

NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING
SEMESTER 2 (2017/2018) MID-TERM EXAMINATION FOR

CS3223: DATABASE SYSTEMS IMPLEMENTATION

March 2018

Time Allowed: 60 minutes

NAME: _____

MATRIC NUMBER: _____

This paper contains 9 questions. You should answer ALL questions. The total marks for this paper is 20 marks.

1. (2 marks) Consider a table with 50 attributes – 20 of these are mandatory (i.e., information must always be captured) while the other 30 are optional. Moreover, the values for all attributes are a fixed size of 10 bytes. Based on historical data, on average, $k\%$ of these optional fields will be captured. Now, consider the following two possible record storage formats:
- (a) Fixed-length record format that reserves space for every attribute.
 - (b) Variable-length record format where all captured fields are tagged. Each tag is 1 byte. The tag contains information that indicates an attribute is present. Note that variable-length record format has no concept of mandatory or optional fields; all fields are deemed optional.

For what range of k values is the fixed format option superior?

Ans:

The expected size for (a) = $50 \times 10 = 500$ bytes

For (b), it is $\# \text{mandatory attributes} \times (10 + 1) + \# \text{optional attribute} \times (10 + 1) \times k/100$

$$= 220 + 3.3k$$

So, we have $500 < 220 + 3.3k \rightarrow k > 84.8$

So, fixed record format is good if more than 85% of the optional attributes are captured.

2. (2 marks) To modify a block on disk, we must read it into main memory, perform the modification, and write it back. Assume that the modification in main memory takes less time than it does for the disk to rotate one block, and that the disk controller postpones other requests for disk access until the block is ready to be written back to the disk.

Consider a disk with the following characteristics:

- (a) Tracks hold 2000 sectors of 1024 bytes each
- (b) 10% of each track is used for gaps
- (c) The disk rotates at 6000 rpm
- (d) Average seek time is 10ms
- (e) A block consists of 8 sectors

What is the time to modify a block (assume the initial read is random)?

Ans: average seek time + average rotational latency + full rotational delay + transfer time

Average seek time = 10 ms;

Rotational delay = 10 ms;

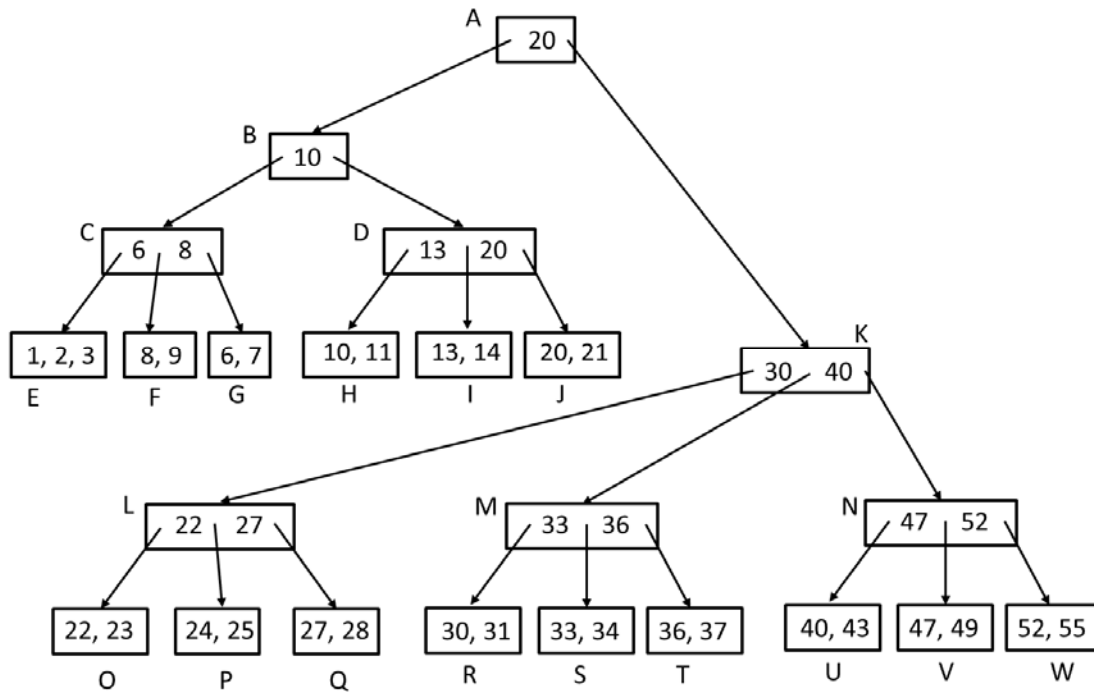
Average rotational delay = 5ms

Read one sector and gap = $1/2000 \times 10 = 0.005$ ms

Read one sector without gap = $9/10 \times 0.005 = 0.0045$ ms

So, ans = $10 + 5 + 10 + 7 \times 0.005 + 0.0045 = 25.0395$ ms

3. (4 marks) Consider the following B⁺-tree structure. Find any/all violations of the B⁺-tree properties. For each bad node, indicate the node (labeled by alphabets A, B, C ... W), and give a brief explanation of each error. Assume the order of the tree is 2, i.e., there are at most 4 key-value pairs and 5 pointers. Complete your answer in the table below (Note that it does not mean that there are as many violations as the number of rows in the table; you can leave some of the rows empty if you have identified fewer).



Node ID	What's wrong with the node?
B	below min key requirement
D	key 20 duplicated from root (20 should be on the right side of the tree) (can also be corrected by making root = 22)
F & G	Nodes F and G are swapped
J	Both keys are not less than root (can also be corrected by making root = 22)
O	key 22 not less than parent key 22
L	Key 22 should be 24

4. (2 marks) In the basic linear hashing that we have discussed in the lecture, we split the buckets in a round-robin fashion one at a time. In fact, the bucket that is split need not be the bucket that overflows (assuming we use overflow as a splitting criterion). Now, consider the following variant: if the overflowed bucket is before the “next” pointer, then we follow the basic algorithm (there is no need to split as the bucket has already been split); however, if the overflowed bucket is at or after the “next” pointer, we split all buckets from the bucket pointed to by the “next” pointer to the overflowed bucket, and move the “next” pointer to the bucket after the overflowed bucket. What are the pros and cons of this variant in comparison to the basic scheme.

Ans: pros: get to split the overflow buckets asap. Hopefully, overflow chains will in general be shorter, and query performance will be better.

Cons: the splitting time is expensive – it has to rehash all buckets that are split. Storage may also be higher.

5. (2 marks) Suppose you are required to create a linear hash index for a table. You decide to initialize k empty primary buckets (the capacity of the k buckets is larger than the number of records). Then you insert the tuples into the buckets accordingly (as we cover in the lecture). You want to minimize the number of split buckets to keep the hash table as small as possible. Which of the following three insertion orders is expected to perform best: (a) Insert the records belonging to the first bucket first, then insert records belonging to the second bucket next, and so on; (b) Insert one record at a time in a round-robin fashion, i.e., one record to bucket one, one record to bucket two and so on; after one round, insert another record to bucket one, and so on. In this way, the buckets are hopefully balanced; (c) Insert the records belonging to the last bucket first, then records belonging to the second last bucket, then third last bucket and so on? Justify your answer.

Ans: (c) is best answer as inserting records from the end would lead to premature splitting of pages in the front. This means that when the front blocks are overloaded, they have already been split.

6. (2 marks) Consider the following sequence of numbers:

20, 41, 39, 7, 49, 12, 10, 23, 89, 48, 76, 55, 50, 77, 33, 28, 11, 46, 40, 29

Suppose each page has only 1 record. Let the number of buffer pages be 4 (we can ignore output buffers). Suppose we use replacement selection to generate sorted runs. List the sorted runs below.

7, 20, 39, 41, 49, 89

10, 12, 23, 48, 50, 55, 76, 77

11, 28, 29, 33, 40, 46

7. (2 marks) Consider a relation R using B buffer pages. Describe the various ways in which the B pages can be allocated in the merging phase for sorting.

Ans: 1 for each input buffer, 1 for output buffer, the remainder are distributed between inputs and outputs.

8. (2 marks) Consider the join of two tables R and S. Let $|R|$ ($|S|$) denote the size of R (S) in terms of number of pages. Let B be the buffer size available to perform the join. Suppose the join result is immediately returned to the user (i.e., there is no need to buffer the output). For the following two join methods (using the basic methods that we learn in our lecture), what is the best/earliest possible time (in terms of I/O cost) that the first output tuple can be returned?

(a) Nested Loops Join

Since answer is returned to user immediately, no need to buffer output.

Ans: 1 buffer for R, 1 buffer for S. Best scenario. But, I also accept $1 + B - 1$.

(b) Hash-Join. You can assume data are uniformly distributed, and memory is sufficient (i.e., $B > \sqrt{|R|}$ or $B > \sqrt{|S|}$). In other words, the scheme we learned in the lecture works.

Ans: $2|R| + 2|S| + R_1 + 1$ where R_1 is first partition of R, and $R_1 = |R|/(B-2)$

R_1 can take on different values as long as $R_1 < B-2$

9. (2 marks) Suppose we want to compute $R \text{ JOIN}_{R.a=S.b} S$. Let $|R| = 100$ pages, $\|R\| = 1000$ tuples. Let $|S| = 100$ pages, $\|S\| = 5000$ tuples. Suppose there is an unclustered B^+ -tree index on $S.b$. As covered in the lecture, the index is in format 2, i.e., it stores (key, pointer)-pairs. Moreover, let the height of the B^+ -tree be 3 (root node is at level 1, and leaf nodes are at level 3). Consider the nested-index join algorithm. Suppose 5% of R has one and only one matching S record while the rest do not have a match. Let every access (be it index or data pages) incurs an I/O, what is the I/O cost to perform the join using the index?

Ans: $|R|$ (cost to scan R) + $1000 \cdot 0.95 \cdot 3$ (95% of the records end at the leaf level – records not found, no need to retrieve the data pages) + $1000 \cdot 0.05 \cdot 4$ (5% need data access).

----- the end -----