Database Management System: Assignment 6

Total Marks : 20

July 29, 2024

Question 1

Marks: 2 **MSQ**

Identify the incorrect statement(s) from the following.

a) In primary index, index entries can be stored in unsorted order of the search key value.

b) In a secondary index file, all the search key values must be presented.

c) A clustered index is built by default on any sorted valued columns.

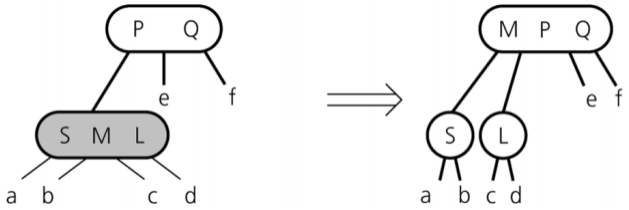
d) Sequential scan using a secondary index is expensive.

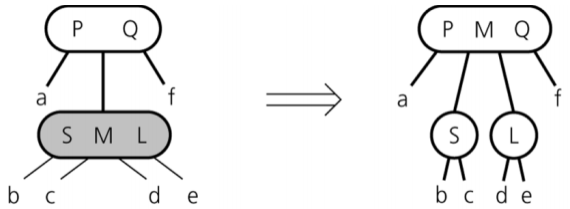
Answer: a), c)
Explanation: secondary indices have to be dense. A Secondary index contains all key values and each key value entry has a pointer to the record that contains the key value. And sequential scan using a secondary index is expensive. So, options b) and d) are correct statement.
Whereas, in the primary index, index entries must be stored in **sorted order** of the search key value. And the clustered index is built by default on unique key columns. So, options a) and c) are incorrect.
Hence, options a) and c) are the answer.

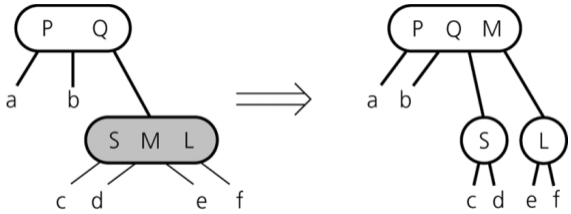
Question 2

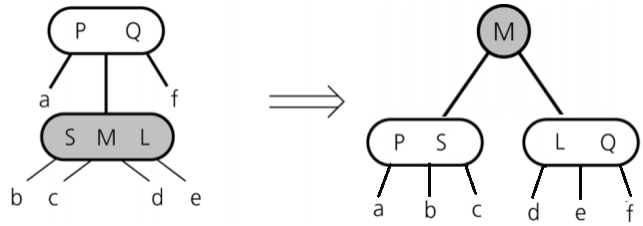
Marks: 2 **MCQ**

Which of the splitting rule is not correct in respect of 2-3-4 Tree?

a) 

b) 

c) 

d) 

Answer: d)
Explanation: Splittings corresponding to options (a), (b) and (c) are correct. Refer to slide 27.
In option (d), links are inserted incorrectly. So, option (d) is the answer.

Question 3

Marks: 2 **MCQ**

Consider a system that uses `B+ tree` for indexing its records. Calculate the order of a `non-leaf` node, if the `block size` is 2 kilobytes, size of the `search key field` is 10 bytes, and the `block pointer size` is 18 bytes?

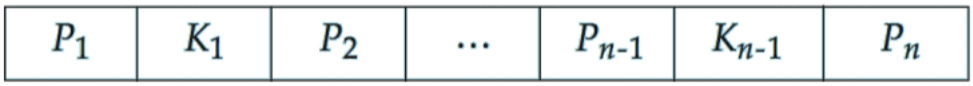
a) 44

b) 73

c) 113

d) 204

Answer: b)
Explanation: A `B+tree` of order m means every node has at most m children.
So, there will be only m number of block pointers and $(m-1)$ number of key value in any non-leaf nodes.



According to the question, maximum size of a non-leaf node $\leq$ 2 KB
size of search key field * $(m-1)$ + block pointer * $m \leq 2048$
10 * $(m-1)$ + 18 * $m \leq 2048$
$10m$ - 10 + $18m \leq 2048$
28m ≤ 2058
So, order of non-leaf node $(m) = 73$
Hence, option b) is correct.

Question 4

Marks: 2 **MSQ**

A hash table contains 10 (indices 0 to 9) buckets and uses linear probing to resolve collisions. The key values are integers and the hash function used is `(key % 10)`. Given the following input `(543, 432, 741, 330, 471, 351, 170, 245)`, which of the following statement(s) is/are true?

a) 245 is stored in bucket 5.

b) 741, 471, 351 are stored in buckets 1, 4, and 5.

c) 432, 543 are stored in buckets 2 and 3.

d) 170 is stored in bucket 4.

Answer: b), c)
Explanation: After inserting all keys, the buckets will be:

330	741	432	543	471	351	170	245		
0	1	2	3	4	5	6	7	8	9

Hence, options b) and c) are correct.

Question 5

Marks: 2 **MCQ**

Suppose, there is a DBMS file with size 16 kilobytes. Calculate the size of the index table, if the size of one record is 64 bytes long which includes 10 bytes key field assuming the system uses primary indexing and the record pointer is 48–bit long?

a) 1 kilobyte

b) 2 kilobytes

c) 3 kilobytes

d) 4 kilobytes

Answer: d)
Explanation: In primary indexing, one index entry contains key value and record pointer.
Hence, one index entry = size of one key + size of record pointer = 10 bytes + 6 bytes
Now, the file size = 16 KB
Size of one record = 64 bytes
So, number of entries in the index table = 16 KB ÷ 64 bytes=2^8
Size of the index table = 2^8 * 16 bytes = 4 KB
Hence, option d) is correct.

Question 6

Marks: 2 **MCQ**

Consider the following table `Patient`. If we want to create an index on `PatID` column, which type of indexing will be preferred?

PatID	PName	ContNumber
2012	Aakash Basu	82404
1016	Tina Mitra	84624
3051	Mita Sarkar	82432
1227	Rana Mall	83668
1355	Barun Sen	81664
1465	Sushma Nandi	81744

a) Sparse index

b) Dense index

c) Secondary index

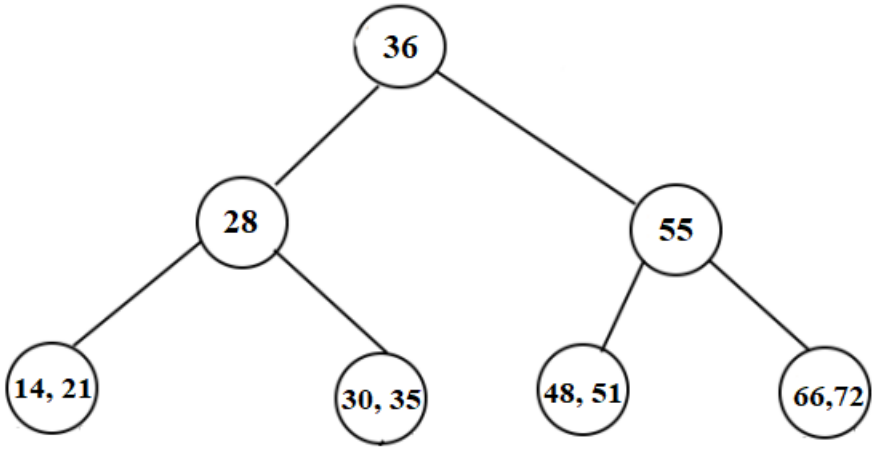
d) Non-clustering index

Answer: b)
Explanation: When the file is not sorted on the indexed field or when the index file is small, compared to the size of the memory, it is preferable to use dense indexing. The above table has only 6 records and **PatID** field is not in sorted order.
Hence, option b) is correct.

Question 7

Marks: 2 **MCQ**

There is a `2-3-4 Tree` as follows:



For searching `34`, how many comparisons will be required ?

a) 11

b) 7

c) 4

d) 3

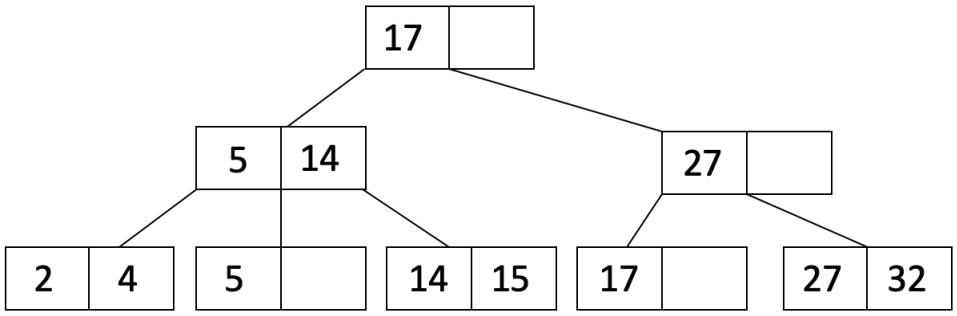
Answer: c)
Explanation: For finding `34`, `4` comparisons will be required: first comparison with `36`, second comparison with `28`, third comparison with `30` and the last comparison with `35`. Then only it can be said that the key is not found.
So, option (c) is correct.

Question 8

Marks: 2 **MCQ**

See the following B+ tree. Find the minimum number of nodes (including the root node) that must be fetched in order to satisfy the following query:

"Find all the records greater than 5 and less than 17"



a) 4

b) 5

c) 6

d) 7

Answer: b)

Explanation: All the leaf nodes are connected with pointers in B+ tree. So first, we need to search for key less than (if 5 does not present) or equal to 5. We start searching from the root node and move to the node with key 5 and 14. As we need the value greater than 5, we will move to the leaf node with 5. We will traverse and search the nodes upto the node having key 17.

So, the total number of nodes fetched = 2 (for searching 5) + 3(for finding keys less than 17) = 5.

So, option b) is correct.

Question 9

Marks: 2 **MCQ**

Consider a shop maintained a relation `Order(Order_No, ItemID, Quantity)` for storing order information. We have created a `bitmap index` on `Order` table and the size of the index file is 512 bytes for `256` number of rows. How many different types of items are availble in the shop assuming `ItemID` as the index column?

a) 16

b) 8

c) 4

d) 2

Answer: a)
Explanation: A `bitmap index` of N number of row with m distinct values in the index column will have N x m bits in the bitmap.
So, the size of the `index file` will be (256 x m) bits = 512 bytes = 512*8 bits
So, $m = (512 * 8)/256 = 16$ bits
Only 16 different types of items are available in that shop for order.
Hence, option a) is correct.

Question 10

Marks: 2 **MCQ**

Consider the following `OrderTable`:

OrderTable			
Order_no	Cust_id	salesman_id	Amount
A001	1005	2	750
A009	3001	4	270
A002	3002	1	1500
B004	5001	3	2300
B007	1005	22	830
A005	6007	51	1100
...	...	...	...

For the `OrderTable` table, assume that there are 10 salesmen, and each salesman is associated with a minimum of 500 orders. Suppose `OrderTable` is commonly queried as:
`SELECT * FROM OrderTable WHERE Order_no = 'XXXX' AND salesman_id = Y;`
For speeding up the execution of queries, it is decided to create an index on the table. Which of the following query will create such an index?

a) `CREATE INDEX ind_Order ON Order(Order_no);`

b) `CREATE INDEX ind_Order ON Order(salesman_id);`

c) `CREATE INDEX ind_Order ON Order(Order_no, salesman_id);`

d) `CREATE MULTI INDEX ind_Order ON Order(Order_no, salesman_id);`

Answer: c)
Explanation: Instead of creating a single column index, one can create a `composite index` (using several columns), and the same index can be used for queries that reference all of these columns, or just some of them.

The **order of columns** in the `CREATE INDEX` statement of `composite index` can affect the performance. Put the column expected to be used **most often first** in the index.

Here, we have to create a `composite index` with the most selective (with most values column `Order_no`) first, `salesman_id` next.
`CREATE MULTI INDEX` command is wrong.
So, option c) is correct.