ASSIGNMENT#7

Describe the four steps in the transfer of data from disk to primary memory.

Four steps in transferring data from disk to primary memory are:

(1) Locating Track: At first, locate the track where data is located by moving the read/write head pointer.
(2) Switching Head: Read/Write head then is moved to a desired location under current head.
(3) Locating Sector: After locating the track, locate the correct sector by rotating the disk until correct sector is obtained (head comes under desired sector).
(4) Transfer Data: This is the last step in which the data is transferred from the disk to the primary memory.

What is an index? Compare the concept of the index in a book to an index in an information system.

Generally, index is an sorted(increasing) list of integers/item names/identifiers which are used to access a location they points now, The concept of an index in a book is also the same concept of an index in an information system, in both cases the names/identifiers of items are being copied into another locations, they can be sorted and associated with the location at which they are found index in a book will help us to move directly to the contents we need, similarity by using index we can also directly move to the information we need in information system.

Describe the idea of the simple linear index. What are its shortcomings?

A simple linear index is a file which is logically organized with a group of key-value pairs where all keys are arranged in an order. Simple linear index may be stored in a main memory or a disk.

Shortcomings:

If database is completely full, then the index cannot be stored in main memory.

If number of disks accessed is mor then, linear indexing becomes expensive.

Describe the idea of the B+-tree index. What are its advantages over the simple linear index?

B+ tree is an n-th tree with a variable but often large number of children per node. A B+ tree consists of a root internal nodes and leaves. The root may be a leaf or a node with two or more children. A B+ tree index is a typical node of B+ tree. It searches the values up to n-1 in key values are k1, k2, k3,….kn.

An index is data structure, that stores values for a specific column in a table. An index is created on a column sortable. So, the index consist of column values from one table and values are stored in data structure. if an index is using the most used data structure type – a B- tree – then the data structure is also *sorted*. Having the column values be sorted can be a major performance enhancement.

Describe what happens to the index tree when you insert new records into a file with a B+-tree index.

Insertion in a B+-tree is like inserting into other search trees, a new record is always inserted at one of the leaf nodes. The complexity added is that insertion could overflow a leaf node that is already full. When such overflow situations occur a brand-new leaf, node is added to the B+-tree at the same level as the other leaf nodes.

Describe the idea of the hashed file. What are its advantages and disadvantages in comparison to indexes?

Hashing in computer science is a concept in which we directly search the location of a data value on the disk instead of making use of index structure. A hash file is generally a file that is to be converted into a numerical string by an algorithm that use mathematical functions and operations. The data of a hashed file can only be understood when it has been decrypted with a key commonly known as hash key. The data here is a numerical depiction of data which is generally not easy to understand. The Hash File structure use hash function to compute on some fields. The output after applying the hash function decide that in which location of disk block the data is to be stored.

Advantages of hashed file over indexes are as follows:
1. The address of any data is determined by hash function therefore to access any record or data becomes much faster.
2. Updating or deleting a record is amazingly fast process.
3. The hashing method can handle multiple transactions, since every record is independent of each other and as there is no dependency on storage address for each record, many records can be accessed at same time.
4. In hash file insertion, updating and deletion is extremely easy.
5. Sorting is reduced in the concept of hashed file.

Disadvantages of hashed file over indexes are as follows:
1. If the columns in a hashed file are shortly updated, then the address of data block is also changed, and therefore each update hence will create new address, which is not accepted at all.
2. The records in a hashed file are stored at random locations, as a result a lot of space is wasted in case of small size files.
3. The system performance generally affects by updating the hashed column more frequently because in this data need to be moved in buckets.
4. If the querying attribute is not the hashed attribute, then the user must need to search entire data record.

What is a collision in a hashed file? Why do collisions occur? Why do collisions occur? Why are they of concern in the application environment?

A collision in hashing technique happens when the hashing algorithm generates a address location to store a record is already occupied by any other record. The hashing algorithm, when used to each of the two different key values generates same address and thus it tries to store or hold both the data records at same location in the file. This same address location concept is known as collision. To deal with this problem, the next available slot which is free or empty will be is allocated to the current hash value. The most common methods to handle collisions in hashed file are open addressing, separate chaining, probabilistic hashing, perfect hashing, and coalesced hashing methods. This collision is of concern because the synonym pointer chain required to handle or resolve the problem takes notably extra time to traverse on future data record access.

Explain why the need to perform joins is an important factor affecting application and database performance.

Join generally means to combine two or more things. In computer science particularly in SQL i.e., structured query language, JOIN refers the combination of two or more tables having similarities. JOIN is a clause which is used to merge the data records from different tables within a database. The order in query in which the tables are joined will determine the result of the join operation. If the query is used to join all the large tables first and then smaller table to be joined afterwards, then this may result in useless processing by the SQL. There are multiple JOIN clauses in structured query language that determines how to combine rows from two or more tables in a database, which is to be dependent on a related column between those tables to produce desired output structure in the form of table itself. These are primarily used when a person wants to take out data from different tables that have one-to-many or many-to-many relationship among them.

There exist many types of joins as follows:
1. INNER join**:** This join selects all those records from the two tables that have matching values.
2. FULL join: This join is the integration of both Right and Left join.
3. RIGHT join: It is opposite of left join i.e., it returns all values from right table with matching values from left table.
4. LEFT join: This join selects all those values from left table along with all the matching values from the right table.
5. NATURAL join: This join type combines records that have same attributes and datatypes but will consists only single copy of all the common columns.

What attributes should be considered as candidates for having indexes built on them? What is the potential problem with building too many indexes?

The attributes which should be considered as candidates for having indexes built on them are Primary keys and search attributes as they are important in direct searches. And another set of attributes are those which are to be players in operations such as joins that involves searches containing SQL ORDER BY and GROUP BY commands.
Building too many indexes can affect the overall system performance. Thus, having too many indexes can be a potential problem.

What is “view”? Which factors affecting application and database performance can be dealt with by using views? Explain.

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

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.

Security can help affect applications and performance because it will restrict certain users from having access to certain files if they are not allowed to. Using a view, this can eliminate wondering eyes that might affect the performance of a database if something gets deleted or corrupted.

View's query plans are cached removing the overhead of planning the query and speeding up performance over the same SQL performed in-line.

Describe the "clustering files" technique. What advantage is gained by using it? What is its disadvantage?

Clustering files technique is nothing but a procedure which is helpful in storing records from two relations near each other in the disk, by organizing the records of the two logical relations.

The advantage of clustering file technique is the access speed. This is obtained while performing an access using the related records. The disadvantage of this process is the low performance. This can be observed while using the related records of only a single relation.

What is the difference between horizontal and vertical partitioning? What is their common advantage? Are their disadvantages the same or different? Explain.

Horizontal partitioning of data refers to storing different rows into different tables. E.g.: Students with their first name starting from A-M are stored in table A, while student with their first name starting from N-Z are stored in table B. Therefore, the partition tables are A and B.

On the contrary, vertical partitioning of data refers to creating tables with fewer columns and using other tables to store the remaining columns. Normalization follows a similar principle. E.g., separately storing Age and Name (frequently needed) in one table and Phone number and address (less frequently used) in another.

Their common advantage is that they can improve performance when a particular query requires data in only one partition. Each part of information could be archived to optimize the performance for its own utilization. Data not applicable to one group of users could be separate from information but the users are permitted to utilize it. This increases the security of data. If one file is harmed there is no problem in accessing the other files, so the impacts of harm are detached only with that file which increase the uptime.

The disadvantages are that the access speed is inconsistent. The access speed is different for different partitions. Smaller records will require more time to recover. Insertion, deletion and modifications anomalies are possible, and it requires programming to keep away these issues.

What is denormalization? Denormalization, while improving performance under certain circumstances, also leads to a serious problem. How does denormalization improve performance and what is this major drawback?

Denormalization is a database optimization technique in which we add redundant data to one or more tables. This can help us avoid costly joins in a relational database. Note that denormalization does not mean doing normalization. It is an optimization technique that is applied after doing normalization. Below are some salient key features points ->

1) For increasing the Query results-> when we join multiple tables it takes lots of time this will slow the database and makes performance low but denormalization will increase the performance.
2) To make Database easier to manage-> A normalized database does not have calculated values that are essential for applications. Calculating these values on-the-fly would require time, slowing down query execution.
You can deformalize a database to provide calculated values. Once they are generated and added to tables, downstream programmers can easily create their own reports and queries without having in-depth knowledge of the app's code or API.
3) To get fast results and data performance increased-> Databases are used to store information both analytical and statistical so when we want any reports of the data it will take lots of time to fetch but with denormalization can help to meet this challenge.

Drawbacks:
1) A denormalized physical implementation can increase hardware costs.
The rows of a denormalized table are always wider than the rows of a fully normalized table. A row cannot span data blocks; therefore, there is a high probability that you will be forced to use a larger data block size for a denormalized table. The greater the degree of a table, the larger the impact on storage space. This impact can be severe in many cases. Row width also affects the transfer rate for all I/O operations; not just for disk access, but also for transmission across the BYNET and to the requesting client.
2) You need bigger servers and need more man power to support these servers it will require more cost but your performance got increases.
3) Sometimes it will decrease the performance of other application so we need to maintain the neutrality of Database.
4) It will increase the Cost and Complexity of the programs.

Duplicating entire tables or parts of tables ("subset tables") obviously introduces data redundancy. What is the advantage of doing this? Do you think it is worth the introduction of redundancy? Explain.

The advantage of duplicating entire table or subset tables is reduction in performance overcrowding. This can be observed while several applications are using the same table, the performance congestion can be reduced by duplicating entire tables or part of tables.

But this introduces the data redundancy. The act of duplicating the tables is worth the introduction if redundancy because the applications will not be executed quickly.