amount of data
4. Data export & import
5. Serving multiple concurrent database requests
6. Gives us a way to manage the data using programming languages.

Types of Databases

There are two types of databases –

1. Relational Database
2. Non-relational Database

Non-relational databases:

Data is **not** organized in form of tables. Data is stored in form of key & value pairs. The examples of non-relational databases are: JSON & XML.

We cannot interact with non-relational databases using SQL.

Relational Databases:

In relational database, data is organized in form of tables. A table contains rows and columns of data. Table has a unique key to identify each row of the table.

SQL is used to interact with relational databases. We often refer relational database as SQL database.

Q) What is a Query ?

A) A Query is a set of instruction given to the database management system, which tells RDBMS what information you would like to get from the database.

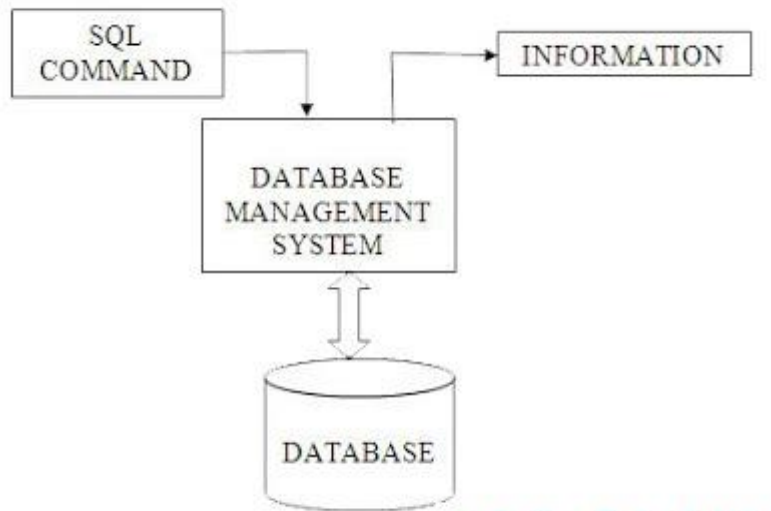
For e.g. to fetch the employee name from the database table EMPLOYEE, we write the SQL Query like this:

```
SELECT employee_name from EMPLOYEE;
```

Q) What is SQL?

A)

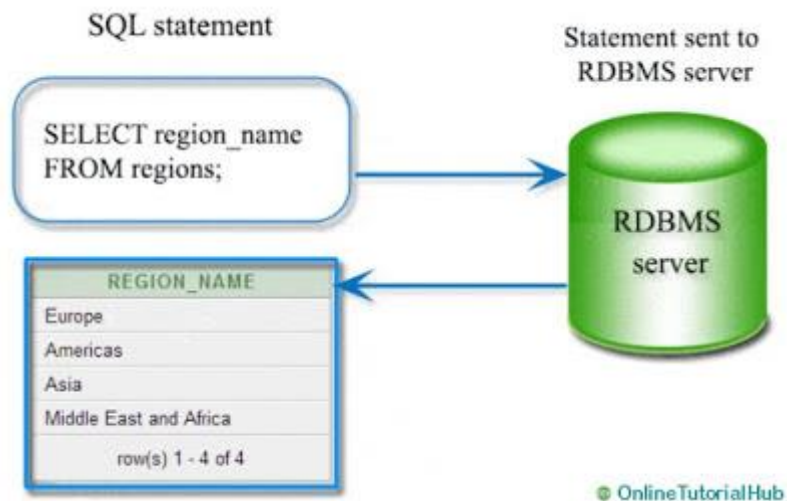
- SQL stands for **Structured Query Language**.
- It is designed for managing data in a relational database management system (RDBMS).
- SQL is a database language, it is used for database creation, deletion, fetching rows and modifying rows etc.
- SQL has reduced the need for lengthy programming to access data.
- The SQL user does not need to know how data is stored. The query language (SQL) takes care of retrieving, replacing, adding, and deleting data.
- A procedural programming language needs every specification about data. SQL is simpler in operation than procedural languages like C/C++ for database management.
- DBMS like MySQL, Oracle, MS Access, Sybase, Informix, Postgres and SQL Server use SQL as standard database language.



Why SQL is required?

SQL is required:

- To create and drop databases and tables.
- To create and drop databases and tables.
- To update records in a database.
- To delete records from a database.
- To retrieve data from a database.
- To set permissions on tables, procedures and views.
- To define the data in a database and manipulate that data.
- To access data in the relational database management systems.
- To create view, stored procedure, functions in a database.



Brief History of SQL

Here, are important landmarks from the history of SQL:

- 1970 - Dr. Edgar F. "Ted" Codd described a relational model for databases.
- 1974 - Structured Query Language appeared.

- 1978 - IBM released a product called System/R.
- 1986 - IBM developed the prototype of a relational database, which is standardized by ANSI.
- 1989- First ever version launched of SQL
- 1999 - SQL 3 launched with features like triggers, object-orientation, etc.
- SQL2003- window functions, XML-related features, etc.
- SQL2006- Support for XML Query Language
- SQL2011-improved support for temporal databases

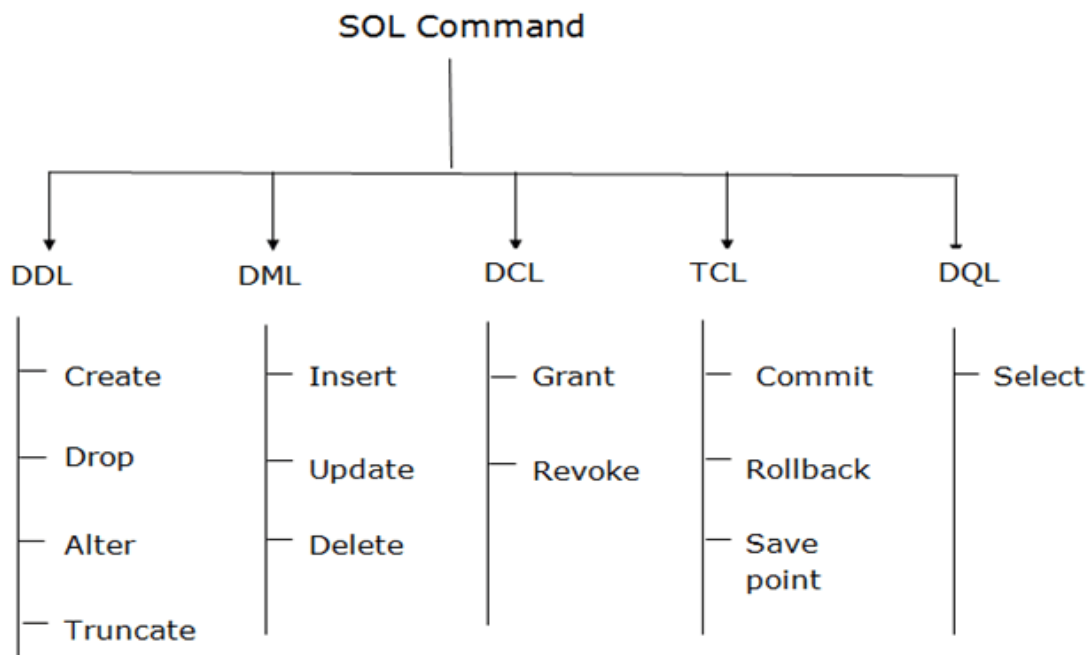
Types of SQL Commands

Q) What is a SQL? Explain types of SQL COMMANDS with example.

A) SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system.

There are five types of SQL commands:

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language (DCL)
4. Transaction Control Language (TCL), and
5. Data Query Language (DQL)



Data Definition Language (DDL)

DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.

All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE
- RENAME

CREATE COMMAND

CREATE is a DDL SQL command used to create a table or a database in relational database management system.

Creating a Database

To create a database in RDBMS, **create** command is used. Following is the syntax,

```
CREATE DATABASE <DB_NAME>;
```

Example for creating Database

```
CREATE DATABASE Test;
```

The above command will create a database named **Test**, which will be an empty schema without any table.

To create tables in this newly created database, we can again use the create command.

Creating a Table

create command can also be used to create tables. Now when we create a table, we have to specify the details of the columns of the tables too. We can specify the **names** and **datatypes** of various columns in the create command itself.

Following is the syntax,

```
CREATE TABLE <TABLE_NAME>  
(  
    column_name1 datatype1,  
    column_name2 datatype2,  
    column_name3 datatype3,  
    column_name4 datatype4  
);
```

create table command will tell the database system to create a new table with the given table name and column information.

Example for creating Table

```
CREATE TABLE Student  
(  
    student_id INT,  
    name VARCHAR(100),  
    age INT  
);
```

The above command will create a new table with name **Student** in the current database with 3 columns, namely student_id, name and age. Where the column student_id will only store integer, name will hold upto 100 characters and age will again store only integer value.

If you are currently not logged into your database in which you want to create the table then you can also add the database name along with table name, using a dot operator .

For example, if we have a database with name **Test** and we want to create a table **Student** in it, then we can do so using the following query:

```
CREATE TABLE Test.Student  
(  
  student_id INT,  
  name VARCHAR(100),  
  age INT  
);
```

ALTER COMMAND

ALTER command is used for altering the table structure, such as,

- to add a column to existing table
- to rename any existing column
- to change datatype of any column or to modify its size.
- to drop a column from the table.

ALTER Command: Add a new Column

Using ALTER command we can add a column to any existing table. Following is the syntax,

```
ALTER TABLE table_name ADD  
(  
  column_name datatype  
);
```

Here is an Example for this,

```
ALTER TABLE student ADD  
(  
  address VARCHAR(200)  
);
```

The above command will add a new column address to the table **student**, which will hold data of type varchar which is nothing but string, of length 200.

ALTER Command: Add multiple new Columns

Using ALTER command we can even add multiple new columns to any existing table. Following is the syntax,

```
ALTER TABLE table_name ADD  
(  
    column_name1 datatype1,  
    column-name2 datatype2,  
    column-name3 datatype3  
);
```

Here is an Example for this,

```
ALTER TABLE student ADD  
(  
    father_name VARCHAR(60),  
    mother_name VARCHAR(60),  
    dob DATE  
);
```

The above command will add three new columns to the **student** table

ALTER Command: Add Column with default value

ALTER command can add a new column to an existing table with a default value too. The default value is used when no value is inserted in the column. Following is the syntax,

```
ALTER TABLE table_name ADD  
(  
    column-name1 datatype1 DEFAULT some_value  
);
```

Here is an Example for this,

```
ALTER TABLE student ADD  
(  
    dob DATE DEFAULT '01-Jan-99'  
);
```

The above command will add a new column with a preset default value to the table **student**.

ALTER Command: Modify an existing Column

ALTER command can also be used to modify data type of any existing column. Following is the syntax,

```
ALTER TABLE table_name modify  
(  
    column_name datatype  
);
```

Here is an Example for this,

```
ALTER TABLE student MODIFY  
(  
    address varchar(300)  
);
```

Remember we added a new column address in the beginning? The above command will modify the address column of the **student** table, to now hold upto 300 characters.

ALTER Command: Rename a Column

Using ALTER command you can rename an existing column. Following is the syntax,

```
ALTER TABLE table_name RENAME old_column_name TO new_column_name;
```

Here is an example for this,

```
ALTER TABLE student RENAME    address TO location;
```

The above command will rename address column to location.

ALTER Command: Drop a Column

ALTER command can also be used to drop or remove columns. Following is the syntax,

```
ALTER TABLE table_name DROP  
(  
    column_name  
);
```

Here is an example for this,

```
ALTER TABLE student DROP  
(  
    Address
```

```
);
```

The above command will drop the address column from the table **student**.

TRUNCATE command

TRUNCATE command removes all the records from a table. But this command will not destroy the table's structure. When we use TRUNCATE command on a table its (auto-increment) primary key is also initialized. Following is its syntax,

```
TRUNCATE TABLE table_name;
```

Here is an example explaining it,

```
TRUNCATE TABLE student;
```

The above query will delete all the records from the table **student**.

In DML commands, we will study about the DELETE command which is also more or less same as the TRUNCATE command. We will also learn about the difference between the two in that tutorial.

DROP Command

DROP command completely removes a table from the database. This command will also destroy the table structure and the data stored in it. Following is its syntax,

```
DROP TABLE table_name;
```

Here is an example explaining it,

```
DROP TABLE student;
```

The above query will delete the **Student** table completely. It can also be used on Databases, to delete the complete database. For example, to drop a database,

```
DROP DATABASE Test;
```

The above query will drop the database with name **Test** from the system.

RENAME Command

RENAME command is used to set a new name for any existing table. Following is the syntax,

```
RENAME TABLE old_table_name to new_table_name;
```

Here is an example explaining it.

```
RENAME TABLE student to students_info;
```

The above query will rename the table **student** to **students_info**.

2. Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT
- UPDATE
- DELETE

1)INSERT command

Insert command is used to insert data into a table. Following is its general syntax,

```
INSERT INTO table_name VALUES(data1, data2, ...)
```

Lets see an example,

Consider a table **student** with the following fields.

s_id	name	age
------	------	-----

```
INSERT INTO student VALUES(101, 'Adam', 15);
```

The above command will insert a new record into **student** table.

s_id	name	age
101	Adam	15

Insert value into only specific columns

We can use the **INSERT** command to insert values for only some specific columns of a row. We can specify the column names along with the values to be inserted like this,

```
INSERT INTO student(id, name) values(102, 'Alex');
```

The above SQL query will only insert id and name values in the newly inserted record.

Insert NULL value to a column

Both the statements below will insert **NULL** value into **age** column of the **student** table.

```
INSERT INTO student(id, name) values(102, 'Alex');
```

Or,

```
INSERT INTO Student VALUES(102,'Alex', null);
```

The above command will insert only two column values and the other column is set to null.

S_id	S_Name	age
101	Adam	15
102	Alex	

Insert Default value to a column

```
INSERT INTO Student VALUES(103,'Chris', default);
```

S_id	S_Name	age
101	Adam	15
102	Alex	
103	chris	14

Suppose the column **age** in our tabel has a default value of 14.

Also, if you run the below query, it will insert default value into the age column, whatever the default value may be.

```
INSERT INTO Student VALUES(103,'Chris');
```

UPDATE

UPDATE command

UPDATE command is used to update any record of data in a table. Following is its general syntax,

```
UPDATE table_name SET column_name = new_value WHERE some_condition;
```

WHERE is used to add a condition to any SQL query, we will soon study about it in detail.

Lets take a sample table **student**,

student_id	name	age
101	Adam	15
102	Alex	
103	chris	14

UPDATE student SET age=18 WHERE student_id=102;

S_id	S_Name	age
101	Adam	15
102	Alex	18
103	chris	14

In the above statement, if we do not use the WHERE clause, then our update query will update age for all the columns of the table to **18**.

Updating Multiple Columns

We can also update values of multiple columns using a single UPDATE statement.

UPDATE student SET name='Abhi', age=17 where s_id=103;

The above command will update two columns of the record which has s_id 103.

s_id	name	age
101	Adam	15
102	Alex	18
103	Abhi	17

UPDATE Command: Incrementing Integer Value

When we have to update any integer value in a table, then we can fetch and update the value in the table in a single statement.

For example, if we have to update the age column of **student** table every year for every student, then we can simply run the following UPDATE statement to perform the following operation:

```
UPDATE student SET age = age+1;
```

As you can see, we have used `age = age + 1` to increment the value of age by 1.

DELETE COMMAND

DELETE command

DELETE command is used to delete data from a table.

Following is its general syntax,

```
DELETE FROM table_name;
```

Let's take a sample table **student**:

s_id	name	age
101	Adam	15
102	Alex	18
103	Abhi	17

Delete all Records from a Table

```
DELETE FROM student;
```

The above command will delete all the records from the table **student**.

Delete a particular Record from a Table

In our **student** table if we want to delete a single record, we can use the WHERE clause to provide a condition in our DELETE statement.

```
DELETE FROM student WHERE s_id=103;
```

The above command will delete the record where s_id is 103 from the table **student**.

S_id	S_Name	age
101	Adam	15
102	Alex	18

Isn't DELETE same as TRUNCATE

TRUNCATE command is different from DELETE command. The delete command will delete all the rows from a table whereas truncate command not only deletes all the records stored in the table, but it also re-initializes the table(like a newly created table).

For eg: If you have a table with 10 rows and an **auto_increment** primary key, and if you use DELETE command to delete all the rows, it will delete all the rows, but will not re-initialize the primary key, hence if you will insert any row after using the DELETE command, the auto_increment primary key will start from 11. But in case of TRUNCATE command, primary key is re-initialized, and it will again start from 1.

DATA QUERY LANGUAGE(DQL)

SELECT SQL Query

SELECT query is used to retrieve data from a table. It is the most used SQL query. We can retrieve complete table data, or partial by specifying conditions using the WHERE clause.

Syntax of SELECT query

SELECT query is used to retrieve records from a table. We can specify the names of the columns which we want in the resultset.

```
SELECT column_name1, column_name2, column_name3, column_nameN ,  
FROM table_name;
```

Consider the following **student** table,

s_id	name	age	address
101	Adam	15	Chennai
102	Alex	18	Delhi
103	Abhi	17	Banglore
104	Ankit	22	Mumbai

```
SELECT s_id, name, age FROM student;
```

The above query will fetch information of s_id, name and age columns of the **student** table and display them,

s_id	name	age
101	Adam	15
102	Alex	18
103	Abhi	17
104	Ankit	22

As you can see the data from address column is absent, because we did not specify it in our SELECT query.

Select all records from a table

A special character **asterisk** * is used to address all the data (belonging to all columns) in a query. SELECT statement uses * character to retrieve all records from a table, for all the columns.

SELECT * FROM student;

The above query will show all the records of **student** table, that means it will show complete dataset of the table.

s_id	name	age	address
101	Adam	15	Chennai
102	Alex	18	Delhi
103	Abhi	17	Bangalore
104	Ankit	22	Mumbai

Select a particular record based on a condition

We can use the **WHERE clause** to set a condition,

SELECT * FROM student WHERE name = 'Abhi';

The above query will return the following result,

103	Abhi	17	Rohtak
-----	------	----	--------

Performing Simple Calculations using SELECT Query

Consider the following **employee** table.

eid	name	age	salary
-----	------	-----	--------

101	Adam	26	5000
102	Ricky	42	8000
103	Abhi	25	10000
104	Rohan	22	5000

Here is our SELECT query,

```
SELECT eid, name, salary+3000 FROM employee;
```

The above command will display a new column in the result, with **3000** added into existing salaries of the employees.

eid	name	salary+3000
101	Adam	8000
102	Ricky	11000
103	Abhi	13000
104	Rohan	8000

So you can also perform simple mathematical operations on the data too using the SELECT query to fetch data from table.

DATA CONTROL LANGUAGE

Data Control Language(DCL) is used to control privileges in Database. To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges. Privileges are of two types,

- **System:** This includes permissions for creating session, table, etc and all types of other system privileges.
- **Object:** This includes permissions for any command or query to perform any operation on the database tables.

In DCL we have two commands,

- **GRANT:** Used to provide any user access privileges or other privileges for the database.
- **REVOKE:** Used to take back permissions from any user.

Privileges are: 1. Create 2. Insert 3.update 4. Delete

Allow a User to create session

When we create a user in SQL, it is not even allowed to login and create a session until and unless proper permissions/privileges are granted to the user.

Following command can be used to grant the session creating privileges.

GRANT CREATE SESSION TO username;

Allow a User to create table

To allow a user to create tables in the database, we can use the below command,

GRANT CREATE TABLE TO username;**GRANT UPDATE TABLE TO USERNAME;**

Provide user with space on tablespace to store table

Allowing a user to create table is not enough to start storing data in that table. We also must provide the user with privileges to use the available tablespace for their table and data.

ALTER USER username QUOTA UNLIMITED ON SYSTEM;

The above command will alter the user details and will provide it access to unlimited tablespace on system.

NOTE: Generally unlimited quota is provided to Admin users.

Grant all privilege to a User

sysdba is a set of privileges which has all the permissions in it. So if we want to provide all the privileges to any user, we can simply grant them the sysdba permission.

GRANT sysdba TO username

Grant permission to create any table

Sometimes user is restricted from creating some tables with names which are reserved for system tables. But we can grant privileges to a user to create any table using the below command,

GRANT CREATE ANY TABLE TO username

Grant permission to drop any table

As the title suggests, if you want to allow user to drop any table from the database, then grant this privilege to the user,

```
GRANT DROP ANY TABLE TO username
```

To take back Permissions

And, if you want to take back the privileges from any user, use the REVOKE command.

```
REVOKE CREATE TABLE FROM username
```

TRANSACTION CONTROL LANGUAGE(TCL)

Transaction Control Language(TCL) commands are used to manage transactions in the database. These are used to manage the changes made to the data in a table by DML statements. It also allows statements to be grouped together into logical transactions.

COMMIT command

COMMIT command is used to permanently save any transaction into the database.

When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.

To avoid that, we use the COMMIT command to mark the changes as permanent.

Following is commit command's syntax,

```
COMMIT;
```

ROLLBACK command

This command restores the database to last committed state. It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction.

If we have used the UPDATE command to make some changes into the database, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not committed using the COMMIT command.

Following is rollback command's syntax,

```
ROLLBACK TO savepoint_name;
```

SAVEPOINT command

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

Following is savepoint command's syntax,

```
SAVEPOINT savepoint_name;
```

In short, using this command we can **name** the different states of our data in any table and then rollback to that state using the ROLLBACK command whenever required.

Using Savepoint and Rollback

Following is the table **class**,

id	name
1	Abhi
2	Adam
4	Alex

Lets use some SQL queries on the above table and see the results.

```
INSERT INTO class VALUES(5, 'Rahul');
```

```
COMMIT;
```

```
UPDATE class SET name = 'Abhijit' WHERE id = '5';
```

```
SAVEPOINT A;
```

```
INSERT INTO class VALUES(6, 'Chris');
```

```
SAVEPOINT B;
```

```
INSERT INTO class VALUES(7, 'Bravo');
```

```
SAVEPOINT C;
```

```
SELECT * FROM class;
```

NOTE: SELECT statement is used to show the data stored in the table.

The resultant table will look like,

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	Chris
7	Bravo

Now let's use the ROLLBACK command to roll back the state of data to the **savepoint B**.

ROLLBACK TO B;

SELECT * FROM class;

Now our **class** table will look like,

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	Chris

Now let's again use the ROLLBACK command to roll back the state of data to the **savepoint A**

ROLLBACK TO A;

SELECT * FROM class;

Now the table will look like,

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit

So now you know how the commands COMMIT, ROLLBACK and SAVEPOINT works.

LIKE clause is used in the condition in SQL query with the WHERE clause. LIKE clause compares data with an expression using wildcard operators to match pattern given in the condition.

Wildcard operators

There are two wildcard operators that are used in LIKE clause.

- **Percent sign %**: represents zero, one or more than one character.
- **Underscore sign _**: represents only a single character.

Example of LIKE clause

Consider the following **Student** table.

s_id	s_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17
104	Eric	18

```
SELECT * FROM Student WHERE s_name LIKE 'A%';
```

%

The above query will return all records where **s_name** starts with character 'A'.

s_id	s_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17

Using _ and %

```
SELECT * FROM Student WHERE s_name LIKE '_d%';
```

The above query will return all records from **Student** table where **s_name** contain 'd' as second character.

s_id	s_Name	age
101	Adam	15

Using % only

SELECT * FROM Student WHERE s_name LIKE '%x';

The above query will return all records from **Student** table where **s_name** contain 'x' as last character.

s_id	s_Name	age
102	Alex	18

SQL ORDER BY Clause

Order by clause is used with SELECT statement for arranging retrieved data in sorted order. The **Order by** clause by default sorts the retrieved data in ascending order. To sort the data in descending order DESC keyword is used with Order by clause.

Syntax of Order By

SELECT column-list|* FROM table-name ORDER BY ASC | DESC;

Using default Order by

Consider the following **Emp** table,

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000

405	Tiger	35	8000
-----	-------	----	------

SELECT * FROM Emp ORDER BY salary;

The above query will return the resultant data in ascending order of the **salary**.

eid	name	age	salary
403	Rohan	34	6000
402	Shane	29	8000
405	Tiger	35	8000
401	Anu	22	9000
404	Scott	44	10000

Using Order by DESC

Consider the **Emp** table described above,

SELECT * FROM Emp ORDER BY salary DESC;

The above query will return the resultant data in descending order of the **salary**.

eid	name	age	salary
404	Scott	44	10000
401	Anu	22	9000
405	Tiger	35	8000
402	Shane	29	8000
403	Rohan	34	6000

Group by Clause

Group by clause is used to group the results of a SELECT query based on one or more columns. It is also used with SQL functions to group the result from one or more tables.

Syntax for using Group by in a statement.

```
SELECT column_name, function(column_name) FROM  
table_name WHERE condition GROUP BY column_name;
```

Example of Group by in a Statement

Consider the following **Emp** table.

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	9000
405	Tiger	35	8000

Here we want to find **name** and **age** of employees grouped by their **salaries** or in other words, we will be grouping employees based on their salaries, hence, as a result, we will get a data set, with unique salaries listed, along side the first employee's name and age to have that salary. Hope you are getting the point here!

group by is used to group different row of data together based on any one column.

SQL query for the above requirement will be,

```
SELECT name, age FROM Emp GROUP BY salary;
```

Result will be,

name	age
Rohan	34
Shane	29
Anu	22

Example of Group by in a Statement with WHERE clause

Consider the following **Emp** table

eid	name	age	salary
------------	-------------	------------	---------------

401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	9000
405	Tiger	35	8000

SQL query will be,

SELECT name, salary FROM Emp WHERE age > 25 GROUP BY salary

Result will be.

name	salary
Rohan	6000
Shane	8000
Scott	9000

You must remember that Group By clause will always come at the end of the SQL query, just like the Order by clause.

HAVING Clause

Having clause is used with SQL Queries to give more precise condition for a statement. It is used to mention condition in Group by based SQL queries, just like WHERE clause is used with SELECT query.

Syntax for HAVING clause is,

SELECT column_name, function(column_name)

FROM table_name

WHERE column_name condition

GROUP BY column_name

HAVING function(column_name) condition

Example of SQL Statement using HAVING

Consider the following **Sale** table.

oid	order_name	previous_balance	customer
11	ord1	2000	Alex
12	ord2	1000	Adam
13	ord3	2000	Abhi
14	ord4	1000	Adam
15	ord5	2000	Alex

Suppose we want to find the **customer** whose **previous_balance** sum is more than **3000**.

We will use the below SQL query,

```
SELECT * FROM sale GROUP BY customer HAVING sum(  
previous_balance ) > 3000
```

Result will be,

oid	order_name	previous_balance	customer
11	ord1	2000	Alex

The main objective of the above SQL query was to find out the name of the customer who has had a **previous_balance** more than **3000**, based on all the previous sales made to the customer, hence we get the first row in the table for customer Alex.

VIEWS

A VIEW in SQL is a logical subset of data from one or more tables. View is used to restrict data access.

Syntax for creating a View,

```
CREATE or REPLACE VIEW view_name  
AS  
SELECT column_name(s)  
FROM table_name
```

WHERE condition

As you may have understood by seeing the above SQL query, a view is created using data fetched from some other table(s). It's more like a temporary table created with data.

Creating a VIEW

Consider following **Sale** table,

oid	order_name	previous_balance	customer
11	ord1	2000	Alex
12	ord2	1000	Adam
13	ord3	2000	Abhi
14	ord4	1000	Adam
15	ord5	2000	Alex

SQL Query to Create a View from the above table will be,

```
CREATE or REPLACE VIEW sale_view  
AS  
SELECT * FROM Sale WHERE customer = 'Alex';
```

The data fetched from **SELECT** statement will be stored in another object called **sale_view**. We can use **CREATE** and **REPLACE** separately too, but using both together works better, as if any view with the specified name exists, this query will replace it with fresh data.

Displaying a VIEW

The syntax for displaying the data in a view is similar to fetching data from a table using a **SELECT** statement.

```
SELECT * FROM sale_view;
```

Copy

Force VIEW Creation

FORCE keyword is used while creating a view, forcefully. This keyword is used to create a View even if the table does not exist. After creating a force View if we create the base table and enter values in it, the view will be automatically updated.

Syntax for forced View is,

```
CREATE or REPLACE FORCE VIEW view_name AS  
  SELECT column_name(s)  
  FROM table_name  
  WHERE condition;
```

Update a VIEW

UPDATE command for view is same as for tables.

Syntax to Update a View is,

```
UPDATE view-name SET VALUE  
WHERE condition;
```

Copy

NOTE: If we update a view it also updates base table data automatically.

Read-Only VIEW

We can create a view with read-only option to restrict access to the view.

Syntax to create a view with Read-Only Access

```
CREATE or REPLACE FORCE VIEW view_name AS  
  SELECT column_name(s)  
  FROM table_name  
  WHERE condition WITH read-only;
```

The above syntax will create view for **read-only** purpose, we cannot Update or Insert data into read-only view. It will throw an **error**.

Types of View

There are two types of view,

- Simple View
- Complex View

Simple View	Complex View
Created from one table	Created from one or more table
Does not contain functions	Contain functions
Does not contain groups of data	Contains groups of data

Q) What are SQL Functions?

A) SQL provides many built-in functions to perform operations on data. These functions are useful while performing mathematical calculations, string concatenations, sub-strings etc. SQL functions are divided into two categories,

1. Aggregate Functions
2. Scalar Functions

Aggregate Functions

These functions **return a single value** after performing calculations on a group of values. Following are some of the frequently used Aggregate functions.

AVG() Function

Average returns average value after calculating it from values in a numeric column.

Its general **syntax** is,

```
SELECT AVG(column_name) FROM table_name
```

Using AVG() function

Consider the following **Emp** table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query to find average salary will be,

SELECT avg(salary) from Emp;

Result of the above query will be,

avg(salary)
8200

COUNT() Function

Count returns the number of rows present in the table either based on some condition or without condition.

Its general **syntax** is,

SELECT COUNT(column_name) FROM table-name;

Using COUNT() function

Consider the following **Emp** table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query to count employees, satisfying specified condition is,

SELECT COUNT(name) FROM Emp WHERE salary = 8000;

Result of the above query will be,

count(name)
2

Example of COUNT(distinct)

Consider the following **Emp** table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query is,

```
SELECT COUNT(DISTINCT salary) FROM emp;
```

Result of the above query will be,

count(distinct salary)
4

FIRST() Function

First function returns first value of a selected column

Syntax for FIRST function is,

```
SELECT FIRST(column_name) FROM table-name;
```

Using FIRST() function

Consider the following **Emp** table

eid	name	age	salary
-----	------	-----	--------

401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query will be,

```
SELECT FIRST(salary) FROM Emp;
```

and the result will be,

first(salary)
9000

LAST() Function

LAST function returns the return last value of the selected column.

Syntax of LAST function is,

```
SELECT LAST(column_name) FROM table-name;
```

Using LAST() function

Consider the following **Emp** table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query will be,

```
SELECT LAST(salary) FROM emp;
```

Result of the above query will be,

last(salary)

8000

MAX() Function

MAX function returns maximum value from selected column of the table.

Syntax of MAX function is,

SELECT MAX(column_name) from table-name;

Using MAX() function

Consider the following **Emp** table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query to find the Maximum salary will be,

SELECT MAX(salary) FROM emp;

Result of the above query will be,

MAX(salary)
10000

MIN() Function

MIN function returns minimum value from a selected column of the table.

Syntax for MIN function is,

SELECT MIN(column_name) from table-name;

Using MIN() function

Consider the following **Emp** table,

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query to find minimum salary is,

```
SELECT MIN(salary) FROM emp;
```

Result will be,

MIN(salary)
6000

SUM() Function

SUM function returns total sum of a selected columns numeric values.

Syntax for SUM is,

```
SELECT SUM(column_name) from table-name;
```

Using SUM() function

Consider the following **Emp** table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query to find sum of salaries will be,

```
SELECT SUM(salary) FROM emp;
```

Result of above query is,

SUM(salary)
41000

Scalar Functions

Scalar functions return a single value from an input value. Following are some frequently used Scalar Functions in SQL.

UCASE() Function

UCASE function is used to convert value of string column to Uppercase characters.

Syntax of UCASE,

```
SELECT UCASE(column_name) from table-name;
```

Using UCASE() function

Consider the following **Emp** table

eid	name	age	salary
401	anu	22	9000
402	shane	29	8000
403	rohan	34	6000
404	scott	44	10000
405	Tiger	35	8000

SQL query for using UCASE is,

```
SELECT UCASE(name) FROM emp;
```

Result is,

UCASE(name)
ANU
SHANE
ROHAN
SCOTT
TIGER

LCASE() Function

LCASE function is used to convert value of string columns to Lowecase characters.

Syntax for LCASE is,

SELECT LCASE(column_name) FROM table-name;

Using LCASE() function

Consider the following **Emp** table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	SCOTT	44	10000
405	Tiger	35	8000

SQL query for converting string value to Lower case is,

SELECT LCASE(name) FROM emp;

Result will be,

LCASE(name)
anu
shane
rohan
scott
tiger

MID() Function

MID function is used to extract substrings from column values of string type in a table.

Syntax for MID function is,

SELECT MID(column_name, start, length) from table-name;

Using MID() function :

Consider the following **Emp** table

eid	name	age	salary
401	anu	22	9000
402	shane	29	8000
403	rohan	34	6000
404	scott	44	10000
405	Tiger	35	8000

SQL query will be,

SELECT MID(name,2,2) FROM emp;

Result will come out to be,

MID(name,2,2)
nu
ha
oh
co
ig

ROUND() Function

ROUND function is used to round a numeric field to number of nearest integer. It is used on Decimal point values.

Syntax of Round function is,

SELECT ROUND(column_name, decimals) from table-name;

Using ROUND() function

Consider the following **Emp** table

eid	name	age	salary
401	anu	22	9000.67
402	shane	29	8000.98
403	rohan	34	6000.45
404	scott	44	10000
405	Tiger	35	8000.01

SQL query is,

SELECT ROUND(salary) from emp;

Result will be,

ROUND(salary)
9001
8001
6000
10000
8000

Q) Explain SQL JOINS in detail.

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. It is used for combining column from two or more tables by using values common to both tables.

JOIN Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is **(n-1)** where **n**, is number of tables. A table can also join to itself, which is known as, **Self Join**.

Types of JOIN

Following are the types of JOIN that we can use in SQL:

- Inner
- Outer

- Left
- Right

Cross JOIN or Cartesian Product

This type of JOIN returns the cartesian product of rows from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Cross JOIN Syntax is,

```
SELECT column-name-list FROM table-name1 CROSS JOIN table-name2;
```

Example of Cross JOIN

Following is the **class** table,

ID	NAME
1	abhi
2	adam
4	alex

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Cross JOIN query will be,

```
SELECT * FROM class CROSS JOIN class_info;
```

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	1	DELHI
4	alex	1	DELHI
1	abhi	2	MUMBAI
2	adam	2	MUMBAI
4	alex	2	MUMBAI
1	abhi	3	CHENNAI
2	adam	3	CHENNAI
4	alex	3	CHENNAI

As you can see, this join returns the cross product of all the records present in both the tables.

INNER Join or EQUI Join

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

Inner Join Syntax is,

```
SELECT column-name-list FROM  
table-name1 INNER JOIN table-name2  
WHERE table-name1.column-name = table-name2.column-name;
```

Example of INNER JOIN

Consider a **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Inner JOIN query will be,

```
SELECT * from class INNER JOIN class_info where class.id = class_info.id;
```

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI

Natural JOIN

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

The syntax for Natural Join is,

```
SELECT * FROM  
table-name1 NATURAL JOIN table-name2;
```

Example of Natural JOIN

Here is the **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Natural join query will be,

```
SELECT * from class NATURAL JOIN class_info;
```

The resultset table will look like,

ID	NAME	Address
1	abhi	DELHI
2	adam	MUMBAI
3	alex	CHENNAI

In the above example, both the tables being joined have **ID** column(same name and same datatype), hence the records for which value of **ID** matches in both the tables will be the result of Natural Join of these two tables.

OUTER JOIN

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

1. Left Outer Join
2. Right Outer Join

3. Full Outer Join

LEFT Outer Join

The left outer join returns a resultset table with the **matched data** from the two tables and then the remaining rows of the **left** table and null from the **right** table's columns.

Syntax for Left Outer Join is,

```
SELECT column-name-list FROM  
table-name1 LEFT OUTER JOIN table-name2  
ON table-name1.column-name = table-name2.column-name;
```

To specify a condition, we use the **ON** keyword with Outer Join.

Left outer Join Syntax for **Oracle** is,

```
SELECT column-name-list FROM  
table-name1, table-name2 on table-name1.column-name = table-name2.column-  
name(+);
```

Example of Left Outer Join

Here is the **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Left Outer Join query will be,

```
SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id = class_info.id);
```

Copy

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null

RIGHT Outer Join

The right outer join returns a resultset table with the **matched data** from the two tables being joined, then the remaining rows of the **right** table and null for the remaining **left** table's columns.

Syntax for Right Outer Join is,

**SELECT column-name-list FROM table-name1 RIGHT OUTER JOIN table-name2
ON table-name1.column-name = table-name2.column-name;**

Right outer Join Syntax for **Oracle** is,

**SELECT column-name-list FROM
table-name1, table-name2
ON table-name1.column-name(+) = table-name2.column-name;**

Example of Right Outer Join

Once again the **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI

3	CHENNAI
7	NOIDA
8	PANIPAT

Right Outer Join query will be,

```
SELECT * FROM class RIGHT OUTER JOIN class_info ON (class.id = class_info.id);
```

Copy

The resultant table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
null	null	7	NOIDA
null	null	8	PANIPAT

Full Outer Join

The full outer join returns a resultset table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.

Syntax of Full Outer Join is,

```
SELECT column-name-list FROM  
table-name1 FULL OUTER JOIN table-name2  
ON table-name1.column-name = table-name2.column-name;
```

Example of Full outer join is,

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Full Outer Join query will be like,

```
SELECT * FROM class FULL OUTER JOIN class_info ON (class.id = class_info.id);
```

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT

Q) What is an SQL Sequence?

A) Sequence is a feature supported by some database systems to produce unique values on demand. Some DBMS like **MySQL** supports **AUTO_INCREMENT** in place of Sequence.

AUTO_INCREMENT is applied on columns, it automatically increments the column value by 1 each time a new record is inserted into the table.

Sequence is also some what similar to **AUTO_INCREMENT** but it has some additional features too.

Creating a Sequence

Syntax to create a sequence is,

```
CREATE SEQUENCE sequence-name  
START WITH initial-value  
INCREMENT BY increment-value  
MAXVALUE maximum-value  
CYCLE | NOCYCLE;
```

- The **initial-value** specifies the starting value for the Sequence.
- The **increment-value** is the value by which sequence will be incremented.
- The **maximum-value** specifies the upper limit or the maximum value upto which sequence will increment itself.
- The keyword CYCLE specifies that if the maximum value exceeds the set limit, sequence will restart its cycle from the beginning.
- And, NO CYCLE specifies that if sequence exceeds MAXVALUE value, an error will be thrown.

Using Sequence in SQL Query

Let's start by creating a sequence, which will start from 1, increment by 1 with a maximum value of 999.

```
CREATE SEQUENCE seq_1 START WITH 1 INCREMENT BY 1 MAXVALUE 999  
CYCLE;
```

Now let's use the sequence that we just created above.

Below we have a **class** table,

ID	NAME
1	abhi
2	adam
4	alex

The SQL query will be,

```
INSERT INTO class VALUE(seq_1.nextval, 'anu');
```

Resultset table will look like,

ID	NAME
1	abhi
2	adam
4	alex
1	anu

Once you use nextval the sequence will increment even if you don't Insert any record into the table.

UNIT –IV

Transaction & Concurrency Database Recovery and Security

Transactions - Concurrent Transactions - Locking Protocol - Serialisable Schedules - Locks Two Phase Locking (2PL) - Deadlock and its Prevention - Optimistic Concurrency Control.

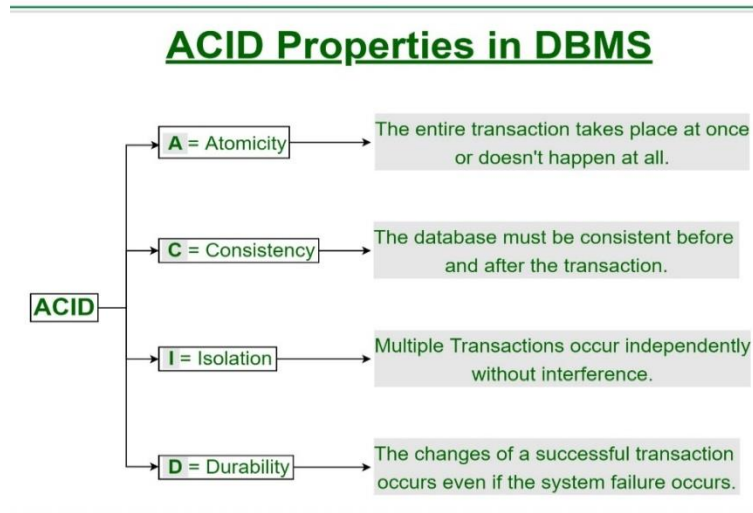
Database Recovery and Security: Database Recovery meaning – Kinds of failures – Failure controlling methods – Database errors – Backup & Recovery Techniques – Security & Integrity – Database Security – Authorization.

Q1) What is Transaction? Write about ACID property in detail.

A) A Set of Logically related operations is known as Transaction. The operations of a transaction are

1. **Read (A):** Read operation Read(A) or R(A) reads the value of A from the database and stores it in a buffer in main memory.
2. **Write (A):** write operation write(A) or W(A) writes the value back to the database from buffer

In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called **ACID** properties.



Atomicity

By this, we mean that either the entire transaction takes place at once or doesn't happen at all. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves the following two operations.

—**Abort:** If a transaction aborts, changes made to database are not visible.

—**Commit:** If a transaction commits, changes made are visible.

Atomicity is also known as the 'All or nothing rule'.

Consider the following transaction **T** consisting of **T₁** and **T₂**: Transfer of 100 from account **X** to account **Y**.

Before: X : 500	Y: 200
Transaction T	
T1	T2
Read (X)	Read (Y)
X: = X – 100	Y: = Y + 100
Write (X)	Write (Y)
After: X : 400	Y : 300

If the transaction fails after completion of **T1** but before completion of **T2**.(say, after **write(X)** but before **write(Y)**), then amount has been deducted from **X** but not added to **Y**. This results in an inconsistent database state. Therefore, the transaction must be executed in entirety in order to ensure correctness of database state.

Consistency

This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database.

Referring to the example above,

The total amount before and after the transaction must be maintained.

Total **before T** occurs = **500 + 200 = 700**.

Total **after T** occurs = **400 + 300 = 700**.

Therefore, database is **consistent**. Inconsistency occurs in case **T1** completes but **T2** fails. As a result T is incomplete.

Isolation

This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of database state. Transactions occur independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.

Let **X=500, Y =500**.

Consider two transactions **T** and **T''**.

T	T''
Read (X)	Read (X)
X: = X*100	Read (Y)
Write (X)	Z: = X + Y
Read (Y)	Write (Z)
Y: = Y – 50	
Write (Y)	

Suppose **T** has been executed till **Read (Y)** and then **T''** starts. As a result , interleaving of operations takes place due to which **T''** reads correct value of **X** but incorrect value of **Y** and sum computed by

T'': (**X+Y = 50, 000+500=50, 500**)

is thus not consistent with the sum at end of transaction:

T: (**X+Y = 50, 000 + 450 = 50, 450**).

This results in database inconsistency, due to a loss of 50 units. Hence, transactions

must take place in isolation and changes should be visible only after they have been made to the main memory.

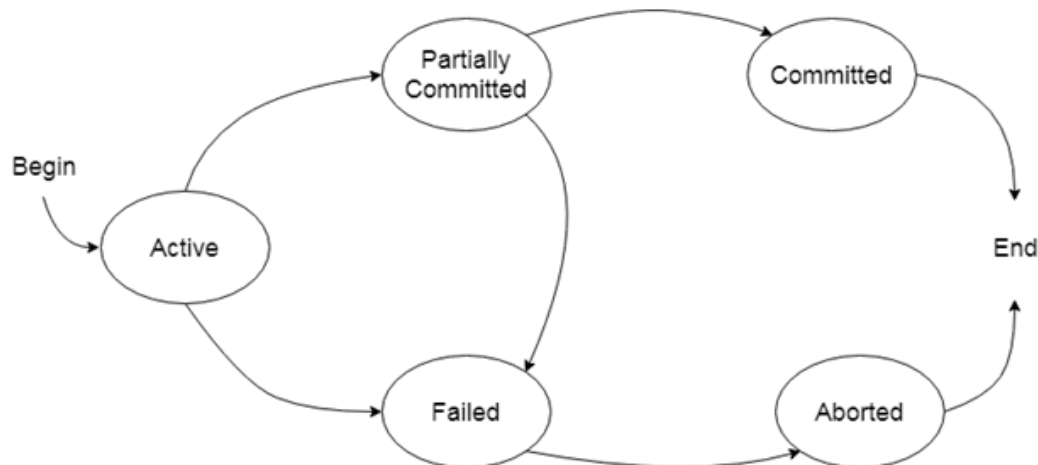
Durability:

This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

The **ACID** properties, in totality, provide a mechanism to ensure correctness and consistency of a database in a way such that each transaction is a group of operations that acts a single unit, produces consistent results, acts in isolation from other operations and updates that it makes are durably stored.

Q2) Write about states of transactions.**A) States of Transaction**

In a database, the transaction can be in one of the following states -

**Active state**

- The active state is the first state of every transaction. In this state, the transaction is being executed.
- For example: Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

Partially committed

- In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database.
- In the total mark calculation example, a final display of the total marks step is executed in this state.

Committed

A transaction is said to be in a committed state if it executes all its operations successfully. In this state, all the effects are now permanently saved on the database system.

Failed state

- I. If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.
- II. In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

Aborted

- If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state. If not then it will abort or roll back the transaction to bring the database into a consistent state.
- If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.
- After aborting the transaction, the database recovery module will select one of the two operations:
 1. Re-start the transaction
 2. Kill the transaction

Q3)What is Concurrency? Explain Concurrency Control techniques in detail.

(Or)

Q) Explain locking techniques in detail.

A) Concurrency control is the procedure in DBMS for managing simultaneous operations without conflicting with each another. Concurrent access is quite easy if all users are just reading data. Though for any practical database, would have a mix of reading and WRITE operations and hence the concurrency is a challenge.

Concurrency control is used to address such conflicts which mostly occur with a multi-user system. It helps you to make sure that database transactions are performed concurrently without violating the data integrity of respective databases.'

Need of Concurrency Control in DBMS

Concurrency is needed for the following reasons:

1. Increase the number of transactions executed per unit time i.e., **improve the throughput.**
2. **Better Resource Utilization** as the CPU and Disk operations can be executed side by side.
3. **Reduced average waiting time** for transactions to complete.

Potential problems of Concurrency

Here, are some issues which you will likely to face while using the DBMS Concurrency Control method:

- **Lost Updates** occur when multiple transactions select the same row and update the row based on the value selected
- Uncommitted dependency issues occur when the second transaction selects a row which is updated by another transaction (**dirty read**)
- **Non-Repeatable Read** occurs when a second transaction is trying to access the same row several times and reads different data each time.
- **Incorrect Summary issue** occurs when one transaction takes summary over the value of all the instances of a repeated data-item, and second transaction update few instances of that specific data-item. In that situation, the resulting summary does not reflect a correct result.

Concurrency Control techniques

Concurrency Control techniques are the schemes which are used by DBMS to ensure that concurrent transactions do not interfere with each other. Some of these Locking Techniques are:

1. Lock – Based Protocol
2. Two –Phase Locking(2PL)
3. Time Stamps-Based and
4. Optimistic Protocols.

in order to ensure the correct execution of concurrent transactions, a number of **concurrency control techniques** are applied in DBMS. One of them is use of Locking Techniques.

1.Lock-Based protocol

In this type of protocol, any transaction cannot read or write data until it acquires an appropriate lock on it. There are two types of lock:

Shared lock : It is also known as a Read-only lock. In a shared lock, the data item can only read by the transaction. It can be shared between the transactions because when the transaction holds a lock, then it can't update the data on the data item.

Exclusive lock : In the exclusive lock, the data item can be both reads as well as written by the transaction. This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously.

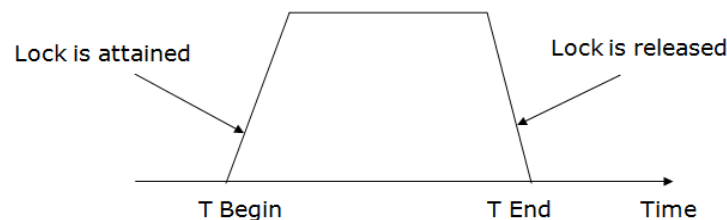
There are two types of Lock Protocol

1. **Simplistic lock protocol:** It is the simplest way of locking the data while transaction. Simplistic lock-based protocols allow all the transactions to get the lock on the data before insert or delete or update on it. It will unlock the data item after completing the transaction.
2. **Pre-claiming Lock Protocol:** Pre-claiming Lock Protocols evaluate the transaction to list all the items on which they need locks. Before initiating an execution of the transaction, it requests DBMS for all the lock on all those data items. If all the locks are granted then this protocol allows the transaction to begin. When the transaction is completed then it releases all the lock. If all the locks are not granted then this protocol allows the transaction to roll back and waits until all the locks are granted.

(V.V.IMP) 2. Two Phase Locking

Q4) Explain Two phase Locking (2PL)

- The two-phase locking protocol divides the execution phase of the transaction into three parts.
- In the first part, when the execution of the transaction starts, it seeks permission for the lock it requires.
- In the second part, the transaction acquires all the locks. The third phase is started as soon as the transaction releases its first lock.
- In the third phase, the transaction cannot demand any new locks. It only releases the acquired locks.



There are two phases of 2PL:

Growing phase: In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released.

Shrinking phase: In the shrinking phase, existing lock held by the transaction may be released, but no new locks can be acquired.

In the below example, if lock conversion is allowed then the following phase can happen:

1. Upgrading of lock (from S(a) to X (a)) is allowed in growing phase.
2. Downgrading of lock (from X(a) to S(a)) must be done in shrinking phase.

Example:

	T1	T2
0	LOCK-S(A)	
1		LOCK-S(A)
2	LOCK-X(B)	
3	—	—
4	UNLOCK(A)	
5		LOCK-X(C)
6	UNLOCK(B)	
7		UNLOCK(A)
8		UNLOCK(C)
9	—	—

The following way shows how unlocking and locking work with 2-PL.

Transaction T1:

- **Growing phase:** from step 1-3
- **Shrinking phase:** from step 5-7
- **Lock point:** at 3

Transaction T2:

- **Growing phase:** from step 2-6
- **Shrinking phase:** from step 8-9
- **Lock point:** at 6

(IMP)Q5) Explain Optimistic concurrency control

The optimistic method of concurrency control is based on the assumption that conflicts of database operations are rare and that it is better to let transactions run to completion and only check for conflicts before they commit. An optimistic concurrency control method is also known as validation or certification methods.

No checking is done while the transaction is executing. The optimistic method does not require locking or time stamping techniques. Instead, a transaction is executed with restrictions until it is committed.

Phases of Optimistic Method

In optimistic methods, each transaction moves through the following phases:

Read phase : During the read phase, the transaction reads the database, executes the needed computations, and makes the updates to a private copy of the database values. All update operations of the transaction are recorded in a temporary update file, which is not accessed by the remaining transactions.

Validation phase : During the validation phase, the transaction is validated to ensure that the changes made will not affect the integrity and consistency of the database. If the validation test is positive, the transaction goes to the write phase. If the validation test is negative, the transaction is restarted and the changes are discarded.

Write phase : During the write phase, the changes are permanently applied to the database. The optimistic approach is acceptable for most read or query database systems that require few update transactions.

Advantages of Optimistic Concurrency Control

- i. This technique is very efficient when conflicts are rare. The occasional conflicts result in the transaction roll back
- ii. The rollback involves only the local copy of data, the database is not involved and thus there will not be any cascading rollbacks.

Problems of Optimistic for Concurrency Control

- a) Conflicts are expensive to deal with, since the conflicting transaction must be rolled
- b) Longer transactions are more likely to have conflicts and may be repeatedly rolled because if conflicts with short transaction

Applications of Optimistic for Concurrency Control

- i. Only suitable for environments where there are few conflicts and no long transactions.
- ii. Acceptable for mostly Read or Query systems that require very few update transactions.

Q6) Explain Serializable Schedules

A scheduler refers to a database process, which provides an order to the transactions that are concurrently executing the same data. This order can be determined by the concurrency control algorithms. But, in a database every transaction is not serializable. i.e., some transactions are serially executed while the other are non-serially executed. The DBMS finds which transactions are serializable and which are non-serializable. Thus, the nonserializable transactions are executed in a First Come First Serve (FCFS) order.

However, the execution of non-serializable transactions in an FCFS manner consume a lot of CPU time because the CPU has to wait until a READ or WRITE operation is entirely completed. As a result, several CPU cycles are lost and the FCFS scheduling of the transactions result in an unacceptable response times in a database system. Thus, to avoid such situation, 'other scheduling algorithms are required so as to increase the overall efficiency of the database system.

A schedule in which the intervened execution of the database transactions produce the same results as that of the transactions that are executed serially are known as a serializable schedule. Thus, the basic responsibility of a scheduler is to establish a serializable schedule for the database transactions. It ensures that the CPU and the storage system are working efficiently and effectively. It also helps in ensuring the data isolation property by maintaining the fact that no two transactions update/modify the same set of data.

T 1	T 2	Output
Read	Read	No conflict
Write	Write	Conflict
Write	Read	Conflict
Read	Write	Conflict

Table: Readwrite Operation Performed by T and T2

In the above table, the following outputs occur during the concurrent execution of transactions,

- When T₁ and T₂ performs a read operation, then no conflict occurs.
- When T₁ and T₂ performs a write operation, then a conflict occurs.
- When T₁ performs a write operation and T₂ concurrently performs a read operation, then a conflict occurs.
- When T₁ performs a read operation and T₂ concurrently performs a write operation, then a conflict occurs.

The conflicting operations can be scheduled for concurrent executions by adopting several methods like timestamping, locking and optimistic methods.

Q7) What is Deadlock?(V.V.IMP)

A) A **deadlock** is a condition wherein two or more tasks are waiting for each other in order to be finished but none of the task is willing to give up the resources that other task needs. In this situation no task ever gets finished and is in waiting state forever.

For example: In the student table, transaction T1 holds a lock on some rows and needs to update some rows in the grade table. Simultaneously, transaction T2 holds locks on some rows in the grade table and needs to update the rows in the Student table held by Transaction T1.

Now, the main problem arises. Now Transaction T1 is waiting for T2 to release its lock and similarly, transaction T2 is waiting for T1 to release its lock. All activities come to a halt state and remain at a standstill. It will remain in a standstill until the DBMS detects the deadlock and aborts one of the transactions.

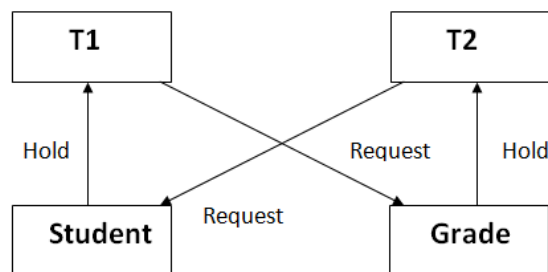


Figure: Deadlock in DBMS

Necessary Conditions for Deadlock

- i. **Mutual Exclusion:** This means that some of the resources involved cannot be used by more than one process at the same time. It means that if the resource (like a printer) is being used by any process and some other process comes then it has to wait for the resources to be released.
- ii. **No Preemption:** If a process is holding a resource then its resources can't be forcibly taken from it.
- iii. **Hold & Wait:** The process is holding some resources and is waiting for the other resources.
- iv. **Circular Wait:** The process is waiting for the resource which is being held by the next process and this forms a chain of waiting. In the above example, Process P1 and Process P2 are forming a chain and waiting for the resource being held by each other. If the above four conditions hold simultaneously then a Deadlock may occur.

Techniques to control deadlocks

The three basic techniques to control deadlocks are :

1. Deadlock Prevention

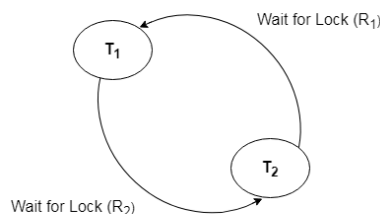
- Deadlock prevention method is suitable for a large database. If the resources are allocated in such a way that deadlock never occurs, then the deadlock can be prevented.
- The Database management system analyzes the operations of the transaction whether they can create a deadlock situation or not. If they do, then the DBMS never allowed that transaction to be executed.

2. Deadlock Detection

In a database, when a transaction waits indefinitely to obtain a lock, then the DBMS should detect whether the transaction is involved in a deadlock or not. The lock manager maintains a Wait for the graph to detect the deadlock cycle in the database.

Wait for Graph

- This is the suitable method for deadlock detection. In this method, a graph is created based on the transaction and their lock. If the created graph has a cycle or closed loop, then there is a deadlock.
- The wait for the graph is maintained by the system for every transaction which is waiting for some data held by the others. The system keeps checking the graph if there is any cycle in the graph.



3. Deadlock Avoidance

- When a database is stuck in a deadlock state, then it is better to avoid the database rather than aborting or restating the database. This is a waste of time and resource.
- Deadlock avoidance mechanism is used to detect any deadlock situation in advance. A method like "wait for graph" is used for detecting the deadlock

situation but this method is suitable only for the smaller database. For the larger database, deadlock prevention method can be used.

The choice of the best deadlock control method to use depends on the database environment. For example, if the probability of deadlocks is low, deadlock detection is recommended. However, if the probability of deadlocks is high, deadlock prevention is recommended. If response time is not high on the system's priority list, deadlock avoidance might be employed. All current DBMSs support deadlock detection in transactional databases, while some DBMSs use a blend of prevention and avoidance techniques for other types of data, such as data warehouses or XML data.

Database Recovery and Security

Q8) what is a system failure?

A)Whenever the system fails to function according to its specifications and doesn't deliver the expected service, that situation is called the failure of the system.

To find that where the problem has occurred, we generalize a failure into the following categories:

1. System Crash
2. Transaction Failure
3. Network Failure
4. Disk Failure
5. Media Failure

- Each transaction has ACID property. If we fail to maintain the ACID properties, it is the failure of the database system.

1. System Crash :
 - a. System crash occurs when there is a hardware or software failure or external factors like a power failure.
 - b. The data in the secondary memory is not affected when system crashes because the database has lots of integrity. Checkpoint prevents the loss of data from secondary memory.
2. Transaction Failure
Reasons for a transaction failure could be
 - a. Logical errors – Where a transaction cannot complete because it has some code error or any internal error condition.
 - b. System errors – Where the database system itself terminates an active transaction because the DBMS is not able to execute it, or it has to stop

because of some system condition. For example, in case of deadlock or resource unavailability, the system aborts an active transaction

3. Network Failure

- a. A network failure occurs when a client – server configuration or distributed database system are connected by communication networks.

4. Disk Failure

- a. Disk Failure occurs when there are issues with hard disks like formation of bad sectors, disk head crash, unavailability of disk etc.

5. Media Failure

- a. Media failure is the most dangerous failure because, it takes more time to recover than any other kind of failures.
- b. A disk controller or disk head crash is a typical example of media failure.
- c. Natural disasters like floods, earthquakes, power failures, etc. damage the data.

6. User Error

Normally, user error is the biggest reason of data destruction or corruption in a database.

Q9) What is a transaction Failure?

A)When a transaction is submitted to a database system, this is the responsibility of the database management system to execute all the operations in the Transaction.

According to **the atomicity property of Transaction**, all the operations in a transaction have to be executed, or none will be completed. There won't be a case where only half of the operations will be executed, or this case will lead to a transaction failure.

The reasons for transaction failure are mentioned below –

System Failure or System Crash

During the Transaction, at a certain point, the system may fail due to various reasons like power failure, network failure, some internal errors in the system, etc., which will lead to transaction failure.

Transaction or System Error

This problem may arise due to programming errors, data storing errors, etc. If the program is not well tested and not executed properly, this may cause a transaction failure too.

Q10) what is database Recovery? Explain some recovery techniques in detail.(v.v.imp)

A) Database Recovery is the process of restoring a database to the correct state in the event of a failure. It ensures that the database is reliable and remains in consistent state in case of a failure. There are both automatic and non-automatic ways for both, backing up of data and recovery from any failure situations

Recovery techniques

There are many disaster recovery tools and techniques available that can assist an organization in database recovery. There are types of techniques, which can help a DBMS in recovering as well as maintain the atomicity of a transaction.

(i) Log based recovery:

In this method, log of each transaction is maintained in some stable storage, so that in case of any failure, it can be recovered from there to recover the database.

But storing the logs should be done before applying the actual transaction on the database.

Every log in this case will have information like:

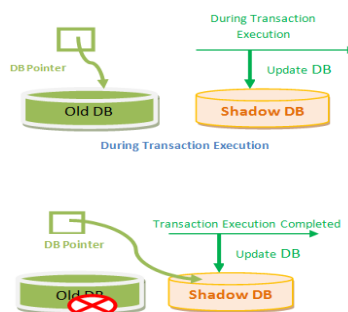
- What transaction is being executed,
- Which values have been modified to which value, and
- State of the transaction.

All these log information will be stored in the order of execution. An update log record describes a single database transaction with:

- Transactions identifier
- Data item identifier
- Old value
- New value

(ii) Shadow paging recovery

This is the method where all the transactions are executed in the primary memory. Once all the transactions completely executed, it will be updated to the database. Hence, if there is any failure in the middle of transaction, it will not be reflected in the database. Database will be updated after all the transaction.



(iii) Check point recovery

Checkpoint acts like a bookmark. During the execution of transaction, such checkpoints are marked and transaction is executed.

The log files will be created as usual with the steps of transactions. When it reaches the checkpoint, the transaction will be updated into database and all the logs till that point will be removed from file.

Log files then are updated with new steps of transaction till next checkpoint and so on.

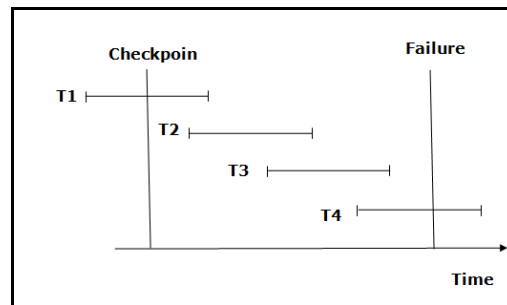


fig: Recovery system recovers the database from this failure

Q11) Explain Database Errors

There are many different types of Errors that can affect database processing.

System Errors:-

In case of system crash, the systems hang up and need to be rebooted. These failures occur due to hardware malfunction or a bug in the database software or the operating system.

User errors:-

Intentionally or unintentionally a user may delete a row or dropping a table, Lack of end-user training also may cause errors.

Carelessness:-

Carelessness is the destruction of data by end-users because of no concentrating on the task at hand or they are not aware of what they are doing.

Statement failure:-

A statement failure can be defined as the inability of the database to execute an SQL statement. While running a user program, a transaction might have multiple statements and one of the statements might fail due to various reasons.

Application software error:-

Application software errors include logical errors in the program that is accessing the database, which causes one or more transactions to fail.

Network errors:-

Network failures can occur while using a client-server configuration or a distributed database system where multiple database servers are connected by communication networks. Network failures such as communication software failures or aborted asynchronous connections will interrupt the normal operations of the database system.

Natural physical disasters:-

Natural and physical disasters are the damage caused to data, hardware and software due to natural disasters like fires, floods, earthquakes, power crashes, etc.

Media errors:-

Media failures are the most dangerous failures. Not only is there a potential to lose data if proper backup procedures are not followed, but it usually takes more time to recover than with other kinds of failures. In addition, the DBA's experience is very important factor in determining the kind of media recovery procedure to use to bring the database up quickly, with little or no data loss.

Sabotage (damage):-

Sabotage is the intentional destruction of data, hardware (or) software facilities.

Q12) Write about Database backup.

A) Database backup

- "It enables the creation of a duplicate copy of a database in case the main database crashes, is corrupted (or) is lost".
- Data and information are important for an organization.
- In case data and information become unavailable due to any reason, all the functionality of the organization stops and it becomes unmanageable loss. In order to avoid such big loss, database administrator (DBA) must take backup of database frequently according to the nature of the database.
- Normally full backup are done after one week but incremental backup are done on daily basis.
- Backup can be defined as second copy of original database. This second copy (or) saved copy of original database is produced on separate storage media such as magnetic tape, external hard disk and on CD-drives.

- These should be placed in a separate and secure location, where unauthorized persons do not have any access.

The best way of storing database backup is online using internet and it can be accessed any time, from any location.

Depending on the nature, there are 2 important types of backup.

- i. Hot backup (or) Online Backup
- ii. Cold backup (or) Offline backup

Hot backup (or) Online Backup:-Hot backup is taken when the database is running and data is being updated, added (or) read by its users but doesn't handle running transactions.Hot backups are dynamic (or) active in nature.

Only Database Management System (DBMS) facility can be used to take hot backups.Make sure that hot backups should only be done during low working hours.Hot backup is taken in environments where database needs to remain online 24 hours a day and 7 days a week, because they do not require downtime as in cold backup.

Cold backup (or) Offline backup:-Cold backup is taken when database is offline, not accessible for updating and users cannot do anything on the database.

DBMS as well as operating system facilities can be used to take the cold backup.Normally cold backup is taken at the end of a working day (or) at weekends when database becomes offline.

Cold backup is suitable in environments where database is not running 24 hours a day.Backups can be taken manually through the DBA (or) these can be taken automatically by configuring the DBMS.

In case of automatic backups, we must specify day, time and location on storage media where DBMS will take the backups.

Q13) Explain Data Availability?

Availability of information refers to ensuring that authorized people are able to access the information when needed. Information only has value if the right people can access it at the right times. Information is unavailable not only when it is lost (or) destroyed, but also when access to the information is denied or delayed (hacked).

Availability of data is maintained when all components of the database system are working properly.Problems in the database system could make it impossible to access information, thereby making the information unavailable.

Backups are used to ensure availability of database information.

Q14) Explain briefly about Database Security?(V.V.IMP)

A) Database security refers to the collective measures used to protect and secure a database or database management software from illegitimate use and malicious cyber threats and attacks.

Database security procedures are aimed at protecting not just the data inside the database, but the database management system and all the applications that access it from intrusion, misuse of data, and damage.



Database security is generally easier to cope with an malicious access to the database, which includes the following

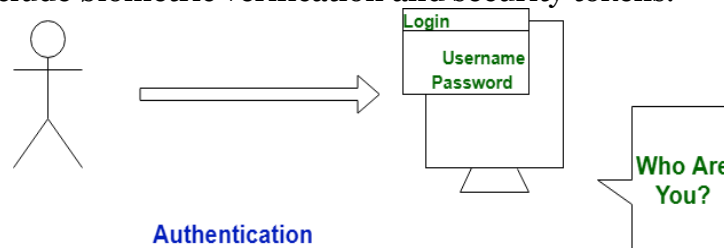
- 1) Theft of information :
- 2) Unauthorized modification of data.
- 3) Unauthorized destruction of data.

We have 3 ways of Database Security

- 1) Authentication
- 2) Authorization
- 3) Encryption

1) Authentication:- One of the most basic concepts in database security is authentication, which is quite simply the process by which it system verifies a use's identity, a user can respond to a request to authenticate by providing a proof of identity (or) User ID and password.

Other options include biometric verification and security tokens.



- a) First level of authentication is giving identity to the user by
- something the user knows, such as password
 - something the user processes such as ID card
 - Physical representation of the user, such a finger print or voice print.

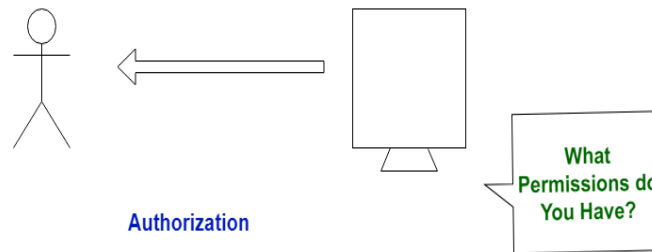
b) A password scheme limits unauthorized system access by creating a password that is difficult to guess but still easy for the user to remember.

c) For creating passwords one scheme can be used is a question-and-answer session With user.

d) If the user unable to supply a valid password with in a set time or no; of trials, they should automatically logoff users.

e) Unauthorized system access can be further controlled by allowing read only functions.

2) Authorization: Authorization is the process through which system obtains information about the authenticated user, including which data base operations that user may perform and which data objects (Tables) that user may access.



- Authorization is the process of granting someone to do something. It means it a way to check if the user has permission to use a resource or not.
- It defines that what data and information one user can access . The authorization usually works with authentication so that the system could know who is accessing the information.

A user may have several forms of authorization on parts of the database . They are:

- 1) **Read authorization**, which allows reading but not modification of data.
- 2) **Insert authorization**, which allows insertion of new data, but not modification of existing data.
- 3) **Update authorization**, which allows modifications, but not deletion of data.
- 4) **Delete authorization**, which allows deletion of data.

To give/ take this authorization to/from the users SQL have2 commands

1. GRANT and
2. REVOKE

1) **GRANT** statement is used to give authorization to the user.

Syntax: - GRANT <insert/alter/delete/select/update> on <view name> to user list>

Ex: - GRANT UPDATE on branch to U1, U2, U3.

In above example branch is the view name, U1, U2, U3 are the users name

2) **REVOKE** command is used to revoke authorization from the user.

Syntax: -REVOKE <command> on <view/table> from <user list>

Ex: -REVOKE UPDATE ON BRANCH FROM U1, U2 and U3.

ENCRYPTION& DECRYPTION

The various authentication and authorization measures that are standard for protecting access to Databases may not be sufficient for highly sensitive data. In such instance, it may be desirable to ENCRYPT the data.

Encrypt: - it means that to convert readable text to unreadable text by use of an algorithm; this method is used to protect sensitive data.

Method of converting Encrypted data to Normal text we called it as Decipher or **Decrypt**

Plaintext: - Which is readable text.

Cipher text: -Encrypted text

Ex: - SUCCESS

Becomes TVDDFTT

But technique we used is very easy. Anyone can decipher it. ENCRPTION has many techniques.

One method or scheme is called public key encryption. In this method, there are two keys, a public key and a private key. Each user has a public key (Ei) and private key (Pi). All the public keys are published. Each user's private key is known only to the user to whom the key belongs. If user wants to store encrypted data, he can encrypt it using public key (Ei). Decryption requires the private key (Pi).

Q15) Why is database security important?

A) Safeguarding the data your company collects and manages is of utmost importance. Database security can guard against a compromise of your database, which can lead to financial loss, reputation damage, consumer confidence disintegration, brand erosion, and non-compliance of government and industry regulation.

Database security safeguards defend against a myriad of security threats and can help protect your enterprise from:

- Deployment failure
- Excessive privileges
- Privilege abuse
- Platform vulnerabilities
- Unmanaged sensitive data
- Backup data exposure
- Weak authentication
- Database injection attacks

-----***-----

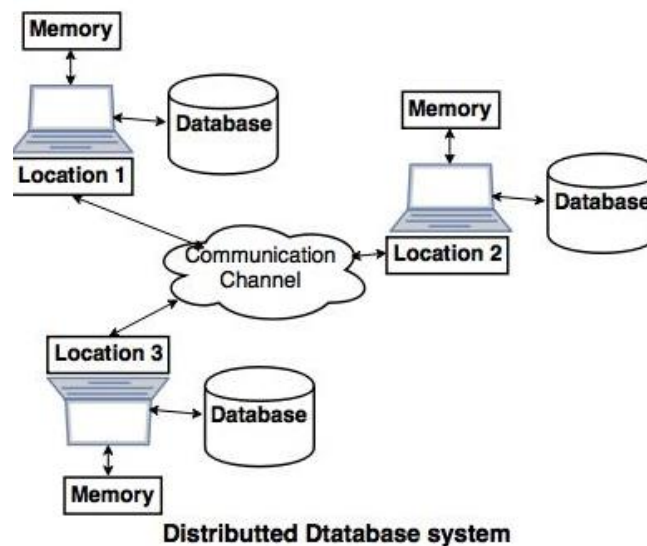
Distributed DBMS

Syllabus

- Need for Distributed Database Systems
- Structure of Distributed Database
- Advantages and Disadvantages of DDBMS
- Advantages of Data Distribution
- Disadvantages of Data Distribution
- Data Replication
- Data Fragmentation.
- Client Server Databases:
 - Emergence of Client Server Architecture
 - Need for Client Server Computing
 - Structure of Client Server Systems & its advantages.

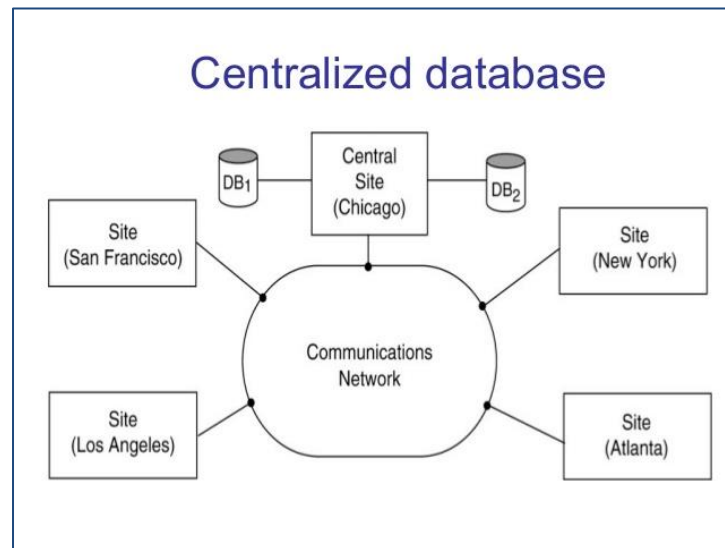
Distributed DBMS**Q: What is Distributed Database Management System (DDBMS).****A)**

- Distributed Database: is a collection of multiple, logically related database, physically distributed over network is called distributed database.
- Distributed database is a system in which storage devices are not connected to a common (centralized) processing unit.
- Database is controlled by Distributed Database Management System and data may be stored at the same location or spread over the interconnected network. It is a loosely coupled system.



- The above diagram is a typical example of distributed database system, in which communication channel is used to communicate with the different locations and every system has its own memory and database.

Q What are Problems with centralized DBMS

Distributed DBMS

- Centralized databases are highly dependent on network connectivity.
- The slower the internet connection is, the longer the database access time needed will be.
- [Bottlenecks](#) can occur as a result of high traffic.
- Limited access by more than one person to the same set of data as there is only one copy of it and it is maintained in a single location.
- This can lead to major decreases in the general efficiency of the system.
- If there is no fault-tolerant setup and hardware failure occurs, all the data within the database will be lost.
- Since there is minimal to no data redundancy, if a set of data is unexpectedly lost it is very hard to retrieve it back, in most cases it would have to be done manually.

Q) Explain Needs and goals for Distributed Database Systems**A) Needs for Distributed Database Systems**

- Business unit autonomy and distribution
- Data sharing
- Data communication costs
- Data communication reliability and costs
- Multiple application vendors
- Database recovery
- Transaction and analytic processing

Goals of Distributed Database system.

The concept of distributed database was built with a goal to improve:

Distributed DBMS

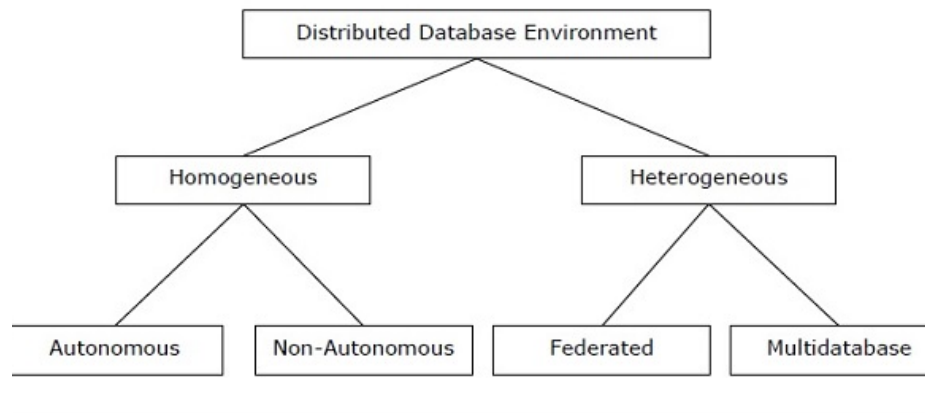
Reliability: In distributed database system, if one system fails down or stops working for some time another system can complete the task.

Availability: In distributed database system reliability can be achieved even if sever fails down. Another system is available to serve the client request.

Performance: Performance can be achieved by distributing database over different locations. So the databases are available to every location which is easy to maintain.

Q) What are the Types of distributed databases.

A) Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments, each with further sub-divisions, as shown in the following illustration.

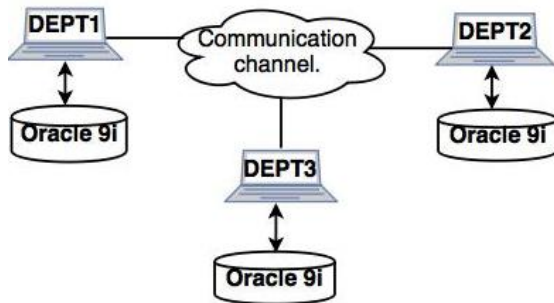


1. Homogeneous distributed databases system:

- Homogeneous distributed database system is a network of two or more databases (With same type of DBMS software) which can be stored on one or more machines.
- So, in this system data can be accessed and modified simultaneously on several databases in the network. Homogeneous distributed system are easy to handle.

Example: Consider that we have three departments using Oracle-9i for DBMS. If some changes are made in one department then, it would update the other department also.

Distributed DBMS



Homogeneous distributed system

Types of Homogeneous Distributed Database

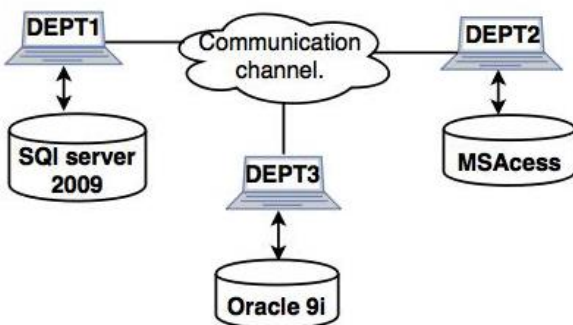
There are two types of homogeneous distributed database –

- **Autonomous** – Each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.
- **Non-autonomous** – Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

2. Heterogeneous distributed database system.

- Heterogeneous distributed database system is a network of two or more databases with different types of DBMS software, which can be stored on one or more machines.
- In this system data can be accessible to several databases in the network with the help of generic connectivity (ODBC and JDBC).

Example: In the following diagram, different DBMS software are accessible to each other using ODBC and JDBC.



Heterogeneous distributed system

Types of Heterogeneous Distributed Databases

- **Federated** – The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.
- **Un-federated** – The database systems employ a central coordinating module through which the databases are accessed.

Distributed DBMS

Q Explain Advantages and Disadvantages of DDBMS**A) Advantages of DDBMS**

- The database is easier to expand as it is already spread across multiple systems and it is not too complicated to add a system.
- The distributed database can have the data arranged according to different levels of transparency i.e. data with different transparency levels can be stored at different locations.
- The database can be stored according to the departmental information in an organization. In that case, it is easier for an organizational hierarchical access.
- there was a natural catastrophe such as fire or an earthquake all the data would not be destroyed it is stored at different locations.
- It is cheaper to create a network of systems containing a part of the database. This database can also be easily increased or decreased.
- Even if some of the data nodes go offline, the rest of the database can continue its normal functions.

Disadvantages of DDBMS

- The distributed database is quite complex and it is difficult to make sure that a user gets a uniform view of the database because it is spread across multiple locations.
- This database is more expensive as it is complex and hence, difficult to maintain.
- It is difficult to provide security in a distributed database as the database needs to be secured at all the locations it is stored. Moreover, the infrastructure connecting all the nodes in a distributed database also needs to be secured.
- It is difficult to maintain data integrity in the distributed database because of its nature. There can also be data redundancy in the database as it is stored at multiple locations.
- The distributed database is complicated and it is difficult to find people with the necessary experience who can manage and maintain it.

Q) Explain Components of DDBMS

Or

Explain Structure of DDBMS

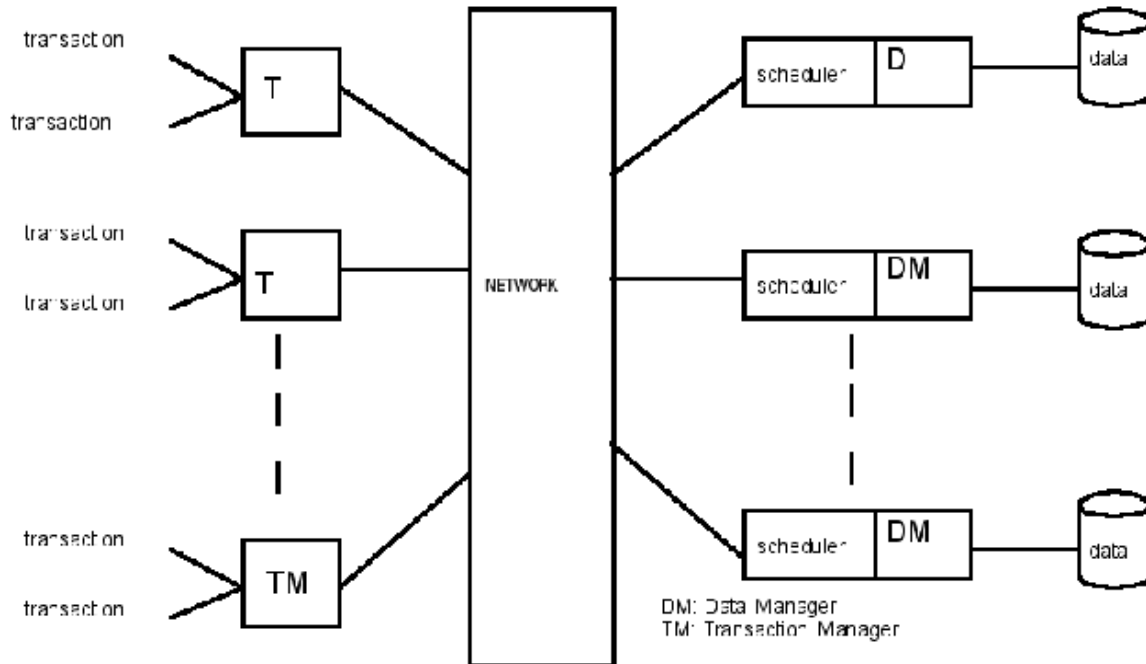
A) The different components of DDBMS are as follows:

- **Computer workstations** or remote devices (sites or nodes) that form the network system. The distributed database system must be independent of the computer system hardware.
- **Network hardware and software components** that reside in each workstation or device. The network components allow all sites to interact and exchange data. Because the components—computers, operating systems, network hardware, and so on—are likely to be supplied by different vendors, it is best to ensure that distributed database functions can be run on multiple platforms.
- **Communications media** that carry the data from one node to another. The DDBMS must be communications media-independent; that is, it must be able to support several types of communications media.

Distributed DBMS

- The **transaction processor (TP)**, which is the software component found in each computer or device that requests data. The transaction processor receives and processes the application's data requests (remote and local). The TP is also known as the application processor (AP) or the transaction manager (TM).
- The **data processor (DP)**, which is the software component residing on each computer or device that stores and retrieves data located at the site. The DP is also known as the data manager (DM). A data processor may even be a centralized DBMS.

The following Figure illustrates the placement of the components and the interaction among them. The communication among TPs and DPs shown in the figure is made possible through a specific set of rules, or protocols, used by the DDBMS.



The protocols determine how the distributed database system will:

- Interface with the network to transport data and commands between data processors (DPs) and transaction processors (TPs).
- Synchronize all data received from DPs (TP side) and route retrieved data to the appropriate TPs (DP side).
- Ensure common database functions in a distributed system. Such functions include security, concurrency control, backup, and recovery.

8Q) What is fragmentation?

Distributed DBMS

- A) The process of dividing the database into a smaller multiple part is called as **fragmentation**. These fragments may be stored at different locations. The data fragmentation process should be carried out in such a way that the reconstruction of original database from the fragments is possible.

Types of data Fragmentation:

There are three types of data fragmentation:

1. Horizontal data fragmentation

As the name suggests, here the data / records are fragmented horizontally. i.e.; horizontal subset of table data is created and are stored in different database in DDB.



For example, consider the employees working at different locations of the organization like India, USA, UK etc. number of employees from all these locations are not a small number. They are huge in number. When any details of any one employee are required, whole table needs to be accessed to get the information. Again the employee table may present in any location in the world. But the concept of DDB is to place the data in the nearest DB so that it will be accessed quickly. Hence what we do is divide the entire employee table data horizontally based on the location. i.e.;

```
SELECT * FROM EMPLOYEE WHERE EMP_LOCATION = 'INDIA';  
SELECT * FROM EMPLOYEE WHERE EMP_LOCATION = 'USA';  
SELECT * FROM EMPLOYEE WHERE EMP_LOCATION = 'UK';
```

Now these queries will give the subset of records from EMPLOYEE table depending on the location of the employees. These sub set of data will be stored in the DBs at respective locations. Any insert, update and delete on the employee records will be done on the DBs at their location and it will be synched with the main table at regular intervals.

Above is the simple example of horizontal fragmentation. This fragmentation can be done with more than one conditions joined by AND or OR clause. Fragmentation is done based on the requirement and the purpose of DDB.

2. Vertical Fragmentation

Distributed DBMS

This is the vertical subset of a relation. That means a relation / table is fragmented by considering the columns of it.



For example consider the EMPLOYEE table with ID, Name, Address, Age, location, DeptID, ProjID. The vertical fragmentation of this table may be dividing the table into different tables with one or more columns from EMPLOYEE.

```
SELECT EMP_ID, EMP_FIRST_NAME, EMP_LAST_NAME, AGE FROM EMPLOYEE;  
SELECT EMP_ID, STREETNUM, TOWN, STATE, COUNTRY, PIN FROM EMPLOYEE;  
SELECT EMP_ID, DEPTID FROM EMPLOYEE;  
SELECT EMP_ID, PROJID FROM EMPLOYEE;
```

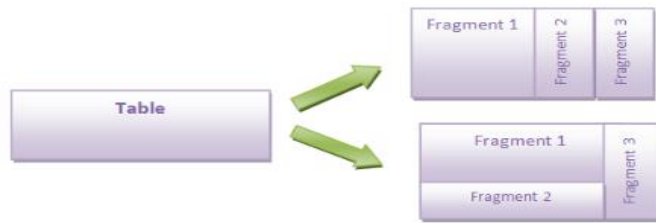
This type of fragment will have fragmented details about whole employee. This will be useful when the user needs to query only few details about the employee. For example consider a query to find the department of the employee. This can be done by querying the third fragment of the table. Consider a query to find the name and age of an employee whose ID is given. This can be done by querying first fragment of the table. This will avoid performing 'SELECT *' operation which will need lot of memory to query the whole table – to traverse whole data as well as to hold all the columns.

In this fragment overlapping columns can be seen but these columns are primary key and are hardly changed throughout the life cycle of the record. Hence maintaining cost of this overlapping column is very least. In addition this column is required if we need to reconstruct the table or to pull the data from two fragments. Hence it still meets the conditions of fragmentation.

3) Hybrid Fragmentation

This is the combination of horizontal as well as vertical fragmentation. This type of fragmentation will have horizontal fragmentation to have subset of data to be distributed over the DB, and vertical fragmentation to have subset of columns of the table.

Distributed DBMS



As we observe in above diagram, this type of fragmentation can be done in any order. It does not have any particular order. It is solely based on the user requirement. But it should satisfy fragmentation conditions.

Consider the EMPLOYEE table with below fragmentations.

```
SELECT EMP_ID, EMP_FIRST_NAME, EMP_LAST_NAME, AGE FROM EMPLOYEE WHERE  
EMP_LOCATION = 'INDIA;
```

```
SELECT EMP_ID, DEPTID FROM EMPLOYEE WHERE EMP_LOCATION = 'INDIA;
```

```
SELECT EMP_ID, EMP_FIRST_NAME, EMP_LAST_NAME, AGE FROM EMPLOYEE WHERE  
EMP_LOCATION = 'US;
```

```
SELECT EMP_ID, PROJID FROM EMPLOYEE WHERE EMP_LOCATION = 'US;
```

This is a hybrid or mixed fragmentation of EMPLOYEE table.

Q : Explain Replication in Distributed System?

A) Data replication is the process in which the data is copied at multiple locations (Different computers or servers) to improve the availability of data.

Goals of data replication

Data replication is done with an aim to:

- Increase the availability of data.
- Speed up the query evaluation.

Types of data replication

There are two types of data replication:

1. Synchronous Replication:

In synchronous replication, the replica will be modified immediately after some changes are made

Distributed DBMS

in the relation table. So there is no difference between original data and replica.

2. Asynchronous replication:

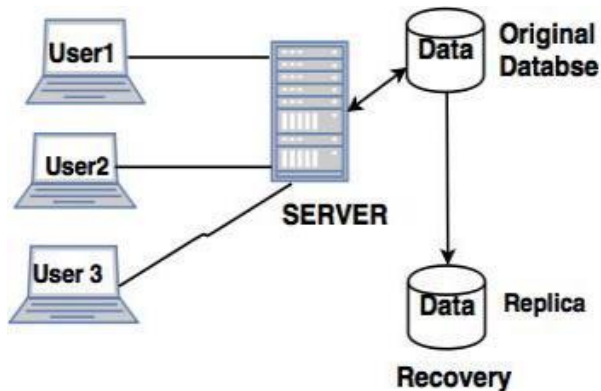
In asynchronous replication, the replica will be modified after commit is fired on to the database.

Replication Schemes

The three replication schemes are as follows:

1. Full Replication

In full replication scheme, the database is available to almost every location or user in communication network.



Full Replication Process In Distributed System

Advantages of full replication

- High availability of data, as database is available to almost every location.
- Faster execution of queries.

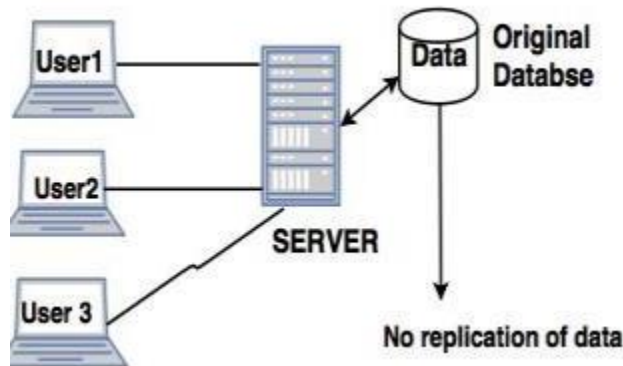
Disadvantages of full replication

- Concurrency control is difficult to achieve in full replication.
- Update operation is slower.

2. No Replication

No replication means, each fragment is stored exactly at one location.

Distributed DBMS



Advantages of no replication

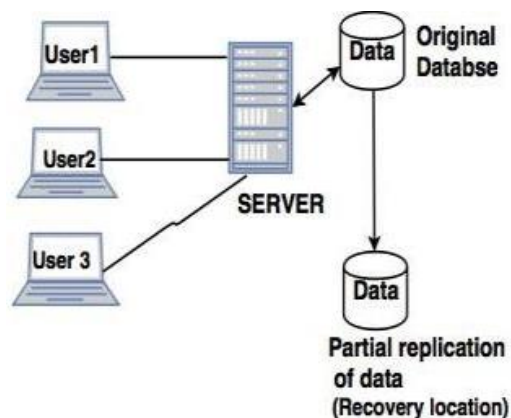
- Concurrency can be minimized.
- Easy recovery of data.

Disadvantages of no replication

- Poor availability of data.
- Slows down the query execution process, as multiple clients are accessing the same server.

3. Partial replication

Partial replication means only some fragments are replicated from the database.



Partial Replication Process In Distributed System

Advantages of partial replication

The number of replicas created for fragments depend upon the importance of data in that fragment.

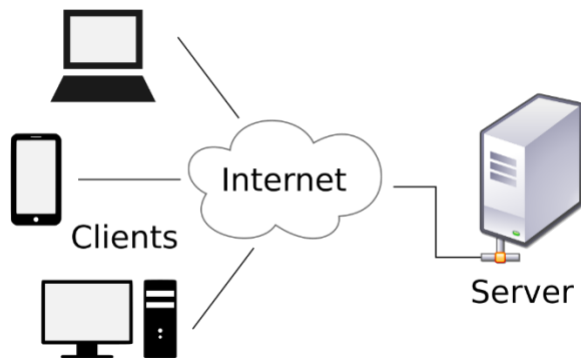
A) What is Client Server Architecture

A) Definition – Client-server architecture is also called of the “Client/Server Network” or “**Network computing Model**”, because in this architecture all services and requests are

Distributed DBMS

spread over the network. Its functionality like as **distributed computing system** because in which all components are performing their tasks independently from each other.

Client-server architecture is a shared computer network architecture where several clients (remote system) send many requests and finally to obtained services from the centralized server machine (host system). Client machine delivers user-friendly interface that helps to users to fire request services of server computer and finally to show your output on client system.



Types of Client-Server Architecture

1-Tier Architecture

In the 1-tier architecture, all client/server configuration setting, user interface environment, data logic, and marketing logic system are existed on the same system. These types of services are reliable but it is very difficult tasks to handle because they contain all data in different variance, which are allotted the replication of entire work. This architecture also contain the different layers.



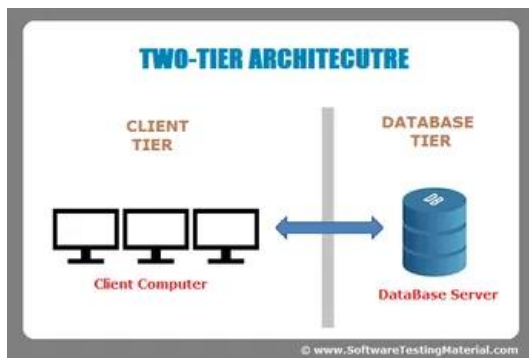
For example – Presentation, Business, Data Access layer with using of single software package. All data is saved on the local machine. Some applications, which manage all three

Distributed DBMS

tiers like as MP3 player, MS Office; but these types of applications are presented under 1-tier architecture applications.

2-Tier Architecture

2-tier architecture provides the best client/server environment that helps to store user interface on the client system and all database is saved on the **server machine**. Business logic and database logic are existed on the client otherwise server, but they are required to be maintained. When data logic and business are gathered on the client terminal then it is known as “fat client thin server architecture”. But if Business Logic and Data Logic are controlled at the server machine then it is known as “thin client fat server architecture”.



In this architecture, client and server machines are connected directly incorporation because if client is firing any input for server terminal then in between should not any intermediate. So, it delivers the output with fastest rate and to ignore misunderstanding between the other clients. **For example** – online ticket reservations program, in which 2-tier architecture is used.

Benefits Are

- Easy to design all applications
- Maximum user satisfaction
- Implementation of Homogeneous Environment
- Best performance

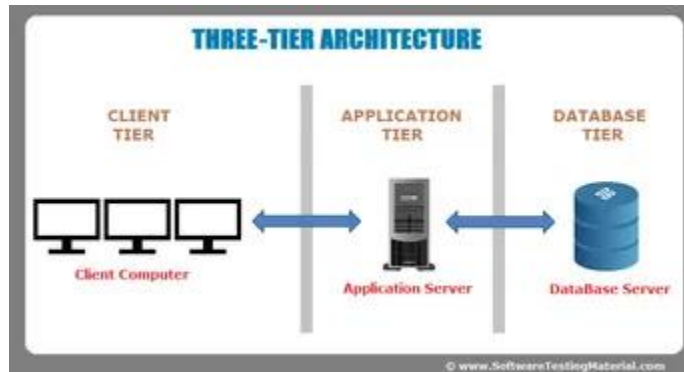
Limitations Are

- Poor performance due to grow number of connections of each user
- Less security
- All clients are totally dependent upon the manufacturer's database.
- Less portability means this architecture is totally dependent upon the particular database.

3-Tier Architecture

Distributed DBMS

In this 3-tier architecture, **middleware** is needed because if client machine sends the request to server machine then firstly this request is received by middle layer, and finally this request is obtained to server. So, firstly response of server is received by middle layer then it is obtained to client machine. All data logic and business logic are stored on the middleware. Due to use of middleware, to improve its flexibility and deliver excellent performance.



3-tier architecture is divided into 3 layers such as presentation layer (Client Tier), Application layer (Business Tier) and Database layer (Data Tier). Client machine handles the presentation layer, Application layer controls the Application layer, and finally Server machine takes care of Database layer.

Benefits Are

- Best performed data integrity
- Improved security to 2-tier architecture
- Hide database structure

Limitation is:

To increase complexity of communication in between client and server because in which middleware is also used

N-Tier Architecture

This architecture is also known as the “Multitier Architecture”, so it is scaled form of 3-tier architecture. In this architecture, entire presentations, application processing, and data management functions are isolated from each other.

Benefit is

It delivers the flexible and reusable applications.

Limitation is

Harder to implement because it uses the complex structure (componentization of tiers)

Distributed DBMS

Examples of Client Server Architecture

There are four **examples of Client Server Architecture**. Below explain each one –

Web Servers – Web server likes as high performance computer system that can host multiples websites. On this server, to install different types of web server softwares like as Apache or Microsoft IIS, which delivers access to hosted several websites on the internet, and these servers are linked with internet through higher speed connection that delivers ultra data transmission rates.

Mail Servers – Email servers helps to send and receive all emails. Some softwares are run on the mail server which allow to administrator to create and handle all email accounts for any domain that is hosted on the server. Mail servers use the some protocols for sending and receiving emails such as SMTP, IMAP, and POP3. **SMTP protocol** helps to fire messages and manages all outgoing email requests. IMAP and **POP3** help to receive all messages and handle all incoming mails.

File Servers – File server is dedicated systems that allow users to access for all files. It works like as centralized file storage location, and it can be accessed by several terminal systems.

DNS – DNS stands for “Domain Name Server”, and it has huge database of different types of public IP addresses, and they link with their hostnames

These **types of server** help to deliver all resources (like as files, directories, shared devices such as applications and printers) to client terminal like as PCs, smart phones, PDAs, laptops, tablets etc.

Components of Client Server Architecture

Client-server architecture contains three components such as **workstations, server, and networking devices**, and they are connected with each other.

Workstation – Workstation is also known as “**Client Computer**”. There are different **types of operating systems**, which are installed on the workstations like as Windows 2000, Windows XP, Windows Vista, Windows 7, and Windows 10. These workstation operating systems are cheaper compare to server’s operating systems.

Server – **Server** is a ultra performer computer system that contains the fastest memory, more hard drive space, and faster speed **processors** because they save and service of several requests which are coming from workstation side. A server plays different types of roles like as mail server, database server, file server, and domain controller at the same time duration.

Distributed DBMS

Network Devices – With the help of **network devices**, workstations and servers are connected with each other. Every network device has own functionality like as hub is used for making connection between server to multiple workstations, repeater is used for moving data from one devices to another device, and bridges helps to isolate of all network segments.