# National University of Computer and Emerging Sciences, Lahore Campus

| | | | | |
|---|---|---|---|---|
| | Course: | Advanced Database Concepts | Course Code: | CS |
| | Program: | BS(Computer Science) | Semester: | Spring 2017 |
| | Out Date: | 22-March-2017 | Total Marks: | |
| | Due Date: | 29-March-2017(start of class) | Weight: | |
| | Section | A | Page(s): | 1 |
| | Assignment: | 3 (Indexing) | | |

**Instruction/Notes:**

Consider the following scenario:

A block pointer is P = 6 bytes long and a record pointer is $P_R$ = 7 bytes long. A file has above records of fixed length. Each record has the following fields: CompanyName (35 bytes), CompanyId (8 bytes), MainWarehouse (9 bytes), Address (41 bytes), Phone (9 bytes), CreationDate (8 bytes), Branches (1 byte), Code (5 bytes), Assets (8 bytes, real number). An additional byte is used as a deletion marker.

**Question 1:** Block Size **B=5041 b**ytes and File Records **r=10m**illion

**Question 2:** Block Size **B=9216 b**ytes and File Records **r=10b**illion

Attempt all following parts first using Question1 conditions and then Question2 conditions.

**a.** Calculate the record size R in bytes.

**b.** Calculate the blocking factor bfr and the number of file blocks b, assuming an un-spanned organization.

**c.** Suppose that the file is *ordered* by the key field CompanyId and we want to construct a *primary* index on CompanyId. Calculate (i) the index blocking factor bfri (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 CompanyId value-using the primary index.

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

**e.** Suppose that the file is not *ordered* by the nonkey field MainWarehouse and we want to construct a *secondary* index on MainWarehouse, with an extra level of indirection that stores record pointers. Assume there are 20000 distinct values of MainWarehouse and that the Company 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 firstlevel 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 MainWarehouse value, using the index.

**f.** Suppose that the file is *ordered* by the nonkey field MainWarehouse and we want to construct a *clustering index* on MainWarehouse that uses block anchors (every new value of MainWarehouse starts at the beginning of a new block). Assume there are 20000 distinct values of MainWarehouse and that the Company 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 MainWarehouse value, using the clustering index (assume that multiple blocks in a cluster are contiguous)

**g.** Suppose the file is not ordered by the key field CompanyId and we want to construct a $B^+$-tree access structure (index) on CompanyId. Calculate (i) the orders p and p leaf of the $B^+$-tree; (ii) the number of leaf-level blocks needed if blocks are approximately 69% full (rounded up for convenience); (iii) the number of levels needed if internal nodes are also 69% 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 SSN value--using the $B^+$-tree.

**h.** Repeat part g, but for a B-tree *rather than for a* $B^+$-tree. Compare your results for the B-tree and for the $B^+$-tree.

**Question 3:**

Assume a relation R (W, Y, Z) is given; Suppose W , Y , Z are integer type values. R is stored as an un-ordered file (un-spanned) on key field W and contains 500 data blocks. Assume there is $B^+$- tree access structure (index) on W of height x=3 (root, 1 intermediate layer, leaf). Moreover, one node of the $B^+$-tree is stored in one block on the disk.

Estimate the number of block fetches needed to compute the following queries:
**a)** SELECT X,Y,Z FROM R WHERE W = 122;
**b)** SELECT Y,Z FROM R WHERE W = 232 AND Y = 597;
**c)** SELECT X,Y,Z FROM R WHERE W = 491 OR W = 237;
**d)** SELECT Y,Z FROM R WHERE W > 250 ;

**Question 4:**

Suppose a table contains 1 million records and an index is created using linked list data structure on this table. State various advantages and disadvantages of using linked list index in this scenario.