CSE 190A Spring 2018 Final Exam

Full Name:

Student ID:

Major:

INSTRUCTIONS

1. You have up to 2 hours and 59 minutes to complete this exam.

2. You can have up to 2 letter/A4-sized sheets of notes, formulas, etc. Apart from this, the exam is closed book/notes/electronics/peers.

3. Please wait until being told to start reading and working on the exam.

4. If you think a question is ambiguous, write down your assumptions, argue that they are reasonable, and then work on the problem using those assumptions.

5. Please ensure that your writing is clear and legible!

<table>
<thead>
<tr>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q 1</td>
<td>20</td>
</tr>
<tr>
<td>Q 2</td>
<td>32</td>
</tr>
<tr>
<td>Q 3</td>
<td>10</td>
</tr>
<tr>
<td>Q 4</td>
<td>20</td>
</tr>
<tr>
<td>Q 5</td>
<td>18</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
</tr>
<tr>
<td>Extra Credit</td>
<td>5</td>
</tr>
</tbody>
</table>
I think we should build a SQL database.

Uh-oh!

Does he understand what he said or is it something he saw in a trade magazine ad?

What color do you want that database?

I think mauve has the most RAM.
Q 1. [20pts] For the following questions, clearly circle True or False.

1. Rename is the only operator in relational algebra whose semantics depends only on the database schema and not the database instance.
   True  False

2. Key-foreign key joins are always inner joins.
   True  False

3. It is always possible to store variable length records in the packed page layout.
   True  False

4. All clustered indexes are primary indexes.
   True  False

5. Internal replacement sort sometimes reduces the number of passes needed for the merge phase of external merge sort.
   True  False

6. Given a physical query plan, there is only one logical query plan that can be translated to it.
   True  False

7. Most RDBMSs use the independence heuristic for estimating the selectivity of a conjunction of predicates.
   True  False
8. It is sometimes possible to obtain superlinear speedup in a parallel DBMS.
   True      False

9. Fault tolerance is one of the key reasons for the success of “Big Data” systems such as MapReduce/Hadoop and Spark compared to parallel DBMSs.
   True      False

10. Variance is not an algebraic aggregate.
    True      False

Q 2. [32pts] For the following questions, **clearly circle** the right answer (only one option is correct).

1. Which of the following relational operators usually can **not** be sped up (in terms of I/O costs) with a B+ tree index at all?
   (a) \( \sigma \)  (b) \( \pi \)  (c) \( \gamma \)  (d) \( \times \)  (e) \( \bowtie \)

2. Given a relation \( R(A, B, C) \), which of the following indexes do **not** match the following query: \( \sigma_{A=a,B=b}(R) \)?
   (a) Hash index on \( R(B) \)  (b) Hash index on \( R(A, B) \)  (c) Hash index on \( R(A, C) \)

3. Which of the following paradigms of parallelism is followed by most parallel DBMSs and “Big Data” systems?
   (a) Shared-memory  (b) Shared-nothing  (c) Shared-everything  (d) Shared-disk

4. A shuffle is **never** needed for which of the following relational operators in a parallel DBMS?
   (a) \( \sigma \)  (b) \( \pi \)  (c) \( \gamma \)  (d) \( \times \)  (e) \( \bowtie \)
5. Which of the following physical join operators is a blocking operator?

(a) Block nested loop   (b) Sort merge   (c) Hash   (d) Index nested loop

6. Which is the most common data partitioning scheme in parallel DBMSs?

(a) Random   (b) Range-based   (c) Hashing-based   (d) Round Robin

7. How many different hash functions does the two-phases “improved” hash join implementation in a parallel DBMS use?

(a) 1   (b) 2   (c) 3   (d) 4   (e) 5

8. Column stores were introduced to speed up primarily which of these relational operations?

(a) Aggregate   (b) Intersection   (c) Set difference   (d) Join

9. In practice, what is the minimum percentage occupancy of a page typically maintained by a B+ tree index?

(a) 0   (b) 25%   (c) 50%   (d) 75%   (e) 99%

10. Which of the following invariants/conditions always hold for an extendible hash index? GD and LD stand for global and local depths respectively.

(a) All LD < GD   (b) All LD ≤ GD   (c) All LD > GD   (d) All LD ≥ GD

11. Which of the following improvements of external merge sort does not affect the fan-in of the merge phase?

(a) Internal replacement sort   (b) Double buffering   (c) Blocked I/O
12. Which typical system resource in a parallel DBMS is not underutilized by a multi-way external merge sort compared to the simple 2-way variant?

(a) Disk  (b) RAM  (c) CPU  (d) Processor caches  (e) Network

13. Which of the following SQL capabilities have no counterpart in (extended) relational algebra?

(a) SELECT DISTINCT  (b) NATURAL JOIN  (c) GROUP BY  (d) ORDER BY

14. Which buffer replacement policy is a variant of LRU with lower overhead?

(a) MRU  (b) Clock  (c) Random  (d) FIFO

15. Which of the following physical join operators benefits when both tables being joined are already stored in the order of the joining attributes' domain?

(a) Block nested loop  (b) Sort merge  (c) Hash  (d) Index nested loop

16. Which typical system resource is not underutilized by the block nested loop join compared to the simple page-nested loop join?

(a) Disk  (b) RAM  (c) CPU  (d) Processor caches
Q 3. [10pts] Operator Implementation. We are given two tables $R(X, Y)$ and $S(X, Y)$ with the same schema, i.e., they are union-compatible. Assume attributes $X$ and $Y$ are of the same size. The table sizes of $R$ and $S$ happen to be $1.2B$ and $3B$ pages respectively, where $B$ is the number of buffer pages available (typically, in the millions). Assume there are no skews in any attribute distributions. Hash table fudge factor is 1.4.

Given the above, what is the smallest I/O cost for each of the following operators in terms of $B$? Exclude the cost of writing the output. Clearly mention the physical operator implementation you chose and briefly explain your calculation of the answer without skipping key steps; otherwise, no points will be given!

1. (2pts) $\pi_X(R)$

2. (4pts) $R \cap S$

3. (4pts) $R \times S$
Q 4. [20pts] Query Processing. Recall the Netflix schema discussed in class:

\[
R(\text{RID}, \text{Stars}, \text{RateDate}, \text{UID}, \text{MID})
\]
\[
U(\text{UID}, \text{UName}, \text{Age}, \text{JoinDate})
\]
\[
M(\text{MID}, \text{MName}, \text{Year}, \text{Director})
\]
R.MID is a foreign key referring to M.MID; R.UID is a foreign key referring to U.UID

We are given the following pieces of information about the database and the system:

\( N_R = 10^7, N_U = 10^6, \) and \( N_M = 10^5 \) (respective number of pages of each table), page size of 8 KB, and allotted buffer memory of 10 GB. Assume that the length of each attribute is 8 B, except for \( U\text{Name}, M\text{Name}, \) and \( \text{Director}, \) each of which is of length 40 B. The fudge factor for hash tables is 1.4.

Answer the following questions. Clearly and briefly explain your derivation/calculation of the answer by mentioning the key steps, including the logical query plan, the chosen physical query plan, whether each operator is pipelined or materialized, and explaining how the I/O costs add up to the given number. Otherwise, no points will be awarded!

1. (6pts) Name 3 indexes, with at least one of them being a hash index, that match the given selection query.

\[
\text{SELECT } * \text{ FROM } M \text{ WHERE Director = “Taika Waititi” AND (Year = 2000 OR Year > 2010)}
\]

2. (3pts) Propose a physical query plan for this query that has a total I/O cost of \( 10^6 \) pages.

\[
\text{SELECT JoinDate, AVG(Age) FROM U GROUP BY JoinDate}
\]
3. (5pts) Propose a physical query plan for this query that is fully pipelined and has a total I/O cost of $101 \times 10^5$ pages.

SELECT Director, AVG(Stars) FROM R, M
WHERE R.MID = M.MID AND Year > 2010 GROUP BY Director
4. (6pts) Propose a physical query plan for this query that has a total I/O cost of $111 \times 10^5$ pages.

```sql
SELECT AVG(Stars) FROM R, M, U
WHERE R.MID = M.MID AND R.UID = U.UID
AND Age <= 35 AND MName = "Avengers: Infinity War"
```
Q 5. [18pts] Query optimization. We are given a relational database schema with the following three relations, wherein $A$ and $B$ are discrete (string) attributes, while $X$ is a numeric attribute.

$$R(A, B, X), S(B, C), \text{ and } T(A, C)$$

For each of the following questions (3pts each), clearly circle either Yes or No for whether the given queries $Q_1$ and $Q_2$ are equivalent.

1. $Q_1 : R \bowtie (S \bowtie T) \text{ and } Q_2 : (R \bowtie S) \bowtie (S \bowtie T)$

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

2. $Q_1 : \sigma_{A=a, B=b, C=c} (R \bowtie S \bowtie T) \text{ and } Q_2 : \sigma_{B=b} (R) \bowtie \sigma_{C=c} (S) \bowtie \sigma_{A=a, C=c} (T)$

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

3. $Q_1 : \pi_{A, B} (R) \cup (\pi_{A} (T) \times \pi_{B} (S)) \text{ and } Q_2 : \pi_{A, B} (R \bowtie S \bowtie T)$

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

4. $Q_1 : \sigma_{A=a, B=b} (R) \bowtie \sigma_{B=b, C=c} (S) \bowtie \sigma_{C=c, A\neq a} (T) \text{ and } Q_2 : (T \cap T) - T$

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

5. $Q_1 : \gamma_{A, AVG(X)} (R) \text{ and } Q_2 : \gamma_{A, AVG(X)} (\pi_{A, X} (R))$

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

6. $Q_1 : \gamma_{COUNT(DISTINCT B)} (R \bowtie S) \text{ and } Q_2 : \gamma_{COUNT(DISTINCT B)} (R \bowtie T)$

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
**Extra Credit Q. [5pts]**. Suppose you are given a Hadoop cluster. You are also given a set of tables in the Netflix schema discussed in class (also provided in Q 4 earlier). Each table is stored as a massive distributed CSV file on HDFS that is hash partitioned on its respective primary key. Assume all ID attributes are stored as integers and there are no majors skews in the distributions of all attributes.

You are asked to determine the total number of ratings for each movie in the Ratings table, as well as the total number of ratings for each user in the same table. But you should output only those movies and users that have at least 10 ratings each.

How will you implement this task in pure MapReduce with no SQL-oriented abstractions on top (Hive, SparkSQL, etc.)? A proper explanation of the Map and Reduce phases are enough; no need to write code in Java, Python, etc. If you had Hive or SparkSQL in the cluster, how will you implement this task using SQL?