CS3402 Database Systems

Homework 3

For Questions 1 – 4, suppose that you have formatted your disk with a block size of 2048 bytes and assume that we have 50,000 CAR records of fixed length. A block pointer is 5 bytes long (P=5), and a record pointer is 6 bytes long (Pr=6). Each CAR record has the following fields:
Model (20 bytes), Registration (8 bytes), Color (9 bytes), License (9 bytes), Location (40 bytes), Mileage (10 bytes), Price (8 bytes), Year (4 bytes), Manufacturer (16 bytes), and Remarks (200 bytes).

Justify your answers *and state any assumptions that you make!*

1. Calculate the blocking factor (bfr) and the number of file blocks needed to store the CAR records. Assume that records are stored unspanned. How much space remains unused per block? **[10 marks]**

   Solution:
   The record size R = 20 + 8 + 9 + 9 + 40 + 10 + 8 + 4 + 16 + 200 = 324.
   To calculate the blocking factor, we get bfr = floor(B / R) = 6 records/block.
   The number of blocks is I = ceiling(50,000 / bfr) = 8334 blocks.
   Therefore, the unused space per block is B – (R * bfr) = 104 bytes.

2. Suppose that the file is ordered by the key field Registration and we want to construct a **primary index** on Registration.
   a. Calculate the index blocking factor bfr_i. **[5 marks]**
   Solution:
   Each index entry has size Ri = size(registration) + P = 8 + 5 = 13 bytes.
   The index blocking factor is bfri = floor(B / Ri) = 157 entries / block.

   b. Assume that the primary index is a single-level index. Calculate the number of index entries and the number of index blocks. **[5 marks]**

   Solution:
   We need one index entry per file block, so to calculate the number of index and we already calculate the number of blocks needed to store all 50,000 file records in Q1, which is I=8334.
   Thus, the number of index blocks: Bi = ceil(I / bfri) = 54 blocks.

   c. Now suppose that we want to make the primary index a multilevel index. How many levels are needed and what is the total number of blocks required by the multilevel index? **[10 marks]**

Solution:
The fan-out for the multilevel index is the same as the index blocking factor (bfri) which is 157 (see above).
The number of 1st level blocks L1 is already calculated in 2b, and so we have L1=54.
The number of 2nd level blocks is L2 = ceil(L1 / fo) = 1.
The total number of index blocks is L1 + L2 = 54 + 1 = 55 blocks.

d. Consider the multilevel index from question 2c. What is the number of block accesses needed to search for and retrieve a record from the file given its Registration value? **[5 marks]**

Solution:
This is given as the number of index levels + 1, which, according to c) is 2+1=3.

3. Suppose that we want to construct a B^+-tree index on the key field License.
   a. Calculate the order p of internal nodes and the order p_{leaf} of leaf nodes. **[10 marks]**

   Solution:
   Field License has size V=9 bytes.
   Therefore, we need to solve
   $(p*P) + ((p-1)*V) \leq B$
   $(p*5) + ((p-1)*9) \leq 1024$, and therefore we can choose
   p = 146.

   For the order pl of leaf nodes, we need to solve
   $(pl * (Pr + V)) \leq B$
   $(pl * (6 + 9)) + 5 \leq 1024$, and therefore we can choose
   pl = 136.

   b. Compute the total number of blocks needed if (internal and leaf) nodes are approximately 70% full. **[10 marks]**

   Solution:
   On average, each internal node will contain 146 * 0.7 = 102 pointers
   On average, each leaf node will contain 136 * 0.7 = 95 pointers.
   Therefore, the average number of leaf blocks is ceil(50000 / 95) = 526 blocks. Let k be the leaf level.
   Since each internal node has 102 pointers, the number of internal blocks on the level k-1 is ceil(526 / 102) = 6.
   Finally, we have one root block on level k-2.
   Summing up, the total number of blocks used is 1 + 6 + 526 = 533.

4. Suppose that several (single-level) secondary indexes exist, namely on the nonkey fields Color, Price, and Year.
    a. Describe how you would use these indexes to *efficiently* search for and retrieve records satisfying the following selection condition:
        (Color='blue' OR Price = 25,000 OR Year = 2018).
        (Note: linear search is not an efficient way!)

[15 marks]

Solution:
1. We use binary search on the secondary index on Color to find all record pointers matching the search condition Color='blue'.
2. We use binary search on the secondary index on Price to find all records matching 'Price=25,000'.
3. We use binary search on the secondary index on Year to find all records matching 'Year=2018'.
4. Finally, we take the union of these 3 sets of record pointers, which will be the set of record pointers satisfying the query condition. Retrieving the corresponding records yields the result.
The method is efficient, because we use the secondary indexes to find these sets (using binary search) instead of performing a linear search over the file.

    b. Suppose that, among the 50,000 CAR records, we have 200 records where Color='blue', 300 records where Price = 25,000, and 200 records where Year=2018.
        What is the *maximum* number of block accesses that your method requires to retrieve the record pointers in the worst case?
        (Note that this does <u>not</u> include accessing the disk blocks where the data records are stored.) **[10 marks]**

Solution:
Since the secondary index is on a non-ordering non-key field, the index uses an additional layer of indirection (see slide #16 in the lecture slides). Thus, binary search on the secondary index for Color requires log_2(8334) = 14 block accesses for finding the block that holds the record pointers for Color='blue'. Note that all such record pointers fit into a single block because a block can hold up to 2048 / 6 = 341 record pointers, and we know there are only 200 records that satisfy Color='blue'.
The same is true when searching the other two indexes. Thus, overall we need 3*(14+1) = 45 block accesses until we have found all the record pointers.

5. Consider three transactions T_1, T_2, and T_3, and the schedules S_1 and S_2 given below. Draw the serialization graphs for S_1 and S_2, and state whether each schedule is serializable or not. If a schedule is serializable, write down all equivalent serial schedule(s):

   T_1: r_1 (X); r_1 (Z); w_1 (X);

   T_2: r_2 (Z); r_2 (Y); w_2 (Z); w_2 (Y);

   T_3: r_3 (X); r_3 (Y); w_3 (Y);

   a. S_1: r_1 (X); r_2 (Z); r_1 (Z); r_3 (X); r_3 (Y); w_1 (X); w_3 (Y); r_2 (Y); w_2 (Z); w_2 (Y);

   [10 marks]

   b. S_2: r_1 (X); r_2 (Z); r_3 (X); r_1 (Z); r_2 (Y); r_3 (Y); w_1 (X); w_2 (Z); w_3 (Y); w_2 (Y);

   [10 marks]

   Solution:

   a) The serialization graph contains the following directed edges:
   T1 → T2, T3 → T1, T3 → T2. It is serializable because the graph doesn't contain a cycle. The only equivalent serial schedule is T3 → T1 → T2.

   b) The serialization graph contains the following directed edges:
   T1 →T2, T2 → T3, T3 → T2, T3 → T1. The schedule isn't serializable because there is a cycle.