EE477 Database and Big Data Systems, Spring 2021
HW4

Due date: 6/4/2021 (11:59pm)

Submission instructions: Use KAIST KLMS to submit your homeworks. Your submission should be one gzipped tar file whose name is `YourStudentID_hw4.tar.gz`. For example, if your student ID is 20211234, and it is for homework #4, please name the file as `20211234_hw4.tar.gz`. You can also use these extensions: tar, gz, zip, tar.zip. Do not use other options not mentioned here.

Your zip file should contain two things: one PDF file (`hw4.pdf`) and an Ethics Oath pdf file. Do not include Korean letters in any file name or directory name when you submit.

Submitting writeup: Prepare answers to the homework questions into a single PDF file. You can use the following template. Please write as succinctly as possible.

Ethics Oath: For every homework submission, please fill out and submit the **PDF** version of this document that pledges your honor that you did not violate any ethics rules required by this course and KAIST. You can either scan a printed version into a PDF file or make the Word document into a PDF file after filling it out. Please sign on the document and submit it along with your other files.

Discussions with other people are permitted and encouraged. However, when the time comes to write your solution, such discussions (except with course staff members) are no longer appropriate: you must write down your own solutions independently. If you received any help, you must specify on the top of your written homework any individuals from whom you received help, and the nature of the help that you received. *Do not, under any circumstances, copy another person's solution.* We check all submissions for plagiarism and take any violations seriously.

1 Hardware

The 5 minute rule states that a page referenced every five minutes should be kept in memory rather than being read from disk each time.

Suppose a page is accessed every X seconds. We compute the cost to keep this page on disk as $CD = \$D/XI$ where

- $\$D$ is the cost of the disk unit and
- I is the number of IOs that unit can perform per second.

That is, in X seconds, the disk can do XI I/O's, so the cost of periodically reading the page is $\$D$ divided by XI.

Now the cost to keep the page in memory can be computed as $CM = \$M/P$ where

- $\$M$ is the cost of 1MB of memory and
- P is the number of pages in 1MB of memory.

If CD is smaller than CM, we keep the page on disk. Otherwise, we keep the page in memory. The break even point is thus $X = \frac{\$DP}{I\$M}$. Interestingly, X has been about 5 minutes for randomly-accessed pages in 1985 [1] and 1997 [2].

In this problem, figure out whether the 5 minute rule still holds today for randomly-accessed pages. In particular, fill in the new values for P, I, $\$D$, $\$M$, and X for year 2021 and specify how you obtained the values (e.g., which memory and disk products did you use?). This problem is open-ended, and your numeric answers are not as important as the arguments you use and the issues you uncover in your analysis.

2 B+ tree

Consider an index organized as a B+ tree. The leaf nodes contain pointers to a total of N records, and each block that makes up the index has m pointers. We wish to choose the value of m that will minimize search times on a particular disk device with the following characteristics:

- For the disk that will hold the index, the time to read a given block into memory can be approximated by $(90 + .07 \times m)$ milliseconds. (The 90 milliseconds represent the seek and latency components of the read, the $.07 \times m$ milliseconds is the transfer time. That is, as m becomes larger, the larger the block will be and the more time it will take to read it into memory.)

[1] Jim Gray & Franco Putzolu, "The 5 minute rule for trading memory for disc accesses and the 10 byte rule for trading memory for CPU time," ACM SIGMOD 1985

[2] Goetz Graefe & Jim Gray, "The five-minute rule ten years later, and other computer storage rules of thumb," ACM SIGMOD Record 1997

- Once the block is in memory a binary search is used to find the correct pointer. So the time to process a block in main memory is $a + b \log_2 m$ milliseconds, for some constants a and b.
- The main memory time constant a is much smaller than the disk seek and latency time of 90 milliseconds.
- The index is full, so that the number of blocks that must be examined per search is $\log_m N$.

Answer the following:

(a) What value of m minimizes the time to search for a given record? An approximate answer is OK. The value you obtain should be independent of b. (HINT: If you come up with an equation which is hard to solve algebraically, try plugging in values to locate the root of the equation.)

(b) What happens as the seek and latency constant (90ms) decreases? For instance, if this constant is cut in half, how does the optimum m value change?

3 Extensible Hashing

Consider an extensible hash structure where buckets can hold up to three records. Initially the structure is empty. The hashed key values are as follows (in binary):

h(Ashley)	=	[00010]
h(Brian)	=	[00101]
h(Chris)	=	[00110]
h(Daniel)	=	[01010]
h(Ethel)	=	[01101]
h(Frank)	=	[10010]
h(George)	=	[10101]
h(Harold)	=	[10110]
h(Jeff)	=	[11101]
h(Karen)	=	[11111]

(a) We insert records in the following order: Ashley, Karen, Brian, Jeff, Chris, Harold, Ethel, George, Frank, Daniel Show the structure after all these records have been inserted.

(b) Suppose, instead we insert records in the following order: Ashley, Brian, Chris, Daniel, Ethel, Frank, George, Harold, Jeff, Karen Show the structure after all these records have been inserted.

(c) Does the final structure depend on the order in which records are inserted? Explain.

(d) Suppose now, we decided to use the opposite bits (low order bits) of the hash but do not change anything else in the algorithm. Show the structure after the records are inserted in the same order as 3(a).

(e) What is the problem that occurred (if any) because we used the low order bits instead of the high order ones? Is this problem serious?

4 Query Processing

Consider the following schema, for an online bookstore:

Cust (CustID, Name, Address, State, Zip)
Book (BookID, Title, Author, Price, Category)
Order (OrderID, CustID, BookID, ShipDate)

This schema represents customers (Cust) and books (Book). When a customer buys a book, a tuple is entered into the Order table. Assume you have an index over Book.Author, but there are no other indexes.

Indicate the number of I/Os required to perform the following operations. You should assume that each operation uses memory efficiently. You can ignore final output I/O cost. If required, explain briefly. Use the following statistics:

- B(Order) = 7,000 blocks
- B(Cust) = 1,000 blocks
- B(Book) = 100 blocks

(a) Selection of Price < 10 over Book. Memory = 10 blocks.

(b) One pass join of Order and Cust. Memory =1001 blocks.

(c) Nested loop join of Order and Cust. Cust is the outer relation. Memory=2 blocks.

(d) Nested loop join of Order and Cust. Cust is the outer relation. Memory=101 blocks.

(e) Nested loop join of Order and Cust. Order is the outer relation. Memory=101 blocks.

(f) Nested loop join of Order and Book. Book is the outer relation. Memory=11 blocks.

(g) Hash join of Order and Book. Memory = 11 blocks. (You may assume that you have a hash function over Order.BookID and Book.BookID that distributes the tuples evenly into equally sized buckets.)

5 Crash Recovery

Consider the following transaction log from the start of the run of a database system that is capable of running undo/redo logging with checkpointing:

1) ⟨START T1⟩
2) ⟨T1, A, 30, 10⟩
3) ⟨START T2⟩
4) ⟨T1, B, 140, 10⟩
5) ⟨T1, A, 80, 30⟩
6) ⟨T2, C, 30, 10⟩
7) ⟨T2, D, 40, 10⟩
8) ⟨COMMIT T1⟩
9) ⟨START T3⟩
10) ⟨T3, E, 70, 10⟩
11) ⟨T2, D, 50, 40⟩
12) ⟨START CKPT (T2,T3)⟩
13) ⟨T2, C, 70, 30⟩
14) ⟨COMMIT T2⟩
15) ⟨START T4⟩
16) ⟨T4, F, 100, 10⟩
17) ⟨T4, G, 100, 10⟩
18) ⟨COMMIT T3⟩
19) ⟨T4, F, 120, 100⟩
20) ⟨END CKPT⟩
21) ⟨T4, F, 130, 120⟩
22) ⟨COMMIT T4⟩

Assume the log entries are in the format ⟨Tid, Variable, New value, Old value⟩

What are the values of data items A, B, C, D, E, F, and G on disk after recovery:

(a) if the system crashes just before line 9 is written to disk?

(b) if the system crashes just before line 13 is written to disk?

(c) if the system crashes just before line 18 is written to disk?

(d) if the system crashes just after line 19 is written to disk?

(e) if the system crashes just before line 22 is written to disk?

(f) if the system crashes just after line 22 is written to disk?

6 Concurrency Control

Below are two transactions, described in terms of their effect on two database elements A and B, which we may assume are integers.

T1: READ(A, t); t := t+10; WRITE(A, t); READ(B, t); t := t*3; WRITE(B, t);
T2: READ(B, s); s := s*2; WRITE(B, s); READ(A, s); s := s+7; WRITE(A, s);

We assume that, whatever consistency constraints there are on the database, these transactions preserve them in isolation. Note that A = B is not the consistency constraint.

(a) It turns out that both serial orders have the same effect on the database; that is, (T1, T2) and (T2, T1) are equivalent. Demonstrate this fact by showing the effect of the two transactions on an arbitrary initial database state.

(b) Give one example each of a serializable schedule and a nonserializable schedule of the 12 actions above.

(c) How many serializable schedules of the 12 actions are there?