# horizontal lineDatabase R&D Exercise

Assignment 8

I confirm that this is my own work and that use of material from other sources, including the Internet, has been properly and fully acknowledged and referenced.

|  |  |
| --- | --- |
| Name: | Pang, Jinhao |
| Date: | 2022.11.22 |
| NYU ID: | N19475049 |
| Course Section Number: | csci-ga.2433-001 |



**Total in points** (100 points total): \_\_\_\_\_

**Professor’s Comments:**

|  |
| --- |
|  |

**16.34.**

**a. What is the total capacity of a track, and what is its useful capacity (excluding interblock gaps)?**

Total capacity: 20\*(512+128) = 12800 bytes.

Useful capacity: 20\*512=10240 bytes.

**b. How many cylinders are there?**

400

**c. What are the total capacity and the useful capacity of a cylinder?**

Total capacity: 15\*2\*12800=38400 bytes

Useful capacity: 20\*512\*10240 =30720 bytes.

**d. What are the total capacity and the useful capacity of a disk pack?**

Total capacity: 400\*38400=15360000 bytes

Useful capacity: 400\*30720=122880000 bytes.

**e. Suppose that the disk drive rotates the disk pack at a speed of 2,400 rpm (revolutions per minute); what are the transfer rate (tr) in bytes/msec and the block transfer time (btt) in msec? What is the average rotational delay (rd) in msec? What is the bulk transfer rate? (See Appendix B.)**

Transfer rate: 12800\*2400/60000 = 512 bytes/msec

Block transfer time: 512/br = 1 msec

Rotational delay: 60000/2400/2 = 12.5msec

Bulk transfer rate: (512/(512+128))\*512 = 409.6 bytes/msec

**f. Suppose that the average seek time is 30 msec. How much time does it take (on the average) in msec to locate and transfer a single block, given its block address?**

30+12.5+1 =43.5msec

**g. Calculate the average time it would take to transfer 20 random blocks, and compare this with the time it would take to transfer 20 consecutive blocks using double buffering to save seek time and rotational delay.**

20\*43.5 =870msec

30+12.5+20\*1 = 62.5msec

**16.36. Suppose that only 80% of the STUDENT records from Exercise 16.28 have a value for Phone, 85% for Major\_dept\_code, 15% for Minor\_dept\_code, and 90% for Degree\_program; and suppose that we use a variable-length record file. Each record has a 1-byte field type for each field in the record, plus the 1-byte deletion marker and a 1-byte end-of-record marker. Suppose that we use a spanned record organization, where each block has a 5-byte pointer to the next block (this space is not used for record storage).**

**a. Calculate the average record length R in bytes.**

There is no Exercise 16.28 in the textbook, so we only calculate the given schema.

R = ((phone+1)\*80%+ (Major\_dept\_code+1) \*85%+ (Minor\_dept\_code+1) \*15%+ (Degree\_program+1)\* 90%+ other attribute length+1+1)

**b. Calculate the number of blocks needed for the file.**

Assume block capacity is B,

The number is R\*numberOfRecords/(B-5)

**16.50. Suppose we have a sequential (ordered) file of 100,000 records where each record is 240 bytes. Assume that B = 2,400 bytes, s = 16 ms, rd = 8.3 ms, and btt = 0.8 ms. Suppose we want to make X independent random record reads from the file. We could make X random block reads or we could perform one exhaustive read of the entire file looking for those X records. The question is to decide when it would be more efficient to perform one exhaustive read of the entire file than to perform X individual random reads. That is, what is the value for X when an exhaustive read of the file is more efficient than random X reads? Develop this as a function of X.**

Exhaustive read time: s+rd+(100000\*240/2400)\*btt = 8024.3

So, X(s+rd+btt)>8024.3, then X>319.693227092

Hence, X should be at least 320.

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

(30+9+9+40+10+8+1+4+4)+1 = 116

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

bfr = 512/116 = 4

b = 30000/4 = 7500

**c. Suppose that the file is ordered by the key field Ssn and we want to con- struct a primary index on Ssn. Calculate**

**(i) the index blocking factor bfri (which is also the index fan-out fo);**

bfri = 512/(9+6) =34

**(ii) the number of first-level index entries and the number of** **first-level index blocks;**

first-level index entries: 7500

first-level index blocks: 7500/34+1= 221

**(iii) the number of levels needed if we make it into a multilevel index;**

Second-level index entries: 221

Second-level index blocks: 221/34 = 7

Third-level index entries: 7

Third-level index blocks: 221/34 = 1

Hence, the number will be 3.

**(iv) the total number of blocks required by the multilevel index; and**

1+7+221 = 229

**(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.**

4

**d. Suppose that 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.**

**(i) the index blocking factor bfri (which is also the index fan-out fo);**

bfri = 512/(9+6) =34

**(ii) the number of first-level index entries and the number of first-level index blocks;**

first-level index entries: 30000

first-level index blocks: 30000/34+1= 883

**(iii) the number of levels needed if we make it into a multilevel index;**

Second-level index entries: 883

Second-level index blocks: 883/34 = 26

Third-level index entries: 26

Third-level index blocks: 26/34 = 1

Hence, the number will be 3.

**(iv) the total number of blocks required by the multilevel index; and**

1+26+883 = 910

**(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.**

4

**e. Suppose that the file is not ordered by the nonkey field Department\_code and we want to construct a secondary index on Department\_code, using**

**option 3 of Section 17.1.3, with an extra level of indirection that stores record pointers. Assume there are 1,000 distinct values of Department\_code and that the EMPLOYEE records are evenly distributed among these values. Calculate**

**(i) the index blocking factor bfri (which is also the index fan-out fo);**

bfri = 512/(9+6) =34

**(ii) the number of blocks needed by the level of indirection that stores record pointers;**

The average number of records 30000/1000=30.

The average number of bytes needed in one block is 30\*7 = 210<512

Hence, the number of blocks is 1000.

**(iii) the number of first-level index entries and the number of first-level index blocks;**

first-level index entries: 1000

first-level index blocks: 1000/34+1= 30

**(iv) the number of levels needed if we make it into a multilevel index;**

Second-level index entries: 30

Second-level index blocks: 30/34+1 = 1

The number of levels is 2.

**(v) the total number of blocks required by the multilevel index and the blocks used in the extra level of indirection; and**

1+30 +1000=1031

**(vi) the approximate number of block accesses needed to search for and retrieve all records in the file that have a specific Department\_code value, using the index.**

2+30+1 = 33

**f. Suppose that the file is ordered by the nonkey field Department\_code and we want to construct a clustering index on Department\_code that uses block anchors (every new value of Department\_code starts at the beginning of a new block). Assume there are 1,000 distinct values of Department\_code and that the EMPLOYEE records are evenly distributed among these values. Calculate**

**(i) the index blocking factor bfri (which is also the index fan-out fo);**

bfri = 512/(9+6) =34

**(ii) the number of first-level index entries and the number of first-level index blocks;**

first-level index entries: 1000

first-level index blocks: 1000/34+1= 30

**(iii) the number of levels needed if we make it into a multi-level index;**

Second-level index entries: 30

Second-level index blocks: 30/34+1 = 1

The number of levels is 2.

**(iv) the total number of blocks required by the multilevel index; and**

1+30 = 31

**(v) the number of block accesses needed to search for and retrieve all records in the file that have a specific Department\_code value, using the clustering index (assume that multiple blocks in a cluster are contiguous).**

Each distinct values has an average of 30000/1000=30 records

Number of block accesses needed to search for the first block is: 2+1 = 3

Number of block accesses needed to search for the last block is: 2+(30/4)=10

**g. Suppose that 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;**

p = (512-6)/(9+6) = 33

p\_leaf = = (512-6)/(9+6+1) = 31

**(ii) the number of leaf-level blocks needed if blocks are approximately 69% full (rounded up for convenience);**

30000/roundup(69%\*p\_leaf) = 1364

**(iii) the number of levels needed if internal nodes are also 69% full (rounded up for convenience);**

Second level: 1364/roundup(69%\*p) = 60

Third level: 60/roundup(69%\*p) = 3

Fourth level: 3/roundup(69%\*p) = 1

The number of levels is 4.

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

1+3+60+1364 = 1428

**(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.**

4+1 = 5

**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.**

**(i) the orders p and p\_leaf of the B+-tree;**

(p-1)\*(9+7) + p\*6 <= 512, then p = 24

p\_leaf = 24\*69% = 16

**(ii) the number of leaf-level blocks needed if blocks are approximately 69% full (rounded up for convenience);**

Second level: 16\*16=256

Third level: 256\*16 = 4096

Hence, the number is 4096

**(iii) the number of levels needed if internal nodes are also 69% full (rounded up for convenience);**

Level 3 entries: 4096\*15 = 61440>30000

Hence, the number is 3+1 = 4

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

1+16+256+4096 = 4369

**(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.**

4+1 = 5

**17.26. It is possible to modify the B+-tree insertion algorithm to delay the case where a new level is produced by checking for a possible redistribution of values among the leaf nodes. Figure 17.17 illustrates how this could be done for our example in Figure 17.12; rather than splitting the leftmost leaf node when 12 is inserted, we do a left redistribution by moving 7 to the leaf node to its left (if there is space in this node). Figure 17.17 shows how the tree would look when redistribution is considered. It is also possible to consider right redistribution. Try to modify the B+-tree insertion algorithm to take redistribution into account.**

For left redistribution: if the nodes need to be inserted in full, move the leftmost value to the leaf node to its left. This step can be considered as inserting the leftmost value into the left leaf, which can be seen as recursively. If all of the leaf nodes are full, try to split to increase the level.

For right redistribution: the procedure is similar, just insert the rightmost value to the leaf node to its right recursively.

**18.14. A file of 4,096 blocks is to be sorted with an available buffer space of 64 blocks. How many passes will be needed in the merge phase of the external sort-merge algorithm?**

nR = b/nB= 4096/64 = 64

dM = min(nR,nB-1) = 63

pass = logdM(nR) = 2

Hence, 2 passes will be needed

**19.19.** **Develop formulas for the hybrid hash-join algorithm for calculating the size of the buffer for the first bucket. Develop more accurate cost estimation formulas for the algorithm.**

bE+bD+bRES