CS3223: Database Management Systems

Tutorial 1

(Week 3, Jan 2022)

1. Consider the Megatron 747 disk with the following characteristics:
   - There are four platters providing eight surfaces, each surface has 2^{13} = 8192 tracks, each track has 2^8 = 256 sectors, and each sector has 2^9 = 512 bytes.
   - The disk rotates at 3840 rpm.
   - To move the head assembly between cylinders takes 1 ms to start and stop, plus 1 ms for every 500 cylinders traveled. Thus, the heads move one track in 1.002 ms, and move from the innermost to the outermost track, a distance of 8192 tracks, in about 17.4 ms.
   - 10% of the space (on top of the usable space) are used for gaps, i.e., between 2 sectors, there is a gap that is not used to store data.

   Answer the following questions.

   a) What is the capacity of the disk?

   b) What is the minimum and maximum time it takes to read a 4096-byte block (8 consecutive sectors) from the disk?

   c) Suppose that we know that the last I/O request accessed cylinder 1000. What is the expected (average) block access time for the next I/O request on the disk?

2. Consider a B+-tree of order 2 for this problem, that is initially empty. Show what the tree looks like at this point after each of the following actions:
   - You insert the key 10 (and a pointer to this record) into the tree.
   - Next you insert keys 2, 5, 14, and 17 (in this order) into the tree.
   - Next you insert keys 1, 3, 4, 6, 7, 8, 9, 11, 12, 13, 15 and 16 (in this order).
   - Delete keys 1, 2 and 3.

3. To manage the buffer pool more efficiently, a database buffer manager might distinguish between different types of disk pages that may have different access patterns. In particular, it has been pointed out that index pages (e.g., B+-tree access) often have a different pattern of access compared to data pages. Consider the example index shown in the figure below:

```
A -->
  --> B -->
        --> C -->
              --> D --> E
                    --> F
                    --> G
                    --> H
```

   Node A is the root node of a B+-tree index, B, C, and D are index nodes, and E, F, G, H leaf nodes that contain the clustered data records. Consider the following logical page reference string corresponding to an index traversal

A_1, B_2, C_3, D_4, E_5, D_6, F_7, D_8, G_9, D_{10}, H_{11}, D_{12}, C_{13}, …

The subscript in each page number denotes the reference sequence number.
Assume that the allocated buffer has 5 pages. Contrast the use of an LRU replacement policy and an Optimal replacement policy (i.e., one which victimizes the page with the longest expected time until its next reference.)

Reference	LRU strategy Least $\rightarrow$ Most frequently used	Optimal Strategy
5	A B C D E	A B C D E
6	A B C E D	A B C D E
7	B C E D F/A	A B C D F/E
8	B C E F D	A B C D F
9	C E F D G / B	A B C D G / F
10	C E F G D	A B C D G
11	E F G D H / C	A B C D H / G
12	E F G H D	A B C D H
13	F G H D C / E	A N C D H

(A buffer entry I/J means that a page replacement has occured with page I replacing page J.)
Complete the above table. Why is LRU not an effective replacement algorithm here. Can you suggest a better strategy than LRU?

LRU is suboptimal because it choses to replace useful pages like B and C which are needed later. This can be seen in line 13 in the above table:
- under the LRU, only one (C) out of fie buffer pages in memory is useful. Under the ideal situation, we have pages A, B, and C which are much more likely to be referenced in the future.

A more optimal strategy here is to choose pages for replacement based on the corresponding level of the page in the B-tree.

Note that the optimal strategy here is NOT MRU since it would have removed D in Step 7 above.

The key objective of this question is to understand that no single buffer replacement strategies work best. Some systems implement different strategies for different types of data: one replacement strategy for index structures; another for data.

Another solution could be to replace based on the level of the B+ tree by allocating buffer for leaf nodes and non-leaf nodes