EECS-4411M: TEST #1

"The Physical Database"

Electrical Engineering & Computer Science

Lassonde School of Engineering

York University

Family Name: _____

Given Name: _____

Student#: _____

EECS Account: _____

Instructor: Parke Godfrey
Exam Duration: 75 minutes
Term: Winter 2017

Instructions

- **rules**
  - The test is closed-note, closed-book. Use of a calculator is permitted.
- **answers**
  - Should you feel a question needs an assumption to be able to answer it, write the assumptions you need along with your answer.
  - If you need more room to write an answer, indicate where you are continuing the answer.
  - For multiple choice questions, choose *one* best answer for each of the following. There is no negative penalty for a wrong answer.
- **notation & assumptions**
  - For questions about indexes, assume that the indexes are *dense*.
  - For physical storage of records, assume row store.
- **points**
  - The number of points a given question is worth is marked. (It is worth one point, if not marked.)
  - There are five major parts worth 10 points each, for 50 points in total.

MARKING BOX	
1.	/10
2.	/10
3.	/10
4.	/10
5.	/10
Total	/50

1. [10pt] **Hardware, I/O, Pages, & Layout.** *But I'm a software person!* SHORT ANSWER

---

a. [2pt] You are a database administrator (DBA). You recently set up a new database system on a server with RAID. When you ran some performance measures, you find that random-access page *writes* to disk are significantly more expensive than random-access page *reads* from disk.

Provide a rational hypothesis why this might be the case.

---

b. [3pt] We generally do not want to change the *slot#* of a record. This can make managing record placement on *pages* more difficult if we handle *variable-length* records.

Describe / draw a page layout strategy for records that can accommodate insertions and deletions of records on the page without changing existing records' slot#'s.

SQL must support NULL values. Thus, so must the database system.

Assume that you only have to support *fixed-format* records.

---

c. [2pt] If a database will have lots of NULL values, why would having support for *variable-length* records in the system be preferable to having only support for *fixed-length* records?

---

d. [3pt] Briefly describe a way to represent variable-length records that may contain NULLs.

2. [10pt] **Buffer Pool.** *Learning to swim.* EXERCISE

a. [2pt] LRU (*least recently used*) is known to be generally a good buffer-pool replacement strategy in support of most SQL operations.
Why?

b. [3pt] Spell out the *steps* that the buffer-pool manager needs to make to handle a *pin* call; e.g., pin(1729). Assume the replacement policy is LRU.

c. [3pt] A *design choice* for the physical database is *how* we will *address* records to locate them. This identifier is called the *record ID* (RID). Our choice is for RID's to be *physical* or to be *indirect*.

What is the difference between these?

---

d. [2pt] When building a database system from scratch, instead of having to implement a buffer-pool manager for our system, it would be tempting to use library calls to the operating system (OS) instead. The OS would handle paging in and out our database pages on disk to and from main memory. (Thus, this would be leveraging the OS's *virtual memory*.)

Is this a good idea? Briefly, why or why not?

3. [10pt] **General.** *Jan, ken, pon!* MULTIPLE CHOICE

---

a. [1pt] B+ tree indexes reduce I/O cost by
A. providing binary search.
B. storing data records with the search keys.
C. use of overflow pages.
D. keeping the data records sorted.
E. increasing fan-out of the index pages.

---

b. [1pt] Which of the following is *false*?
A. The technology trend is that the ratio of CPU to disk I/O speed is growing over time.
B. Page size is determined by the hardware.
C. Sequential reads and writes are important to a database system's performance.
D. CPU time usually dominates I/O time in database operations.
E. Many records fit on a page, on average.

---

c. [1pt] How many distinct search keys exist for table **T** with six attributes A, B, C, D, E, and F? (Assume *ascending* in all cases.)
A. 1
B. 6
C. 21
D. 63
E. 720
F. 1956

---

d. [1pt] An operation is considered to be *I/O bound* if it
A. runs as efficiently as possible.
B. stalls frequently waiting on I/O calls to complete.
C. uses no I/O resources.
D. uses no CPU resources.
E. runs entirely on disk, not in main memory.

---

e. [1pt] In database systems, hash indexes have the *advantage* over tree indexes in that
A. they can be used for all queries that tree indexes can be, but not vice versa.
B. they use no disk space.
C. they can be used for *unique* search keys; that is, a search key that is also a (logical) candidate key.
D. they typically use less I/O for equality searches.
E. they can accommodate composite search keys, while tree indexes cannot.

---

f. [1pt] Assume a B+ tree in which up to four index records (with five pointers) *or* four data records fit per page. Assume the index is primary / direct; the data records themselves are stored in the index. And assume the tree is storing millions of records.

In *worst* case, a B+ tree index may take up (approximately) how much more disk space than the data it stores?

A. $1\frac{1}{2}$ times.
B. 2 times.
C. 3 times.
D. 16 times.
E. *Indefinitely more space.*

---

g. [1pt] *Double buffering* is a technique that

A. doubles *virtually* the space in the buffer pool.
B. involves reading *twice* each page fetched into the buffer pool.
C. addresses I/O boundedness.
D. addresses CPU boundedness.
E. speeds up significantly individual I/O operations.

---

h. [1pt] People often use the phrase "Moore's Law" to refer to the periodic doubling of a hardware attribute.

All the following "Moore's Laws" *but* which have we observed in the industry over time?

A. The number of transistors on a integrated circuit.
B. CPU clock speed (until recently).
C. Network bandwidth speed.
D. Speed of disk I/O operations.
E. Number of bytes on commodity hard disks.

---

i. [1pt] Currently, SSDs compared with HDDs (*hard disk drives*) are

I. faster
II. more reliable
III. less expensive per byte

A. *None of those.*
B. Just **I**.
C. Just **I** and **II**.
D. Just **I** and **III**.
E. *All of those.*

---

j. [1pt] For database systems, RAID is used to

A. speed up transaction management, allowing more operations to run concurrently.
B. store only data for which we can tolerate loss such as indexes, but not data for which we cannot tolerate loss, such as tables.
C. provide variable-sized physical pages.
D. back up the system's databases reliably.
E. survive better data loss due to hard-disk failure.

---

4. [10pt] **Index Mechanics.** *It's the carburetor, of course.* EXERCISE

   Assume the search-key rule to go *left* if '<' and to go *right* if '≥'.

---

a. [2pt] You conduct some disk forensics and you discover a B+ tree structure with the data as in Figure 1.

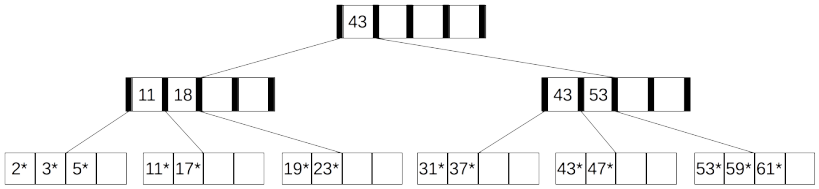

Figure 1: Recovered forensics B+ tree.

You suspect there is something wrong with this B+ tree. What is wrong?

---

b. [3pt] Add a record with search-key value F to the B+ tree in Figure 2.

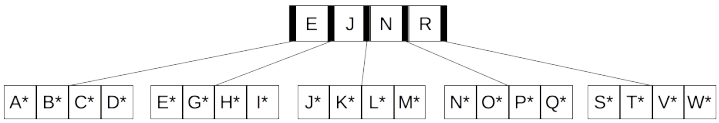

Figure 2: B+ tree for addition.

c. [3pt] Consider a dense B+ tree index for a table **Foo** with search key on column bar. (**Foo** has other columns too.) The search key is *not* unique. **Foo** has 10,000,000 records. The average fan-out of the tree is 200. It is secondary / indirect, so its leaf pages contain *data-entry* records. Each data-entry record contains the search-key attribute's value and an RID that provides the address to the actual record. The index is *clustered*.

50 data-entry records are on a leaf page, on average. In the file that contains the actual data records of the file, 15 records are on a page, on average.

Say that you use the index to fetch the records (with *all* their fields) from **Foo** where bar = 'Jabberwocky'. Assume that 75 records match.

Calculate the number of I/O's the fetch via the index costs. Explain the steps.

---

d. [2pt] Consider the index in Question 4c. However, this time, consider that the index is *unclustered*.

Calculate the number of I/O's the fetch costs now for the 75 records where bar = 'Jabberwocky'.

(You can just explain what are the additional or reduced I/O costs in comparison to your answer to Question 4c.)

5. [10pt] **Design Choices.** *Wow! Much choice!* ANALYSIS

The Toronto-based start-up company *Flash Bash* has just announced that they have created a new flash *main memory*. Their marketing claims the following.

- Faster (slightly) than current main-memory technologies!
- Random access!
  (Per byte, *not* per page, just like other main-memory technologies.)
- Less expensive than disk, per byte!
- Non-volatile (like disk)!

They claim their memory technology is robust. (That is, it will last a long time without faults.) They plan to start producing 1-, 2-, and 5-terabyte chips next month, and the price will be comparable to disk, per byte.[1]

Infamous database researcher Dr. Mark Dogfurry has incorporated *Flash-IT* to build a new SQL-based relational database system that takes advantage of *Flash Bash*'s new memory technology. He has hired you (with stock options!) on the team to build this new database system.

The system will run on a new server machine *without* traditional main memory and disks; instead, it will have a single, large Flash-Bash memory.

---

a. [2pt] Do we still need a buffer pool and a buffer pool manager?
Briefly, why or why not?

---

b. [3pt] Would we still need to support tree indexes?
If so, should we implement B+ tree indexes?
Briefly explain.

[1] I am making all this up, unfortunately.

c. [2pt] Would really long records still be a design concern for us?
Briefly, why or why not?

---

d. [3pt] Would we still need *record IDs* (RIDs)?
Briefly explain.

Extra Space

Did you see your shadow?

Extra Space

Happy Groundhog Day!

EXTRA SPACE

YOU REACHED THE END. TURN IN YOUR TEST. RETURN TO THE WILD.