IARE '24 DISCORD TEAM

II-II

DBMS MODULE 5 SOLUTIONS

SYED IKRAM • VISHNU • UJJWAL • VARUN TEJA

DATA STORAGE AND QUERY PROCESSING

Completed

DBMS MODULE 5

PART A

1. Consider a B+ tree in which the maximum number of keys in a node is 5, Calculate the minimum number of keys in any non - root node.

Since the maximum number of keys is 5, the maximum number of children a node can have is 6. Definition of B Tree, minimum children that a node can have would be 6/2 = 3. Therefore, the minimum number of keys that a node can have becomes (3-1)=2 keys.

2. In the index allocation schema of blocks to a file, Calculate on what maximum possible size of the file depends.

In the indexed scheme of blocks to a file, the maximum possible size of the file depends on the number of blocks used for index and the size of index.

3. A clustering index is defined on the fields of which type. Analyse them.

We are given that the clustering index is defined on the fields. Now, if the records of the file are physically ordered on a non-key field since it is a non-key, it will not have a distinct value for each record. Therefore, the clustering index is defined on the fields of type non-key and ordering.

4. Calculate the minimum space utilisation for a B+ tree index.

By the definition of a B+ tree, each index page, except for the root, has at least d and at most 2d key entries. Therefore—with the exception of the root—the minimum space utilisation guaranteed by a B+ tree index is 50 percent.

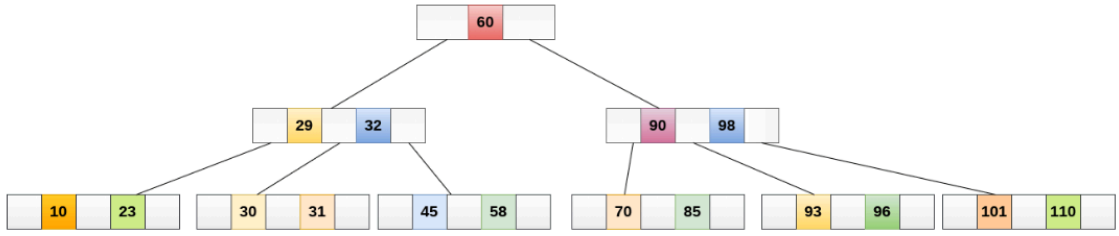
5. Explain about the B tree and the structure of B+ tree in detail with an example.

B Tree is a specialised m-way tree that can be widely used for disk access. A B Tree of order m can have at most m-1 keys and m children. One of the main reasons for using B tree is its capability to store a large number of keys in a single node and large key values by keeping the height of the tree relatively small.

A B tree of order m contains all the properties of an M way tree. In addition, it contains the following properties.

- Every node in a B Tree contains at most m children.
- Every node in a B-Tree except the root node and the leaf node contain at least m/2 children.
- The root nodes must have at least 2 nodes.
- All leaf nodes must be at the same level.

A B tree of order 4 is shown in the following image.



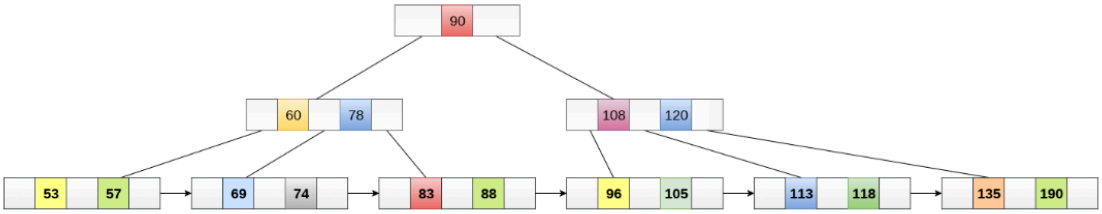
B+ Tree is an extension of B Tree which allows efficient insertion, deletion and search operations.

In B Tree, Keys and records both can be stored in the internal as well as leaf nodes. Whereas, in B+ tree, records (data) can only be stored on the leaf nodes while internal nodes can only store the key values.

The leaf nodes of a B+ tree are linked together in the form of singly linked lists to make the search queries more efficient.

B+ Tree are used to store a large amount of data which can not be stored in the main memory. Due to the fact that, size of main memory is always limited, the internal nodes (keys to access records) of the B+ tree are stored in the main memory whereas leaf nodes are stored in the secondary memory.

The internal nodes of B+ tree are often called index nodes. A B+ tree of order 3 is shown in the following figure.



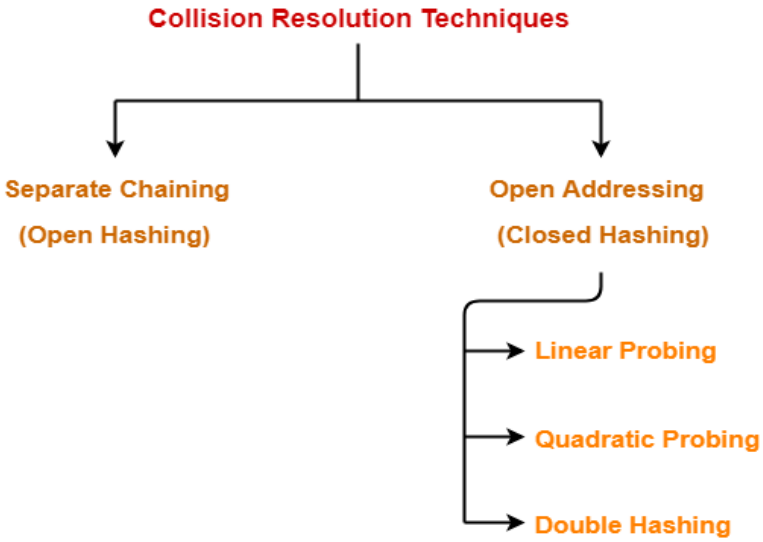
6. Explain the distinction between closed and open hashing. Discuss the relative merits of each technique in database applications

Open hashing may place keys with the same hash function value in different buckets. Closed hashing always places such keys together in the same bucket. Thus in this case, different buckets can be of different sizes, though the implementation may be by linking together fixed size buckets using overflow chains.

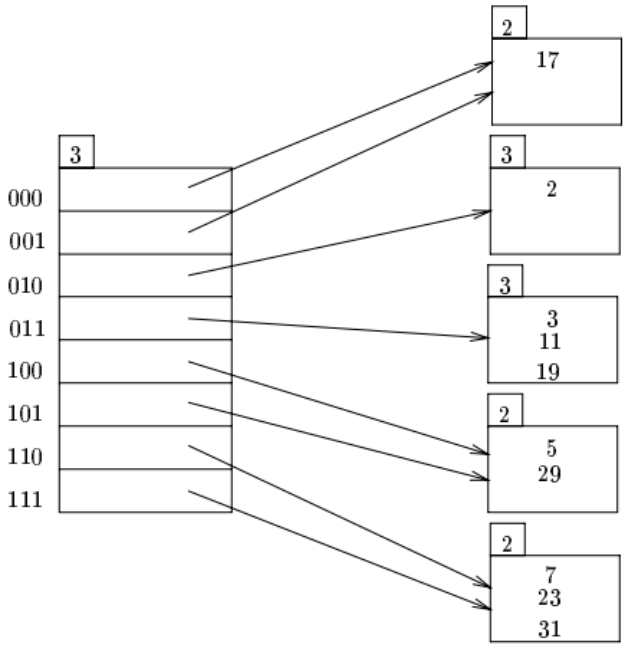
Deletion is difficult with open hashing as all the buckets may have to be inspected before we can ascertain that a key value has been deleted, Whereas in closed hashing only that bucket whose address is obtained by hashing the key value needs to be inspected. Deletions are more common in databases and

hence closed hashing is more appropriate for them. For a small, static set of data lookups may be more efficient using open hashing.

The symbol table of a compiler would be a good example.



7. Suppose that we are using extendible hashing on a file following search - key values: 2, 3, , 7 , 11, 17, 19, 23, 29, 31. Show the extendable hash structure for this file if the hash function is h(x) = x mod 8 and buckets can hold three records



8. Explain various steps in Query Processing and write any two techniques to optimise query.

Query Processing is the activity performed in extracting data from the database. In query processing, it takes various steps for fetching the data from the database. The steps involved are:

1. Parsing and translation
2. Optimization
3. Evaluation

Parsing

the parser of the query processor module checks the syntax of the query, the user's privileges to execute the query, the table names and attribute names, etc.

Parsing and Translation

As query processing includes certain activities for data retrieval. Initially, the given user queries get translated in high-level database languages such as SQL. It gets translated into expressions that can be further used at the physical level of the file system. After this, the actual evaluation of the queries and a variety of query -optimising transformations takes place. Thus before processing a query, a computer system needs to translate the query into a human-readable and understandable language.

Evaluation

- For this, with addition to the relational algebra translation, it is required to annotate the translated relational algebra expression with the instructions used for specifying and evaluating each operation. Thus, after translating the user query, the system executes a query evaluation plan.

Query Evaluation Plan

- In order to fully evaluate a query, the system needs to construct a query evaluation plan.

- The annotations in the evaluation plan may refer to the algorithms to be used for the particular index or the specific operations.
- A query execution engine is responsible for generating the output of the given query. It takes the query execution plan, executes it, and finally makes the output for the user query.

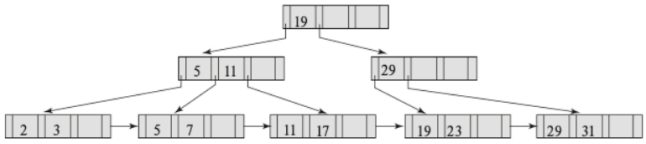
Optimization

- The cost of the query evaluation can vary for different types of queries. Although the system is responsible for constructing the evaluation plan, the user does not need to write their query efficiently.
- Usually, a database system generates an efficient query evaluation plan, which minimises its cost. This type of task is performed by the database system and is known as Query Optimization.
- For optimising a query, the query optimizer should have an estimated cost analysis of each operation. It is because the overall operation cost depends on the memory allocations to several operations, execution costs, and so on.
- Finally, after selecting an evaluation plan, the system evaluates the query and produces the output of the query.

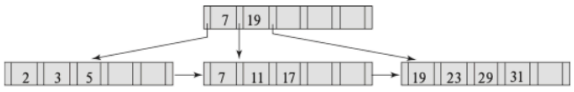
9. Construct a B+ tree for the following (2, 3, 5, 7, 11, 17, 19, 23, 29, 31). Assume that the tree is initially empty and values are added in ascending order. Construct B+ tree for the cases where the number of pointers that will fit in one node is as follows.

a. **Four**
b. **Six**
c. **Eight**

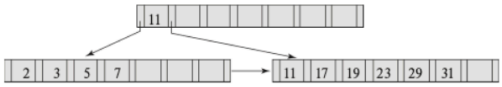
[4] $M = 4$



$M = 6$



$M = 8$



10. Consider the B+ tree index of order d = 2 shown in (fig10.1)

1. **Show the tree that would result from inserting a data entry with key 9 into this tree.**
2. **Show B+ tree that would result from inserting a data entry with key 3 into the original tree. How many page reads and page writes does the insertion require?**
3. **Show the B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the left sibling is checked for possible distribution.**
4. **Show B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the right sibling is checked for possible redistribution.**
5. **Show the B+ tree that would result from starting with the original tree, inserting a data entry with key 46 and then deleting the data entry with key 52.**
6. **Show the B+ tree that would result from deleting the data entry with key 91 from the original tree.**

PART B

1. Write in detail about hash based indexing and tree based indexing.

In DBMS, hashing is a technique to directly search the location of desired data on the disk without using index structure. Hashing method is used to index and retrieve items in a database as it is faster to search that specific item using the shorter hashed key instead of using its original value.

B+ tree file organisation is the advanced method of an indexed sequential access method. It uses a tree-like structure to store records in File. It uses the same concept of key-index where the primary key is used to sort the records. For each primary key, the value of the index is generated and mapped with the record.

2. Compare I/O costs for all File Organisations.

1. Sequential File Organisation
2. Heap File Organisation
3. Hash/Direct File Organisation
4. Indexed Sequential Access Method
5. B+ Tree File Organisation
6. Cluster File Organisation

- A heap file has good storage efficiency and supports fast scanning and insertion of records. However, it is slow for searches and deletions.
- A sorted file also offers good storage efficiency, but insertion and deletion of records is slow. Searches are faster than in heap files.
- A clustered file offers all the advantages of a sorted file and supports inserts and deletes efficiently. Searches are even faster than in sorted files, although a sorted file can be faster when a large number of records are retrieved sequentially, because of blocked I/O efficiencies.

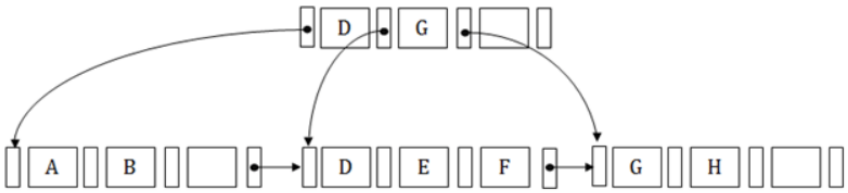
- Unclustered tree and hash indexes offer fast searches, insertion, and deletion, but scans and range searches with many matches are slow. Hash indexes are a little faster on equality searches, but they do not support range searches

3. Explain in detail about ISAM.

ISAM (Indexed Sequential Access Method) is a file management system developed at IBM which is a method for creating, maintaining, and manipulating computer files of data so that records can be retrieved sequentially or randomly by one or more keys. Indexes of key fields are maintained to achieve fast retrieval of required file records in Indexed files. IBM originally developed ISAM for mainframe computers, but implementations are available for most computer systems.

4. Explain B+ trees. Discuss about this Dynamic Index structure.

- The B+ tree is a balanced binary search tree. It follows a multi-level index format.
- In the B+ tree, leaf nodes denote actual data pointers. B+ tree ensures that all leaf nodes remain at the same height.
- In the B+ tree, the leaf nodes are linked using a link list. Therefore, a B+ tree can support random access as well as sequential access.
- In the B+ tree, every leaf node is at equal distance from the root node. The B+ tree is of the order n where n is fixed for every B+ tree.
- It contains an internal node and leaf node.



Internal node

- An internal node of the B+ tree can contain at least n/2 record pointers except the root node.
- At most, an internal node of the tree contains n pointers.

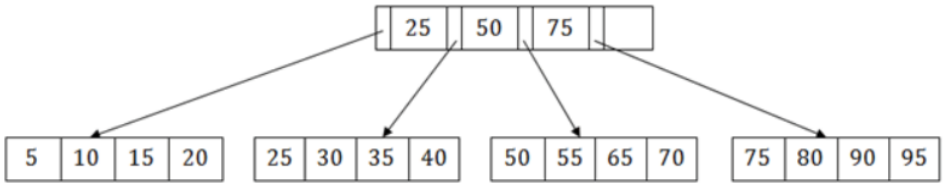
Leaf node

- The leaf node of the B+ tree can contain at least n/2 record pointers and n/2 key values.
- At most, a leaf node contains n record pointer and n key values.
- Every leaf node of the B+ tree contains one block pointer P to point to the next leaf node.

5. Demonstrate searching for a given element in B+ trees. Explain with examples.

Suppose we have to search 55 in the below B+ tree structure. First, we will fetch the intermediary node which will direct to the leaf node that can contain a record for 55.

So, in the intermediary node, we will find a branch between 50 and 75 nodes. Then at the end, we will be redirected to the third leaf node. Here DBMS will perform a sequential search to find 55.

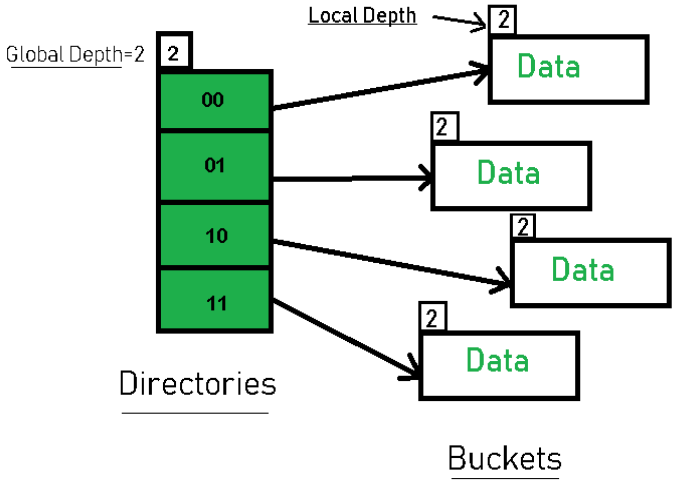


6. Compare and contrast Extendible Hashing with Linear Hashing

Extendible Hashing is a dynamic hashing method wherein directories, and buckets are used to hash data. It is an aggressively flexible method in which the hash function also experiences dynamic changes.

Main features of Extendible Hashing: The main features in this hashing technique are:

Directories: The directories store addresses of the buckets in pointers. An id is assigned to each directory which may change each time when Directory Expansion takes place.
Buckets: The buckets are used to hash the actual data.
Basic Structure of Extendible Hashing:



Extendible Hashing

- Linear hashing (LH) is a dynamic data structure which implements a hash table and grows or shrinks one bucket at a time.
- The file structure of a dynamic hashing data structure adapts itself to changes in the size of the file, so expensive periodic file reorganisation is avoided.
- A Linear Hashing file expands by splitting a predetermined bucket into two and contracts by merging two predetermined buckets into one.
- In Linear Hashing there are two types of buckets, those that are to be split and those already split. While extendible hashing splits only overflowing buckets, spiral hashing (a.k.a. spiral storage) distributes records unevenly over the buckets such that buckets with high costs of insertion, deletion, or retrieval are earliest in line for a split.

Linear Hashing vs. Extendible Hashing

- Less Code: LH
- Less Space: LH
- Higher Performance: EH potentially
- No Overflow Buckets: EH
- No Directory: LH
- Complexity of EH: B O(1), E O(1), W O(1)
- Complexity of LH: B O(1), E O(1), W O(n)

7. How does Extendible hashing use a directory of buckets? How does it handle insert and delete operations?

Extendible Hashing is a dynamic hashing method wherein directories, and buckets are used to hash data. It is an aggressively flexible method in which the hash function also experiences dynamic changes.

Main features of Extendible Hashing: The main features in this hashing technique are:

- Directories: The directories store addresses of the buckets in pointers. An id is assigned to each directory which may change each time when Directory Expansion takes place.
- Buckets: The buckets are used to hash the actual data.

Refer this link for insert and delete operations
Extendible Hashing (Dynamic approach to DBMS) - GeeksforGeeks

8. Explain how insert and delete operations are handled in a static hash index.

- Insert a Record

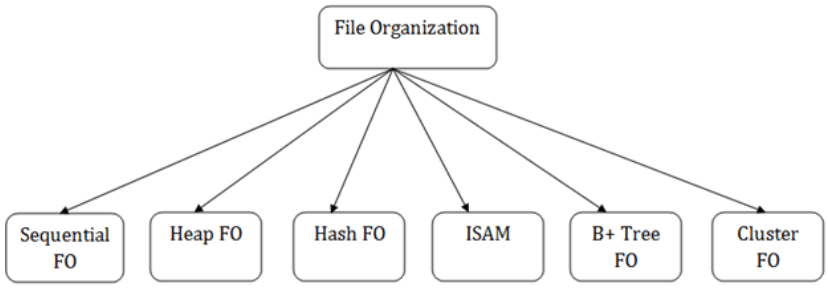
When a new record is inserted into the table, then we will generate an address for a new record based on the hash key and the record is stored in that location.

- Delete a Record

To delete a record, we will first fetch the record which is supposed to be deleted. Then we will delete the records for that address in memory.

9. Explain the organisation of records in files in detail

- 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 organisation which was used for a given set of records.
- File organisation is a logical relationship among various records. This method defines how file records are mapped onto disk blocks.
- File organisation 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 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.



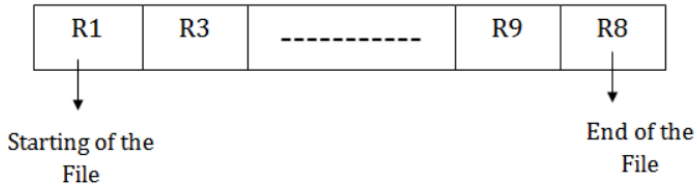
10. Explain about sequential file and heap file organisations.

Sequential

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

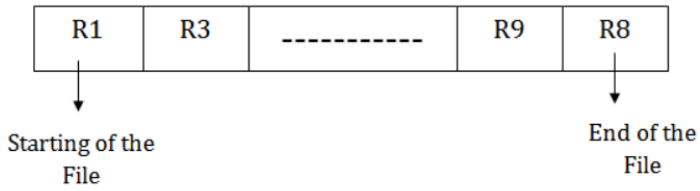
1. Pile File Method:

- 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.



2. 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.

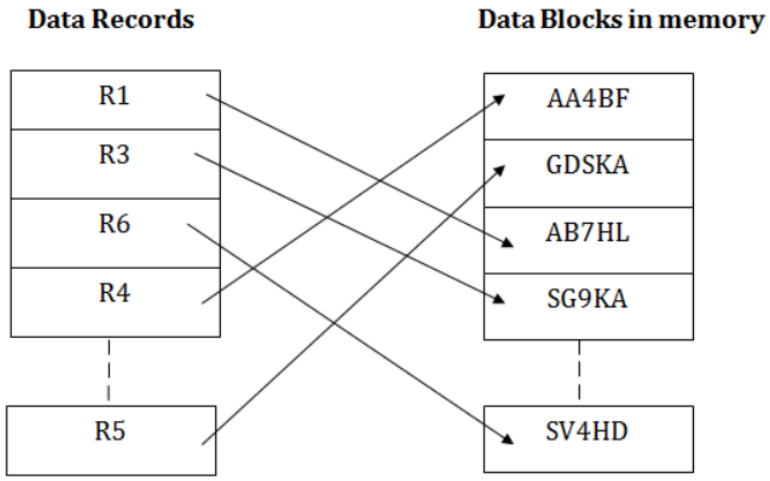


Heap File

- It is the simplest and most basic type of organisation. It works with data blocks. In heap file organisation, 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 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.

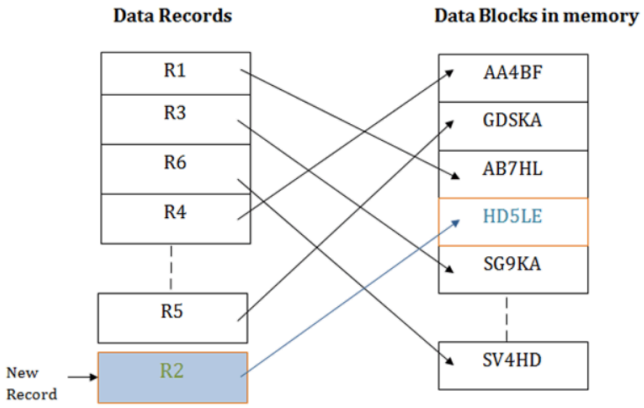
11. Explain the hash file organisation.

Hash File Organisation uses the computation of hash function on some fields of the records. The hash function's output determines the location of the 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 the record is directly inserted. The same process is applied in the case of delete and update.

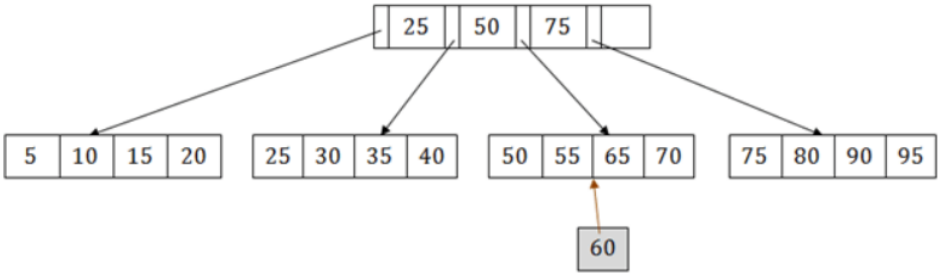
In this method, there is no effort for searching and sorting the entire file. In this method, each record will be stored randomly in the memory.



12. Illustrate insertion of an element in B+ trees with an example.

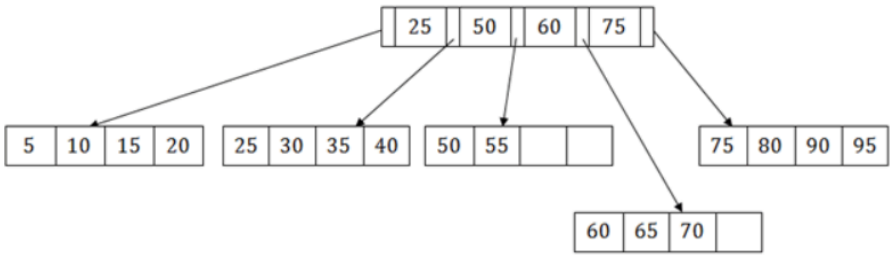
Suppose we want to insert a record 60 in the below structure. It will go to the 3rd leaf node after 55. It is a balanced tree, and a leaf node of this tree is already full, so we cannot insert 60 there.

In this case, we have to split the leaf node, so that it can be inserted into the tree without affecting the fill factor, balance and order.



The 3rd leaf node has the values (50, 55, 60, 65, 70) and its current root node is 50. We will split the leaf node of the tree in the middle so that its balance is not altered. So we can group (50, 55) and (60, 65, 70) into 2 leaf nodes.

If these two have to be leaf nodes, the intermediate node cannot branch from 50. It should have 60 added to it, and then we can have pointers to a new leaf node.

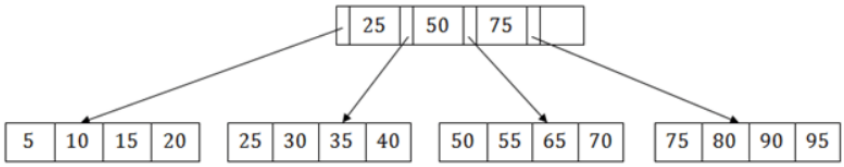


This is how we can insert an entry when there is overflow. In a normal scenario, it is very easy to find the node where it fits and then place it in that leaf node.

13. Illustrate deletion of an element in B+ trees with an example.

Suppose we want to delete 60 from the above example. In this case, we have to remove 60 from the intermediate node as well as from the 4th leaf node too. If we remove it from the intermediate node, then the tree will not satisfy the rule of the B+ tree. So we need to modify it to have a balanced tree.

After deleting node 60 from above B+ tree and re-arranging the nodes, it will show as follows:



14. Write in detail about static hashing.

Static hashing is a method of hashing, or shortening a string of characters in computer programming, in which the set of shortened characters remains the same length to improve the ease with which data can be accessed. All objects listed in an object dictionary are static and may not change when static hashing is applied. This method is often compared to the alternative, dynamic hashing.

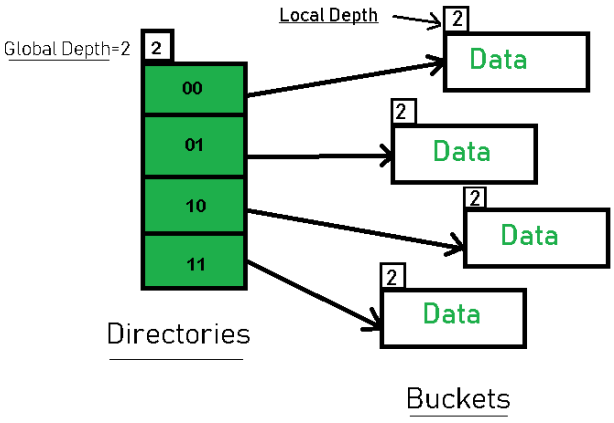
The process of static hashing creates a smaller, adaptable string of characters, making it faster and easier for users to find objects in a dictionary or groups of objects stored in a containing data structure.

15. Explain in detail about extendible hashing.

Extendible Hashing is a dynamic hashing method where directories, and buckets are used to hash data. It is an aggressively flexible method in which the hash function also experiences dynamic changes.

Main Features of Extendible hashing: The main features in this hashing techniques are:

- Directories: The directories store addresses of the buckets in pointers. An ID is assigned to each directory which may change each time when Directory Expansion takes place.
- Buckets: The buckets are used to hash the actual data.



Extendible Hashing

16. Explain in detail about linear hashing.

Linear hashing is a dynamic hashing technique. It allows a file to extend or shrink its number of buckets without a directory as used in Extendible Hashing. Suppose you start with a number of buckets N to put records in the buckets 0 to N-1. Let this be round i which will be 0 initially. You start with an initial mod

hash function hi(K) = K mod 2iN. When there is a collision, the first bucket, i.e., bucket 0, is split into two buckets: bucket 0 and a new bucket N at the end of the file. The records in bucket 0 are redistributed between the two buckets using another hash function hi+1 = K mod 2(i+1)N. Important property of the hash function is that records hashed into bucket 0 based on hi will be hashed to bucket 0 or bucket N based on the new hash function hi+1.

Here is a simple example of using linear hashing to store 14 records with number of initial buckets N = 4.

Bucket#		Buckets			
0	s = 0	32	44	36	
1		9	25	5	
2		14	18	10	30
3		31	35	7	11

17. Explain about storage devices and memory hierarchy.

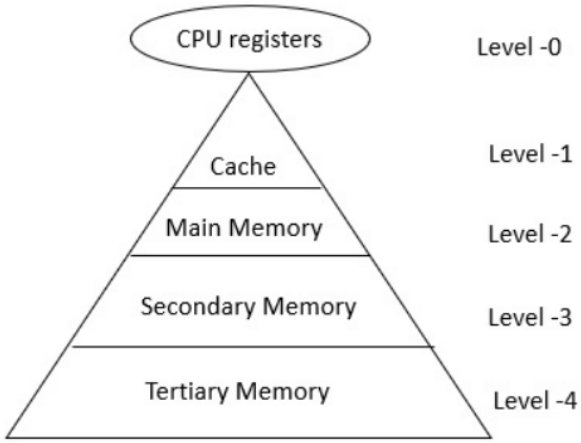
A storage device is a kind of hardware, which is also known as storage, storage medium, digital storage, or storage media that has the ability to store information either temporarily or permanently. Generally, it is used to hold, port, and extract data files.

Storage devices are available in various form factors; for example, a computer device includes different storage media such as hard disk, RAM, cache. They also have optical disk drives and externally connected USB drives.

Two types of storage devices, primary and secondary are available there to store data:

- Primary storage devices: They are fit internally to the computer and very fast in terms of accessing data files. The RAM and cache memory are the examples of the primary storage devices.
- Secondary storage devices: The hard disk, USB storage devices and optical disk drive are examples of secondary storage devices, which are designed to store data permanently. They include a large storage capacity while comparing with primary storage devices.

Memory hierarchy is arranging different kinds of storage present on a computing device based on speed of access. At the very top, the highest performing storage is CPU registers which are the fastest to read and write to.



18. Explain different RAID levels in disks.

RAID stands for a Redundant Array of Independent Disks. The technology combines two or more physical drives into a logical unit presented as a single hard drive to the operating system.
RAID levels are defined by the combination of the techniques used; they also provide varying degrees of reliability (ability to withstand drive failure) and availability (speed of I/O). There are six basic RAID levels:

- RAID Level 0 stripes data across two or more drives. No parity.
- RAID Level 1 mirrors data to two or more drives. No parity.
- RAID Level 0+1 is striped sets in a mirrored set. RAID 0+1 creates a striped set that mirrors a primary striped set.
- RAID Level 1+0 (RAID 10) is mirrored in a striped set. RAID 0+1 creates a striped set from a series of mirrored drives.
- RAID Level 3 is byte-level striping with a dedicated parity disk.
- RAID Level 4 is block-level striping with a dedicated parity disk.
- RAID Level 5 is striping with distributed (interleaved) parity. No dedicated parity disk.

19. Compare Tree indices and hash indices.

B-Tree Index Characteristics

A B-tree index can be used for column comparisons in expressions that use the =, >, >=, <, <=, or BETWEEN operators. The index also can be used for LIKE comparisons if the argument to LIKE is a constant string that does not start with a wildcard character. For example, the following SELECT statements use indexes:

```
SELECT * FROM tbl_name WHERE key_col LIKE 'Patrick%';
SELECT * FROM tbl_name WHERE key_col LIKE 'Pat%_ck%';
```

Hash Index Characteristics

Hash indexes have somewhat different characteristics from those just discussed:

They are used only for equality comparisons that use the = or ⇔ operators (but are very fast). They are not used for comparison operators such as < that find a range of values. Systems that rely on this type of single-value lookup are

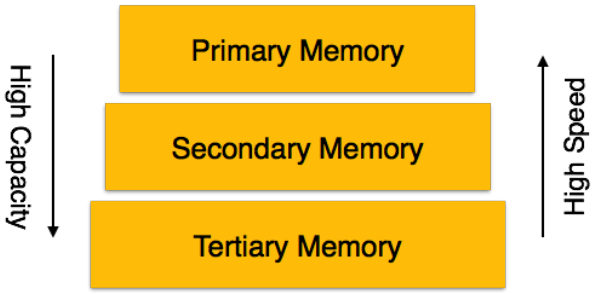
known as “key-value stores”; to use MySQL for such applications, use hash indexes wherever possible.
The optimizer cannot use a hash index to speed up ORDER BY operations. (This type of index cannot be used to search for the next entry in order.)

20. Explain the steps in query processing and write about measures of query cost.

PART C

1. Write about data on external storage.

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.



2. Illustrate Clustered Indexes.

- In a clustered index, records themselves are stored in the Index and not pointers. Sometimes the Index is created on non-primary key columns which might not be unique for each record.
- In such a situation, you can group two or more columns to get the unique values and create an index which is called clustered Index. This also helps you to identify the record faster.

Example:

- Let's assume that a company recruited many employees in various departments. In this case, clustering indexing in DBMS should be created for all employees who belong to the same dept.
- It is considered in a single cluster, and index points point to the cluster as a whole. Here, Department _no is a non-unique key.

3. Discuss the primary and secondary indexes.

Indexing in Database is defined based on its indexing attributes. Two main types of indexing methods are:

- Primary Indexing
- Secondary Indexing

Primary Index in DBMS

Primary Index is an ordered file which is fixed length with two fields. The first field is the same as a primary key and the second field is pointed to that specific data block. In the primary Index, there is always one to one relationship between the entries in the index table.

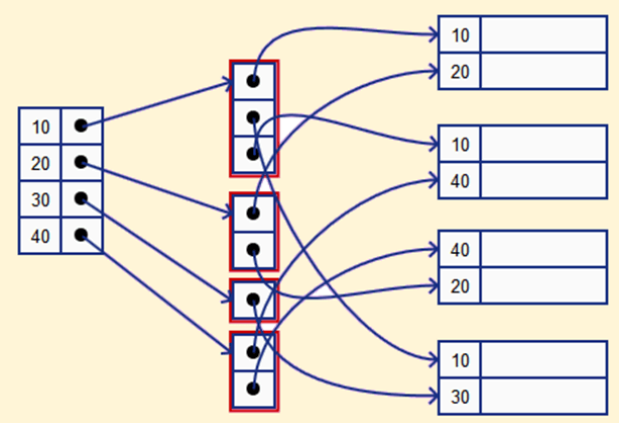
The primary Indexing in DBMS is also further divided into two types.

- Dense Index
- Sparse Index

Secondary Index in DBMS

The secondary Index in DBMS can be generated by a field which has a unique value for each record, and it should be a candidate key. It is also known as a non-clustering index.

This two-level database indexing technique is used to reduce the mapping size of the first level. For the first level, a large range of numbers is selected because of this; the mapping size always remains small.



Refer this link: [Indexing in DBMS: What is, Types of Indexes with EXAMPLES]

4. Define Tree hashing.

5. Describe Storage Hierarchy.

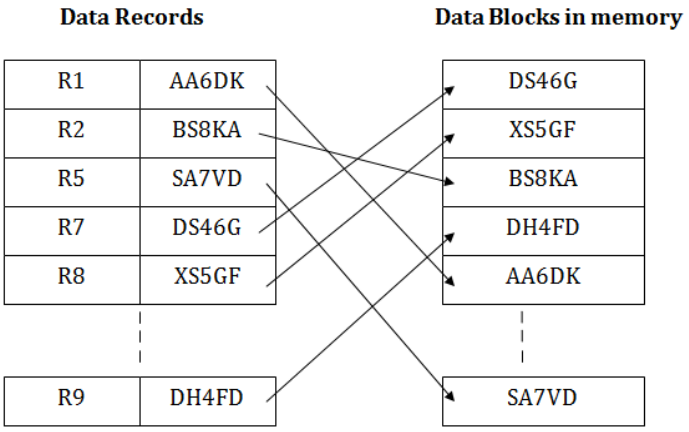
Refer question no:17(part-B)

6. Discuss the intuition for tree indexes.

7. Define Indexed Sequential Access Method.

Indexed sequential access method (ISAM):

ISAM method is an advanced sequential file organisation. 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.



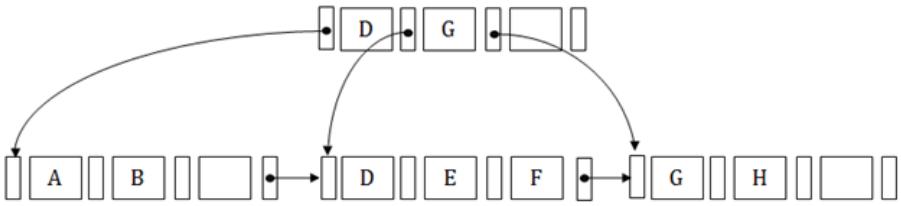
➔ If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

8. Discuss about Overflow pages and locking considerations of ISAM.

9. Describe structure of B+ tree nodes.

Structure of B+ Tree

- In the B+ tree, every leaf node is at equal distance from the root node. The B+ tree is of the order n where n is fixed for every B+ tree.
- It contains an internal node and leaf node.



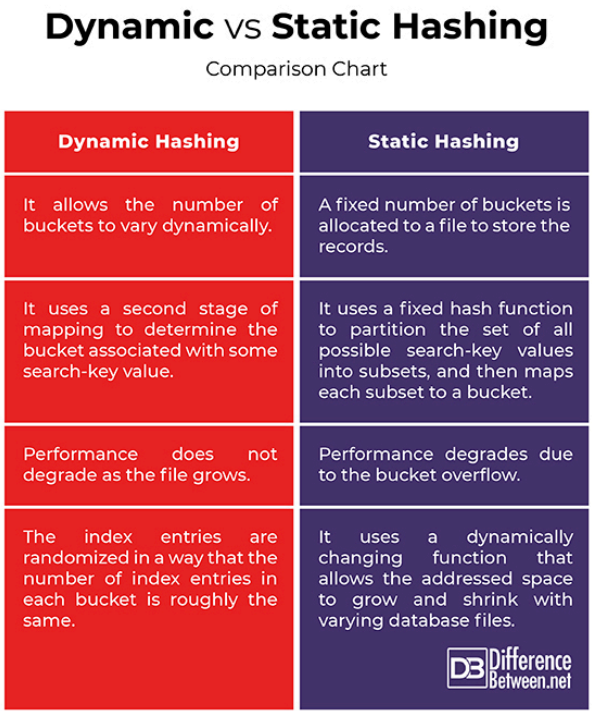
Internal node

- An internal node of the B+ tree can contain at least n/2 record pointers except the root node.
- At most, an internal node of the tree contains n pointers.

Leaf node

- The leaf node of the B+ tree can contain at least n/2 record pointers and n/2 key values.
- At most, a leaf node contains n record pointer and n key values.
- Every leaf node of the B+ tree contains one block pointer P to point to next leaf node.

10. Compare dynamic and static hash techniques.



Dynamic vs Static Hashing

Comparison Chart

Dynamic Hashing	Static Hashing
It allows the number of buckets to vary dynamically.	A fixed number of buckets is allocated to a file to store the records.
It uses a second stage of mapping to determine the bucket associated with some search-key value.	It uses a fixed hash function to partition the set of all possible search-key values into subsets, and then maps each subset to a bucket.
Performance does not degrade as the file grows.	Performance degrades due to the bucket overflow.
The index entries are randomized in a way that the number of index entries in each bucket is roughly the same.	It uses a dynamically changing function that allows the addressed space to grow and shrink with varying database files.

Refer: [Difference Between Dynamic and Static Hashing]

11. What is a timestamp?

- Timestamp is a unique identifier created by the DBMS to identify the relative starting time of a transaction.
- Typically, timestamp values are assigned in the order in which the transactions are submitted to the system.

- So, a timestamp can be thought of as the transaction start time. Therefore, time stamping is a method of concurrency control in which each transaction is assigned a transaction timestamp.

12. List the different file organisations.

➔ File Organisation refers to the logical relationships among various records that constitute the file, particularly with respect to the means of identification and access to any specific record.

➔ In simple terms, Storing the files in a certain order is called file Organisation.

Types of File Organisations –

Various methods have been introduced to Organise files.

- Sequential File Organisation
- Heap File Organisation
- Hash File Organisation
- B+ Tree File Organisation
- Clustered File Organisation

13. What is log?

- Log is nothing but a file which contains a sequence of records, each log record refers to a write operation. All the log records are recorded step by step in the log file. We can say, log files store the history of all updates activities.
- Log contains start of transaction, transaction number, record number, old value, new value, end of transaction etc. For example, mini statements in bank ATMs.

- If within an ongoing transaction, the system crashes, then by using log files, we can return back to the previous state as if nothing has happened to the database.
- The log is kept on disk so that it is not affected by failures except disk and failures.

14. List the steps in Query processing.

Query Processing is the activity performed in extracting data from the database. In query processing, it takes various steps for fetching the data from the database. The steps involved are:

1. Parsing and translation
2. Optimization
3. Evaluation

The query processing works in the following way:

Parsing

the parser of the query processor module checks the syntax of the query, the user's privileges to execute the query, the table names and attribute names, etc.

Translation

If we have written a valid query, then it is converted from high level language SQL to low level instruction in Relational Algebra.

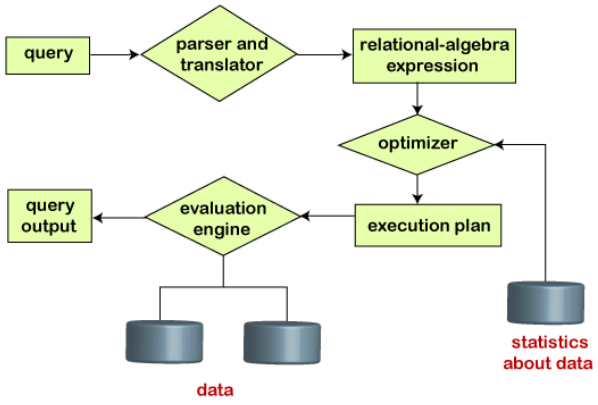
Optimizer

Optimizer uses the statistical data stored as part of a data dictionary. The statistical data are information about the size of the table, the length of

records, the indexes created on the table, etc. Optimizer also checks for the conditions and conditional attributes which are parts of the query.

Evaluation:

At this stage, we choose one execution plan of the several we have developed. This Execution plan accesses data from the database to give the final result.



Steps in query processing

15. What is blind writing?

Blind write is simply when a transaction writes without reading. i.e a transaction has WRITE(Q), but no READ(Q) before it. So, the transaction is writing to the database "blindly" without reading the previous value.

If there is no read that happens prior to the first write then it is said to be a *blind write*.

T1	T2	T3
	$R_2(X)$	
$R_1(X)$		
		$W_3(X)$
	$W_2(X)$	

- W3(X) is a blind write, as there is no read before write [R3(X) before W3(X)]
- W2(X) is not a blind write, as a read happens before write [R2(X) before W2(X)]

16. Describe immediate database modification.

The immediate database modification technique allows database modification to be output to the database while the transaction is still in the active state. The data modification written by active transactions are called **"uncommitted modification"**.
If the system crashes or transaction aborts, then the old value field of the log records is used to restore the modified data items to the value they had prior to the start of the transaction. This restoration is accomplished through the undo operation.
Refer: What is Immediate database modification? | Practice | GeeksforGeeks

17. Discuss the advantages of heap file organisation.

Heap File Organisation

This is the simplest form of file organisation. 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.

Advantages of Heap file organisation:

- Very good method of file organisation for bulk insertion. i.e.; when there is a huge amount of data needed to load into the database at a time, then this method of file organisation 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.

18. What is transaction failure?

The transaction failure occurs when it fails to execute or when it reaches a point from where it can't go any further. If a few transactions or processes are hurt, then this is called a transaction failure.
Reasons for a transaction failure could be -

1. Logical errors: If a transaction cannot complete due to some code error or an internal error condition, then the logical error occurs.
2. Syntax error: It occurs where the DBMS itself terminates an active transaction because the database system is not able to execute it. For example, The system aborts an active transaction, in case of deadlock or resource unavailability.

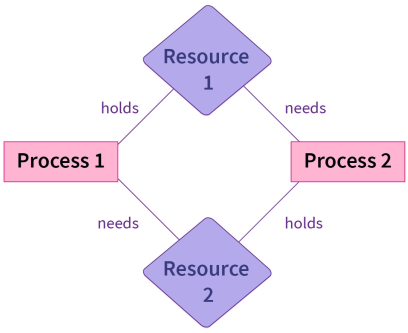
19. Identify when a transaction system is in a deadlock state.

- In a database, a deadlock is an unwanted situation in which 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 it brings the whole system to a Halt.

Necessary Conditions for Deadlock:
The four necessary conditions for a deadlock to arise are as follows.

- Mutual Exclusion: Only one process can use a resource at any given time i.e. the resources are non-sharable.

- Hold and wait: A process is holding at least one resource at a time and is waiting to acquire other resources held by some other process.
- No preemption: The resource can be released by a process voluntarily i.e. after execution of the process.
- Circular Wait: A set of processes are waiting for each other in a circular fashion.



20. What is the locking protocol?

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

1. Shared lock:

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

2. Exclusive lock:

- In the exclusive lock, the data item can be both read and written by the transaction.
- This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously.