Relational Database Management Systems – Oct 2015

I. Section-A: **5 X 4 =20 Marks**

1. Data Consistency

Files and application programs are created by different programmers over a long period of time, the files are likely to be having different formats and the programs may be written in several programming languages. Moreover, the same piece of information may be duplicated in several places. This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency, i.e. the various copies of same data may no longer agree.

If the amount of data redundancy is controlled, it will reduce data inconsistency also. It is also highly recommended to maintain the same version of the data at all the locations. For example, when a customer address is stored at only one location, if the customer changes the address, it will be automatically reflected in all the applications related to that particular customer.

2. Database.

Database is a collection of inter-related data which contains the information of an enterprise. It is obtained by collecting the data from all sources of the organization. Database can be shared, integrated and managed with other computer applications

3. Relational

Relational algebra is a formal language describing how new relations are created from old ones. It is a useful tool for describing queries on a database management system. Each row in the table represents one tuple from the relationship, and each column one attribute. Relational algebra is expressed as a language that can be used to describe operations for creating new tables from existing tables in a database management system.
If the tables in the database as termed as "*existing relations*", and the result set of the query as the "*new relations*", then relational algebra is a language that can be used to describe data base queries that will return a result set from the existing database.

result set = query (existing database)

4. Dual Table in Oracle

Dual is a table that is created by Oracle together with data dictionary. It is suitable for in selecting a pseudo column such as SYSDATE or USER. The table has a single VARCHAR2(1) column called DUMMY that has a value of 'X'.

DUAL was originally a table and the database engine would perform disk IO on the table when selecting from DUAL. This disk IO was usually logical IO (not involving physical disk access) as the disk blocks were usually already cached in memory. This resulted in a large amount of logical IO against the DUAL table.

```
SQL> desc dual

Name                          Null?    Type
-----------------------       -------- ----------------
DUMMY                                  VARCHAR2 (1)
```

The DUAL table is used because the relational model does not have a placeholder for calculations because every command must be a SQL statement.

Note that the outer query references the dummy table called "dual" the dual table is used in Oracle when one need to run SQL that does not logically have a table name. Example, To select the current date from dual.

```
select sysdate from dual;

SYSDATE
---------
15-OCT-2015
```

5. Schema

A database schema is described in a formal language supported by the database management system (DBMS). In a relational database, the schema defines the tables, the fields in each table, and the relationships between fields and tables. Schemas are generally stored in a data dictionary. Although a schema is defined in text database language, the term is often used to refer to a graphical depiction of the database structure.

A Data Schema aims to identify and describe a thematic set of core variables that are of particular value in a specified scientific setting, and which are recommended to optimize potential data sharing between bio banks. Data Schema 's can facilitate data sharing for an emerging group of studies which will work together on a specific topic, or to provide a tool for existing studies which intend to share data. Each Data Schema is comprised of variables that may be derived from a number of different data sources

Each Data Schema includes:

1. A list of variables, each with an associated description, definition and format.
2. A list of domains, each with an associated definition;
3. A list of themes and modules with their respective definitions.

6. Backup

Database backups can be either physical or logical. **Physical backups**, which are the primary concern in a backup and recovery strategy, are copies of physical database files. A DBA is responsible to make physical backups with RMAN or operating system utilities.

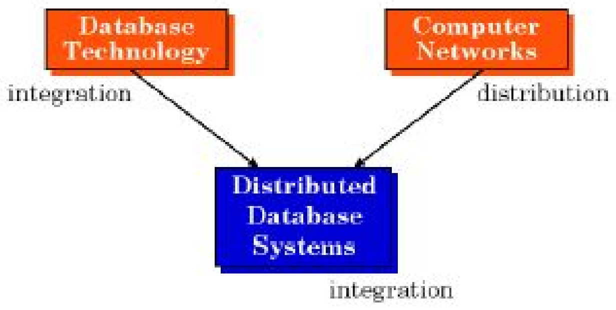
In contrast, **logical backups** contain logical data such as tables and stored procedures. DBA can extract logical data with an Oracle Database utility such as Data Pump Export and store it in a binary file. Logical backups can supplement physical backups.

Physical backups have large granularity and limited transportability, but are very fast. Logical backups have fine granularity and complete transportability, but are slower than physical backups.

7. Distributed Database

A distributed database is integrated database which is built on top of a computer network rather than on a single computer. The data which constitute the database are stored at the different sites of the computer network, and the application programs which are run by the computers access data at different sites. Databases may involve different database management systems, running on different architectures, that distributes the execution of transactions.

A Distributed Database Management System (DDBMS) is defined as the software that handles the management of the DDB (Distributed Database) and makes the operation of such a system appear to the user as a centralized database.



8. Data Integrity

A distributed database system differs from a centralized database system in that its database resides at a set of sites S. As might be expected, control data integrity becomes a harder problem in the network environment. Transactions are no longer linearly ordered sequences of actions on data. As the data is distributed, the transaction activities may take place at a number of sites, and it can be difficult to maintain a time ordering among actions.

The most common problem is when two (or more) transactions are executing at the same time, and both require access to the same data record in order to complete their processing. There may be multiple copies of the same record. All copies must have the same value at all times, or else transactions may operate on inaccurate data in DDB.
Most concurrency control algorithms for distributed database systems use some form of check to see that the result of a transaction is the same as if its actions were executed serially.
To implement concurrency control, the following must be known:

1. The type of scheduling algorithm used.
2. The location of the scheduler.
3. How replicated data are controlled.

II. Section-B: **5 X 10 =50 Marks**

9(a). Explain Data Models

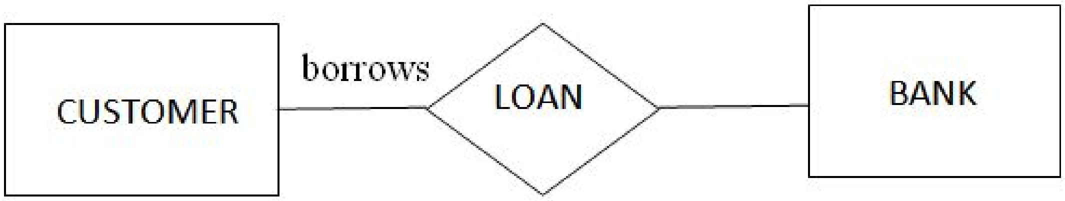
Data Model: A conceptual method of structuring data is called Data Model.
The development of systems based on following data models. They are

1. Entity-Relationship Model
2. Object Oriented Model
3. Relational Model
4. Hierarchical Model
5. Network Model

Entity-Relationship Model:
The entity-relationship data model is based on real world objects called entities and relationship among these objects. E-R model is represented graphically by an E-R diagram. The following are the components of E-R diagram.

1. Rectangle represents Entity set.
2. Ellipses represent attributes.
3. Diamonds represents relationships among entity sets.

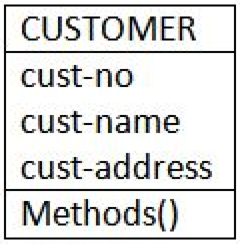
4. Lines represent link attributes to entity sets and entity sets to Relationships.



Object Oriented Model:

The object-oriented model is based on a collection of objects, like the E-R model.

1. An object contains values stored in **instance variables** within the object.
2. An object also contains bodies of code that operate on the object. These bodies of code are called **methods**.
3. Objects that contain the same types of values and the same methods are grouped into **classes**.



Relational Model:
The relational model uses a collection of tables to represent both data and relationships Among those data, each table has multiple columns and each column has a unique name. Here relation refers to a two dimensional table containing rows and columns of data.

CUSTOMER TABLE

Cust-no	**Cust-name**	**Cust-address**
11500	ABC	Hyderabad
11501	DEF	Mumbai
11502	IJK	Chennai

Hierarchical Model:

Data in the hierarchical model are represented b collection of record and relationship among data are represented by links. The records in the database are organized as Collection of trees

Network Model:
Data in the network model are represented as collection of records and relationships among data are represented by lines. The records in the database are organized as collections of arbitrary graphs.

9(b). Write about advantages of

Program Data Independence:

If a database approach is used, data is stored in a central location called repository. The process of database allows an enterprise's data to change the database without modifying the application programs which are able to process this data. This allows the separation of database from the application programs.

Minimal Data Redundancy:

Data Redundancy exists when the same data are stored unnecessarily at different places. The design with the database approach is that separate data files are integrated into a single structure. The database approach does not eliminate redundancy completely, but it provides the facilities to the designer to carefully control the amount of redundancy.

Improved Data Consistency:

If the amount of data redundancy is controlled, it will reduce data inconsistency also. It is also highly recommended to maintain the same version of the data at all the locations. For example, when a customer address is stored at only one location, if the customer changes the address, it will be automatically reflected in all the applications related to that particular customer.

Improved Data Sharing:

A database is designed as a sharable component. The DBMS helps in creating an environment in which end users have better access to more data and better manages data. Such access makes it possible for end users to respond quickly to change in their environment. Users are allowed to utilize the services of the database by authentication and authorization.

Enforcements of Standards:

To facilitate the services of database management, every database administrator designs establishing procedures and enforcement of standards. Procedures are the instructions and rules that govern the design and use of the database system. Procedures play an important role in an enterprise because they enforce the standards by which business is generated within the organization and with customers. Procedures are also used to ensure that there is an organized way to monitor and audit both the data and the information that is generated through the use of the data.

Improved Quality:

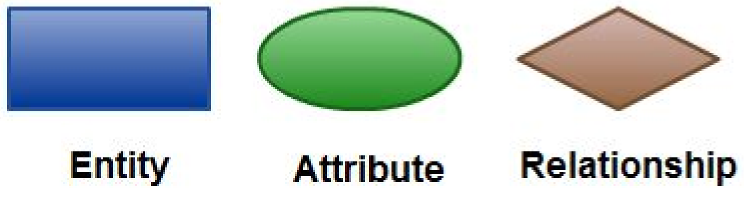
The database approach provides an optimum number of tools and processes to improve data quality. Every data designer can specify a rule called integrity constraint which users cannot violate. The availability of data combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between the success and failure in the global economy.

10(a). Explain E-R diagram with an example.

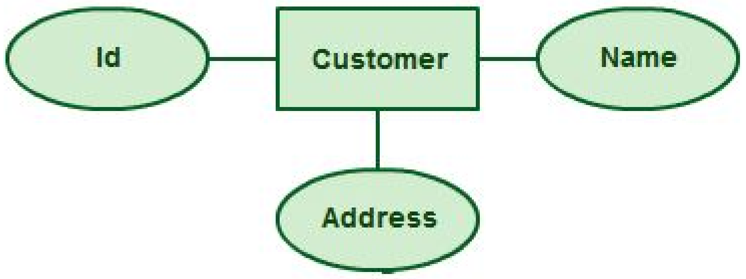
The E-R (Entity-Relationship) model views the real world as a set of basic objects and **relationships** among these objects. This represents the overall logical structure of the database.

The basic elements of E-R Model are Entity, Attribute and Relationship.

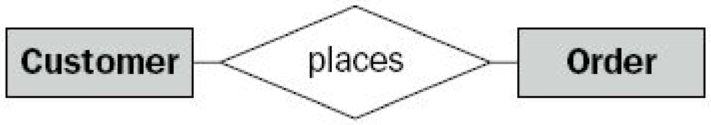
E-R Diagram: E-R Model is pictorial represented as E-R Diagram. The notations of this diagram is as follows:



An Entity (Customer) contains Attributes (Id, Name, Address) is represented as follows:



Two Entities (Customer) & (Order) are related as follows:



The relationship between the two entities is given the name "place". It gives the meaning of the Customer places an Order.

An Entity is a real-world object which is distinguishable from other objects. An entity may be defined as a thing which is recognized as being capable of an independent existence and which can be uniquely identified. Preferably, every entity must have a minimal set of uniquely identifying attributes, which is called the entity's primary key.

General examples for entities are:

- A physical object like computer or bike.
- A human-being
- An organization or a department of an organization
- Country or a place e.g. Hyderabad
- Software like Oracle, MS-Access etc.

Entity Set: A collection of similar entities is called Entity Set or Entity Type.
Some examples of Entity Set are:

- All employees of an organization, EMP.
- All students of a college or university, STUDENT.
- All departments of a university, DEPT.

Attribute: An Attribute is a property or characteristic of an entity. In other words, An Entity is described using Attributes in database. All entities in an entity set have the same set of attributes.
Some examples of Attributes:

- EMP contains emp-no, emp-name, sal
- STUDENT contains student-no, student-name, course
- DEPT contains dept-no, dept-name.

Relationship:

A Relationship is an association of two or more entities. A meaningful relationship is called Relationship Set or Relationship Type. Relationships can be classified by their degree.

Some examples of relationship set are:

- Working is the relationship between EMP and DEPT.
- Borrows is the relationship between CUSTOMER and LOAN.
- Performs is the relationship between ARTIST and SONG.

10(b). Explain Normalization and Comparison of BCNF & 3NF

Normalization is the process of converting a relation into standard form.

Un-Normalized Form (UNF): If a table contains non-atomic values at each row, it is said to be in UNF. An atomic value is something that cannot be further decomposed. A non-atomic value, as the name suggests, can be further decomposed and simplified.

EmpNo	**EmpName**	**Month**	**Sales**	**Bankno**	**BankName**
100	ABC	Jan	1000	B01	SBI
		Feb	1200		
		Mar	850		
101	DEF	Jan	2200	B02	HDFC
		Feb	2500		
102	XYZ	Jan	1700	B01	SBI

In the above table, there are multiple occurrences of rows under each key EmpNo. Although considered to be the primary key, EmpNo cannot give us the unique identification facility for any single row. Further, each primary key points to a variable length record(3 for Empno.100 , 2 for Empno.101 and 1 for Empno.102).

3rd Normal Form: A table is in third normal form (3NF) if and only if it is in 2NF and every non key attribute is non transitively dependent on the primary key (i.e. there are no transitive dependencies)

1. Anomalies can occur when a relation contains one or more transitive dependencies.
2. A relation is in 3NF when it is in 2NF and has no transitive dependencies.
3. A relation is in 3NF when 'All non-key attributes are dependent on the key, the whole key and nothing but the key'.

Boyce- Codd Normal Form: A relationship is said to be in BCNF if it is already in 3NF and every key is a candidate key. A relation which is in 3NF is almost always in BCNF. This could be same situation when a 3NF relation may not be in BCNF the following conditions are found true.

1. The candidate keys are composite.
2. There are more than one candidate keys in the relation.
3. There are some common attributes in the relation.

11(a). Explain Table Creation with example in SQL.

Table: Table are defined in three steps.

1. The name of the table is given.
2. Each column is defined, possibly including column constraints.
3. Table constraints are defined.

```
Create table Tablename(column-name1, data type(number of characters))
                     (column-name2, data type(number of characters))
                      …………………………………………………..
                     (column-name, data type(number of characters))
```

11(b). Explain operators present in SQL.

An operator manipulates individual data items and returns a result. The data items are called *operands* or *arguments*. Operators are represented by special characters or by keywords. For example, the multiplication operator is represented by an asterisk (*) and the operator that tests for nulls is represented by the keywords IS NULL. There are two general classes of operators: unary and binary. SQL also supports with set operators.

UNARY OPERATORS:

A unary operator uses only one operand. A unary operator typically appears with its operand in the following format:

Syntax:- operator operand

BINARY OPERATOR

A binary operator uses two operands. A binary operator appears with its operands in the following format:

Syntax:- operand1 operator operand2

CHARACTER OPERATORS
Character operators are used in expressions to manipulate character strings.

Syntax:- Character Operators

Operator : ||

Description : Concatenates character strings

Example: SELECT 'The Name of the employee is: ' || ENAME FROM EMP;

SQL* ARITHMETIC *Operators:

Operator	**Description**	**Example**
+ (unary)	Makes operand positive	SELECT +3 FROM DUAL;
- (unary)	Negates operand	SELECT -4 FROM DUAL;
/	Division (numbers and dates)	SELECT SAL / 10 FROM EMP;
*	Multiplication	SELECT SAL * 5 FROM EMP;
+	Addition (numbers and dates)	SELECT SAL + 200 FROM EMP;
-	Subtraction (numbers and dates)	SELECT SAL - 100 FROM EMP;

SQL COMPARISON OPERATORS

Operator	**Description**	**Example**
=	Equality test.	SELECT ENAME "Employee" FROM EMP WHERE SAL = 1500;
!=, ^=, <>	Inequality test.	SELECT ENAME FROM EMP WHERE SAL ^= 5000;
>	Greater than test.	SELECT ENAME "Employee", JOB "Title" FROM EMP WHERE SAL > 3000;
<	Less than test.	SELECT * FROM PRICE WHERE MINPRICE < 30;
>=	Greater than or equal to test.	SELECT * FROM PRICE WHERE MINPRICE >= 20;
<=	Less than or equal to test.	SELECT ENAME FROM EMP WHERE SAL <= 1500;
IN	"Equivalent to any member of" test. Equivalent to "= ANY".	SELECT * FROM EMP WHERE ENAME IN ('SMITH', 'WARD');
ANY/ SOME	Compares a value to each value in a list or returned by a query. Must be	SELECT * FROM DEPT WHERE LOC = SOME ('NEW YORK','DALLAS');

	preceded by =, !=, >, <, <=, or >=. Evaluates to FALSE if the query returns no rows.	
NOT IN	Equivalent to "!= ANY". Evaluates to FALSE if any member of the set is NULL.	SELECT * FROM DEPT WHERE LOC NOT IN ('NEW YORK', 'DALLAS');
ALL	Compares a value with every value in a list or returned by a query. Must be preceded by =, !=, >, <, <=, or >=. Evaluates to TRUE if the query returns no rows.	SELECT * FROM emp WHERE sal >= ALL (1400, 3000);
[NOT] BETWEEN *x* and *y*	[Not] greater than or equal to *x* and less than or equal to *y*.	SELECT ENAME, JOB FROM EMP WHERE SAL BETWEEN 3000 AND 5000;
EXISTS	TRUE if a sub-query returns at least one row.	SELECT * FROM EMP WHERE EXISTS (SELECT ENAME FROM EMP WHERE MGR IS NULL);
x [NOT] LIKE *y* [ESCAPE *z*]	TRUE if *x* does [not] match the pattern *y*. Within *y*, the character "%" matches any string of zero or more characters except null. The character "_" matches any single character. Any character following ESCAPE is interpretted litteraly, useful when *y* contains a percent (%) or underscore (_).	SELECT * FROM EMP WHERE ENAME LIKE '%E%';
IS [NOT] NULL	Tests for nulls. This is the only operator that should be used to test for nulls.	SELECT * FROM EMP WHERE COMM IS NOT NULL AND SAL > 1500;

SQL LOGICAL OPERATORS

NOT	Returns TRUE if the following condition is FALSE. Returns FALSE if it is TRUE. If it is UNKNOWN, it remains UNKNOWN.	SELECT * FROM EMP WHERE NOT (job IS NULL)
		
		SELECT * FROM EMP WHERE NOT (sal BETWEEN 1000 AND 2000)

AND	Returns TRUE if both component conditions are TRUE. Returns FALSE if either is FALSE; otherwise returns UNKNOWN.	SELECT * FROM EMP WHERE job='CLERK' AND deptno=10
OR	Returns TRUE if either component condition is TRUE. Returns FALSE if both are FALSE. Otherwise, returns UNKNOWN.	SELECT * FROM emp WHERE job='CLERK' OR deptno=10

SET OPERATORS:-

Set operators combine sets of rows returned by queries, instead of individual data items. All set operators have equal precedence.

Operator	Description	Example
UNION	Returns all distinct rows selected by either query.	SELECT * FROM (SELECT ENAME FROM EMP WHERE JOB = 'CLERK' UNION SELECT ENAME FROM EMP WHERE JOB = 'ANALYST');
UNION ALL	Returns all rows selected by either query, including all duplicates.	SELECT * FROM (SELECT SAL FROM EMP WHERE JOB = 'CLERK' UNION SELECT SAL FROM EMP WHERE JOB = 'ANALYST');
INTERSECT and INTERSECT ALL	Returns all distinct rows selected by both queries.	SELECT * FROM orders_list1 INTERSECT SELECT * FROM orders_list2
MINUS	Returns all distinct rows selected by the first query but not the second.	SELECT * FROM (SELECT SAL FROM EMP WHERE JOB = 'PRESIDENT' MINUS SELECT SAL FROM EMP WHERE JOB = 'MANAGER');

12(a). Explain types of File Organization.

There are three basic ways of physically organizing files on storage devices.

- Sequential organization
- Indexed-Sequential organization
- Direct organization.

The terms organization and access are often used loosely if not interchangeably. The reason is that the way in which data are stored is closely intertwined with the method access.

1. Sequential File Organization: Sequential file organizational means that records are stored adjacent to one another according to a key such as employee number, account number, and so forth. A conventional implementation arranges the records in ascending order of key values. This is efficient method of organizing records when an application. Such as a payroll program, will be updating a significant number of the stored records.

If a sequential file is maintained on magnetic tape, its records can only be accessed in a sequential manner. That is, if access to the tenth record in sequence is desired generally the preceding nine records must be read. Direct access of a particular record is impossible. Consequently magnetic tapes are not well suited for database operations and are usually relegated log files and recording archival information.

2. Indexed- Sequential File Organization : - When files are sequentially organized on a disk pack, however, direct access of records is possible. Indexed-sequential file organization provides facilities for accessing records both sequentially and directly. Records are stored in the usual physical sequence by primary key. In addition an index of record locations is stored on the disk. This allows records to be accessed sequentially for applications requiring the updating of large numbers of records, as well as providing the ability to access records directly in response to user queries.

3. Direct File Organization: The third type of file organization is called direct or hashed.
It is of two types:

- Static Hash Function
- Dynamic Hash Function

(a). Static Hash Function: The use of hashing is a method of record addressing that eliminates the need for maintaining and searching indexes. The basic idea is that of trading the time and effort associated with storing, maintaining, and searching an index for the time required for the central processing unit (CPU) to execute a hashing algorithm, which generates the record address. The hashing algorithm is a procedure for calculating a record address from some field in the record, usually the key.

(b). Dynamic hash functions: The static hash function is fairly simple. As the database grows, however, the static hash function loses its appeal. One strategy for dealing with this problem is to allocate estimated space for future requirements at the outset; but this wastes storage space. Another scheme is to allocate additional storage and reorganize the file as it grows.

A better approach is provided by the dynamic hash function. This hashing splits and combines blocks as the database grows or shrinks. This ensures efficient space utilization. Moreover, since reorganization involves only one block at a time, the associated overhead is minimal.

Dynamic hashing uses a hash function "h" that has the useful characteristics of randomness and uniformity. It also (typically) uses a 32-bit binary string in order to create and identify block indexes.

12(b). Explain functions of DBA.

Database Administrator is a person with the responsibility of controlling and protecting the data. The **DBA** should coordinate the design of the database, guide the development and implementation of data security procedures, protect the integrity of data values and make sure system performance is satisfactory.

In a small organization, one person carries out all these responsibilities. Often, these functions are assigned to a group of people. This is most likely in a large organization where DBA responsibilities are divided among several people managed by a chief administrator.

Functions of DBA:

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

2. **Storage structure and access-method definition**: writing a set of definitions translated by the data storage and definition language compiler

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

4. **Granting of authorization for data access**. By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.

Integrity constraint specification: generating integrity constraints. These are consulted by the database manager module whenever updates occur

13(a). Define Client-Server Systems

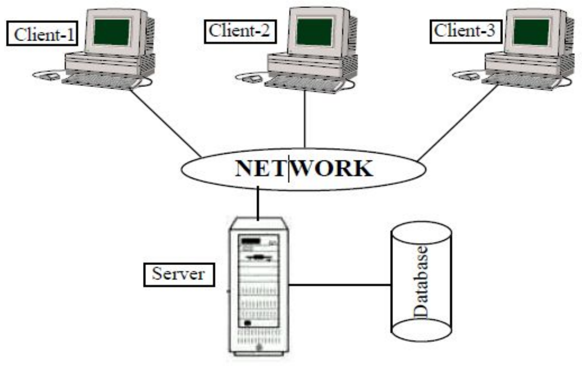
The term client/server was first used in the 1980s in reference to personal computers (PCs) on a network. The actual client/server model started gaining acceptance in the late 1980s. The client/server software architecture is a versatile, message-based and modular infrastructure that is intended to improve usability, flexibility, interoperability and scalability. A client is defined as a requester of services and a server is defined as the provider of services.

A single machine can be both a client and a server depending on the software configuration. In a network environment, a file server stores files required by users on the network. When users need data from a file, the entire file is sent. In client/server architecture, the server is a computer providing data to the clients, which are the computers that are connected to a network and that people use to access data stored on the server. The DBMS runs on the server and a client sends a request for specific data to the server. Only the necessary data and not the entire file or files are sent.

Client/server architecture may be either two-tier or three-tier. In a two-tier architecture, the server performs database functions and the clients perform the presentation (user interface) functions. Either the server or the clients may perform business functions. The term fat client refers to an arrangement where the clients perform the business functions. If the business functions reside on the server, each client is called a thin client. In a three-tier architecture, the clients perform the presentation functions, a database server performs the database functions, and separate computers, called application servers, perform the business functions and act as interface between clients and database server. The advantages to using a client/server system instead of a file server are:

- Lower network traffic
- Improved processing distribution
- Thinner clients
- Greater processing transparency
- Increased network
- Hardware and software transparency
- Improved security
- Decreased costs and increased scalability.

Triggers, which are actions that occur automatically in response to associated database operations, provide additional integrity support.



13(b). Explain need of Distributed database

A database that is distributed among a network of geographically separated locations. A distributed database is not entirely stored in one central location but is distributed among a network of locations that are geographically separated and connected by communication links. Each location has its own database and it also able to access data maintained at other locations.

The reasons for the development and use of distributed database systems are several and include the following:

Need of Distributed Database: -

1. Often organizations have branches or divisions in different locations. For a given location, L, there may be a set of data that is used frequently perhaps exclusively, at L. In addition, L may sometimes need data that are used more frequently at another location, L.
2. Allowing each site to store and maintain its own database allows immediate and efficient access to data that are used most frequently. Such data may be used at others site as well, but usually with less frequency. Similarly, data stored at other locations can be accessed as required.
3. Distributed database can upgrade reliability. If one site's computer fails, or if a communication link goes down, the rest of the network can possibly continue functioning. Moreover when data are replicated at two or more sites, required data may still be available from a site, which is still operate.
4. Allowing local control over the data used most frequently at a site can improve user satisfaction with the database system. That is to say, local database can more nearly reflect an organization's administrative structure and thereby better service its manager's needs.