CS 6360.002 Assignment-5
Akhila Kancharana – AXK180025

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.

(70 points)

(a) Calculate the record size R in bytes:

A) Record Size= 30+10+10+30+10+10+1+4+4+1=110

(b) Calculate the blocking factor bfr and the number of file blocks b assuming an unspanned organization:

A) Blocking factor bfr = floor (B/R) = floor (512/110) = 4 records per block
Number of blocks needed for file = ceiling(r/bfr) = ceiling (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:

A) Index record size R_i = (V SSN + P) = (10 + 6) = 16 bytes
Index blocking factor bfr_i = fo = floor (B/R_i) = floor(512/16) = 32

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

A) Number of first-level index entries r_1 = number of file blocks b = 750 entries
Number of first-level index blocks b_1 = ceiling (r_1 / bfr_i) = ceiling(7500/32) = 24 blocks

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

Number of second-level index entries r_2 = number of first-level blocks b_1 = 24 entries
Number of second-level index blocks b_2 = ceiling (r 2 /bfr_i) = ceiling (24/32) = 1 blocks
Since the second level has only one block, it is the top index level.
Hence, the index has x = 2 levels

(iv) the total number of blocks required by the multi-level index; and

CS 6360.002 Assignment-5
Akhila Kancharana – AXK180025

a) Total number of blocks for the index $b_i = b_1 + b_2 = 24 + 1$

= 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.

A) Number of block accesses to search for a record = x + 1 = 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.

a)

i. Index record size R_i = (V SSN + P) = (10 + 6) = 16 bytes

Index blocking factor bfr_i = (fan-out) fo = floor (B/R_i) = floor (512/15) = 32 index records per block (This has not changed from part (c) above)

ii. Number of first-level index entries r_1 = number of file records r = 30000

Number of first-level index blocks b_1 = ceiling (r_1/bfr_i) = ceiling (3000/32)

= 94 blocks

iii. We can calculate the number of levels as follows:

Number of second-level index entries r_2 = number of first-level index blocks b_1 = 94 entries

Number of second-level index blocks b_2 = ceiling (r_2/bfr_i) = ceiling (94/32) = 3 blocks

Number of third-level index entries r_3 = number of second-level index blocks b_2 = 3 entries

Number of third-level index blocks b_3 = ceiling (r_3/bfr_i) = ceiling (3/32) = 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 $b_i = b_1 + b_2 + b_3 = 94 + 3 + 1 = 98$

v. Number of block accesses to search for a record = x + 1 = 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

CS 6360.002 Assignment-5
Akhila Kancharana – AXK180025

(i) the index blocking factor bfr i;

A)

Index record size R_i = (V DEPARTMENTCODE + P) = (10 + 6) = 16 bytes

Index blocking factor bfr_i = (fan-out) fo = floor (B/R_i) = floor (512/16) = 32 index records per block

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

A)

*There are 100 distinct values of DEPARTMENTCODE, so the average number of records for each value is (r/1000) = (3000/100) = 30
Since a record pointer size PR = 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 1 block. Hence, 100 blocks are needed for the level of indirection.

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

A)

Number of first-level index entries r_1 = number of distinct values of DEPARTMENTCODE = 100 entries

Number of first-level index blocks b_1 = ceiling (r_1/bfr_i) = ceiling (100/32) = 4 blocks

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

A)

We can calculate the number of levels as follows:
Number of second-level index entries r_2 = number of first-level index blocks b_1 = 4 entries
Number of second-level index blocks b_2 = ceiling (r_2/bfr_i) = ceiling (4/32) = 1
Hence, the index has x = 2 levels

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

A)

Total number of blocks for the index $b_i = b_1 + b_2$ = 4 + 1 = 5 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.

A)

Number of block accesses to search for a record = x + 1 = 2 + 1 = 3

CS 6360.002 Assignment-5
Akhila Kancharana – AXK180025

(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;
A)
Index record size R i = (V DEPARTMENTCODE + P) = (10 + 6) = 16 bytes
Index blocking factor bfr i = (fan-out) fo = floor (B/R i) = floor (512/16) = 32 index records per block

(ii) the number of first-level index entries and the number of first-level index blocks;
A)
Number of first-level index entries r 1 = number of distinct DEPARTMENTCODE values= 100 entries
Number of first-level index blocks b 1 = ceiling (r 1 /bfr i) = ceiling (100/32) = 4 blocks

(iii) the number of levels needed if we make it a multi-level index;
a)
We can calculate the number of levels as follows:
Number of second-level index entries r 2 = number of first-level index blocks b 1= 4 entries
Number of second-level index blocks b 2 = ceiling (r 2 /bfr i) = ceiling (4/32) = 1
Since the second level has one block, it is the top index level.
Hence, the index has x = 2 levels

(iv) the total number of blocks required by the multi-level index; and
a)
Total number of blocks for the index b i = b 1 + b 2 = 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).
a)
Number of block accesses to search for the first block in the cluster of blocks= x + 1 = 2 + 1 = 3
The 32 records are clustered in ceiling(30/bfr) = ceiling (30/4) = 8 blocks.
Hence, total block accesses needed on average to retrieve all the records with a given DEPARTMENTCODE = x + 8 = 2 + 8 = 10 block accesses

<u>(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</u>

(i) the orders p and p leaf of the B+ tree;

CS 6360.002 Assignment-5
Akhila Kancharana – AXK180025

A) For a B + -tree of order p, the following inequality must be satisfied for each internal tree node: (p * P) + ((p - 1) * V SSN) < B, or
(p * 7) + ((p - 1) * 10) < 512, which gives 17p < 522, so p=31
For leaf nodes, if record pointers are included in the leaf nodes, the following inequality must be satisfied: (p leaf * (V SSN +P R)) + P < B, or
(p leaf * (10+7)) + 7 < 512, which gives 17p leaf < 505, so p leaf =30

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

A) Assuming that nodes are 69% full on the average, the average number of key-values in a leaf node is 0.69*p leaf = 0.69*30 = 20.7. If we round this up for convenience, we get 21 key values (and 21 record pointers) per leaf node. Since the file has 3000 records and hence 3000 values of SSN, the number of leaf-level nodes (blocks) needed is b 1 = ceiling (3000/21) = 143 blocks

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

A)

We can calculate the number of levels as follows:

The average fan-out for the internal nodes (rounded up for convenience) is
fo = ceiling (0.69*p) = ceiling (0.69*31) = ceiling (21.39) = 22

number of second-level tree blocks b 2 = ceiling (b 1 /fo) = ceiling (143/22) = 7 blocks

number of third-level tree blocks b 3 = ceiling (b 2 /fo) = ceiling (7/22) = 1

Since the third level has only one block, the tree has x = 3 levels (counting the leaf level). Note: We could use the formula:
x = ceiling (log fo (b 1)) + 1 = ceiling (log 22 143) + 1 = 2 + 1 = 3 levels

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

A)

total number of blocks for the tree b i = b 1 + b 2 + b 3 = 143 + 7 + 1 = 151 blocks

(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.

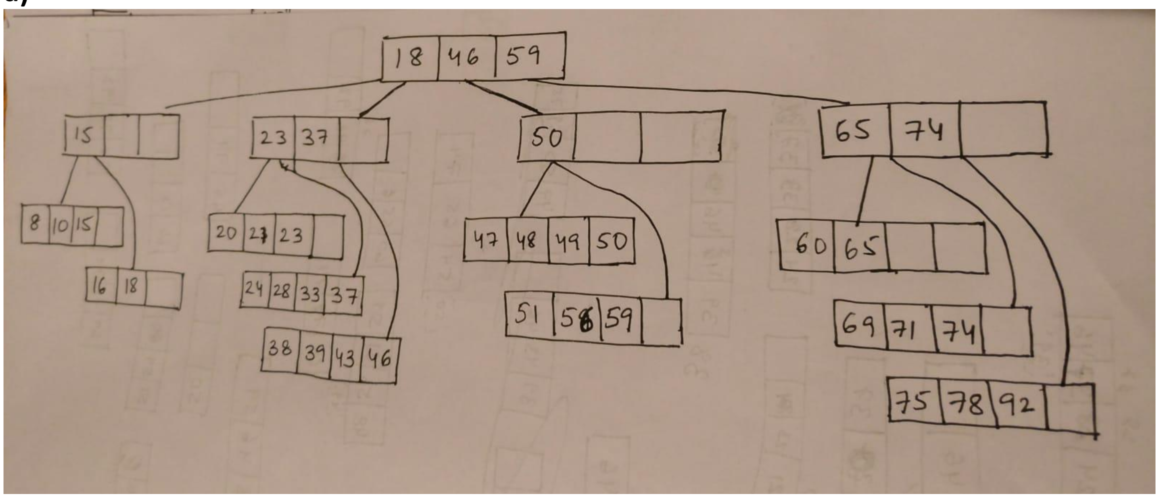
A)

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

CS 6360.002 Assignment-5
Akhila Kancharana – AXK180025

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)**

a)



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 in Chicago and the manager of the project's controlling department started after January 1, 2009.

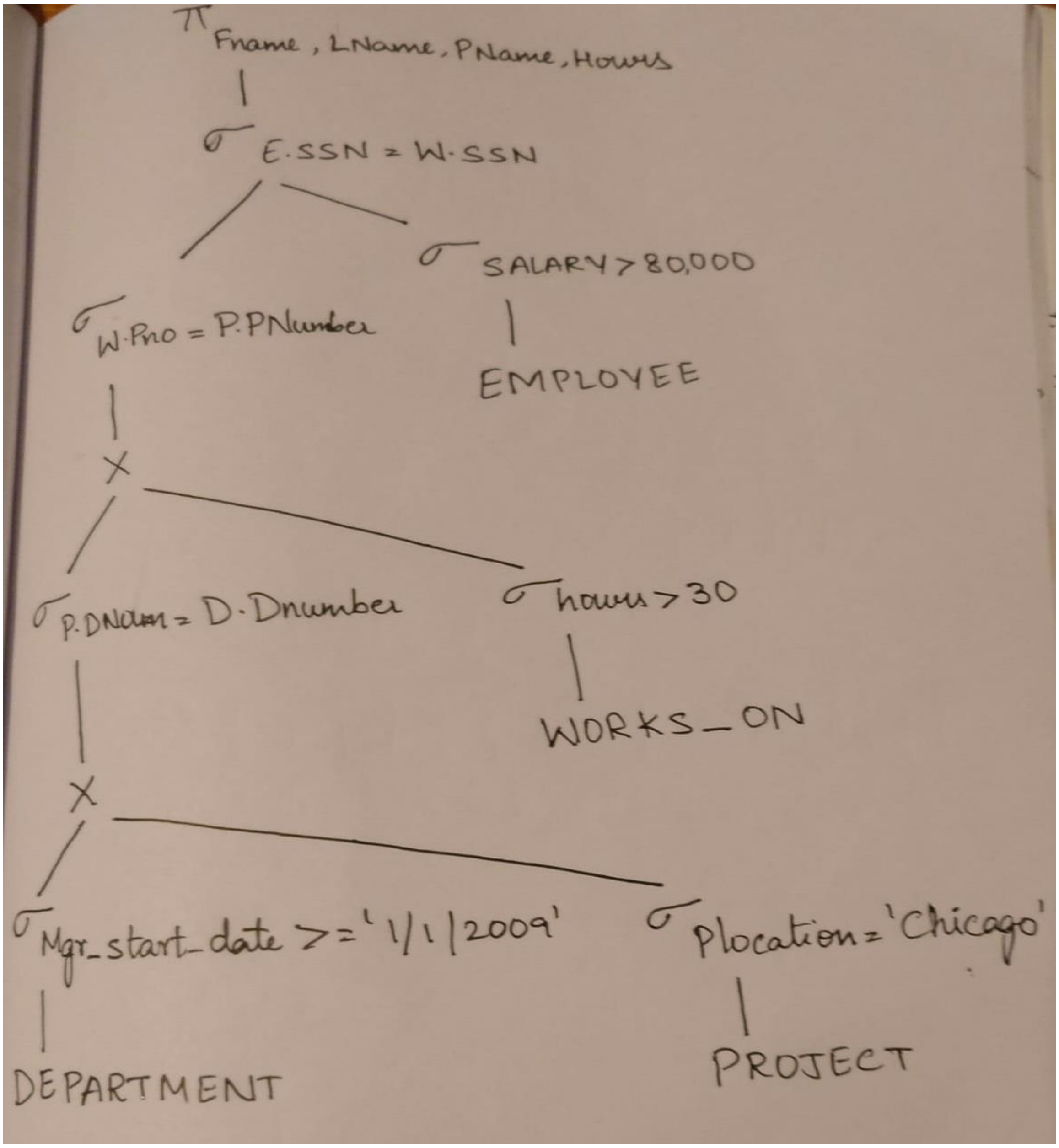
Show the final query tree. **(10 points)**

A) Optimized query:

```
Select E.Lname, E.Fname, P.Pname, W.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  P.Plocation = 'Chicago'
  and W. Hours > 30
  and  E.Salary > 80000
  and  D.Mgr_start_date >= '1/1/2009'
```

CS 6360.002 Assignment-5
Akhila Kancharana – AXK180025

Query Tree:



CS 6360.002 Assignment-5
Akhila Kancharana – AXK180025