|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
| final design | **Course:** | **Advance Database Systems** | **Course Code:** | **CS451** |
| **Program:** | **BS(Computer Science)** | **Semester:** | **Spring 2018** |
| **Out Date:** | **28-March-2018** | **Total Marks:** | **50** |
| **Due Date:** | **Tue 03-April-2018 *(start of class)*** | **Weight:** |  |
| **Section** | **CS** | **Page(s):** | **2** |
| **Assignment:** | **3 (Indexing)** |  |  |
|  |  | | | |

*Instructions:**•* Use proper assignment papers for solving your assignment questions. Assignment done on diary pages, register pages, rough pages will not be credited. *•* Do not copy the work of your peers. In case cheating is detected, then your case will be referred to DC

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

1. Block Size **B=4096 b**ytes and File Records **r=10m**illion
2. Block Size **B=4096 b**ytes and File Records **r=10b**illion
3. Block Size **B=8192 b**ytes and File Records **r=10m**illion
4. Block Size **B=8192 b**ytes and File Records **r=10b**illion

A block pointer is P = 6 bytes long and a record pointer is PR = 7 bytes long. A file has above records of fixed length*.* Each record has the following fields: NAME (30 bytes), SSN (9 bytes), DEPARTMENTCODE (9 bytes), ADDRESS (40 bytes), PHONE (9 bytes), BIRTHDATE (8 bytes), SEX(1 byte), JOBCODE (4 bytes), SALARY (4 bytes, real number). An additional byte is used as a deletion marker.

1. Calculate the record size R in bytes.
2. Calculate the blocking factor bfr and the number of file blocks b, assuming an unspanned organization.
3. Suppose that 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 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 SSN value-using the primary index.
4. 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.
5. Suppose that the file is not *ordered* by the nonkey field DEPARTMENTCODE and we want to construct a *secondary* index on DEPARTMENTCODE, with an extra level of indirection that stores record pointers. Assume there are 20000 distinct values of DEPARTMENTCODE and that the EMPLOYEE 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 DEPARTMENTCODE value, using the index.
6. Suppose that the file is *ordered* by the nonkey 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 20000 distinct values of DEPARTMENTCODE and that the EMPLOYEE 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 DEPARTMENTCODE value, using the clustering index (assume that multiple blocks in a cluster are contiguous)
7. 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; (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.
8. 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.

Solution:

Q1)

1. 30+9+9+40+9+8+1+4+4+1 = 115 byes
2. bfr = floor(B/R) = floor(4096 / 115) = floor(35.6) = 35 (not 36)

b = ceiling(r/bfr) = ceiling(10000000/35) = ceiling(285714.2) = 285715 (not 285716)

1. Ri = SsnSize + Pb = 9+6 = 15bytes

bfri = floor(B/Ri) = floor(4096/15) = floor(273.06) = 273

first level index records = b

r1 = 285715,

b1 = ceiling(r1/bfri) = ceiling(285715/273) = ceiling(1046.5) = 1047 (not 1046)

r2 = b1 = 1047

b2 = ceiling(r2/bfri) = ceiling(1047/273) = ceiling(3.8) = 4

r3 = b2 = 4

b3 = ceiling(r2/bfri) = ceiling(4/273) = ceiling(0.01) = 1

so x=3

bi = b1+b2+b3 = 1047+4+1 =1052 blocks

number of block accesses = x+1 = 3+1=4

1. Ri = SsnSize + Pb = 9+6 = 15bytes

bfri = floor(B/Ri) = floor(4096/15) = floor(273.06) = 273

b1 = ceiling(10000000/273) = ceiling(36630.03) = 36631

b2 = ceiling(36631/273) = ceiling(134.1) = 135

b3 = ceiling(135/273) = ceiling(0.4) = 1

so x=3

bi = b1+b2+b3 = 36631+135+1 = 36767 blocks

number of block accesses = x+1 = 3+1 = 4

1. Ri = DeptCodeSize + Pb = 9+6 = 15bytes

bfri = floor(B/Ri) = floor(4096/15) = floor(273.06) = 273

total different values for department code = 20000

r/differentValues = 10000000/20000 = 500

bytes for indirection = 7\*500 = 3500

it fits in one block.

so bindirection = 1\*20,000 = 20,000

b1 = ceiling(r1/bfri) = ceiling(20,000/273) = ceiling(73.2) = 74

b2 = ceiling(r2/bfri) = ceiling(74/273) = ceiling(0.2) = 1

so x=2

bi = b1+b2+bindirecton = 74+1+20000 = 20075

block accesses = x+1+average record for each value = 2+1+500 = 503

1. Ri = DeptCodeSize + Pb = 9+6 = 15bytes

bfri = floor(B/Ri) = floor(4096/15) = floor(273.06) = 273

b1 = ceiling(r1/bfri) = ceiling(20,000/273) = ceiling(73.2) = 74

b2 = ceiling(r2/bfri) = ceiling(74/273) = ceiling(0.2) = 1

so x=2

for bi,

bfr for cluster = floor(B/R) = floor(35.6) = 35

block cluster = ceiling(500/35) = ceiling(14.2) = 15

total block accesses = x+blockCluster = 2+15 = 17

1. (p\*P+((p-1) \* (Vssn))) < B

p < 273.6

p=273 not 274

(pleaf \* (Vssn + Pr)) + P < B

pleaf < 255.6

p=255 not 256

0.69\*255 = 175.9 approx = 176

b1 = ceiling(10000000/176) = ceiling(56818.1) = 56819

0.69\*273 = 188.3 approx = 189

b2 = (ceiling(b1/fo)) = ceiling(56819/189) = ceiling(300.63) = 301

b3 = ceiling(b2/fo) = ceiling(301/189) = ceiling(1.5) = 2

b4 = ceiling(b2/fo) = ceiling(2/189) = ceiling(0.01) = 1

total number of blocks = b1+b2+b3+b4 = 56819+301+2+1 = 57123

block accesses = x+1 = 4+1 = 5

1. (p\*P) + ((p-1) \* (Vssn + Pr)) < B

p < 186.9

p = 186

0.69\*186 = 128.3 = 129

b1 = ceiling(10000000/129) = ceiling(77519.3) = 77520

b2 = ceiling(77520/129) = ceiling(600.9) = 601

b3 = ceiling(601/129) = ceiling(4.6) = 5

b4 = ceiling(5/129) = ceiling(0.03) = 1

total blocks = b1+b2+b3+b4 = 77520+601+5+1 = 78127

total blocks accesses = x+1 = 4+1 = 5

you can solve Q2,3,4 in the same way.

**Q 5: *B+-tree and B-tree*(10 points)**

Consider a DBMS that has the following characteristics:

* 2KB fixed-size blocks
* 12-byte pointers
* 56-byte block headers

We want to build an index on a search key that is 8 bytes long. Calculate the maximum number of records we can index with

a) a 3-level B+ tree (including the root level)

b) a 3-level B tree (including the root level)

**Solution:**

a)

Let each node of a B+-tree contain at most n pointers and n-1 keys. (n\*12) + ((n-1)\*8) + 56 <= 2048. Therefore, n <= 100. The leaf level of a B+-tree can hold at most 99 \* 100 \* 100 record pointers. Therefore, the maximum number of records that can be indexed is 990,000.

b)

Let each inner node of a B-tree contain at most n index pointers, n-1 keys, and n-1 record pointers. ((n-1)\*8) + ((2n-1)\*12) + 56 <= 2048. Therefore, n <= 62. The first level of a B-tree can hold at most 61 record pointers. The second level can hold at most 62 \* 61 record pointers. The leaf level can hold at most 62 \* 62 \* 61 record pointers. Therefore, the maximum number of records that can be indexed is 61 + (62 \* 61) + (62 \* 62 \* 61) = 238,327.

NOTE: You can also assume that each leaf node of a B-tree can hold at most 99 record pointers. Then, the answer is 61 + (62 \* 61) + (62 \* 62 \* 99) = 384,399.