

# CS 245 Midterm Exam

## Spring 2019

- Please read all instructions (including these) carefully.
- There are 4 problems, some with multiple parts, for a total of 60 points. You have 80 minutes to work on the exam.
- The exam is open notes. You may use a laptop with all communication facilities (Wi-Fi, Bluetooth, etc) disabled.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the backs of the exam pages as scratch paper. Please do not use any additional scratch paper.
- Solutions will be graded on correctness and clarity. For the long-answer problems, please show your intermediate work. Each problem has a relatively simple to explain solution, and we may deduct points if your solution is much more complex than necessary. Partial solutions will be graded for partial credit.

NAME: \_\_\_\_\_

SUID: \_\_\_\_\_

In accordance with both the letter and spirit of the Honor Code, I have neither given nor received assistance on this examination.

SIGNATURE: \_\_\_\_\_

Grading: P1=20 points, P2=8 points, P3=14 points, P4=18 points, total: 60 points.

## Problem 1: Short Answer Questions (20 points)

**1. (2 points)** What was the main benefit of System R over navigational databases such as CODASYL? (*Circle your answer.*)

- (a) System R could provide higher performance for hand-optimized applications.
- (b) System R supported recovery from crashes.
- (c) System R provided data independence.
- (d) System R was implemented using B-trees.

**2. (2 points)** For each of the following abstractions, circle whether they are a logical data model or a physical data structure.

- |      |             |         |          |
|------|-------------|---------|----------|
| i.   | B+ tree:    | LOGICAL | PHYSICAL |
| ii.  | Relation:   | LOGICAL | PHYSICAL |
| iii. | Graph:      | LOGICAL | PHYSICAL |
| iv.  | Hash table: | LOGICAL | PHYSICAL |

**3. (2 points)** During System R's development, its storage system changed from the XRM data structure, which had a tuple list with pointers to "domain" files for the fields of each record, to a B-tree. Which of the following queries are likely to be faster using a B-tree than XRM, assuming both the B-tree and XRM have an index or inversion on the "salary" field? (*Circle your answer.*)

- (a) `SELECT name, dept, salary FROM users WHERE salary >= 100000`
- (b) `SELECT AVG(salary) FROM users`
- (c) Both these queries are likely to be faster using a B-tree.
- (d) Neither of these queries is likely to be faster using a B-tree.

**4. (2 points)** How many failed disks can the RAID 0, RAID 1, and RAID 5 storage levels each tolerate without losing data? (*Circle your answer.*)

- (a) 0, 1 and 1 respectively for RAID 0, 1 and 5.
- (b) 0, 0 and 1 respectively for RAID 0, 1 and 5.
- (c) 0, 0 and 4 respectively for RAID 0, 1 and 5.
- (d) 0, 1 and 5 respectively for RAID 0, 1 and 5.

**5. (2 points)** Why does sequential access generally provide higher throughput than random access even for data in Random Access Memory (RAM)?

**6. (2 points)** In C-Store, which of the following compression formats would allow a SELECT COUNT(\*) GROUP BY C query on column C without decompressing it? (*Circle all that apply.*)

- i. Null suppression
- ii. Dictionary encoding
- iii. Run length encoding
- iv. Lempel-Ziv

**7. (2 points)** Most databases have sparse primary indexes, but dense secondary indexes. Why is it a bad idea to have a sparse secondary index?

**8. (2 points)** We need to store a table called Users with three fields: Name, Country and Job. We are considering several ways to sort the records in this table. Which sort order(s) are likely to lead to the fastest performance (fewest I/Os) for each of the following queries? (*Check the box with fastest performance in each row; if several options are equally fast, check all of them.*)

Query (SELECT * WHERE ...)	Order 1: by Country	Order 2: by Country and then by Job within each country	Order 3: Z-order by Country & Job	Order 4: by Name
Name='Bob'				
Country='US'				
Country='US' AND Job='SRE'				
Job='SRE'				

**9. (2 points)** Why does Spark SQL provide a DataFrame API embedded in languages like Scala and Python in addition to pure SQL statements? (*Circle your answer.*)

- (a) SQL would not allow users to embed user-defined functions.
- (b) The DataFrame API lets users structure their code using standard features in their programming language, such as functions and classes.
- (c) The DataFrame API gives more control because it does not go through a SQL optimizer.
- (d) All of the above.

**10. (2 points)** Which of the following optimizations in relational databases would it **not** make sense to apply in a deep learning framework like TensorFlow? (*Circle your answer.*)

- (a) Data layout optimization to reduce random accesses.
- (b) Compilation of operator graphs into executable code.
- (c) Selectivity estimation based on data statistics.
- (d) Substitution rules to simplify algebraic expressions.

## Problem 2: Relational Algebra (8 points)

a) (3 points) Write the relational algebra expression that represents the following SQL query:

```
SELECT students.name
FROM students NATURAL JOIN enrolments
WHERE enrolments.class = 'CS245' AND students.level = 'PhD'
```

b) (5 points) For each of these following relational algebra expressions, circle whether it is always true or can be false. If it can be false, briefly explain why.

i.  $(R \bowtie S) \bowtie T = (T \bowtie R) \bowtie S$ :                      TRUE              FALSE

ii.  $\sigma_A(\sigma_B(R)) = \sigma_B(\sigma_A(R))$ :                      TRUE              FALSE

iii.  $\Pi_A(\Pi_B(R)) = \Pi_B(\Pi_A(R))$ :                      TRUE              FALSE

iv.  $R \bowtie (S \cap T) = (R \bowtie S) \cap (R \bowtie T)$ :                      TRUE              FALSE

v.  $\sigma_A(R \bowtie S) = \sigma_A(R) \bowtie S$ :                      TRUE              FALSE

### Problem 3: Storage Tiers (14 points)

Green Storage Inc. has built a DNA-based storage device that holds a large amount of data, but can only be read in 1 MB blocks at a rate of 100 blocks per second. This device costs \$1000. In contrast, 100 GB of DRAM costs \$500 but can serve 50 GB/s of accesses.

**a) (6 points)** Netpics.com currently stores photos that it serves on its website in DRAM, and these photos are each around 1 MB in size. How rarely should a photo be accessed for it to be more cost-effective for Netpics.com to place it in DNA storage? *(Please explain your work.)*

**b) (8 points)** Netpics.com now wants to design a B+ tree to index its data on the DNA device. To do this, they need to figure out the optimal degree for the tree. Suppose that Netpics.com wants to store a total of 200 million ( $2 \times 10^8$ ) photos on the device, that the keys for these records are 90 bytes each, and that pointers to blocks on the device are 9 bytes each.

i. If Netpics.com decides to build a B+ tree of degree 10,000 (where each node can have 10,000 keys), what **minimum depth** does the tree need to have to index  $2 \times 10^8$  records, and what is the **number of I/Os** required to look up a key in the index when it has that depth? (*Don't count the cost to then read that photo from storage; just the cost to look up its address using the index.*)

ii. If Netpics.com decides to build a B+ tree of degree 20,000 instead, what **minimum depth** does the tree need to have to index  $2 \times 10^8$  records, and what is the **number of I/Os** to look up a key in the index when it has this depth? (*As above, don't count the cost to then read that photo.*)

## Problem 4: Cost-Based Optimization (18 points)

In this problem, we will be optimizing the query  $X \bowtie Y \bowtie Z$  on relations  $X(A, B)$ ,  $Y(B, C)$  and  $Z(C, D)$ . (Recall that the notation  $X(A, B)$  means that relation  $X$  has attributes  $A$  and  $B$ .) We have collected the following statistics about our relations, where  $T(R)$  is the number of tuples in a relation and  $V(R, A)$  is the number of distinct values of attribute  $A$  in a relation:

$$T(X) = 200$$

$$V(X, A) = 100$$

$$V(X, B) = 20$$

$$T(Y) = 1000$$

$$V(Y, B) = 10$$

$$V(Y, C) = 1000$$

$$T(Z) = 100$$

$$V(Z, C) = 100$$

$$V(Z, D) = 20$$

**a) (8 points)** As a first step to cost-based optimization, we must estimate the statistics of some potential intermediate and final tables in our query. Use the estimation rules based on  $T()$  and  $V()$  that we discussed in class to estimate the  $T()$  and  $V()$  statistics for the following tables:

i.  $X \bowtie Y$

ii.  $Y \bowtie Z$

iii.  $X \bowtie Y \bowtie Z$



**b) (10 points)** We also need to estimate the cost of physical operations we might run in the plan. Suppose that our database engine only supports **sort-merge joins**, where the two input tables must be sorted by the join attribute to produce an output table that also goes to disk (the engine does not support streaming these results directly into another join). When tables are not sorted by the join attribute, the engine first uses a **sort operator** that takes  $4 \cdot b$  disk I/Os for a table containing  $b$  blocks. Finally, suppose that records in our tables (including intermediate tables) are 10 bytes, a disk block is 1000 bytes, and our original tables are sorted as follows:

$X(A,B)$  is sorted by field B       $Y(B,C)$  is sorted by field B       $Z(C,D)$  is sorted by field C

Using this information, estimate the number of disk I/Os for the following physical plans, counting the I/Os for both reading and writing. Please also briefly explain how each plan runs and how any intermediate tables are sorted.

i.  $X \bowtie Y$

ii.  $Y \bowtie Z$

iii.  $(X \otimes Y) \otimes Z$

iv.  $X \otimes (Y \otimes Z)$