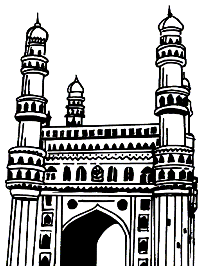
Rahul's ✔
Topper's Voice

AS PER CBCS SYLLABUS



B.Com.

II Year III Sem

(All Universities in Telangana)

Latest 2022 Edition

RELATIONAL DATABASE MANAGEMENT SYSTEM

- ☞ **Study Manual**
- ☞ **FAQ's & Important Questions**
- ☞ **Short Question and Answers**
- ☞ **Choose the Correct Answer**
- ☞ **Fill in the blanks**
- ☞ **Solved Previous Question Papers**
- ☞ **Solved Model Papers**
- ☞ **Lab Programs**

Price : ` .179/-

- by -

WELL EXPERIENCED LECTURER

Rahul Publications™
Hyderabad. Ph : 66550071, 9391018098

B.Com.

II Year III Sem

RELATIONAL DATABASE MANAGEMENT SYSTEM

Price `. 179-00

Sole Distributors : **☎ : 66550071, Cell : 9391018098**

VASU BOOK CENTRE

Shop No. 2, Beside Gokul Chat, Koti, Hyderabad.

Maternity Hospital Opp. Lane, Narayan Naik Complex, Koti, Hyderabad.
Near Andhra Bank, Subway, Sultan Bazar, Koti, Hyderabad -195.

CONTENTS

RELATIONAL DATABASE MANAGEMENT SYSTEM

SYLLABUS

UNIT - I

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.

UNIT - II

DATABASE INTEGRITY AND NORMALISATION: Relational Database Integrity - TheKeys - 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 - Losslessjoin 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.

UNIT - III

STRUCTURES QUERY LANGUAGE (SQL): Meaning–SQL commands - DataDefinition 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.

UNIT - IV

TRANSACTIONS AND CONCURRENCY MANAGEMENT: Transactions - ConcurrentTransactions - 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.

UNIT - V

DISTRIBUTED AND CLIENT SERVER DATABASES: Need for Distributed DatabaseSystems - 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.

ADVANCED TOPICS: Overview: Parallel Database - Multimedia Database - Mobile Database - Web Database - Multidimensional Database. Data Warehouse - OLTP Vs OLAP - NoSQL Database.

LAB: SQL QUERIES BASED ON VARIOUS COMMANDS.

Contents

UNIT - I

Topic **Page No.**

Frequently Asked & Important Questions

UNIT - I

1. **What are the advantages of DBMS over File based system ?**

Ans : **(Oct.-20, Oct.-19, June-19, June-8, Imp.)**

Refer Unit-I, Q.No. 8

2. **Explain the Logical Architecture of DBMS?**

Ans : **(July - 21, Imp.)**

Refer Unit-I, Q.No. 10

3. **Explain in detail about ER – Model ?**

Ans : **(July-21,Oct.-20, Oct.-19,June-18, Imp.)**

Refer Unit-I, Q.No. 28

4. **Who is DBA ? What is the role and responsibilities DBA ?**

Ans : **(June-19, Imp.)**

Refer Unit-I, Q.No. 14

5. **What are the functions of DBA ?**

Ans : **(June-19, Imp.)**

Refer Unit-I, Q.No. 15

UNIT - II

1. **What is Normalization?**

Ans : **(July-21)**

Refer Unit-II, Q.No. 8

2. **Explain briefly about 1NF.**

Ans : **(July-21, Oct.-20, Oct.-19, June-19, June-18, Imp.)**

Refer Unit-II, Q.No. 10

3. **Explain briefly about 2NF.**

Ans : **(July-11, Oct.-20, Oct.-19, June-19, June-18, Imp.)**

Refer Unit-II, Q.No. 11

4. Explain briefly about 3NF.

Ans : **(Oct.-20, Oct.-19, June-19, Imp.)**

Refer Unit-II, Q.No. 12

5. What is File Organization ? What are the objectives of file organization.

Ans : **(June-18)**

Refer Unit-II, Q.No. 21

6. Explain different types of File Organizations in DBMS.

Ans : **(July-21, Oct.-20, Oct.-19, June-18, Imp.)**

Refer Unit-II, Q.No. 22

UNIT - III

1. Describe in detail about DDL commands in SQL ?

Ans : **(July-21, Oct.-20, June -19, June-18, Imp.)**

Refer Unit-III, Q.No. 7

2. Explain in detail about DML commands ?

Ans : **(Oct.-19, June-18, Imp.)**

Refer Unit-III, Q.No. 8

3. Explain in detail about joins in SQL ?

Ans : **(Oct.-20, June-19, Imp.)**

Refer Unit-III, Q.No. 21

4. Describe briefly about SQL Views? How can we create views in SQL?

Ans : **(July-21, Oct.-19, Imp.)**

Refer Unit-III, Q.No. 23

5. Explain sequences in SQL Queries ?

Ans : **(July-21, Imp.)**

Refer Unit-III, Q.No. 24

UNIT - IV

1. Define lock? Explain different types of locks.

Ans : **(July-21, Imp.)**

Refer Unit-IV, Q.No. 5

2. Explain in detail about Two phase Locking (2PL) Protocol ?

Ans : **(July-21, Oct.-20, Imp.)**

Refer Unit-IV, Q.No. 9

3. What is meant by Deadlock? State the prevention of dead lock.

Ans : **(Oct.-19, June-18)**

Refer Unit-IV, Q.No. 10

4. Explain in detail about Optimistic concurrency control ?

Ans : **(June-19, Imp.)**

Refer Unit-IV, Q.No. 11

5. Define Database Recovery ?

Ans : **(June-19, Imp.)**

Refer Unit-IV, Q.No. 12

6. How can we Backup our Database ?

Ans : **(Oct.-20, June-18)**

Refer Unit-IV, Q.No. 16

7. Describe about Database Security and Integrity

Ans : **(June-18, Imp.)**

Refer Unit-IV, Q.No. 18

UNIT - V

1. What is the need of Distributed Database ?

Ans : **(Oct.-19, Imp.)**

Refer Unit-V, Q.No. 4

2. Explain logical structure of Distributed Database.

Ans : **(July-21, Imp.)**

Refer Unit-V, Q.No. 5

3. List out the advantages and disadvantages of DDBMS.

Ans : **(Oct.-20, June-19, Imp.)**

Refer Unit-V, Q.No. 6

4. What is the need of client server computing ? Explain its basic characteristics and advantages and disadvantages of client server computing.

Ans : **(July-21,Oct.-20, June-18, Imp.)**

Refer Unit-V, Q.No. 15

5. Explain the structure of client server system and list out its advantages and disadvantages.

Ans : **(Oct.-19, June-19, Imp.)**

Refer Unit-V, Q.No. 16

UNIT I

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.

1.1 Database Management System

Q1. Define the terms

(a) Data

(b) Information

(c) Data process

(d) Database

(e) Database system

(f) DBMS

Ans :

(a) Data

Data can be defined as a representation of facts, concepts, or instructions in a formalized manner, which should be suitable for communication, interpretation, or processing by human or electronic machine.

Data is represented with the help of characters such as alphabets (A-Z, a-z), digits (0-9) or special characters (+,-,/,*,<,>,= etc.)

(b) Information

Information is organized or classified data, which has some meaningful values for the receiver. Information is the processed data on which decisions and actions are based.

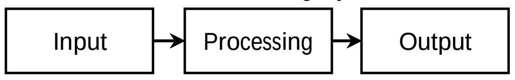
For the decision to be meaningful, the processed data must qualify for the following characteristics :-

- **Timely:** Information should be available when required.
- **Accuracy:** Information should be accurate.
- **Completeness:** Information should be complete.

(c) Data Processing

Data processing is the re-structuring or re-ordering of data by people or machine to increase their usefulness and add values for a particular purpose. Data processing consists of the following basic steps - input, processing, and output. These three steps constitute the data processing cycle.

Data Processing Cycle



- **Input:** In this step, the input data is prepared in some convenient form for processing. The form will depend on the processing machine. For example, when electronic computers are used, the input data can be recorded on any one of the several types of input medium, such as magnetic disks, tapes, and so on.
- **Processing:** In this step, the input data is changed to produce data in a more useful form. For example, pay-checks can be calculated from the time cards, or a summary of sales for the month can be calculated from the sales orders.

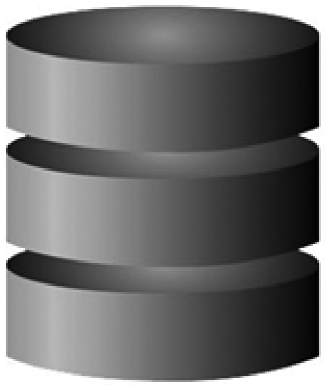
- **Output:** At this stage, the result of the proceeding processing step is collected. The particular form of the output data depends on the use of the data. For example, output data may be pay-checks for employees.

(d) Database

A database is a collection of information that is organized so that it can be easily accessed, managed and updated. **Data** is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information.

A database is a collection of **information** that is organized so that it can be easily accessed, managed and updated.

Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it.



Alternatively referred to as a databank or a datastore, and sometimes abbreviated as a DB, a database is a large quantity of indexed digital information. It can be searched, referenced, compared, changed or otherwise manipulated with optimal speed and minimal processing expense.

A database is built and maintained by using a database programming language. The most common database language is **SQL,** but there are multiple "flavors" of SQL, depending on the type of database being used. Each flavor of SQL has differences in the SQL syntax and are designed to be used with a specific type of database. For example, an **Oracle database** uses PL/SQL and Oracle SQL (Oracle's version of SQL). A Microsoft database uses T-SQL (Transact-SQL).

Components

A database is made up of several main components.

- **Schema:** A database contains one or more **schemes**, which is basically a collection of one or more **tables** of data.
- **Table:** Each table contains multiple columns, which are similar to columns in a spreadsheet. A table can have as little as two columns and as many as one hundred or more columns, depending on the type of data being stored in the table.
- **Column:** Each column contains one of several types of data or values, like dates, numeric or integer values, and alphanumeric values (also known as varchar).
- **Row:** Data in a table is listed in rows, which are like rows of data in a spreadsheet. Often there are hundreds or thousands of rows of data in a table.

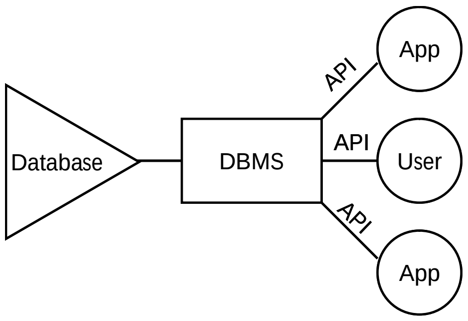
(e) Database System

The system which is used to create and manage the information as per the requirements of the user/client is called Database System.

(f) Database Management System (DBMS)

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.

A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible.



Q2. What is DBMS? What are the characteristics of DBMS? Write its advantages and disadvantages ?

Ans : **(Imp.)**

Database Management System

- Database management system is a software which is used to manage the database.

  For example: MySQL, Oracle, etc are a very popular commercial database which is used in different applications.

- DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
- It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

DBMS allows users the following tasks:

- **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
- **Data Updation:** It is used for the insertion, modification, and deletion of the actual data in the database.
- **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.
- **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

Characteristics

- It uses a digital repository established on a server to store and manage the information.
- It can provide a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.
- It contains ACID properties which maintain data in a healthy state in case of failure.
- It can reduce the complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It can view the database from different viewpoints according to the requirements of the user.

Advantages

- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.

- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **Multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces.

Disadvantages

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

Q3. What are the functions of DBMS?

Ans :

1. Data Dictionary Management

This function refers to managing the data dictionary in which definitions of several data elements and their corresponding relationships are stored by DBMS. The DBMS helps in executing the programs which are used for accessing the data within the database environment. The user doesn't need to code the complex relationships in every program because it identifies the data elements and their relationships by using the data dictionary. If changes are made to the data in the database structure, then these changes are automatically stored in the data dictionary. This in turn removes the necessity of updating each program that successes the changed structure. Consequently, it can be said that data abstraction is provided by the DBMS, which in turn eliminates the structural and data dependency issues from the system.

2. Data Storage Management

This function refers to management and creation of complex data structures, which are stored within the database structure. Thus, it avoids the need to define and organize the physical data characteristics. An advanced Data Storage Management provides the storage structure for storing data, reporting definitions, screening definitions, defining structures to handle video and picture formats, defining data validation rules, etc. It. helps in the database performance tuning, with is a set of activities using which the storage and access speed of the database increases significantly.

From the user's perspective, the database is viewed as a single data storage component. But, actually the database is stored in several physical data files by the DBMS. These files are stored on several storage devices using which the DBMS can access the database requests without waiting for a disk request to complete. This means it can respond to the database requests in a concurrent manner.

3. Data Transformation and Presentation

This function is used to transform the data in such a way that it is conformed to the required data structures. It also presents the data in such a way that it provides a clear distinction between the physical data format and logical data format.

4. Security Management

This function provides a security system, which enforces data privacy and user security rules. These rules specify the users that can access the database, the data operations that can be performed by the user and the data items that can be accessed by the users. Thus, security rules are very much important in multiuser database systems where several users access the database concurrently. The DBMS provides an authenticated username and password or a biometric authentication like a fingerprint scan to every user so that he/she can access the database components like queries and reports.

5. Multi-user Access Control

This function refers to the management of multiple users that access the database simultaneously without the loss of data consistency and data integrity. This can be achieved by DBMS which uses complex and sophisticated algorithms.

1.2 File Based System

Q4. Explain in detail about File based system ?

Ans :

File Based Systems or File Management System is one of the first attempt to store information and organizing it into compute base storing device like a hard disk, an SSD, a CD-ROM, a USB key, a floppy disk, etc.

File based systems can use indexing system to locate the information. Such file management makes it possible to process, preserve large amounts of data as well as share them between several computer programs. It offers the user an abstract view of their data and allows them to be located from a path.

Through this system files can be *saved*, *read*, *changed* or *deleted data and* easy *retrieval* and *secure* storage are essential tasks of a file system.

Files in a file system almost always have at least one file name and attributes that give more information about the file. These file names are stored in directories. All files can be accessed via their file names that serves as a unique address.

Thus, through a File Management System one can easily find a file name (and thus a file) as well as the data belonging to the file. A file system thus forms a namespace. The name of a file and other information associated with the stored files are called metadata.

Functions

- The computer's file system is a method of storing and organizing computer data. It makes it easy to access and find it.
- Its main functions are the allocation of space to the files and the administration of free space and access to protected data.
- They structure the information stored in a data storage device or storage unit which will then be represented either textually or graphically using a file manager. It divides the space in the device into blocks of a specific size.
- The data is stored in these blocks and the size is modified to occupy an integer number of blocks. The file system software is responsible for organizing these blocks into files and directories and recording which blocks are assigned to which files and which blocks are not used.
- In the concept, the user uses the file system to save data without having to care about the data blocks actually stored in the address of the hard disk (or optical disk), and only needs to remember the directory and file name of the file.

Q5. Explain the types of file based system and list out the limitations of File based system?

Ans :

Types of File Based Systems

(i) File Allocation Table

File Allocation Table is a Microsoft invented file system for MS-DOS use. The FAT file system is not complicated, so almost all of the PC's operating systems are supported. This feature makes it an ideal file system for floppy disks and memory cards, as well as for the exchange of data in different operating systems.

However, FAT has a serious drawback: when the file is deleted and new data is written, FAT will not organize the file into complete segments and then write it. After long-term use, the file data will gradually become scattered, which slows down the reading and writing speed.

(ii) High-performance file system

High-performance file system is IBM 's file system for OS/2 operating system. It is the prototype of NTFS, split due to disagreement between Microsoft and IBM in the development of OS/2.

(iii) New Technology File System

New Technology File System is a dedicated file system developed by Microsoft Corporation and has been a standard file system of the Windows NT family since Windows NT 3.1

NTFS replaces FAT and HPFS and carries out a series of improvements, such as enhanced metadata support, use of more advanced data structures to improve performance, reliability, and disk space utilization, and comes with a series of enhancements, such as access control lists (ACLs) and file system logs.

(iv) Cluster file system

A cluster file system is a file system that runs on multiple computers and communicates with each other in a certain way so that all the cluster storage resource consolidation, virtualization, and provide file access services outside of the file system.

Unlike the purpose of local file systems such as NTFS, the cluster file system runs in a cluster environment, while the local file system runs in a stand-alone environment, purely between the management block and the file mapping and file attributes.

(v) Disk file systems

A disk file system is designed for the storage of files on a disk drive , which may be directly or indirectly connected to the computer .

(vi) Network file systems

A network file system is the one that accesses your files through a network of computers .

(vii) MacOS file system

The file systems used by MacOS have their peculiar way of working, which is totally different from that of Microsoft Windows and GNU / Linux, they work through hierarchies.

Limitations

- Due to the storage of the data in isolated files it becomes difficult to access them, especially when the data has to be retrieved from more than two files as a large amount of data has to be searched.
- Due to the decentralised approach, the file system leads to duplication of data that leads to wastage of a lot of storage space. It also uneconomical to enter the data more than once.
- The data in a file system can become inconsistent if more than one person modifies the data concurrently.
- The physical structure and storage of data files and records are defined in the application code. This makes it extremely difficult to make changes to the existing structure as the programmer would have to identify all the affected programs, modify them and retest them. This is known as data dependence.
- The structure of the files generated by a particular programing languages may be quite different from other programming language. This incompatibility makes them difficult to process them jointly.

Besides the above, the maintenance of the File Based System is difficult and there is no provision for security. In order to overcome the limitations of a file system, a new approach was required. Hence a database approach emerged.

Q6. Compare and Contrast DBMS and File based system?

Ans :

DBMS	File Based System
DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	File system is a collection of data. In this system, the user has to write the procedures for managing the database.
DBMS gives an abstract view of data that hides the details.	File system provides the detail of the data representation and storage of data.
DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure.	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 lost.
DBMS provides a good protection mechanism.	It is very difficult to protect a file under the file system.
DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	File system can't efficiently store and retrieve the data.
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 other deleting some information or updating some information.

Q7. List out advantages and disadvantages of File Based System ?

Ans :

Advantages of File-Based System

1. **Backup**
   - ➢ It is possible to take faster and automatic back-up of database stored in files of computer-based systems.
   - ➢ computer systems provide functionalities to serve this purpose. It is also possible to develop specific application program for this purpose.
2. **Compactness**
   - ➢ It is possible to store data compactly.
3. **Data Retrieval**
   - ➢ Computer-based systems provide enhanced data retrieval techniques to retrieve data stored in files in easy and efficient way.
4. **Editing**
   - ➢ It is easy to edit any information stored in computers in form of files.
   - ➢ Specific application programs or editing software can be used for this purpose.
5. **Remote Access**
   - ➢ In computer-based systems, it is possible to access data remotely.

- so, to access data it is not necessary for a user to remain present at location where these data are kept.

6. Sharing

- Data stored in files of computer-based systems can be shared among multiple users at a same time.

Disadvantages of File-Based System:

1. Data Redundancy

- It is possible that the same information may be duplicated in different files. This leads to data redundancy results in memory wastage.

2. Data Inconsistency

- Because of data redundancy, it is possible that data may not be in consistent state.

3. Difficulty in Accessing Data

- Accessing data is not convenient and efficient in file processing system.

4. Limited Data Sharing

- Data are scattered in various files also different files may have different formats and these files may be stored in different folders may be of different departments.
- So, due to this data isolation, it is difficult to share data among different applications.

5. Integrity Problems

- Data integrity means that the data contained in the database in both correct and consistent for this purpose the data stored in database must satisfy correct and constraints.

6. Atomicity Problems

- Any operation on database must be atomic.
- this means, it must happen in its entirely or not at all.

7. Concurrent Access Anomalies

- Multiple users are allowed to access data simultaneously. This is for the sake of better performance and faster response.

8. Security Problems

- Database should be accessible to users in limited way.
- Each user should be allowed to access data concerning his requirements only.

1.3 Advantages of DBMS over File Based System

Q8. What are the advantages of DBMS over File based system ?

Ans : **(Oct.-20, Oct.-19, June-19, June-8, Imp.)**

Advantages of DBMS over File Based System

1. Data redundancy and inconsistency

Redundancy is the concept of repetition of data i.e. each data may have more than a single copy. The file system cannot control redundancy of data as each user defines and maintains the needed files for a specific application to run. There may be a possibility that two users are maintaining same files data for different applications.

Hence changes made by one user does not reflect in files used by second users, which leads to inconsistency of data. Whereas DBMS controls redundancy by maintaining a single repository of data that is defined once and is accessed by many users. As there is no or less redundancy, data remains consistent.

2. Data sharing

File system does not allow sharing of data or sharing is too complex. Whereas in DBMS, data can be shared easily due to centralized system.

3. Data concurrency

Concurrent access to data means more than one user is accessing the same data at the same time. Anomalies occur when changes made by one user gets lost because of changes made by other user. File system does not provide any procedure to stop anomalies. Whereas DBMS provides a locking system to stop anomalies to occur.

4. **Data searching**

For every search operation performed on file system, a different application program has to be written. While DBMS provides inbuilt searching operations. User only have to write a small query to retrieve data from database.

5. **Data integrity**

There may be cases when some constraints need to be applied on the data before inserting it in database. The file system does not provide any procedure to check these constraints automatically. Whereas DBMS maintains data integrity by enforcing user defined constraints on data by itself.

1.4 Database Approach

Q9. Explain in detail about Database Approach.

Ans :

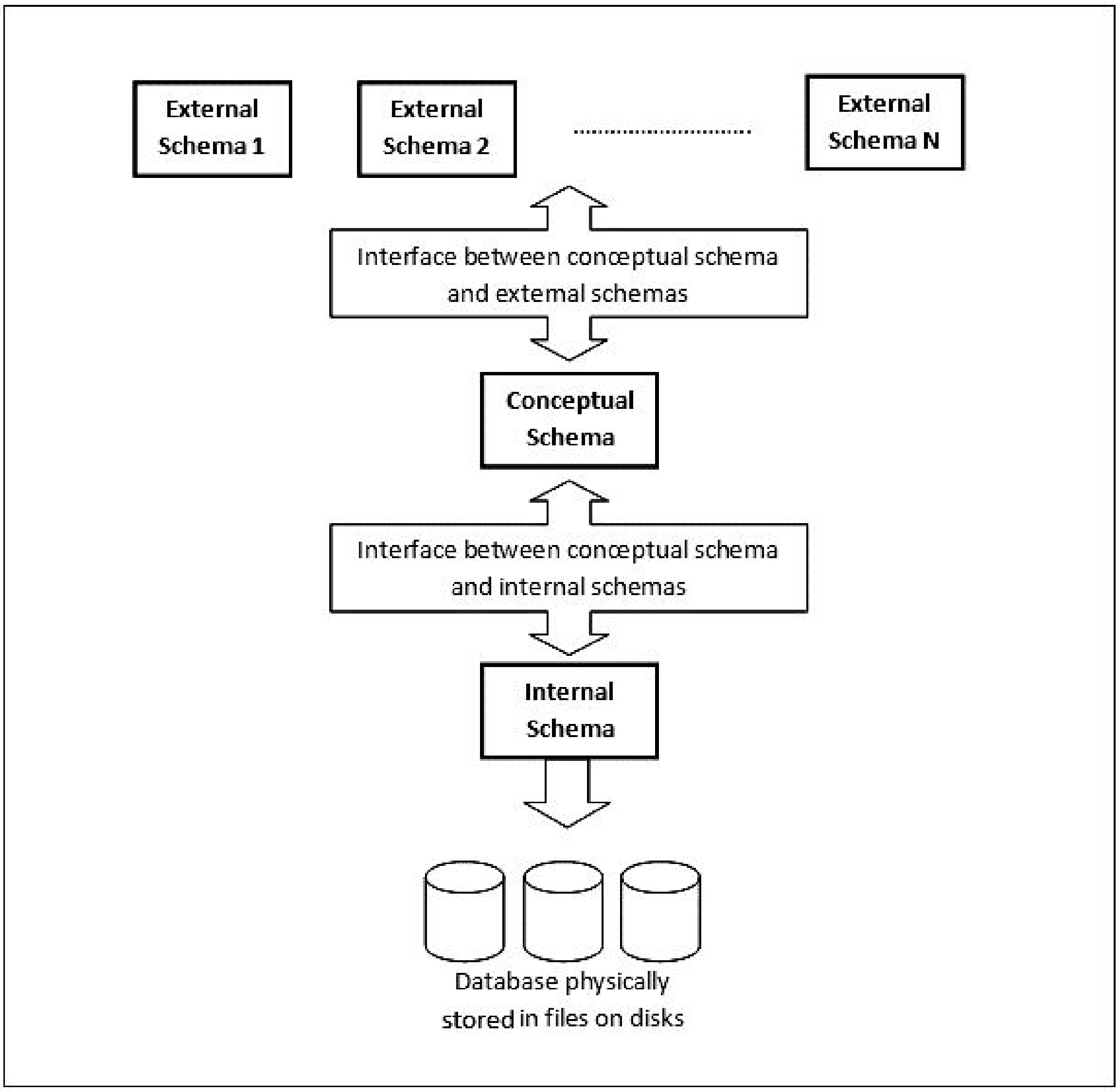
The Database Approach

The database approach is an improvement on the shared file solution as the use of a database management system (DBMS) provides facilities for querying, data security and integrity, and allows simultaneous access to data by a number of different users. At this point we should explain some important terminology:

- **Database:** A database is a collection of related data.
- **Database management system:** The term ‘database management system’, often abbreviated to DBMS, refers to a software system used to create and manage databases. The software of such systems is complex, consisting of a number of different components, which are described later in this chapter. The term database system is usually an alternative term for database management system.
- **System catalogue/Data dictionary:** The description of the data in the database management system.
- **Database application:** Database application refers to a program, or related set of programs, which use the database management system to perform the computer-related tasks of a particular business function, such as order processing.

One of the benefits of the database approach is that the problem of physical data dependence is resolved; this means that the underlying structure of a data file can be changed without the application programs needing amendment. This is achieved by a hierarchy of levels of data specification. Each such specification of data in a database system is called a schema. The different levels of schema provided in database systems are described below.

The Systems Planning and Requirements Committee of the American National Standards Institute encapsulated the concept of schema in its three-level database architecture model, known as the ANSI/ SPARC architecture, which is shown in the diagram below:


External Schema 1
External Schema 2
External Schema N
Interface between conceptual schema and external schemas
Conceptual Schema
Interface between conceptual schema and internal schemas
Internal Schema
Database physically stored in files on disks

(i) The External Schema

The external schemas describe the database as it is seen by the user, and the user applications. The external schema maps onto the conceptual schema, which is described below.

There may be many external schemas, each reflecting a simplified model of the world, as seen by particular applications. External schemas may be modified, or new ones created, without the need to make alterations to the physical storage of data. The interface between the external schema and the conceptual schema can be amended to accommodate any such changes.

The external schema allows the application programs to see as much of the data as they require, while excluding other items that are not relevant to that application. In this way, the external schema provides a view of the data that corresponds to the nature of each task.

The external schema is more than a subset of the conceptual schema. While items in the external schema must be derivable from the conceptual schema, this could be a complicated process, involving computation and other activities.

(ii) The Conceptual Schema

The conceptual schema describes the universe of interest to the users of the database system. For a company, for example, it would provide a description of all of the data required to be stored in a database system. From this organisation-wide description of the data, external schemas can be derived to provide the data for specific users or to support particular tasks.

At the level of the conceptual schema we are concerned with the data itself, rather than storage or the way data is physically accessed on disk. The definition of storage and access details is the preserve of the internal schema.

(iii) The Internal Schema

A database will have only one internal schema, which contains definitions of the way in which data is physically stored. The interface between the internal schema and the conceptual schema identifies how an element in the conceptual schema is stored, and how it may be accessed.

If the internal schema is changed, this will need to be addressed in the interface between the internal and the conceptual schemas, but the conceptual and external schemas will not need to change. This means that changes in physical storage devices such as disks, and changes in the way files are organised on storage devices, are transparent to users and application programs.

In distinguishing between 'logical' and 'physical' views of a system, it should be noted that the difference could depend on the nature of the user. While 'logical' describes the user angle, and 'physical' relates to the computer view, database designers may regard relations (for staff records) as logical and the database itself as physical. This may contrast with the perspective of a systems programmer, who may consider data files as logical in concept, but their implementation on magnetic disks in cylinders, tracks and sectors as physical.

(iv) Physical Data Independence

In a database environment, if there is a requirement to change the structure of a particular file of data held on disk, this will be recorded in the internal schema. The interface between the internal schema and the conceptual schema will be amended to reflect this, but there will be no need to change the external schema. This means that any such change of physical data storage is not transparent to users and application programs. This approach removes the problem of physical data dependence.

(v) Logical Data Independence

Any changes to the conceptual schema can be isolated from the external schema and the internal schema; such changes will be reflected in the interface between the conceptual schema and the other levels. This achieves logical data independence. What this means, effectively, is that changes can be made at the conceptual level, where the overall model of an organisation's data is specified, and these changes can be made independently of both the physical storage level, and the external level seen by individual users. The changes are handled by the interfaces between the conceptual, middle layer, and the physical and external layers.

1.5 Logical DBMS Architecture (3 Level Architecture of DBMS (or) Logical DBMS Architecture)

Q10. Explain the Logical Architecture of DBMS?

(OR)

Explain three level architecture of DBMS.

Ans : **(July - 21, Imp.)**

This framework is used to describe the structure of a specific database system. The three schema architecture is also used to separate the user applications and physical database. The three schema architecture contains three-levels. It breaks the database down into three different categories.

Following are the three levels of database architecture:

1. Physical Level
2. Conceptual Level
3. External Level

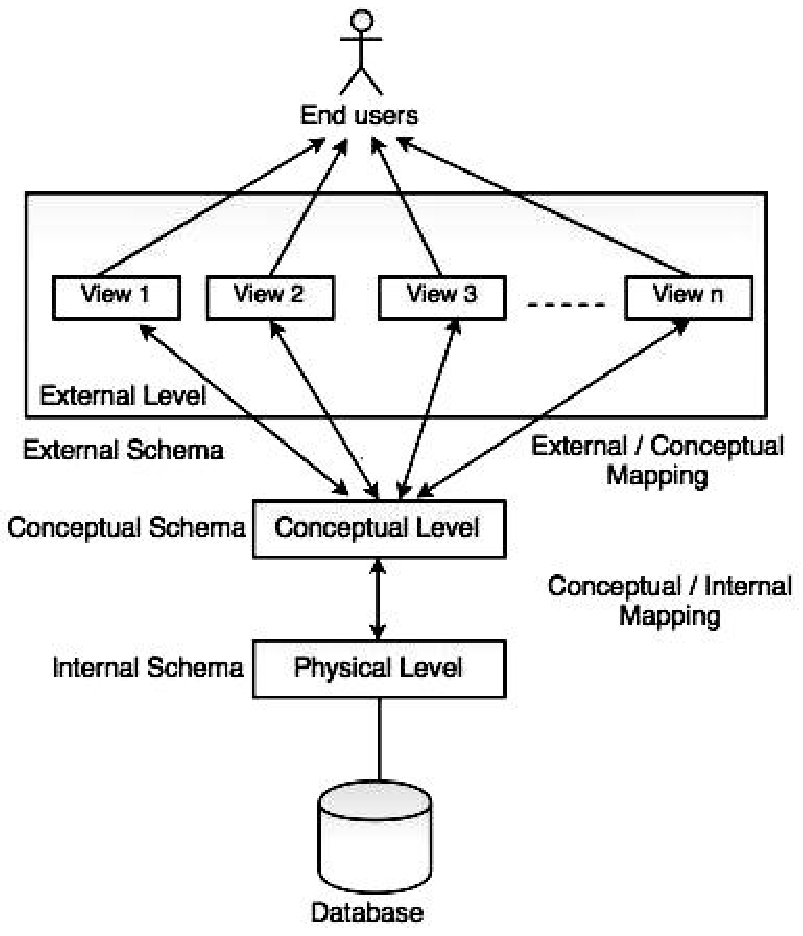


Fig. : Three Level Architecture of DBMS

In the above diagram,

- It shows the architecture of DBMS.
- Mapping is the process of transforming request response between various database levels of architecture.
- Mapping is not good for small database, because it takes more time.
- In External / Conceptual mapping, DBMS transforms a request on an external schema against the conceptual schema.
- In Conceptual / Internal mapping, it is necessary to transform the request from the conceptual to internal levels.

1. Physical Level

- Physical level describes the physical storage structure of data in database.

- It is also known as Internal Level.
- This level is very close to physical storage of data.
- At lowest level, it is stored in the form of bits with the physical addresses on the secondary storage device.
- At highest level, it can be viewed in the form of files.
- The internal schema defines the various stored data types. It uses a physical data model.

2. Conceptual Level

- Conceptual level describes the structure of the whole database for a group of users.
- It is also called as the data model.
- Conceptual schema is a representation of the entire content of the database.
- These schema contains all the information to build relevant external records.
- It hides the internal details of physical storage.

3. External Level

- External level is related to the data which is viewed by individual end users.
- This level includes a no. of user views or external schemas.
- This level is closest to the user.
- External view describes the segment of the database that is required for a particular user group and hides the rest of the database from that user group.

1.5.1 Need for Three Level Architecture

Q11. Explain the need for three level architecture ?

Ans :

The need for employing a three level DBMS architecture is to provide a platform on which users can efficiently access database irrespective of its physical representation or structure. The reasons for providing such a platform are,

- To provide different views of same data to different users.
- To facilitate the user in accessing the data in desired way without affecting others.
- To make the database access independent of storage factors such as indexing, hashing etc.
- To make the DBA (Database Administrator) capable of modifying storage structure without affecting the users.
- To make the internal structure of data independent of modifications made on it externally such as change in location,
- To make the DBA capable of modifying the conceptual database structure without affecting users.

1.6 Physical DBMS Architecture

Q12. Discuss in brief about physical centralized database structure.

Ans : **(Imp.)**

Centralized Database Structure

A centralized database can be described as physical entity situated at single location. Also, its operations are undertaken by a single computer only. The centralized databases ease the execution of most of the functions for which databases are created. Apparently, operations such as data update, data back up, query and control access becomes easy if the user is aware of the data storage location and its software.

The size and location of the database does not have any impact on whether the database is centralized, distributed or individual. For instance, the database in a small company can be stored in a personal computer which serves as centralized database, whereas in large companies, database will be stored in multiple computers which are controlled by mainframe.

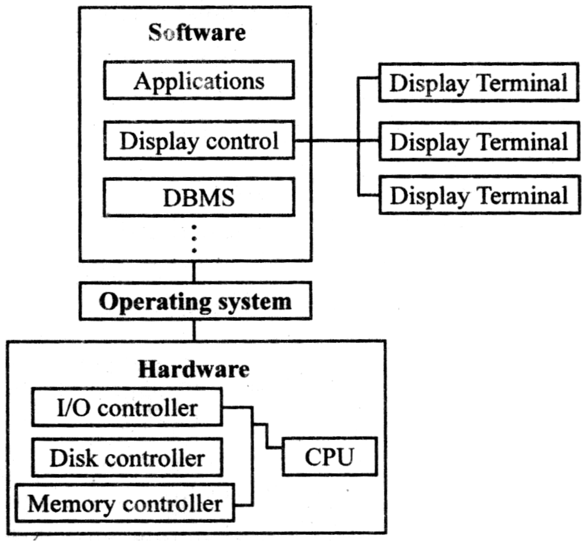


Fig.: Centralized Database Structure

Q13. Describe the overall database system structure with block diagram.

Ans :

Database System Structure

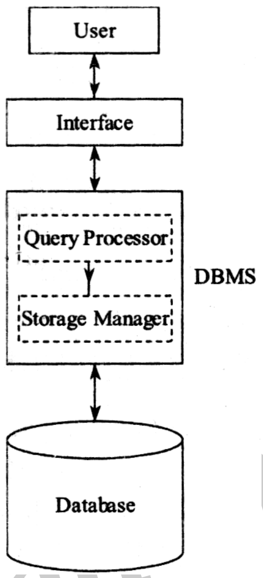


Fig.: Database System Structure

The user interacts with the database management system through an interface. The DBMS does the processing and retrieves the data from the database.

Database system is divided into two modules,

(i) Storage management

(ii) Query processing

Data stored in database accounts is about trillion bytes of data. However, main memory cannot accommodate for such large amount of data. Therefore the data is stored in disks. But for processing, data needs to be transferred from disk to the main memory. This transfer consumes processor time and hence the data needs to be arranged such that the data transfer rate is minimum. This care is taken by storage manager.

The main task of database system is to provide the user with simplified view of data. This is achieved by hiding the physical level implementation details from the user. The user is provided with only high-level view. But for faster processing, the operations need to be done at physical level.

(i) Storage Management

Storage management works as an interface between data stored in database and the application programs.

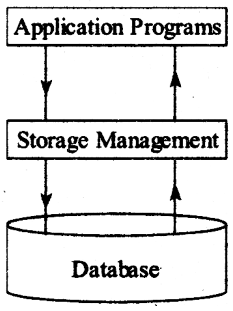


Fig.: Storage Management

Data is stored in disk as file system using operating system conventions. The storage manager provides interaction with file manager and converts the complex DML statements into low level file system commands. Storage management is divided into several components.

(a) Transaction Management

As the name specifies, it manages the transactions such that data remain in consistent state even if the transaction fails. It also provides support for optimal concurrent transaction executions.

(b) File Management

File management allocates space for the data on disk and provides a data structure to define the information saved on disk so as to provide easy access.

(c) Buffer Management

Buffer management handles the transfer of data from disk onto the main memory and decides on what data is to be kept in main memory.

(d) Authorization Management

This module allows only authorized users to access the data.

(e) Integrity Management

This module handles the integrity constraints that resides on the data.

Storage management uses the following data structures.

- Data files
- Data dictionary
- Indices.

(ii) Query Processing

- **Interpreter of Data Definition Language Statements:** DDL statements written by DBA to define the schema are interpreted and stored in the data dictionary.
- **Compiler of Data Manipulation Language Statements:** As any other compiler, DML compiler converts the DML statements in low-level instructions. It also optimizes the query i.e., perform "query optimization"

- **Query Evaluation:** DML compiler converts DML statements into low level instructions which are evaluated by query evaluation machine.

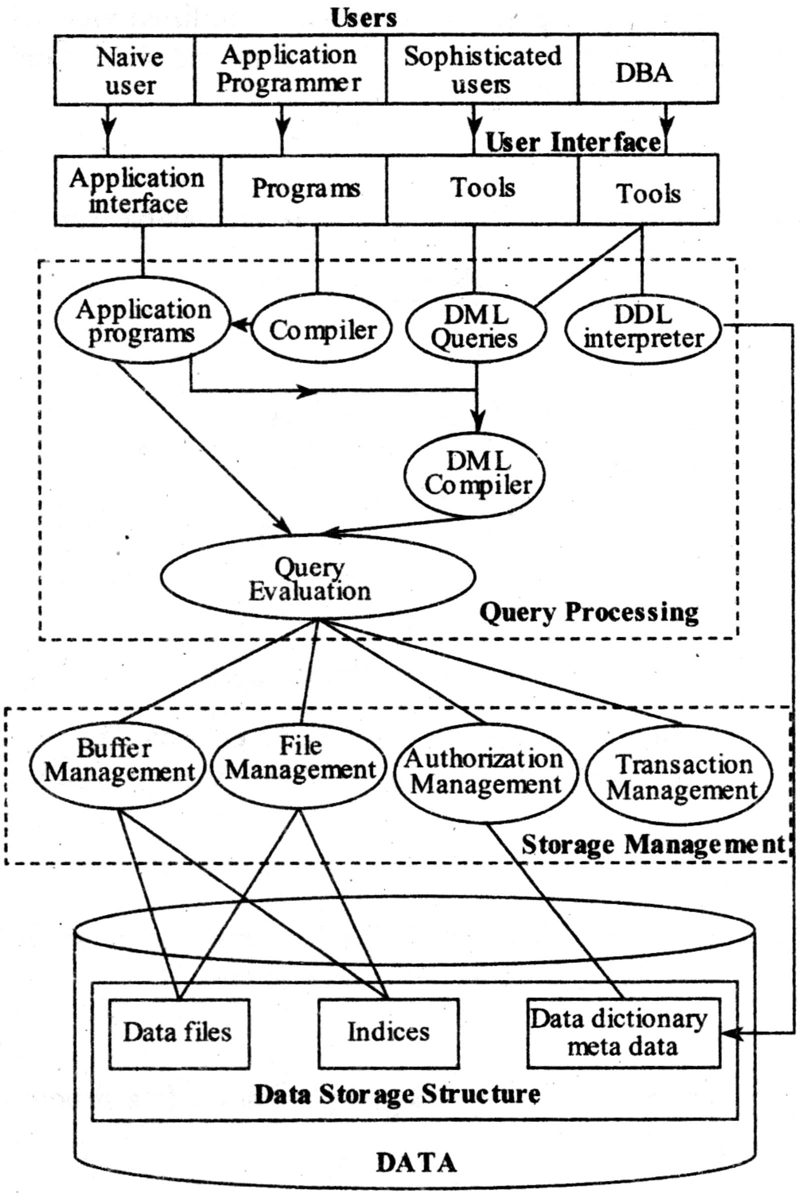


Fig.: Architecture of Database System

1.7 Database Administrator - DBA

1.7.1 Role

Q14. Who is DBA ? What is the role and responsibilities DBA ?

Ans : **(June-19, Imp.)**

Meaning

A Database Administrator is a person or a group of person 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 person. 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.

DBA is responsible for installing the database software. He configure the software of database and then upgrades it if needed. There are many database software like oracle, Microsoft SQL and MySQL in the industry so DBA decides how the installing and configuring of these database software will take place.

1. 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 devise 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 require 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 an modifying sql 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 kind 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.

1.7.2 Functions

Q15. What are the functions of DBA ?

Ans : **(June-19, Imp.)**

1. Schema Definition

- The DBA definition the logical Schema of the database. A Schema refers to the overall logical structure of the database.
- According to this schema, database will be developed to store required data for an organization.

2. Storage Structure and Access Method Definition

- The DBA decides how the data is to be represented in the stored database.

3. Assisting Application Programmers

- The DBA provides assistance to application programmers to develop application programs.

4. Physical Organization Modification

- The DBA modifies the physical organization of the database to reflect the changing needs of the organization or to improve performance.

5. Approving Data Access

- The DBA determines which user needs access to which part of the database.
- According to this, various types of authorizations are granted to different users.

6. Monitoring Performance

- The DBA monitors performance of the system. The DBA ensures that better performance is maintained by making changes in physical or logical schema if required.

7. Backup and Recovery

- Database should not be lost or damaged.
- The DBA ensures this periodically backing up the database on magnetic tapes or remote servers.
- In case of failure, such as virus attack database is recovered from this backup.

1.8 Data Files Indices and Data Dictionary

Q16. Define :

(i) Datafiles

(ii) Indices

Ans :

(i) Datafiles

From user's perspective, the database is viewed as a single data storage component. But, the database is stored in several files called data files. These files are stored on several storage devices using which the DBMS can serve the database requests without waiting for a disk request to complete. This means that using data files, DBMS can respond to the queries in a concurrent manner.

(ii) Indices

The term 'index' is defined as an arrangement that follows a particular order and thus can help in accessing the rows of a table.

An index includes two components, they are: Group of pointers and an index key. Each index key is associated with a set of pointers. The index key is the reference point of an index and it points to those data locations that are referenced by the key.

Q17. What is Data Dictionary? Explain its importance over Database?

Ans :

A metadata (also called the data dictionary) is the data about the data. It is the self describing nature of the database that provides program-data independence. It is also called as the System Catalog. It holds the following information about each data element in the databases, it normally includes:

- Name
- Type
- Range of values
- Source
- Access authorization

+ Indicates which application programs use the data so that, when a change in a data structure is contemplated, a list of the affected programs can be generated.

Data dictionary is used to actually control the database operation, data integrity and accuracy. Metadata is used by developers to develop the programs, queries, controls and procedures to manage and manipulate the data. Metadata is available to database administrators (DBAs), designers and authorized user as on-line system documentation. This improves the control of database administrators (DBAs) over the information system and the user's understanding and use of the system.

Active and Passive Data Dictionaries

Data dictionary may be either active or passive.

An active data dictionary (also called integrated data dictionary) is managed automatically by the database management software. Consistent with the current structure and definition of the database. Most of the relational database management systems contain active data dictionaries that can be derived from their system catalog.

The passive data dictionary (also called non-integrated data dictionary) is the one used only for documentation purposes. Data about fields, files, people and so on, in the data processing environment are. Entered into the dictionary and cross-referenced. Passive dictionary is simply a self-contained application. It is managed by the users of the system and is modified whenever the structure of the database is changed. Since this modification must be performed manually by the user, it is possible that the data dictionary will not be current with the current structure of the database. However, the passive data dictionaries may be maintained as a separate database. Thus, it allows developers to remain independent from using a particular relational database management system. It may be extended to contain information about organizational data that is not computerized.

Importance

Data dictionary is essential in DBMS because of the following reasons:

- Data dictionary provides the name of a data element, its description and data structure in which it may be found.
- Data dictionary provides great assistance in producing a report of where a data element is used in all programs that mention it.
- It is also possible to search for a data name, given keywords that describe the name. For example, one might want to determine the name of a variable that stands for net pay. Entering keywords would produce a list of possible identifiers and their definitions. Using keywords one can search the dictionary to locate the proper identifier to use in a program.

These days, commercial data dictionary packages are available to facilitate entry, editing and to use the data elements.

1.9 Types of Databases

Q18. Explain the different types of databases.

Ans :

A DBMS can support many different types of databases. There exist different database types according to the number of users, the database location(s), and the expected type and extent of use.

Types

1. Depending on Number of Users

The number of users determines whether the database is classified as single-user or multiuser.

(i) Single User Database

A single-user database supports only one user at a time. In other words, if user A is using the database, users B and C must wait until user A is done. A single-user database that runs on a personal computer is called a desktop database.

(ii) Multi User Database

A multiuser database supports multiple users at the same time. When the multiuser database supports a relatively small number of users (usually fewer than 50) or a specific department within an organization, it is called a workgroup database. When the database is used by the entire organization and supports many users (more than 50, usually hundreds) across many departments, the database is known as an enterprise database.

2. Depending on Database location

Location might also be used to classify the database. For example, a database that supports data located at a single site is called a centralized database. A database that supports data distributed across several different sites is called a distributed database.

3. Based on type of use

The most popular way of classifying databases today, however, is based on how they will be used and on the time sensitivity of the information gathered from them. For example, transactions such as product or service sales, payments, and supply purchases reflect critical day-to-day operations. Such transactions must be recorded accurately and immediately. A database that is designed primarily to support a company's day-to-day operations is classified as an operational database (sometimes referred to as a transactional or production database).

In contrast, a data warehouse focuses primarily on storing data used to generate information required to make tactical or strategic decisions. Such decisions typically require extensive "data massaging" (data manipulation) to extract information to formulate pricing decisions, sales forecasts, market positioning, and so on.

4. Based on Structure

Databases can also be classified to reflect the degree to which the data are structured.

(i) Unstructured Data

Unstructured data are data that exist in their original (raw) state, that is, in the format in which they were collected. Therefore, unstructured data exist in a format that does not lend itself to the processing that yields information.

(ii) Structured Data

Structured data are the result of taking unstructured data and formatting (structuring) such data to facilitate storage, use, and the generation of information. You apply structure (format) based on the type of processing that you intend to perform on the data.

1.10 Relational and ER Models

1.10.1 Data Models

Q19. Define data model. State the importance of data models.

Ans :

1. It helps the users in understanding the complexities of real-world environment.
2. It enables communication between designer, application programmers and end users.
3. It provides a clear and improved understanding about the information requirement of the organization to the designers.
4. It helps in organizing data in accordance to end-user's view.
5. It describes data, their relationships, semantics and constraints in order to easily understand a particular problem domain.
6. It helps the users in easily understanding the meaning (semantics) of data and its usage across different application areas.

7. It helps in translating conceptual schema into physical data structures.
8. It helps the users in understanding the nature of data without requiring any physical implementation detail.

Q20. Explain the different types of Data models in DBMS ?

Ans : **(Imp.)**

The following are various data models :

1. Hierarchical Model
2. Network Model
3. Entity-relationship Model
4. Relational Model

1. Hierarchical Model

This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the Root data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like index of a book, recipes etc.

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

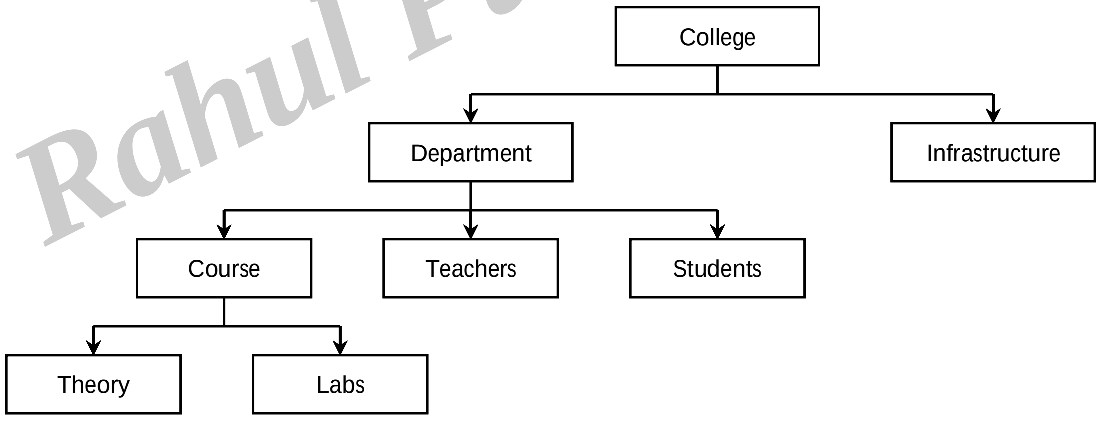


Fig. : Hierarchical Model

Advantages

(i) The design of hierarchical database is simple because the relationship between different levels of a structure is logically simple.

(ii) The model is capable of providing data security and promoting data sharing.

(iii) The model ensures data integrity and conceptual simplicity because of link present between parent and child segment.

(iv) It is efficient when database consists of many 1 :M relationships and transactions.

Disadvantages

(i) It is difficult to implement the model as database designer needs to have complete knowledge about the characteristics of physical data storage.

(ii) It is difficult to maintain the database and applications of hierarchical database because changes in database structure must also be reflected in every application program.

(iii) The model doesn't support the concept of structural independence.

(iv) The model doesn't support multiparent or M:N relationship.

(v) The model doesn't conform to any standard, thereby making it less portable.

2. Network Model

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

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

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

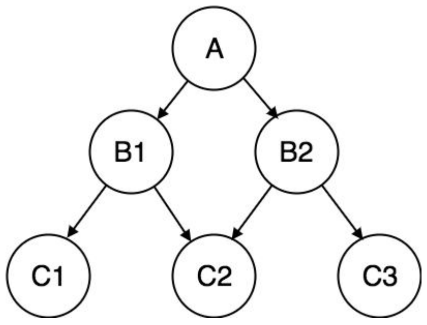


Fig. : Network Model

Advantages

(i) The model is easy to design and is conceptually simple.

(ii) The model can handle both 1:M and M:M relationships.

(iii) The model ensures data integrity by allowing member record to have owner record.

(iv) The model ensures data independence as it isolates the application program from the detail regarding the physical storage.

(v) The model is developed in accordance to the standard, thereby making it model highly portable.

(vi) The model enhances database administration as it includes DDL and DML in DBMS.

Disadvantages

(i) It fails to achieve structural independence due to which it is difficult to maintain database and application programs.

(ii) The model lacks adhoc query capability, thereby making it difficult for database designer in generating simple and complex reports.

(iii) The implementation process is very complex because of navigational data access mechanism.

(iv) The model cannot be used for creating user-friendly DBMS.

3. Entity-relationship Model

In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.

Different entities are related using relationships.

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

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.

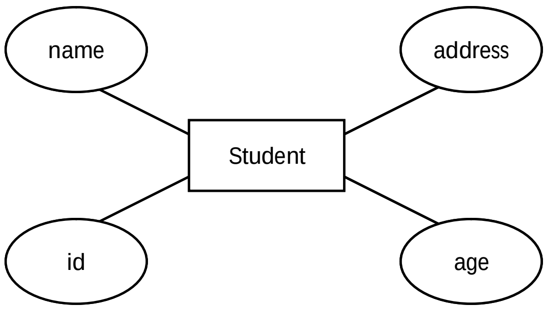


Fig. : Entity - Relationship Model

4. Relational Model

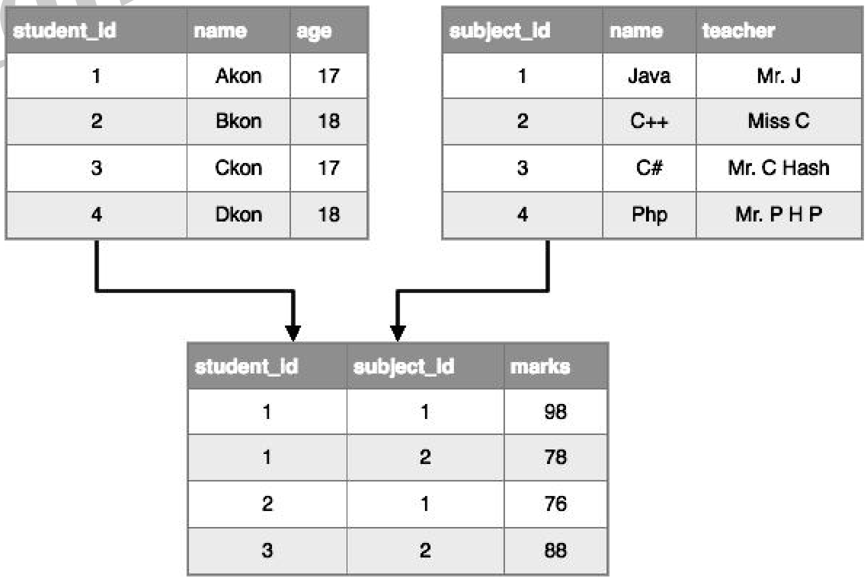
In this model, data is organised in two-dimensional tables and the relationship is maintained by storing a common field.

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.



student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

Fig. : Relational Model

Advantages

(i) The model provides minimum level of redundancy.

(ii) The model promotes data independence and structural independence by using independent tables (i.e change in a table structure is not reflected in other application programs).

(iii) The model is easy to design, manage and implement.

(iv) The model is conceptually simple as database designer need not have to maintain any physical storage details.

(v) The model supports flexible and easy-to-use adhoc query capability, which is based on SQL.

Disadvantages

(i) Relational database system requires huge amount of powerful hardware and system softwares in order to hide the implementation complexities from the users.

(ii) The ease of designing the model may lead to incorrect development and implementation of database management system which in turn results in degrading the performance.

(iii) The model results in islands of information problems (like data incomplete, data duplication, data inconsistency) as every individual develops their own independent applications.

1.10.2 Relational Model (Domain, Tuple, Relation)

Q21. Explain in detail about Relational model concept ?

Ans :

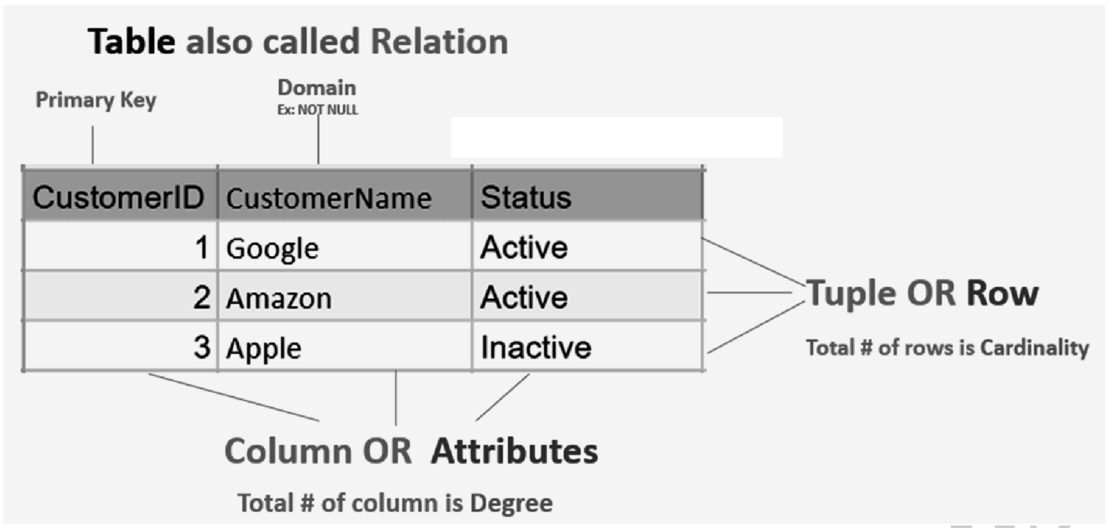
The relational model represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

Relational Model Concepts

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key** - Every row has one, two or multiple attributes, which is called relation key.

10. **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain.



Relational model can represent as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute.

- **Domain:** It contains a set of atomic values that an attribute can take.
- **Attribute:** It contains the name of a column in a particular table. Each attribute Ai must have a domain, dom(Ai)
- **Relational Instance:** In the relational database system, the relational instance is represented by a finite set of tuples. Relation instances do not have duplicate tuples.
- **Relational Schema:** A relational schema contains the name of the relation and name of all columns or attributes.
- **Relational Key:** In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

Example: STUDENT Relation

NAME	ROLL_NO	PHONE_NO	ADDRESS	AGE
Ram	14795	7305758992	Noida	24
Shyam	12839	9026288936	Delhi	35
Laxman	33289	8583287182	Gurugram	20
Mahesh	27857	7086819134	Ghaziabad	27
Ganesh	17282	9028 9i3988	Delhi	40

- In the given table, NAME, ROLL_NO, PHONE_NO, ADDRESS, and AGE are the attributes.
- The instance of schema STUDENT has 5 tuples.
- t3 = <Laxman, 33289, 8583287182, Gurugram, 20>

Properties of Relations

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name
- Attribute domain has no significance
- tuple has no duplicate value
- Order of tuple can have a different sequence

1.10.3 Keys

Q22. What is a Key? Explain types of Relational Keys ?

Ans :

A key is a set of one or more attributes that can uniquely identify each row in a table. It not only identifies the rows of a table but also relates two or more tables.

Types of Keys

1. **Primary Key:** A primary is a column or set of columns in a table that uniquely identifies tuples (rows) in that table.
2. **Super Key:** A super key is a set of one of more columns (attributes) to uniquely identify rows in a table.
3. **Candidate Key:** A super key with no redundant attribute is known as candidate key.
4. **Alternate Key:** Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate or secondary keys.
5. **Composite Key:** A key that consists of more than one attribute to uniquely identify rows (also known as records & tuples) in a table is called composite key.
6. **Foreign Key:** Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

1.10.4 Super Keys - Candidate Keys - Primary Keys and Foreign Key for the Relations

Q23. Explain in detail with examples about Relational keys.

Ans : **(Imp.)**

1. **Candidate Key**

   The minimal set of attribute which can uniquely identify a tuple is known as candidate key.

   For Example, STUD_NO in STUDENT relation.

   - The value of Candidate Key is unique and non-null for every tuple.
   - There can be more than one candidate key in a relation. For Example, STUD_NO as well as STUD_PHONE both are candidate keys for relation STUDENT.
   - The candidate key can be simple (having only one attribute) or composite as well. For Example, {STUD_NO, COURSE_NO} is a composite candidate key for relation STUDENT_COURSE.

Note: In Sql Server a unique constraint that has a nullable column, allows the value 'null' in that column only once. That's why STUD_PHONE attribute as candidate here, but can not be 'null' values in primary key attribute.

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajasthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

2. Super Key

The set of attributes which can uniquely identify a tuple is known as Super Key. For Example, STUD_NO, (STUD_NO, STUD_NAME) etc.

- Adding zero or more attributes to candidate key generates super key.
- A candidate key is a super key but vice versa is not true.

3. Primary Key

There can be more than one candidate key in a relation out of which one can be chosen as primary key.

For Example, STUD_NO as well as STUD_PHONE both are candidate keys for relation STUDENT but STUD_NO can be chosen as primary key (only one out of many candidate keys).

4. Alternate Key

The candidate key other than primary key is called as alternate key. For Example, STUD_NO as well as STUD_PHONE both are candidate keys for relation STUDENT but STUD_PHONE will be alternate key (only one out of many candidate keys).

5. Foreign Key

If an attribute can only take the values which are present as values of some other attribute, it will be foreign key to the attribute to which it refers. The relation which is being referenced is called referenced relation and corresponding attribute is called referenced attribute and the relation which refers to referenced relation is called referencing relation and corresponding attribute is called referencing attribute. Referenced attribute of referencing attribute should be primary key. For

Example, STUD_NO in STUDENT_COURSE is a foreign key to STUD_NO in STUDENT relation.

1.10.5 Relational Constraints - Domain, Key, Integrity

Q24. Explain the different types of Relational constrains ?

Ans :

Relational Integrity Constraints

Relational Integrity constraints is referred to conditions which must be present for a valid relation. These integrity constraints are derived from the rules in the mini-world that the database represents.

There are many types of integrity constraints. Constraints on the Relational database management system is mostly divided into three main categories are:

1. Domain constraints
2. Key constraints
3. Referential integrity constraints

1. Domain Constraints

Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.

Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

Example:

Create DOMAIN CustomerName

CHECK (value not NULL)

The example shown demonstrates creating a domain constraint such that CustomerName is not NULL

2. Key Constraints

An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.

Example:

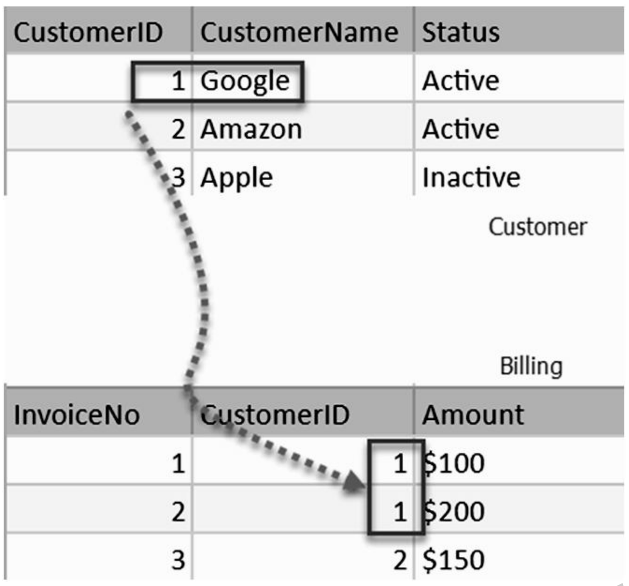
In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

3. Referential Integrity Constraints

Referential integrity constraints is base on the concept of Foreign Keys. A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

Example:



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Customer

Billing

InvoiceNo	CustomerID	Amount
1	1	$100
2	1	$200
3	2	$150

In the above example, we have 2 relations, Customer and Billing.

Tuple for CustomerID =1 is referenced twice in the relation Billing. So we know CustomerName=Google has billing amount $300.

1.10.6 Update Operations and Dealing with Constraint Violations, Relational Operations

Q25. Explain in detail about enforcement of integrity constraint with update operations.

Ans :

Enforcing Integrity Constraints

Integrity constraints are the rules that when applied on relations restrict the insertion of incorrect data. They also help to prevent deletion and updation of consistent data that may lead to loss of data integrity. And, therefore one should be very careful when applying integrity constraints on relations.

The operations such as insertion, deletion and updation must be discarded if they are found to violate integrity constraints.

Example

Consider the Employee relation where "Eid" is the primary key. The following statement,

INSERT INTO Employee(Ename, Eid, Bdate, Address, DNo, Age, Phonenumber)

VALUES('hansen', 12345265, 12-12-1975,' 'New Jersey', 8, 31, 7732132187);

violates the primary key constraints as Eid with 12345265 already exists in the same relations. Since "Eid" is a primary key, its value must be unique for every tuple of relation. Moreover, a primary key cannot contain a null value. Insertion of null values for the field "Eid" again violates the primary key constraint. The following statement shows the same,

INSERT INTO Employee (Ename, Eid, Bdate, Address, DNo, Age, Phone_number)

VALUES ('hensen', null, 12-12-1975, 'New Jersy', 8, 31, 7732132187);

The statement given below,

INSERT INTO Employee (Ename, Eid, Bdate, Address, DNo, Age, Phone number)

VALUES (12345, 12345260, 12-12-1975, 'New Jersy', 8, 31, 7732132187);

causes a violation for domain constraint as the domain for Ename in character but the inserted value is an integer. Their domain constraints is not satisfied again causing a violation. So, while inserting values into the table, one must take care of primary key, domain and unique constraints. When applying any update operation all these violations may again come into picture. But with delete operation - no primary key, domain and unique constraints are violated.

Example for Update Operation

UPDATE Employee E

SET E.Eid= } 2345263

WHERE E.Eid = 12345267;

This statement causes the violation and hence is rejected because the employee with 12345263 already exists. Performing this updation means violating unique constraint for primary key.

In short, it can be said that - these rules are collectively called as entity integrity rules and must be appropriately applied. The second rule called as - referential integrity rule is applied as foreign key. This rule states that either foreign key values must be null or must match the values of the primary key attribute. To understand the concept of referential integrity, consider the same two relations - employee and department where the foreign key - "Managerid" uses Eid of employee as a reference,

The following statement,

INSERT INTO Department(Dname, DNo, Manager id)

VALUES ('headquarters', 9, 12345260);

causes the violation of referential integrity as no such Eid-12345260 exists in employer relation. Thus, insertion in department tables violates the integrity rule. But insertion in employee table does not violate any referential integrity rule. This point is reversed for deletion operations. That is deleting a tuple from department table is simple - it is not a compulsion that the tuple deleted from department must also be deleted from employee relation. Whereas when a tuple is deleted from employee relation, then it must also be deleted from department table otherwise department table will contain the information about the deleted employee leading to data inconsistency. However, the application of update operation which may result in the modification of primary key (Eid) either in department or employee relation may lead to violation of referential integrity.

Q26. Explain different types of relations operations.

Ans :

There are three basic operations to be performed on relations.

1. Insert Operation
2. Delete Operation
3. Update Operation

1. Insert Operation

The Insert operation provides a list of attribute values for a new tuple t that is to be inserted into a relationR. Insert can violate any of the four types of constraints dis-cussed in the previous section. Domain constraints can be violated if an attribute value is given that does not appear in the corresponding domain or is not of the appropriate data type. Key constraints can be violated if a key value in the new tuple t already exists in another tuple in the relation r(R). Entity integrity can be violated if any part of the primary key of the new tuple t is NULL. Referential integrity can be violated if the value of any foreign key in t refers to a tuple that does not exist in the referenced relation

2. Delete Operation

The Delete operation can violate only referential integrity. This occurs if the tuple being deleted is referenced by foreign keys from other tuples in the database. To specify deletion, a condition on the attributes of the relation selects the tuple (or tuples) to be deleted. Here are some examples.

Operation:

Delete the WORKS_ON tuple with Essn = ‘999887777’ and Pno = 10.

Result: This deletion is acceptable and deletes exactly one tuple.

3. Update Operation

The Update (or Modify) operation is used to change the values of one or more attributes in a tuple (or tuples) of some relation R. It is necessary to specify a condition on the attributes of the relation to select the tuple (or tuples) to be modified. Here are some examples.

Operation:

Update the salary of the EMPLOYEE tuple with Ssn = ‘999887777’ to 28000.

Result: Acceptable.

1.11 Relational Operations

Q27. Explain in detail about Relational Operations?

Ans : **(Imp.)**

Relational database systems are expected to be equipped with a query language that can assist its users to query the database instances. There are two kinds of query languages

1. Relational algebra
2. Relational calculus.

1. Relational Algebra

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows -

(i) Select

(ii) Project

(iii) Union

(iv) Set different

(v) Cartesian product

(vi) Rename

We will discuss all these operations in the following sections.

(i) Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation - $\sigma_p(r)$

Where **ó** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like – $=, \neq, \geq, <, >, \leq$.

For example -

$\sigma_{subject = \text{“database”}}$(Books)

Output “ Selects tuples from books where subject is ‘database’.

$\sigma_{\text{subject = “database” and price = “450”}}$ (Books)

Output - Selects tuples from books where subject is ‘database’ and ‘price’ is 450.

$\sigma_{\text{subject = “database” and price = “450” or year > “2010”}}$ (Books)

Output - Selects tuples from books where subject is ‘database’ and ‘price’ is 450 or those books published after 2010.

(ii) Project Operation (Π)

It projects column(s) that satisfy a given predicate.

Notation – $\Pi_{A1, A2, An}$ (r)

Where A_1, A_2, A_n are attribute names of relation r.

Duplicate rows are automatically eliminated, as relation is a set.

For example -

$\Pi_{\text{subject, author}}$ (Books)

Selects and projects columns named as subject and author from the relation Books.

(iii) Union Operation ($\cup$)

It performs binary union between two given relations and is defined as -

$r \cup s = \{ t \mid t \in r \text{ or } t \in s\}$

Notation – r $\cup$ s

Where **r** and **s** are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold -

- r, and s must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

Π_{author} (Books) $\cup$ Π_{author} (Articles)

Output - Projects the names of the authors who have either written a book or an article or both.

(iv) Set Difference (–)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation – **r** – **s**

Finds all the tuples that are present in r but not in s.

Π_{author} (Books) – Π_{author} (Articles)

Output – Provides the name of authors who have written books but not articles.

(v) Cartesian Product (×)

Combines information of two different relations into one.

Notation – r × s

Where **r** and **s** are relations and their output will be defined as "

$r \times s = \{q\,t \mid q \in r \text{ and } t \in s\}$

$\sigma_{author = \text{'tutorialspoint'}}$ (Books × Articles)

Output – Yields a relation, which shows all the books and articles written by tutorialspoint.

(vi) Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter rhoρ.

Notation – ρ_x (E)

Where the result of expression E is saved with name of x.

Additional operations are -

- Set intersection
- Assignment
- Natural join

2. Relational Calculus

In contrast to Relational Algebra, Relational Calculus is a non-procedural query language, that is, it tells what to do but never explains how to do it.

Relational calculus exists in two forms -

Tuple Relational Calculus (TRC)

Filtering variable ranges over tuples

Notation – {T | Condition}

Returns all tuples T that satisfies a condition.

For example –

{ T.name | Author(T) AND T.article = 'database' }

Output – Returns tuples with 'name' from Author who has written article on 'database'.

TRC can be quantified. We can use Existential ($\exists$) and Universal Quantifiers ($\forall$).

For example –

{ R| $\exists$T $\in$ Authors(T.article='database' AND R.name=T.name)}

Output - The above query will yield the same result as the previous one.

Domain Relational Calculus (DRC)

In DRC, the filtering variable uses the domain of attributes instead of entire tuple values (as done in TRC, mentioned above).

Notation

$\{ a_1, a_2, a_3, ..., a_n \mid P (a_1, a_2, a_3, ... ,a_n)\}$

Where a1, a2 are attributes and P stands for formulae built by inner attributes.

For example

{< article, page, subject > | $\in$ TutorialsPoint $\wedge$ subject = 'database'}

Output - Yields Article, Page, and Subject from the relation TutorialsPoint, where subject is database.

Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators.

The expression power of Tuple Relation Calculus and Domain Relation Calculus is equivalent to Relational Algebra.

1.12 Entity Relationship (ER) Model

1.12.1 Entities

Q28. Explain in detail about ER – Model ?

Ans : **(July-21,Oct.-20, Oct.-19,June-18, Imp.)**

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Attributes

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Types of Attributes

- **Simple attribute** - Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
- **Composite attribute** - Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
- **Derived attribute** - Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.
- **Single-value attribute** - Single-value attributes contain single value. For example " Social_Security_Number.
- **Multi-value attribute** - Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

These attribute types can come together in a way like -

- simple single-valued attributes
- simple multi-valued attributes

- composite single-valued attributes
- composite multi-valued attributes

Entity-Set and Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the roll_number of a student makes him/her identifiable among students.

- **Super Key** - A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key** - A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key** - A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Relationship

The association among entities is called a relationship. For example, an employee works_at a department, a student enrolls in a course. Here, Works_at and Enrolls are called relationships.

Relationship Set

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

Degree of Relationship

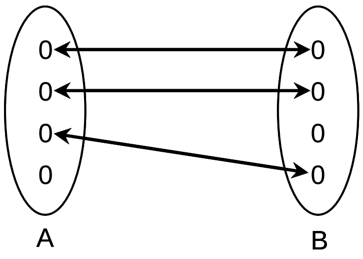
The number of participating entities in a relationship defines the degree of the relationship.

- Binary = degree 2
- Ternary = degree 3
- n-ary = degree

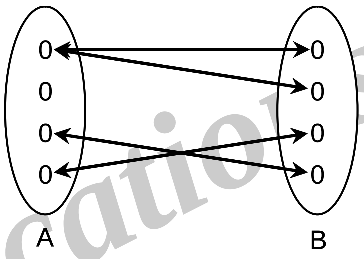
Mapping Cardinalities

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

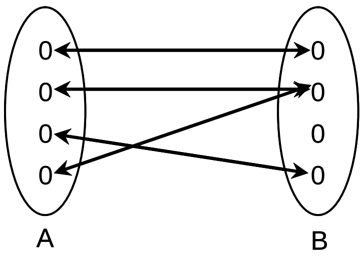
- **One-to-one** - One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



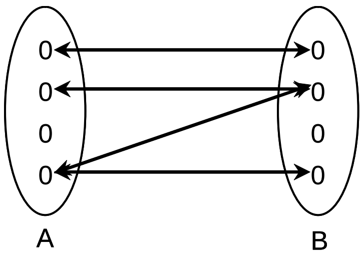
- **One-to-many** - One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



- **Many-to-one** - More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



- **Many-to-many** - One entity from A can be associated with more than one entity from B and vice versa.



1.12.2 Attributes

Q29. Explain about different Types of attributes?

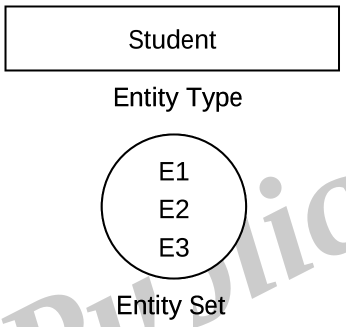
Ans :

ER Model is used to model the logical view of the system from data perspective which consists of these components:

Entity, Entity Type, Entity Set

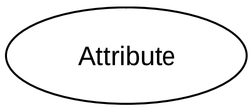
An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

An Entity is an object of Entity Type and set of all entities is called as entity set. e.g.; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



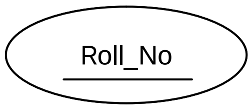
Attribute(s):

Attributes are the properties which define the entity type. For example, Roll_No, Name, DOB, Age, Address, Mobile_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.



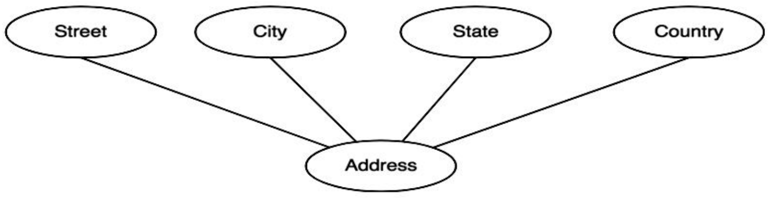
1. **Key Attribute –**

   The attribute which uniquely identifies each entity in the entity set is called key attribute.For example, Roll_No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.



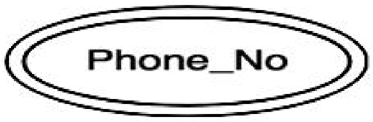
2. **Composite Attribute**

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



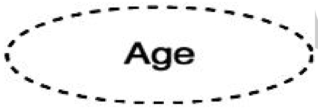
3. Multivalued Attribute

An attribute consisting more than one value for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.

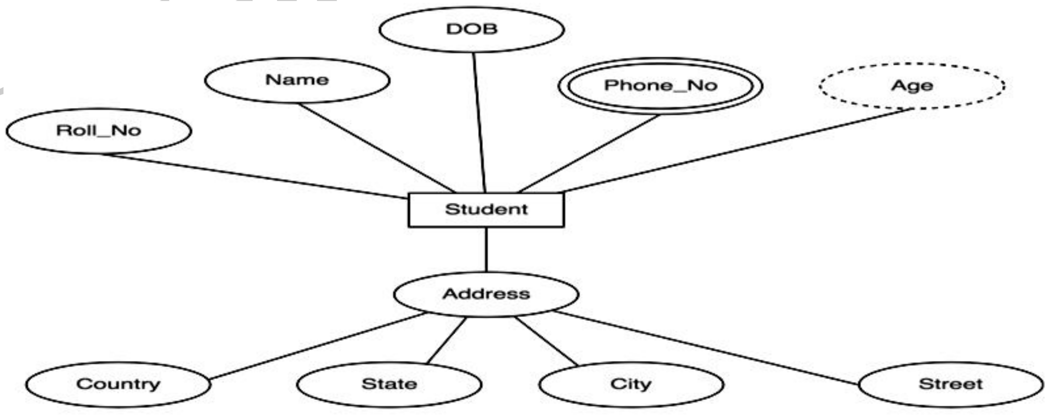


4. Derived Attribute

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



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

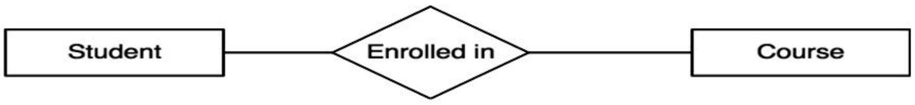


1.12.3 Relationships

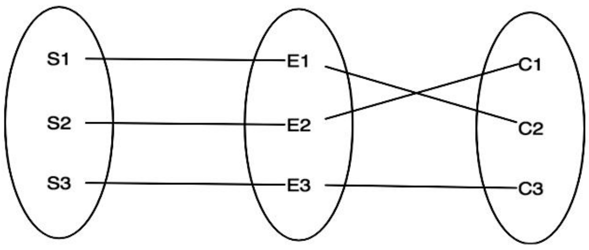
Q30. Explain about different Relationships in ER Model ?

Ans :

A relationship type represents the association between entity types. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course. In ER diagram, relationship type is represented by a diamond and connecting the entities with lines.



A set of relationships of same type is known as relationship set. The following relationship set depicts S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3.

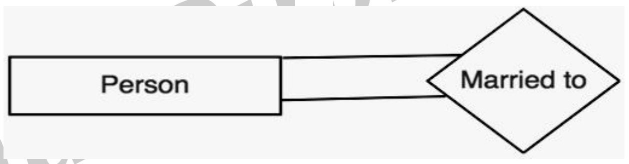


Degree of a relationship set:

The number of different entity sets participating in a relationship set is called as degree of a relationship set.

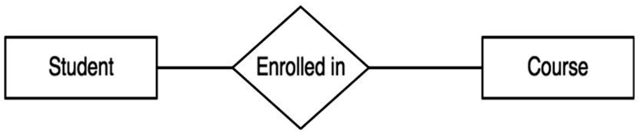
1. Unary Relationship

When there is only ONE entity set participating in a relation, the relationship is called as unary relationship. For example, one person is married to only one person.



2. Binary Relationship

When there are TWO entities set participating in a relation, the relationship is called as binary relationship. For example, Student is enrolled in Course.



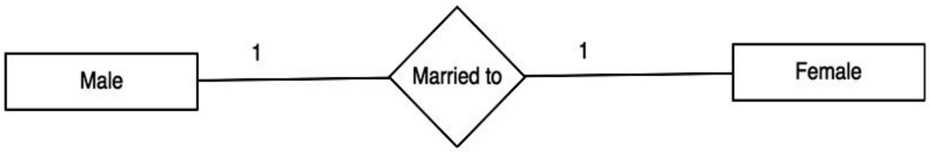
3. n-ary Relationship

When there are n entities set participating in a relation, the relationship is called as n-ary relationship.

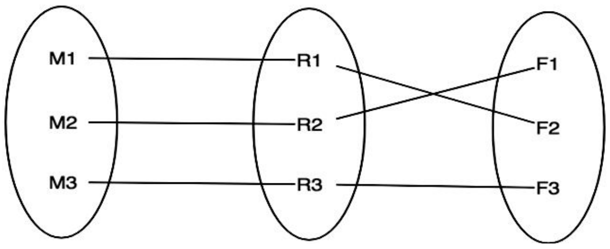
Cardinality:

The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:

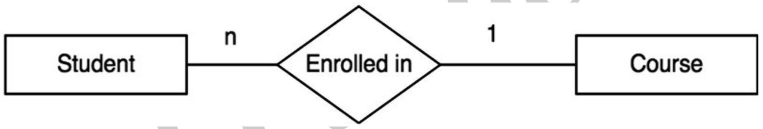
1. **One to one –** When each entity in each entity set can take part **only once in the relationship**, the cardinality is one to one. Let us assume that a male can marry to one female and a female can marry to one male. So the relationship will be one to one.



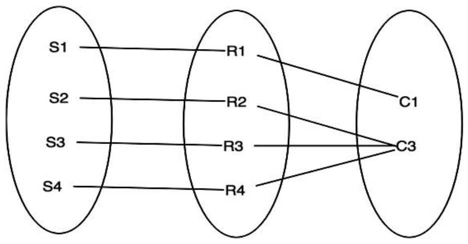
Using Sets, it can be represented as:



2. **Many to one –** When entities in one entity set **can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set,** cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course.

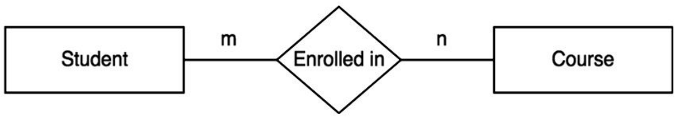


Using Sets, it can be represented as:

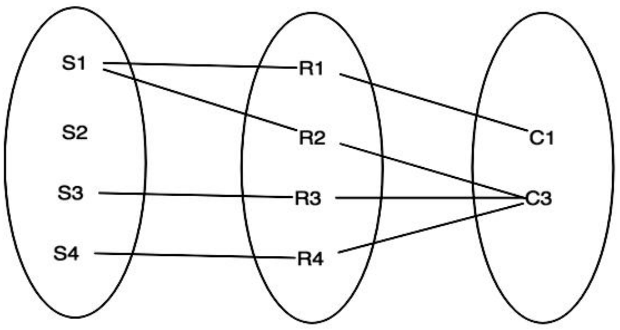


In this case, each student is taking only 1 course but 1 course has been taken by many students.

3. **Many to many –** When entities in all entity sets can **take part more than once in the relationship** cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.



Using sets, it can be represented as:



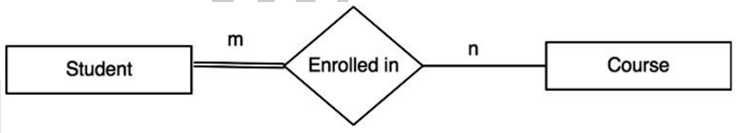
In this example, student S1 is enrolled in C1 and C3 and Course C3 is enrolled by S1, S3 and S4. So it is many to many relationships.

Participation Constraint:

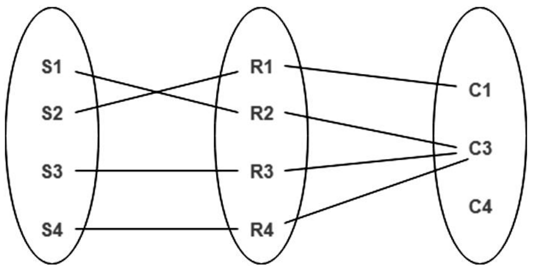
Participation Constraint is applied on the entity participating in the relationship set.

1. **Total Participation –** Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of student will be total. Total participation is shown by double line in ER diagram.
2. **Partial Participation –** The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the student, the participation of course will be partial.

The diagram depicts the 'Enrolled in' relationship set with Student Entity set having total participation and Course Entity set having partial participation.



Using set, it can be represented as,



Every student in Student Entity set is participating in relationship but there exists a course C4 which is not taking part in the relationship.

Q31. Explain in detail about Entity types in ER Model ?

Ans :

Entity:

An Entity is an object of interest to the end user. An entity actually refers to the entity set and not to a single entity occurrence. In other words, the word entity in the ERM corresponds to a table—not to a row—in the relational environment. The ERM refers to a table row as an entity instance or entity occurrence. In both the Chen and Crow's Foot notations, an entity is represented by a rectangle containing the entity's name. The entity name, a noun, is usually written in all capital letters. Different types of Entities are as follows.

Strong Entities:

If an entity can exist apart from all of its related entities (it is existence-independent), then that entity is referred to as a strong entity or regular entity. For example, suppose that the XYZ Corporation uses parts to produce its products. Furthermore, suppose that some of those parts are produced in-house and other parts are bought from vendors. In that scenario, it is quite possible for a PART to exist independently from a VENDOR in the relationship "PART is supplied by VENDOR," because at least some of the parts are not supplied by a vendor. Therefore, PART is existence independent from VENDOR.

Weak Entities:

A weak entity is one that meets two conditions:

1. The entity is existence-dependent; that is, it cannot exist without the entity with which it has a relationship.
2. The entity has a primary key that is partially or totally derived from the parent entity in the relationship.

For example, a company insurance policy insures an employee and his/her dependents. For the purpose of describing an insurance policy, an EMPLOYEE might or might not have a DEPENDENT, but the DEPENDENT must be associated with an EMPLOYEE. Moreover, the DEPENDENT cannot exist without the EMPLOYEE; that is, a person cannot get insurance coverage as a dependent unless s(he) happens to be a dependent of an employee. DEPENDENT is the weak entity in the relationship "EMPLOYEE has DEPENDENT."

Basis for Comparison	Strong Entity	Weak Entity
Basic	The Strong entity has a primary key.	The weak entity has a partial discriminator key.
Depends	The Strong entity is independent of any other entity in a schema.	Weak entity depends on the strong entity for its existence.
Denoted	Strong entity is denoted by a single rectangle.	Weak entity is denoted with the double rectangle.
Relation	The relation between two strong entities is denoted by a single diamond simply called relationship.	The relationship between a weak and a strong entity is denoted by Identifying Relationship denoted with double diamond.
Participation	Strong entity may or may not have total participation in the relationship.	Weak entity always has total participation in the identifying relationship shown by double line.

1.13 More about Entities and Relationships

Q32. Explain the features of ER-Model ?

Ans :

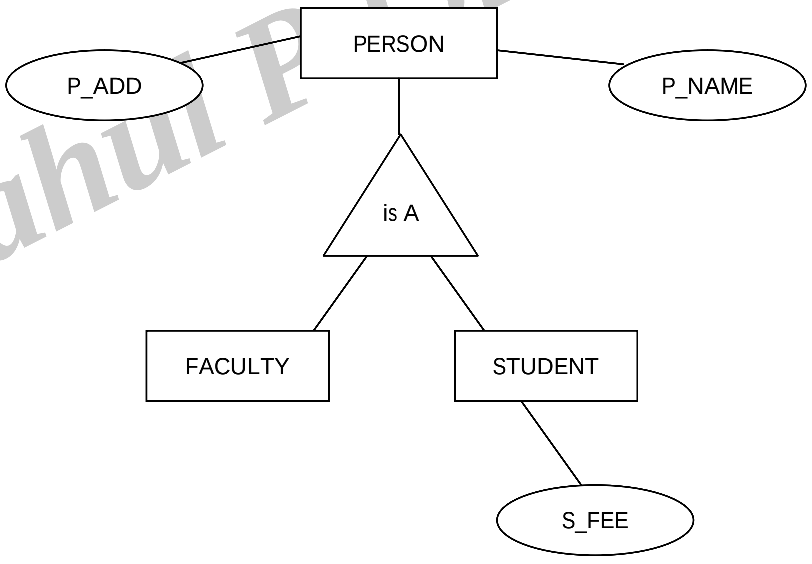
The basic features of ER - Diagrams are sufficient to design many database situations. However, with more complex relations and advanced database applications, it is required to move to enhanced features of ER- model.

The three such features are.

1. Genralization
2. Specialization
3. Aggregation

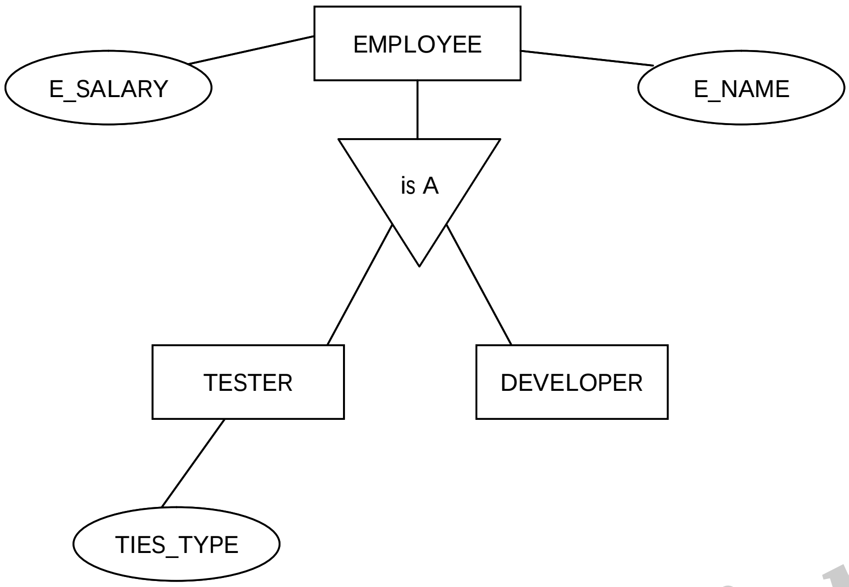
1. Generalization

Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it. It is a bottom-up approach in which two or more entities can be generalized to a higher level entity if they have some attributes in common. For Example, STUDENT and FACULTY can be generalized to a higher level entity called PERSON as shown in Figure 1. In this case, common attributes like P_NAME, P_ADD become part of higher entity (PERSON) and specialized attributes like S_FEE become part of specialized entity (STUDENT).



2. Specialization

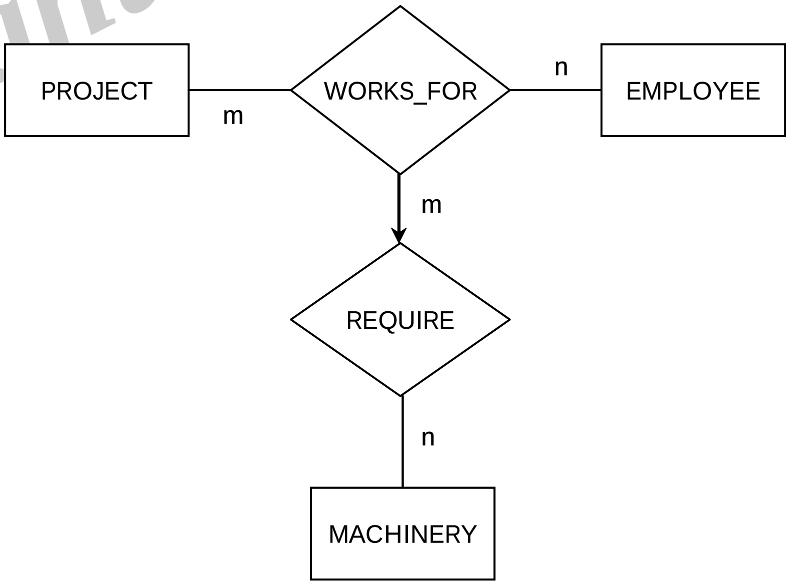
In specialization, an entity is divided into sub-entities based on their characteristics. It is a top-down approach where higher level entity is specialized into two or more lower level entities. For Example, EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc. as shown in Figure 2. In this case, common attributes like E_NAME, E_SAL etc. become part of higher entity (EMPLOYEE) and specialized attributes like TES_TYPE become part of specialized entity (TESTER).



Specialization

3. Aggregation

An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher level entity. For Example, Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS_FOR and entity MACHINERY. Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.



Aggregation

Q33. List out the differences between Generalization and Specialization ?

Ans :

Key Differences Between Generalization and Specialization in DBMS

1. The fundamental difference between generalization and specialization is that Generalization is a bottom-up approach. However, specialization is a top-down approach.
2. Generalization club all the entities that share some common properties to form a new entity. On the other hands, specialization spilt an entity to form multiple new entities that inherit some properties of the spiltted entity.
3. In generalization, a higher entity must have some lower entities whereas, in specialization, a higher entity may not have any lower entity present.
4. Generalization helps in reducing the size of schema whereas, specialization is just opposite it increases the number of entities thereby increasing the size of a schema.
5. Generalization is always applied to the group of entities whereas, specialization is always applied on a single entity.
6. Generalization results in a formation of a single entity whereas, Specialization results in the formation of multiple new entities.

1.14 Defining Relationship for College Database

Q34. Explain an ER Diagram for College Database ?

Ans :

A college database maintains all the information related to the students such as their admission details, fee structure, performance etc. It also maintains information related to the faculty and course offerings. This database typically carries information related to students and faculty only. For both or these fields, there exist certain common attributes such as Name, Sex, Address, Phone numbers etc.

The following relationships can be defined between various attributes in a college database.

- A student can belong to not more than a single attribute of Course field therefore the relationship is one-to-one among Student and Course.
- A faculty can belong to more than one department and therefore a one-to-many relationship is defined among Faculty and Department.
- With respect to departments, the faculty as well as students can be mapped. Therefore, a many-to-many relationship is defined between Department, Faculty and Students.

  For most of the entities like Name, Address, Faculty, Course etc., there exist some sub entities which are as follows,
- For ‘Name’ entity, the sub-entities could be First Name, Middle Name and Last_Name.
- For ‘Address’ entity, the sub-entities could be Door_No, City and State.
- For ‘Faculty’ entity, the sub-entities could be Department, Salary, Contact No, etc.
- For ‘Course’ entity, the sub-entities could be Course Name, Fee, No of Students etc.

For all these entities and sub-entities, Is-A relationship or class/sub-class relationship is defined.

Consider an example ER diagram that captures the full information about the university. Use only the basic ER model here, that is entities, relationships and attributes. Be sure to indicate any key and participation constraints with underlines and arrows.

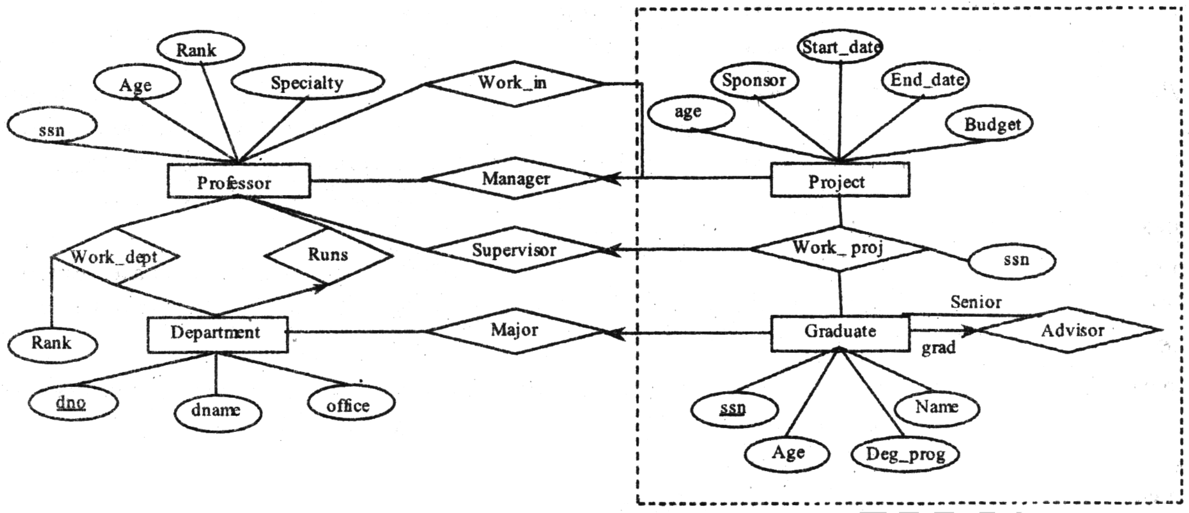


Fig.: University ER Diagram

College Database ER Diagram

An ER diagram for College database is constructed with the following information.

(a) Professors have an SSN, name, age, rank and research speciality.

(b) Projects have a project number, a sponsor name, a starting date, an ending date and a budget.

(c) Graduate students have an SSN, a name, age and a degree program (Example : M.S or Ph.D.)

(d) Each project is managed by one professor.

(e) Each project is worked by one or more professors.

(f) Professors can manage and/or work on multiple projects.

(g) Each project is worked by one or more graduate students (known as the projects research assistants).

(h) When graduate students work on a project, a professor must supervise their work on the project. Graduate students can work on multiple projects, in this case they can have a different supervisor for each one.

(i) Departments have a department number, department name and a main office.

(j) Departments have a professor who runs the department.

(k) Professors work in one or more departments, and for each department that they work in a time percentage is associated with their job.

(*l*) Graduate students have a major department in which they are working on their degree.

(m) Each graduate student has another, more senior graduate student (known as student advisor) who advises him or her on what courses to take.

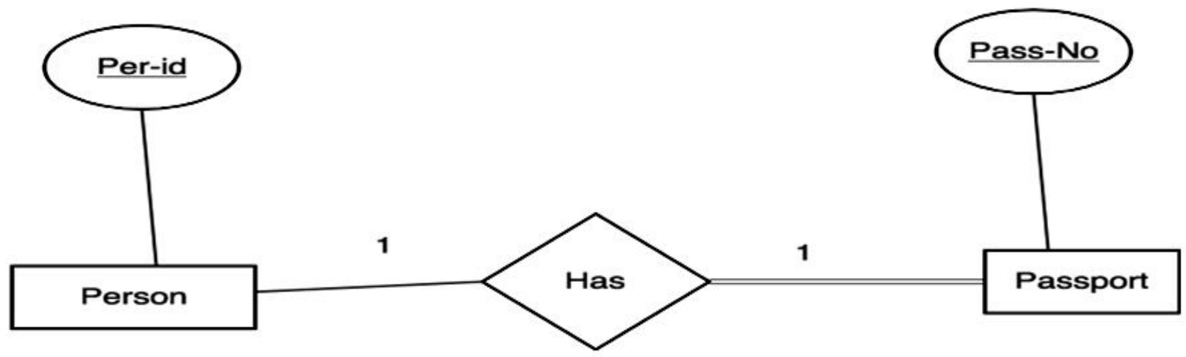
1.15 Conversion of ER Diagram to Relational Database

Q35. How can we convert ER Diagram into Relational model ?

Ans : **(Imp.)**

After designing the ER diagram of system, we need to convert it to Relational models which can directly be implemented by any RDBMS like Oracle, MySQL etc. In this article we will discuss how to convert ER diagram to Relational Model for different scenarios.

Case 1 : Binary Relationship with 1:1 cardinality with total participation of an entity



A person has 0 or 1 passport number and Passport is always owned by 1 person. So it is 1:1 cardinality with full participation constraint from Passport.

First Convert each entity and relationship to tables. Person table corresponds to Person Entity with key as Per-Id. Similarly Passport table corresponds to Passport Entity with key as Pass-No. Has Table represents relationship between Person and Passport (Which person has which passport). So it will take attribute Per-Id from Person and Pass-No from Passport.

Person	
Per-Id	**Other Person Attribute**
PR1	–
PR2	–
PR3	–

Has	
Per-Id	**Pass-No**
PR1	PS1
PR2	PS2

Passport	
Pass-No	**Other PassportAttribute**
PS1	–
PS2	–

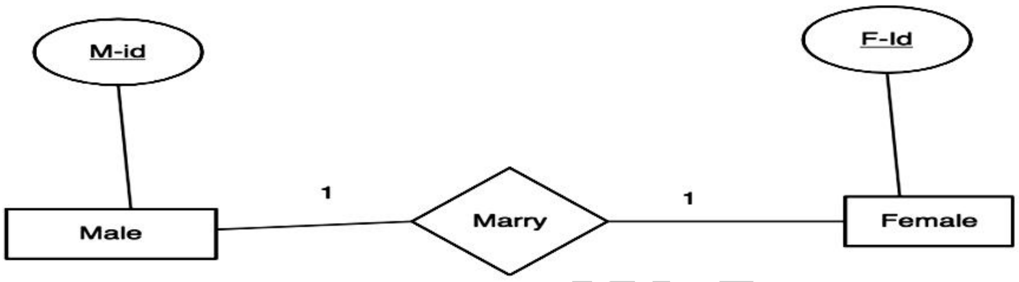
Table 1

As we can see from Table 1, each Per-Id and Pass-No has only one entry in Has table. So we can merge all three tables into 1 with attributes shown in Table 2. Each Per-Id will be unique and not null. So it will be the key. Pass-No can't be key because for some person, it can be NULL.

Per-Id	Other Person Attribute	Pass-No	Other PassportAttribute
PR_1	–	PS_1	–
PR_2	–	PS_2	–
PR_3	–	–	–

Table 2

Case 2 : Binary Relationship with 1:1 cardinality and partial participation of both entities



A male marries 0 or 1 female and vice versa as well. So it is 1:1 cardinality with partial participation constraint from both. First Convert each entity and relationship to tables. Male table corresponds to Male Entity with key as M-Id. Similarly Female table corresponds to Female Entity with key as F-Id. Marry Table represents relationship between Male and Female (Which Male marries which female). So it will take attribute M-Id from Male and F-Id from Female.

Male		**Marry**		**Female**	
M-Id	Other Male Attribute	M-Id	F-Id	F-Id	Other FemaleAttribute
M1	–	M1	F2	F1	–
M2	–	M2	F1	F2	–
M3	–			F3	–

Table 3

As we can see from Table 3, some males and some females do not marry. If we merge 3 tables into 1, for some M-Id, F-Id will be NULL. So there is no attribute which is always not NULL. So we can't merge all three tables into 1. We can convert into 2 tables. In table 4, M-Id who are married will have F-Id associated. For others, it will be NULL. Table 5 will have information of all females. Primary Keys have been underlined.

M-Id	Other Male Attribute	F-Id

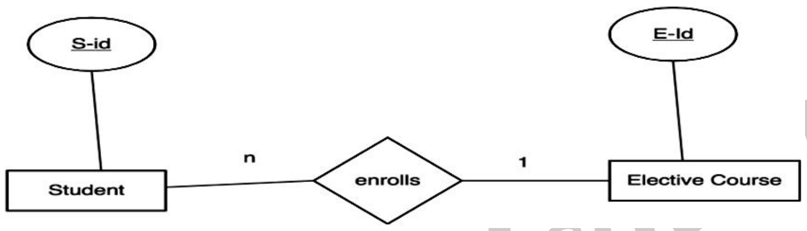
Table 4

F-Id	Other FemaleAttribute

Table 5

Note: Binary relationship with 1:1 cardinality will have 2 table if partial participation of both entities in the relationship. If atleast 1 entity has total participation, number of tables required will be 1.

Case 3: Binary Relationship with n: 1 cardinality



In this scenario, every student can enroll only in one elective course but for an elective course there can be more than one student. First Convert each entity and relationship to tables. Student table corresponds to Student Entity with key as S-Id. Similarly Elective_Course table corresponds to Elective_Course Entity with key as E-Id. Enrolls Table represents relationship between Student and Elective_Course (Which student enrolls in which course). So it will take attribute S-Id from and Student E-Id from Elective_Course.

Student

S-Id	Other Student Attribute
S1	–
S2	–
S3	–
S4	–

Enrolls

S-Id	E-Id
S1	E1
S2	E2
S3	E1
S4	E1

Elective_Course

E-Id	Other Elective CourseAttribute
E1	–
E2	–
E3	–

Table 6

As we can see from Table 6, S-Id is not repeating in Enrolls Table. So it can be considered as a key of Enrolls table. Both Student and Enrolls Table's key is same; we can merge it as a single table. The resultant tables are shown in Table 7 and Table 8. Primary Keys have been underlined.

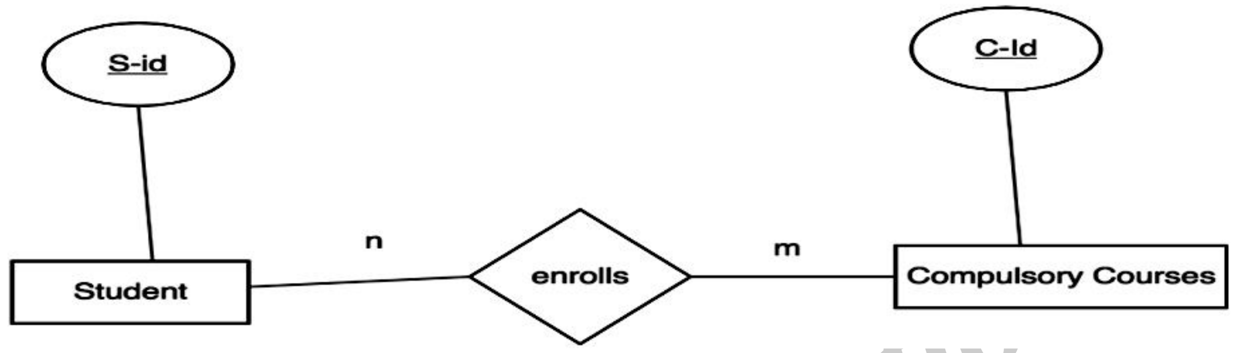
S-Id	Other Student Attribute	E-Id
S_1	–	E_1
S_2	–	E_2
S_3	–	E_3

Table 7

E-Id	Other Elective
E_1	–
E_2	–
E_3	–

Table 8

Case 4: Binary Relationship with m: n cardinality



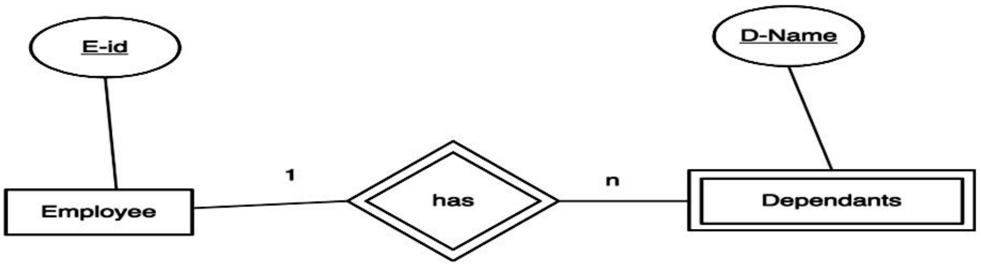
In this scenario, every student can enroll in more than 1 compulsory course and for a compulsory course there can be more than 1 student. First Convert each entity and relationship to tables. Student table corresponds to Student Entity with key as S-Id. Similarly Compulsory_Courses table corresponds to Compulsory Courses Entity with key as C-Id. Enrolls Table represents relationship between Student and Compulsory_Courses (Which student enrolls in which course). So it will take attribute S-Id from Person and C-Id from Compulsory_Courses.

Student		Enrolls		Compulsory_Courses	
S-Id	Other Student Attribute	S-Id	C-Id	C-Id	Other Compulsory CourseAttribute
S1	–	S1	C1	C1	–
S2	–	S1	C2	C2	–
S3	–	S3	C1	C3	–
S4	–	S4	C3	C4	–
		S4	C2		
		S3	C3		

Table 9

As we can see from Table 9, S-Id and C-Id both are repeating in Enrolls Table. But its combination is unique; so it can be considered as a key of Enrolls table. All tables' keys are different, these can't be merged. Primary Keys of all tables have been underlined.

Case 5: Binary Relationship with weak entity



In this scenario, an employee can have many dependants and one dependant can depend on one employee. A dependant does not have any existence without an employee (e.g; you as a child can be dependant of your father in his company). So it will be a weak entity and its participation will always be total. Weak Entity does not have key of its own. So its key will be combination of key of its identifying entity (E-Id of Employee in this case) and its partial key (D-Name).

First Convert each entity and relationship to tables. Employee table corresponds to Employee Entity with key as E-Id. Similarly Dependants table corresponds to Dependant Entity with key as D-Name and E-Id. Has Table represents relationship between Employee and Dependants (Which employee has which dependants). So it will take attribute E-Id from Employee and D-Name from Dependants.

Employee		**Has**		**Dependants**		
E-Id	Other Employee Attribute	E-Id	D-Name	D-Name	E-Id	Other DependantsAttribute
E1	–	E1	RAM	RAM	E1	–
E2	–	E1	SRINI	SRINI	E1	–
E3	–	E2	RAM	RAM	E2	–
		E3	ASHISH	ASHISH	E3	–

Table 10

As we can see from Table 10, E-Id, D-Name is key for **Has** as well as Dependants Table. So we can merge these two into 1. So the resultant tables are shown in Tables 11 and 12. Primary Keys of all tables have been underlined.

E-id	Other Employee Attribute
E_1	–
E_2	–
E_3	–

Table 11

D-Name	E-id	Other DependantsAttribute
RAM	E_1	–
SRINI	E_1	–
RAM	E_2	–
ASHISH	E_3	–

Table 12

Short Question and Answers

1. Database

Ans :

A database is a collection of information that is organized so that it can be easily accessed, managed and updated. Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information.

A database is a collection of information that is organized so that it can be easily accessed, managed and updated.

Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it.

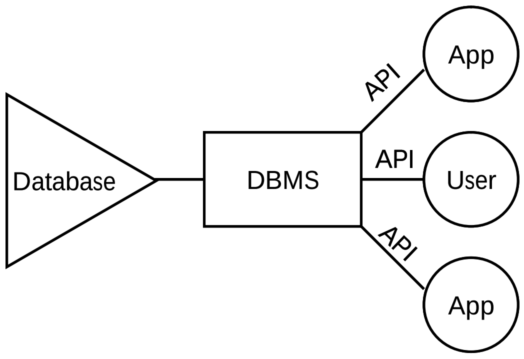
2. Define DBMS.

Ans :

Database Management System (DBMS)

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.

A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible.



3. Differentiate DBMS with File based system.

Ans :

There are following differences between DBMS and File system:

DBMS	File System
DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	File system is a collection of data. In this system, the user has to write the procedures for managing the database.
DBMS gives an abstract view of data that hides the details.	File system provides the detail of the data representation and storage of data.
DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure.	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 lost.
DBMS provides a good protection mechanism.	It is very difficult to protect a file under the file system.
DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	File system can't efficiently store and retrieve the data.
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 other deleting some information or updating some information.

4. Define Data Dictionary.

Ans :

A metadata (also called the data dictionary) is the data about the data. It is the self describing nature of the database that provides program-data independence. It is also called as the System Catalog. It holds the following informationabout each data element in the databases, it normally includes:

- Name
- Type
- Range of values
- Source
- Access authorization

+ Indicates which application programs use the data so that, when a change in a data structure is contemplated, a list of the affected programs can be generated.

Data dictionary is used to actually control the database operation, data integrity and accuracy. Metadata is used by developers to develop the programs, queries, controls and procedures to manage and manipulate the data. Metadata is available to database administrators (DBAs), designers and authorized user as on-line system documentation. This improves the control of database administrators (DBAs) over the information system and the user's understanding and use of the system.

5. Define MVA.

Ans :

Multi-value attribute - Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

These attribute types can come together in a way like -

- simple single-valued attributes
- simple multi-valued attributes
- composite single-valued attributes
- composite multi-valued attributes

6. What are the advantages of DBMS ?

Ans :

Advantages of DBMS

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

7. List the functions of File Management System.

Ans :

- The computer's file system is a method of storing and organizing computer data. It makes it easy to access and find it.
- Its main functions are the allocation of space to the files and the administration of free space and access to protected data.
- They structure the information stored in a data storage device or storage unit which will then be represented either textually or graphically using a file manager. It divides the space in the device into blocks of a specific size.
- The data is stored in these blocks and the size is modified to occupy an integer number of blocks. The file system software is responsible for organizing these blocks into files and directories and recording which blocks are assigned to which files and which blocks are not used.

- In the concept, the user uses the file system to save data without having to care about the data blocks actually stored in the address of the hard disk (or optical disk), and only needs to remember the directory and file name of the file.

8. Who is DBA ? What is his Role ?

Ans :

A Database Administrator is a person or a group of person 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 person. 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.

DBA is responsible for installing the database software. He configure the software of database and then upgrades it if needed. There are many database software like oracle, Microsoft SQL and MySQL in the industry so DBA decides how the installing and configuring of these database software will take place.

9. Define Derive attribute.

Ans :

Derived attribute - Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.

10. Differentiate Strong & Weak Entities.

Ans :

Basis for Comparison	Strong Entity	Weak Entity
Basic	The Strong entity has a primary key.	The weak entity has a partial discriminator key.
Depends	The Strong entity is independent of any other entity in a schema.	Weak entity depends on the strong entity for its existence.
Denoted	Strong entity is denoted by a single rectangle.	Weak entity is denoted with the double rectangle.
Relation	The relation between two strong entities is denoted by a single diamond simply called relationship.	The relationship between a weak and a strong entity is denoted by Identifying Relationship denoted with double diamond.
Participation	Strong entity may or may not have total participation in the relationship.	Weak entity always has total participation in the identifying relationship shown by double line.

11. List out the differences between Generalization and Specialization ?

Ans :

Key Differences Between Generalization and Specialization in DBMS

1. The fundamental difference between generalization and specialization is that Generalization is a bottom-up approach. However, specialization is a top-down approach.

2. Generalization club all the entities that share some common properties to form a new entity. On the other hands, specialization spilt an entity to form multiple new entities that inherit some properties of the spiltted entity.
3. In generalization, a higher entity must have some lower entities whereas, in specialization, a higher entity may not have any lower entity present.
4. Generalization helps in reducing the size of schema whereas, specialization is just opposite it increases the number of entities thereby increasing the size of a schema.
5. Generalization is always applied to the group of entities whereas, specialization is always applied on a single entity.
6. Generalization results in a formation of a single entity whereas, Specialization results in the formation of multiple new entities.

12. Relational model

Ans :

The relational model represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

Relational Model Concepts

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.

Choose the Correct Answer

1. Logical DBMS architecture contains ________ level [d]
   (a) Physical (b) Conceptual
   (c) External (d) All

2. ________ is introduced to overcome the limitations of file based system [a]
   (a) DataBase approach (b) Objective model
   (c) Indexes (d) ER-Model

3. Hierarchical and Network Data Models were developed for complex data in the year of ________ [a]
   (a) 1960 (b) 1970
   (c) 1980 (d) 1990

4. Following is a problem faced by File-Based system [d]
   (a) Data redundancy (b) Data isolation
   (c) Security & Integrity (d) All

5. A person who has total control over the system is called [c]
   (a) Data Manager (b) DataBase User
   (c) DataBase Administrator (d) Programmer

6. Row in a table is also termed as ________ . [a]
   (a) Tuple (b) Attribute
   (c) Domain (d) Relation

7. ________ is an attribute, used to identify data in Relation. [b]
   (a) Column (b) Key
   (c) Row (d) Relation

8. A key that contain more than one attribute is called ________ . [c]
   (a) Candidate key (b) Forign key
   (c) Composite key (d) Primary key

9. A Relational model contains ________ constraints [d]
   (a) Domain (b) Key
   (c) Integrity (d) All

10. An Entity always depends on another entity is termed as [b]
    (a) Strong Entity (b) Weak Entity
    (c) Simple Entity (d) All

Fill in the blanks

1. ________ is the collection of suumarized (or) organized data.
2. In logical DBMS architecture ________ level is closest to the user.
3. The name of each column in a table is called an ________
4. ________ key is select as the principal unique indentifier.
5. ________ views the real world as a set of basic objects and Relationships among these objects.
6. Multivalued attribute's are represented by ________ symbol.
7. ________ refers to the no.of. associated entities.
8. Specialization is a ________ approach.
9. If an attribute can be split into components it is called a ________ .
10. An attribute which is calculated from other attribute is called ________ .

Answers

1. Information
2. External
3. Attribute
4. Primary
5. Entity & Relationship Model
6. Double ellipse
7. Relationship Degree
8. Top-Down
9. Composite attribute
10. Derived attribute

One Mark Answers

1. DBMS

Ans :

DataBase Management System is a collection of inter related data and set of programs to access those data. The goal of DBMS is to fecilitate an interface which is suitable for the users to access and store data.

2. DBA

Ans :

One of the main reason for using DBMS is to have a central control of both data and the programs accessed those data. A person who has such control over the system is called a DataBase administrator (DBA).

3. Define Foreign key.

Ans :

It is an attribute that appears as a non-key attribute in one relation and as a primary key attribute in another relation.

4. ER-Model

Ans :

Entity - Relationship model views the real world as a set of basic objects and Relationships among these objects. This Represents the overall logical structure of the DataBase.

ER-Model is a pictorial Representation of ER-Diagrams including Entities, Attributes & Relationships.

5. Aggrigation

Ans :

Aggrigation is an abstraction through which Relationships are treated as higher level Entities.

6. Define Database.

Ans :

A database is an integrated, shared collection of data. It consists of both ‘end-user-data’, which includes raw facts of interest and ‘metadata’, which includes information about data using which end-user data are managed and integrated. it is usually so large that it has to be stored on the secondary storage devices like disks or tapes. Such data can be maintained as a collection of operating system files and stored in a database structure called Database Management System (DBMS).

7. Explain primary key.

Ans :

It is an attribute or set of attributes that uniquely identify an entity (row) in the entity set (table). The main difference between the primary key and the candidate key is that primary key doesn't contain null values. Always the attributes that can never or rarely changed are chosen as primary keys. For example the EMP_ID attribute can be selected as a primary key in the above example.

8. Define derived attribute.

Ans :

An attribute whose value is obtained from the value of other related attributes of an entity is called 'derived attribute'. The values of derived attributes can be obtained using calculations, procedures or algorithms.

UNIT II

DATABASE INTEGRITY AND NORMALISATION

Relational Database Integrity - TheKeys - 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 - Losslessjoin 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

2.1 Relational Database Integrity

Q1. Define Database integrity ?

(OR)

Explain briefly about database integrity.

Ans : **(Imp.)**

Meaning

Data integrity is the overall completeness, accuracy and consistency of data. This can be indicated by the absence of alteration between two instances or between two updates of a data record, meaning data is intact and unchanged. Data integrity is usually imposed during the database design phase through the use of standard procedures and rules. Data integrity can be maintained through the use of various error-checking methods and validation procedures.

Data integrity is enforced in both hierarchical and relational database models.

The following three integrity constraints are used in a relational database structure to achieve data integrity:

1. **Entity Integrity**

   This is concerned with the concept of primary keys. The rule states that every table must have its own primary key and that each has to be unique and not null.

2. **Referential Integrity**

   This is the concept of foreign keys. The rule states that the foreign key value can be in two states. The first state is that the foreign key value would refer to a primary key value of another table, or it can be null. Being null could simply mean that there are no relationships, or that the relationship is unknown.

3. **Domain Integrity**

   This states that all columns in a relational database are in a defined domain.

The concept of data integrity ensures that all data in a database can be traced and connected to other data. This ensures that everything is recoverable and searchable. Having a single, well-defined and well-controlled data integrity system increases stability, performance, reusability and maintainability. If one of these features cannot be implemented in the database, it must be implemented through the software.

2.2 The Keys

Q2. What is the use of key on tables? Explain the types of keys available in RDBMS ?

Ans :

Keys are very important part of Relational database model. They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table.

A Key can be a single attribute or a group of attributes, where the combination may act as a key.

Need

In real world applications, number of tables required for storing the data is huge, and the different tables are related to each other as well.

Also, tables store a lot of data in them. Tables generally extends to thousands of records stored in them, unsorted and un organized.

Now to fetch any particular record from such dataset, you will have to apply some conditions, but what if there is duplicate data present and every time you try to fetch some data by applying certain condition, you get the wrong data. How many trials before you get the right data?

To avoid all this, Keys are defined to easily identify any row of data in a table.

Let's try to understand about all the keys using a simple example.

student_id	name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	17

Let's take a simple **Student** table, with fields student_id, name, phone and age.

1. Super Key

Super Key is defined as a set of attributes within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.

In the table defined above super key would include student_id, (student_id, name), phone etc.

The first one is pretty simple as student_id is unique for every row of data, hence it can be used to identity each row uniquely.

Next comes, (student_id, name), now name of two students can be same, but their student_id can't be same hence this combination can also be a key.

Similarly, phone number for every student will be unique, hence again, phone can also be a key.

So they all are super keys.

2. Candidate Key

Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table. It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table. There can be more than one candidate key.

In our example, student_id and phone both are candidate keys for table Student**.**

- A candidate key can never be NULL or empty. And its value should be unique.
- There can be more than one candidate keys for a table.
- A candidate key can be a combination of more than one columns (attributes).

3. Primary Key

Primary key is a candidate key that is most appropriate to become the main key for any table. It is a key that can uniquely identify each record in a table.

Primary Key for this table

↓

student_id	Name	Age	phone

For the table **Student** we can make the student_id column as the primary key.

4. Composite Key

Key that consists of two or more attributes that uniquely identify any record in a table is called Composite key. But the attributes which together form the Composite key are not a key independently or individually.

Composite Key

↑

Student_id	Subject_id	marks	exam_name

Score Table - To save scores of the student for various subjects

In the above picture we have a **Score** table which stores the marks scored by a student in a particular subject.

In this table student_id and subject_id together will form the primary key, hence it is a composite key.

5. Secondary (or) Alternative key

The candidate key which are not selected as primary key are known as secondary keys or alternative keys.

➢ **Non-key Attributes**

Non-key attributes are the attributes or fields of a table, other than candidate key attributes/ fields in a table.

➢ **Non-prime Attributes**

Non-prime Attributes are attributes other than Primary Key attribute(s).

6. Foreign key

A foreign key is a key used to link two tables together.

A foreign key is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

Look at the following two tables :

"Persons" table

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

"Orders" table

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

The foreign key constraint is used to prevent actions that would destroy links between tables.

The foreign key constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

2.3 Referential Integrity

Q3. What is Referential Integrity ? Explain its constrains ?

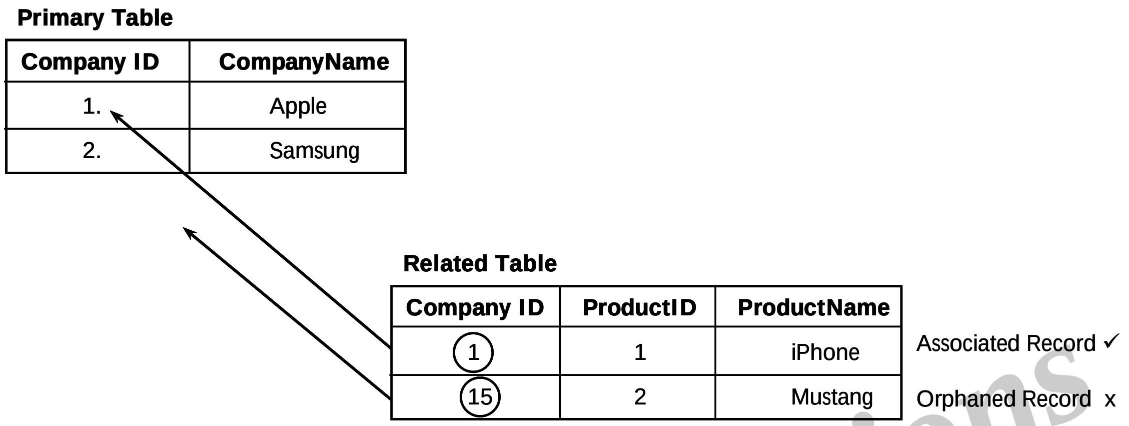
Ans :

Meaning

Referential integrity refers to the accuracy and consistency of data within a relationship.

In relationships, data is linked between two or more tables. This is achieved by having the foreign key (in the associated table) reference a primary key value (in the primary – or parent – table). Because of this, we need to ensure that data on both sides of the relationship remain intact.

So, referential integrity requires that, whenever a foreign key value is used it must reference a valid, existing primary key in the parent table or example, if we delete record number 15 in a primary table, we need to be sure that there's no foreign key in any related table with the value of 15. We should only be able to delete a primary key if there are no associated records. Otherwise, we would end up with an orphaned record.



Primary Table

Company ID	CompanyName
1.	Apple
2.	Samsung

Related Table

Company ID	ProductID	ProductName	
1	1	iPhone	Associated Record ✓
15	2	Mustang	Orphaned Record x

Here, the related table contains a foreign key value that doesn't exist in the primary key field of the primary table (i.e. the "CompanyId" field). This has resulted in an "orphaned record".

So referential integrity will prevent users from :

- ➢ Adding records to a related table if there is no associated record in the primary table.
- ➢ Changing values in a primary table that result in orphaned records in a related table.
- ➢ Deleting records from a primary table if there are matching related records.

A lack of referential integrity in a database can lead to incomplete data being returned, usually with no indication of an error. This could result in records being "lost" in the database, because they're never returned in queries or reports.

It could also result in strange results appearing in reports (such as products without an associated company).

Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL :

- ➢ **Not Null** - Ensures that a column cannot have a NULL value
- ➢ **Unique** - Ensures that all values in a column are different
- ➢ **Primary Key** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

- **Foreign Key** - Uniquely identifies a row/record in another table
- **Check** - Ensures that all values in a column satisfies a specific condition
- **Default** - Sets a default value for a column when no value is specified
- **Index** - Used to create and retrieve data from the database very quickly

2.4 Redundancy and Associated Problems

Q4. What is Data Redundancy ? How it create Problem in Database ?

Ans :

Data redundancy is a condition created within a database or data storage technology in which the same piece of data is held in two separate places.

This can mean two different fields within a single database, or two different spots in multiple software environments or platforms. Whenever data is repeated, this basically constitutes data redundancy. This can occur by accident, but is also done deliberately for backup and recovery purposes.

Redundancy means having multiple copies of same data in the database. This problem arises when a database is well 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	Himashree	7300936751	GEU	Btech	1
101	Ankit	7300936758	GEU	Btech	1
102	Ayush	7300936759	GEU	Btech	1
103	Ravi	73009367556	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:

1. Insertion Anomaly,
2. Deletion Anomaly, and
3. Updation Anomaly.

1. Insertion Anomaly

If a student detail has to be inserted whose course is not being decided yet then insertion will not be possible till the time course is decided for student.

Student_ID	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU		1

This problem happens when the insertion of a data record is not possible without adding some additional unrelated data to the record.

2. Deletion Anomaly

If the details of students in this table is deleted then the details of college will also get deleted which should not occur by common sense.

This anomaly happens when deletion of a data record results in losing some unrelated information that was stored as part of the record that was deleted from a table.

3. Updation Anomaly

Suppose if the rank of the college changes then changes will have to be all over the database which will be time-consuming and computationally costly.

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

All places should be updated

If updation do not occur at all places then database will be in inconsistent state.

2.5 SINGLE VALUED DEPENDENCIES (SVD)

Q5. Define Single value dependency and Functional Dependency ?

Ans : **(Imp.)**

i) Single Valued Dependency

If a single value or attribute in a relation or table corresponds to a single attribute or value within the relation, then it is called a single value dependency.

ii) Functional Dependency

Functional Dependency is when one attribute determines another attribute in a DBMS system. Functional Dependency plays a vital role to find the difference between good and bad database design.

Example

Employee number	**Employee Name**	**Salary**	**City**
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc.

By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

A functional dependency is denoted by an arrow $\rightarrow$

The functional dependency of X on Y is represented by X $\rightarrow$ Y

Q6. List and explain different types of Functional Dependencies ?

Ans :

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

1. Multivalued dependency

Multivalued dependency occurs in the situation where there are multiple independent multivalued attributes in a single table. A multivalued dependency is a complete constraint between two sets of attributes in a relation. It requires that certain tuples be present in a relation.

Example

Car_model	Maf_year	Color
H001	2017	Metallic
H001	2017	Green
H005	2018	Metallic
H005	2018	Blue
H010	2015	Metallic
H033	2012	Gray

In this example, maf_year and color are independent of each other but dependent on car_model. In this example, these two columns are said to be multivalue dependent on car_model.

This dependence can be represented like this:

car_model -> maf_year

car_model-> colour

2. Trivial Functional dependency

The Trivial dependency is a set of attributes which are called a trivial if the set of attributes are included in that attribute.

So, X -> Y is a trivial functional dependency if Y is a subset of X.

For example

Emp_id	Emp_name
AS555	Harry
AS811	George
AS999	Kevin

Consider this table with two columns Emp_id and Emp_name.

{Emp_id, Emp_name} -> Emp_id is a trivial functional dependency as Emp_id is a subset of {Emp_id,Emp_name}.

3. Non trivial functional dependency

Functional dependency which also known as a nontrivial dependency occurs when A->B holds true where B is not a subset of A. In a relationship, if attribute B is not a subset of attribute A, then it is considered as a non-trivial dependency.

Example

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Apple	Tim Cook	57

(Company} -> {CEO} (if we know the Company, we knows the CEO name)

But CEO is not a subset of Company, and hence it's non-trivial functional dependency.

4. Transitive dependency:

A transitive is a type of functional dependency which happens when it is indirectly formed by two functional dependencies.

Example

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Alibaba	Jack Ma	54

{Company} -> {CEO} (if we know the compay, we know its CEO's name)

{CEO } -> {Age} If we know the CEO, we know the Age

Therefore according to the rule of rule of transitive dependency :

{Company} -> {Age} should hold, that makes sense because if we know the company name, we can know his age.

Q7. State the Advantages of Functional Dependency.

Ans :

1. Functional Dependency avoids data redundancy. Therefore same data do not repeat at multiple locations in that database
2. It helps you to maintain the quality of data in the database
3. It helps you to defined meanings and constraints of databases
4. It helps you to identify bad designs
5. It helps you to find the facts regarding the database design

2.6 Normalization

Q8. What is Normalization?

(OR)

Explain briefly about Normalization.

Ans : **(July-21)**

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anamolies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e., data is logically stored.

Problems Without Normalization

If a table is not properly normalized and have data redundancy then it will not only eat up extra memory space but will also make it difficult to handle and update the database, without facing data loss. Insertion, Updation and Deletion anomalies are very frequent if database is not normalized. To understand these anomalies let us take an example of a Student table.

Roll No	name	branch	hod	office_tel
401	Akon	CSE	Mr. X	53337
402	Bkon	CSE	Mr. X	53337
403	Ckon	CSE	Mr. X	53337
404	Dkon	CSE	Mr. X	53337

In the table above, we have data of 4 Computer Sci. students. As we can see, data for the fields branch, hod (Head of Department) and office_tel is repeated for the students who are in the same branch in the college, this is Data Redundancy.

i) Insertion Anomaly

Suppose for a new admission, until and unless a student opts for a branch, data of the student cannot be inserted, or else we will have to set the branch information as NULL.

Also, if we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students.

These scenarios are nothing but Insertion anomalies.

ii) Updation Anomaly

What if Mr. X leaves the college? or is no longer the HOD of computer science department? In that case all the student records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency. This is Updation anomaly.

iii) Deletion Anomaly

In our Student table, two different informations are kept together, Student information and Branch information. Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information. This is Deletion anomaly.

Normalization Rule

Normalization rules are divided into the following normal forms :

- ➢ First Normal Form (1NF)
- ➢ Second Normal Form (2NF)
- ➢ Third Normal Form (3NF)
- ➢ BCNF
- ➢ Fourth Normal Form (4NF)

2.6.1 Rules of Data Normalization

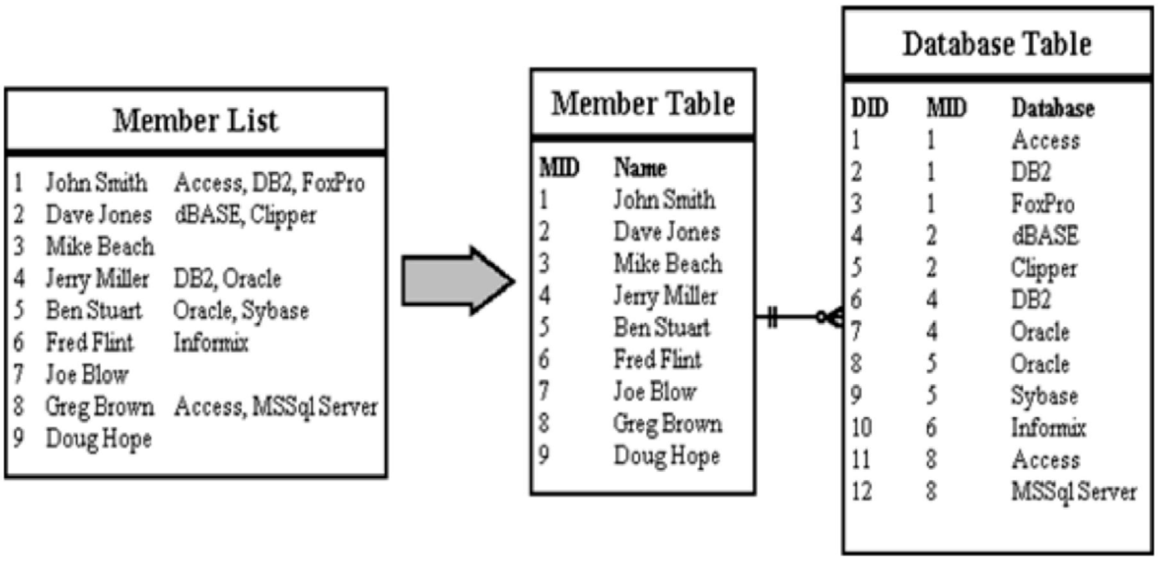
Q9. Explain the Rules of Normalization ?

Ans :

1. Eliminate Repeating Groups

In the original member list, each member name is followed by any databases that the member has experience with. Some might know many, and others might not know any. We need to perform an awkward scan of the list looking for references to DB2. This is inefficient and an extremely untidy way to store information.

Moving the known databases into a separate table helps a lot. Separating the repeating groups of databases from the member information results in first normal form. The MemberID in the database table matches the primary key in the member table, providing a foreign key for relating the two tables with a join operation. Now we can answer the question by looking in the database table for "DB2" and getting the list of members.



Member List

1	John Smith	Access, DB2, FoxPro
2	Dave Jones	dBASE, Clipper
3	Mike Beach	
4	Jerry Miller	DB2, Oracle
5	Ben Stuart	Oracle, Sybase
6	Fred Flint	Informix
7	Joe Blow	
8	Greg Brown	Access, MSSql Server
9	Doug Hope	

Member Table

MID	Name
1	John Smith
2	Dave Jones
3	Mike Beach
4	Jerry Miller
5	Ben Stuart
6	Fred Flint
7	Joe Blow
8	Greg Brown
9	Doug Hope

Database Table

DID	MID	Database
1	1	Access
2	1	DB2
3	1	FoxPro
4	2	dBASE
5	2	Clipper
6	4	DB2
7	4	Oracle
8	5	Oracle
9	5	Sybase
10	6	Informix
11	8	Access
12	8	MSSql Server

2. Eliminate Redundant Data

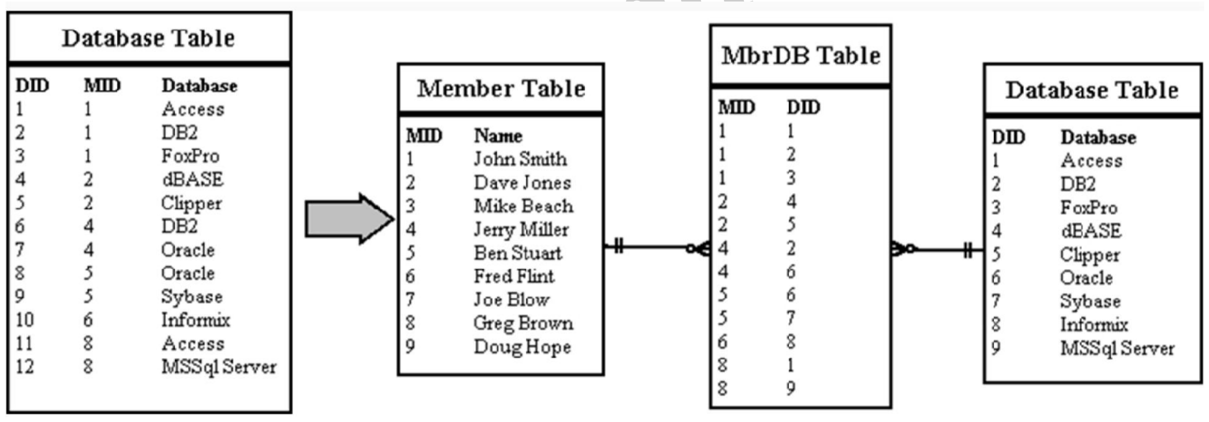
In the Database Table, the primary key is made up of the MemberID and the DatabaseID. This makes sense for the "Where Learned" and "Skill Level" attributes, since they will be different for every member/database combination. But the database name depends only on the DatabaseID. The same database name will appear redundantly every time its associated ID appears in the Database Table.

Suppose you want to reclassify a database - give it a different DatabaseID. The change has to be made for every member that lists that database! If you miss some, you'll have several members with the same database under different IDs. This is an update anomaly.

Or suppose the last member listing a particular database leaves the group. His records will be removed from the system, and the database will not be stored anywhere! This is a delete anomaly. To avoid these problems, we need **second normal form**.

To achieve this, separate the attributes depending on both parts of the key from those depending only on the DatabaseID. This results in two tables: "Database" which gives the name for each DatabaseID, and "MemberDatabase" which lists the databases for each member.

Now we can reclassify a database in a single operation: look up the DatabaseID in the "Database" table and change its name. The result will instantly be available throughout the application.



Database Table

DID	MID	Database
1	1	Access
2	1	DB2
3	1	FoxPro
4	2	dBASE
5	2	Clipper
6	4	DB2
7	4	Oracle
8	5	Oracle
9	5	Sybase
10	6	Informix
11	8	Access
12	8	MSSql Server

Member Table

MID	Name
1	John Smith
2	Dave Jones
3	Mike Beach
4	Jerry Miller
5	Ben Stuart
6	Fred Flint
7	Joe Blow
8	Greg Brown
9	Doug Hope

MbrDB Table

MID	DID
1	1
1	2
1	3
2	4
2	5
4	2
4	6
5	6
5	7
6	8
8	1
8	9

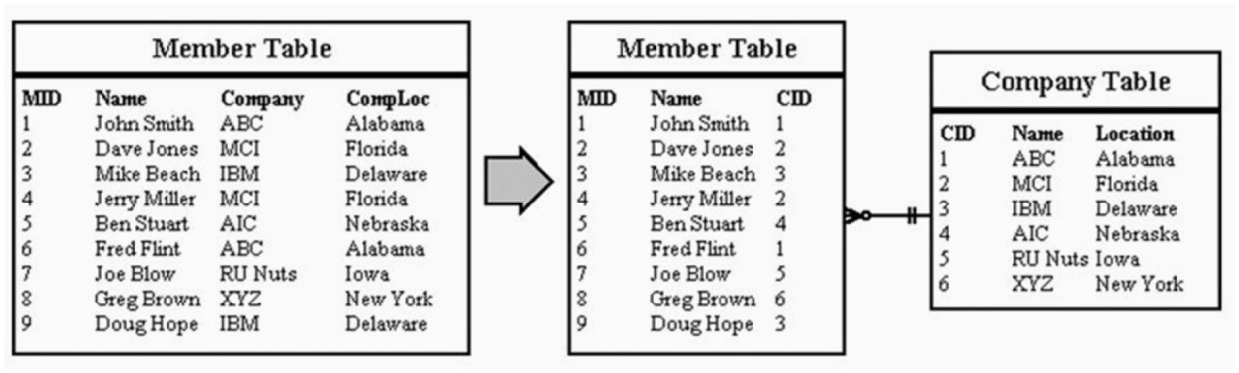
Database Table

DID	Database
1	Access
2	DB2
3	FoxPro
4	dBASE
5	Clipper
6	Oracle
7	Sybase
8	Informix
9	MSSql Server

3. Eliminate Columns Not Dependent On Key

The Member table satisfies first normal form - it contains no repeating groups. It satisfies second normal form - since it doesn't have a multivalued key. But the key is MemberID, and the company name and location describe only a company, not a member. To achieve third normal form, they must be moved into a separate table. Since they describe a company, CompanyCode becomes the key of the new "Company" table.

The motivation for this is the same for second normal form: we want to avoid update and delete anomalies. For example, suppose no members from the IBM were currently stored in the database. With the previous design, there would be no record of its existence, even though 20 past members were from IBM!



Member Table

MID	Name	Company	CompLoc
1	John Smith	ABC	Alabama
2	Dave Jones	MCI	Florida
3	Mike Beach	IBM	Delaware
4	Jerry Miller	MCI	Florida
5	Ben Stuart	AIC	Nebraska
6	Fred Flint	ABC	Alabama
7	Joe Blow	RU Nuts	Iowa
8	Greg Brown	XYZ	New York
9	Doug Hope	IBM	Delaware

Member Table

MID	Name	CID
1	John Smith	1
2	Dave Jones	2
3	Mike Beach	3
4	Jerry Miller	2
5	Ben Stuart	4
6	Fred Flint	1
7	Joe Blow	5
8	Greg Brown	6
9	Doug Hope	3

Company Table

CID	Name	Location
1	ABC	Alabama
2	MCI	Florida
3	IBM	Delaware
4	AIC	Nebraska
5	RU Nuts	Iowa
6	XYZ	New York

4. Isolate Independent Multiple Relationships

This applies only to designs that include one-to-many and many-to-many relationships. An example of one-to-many is that one company can employ many members. An example of a many-to-many relationship is that a member can know many databases and several members might know the same database.

Suppose we want to add a new attribute to the MemberDatabase table called "Attire". This way we can look for members that not only know DB2, but also typically wear a suit and tie. Fourth normal form dictates against this (using the MemberDatabase table, not wearing suits and ties). The two attributes do not share a meaningful relationship.

5. Isolate Semantically Related Multiple Relationships

Usually, related attributes belong together. For example, if we really wanted to record which databases each member works on wearing which kinds of clothes, we would want to keep the attire attribute in the MemberDatabase table. But there are times when special characteristics of the data make it more efficient to separate even logically related attributes.

Imagine that our system will record which jobs are available in each company, and which schools typically supply candidates to those companies. This suggests a CompanySchoolJob table which satisfies fourth normal form. As long as any company can use any candidates from any school, this works fine.

Now suppose a law is passed to prevent exclusive arrangements: a company accepting candidates must accept them from all schools it deals with. In other words, if IBM is hiring DBAs and wants to maintain a relationship with MIT, it must accept DBAs from MIT.

The need for fifth normal form becomes clear when we consider inserts and deletes. Suppose a company decides to create 3 new jobs types: HTML DBA, Java Programmer and Underwater DB2 DBA. Suppose further that it already deals with three schools that can supply candidates for those positions. This will require nine new rows in the database, one for each school/job combination.

Breaking up the table reduces the number of inserts to six. Here are the tables necessary for fifth normal form, shown with the six newly inserted rows in bold type. If an application involves significant update activity, fifth normal form can mean important savings. Note that these combination tables develop naturally out of entity-relationship analysis.

2.6.2 First Normal form (1NF)

Q10. Explain briefly about 1NF.

Ans : **(July-21, Oct.-20, Oct.-19, June-19, June-18, Imp.)**

A relation which contains no multivalued attributes i.e., In a relation the value at the intersection of each row and column must be atomic. Thus a table that contains multivalued attributes or repeating groups is not a relation.

Example: Give an example which is not in 1NF.

empno	ename	Dept-Name	Salary	Course-Title	Date-completed
100	Venkat	Comp. Sc.	22000	C++	8/9/2001
			22000	Oracle	10/9/2001
101	VASU	Electronics	50000	Java	19/9/2001
				ASP	20/9/2001

The above table contain multivalued attributes. So the above relation is not in 1NF. A table with multivalued attributes is converted to a relation in first normal form by extending the data in each column to fill cells that are empty because of the multivalued attributes.

empno	ename	Dept-Name	Salary	Course-Title	Date-completed
100	Venkat	Comp. Sc.	22000	C++	8/9/2001
100	Venkat	Comp. Sc.	22000	Oracle	10/9/2001
101	VASU	Electronics	50000	Java	19/9/2001
101	VASU	Electronics	50000	ASP	20/9/2001

So the above table is not having any multivalued attributes. The above table is an example for the relation which is 1NF.

2.6.3 Second Normal Form (2NF)

Q11. Explain briefly about 2NF.

Ans : **(July-11, Oct.-20, Oct.-19, June-19, June-18, Imp.)**

A relation is in second normal form (2NF) if it is in first normal form and every nonkey attributes is fully functionally dependent on the primary key. Thus no nonkey attribute is functionally dependent on part (but not all) of the primary key. A relation that is in first normal form will be in second normal form if any one of the following conditions applies :

1. No nonkey attributes exist in the relation (thus all of the attributes in the relation are components of the primary key).
2. The primary key consists of only one attribute.
3. Every nonkey attribute is functionally dependent on the full set of primary key attributes.

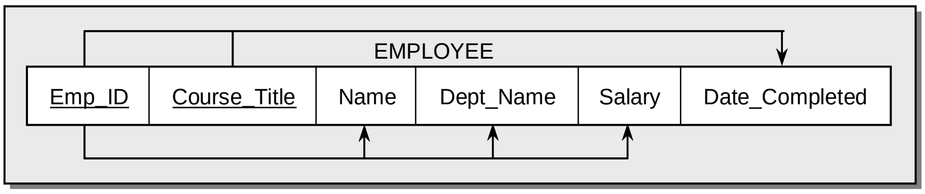


Fig.: Functional dependencies in EMPLOYEE

The figure EMPLOYEE above is an example of a relation that is not in second normal form. The primary key for this relation is the composite key Emp_ID, Course_Title. Therefore the nonkey attributes Name, Dept_Name, and Salary are functionally dependent on part of the primary key (Emp_ID) but not on Course_Title. These dependencies are shown graphically in figure above.

A partial functional dependency is a functional dependency in which one or more nonkey attributes are functionally dependent on part of the primary key. The partial functional dependency in EMPLOYEE creates redundancy in that relation, which results in anomalies when the table is updated.

To convert a relation to second normal form, we decompose the relation into new relations that satisfy one (or more) of the conditions described above. EMPLOYEE is decomposed into the following two relations :

1. EMPLOYEE (<u>Emp_ID</u>, Name, Dept_Name, Salary) This relation satisfies condition 1 above and is in second normal form.
2. EMP_COURSE (<u>Emp_ID</u>, <u>Course_Title</u>, Date_Completed) This relation satisfies property 3 above and is also in second normal form.

2.6.4 Third Normal Form (3NF)

Q12. Explain briefly about 3NF.

Ans : **(Oct.-20, Oct.-19, June-19, Imp.)**

Third Normal Form (3NF)

A relation is said to be in third normal form if the following two conditions are satisfied,

(i) A relation must be in 2NF

(ii) A relation must not have any transitive dependencies.

In other words, it can be said that a relation is in 3NF, if every determinant is a key i.e., for every functional dependency A $\rightarrow$ B, A is a key. Basically, determinant is an attribute whose value determines the value of other attributes.

Converting 2NF to 3NF

The conversion of a relation in 2NF into a relation in 3NF can be done by performing the following steps,

Step 1

In this step, determinants in transitive dependencies are identified and are written as a primary key for a new table.

Step 2

In this step, the attributes that depend on the determinant are identified along with the dependency between the attribute and its respective determinant.

Step 3

In this step, the dependent attributes that were identified in step 2 are removed from every table consisting of the corresponding transitive relationship. This deletion results in a new table that consists of determinant and also its dependent attribute that was present in the transitive relationship. The determinant acts as the primary key in the new table.

The table generated after performing these steps does not contain any inappropriate dependencies and therefore it can be said that the original table is now in 3NF.

Example

Consider the dependency diagram of Employee table from 2NF.

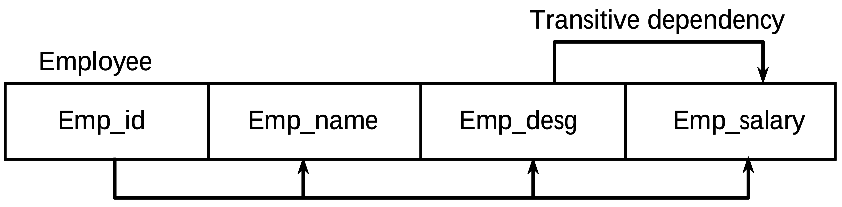


Fig.: Dependency Diagram of "Employee"

The above dependency diagram consists of a transitive dependency, Emp desg $\rightarrow$ Emp_salary, which must be eliminated in order to convert the relation into 3NF. This elimination is done by performing the above mentioned steps.

Step 1

Identify the determinant in transitive dependency Emp_desg

Step 2

Identify dependent attributes Emp_desg $\rightarrow$ Empsalary

'Empsalary' is the dependent attribute since its value depends on the value of its determinant (i.e., Emp_desg)

Step 3

Remove the dependent attribute

Emp_salary is deleted

$\therefore$ The dependency of "Employee" table is defined as,

Emp id $\rightarrow$ Emp_name, Emp_desg

Where Emp_desg acts as a foreign key in Employee table, The relation schema and dependency diagrams generated after completion of step 1 to step 3 for Employee relations are,

(i) Department

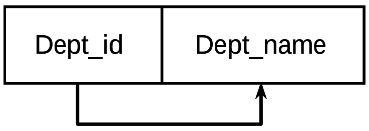


Fig.: Dependency Diagram of "Department" Table

Schema: Department $\left(\frac{\text{Dept\_id}}{\text{Primary key}}, \text{Dept\_name}\right)$

(ii) Employee

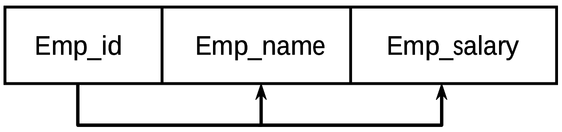


Fig.: Dependency Diagram for "Employee" Table

Schema: Employee $\left(\frac{\text{Emp\_id}}{\text{Primary key}}, \text{Emp\_name, Emp\_desg}\right)$

(iii) Set_salary

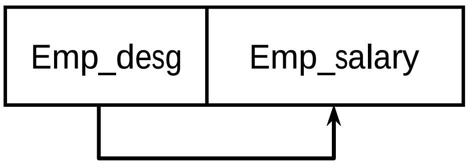


Fig.: Dependency Diagram for "Salary" Table

Schema: Set_salary $\left(\frac{\text{Emp\_desg}}{\text{Primary key}}, \text{Emp\_salary}\right)$

(iv) Salary

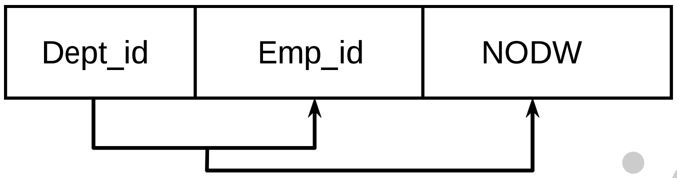


Fig.: Dependency Diagram for Salary

Schema: Salary $\left(\frac{\text{Dept\_id, Emp\_id}}{\text{Primary key}}, \text{NODW}\right)$

This way of concerting a 21NF relation into 3NF relation helps in eliminating transitive dependency which in turn removes all the possibilities of occurrence of different data anomalies.

2.6.5 Boyee Codd Normal Form (BCNF)

Q13. Explain briefly about BCNF.

Ans : **(Imp.)**

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Example

Suppose there is a company wherein employees work in **more than one department**. They store the data like this :

emp_id	**emp_nationality**	**emp_dept**	**dept_type**	**dept_no_of_emp**
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above

emp_id -> emp_nationality

emp_dept -> {dept_type, dept_no_of_emp}

Candidate key: {emp_id, emp_dept}

The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:

emp_nationality table

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies

emp_id -> emp_nationality

emp_dept -> {dept_type, dept_no_of_emp}

Candidate keys

For first table: emp_id

For second table: emp_dept

For third table: {emp_id, emp_dept}

This is now in BCNF as in both the functional dependencies left side part is a key.

Q14. Explain the advantages and disadvantages of normalization.

Ans :

Advantages of Normalization

Here we can see why normalization is an attractive prospect in RDBMS concepts.

1. A smaller database can be maintained as normalization eliminates the duplicate data. Overall size of the database is reduced as a result.
2. Better performance is ensured which can be linked to the above point. As databases become lesser in size, the passes through the data becomes faster and shorter thereby improving response time and speed.
3. Narrower tables are possible as normalized tables will be fine-tuned and will have lesser columns which allows for more data records per page.
4. Fewer indexes per table ensures faster maintenance tasks (index rebuilds).
5. Also realizes the option of joining only the tables that are needed.

Disadvantages of Normalization

1. More tables to join as by spreading out data into more tables, the need to join table's increases and the task becomes more tedious. The database becomes harder to realize as well.
2. Tables will contain codes rather than real data as the repeated data will be stored as lines of codes rather than the true data. Therefore, there is always a need to go to the lookup table.
3. Data model becomes extremely difficult to query against as the data model is optimized for applications, not for ad hoc querying. (Ad hoc query is a query that cannot be determined before the issuance of the query. It consists of an SQL that is constructed dynamically and is usually constructed by desktop friendly query tools.). Hence it is hard to model the database without knowing what the customer desires.
4. As the normal form type progresses, the performance becomes slower and slower.
5. Proper knowledge is required on the various normal forms to execute the normalization process efficiently. Careless use may lead to terrible design filled with major anomalies and data inconsistency.

Q15. What are the differences between 3NF and BCNF?

Ans : **(Imp.)**

BCNF	3NF
1. If a functional dependency A $\rightarrow$ B is present, then BCNF allows this dependency in a relation only if 'A' is a candidate key.	1. If a functional dependency A $\rightarrow$ B is present, then 3NF allows this dependency to be in a relation if and only if, 'B' is a primary key attribute and 'A' is not a candidate key.
2. A relation in BCNF is also in 3NF.	2. A relation in 3NF need not be in BCNF.
3. Decomposition of BCNF relations is not possible.	3. It is possible to decompose any relational schema into a set of 3NF relations, by using decompositions that possess certain desirable properties like lossless-join, dependency preservation.
4. BCNF guarantees that no redundancy can occur using only the information about functional dependencies.	4. There is a possibility of some redundancy to occur because if every transitive dependency is not eliminated, then null values are to be used in order to present some meaningful relationship.

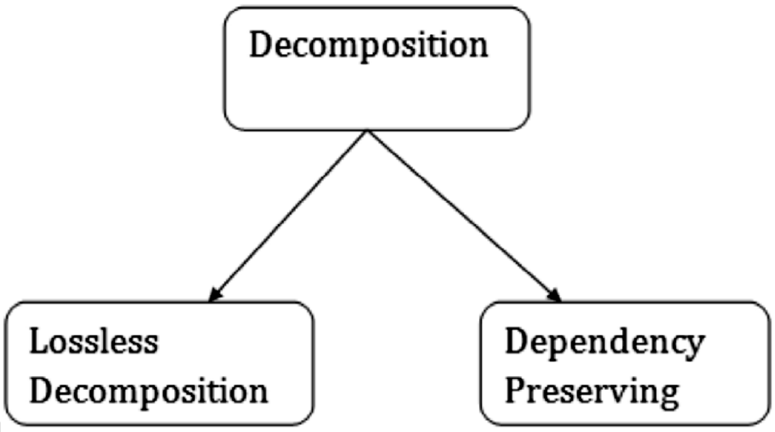
2.7 Decomposition (Attribute Preservation, Lossless Join Dependency, Dependency Preservation)

Q16. Define Decomposition ? Explain the types of decomposition.

Ans :

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

Types of Decomposition



1. Lossless Decomposition

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

Example

EMPLOYEE_DEPARTMENT table

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

The above relation is decomposed into two relations EMPLOYEE and DEPARTMENT

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY
22	Denim	28	Mumbai
33	Alina	25	Delhi
46	Stephan	30	Bangalore
52	Katherine	36	Mumbai
60	Jack	40	Noida

DEPARTMENT table

DEPT_ID	EMP_ID	DEPT_NAME
827	22	Sales
438	33	Marketing
869	46	Finance
575	52	Production
678	60	Testing

Now, when these two relations are joined on the common column "EMP_ID", then the resultant relation will look like:

Employee $\bowtie$ Department

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

Hence, the decomposition is Lossless join decomposition.

2. Dependency Preservation

- 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 relation R1 and R2, then the dependencies of R either must be a part of R1 or R2 or 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->BC). The relational R is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of relation R1(ABC).

Q17. What is meant by Attribute preservation.

Ans :

This is simple requirement that involves preserving all the attributes that were there in the relation that is being decomposed.

All attributes must be preserved through the process of normalization.

Start with universal relation schema R

R={A1,A2,A3,——,An}, the set of attributes

D is a decomposition of R such that

D = {R1, R2, —— Rm} and R=U Ri.

2.8 File Organization

Q18. Define File organization ? List the factors to be considered in file organization?

Ans : **(July - 21, Imp.)**

- The File is a collection of records. Using the primary key, we can access the records. The type and frequency of access can be determined by the type of file organization which was used for a given set of records.
- File organization is a logical relationship among various records. This method defines how file records are mapped onto disk blocks.
- File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.
- The first approach to map the database to the file is to use the several files and store only one fixed length record in any given file. An alternative approach is to structure our files so that we can contain multiple lengths for records.
- Files of fixed length records are easier to implement than the files of variable length records.

Factors to be considered on File organization

There are several methods of file organization and each one is suited for a particular task or purpose. Here are the factors to consider before choosing a file organization method;

1. **Frequency of update:** A file that needs to be updated every now and then needs an organization method that will allow easy retrieval of information and ease of updating, example of such a file is the transaction file.
2. **File activity:** Different files have different activities, example a sort file is used to sort data in sequential order and therefore sequential method would be appropriate for such a file.
3. **File access method:** Definitely different files have different methods of being accessed, example a reference file is accessed using random method for easy retrieval of data.
4. **Nature of the system:** Files that are used in a particular system will depend on the nature of the system i.e., the suitable organization method for that particular system.
5. **Master file medium:** The master file is the main file for keeping permanent updates of records from transaction files and other sources, the medium by which it is updated will determine the organization method to be used.

2.8.1 Physical Database Design Issues

Q19. Discuss physical database designing issues?

Ans :

1. **Relying on the defaults:** Often times the default settings are not optimal and can cause performance problems (or operational problems)
2. **Not basing the physical on a logical model:** Every database should begin its life as part of a logical data model; failure to do so will probably cause data integrity issues.
3. **Over-relying on logical design:** I sometimes call this the dreaded disease of ERwin-itis; you know, where the DDL is spit

out by the design tool and no changes can be made. This can be as bad as no logical model.

4. **Normalization problems:** Could exhibit itself as too much denormalization… or even over-normalization. Be sure the design is usable given today's DBMS capabilities and your organization's requirements.

5. **Not enough indexes:** Indexes optimize data access – build as many of them as you need to assure optimal performance (without causing data modification issues… because indexes need to be modified when the table data is updated/inserted/deleted).

6. **Indexing by table, not by workload:** Indexes should be created to optimize your query workload. Yet many still create indexes when creating tables… well before the SQL to access the data is known.

7. **Too much (or not enough) free space:** Use free space appropriately. Free space provides room for data to grow in between reorganizations. Don't specify too much free space (or queries that must read multiple pages of data may not perform well). Don't specify too little free space (or data may become disorganized too quickly). And if the data is static, then don't specify any free space at all!

8. **Failing to plan for data purging or archiving:** If you never plan to remove data from your tables then they will grow and grow and grow and… eventually, they may become unmanageable.

9. **Failure to share data:** Not IBM DB2 Data Sharing, but sharing data. Databases are designed for sharing data among users and applications. Failing to share data is the reason you may have 287 different customer databases (for example).

10. **Kludging:** By kludging, I mean trying to avoid a certain feature/function and instead using something in a way it was not intended… for example: avoiding NULLs at all costs, storing numbers in CHAR columns, storing dates in CHAR columns, etc

2.8.2 Storage of Database on Hard Disks

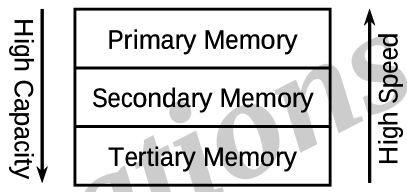
Q20. Explain the hierarchy of storing data in memory.

(OR)

Discuss briefly about storage of data on hard disk.

Ans : **(Imp.)**

Databases are stored in file formats, which contain records. At physical level, the actual data is stored in electromagnetic format on some device. These storage devices can be broadly categorized into three types -



➢ **Primary Storage:** The memory storage that is directly accessible to the CPU comes under this category. CPU's internal memory (registers), fast memory (cache), and main memory (RAM) are directly accessible to the CPU, as they are all placed on the motherboard or CPU chipset. This storage is typically very small, ultra-fast, and volatile. Primary storage requires continuous power supply in order to maintain its state. In case of a power failure, all its data is lost.

➢ **Secondary Storage:** Secondary storage devices are used to store data for future use or as backup. Secondary storage includes memory devices that are not a part of the CPU chipset or motherboard, for example, magnetic disks, optical disks (DVD, CD, etc.), hard disks, flash drives, and magnetic tapes.

➢ **Tertiary Storage:** Tertiary storage is used to store huge volumes of data. Since such storage devices are external to the computer system, they are the slowest in speed. These storage devices are mostly used to take the back up of an entire system. Optical disks and magnetic tapes are widely used as tertiary storage.

➢ **Memory Hierarchy**

A computer system has a well-defined hierarchy of memory. A CPU has direct access to it main memory as well as its inbuilt registers. The access time of the main memory is obviously less than the CPU speed. To minimize this speed mismatch, cache memory is introduced. Cache memory provides the fastest access time and it contains data that is most frequently accessed by the CPU.

The memory with the fastest access is the costliest one. Larger storage devices offer slow speed and they are less expensive, however they can store huge volumes of data as compared to CPU registers or cache memory.

➢ **Magnetic Disks**

Hard disk drives are the most common secondary storage devices in present computer systems. These are called magnetic disks because they use the concept of magnetization to store information. Hard disks consist of metal disks coated with magnetizable material. These disks are placed vertically on a spindle. A read/write head moves in between the disks and is used to magnetize or de-magnetize the spot under it. A magnetized spot can be recognized as 0 (zero) or 1 (one).

Hard disks are formatted in a well-defined order to store data efficiently. A hard disk plate has many concentric circles on it, called tracks. Every track is further divided into sectors. A sector on a hard disk typically stores 512 bytes of data.

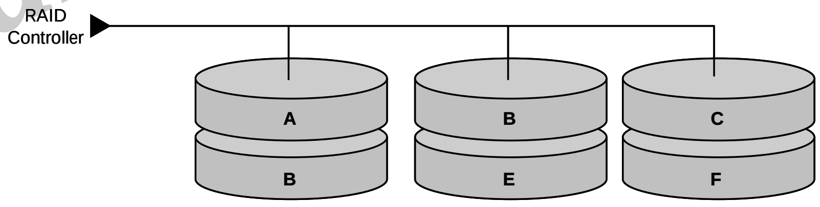
➢ **Redundant Array of Independent Disks**

RAID or Redundant Array of Independent Disks, is a technology to connect multiple secondary storage devices and use them as a single storage media.

RAID consists of an array of disks in which multiple disks are connected together to achieve different goals. RAID levels define the use of disk arrays.

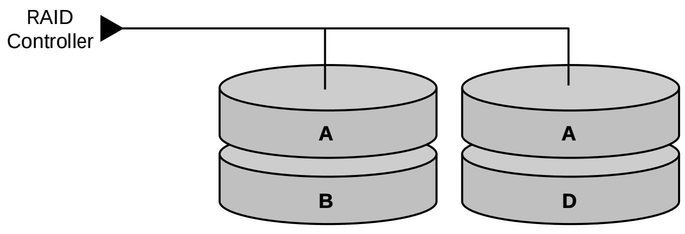
➢ **RAID 0**

In this level, a striped array of disks is implemented. The data is broken down into blocks and the blocks are distributed among disks. Each disk receives a block of data to write/read in parallel. It enhances the speed and performance of the storage device. There is no parity and backup in Level 0.



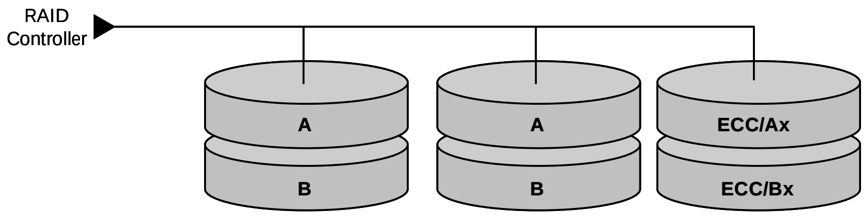
➢ **RAID 1**

RAID 1 uses mirroring techniques. When data is sent to a RAID controller, it sends a copy of data to all the disks in the array. RAID level 1 is also called mirroring and provides 100% redundancy in case of a failure.



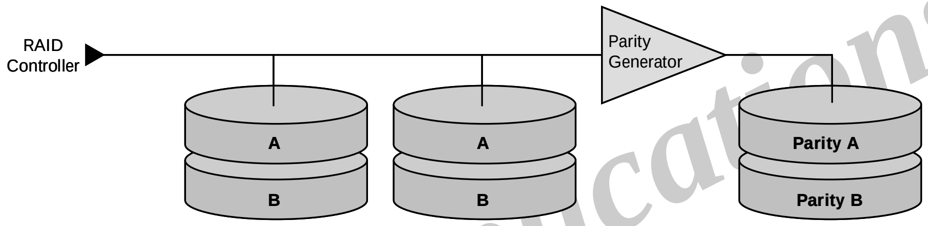
- **RAID 2**

RAID 2 records Error Correction Code using Hamming distance for its data, striped on different disks. Like level 0, each data bit in a word is recorded on a separate disk and ECC codes of the data words are stored on a different set disks. Due to its complex structure and high cost, RAID 2 is not commercially available.



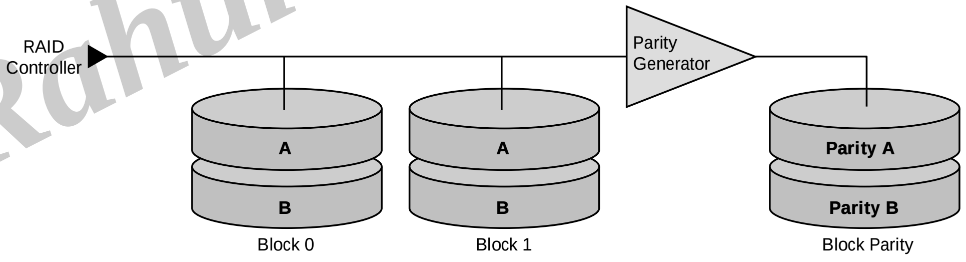
- **RAID 3**

RAID 3 stripes the data onto multiple disks. The parity bit generated for data word is stored on a different disk. This technique makes it to overcome single disk failures.



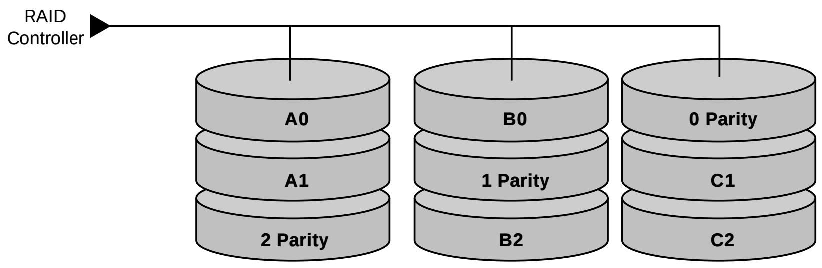
- **RAID 4**

In this level, an entire block of data is written onto data disks and then the parity is generated and stored on a different disk. Note that level 3 uses byte-level striping, whereas level 4 uses block-level striping. Both level 3 and level 4 require at least three disks to implement RAID.



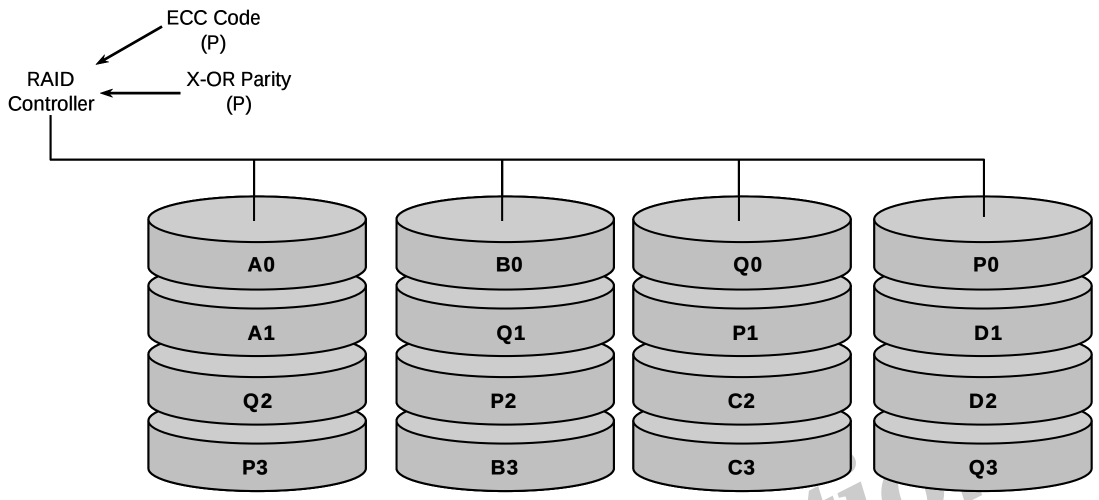
- **RAID 5**

RAID 5 writes whole data blocks onto different disks, but the parity bits generated for data block stripe are distributed among all the data disks rather than storing them on a different dedicated disk.



- **RAID 6**

  RAID 6 is an extension of level 5. In this level, two independent parities are generated and stored in distributed fashion among multiple disks. Two parities provide additional fault tolerance. This level requires at least four disk drives to implement RAID.



2.9 File Organization

Q21. What is File Organization ? What are the objectives of file organization.

Ans : **(June-18)**

- The File is a collection of records. Using the primary key, we can access the records. The type and frequency of access can be determined by the type of file organization which was used for a given set of records.
- File organization is a logical relationship among various records. This method defines how file records are mapped onto disk blocks.
- File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.
- The first approach to map the database to the file is to use the several files and store only one fixed length record in any given file. An alternative approach is to structure our files so that we can contain multiple lengths for records.
- Files of fixed length records are easier to implement than the files of variable length records.

Objective

- It contains an optimal selection of records, i.e., records can be selected as fast as possible.
- To perform insert, delete or update transaction on the records should be quick and easy.
- The duplicate records cannot be induced as a result of insert, update or delete.
- For the minimal cost of storage, records should be stored efficiently.

2.9.1 Types

2.9.1.1 Sequential, Indexed, Heap, Hash

Q22. Explain Different types of File Organizations in DBMS.

(OR)

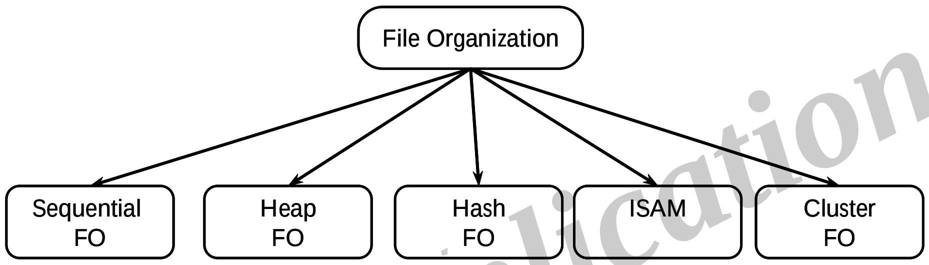
Explain types of file organizations.

Ans : **(July-21, Oct.-20, Oct.-19, June-18, Imp.)**

Types of file organization

File organization contains various methods. These particular methods have pros and cons on the basis of access or selection. In the file organization, the programmer decides the best-suited file organization method according to his requirement.

Types of file organization are as follows:



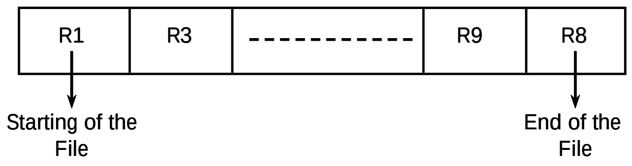
1. Sequential File Organization

This method is the easiest method for file organization. In this method, files are stored sequentially. This method can be implemented in two ways:

(a) File Method

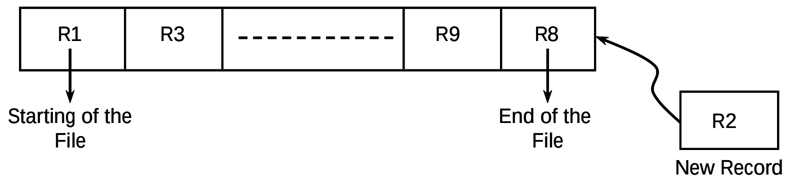
➢ It is a quite simple method. In this method, we store the record in a sequence, i.e., one after another. Here, the record will be inserted in the order in which they are inserted into tables.

➢ In case of updating or deleting of any record, the record will be searched in the memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.



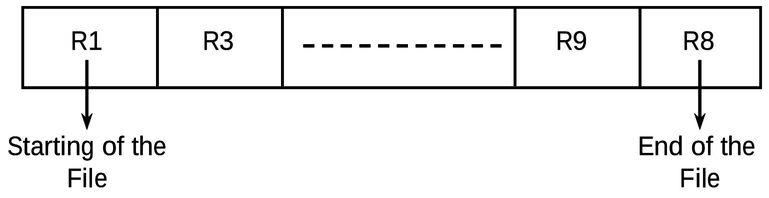
Insertion of the new record

Suppose we have four records R1, R3 and so on upto R9 and R8 in a sequence. Hence, records are nothing but a row in the table. Suppose we want to insert a new record R2 in the sequence, then it will be placed at the end of the file. Here, records are nothing but a row in any table.



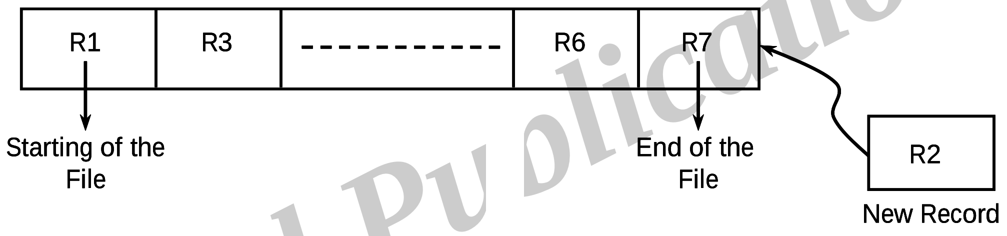
(b) Sorted File Method

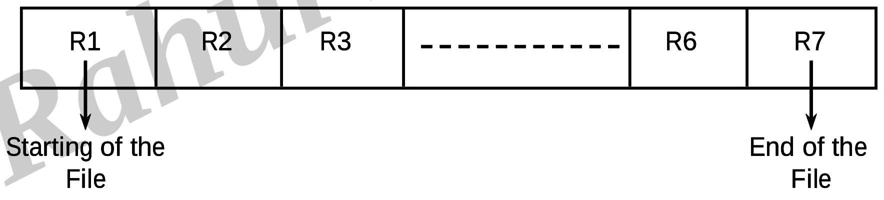
- In this method, the new record is always inserted at the file's end, and then it will sort the sequence in ascending or descending order. Sorting of records is based on any primary key or any other key.
- In the case of modification of any record, it will update the record and then sort the file, and lastly, the updated record is placed in the right place.



Insertion of the new record

Suppose there is a preexisting sorted sequence of four records R1, R3 and so on upto R6 and R7. Suppose a new record R2 has to be inserted in the sequence, then it will be inserted at the end of the file, and then it will sort the sequence.





Pros of sequential file organization

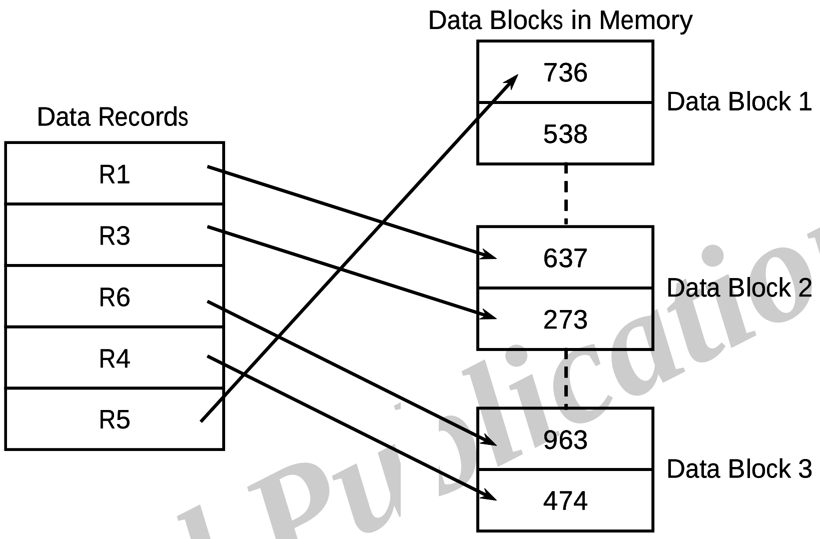
- It contains a fast and efficient method for the huge amount of data.
- In this method, files can be easily stored in cheaper storage mechanism like magnetic tapes.
- It is simple in design. It requires no much effort to store the data.
- This method is used when most of the records have to be accessed like grade calculation of a student, generating the salary slip, etc.
- This method is used for report generation or statistical calculations.

Cons of sequential file organization

- It will waste time as we cannot jump on a particular record that is required but we have to move sequentially which takes our time.
- Sorted file method takes more time and space for sorting the records.

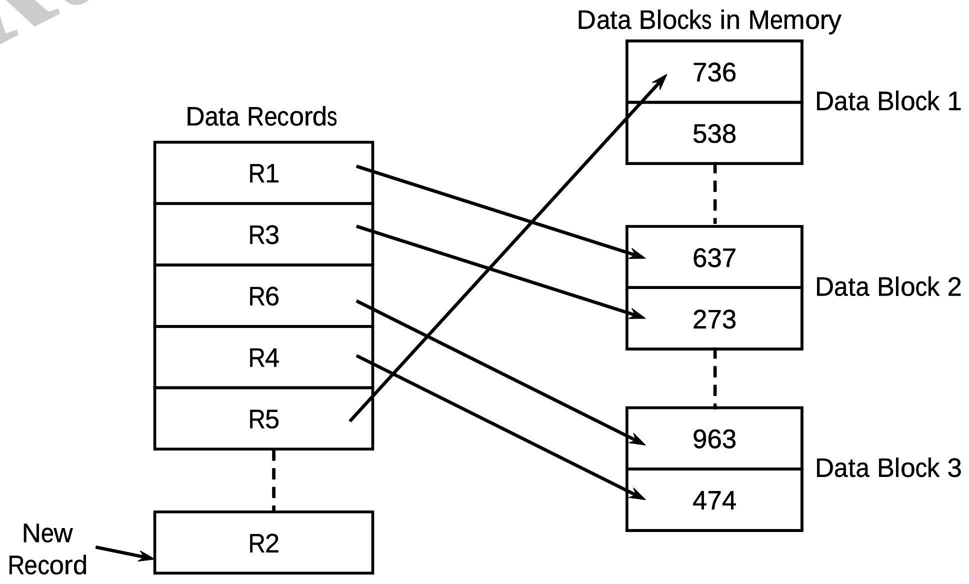
2. Heap file organization

- It is the simplest and most basic type of organization. It works with data blocks. In heap file organization, the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records.
- When the data block is full, the new record is stored in some other block. This new data block need not to be the very next data block, but it can select any data block in the memory to store new records. The heap file is also known as an unordered file.
- In the file, every record has a unique id, and every page in a file is of the same size. It is the DBMS responsibility to store and manage the new records.



Insertion of a new record

Suppose we have five records R1, R3, R6, R4 and R5 in a heap and suppose we want to insert a new record R2 in a heap. If the data block 3 is full then it will be inserted in any of the database selected by the DBMS, let's say data block 1.



If we want to search, update or delete the data in heap file organization, then we need to traverse the data from staring of the file till we get the requested record.

If the database is very large then searching, updating or deleting of record will be time-consuming because there is no sorting or ordering of records. In the heap file organization, we need to check all the data until we get the requested record.

Pros of Heap file organization

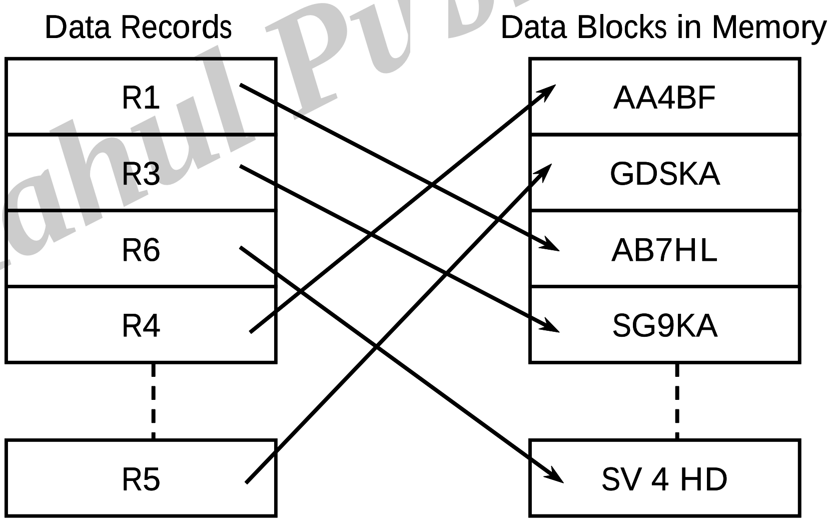
- It is a very good method of file organization for bulk insertion. If there is a large number of data which needs to load into the database at a time, then this method is best suited.
- In case of a small database, fetching and retrieving of records is faster than the sequential record.

Cons of Heap file organization

- This method is inefficient for the large database because it takes time to search or modify the record.
- This method is inefficient for large databases.

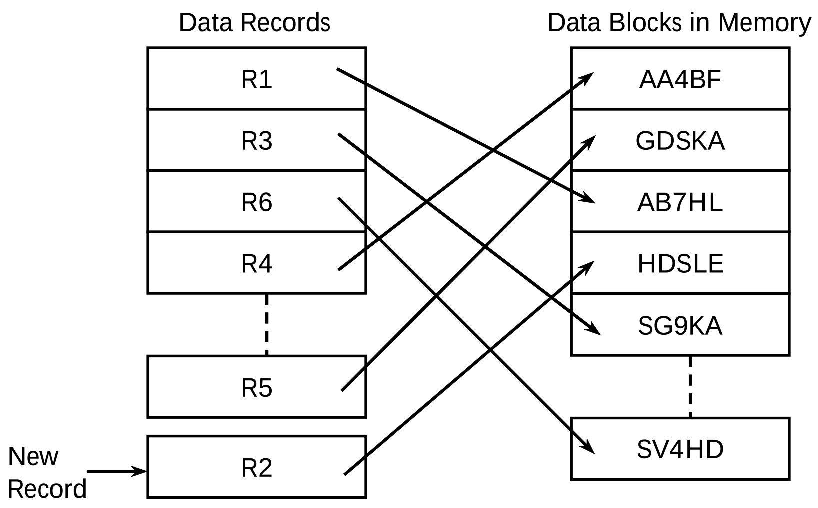
3. Hash File Organization

Hash File Organization uses the computation of hash function on some fields of the records. The hash function's output determines the location of disk block where the records are to be placed.



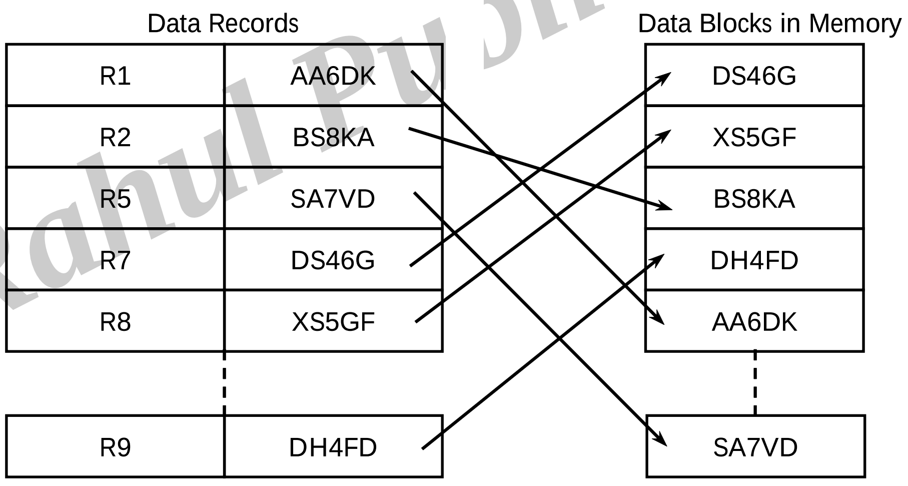
When a record has to be received using the hash key columns, then the address is generated, and the whole record is retrieved using that address. In the same way, when a new record has to be inserted, then the address is generated using the hash key and record is directly inserted. The same process is applied in the case of delete and update.

In this method, there is no effort for searching and sorting the entire file. In this method, each record will be stored randomly in the memory.



4. 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. 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.

Pros 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.

Cons 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.

5. Cluster file organization

- When the two or more records are stored in the same file, it is known as clusters. These files will have two or more tables in the same data block, and key attributes which are used to map these tables together are stored only once.
- This method reduces the cost of searching for various records in different files.
- The cluster file organization is used when there is a frequent need for joining the tables with the same condition. These joins will give only a few records from both tables. In the given example, we are retrieving the record for only particular departments. This method can't be used to retrieve the record for the entire department.

EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEP_ID
1.	John	Delhi	14
2.	Robert	Gujarat	12
3.	David	Mumbai	15
4.	Amelia	Meerut	11
5.	Kristen	Noida	14
6.	Jackson	Delhi	13
7.	Amy	Bihar	10
8.	Sonoo	UP	12

DEPARTMENT

DEP_ID	DEP_NAME
10	Math
11	English
12	Java
13	Physics
14	Civil
15	Chemistry

Cluster Key

DEP_ID	DEP_NAME	EMP_ID	EMP_NAME	ADDRESS
10	Math	7	Amy	Bihar
11	English	4	Amelia	Meerut
12	Java	2	Robert	Gujarat
12		8	Sonoo	UP
13	Physics	6	Jackson	Delhi
14	Civil	1	John	Delhi
14		5	Kristen	Noida
15	Chemistry	3	David	Mumbai

In this method, we can directly insert, update or delete any record. Data is sorted based on the key with which searching is done. Cluster key is a type of key with which joining of the table is performed.

Types of Cluster File Organization

Cluster file organization is of two types:

1. **Indexed Clusters**

   In indexed cluster, records are grouped based on the cluster key and stored together. The above EMPLOYEE and DEPARTMENT relationship is an example of an indexed cluster. Here, all the records are grouped based on the cluster key- DEP_ID and all the records are grouped.

2. **Hash Clusters**

   It is similar to the indexed cluster. In hash cluster, instead of storing the records based on the cluster key, we generate the value of the hash key for the cluster key and store the records with the same hash key value.

Pros of Cluster file organization

- ➢ The cluster file organization is used when there is a frequent request for joining the tables with same joining condition.
- ➢ It provides the efficient result when there is a 1:M mapping between the tables.

Cons of Cluster file organization

- ➢ This method has the low performance for the very large database.
- ➢ If there is any change in joining condition, then this method cannot use. If we change the condition of joining then traversing the file takes a lot of time.

2.10 Types of Indexes and Tree Structure

Q23. What is Index? How index is helpful in File organization ? Explain the types of Indexes?

Ans :

We know that data is stored in the form of records. Every record has a key field, which helps it to be recognized uniquely.

Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing in database systems is similar to what we see in books.

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

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

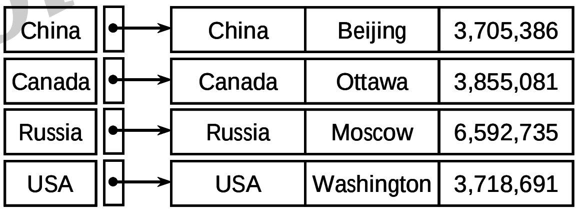
   Primary Index are two types:

   (i) Dense Index

   (ii) Sparse Index

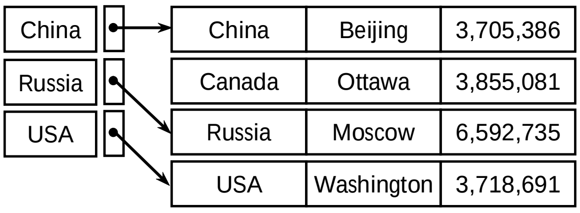
(i) Dense Index

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.


China
Canada
Russia
USA
China | Beijing | 3,705,386
Canada | Ottawa | 3,855,081
Russia | Moscow | 6,592,735
USA | Washington | 3,718,691

(ii) 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.


China
Russia
USA
China | Beijing | 3,705,386
Canada | Ottawa | 3,855,081
Russia | Moscow | 6,592,735
USA | Washington | 3,718,691

2. **Secondary Index:** Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.

3. **Clustering Index:** Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

Q24. List the properties of indexes.

Ans :

The properties of indexes are as follows,

1. Indexes enhance the performance level of the databases.
2. They can retrieve the records in a particular sequence order, thereby reducing the work load of the database manager.
3. They are capable of addressing the requirements of the application program.
4. They consume less time to locate the file records.
5. They eliminate the need to analyze each entry during the query execution i.e., they reduce the amount of file that is to be searched.
6. They increase the speed of accessing the data records.
7. Indexes can be added or removed by the database designers without modifying the application logic, due to which the maintenance costs is decreased whenever the size of the database is increased.
8. They can perform binary search on variable-length file records.

Q25. What are the differences between primary index and secondary index?

(OR)

Differentiate between 'primary indexing' and 'secondary indexing'.

Ans :

Primary Index	Secondary Index
1. An index that contains a primary key is called a primary index.	1. An index that does not contain a primary key is called a secondary index.
2. It does not have any duplicates i.e., it have unique values in each record.	2. It may have duplicates.
3. It consumes less storage space.	3. It consumes more storage space.
4. The time required to search a particular record is less.	4. The time required to search a particular record is more.
5. An index entry is created for each block.	5. An index entry is created for each record present within the data file.
6. Primary index can also be referred to as sparse index.	6. Secondary index can also be referred to as dense index.
7. Block anchors (the first record in each data block) can be used.	7. Block anchors cannot be used.
8. The file records are physically ordered based on the primary key field.	8. The file records are not physically ordered due to the absence of the primary key field.

Q26. Differentiate between sparse and dense indices.

Ans :

	Sparse Indice		Dense Indice
1.	Index entry is available for only few of the search key values.	1.	Index entry is available for every search key value in the file.
2.	It is applicable only when the relation is present in sorted order of the search key.	2.	It is applicable in either case
3.	It is complex.	3.	It is simple.
4.	It consumes more time.	4.	It consumes less time.
5.	It locates the records indirectly.	5.	It locates the records directly.

Q27. Explain briefly about tree based indexing.

Ans :

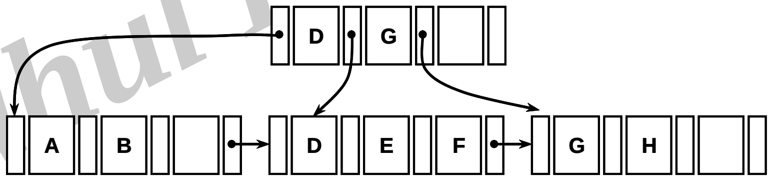
TREE STRUCTURES

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 **no** where **n** is fixed for every B^+ tree.



Internal nodes

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

Leaf nodes

- Leaf nodes contain at least [n/2] record pointers and [n/2] 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.

B^+ Tree Insertion

- B^+ trees are filled from bottom and each entry is done at the leaf node.
- If a leaf node overflows -
  - o Split node into two parts.
  - o Partition at $i = [(m+1)_{/2}]$.
  - o First i entries are stored in one node.
  - o Rest of the entries (i+1 onwards) are moved to a new node.
  - o i^{th} key is duplicated at the parent of the leaf.

- If a non-leaf node overflows -
  - o Split node into two parts.
  - o Partition the node at $i = [(m+1)_{/2}]$.
  - o Entries up to i are kept in one node.
  - o Rest of the entries are moved to a new node.

B^+ Tree Deletion

- B^+ tree entries are deleted at the leaf nodes.
- The target entry is searched and deleted.
  - o If it is an internal node, delete and replace with the entry from the left position.
- After deletion, underflow is tested,
  - o If underflow occurs, distribute the entries from the nodes left to it.
- If distribution is not possible from left, then
  - o Distribute from the nodes right to it.
- If distribution is not possible from left or from right, then
  - o Merge the node with left and right to it.

2.11 Multikey File Organization

2.11.1 Need for Multiple Access Paths

Q28. Explain multi-key file organization and discuss the need for multiple access paths.

Ans : **(Imp.)**

Unlike single key file organization, the multi-key file organization support the access of file records using more than one key. It has different approaches in which most of them follow building indexes. It allows a single data file to support multiple access path i.e., single data file can be accessed using different keys. For example, in a banking system of different users such as loan officer, branch manager and so on may require to access the same data in different ways and according to their requirements. The branch manager may require to access the data record with the key values such as BRANCH and TYPE. A loan officer may require to access the data record with the key value OVERDRAW_LIMIT, similarly the other workers want to access the data according to their requirements. Hence, they need to use different access keys. They submit different types of queries based on their required key values. Hence, to maintain such queries separate files must be maintained i.e., each file must serve a single request. Example, a separate sequential account file must serve the record, ordered by the key value "ID". Similarly, a separate relative account file must serve the records ordered by the relative keys NAME and TYPE.

Multiple access paths are needed in relational databases to make DBMS flexible and user friendly. The major need of providing multiple access paths is to allow the users to refer identical data with distinct key values and in distinct format.

Example

In a bank database, different users such as accountants, users, managers, loan officers need to access the database in different formats. That is, an accountant might want to access the database to update the customer accounts, users might want to access the database to know their current status. Managers want to access the same database to know the most dominating customers etc.

2.11.2 Multi list file organization, Inverted file organisation

Q29. Explain briefly about multilisted and inverted file organization.

Ans : **(Imp.)**

There are numerous methods that have been used to execute multi-key file Organisation. Most of these methods are based on building indexes to provide direct access by the key value. Two of the commonest methods for this Organisation are:

1. Multi-list file Organisation
2. Inverted file Organisation

1. Multi-list file Organisation

Multi-list file organisation is a multi-index linked file organisation. A linked file organisation is a logical organisation where physical ordering of records is not of concern. In linked organisation the series of records is governed by the links that verify the next record in series. Linking of records can be unordered but such a linking is very costly for searching of information from a file. Thus, it may be a good idea to link records in the order of increasing primary key. This will facilitate deletion and insertion algorithms. Also this really helps the search performance. In addition to making order during linking, search by a file can be further facilitated by producing primary and secondary indexes. All these ideas are supported in the multi-list file organisation. Let us describe these concepts further with the help of an example.

Consider the employee data as given in Figure. The record numbers are given as alphabets for better explanation. Suppose that the Empid is the key field of the data records. Let us describe the Multi-list file organisation for the data file.

Record Number	Empid	Name	Job	Qualification	Gender	City	Married/ Single	Salary
A	800	Jain	Software Engineer	B. Tech.	Male	New Delhi	Single	15,000/-
B	500	Inder	Software Manager	B. Tech.	Female	New Delhi	Married	18,000/-
C	900	Rashi	Software Manager	MCA	Female	Mumbai	Single	16,000/-
D	700	Gurpreet	Software Engineer	B. Tech.	Male	Mumbai	Married	12,000/-
E	600	Meena	Software Manager	MCA	Female	Mumbai	Single	13,000/-

Fig.. : Sample data for Employee file

Since, the primary key of the file is Empid, thus the linked order of records should be defined as B (500), E(600), D(700), A(800), C(900). Though, as the file size will grow the search performance of the file would deteriorate. Therefore, we can make a primary index on the file (please note that in this file the records are in the logical series and tied together using links and not physical placement, thus, the primary index will be a linked index file rather than block indexes).

2. Inverted File Organization

Conceptually, inverted files are similar to multilists. The difference is that while in multilists records with the same key value are linked together with link information being kept in individual records, in the case of inverted files this link information is kept in the index itself. Figure 27 shows the indexes for the file of figure 24. A slightly different strategy has been used in the E# and salary indexes than was used in figure 26, though the same strategy could have been used here too.

To simplify further discussion, we shall assume that the index for every key is dense and contains a value entry for each distinct value in the file. Since the index entries are variable length (the number of records with the same key value is variable), index maintenance becomes more complex than for multilist. However, several benefits accrue from this scheme. Boolean queries require only one access per record satisfying the query (plus some accesses to process the indexes). Queries of the type K1 = XX and K2 = XY. These two lists are then merged to obtain a list of all records satisfying the query. K1 = XX and K2 = XY can be handled similarly by intersecting the two lists. K1 =.not. XX can be handled by maintaining a universal list, U, with the addresses of all records. Then, K1 = .not. XX is just the difference between U and the list for K1 = Y-X. Any complex Boolean query may be handled in this way. The retrieval works in two steps. In the first step, the indexes are processed to obtain a list of records satisfying the query and in the second, these records are retrieved using this list. The number of disk accesses needed is equal to the number of records being retrieved plus the number to process the indexes.

Inverted files represent one extreme of file organization in which only the index structures are important. The records themselves may be stored in any way (sequentially ordered by primary key, random, linked ordered by primary key etc.).

E# index

510	
620	
750	
800	
950	

Occupation Index

Analyst	B,C
Programmer	A,D,E

Sex Index

Female	B,C,D
Male	A,E

Salary Index

9,000	E
10,000	A
12,000	C, D
15,000	B

Fig. : Indexes for fully inverted file

Inverted files may also result in space saving compared with other file structures when record retrieval does not require retrieval of key fields. In this case, the key fields may be deleted from the records. In the case of multilist structures, this deletion of key fields is possible only with significant loss in system retrieval performance. Insertion and deletion of records requires only the ability to insert and delete within indexes.

Short Question and Answers

1. Define Referential Integrity

Ans :

This is the concept of foreign keys. The rule states that the foreign key value can be in two states. The first state is that the foreign key value would refer to a primary key value of another table, or it can be null. Being null could simply mean that there are no relationships, or that the relationship is unknown.

2. Define Foreign Key

Ans :

A Foreign Key is a key used to link two tables together.

A Foreign Key is a field (or collection of fields) in one table that refers to the Primary Key in another table.

The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

Look at the following two tables :

"Persons" table :

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

"Orders" table :

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

3. Define Data Redundancy

Ans :

Data redundancy is a condition created within a database or data storage technology in which the same piece of data is held in two separate places.

This can mean two different fields within a single database, or two different spots in multiple software environments or platforms. Whenever data is repeated, this basically constitutes data redundancy. This can occur by accident, but is also done deliberately for backup and recovery purposes.

4. What is Functional Dependency

Ans :

Functional Dependency is when one attribute determines another attribute in a DBMS system. Functional Dependency plays a vital role to find the difference between good and bad database design.

Example :

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc.

By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

A functional dependency is denoted by an arrow $\rightarrow$

The functional dependency of X on Y is represented by X $\rightarrow$ Y

5. Define MVD

Ans :

Multivalued dependency occurs in the situation where there are multiple independent multivalued attributes in a single table. A multivalued dependency is a complete constraint between two sets of attributes in a relation. It requires that certain tuples be present in a relation.

Example :

Car_model	Maf_year	Color
H001	2017	Metallic
H001	2017	Green
H005	2018	Metallic
H005	2018	Blue
H010	2015	Metallic
H033	2012	Gray

In this example, maf_year and color are independent of each other but dependent on car_model.

In this example, these two columns are said to be multivalue dependent on car_model.

This dependence can be represented like this:

car_model -> maf_year

car_model-> colour

6. Write the advantages of Normalization.

Ans :

Here we can see why normalization is an attractive prospect in RDBMS concepts.

1) A smaller database can be maintained as normalization eliminates the duplicate data. Overall size of the database is reduced as a result.

2) Better performance is ensured which can be linked to the above point. As databases become lesser in size, the passes through the data becomes faster and shorter thereby improving response time and speed.

3) Narrower tables are possible as normalized tables will be fine-tuned and will have lesser columns which allows for more data records per page.

4) Fewer indexes per table ensures faster maintenance tasks (index rebuilds).

5) Also realizes the option of joining only the tables that are needed.

7. Define File Organization

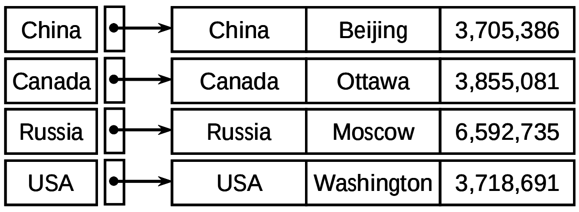
Ans :

- The File is a collection of records. Using the primary key, we can access the records. The type and frequency of access can be determined by the type of file organization which was used for a given set of records.
- File organization is a logical relationship among various records. This method defines how file records are mapped onto disk blocks.
- File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.
- The first approach to map the database to the file is to use the several files and store only one fixed length record in any given file. An alternative approach is to structure our files so that we can contain multiple lengths for records.
- Files of fixed length records are easier to implement than the files of variable length records.

8. What is Dense Index.

Ans :

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.



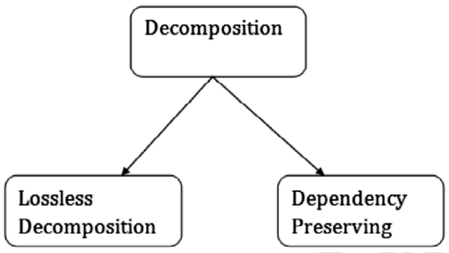
9. Define Decomposition. Explain the types of decomposition.

Ans :

Meaning

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

Types of Decomposition



1. Lossless Decomposition

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

2. Dependency Preservation

- 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 relation R1 and R2, then the dependencies of R either must be a part of R1 or R2 or must be derivable from the combination of functional dependencies of R1 and R2.

10. What is Normalization?

Ans :

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anamolies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

Choose the Correct Answers

1. A Relation is in 1NF if it does not contains [b]
   (a) Determinants (b) Repeating Groups
   (c) Null values in primary key (d) None
2. A Functional Dependency is a relation between [d]
   (a) Tables (b) Relations
   (c) Rows (d) Attributes
3. If an attribute A determines B. then. [a]
   (a) A is determinant (b) B is determinant
   (c) A is dependent (d) None
4. A table can have only one _______ key. [b]
   (a) Candidate (b) Primary
   (c) Forign (d) Composite
5. What is RDBMS terminology for a Table. [b]
   (a) Attribute (b) Relation
   (c) Tuple (d) Domain
6. Describe the property of De-composition. [d]
   (a) Attribute preservation (b) Loss-less-join
   (c) Dependency preservation (d) All
7. A _______ is the smallest unit of data transfer between the hard disk and the processor of the computer. [a]
   (a) Block (b) Drive
   (c) Disk (d) Address
8. _______ is a way of arranging the records in a file when the file is stored on the disk. [c]
   (a) Decomposition (b) Normalization
   (c) File organization (d) Genralization
9. With _______ sequential file organization Random access is possible? [a]
   (a) Index (b) Harshed file
   (c) Duster (d) None
10. A Relation which is in 3NF is almost always in [b]
   (a) UNF (b) BCNF
   (c) 2NF (d) Domain

Fill in the blanks

1. Candidate keyh must be _______ and cannot be _______.
2. An attribute is a _______ if its value is determined by another attribute.
3. _______ is storing the same data item in more one place.
4. _______ is a process of splitting a relation into its projections that will not be disjoint.
5. _______ is a file whose records can be accessed on the order of their appearance in the file.
6. Index is splitted into _______ Index.
7. BST stands for _______.
8. Central Data Defination and Data control facility is known as _______.
9. _______ Developed the process of Normalization.
10. Heap files is also called _______.

ANSWERS

1. Unique, Null
2. Functionally dependent
3. Redundancy
4. Decomposition
5. Sequential File
6. Primary, Cluster, Secondary
7. Binary Search Tree
8. Data Dictionary
9. Dr. E.F. Codd
10. Un-Ordered files

One Mark Answers

1. Entity Intigity constraint.

Ans :

It states that no primary key value can be null. This is because the primary key is used to identify individual tuple in the Relation. So we will not be able to identify the records uniquely containing null values for the primary key attributes.

2. Functional Dependency.

Ans :

"An attribute is functionally dependent if its value is determined by another attribute" that is if one data item is known, then it is possible to determine the other data value.

3. File Organization.

Ans :

A file organization is a technique to organize data in the secondary memory.

4. Give any 3 Drawbacks of Redundency Problem.

Ans :

- Entering same data more than one during data insertion.
- Modifing data in more than one place.
- Deleting data from more than one place.

5. Attribute Preservation.

Ans :

This is a simple requirement that involves preserving all the attributes that were there in the relation that is being Decomposed.

6. What is the relationship between files and indexes?

Ans :

A file is a collection or sequence of records that can be built and destroyed whereas, an index is list of keys or keywords, that form a disk based data structure. Allocation of an index onto a file is done in order to speed up the searching and retrieval of records that satisfy search conditions on the search key fields of the index.

7. What is the search key for an index?

Ans :

The fields stored in an index which allow efficient searching and retrieval of all the records that satisfy search conditions is called search key for an index.

8. What is hard disk?

Ans :

Hard disk is the primary storage unit of the computer system. It is also known as hard disk or fixed disk. This disk comprises of stack of disk platters. These platters are made of aluminium alloy that have a magnetic material coating. It also consists of protective layer, that protect the disk from being crashed.

UNIT III

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.

3.1 Structured Query Language (SQL)

Q1. Give a Basic introduction about SQL?

(OR)

Define SQL.

Ans : **(Imp.)**

Meaning

SQL stands for Structured Query Language.

SQL is a language that enables you to work with a database. Using SQL, you can insert records, update records, and delete records. You can also create new database objects such as databases and tables. And you can drop (delete) them.

More advanced features include creating stored procedures (self contained scripts), views (pre-made queries), and setting permissions on database objects (such as tables, stored procedures, and views).

Although SQL is an ANSI (American National Standards Institute) standard, there are many different versions of SQL. Different database vendors have their own variations of the language.

Having said this, to be in compliance with the ANSI standard, they need to at least support the major commands such as DELETE, INSERT, UPDATE, WHERE etc. Also, you will find that many vendors have their own extensions to the language — features that are only supported in their database system.

Furthermore, transact-SQL is an extension to the ANSI standard and provides extra functionality.

Using SQL

To run the SQL queries in this tutorial, you will need a database management system such as MySQL, Oracle, Microsoft Access, SQL Server, etc.

If you need to build a website with a database providing the content, you will generally need knowledge of the following:

- A server side scripting language (i.e. ColdFusion, PHP, ASP/.NET)
- A database query language (eg, SQL)
- A client side markup language and style sheets (eg, HTML/CSS)

Although SQL can be quite involved, you can achieve a lot with a handful of SQL statements. When using SQL on a website, you will often find yourself either selecting a record, inserting a record, updating a record, or deleting a record. Fortunately, SQL has commands for performing each of these actions.

In a simple manner, SQL is a non-procedural, English like language that processes data in group of records rather than one record at a time. Few basic functions of SQL listed below

- Storing of Data
- Modifying of data
- Retrieving of data
- Inserting of data
- Deleting of data
- Creating tables and other database objects.

Q2. List the features of SQL.

Ans :

- SQL is a non-procedural and English-like language containing statements in English sentences. Hence SQL statements are easily understandable.
- It minimizes the time required in creating and maintaining the database systems.
- It is a portable language and can be executed on various computer systems.
- It gives quick solutions to complex queries.
- It enables communication between clients and servers over a network.
- It allows users to store data on adhoc bases.
- It enables programmers to w rite applications for accessing database.

Q3. List and explain basic data types supported by SQL ?

Ans :

The data type of a column defines what value the column can hold: integer, character, money, date and time, binary, and so on.

SQL Data Types

Each column in a database table is required to have a name and a data type.

An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

MySQL Data Types

In MySQL there are three main data types:

1. Text data type
2. Number data type
3. Date data type

1. Text Data Types

Data type	**Description**
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters

BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. **Note:** The values are sorted in the order you enter them.You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

2. Number Data Types

Data type	**Description**
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

*The integer types have an extra option called UNSIGNED. Normally, the integer goes from an negative to positive value. Adding the UNSIGNED attribute will move that range up so it starts at zero instead of a negative number.

3. Date Data Types

Data type	Description
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

*Even if DATETIME and TIMESTAMP return the same format, they work very differently. In an INSERT or UPDATE query, the TIMESTAMP automatically set itself to the current date and time. TIMESTAMP also accepts various formats, like YYYYMMDDHHMISS, YYMMDDHHMISS, YYYYMMDD, or YYMMDD

Q4. Explain various types of operators are used in SQL.

Ans :

An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations. These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

1. Arithmetic operators
2. Comparison operators
3. Logical operators

1. Arithmetic Operators

Assume 'variable a' holds 10 and 'variable b' holds 20, then

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator.	a + b will give 30
- (Subtraction)	Subtracts right hand operand from left hand operand.	a - b will give -10
* (Multiplication)	Multiplies values on either side of the operator.	a * b will give 200
/ (Division)	Divides left hand operand by right hand operand.	b / a will give 2
% (Modulus)	Divides left hand operand by right hand operand and returns remainder.	b % a will give 0

2. Comparison Operators

Assume 'variable a' holds 10 and 'variable b' holds 20, then

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(a = b) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a != b) is true.
< >	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a < > b) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
> =	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
< =	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.	(a !< b) is false.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.	(a !> b) is true.

3. Logical Operators

Here is a list of all the logical operators available in SQL.

Sr.No.	Operator & Description
1	**ALL** The ALL operator is used to compare a value to all values in another value set.
2	**AND** The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.

3	**ANY** The ANY operator is used to compare a value to any applicable value in the list as per the condition.
4	**BETWEEN** The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
5	**EXISTS** The EXISTS operator is used to search for the presence of a row in a specified table that meets a certain criterion.
6	**IN** The IN operator is used to compare a value to a list of literal values that have been specified.
7	**LIKE** The LIKE operator is used to compare a value to similar values using wildcard operators.
8	**NOT** The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. **This is a negate operator.**
9	**OR** The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
10	**IS NULL** The NULL operator is used to compare a value with a NULL value.
11	**UNIQUE** The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

Q5. Explain about Expressions in SQL ?

Ans :

Meaning

An expression is a combination of one or more values, operators and SQL functions that evaluate to a value. These SQL EXPRESSIONs are like formulae and they are written in query language. You can also use them to query the database for a specific set of data.

Syntax

Consider the basic syntax of the SELECT statement as follows -

```
SELECT column1, column2, columnN
FROM table_name
WHERE [CONDITION|EXPRESSION];
```

There are different types of SQL expressions, which are mentioned below “

1. Boolean Expression
2. Numeric Expression
3. Date Expression

Let us now discuss each of these in detail.

1. Boolean Expressions

SQL Boolean Expressions fetch the data based on matching a single value. Following is the syntax-

```
SELECT column1, column2, columnN
FROM table_name
WHERE SINGLE VALUE MATCHING EXPRESSION;
```

Consider the CUSTOMERS table having the following records -

SQL> SELECT * FROM CUSTOMERS;

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

7 rows inset(0.00 sec)

The following table is a simple example showing the usage of various SQL Boolean Expressions -

SQL> SELECT * FROM CUSTOMERS WHERE SALARY =10000;

ID	NAME	AGE	ADDRESS	SALARY
7	Muffy	24	Indore	10000.00

1 row inset(0.00 sec)

2. Numeric Expression

These expressions are used to perform any mathematical operation in any query. Following is the syntax -

```
SELECT numerical_expression as  OPERATION_NAME
[FROM table_name
WHERE CONDITION] ;
```

Here, the numerical_expression is used for a mathematical expression or any formula. Following is a simple example showing the usage of SQL Numeric Expressions -

SQL> SELECT (15+6) AS ADDITION

ADDITION
21

1 row inset(0.00 sec)

There are several built-in functions like avg(), sum(), count(), etc., to perform what is known as the aggregate data calculations against a table or a specific table column.

```
SQL > SELECT COUNT(*) AS "RECORDS" FROM CUSTOMERS;
```

RECORDS
7

1 row inset(0.00 sec)

3. Data Expression

Date Expressions return current system date and time values "

```
SQL >  SELECT CURRENT_TIMESTAMP;
```

Current_Timestamp
2009-11-1206:40:23

1 row inset(0.00 sec)

Another date expression is as shown below -

```
SQL >  SELECT  GETDATE();;
```

GETDATE
2009-10-2212:07:18.140

1 row inset(0.00 sec)

3.2 SQL Commands

Q6. What is the use of SQL ? Catagorize SQL commands ?

Ans : **(Imp.)**

Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like Create, Drop, Insert etc. to carry out the required tasks.

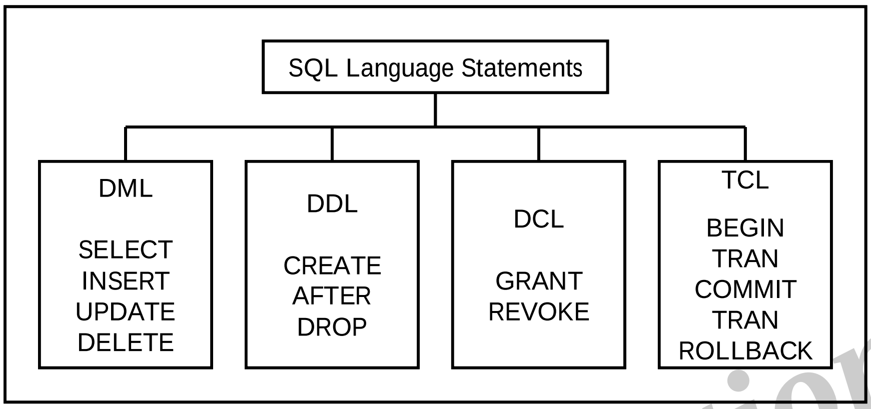
These SQL commands are mainly categorized into four categories as discussed below:

1. **DDL(Data Definition Language) :** DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database.

   Examples of DDL commands:

   - **CREATE:** Is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
   - **DROP:** Is used to delete objects from the database.
   - **ALTER:** Is used to alter the structure of the database.

- **TRUNCATE**: Is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT:** Is used to add comments to the data dictionary.
- **RENAME:** Is used to rename an object existing in the database.



2. **DML(Data Manipulation Language) :** The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

Examples of DML:

- **SELECT:** Is used to retrieve data from the a database.
- **INSERT:** Is used to insert data into a table.
- **UPDATE:** Is used to update existing data within a table.
- **DELETE:** Is used to delete records from a database table.

3. **DCL(Data Control Language) :** DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

Examples of DCL commands:

- **GRANT:** It gives user's access privileges to database.
- **REVOKE:** It withdraw user's access privileges given by using the GRANT command.

4. **TCL(transaction Control Language) :** TCL commands deals with the transaction within the database.

Examples of TCL commands:

- **COMMIT:** It commits a Transaction.
- **ROLLBACK:** It rollbacks a transaction in case of any error occurs.
- **SAVEPOINT:** It sets a savepoint within a transaction.
- **SET TRANSACTION:** It specify characteristics for the transaction.

3.3 Data Definition Language - DDL

Q7. Describe in detail about DDL commands in SQL ?

(OR)

Explain DDL Commands.

Ans : **(July-21, Oct.-20, June -19, June-18, Imp.)**

Data definition language commands are used to create, Modify and delete the structure of the object in the database. The syntax of the DDL command definitely includes a table keyword after the command name.

All DDL command are given below;

1. Create
2. Describe
3. Alter
4. Drop
5. Truncate and
6. Remove

1. CREATE

Creating a basic table involves naming the table and defining its columns and each column's data type.

The SQL **CREATE TABLE** statement is used to create a new table.

Syntax

The basic syntax of the CREATE TABLE statement is as follows -

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
 PRIMARY KEY( one or more columns )
);
```

CREATE TABLE is the keyword telling the database system what you want to do. In this case, you want to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement.

Then in brackets comes the list defining each column in the table and what sort of data type it is. The syntax becomes clearer with the following example.

A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement.

Example

```
CREATE TABLE Employee
(
Emp id number(5),
name varchar(20),
salary number(10),
);
```

2. Describe

As the name suggests, DESCRIBE is used to describe something. Since in database we have tables, that's why we use **DESCRIBE** or **DESC**(both are same) command to describe the **structure** of a table.

Syntax:

```
DESCRIBE employee;
          OR
DESC employee;
```

Then for the above we will get sample output as

```
Empid    varchar(5)
Name    varchar(20)
Salary    number(10)
```

3. ALTER

The SQL **ALTER TABLE** command is used to add, delete or modify columns in an existing table. You should also use the ALTER TABLE command to add and drop various constraints on an existing table.

Syntax

The basic syntax of an ALTER TABLE command to add a **New Column** in an existing table is as follows.

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

The basic syntax of an ALTER TABLE command to **DROP COLUMN** in an existing table is as follows.

```
ALTER TABLE table_name DROP COLUMN column_name;
```

The basic syntax of an ALTER TABLE command to change the **DATA TYPE** of a column in a table is as follows.

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

The basic syntax of an ALTER TABLE command to add a **NOT NULL** constraint to a column in a table is as follows.

```
ALTER TABLE table_name MODIFY column_name datatype NOT NULL;
```

The basic syntax of ALTER TABLE to **ADD UNIQUE CONSTRAINT** to a table is as follows.

ALTER TABLE table_name

ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);

The basic syntax of an ALTER TABLE command to **ADD CHECK CONSTRAINT** to a table is as follows.

ALTER TABLE table_name

ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);

The basic syntax of an ALTER TABLE command to **ADD PRIMARY KEY** constraint to a table is as follows.

ALTER TABLE table_name

ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);

The basic syntax of an ALTER TABLE command to **DROP CONSTRAINT** from a table is as follows.

ALTER TABLE table_name

DROP CONSTRAINT MyUniqueConstraint;

If you're using MySQL, the code is as follows "

ALTER TABLE table_name

DROP INDEX MyUniqueConstraint;

The basic syntax of an ALTER TABLE command to **DROP PRIMARY KEY** constraint from a table is as follows.

ALTER TABLE table_name

DROP CONSTRAINT MyPrimaryKey;

If you're using MySQL, the code is as follows "

ALTER TABLE table_name

DROP PRIMARY KEY;

Example

Consider the CUSTOMERS table having the following records -

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is the example to ADD a **New Column** to an existing table “

ALTER TABLE CUSTOMERS ADD SEX char(1);

Now, the CUSTOMERS table is changed and following would be output from the SELECT statement.

ID	NAME	AGE	ADDRESS	SALARY	SEX
1	Ramesh	32	Ahmedabad	2000.00	NULL
2	Ramesh	25	Delhi	1500.00	NULL
3	kaushik	23	Kota	2000.00	NULL
4	kaushik	25	Mumbai	6500.00	NULL
5	Hardik	27	Bhopal	8500.00	NULL
6	Komal	22	MP	4500.00	NULL
7	Muffy	24	Indore	10000.00	NULL

Following is the example to DROP sex column from the existing table.

ALTER TABLE CUSTOMERS DROP SEX;

Now, the CUSTOMERS table is changed and following would be the output from the SELECT statement.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Ramesh	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	kaushik	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

4. DROP

The SQL **DROP TABLE** statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.

NOTE - You should be very careful while using this command because once a table is deleted then all the information available in that table will also be lost forever.

Syntax

The basic syntax of this DROP TABLE statement is as follows “

DROP TABLE table_name;

Example

Let us first verify the CUSTOMERS table and then we will delete it from the database as shown below -

```
SQL >  DESC  CUSTOMERS
```

Field Type	Null Key	Default	Extra
ID	int(11)	NO	PRI
NAME	varchar(20)	NO	—
AGE	int(11)	NO	—
ADDRESS	char(25)	YES	NULL
SALARY	decimal(18,2)	YES	NULL

```
5 rows inset(0.00 sec)
```

This means that the CUSTOMERS table is available in the database, so let us now drop it as shown below.

```
SQL > DROP TABLE CUSTOMERS;
Query OK,0 rows affected (0.01 sec)
```

Now, if you would try the DESC command, then you will get the following error "

```
SQL > DESC CUSTOMERS;
OUTPUT:
ERROR 1146(42S02):Table'TEST.CUSTOMERS' doesn't exist
```

5. TRUNCATE

The SQL **TRUNCATE TABLE** command is used to delete complete data from an existing table.

You can also use DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.

Syntax

The basic syntax of a **TRUNCATE TABLE** command is as follows.

```
TRUNCATE TABLE  table_name;
```

Example

Consider a CUSTOMERS table having the following records -

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is the example of a Truncate command.

```
SQL > TRUNCATE TABLE CUSTOMERS;
```

Now, the CUSTOMERS table is truncated and the output from SELECT statement will be as shown in the code block below “

```
SQL> SELECT * FROM CUSTOMERS;
OUTPUT:
Empty set (0.00 sec)
```

6. RENAME

SQL RENAME Statement

With RENAME statement you can rename a table.

Syntax for SQL RENAME is:

```
RENAME TABLE {tbl_name} TO {new_tbl_name};
```

Where {tbl_name} table that exists in the current database, and {new_tbl_name} is new table name.

```
ALTER TABLE {tbl_name} RENAME TO {new_tbl_name};
```

As Example

```
CREATE TABLE employees
( id NUMBER(6),
name VARCHAR(20)
);
INSERT INTO employees( id, name ) values( 1, ‘name 1’);
INSERT INTO employees( id, name ) values( 2, ‘name 2’);
INSERT INTO employees( id, name ) values( 3, ‘name 3’);

SELECT * FROM employees;
```

SELECT Output:

id	name
1	name 1
2	name 2
3	name 3

```
RENAME TABLE employees TO employees_new;
SELECT * FROM employees_new;
```

SELECT Output:

id	name
1	name 1
2	name 2
3	name 3

3.4 Data Manipulation Language - DML

Q8. Explain in detail about DML commands ?

Ans : **(Oct.-19, June-18, Imp.)**

DML(Data Manipulation Language) : The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

Examples of DML:

1. **INSERT –** is used to insert data into a table.
2. **SELECT –** is used to retrieve data from the a database.
3. **UPDATE –** is used to update existing data within a table.
4. **DELETE –** is used to delete records from a database table.

1. INSERT

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax

There are two basic syntaxes of the INSERT INTO statement which are shown below.

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);
```

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

The **SQL INSERT INTO** syntax will be as follows -

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

Example

The following statements would create six records in the CUSTOMERS table.

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1,'Ramesh',32,'Ahmedabad',2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2,'Khilan',25,'Delhi',1500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3,'kaushik',23,'Kota',2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4,'Chaitali',25,'Mumbai',6500.00);
```

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5,'Hardik',27,'Bhopal',8500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6,'Komal',22,'MP',4500.00);
```

You can create a record in the CUSTOMERS table by using the second syntax as shown below.

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (7, 'Muffy', 24, 'Indore', 10000.00 );
```

All the above statements would produce the following records in the CUSTOMERS table as shown below.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

2. SELECT

The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

Syntax

The basic syntax of the SELECT statement is as follows "

```
SELECT column1, column2, columnN FROM table_name;
```

Here, column1, column2... are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax.

```
SELECT * FROM table_name;
```

Example

Consider the CUSTOMERS table having the following records -

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code is an example, which would fetch the ID, Name and Salary fields of the customers available in CUSTOMERS table.

```
SQL > SELECT ID, NAME, SALARY FROM CUSTOMERS;
```

This would produce the following result -

ID	NAME	SALARY
1	Ramesh	2000.00
2	Khilan	1500.00
3	kaushik	2000.00
4	Chaitali	6500.00
5	Hardik	8500.00
6	Komal	4500.00
7	Muffy	10000.00

If you want to fetch all the fields of the CUSTOMERS table, then you should use the following query.

```
SQL > SELECT * FROM CUSTOMERS;
```

This would produce the result as shown below.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

3. UPDATE

The SQL **UPDATE** Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

Syntax

The basic syntax of the UPDATE query with a WHERE clause is as follows -

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

You can combine N number of conditions using the AND or the OR operators.

Example

Consider the CUSTOMERS table having the following records “

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following query will update the ADDRESS for a customer whose ID number is 6 in the table.

```
SQL> UPDATE CUSTOMERS
SET ADDRESS = 'Pune'
WHERE ID = 6;
```

Now, the CUSTOMERS table would have the following records -

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Pune	4500.00
7	Muffy	24	Indore	10000.00

If you want to modify all the ADDRESS and the SALARY column values in the CUSTOMERS table, you do not need to use the WHERE clause as the UPDATE query would be enough as shown in the following code block.

```
SQL> UPDATE CUSTOMERS
SET ADDRESS = 'Pune', SALARY = 1000.00;
```

Now, CUSTOMERS table would have the following records “

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Pune	1000.00
2	Khilan	25	Pune	1000.00
3	kaushik	23	Pune	1000.00
4	Chaitali	25	Pune	1000.00
5	Hardik	27	Pune	1000.00
6	Komal	22	Pune	1000.00
7	Muffy	24	Pune	1000.00

4. DELETE

The SQL DELETE Query is used to delete the existing records from a table.

You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

Syntax

The basic syntax of the DELETE query with the WHERE clause is as follows “

```
DELETE FROM table_name
WHERE [condition];
```

You can combine N number of conditions using AND or OR operators.

Example

Consider the CUSTOMERS table having the following records “

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code has a query, which will DELETE a customer, whose ID is 6.

```
SQL> DELETE FROM CUSTOMERS
WHERE ID = 6;
```

Now, the CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

If you want to DELETE all the records from the CUSTOMERS table, you do not need to use the WHERE clause and the DELETE query would be as follows -

```
SQL> DELETE FROM CUSTOMERS;
```

Now, the CUSTOMERS table would not have any record.

3.5 Data Control Language - DCL

Q9. Explain in detail about DCL commands?

Ans : **(Imp.)**

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,

1. **GRANT:** It used to provide any user access privileges or other privileges for the database.
2. **REVOKE:** It used to take back permissions from any user.

1. GRANT

- GRANT command gives user's access privileges to the database.
- This command allows specified users to perform specific tasks.

Syntax:

```
GRANT <privilege list>
ON <relation name or view name>
TO <user/role list>;
```

Example : GRANT Command

```
GRANT ALL ON employee
TO ABC;
[WITH GRANT OPTION]
```

In the above example, user 'ABC' has been given permission to view and modify the records in the 'employee' table.

2. REVOKE

- REVOKE command is used to cancel previously granted or denied permissions.
- This command withdraw access privileges given with the GRANT command.
- It takes back permissions from user.

Syntax:

```
REVOKE <privilege list>
ON <relation name or view name>
FROM <user name>;
```

Example : REVOKE Command

```
REVOKE UPDATE
ON employee
FROM ABC;
```

Q10. Explain in detail about Grant , Revoke commands in DCL ?

Ans :

1. SQL GRANT REVOKE Commands

DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. Only Database Administrator's or owner's of the database object can provide/remove privileges on a database object.

SQL GRANT Command

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

The Syntax for the GRANT command is:

```
GRANT privilege_name
ON object_name
TO {user_name |PUBLIC |role_name}
[WITH GRANT OPTION];
```

- **privilege_name** is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
- **object_name** is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
- **user_name** is the name of the user to whom an access right is being granted.
- **user_name** is the name of the user to whom an access right is being granted.

- **PUBLIC** is used to grant access rights to all users.
- **ROLES** are a set of privileges grouped together.
- **WITH GRANT OPTION** - allows a user to grant access rights to other users.

For Example: GRANT SELECT ON employee TO user1; This command grants a SELECT permission on employee table to user1.You should use the WITH GRANT option carefully because for example if you GRANT SELECT privilege on employee table to user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on employee table to another user, such as user2 etc. Later, if you REVOKE the SELECT privilege on employee from user1, still user2 will have SELECT privilege on employee table.

2. SQL REVOKE Command:

The REVOKE command removes user access rights or privileges to the database objects.

The Syntax for the REVOKE command is:

```
REVOKE privilege_name
ON object_name
FROM {user_name |PUBLIC |role_name}
```

For Example: REVOKE SELECT ON employee FROM user1; This command will REVOKE a SELECT privilege on employee table from user1. When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one users, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.

Privileges and Roles:

Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

1. **System privileges** - This allows the user to CREATE, ALTER, or DROP database objects.
2. **Object privileges** - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Few CREATE system privileges are listed below:

System Privileges	Description
CREATE object	allows users to create the specified object in their own schema.
CREATE ANY object	allows users to create the specified object in any schema.

The above rules also apply for ALTER and DROP system privileges.

Few of the object privileges are listed below:

Object Privileges	Description
INSERT	allows users to insert rows into a table.
SELECT	allows users to select data from a database object.
UPDATE	allows user to update data in a table.
EXECUTE	allows user to execute a stored procedure or a function.

Roles: Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles, you can grant or revoke privileges to users, thereby automatically granting or revoking privileges. You can either create Roles or use the system roles pre-defined by oracle.

Some of the privileges granted to the system roles are as given below:

System Role	Privileges Granted to the Role
CONNECT	CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE SESSION etc.
RESOURCE	CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER etc. The primary usage of the RESOURCE role is to restrict access to database objects.
DBA	ALL SYSTEM PRIVILEGES

Creating Roles:

The Syntax to create a role is:

CREATE ROLE role_name

[IDENTIFIED BY password];

For Example: To create a role called "developer" with password as "pwd", the code will be as follows

CREATE ROLE testing

[IDENTIFIED BY pwd];

It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user. If a role is identified by a password, then, when you GRANT or REVOKE privileges to the role, you definitely have to identify it with the password.

We can GRANT or REVOKE privilege to a role as below.

For example: To grant CREATE TABLE privilege to a user by creating a testing role:

First, create a testing Role

```
CREATE ROLE testing
```

Second, grant a CREATE TABLE privilege to the ROLE testing. You can add more privileges to the ROLE.

```
GRANT CREATE TABLE TO testing;
```

Third, grant the role to a user.

```
GRANT testing TO user1;
```

To revoke a CREATE TABLE privilege from testing ROLE, you can write:

```
REVOKE CREATE TABLE FROM testing;
```

The Syntax to drop a role from the database is as below:

```
DROP ROLE role_name;
```

For example: To drop a role called developer, you can write:

```
DROP ROLE testing;
```

Q11. How Grant command is differ from Revoke.

Ans :

GRANT	REVOKE
i) Grant command allows a user to perform certain activities on the database.	i) Revoke command disallows a user to perform certain activities.
ii) It grants access privileges for database objects to other users.	ii) It revokes access privileges for database objects previously granted to other users.

Example:
GRANT privilege_name
ON object_name
TO
{
user_name|PUBLIC|role_name
}
[WITH GRANT OPTION];
	Example:
	REVOKE privilege_name
	ON object_name
	FROM
	{
	user_name|PUBLIC|role_name
	}

3.6 Transaction Control Language - TCL

Q12. Explain in detail about TCL - Commands.

Ans : **(Imp.)**

Introduction

- TCL stands for **Transaction Control Language.**

- This command is used to manage the changes made by DML statements.
- TCL allows the statements to be grouped together into logical transactions.

TCL commands are as follows:

1. COMMIT
2. SAVEPOINT
3. ROLLBACK
4. SET TRANSACTION

1. COMMIT

- **COMMIT command** saves all the work done.
- It ends the current transaction and makes permanent changes during the transaction.

Syntax:

commit;

2. SAVEPOINT

- **SAVEPOINT command** is used for saving all the current point in the processing of a transaction.
- It marks and saves the current point in the processing of a transaction.

Syntax:

SAVEPOINT <savepoint_name>

Example:

SAVEPOINT no_update;

- It is used to temporarily save a transaction, so that you can rollback to that point whenever necessary.

3. ROLLBACK

- ROLLBACK command restores database to original since the last COMMIT.
- It is used to restores the database to last committed state.

Syntax:

ROLLBACK TO SAVEPOINT <savepoint_name>;

Example:

ROLLBACK TO SAVEPOINT no_update;

4. SET TRANSACTION

- **SET TRANSACTION** is used for placing a name on a transaction.

Syntax:

SET TRANSACTION [Read Write | Read Only];

- You can specify a transaction to be read only or read write.
- This command is used to initiate a database transaction.

Q13. What are the Differences between ROLLBACK and COMMIT Commands?

Ans :

ROLLBACK	COMMIT
i) ROLLBACK command is used to undo the changes made by the DML commands.	i) The COMMIT command is used to save the modifications done to the database values by the DML commands.
ii) It rollbacks all the changes of the current transaction.	ii) It will make all the changes permanent that cannot be rolled back.
Syntax: DELETE FROM table_name ROLLBACK	**Syntax:** COMMIT;

3.7 Queries using - Order By, Where, Group By Nested Queries

Q14. Explain in detail about Order by clause in SQL ?

Ans : **(Imp.)**

The ORDER BY clause is used in a SELECT statement to sort results either in ascending or descending order. Oracle sorts query results in ascending order by default.

Syntax for using SQL ORDER BY clause to sort data is:

```
SELECT column-list
FROM table_name [WHERE condition]
[ORDER BY column1 [, column2, .. columnN] [DESC]];
```

database table "employee";

id	name	dept	age	salary	location
100	Ramesh	Electrical	24	25000	Bangalore
101	Hrithik	Electronics	28	35000	Bangalore
102	Harsha	Aeronautics	28	35000	Mysore
103	Soumya	Electronics	22	20000	Bangalore
104	Priya	InfoTech	25	30000	Mangalore

For Example: If you want to sort the employee table by salary of the employee, the sql query would be.

```
SELECT name, salary FROM employee ORDER BY salary;
```

The output would be like

name	salary
----------	----------
Soumya	20000
Ramesh	25000
Priya	30000
Hrithik	35000
Harsha	35000

The query first sorts the result according to name and then displays it.

You can also use more than one column in the ORDER BY clause.

If you want to sort the employee table by the name and salary, the query would be like,

SELECT name, salary FROM employee ORDER BY name, salary;

The output would be like:

name	salary
-------------	-------------
Soumya	20000
Ramesh	25000
Priya	30000
Harsha	35000
Hrithik	35000

NOTE: The columns specified in ORDER BY clause should be one of the columns selected in the SELECT column list.

You can represent the columns in the ORDER BY clause by specifying the position of a column in the SELECT list, instead of writing the column name.

The above query can also be written as given below,

SELECT name, salary FROM employee ORDER BY 1, 2;

By default, the ORDER BY Clause sorts data in ascending order. If you want to sort the data in descending order, you must explicitly specify it as shown below.

```
SELECT name, salary
FROM employee
ORDER BY name, salary DESC;
```

The above query sorts only the column 'salary' in descending order and the column 'name' by ascending order.

If you want to select both name and salary in descending order, the query would be as given below.

```
SELECT name, salary
FROM employee
ORDER BY name DESC, salary DESC;
```

Q15. How to use expressions in the Order-By Clause?

Ans :

Expressions in the ORDER BY clause of a SELECT statement.

For example: If you want to display employee name, current salary, and a 20% increase in the salary for only those employees for whom the percentage increase in salary is greater than 30000 and in descending order of the increased price, the SELECT statement can be written as shown below

```
SELECT name, salary, salary*1.2 AS new_salary
FROM employee
WHERE salary*1.2 > 30000
ORDER BY new_salary DESC;
```

The output for the above query is as follows.

name	salary	new_salary
----------	----------	-------------
Hrithik	35000	37000
Harsha	35000	37000
Priya	30000	36000

Q16. Explain in detail about where clause in SQL ?

Ans :

The WHERE Clause is used when you want to retrieve specific information from a table excluding other irrelevant data. For example, when you want to see the information about students in class 10th only then you do need the information about the students in other class. Retrieving information about all the students would increase the processing time for the query.

So SQL offers a feature called WHERE clause, which we can use to restrict the data that is retrieved. The condition you provide in the WHERE clause filters the rows retrieved from the table and gives you only those rows which you expected to see. WHERE clause can be used along with SELECT, DELETE, UPDATE statements.

Syntax of SQL WHERE Clause

WHERE {column or expression} comparison-operator value

Syntax for a WHERE clause with Select statement is:

SELECT *column_list* FROM *table-name*

WHERE *condition*;

- *column or expression* - Is the column of a table or a expression
- *comparison-operator* - operators like = < > etc.
- *value* - Any user value or a column name for comparison

For Example: To find the name of a student with id 100, the query would be like:

SELECT first_name, last_name FROM student_details

WHERE id = 100;

Comparison Operators and Logical Operators are used in WHERE Clause.

NOTE: Aliases defined for the columns in the SELECT statement cannot be used in the WHERE clause to set conditions. Only aliases created for tables can be used to reference the columns in the table.

Q17. How to use expressions in the where Clause?

Ans :

Expressions can also be used in the WHERE clause of the SELECT statement.

For example: Lets consider the employee table. If you want to display employee name, current salary, and a 20% increase in the salary for only those products where the percentage increase in salary is greater than 30000, the SELECT statement can be written as shown below

SELECT name, salary, salary*1.2 AS new_salary FROM employee

WHERE salary*1.2 > 30000;

Output:

name	salary	new_salary
-----------	----------	----------------
Hrithik	35000	37000
Harsha	35000	37000
Priya	30000	360000

NOTE: Aliases defined in the SELECT Statement can be used in WHERE Clause.

Q18. Explain in detail about Group by clause in SQL ?

Ans :

The SQL GROUP BY Clause is used along with the group functions to retrieve data grouped according to one or more columns.

For Example: If you want to know the total amount of salary spent on each department, the query would be:

SELECT dept, SUM (salary)

FROM employee

GROUP BY dept;

The output would be like:

dept	salary
----------------	--------------
Electrical	25000
Electronics	55000
Aeronautics	35000
InfoTech	30000

NOTE: The group by clause should contain all the columns in the select list expect those used along with the group functions.

```
SELECT location, dept, SUM (salary)
FROM employee
GROUP BY location, dept;
```

The output would be like:

location	dept	salary
-------------	---------------	-----------
Bangalore	Electrical	25000
Bangalore	Electronics	55000
Mysore	Aeronautics	35000
Mangalore	InfoTech	30000

Q19. Explain in detail about Having clause in SQL ?

Ans :

Having clause is used to filter data based on the group functions. This is similar to WHERE condition but is used with group functions. Group functions cannot be used in WHERE Clause but can be used in HAVING clause.

SQL HAVING Clause Example

If you want to select the department that has total salary paid for its employees more than 25000, the sql query would be like;

```
SELECT dept, SUM (salary)
FROM employee
GROUP BY dept
HAVING SUM (salary) > 25000
```

The output would be like:

dept	salary
-------------	-------------
Electronics	55000
Aeronautics	35000
InfoTech	30000

When WHERE, GROUP BY and HAVING clauses are used together in a SELECT statement, the WHERE clause is processed first, then the rows that are returned after the WHERE clause is executed are grouped based on the GROUP BY clause.

Finally, any conditions on the group functions in the HAVING clause are applied to the grouped rows before the final output is displayed.

Q20. Explain in detail about Nested Queries.

Ans : **(Imp.)**

A Subquery (or) Inner query(or) a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are a few rules that subqueries must follow -

➢ Subqueries must be enclosed within parentheses.

➢ A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.

- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A subquery cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

(i) Subqueries with the SELECT Statement

Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows “

```
SELECT column_name [, column_name ]
FROM   table1 [, table2 ]
WHERE  column_name OPERATOR
      (SELECT column_name [, column_name ]
      FROM table1 [, table2 ]
      [WHERE])
```

Example

Consider the CUSTOMERS table having the following records -

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now, let us check the following subquery with a SELECT statement.

```
SQL> SELECT *
  FROM CUSTOMERS
  WHERE ID IN (SELECT ID
      FROM CUSTOMERS
      WHERE SALARY >4500);
```

This would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

(ii) Subqueries with the INSERT Statement

Subqueries also can be used with INSERT statements. The INSERT statement uses the data returned from the subquery to insert into another table. The selected data in the subquery can be modified with any of the character, date or number functions.

The basic syntax is as follows.

```
INSERT INTO table_name [ (column1 [, column2 ]) ]
  SELECT [ *|column1 [, column2 ]
  FROM table1 [, table2 ]
  [ WHERE VALUE OPERATOR ]
```

Example

Consider a table CUSTOMERS_BKP with similar structure as CUSTOMERS table. Now to copy the complete CUSTOMERS table into the CUSTOMERS_BKP table, you can use the following syntax.

```
SQL > INSERT INTO CUSTOMERS_BKP
  SELECT * FROM CUSTOMERS
  WHERE ID IN (SELECT ID
  FROM CUSTOMERS) ;
```

(iii) Subqueries with the UPDATE Statement

The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

The basic syntax is as follows.

```
UPDATE table
SET column_name = new_value
[ WHERE OPERATOR [ VALUE ]
       (SELECT COLUMN_NAME
       FROM TABLE_NAME)
       [ WHERE) ]
```

Example

Assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table. The following example updates SALARY by 0.25 times in the CUSTOMERS table for all the customers whose AGE is greater than or equal to 27.

```
SQL > UPDATE CUSTOMERS
  SET SALARY = SALARY *0.25
  WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP
         WHERE AGE >=27);
```

This would impact two rows and finally CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	125.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	2125.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

(iv) Subqueries with the DELETE Statement

The subquery can be used in conjunction with the DELETE statement like with any other statements mentioned above.

The basic syntax is as follows.

```
DELETE FROM TABLE_NAME
[ WHERE OPERATOR [ VALUE ]
  (SELECT COLUMN_NAME
  FROM TABLE_NAME)
  [ WHERE) ]
```

Example

Assuming, we have a CUSTOMERS_BKP table available which is a backup of the CUSTOMERS table. The following example deletes the records from the CUSTOMERS table for all the customers whose AGE is greater than or equal to 27.

```
SQL > DELETE FROM CUSTOMERS
    WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP
         WHERE AGE >=27);
```

This would impact two rows and finally the CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

3.8 Joins

Q21. Explain in detail about joins in SQL ?

(OR)

Explain briefly about Joins concept.

Ans : **(Oct.-20, June-19, Imp.)**

SQL Joins are used to relate information in different tables. A Join condition is a part of the sql query that retrieves rows from two or more tables. A SQL Join condition is used in the SQL *WHERE Clause* of select, update, delete statements.

Joins in SQL

The **SQL Syntax** for joining two tables is:

```
SELECT col1, col2, col3...
FROM table_name1, table_name2
WHERE table_name1.col2 = table_name2.col1;
```

If a sql join condition is omitted or if it is invalid the join operation will result in a Cartesian product. The Cartesian product returns a number of rows equal to the product of all rows in all the tables being joined. For example, if the first table has 20 rows and the second table has 10 rows, the result will be 20 * 10, or 200 rows. This query takes a long time to execute.

Example

Lets use the below two tables to explain the sql join conditions.

Database table "product";

product_id	product_name	supplier_name	unit_price
100	Camera	Nikon	300
101	Television	Onida	100
102	Refrigerator	Vediocon	150
103	Ipod	Apple	75
104	Mobile	Nokia	50

Database table "order_items";

order_id	product_id	total_units	customer
5100	104	30	Infosys
5101	102	5	Satyam
5102	103	25	Wipro
5103	101	10	TCS

SQL Joins can be classified into Equi join and Non Equi join.

1. SQL Equi joins

It is a simple sql join condition which uses the equal sign as the comparison operator.

For example: You can get the information about a customer who purchased a product and the quantity of product.

An equi-join is further classified into two categories:

(a) SQL Inner Join

(b) SQL Outer Join

(a) SQL Inner Join:

All the rows returned by the sql query satisfy the sql join condition specified.

SQL Inner Join Example:

If you want to display the product information for each order the query will be as given below. Since you are retrieving the data from two tables, you need to identify the common column between these two tables, which is the product_id.

The query for this type of sql joins would be like,

```
SELECT order_id, product_name, unit_price, supplier_name, total_units
FROM product, order_items
WHERE order_items.product_id = product.product_id;
```

The columns must be referenced by the table name in the join condition, because product_id is a column in both the tables and needs a way to be identified. This avoids ambiguity in using the columns in the SQL SELECT statement.

The number of join conditions is (n-1), if there are more than two tables joined in a query where 'n' is the number of tables involved. The rule must be true to avoid Cartesian product.

We can also use aliases to reference the column name, then the above query would be like,

```
SELECT o.order_id, p.product_name, p.unit_price, p.supplier_name, o.total_units
FROM product p, order_items o
WHERE o.product_id = p.product_id;
```

(b) SQL Outer Join:

This sql join condition returns all rows from both tables which satisfy the join condition along with rows which do not satisfy the join condition from one of the tables. The sql outer join operator in Oracle is (+) and is used on one side of the join condition only.

The syntax differs for different RDBMS implementation. Few of them represent the join conditions as "sql left outer join", "sql right outer join".

If you want to display all the product data along with order items data, with null values displayed for order items if a product has no order item, the sql query for outer join would be as shown below:

```
SELECT p.product_id, p.product_name, o.order_id, o.total_units
FROM order_items o, product p
WHERE o.product_id (+) = p.product_id;
```

The output would be like,

product_id	product_name	order_id	total_units
-------------	-------------	-------------	-------------
100	Camera		
101	Television	5103	10
102	Refrigerator	5101	5
103	Ipod	5102	25
104	Mobile	5100	30

NOTE: If the (+) operator is used in the left side of the join condition it is equivalent to left outer join. If used on the right side of the join condition it is equivalent to right outer join.

SQL Self Join:

A Self Join is a type of sql join which is used to join a table to itself, particularly when the table has a FOREIGN KEY that references its own PRIMARY KEY. It is necessary to ensure that the join statement defines an alias for both copies of the table to avoid column ambiguity.

The below query is an example of a self join,

```
SELECT a.sales_person_id, a.name, a.manager_id, b.sales_person_id, b.name
FROM sales_person a, sales_person b
WHERE a.manager_id = b.sales_person_id;
```

2. SQL Non equi joins

It is a sql join condition which makes use of some comparison operator other than the equal sign like >, <, >=, <=

A Non Equi Join is a SQL Join whose condition is established using all comparison operators except the equal (=) operator. Like >=, <=, <, >

SQL Non Equi Join Example:

If you want to find the names of students who are not studying either Economics, the sql query would be like, (lets use student_details table defined earlier.)

```
SELECT first_name, last_name, subject
FROM student_details
WHERE subject != 'Economics'
```

The output would be something like,

first_name	last_name	subject
-------------	-------------	-------------
Anajali	Bhagwat	Maths
Shekar	Gowda	Maths
Rahul	Sharma	Science
Stephen	Fleming	Science

Q22. Explain with diagrams about Left, Right, Full, Self joins ?

Ans :

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Let’s look at a selection from the “Orders” table:

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Then, look at a selection from the “Customers” table:

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

Notice that the “CustomerID” column in the “Orders” table refers to the “CustomerID” in the “Customers” table. The relationship between the two tables above is the “CustomerID” column.

Then, we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values in both tables:

Example

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNERJOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

and it will produce something like this:

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

Different Types of SQL JOINs

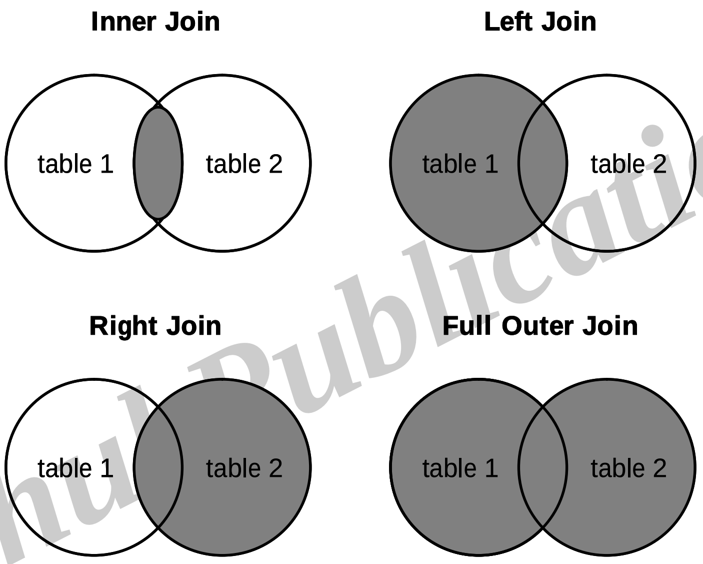
Here are the different types of the JOINs in SQL:

(i) INNER JOIN: Returns records that have matching values in both tables.

(ii) LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table.

(iii) RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table.

(iv) FULL (OUTER) JOIN: Return all records when there is a match in either left or right table.



(i) SQL INNER JOIN Keyword

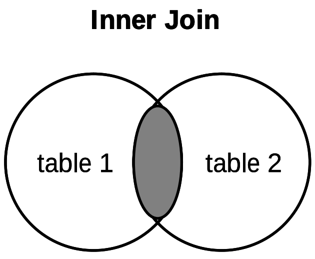
The INNER JOIN keyword selects records that have matching values in both tables.

INNER JOIN Syntax

SELECT*column_name(s)*

FROM*table1*

INNERJOIN*table2* ON*table1.column_name* = *table2.column_name*;



Demo Database

We will use the well-known Northwind sample database.

Below is a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

And a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Postal Code	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

(ii) SQL LEFT JOIN Keyword

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

LEFT JOIN Syntax

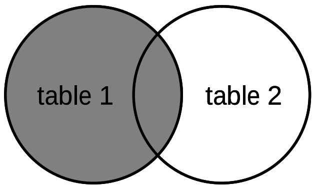
SELECT*column_name(s)*

FROM*table1*

LEFTJOIN*table2* ON*table1.column_name* = *table2.column_name*;

Note: In some databases LEFT JOIN is called LEFT OUTER JOIN.

Left Join



Demo Database

We will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Postal Code	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

And a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

SQL LEFT JOIN Example

The following SQL statement will select all customers, and any orders they might have:

Example

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFTJOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDERBY Customers.CustomerName;
```

(iii) SQL RIGHT JOIN Keyword

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

RIGHT JOIN Syntax

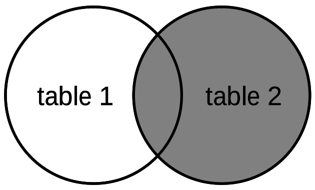
SELECT*column_name(s)*

FROM*table1*

RIGHTJOIN*table2* ON*table1.column_name* = *table2.column_name*;

Note: In some databases RIGHT JOIN is called RIGHT OUTER JOIN.

Right Join



Demo Database

We will use the well-known Northwind sample database.

Below is a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

And a selection from the "Employees" table:

EmployeeID	LastName	FirstName	BirthDate	Photo
1	Davolio	Nancy	12/8/1968	EmpID1.pic
2	Fuller	Andrew	2/19/1952	EmpID2.pic
3	Leverling	Janet	8/30/1963	EmpID3.pic

SQL RIGHT JOIN Example

The following SQL statement will return all employees, and any orders they might have placed:

Example

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHTJOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDERBY Orders.OrderID;
```

SQL FULL OUTER JOIN Keyword

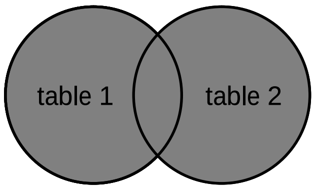
The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.

Note: FULL OUTER JOIN can potentially return very large result-sets!

FULL OUTER JOIN Syntax

```
SELECTcolumn_name(s)
FROMtable1
FULLOUTERJOINtable2 ONtable1.column_name = table2.column_name;
```

Full Outer Join



Demo Database

We will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Postal Code	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

And a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

(iv) SQL FULL OUTER JOIN Example

The following SQL statement selects all customers, and all orders:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULLOUTERJOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

SQL Self JOIN

A self JOIN is a regular join, but the table is joined with itself.

Self JOIN Syntax

SELECT*column_name(s)*

FROM*table1 T1, table1 T2*

WHERE*condition*;

Demo Database

We will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Postal Code	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

SQL Self JOIN Example

The following SQL statement matches customers that are from the same city:

Example

```
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID < > B.CustomerID
AND A.City = B.City
ORDERBY A.City;
```

3.9 Views

Q23. Describe briefly about SQL Views? How can we create views in SQL?

(OR)

Explain Views with example.

Ans : **(July-21, Oct.-19, Imp.)**

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following “

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

Creating Views

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic **CREATE VIEW** syntax is as follows -

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

You can include multiple tables in your SELECT statement in a similar way as you use them in a normal SQL SELECT query.

Example

Consider the CUSTOMERS table having the following records “

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example to create a view from the CUSTOMERS table. This view would be used to have customer name and age from the CUSTOMERS table.

```
SQL > CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM  CUSTOMERS;
```

Now, you can query CUSTOMERS_VIEW in a similar way as you query an actual table. Following is an example for the same.

```
SQL > SELECT * FROM CUSTOMERS_VIEW;
```

This would produce the following result.

Name	Age
Ramesh	32
Khilan	25
kaushik	23
Chaitali	25
Hardik	27
Komal	22
Muffy	24

The WITH CHECK OPTION

The WITH CHECK OPTION is a CREATE VIEW statement option. The purpose of the WITH CHECK OPTION is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition.

If they do not satisfy the condition(s), the UPDATE or INSERT returns an error.

The following code block has an example of creating same view CUSTOMERS_VIEW with the WITH CHECK OPTION.

```
CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM  CUSTOMERS
WHERE age IS NOT NULL
WITH CHECK OPTION;
```

The WITH CHECK OPTION in this case should deny the entry of any NULL values in the view's AGE column, because the view is defined by data that does not have a NULL value in the AGE column.

Updating a View

A view can be updated under certain conditions which are given below -

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.
- The SELECT clause may not contain an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain subqueries.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.
- All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So, if a view satisfies all the above-mentioned rules then you can update that view. The following code block has an example to update the age of Ramesh.

```
SQL > UPDATE CUSTOMERS_VIEW
      SET AGE = 35
      WHERE name = 'Ramesh';
```

This would ultimately update the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Inserting Rows into a View

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command.

Here, we cannot insert rows in the CUSTOMERS_VIEW because we have not included all the NOT NULL columns in this view, otherwise you can insert rows in a view in a similar way as you insert them in a table.

Deleting Rows into a View

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

Following is an example to delete a record having AGE = 22.

```
SQL > DELETE FROM CUSTOMERS_VIEW
  WHERE age =22;
```

This would ultimately delete a row from the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

Dropping Views

Obviously, where you have a view, you need a way to drop the view if it is no longer needed. The syntax is very simple and is given below “

```
DROP VIEW view_name;
```

Following is an example to drop the CUSTOMERS_VIEW from the CUSTOMERS table.

```
DROP VIEW CUSTOMERS_VIEW;
```

3.10 Sequences

Q24. Explain sequences in SQL Queries ?

Ans : **(July-21)**

Meaning

Sequence is a set of integers 1, 2, 3, … that are generated and supported by some database systems to produce unique values on demand.

- A sequence is a user defined schema bound object that generates a sequence of numeric values.
- Sequences are frequently used in many databases because many applications require each row in a table to contain a unique value and sequences provides an easy way to generate them.
- The sequence of numeric values is generated in an a**scending or descending order** at defined intervals and can be configured to restart when exceeds max_value.

Syntax:

```
CREATE SEQUENCE sequence_name
START WITH initial_value
INCREMENT BY increment_value
MINVALUE minimum value
MAXVALUE maximum value
CYCLE|NOCYCLE ;
```

Following is the sequence query creating sequence in ascending order.

Example:

```
CREATE SEQUENCE sequence_1
start with 1
increment by 1
minvalue 0
maxvalue 100
cycle;
```

Above query will create a sequence named *sequence_1*.Sequence will start from 1 and will be incremented by 1 having maximum value 100. Sequence will repeat itself from start value after exceeding 100.

3.11 Indexes and Synonyms

Q25. What is the use of Index in SQL ? How can we create Indexes in SQL ?

Ans :

Indexes are special lookup tables that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.

For example, if you want to reference all pages in a book that discusses a certain topic, you first refer to the index, which lists all the topics alphabetically and are then referred to one or more specific page numbers.

An index helps to speed up **SELECT** queries and **WHERE** clauses, but it slows down data input, with the **UPDATE** and the **INSERT** statements. Indexes can be created or dropped with no effect on the data.

Creating an index involves the **CREATE INDEX** statement, which allows you to name the index, to specify the table and which column or columns to index, and to indicate whether the index is in an ascending or descending order.

Indexes can also be unique, like the **UNIQUE** constraint, in that the index prevents duplicate entries in the column or combination of columns on which there is an index.

The CREATE INDEX Command

The basic syntax of a **CREATE INDEX** is as follows.

```
CREATE INDEX index_name ON table_name;
```

Single-Column Indexes

A single-column index is created based on only one table column. The basic syntax is as follows.

```
CREATE INDEX index_name
ON table_name (column_name);
```

- **Unique Indexes**

  Unique indexes are used not only for performance, but also for data integrity. A unique index does not allow any duplicate values to be inserted into the table. The basic syntax is as follows.

```
CREATE UNIQUE INDEX index_name
on table_name (column_name);
```

- **Composite Indexes**

  A composite index is an index on two or more columns of a table. Its basic syntax is as follows.

```
CREATE INDEX index_name
on table_name (column1, column2);
```

Whether to create a single-column index or a composite index, take into consideration the column(s) that you may use very frequently in a query's WHERE clause as filter conditions.

Should there be only one column used, a single-column index should be the choice. Should there be two or more columns that are frequently used in the WHERE clause as filters, the composite index would be the best choice.

- **Implicit Indexes**

  Implicit indexes are indexes that are automatically created by the database server when an object is created. Indexes are automatically created for primary key constraints and unique constraints.

- **The Drop Index Command**

  An index can be dropped using SQL **DROP** command. Care should be taken when dropping an index because the performance may either slow down or improve.

  The basic syntax is as follows -

```
DROP INDEX index_name;
```

When should indexes be avoided?

Although indexes are intended to enhance a database's performance, there are times when they should be avoided.

The following guidelines indicate when the use of an index should be reconsidered.

- Indexes should not be used on small tables.
- Tables that have frequent, large batch updates or insert operations.
- Indexes should not be used on columns that contain a high number of NULL values.
- Columns that are frequently manipulated should not be indexed.

Q26. Discuss in detail about synonyms management.

Ans :

Synonyms are generally the alias name given to a base table or any schema object. The advantage of using the concept of synonyms is that,

(i) It is possible to provide location transparency

(ii) Level of security can be provided by hiding the name and owner of an object.

(iii) The complexity of SQL statement can be reduced

(iv) The objects can be renamed or moved. This operation does not have any impact on the functionality of the application.

Types of Synonym

Synonyms can be categorized as,

(i) Public synonym

(ii) Private synonym.

(i) Public Synonyms

A synonym is referred to as public synonym if it is owned by a user group which is named "PUBLIC". This synonym can be accessed by any database user.

(ii) Private Synonym

A synonym is referred to as private synonym if it is defined within the schema of a specific user.

Creating Synonyms

A private synonym can be created in the user's schema or in some other user's schema. This is done using the privilege "CREATE SYNONYM" and "CREATE ANY SYNONYM" respectively. Moreover, a public synonym is created using the privilege "CREATE PUBLIC SYNONYM".

Example

CREATE PUBLIC SYNONYM pubstd FOR Robertstd.

The above statement creates a public synonym called "pub std" on the "std" table defined in the schema of Robert.

Dropping Synonym

The private synonym created in a user's schema is deleted using the statement.

"DROP SYNONYM"

Syntax

DROP SYNONYM SYNONYM_Name

Example

DROP SYNONYM std;

The private synonym created in another users schema is deleted using the statement "DROP ANY SYNONYM".

Syntax

DROP ANY SYNONYM synonym name

Example

DROP ANY SYNONYM pub_std

The public synonym can be dropped using the statement "DROP PUBLIC SYNONYM".

Syntax

DROP PUBLIC SYNONYM synonym name;

Example

DROP PUBLIC SYNONYM pub_std;

Dropping a synonym removes it from the data dictionary.

3.12 Table Handling

Q27. How can we handle the tables in SQL?

Ans : **(Imp.)**

In RDBMS more than one table can be handled at a time by using join operation.

Joins operation is a relational operation that causes 2 tables with a common domain to be combined into single table or view with the help of constrains of

1. Primary key
2. Foreign key

SQL Create Constraints

Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

Syntax

```
CREATETABLEtable_name (
    column1 datatypeconstraint,
    column2 datatypeconstraint,
    column3 datatypeconstraint,
    ....
);
```

SQL Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/ record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified
- **INDEX** - Used to create and retrieve data from the database very quickly

SQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are different.

Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint.

However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

SQL UNIQUE Constraint on CREATE TABLE

The following SQL creates a UNIQUE constraint on the "ID" column when the "Persons" table is created:

SQL Server / Oracle / MS Access:

```
CREATETABLE Persons (
    ID int NOTNULLUNIQUE,
    LastName varchar(255) NOTNULL,
    FirstName varchar(255),
    Age int
);
```

MySQL:

```
CREATETABLE Persons (
    ID int NOTNULL,
    LastName varchar(255) NOTNULL,
    FirstName varchar(255),
    Age int,
    UNIQUE (ID)
);
```

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only one primary key, which may consist of single or multiple fields.

SQL PRIMARY KEY on CREATE TABLE

The following SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

MySQL:

```
CREATETABLE Persons (
    ID int NOTNULL,
    LastName varchar(255) NOTNULL,
    FirstName varchar(255),
    Age int,
    PRIMARYKEY (ID)
);
```

SQL FOREIGN KEY Constraint

A FOREIGN KEY is a key used to link two tables together.

A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

Look at the following two tables:

"Persons" table:

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

"Orders" table:

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

SQL FOREIGN KEY on CREATE TABLE

The following SQL creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created:

MySQL:

```
CREATETABLE Orders (
    OrderID int NOTNULL,
    OrderNumber int NOTNULL,
    PersonID int,
    PRIMARYKEY (OrderID),
    FOREIGNKEY (PersonID) REFERENCES Persons(PersonID)
);
```

SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a single column it allows only certain values for this column.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

SQL CHECK on CREATE TABLE

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that you can not have any person below 18 years:

MySQL:

```
CREATETABLE Persons (
    ID int NOTNULL,
    LastName varchar(255) NOTNULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

SQL DEFAULT Constraint

The DEFAULT constraint is used to provide a default value for a column.

The default value will be added to all new records IF no other value is specified.

SQL DEFAULT on CREATE TABLE

The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:

My SQL / SQL Server / Oracle / MS Access:

```
CREATETABLE Persons (
    ID int NOTNULL,
    LastName varchar(255) NOTNULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT'Sandnes'
);
```

SQL CREATE INDEX Statement

The CREATE INDEX statement is used to create indexes in tables.

Indexes are used to retrieve data from the database very fast. The users cannot see the indexes, they are just used to speed up searches/queries.

Note: Updating a table with indexes takes more time than updating a table without (because the indexes also need an update). So, only create indexes on columns that will be frequently searched against.

CREATE INDEX Syntax

Creates an index on a table. Duplicate values are allowed:

CREATEINDEX*index_name*

ON*table_name* (*column1*, *column2*, ...);

CREATE UNIQUE INDEX Syntax

Creates a unique index on a table. Duplicate values are not allowed:

CREATEUNIQUEINDEX*index_name*

ON*table_name* (*column1*, *column2*, ...);

Short Question and Answers

1. SQL

Ans :

SQL stands for Structured Query Language.

SQL is a language that enables you to work with a database. Using SQL, you can insert records, update records, and delete records. You can also create new database objects such as databases and tables. And you can drop (delete) them.

More advanced features include creating stored procedures (self contained scripts), views (pre-made queries), and setting permissions on database objects (such as tables, stored procedures, and views).

Although SQL is an ANSI (American National Standards Institute) standard, there are many different versions of SQL. Different database vendors have their own variations of the language.

2. Define operation ? List out the operators of SQL ?

Ans :

An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations. These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

- Arithmetic operators
- Comparison operators
- Logical operators
- Operators used to negate conditions

3. Explain about TRUNCATE Command.

Ans :

The SQL **TRUNCATE TABLE** command is used to delete complete data from an existing table.

You can also use DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.

Syntax

The basic syntax of a **TRUNCATE TABLE** command is as follows.

```
TRUNCATE TABLE  table_name;
```

4. Explain in detail about Having clause in SQL ?

Ans :

SQL HAVING Clause

Having clause is used to filter data based on the group functions. This is similar to WHERE condition but is used with group functions. Group functions cannot be used in WHERE Clause but can be used in HAVING clause.

SQL HAVING Clause Example

If you want to select the department that has total salary paid for its employees more than 25000, the sql query would be like;

```
SELECT dept, SUM (salary)
FROM employee
GROUP BY dept
HAVING SUM (salary) > 25000
```

The output would be like:

dept	salary
-------------	-------------
Electronics	55000
Aeronautics	35000
InfoTech	30000

When WHERE, GROUP BY and HAVING clauses are used together in a SELECT statement, the WHERE clause is processed first, then the rows that are returned after the WHERE clause is executed are grouped based on the GROUP BY clause.

Finally, any conditions on the group functions in the HAVING clause are applied to the grouped rows before the final output is displayed.

5. Differentiate between RollBack and Commit ?

Ans :

ROLLBACK	COMMIT
ROLLBACK command is used to undo the changes made by the DML commands.	The COMMIT command is used to save the modifications done to the database values by the DML commands.
It rollbacks all the changes of the current transaction.	It will make all the changes permanent that cannot be rolled back.
Syntax: DELETE FROM table_name ROLLBACK	**Syntax:** COMMIT;

6. Define view in SQL ?

Ans :

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following “

- ➢ Structure data in a way that users or classes of users find natural or intuitive.
- ➢ Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- ➢ Summarize data from various tables which can be used to generate reports.

Creating Views

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic **CREATE VIEW** syntax is as follows -

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

7. Define Index.

Ans :

Indexes are **special lookup tables** that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.

For example, if you want to reference all pages in a book that discusses a certain topic, you first refer to the index, which lists all the topics alphabetically and are then referred to one or more specific page numbers.

An index helps to speed up **SELECT** queries and **WHERE** clauses, but it slows down data input, with the **UPDATE** and the **INSERT** statements. Indexes can be created or dropped with no effect on the data.

Creating an index involves the **CREATE INDEX** statement, which allows you to name the index, to specify the table and which column or columns to index, and to indicate whether the index is in an ascending or descending order.

Indexes can also be unique, like the **UNIQUE** constraint, in that the index prevents duplicate entries in the column or combination of columns on which there is an index.

The CREATE INDEX Command

The basic syntax of a **CREATE INDEX** is as follows.

```
CREATE INDEX index_name ON table_name;
```

8. List the different types of view.

Ans :

There are two types of views. They are,

(i) Simple view

(ii) Complex view

(i) **Simple view:** It refers to the view which is used for driving data from a single table. It does not have any function or group of data.

(ii) **Complex:** It refers to the view which is used for driving data from multiple tables. It has function or group of data.

9. List the advantages of views.

Ans :

The advantages of views include the following :

(i) They can store complex queries which are most oftenly executed.

(ii) They are virtual and therefore doesn't require space.

(iii) They provide security of avoid undesired access.

(iv) They are capable of displaying various types of data for different kinds of users.

10. Differentiate between table and view.

Ans :

The difference between table and view is that, a table stores original data and uses physical storage while a view stores no data and hence users no storage, view only stores view definition.

Tables are the basic unit of data management. They are used for data storage and manipulation. Data within a table is stored in rows and columns. Columns describe the data, while rows are the instances of table data.

Views can be defined as a logical representation of some other table or mixture of tables. The purpose of the view is to stop the user from viewing all columns from a tale.

11. Write short notes on special operators.

Ans :

Special Operators

These operators are also used for comparing two expressions. Such operators include the following:

(i) BETWEEN

This operator is used for checking whether an attribute value lies within the range or not

(ii) IN

This operator is used for checking whether an attribute value matches any of the values in the specified list.

(iii) IS NULL

This operator is used for checking whether an attribute value returns a null value.

(iv) LIKE

This operator is used for checking whether an attribute value matches a specicied string pattern.

(v) EXISTS

This operator is used for checking whether a set of rows satisfying the conditions return a value.

(vi) DISTINCT

This operator is used for retrieving unique values from the table i.e., removing duplicate rows from the final result set.

Choose the Correct Answers

1. SQL stands for ______. [a]
   (a) Structured Query Language (b) Simple Query Language
   (c) Standard Query Language (d) Significant Query Language
2. Find which are functions of SQL [d]
   (a) Store & Modify data (b) Retrive & Delete data
   (c) Crete tables & DB objects (d) All
3. Describe the size of varchar 2 ( ) [b]
   (a) Up to 4000 bytes (b) Up to 2000 bytes
   (c) Up to 3000 bytes (d) Up to 8000 bytes
4. DATE data type is used to store ______ values [c]
   (a) DATE (b) TIME
   (c) DATE and TIME (d) All
5. TCL stands for ______. [b]
   (a) Transactional Control Language (b) Transaction Control Language
   (c) Transmission Control Language (d) None
6. Insert Command comes under ______. [b]
   (a) DDL (b) DML
   (c) DCL (d) TCL
7. < = is comes under which operator in SQL [b]
   (a) Arthmetic Operator (b) Comparison Operator
   (c) Logical Operator (d) Set Operator
8. Which of the following is Unary operator ______. [a]
   (a) Select (b) Union
   (c) Project (d) Difference
9. In precedence of set operators the expression is Evaluated from ______. [c]
   (a) Left to Left (b) Right to Right
   (c) Left to Right (d) Right to Left
10. Command is used to add a new column in the selected table is ______. [c]
   (a) Update (b) Add
   (c) Alter (add) (d) Alter (modify)

Fill in the blanks

1. A Table joined with itself is called _______.
2. DDL stands for _______ .
3. Sub Query is also termed as _______ .
4. UPPER (string -value) is a _______ function.
5. COMMIT and ROLL BACK comes under _______.
6. _______ Operator is used to select rows that satisfy all the given conditions.
7. _______ is a term is used to represent a missing value.
8. In SQL, SUM, COUNT, AVG, MAX, MIN comes under _______ functions.
9. _______ Clause is used to specify a condition while fetching data from single Table (or) joining with multiple tables.
10. DCL stands for _______.

ANSWERS

1. Self join
2. Data Defination Language
3. Nested Query
4. Character
5. TCL
6. AND
7. NULL
8. Aggrigate
9. WHERE
10. Data Control Language

One Mark Answers

1. Define Nested Query.

Ans :

A Query with in another Query is called Nested - Query

2. Define Functional Dependency.

Ans :

SQL Joins are used to relate information in different tables.

3. SQL Integrity Constraints.

Ans :

Integrity constraints are used to apply business rules for the database tables.

The constraints available in SQL are

- Forign Key
- Not Null
- Unique
- Check

4. SQL Operations

Ans :

An operator is a Reserved word used primarily in SQL statements WHERE clause to perform operation Following are basic operators in SQL

- Arthmetic
- Comparison
- Logical
- Set Operators

5. DML.

Ans :

Data Manuplation Language Commands are used for storing , Retriving, Modifing and Deleting Data.

6. Give the syntax for INSERT command.

Ans :

The syntax for INSERT command is as follows,

INSERT INTO table_name (column 1, column 2, ...) VALUES (value 1, value 2, ...),

OR

INSERT INTO table_name VALUES (value 1, value 2, ...).

7. Explain about dropping a table with an example.

Ans :

The DROP TABLE statement drops a table completely from a database. It deletes the table with corresponding constraints and indexes.

Syntax

```
DROP TABLE    table_name;
```

Example

```
DROP TABLE  Employee;
```

UNIT IV

TRANSACTIONS AND CONCURRENCY MANAGEMENT

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.

4.1 Transactions

Q1. Define Transaction, Explain the states of transactions.

Ans : **(Imp.)**

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Let's take an example of a simple transaction. Suppose a bank employee transfers Rs. 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

A's Account

```
Open_Account(A)
Old_Balance = A.balance
New_Balance = Old_Balance - 500
A.balance = New_Balance
Close_Account(A)
```

B's Account

```
Open_Account(B)
Old_Balance = B.balance
New_Balance = Old_Balance + 500
B.balance = New_Balance
Close_Account(B)
```

States of Transactions

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

- **Active:** In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed:** When a transaction executes its final operation, it is said to be in a partially committed state.
- **Failed:** A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.

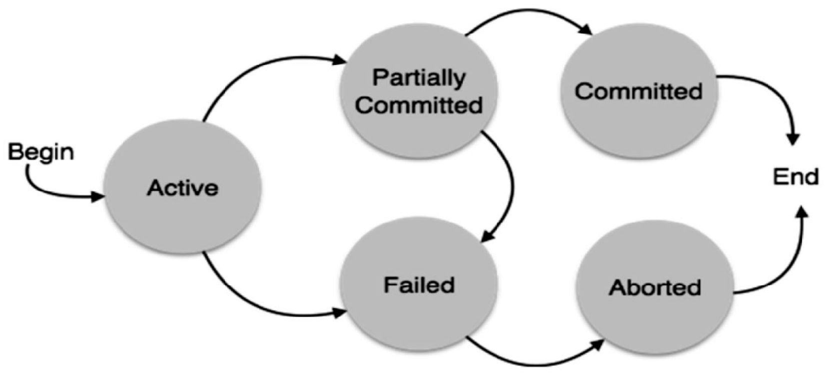


Fig.: States of Transactions

- **Aborted:** If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts:
  - Re-start the transaction
  - Kill the transaction
- **Committed:** If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

Q2. List and explain in detail Transaction Properties.

(OR)

Explain briefly about ACID properties.

Ans : **(Imp.)**

A transaction is a very small unit of a program and it may contain several low level tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability - commonly known as ACID properties - in order to ensure accuracy, completeness, and data integrity.

(i) **Atomicity:** This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.

(ii) **Consistency:** The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.

(iii) **Isolation:** In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

(iv) **Durability**: The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

4.2 Concurrent Transactions

Q3. Define Concurrency? Explain the uses of Equivalence ?

Ans :

Concurrency is the ability of a database to allow multiple users to affect multiple transactions. This is one of the main properties that separates a database from other forms of data storage like spreadsheets.

The ability to offer concurrency is unique to databases. Spreadsheets or other flat file means of storage are often compared to databases, but they differ in this one important regard. Spreadsheets cannot offer several users the ability to view and work on the different data in the same file, because once the first user opens the file it is locked to other users. Other users can read the file, but may not edit data.

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.

➢ **Schedule:** A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/ tasks.

➢ **Serial Schedule:** It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

In a multi-transaction environment, serial schedules are considered as a benchmark. The execution sequence of an instruction in a transaction cannot be changed, but two transactions can have their instructions executed in a random fashion. This execution does no harm if two transactions are mutually independent and working on different segments of data; but in case these two transactions are working on the same data, then the results may vary. This ever-varying result may bring the database to an inconsistent state.

To resolve this problem, we allow parallel execution of a transaction schedule, if its transactions are either serialized or have some equivalence relation among them.

Equivalence Schedules

An equivalence schedule can be of the following types:

(i) **Result Equivalence**

If two schedules produce the same result after execution, they are said to be result equivalent. They may yield the same result for some value and different results for another set of values. That's why this equivalence is not generally considered significant.

(ii) **View Equivalence**

Two schedules would be view equivalence if the transactions in both the schedules perform similar actions in a similar manner.

For example:

➢ If T reads the initial data in S1, then it also reads the initial data in S2.

➢ If T reads the value written by J in S1, then it also reads the value written by J in S2.

➢ If T performs the final write on the data value in S1, then it also performs the final write on the data value in S2.

(iii) **Conflict Equivalence**

Two schedules would be conflicting if they have the following properties:

➢ Both belong to separate transactions.

➢ Both accesses the same data item.

➢ At least one of them is "write" operation.

Two schedules having multiple transactions with conflicting operations are said to be conflict equivalent if and only if:

- Both the schedules contain the same set of Transactions.
- The order of conflicting pairs of operation is maintained in both the schedules.

Q4. What is mean by concurrency control ?

Ans :

Concurrency controls prevent data integrity problems, which can arise when two update processes access the same data item at the same time. Access controls restrict updating of the database to authorized users, and controls such as passwords prevent the inadvertent or unauthorized disclosure of data from the database.

Concurrency controls prevent data integrity problems, which can arise when two update processes access the same data item at the same time. Access controls restrict updating of the database to authorized users, and controls such as passwords prevent the inadvertent or unauthorized disclosure of data from the database. Quality controls, such as edits, ensure the accuracy, completeness and consistency of data maintained in the database.

In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions. We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols can be broadly divided into two categories -

4.3 Locking Protocols

4.3.1 Types of Locks

Q5. Define lock? Explain different types of locks.

Ans : **(July-21, Imp.)**

A lock is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied to it. Generally, there is one lock for each data item in the database. Locks are used as a means of synchronizing the access by concurrent transactions to the database item.

Types of Lock

The following are the two different types of lock,

1. Binary locks
2. Shared/exclusive locks

1. Binary Locks

Binary locks are the locks that consist of two states, locked and unlocked. Binary values ‘1’ and ‘0’ are used for representing locked state and unlocked state respectively. If a data item is in lock state, then it implies that it cannot be accessed by any requested database operation. If the data item is in unlock state, then any database operation that have sent a request for accessing the data item can access it.

Whenever a transaction sends a request for accessing data item (say x) then database system must ensure whether ‘x’ is in unlocked state. If it is in unlocked state, then DBMS grants the lock to the requested transaction, otherwise, the transaction have to wait until the required data item is unlocked.

2. Shared/Exclusive Locks

(i) Shared Lock (S): A transaction that acquired shared lock on data item *P*, can execute read instruction but not the write instruction i.e., these locks support only read integrity. Using these locks multiple users can read the data concurrently, but cannot change/write the data value. The intention of this lock is to ensure that data items are not modified while they are locked using shared locking mode.

(ii) Exclusive Lock (X): A transaction that acquired an exclusive lock on data item P, can execute both read as well as write instruction i.e., these locks support read and write integrity. Only a single transaction can update the data at a time. The intention of this lock is to provide exclusive use of the data item to one transaction i.e., if a transaction T_A holds an exclusive lock on data item P, then no other transaction can read/update that data item until T_A releases the lock.

Whenever a data item is locked using shared lock, then it cannot be locked using exclusive lock. On the other hand, if it is locked using exclusive lock, then it cannot acquire either shared lock or another exclusive lock until the first exclusive lock is released.

Before accessing any data item, a transaction must send a lock request to concurrency control manager. This lock request should specify the appropriate locking mode, which the transaction wants to acquire on the data item. When the lock manager receives the lock request, it checks whether the lock can be granted or not. If a data item is already locked in shared mode, then lock manager will grant the lock permission, but if it is locked in exclusive mode then permission is not granted. In such situation, the transaction that sends the lock request have to wait until the lock is released.

The following are the major problems caused by the locks even if they are used to prevent the serious data inconsistencies,

(i) Non-serializable transaction schedules may occur.

(ii) Deadlocks may occur because of the improper scheduling of the transactions.

Q6. Explain different levels of lock.

Ans :

The following are the different levels of lock in the databases,

1. Database levels

A database levels that are executed in batches. However, they are not applicable for the multiple users that are accessing the same database. The database level locks also make the access time of data slow because several transactions have to wait until the previous transactions are executed completely. Though, two transactions T_1 and T_2 use different tables for execution, the database level lock does not facilitate those transactions to access the same database.

2. Table Level

A table level lock refers to a lock that is provided on the complete table so as to restrict the transaction T_2 to access a row in the table when another transaction T_1 is accessing any other row in the same table. Thus, different transactions can use the same database only when they need different tables for accessing.

In comparison to the database level locks, the table level locks are less restrictive because several transactions have to wait until the previous transactions complete accessing the same table (but not the same database). As a result, the table level locks are not applicable for the multiusers DBMS.

3. Page Level

A page level lock refers to a lock that is provided on an entire diskpage so as to restrict a transaction T_2 to access a page that is locked when another transaction T_1 is accessing the same page. Here, a page/diskpage is similar to the diskblock, which stores the addresses of the diskpages. These pages have a storage capacity like 4 k, 8 k or 16 k. If 73 bytes are used from a 4 k page, then the complete 4k page is taken from the disk, loaded in the memory and then written back to the disk. Each page can have multiple rows of atleast one table and multiple pages can be stored in a single table. As a result, the page level blocks are mostly used for the multiuser DBMSs.

4. Row Level

A row level lock refers to a lock that is provided on a page so as to restrict a transaction T_2 to access a page when another transaction T_1 is accessing the same page. Since every row in a table is locked, conflict transactions may arise due to which, the management of the row level locks require huge amount of CPU overhead. But, however these type of locks increase the availability of level of data. In addition to this, they are much less restrictive when compared to the other level of locks.

When multiple locks are to be requested on the same page, the modem DBMSs automatically change the locks from row level to page level.

5. Field Level

A field level lock refers to a lock that is provided on the fields so as to restrict a transaction T_2 to access a field when another transaction T_1 is accessing the same field. It is mostly applicable for the flexible multiuser DBMSs but rarely implemented in the system because of the high level of overhead requirement. As a result, the field level locks allow multiple transactions to access the same row only when they had to access different field of that particular row)

4.4 Serializable Schedules

Q7. Explain briefly about Serializable schedule ?

Ans : **(Imp.)**

Meaning

Serializability is the classical concurrency scheme. It ensures that a schedule for executing concurrent transactions is equivalent to one that executes the transactions serially in some order. It assumes that all accesses to the database are done using read and write operations.

Conflict Serializable Schedule

- A schedule is called conflict serializability if after swapping of nonconflicting operations, it can transform into a serial schedule.
- The schedule will be a conflict serializable if it is conflict equivalent to a serial schedule.

Conflicting Operations

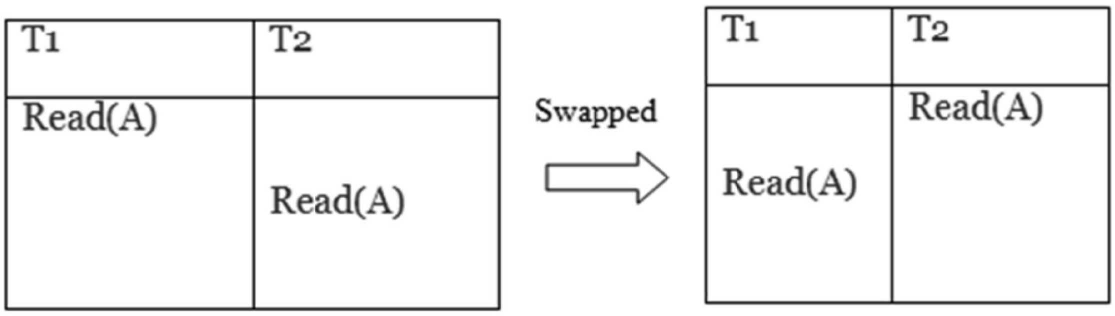
The two operations become conflicting if all conditions satisfy:

1. Both belong to separate transactions.
2. They have the same data item.
3. They contain at least one write operation.

Example:

Swapping is possible only if S1 and S2 are logically equal.

1. T1: Read (A) T2: Read (A)



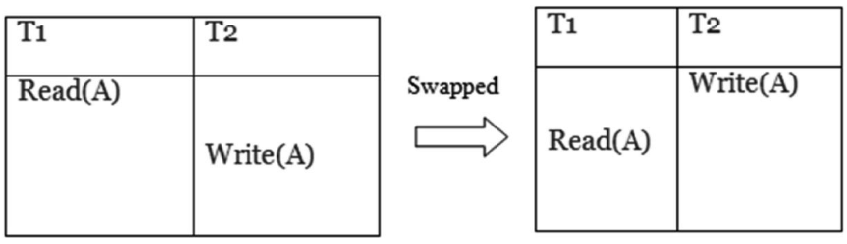
T1	T2
Read(A)	
	Read(A)

T1	T2
	Read(A)
Read(A)	

Schedule S1 **Schedule S2**

Here, S1 = S2. That means it is non-conflict.

2. T1: Read (A) T2: Write(A)



T1	T2
Read(A)	
	Write(A)

T1	T2
	Write(A)
Read(A)	

Schedule S1 **Schedule S2**

Here, S1 ‘“ S2. That means it is conflict.

Conflict Equivalent

In the conflict equivalent, one can be transformed to another by swapping non-conflicting operations. In the given example, S2 is conflict equivalent to S1 (S1 can be converted to S2 by swapping non-conflicting operations).

Two schedules are said to be conflict equivalent if and only if:

1. They contain the same set of the transaction.
2. If each pair of conflict operations are ordered in the same way.

Example

Non-serial schedule		**Serial Schedule**	
T1	**T2**	**T1**	**T2**
Read(A)		Read(A)	
Write(A)		Write(A)	
	Read(A)	Read(B)	
	Write(A)	Write(B)	
Read(B)			Read(A)
Write(B)			Write(A)
	Read(B)		Read(B)
	Write(B)		Write(B)

Schedule S1 **Schedule S2**

Schedule S2 is a serial schedule because, in this, all operations of T1 are performed before starting any operation of T2. Schedule S1 can be transformed into a serial schedule by swapping non-conflicting operations of S1.

After swapping of non-conflict operations, the schedule S1 becomes:

T1	**T2**

Read(A)
Write(A)
Read(B)
Write(B)
	Read(A)
	Write(A)
	Read(B)
	Write(B)

Since, S1 is conflict serializable.

Q8. Explain the basic protocols for concurrency control ?

Ans :

In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions. We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols can be broadly divided into two categories -

A) Lock based protocols

B) Time stamp based protocols

A) Lock-based Protocols

Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it. Locks are of two kinds -

➢ **Binary Locks:** A lock on a data item can be in two states; it is either locked or unlocked.

➢ **Shared/exclusive:** This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

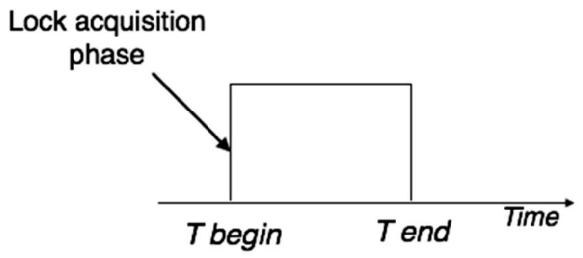
There are four types of lock protocols available:

Simplistic Lock Protocol

Simplistic lock-based protocols allow transactions to obtain a lock on every object before a 'write' operation is performed. Transactions may unlock the data item after completing the 'write' operation.

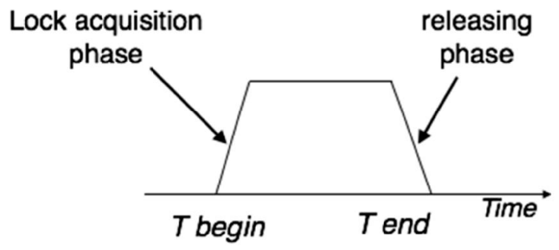
Pre-claiming Lock Protocol

Pre-claiming protocols evaluate their operations and create a list of data items on which they need locks. Before initiating an execution, the transaction requests the system for all the locks it needs beforehand. If all the locks are granted, the transaction executes and releases all the locks when all its operations are over. If all the locks are not granted, the transaction rolls back and waits until all the locks are granted.



Two-Phase Locking (2PL)

This locking protocol divides the execution phase of a transaction into three parts. In the first part, when the transaction starts executing, it seeks permission for the locks it requires. The second part is where the transaction acquires all the locks. As soon as the transaction releases its first lock, the third phase starts. In this phase, the transaction cannot demand any new locks; it only releases the acquired locks.

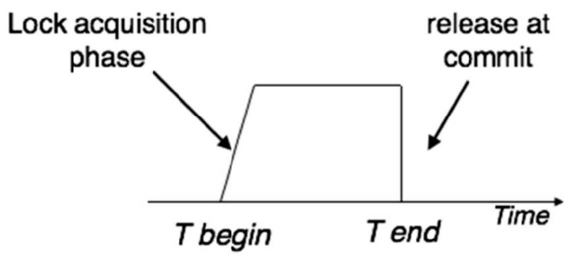


Two-phase locking has two phases, one is growing, where all the locks are being acquired by the transaction; and the second phase is shrinking, where the locks held by the transaction are being released.

To claim an exclusive (write) lock, a transaction must first acquire a shared (read) lock and then upgrade it to an exclusive lock.

Strict Two-Phase Locking (S2PL)

The first phase of Strict-2PL is same as 2PL. After acquiring all the locks in the first phase, the transaction continues to execute normally. But in contrast to 2PL, Strict-2PL does not release a lock after using it. Strict-2PL holds all the locks until the commit point and releases all the locks at a time.



Strict-2PL does not have cascading abort as 2PL does.

B) Time stamp-based Protocols

The most commonly used concurrency protocol is the timestamp based protocol. This protocol uses either system time or logical counter as a timestamp.

Lock-based protocols manage the order between the conflicting pairs among transactions at the time of execution, whereas timestamp-based protocols start working as soon as a transaction is created.

Every transaction has a timestamp associated with it, and the ordering is determined by the age of the transaction. A transaction created at 0002 clock time would be older than all other transactions that come after it. For example, any transaction 'y' entering the system at 0004 is two seconds younger and the priority would be given to the older one.

In addition, every data item is given the latest read and write-timestamp. This lets the system know when the last 'read and write' operation was performed on the data item.

Timestamp Ordering Protocol

The timestamp-ordering protocol ensures serializability among transactions in their conflicting read and write operations. This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.

- The timestamp of transaction Ti is denoted as TS(Ti).
- Read time-stamp of data-item X is denoted by R-timestamp(X).
- Write time-stamp of data-item X is denoted by W-timestamp(X).

Timestamp ordering protocol works as follows:

- **If a transaction Ti issues a read (X) operation -**
  - i) If TS(Ti) < W-timestamp (X)
    - Operation rejected.
  - ii) If TS(Ti) > = W-timestamp(X)
    - Operation executed.
  - iii) All data-item time stamps updated.
- **If a transaction Ti issues a write (X) operation –**
  - i) If TS(Ti) < R-timestamp(X)
    - Operation rejected.
  - ii) If TS(Ti) < W-timestamp(X)
    - Operation rejected and Ti rolled back.
  - iii) Otherwise, operation executed.

Thomas' Write Rule

This rule states if TS(Ti) < W-timestamp(X), then the operation is rejected and Ti is rolled back.

Time-stamp ordering rules can be modified to make the schedule view serializable.

Instead of making Ti rolled back, the 'write' operation itself is ignored.

4.5 Locks Two Phase Locking (2PL)

Q9. Explain in detail about Two phase Locking (2PL) Protocol ?

Ans : **(July-21, Oct.-20, Imp.)**

Two-phase locking (2PL)

According to the two-phase locking protocol, a transaction handles its locks in two distinct, consecutive phases during the transaction's execution:

1. **Expanding phase** (aka Growing phase): locks are acquired and no locks are released (the number of locks can only increase).
2. **Shrinking phase** (aka Contracting phase): locks are released and no locks are acquired.

The two phase locking rule can be summarized as: never acquire a lock after a lock has been released. The serializability property is guaranteed for a schedule with transactions that obey this rule.

Typically, without explicit knowledge in a transaction on end of phase-1, it is safely determined only when a transaction has completed processing and requested commit. In this case, all the locks can be released at once (phase-2).

Conservative two-phase locking(C2PL)

The difference between 2PL and C2PL is that C2PL's transactions obtain all the locks they need before the transactions begin. This is to ensure that a transaction that already holds some locks will not block waiting for other locks. Conservative 2PL prevents deadlocks.

Strict two-phase locking (S2PL)

To comply with the S2PL protocol, a transaction needs to comply with 2PL, and release its write (exclusive) locks only after it has ended, i.e., being either committed or aborted. On the other hand, read (shared) locks are released regularly during phase 2. This protocol is not appropriate in B-trees because it causes Bottleneck (while B-trees always starts searching from the parent root).

Strong strict two-phase locking(SS2PL)or Rigorousness, or Rigorous scheduling, or Rigorous two-phase locking

To comply with strong strict two-phase locking (SS2PL) the locking protocol releases both write (exclusive) and read (shared) locks applied by a transaction only after the transaction has ended, i.e., only after both completing executing (being ready) and becoming either committed or aborted. This protocol also complies with the S2PL rules. A transaction obeying SS2PL can be viewed as having phase-1 that lasts the transaction's entire execution duration, and no phase-2 (or a degenerate phase-2). Thus, only one phase is actually left, and "two-phase" in the name seems to be still utilized due to the historical development of the concept from 2PL, and 2PL being a super-class.

The SS2PL property of a schedule is also called Rigorousness. It is also the name of the class of schedules having this property, and an SS2PL schedule is also called a "rigorous schedule". The term "Rigorousness" is free of the unnecessary legacy of "two-phase," as well as being independent of any (locking) mechanism (in principle other blocking mechanisms can be utilized). The property's respective locking mechanism is sometimes referred to as Rigorous 2PL.

4.6 Deadlock and its Prevention

Q10. What is meant by Deadlock? State the prevention of dead lock.

(OR)

Explain Deadlock and its prevention.

Ans : **(Oct.-19, June-18)**

Meaning

A deadlock is a condition where two or more transactions are waiting indefinitely for one another to give up locks. Deadlock is said to be one of the most feared complications in DBMS as no task ever gets finished and is in waiting state forever.

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.

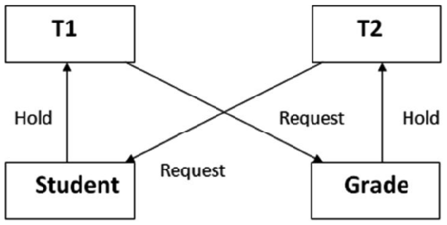


Fig.: Deadlock in DBMS

Deadlock Avoidance

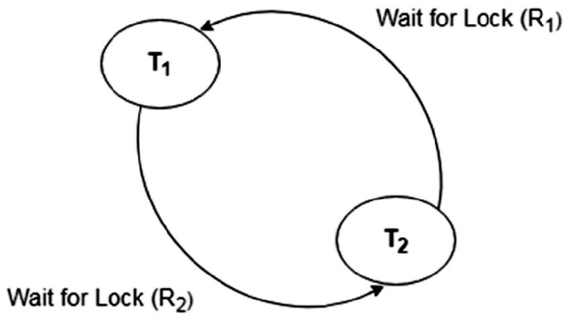
- When a database is stuck in a deadlock state, then it is better to avoid the database rather than aborting or restating the database. This is a waste of time and resource.
- Deadlock avoidance mechanism is used to detect any deadlock situation in advance. A method like "wait for graph" is used for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

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.

- 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.

The wait for a graph for the above scenario is shown below:



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.

(i) Wait-Die scheme

In this scheme, if a transaction requests for a resource which is already held with a conflicting lock by another transaction then the DBMS simply checks the timestamp of both transactions. It allows the older transaction to wait until the resource is available for execution.

Let's assume there are two transactions Ti and Tj and let TS(T) is a timestamp of any transaction T. If T2 holds a lock by some other transaction and T1 is requesting for resources held by T2 then the following actions are performed by DBMS:

1. Check if TS(Ti) < TS(Tj) - If Ti is the older transaction and Tj has held some resource, then Ti is allowed to wait until the data-item is available for execution. That means if the older transaction is waiting for a resource which is locked by the younger transaction, then the older transaction is allowed to wait for resource until it is available.
2. Check if TS(Ti) < TS(Tj) - If Ti is older transaction and has held some resource and if Tj is waiting for it, then Tj is killed and restarted later with the random delay but with the same timestamp.

(ii) Wound wait scheme

- In wound wait scheme, if the older transaction requests for a resource which is held by the younger transaction, then older transaction forces younger one to kill the transaction and release the resource. After the minute delay, the younger transaction is restarted but with the same timestamp.
- If the older transaction has held a resource which is requested by the Younger transaction, then the younger transaction is asked to wait until older releases it.

Advantages of Wound-wait Scheme

(i) Deadlock never occurs because higher priority transactions need not wait for lower priority transaction.

(ii) It also avoids starvation.

Disadvantage of Wound-wait Scheme

Unnecessary rollbacks may occur.

It is necessary to ensure that, when a transaction is aborted and restarted again it should have the timestamp value equal to the timestamp value that was prior to aborting the transaction, otherwise reassignment of time stamps causes each transaction to become the oldest transaction.

Deadlock Detection

In order to detect and recover from the deadlocks a system must perform the following operations.

1. It should maintain the information about the allocation of the locks to different transactions and the outstanding requests.

2. It should provide an algorithm that determines the occurrence of deadlock.

3. Whenever a deadlock has been detected find out the ways to recover from it. For deadlock detection the lock manager maintains a structure called a "waits for" graph in which nodes represents the active transactions and the are from T_i to T_j ($T_i \rightarrow T_j$) represents that T_i is waiting for T_j to release a lock. These edges are added to the graph by the lock manager whenever a transaction requests a lock and are removed when it grants lock requests.

4.7 Optimistic Concurrency Control

Q11. Explain in detail about Optimistic concurrency control ?

(OR)

Discuss briefly about optimistic concurrency control.

Ans : **(June-19, Imp.)**

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 timestamping techniques. Instead, a transaction is executed without restrictions until it is committed. In optimistic methods, each transaction moves through the following phases:

(a) Read phase.

(b) Validation (or) certification phase.

(c) Write phase.

(a) Read phase

In a Read phase, the updates are prepared using private (or local) copies (or versions) of the granule. In this phase, the transaction reads values of committed data from 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.

It is conventional to allocate a timestamp to each transaction at the end of its Read to determine the set of transactions that must be examined by the validation procedure. These set of transactions are those who have finished their Read phases since the start of the transaction being verified

(b) Validation or certification phase

In a validation (or certification) phase, the transaction is validated to assure 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. Thus, in this phase the list of granules is checked for conflicts. If conflicts are detected in this phase, the transaction is aborted and restarted. The validation algorithm must check that the transaction has :

- Seen all modifications of transactions committed after it starts.
- Not read granules updated by a transaction committed after its start.

(c) Write phase

In a Write phase, the changes are permanently applied to the database and the updated granules are made public. Otherwise, the updates are discarded and the transaction is restarted. This phase is only for the Read-Write transactions and not for Read-only transactions.

Q. Explain the Advantages, Problems and Applications of optimistic concurrency control.

Ans :

Advantages

(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

(i) Conflicts are expensive to deal with, since the conflicting transaction must be rolled back.

(ii) Longer transactions are more likely to have conflicts and may be repeatedly rolled back because of conflicts with short transactions.

Applications

(i) Only suitable for environments where there are few conflicts and no long transactions.

(ii) Acceptable for mostly Read or Query database systems that require very few update transactions

4.8 Database Recovery and Security

4.8.1 Meaning

Q12. Define Database Recovery ?

Ans : **(June-19)**

During the life of a transaction, that is, a after the start of a transaction but before the transaction commits, several changes may be made in a database state. The database during such a state is in an inconsistent state. What happens when a failure occurs at this stage?

Let us explain this with the help of an example: Assume that a transaction transfers Rs. 2000/ from A's account to B's account. For simplicity we are not showing any error checking in the transaction. The transaction may be written as:

```
Transaction T1:
READ A
A = A –2000
WRITE A
Failure
READ B
B = B + 2000
WRITE B
COMMIT
```

Failures and Recovery

In practice several things might happen to prevent a transaction from completing.

Recovery techniques are used to bring database, which does not satisfy consistency requirements, into a consistent state. If a transaction completes normally and commits then all the changes made by the transaction on the database are permanently registered in the database. They should not be lost But, if a transaction does not complete normally and terminates abnormally then all the changes made by it should be discarded. An abnormal termination of a transaction may be due to several reasons, including:

a) user may decide to abort the transaction issued by him/ her

b) there might be a deadlock in the system

c) there might be a system failure.

The recovery mechanisms must ensure that a consistent state of database can be restored under all circumstances. In case of transaction abort or deadlock, the system remains in control and can deal with the failure but in case of a system failure the system loses control because the computer itself has failed. Will the results of such failure be catastrophic? A database contains a huge amount of useful information and any system failure should be recognized on the restart of the system. The DBMS should recover from any such failures.

4.8.2 Kinds of Failure

Q13. Explain the different types of database failures ?

Ans :

The kinds of failures that a transaction program during its execution can encounter are:

1. Software Failures

In such cases, a software error abruptly stops the execution of the current transaction (or all transactions), thus leading to losing the state of program execution and the state/ contents of the buffers. But what is a buffer? A buffer is the portion of RAM that stores the partial contents of database that is currently needed by the transaction. The software failures can further be subdivided as:

a) Statement or application program failure

b) Failure due to viruses

c) DBMS software failure

d) Operating system failure

A Statement of program may cause abnormal termination if it does not execute completely. This happens if during the execution of a statement, an integrity constraint gets violated. This leads to abnormal termination of the transaction due to which any prior updates made by the transaction may still get reflected in the database leaving it in an inconsistent state. A failure of transaction can occur if some code in a transaction program leads to its abnormal termination. For example, a transaction can go into an infinite loop.

In such a case the only way to break the loop is to abort the program. Similarly, the failure can be traced to the operating system or DBMS and transactions are aborted abruptly.

Thus part of the transaction that was executed before abort may cause some updates in database, and hence the database is updated only partially which leads to an inconsistent state of database.

2. Hardware Failure

Hardware failures are those failures when some hardware chip or disk fails. This may result in loss of data. One such problem can be that a disk gets damaged and cannot be read any more. This may be due to many reasons. For example, a voltage fluctuation in the power supply to the computer makes it go off or some bad sectors may come on disk or there is a disk crash. In all these cases, the database gets into an inconsistent state.

3. External Failure

A failure can also result due to an external cause, such as fire, earthquakes, floods, etc. The database must be duly backed up to avoid problems occurring due to such failures. In practice software failures are more common than hardware failures. Fortunately, recovery from software failures is much quicker.

4.8.3 Failure Controlling Methods

Q14. Explain briefly about failure controlling methods.

Ans :

Any transaction must maintain one of the following two states to maintain data integrity. The two states are,

1. Aborted state
2. Committed state

1. Aborted State

An aborted state is a state in which the database is restored back to the state which exists before execution. Any incompleted transaction (transaction which does not complete its process) must be aborted since, it does not affects the consistent state of the database.

2. Committed State

A committed state is a state in which a transaction completes its processing successfully. Committed transactions always leaves the database in consistent state.

The following are the three procedures used for database recovery,

(a) Deferred database modifications

(b) Immediate database modifications

(c) Checkpointing

(a) Deferred Database Modification

Using this technique, all the database modifications can be recorded in the log which ensures the transaction's atomicity. But, the execution of all the write operations should not be deferred till a transaction is partially committed and a transaction is said to be partially committed only when the execution of the transaction is fully completed. In the deferred-modification technique, the transactions are executed serially.

The deferred write can be executed by using the information of the partially committing the transaction. But, if the system fails or the transaction aborts then its associated information on the log is omitted.

(b) Immediate Database Modification

Using this technique, the data modifications are performed on the database even in the active state. If the modifications done to the data are written by active transactions then they are referred to as uncommitted modifications. Upon a system crash or transaction failure, the old value of the records in the log must be used by the system so that the modified data items can retrieve the previously assigned values.

(c) Checkpointing

To overcome the drawbacks of deferred database modification and immediate database modification, checkpointing is introduced.

Checkpoints are the synchronization points between the database and the log file for reducing the time taken to recover from the system crash.

In this, the deferred database modification and immediate database modification techniques helps in maintaining the log in the system while executing the transactions. Also, the system performs checkpointing on a regular basis. Here, checkpoints refers to a sequence of actions carried out in the following manners,

(i) All the log records that are currently available inside the main memory must be moved to the stable storage.

(ii) All the updated buffer blocks must be moved to the disk.

(iii) A log record checkpoint L> must be moved to the stable storage.

4.9 Data Base Errors

Q15. What is Error ? Explain the types of Database Errors ?

Ans : **(Imp.)**

An error is said to have occurred if the execution of a command to manipulate the database cannot be successfully completed either due to inconsistent data or due to state of program. For example, there may be a command in program to store data in database. On the execution of command, it is found that there is no space/place in database to accommodate that additional data. Then it can be said that an error has occurred. This error is due to the physical state of database storage.

Broadly errors are classified into the following categories:

1. User error

This includes errors in the program (e.g., Logical errors) as well as errors made by online users of database. These types of errors can be avoided by applying some check conditions in programs or by limiting the access rights of online users e.g., read only. So only updating or insertion operation require appropriate check routines that perform appropriate checks on the data being entered or modified. In case of an error, some prompts can be passed to user to enable him/her to correct that error.

2. Consistency error

These errors occur due to the inconsistent state of database caused may be due to wrong execution of commands or in case of a transaction abort. To overcome these errors the database system should include routines that check for the consistency of data entered in the database.

3. System error

These include errors in database system or the OS, e.g., deadlocks. Such errors are fairly hard to detect and require reprogramming the erroneous components of the system software.

Database errors can result from failure or can cause failure and thus will require recovery. However, one of the main tasks of database system designers is to make sure that errors minimized. These concepts are also related to database integrity and have also been discusses in a later section.

4.10 Backup and Recovery Techniques

Q16. How can we Backup our Database ?

Ans : **(Oct.-20, June-18)**

Database Backup

- Database Backup is storage of data that means the copy of the data.
- It is a safeguard against unexpected data loss and application errors.
- It protects the database against data loss.
- If the original data is lost, then using the backup it can reconstructed.

The backups are divided into two types:

1. Physical Backup
2. Logical Backup

1. Physical backups

- Physical Backups are the backups of the physical files used in storing and recovering your database, such as data files, control files and archived redo logs, log files.
- It is a copy of files storing database information to some other location, such as disk, some off-line storage like magnetic tape.
- Physical backups are the foundation of the recovery mechanism in the database.
- Physical backup provides the minute details about the transaction and modification to the database.

2. Logical backup

- Logical Backup contains logical data which is extracted from a database.
- It includes backup of logical data like views, procedures, functions, tables, etc.
- It is a useful supplement to physical backups in many circumstances but not a sufficient protection against data loss without physical backups, because logical backup provides only structural information.

Importance of Backups

- Planning and testing backup helps against failure of media, operating system, software and any other kind of failures that cause a serious data crash.
- It determines the speed and success of the recovery.
- Physical backup extracts data from physical storage (usually from disk to tape). Operating system is an example of physical backup.
- Logical backup extracts data using SQL from the database and store it in a binary file.
- Logical backup is used to restore the database objects into the database. So the logical backup utilities allow DBA (Database Administrator) to back up and recover selected objects within the database.

Causes of Database Failures

- A database includes a huge amount of data and transaction.
- If the system crashes or failure occurs, then it is very difficult to recover the database.

Q17. Explain the causes of failures.

Ans :

1. System Crash

- System crash occurs when there is a hardware or software failure or external factors like a power failure.
- 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

- The transaction failure is affected on only few tables or processes because of logical errors in the code.
- This failure occurs when there are system errors like deadlock or unavailability of system resources to execute the transaction.

3. Network Failure

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

4. Disk Failure

- 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

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

Q17. Explain the Database Recovery Techniques ?

(OR)

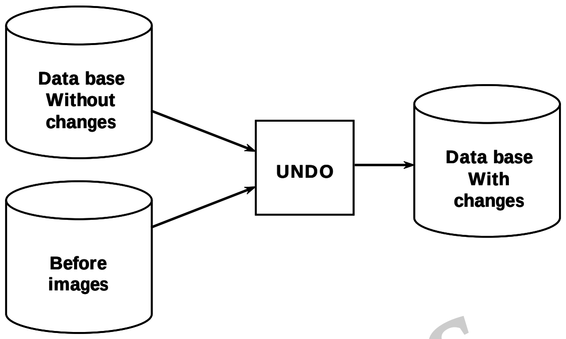
Explain briefly about data base recovery.

Ans : **(July-21, Imp.)**

After going through the types of failures and database errors, let us discuss how to recover from the failures. Recovery can be done using/restoring the previous consistent state (backward recovery) or by moving forward to the next consistent state as per the committed transactions (forward recovery) recovery. Please note that a system can recover from software and hardware failures using the forward and backward recovery only if the system log is intact.

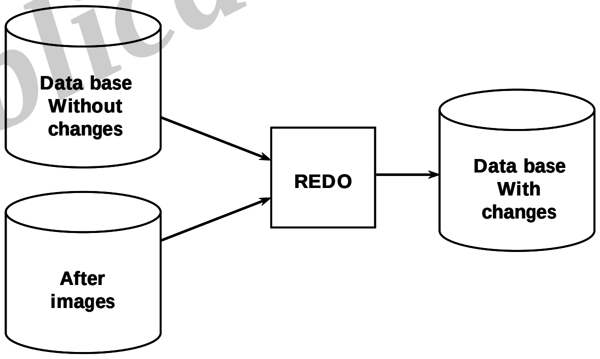
1. Backward Recovery (UNDO)

In this scheme the uncommitted changes made by a transaction to a database are undone. Instead the system is reset to the previous consistent state of database that is free from any errors.



2. Forward Recovery (Redo)

In this scheme the committed changes made by a transaction are reapplied to an earlier copy of the database.



In simpler words, when a particular error in a system is detected, the recovery system makes an accurate assessment of the state of the system and then makes the appropriate adjustment based on the anticipated results -had the system been error free.

One thing to be noted is that the Undo and Redo operations must be idempotent, i.e., executing them several times must be equivalent to executing them once. This characteristic is required to guarantee correct behavior of database even if a failure occurs during the recovery process.

Depending on the above discussed recovery scheme, several types of recovery methods have been used. However, we define the most important recovery schemes used in most of the commercial DBMS.

Log based Recovery

Let us first define the term transaction log in the context of DBMS. A transaction log is a record in DBMS that keeps track of all the transactions of a database system that update any data values in the database. A log contains the following information about a transaction

- A transaction begin marker
- The transaction identification: The transaction id, terminal id or user id etc.
- The operations being performed by the transaction such as update, delete, insert.
- The data items or objects that are affected by the transaction including name of the table, row number and column number.
- The before or previous values (also called UNDO values) and after or changed values (also called REDO values) of the data items that have been updated.
- A pointer to the next transaction log record, if needed.
- The COMMIT marker of the transaction.

4.11 Security & Integrity

Q18. Describe about Database Security and Integrity

Ans : **(June-18)**

Data base security is the protection of information that is maintained in a database. It deals with ensuring only the "right people" get the right to access the "right data". By right people we mean those people who have the right to access/update the data that they are requesting to access/update with the database. This should also ensure the confidentiality of the data. For example, in an educational institution, information about a student's grades should be made available only to that student, whereas only the university authorities should be able to update that information. Similarly, personal information of the employees should be accessible only to the authorities concerned and not to everyone. Another example can be the medical records of patients in a hospital. These should be accessible only to health care officials.

Thus, one of the concepts of database security is primarily a specification of access rules about who has what type of access to what information. This is also known as the problem of Authorisation. These access rules are defined at the time database is defined.

The person who writes access rules is called the authorizer. The process of ensuring that information and other protected objects are accessed only in authorized ways is called access control. There may be other forms of security relating to physical, operating system, communication aspects of databases. However, in this unit, we will confine ourselves mainly to authorisation and access control using simple commands.

The term integrity is also applied to data and to the mechanism that helps to ensure its consistency. Integrity refers to the avoidance of accidental loss of consistency. Protection of database contents from unauthorized access includes legal and ethical issues, organization policies as well as database management policies. To protect database several levels of security measures are maintained:

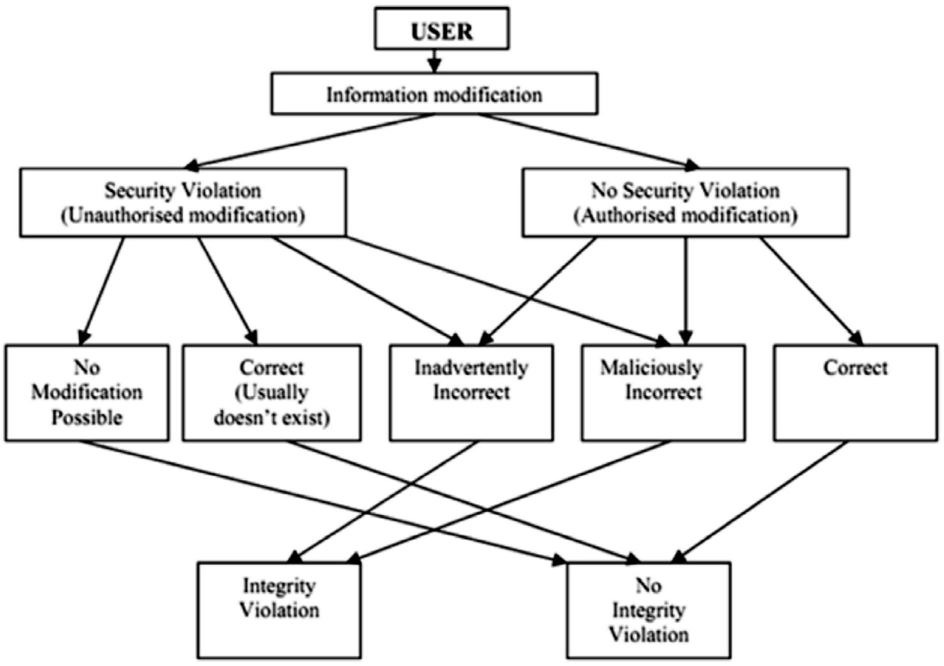
1. **Physical:** The site or sites containing the computer system must be physically secured against illegal entry of unauthorized persons.
2. **Human**: An Authorisation is given to a user to reduce the chance of any information leakage and unwanted manipulations.
3. **Operating System:** Even though foolproof security measures are taken to secure database systems, weakness in the operating system security may serve as a means of unauthorised access to the database.
4. **Network:** Since databases allow distributed or remote access through terminals or network, software level security within the network software is an important issue.
5. **Database system:** The data items in a database need a fine level of access control. For example, a user may only be allowed to read a data item and is allowed to issue queries but would not be allowed to deliberately modify the data. It is the responsibility of the database system to ensure that these access restrictions are not violated.

Q19. What is the relation between Database integrity and security ?

Ans :

Relationship between Security and Integrity

Database security usually refers to access, whereas database integrity refers to avoidance of accidental loss of consistency. But generally, the turning point or the dividing line between security and integrity is not always clear. *Figure* shows the relationship between data security and integrity



4.12 Database Security - Authorization

Q20. Explain in detail about Database security ? and How authorization process helps you to provide security for the database ?

(OR)

Explain briefly about Data base security.

Ans : **(July-21, Oct.-19, Imp.)**

DB2 database and functions can be managed by two different modes of security controls:

1. Authentication
2. Authorization

1. Authentication

Authentication is the process of confirming that a user logs in only in accordance with the rights to perform the activities he is authorized to perform. User authentication can be performed at operating system level or database level itself. By using authentication tools for biometrics such as retina and figure prints are in use to keep the database from hackers or malicious users.

The database security can be managed from outside the db2 database system. Here are some type of security authentication process:

- Based on Operating System authentications.
- Lightweight Directory Access Protocol (LDAP)

For DB2, the security service is a part of operating system as a separate product. For Authentication, it requires two different credentials, those are user id or username, and password.

2. Authorization

You can access the DB2 Database and its functionality within the DB2 database system, which is managed by the DB2 Database manager. Authorization is a process managed by the DB2 Database manager. The manager obtains information about the current authenticated user, that indicates which database operation the user can perform or access.

Here are different ways of permissions available for authorization:

(i) Primary permission:

Grants the authorization ID directly.

(ii) Secondary permission:

Grants to the groups and roles if the user is a member

(iii) Public permission:

Grants to all users publicly.

(iv) Context-sensitive permission:

Grants to the trusted context role.

Authorization can be given to users based on the categories below:

- System-level authorization
- System administrator [SYSADM]
- System Control [SYSCTRL]
- System maintenance [SYSMAINT]
- System monitor [SYSMON]

Authorities provide of control over instance-level functionality. Authority provide to group privileges, to control maintenance and authority operations. For instance, database and database objects.

- Database-level authorization
- Security Administrator [SECADM]
- Database Administrator [DBADM]
- Access Control [ACCESSCTRL]
- Data access [DATAACCESS]
- SQL administrator. [SQLADM]
- Workload management administrator [WLMADM]
- Explain [EXPLAIN]

Authorities provide controls within the database. Other authorities for database include with LDAD and CONNECT.

- **Object-Level Authorization**:

Object-Level authorization involves verifying privileges when an operation is performed on an object.

- **Content-based Authorization**:

User can have read and write access to individual rows and columns on a particular table using Label-based access Control [LBAC].

DB2 tables and configuration files are used to record the permissions associated with authorization names. When a user tries to access the data, the recorded permissions verify the following permissions:

- Authorization name of the user
- Which group belongs to the user
- Which roles are granted directly to the user or indirectly to a group
- Permissions acquired through a trusted context.

While working with the SQL statements, the DB2 authorization model considers the combination of the following permissions:

- Permissions granted to the primary authorzation ID associated with the SQL statements.
- Secondary authorization IDs associated with the SQL statements.
- Granted to PUBLIC
- Granted to the trusted context role.

Short Question & Answers

1. Define Transaction

Ans :

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Let's take an example of a simple transaction. Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

A's Account

Open_Account(A)

Old_Balance = A.balance

New_Balance = Old_Balance - 500

A.balance = New_Balance

Close_Account(A)

B's Account

Open_Account(B)

Old_Balance = B.balance

New_Balance = Old_Balance + 500

B.balance = New_Balance

Close_Account(B)

2. ACID Properties

Ans :

A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability " commonly known as ACID properties " in order to ensure accuracy, completeness, and data integrity.

- **Atomicity:** This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
- **Consistency:** The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
- **Durability:** The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.
- **Isolation:** In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

3 Define Time stamp-based Protocols

Ans :

The most commonly used concurrency protocol is the timestamp based protocol. This protocol uses either system time or logical counter as a timestamp.

Lock-based protocols manage the order between the conflicting pairs among transactions at the time of execution, whereas timestamp-based protocols start working as soon as a transaction is created.

Every transaction has a timestamp associated with it, and the ordering is determined by the age of the transaction. A transaction created at 0002 clock time would be older than all other transactions that come after it. For example, any transaction 'y' entering the system at 0004 is two seconds younger and the priority would be given to the older one.

In addition, every data item is given the latest read and write-timestamp. This lets the system know when the last 'read and write' operation was performed on the data item.

4. Define Deadlock

Ans :

A deadlock is a condition where two or more transactions are waiting indefinitely for one another to give up locks. Deadlock is said to be one of the most feared complications in DBMS as 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.

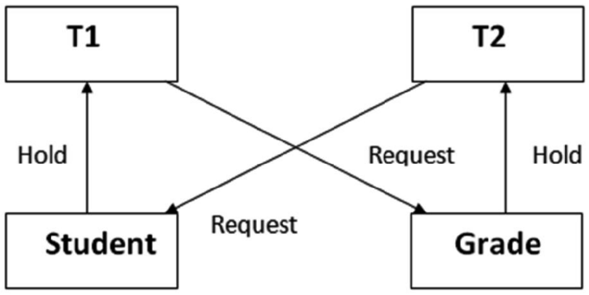


Fig. : Deadlock in DBMS

5. What is Error ? Explain the types of Database Errors ?

Ans :

An error is said to have occurred if the execution of a command to manipulate the database cannot be successfully completed either due to inconsistent data or due to state of program. Forexample, there may be a command in program to store data in database. On the execution of command, it is found that there is no space/place in database to accommodate that additional data. Then it can be said that an error has occurred. This error is due to the physical state of database storage.

Broadly errors are classified into the following categories:

(i) User error

This includes errors in the program (e.g., Logical errors) as well as errors made by online users of database. These types of errors can be avoided by applying some check conditions in programs or by limiting the access rights of online users e.g., read only. So only updating or insertion operation require appropriate check routines that perform appropriate checks on the data being entered or modified. In case of an error, some prompts can be passed to user to enable him/her to correct that error.

(ii) Consistency error

These errors occur due to the inconsistent state of database caused may be due to wrong execution of commands or in caseof a transaction abort. To overcome these errors the database system should include routines that check for the consistency of data entered in the database.

(iii) System error

These include errors in database system or the OS, e.g., deadlocks. Such errors are fairly hard to detect and require reprogramming the erroneous components of the system software.

Database errors can result from failure or can cause failure and thus will require recovery. However, one of the main tasks of database system designers is to make sure that errors minimized. These concepts are also related to database integrity and have also been discusses in a later section.

6. How can we Backup our Database ?

Ans :

Database Backup

- Database Backup is storage of data that means the copy of the data.
- It is a safeguard against unexpected data loss and application errors.
- It protects the database against data loss.
- If the original data is lost, then using the backup it can reconstructed.

The backups are divided into two types,

1. Physical Backup
2. Logical Backup

1. Physical backups

- Physical Backups are the backups of the physical files used in storing and recovering your database, such as datafiles, control files and archived redo logs, log files.
- It is a copy of files storing database information to some other location, such as disk, some offline storage like magnetic tape.
- Physical backups are the foundation of the recovery mechanism in the database.
- Physical backup provides the minute details about the transaction and modification to the database.

2. Logical backup

- Logical Backup contains logical data which is extracted from a database.
- It includes backup of logical data like views, procedures, functions, tables, etc.
- It is a useful supplement to physical backups in many circumstances but not a sufficient protection against data loss without physical backups, because logical backup provides only structural information.

7. Define authentication and authorization.

Ans :

Authentication

Authentication is the process of confirming that a user logs in only in accordance with the rights to perform the activities he is authorized to perform. User authentication can be performed at operating system level or database level itself. By using authentication tools for biometrics such as retina and figure prints are in use to keep the database from hackers or malicious users.

The database security can be managed from outside the db2 database system. Here are some type of security authentication process:

- Based on Operating System authentications.
- Lightweight Directory Access Protocol (LDAP)

For DB2, the security service is a part of operating system as a separate product. For Authentication, it requires two different credentials, those are userid or username, and password.

Authorization

You can access the DB2 Database and its functionality within the DB2 database system, which is managed by the DB2 Database manager. Authorization is a process managed by the DB2 Database manager. The manager obtains information about the current authenticated user, that indicates which database operation the user can perform or access.

8. Write short notes on database security.

Ans :

Database security contain policies and mechanisms to protect the data and ensure that it is not accessed, altered, or deleted without proper authorization. Database security method focus on preventing unauthorized users from accessing the database. DBMS allows the users to access and manipulate database easily, thereby opening doors for intruders. However, most of the DBMSs include security features that allow only authorized persons or programs to access data and then restrict the types of processing that can be accomplished once the access is made.

9. What is optimistic concurrency control?

Ans :

If read only transactions are executed without employing any of the concurrency control mechanisms, then the result generated is in inconsistent state. However, if concurrency control schemes are used then the execution of transactions may be delayed and overhead may be resulted. To avoid such issue, optimistic concurrency control mechanism is used that reduces the execution overhead. But the problem in reducing the overhead is that, prior knowledge regarding the conflicting transactions will not be known. Therefore, a mechanism called "monitoring" the system is required to gain such knowledge. Let us, consider that every transaction T_A is executed in two or three-phase during its life-time.

10. Differentiate between authorization and authentication.

Ans :

	Authorization		Authentication
1.	Authorization is a technique that helps in identifying the resources which can be accessed by an authenticated user.	1.	Authentication is a technique that verifies the identity of the person who is interested in using the system.
2.	The term authorization specifies granting permission.	2.	The term authentication specifies restrictions.
3.	It depends on authentication.	3.	It does not depend on authorization.
4.	Threats and attacks in authorization are Data tampering, Luring attacks, Token stealing, Elevation of privilege and Disclosure of confidential data.	4.	Threats and attacks in authentication are Network Eavesdropping. Brute force attacks, Dictionary attacks, credential theft and cookie reply attacks.

Choose the Correct Answers

1. In ACID Properties, I- Stands for ______. [b]

   (a) Investigation (b) Isolation

   (c) Insulation (d) None

2. 2-phase locking protocol contains ______ phase ? [c]

   (a) Growing (b) Shrinking

   (c) A & B (d) None

3. ______ is a common problem in multiprocessing where many processes share a specific type of mutually Exclusive Resouce. [b]

   (a) Locking (b) Deadlock

   (c) Recovary (d) Time stamping

4. Find the Reason for DataBase failure? [c]

   (a) Software failure (b) Hardware failure

   (c) External failure (d) All

5. In this state the transaction is being Executed. [a]

   (a) Active (b) Commited

   (c) Aborted (d) Failed

6. If transaction Executes all its Operations Successfully it is said to be ______. [b]

   (a) Active (b) Commited

   (c) Aborted (d) Completed

7. A chronological Execution Sequence of transaction is called [b]

   (a) Concurrency (b) Schedule

   (c) Equivalence (d) All

8. Backward Recovery is also called as [a]

   (a) Undo (b) Redo

   (c) A & B (d) None

9. Find which is a main Reason for DataBase loss. [c]

   (a) Theft of Information (b) Unauthorized access

   (c) Unauthorized destruction of Data (d) All

10. ______ is a process of converting plain text into cipher text. [a]

    (a) Encryption (b) Decription

    (c) Authorization (d) All

Fill in the blanks

1. _______ is a collection of actions that transforms the DB from one consistent state to other.
2. DBMS has to Ensure that the _______ properties are fulfilled for any transactions.
3. Two schedules are _______ if transactions in both schedules perform similes actions in similer manner.
4. _______ is a process of managing simultaneous Execution of transactions an a shared DataBase.
5. In ACID properties, D stands for _______.
6. Specify any 2 methods of view Access _______.
7. Data loss can be _______.
8. Database backup can be either _______.
9. _______ the properly in which 2 (or) more computing process are Executing at the same time.
10. _______ is a collection & Rows and columns.

Answers

1. Transaction
2. ACID
3. View Eqialence
4. Concurrency control
5. Durability
6. Read Authorization, Insert Authorization
7. Partial, Total
8. Physical Logical
9. Concurrency
10. Table

One Mark Answers

1. Deadlock

Ans :

Deadlock refers to a particular situation where 2 (or) more process are waiting for resources in a circular chain. And It is a common problem in a multiprocessing where many processes share a specific type of mutually Exclusive Resource.

2. Functional Dependency

Ans :

It is most commonly used authentication scheme. In this. The user is asked to enter the username and password to log-in into the Data Base. Then DBMS verifies allows the user to access.

3. View

Ans :

A View is a mean of providing a user with personalized model of the Database. It is also a useful way of limiting user's access to various portions of DB.

4. Database - Security

Ans :

Database security is the protection of Information that is maintained in a Database. It deals with Ensuring only the Right people to get Right Information.

5. Time Stamping

Ans :

It is a concurrency control mechanism that Eliminates Dead lock. This method doesn't use locks to control concurrency, so It is Impossible for deadlock to occur.

6. Define Lock Granularity.

Ans :

Lock granularity refers to a process that specifies the level of the lock that is being used. The following are the different levels of lock in the databases,

1. Database level
2. Table level
3. Page level
4. Row level
5. Field level.

7. What is access control?

Ans :

Access control is a way of prohibiting uanuthorized users from data accessing.

8. What is full backup?

Ans :

In full backup level all the contents of the database are copied to permanent storage device. Examples: Disks, tapes etc.

9. What is system crash?

Ans :

System crash is a condition in which a program halts the execution of the functions and also stops responding to the request made by the other parts of system.

UNIT V

DISTRIBUTED AND CLIENT SERVER DATABASES

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.

ADVANCED TOPICS: Overview: Parallel Database - Multimedia Database - Mobile Database - Web Database - Multidimensional Database. Data Warehouse - OLTP Vs OLAP - NoSQL Database.

5.1 Distributed Database

Q1. Explain briefly about Distributed Database ?

Ans : **(Imp.)**

A distributed database is a database that consists of two or more files located in different sites either on the same network or on entirely different networks. Portions of the database are stored in multiple physical locations and processing is distributed among multiple database nodes.

A distributed database is basically a database that is not limited to one system, it is spread over different sites, i.e, on multiple computers or over a network of computers. A distributed database system is located on various sited that don't share physical components. This maybe required when a particular database needs to be accessed by various users globally. It needs to be managed such that for the users it looks like one single database.

- Distributed database is a system in which storage devices are not connected to a common 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.
- Shared nothing architecture is used in distributed databases.

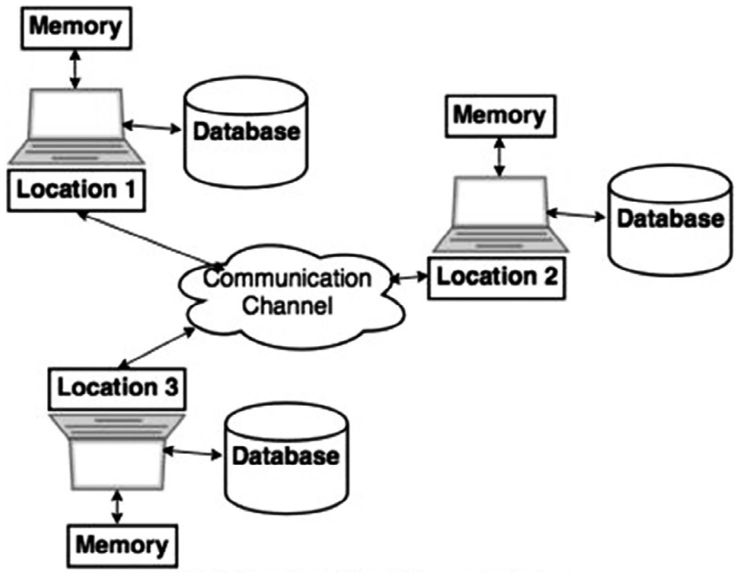


Fig.: Distributeed Database System

- 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.

Goals of Distributed Database system

The concept of distributed database was built with a goal to improve:

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.

Types of Distributed Databases

Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments, each with further subdivisions, as shown in the following illustration.

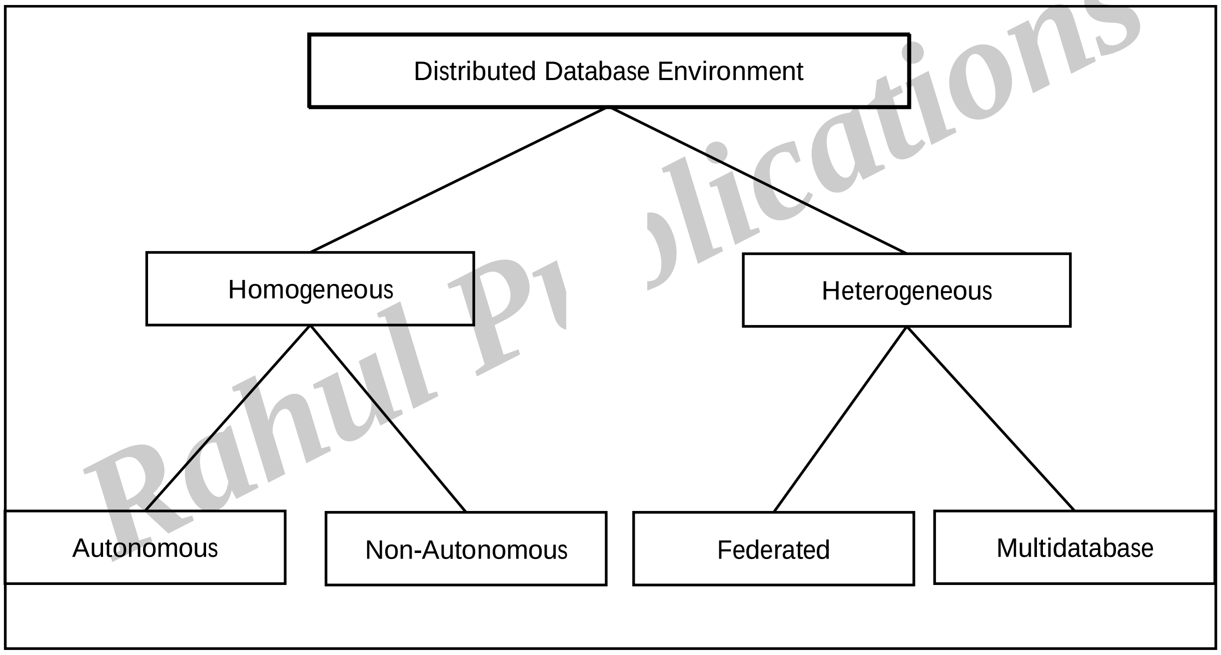


Fig.: Distributed Database Environment

A) Homogeneous Distributed Databases

In a homogeneous distributed database, all the sites use identical DBMS and operating systems. Its properties are:

- The sites use very similar software.
- The sites use identical DBMS or DBMS from the same vendor.
- Each site is aware of all other sites and cooperates with other sites to process user requests.
- The database is accessed through a single interface as if it is a single database.

Types of Homogeneous Distributed Database

There are two types of homogeneous distributed database:

i) **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.

ii) **Non-autonomous:** Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

B) Heterogeneous Distributed Databases

In a heterogeneous distributed database, different sites have different operating systems, DBMS products and data models. Its properties are

- Different sites use dissimilar schemes and software.
- The system may be composed of a variety of DBMSs like relational, network, hierarchical or object oriented.
- Query processing is complex due to dissimilar schemas.
- Transaction processing is complex due to dissimilar software.
- A site may not be aware of other sites and so there is limited co-operation in processing user requests.

Types of Heterogeneous Distributed Databases

i) **Federated:** The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.

ii) **Un-federated:** The database systems employ a central coordinating module through which the databases are accessed.

Q2. List out the characteristics of DDBMS?

Ans :

Characteristics of Distributed Database Management Systems:

A DDBMS governs the storage and processing of logically related data over interconnected computer systems in which both data and processing functions are distributed among several sites. A DBMS must have at least the following functions to be classified as distributed:

- Application interface to interact with the end user, application programs, and other DBMSs within the distributed database.
- Validation to analyze data requests for syntax correctness.
- Transformation to decompose complex requests into atomic data request components.
- Query optimization to find the best access strategy. (Which database fragments must be accessed by the query, and how must data updates, if any, be synchronized?)
- Mapping to determine the data location of local and remote fragments.
- I/O interface to read or write data from or to permanent local storage.
- Formatting to prepare the data for presentation to the end user or to an application program.
- Security to provide data privacy at both local and remote databases.
- Backup and recovery to ensure the availability and recoverability of the database in case of a failure.
- Backup and recovery to ensure the availability and recoverability of the database in case of a failure.
- DB administration features for the database administrator.
- DB administration features for the database administrator.
- Concurrency control to manage simultaneous data access and to ensure data consistency across database fragments in the DDBMS.
- Concurrency control to manage simultaneous data access and to ensure data consistency across database fragments in the DDBMS.
- Transaction management to ensure that the data moves from one consistent state to another. This activity includes the synchronization of local and remote transactions as well as transactions across multiple distributed segments.

Q3. What are the basic components of DDBMS ?

Ans :

1. **Computer work stations:** DDBMS must be independent of the computer system hardware.
2. **Network hard ware & software components :** This is present in each work station . The network components allows all sites to internet & exchange data.

   Because the components like computers, as network hard ware & so on to be supplied by different vendors so that's why DDBMS functions can be run on multiple platforms.
3. **Communication media :** This carry the data from one workstation to another. The DDBMS must be communications media independent.

   It able to support several types of communication media.
4. **Transaction processor (TP):** It is a software component found in each computer that requests data.

   TP receive & process the applications data requests.

   TP also known as application processor (AP) or the Transaction manager (TM).
5. **Data processor (Dp) :** It is a software component found in each computer that stores & retrieves data located at the size.

   DP also known as data manager (DM).

   DTP & TPS can be added to the system without affecting the operation of the other components.

5.1.1 Need Of Distributed Data Bases System

Q4. What is the need of Distributed Data Base ?

(OR)

Explain the Need for distributed data base system.

Ans : **(Oct.-19, Imp.)**

Distributed databases offer some key advantages over centralized databases. Many companies are switching to distributed databases (in which the database, as its name implies, is distributed throughout an array of servers in various locations), for a variety of reasons.

There are several reasons for developing distributed databases. Some of them includes the following,

1. **Sharing Data**

   The major advantage in building a distributed database system is the provision of an environment where users at one site may be able to access the data residing at other sites. For instance, in a distributed banking system, where each branch stores data present at another branch. Without this capability, a user wishing to transfer funds from one branch to another would have to resort to some external mechanism that connect existing systems.
2. **Organizational and Economic Motivation**

   Distributed organizational structure plays a vital role while designing decentralized organizations. A distributed database consumes less cost when compared to the large and centralized databases. Hence, organizations and economy-of-scale are the basic factors considered while developing distributed databases.

3. Connecting Existing Databases with Each Other

!f a large number of databases exist in a particular organization, then a distributed database approach is required. Because of these databases, global applications need to be performed. In this situation, the distributed database is developed by employing bottom-up approach and initiating from the already available local databases. This process needs less amount of local restructuring, when compared to the construction of new centralized databases.

4. Smooth Incremental Growth

Smooth incremental growth is supported by distributed databases when new, autonomous units are added so as to expand the organization. This growth, however, doesn't have much impact on the previously existing organizational units. But, in the centralized databases the growth has a large impact both on the new applications and also on the already existing units. The initial dimensions of the system are responsible for managing the future expansions, which would otherwise be difficult to predict and at the same time would be expensive to implement.

5. Decreased Communication Overhead

Distributed databases consist of more communication overhead when compared to centralized databases. The communication overhead in distributed databases can be reduced by localizing the applications (i.e., every application can access its own database). Therefore, the main goal while designing a distributed database is to maximize the level of localization.

5.1.2 Structure of Distributed Data Base

Q5. Explain logical structure of Distributed Data Base.

Ans : **(July-21, Imp.)**

A Distributed Database Management System (DDBMS) is basically a software, which is used for managing the distributed databases. This system consist of multiple DBMSs, each of which are locally executed. The connection between these DBMSs is done using message handling mechanism. The major component of DDBMS is data dictionary, which contains information for managing,

1. Data
2. Data location
3. Data replication and
4. Data fragmentation.

The purpose of data dictionary is to provide relevant information about location and replication during query processing. It provides the information by assuring that the updates have been transferred to the desired locations. The data dictionary can be managed at centralized location or at distributed locations. However, to obtain complete data dictionary, all the distributed subsets of the dictionary must be integrated. DDBMS provides a provision of user interaction. This interaction is done by means of transactions. Transaction is the mechanism of executing programs. DDBMS transactions consists of multiple processes, each of which are controlled by independent software modules.

Every process involved in the transaction is referred to as an agent. It is possible that a transaction may require a single agent or multiple agents. In the former case, the transaction is referred as local transaction and in the latter, it is referred as global transaction. The purpose of an agent is to access only the data which is under the control of its local data management software. Basically, the execution of a transaction requires an initiating agent which can activate agents of other sites by means of a request sent so as to get access over the required data.

5.2 Advantages and Disadvantages of DDBMS

Q6. List out the advantages and disadvantages of DDBMS.

Ans : **(Oct.-20, June-19, Imp.)**

Advantages

1. **Data are located near the greatest demand site:** The data in a distributed database system are dispersed to match business requirements which reduce the cost of data access.
2. **Faster data access:** End users often work with only a locally stored subset of the company's data.
3. **Faster data processing:** A distributed database system spreads out the systems workload by processing data at several sites.
4. **Growth facilitation:** New sites can be added to the network without affecting the operations of other sites.
5. **Improved communications:** Because local sites are smaller and located closer to customers, local sites foster better communication among departments and between customers and company staff.
6. **Reduced operating costs:** It is more cost-effective to add workstations to a network than to update a mainframe system. Development work is done more cheaply and more quickly on low-cost PCs than on mainframes.
7. **User-friendly interface:** PCs and workstations are usually equipped with an easy-to-use graphical user interface (GUI). The GUI simplifies training and use for end users.
8. **Less danger of a single-point failure:** When one of the computers fails, the workload is picked up by other workstations. Data are also distributed at multiple sites.
9. **Processor independence:** The end user is able to access any available copy of the data, and an end user's request is processed by any processor at the data location.

Disadvantages

1. **Complexity of management and control:** Applications must recognize data location, and they must be able to stitch together data from various sites. Database administrators must have the ability to coordinate database activities to prevent database degradation due to data anomalies.
2. **Technological difficulty:** Data integrity, transaction management, concurrency control, security, backup, recovery, query optimization, access path selection, and so on, must all be addressed and resolved.
3. **Security:** The probability of security lapses increases when data are located at multiple sites. The responsibility of data management will be shared by different people at several sites.
4. **Lack of standards:** There are no standard communication protocols at the database level. (Although TCP/IP is the de facto standard at the network level, there is no standard at the application level.) For example, different database vendors employ different - and often incompatible - techniques to manage the distribution of data and processing in a DDBMS environment.
5. **Increased storage and infrastructure requirements:** Multiple copies of data are required at different sites, thus requiring additional disk storage space.
6. **Increased training cost:** Training costs are generally higher in a distributed model than they would be in a centralized model, sometimes even to the extent of offsetting operational and hardware savings.

5.2.1 Data Distribution

Q7. What is meant by Data Distribution ? Explain the characteristics of Data distribution.

Ans : **(Imp.)**

In distributed database system, the database is shared on several computers. The computers in a distributed system communicate with one another through various communication media, such as high-speed networks or telephone lines. They do not share main memory or disks.

The computers in distributed system may vary in size and function, ranging from workstations up to mainframe systems.

The computers in distributed system are referred to by a number of different names, such as Sites or Nodes depending on the context in which they are mentioned.

A distributed database system consists of single logical database which is split into different fragments. Each fragment is stored on one or more computers under the control of separate DBMS with computers connected by communication network. Each site is capable of independently processing user requests that require access to local data or file system and is also capable of performing processing on remote machines in the network.

Characteristics

- Data set can be split in to fragments and can be distributed across different nodes within network.
- Individual data fragments can be replicated and allocated across different nodes.
- Data at each site is under control of a DBMS.
- DBMS at each site can handle local applications autonomously.
- Each DBMS site will participate in at least one global application.

5.2.1.1 Advantages of Data Distribution

Q8. List out the advantages of Data distribution.

Ans : **(Imp.)**

Distributed databases basically provide us the advantages of distributed computing to the database management domain. Basically, we can define a Distributed database as a collection of multiple interrelated databases distributed over a computer network and a distributed database management system as a software system that basically manages a distributed database while making the distribution transparent to the user.

Distributed database management basically proposed for the various reason from organizational decentralization and economical processing to greater autonomy.

Some of these advantages are as follows:

1. **Management of data with different level of transparency:**

   Ideally, a database should be distribution transparent in the sense of hiding the details of where each file is physically stored within the system. The following types of transparencies are basically possible in the distributed database system:

   - **Network transparency**

     This basically refers to the freedom for the user from the operational details of the network. These are of two types Location and naming transparency.

   - **Replication transparencies**

     It basically made user unaware of the existence of copies as we know that copies of data may be stored at multiple sites for better availability performance and reliability.

   - **Fragmentation transparency**

     It basically made user unaware about the existence of fragments it may be the vertical fragment or horizontal fragmentation.

2. **Increased Reliability and availability :**

   Reliability is basically defined as the probability that a system is running at a certain time whereas Availability is defined as the probability that the system is continuously available during a time interval. When the data and DBMS software are distributed over several sites one site may fail while other sites continue to operate and we are not able to only access the data that exist at the failed site and this basically leads to improvement in reliability and availability.

3. Easier Expansion

In a distributed environment expansion of the system in terms of adding more data, increasing database sizes, or adding more data, increasing database sizes or adding more processor is much easier.

4. Improved Performance

We can achieve interquery and intraquery parallelism by executing multiple queries at different sites by breaking up a query into a number of subqueries that basically executes in parallel which basically leads to improvement in performance.

5.2.1.2 Disadvantages of Data Distribution

Q9. List out disadvantages of Data Distribution.

Ans :

The added complexity required to ensure proper co-ordination among the sites, is the major disadvantage. This increased complexity takes various forms :

- **Software Development Cost :** It is more difficult to implement a distributed database system; thus it is more costly.
- **Greater Potential for Bugs :** Since the sites that constitute the distributed database system operate parallel, it is harder to ensure the correctness of algorithms, especially operation during failures of part of the system, and recovery from failures. The potential exists for extremely subtle bugs.
- **increased Processing Overhead :** The exchange of information and additional computation required to achieve intersite co-ordination are a form of overhead that does not arise in centralized system.

5.2.2 Data Replication

Q10. Define data replication? Explain the types of data replication.

Ans : **(Imp.)**

Data Replication is the process of storing data in more than one site or node. It is useful in improving the availability of data. It is simply copying data from a database from one server to another server so that all the users can share the same data without any inconsistency.

Data Replication is the process of storing data in more than one site or node. It is useful in improving the availability of data. It is simply copying data from a database from one server to another server so that all the users can share the same data without any inconsistency. The result is a distributed database in which users can access data relevant to their tasks without interfering with the work of others.

Data replication encompasses duplication of transactions on an ongoing basis, so that the replicate is in a consistently updated state and synchronized with the source. However in data replication data is available at different locations, but a particular relation has to reside at only one location.

There can be full replication, in which the whole database is stored at every site. There can also be partial replication, in which some frequently used fragment of the database are replicated and others are not replicated.

Types of Data Replication

1. **Transactional Replication:** In Transactional replication users receive full initial copies of the database and then receive updates as data changes. Data is copied in real time from the publisher to the receiving database (subscriber) in the same order as they occur with the publisher therefore in this type of replication, transactional consistency is guaranteed. Transactional replication is typically used in server-to-server environments. It does not simply copy the data changes, but rather consistently and accurately replicates each change.
2. **Snapshot Replication :** Snapshot replication distributes data exactly as it appears at a specific moment in time does not monitor for updates to the data. The entire snapshot is generated and sent to Users. Snapshot replication is generally used when data changes are infrequent. It is bit slower than transactional because on each attempt it moves multiple records from one end to the other end. Snapshot replication is a good way to perform initial synchronization between the publisher and the subscriber.

3. **Merge Replication :** Data from two or more databases is combined into a single database. Merge replication is the most complex type of replication because it allows both publisher and subscriber to independently make changes to the database. Merge replication is typically used in server-to-client environments. It allows changes to be sent from one publisher to multiple subscribers.

Replication Schemes

1. **Full Replication :** The most extreme case is replication of the whole database at every site in the distributed system. This will improve the availability of the system because the system can continue to operate as long as atleast one site is up.

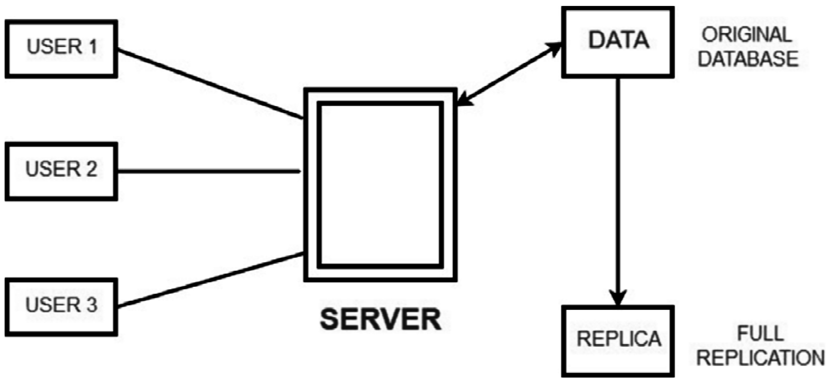


Fig.: Full Replication

Advantages of full replication

- High Availability of Data.
- Improves the performance for retrieval of global queries as the result can be obtained locally from any of the local site.
- Faster execution of Queries.

Disadvantages of full replication

- Concurrency is difficult to achieve in full replication.
- Slow update process as a single update must be performed at different databases to keep the copies consistent.

2. **No Replication :** The other case of replication involves having No replication – that is, each fragment is stored at only one site.

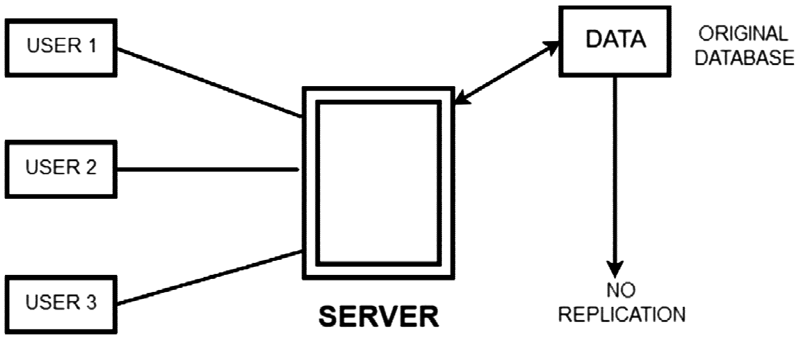


Fig.: No Replication

Advantages of No replication

- The data can be easily recovered.
- Concurrency can be achieved in no replication.

Disadvantages of No replication

- Since multiple users are accessing the same server, it may slow down the execution of queries.
- The data is not easily available as there is no replication.

3. **Partial Replication :** In this type of replication some fragments of the database may be replicated whereas others may not. The number of copies of the fragment may range from one to the total number of sites in the distributed system. The description of replication of fragments is sometimes called the replication schema.

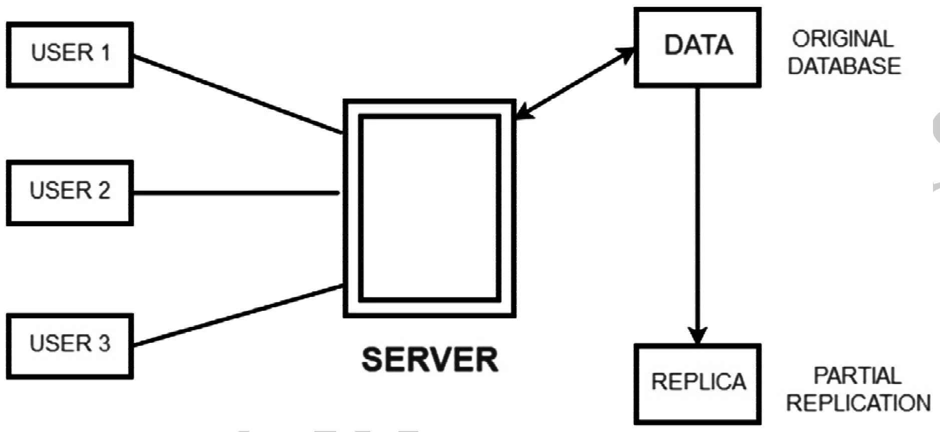


Fig.: Partial Replication

Advantages of Partial replication

- The number of copies of the fragment depends upon the importance of data.

ADVANTAGES OF DATA REPLICATION – Data Replication is generally performed to:

- To provide a consistent copy of data across all the database nodes.
- To increase the availability of data.
- The reliability of data is increased through data replication.
- Data Replication supports multiple users and gives high performance.
- To remove any data redundancy, the databases are merged and slave databases are updated with outdated or incomplete data.
- Since replicas are created there are chances that the data is found itself where the transaction is executing which reduces the data movement.
- To perform faster execution of queries.

Disadvantages of Data Replication

- More storage space is needed as storing the replicas of same data at different sites consumes more space.
- Data Replication becomes expensive when the replicas at all different sites need to be updated.
- Maintaining Data consistency at all different sites involves complex measures.

5.2.3 Data Fragmentation

Q11. Define Data Fragmentation ? Explain the advantages and disadvantages of fragmentation.

Ans :

Meaning

Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called fragments. Fragmentation can be of three types: horizontal, vertical, and hybrid (combination of horizontal and vertical). Horizontal fragmentation can further be classified into two techniques: primary horizontal fragmentation and derived horizontal fragmentation.

Fragmentation should be done in a way so that the original table can be reconstructed from the fragments. This is needed so that the original table can be reconstructed from the fragments whenever required. This requirement is called “reconstructiveness”.

Advantages

➢ Since data is stored close to the site of usage, efficiency of the database system is increased.

➢ Local query optimization techniques are sufficient for most queries since data is locally available.

➢ Since irrelevant data is not available at the sites, security and privacy of the database system can be maintained.

Disadvantages

➢ When data from different fragments are required, the access speeds may be very high.

➢ In case of recursive fragmentations, the job of reconstruction will need expensive techniques.

➢ Lack of back-up copies of data in different sites may render the database ineffective in case of failure of a site.

Q12. Explain different types of fragmentation.

Ans :

1. **Vertical Fragmentation**

   In vertical fragmentation, the fields or columns of a table are grouped into fragments. In order to maintain reconstructiveness, each fragment should contain the primary key field(s) of the table. Vertical fragmentation can be used to enforce privacy of data.

   For example, let us consider that a University database keeps records of all registered students in a Student table having the following schema.

   STUDENT

   Regd_No Name Course Address Semester Fees Marks

   Now, the fees details are maintained in the accounts section. In this case, the designer will fragment the database as follows:

   CREATE TABLE STD_FEES AS

   SELECT Regd_No,Fees

   FROM STUDENT

2. Horizontal Fragmentation

Horizontal fragmentation groups the tuples of a table in accordance to values of one or more fields. Horizontal fragmentation should also confirm to the rule of reconstructiveness. Each horizontal fragment must have all columns of the original base table.

Regd_No	Name	Course	Address	Semester	Fees	Marks

For example, in the student schema, if the details of all students of Computer Science Course needs to be maintained at the School of Computer Science, then the designer will horizontally fragment the database as follows:

CREATE COMP_STD AS

SELECT * FROM STUDENT

WHERE COURSE =”Computer Science”;

3. Hybrid Fragmentation :

In hybrid fragmentation, a combination of horizontal and vertical fragmentation techniques are used. This is the most flexible fragmentation technique since it generates fragments with minimal extraneous information. However, reconstruction of the original table is often an expensive task.

Hybrid fragmentation can be done in two alternative ways.

- At first, generate a set of horizontal fragments; then generate vertical fragments from one or more of the horizontal fragments.
- At first, generate a set of vertical fragments; then generate horizontal fragments from one or more of the vertical fragments.

5.3 Client Server Databases

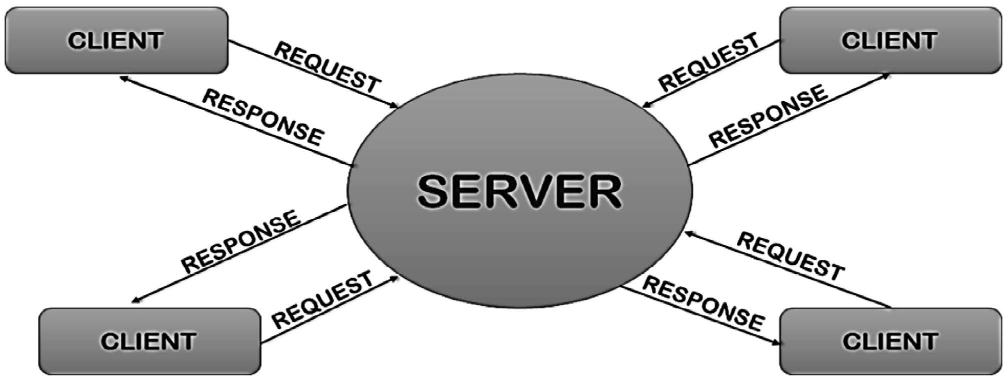
Q13. Explain briefly about Client and server database ?

Ans :

Client-Server Architecture is a distributed system architecture where the workload of client server are separated. Clients are those who request for the services or resources and Server means the resource provider. The server host several programs at its end for sharing resources to its clients whenever requested. Client and server can be on the same system or may be in a network.

Client Server architecture is centralised resource system where Server contain all the resources. The server is highly secured and scalable to respond clients. Client/Server Architecture is Service Oriented Architecture that means client service will never be disrupted.

Client/Server Architecture reduced network traffic by responding to the queries of the clients instead of complete file transfer. It replaced the file server with database server. RDBMS is used by the server to answer client’s request directly.



Client/Server architecture of database system has two logical components namely client, and server. Clients are generally personal computers or workstations whereas server is large workstations, mini range computer system or a mainframe computer system. The applications and tools of DBMS run on one or more client platforms, while the DBMS software reside on the server. The server computer is called back end and the client's computer is called front end.

These server and client computers are connected into a network. The applications and tools act as clients of the DBMS, making requests for its services. The DBMS, in turn, processes these requests and returns the results to the client(s). Client/Server architecture handles the Graphical User Interface (GUI) and does computations and other programming of interest to the end user. The server handles parts of the job that are common to many clients, for example, database access and updates.

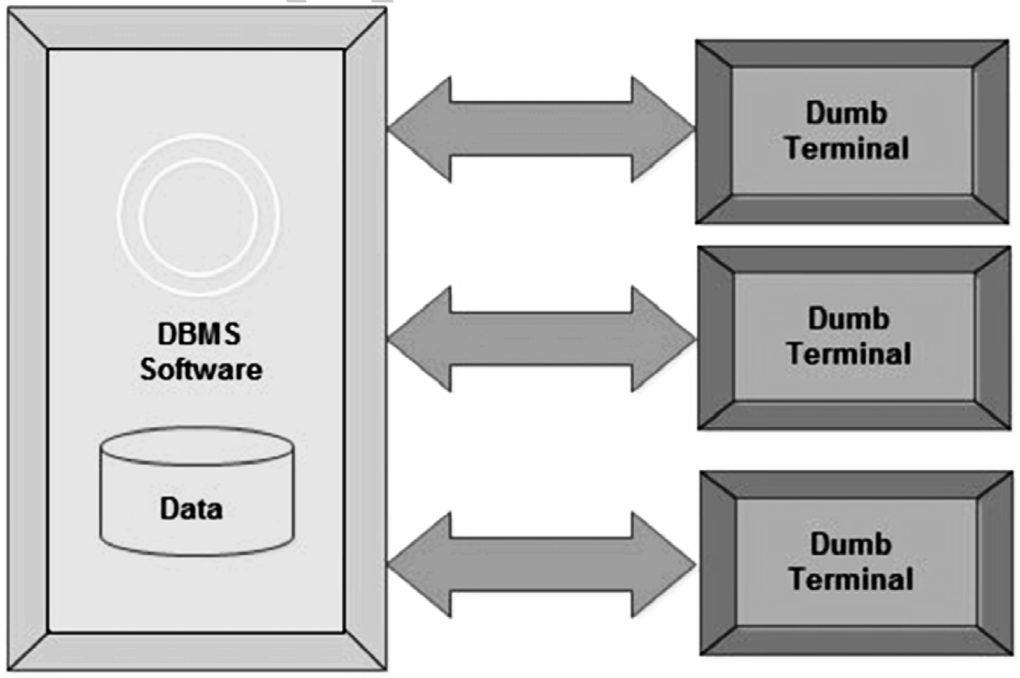


Fig.: Single tier Client Server Computing Model

Types of Client Server Architecture :

- 1 tier architecture
- 2 tier architecture
- 3 tier architecture

5.3.1 Emergence of Client server architecture

Q14. What are the contemporary technologies required to set-up a distributed environment by using client, server model?

Ans :

Some of the pioneering work that was done by some of the relational databases vendors allowed the computing to be distributed on multiple computers on network using contemporary technologies involving.

- Low cost, High performance PCs and Servers
- Graphical user Interface (GUI)
- Open systems
- Object Orientations
- Work group computing
- EDI and Email
- Relational Databases
- Networking and data communications

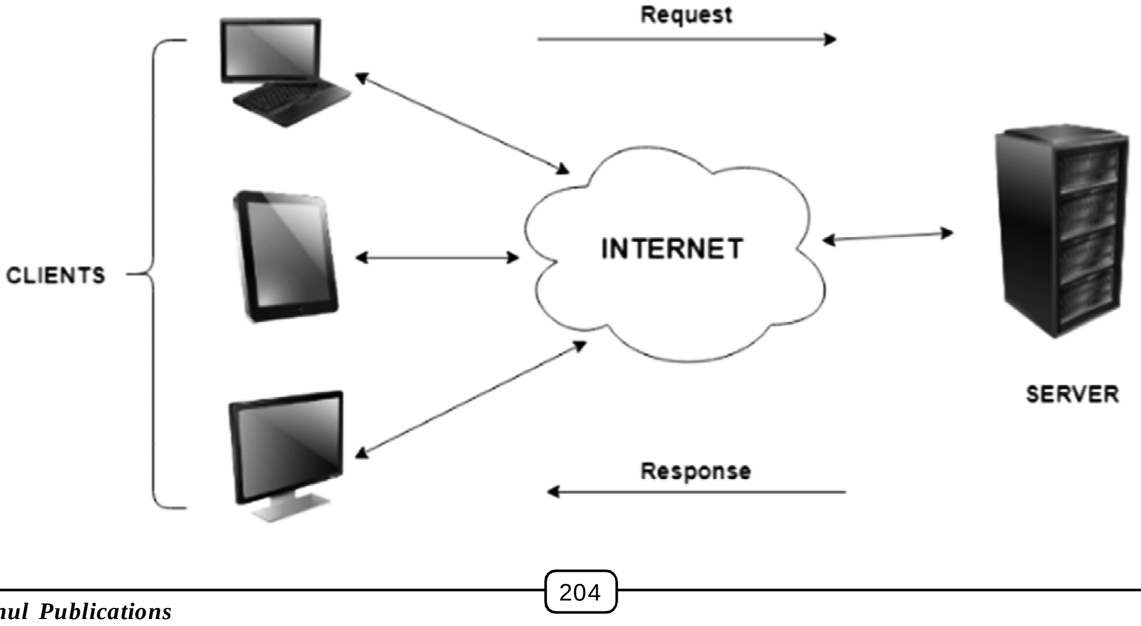
5.3.2 Need for client server computing

Q15. What is the need of client server computing ? Explain its basic characteristics and advantages and disadvantages of client server computing.

Ans : **(July-21,Oct.-20, June-18, Imp.)**

In client server computing, the clients requests a resource and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network but sometimes they may reside in the same system.

An illustration of the client server system is given as follows:



Characteristics

The salient points for client server computing are as follows:

- The client server computing works with a system of request and response. The client sends a request to the server and the server responds with the desired information.
- The client and server should follow a common communication protocol so they can easily interact with each other. All the communication protocols are available at the application layer.
- A server can only accommodate a limited number of client requests at a time. So it uses a system based to priority to respond to the requests.
- Denial of Service attacks hindera servers ability to respond to authentic client requests by inundating it with false requests.
- An example of a client server computing system is a web server. It returns the web pages to the clients that requested them.

Advantages

The different advantages of client server computing are:

- All the required data is concentrated in a single place i.e. the server. So it is easy to protect the data and provide authorisation and authentication.
- The server need not be located physically close to the clients. Yet the data can be accessed efficiently.
- It is easy to replace, upgrade or relocate the nodes in the client server model because all the nodes are independent and request data only from the server.
- All the nodes i.e clients and server may not be build on similar platforms yet they can easily facilitate the transfer of data.

Disadvantages

The different disadvantages of client server computing are:

- If all the clients simultaneously request data from the server, it may get overloaded. This may lead to congestion in the network.
- If the server fails for any reason, then none of the requests of the clients can be fulfilled. This leads of failure of the client server network.
- The cost of setting and maintaining a client server model are quite high.

5.3.3 Structure of client server system and its advantages

Q16. Explain the structure of client server system and list out its advantages and disadvantages.

(OR)

Explain client server system.

Ans : **(Oct.-19, June-19, Imp.)**

The client-server model describes how a server provides resources and services to one or more clients. Examples of servers include web servers, mail servers, and file servers. Each of these servers provide resources to client devices, such as desktop computers, laptops, tablets, and smartphones. Most servers have a one-to-many relationship with clients, meaning a single server can provide resources to multiple clients at one time.

When a client requests a connection to a server, the server can either accept or reject the connection. If the connection is accepted, the server establishes and maintains a connection with the client over a specific protocol. For example, an email client may request an SMTP connection to a mail server in order to send a message. The SMTP application on the mail server will then request authentication from the client, such as the email address and password. If these credentials match an account on the mail server, the server will send the email to the intended recipient.

Which has following structures

i) Two Tier client/server Model

ii) Three Tier client/Server Model

i) Two-Tier Architecture

The two-tier is based on Client Server architecture. The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server. Because of tight coupling a 2 tiered application will run faster.

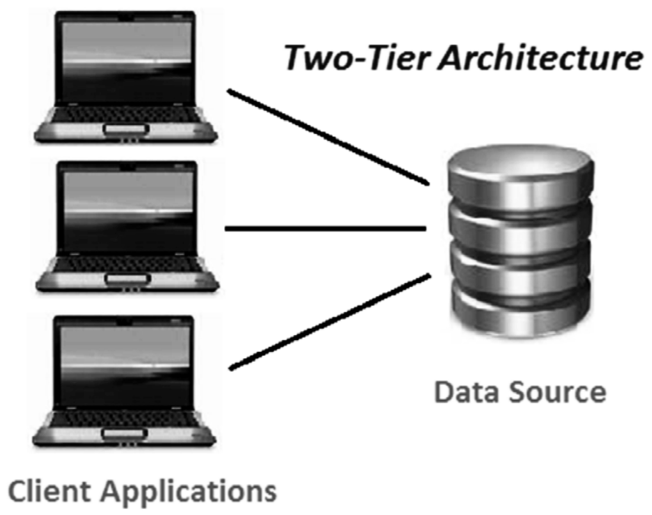


Fig.: Two Tier Architecture

The above figure shows the architecture of two-tier. Here the direct communication between client and server, there is no intermediate between client and server.

Let's take a look of real life example of Railway Reservation two-tier architecture:

Let's consider that first Person is making Railway Reservation for Mumbai to Delhi by Mumbai Express at Counter No. 1 and at same time second Person is also try to make Railway reservation of Mumbai to Delhi from Counter No. 2

If staff from Counter No. 1 is searching for availability into system & at the same staff from Counter No. 2 is also looking for availability of ticket for same day then in this case there is might be good change of confusion and chaos occurs. There might be chance of lock the Railway reservation that reserves the first.

But reservations can be making anywhere from the India, then how it is handled?

So here if there is difference of micro seconds for making reservation by staff from Counter No. 1 & 2 then second request is added into queue. So in this case the Staff is entering data to Client Application and reservation request is sent to the database. The database sends back the information/data to the client.

In this application the Staff user is an end user who is using Railway reservation application software. He gives inputs to the application software and it sends requests to Server. So here both Database and Server are incorporated with each other, so this technology is called as "**Client-Server Technology**".

The Two-tier architecture is divided into two parts:

1) Client Application (Client Tier)

2) Database (Data Tier)

On client application side the code is written for saving the data in the SQL server database. Client sends the request to server and it process the request & send back with data. The main problem of two tier architecture is the server cannot respond multiple request same time, as a result it cause a data integrity issue.

Advantages

1. Easy to maintain and modification is bit easy
2. Communication is faster

Disadvantages

1. In two tier architecture application performance will be degrade upon increasing the users.
2. Cost-ineffective

ii) Three-Tier Architecture

Three-tier architecture typically comprise a presentation tier, a business or data access tier, and a data tier. Three layers in the three tier architecture are as follows:

1) Client layer
2) Business layer
3) Data layer

1) Client layer

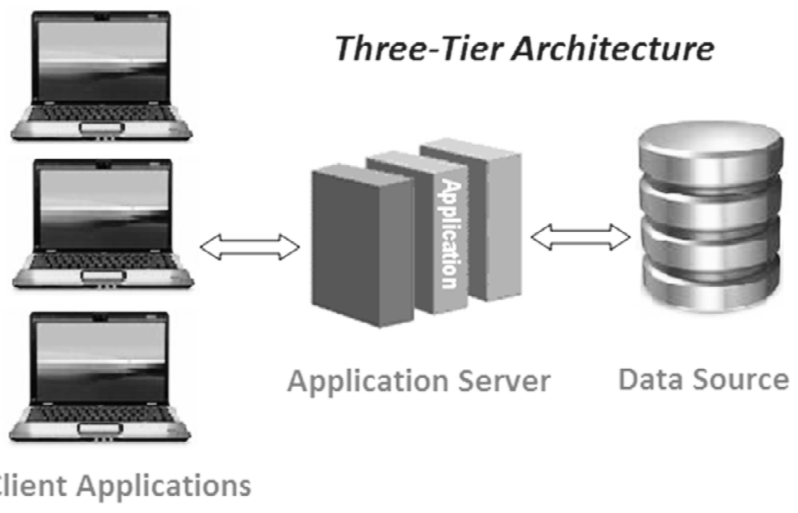
It is also called as Presentation layer which contains UI part of our application. This layer is used for the design purpose where data is presented to the user or input is taken from the user. For example designing registration form which contains text box, label, button etc.

2) Business layer

In this layer all business logic written like validation of data, calculations, data insertion etc. This acts as a interface between Client layer and Data Access Layer. This layer is also called the intermediary layer helps to make communication faster between client and data layer.

3) Data layer

In this layer actual database is comes in the picture. Data Access Layer contains methods to connect with database and to perform insert, update, delete, get data from database based on our input data.



Three-tier Architecture

Advantages

1. High performance, lightweight persistent objects
2. Scalability – Each tier can scale horizontally
3. Performance – Because the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers.
4. High degree of flexibility in deployment platform and configuration
5. Better Re-use
6. Improve Data Integrity
7. Improved Security – Client is not direct access to database.
8. Easy to maintain and modification is bit easy, won't affect other modules
9. In three tier architecture application performance is good.

5.4 Parallel Databases

Q17. What do you mean by parallel databases? What are the typical applications of parallel databases?

Ans : **(Oct.-19)**

In parallel database systems, multiple CPUs work in parallel to improve performance through parallel implementation of various operations such as loading data, building indexes and evaluating queries. Parallel processing divides a large task into many smaller tasks and executes the smaller tasks concurrently on several nodes (CPUs). As a result, the larger tasks completes more quickly. Parallel database systems improve processing and input/output (I/O) speeds by using multiple CPUs and disks working in parallel. Parallel databases are especially useful for applications that have to query large databases and process large number of transactions per second. In parallel processing, many operations are performed simultaneously, as opposed to centralized processing, in which serial computation is performed.

Applications

The goal of parallel database systems is usually to ensure that the database system can continue to perform at an acceptable speed, even as the size of the database and the number of transactions increases. Increasing the capacity of the system by increasing the parallelism provides a smoother path for growth for an enterprise than does replacing a centralized system by a faster machine.

Parallel database systems are usually designed from the ground up to provide best cost-performance and they are quite uniform in site machine (computer) architecture. The cooperation between site machines is usually achieved at the level of the transaction manager module of a database system.

Q18. List advantages and disadvantages of parallel databases?

Ans :

Advantages of Parallel Databases

- Increased throughput (scale-up).
- Improved response time (speed-up).
- Useful for the applications to query extremely large databases and to process an extremely large number of transactions rate (in the order of thousands of transactions per second).
- Substantial performance improvements.
- Increased availability of system.

- Greater flexibility.
- Possible to serve large number of users.

Disadvantages of Parallel Databases

- More start-up costs.
- Interference problem.
- Skew problem.

5.4.1 Architecture of Parallel Databases

Q19. Discuss the architecture of parallel databases. What is shared-memory architecture? Explain with a neat sketch. What are its benefits and limitations?

Ans :

In parallel database architecture, there are multiple central processing units (CPUs) connected to a computer system. There are several architectural models for parallel machines.

- Shared-memory multiple CPU.
- Shared-disk multiple CPU.
- Shared-nothing multiple CPU.

Shared-memory Multiple CPU

In a shared-memory system, a computer has several (multiple) simultaneously active CPUs that are attached to an interconnection network and can share (or access) a single (or global) main memory and a common array of disk storage. Thus, in shared-memory architecture, a single copy of a multithreaded operating system and multithreaded DBMS can support multiple CPUs.

The shared-memory architecture of parallel database system is closest to the traditional single-CPU processor of centralized database systems, but much faster in performance as compared to the single-CPU of the same power. This structure is attractive for achieving moderate parallelism. Many commercial database systems have been ported to shared-memory platforms with relative ease.

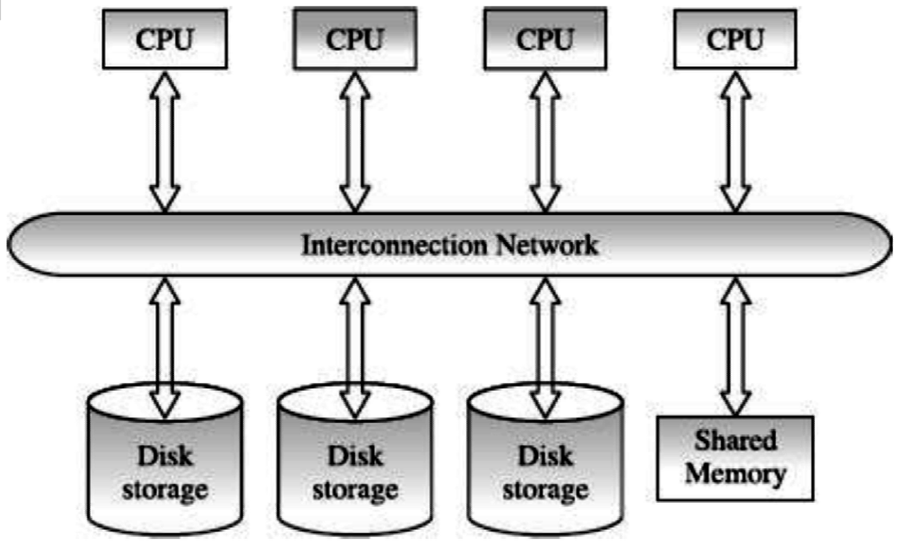


Fig.: Shared Memory Multiple CPU

Benefits of Shared-memory Architecture

- Communication between CPUs is extremely efficient. Data can be accessed by any CPU without being moved with software. A CPU can send messages to the other CPUs much faster by using memory writes, which usually takes less than a microsecond, than by sending a message through a communication mechanism.
- The communication overheads are low, because main memory can be used for this purpose and operating system services can be leveraged to utilize the additional CPUs.

Limitations of Shared-memory Architecture

- Memory access uses a very high-speed mechanism that is difficult to partition without losing efficiency. Thus, the design must take special precautions that the different CPUs have equal access to the common memory. Also, the data retrieved by one CPU should not be unexpectedly modified by another CPU acting in parallel.
- Since the communication bus or interconnection network is shared by all the CPUs, the shared-memory architecture is not scalable beyond 80 or 100 CPUs in parallel. The bus or the interconnection network becomes a bottleneck as the number of CPUs increases.
- The addition of more CPUs causes CPUs to spend time waiting for their turn on the bus to access memory.

Q20. What is shared-disk architecture? Explain with a neat sketch. What are its benefits and limitations?

Ans :

In a shared disk system, multiple CPUs are attached to an interconnection network and each CPU has its own memory but all of them have access to the same disk storage or, more commonly, to a shared array of disks. The scalability of the system is largely determined by the capacity and throughput of the interconnection network mechanism. Since memory is not shared among CPUs, each node has its own copy of the operating system and the DBMS. It is possible that, with the same data accessible to all nodes, two or more nodes may want to read or write the same data at the same time. Therefore, a kind of global (or distributed) locking scheme is required to ensure the preservance of data integrity. Sometimes, the shared-disk architecture is also referred to as a parallel database system.

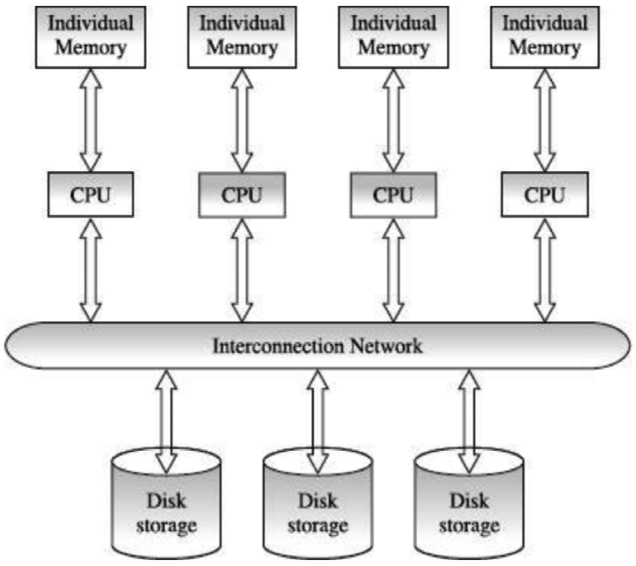


Fig.: Shared Disk Architecture

Benefits of Shared-disk Architecture

- Shared-disk architecture is easy to load-balance, because data does not have to be permanently divided among available CPUs.
- Since each CPU has its own memory, the memory bus is not a bottleneck.
- It offers a low cost solution to provide a degree of fault tolerance. In case of a CPU or memory failure, the other CPUs take over its task; since the database is resident on disks that are accessible form all CPUs.
- It has found acceptance in wide applications.

Limitations of Shared-disk Architecture

- Shared-disk architecture also faces similar problems of interference and memory contention bottleneck as the number of CPUs increases. As more CPUs are added, existing CPUs are slowed down because of the increased contention for memory accesses and network bandwidth.
- Shared-disk architecture also has a problem of scalability. The interconnection to the disk subsystem becomes bottleneck, particularly when the database makes a large number of accesses to the disks.

Q21. What is shared-nothing architecture? Explain the benefits and limitations of shared nothing architecture.

Ans :

In a shared-nothing system, multiple CPUs are attached to an interconnection network through a node and each CPU has a local memory and disk storage, but no two CPUs can access the same disk storage area. All communication between CPUs is through a high-speed interconnection network. Node functions as the server for the data on the disk or disks that the node owns. Thus, shared-nothing environments involve no sharing of memory or disk resources. Each CPU has its own copy of operating system, its own copy of the DBMS, and its own copy of a portion of data managed by the DBMS.

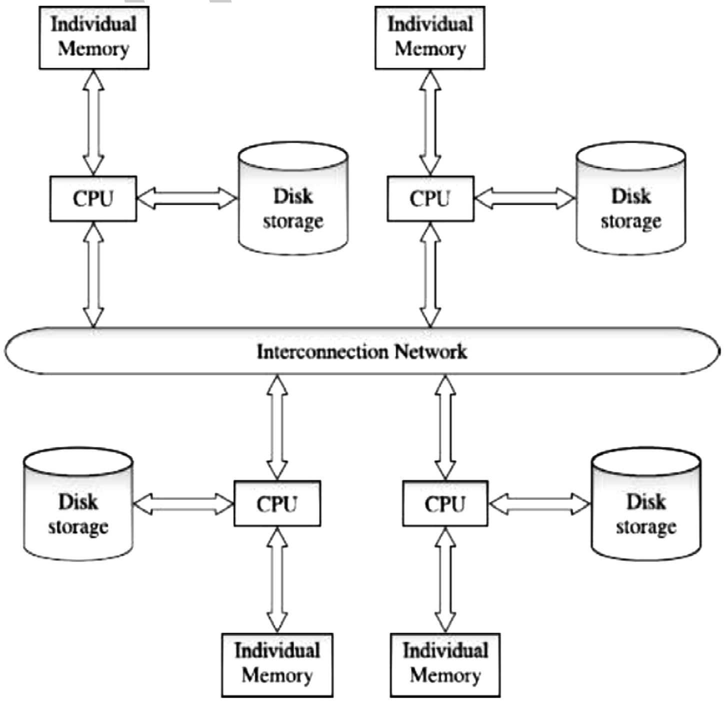


Fig.: Shared Nothing System

CPUs sharing responsibility for database services usually split up the data among themselves. CPUs then perform transactions and queries by dividing up the work and communicating by message over the high-speed network (at the rate of megabits per second).

Benefits of Shared-nothing Architecture

- Shared-nothing architectures minimize contention among CPUs by not sharing resources and therefore offer a high degree of scalability.
- Since local disk references are serviced by local disks at each CPU, the shared-nothing architecture overcomes the limitations of requiring all I/O to go through a single interconnection network. Only queries, accesses to non-local disks and result relations pass through the network.
- The interconnection networks for shared-nothing architectures are usually designed to be scalable. Thus, adding more CPUs and more disks enables the system grow (or scale) in a manner that is proportionate to the power and capacity of the newly added components. This provides for scalability that is nearly linear, enabling users to get a large return on their investment in new hardware (resources). In other words, shared-nothing architecture provides linear *speed-up* and linear *scale-up*.
- Linear speed-up and scale-up properties increase the transmission capacity of shared-nothing architecture as more nodes are added and therefore, it can easily support large number of CPUs.

Limitations of Shared-nothing Architecture

- Shared-nothing architectures are difficult to load-balance. In many multi CPU environments, it is necessary to split the system workload in some way so that all system resources are being used efficiently. Proper splitting or balancing this workload across a shared-nothing system requires an administrator to properly partition or divide the data across the various disks such that each CPU is kept roughly as busy as the others. In practice this is difficult to achieve.
- Adding new CPUs and disks to shared-nothing architecture means that the data may need to be redistributed in order to make advantage of the new hardware (resource) and thus requires more extensive reorganization of the DBMS code.
- The costs of communication and non-local disk access are higher than in shared-disk or shared-memory architecture since sending data involves software interaction at both ends.
- The high-speed networks are limited in size, because of speed-of-light considerations. This leads to the requirement that a parallel architecture has CPUs that are physically close together. This network architecture is also known as local area network (LAN).
- Shared-nothing architectures introduce a single point of failure to the system. Since each CPU manages its own disk(s), data stored on one or more of these disks become inaccessible if its CPU goes down.
- It requires an operating system that is capable of accommodating the heavy amount of messaging required to support inter-processor communications.

5.4.2 I/O Parallelism (Data Partitioning)

Q22. Write short notes on the following:

(i) Hash partitioning.

(ii) Round-robin partitioning.

(iii) Range partitioning.

(iv) Schema partitioning.

Ans :

Input/output (I/O) parallelism is the simplest form of parallelism in which the relations (tables) are partitioned on multiple disks to reduce the retrieval time of relations from disk. In I/O parallelism, the input data is partitioned and then each partition is processed in parallel. The results are combined after the processing of all partitioned data. I/O parallelism is also called data partitioning. The following four types of partitioning techniques can be used:

(i) Hash partitioning.

(ii) Range partitioning.

(iii) Round-robin partitioning.

(iv) Schema partitioning.

(i) Hash Partitioning

In the technique of hash partitioning a hash function is applied to the attribute value whose range is $(0, 1, 2, \ldots, n-1)$. Each tuple (row) of the original relation is hashed on the partitioning attributes. The output of this function causes the data for that tuple to be targeted for placement on a particular disk. For example, let us assume that there are n disks d_1, d_2, d_3,... d_n, across which the data are to be partitioned. Now, if the hash function returns 2, then the tuple is placed on disk d_2.

Advantages

- Hash partitioning has the advantage of providing for even distribution of data across the available disk, helping to prevent skewing.
- Skew can slow the performance caused by one or more CPUs and disks getting more work than others.
- Hash partitioning is best suited for point queries (involving exact matches) based on the partitioning attribute.
- For example, if a relation is partitioned on the employee identification numbers (EMP-ID), then we can answer the query "Find the record of the employee with employee identification number = 106519" using SQL statement as follows:

```
SELECT * FROM EMPLOYEE
  WHERE EMP-ID=106519;
```

Hash partitioning is also useful for sequential scans of the entire relation (table) placed on *n* number of disks. The time taken to scan the relation is approximately 1/n of the time required to scan the relation in a single disk system.

Disadvantages

- Hash partitioning technique is not well suited for point queries on non-partitioning attributes.
- It is also not well suited for answering range queries, since, typically hash functions do not preserve proximity within a range.

(ii) Range Partitioning

In the technique of range partitioning an administrator specifies that attribute-values within a certain range are to be placed on a certain disk. In other words, range partitioning distributes contiguous attribute-value ranges to each disk. For example, range partitioning with three disks numbered as 0, 1, 2, ..., n might place tuples for employee numbers with up to 100000 on disk 0, tuples for employees identification numbers 100001-150000 on disk 1, tuples for employee 150001-200000 on disk 2 and so forth.

Advantages

- Range partitioning involves placing tuples containing attribute values that fall within a certain range on a disk.
- This offers good performance for range-based queries and also provides reasonable performance for exact-match (point) queries involving the partitioning attribute.
- For point queries, the partitioning vector can be used to locate the disk where the tuples reside.
- For range queries, the partitioning vector is used to find the range of disks on which the tuples may reside.
- In both cases, the search narrows to exactly those disks that might have any tuples of interest.

Disadvantages

Range partitioning can cause skewing in some cases. For example, consider an EMPLOYEE relation that is partitioned across disk according to employee identification numbers. If tuples containing numbers 100000 – 150000 are placed on disk 0(d_0) and tuples containing numbers 150001–200000 are placed on disk 1(d_1), data will be evenly distributed if the company employs 200000 employees. However, if the company employs only 160000 employees currently and most are assigned numbers 100000 – 150000, the bulk of the tuples for this relation will be skewed towards disk 0($d0_0$).

(iii) Round-robin Partitioning

In the round-robin partitioning technique, the relations (tables) are scanned in any order and i^{th} tuple is send to disk number di mode n. In other word, disks 'take turns' receiving new tuples of data. For example, a system with n disks would place tuple A on disk 0 (d_0), tuple B on disk 1 (d_1), tuple C on disk 2 (d_2) and so forth. Round-robin technique ensures an even distribution of tuples across disks. That is, each disk has approximately the same number of tuples as the others.

Advantages

Round-robin partitioning is ideally suited for applications that wish to read the entire relation sequentially for each query.

Disadvantages

With round-robin partitioning technique, both point queries and range queries are complicated to process, since each of the n disks must be used for search.

(iv) Schema Partitioning

In schema partitioning technique, different relations (tables) within a database are placed on different disks.

Disadvantages

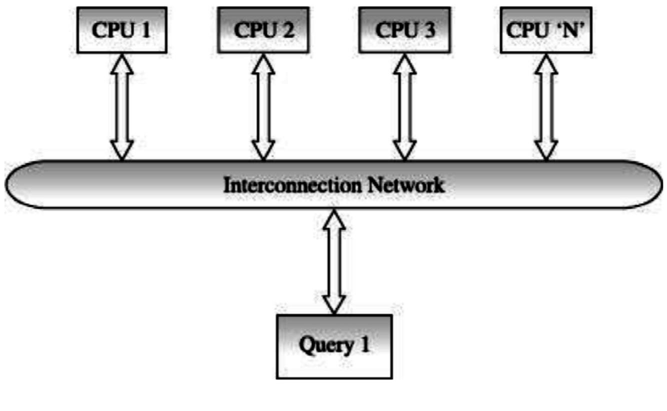
Schema partitioning is more prone to data skewing. Most vendors support schema partitioning along with one or more other techniques.

5.4.3 Intraquery Parallelism

Q23. Write short notes on Intraquery parallelism.

Ans :

Intra-query parallelism refers to the execution of a single query in parallel on multiple CPUs using shared-nothing parallel architecture technique. Intra-query parallelism is sometimes called parallel query processing. For example, suppose that a relation (table) has been partitioned across multiple disks by range partitioning on some attribute and now user wants to 'sort' on the partitioning attribute. The 'sort' operation can be implemented by sorting each partition in parallel, then concatenating the sorted partitions to get the final sorted relation. Thus, a query can be parallelised by parallelising individual operations.



Generally two approaches are used in intra-query parallelism. In the first approach, each CPU can execute the same task against some portion of the data. This approach is the most common approach to parallel query processing in commercial products. In the second approach, the task can be divided into different subtasks with each CPU executing a different subtask. Both approaches presume that the data is portioned across disks in an appropriate manner.

Advantages

- Intra-query parallelism speeds up long-running queries.
- They are beneficial for decision support applications that issue complex, read-only queries, including queries involving multiple joins.

5.4.4 Interquery Parallelism

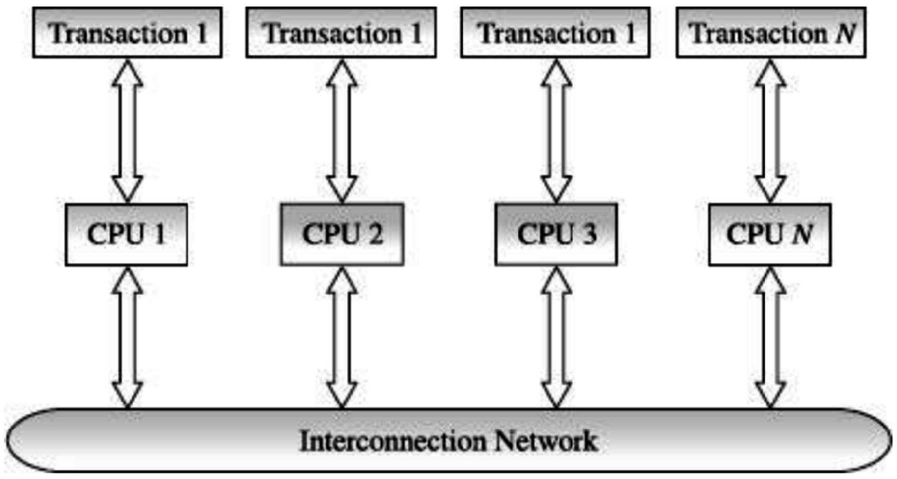
Q24. Write short notes on Interquery parallelism.

Ans :

In an inter-query parallelism, multiple transactions are executed in parallel, one by each (CPU). Inter-query is sometimes also called parallel transaction processing. The primary use of inter-query parallelism is to scale-up a transaction-processing system to support a larger number of transactions per second.

To support inter-query parallelism, the DBMS generally uses means of task or transaction dispatching. This helps to ensure that incoming requests are routed to the least busy processor, enabling the overall workload to be kept balanced. However, it may be difficult to fully automate this process, depending on the underlying hardware architecture of the computer. For example, a shared-nothing architecture dictates that data stored on certain disks be accessible only to certain CPUs. Therefore, requests that involve this data cannot be dispatched to just any CPU.

Efficient lock management is another method used by DBMS to support inter-query parallelism, particularly in shared-disk architecture. Since, in the inter-query parallelism each query is run sequentially, it does not help in speeding up long-running queries. In such cases, the DBMS must understand the locks held by different transactions executing on different CPUs in order to preserve overall data integrity. If memory is shared among CPUs, lock information can be kept in buffers in global memory and updated with little overhead. However, if only disks are shared (and not memory), this lock information must be kept on the only shared resource, that is disk. Inter-query parallelism on shared-disk architecture performs best when transactions that execute in parallel do not access the same data. The Oracle 8 and Oracle Rdb systems are examples of shared-disk parallel database systems that support inter-query parallelism.



Advantages

- Easiest form of parallelism to support in a database system, particularly in shared-memory parallel system.
- Increased transaction throughput.
- It scales up a transaction-processing system to support a larger number of transactions per second.

Disadvantages

- Response times of individual transactions are no faster than they would be if the transactions were run in isolation.
- It is more complicated in a shared-disk or shared-nothing architecture.

5.4.5 Intra-operation Parallelism

Q25. Write a short notes on Intra Operation Parallelism?

Ans :

In intra-operation parallelism, we parallelise the execution of each individual operation of a task, such as sorting, projection, join and so on.

Since the number of operations in a typical query is small, compared to the number of tuples processed by each operation, intra-operation parallelism scales better with increasing parallelism.

Advantages

- Intra-operation parallelism is natural in a database.
- Degree of parallelism is potentially enormous.

5.4.6 Inter-operation Parallelism

Q26. Write a short notes on Inter Operation Parallelism?

Ans :

In inter-operation parallelism, the different operations in a query expression are executed in parallel. The following two types of inter-operation parallelism are used:

- Pipelined parallelism
- Independent parallelism

1. Pipelined Parallelism

In pipelined parallelism, the output tuples of one operation A are consumed by a second operation B, even before the first operation has produced the entire set of tuples in its output. Thus, it is possible to run operations A and B simultaneously on different processors (CPUs), so that operation B consumes tuples in parallel with operation A producing them. The major advantage of pipelined parallelism in a sequential evaluation is that we can carry out a sequence of such operations without writing any of the intermediate results to disk.

Advantages

Pipelined parallelism is useful with a small number of CPUs. Also, pipelined executions avoid writing intermediate results to disk.

Disadvantages

Pipelined parallelism does not scale up well. First, pipeline chains generally do not attain sufficient length to provide a high degree of parallelism. Second, it is not possible to pipeline relational operators that do not produce output until all inputs have been accessed. Third, only marginal speed-up is obtained for the frequent cases in which one operator's execution cost is much higher than are those of the others.

2. Independent Parallelism

In an independent parallelism, the operations in a query expression that do not depend on one another can be executed in parallel.

Advantages

Independent parallelism is useful with a lower degree of parallelism.

Disadvantages

Like pipelined parallelism, independent parallelism does not provide a high degree of parallelism. It is less useful in a highly parallel system.

5.5 Multimedia Database

Q27. What is Multimedia Databases?

Ans : **(Imp.)**

Multimedia Databases are databases that contain and allow key data management operations with multimedia data. Traditional databases contained alphanumeric data and managed it for various applications. Increasingly, applications now contain multimedia data that requires defining additional types and requires development of operations for storage, management, access, and presentation of multimedia data. Multimedia databases must increasingly deal with issues related to managing multimedia data as well as the traditional data. Commonly, databases that manage images, audio, and video in addition to metadata related to these and other alphanumeric information are called multimedia databases. When databases contain only one of the images, audio, or video, they are called image databases, audio databases, and video databases, respectively. Considering the current trend, it is likely that most databases will slowly become multimedia databases.

Content of Multimedia Database management system:

- **Media data:** The actual data representing an object.
- **Media format data:** Information such as sampling rate, resolution, encoding scheme etc. about the format of the media data after it goes through the acquisition, processing and encoding phase.
- **Media keyword data:** Keywords description relating to the generation of data. It is also known as content descriptive data. Example: date, time and place of recording.
- **Media feature data:** Content dependent data such as the distribution of colors, kinds of texture and different shapes present in data.

  There are still many challenges to multimedia databases, some of which are:

- **Modelling:** Working in this area can improve database versus information retrieval techniques thus, documents constitute a specialized area and deserve special consideration.
- **Design:** The conceptual, logical and physical design of multimedia databases has not yet been addressed fully as performance and tuning issues at each level are far more complex as they consist of a variety of formats like JPEG, GIF, PNG, MPEG which is not easy to convert from one form to another.
- **Storage:** Storage of multimedia database on any standard disk presents the problem of representation, compression, mapping to device hierarchies, archiving and buffering during input-output operation. In DBMS, a “BLOB” (Binary Large Object) facility allows untyped bitmaps to be stored and retrieved.
- **Performance:** For an application involving video playback or audio-video synchronization, physical limitations dominate. The use of parallel processing may alleviate some problems but such techniques are not yet fully developed. Apart from this multimedia database consume a lot of processing time as well as bandwidth.
- **Queries and retrieval:** For multimedia data like images, video, audio accessing data through query opens up many issues like efficient query formulation, query execution and optimization which need to be worked upon.

Q28. What are multimedia sources? Explain each one of them.

Ans :

Multimedia databases use wide variety of multimedia sources, such as:

i) Images

ii) Video clips

iii) Audio clips

iv) Text (or) documents

The fundamental characteristics of multimedia systems are that they incorporate continuous media, such as voice (audio), video and animated graphics.

i) Images

Images include photographs, drawings and so on. Images are usually stored in raw form as a set of pixel or cell values, or in a compressed form to save storage space. The image shape descriptor describes the geometric shape of the raw image, which is typically a rectangle of cells of a certain width and height. Each cell contains a pixel value that describes the cell content. In black/white images, pixels can be one bit. In gray scale or colour images, pixel is multiple bits. Images require very large storages space. Hence, they are often stored in a compressed form, such as GIF, JPEG. These compressed forms use various mathematical transformations to reduce the number of cells stored, without disturbing the main image characteristics.

In order to identify the particular objects in an image, the image is divided into two homogeneous segments using a homogeneity predicate. The homogeneity predicate defines the conditions for how to automatically group those cells. Inexpensive image-capture and storage technologies have allowed massive collections of digital images to be created.

ii) Video Clips

Video clippings include movies, newsreels, home videos and so on. A video source is typically represented as a sequence of frames, where each frame is a still image. However, rather than identifying the objects and activities in every individual frame, the video is divided into video segments. Each video segment is made up of a sequence of contiguous frames that includes the same objects or activities. Its starting and ending frames identify each segment. The objects and activities identified in each video segment can be used to index the segments. An indexing technique called frame segment trees are used for video indexing. The index includes both objects (such as persons, houses, cars and others) and activities (such as a person delivering a speech, two persons talking and so on). Videos are also often compressed using standards such as MPEG.

iii) Audio Clips

Audio clips include phone messages, songs, speeches, class presentations, surveillance recording of phone messages and conversations by law enforcement and others. Here, discrete transforms are used to identify the main characteristics of a certain person's voice in order to have similarity based indexing and retrieval Audio characteristic features include loudness, intensity, pitch and clarity.

iv) Text or Documents

Text or document sources include articles, books, journals and so on. A text or document is basically the full text of some article, book, magazine or journal. These sources are typically indexed by identifying the keywords that appear in the text and their relative frequencies. However, filler words are eliminated from that process. Because a technique called singular value decompositions (SVD) based on matrix transformation is used to reduce the number of keywords in collection of document. An indexing technique called telescoping vector trees or TV-trees, can then be used to group similar documents together.

Multimedia Database Queries

Multimedia databases provide features that allow users to store and query different types of multimedia information. As discussed in the previous section, the multimedia information includes images (for example, pictures, photographs, drawings and more), video (for example, movies, newsreels, home videos and others), audio (for example, songs, speeches, phone messages and more) and text or documents (for example, books, journals, articles and others). The main types of multimedia database queries are the ones that help in locating multimedia data containing certain objects of interest.

Q29. What do you mean by contest-based retrieval in multimedia databases?

Ans :

In multimedia databases, content-based queries are widely used. For example, locating multimedia sources that contain certain objects of interest such as locating all video clippings of in a video database that include certain famous hills, say Mount Everest, or retrieving all photographs with the picture of a computer from our photo gallery, or retrieving video clips that contain a certain person from the video database. One may also want to retrieve video clips based on certain activities included in them, for example, video clips of all sixes in cricket test matches. These types of queries are also called content-based retrieval, because the multimedia source is being retrieved based on its containing certain objects or activities.

Content-based retrieval is useful in database applications where the query is semantically of the form, “find objects that look like this one”. Such applications include the following:

- Medical imaging
- Trademarks and copyrights
- Art galleries and museums
- Retailing
- Fashion and fabric design
- Interior design or decorating
- Law enforcement and criminal investigation

Q30. How are multimedia sources identified in multimedia databases? Explain.

Ans :

Multimedia databases use some model to organise and index the multimedia sources based on their contents. Two approaches, namely automatic analysis and manual identification are used for this purpose. In the first approach, an automatic analysis of the multimedia sources is done to identify certain mathematical characteristics of their contents. The automatic analysis approach uses different techniques depending on the type of multimedia source, for example image, video, text or audio. In the second approach, manual identification of the objects and activities of interests is done in each multimedia source. This information is used to index the sources. Manual identification approach can be applied to all the different multimedia sources. However, it requires a manual pre-processing phase where a person has to scan each multimedia source to identify and catalog the objects and activities it contains so that they can be used to index these sources.

A typical image database query would be to find images in the database that are similar to a given image. The given image could be an isolated segment that contains, say, a pattern of interest and the query is to locate other images that contain that same pattern. There are two main techniques for this type of search. The first technique uses distance function to compare the given image with the stored images and their segments. If the distance value returned is small, the probability of match is high. Indexes can be created to group together stored images that are close in the distance metric so as to limit the search space. The second technique is called the transformation approach, which measures image similarity by having a small number of transformations that can transform one image’s cells to match the other image. Transformations include rotations, translations and scaling.

Q31. List Few Characteristics and Operation of MDBMS?

Ans :

- **A MDBMS (Multimedia Database Management System):** can be characterized based on its objectives at the time of handling multimedia objects.
- **Corresponding storage media:** Multimedia data must be stored and managed according to the specific characteristics of the available storage media.
- **Comprehensive search methods:** During a search in the database, an entry, given in the form of text or a graphical image, is found using different search queries and the corresponding search methods.
- **Format independent interface:** database queries should be independent of media format. MDBMS should provide information in formats requested by the application.
- **Simultaneous data access:** The same multimedia data can be accessed (even simultaneously) through different queries by several applications. Hence, consistent access to shared data can be implemented.
- **Management of large amount of data:** The MDBMS must be capable of handling and managing large amounts of data.

- **Long Transaction:** The performance of a transaction in a MDBMS means that transfer of a large amount of data will take a long time and must be done in a reliable manner.
- **Real-time Data:** The read and write operations of continuous data must be done in real-time. The data transfer of continuous data has a higher priority than other database management actions.

Operations of Multimedia Databases

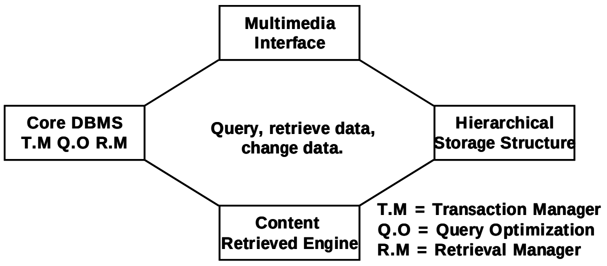
- **Input (insert/record) operation:** The data will be written to the database. The raw and registering data are always needed; descriptive data can be attached later.
- **Output (play) operation:** It involves reading the raw data from the database according to the registered data.
- **Modification:** It involves changing of raw, registering and descriptive data. Modification can also be understood as a data conversion from one format to another.
- **Deletion Operation:** This operation removes an entry from the database. The consistency of the data must be preserved.

Q32. Explain different architecture for content organization in multimedia databases.

Ans :

Architecture of a multi-user database can become complex. It is not clear which architecture would be the best option for a multimedia database. A transaction involving multimedia data will in general be expected to take longer. Locks will have to be maintained for longer periods. MDBMS (Multimedia Database Management System) has formal database architecture. It has a separate user view from the system view

MMDBs require all the basic attributes of a database management system such as a transaction manager, query optimizer, recovery manager etc. as well as special storage structures and specialized search and querying modules.



Since existing relational and OO databases comprise the basic requirements of any database, it is natural that many multimedia and imaging DB applications are constructed within such existing systems. In order to support such applications, many DBMS vendors offer facilities suitable for MM.

These include:

- Long bit and byte strings
- BLOBS
- Paths or references of images where the actual image stored elsewhere, such as on an optical storage subsystem.

The reasons for this are that document imaging systems need on-line, near-line and off-line storage of images, including archiving. This may be achieved by the use of optical jukeboxes but most commercial DBMSs do not directly support optical storage subsystems (Informix Online/optical is an exception).

Content retrieval capabilities. In conventional relational and OO DBs querying is based on the attributes of objects. Information retrieval and document imaging systems require searching the content of documents. This ability can be generalized to still images, audio and video.

Multimedia Database basically constitutes a three layer architecture. A simple representation is shown below:

Interface

The interface between the user and the database is used for the following activities:

- Object browsing
- Compose

Object Composition

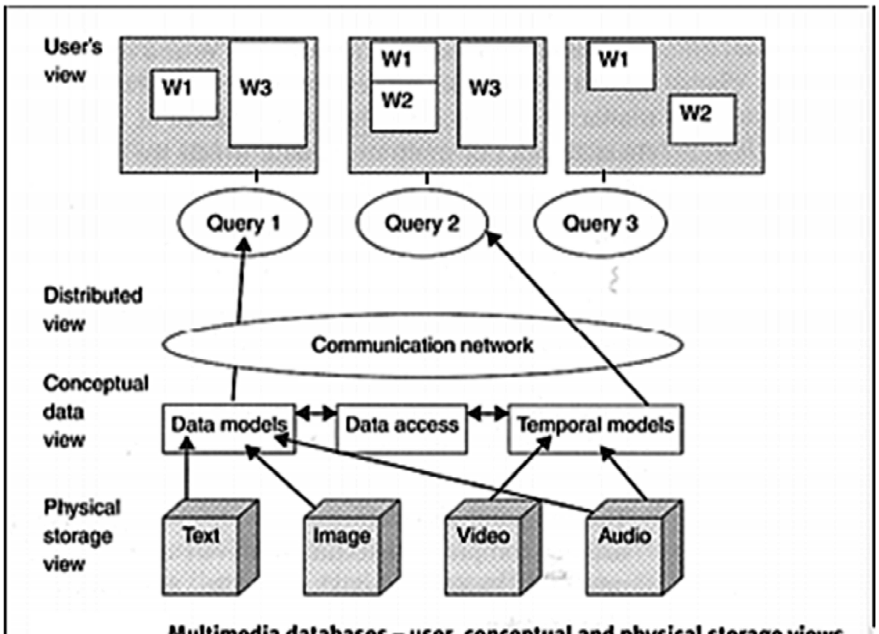
The object composition part of the multimedia database manages the multimedia objects

Storage

Storage functions of multimedia database involve clustering and indexing of multimedia data.

Three-layer architecture

- **Database (Data) Tier:** At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier:** At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
- **User (Presentation) Tier:** End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.



Multimedia databases – user, conceptual and physical storage views

5.5.1 Multimedia Database Applications

Q33. What are the applications of multimedia databases?

Ans :

Multimedia data may be stored, delivered and utilized in many different ways. Some of the important applications are as follows:

- Repository applications.
- Presentation applications.
- Collaborative work using multimedia information.

- Documents and records management.
- Knowledge dissemination.
- Education and training.
- Marketing, advertising, retailing, entertainment and travel.
- Real-time control and monitoring.

5.6 Mobile Database

Q34. Define Mobile Database? Explain the components of Mobile Database.

Ans : **(Imp.)**

Mobile databases are separate from the main database and can easily be transported to various places. Even though they are not connected to the main database, they can still communicate with the database to share and exchange data.

Recent advances in portable and wireless technology led to mobile computing, a new dimension in data communication and processing.

- Portable computing devices coupled with wireless communications allow clients to access data from virtually anywhere and at any time.
- There are a number of hardware and software problems that must be resolved before the capabilities of mobile computing can be fully utilized.
- Some of the software problems – which may involve data management, transaction management, and database recovery – have their origins in distributed database systems.

The mobile database includes the following components

- The main system database that stores all the data and is linked to the mobile database.
- The mobile database that allows users to view information even while on the move. It shares information with the main database.
- The device that uses the mobile database to access data. This device can be a mobile phone, laptop etc.
- A communication link that allows the transfer of data between the mobile database and the main database.

In mobile computing, the problems are more difficult, mainly:

- The limited and intermittent connectivity afforded by wireless communications.
- The limited life of the power supply (battery).
- The changing topology of the network.

Q35. List out Advantages and Disadvantages of MDB?

Ans :

Advantages of Mobile Databases

- The data in a database can be accessed from anywhere using a mobile database. It provides wireless database access.
- The database systems are synchronized using mobile databases and multiple users can access the data with seamless delivery process.
- Mobile databases require very little support and maintenance.
- The mobile database can be synchronized with multiple devices such as mobiles, computer devices, laptops etc.

Disadvantages of Mobile Databases

Some disadvantages of mobile databases are:

- The mobile data is less secure than data that is stored in a conventional stationary database. This presents a security hazard.
- The mobile unit that houses a mobile database may frequently lose power because of limited battery. This should not lead to loss of data in database.

Q36. Explain briefly about Mobile Database Architecture?

Ans :

- The general architecture of a mobile platform is illustrated in Figure.
- It is distributed architecture where a number of computers, generally referred to as Fixed Hosts and Base Stations are interconnected through a high-speed wired network.

- Fixed hosts are general purpose computers configured to manage mobile units.
- Base stations function as gateways to the fixed network for the Mobile Units.

Wireless Communications

- The wireless medium have bandwidth significantly lower than those of a wired network.
- The current generation of wireless technology has data rates range from the tens to hundreds of kilobits per second (2G cellular telephony) to tens of megabits per second (wireless Ethernet, popularly known as WiFi).
- Modern (wired) Ethernet, by comparison, provides data rates on the order of hundreds of megabits per second.

  The other characteristics distinguish wireless connectivity options:
- Interference, locality of access, range,
- Support for packet switching,
- Seamless roaming throughout a geographical region.
- Some wireless networks, such as WiFi and Bluetooth, use unlicensed areas of the frequency spectrum, which may cause interference with other appliances, such as cordless telephones.
- Modern wireless networks can transfer data in units called packets, that are used in wired networks in order to conserve bandwidth.

Client/Network Relationships

- Mobile units can move freely in a geographic mobility domain, an area that is circumscribed by wireless network coverage.
- To manage entire mobility domain is divided into one or more smaller domains, called cells, each of which is supported by at least one base station.
- Mobile units be unrestricted throughout the cells of domain, while maintaining information access contiguity.
- The communication architecture described earlier is designed to give the mobile unit the impression that it is attached to a fixed network, emulating a traditional client-server architecture.
- Wireless communications, however, make other architectures possible. One alternative is a mobile ad-hoc network (MANET).
- In a MANET, co-located mobile units do not need to communicate via a fixed network, but instead, form their own using cost-effective technologies such as Bluetooth.
- In a MANET, mobile units are responsible for routing their own data, effectively acting as base stations as well as clients.
- Moreover, they must be robust enough to handle changes in the network topology, such as the arrival or departure of other mobile units.
- MANET applications can be considered as peer-to-peer, meaning that a mobile unit is simultaneously a client and a server.
- Transaction processing and data consistency control become more difficult since there is no central control in this architecture.
- Resource discovery and data routing by mobile units make computing in a MANET even more complicated.
- Sample MANET applications are multi-user games, shared whiteboard, distributed calendars, and battle information sharing.

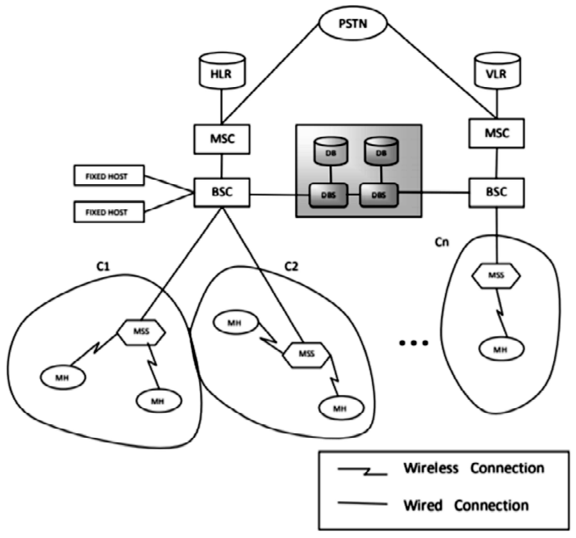


Fig.: Architecture of a Mobile Database System

5.7 Web Database

5.7.1 Internet Databases

Q37. What is Internet database?

Ans :

The Internet revolution of the late 90s have resulted into explosive growth of World Wide Web (WWW) technology and sharply increased direct user access to databases. Organizations converted many of their phone interfaces to databases into Web interfaces and made a variety of services and information available on-line. The transaction requirements of organizations have grown with increasing use of computers and the phenomenal growth in the Web technology. These developments have created many sites with millions of viewers and the increasing amount of data collected from these viewers has produced extremely large databases at many companies.

Internet Technology

As its name suggest, the Internet is not a single homogeneous network. It is an interconnected group of independently managed networks. Each network supports the technical standards needed for inter-connection - the Transmission Control Protocol/Internet Protocol (TCP/IP) family of protocols and a common method for identifying computers - but in many ways the separate networks are very different. The various sections of the Internet use almost every kind of communications channel that can transmit data. They range from fast and reliable to slow and erratic. They are privately owned or operated as public utilities. They are paid for in different ways. The Internet is sometimes called an information highway. A better comparison would be the international transportation system, with everything from airlines to dirt tracks.

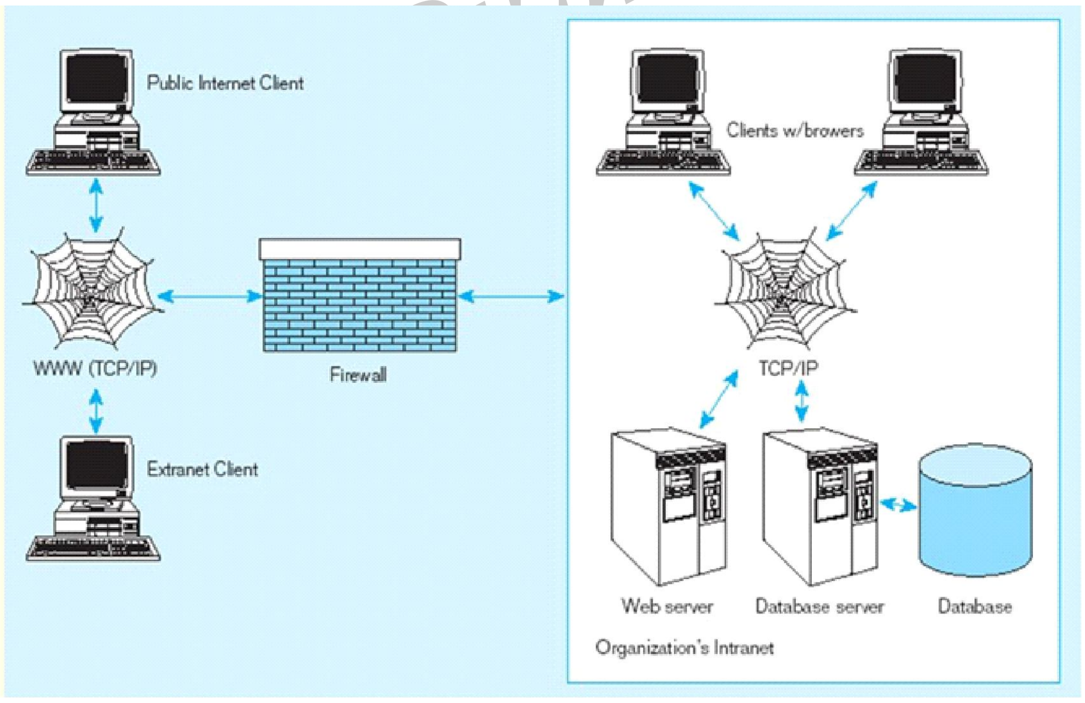


Fig.: Database-Enabled Intranet/Internet Environment

Thus, the Internet may be defined as a network of networks, scattered geographically all over the world. It is a worldwide collection of computer networks connected by communication media that allow users to view and transfer information between computers. Internet is made up of many separate but interconnected networks belonging to commercial, educational and government organisations and Internet Service Providers (ISPs). Thus, the Internet is not a single organization but cooperative efforts by multiple organisations managing a variety of computers and different operating systems.

Q38. What are the available Internet services?

Ans :

Wide varieties of services are available on the Internet.

Category	Services	Description
Communication	Electronic Mail	Electronic messages sent or received from one computer to another, commonly referred to as e-mail.
	Newsgroups	Computer discussion groups where participants with common interests (like hobbies or professional associations) post messages called "articles" that can be read and responded to by other participants around the world via "electronic bulletin boards".
	Mailing Lists	Similar to Newsgroups except participants exchange information via e-mail.
	Chat	Real-time on-line conversations where participants type messages to other chat group participants and receive responses they can read on their screens.
File Access	File Transfer Protocol (FTP)	Sending (uploading) or receiving (down-loading) computer files via the File Transfer Protocol (FTP) communication rules.
Searching Tools	Search Engines	Programs that maintain indices of the contents of files at computers on the Internet. Users can use search engines to find files by searching indices for specific words or phrases.
World Wide Web (WWW)	Web Interfaces	A subset of the Internet using computers called Web servers that store multimedia files, or "pages" that contain text, graphics, video, audio and links to other pages that are accessed by software programs called Web browsers.
E-commerce	Electronic Commerce	Customers can place and pay for orders via the business's Web site.
E-Business	Electronic Business	Complete integration of Internet technology into the economic infrastructure of the business.

Q39. What do you mean by web databases?

Ans :

A web database is a wide term for managing data online. A web database gives you the ability to build your own databases/data storage without you being a database guru or even a technical person.

Examples: banks, airline and rental car reservations, university course registration and so on

- The Web is a distributed information system base on hypertext.
- Most Web documents are hypertext documents formatted via HTML
- HTML Documents contain
- Text along with font specifications, and other formatting instructions
- Hypertext links to other documents, which can be associated with region of the text.

Data Organization

Web databases enable collected data to be organized and cataloged thoroughly within hundreds of parameters. The Web database does not require advanced computer skills, and many database software programs provide an easy "click-and-create" style with no complicated coding. Fill in the fields and save each record. Organize the data however you choose, such as chronologically, alphabetically or by a specific set of parameters.

Web Database Software

Web database software programs are found within desktop publishing programs, such as Microsoft Office Access and Open Office Base. Other programs include the Webex WebOffice database and Form Logix Web database. The most advanced software applications can set up data collection forms, polls, feedback forms and present data analysis in real time.

5.7.2 Features of Web Database

Q40. List out few features of web database?

Ans :

1. **Save Money**

   One of the advantages of online database software is that it can save your business money. When you don't need to buy a software program for your business, this could result in a major savings overall. In most cases, businesses pay for a software program and then pay for a licensing fee for each computer that uses it. Using an online database may prove cheaper, depending on the number of computers you use.

2. **Flexible Use**

   Another benefit of using an online database program is that it allows your business to be flexible. You only pay for the amount of storage that you use. You need not worry about purchasing servers as you go or eliminating them when they are no longer needed. If your business grows or shrinks, you do not need to be concerned about the costs of database management software or servers.

3. **Technical Support**

   Another advantage of using a Web-based database program is that you can shift the technical support burden to someone else. Paying a company for access to an online database includes technical support. If the database has problems, you simply contact the company and the staff handles it. You don't need to pay for an information technology professional for this purpose. If you already have an IT department, your employees can focus on other things.

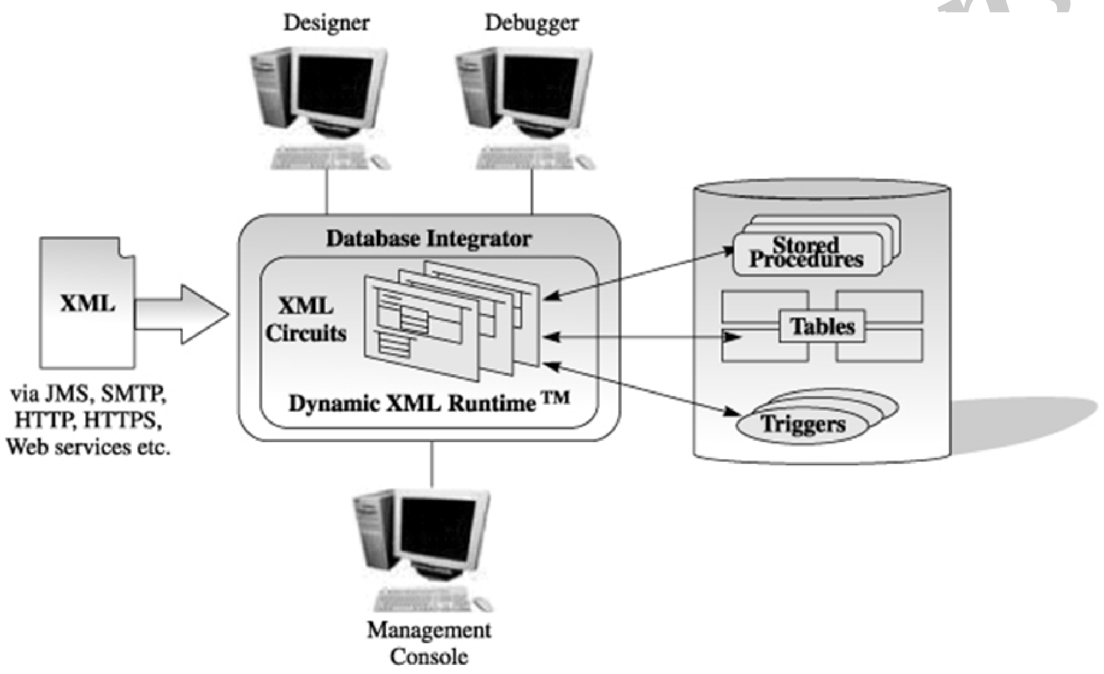
4. Access

Having access to the database at all times from multiple locations is another major advantage of this type of database. With an online database, you could theoretically access the information in the database from any computer. The information is also available 24 hours a day, seven days a week. This means that all employees have access to the same information and can collaborate with one another on projects — regardless of location. This advantage can increase productivity and improve efficiency.

- It's based on a file management system (no actual database)
- It is a table with several million entries, each entry being a keyword and a related keyword, plus metrics that measure the quality of the match (how strong the relationship between the two keywords is), as well as frequencies attached to these two keywords, and when they are jointly found. The function to measure the quality of the match can be customized by the user.

Q41. What are Web database tools? Explain.

Ans :



Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) is a protocol for transferring HTML documents through Internet, between Web browsers and Web servers, such as Web pages and so on. It is a generic object-oriented, stateless protocol to transmit information between servers and clients. In computing, a protocol is a set of rules that are used to send messages between computer systems.

A typical protocol includes description of the formats to be used, the various messages, the sequences in which they should be sent, appropriate responses, error conditions and so on.

In addition to the transfer of a document, HTTP also provides powerful features, such as the ability to execute programs with arguments supplied by the user and deliver the results back as an HTML document. The basic message type in HTTP is 'get'. For example, clicking on the hyperlink with the URL:

http://www.dlib.org/dlib.html

Specifies an HTTP a get command. An informal description of this command is:

- Open a connection between the browser and the Web server that has the domain name "www.dlib.org".
- Copy the file "dlib.html" from the Web server to the browser.
- Close the connection.

Uniform Resource Locator (URL)

Uniform resource locator (URL) is a key component of the Web. It is a globally unique name for each document that can be accessed on the Web. URL provides a simple addressing mechanism that allows the Web to link information on computers all over the world. It is a string of alphanumeric characters that represent the location or address of a resource on the Internet and how that resource should be accessed. URL is a special code to identify each Web page on the World Wide Web. An example of a URL is

http://www.google.co.in/google.html

The Web technology was developed in about 1990 by Tim Berners-Lee and colleagues at CERN, the European research centre for high-energy physics in Switzerland. It was made popular by the creation of a user interface, known as Mosaic, which was developed by Marc Andreessen and others at the University of Illinois, Urbana-Champaign. Mosaic was released in 1993. Within a few years, numerous commercial versions of Mosaic followed. The most widely used are the Netscape Navigator and Microsoft's Internet Explorer. These user interfaces are called Web browsers, or simply browsers.

Features of Web

The basic reason for the success of the Web can be summarized succinctly.

- It provides a convenient way to distribute information over the Internet.
- Individuals can publish information and users can access that information by themselves, with no training and no help from outsiders.
- A small amount of computer knowledge is needed to establish a Web site. It is very easy to use a browser to access the information.

Web Technology

Technically, the Web is based on the following simple techniques:

- Internet service providers (ISPs)
- IP address
- Hypertext Markup Language (HTML)
- Hypertext Transfer Protocol (HTTP)
- Uniform Resource Locators (URLs)
- Multipurpose Internet Mail Extension (MIME) Data Types

Internet Service Providers (ISPs)

Internet service providers (ISPs) are commercial agents who maintain the host computer, serve as gateway to the Internet and provide an electronic mail box with facilities for sending and receiving e-mails. ISPs connect the client computers to the host computers on the Internet. Commercial ISPs usually charge for the access to the Internet and e-mail services. They supply the communication protocols and front-end tools that are needed to access the Internet.

Internet Protocol (IP) Address

All host computers on the Internet are identified by a unique address called IP address. IP address consists of a series of numbers. Computers on the Internet use these IP address numbers to communicate with each other. ISPs provide this IP address so that we can enter it as part of the setup process when we originally setup our communication connection to the ISP.

Hypertext Markup Language (HTML)

Hypertext Markup Language (HTML) is an Internet language for describing the structure and appearance of text documents. It is used to create Web pages stored at web sites. The Web pages can contain text, graphics, video, audio and links to other areas of the same Web page, other Web pages at the same Web site, or to a Web page at a different Web site. These links are called hypertext link and are used to connect Web pages. They allow the user to move from one Web page to another. When a link is clicked with the mouse pointer, another area of same Web page, another Web page at the same Web site, or a Web page at different Web site appears in the browser window.

```
<html>
<head>
<title>D-Lib</title>
</head>

<body>
<h1>D-Lib Magazine</h1>
<img src = "log.gif">

<p> Since the first issue appeared in July 1995,

<a href = "http://www.dlib.org/dlib.html">D-Lip Magazine</a> has appeard monthly as a
compendium of research, news and progress in digital libraries.</p>

<p><i> William Y.Arms
<br>January 1, 1999</i></p>

</body>
</html>
```

Multipurpose Internet Mail Extension (MIME) Data Types

A file of data in a computer is simply a set of bits, but, to be useful the bits need to be interpreted. Thus, in the previous example, in order to display the file "google.html" correctly, the browser must know that it is in the HTML format. The interpretation depends upon the data type of the file. Common data types are "html" for a file of text that is marked-up in HMTL format and "jpeg" for a file that represents an image encoded in the jpeg format.

In the Web and in a wide variety of Internet applications, the data type is specified by a scheme called MIME, also called Internet Media Types. MIME was originally developed to describe information sent by electronic mail. It uses a two part encoding, a generic part and a specific part. Thus text/ascii is the MIME type for text encoded in ASCII, image/jpeg is the type for an image in the jpeg format and text/html is text marked-up with HMTL tags. There is a standard set of MIME types that are used by numerous computer programs and additional data types can be described using experimental tags.

The importance of MIME types in the Web is that the data transmitted by an HTTP get command has a MIME type associated with it. Thus, the file "dlib.html" has the MIME type text/html. When the browser receives a file of this type, it knows that the appropriate way to handle this file is to render it as HTML text and display it in the screen.

Many computer systems use file names as a crude method of recording data types. Thus, some Windows programs use file names that end in ".htm" for file of HMTL data and Unix computers use ".html" for the same purpose. MIME types are a more flexible and systematic method to record and transmit typed data.

Q42. What are the advantages and disadvantages of Web databases?

Ans :

Advantages of Web Databases

- Simple to use HTML both for developers and end-users.
- Platform-independent.
- Good graphical user interface (GUI).
- Standardisation of HTML.

- Cross-platform support.
- Transparent network access.
- Scalable deployment.
- Web enables organisations to provide new and innovative services and reach new customers through globally accessible applications.

Disadvantages of Web Databases

- The internet not yet very reliable.
- Slow communication medium.
- Security concern.
- High cost for meeting increasing demands and expectations of customers.
- Scalability problem due to enormous peak loads.
- Limited functionality of HTML.

5.8 Multidimensional Database

Q43. Explain in detail about Multidimensional Database with a proper example?

Ans :

Multidimensional databases are used mostly for OLAP (online analytical processing) and data warehousing. They can be used to show multiple dimensions of data to users.

A multidimensional database is created from multiple relational databases. While relational databases allow users to access data in the form of queries, the multidimensional databases allow users to ask analytical questions related to business or market trends.

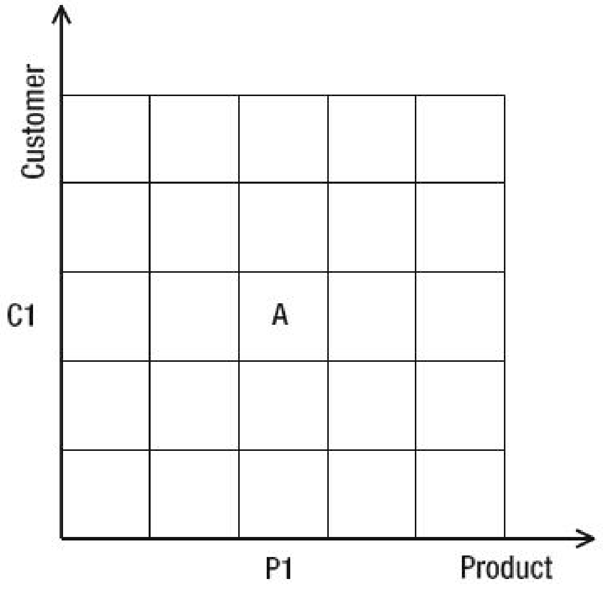
The multidimensional databases uses MOLAP (multidimensional online analytical processing) to access its data. They allow the users to quickly get answers to their requests by generating and analyzing the data rather quickly.

The data in multidimensional databases is stored in a data cube format. This means that data can be seen and understood from many dimensions and perspectives.

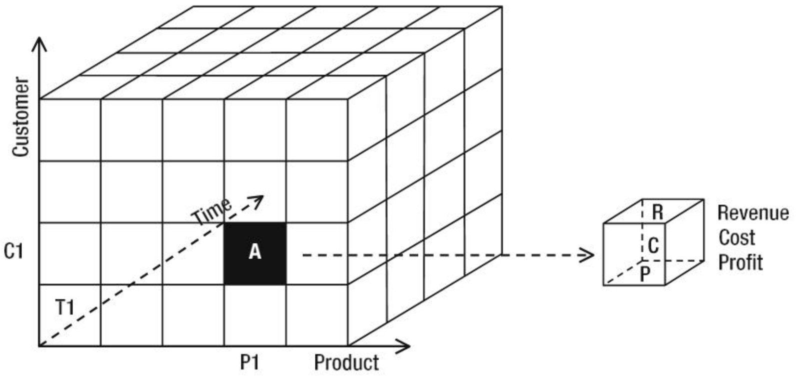
A multidimensional database is a form of data base where the data is stored incells and the position of each cell is defined by a number of hierarchical called dimensions. Each cell represents a business event, and the value of the dimensions shows when and where this even happened.

The structure stores the aggregate values as well as the base values, typically incom- pressed multidimensional array format, rather than in RDBMS tables. Aggregate values are precomputed summaries of the base values.

Physically, an MD Bisafile. To help you understand how an MDB file is internally structured, you can visualize it as a matrix or a cube. MDB with two dimensions; it looks like a matrix. The two dimensions are customer and product. The combination of these two dimensions points to a cell in the matrix. The cell contains one or more measure ment values (or none/empty). A cell represents a business event. The value of the dimensions shows when and where this event happened. Event A is created by customer C1 and product P1.



Multi dimensional databases are typically used for business intelligence (BI), especially for online an alytical processing (OLAP) and datamining (DM). The advantages of using multidimensional databases for OLAP and D Mrather than a relational data base such as a dimensional data store (DSS) are that they use less disks pace and have better performance. A multidimensional data base occupies less disk space compared to a relational dimensional database (like DDS) because it is compressed and because it does not use in dexinglikea DDS. Instead, it uses multidimensional off setting to locate the data. A multi dimensional database performs better on OLAP operations because the aggregates a repre calculated and because the way the data is physically stored (compressed multidimensional array format with offset positioning) minimizes the number off operations (disk reads), compared to storing tables in an RDBMS.



Q44. How many scheme a as are in a database? Explain its Characteristics?

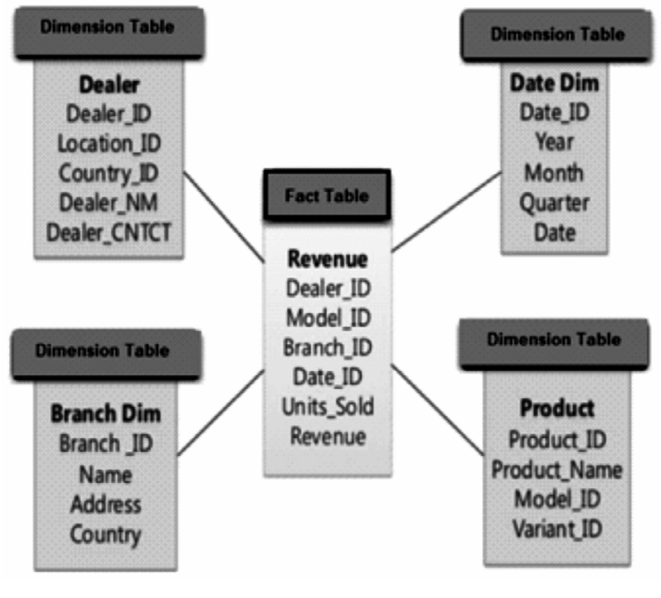
Ans :

3 chief types of multidimensional schemas each having its unique advantages.

1. Star Schema
2. Snowflake Schema
3. Galaxy Schema

1. Star Schema

In the STAR Schema, the center of the star can have one fact table and a number of associated dimension tables. It is known as star schema as its structure resembles a star. The star schema is the simplest type of Data Warehouse schema. It is also known as Star Join Schema and is optimized for querying large data sets.

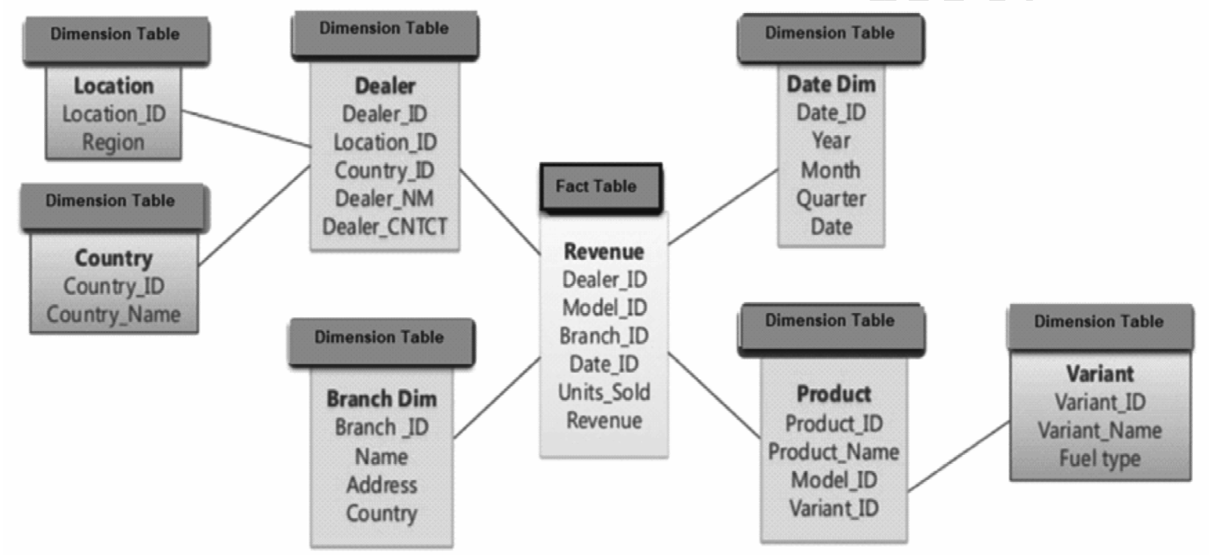


Characteristics of Star Schema

- Every dimension in a star schema is represented with the only one-dimension table.
- The dimension table should contain the set of attributes.
- The dimension table is joined to the fact table using a foreign key
- The dimension table are not joined to each other
- Fact table would contain key and measure
- The Star schema is easy to understand and provides optimal disk usage.
- The dimension tables are not normalized. For instance, in the above figure, Country_ID does not have Country lookup table as an OLTP design would have.
- The schema is widely supported by BI Tools

2. Snowflake Schema

SNOWFLAKE SCHEMA is a logical arrangement of tables in a multidimensional database such that the ER diagram resembles a snowflake shape. A Snowflake Schema is an extension of a Star Schema, and it adds additional dimensions. The dimension tables are normalized which splits data into additional tables.

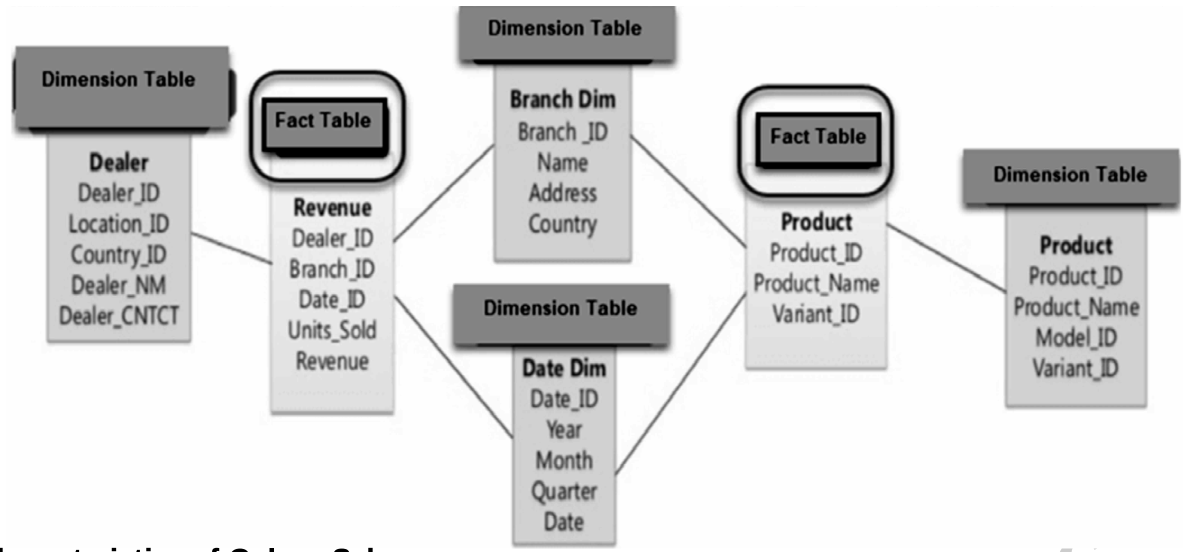


Characteristics of Snowflake Schema

- The main benefit of the snowflake schema it uses smaller disk space.
- Easier to implement a dimension is added to the Schema
- Due to multiple tables query performance is reduced
- The primary challenge that you will face while using the snowflake Schema is that you need to perform more maintenance efforts because of the more lookup tables.

3. Galaxy schema

A GALAXY SCHEMA contains two fact table that share dimension tables between them. It is also called Fact Constellation Schema. The schema is viewed as a collection of stars hence the name Galaxy Schema.



Characteristics of Galaxy Schema

- The dimensions in this schema are separated into separate dimensions based on the various levels of hierarchy.
- For example, if geography has four levels of hierarchy like region, country, state, and city then Galaxy schema should have four dimensions.
- Moreover, it is possible to build this type of schema by splitting the one-star schema into more Star schemes.
- The dimensions are large in this schema which is needed to build based on the levels of hierarchy.
- This schema is helpful for aggregating fact tables for better understanding.

5.9 Data Warehouse

Q45. What is a data warehouse? How does it differ from a database?

Ans : **(Imp.)**

A data warehouse is a type of contemporary database system designed to fulfil decision-support needs. However, a data warehouse differs from a conventional decision-support database in a number of ways:

- **Volume of data:** A data warehouse is likely to hold far more data than a decision-support database. Volumes of the order of over 400 gigabytes of data are commonplace
- **Diverse data sources:** The data stored in a warehouse is likely to have been extracted from a diverse range of application systems, only some of which may be database systems. These systems are described as data sources
- **Dimensional access:** A warehouse is designed to fulfil a number of distinct ways (dimensions) in which users may wish to retrieve data. This is some times referred to as the need to facilitate ad-hoc query.

A data warehouse as being 'a subject-oriented, integrated, time-variant, and non-volatile collection of data used in support of management decision-making':

- **Subject-oriented:** A data warehouse is structured in terms of the major subject are as of the organisation such as, in the case of a university, students, lecturers and modules, rather than in terms of application areas such as enrolment, payroll and time tabling.
- **Integrated:** A data warehouse provides a data repository which integrates data from different systems with data frequently in different formats. The objective is to provide a unified view of data for users.
- **Time-variant:** A data warehouse explicitly associates time with data. Data in a warehouse is only valid for some point or period in time
- **Non-volatile:** The data in a data warehouse is not updated in real-time. Instead, it is refreshed from data in operational systems on a regular basis. A consequence of this is that the management of data integrity is not a critical issue for data warehouses.

Q46. What are the benefits and goals of a data warehouse?

Ans :

A data warehouse is seen to deliver three major benefits for organisations:

- A data warehouse provides a single manageable structure for decision support data.
- A data warehouse enables organisational users to run complex queries ondata that traverses a number of business areas.
- A data warehouse enables a number of business intelligence applications such as on-line analytical processing and data mining.

The overall objective for a data warehouse is to increase the productivity and effectiveness of decision-making in organisations. This, in turn, is expected to deliver competitive advantage to organisations.

Challenges of Data Warehousing

Data warehousing projects are large-scale development projects. Typically a data warehousing project may take of the order of three years. Some of the challenges experienced in such projects are indicated below:

- Knowing in advance what the data users require, and determining the ownership and responsibilities in terms of data sources
- Selecting installing and integrating different hardware and software required to set up the warehouse. The large volume of data needed in terms of a data warehouse requires large amounts of disk space. This means that estimation of storage volume is a significant activity.
- Identifying reconciling and cleaning existing production data and loading it into the warehouse. The diverse sources of data feeding a data ware house introduces problems of design in terms of creating a homogeneous data store. Problems are also introduced in terms of the effort required to extract, clean and load data into the warehouse.

Q47. What are characteristics of data warehouse?

Ans :

As listed by E.F. Codd (1993), data warehouses have the following distinct characteristics:

- Multidimensional conceptual view.
- Generic dimensionality.
- Unlimited dimensions and aggregation levels.
- Unrestricted cross-dimensional operations.
- Dynamic sparse matrix handling.
- Client/server architecture.
- Multi-user support.
- Accessibility.
- Transparency.
- Intuitive data manipulation.
- Consistent reporting performance.
- Flexible reporting.

Q48. List the steps used in building the Data warehouse?

Ans :

The key steps involved in a data warehousing project are outlined below:

- Users specify information needs
- Analysts and users create a logical and physical design
- Sources of data are identified in operational systems, external sources etc.
- Source data is scrubbed, extracted and transformed
- Data is transferred and loaded into the warehouse periodically
- Users are given access to the warehouse data
- The warehouse is maintained in terms of changing requirements.

Q49. List and explain in brief the components of Data Warehouse?

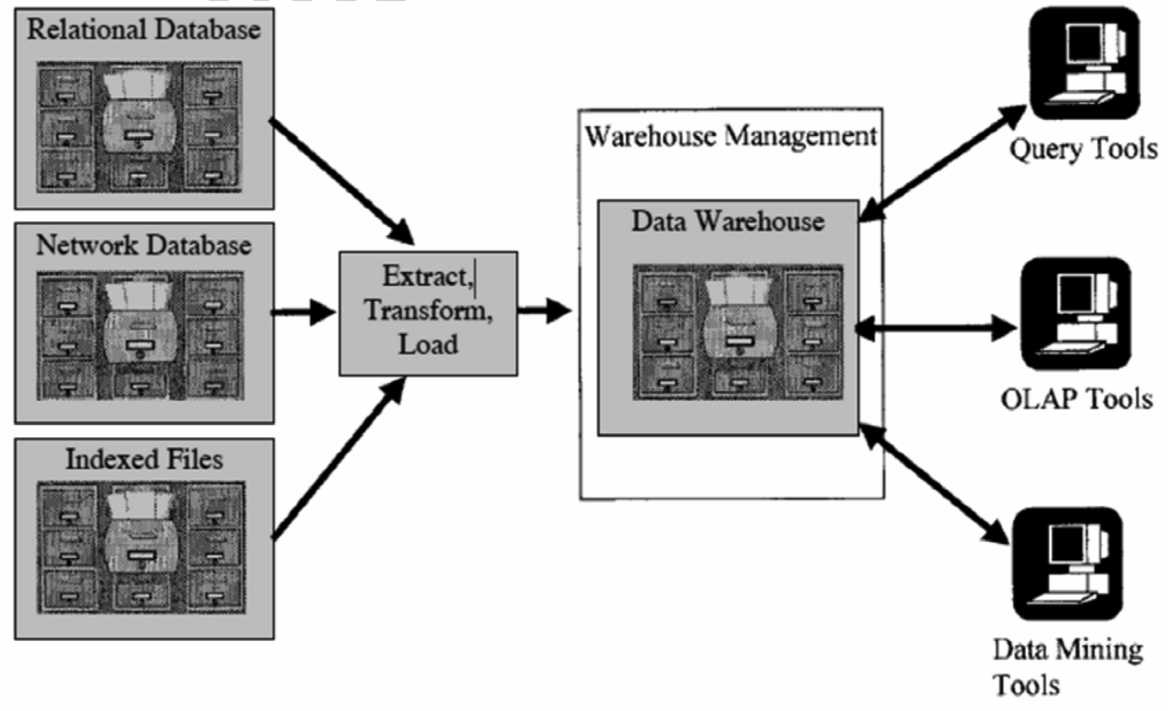
(OR)

What are the different components of a data warehouse?

Ans :

Some of the major components of a data warehouse:

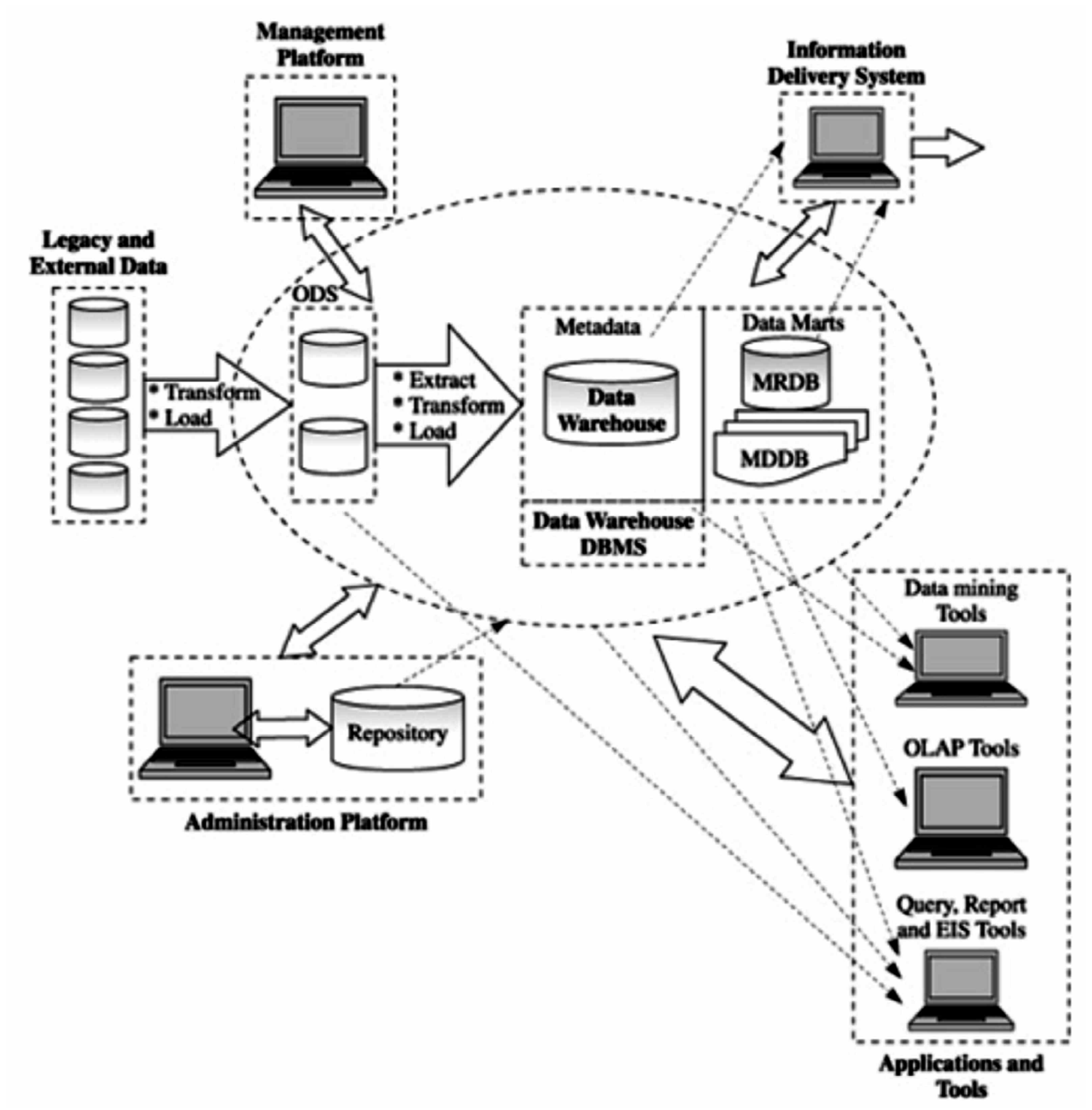
- **Operational data:** Data for the warehouse may be sourced in a number of ways, e.g. from mainframe-based hierarchical or network databases, from relational databases and from data in proprietary file systems.
- **Extraction, transformation and loading functions**: These ETL operations or functions are concerned with extracting data from source systems, transforming into a suitable form and loading the transformed data into the data warehouse
- **Warehouse management:** A series of functions must be provided to manage the warehouse: consistency analysis, indexing, denormalisation, aggregation, backup and archiving
- **Query management:** The warehouse must perform a series of operations concerned with the management of queries for use by a variety of actors. reporting and query tools, OLAP tools or tools for data mining.



Q50. Present a diagrammatic representation of the typical architecture and main components of a data warehouse.

Ans :

The data warehouse structure is based on a relational database management system server that functions as the central repository for informational data. In the data warehouse structure, operational data and processing is completely separate from data warehouse processing. This central information repository is surrounded by a number of key components, designed to make the entire environment functional, manageable and accessible by both the operational systems that source data into the warehouse and by end user query and analysis tools.



Following are the main components of data warehouse structure:

- Operational and external data sources.
- Data warehouse DBMS.
- Repository system.

- Data marts.
- Application tools.
- Management platform.
- Information delivery system.

Typically, source data for the warehouse comes from the operational applications or from an operational data store (ODS). The operational data store (ODS) is a repository of current and integrated operational data used for analysis. The ODS is often created when legacy operational systems are found to be incapable of achieving reporting requirements. The ODS provides users with the ease to use of a relational database while remaining distant from the decision support functions of the data warehouse. ODS is one of the more recent concepts in data warehousing. Its main purpose is to address the need of users, particularly clerical and operational managers, for an integrated view of current data. Data in ODS is subject-oriented, integrated, volatile and current or near current. The subject-oriented and integrated correspond to modelled and reconciled, while volatile and current or near-current correspond to read/write, transient and current. The data processed by ODS is a mixture of real-time and reconciled.

As the data enters the data warehouse, it is transformed into an integrated structure and format. The transformation process may involve conversion, summarisation, filtering and condensation of data. Because data within the data warehouse contains large historical components, the data warehouse must be capable of holding and managing large volumes of data as well as different data structures for the same database over time.

The central data warehouse DBMS is a cornerstone of data warehousing environment. It is almost always implemented on the relational database management system (RDBMS) technology. However, different technology approaches, such as parallel databases, multi-relational databases (MRDBs), multidimensional databases (MDDBs) and so on are also being used in data warehouse environment to fulfil the need for flexible user view creation including aggregates, multi-table joins and drill-downs. Data warehouse also includes metadata, which is data about data that describes the data warehouse. It is used for building, maintaining, managing and using the data warehouse. Metadata provides interactive access to users to help understand content and find data. Metadata management is provided via metadata repository and accompanying software. Metadata repository software can be used to map the source data to the target database, generate code for data transformations, integrate and transform the data, and control moving data to the warehouse. This software typically runs on a workstation and enables users to specify how the data should be transformed, such as by data mapping, conversion, and summarisation. Metadata repository maintains information directory that helps technical and business users to exploit the power of data warehousing. This directory helps integrate, maintain, and view the contents of the data warehousing system.

Multidimensional databases (MDDBs) are tightly coupled with the online analytical processing (OLAP) and other application tools that act as clients to the multidimensional data stores. These tools architecturally belong to a group of data warehousing components jointly categorised as the data query, reporting, analysis and mining tools. These tools provide information to business users for strategic decision making.

Q51. What are data marts? What are its advantages and limitations?

Ans :

Data mart is a generalised term used to describe data warehouse environments that are somehow smaller than others. It is a subsidiary of data warehouse of integrated data. It is a localised, single-purpose data warehouse implementation. Data mart is a relative and subjective term often used to describe small, single-purpose mini-data warehouses. The data mart is directed at a partition of data that is created for the use of a dedicated group of users. It delivers specific data to groups of users as required. Thus, data mart can be defined as "a specialised, subject-oriented, integrated, time-variant, volatile data store in support of specific subset of management's decisions". A data mart may contain summarised, de-normalized, or aggregated departmental data and can be customised to suit the needs of a particular department that owns the data. Data mart is used to describe an approach in which each individual department of a big enterprise implements its own management information system (MIS), often based on a large, parallel, relational database or on a smaller multidimensional or spreadsheet-like system.

In a large enterprise, data marts tend to be a way to build a data warehouse in a sequential, phased approach. A collection of data marts composes an enterprise-wide data warehouse. Conversely, a data warehouse may be construed as a collection of subset of data marts. Normally data marts are resident on a separate database servers, often on the local area network serving a dedicated user group. Data mart uses automated data replication tools to populate the new databases, rather than the manual processes and specially developed programs as being used previously.

Advantages of Data Marts

- Data marts enable departments to customise the data as it flows into the data mart from the data warehouse. There is no need for the data in the data mart to serve the entire enterprise. Therefore, the department can summarise, sort, select and structure their own department's data independently.
- Data marts enable department to select a much smaller amount of historical data than that which is found in the data warehouse.
- The department can select software for their data mart that is tailored to fit their needs.
- Very cost-effective.

Limitations of Data Marts

- Once in production, data marts are difficult to extend for use by other departments because of inherent design limitations in building for a single set of business needs, and disruption of existing users caused by any expansion of scope.
- Scalability problem in situations where an initial small data mart grows quickly in multiple dimensions.
- Data integration problem.

5.10 Online Analytical Processing (OLAP)

Q52. Explain briefly about online analytical processing.

Ans :

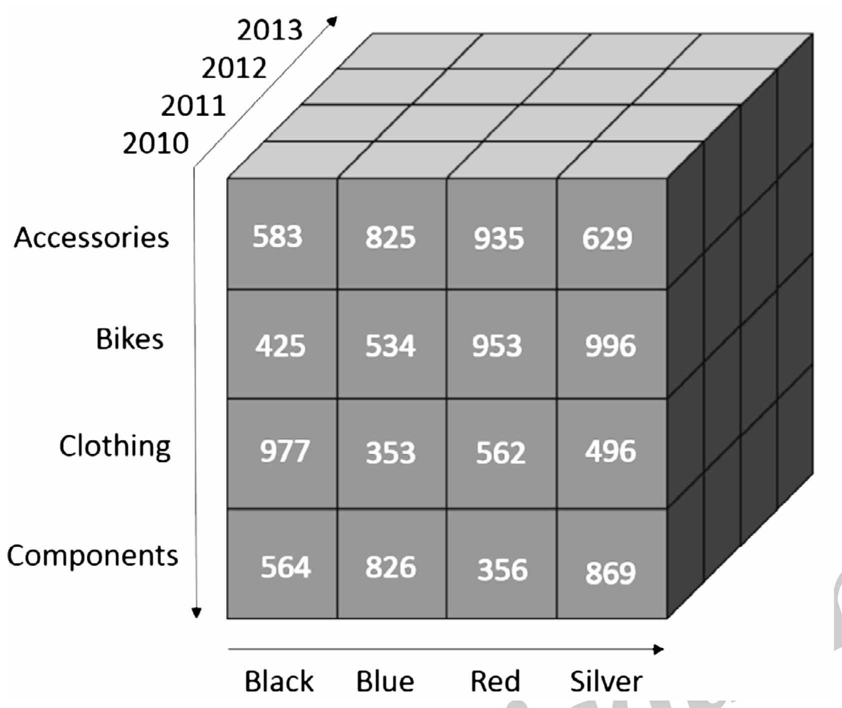
Online Analytical Processing (OLAP) is a category of software that allows users to analyze information from multiple database systems at the same time. It is a technology that enables analysts to extract and view business data from different points of view.

Analysts frequently need to group, aggregate and join data. These operations in relational databases are resource intensive. With OLAP data can be pre-calculated and pre-aggregated, making analysis faster.

OLAP databases are divided into one or more cubes. The cubes are designed in such a way that creating and viewing reports become easy. OLAP stands for Online Analytical Processing.

At the core of the OLAP concept, is an OLAP Cube. The OLAP cube is a data structure optimized for very quick data analysis.

The OLAP Cube consists of numeric facts called measures which are categorized by dimensions. OLAP Cube is also called the hypercube.



Usually, data operations and analysis are performed using the simple spreadsheet, where data values are arranged in row and column format. This is ideal for two-dimensional data. However, OLAP contains multidimensional data, with data usually obtained from a different and unrelated source. Using a spreadsheet is not an optimal option. The cube can store and analyze multidimensional data in a logical and orderly manner.

Q53. List and Discuss the Guidelines of OLAP?

Ans :

Dr. E.F. Codd, the "father" of the relational model, has formulated a list of 12 guidelines and requirements as the basis for selecting OLAP systems:

1. **Multidimensional Conceptual View:** This is the central features of an OLAP system. By needing a multidimensional view, it is possible to carry out methods like slice and dice.

2. **Transparency:** Make the technology, underlying information repository, computing operations, and the dissimilar nature of source data totally transparent to users. Such transparency helps to improve the efficiency and productivity of the users.

3. **Accessibility:** It provides access only to the data that is actually required to perform the particular analysis, present a single, coherent, and consistent view to the clients. The OLAP system must map its own logical schema to the heterogeneous physical data stores and perform any necessary transformations. The OLAP operations should be sitting between data sources (e.g., data warehouses) and an OLAP front-end.

4. **Consistent Reporting Performance:** To make sure that the users do not feel any significant degradation in documenting performance as the number of dimensions or the size of the database increases. That is, the performance of OLAP should not suffer as the number of dimensions is increased. Users must observe consistent run time, response time, or machine utilization every time a given query is run.

Multidimensional Conceptual View
Transparency
Accessibility
Consistency Reporting Performance
Client/Server Architecture
General Dimensionality
Dynamic Sparse Matrix Handling
Multiuser Support
Unrestricted Cross-dimensional Operations
Intuitive Data Manipulation
Flexible Reporting
Unlimited Dimensions and Aggregation Levels

5. **Client/Server Architecture:** Make the server component of OLAP tools sufficiently intelligent that the various clients to be attached with a minimum of effort and integration programming. The server should be capable of mapping and consolidating data between dissimilar databases.

6. **Generic Dimensionality:** An OLAP method should treat each dimension as equivalent in both is structure and operational capabilities. Additional operational capabilities may be allowed to selected dimensions, but such additional tasks should be grantable to any dimension.

7. **Dynamic Sparse Matrix Handling:** To adapt the physical schema to the specific analytical model being created and loaded that optimizes sparse matrix handling. When encountering the sparse matrix, the system must be easy to dynamically assume the distribution of the information and adjust the storage and access to obtain and maintain a consistent level of performance.

8. **Multiuser Support:** OLAP tools must provide concurrent data access, data integrity, and access security.

9. **Unrestricted cross-dimensional Operations:** It provides the ability for the methods to identify dimensional order and necessarily functions roll-up and drill-down methods within a dimension or across the dimension.

10. **Intuitive Data Manipulation:** Data Manipulation fundamental the consolidation direction like as reorientation (pivoting), drill-down and roll-up, and another manipulation to be accomplished naturally and precisely via point-and-click and drag and drop methods on the cells of the scientific model. It avoids the use of a menu or multiple trips to a user interface.

11. **Flexible Reporting:** It implements efficiency to the business clients to organize columns, rows, and cells in a manner that facilitates simple manipulation, analysis, and synthesis of data.

12. **Unlimited Dimensions and Aggregation Levels:** The number of data dimensions should be unlimited. Each of these common dimensions must allow a practically unlimited number of customer-defined aggregation levels within any given consolidation path.

Q54. Briefly Explain FASMI Characteristics of OLAP?

(OR)

Explain the characteristics of OLAP.

Ans :

In the FASMI characteristics of OLAP methods, the term derived from the first letters of the characteristics are:

i) Fast

It defines which the system targeted to deliver the most feedback to the client within about five seconds, with the elementary analysis taking no more than one second and very few taking more than 20 seconds.

ii) Analysis

It defines which the method can cope with any business logic and statistical analysis that is relevant for the function and the user, keep it easy enough for the target client. Although some preprogramming may be needed we do not think it acceptable if all application definitions have to be allow the user to define new Adhoc calculations as part of the analysis and to document on the data in any desired method, without having to program so we excludes products (like Oracle Discoverer) that do not allow the user to define new Adhoc calculation as part of the analysis and to document on the data in any desired product that do not allow adequate end user-oriented calculation flexibility.

iii) Share

It defines which the system tools all the security requirements for understanding and, if multiple write connection is needed, concurrent update location at an appropriated level, not all functions need customer to write data back, but for the increasing number which does, the system should be able to manage multiple updates in a timely, secure manner.

iv) Multidimensional

This is the basic requirement. OLAP system must provide a multidimensional conceptual view of the data, including full support for hierarchies, as this is certainly the most logical method to analyze business and organizations.

v) Information

The system should be able to hold all the data needed by the applications. Data sparsity should be handled in an efficient manner.

5.11 Difference between OLTP and OLAP

Q55. What are the main differences between OLAP and OLTP ?

Ans : **(Imp.)**

OLTP (On-Line Transaction Processing) is featured by a large number of short on-line transactions (INSERT, UPDATE, and DELETE). The primary significance of OLTP operations is put on very rapid query processing, maintaining record integrity in multi-access environments, and effectiveness consistent by the number of transactions per second. In the OLTP database, there is an accurate and current record, and schema used to save transactional database is the entity model (usually 3NF).

OLAP (On-line Analytical Processing) is represented by a relatively low volume of transactions. Queries are very difficult and involve aggregations. For OLAP operations, response time is an effectiveness measure. OLAP applications are generally used by Data Mining techniques. In OLAP database there is aggregated, historical information, stored in multi-dimensional schemas.

OLAP (ONLINE ANALYTICAL PROCESSING)	OLTP (ONLINE TRANSACTION PROCESSING)
Consists of historical data from various Databases.	Consists only operational current data.
It is subject oriented. Used for Data Mining, Analytics, Decision making, etc.	It is application oriented. Used for business tasks.
The data is used in planning, problem solving and decision making.	The data is used to perform day to day fundamental operations.
It reveals a snapshot of present business tasks.	It provides a multi-dimensional view of different business tasks.
Large amount of data is stored typically in TB, PB	The size of the data is relatively small as the historical data is archived. For ex MB, GB
Relatively slow as the amount of data involved is large. Queries may take hours.	Very Fast as the queries operate on 5% of the data.
It only need backup from time to time as compared to OLTP.	Backup and recovery process is maintained religiously
This data is generally managed by CEO, MD, GM.	This data is managed by clerks, managers.
Only read and rarely write operation.	Both read and write operations.

5.12 NoSQL Database

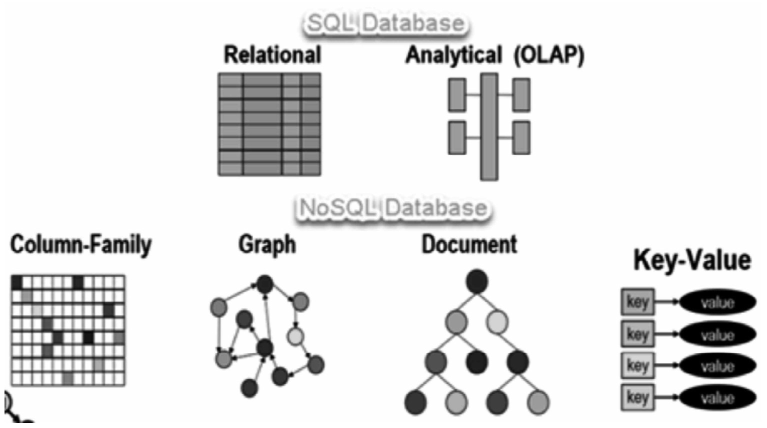
Q56. What is NoSQL Database?

Ans : **(Imp.)**

NoSQL Database is a non-relational Data Management System, that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.

NoSQL database stands for “Not Only SQL” or “Not SQL.” Though a better term would be “NoREL”, NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

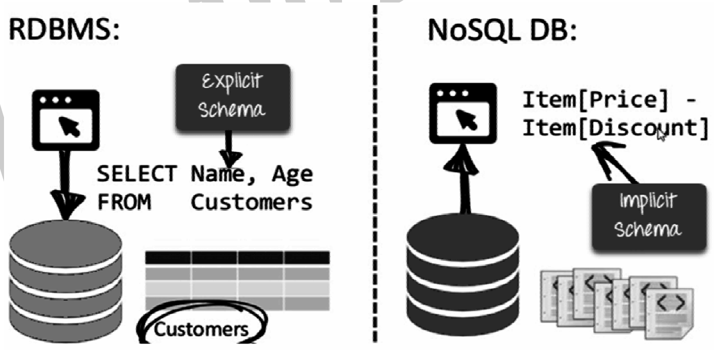


Q57. Explain the features of NoSQL

Ans :

i) Non-relational

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn't require object-relational mapping and data normalization
- No complex features like query languages, query planners, referential integrity joins, ACID.



ii) Schema-free

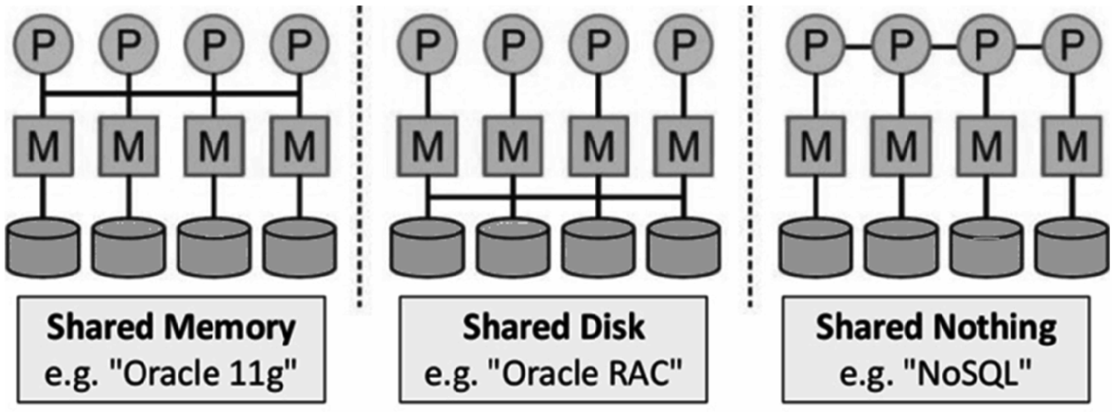
- NoSQL databases are either schema-free or have relaxed schemas.
- Do not require any sort of definition of the schema of the data.
- Offers heterogeneous structures of data in the same domain.

iii) Simple API

- Offers easy to use interfaces for storage and querying data provided.
- APIs allow low-level data manipulation and selection methods.
- Text-based protocols mostly used with HTTP REST with JSON.
- Mostly used no standard based query language.
- Web-enabled databases running as internet-facing services.

Distributed

- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Often ACID concept can be sacrificed for scalability and throughput
- Mostly no synchronous replication between distributed nodes Asynchronous Multi-Master Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency
- Shared Nothing Architecture. This enables less coordination and higher distribution.



Q58. Explain the Query Processing in NoSQL Database?

Ans :

The most common data retrieval mechanism is the REST-based retrieval of a value based on its key/ ID with GET resource Document store Database offers more difficult queries as they understand the value in a key-value pair. For example, CouchDB allows defining views with MapReduce

CAP Theorem

CAP theorem is also called brewer's theorem. It states that is impossible for a distributed data store to offer more than two out of three guarantees

i) Consistency

ii) Availability

iii) Partition Tolerance

iv) Eventual Consistency

i) Consistency

The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.

ii) Availability

The database should always be available and responsive. It should not have any downtime.

iii) Partition Tolerance

Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

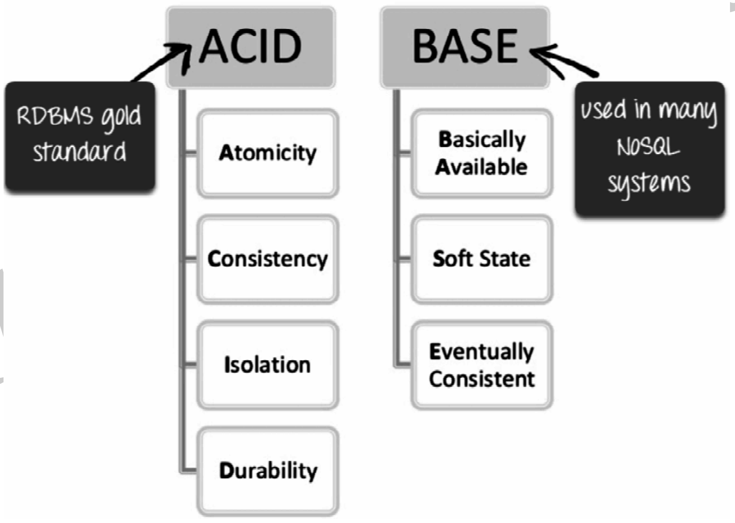
iv) Eventual Consistency

The term "eventual consistency" means to have copies of data on multiple machines to get high availability and scalability. Thus, changes made to any data item on one machine has to be propagated to other replicas.

Data replication may not be instantaneous as some copies will be updated immediately while others in due course of time. These copies may be mutually, but in due course of time, they become consistent. Hence, the name eventual consistency.

BASE: Basically Available, Soft state, Eventual consistency

- Basically, available means DB is available all the time as per CAP theorem
- Soft state means even without an input; the system state may change
- Eventual consistency means that the system will become consistent over time



5.12.1 Advantages and Dis-Advantages of NoSQL

Q59. List Few Advantages and Dis-Advantages of NoSQL?

Ans :

Advantages of NoSQL

- Can be used as Primary or Analytic Data Source
- Big Data Capability
- No Single Point of Failure
- Easy Replication
- No Need for Separate Caching Layer

- It provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect
- Object-oriented programming which is easy to use and flexible
- NoSQL databases don't need a dedicated high-performance server
- Support Key Developer Languages and Platforms
- Simple to implement than using RDBMS
- It can serve as the primary data source for online applications.
- Handles big data which manages data velocity, variety, volume, and complexity
- Excels at distributed database and multi-data center operations
- Eliminates the need for a specific caching layer to store data
- Offers a flexible schema design which can easily be altered without downtime or service disruption

Disadvantages of NoSQL

- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values as keys become difficult
- Doesn't work as well with relational data
- The learning curve is stiff for new developers
- Open source options so not so popular for enterprises.

Short Question & Answers

1. List few Operations of multimedia databases.

Ans :

- **Input (insert/record) operation:** The data will be written to the database. The raw and registering data are always needed; descriptive data can be attached later.
- **Output (play) operation:** It involves reading the raw data from the database according to the registered data.
- **Modification:** It involves changing of raw, registering and descriptive data. Modification can also be understood as a data conversion from one format to another.
- **Deletion Operation:** This operation removes an entry from the database. The consistency of the data must be preserved.

2. Define Wireless Communications.

Ans ;

- The wireless medium have bandwidth significantly lower than those of a wired network.
- The current generation of wireless technology has data rates range from the tens to hundreds of kilobits per second (2G cellular telephony) to tens of megabits per second.
- Modern (wired) Ethernet, by comparison, provides data rates on the order of hundreds of megabits per second.

3. What is WWW?

Ans :

The World Wide Web, or “the Web” as it is colloquially called, has been one of the great successes in the history of computing. After the conception in 1989, Web is the most popular and powerful-networked information system till date. The combinations of the Web technology and databases have resulted into many new opportunities for creating advanced database applications.

4. Define Uniform Resource Locator (URL)

Ans :

Uniform resource locator (URL) is a key component of the Web. It is a globally unique name for each document that can be accessed on the Web. URL provides a simple addressing mechanism that allows the Web to link information on computers all over the world. It is a string of alphanumeric characters that represent the location or address of a resource on the Internet and how that resource should be accessed. URL is a special code to identify each Web page on the World Wide Web.

5. Define Internet Protocol (IP) Address

Ans :

All host computers on the Internet are identified by a unique address called IP address. IP address consists of a series of numbers. Computers on the Internet use these IP address numbers to communicate with each other. ISPs provide this IP address so that we can enter it as part of the setup process when we originally setup our communication connection to the ISP.

6. What is Multidimensional Database?

Ans :

A multidimensional data base is a form of data base where the data is stored in cell sand the position of each cell is defined by a number of hierarchical called dimensions. Each cell represents a businesse vent, and the value of the dimensions shows when and where this event happened.

The structures to rest he aggregate values as well as the base values, typically incompressed multidimensional array format, rather than in RDBMS tables. Aggregate values are pre computed summaries of the base values.

7. List and Define Schemas for Multi-dimensional Data Model

Ans :

i) Star Schema: In the STAR Schema, the center of the star can have one fact table and a number of associated dimension tables. It is known as star schema as its structure resembles a star. The star schema is the simplest type of Data Warehouse schema.

ii) Snowflake Schema: SNOWFLAKE SCHEMA is a logical arrangement of tables in a multidimensional database such that the ER diagram resembles a snowflake shape. A Snowflake Schema is an extension of a Star Schema, and it adds additional dimensions.

iii) Galaxy schema: A GALAXY SCHEMA contains two fact table that share dimension tables between them. It is also called Fact Constellation Schema. The schema is viewed as a collection of stars hence the name Galaxy Schema.

8. What are the benefits of Data warehousing?

Ans :

A data warehouse is seen to deliver three major benefits for organisations:

- A data warehouse provides a single manageable structure for decisionsupport data
- A data warehouse enables organisational users to run complex queries on data that traverses a number of business areas.
- A data warehouse enables a number of business intelligence applications such as on-line analytical processing and data mining.

The overall objective for a data warehouse is to increase the productivity and effectiveness of decision-making in organisations. This, in turn, is expected to deliver competitive advantage to organisations.

9. List the steps used in building the Data warehouse?

Ans :

The key steps involved in a data warehousing project are outlined below (Inmon, 2000):

- Users specify information needs
- Analysts and users create a logical and physical design
- Sources of data are identified in operational systems, external sources etc.
- Source data is scrubbed, extracted and transformed
- Data is transferred and loaded into the warehouse periodically
- Users are given access to the warehouse data
- The warehouse is maintained in terms of changing requirements.

10. Define Metadata.

Ans :

Meta-data- Data about data is needed to enable the extraction, transformation and loading processes by mapping data sources to the warehouse schema. Meta-data is also used to automate the production of summary data and to facilitate query management.

11. What is Data Mart?

Ans :

A DATA MART is focused on a single functional area of an organization and contains a subset of data stored in a Data Warehouse. A Data Mart is a condensed version of Data Warehouse and is designed for use by a specific department, unit or set of users in an organization.

12. List few Types of Data Mart

Ans :

There are three main types of data marts are:

i) Dependent: Dependent data marts are created by drawing data directly from operational, external or both sources.

ii) Independent: Independent data mart is created without the use of a central data warehouse.

iii) Hybrid: This type of data marts can take data from data warehouses or operational systems.

13. What is OLAP?

Ans :

Online Analytical Processing (OLAP) is a category of software that allows users to analyze information from multiple database systems at the same time. It is a technology that enables analysts to extract and view business data from different points of view.

Analysts frequently need to group, aggregate and join data. These operations in relational databases are resource intensive. With OLAP data can be pre-calculated and pre-aggregated, making analysis faster.

14. Explain about distributed database.

Ans :

A distributed database is a database that consists of two or more files located in different sites either on the same network or on entirely different networks. Portions of the database are stored in multiple physical locations and processing is distributed among multiple database nodes.

A distributed database is basically a database that is not limited to one system, it is spread over different sites, i.e, on multiple computers or over a network of computers. A distributed database system is located on various sited that don't share physical components. This maybe required when a particular database needs to be accessed by various users globally. It needs to be managed such that for the users it looks like one single database.

- Distributed database is a system in which storage devices are not connected to a common 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.
- Shared nothing architecture is used in distributed databases.

15. Write the advantages of DDBMS.

Ans :

(i) Data are located near the greatest demand site. The data in a distributed database system are dispersed to match business requirements which reduce the cost of data access.

(ii) Faster data access. End users often work with only a locally stored subset of the company's data.

(iii) Faster data processing. A distributed database system spreads out the systems workload by processing data at several sites.

(iv) Growth facilitation. New sites can be added to the network without affecting the operations of other sites.

(v) Improved communications. Because local sites are smaller and located closer to customers, local sites foster better communication among departments and between customers and company staff.

(vi) Reduced operating costs. It is more cost-effective to add workstations to a network than to update a mainframe system. Development work is done more cheaply and more quickly on low-cost PCs than on mainframes.

(vii) User-friendly interface. PCs and workstations are usually equipped with an easy-to-use graphical user interface (GUI). The GUI simplifies training and use for end users.

(viii) Less danger of a single-point failure. When one of the computers fails, the workload is picked up by other workstations. Data are also distributed at multiple sites.

(ix) Processor independence. The end user is able to access any available copy of the data, and an end user's request is processed by any processor at the data location.

16. Define data replication.

Ans :

Data Replication is the process of storing data in more than one site or node. It is useful in improving the availability of data. It is simply copying data from a database from one server to another server so that all the users can share the same data without any inconsistency.

Data Replication is the process of storing data in more than one site or node. It is useful in improving the availability of data. It is simply copying data from a database from one server to another server so that all the users can share the same data without any inconsistency. The result is a distributed database in which users can access data relevant to their tasks without interfering with the work of others.

17. What is Fragmentation? Write its advantages and disadvantages.

Ans :

Fragmentation

Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called fragments. Fragmentation can be of three types: horizontal, vertical, and hybrid (combination of horizontal and vertical). Horizontal fragmentation can further be classified into two techniques: primary horizontal fragmentation and derived horizontal fragmentation.

Fragmentation should be done in a way so that the original table can be reconstructed from the fragments. This is needed so that the original table can be reconstructed from the fragments whenever required. This requirement is called "reconstructiveness."

Advantages of Fragmentation

- Since data is stored close to the site of usage, efficiency of the database system is increased.
- Local query optimization techniques are sufficient for most queries since data is locally available.
- Since irrelevant data is not available at the sites, security and privacy of the database system can be maintained.

Disadvantages of Fragmentation

- When data from different fragments are required, the access speeds may be very high.
- In case of recursive fragmentations, the job of reconstruction will need expensive techniques.
- Lack of back-up copies of data in different sites may render the database ineffective in case of failure of a site.

18. Explain 3-tier architecture

Ans :

Three-tier architecture typically comprise a presentation tier, a business or data access tier, and a data tier. Three layers in the three tier architecture are as follows:

(i) Client layer

(ii) Business layer

(iii) Data layer

(i) Client layer

It is also called as Presentation layer which contains UI part of our application. This layer is used for the design purpose where data is presented to the user or input is taken from the user. For example designing registration form which contains text box, label, button etc.

(ii) Business layer

In this layer all business logic written like validation of data, calculations, data insertion etc. This acts as a interface between Client layer and Data Access Layer. This layer is also called the intermediary layer helps to make communication faster between client and data layer.

(iii) Data layer

In this layer actual database is comes in the picture. Data Access Layer contains methods to connect with database and to perform insert, update, delete, get data from database based on our input data.

19. Define DDBMS.

Ans :

A distributed database management system (DDBMS) is a set of multiple, logically interrelated databases distributed over a network. They provide a mechanism that makes the distribution of data transparent to users.

DDBMS is a centralized application that manages a distributed database. This database system synchronizes data periodically and ensures that any change in data made by users is universally updated in the database.

DDBMS is widely used in data warehousing, where huge volumes of data are processed and accessed by numerous users or database clients at the same time. This database system is used to manage data in networks, maintain confidentiality and handle data integrity. A distributed database management system is designed for heterogeneous database platforms that focus on heterogeneous database management systems.

The software system that permits the management of the distributed database and makes the distribution transparent to users.

20. Write Disadvantages of DDBMS.

Ans :

1. Complexity of management and control. Applications must recognize data location, and they must be able to stitch together data from various sites. Database administrators must have the ability to coordinate database activities to prevent database degradation due to data anomalies.
2. Technological difficulty. Data integrity, transaction management, concurrency control, security, backup, recovery, query optimization, access path selection, and so on, must all be addressed and resolved.
3. Security. The probability of security lapses increases when data are located at multiple sites. The responsibility of data management will be shared by different people at several sites.
4. Lack of standards. There are no standard communication protocols at the database level. (Although TCP/IP is the de facto standard at the network level, there is no standard at the application level.) For example, different database vendors employ different—and often incompatible—techniques to manage the distribution of data and processing in a DDBMS environment.
5. Increased storage and infrastructure requirements. Multiple copies of data are required at different sites, thus requiring additional disk storage space.
6. Increased training cost. Training costs are generally higher in a distributed model than they would be in a centralized model, sometimes even to the extent of offsetting operational and hardware savings.

21. Define vertical Fragmentation.

Ans :

Vertical Fragmentation

In vertical fragmentation, the fields or columns of a table are grouped into fragments. In order to maintain reconstructiveness, each fragment should contain the primary key field(s) of the table. Vertical fragmentation can be used to enforce privacy of data.

For example, let us consider that a University database keeps records of all registered students in a Student table having the following schema.

STUDENT

Regd_No Name Course Address Semester Fees Marks

Now, the fees details are maintained in the accounts section. In this case, the designer will fragment the database as follows:

CREATE TABLE STD_FEES AS

SELECT Regd_No,Fees

FROM STUDENT;

22. List advantages of Client Server Computing.

Ans :

- All the required data is concentrated in a single place i.e. the server. So it is easy to protect the data and provide authorisation and authentication.
- The server need not be located physically close to the clients. Yet the data can be accessed efficiently.
- It is easy to replace, upgrade or relocate the nodes in the client server model because all the nodes are independent and request data only from the server.
- All the nodes i.e clients and server may not be build on similar platforms yet they can easily facilitate the transfer of data.

23. List out advantages of 3tier architecture

Ans :

1. High performance, lightweight persistent objects
2. Scalability – Each tier can scale horizontally
3. Performance – Because the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers.
4. High degree of flexibility in deployment platform and configuration
5. Better Re-use
6. Improve Data Integrity
7. Improved Security – Client is not direct access to database.
8. Easy to maintain and modification is bit easy, won't affect other modules
9. In three tier architecture application performance is good.

24. Compare centralized database with distributed database.

Ans :

Centralized Database	Distributed Database
1. It allows storage of data at single location.	1. It allows the storage of data at multiple physical locations.
2. Remote users can access the database using Wise Area Network (WAN).	2. All the end users can access the data present in database using WAN.
3. It is easy to store, retrieve and update data in centralized database.	3. Data updation on each site is a complex task as each site must be updated.
4. It does not support dynamic business environment.	4. It supports dynamics business environment.

Choose the Correct Answer

1. Which of the following is a benefit of a parallel database system? [d]
   (a) Improved performance (b) Greater flexibility
   (c) Better availability (d) All of these.

2. Parallel database system has the disadvantage of ________ [d]
   (a) More start-up cost. (b) Interference problem
   (c) Skew problem (d) All of these.

3. Which of the following is an advantage of data warehousing? [d]
   (a) Better enterprise intelligence (b) Business reengineering
   (c) Cost-effective decision-making. (d) All of these.

4. Data mart is a data store which is ________ [d]
   (a) Specialised and subject-oriented (b) Integrated and time-variant.
   (c) Volatile data store. (d) All of these.

5. Which of the following is not an Internet addressing system? [c]
   (a) Domain name (b) URL
   (c) HTTP (d) IP address

6. Which of the following must be unique in Internet? [d]
   (a) IP address (b) E-mail address
   (c) Domain name (d) All of these.

7. Spatial databases keep track of objects in a [a]
   (a) Multidimensional space (b) Single dimensional space.
   (c) Both (a) & (b). (d) None of these.

8. OLAP stands for ________ [a]
   (a) Online analytical processing (b) Online analysis processing
   (c) Online transaction processing (d) Online aggregate processing

9. Data that can be modelled as dimension attributes and measure attributes are called ________ data [b]
   (a) Single dimensional (b) Multidimensional
   (c) Measured (d) Dimensional

10. Find the advantages of DDMBS. [d]
    (a) Fast access of data (b) Improved communications
    (c) User friendly interface (d) All

11. ________ is the division of DB into segments [c]

(a) Fragmentation (b) Replication

(c) Partition (d) None of the above

12. Distribute D.B devided into ________ types? [b]

(a) 3 (b) 2

(c) 4 (d) 5

13. Client / Server systems may contain different types of models. [a]

(a) 2 (b) 3

(c) 4 (d) 5

14. ________ refers to users receives full intial copies of the DataBase and then receive periodic updates as data changes. [d]

(a) Fully Replicated (b) Partial replicated

(c) Un Replicated (d) Transactional Replicated

15. Table is a collection of [c]

(a) Rows (b) Columns

(c) a & b (d) None

16. Find the drawback of data distribution [d]

(a) Complexily (b) Security

(c) Lack of standards (d) All

17. ________ refers to, maintain duplicated data files at different sites [b]

(a) Partitioning (b) Replication

(c) Transparency (d) None

18. 2PL stands for ________ . [ ]

(a) 2 phase lock and acequition (b) c & d

(c) 2 phase locking protocol (d) Strict 2 phase locking

19. ________ allows to break a single object into 2 (or) more segments [c]

(a) Partition (b) Replication

(c) Fragmentation (d) All

Fill in the blanks

1. ________ divides larger tasks into many smaller tasks, and executes the smaller tasks concurrently on several communication nodes. (Parallel Processing)
2. The World Wide Web is a subset of ________ that uses computers called ________ to store multimedia files. (Internet, Web Server)
3. HTML is the abbreviation of ________. (Hyper Text Markup Language)
4. URL is the abbreviation for ________. (Uniform Resource Locator).
5. An ________ is a unique number that identifies computers on the Internet. (IP Address)
6. ________ provide features that allow users to store and query different types of multimedia information like images, videos , audio clips and text or documents.(Multi Media Database)
7. The multimedia queries are called ________ queries. (Content Based)
8. Spatial databases keep track of objects in a ________ space. (Multidimensional)
9. The operation of moving from finer-granularity data to a coarser granularity is called a ________. (Rollup)
10. Time stamped based concurrency control protocol can be used in ________ system.
11. ________ is the first activity taken place in 2 phase locking protocol.
12. ________ fragmentation refers to the division of a relation into subset of Rows.
13. DDBMS makes the operation of DB-System appear to the user as a ________ DataBase.
14. ________ is the function of DB maintenance.
15. DDMBS divided into ________ & ________ DB'S.
16. ________ refers to the storage of data copies at multiple sites.
17. ________ fragmentation refers to combination of Vertical & Horizantal stratigies .
18. ________ is a division of a logical database.
19. DDBMS stands for ________ .

ANSWERS

1. Parallel Processing
2. Internet, Web Server
3. Hyper Text Markup Language
4. Uniform Resource Locator.
5. IP Address
6. Multi Media Database
7. Content Based
8. Multidimensional

9. Rollup
10. Distributed
11. Lock acquisition
12. Horizontal
13. Centralized
14. Managing Resources
15. Homogenious & Hetrogenious
16. Data Replication
17. Mixed
18. Partition
19. Distributed Database Management System

One Mark Answers

1. Define Data Warehousing.

Ans :

Storage and access of data from the central location in order to take some strategic decision is called Data Warehousing. Enterprise management is used for managing the information whose framework is known as Data Warehousing.

2. What is OLTP?

Ans :

OLTP is abbreviated as On-Line Transaction Processing, and it is an application that modifies the data whenever it received and has large number of simultaneous users.

3. What is OLAP?

Ans :

OLAP is abbreviated as Online Analytical Processing, and it is set to be a system which collects, manages, processes multi-dimensional data for analysis and management purposes.

4. What is Datamart?

Ans :

A Datamart is a specialized version of Datawarehousing and it contains a snapshot of operational data that helps the business people to decide with the analysis of past trends and experiences. A data mart helps to emphasizes on easy access to relevant information.

5. What are NoSQL databases?

Ans :

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases (like SQL, Oracle, etc.).

6. What are the different types of NoSQL databases?

Ans :

Types of NoSQL databases:

- Document Oriented
- Key Value
- Graph
- Column Oriented

7. What are the features of NoSQL?

Ans :

When compared to relational databases, NoSQL databases are more scalable and provide superior performance, and their data model addresses several issues that the relational model is not designed to address:

- Large volumes of structured, semi-structured, and unstructured data
- Object-oriented programming that is easy to use and flexible
- Efficient, scale-out architecture instead of expensive, monolithic architecture.

8. What Is a Web Database?

Ans :

A Web database is a database application designed to be managed and accessed through the Internet. Website operators can manage this collection of data and present analytical results based on the data in the Web database application.

9. List the Benefits of Web database.

Ans :

Here are various benefits that come through the use of web-based DBMS are:

- Web-DBMS is Platform independence
- Provides Graphical User Interface (GUI)
- Provides Cross-platform support
- Facilitates transparent network access
- Scalability

10. What is multimedia?

Ans :

Multimedia is a technique that incorporates

- text,
- graphics,
- sound,
- animations and
- video elements.

11. What is Metadata?

Ans :

Metadata is defined as data about the data. The metadata contains information like number of columns used, fix width and limited width, ordering of fields and data types of the fields.

12. What is the definition of Cube in Datawarehousing?

Ans :

Cubes are logical representation of multidimensional data. The edge of the cube has the dimension members,and the body of the cube contains the data values.

13. What are the Advantages of Multidimensional Databases

Ans :

Some advantages of multidimensional databases are:

i) Increased performance

The performance of the multidimensional databases is much superior to that of normal databases such as relational database.

ii) Easy maintenance

The multidimensional database is easy to handle and maintain

iii) Better data presentation

The data in a multidimensional database is multi faced and contains many different factors.

14. Mention the major uses of Multimedia

Ans :

- Multimedia is heavily used in the entertainment industry, especially to develop special effects in moves and animation for cartoon characters.
- Multimedia games are a popular pastime and these are software programs available either as CD ROMs or online.
- Some video games also use multimedia features.

15. Distributed DataBase

Ans :

A logically interrelated collection of shared data. Physically distributed over a computer network.

16. Define DDBMS

Ans :

Distributed DataBase management system is a software that handles the management of distributed DataBase and makes the operation of such a system appear to the user as a centralized DataBase.

17. Data Replication

Ans :

It Refers to the storage of data copies at multiple sites by a computer network.

18. Data Fragmentation

Ans :

It allows to break a single object two (or) more segments (or) fragments. The object might be a user's DB a system D.B (or) a Table.

19. List any 2 advantages of client server computing

Ans :

- They provide low cost and User-Friendly Environment.
- They allow connectivity with the heterogeneous machines deals with real time.

Lab Programs with Solutions

Create a Supplier table as shown below

Sup_No	Sup_Name	Item_Supplied	Item_Price	City
S1	Suresh	Keyboard	400	Hyderabad
S2	Kiran	Processor	8000	Delhi
S3	Mohan	Mouse	600	Delhi
S4	Ramesh	Processor	9000	Bangalore
S5	Manish	Printer	6000	Mumbai
S6	Srikanth	Processor	8500	Chennai

Create table supplier(Sup_No varchar2(3) primary key, Sup_Name char(10), Item_Supplied char(10), Item_Price number(4), City char(10));

```
SQL>
SQL> create table Supplier(Sup_No varchar2(3) primary key, Sup_Name char(10),
Item_Supplied char(lO), ltem_Price number(4), city char(10));
```

Table created.

```
SQL> insert into Supplier
values('&Sup_No','&Sup_Name','&Item_Supplied',&Item_Price,'&City');
SQL>
SQL> insert into Supplier values('&Sup_No','&Sup_Name','&Item_Supplied',&Item_Price,'&city');
Enter value for sup_no: S1
Enter value for sup_name: Suresh
Enter value for item_supplied: Keyboard
Enter value for item_price: 400
Enter value for city: Hyderabad
old 1: insert into Supplier values('&Sup_No', '&Sup_Name','&Item_Supplied',& ltem_Price,' &city')
new 1: insert into Supplier values('S1','Suresh','Keyboard',400,'Hyderabad')
```

1 row created.

```
SQL> /
Enter value for sup_no: S2
Enter value for sup_name: Kiran
Enter value for item_supplied: Processor
Enter value for item_price: 8000
Enter value for city: Delhi
old 1: insert into Supplier values('&Sup_No','&Sup_Name','&Item_Supplied',&Item_Price,'& City')
new 1: insert into Supplier values('S2KiranProcessor8000,'Delhi')
```

1 row created.

```
SQL> /
Enter value for sup_no: S3
Enter value for sup_name: Mohan
Enter value for item_supplied: Mouse
Enter value for item_price: 600
Enter value for city: Delhi
old 1:insert into Supplier values('&Sup_No','&Sup_Name','&Item_Supplied',&Item_Price,'& city')
new 1: insert into Supplier values('S3','Mohan','Mouse',600,'Delhi')
```
1 row created.
```
SQL> /
Enter value for sup_no: S4
Enter value for sup_name: Ramesh
Enter value for item_supplied: Processor
Enter value for item_price: 9000
Enter value for city: Bangalore
old 1: insert into Supplier values('&Sup_No*,'&Sup_Name','&Item_Supplied',&Item_Price,'& city')
new 1: insert into Supplier values('S4','Ramesh','Processor',9000,'Bangalore')
```
1 row created.
```
SQL> /
Enter value for sup_no: S5
Enter value for sup_name: Manish
Enter value for item_supplied: Printer
Enter value for item_price: 6000
Enter value for city: Mumbai
old 1: insert into Supplier values('&Sup_No't'&Sup_Name','&Item_Supplied',&Item_Price,'&City')
new 1: insert into Supplier values('s5','Mamsh','Printer',6000,'Mumbai')
```
1 row created.
```
SQL> /
Enter value for sup_no: S6
Enter value for sup_name: Srikanth
Enter value for item_supplied: Processor
Enter value for item_price: 8500
Enter value for city: Chennai
old 1: insert into Supplier values('&Sup_No','&Sup_Name','&Item_Supplied',&Item_Price,'&city')
new 1: insert into Supplier values('s6','Srikanth','Processor',8500,'Chennai')
```
1 row created.

Select * from Supplier

SQL> select * from Supplier;

SUP	SUP_NAME	ITEM_SUPPL	ITEM_PRICE	CITY
S1	Suresh	Keyboard	400	Hyderabad
S2	Ki ran	Processor	8000	Delhi
S3	Mohan	Mouse	600	Delhi
S4	Ramesh	Processor	9000	Bangalore
S5	Manish	Printer	6000	Mumbai
S6	Srikanth	Processor	8500	Chennai

6 rows selected.

1. Write SQL query to display Supplier number and Supplier Name whose names starts with "S".

Ans :

selectSup_No,Sup_Name from Supplier where Sup_Name like 'S%';

SUP	SUP_NAME
S1	SURESH
S6	Srikanth

2. Write a SQL query to display the details of supplier who supply Processor and Whose City is Delhi.

Ans :

select * from Supplier where Item_Supplied='Processor' AND city='Delhi';

SQL> select * from Supplier where Item_Supp1ied='Processor' AND city='Del hi';

SUP	SUP_NAME	ITEM_SUPPL	ITEM_PRICE	CITY
S2	Kiran	Processor	8000	Delhi

3. Write SQL Query to increase the price of keyboard by 200.

Ans :

update Supplier set Item_Price=Item_price+200 where Item_Supplied='Keyboard';
SQL> select * from Supplier;

SUP	SUP_NAME	ITEM_SUPPL	ITEM_PRICE	CITY
S1	Suresh	Keyboard	400	Hyderabad
S2	Ki ran	Processor	8000	Delhi
S3	Mohan	Mouse	600	Delhi
S4	Ramesh	Processor	9000	Bangalore
S5	Manish	Printer	6000	Mumbai
S6	Srikanth	Processor	8500	Chennai

6 rows selected.

```
SQL> update Supplier set Item_Price=Item_price+200 where Item_Supplied='Keyboard';
1 row updated.
SQL> select * from supplier;
```

SUP	SUP_NAME	ITEM_SUPPL	ITEM_PRICE	CITY
S1	Suresh	Keyboard	600	Hyderabad
S2	Kiran	Processor	8000	Delhi
S3	Mohan	Mouse	600	Delhi
S4	Ramesh	Processor	9000	Bangalore
S5	Manish	Printer	6000	Mumbai
S6	Srikanth	Processor	8500	Chennai

```
6 rows selected.
```

4. Write SQL query to display Sup_No, Sup_Nameand Item_Price from Supplier in Delhi in the Ascending order of Item price.

Ans :

Select Sup_No,Sup_Name,Item_Supplied,Item_price from Supplier where city='Delhi' order by Item_price ASC;

SUP	SUP_NAME	ITEM_SUPPL	ITEM_PRICE
S3	Mohan	Mouse	600
S2	Kiran	Processor	8000

5. Write a SQL query to add a column called Contact Number.

Ans :

Alter table Supplier add Contact_Nonumber(10);

```
SQL> desc Supplier;
```

Name	Null ?	Type
SUP_NO	NOT NULL	VARCHAR2(3)
SUP_NAME		CHAR (10)
ITEM_SUPPLIED		CHAR (10)
ITEM_PRICE		NUMBER (4)
CITY		CHAR (10)

```
SQL> Alter table Supplier add Contact_No number(10):
Table altered.
SQL> desc Supplier;
```

Name	Null ?	Type
SUP_NO	NOT NULL	VARCHAR2(3)
SUP_NAME		CHAR (10)
ITEM_SUPPLIED		CHAR (10)
ITEM_PRICE		NUMBER (4)
CITY		CHAR (10)
CONTACT_NO		NUMBER (10)

6. Create a view on the table which displays only Supplier Number and Supplier names.

Ans :

Create view Supplierview as select sup_No,Sup_Name from Supplier;

SQL> create view Supplierview as select sup_No,Sup_Name from Supplier;

view created.

SQL> select * from Supplierview;

SUP	SUP_NAME
S1	Suresh
S2	Kiran
S3	Mohan
S4	Ramesh
S5	Manish
S6	Srikanth

6 rows selected.

7. Write query to display the record in descending order of item price for each item supplied.

Ans :

selectItem_Supplied,Item_Price from Supplier order by Item_price DESC;

SQL> select Item_Supplied, Item_Price from Supplier order by Item_price DESC;

ITEM_SUPPL	ITEM_PRICE
Processor	9000
Processor	8500
Processor	8000
Printer	6000
Keyboard	600
Mouse	600

6 rows selected

8. Write a SQL query to display the records of supplier who supply items other than Processor or Keyboard.

Ans ;

select * from Supplier where NOT Item_Supplied=’Processor’ OR Item_Supplied=’Keyboard’;

SQL> select * from Supplier where NOT Item_Supplied=’Processor’ OR Item_Supplied =’ Keyboard’;

SUP	SUP_NAME	ITEM_SUPPL	ITEM_PRICE	CITY	CONTACT_NO
S1	Suresh	Keyboard	600	Hyderabad	
S3	Mohan	Mouse	600	Delhi	
S5	Manish	Printer	6000	Mumbai	

9. Write a SQL query to delete the record whose Item price is the lowest of all the items supplied.

Ans :

delete from Supplier where Item_Price=(Select min(Item_Price) from Supplier);

Select * from supplier;

SUP	SUP_NAME	ITEM_SUPPL	ITEM_PRICE	CITY	CONTACT_NO
S1	Suresh	Keyboard	600	Hyderabad	
S2	Kiran	Processor	8000	Delhi	
S3	Mohan	Mouse	600	Delhi	
S4	Ramesh	Processor	9000	Bangalore	
S5	Manish	Printer	6000	Mumbai	
S6	Srikanth	Processor	8500	Chennai	

6 rows selected.

SQL> delete from Supplier where Item_Price=(Select min(Item_Price) from Supplier);

2 rows deleted.

SQL> select * from Supplier;

SUP	SUP_NAME	ITEM_SUPPL	ITEM_PRICE	CITY	CONTACT_NO
S2	Kiran	Processor	8000	Delhi	
S4	Ramesh	Processor	9000	Bangalore	
S5	Manish	Printer	6000	Mumbai	
S6	Srikanth	Processor	8500	Chennai	

Below are the details of the Employees working for a software Company. Create the table called EmpDetailsWith the details Mentioned.

EID	EName	DOB	Designation	Salary	DOJ
E101	Suma	29-Dec-89	Designer	20000	01-Apr-10
E102	Amit	10-Jan-95	Programmer	25000	18-Feb-18
E103	Payal	15-Aug-85	Tester	35000	12-Jun-11
E104	Kiran	20-Apr-90	Programmer	40000	07-Mar-14
E105	Meenal	29-May-83	DBA	50000	09-Dec-11
E106	Sheila	01-May-70	Analysist	60000	25-Sep-18
E107	Swamy	13-Jan-85	Programmer	45000	14-feb-16
E108	Sushma	22-Dec-76	DBA	45000	31-Jan-12

create table EmpDetails(Eid varchar2(5) primary key, EName char(8),DOB date, Designationchar (12), Salary number(6), DOJ date);

SQL > create table EmpDetai1s(Eid varchar2(5) primary key, EName char(8),D0B date,Designation char(2),Salary number(6), DOJ date);

Table Created

SQL > insert into EmpDetails values('&Eid','&EName',to_date('&DOB','dd-mon-yy'),' & Designation', & Salary,to_date('&DOJ','dd-mon-yy'));

Enter value for eid: E101

Enter value for ename: Suma

Enter value for dob: 29-Dec-89

Enter value for designation: Designer

Enter value for salary: 20000

Enter value for doj: 01-Apr-10

old 1: insert into EmpDetails values('&Eid','&EName',to_date('&DOB','dd-mon-yy'),'&Designation',&Salary,to_date('&DOJ','dd-mon-yy'))

new 1: insert into EmpDetails values('ElOl','suma',to_date('29-Dec-89','dd-mon-yy'),'Designer',20000,to_date('01-Apr-10','dd-mon-yy'))

1 row created.

SQL > /

Enter value for eid: E102

Enter value for ename: Amit

Enter value for dob: 10-Jan-95

Enter value for designation: Programmer

Enter value for salary: 25000

Enter value for doj: 18-Feb-18

```
old 1: insert into EmpDetails values('&Eid','&EName',to_date('&DOB','dd-mon-yy'),'&Designation',&Salary,to_date('&D03','dd-mon-yy'))

new 1: insert into EmpDetails values('E102','Amit',to_date('10-Jan-95','dd-mon-yy'),'Programmer',
25000,to_date('18-Feb-18','dd-mon-yy'))
```

1 row created.

```
SQL> /
Enter value for eid: E103
Enter value for ename: Payal
Enter value for dob: 15-Aug-85
Enter value for designation: Tester
Enter value for salary: 35000
Enter value for doj: 13-Jun-11
old 1: insert into EmpDetails values('&Eid','&EName',to_date('&DOB','dd-mon-yy'), '& Designation',& Salary,to_date('&DOJ','dd-mon-yy'))

new 1: insert into EmpDetails values('E103Payal',to_date('15-Aug-85','dd-mon-yy'),'Tester',35000,to_date('13-Jun-ll','dd-mon-yy'))
```

1 row created.

```
SQL> /
Enter value for eid: E104
Enter value for ename: Kiran
Enter value for dob: 20-Apr-90
Enter value for designation: Programmer
Enter value for salary: 40000
Enter value for doj: 07-Mar-14
old 1: insert into EmpDetails values('&Eid','&EName',to_date('&D0B','dd-mon-yy'),'&Designation',& Salary, to_date('&DOJ','dd-mon-yy'))

new 1: insert into EmpDetails values('E104','Kiran',to_date('20-Apr-90','dd-mon-yy'),'Programmer',
40000,to_date('07-Mar-14','dd-mon-yy'))
```

1 row created.

```
SQL> /
Enter value for eid: E105
Enter value for ename: Meenal
Enter value for dob: 29-May-83
Enter value for designation: DBA
Enter value for salary: 50000
Enter value for doj: 09-Dec-ll
old 1: insert into EmpDetails values('&Eid','&EName',to_date('&DOB','dd-mon-yy'),'&Designation',& Salary,to_date('&DOJ','dd-mon-yy'))

new 1: insert into EmpDetails values('E105','Meenal',to_date('29-May-83','dd-mon-yy'),'DBA',50000, to_date('09-Dec-ll','dd-mon-yy'))
```

```
1 row created.
SQL> /
Enter value for eid: E106
Enter value for ename: Sheila
Enter value for dob: 01-May-70
Enter value for designation: Analyst
Enter value for salary: 60000
Enter value for doj: 25-sep-18
old 1: insert into EmpDetails values('&Eid','&EName',to_date('&DOB','dd-mon-yy'),'&Designation',& Salary,to_date('&DOJ','dd-mon-yy'))
new 1: insert into EmpDetails values('E106','Sheila',to_date('01-May-70','dd-mon-yy'),'Analyst',60000,to_date('25-Sep-18','dd-mon-yy'))
1 row created.
SQL> /
Enter value for eid: E107
Enter value for ename: Swamy
Enter value for dob: 13-jan-85
Enter value for designation: Programmer
Enter value for salary: 45000
Enter value for doj: 14-Feb-16
old 1: insert into EmpDetails values('&Eid','SEName',to_date('&DOB','dd-mon-yy'), '&Designation',& salary,to_date('&DOJ','dd-mon-yy'))
new 1: insert into EmpDetails values('E107','Swamy',to_date('13-jan-85dd-mon-yy'),'Programmer', 45000,to_date('14-Feb-16','dd-mon-yy'))
1 row created.
SQL> /
Enter value for eid: E108
Enter value for ename: Sushma
Enter value for dob: 22-Dec-76
Enter value for designation: DBA
Enter value for salary: 45000
Enter value for doj: 31-Jan-12
old 1: insert into EmpDetails values('&Eid','&EName',to_date('&DOB','dd-mon-yy'), '&Designation',& Salary,to_date('&DOJ','dd-mon-yy'))
new 1: insert into EmpDetails values('El08','sushma',to_date('22-Dec-76','dd-mon-yy'),'DBA',45000, to_date('31-Jan-12','dd-mon-yy'))
1 row created.
```

Select * from EmpDetails;

SQL> select * from EmpDetails;

EID	ENAME	DOB	DESIGNATION	SALARY	DOJ
E101	Suma	29-DEC-89	Designer	20000	01-APR-10
E102	Amit	10-JAN-95	Programmer	25000	18-FEB-18
E103	Payal	15-AUG-85	Tester	35000	13-JUN-11
E104	Kiran	20-APR-90	Programmer	40000	07-MAR-14
E105	Meenal	29-MAY-83	DBA	50000	09-DEC-11
E106	Sheila	01-MAY-70	Analyst	60000	25-SEP-18
E107	Swamy	13-JAN-85	Programmer	45000	14-FEB-16
E108	Sushma	22-DEC-76	DBA	45000	31-JAN-12

8 rows selected.

10. Write a SQL query to display all the employees whose designation is Programmer.

Ans :

select * from EmpDetails where Designation='Programmer';

SQL> select * from EmpDetails where Designation = 'Programmer';

EID	ENAME	DOB	DESIGNATION	SALARY	DOJ
E102	Amit	10-JAN-95	Programmer	25000	18-FEB-18
E104	Kiran	20-APR-90	Programmer	40000	07-MAR-14
E107	Swamy	13-JAN-85	Programmer	45000	14-FEB-16

11. Write a SQL query to display all the employees whose name starts with "A".

Ans :

select * from EmpDetails where EName like 'A%';

SQL> select * from EmpDetails where EName like 'A%';

EID	ENAME	DOB	DESIGNATION	SALARY	DOJ
E102	Amit	10-JAN-95	Programmer	25000	18-FEB-18

12. Write a SQL query to display the details of the employee whose name contain "ee".

Ans :

select * from EmpDetails where EName LIKE '%ee%';

SQL> select * from EmpDetails where EName LIKE '%ee%';

EID	ENAME	DOB	DESIGNATION	SALARY	DOJ
E105	Meenal	29-MAY-83	DBA	50000	09-DEC-11

13. Write SQL query to display the total salary of all the employees whose designation is Programmer.

Ans :

```
select sum(salary) from EmpDetails where Designation='Programmer';
SQL> select sum(salary) from EmpDetails where Designation = 'Programmer';
SUM(SALARY)
        110000
```

14. Write SQL query to display all the employee names in UPPER case.

Ans :

```
select upper(EName) from EmpDetails;
SQL> select upper(EName) from EmpDetails;
UPPER(EN
SUMA
AMIT
PAYAL
KIRAN
MEENAL
SHEILA
SWAMY
SUSHMA
8 rows selected.
```

15. Write SQL query to increase the salaries of the employees by 5000 whose Designation is DBA.

Ans :

```
updateEmpDetails set Salary=Salary+5000 where Designation='DBA';
SQL> select * from EmpDetails;
```

EID	ENAME	DOB	DESIGNATION	SALARY	DOJ
E101	Suma	29-DEC-89	Designer	20000	01-APR-10
E102	Amit	10-IAN-95	Programmer	25000	18-FEB-18
E103	Payal	15-AUG-85	Tester	35000	13-JUN-11
E104	Kiran	20-APR-90	Programmer	40000	07-MAR-14
E105	Meenal	29-MAY-83	DBA	50000	09-DEC-11
E106	Sheila	01-MAY-70	Analyst	60000	25-SEP-18
E107	Swamy	13-IAN-85	Programmer	45000	14-FEB-16
E108	Sushma	22-DEC-76	DBA	45000	31-IAN-12

8 rows selected.

SQL > update EmpDetails set Salary = Salary + 5000 where Designation = 'DBA';

2 rows updated.

SQL > select * from EmpDetails;

EID	ENAME	DOB	DESIGNATION	SALARY	DOJ
E101	Suma	29-DEC-89	Designer	20000	01-APR-10
E102	Amit	10-JAN-95	Programmer	25000	18-FEB-18
E103	Payal	15-AUG-85	Tester	35000	13-JUN-11
E104	Kiran	20-APR-90	Programmer	40000	07-MAR-14
E105	Meenal	29-MAY-83	DBA	55000	09-DEC-11
E106	Sheila	01-MAY-70	Analyst	60000	25-SEP-18
E107	Swamy	13-JAN-85	Programmer	45000	14-FEB-16
E108	Sushma	22-DEC-76	DBA	50000	31-JAN-12

8 rows selected.

16. Write a SQL query to display all the Employees whose salary is greater than the average of the salary of all the employees.

Ans :

select * from EmpDetails where Salary > (Select avg(Salary) from EmpDetails);

SQL > select * from EmpDetails where Salary > (Select avg(Salary) from EmpDetails);

EID	ENAME	DOB	DESIGNATION	SALARY	DOJ
E105	Meenal	29-MAY-83	DBA	55000	09-DEC-11
E106	Sheila	01-MAY-70	Analyst	60000	25-SEP-18
E107	Swamy	13-JAN-85	Programmer	45000	14-FEB-16
E108	Sushma	22-DEC-76	DBA	50000	31-JAN-12

17. Find the Maximum, Minimum and Total Salary paid by the company.

Ans :

select Max(Salary) As MaxSalary,Min(Salary) As MinSalary,Sum(Salary) As TotSalary from EmpDetails;

SQL > select Max(Salary) As MaxSalary, Min(Salary) As MinSalary, Sum(Salary) As TotSalary from EmpDetails;

MAXSALARY	MINSALARY	TOTSALARY
60000	20000	330000

18. Write SQL Query to display the unique designation for the Employees.

Ans :

```
select Distinct Designation from EmpDetails;
SQL > select Distinct Designation from EmpDetails;
DESIGNATION
Tester
Designer
DBA
Analyst
Programmer
```

19. Write SQL query to display the employee with Experience.

Ans :

selectEID,EName, round(to_number(months_between(sysdate,DOJ)/12),0) As Experience from EmpDetails;

SQL > select EID,EName, round(to_number(months_between(sysdate,DOI)/12),0) As Experience from EmpDetails;

EID	ENAME	EXPERIENCE
E101	Suma	10
E102	Amit	3
E103	Payal	9
E104	Kiran	6
E105	Meenal	9
E106	Sheila	2
E107	Swamy	5
E108	Sushma	9

8 rows selected

Below are the details of Students Enrolled in various courses of B.Com. Create the table called Student with the details below.

SID	SName	DOB	State	Gender	Category	Course
1001	Neha	29-Dec-02	Telangana	F	Gen	Comp
1002	Arun	10-Jan-02	Telangana	M	OBC	Honors
1003	Payal	15-Aug-01	Maharashtra	F	Gen	Appl
1004	Amrita	20-Apr-02	Karnataka	F	OBC	Honors
1005	Pavan	29-May-03	Andhra Pradesh	M	ExService	Comp
1006	Anchal	01-May-03	Gujrat	F	OBC	Comp
1007	Ramya	13-Jan-02	Telangana	F	Gen	Appl
1008	Rakesh	22-Dec-01	Andhra Pradesh	M	Sports	Comp

SQL>create table student(sid Number(5) primary key,SName char(8), DOB date,state char(15).Gender char(2).Category char(15).Course char(10));

Table created.

insert into Student values(&Sid,'&SName',to_date('&DOB','DD-MON-YY'),'&State','& Gender','& Category','&Course');

SQL> insert into Student values(&Sid,'&SName',to_date('&DOB','DD-MON-YY'),'&State',' &Gender','&category','&Course');

```
Enter value for sid: 1001
Enter value for sname: Neha
Enter value for dob: 29-Dec-02
Enter value for state: Telangana
Enter value for gender: F
Enter value for category: Gen
Enter value for course: Comp
```

old 1: insert into Student values(&sid,'&SName',to_date('&DOB','DD-MON-YY'),'&state',' & Gender','& Category'$_t$'&Course')

new 1: insert into Student values(1001,'Neha',to_date('29-Dec-02','DD-MON-YY'),'Telangana','F','Gen','Comp')

1 row created.

SQL> /

```
Enter value for sid: 1002
Enter value for sname: Arun
Enter value for dob: 10-Jan-02
Enter value for state: Telangana
Enter value for gender: M
Enter value for category: OBC
Enter value for course: Honors
```

old 1: insert into Student values(&sid,'&SName',to_date('&DOB','DD-MON-YY'),'&state',' &Gender','& Category','&Course')

new 1: insert into Student values(1002,'Arun',to_date('10-Jan-02','DD-MON-YY'),'Telangana','M','O BC','Honors')

1 row created.

SQL> /

```
Enter value for sid: 1003
Enter value for sname: Payal
Enter value for dob: 15-Aug-01
Enter value for state: Maharashtra
Enter value for gender: F
Enter value for category: Gen
Enter value for course: Appl
```

old 1: insert into Student values(&Sid,'&SName',to_date('&DOB','DD-MON-YY*),'&State','& Gender','& Category','&Course')

```
new 1: insert into Student values(1003,'Payal',to_date('15-Aug-01','DD-MON-YY'),'Maharashtra','F'
,'Gen','Appl')
1 row created.
SQL> /
Enter value for sid: 1004
Enter value for sname: Amritha
Enter value for dob: 20-Apr-02
Enter value for state: Karnataka
Enter value for gender: F
Enter value for category: OBC
Enter value for course: Honors
old 1: insert into Student values(&sid,'ASName',to_date('&DOB*,'DD-MON-YY'),'Astate','AGender',
' A Category','ACourse')
new 1: insert into Student values(1004,'Amritha',to_date('20-Apr-02','DD-MON-YY'),'Karnataka',
'F','OBC','Honors')
1 row created.
SQL> /
Enter value for sid: 1005
Enter value for sname: Pavan
Enter value for dob: 29-May-03
Enter value for state: Andnrapradesh
Enter value for gender: M
Enter value for category: Exservicemen
Enter value for course: Comp
old 1: insert into student values(Asid,'ASName',to_date('ADOB','DD-MON-YY'),'Astate','AGender','A
Category'.'ACourse')
new 1: insert into Student values(1005,'Pavan',to_date('29-May-03','DD-MON-
YY'),'Andhrapradesh','M', 'Exservicemen','Comp')
1 row created.
SQL> /
Enter value for sid: 1006
Enter value for sname: Anchal
Enter value for dob: 01-May-03
Enter value for state: Gujarat
Enter value for gender: F
Enter value for category: OBC
Enter value for course: Comp
old 1: insert into Student values(Asid,'ASName',to_date('ADOB','DD-MON-YY'),
'Astate','AGender','A Category's'ACourse')
new 1: insert into Student values(1006,'Anchal',to_date('01-May-03','DD-MON-YY'),'Gujarat','F','O
BC','Comp')
1 row created.
```

```
SQL> /
Enter value for sid: 1007
Enter value for sname: Ramya
Enter value for dob: 13-Ian-02
Enter value for state: Telangana
Enter value for gender: F
Enter value for category: Gen
Enter value for course: Appl
old 1: insert into Student values(&sid,'&SName',to_date('&D0B','DD-MON-YY'),'&state','&
Gender','& Category','&course')
new 1: insert into Student values(1007,'Ramya',to_date('13-Jan-02','DD-MON-YY'),'Telangana','F','
Gen','Appl')
1 row created.
SQL> /
Enter value for sid: 1008
Enter value for sname: Rakesh
Enter value for dob: 22-Dec-01
Enter value for state: Andhrapradesh
Enter value for gender: M
Enter value for category: Sports
Enter value for course: Comp.
old 1: insert into Student values(&sid,'&SName',to_date('&D0B','DD-MON-YY'),'&state','&
Gender','& Category','&course')
new 1: insert into Student values(1008,'Rakesh',to_date('22-Dec-01','DD-MON-
YY'),'Andhrapradesh', 'M','sports','Comp.')
1 row created.
select * from Student;
SQL> select * from Student;
```

SID	SNAME	DOB	STATE	GE	CATEGORY	COURSE
1001	Neha	29-DEC-02	Telangana	F	Gen	Comp
1002	Arun	10-JAN-02	Telangana	M	OBC	Honors
1003	Payal	15-AUG-01	Maharashtra	F	Gen	Appl
1004	Amritha	20-APR-02	Karnataka	F	OBC	Honors
1005	Pavan	29-MAY-03	Andhrapradesh	M	Exservicemen	Comp
1006	Anchal	01-MAY-03	Gujarat	F	OBC	Comp
1007	Ramya	13-JAN-02	Telangana	F	Gen	Appl
1008	Rakesh	22-DEC-01	Andhrapradesh	M	Sports	Comp.

```
8 rows selected.
```

20. Write SQL query to display the students who are not from Telangana or Andhrapradesh.

Ans :

select * from Student where NOT State= 'Telangana' AND NOT State= 'Andhrapradesh';

SQL> select * from student where NOT state='Telangana' AND NOT state='Andhrapradesh';

SID	SNAME	DOB	STATE	GE	CATEGORY	COURSE
1003	Payal	15-AUG-01	Maharashtra	F	Gen	Appl
1004	Amritha	20-APR-02	Karnataka	F	OBC	Honors
1006	Anchal	01-MAY-03	Gujarat	F	OBC	Comp

21. Create a view to display the columns Sid and SName from Student belonging to Telangana.

Ans :

create view Std as select Sid, SName from Student where State='Telangana';

SQL> create view Std as select sid, SName from student where state='Telangana';

view created.

SQL> select * from std;

SID	SNAME
1001	Neha
1002	Arun
1007	Ramya

22. Write SQL query to create index on Column SName.

Ans :

create index SNameIndex on Student(SName);

SQL> create index SNameIndex on Student (SName):

Index created.

23. Write SQL query to display all the female students enrolled under Comp course and who belong to OBC.

Ans :

select * from Student where Gender='F'AND Course='Comp' AND Category='OBC';

SQL> select; * from Student where Gender='F'AND Course='Comp' AND Category= 'OBC';

SID	SNAME	DOB	STATE	GE	CATEGORY	COURSE
1006	Anchal	01-MAY-03	Gujarat	F	OBC	Comp

24. Write SQL query to add two new columns contactno and emailed to the existing fields.

Ans :

alter table student ADD contactno number(10);

alter table student ADD email varchar2(255);

SQL > desc student;

Name	Null?	Type
SID	NOT NULL	NUMBER(5)
SNAME		CHAR(8)
DOB		DATE
STATE		CHAR(15)
GENDER		CHAR(2)
CATEGORY		CHAR(15)
COURSE		CHAR(10)

SQL > alter table student ADD contactno number(IO); able altered.

SQL > alter table student ADD email varchar2(255);

Table altered.

SQL > desc student;

Name	Null?	Type
SID	NOT NULL	NUMBER(5)
SNAME		CHAR(8)
DOB		DATE
STATE		CHAR(15)
GENDER		CHAR(2)
CATEGORY		CHAR(15)
COURSE		CHAR(10)
CONTACTNO		NUMBER(10)
EMAIL		VARCHAR2(255)

25. Write an SQL query to display all the student names where the length of the names is 5 characters.

Ans :

select SName,substr(Sname,1,6) from Student;

SQL > select SName,substr(Sname,l,6) from Student;

SNAME	SUBSTR
Neha	Neha
Arun	Arun
Payal	Payal
Amritha	Amrith
Pavan	Pavan
Anchal	Anchal
Ramya	Ramya
Rakesh	Rakesh

8 rows selected.

26. Write a SQL query to display all the students names prefixed by Mr./Mrs. Based on Gender.

Ans :

```
SQL> select SID, Case
when Gender= 'M'then 'Mr.' when Gender= 'F'then 'Mrs.'
end, SName from Student;
SQL> select SID,Case
2 when Gender='M'then 'Mr.'
3 when Gender='F'then 'Mrs.'
4 end, SName from Student;
```

SID	CASE	SNAME
1001	Mrs.	Neha
1002	Mr.	Arun
1003	Mrs.	Payal
1004	Mrs.	Amritha
1005	Mr.	Pavan
1006	Mrs.	Anchal
1007	Mrs.	Ramya
1008	Mr.	Rakesh

8 rows selected.

27. Write SQL query to Delete all the students who enrolled for Comp course.

Ans :

```
delete from Student where Course='Comp';
SQL> select * from student;
```

SID	SNAME	DOB	STATE	GE	CATEGORY	COURSE
1001	Neha	29-DEC-02	Telangana	F	Gen	Comp
1002	Arun	10-1AN-02	Telangana	M	OBC	Honors
1003	Payal	15-AUG-01	Maharashtra	F	Gen	Appl
1004	Amritha	20-APR-02	Karnataka	F	OBC	Honors
1005	Pavan	29-MAY-03	Andhrapradesh	M	Exservicemen	Comp
1006	Anchal	01-MAY-03	Gujarat	F	OBC	Comp
1007	Ramya	13-JAN-02	Telangana	F	Gen	Appl
1008	Rakesh	22-DEC-01	Andhrapradesh	M	Sports	Comp.

8 rows selected.

SQL> delete from Student where Course='Comp';

3 rows deleted.

SQL> select * from student;

SID	SNAME	DOB	STATE	GE	CATEGORY	COURSE
1002	Arun	10-1AN-02	Telangana	M	OBC	Honors
1003	Payal	15-AUG-01	Maharashtra	F	Gen	Appl
1004	Amritha	20-APR-02	Karnataka	F	OBC	Honors
1007	Ramya	13-1AN-02	Telangana	F	Gen	Appl
1008	Rakesh	22-DEC-01	Andhrapradesh	M	Sports	Comp.

28. Write SQL query to display the Sid, Sname and their present ages.

Ans :

select SID,Sname,round(to_number(months_between(sysdate,DOB)/12),0) As AGE from Student;

SQL> select SID, Sname, round(to_number(months_between(sysdate,DOB)/12),0) As AGE from Student

SID	SNAME	AGE
1001	Neha	18
1002	Arun	19
1003	Payal	19
1004	Amritha	18
1005	Pavan	17
1006	Anchal	17
1007	Ramya	19
1008	Rakesh	19

8 rows selected.

Create a Table for Library information with Library.

BookId	BookName	Author	DatePurchased	Publisher	Price
B101	CostAccounting	Jain Narang	11-Feb-13	Kalyani	800
B102	Business Statistics	OP Aggrawal	22-Dec-11	Himalaya	750
B103	RDBMS	C J Date	02-Mar-15	THM	900
B104	MGMT Accounting	RK Sharma	19-Apr-16	Kalyani	450
B105	Operating System	Galvin	25-Nov-13	PHI	750
B106	Adv. Accounting	SC Gupta	16-Apr-18	Himalaya	600

create table Library(BookId Varchar(5) Primary key, BookName char(15) Not Null,Author char(10),DatePurchaseddate,Publisher char(10),Price Number(5));

```
SQL> create table Library(BookId Varchar(5) Primary key, BookName char(15) Not Null,Author
char(10), DatePurchased date,Publisher char(10),Price Number(5));
Table created.
insert into Library values('&BookId','&BookName','&Author',to_date('&DatePurchased','DD-
MON-YY'),'&Publisher',&Price);
SQL> insert into Library values('&BookId','&BookName','&Author', to_date('&DatePurchased',
'DD-MON-YY'),'&Publisher',&Price);
Enter value for bookid: B101
Enter value for bookname: Cost Accounting
Enter value for author: J Narang
Enter value for datepurchased: ll-feb-13
Enter value for publisher: Kalyani
Enter value for price: 800
old 1: insert into Library values('&BookId','&BookName','&Author',to_date('&DatePurchased','DD-
MO N-YY' ),'&Publisher',&Price)
new 1: insert into Library values('BlOl','cost Accounting', 'J Narang',to_date('ll-feb-13','DD-MON
-YY'),'Kalyani',800)
1 row created.
SQL> /
Enter value for bookid: B102
Enter value for bookname: Business Stats
Enter value for author: OP Agarwal
Enter value for datepurchased: 22-Dec-II
Enter value for publisher: Himalaya
Enter value for price: 750
old 1: insert into Library values('SBookId','&BookName','SAuthor',to_date('&DatePurchased','DD-
MO N-YY'),'&Publi sher',&Pri ce)
new 1: insert into Library values('B102','Business Stats','OP Agarwal',to_date('22-Dec-ll','DD-MO
N-YY'),'Himalaya',750)
1 row created.
SQL> /
Enter value for bookid: B103
Enter value for bookname: RDBMS
Enter value for author: C J Date
Enter value for datepurchased: 02-mar-15
Enter value for publisher: TMH
Enter value for price: 900
old 1: insert into Library values('&BookId','&BookName','&Author',to_date('&DatePurchased','DD-
MON-YY'),'SPublisher',&Price)
new 1: insert into Library values('B103','RDBMS','C J Date',to_date('02-mar-15','DD-MON-
YY'),'TMH',900)
1 row created.
```

```
SQL>
Enter value for bookid: B104
Enter value for bookname: MGMT Accounting
Enter value for author: RK Sharma
Enter value for datepurchased: 19-Apr-16
Enter value for publisher: Kalyani
Enter value for price: 450
old 1: insert into Library values('&BookId','&BookName','&Author',to_date('&DatePurchased','DD-MO N-YY'),'SPublisher', &Price)
new 1: insert into Library values('B104','MGMT Accounting','RK Sharma',to_date('19-Apr-16','DD-MO N-YY'),'Kalyani',450)
1 row created.
SQL> /
Enter value for bookid: B105
Enter value for bookname: OS
Enter value for author: Galvin
Enter value for datepurchased: 25-Nov-13
Enter value for publisher: PHI
Enter value for price: 750
old 1: insert into Library values('&BookId','&BookName','SAuthor',to_date('&DatePurchased','DD-MON-YY'),'&Publisher',&Price)
new 1: insert into Library values('Bl05','OS','Galvin',to_date('25-Nov-13','DD-MON-YY'),'PHI',750)
1 row created.
SQL> /
Enter value for bookid: B106
Enter value for bookname: Adv.Accounting
Enter value for author: SCGupta
Enter value for datepurchased: 16-Apr-18
Enter value for publisher: Himalaya
Enter value for price: 600
old 1: insert into Library values('&BookId','&BookName','&Author',to_date('&DatePurchased','DD-MON-YY'),'&Publisher',&Price)
new 1: insert into Library values('B106','Adv.Accounting','SCGupta',to_date('16-Apr-18','DD-MON-Y Y'),'Hi malaya',600)
1 row created.
select * from Library;
```

SQL> select * from Library;

BOOK1	BOOKNAME	AUTHOR	DATEPURCH	PUBLISHER	PRICE
B101	Cost Accounting	J Narang	11-FEB-13	Kalyani	800
B102	Business stats	OP Agarwal	22-DEC-11	Himalaya	750
B103	RDBMS	CJ Date	02-MAR-15	TMH	900
B104	MGMT Accounting	R.K. Sharma	19-NOV-13	PHI	750
B105	OS	Galvin	25-NOV-13	PHI	750
B106	Adv. Accounting	SCGupta	16-APR-18	Himalaya	600

6 rows selected

29. Write SQL query to display the list of authors from Himalaya Publications.

Ans :

select * from Library where Publisher='Himalaya';

SQL> select * from Library where Publisher = 'Himalaya';

BOOKI	BOOKNAME	AUTHOR	DATEPURCH	PUBLISHER	PRICE
B102	Business Stats	OP Agarwal	22-DEC-11	Himalaya	750
B106	Adv. Accounting	SC Gupta	16-APR-18	Himalaya	600

30. Write SQL Query to display the total cost of books purchased publishers wise.

Ans :

select sum(Price), Publisher from Library group by Publisher;

SQL> Select sum(price), publisher from library group by publisher;

SUM(PRICE)	PUBLISHER
1350	Himalaya
1250	Kalyani
750	PHI
900	TMH

31. Write SQL Query to count the no.of Books under Kalyani Publications.

Ans :

select count (BookName), Publisher from Library where Publisher = 'Kalyani' group by publisher;

SQL> select count (BookName), Publisher from Library where Publisher = 'Kalyani' group by publisher;

COUNT(BOOKNAME)	PUBLISHER
2	Kalyani

32. Write SQL query to rname the column publisher to Publications.

Ans :

alter table Library Rename column publisher to Publications;

SQL > desc Library;

Name	Null?	Type
BOOKID	NOT NULL	VARCHAR2(5)
BOOKNAME	NOT NULL	CHAR(15)
AUTHOR		CHAR(10)
DATEPURCHASED		DATE
PUBLISHER		CHAR(10)
PRICE		NUMBER(5)

SQL > alter table Library Rename column publisher to Publications;

Table altered.

SQL > desc Library;

Name	Null?	Type
BOOKID	NOT NULL	VARCHAR2(5)
BOOKNAME	NOT NULL	CHAR(15)
AUTHOR		CHAR(10)
DATEPURCHASED		DATE
PUBLICATIONS		CHAR(10)
PRICE		NUMBER(5)

33. Write SQL query to display the books in Ascending order of datepurchased.

Ans :

select * from Library Order by datepurchased ASC;

SQL > select * from Library Order by datepurchased ASC;

BOOK1	BOOKNAME	AUTHOR	DATEPURCH	PUBLICATIO	PRICE
B102	Business Stats	OP Agarwal	22-DEC-11	Himalaya	750
3101	Cost Accounting	J Narang	11-FEB-13	Kalyani	800
B105	OS	Galvi N	25-NOV-13	PHI	750
B103	RDBMS	C J Date	02-MAR-15	TMH	900
B104	MGMT. Accounting	RK Sharma	19-APR-16	Kalyani	450
B106	Adv. Accounting	SC Gupta	16-APR-18	Himalaya	600

6 rows selected.

34. Write SQL query to create Index on the fields BookName and Author.

Ans :

create index LibraryIndex ON Library(BookName,Author);

SQL > create index LibraryIndex ON Library(BookName, Author);

Index created.

35. Write SQL query to display the books whose price is between 500 and 900

Ans :

select * from Library where Price between 500 AND 900;

SQL> select * from Library where Price between 500 AND 900;

BOOK1	BOOKNAME	AUTHOR	DATEPURCH	PUBLICATIO	PRICE
3101	Cost Accounting	J Narang	11-FEB-13	Kalyani	800
B102	Business Stats	OP Agarwal	22-DEC-11	Himalaya	750
B103	RDBMS	C J Date	02-MAR-15	TMH	900
B105	OS	Galvin	25-NOV-13	PHI	750
B106	Adv.Accounti ng	SC Gupta	16-APR-18	Himalaya	600

36. Write a SQL query to increase the price of all books by 200 for publishers other than Himalaya or Kalyani.

Ans :

update Library set Price=Price+200 where NOT publications='Himalaya'AND NOT Publications='Kalyani';

SQL> update Library set Price=Price+200 where NOT publications='Himalaya' AND NOT Publications = 'Kalyani';

2 rows updated.

SQL> select * from Library:

BOOK1	BOOKNAME	AUTHOR	DATEPURCH	PUBLICATIO	PRICE
3101	Cost Accounting	J Narang	11-FEB-13	Kalyani	800
B102	Business Stats	OP Agarwal	22-DEC-11	Himalaya	750
B103	RDBMS	C J Date	02-MAR-15	TMH	1100
B104	MGMT Accounting	RK Sharma	19-APR-16	Kalyani	450
B105	OS	Galvin	25-NOV-13	PHI	950
B106	Adv.Accounting	SC Gupta	16-APR-18	Himalaya	600

6 rows selected.

37. Write SQL query to display the book details where author namecontains the name sharma.

Ans :

select * from Library where Author like '%Sharma%';

SQL> select * from Library where Author like '%Sharma%';

BOOK1	BOOKNAME	AUTHOR	DATEPURCH	PUBLICATIO	PRICE
B104	MGMT Accounting	RK Sharma	19-APR-16	Kalyani	450

38. Create a view to display the fields BookId and BookName where publishers is Himalaya.

Ans :

create view LibraryView as select BookId, BookName from library where Publications = 'Himalaya';

SQL> create view LibraryView as select BookId, BookName from library where Publications = 'Himalaya';

view created.

SQL> select * from Libraryview:

BOOK1	**BOOKNAME**
B102	Business stats
B106	Adv. Accounting

FACULTIES OF COMMERCE

B.Com. III – Semester (CBCS) Examination

Model Paper - I

RELATIONAL DATABASE MANAGEMENT SYSTEM

(Only for Computer Applications Courses)

Time : $1^1/_2$ Hours] **[Max. Marks : 50**

PART - A (5 × 2 = 10 Marks)

[Short Answer Type]

Note: Answer any five of the following questions.

		ANSWERS
1.	Define DBMS.	**(Unit-I, SAQ-2)**
2.	Relational model	**(Unit-I, SAQ-12)**
3.	Define File organization.	**(Unit-II, SAQ-7)**
4.	What is Functional dependency?	**(Unit-II, SAQ-4)**
5.	SQL	**(Unit-III, SAQ-1)**
6.	Defin Transaction.	**(Unit-IV, SAQ-1)**
7.	Define Data Replication.	**(Unit-V, SAQ-16)**
8.	What is Data Mart ?	**(Unit-V, SAQ-11)**

PART - B (5 × 8 = 40 Marks)

[Essay Answer Type]

Note: Answer all from the following questions.

9. (a) What are the advantages of DBMS over File based system ? **(Unit-I, Q.No. 8)**

OR

(b) How can we convert ER Diagram into Relational model ? **(Unit-I, Q.No. 35)**

10. (a) Explain briefly about 1NF, 2NF, and 3NF. **(Unit-II, Q.No. 10, 11, 12)**

OR

(b) Explain different types of file organizations. **(Unit-II, Q.No. 22)**

11. (a) Explain in detail about TCL - Commands. **(Unit-III, Q.No. 12)**

OR

(b) Explain in detail about Nested Queries. **(Unit-III, Q.No. 20)**

12. (a) Define Transaction. Explain the properties of transactions. **(Unit-IV, Q.No. 1, 2)**

OR

(b) Explain briefly about data base recovery. **(Unit-IV, Q.No. 17)**

13. (a) Explain the advantages and disadvantages of Data distribution. **(Unit-V, Q.No. 8, 9)**

OR

(b) Explain briefly about

(i) Mobile Database **(Unit-V, Q.No. 34)**

(i) Multidimensional Database **(Unit-V, Q.No. 43)**

FACULTIES OF COMMERCE

B.Com. III – Semester (CBCS) Examination

Model Paper - II

RELATIONAL DATABASE MANAGEMENT SYSTEM

(Only for Computer Applications Courses)

Time : $1^1/_2$ Hours] **[Max. Marks : 50**

PART - A (5 × 2 = 10 Marks)

[Short Answer Type]

Note: Answer any five of the following questions.

		Answers
1.	Multi value attribute.	**(Unit-I, SAQ-5)**
2.	Define Data Dictionary.	**(Unit-I, SAQ-4)**
3.	Define Decomposition.	**(Unit-II, SAQ-9)**
4.	What is normalization?	**(Unit-II, SAQ-10)**
5.	Differentiate between Roll back and Commit.	**(Unit-III, SAQ-5)**
6.	Optimistic concurrency control.	**(Unit-IV, SAQ-9)**
7.	OLAP	**(Unit-V, SAQ-13)**
8.	DDBMS	**(Unit-V, SAQ-19)**

PART - B (5 × 8 = 40 Marks)

[Essay Answer Type]

Note: Answer all from the following questions.

9. (a) Explain three level architecture of DBMS. **(Unit-I, Q.No. 10)**

OR

(b) Explain in detail about ER – Model ? **(Unit-I, Q.No. 28)**

10. (a) What is Data Redundancy ? How it create Problem in Database ? **(Unit-II, Q.No. 4)**

OR

(b) What are the differences between 3NF and BCNF? **(Unit-II, Q.No. 15)**

11. (a) Explain in detail about DCL commands? **(Unit-III, Q.No. 9)**

OR

(b) Describe briefly about SQL Views? How can we create views in SQL? **(Unit-III, Q.No. 23)**

12. (a) Define lock? Explain different types of locks. **(Unit-IV, Q.No. 5)**

OR

(b) Explain in detail about Database security ? and How authorization process helps you to provide security for the database ? **(Unit-IV, Q.No. 20)**

13. (a) List out the advantages and disadvantages of DDBMS. **(Unit-V, Q.No. 6)**

OR

(b) Define Datawarehouse. Explain the components of Datawarehouse. **(Unit-V, Q.No. 45, 49)**

FACULTIES OF COMMERCE

B.Com. III – Semester (CBCS) Examination

Model Paper - III

RELATIONAL DATABASE MANAGEMENT SYSTEM

(Only for Computer Applications Courses)

Time : $1^1/_2$ Hours] **[Max. Marks : 50**

PART - A (5 × 2 = 10 Marks)

[Short Answer Type]

Note: Answer any five of the following questions.

		Answers
1.	What are the advantages of DBMS ?	**(Unit-I, SAQ-6)**
2.	List out the differences between Generalization and Specialization ?	**(Unit-I, SAQ-11)**
3.	Define Foreign Key	**(Unit-II, SAQ-2)**
4.	Define File Organization	**(Unit-II, SAQ-7)**
5.	Having Clause in SQL.	**(Unit-III, SAQ-4)**
6.	Deadlock	**(Unit-IV, SAQ-4)**
7.	Database Security.	**(Unit-IV, SAQ-8)**
8.	Fragmentation	**(Unit-V, SAQ-17)**

PART - B (5 × 8 = 40 Marks)

[Essay Answer Type]

Note: Answer all from the following questions.

9. (a) Who is DBA ? What is the role and responsibilities DBA ? **(Unit-I, Q.No. 14)**

OR

(b) How can we convert ER Diagram into Relational model ? **(Unit-I, Q.No. 35)**

10. (a) Discuss briefly about storage of data on hard disk. **(Unit-II, Q.No. 20)**

OR

(b) Explain briefly about multilisted and inverted file organization. **(Unit-II, Q.No. 29)**

11. (a) Describe in detail about DDL commands in SQL ? **(Unit-III, Q.No. 7)**

OR

(b) Explain briefly about Joins concept. **(Unit-III, Q.No. 21)**

12. (a) Explain briefly about Serializable schedule ? **(Unit-IV, Q.No. 7)**

OR

(b) Explain in detail about Optimistic concurrency control ? **(Unit-IV, Q.No. 11)**

13. (a) Explain briefly about Distributed Database ? **(Unit-V, Q.No. 1)**

OR

(b) What are the main differences between OLAP and OLTP ? **(Unit-V, Q.No. 55)**

FACULTIES OF COMMERCE

B.Com. VI - Semester (CBCS) Examination

September / October – 2020

RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS)

(Only for Computer Applications)

Time: 2 Hours] [Max. Marks : 80

PART – A (4 × 5 = 20 Marks)

Note : Answer any **four** Questions

1. DBMS **(Unit-I, SQA - 2)**
2. Referential Integrity **(Unit-II, SQA - 1)**
3. SQL **(Unit-III, SQA - 1)**
4. Dead Lock **(Unit-IV, SQA - 4)**
5. Data Replication **(Unit-V, SQA - 16)**
6. Database **(Unit-I, SQA - 1)**
7. Normalization **(Unit-II, SQA - 10)**
8. Client server system **(Unit-V, Q.No. 13)**

PART – B (4 × 15 = 60 Marks)

Note: Answer any **four** Questions

9. Explain advantages of DBMS over file based system. **(Unit-I, Q.No. 8)**
10. Explain Entity Relationship (ER) Model with examples. **(Unit-I, Q.No. 28)**
11. Explain INF, 2NF and 3NF. **(Unit-II, Q.No. 10, 11, 12)**
12. Explain types of File Organization. **(Unit-II, Q.No. 22)**
13. Explain Data Definition Language (DDL) command with examples. **(Unit-III, Q.No. 7)**
14. Explain Joins concept in Structures Query language (SQL). **(Unit-III, Q.No. 21)**
15. Explain Two Phase Locking Protocol. **(Unit-IV, Q.No. 9)**
16. Explain Backup anri Recovery techniques. **(Unit-IV, Q.No. 16)**
17. Explain the advantages and disadvantages of DDBMS. **(Unit-V, Q.No. 6)**
18. Explain need for Client Server Computing. **(Unit-V, Q.No. 15)**

FACULTIES OF COMMERCE

B.Com. VI – Semester (CBCS) Examination

May/June - 2019

RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS)

Time : 3 Hours] **[Max. Marks : 80**

PART - A (5 × 4 = 20 Marks)
[Short Answer Type]

Note: Answer any five of the following questions not exceeding 20 lines each.

1. Database Approach **(Unit-I, Q.No. 9)**
2. Data Redundancy

Ans :

Redundancy is the concept of repetition of data i.e. each data may have more than a single copy. The file system cannot control redundancy of data as each user defines and maintains the needed files for a specific application to run. There may be a possibility that two users are maintaining same files data for different applications.

3. Nested Query **(Unit-III, Q.No. 20)**
4. Database Error **(Unit-IV, SQA - 5)**
5. Distributed Database **(Unit-V, SQA - 14)**
6. Concurrent Transactions **(Unit-IV, Q.No. 3)**
7. Normalization **(Unit-II, SQA - 10)**
8. Primary Key **(Unit-I, Q.No. 22)**

PART - B (5 × 12 = 60 Marks)
[Essay Answer Type]

Note: Answer all the questions

9. a) Explain the DBA functions and Role. **(Unit-I, Q.No. 14, 15)**

OR

b) Explain the advantages of DBMS over file based system. **(Unit-I, Q.No. 8)**

10. a) Explain 1NF, 2NF and 3NF. **(Unit-II, Q.No. 10, 11, 12)**

OR

b) Storage of Database on Hard Disks.

Ans :

Information is stored in the form of a magnetic pattern by using a circular, flat platters. These platters are made of either glass (or) metals on both sides.

The information is stored in tracks are divided into small sectors. The positioning of the arms is done by a device called actuator.

Virtual file allocation table (VFAT) is initially read into windows operating system, when the hard disk is being portioned.

Information Requirements

➢ The information needs to be provided in the form of normalized relations.

➢ Information describing the creation and modification of data.

➢ Information describing the technologies used.

Decisions

➢ Deciding the arrangement of similar records.

➢ Use of indexes to make the retrieval efficient.

11. a) Explain DDL commands. **(Unit-III, Q.No. 7)**

OR

b) Explain joins concept. **(Unit-III, Q.No. 21)**

12. a) Explain optimistic concurrency control. **(Unit-IV, Q.No. 11)**

OR

b) Explain Database Security and Recovery. **(Unit-IV, Q.No. 12)**

13. a) Explain the advantages and disadvantages of DDBMs. **(Unit-V, Q.No. 6)**

OR

b) Explain the structure of Client Server Systems and its advantages. **(Unit-V, Q.No. 16)**

FACULTIES OF COMMERCE

B.Com. VI – Semester (CBCS) (Instant) Examination

September / October - 2019

RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS)

Time : 3 Hours] [Max. Marks : 80

PART - A (5 × 4 = 20 Marks)

Note: Answer any five of the following questions not exceeding 20 lines each.

1. DBMS **(Unit-I, SQA - 2)**
2. Referential Integrity **(Unit-II, SQA - 1)**
3. SQL **(Unit-III, SQA - 1)**
4. Database Recovery **(Unit-IV, Q.No. 17)**
5. Data Replication **(Unit-V, SQA - 16)**
6. Relational Model **(Unit-I, SQA - 12)**
7. Normalization **(Unit-II, SQA - 10)**
8. Client server systems **(Unit-V, Q.No. 13)**

PART - B (5 × 12 = 60 Marks)

Note: Answer ALL the questions in not exceeding 4 pages each.

9. (a) Advantages of DBMS over file based system. **(Unit-I, Q.No. 8)**

OR

(b) Explain Entity Relationship model. **(Unit-I, Q.No. 28)**

10. (a) Explain First, Second and Third Normal Form. **(Unit-II, Q.No. 10, 11, 12)**

OR

(b) Explain types of File organization. **(Unit-II, Q.No. 22)**

11. (a) Explain Data Manipulation language. **(Unit-III, Q.No. 8)**

OR

(b) Explain views with example. **(Unit-III, Q.No. 23)**

12. (a) Explain Deadlock and its prevention. **(Unit-IV, Q.No. 10)**

OR

(b) Explain Database Security and Authorization. **(Unit-IV, Q.No. 20)**

13. (a) Explain need for Distributed Database systems. **(Unit-V, Q.No. 4)**

OR

(b) Explain client server system and its advantages. **(Unit-V, Q.No. 16)**

FACULTIES OF COMMERCE

B.Com. (CBCS) II – Semester Examination

May/June - 2018

RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS)

(Only for Computer Applications Courses)

Time : 3 Hours] **[Max. Marks : 80**

PART - A (5 × 4 = 20 Marks)

Note: Answer any **FIVE** of the following questions not exceeding 20 lines each

ANSWERS

1. Database Management System **(Unit-I, SQA - 2)**

2. Entity

Ans :

An entity is a real-world object that is distinguishable from other objects. In ER-diagram an entity is represented by a rectangel (entity box). The name of the entity is a noun and it is written in the centre of rectangle. Whenever ER-diagram is applied to relational modl, an entity is mapped to relational table wherein each row represents an entity instance.

3. Normalization **(Unit-II, SQA - 10)**

4. Index **(Unit-II, Q.No. 23)**

5. SQL **(Unit-III, SQA - 1)**

6. Backup **(Unit-IV, SQA - 6)**

7. Data Fragmentation **(Unit-V, SQA - 17)**

8. Data Replication **(Unit-V, SQA - 16)**

PART - B (5 × 12 = 60 Marks)

Note: Answer all the questions in not exceeding 4 pages each.

9. (a) Explain advantages of DBMS over File Oriented System. **(Unit-I, Q.No. 8)**

OR

(b) Explain entity Relationship (ER) Model. **(Unit-I, Q.No. 28)**

10. (a) Explain First Normal Form and Second Normal Form. **(Unit-II, Q.No. 10, 11)**

OR

(b) Explain File Organization and its types. **(Unit-II, Q.No. 21, 22)**

11. (a) Explain insert, update command with examples. **(Unit-III, Q.No. 8)**

OR

(b) Explain data definition language. **(Unit-III, Q.No. 7)**

12. (a) Explain deadlock and its prevention. **(Unit-IV, Q.No. 10)**

OR

(b) Explain database recovery and database security. **(Unit-IV, Q.No. 16, 18)**

13. (a) Explain advantages and disadvantages of RDBMS.

Ans :

Advantages

i) It is easy to design, manage and implement.

ii) It allows the backup and recovery of the data.

iii) It provides minimum level of redundancy.

iv) It maintains data integrity.

v) It supports flexible and easy to use adhoc query capability.

Disadvantages

i) It does not recover the lost data easily.

ii) The implementation of RDBMS needs skill full human resources.

iii) It is a slow process in case of logic programming language.

OR

(b) Explain need for client server computing. **(Unit-V, Q.No. 15)**

FACULTY OF COMMERCE

B.Com (CBCS) III - Semester Examination

July - 2021

RELATIONAL DATABASE MANAGEMENT SYSTEMS

Time : $1^{1/2}$ Hours] **[Max. Marks : 50**

PART - A - (5 × 2 = 10 Marks)

ANSWERS

Note : Answer the following any Five Question.

1.	DBA	**(Unit - I, SQA-8)**
2.	Referential Integrity	**(Unit - II, SQA-1)**
3.	Where clause	**(Unit - III, Q.No.16)**
4.	Database Security	**(Unit - IV, SQA-8)**
5.	Data Fragmentation	**(Unit - V, SQA-17)**
6.	Database Error	**(Unit - IV, SQA-15)**
7.	Nested Query	**(Unit - III, SQA-20)**
8.	Composite attribute	

Ans :

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

PART - B - (5 × 8 = 40 Marks)

Note : Answer the following Five Question.

9.	Explain about logical DBMS architecture.	**(Unit - I, Q.No.10)**
10.	Explain Entity Relationship (ER) Model with examples.	**(Unit - I, Q.No.28)**
11.	What is Normalization? Explain 1 NF and 2 NF	**(Unit - II, Q.No.8, 10, 11)**
12.	What is File Organization? Explain different types of File Organization.	**(Unit - I, Q.No.18,22)**
13.	Explain the various DDL commands used in SQL with examples.	**(Unit - III, Q.No.7)**
14.	What are Views and Sequences?	**(Unit - III, Q.No.23, 24)**
15.	What is a Lock? Explain Two Phase Locking (2PL)	**(Unit - IV, Q.No.5,9)**
16.	Explain the database security and the database recovery.	**(Unit - IV, Q.No.17,20)**
17.	Explain the structure of Distributed Database.	**(Unit - V, Q.No.5)**
18.	Explain the need for Client Server computing.	**(Unit - V, Q.No.15)**