Q1) Database Normalization

In relational database design, we not only want to create a structure that stores all of the data, but we also want to do it in a way that minimize potential errors when we work with the data. The default language for accessing data from a relational database is SQL. In particular, SQL can be used to manipulate data in the following ways: insert new data, delete unwanted data, and update existing data. Similarly, in an un-normalized design, there are 3 problems that can occur when we work with the data:

INSERT ANOMALY: This refers to the situation when it is impossible to insert certain types of data into the database.

DELETE ANOMALY: The deletion of data leads to unintended loss of additional data, data that we had wished to preserve.

UPDATE ANOMALY: This refers to the situation where updating the value of a column leads to database inconsistencies (i.e., different rows on the table have different values).

To address the 3 problems above, we go through the process of normalization. When we go through the normalization process, we increase the number of tables in the database, while decreasing the amount of data stored in each table. There are several different levels of database normalization:

- 1st Normal Form (1NF)
- 2nd Normal Form (2NF)
- 3rd Normal Form (3NF)
- Bryce-Codd Normal Form (BCNF)
- 4th Normal Form (4NF)
- 5th Normal Form (5NF)

The opposite of normalization is denormalization, where we want to combine multiple tables together into a larger table. Denormalization is most frequently associated with designing the fact table in a data warehouse.

1st Normal Form Definition

A database is in first normal form if it satisfies the following conditions:

- Contains only atomic values
- There are no repeating groups

An atomic value is a value that cannot be divided. For example, in the table shown below, the values in the [Color] column in the first row can be divided into "red" and "green", hence [TABLE_PRODUCT] is not in 1NF.

A repeating group means that a table contains two or more columns that are closely related. For example, a table that records data on a book and its author(s) with the following columns: [Book ID], [Author 1], [Author 2], [Author 3] is not in 1NF because [Author 1], [Author 2], and [Author 3] are all repeating the same attribute.

1st Normal Form Example

How do we bring an unnormalized table into first normal form? Consider the following example

TABLE_PRODUCT

Product ID	Color	Price
1	red, green	15.99
2	yellow	23.99
3	green	17.50
4	yellow, blue	9.99
5	red	29.99

This table is not in first normal form because the [Color] column can contain multiple values. For example, the first row includes values "red" and "green."

To bring this table to first normal form, we split the table into two tables and now we have the resulting tables:

TABLE_PRODUCT_PRICE

Product ID	Price
1	15.99
2	23.99
3	17.50
4	9.99
5	29.99

TABLE_PRODUCT_COLOR

Product ID	Color
1	red
1	green
2	yellow
3	green
4	yellow
4	blue
5	red

Now first normal form is satisfied, as the columns on each table all hold just one value

2nd Normal Form Definition

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

- It is in first normal form
- All non-key attributes are fully functional dependent on the primary key

In a table, if attribute B is functionally dependent on A, but is not functionally dependent on a proper subset of A, then B is considered fully functional dependent on A. Hence, in a 2NF table, all non-key attributes cannot be dependent on a subset of the primary key. Note that if the primary key is not a composite key, all non-key attributes are always fully functional dependent on the primary key. A table that is in 1st normal form and contains only a single key as the primary key is automatically in 2nd normal form.

2nd Normal Form Example

Consider the following example:

TABLE_PURCHASE_DETAIL

Customer ID	Store ID	Purchase Location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angeles
3	2	New York
4	3	San Francisco

This table has a composite primary key [Customer ID, Store ID]. The non-key attribute is [Purchase Location]. In this case, [Purchase Location] only depends on [Store ID], which is only part of the primary key. Therefore, this table does not satisfy second normal form.

To bring this table to second normal form, we break the table into two tables, and now we have the following:

TABLE_PURCHASE

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

TABLE_STORE

Store ID	Purchase Location
1	Los Angeles
2	New York
3	San Francisco

What we have done is to remove the partial [illegible] cy that we initially ha
the table [TABLE_STORE], the column [Purchase Location] is fully dependent on th
key of that table, which is [Store ID].

3rd Normal Form Definition

A database is in third normal form if it satisfies the following conditions:

- It is in second normal form
- There is no transitive functional dependency

By transitive functional dependency, we mean we have the following relationships in the
A is functionally dependent on B, and B is functionally dependent on C. In this case
transitively dependent on A via B.

3rd Normal Form Example

Consider the following example:

TABLE_BOOK_DETAIL

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

In the table able, [Book ID] determines [Genre ID], and [Genre ID] determines [Genre Ty
Therefore, [Book ID] determines [Genre Type] via [Genre ID] and we have transitive functio
dependency, and this structure does not satisfy third normal form.

To bring this table to third normal form, we split the table into two as follows:

TABLE_BOOK

Book ID	Genre ID	Price
1	1	25.99
2	2	14.99
3	1	10.00
4	3	12.99
5	2	17.99

TABLE_GENRE

Genre ID	Genre Type
1	Gardening
2	Sports
3	Travel

Now all non-key attributes are fully functional dependent only on the primary key. In
[TABLE_BOOK], both [Genre ID] and [Price] are only dependent on [Book ID]. In
[TABLE_GENRE], [Genre Type] is only dependent on [Genre ID]

Q2) what are advantage of database over file processing system? Explain

Advantages of DBMS over file system

Drawbacks of File system:

- Data Isolation: Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.
- Duplication of data – Redundant data
- Dependency on application programs – Changing files would lead to change in application programs.

Advantage of DBMS over file system

There are several advantages of Database management system over file system. Few of them are as follows:

Flexibility: Because programs and data are independent, programs do not have to be modified when types of unrelated data are added to or deleted from the database, or when physical storage changes.

• **Fast response to information requests:** Because data is integrated into a single database, complex requests can be handled much more rapidly than locating data separately. In many businesses, faster response means better customer service.

• **Multiple access:** Database software allows data to be accessed in a variety of ways (through various key fields), by using several programming languages (both3GL and nonprocedural4GL programs).

• **Lower user training costs:** Users often find it easier to learn such systems and training costs may be reduced. Also, the total time taken to process requests may be less, which would increase user productivity.

• **Less storage:** Theoretically, all occurrences of data items need be stored only once, thereby eliminating the storage of redundant data. System developers and database designers often use data normalization to minimize data redundancy.

Disadvantages of DBMS:

- DBMS implementation cost is high compared to the file system
- Complexity: Database systems are complex to understand
- Performance: Database systems are generic, making them suitable for various applications. However this feature affect their performance for some applications

Q3) Explain Physical Database Design concept?

Physical database design translates the logical data model into a set of SQL statements that define the database. For relational database systems, it is relatively easy to translate from a logical data model into a physical database.

Rules for translation:

- Entities become tables in the physical database.
- Attributes become columns in the physical database. Choose an appropriate data type for each of the columns.
- Unique identifiers become columns that are not allowed to have NULL values. These are referred to as primary keys in the physical database. Consider creating a unique index on the identifiers to enforce uniqueness.
- Relationships are modeled as foreign keys.

Spaces are not allowed in entity names in a physical schema because these names must translate into SQL calls to create the tables. Table names should therefore conform to SQL naming rules.

Because primary key attributes are complete inventions, they can be of any indexable data type. (Each database engine has different rules about which data types can be indexable.) Making primary keys of type INT is almost purely arbitrary.

It is almost arbitrary because it is actually faster to search on numeric fields in many database engines. However, one could just have well have chosen CHAR as the type for the primary key fields. The bottom line is that this choice should be driven by the criteria for choosing identifiers.

Physical Table Definitions			
Table	**Column**	**Data Type**	**Notes**
CD	CDId CDTitle	INT TEXT(50)	Primary Key
Artist	ArtistId ArtistName	INT TEXT(50)	Primary Key
Song	SongId SongName	INT TEXT(50)	Primary Key
RecordLabel	RecordLabelId RecordLabelName	INT TEXT(50)	Primary Key

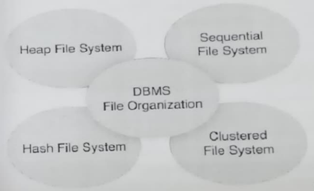
Q4) Explain database file organization concept?

File Organization

Relative data and information is stored collectively in file formats. A file is a sequence of records stored in binary format. A disk drive is formatted into several blocks that can store records. File records are mapped onto those disk blocks.

File Organization

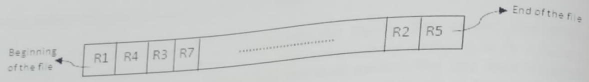
File Organization defines how file records are mapped onto disk blocks. We have four types of File Organization to organize file records –



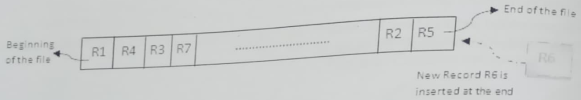
1) Sequential File Organization

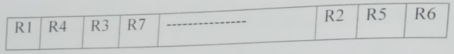
Every file record contains a data field (attribute) to uniquely identify that record. In sequential file organization, records are placed in the file in some sequential order based on the unique key field or search key. Practically, it is not possible to store all the records sequentially in physical form. It is one of the simple methods of file organization. Here each file/records are stored one after the other in a sequential manner. This can be achieved in two ways:

- Records are stored one after the other as they are inserted into the tables. This method is called **pile file method**. When a new record is inserted, it is placed at the end of the file. In the case of any modification or deletion of record, the record will be searched in the memory blocks. Once it is found, it will be marked for deleting and new block of record is entered.

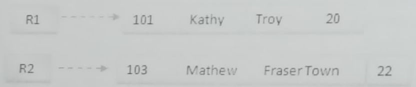


Inserting a new record:

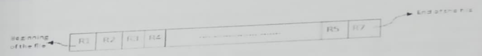




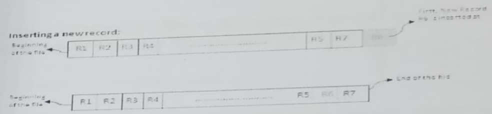
In the diagram above, R1, R2, R3 etc are the records. They contain all the attribute of a row. i.e.; when we say student record, it will have his id, name, address, course, DOB etc. Similarly R1, R2, R3 etc can be considered as one full set of attributes.



- In the second method, records are sorted (either ascending or descending) each time they are inserted into the system. This method is called **sorted file method**. Sorting of records may be based on the primary key or on any other columns. Whenever a new record is inserted, it will be inserted at the end of the file and then it will sort - ascending or descending based on key value and placed at the correct position. In the case of update, it will update the record and then sort the file to place the updated record in the right place. Same is the case with delete.



Inserting a new record:



Advantages of Sequential File Organization

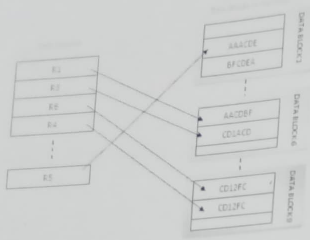
- The design is very simple compared other file organization. There is no much effort involved to store the data.
- When there are large volumes of data, this method is very fast and efficient. This method is helpful when most of the records have to be accessed like calculating the grade of a student, generating the salary slips etc where we use all the records for our calculations
- This method is good in case of report generation or statistical calculations.
- These files can be stored in magnetic tapes which are comparatively cheap.

Disadvantages of Sequential File Organization

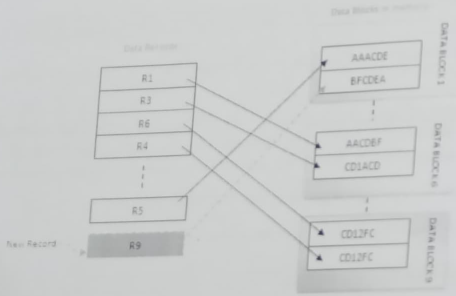
- Sorted file method always involves the effort for sorting the record. Each time any insert/update/ delete transaction is performed, file is sorted. Hence identifying the record, inserting/ updating/ deleting the record, and then sorting them always takes some time and may make system slow

2) Heap File Organization

This is the simplest form of file organization. Here records are inserted at the end of the file as and when they are inserted. There is no sorting or ordering of the records. Once the data block is full, the next record is stored in the new block. This new block need not be the very next block. This method can select any block in the memory to store the new records. It is similar to pile file in the sequential method, but here data blocks are not selected sequentially. They can be any data blocks in the memory. It is the responsibility of the DBMS to store the records and manage them.



If a new record is inserted, then in the above case it will be inserted into data block 1.



When a record has to be retrieved from the database, in this method, we need to traverse from the beginning of the file till we get the requested record. Hence fetching the records in very huge tables, it is time consuming. This is because there is no sorting or ordering of the records. We need to check all the data.

Similarly if we want to delete or update a record, first we need to search for the record. Again, searching a record is similar to retrieving it- start from the beginning of the file till the record is fetched. If it is a small file, it can be fetched quickly. But larger the file, greater amount of time needs to be spent in fetching.

In addition, while deleting a record, the record will be deleted from the data block. But it will not be freed and it cannot be re-used. Hence as the number of record increases, the memory size also increases and hence the efficiency. For the database to perform better, DBA has to free this unused memory periodically.

Advantages of Heap File Organization

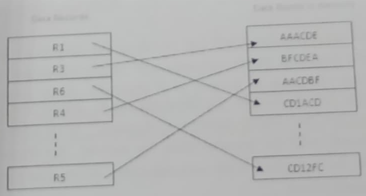
- Very good method of file organization for bulk insertion. i.e.; when there is a huge number of data needs to load into the database at a time, then this method of file organization is best suited. They are simply inserted one after the other in the memory blocks.

It is suited for very small files as the fetching of records is faster in them. As the file size grows, linear search for the record becomes time consuming

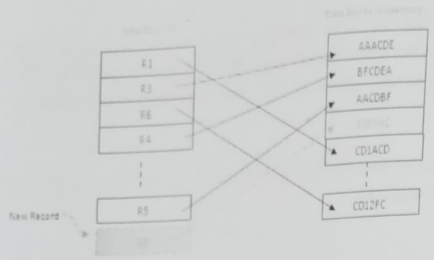
2) Hash File Organization

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

In this method of file organization, hash function is used to calculate the address of the block to store the records. The hash function can be any simple or complex mathematical function. The hash function is applied on some columns/attributes – either key or non-key columns to get the block address. Hence each record is stored randomly irrespective of the order they come. Hence this method is also known as Direct or Random file organization. If the hash function is generated on key column, then that column is called hash key, and if hash function is generated on non-key column, then the column is hash column.



When a record has to be retrieved, based on the hash key column, the address is generated directly from that address whole record is retrieved. Here no effort to traverse through whole Similarly when a new record has to be inserted, the address is generated by hash key and rec is directly inserted. Same is the case with update and delete. There is no effort for searching entire file nor sorting the files. Each record will be stored randomly in the memory.



These types of file organizations are useful in online transaction systems, where retrieval or insertion/updation should be faster.

Advantages of Hash File Organization

- Records need not be sorted after any of the transaction. Hence the effort of sorting is reduced in this method.
- Since block address is known by hash function, accessing any record is very faster. Similarly updating or deleting a record is also very quick.
- This method can handle multiple transactions as each record is independent of other. i.e.; since there is no dependency on storage location for each record, multiple records can be accessed at the same time.
- It is suitable for online transaction systems like online banking, ticket booking system etc.

Disadvantages of Hash File Organization

- This method may accidentally delete the data. For example, In Student table, when hash field is on the STD_NAME column and there are two same names – 'Antony', then same address is generated. In such case, older record will be overwritten by newer. So there will be data loss. Thus hash columns needs to be selected with utmost care. Also, correct backup and recovery mechanism has to be established.
- Since all the records are randomly stored, they are scattered in the memory. Hence memory is not efficiently used.
- If we are searching for range of data, then this method is not suitable. Because, each record will be stored at random address. Hence range search will not give the correct address range and searching will be inefficient. For example, searching the employees with salary from 20K to 30K will be efficient.
- Searching for records with exact name or value will be efficient. If the Student name starting with 'B' will not be efficient as it does not give the exact name of the student.
- If there is a search on some columns which is not a hash column, then the search will not be efficient. This method is efficient only when the search is done on hash column. Otherwise, it will not be able find the correct address of the data.

- If there is multiple hash columns – say name and phone number of a person, to generate the address, and if we are searching any record using phone or name alone will not give correct results.
- If these hash columns are frequently updated, then the data block address is also changed accordingly. Each update will generate new address. This is also not acceptable.

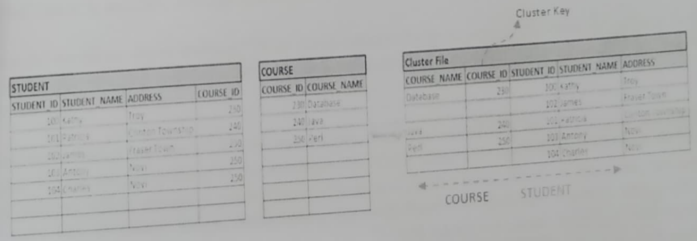
4) Clustered File Organization

Clustered file organization is not considered good for large databases. In this mechanism, related records from one or more relations are kept in the same disk block, that is, the ordering of records is not based on primary key or search key.

In all the file organization methods described above, each file contains single table and are all stored in different ways in the memory. In real life situation, retrieving records from single table is comparatively less. Most of the cases, we need to combine/join two or more related tables and retrieve the data. In such cases, above all methods will not be faster to give the result. Those methods have to traverse each table at a time and then combine the results of each to give the requested result. This is obvious that the time taken for this is more. So what could be done to overcome this situation?

Another method of file organization – Cluster File Organization is introduced to handle above situation. In this method two or more table which are frequently used to join and get the results are stored in the same file called clusters. These files will have two or more tables in the same data block and the key columns which map these tables are stored only once. This method hence reduces the cost of searching for various records in different files. All the records are found at one place and hence making search efficient.

For example, we want to see the students who have taken particular course. The tables are shown in below diagram. We can see there are two students who have opted for 'Database' and 'Perl' course each. Though it is stored in separate tables in logical view, when it is stored in physical view, we have combined them. This can be seen in cluster file below. This is the result of join. So do not have to put any effort or time for joining. Hence it will give faster results.



STUDENT

STUDENT_ID	STUDENT_NAME	ADDRESS	COURSE_ID
100	Kathy	Troy	230
101	Patricia	Clinton Township	240
102	James	Fraser Town	230
103	Antony	Novi	250
104	Charles	Novi	250

COURSE

COURSE_ID	COURSE_NAME
230	Database
240	Java
250	Perl

Cluster File

COURSE_NAME	COURSE_ID	STUDENT_ID	STUDENT_NAME	ADDRESS
Database	230	100	Kathy	Troy
		102	James	Fraser Town
Java	240	101	Patricia	Clinton Township
Perl	250	103	Antony	Novi
		104	Charley	Novi

If we have to insert or update or delete any record, we can directly do so. Here data are based on the primary key or the key with which we are searching the data. Also, clusters formed based on the join condition. The key with which we are joining the tables is know cluster key.

Advantages of Clustered File Organization

- This method is best suited when there is frequent request for joining the tables with joining condition.
- When there is a 1:M mapping between the tables, it results efficiently

Disadvantages of Clustered File Organization

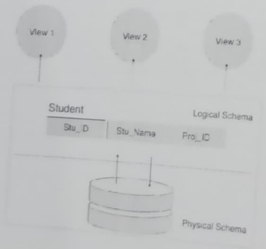
- This method is not suitable for very large databases since the performance of this meth on them is low.
- We cannot use this clusters, if there is any change is joining condition. If the join condition changes, the traversing the file takes lot of time.
- This method is not suitable for less frequently joined tables or tables with 1:1 condition

Q5) What is database schema?

Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.



A database schema can be divided broadly into two categories –

- **Physical Database Schema** – This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- **Logical Database Schema** – This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Q 6) What is DBMS indexing?

Q7) How do we choose indexes? What are tools available

Introduction of Indexing

The main goal of designing the database is faster access to any data in the database and quicker insert/delete/update to any data. This is because no one likes waiting. When a database is very huge, even a smallest transaction will take time to perform the action. In order to reduce the time spent in transactions, Indexes are used. Indexes are similar to book catalogues in library or even like an index in a book. What it does? It makes our search simpler and quicker. Same concept is applied here in DBMS to access the files from the memory.

When records are stored in the primary memory like RAM, accessing them is very easy and quick. But records are not limited in numbers to store in RAM. They are very huge and we have to store it in the secondary memories like hard disk. As we have seen already, in memory we cannot store records like we see – tables. They are stored in the form of files in different data blocks. Each block is capable of storing one or more records depending on its size.

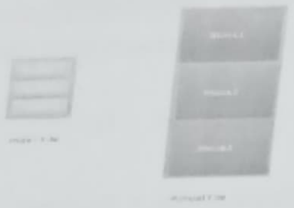
When we have to retrieve any required data or perform some transaction on those data, we have to pull them from memory, perform the transaction and save them back to the memory. In order to do all these activities, we need to have a link between the records and the data blocks so that we can know where these records are stored. This link between the records and the data block is called index. It acts like a bridge between the records and the data block.

How do we create these indexes? How these indexes help to access the data? Is linking between records and data block address enough to give better performance? Answers all these questions are learnt in this article.

How do we index in a book? We list main topics first and under that we group different sub-topic right? We do the same thing in the database too. Each table will have unique column or primary key column which uniquely determines each record in the table. Most of the time, we use this primary key to create index. Sometimes, we will have to fetch the records based on other columns in the table which are not primary key. In such cases we create index on those columns. But what is this index? Index in databases is the pointer to the block address in the memory. But these pointers are stored as (column, block_address) format.

1)Primary Index

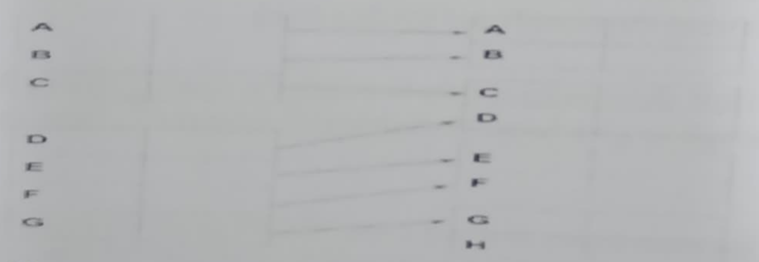
If the index is created on the primary key of the table then it is called as Pri Indexing. Since these primary keys are unique to each record and it has 1:1 relation betwee records, it is much easier to fetch the record using it. Also, these primary key are kept in so form which helps in performance of the transactions. The primary indexing is of two typ Dense Index and Sparse Index.



1. Dense Index

In this case, indexing is created for primary key as well as on the columns on which we perfo transactions. That means, user can fire query not only based on primary key column. He c query based on any columns in the table according to his requirement. But creating index o on primary key will not help in this case. Hence index on all the search key columns are store This method is called dense index.

For example, Student can be searched based on his ID which is a primary key. In addition, w search for student by his first name, last name, particular age group, residing in some plac opted for some course etc. That means most of the columns in the table can be used for searchin the student based on different criteria. But if we have index on his ID, other searches will not b efficient. Hence index on other search columns are also stored to make the fetch faster.



Dense Index

1. **Planning Phase:**In the planning phase of benchmark testing, the standards and requirements are identified and prioritised. Further the benchmark criteria is decided and the benchmark test process is defined.

2. **Analysis Phase:** In this stage of benchmark testing, the root cause of the error is identified to improve the quality of the software application. Moreover, goals are set for the test process.

3. **Integration Phase:** The third stage of benchmark testing is the integration stage. Here, the outcomes are shared with the concerned person to get approval. Also, functional goals are also defined established by the team here.

4. **Action Phase:** This is the final phase of benchmark testing, where the test plan and documentation is developed and the actions specified in previous phases are implemented. Moreover the progress is also monitored here, while the process runs continuously.

Components of Benchmark Testing:

TPerformed using the same software and hardware that runs in the production network, benchmark testing helps testers and developers in identifying any snags or errors that may occur after the product is released. The hallmark of this type of testing is the repeatability of the test suites. Therefore, to help you get a better understanding of this type of testing, here are the components of benchmark testing:

a. **Workload Specifications:** Determine the type and frequency of request that is to be submitted to the system under test (SUT).

b. **Specifications of Metrics:** It determines which elements will be measured. For example: download speed, or more.

c. **Specifications of Measurements:** Discern how to measure the specified elements to appropriate values.

Prerequisites of Benchmark Testing:

Before initiating the process of benchmark testing, it is vital for software testers to ensure that all the necessary conditions and requirements are fulfilled and the software is ready for benchmark testing. This allows them to take all the crucial precautions as well as enables them to perform the process of testing smoothly. Therefore, to assist testers in cross-checking all the prerequisites of benchmark testing, provided here is the list for the same.

- Make sure all the software components are in the accurate working condition.
- Consistency and control are important measures to perform benchmark testing.

- There should be a clear understanding of the system architecture to design test criteria and test data.
- Testers ought to ensure that the initial static data is examined and updated according to the number of users.
- Check for operating systems updates and real world configurations.
- Each time the tests should be executed under the same environmental condition.
- The elements of the system should be split according to its functionalities.
- Every system has different architecture and design, which needs to be taken into consideration while performing benchmark testing.
- The software and hardware components should be in line with the specification of the production environment.

Advantages of Benchmark Testing:

As stated above, benchmark testing offers numerous benefits to developers and testers, as well as the users. With the assistance of this testing technique, the team of testers can effortlessly ensure the quality, performance, efficiency of the end product. Other advantages of benchmark testing are: Helps measure how a change or modification affects the performance characteristics of a software product.

- Ensures that the initial static data is examined and updated according to the number of users.
- Validates that the software components are in an accurate working condition.
- It helps build an application whose characteristics are well understood and that can stand up to the rigors of the real world users.
- With its assistance software developers are able to launch a software or application with confidence, as they are sure that their users will be satisfied with the performance as well as the effectiveness of the released product.