CS 6360.002/003 Assignment-5

Due Date: Nov 17, 2017, 11:59PM

1. Consider a disk with block size B=512 bytes. A block pointer is P=6 bytes long, and a record pointer is P R =7 bytes long. A file has r=3000 EMPLOYEE records of fixed-length. Each record has the following fields: NAME (30 bytes), SSN (10 bytes), DEPARTMENTCODE (10 bytes), ADDRESS (30 bytes), PHONE (10 bytes), BIRTHDATE (10 bytes), GENDER (1 byte), JOBCODE (4 bytes), SALARY (4 bytes, real number). An additional byte is used as a deletion marker.

(a) Calculate the record size R in bytes.
Record size = 30+10+10+30+10+10+1+4+4+1 = 110 bytes

(b) Calculate the blocking factor bfr and the number of file blocks b assuming an unspanned organization.
bfr = floor(512/110) = floor(4.65) = 4 records in a single block {unspanned organization}
b = 3000/4 = 750

(c) Suppose the file is ordered by the key field SSN and we want to construct a primary index on SSN. Calculate

(i) the index blocking factor bfr i;
index record size = SSN + Block Pointer = 10+ 6 = 16 bytes
bfr_i = floor(512/16) = 32

(ii) the number of first-level index entries and the number of first-level index blocks;
No of first-level index entries = No of file blocks = 750
No of first-level index blocks = ceil(750/bfr_i) = ceil(750/32) = ceil(23.43) = 24

(iii) the number of levels needed if we make it into a multi-level index;
b1 = ceil(750/32) = 24
b2 = ceil(24/32) = 1
No of levels needed = 2

(iv) the total number of blocks required by the multi-level index; and
Total No of blocks required by the multi-level index = 24 + 1 = 25 blocks

(v) the number of block accesses needed to search for and retrieve a record from the file--given its SSN value--using the primary index.
No of Block accesses = 2 + 1 = 3

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

Size of index R_i = 10+6 = 16 bytes

bfr_i = ceil(512/16) = 32 records per block

No of first-level index entries = 3000
No of first-level index blocks = ceil(3000/32) = 94 blocks

No of level needed:
No of 2nd level entries = 94
No of 2nd level blocks = ceil(94/32) = 3 blocks

No of 3rd level entries = 3
No of 3rd level blocks = ceil(3/32) = 1 block
Hence, No of levels needed is 3.

Total No of blocks required by the multi-level index = 94 + 3 + 1 = 98 blocks

No of Block accesses = 3 + 1 = 4

(e) Suppose the file is not ordered by the non-key field DEPARTMENTCODE and we want to construct a secondary index on DEPARTMENTCODE using Option 3 of Section 18.1.3, with an extra level of indirection that stores record pointers.
Assume there are 100 distinct values of DEPARTMENTCODE, and that the EMPLOYEE records are evenly distributed among these values. Calculate

(i) the index blocking factor bfr_i;
Size of index R_i = 10+6 = 16 bytes
bfr_i = ceil(512/16) = 32 records per block

(ii) the number of blocks needed by the level of indirection that stores record pointers;
There are 100 distict values of DEPARTMENTCODE,
Average no of records of each value = 3000/100 = 30
Record Pointer = 7 bytes
The No of blocks needed by the level of indirection = 7 * 30 = 210 bytes
So, 210 bytes fits into one block. And therefore, we would need 100 blocks by the level of indirection

(iii) the number of first-level index entries and the number of first-level index blocks;
No of first-level entries = 100
No of first-level index blocks = ceil(100/32) = 4 blocks

(iv) the number of levels needed if we make it a multi-level index;
No of second-level entries = 4
No of second-level index blocks = ceil(4/32) = 1 block
No of levels needed to make it a multi-level index = 2

(v) the total number of blocks required by the multi-level index and the blocks used in the extra level of indirection; and
The total number of blocks required by the multi-level index = 4 + 1 = 5 blocks
The blocks used in the extra level of indirection = 100 + 5 = 105 blocks

(vi) the approximate number of block accesses needed to search for and retrieve all records in the file having a specific DEPARTMENTCODE value using the index.
The No of block accesses = 1 + 2 + 30 (additional block access to retrieve all 3) = 33 block accesses

(f) Suppose the file is ordered by the non-key field DEPARTMENTCODE and we want to construct a clustering index on DEPARTMENTCODE that uses block anchors (every new value of DEPARTMENTCODE starts at the beginning of a new block).
Assume there are 100 distinct values of DEPARTMENTCODE, and that the EMPLOYEE records are evenly distributed among these values. Calculate

(i) the index blocking factor bfr_i;
Size of index R_i = 10+6 = 16 bytes
bfr_i = ceil(512/16) = 32 records per block

(ii) the number of first-level index entries and the number of first-level index blocks;
No of first-level index entries = 100
No of first-level index blocks = ceil(100/32) = 4 blocks

(iii) the number of levels needed if we make it a multi-level index;
No of second-level entries = 4
No of second-level index blocks = ceil(4/32) = 1 block
No of levels needed to make it a multi-level index = 2

(iv) the total number of blocks required by the multi-level index; and
The total number of blocks required by the multi-level index = 4 + 1 = 5 blocks

(v) the number of block accesses needed to search for and retrieve all records in the file having a specific DEPARTMENTCODE value using the clustering index (assume that multiple blocks in a cluster are either contiguous or linked by pointers).
The 30 records can be clustered in ceil(30/4) = 8 blocks
No of total block accesses needed on an average to retrieve all records = 2+8 = 10

(g) Suppose the file is not ordered by the key field Ssn and we want to construct a B+ tree access structure (index) on SSN. Calculate
(i) the orders p and p leaf of the B+ tree;
Here, Key field, k = 10 bytes
Block Pointer, b = 6 bytes
Record Pointer, r = 7 bytes

let the Order be 'p' of the B+ tree,
As internal node of the B+ tree can have up to n tree pointers and n-1 search field values, these must fit into a single block. Hence, we have,

$$(n * b) + (n - 1) * k <= \text{block size}$$
$$(n * 6) + (n - 1) * 10 <= 512$$
$$16n <= 522$$
$$n <= 522/ 16$$
$$n <= 32.62$$
$$n = 32$$

OR

here, n:= degree of B+ tree which can be calculated if we know maximum no of key a internal node can have. So, the order of the internal node be 'n'. Hence, using the formula:
(n - 1) k(key size)+ n b(block pointer size) = block size
(n - 1) 10 + 6n = 512
n = 522/16 = floor(32.62)
$n_{internal}$ = 32
which means that each internal node can hold up to 32 tree pointers, and 31 search key values.

The order of a leaf node in a B+ tree is the maximum number of (value, data record pointer) pairs it can hold. So, the order of the leaf node be 'n'
order of leaf node B+ tree can be determined by formula:
n * k(key size)+ n* r(record pointer size) + b = block size
n * k + n * r + b = block size
n * (10 + 7) + 6 = 512
n = floor(506/17)

n_{leaf} = 29
which means each leaf node can hold up to n_{leaf} = 29 value/data pointer combinations, assuming data pointers are record pointers.

(ii) the number of leaf-level blocks needed if blocks are approximately 69% full (rounded up for convenience);
Assuming that each leaf-level node of the B+ tree is 69% full, each leaf-level node on an average will have
n_{leaf}*0.69 = 29*0.69 = 20.01 = 20 nodes.
Therefore, the tree will have approximately 20 data record pointers (p_{leaf}).

(iii) the number of levels needed if internal nodes are also 69% full (rounded up for convenience);
Assuming that each internal node of the B+ tree is 69% full, each internal node on the average will have
$n_{internal}$*0.69 = 32*0.69 = 22.08 = 22 nodes.
Therefore, the tree will have approximately 22 pointers ($p_{internal}$), with 21 values. So,

No of second-level index blocks = 150/22 = 6.81 = 7
No of thired-level index blocks = 7/22 = 1 block

Hence, No of Levels needed = 3 (not including root)

(iv) the total number of blocks required by the B+ tree; and

The total number of blocks needed by the B+ Tree = 150 + 7 + 1 = 158

(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.

The number of block accesses needed = No. of levels of the tree + 1 = 3 + 1 = 4

Root	1 node	21 entries	22 pointers
Level 1	22 nodes	21 * 22 = 462 entries	22*22 = 484 pointers
Level 2	484 nodes	21*484 = 10164 entries	22*484 = 10648 pointers
Leaf Level	10648 nodes	20 * 10648 = 212960 record pointers	

2. A PARTS file with Part# as key field includes records with the following Part# values: 23, 65, 37, 60, 46, 92, 48, 71, 56, 59, 18, 21, 10, 74, 78, 15, 16, 20, 24, 28, 39, 43, 47, 50, 69, 75, 8, 49, 33, 38. Suppose the search field values are inserted in the given order in a B+ tree of order p=4 and p-leaf =4; show how the final tree looks like. (20 points)

3. Optimize the following SQL query on the Company Database to find names of employees earning over $80,000 per year, names of projects in which they work more than 30 hours, where the project is located in Chicago and the manager of the project's controlling department started after January 1, 2009.

```
Select Lname, Fname, Pname, Hours
From   Project P, Employee E, Department D, Works_on W
Where  E.Ssn = W.Essn
  and  P.Dnum = D.Dnumber
  and  W.Pno = P.Pnumber
  and  Plocation = 'Chicago'
  and  Hours > 30
  and  Salary > 80000
  and  Mgr_start_date >= '1/1/2009'
```

Show the final query tree. (10 points)