Homework 2

Jeremy Desser

December 1, 2015

17.28

a)

$R = 30 + 9 + 40 + 10 + 8 + 1 + 4 + 4 + 4 + 3 + 1 = 114$ bytes

b)

Block size $B = 512$
$512/114 = 4$ R 56
$bfr = 4$

c)

(i)
$b/2(b = numblocks) = 20000/4/2 = 2500$
With double blocking, only needed for 1st one:
avg seek time = 30 msec

$rd = 12.5msec$

$avgTime = seek + rd + 2500 * btt = 30 + 12.5 + 2500 * 5.76 = 14442.5msec$
(ii)
The file blocks are not stored contiguously.
$avgTime = (seek + rd + btt) * 2500 = 120700msec$

d)

$log2(b) * 30msec = log2(5000) - 12.287$
$avgTime = (seek + rd + btt) * 12.287 = 593msec$

18.18

a)

Record length $R = (30 + 9 + 9 + 40 + 9 + 8 + 1 + 4 + 4) + 1 = 115$bytes

b)

bfr = $\lfloor B/R \rfloor = \lfloor 512/115 \rfloor = 4$ records per block

Number of blocks needed for file= $\lceil r/bfr \rceil = \lceil 30000/4 \rceil = 7500$

c)

(i)

Index record size $R_i = (V_{SSN} + P) = (9 + 6) = 15bytes$
Index blocking factor $bfr_i = fo = \lfloor B/R_i \rfloor) = \lfloor 512/15 \rfloor = 34$

(ii)

Number of first-level index entries r_1 = number of file blocks b = 7500 entries
Number of first-level index blocks $b_1 = \lceil r_1/bfr_i \rceil = \lceil 7500/34 \rceil = 221$ blocks

(iii)

Number of second-level index entries $r_2 = numberoffirst - levelblocksb_1 = 221entries$
Number of second-level index blocks$b_2 = \lceil r2/bfr_i \rceil = \lceil 221/34 \rceil = 7blocks$
Number of third-level index entries $r_3 = numberofsecond - levelindexblocksb_2 = 7entries$
Number of third-level index blocks$b_3 = \lceil r_3/bfr_i \rceil = \lceil 7/34 \rceil = 1$
Since the third level has only one block, it is the top index level. Hence, the index has $x = 3$ levels

(iv)

Total number of blocks for the index $b_i = b_1 + b_2 + b_3 = 221 + 7 + 1 = 229$ blocks

(v)

Number of block accesses to search for a record $= x + 1 = 3 + 1 = 4$

d)

(i)

Index record size $R_i = (V_{SSN} + P) = (9 + 6) = 15bytes$
Index blocking factor $bfr_i = fo = \lfloor B/R_i \rfloor) = \lfloor 512/15 \rfloor = 34$

(ii)

Number of first-level index entries r_1 =number of file records$r = 30000$
Number of first-level index blocks$b_1 = \lceil r_1/bfr_i \rceil = \lceil 30000/34 \rceil = 883blocks$

(iii)

We can calculate the number of levels as follows:
Number of second-level index entriesr_2 = number of first-level index blocks $b_1 = 883entries$
Number of second-level index blocks$b_2 = \lceil r_2/bfr_i \rceil = \lceil 883/34 \rceil = 26blocks$
Number of third-level index entries r_3 = number of second-level index blocks$b_2 = 26entries$
Number of third-level index blocks$b_3 = \lceil r_3/bfr_i \rceil = \lceil 26/34 \rceil = 1$ Since the third level has only one block, it is the top index level. The index has x = 3 levels

(iv)

Total number of blocks for the index$b_i = b_1 + b_2 + b_3 = 883 + 26 + 1 = 910$

(v)

Number of block accesses to search for a record= $x + 1 = 3 + 1 = 4$

e)

(i)

Index record size $R_i = (VdepartmentCode + P) = (9 + 6) = 15$ bytes

(ii)

There are 1000 values of DepartmentCode, the average number of records for each value is $r/1000 = 30000/1000 = 30$

Since a record pointer size P R = 7 bytes, the number of bytes needed at the level of indirection for each value of DepartmentCode is $7 * 30 = 210$ bytes which fits in one block. So 1000 blocks are needed for the level of indirection.

(iii)

Number of first level index entries R_1 =number of distinct values of DepartmentCode = 1000 entries
Number of first level index blocks $b_1 = \lceil R_1/bfr_i \rceil = \lceil 1000/34 \rceil = 30$ blocks

(iv)

Number of second level index entries r_2 = Number of first level index blocks $b_1 = 30$ blocks
Number of second level index blocks $b_2 \lceil r_2/bfr_i \rceil = \lceil 30/34 \rceil = 1$ so the index has $x = 2$ levels.

(v)

total number of blocks for the index $b_i = b_1 + b_2 + b$ indirection $= 30 + 1 + 1000 = 1031$ blocks.

(vi)

Number of block accesses to search for and retrieve the block containing the record pointers at the level of indirection $= x + 1 = 2 + 1 = 3$ block accesses.
Assuming that the 30 records are distributed over 30 distinct blocks we will need an additional 30

block accesses to retrieve all 30 records. So, total block accesses needed on average to retrieve all records with a given value for DepartmentCode = $x + 1 + 30 = 33$

f)

(i)

Index record size $R_i = (VDepartmentCode + P) = (9 + 6) = 15$ bytes
Index blocking factor $bfr_i fo = \lfloor B/R_i \rfloor = \lfloor 512/15 \rfloor = 34 index records per block$

(ii)

Number of first level index entries r_1 = number of distinct values of DepartmentCode = 1000 entries
Number of first level index blocks $b_1 = \lceil r_1/bfr_i \rceil = \lceil 1000/34 \rceil = 30$ blocks.

(iii)

Number of second level index entries r_2 = number of first level index blocks $b_1 = 30$ entries
Number of second level index blocks $b_2 = \lceil r_2/bfr_i \rceil = \lceil 30/34 \rceil = 1$ Since the second level has one block, it is the top level index.
the index has $x = 2$ levels

(iv)

total number of blocks for the index $b_i = b_1 + b_2 = 30 + 1$ blocks

(v)

Number of block accesses to search for the first block in the cluster of blocks = $x + 1 = 2 + 1 = 3$

The 30 records are clustered in $\lceil 30/bfr \rceil = \lceil 30/4 \rceil = 8$ blocks.
So, total block accesses needed on average to retrieve all records with a given value for DepartmentCode = $x + 8 = 2 + 8 = 10$ block accesses

g)

(i)

For a B+ tree of order p, the following inequality must be satisfied for each internal tree node:
$(p * P) + ((p - 1) * V_{snn}) < B$ or $(p * 6) + ((p - 1) * 9) < 512 = 15p < 521$, $p = 34$
For leaf nodes:
$(p_{leaf} * (V_{ssn} + P_R)) + P < B$ or $(p_{leaf} * (9 + 7)) + 6 < 512 = 15p < 512$, so $p = 34$

(ii)

Assuming nodes are 69 percent full on average, the number of key values in a leaf node are:
$0.69 * p_{leaf} = 0.69 * 31 = 21.39$ rounded up we get 22 key values and 22 record pointers per leaf nodes. Since the file has 30000 records and thus 30000 values of SSN, the number of leaf level nodes needed is $b_1 = \lceil 30000/22 \rceil = 1364$ blocks

(iii)

The number of levels are calculated as follows:
The average fan-out for the internal nodes (rounded up) is fo $= \lceil 0.69 * p \rceil = \lceil 0.69 * 34 \rceil = \lceil 23.46 \rceil = 24$
number of second level tree blocks, $b_2 = \lceil b_1/fo \rceil = \lceil 1364/24 \rceil = 57$ blocks
number of third level tree blocks $b_3 = \lceil b_2/fo \rceil = \lceil 57/24 \rceil = 3$
number of fourth level tree blocks $b_4 = \lceil b_3/fo \rceil = \lceil 3/24 \rceil = 1$
since the fourth level has one block, the tree has $x = 4$ levels

(iv)

total number of blocks for the tree $b_i = b_1 + b_2 + b_3 + b_4 = 1364 + 57 + 3 + 1 = 1425$ blocks

(v)

number of block acceses to search for a record $= x + 1 = 4 + 1 = 5$

19.13

a)

b)

c)

19.17

Yes,for the MIN, MAX operator, the optimizer can use the nondense-index to search for the smallest and biggest value in a similar way to dense index. For instance, to search for the biggest value, it will follow the rightmost pointer in each index node from the root to the right most leaf.
For COUNT, AVERAGE and SUM operators, it is slightly different. The actual number of records associated with each index entry must be used for a correct computation.

19.22