National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Advance Database Concepts**	**Course Code:**	**CS4064**
Program:	**BS (Computer Science)**	**Semester:**	**Spring 2025**
Out Date:	**21-Mar-2025**	**Total Marks:**	
Due Date:	**Thu 27-Mar-2025** ***(Start of class)***	**Weight:**	
		Page(s):	**2**
Assignment:	**3 (Indexing Structures)**		

Instructions:

- Use any valid assumption where needed.
- You are required to submit the hard copy of your assignment at the start of your class.
- For any queries, please contact your TA.

Take the following assumptions for the block size and order file size to solve the questions:

Q1. Block Size **B= 4096 b**ytes and File Records (fixed length and un-spanned) **r= 10b**illion
Q2. Block Size **B= 8192 b**ytes and File Records (fixed length and un-spanned) **r= 100m**illion

A block pointer (P) is 5 bytes long and a record pointer (P_R) is 6 bytes long. Record length (R) is 120 bytes long and assume that the size of each field is 10 bytes long.

a. Suppose that the file is *ordered* by the key field OrderNo and we want to construct a *primary* index on OrderNo. Calculate (i) the index blocking factor bfr_i (which is also the index fan-out *fa);* (ii) the number of first-level index entries and the number of first-level index blocks; (iii) the number of levels needed if we make it into a multilevel index; (iv) the total number of blocks required by the multilevel index; and (v) the number of block accesses needed to search for and retrieve a record from the file given its OrderNo value using the primary index.

b. Suppose that the file is not *ordered* by the key field OrderNo and we want to construct a *secondary* index on OrderNo. Repeat the previous (part a) for the secondary index and compare it with the primary index.

c. Suppose that the file is not *ordered* by the non-key field CustID and we want to construct a *secondary* index on CustID, with an extra level of indirection that stores record pointers. Assume there are 50,000 distinct values of CustID and that the Order records are evenly distributed among these values. Calculate (i) the index blocking factor bfr, (which is also the index fan-out *fa);* (ii) the number of blocks needed by the level of indirection that stores record pointers; (iii) the number of first level index entries and the number of first-level index blocks; (iv) the number of levels needed if we make it into a multilevel index; (v) the total number of blocks required by the multilevel index and the blocks used in the extra level of indirection; and (vi) the approximate number of block accesses needed to search for and retrieve all records in the file that have a specific CustID value, using the index.

d. Suppose that the file is *ordered* by the non-key field CustID and we want to construct a *clustering index* on CustID that uses block anchors (every new value of CustID starts at the beginning of a new block). Assume there are 50,000 distinct values of CustID and that the Order records are evenly distributed among these values. Calculate (i) the index blocking factor bfr, (which is also the index fan-out *fa);* (ii) the number of first-level index entries and the number of first-level index blocks; (iii) the number of levels needed if we make it into a multilevel index; (iv) the total number of blocks required by the multilevel index; and (v) the number of block accesses needed to search for and retrieve all records in the file that have a specific CustID value, using the clustering index (assume that multiple blocks in a cluster are contiguous).

e. Suppose the file is not ordered by the key field OrderNo and we want to construct a B^+-tree access structure (index) on OrderNo. Calculate (i) the orders p and p leaf of the B^+-tree; (ii) the number of leaf-level blocks needed if blocks are approximately 70% full (rounded up for convenience); (iii) the number of levels needed if internal nodes are also 70% full

(rounded up for convenience); (iv) the total number of blocks required by the B^+-tree; and (v) the number of block accesses needed to search for and retrieve a record from the file--given its OrderNo value--using the B^+-tree.

f. Repeat (part e), but for a B-tree *rather than for a* B^+-tree. Compare your results with the B^+-tree.

Q3. Consider a DBMS that has the following characteristics:

- 2KB fixed-size blocks
- 6-bytes block pointer and 7-bytes record pointer
- 50-bytes block headers

We want to build an index on a search key that is 10 bytes long. Calculate the maximum number of records we can index with a

a. 4 Level B^+- tree index (including the root level)
b. 4 Level B-tree index (including the root level)

Q4. Assume an un-ordered relation Customer *(CustID, Rating, Age)* is given with CustID as a key attribute and the data types of all attributes are integer. Size of the relation is 10,000 data blocks. Assume there is a four-level of B^+-tree index exist on CustID attribute. Moreover, one node of the B^+-tree is stored in one block on the disk. Take any valid assumption where needed.

Estimate the number of block fetches needed to compute the following queries:

a. SELECT * FROM Customer WHERE CustID= 100;
b. SELECT age FROM Customer WHERE CustID= 200 AND Rating= 5;
c. SELECT * FROM Customer WHERE CustID= 100 OR CustID= 200;
d. SELECT CustID, age FROM Customer WHERE Rating= 5;
e. SELECT * FROM Customer WHERE CustID>= 100;
f. SELECT COUNT(*) FROM Customer WHERE CustID>= 100;