CompSci 516: Database Systems

QUAL EXAM

Fall 2019

1. You are strongly encouraged to attempt all questions. If you cannot solve a problem fully, feel free to write partial solutions or your thought process.
2. Do not spend too much time on a problem that you find difficult to solve - move on to other problems.
3. The problems are organized in no particular order, easier problems may appear later.
4. Clearly write any additional assumption you need in your solution.

Problem 1	/ 30	Problem 2	/ 10	Problem 3	/ 24
Problem 4	/ 16	Problem 5	/ 20	Total	/ 100

Q1. (30 = 10 + 10+ 10 pts) RA and SQL

Consider the following tables storing information about an international singing competition with multiple events.
The primary keys are underlined and `P.aid`,`P.eid` are foreign keys referring to the primary keys of `A` and `E` respectively.

- `Event: E(eid, ename, genre)`
- `Artist: A(aid, aname, country)`
- `Participated: P(aid, eid, rank)`

Q1a: (10 pts) Relational Algebra

Write a Relational Algebra expression (or a logical query plan tree) to output `aname` (names) of all the artists who had rank = 1 in **all** events they participated.
Note that you can use σ (select), π (project), $\bowtie$, (join), ρ (rename), and $-$ (set difference) operators.
You may or may not include the artists who did not participate in any events.

Q1b: (10 pts) SQL

$E(\underline{eid}, ename, genre)$	$A(\underline{aid}, aname, country)$	$P(\underline{aid, eid}, rank)$

(same query in SQL)

Write a SQL query to output `aname` (names) of all the artists who had rank = 1 in **<u>all</u>** events they participated.
You may or may not include the artists who did not participate in any event.

$E(\underline{eid}, ename, genre)$	$A(\underline{aid}, aname, country)$	$P(\underline{aid, eid}, rank)$

Q1c: (10 pts) More SQL

Write a SQL query using **only one SELECT** to output the `ename` of all the events where **all** participating artists are from the same country.

Note: there cannot be any sub-query in your solution!

Q2. (10 pts) Normalization

Consider the following schema

R(aid, eid, ename, genre, aname, country, rank)

and the following set of functional dependencies:

- **FD1:** aid, eid → rank
- **FD2:** eid → genre, ename
- **FD3:** aid → aname, country
- **FD4:** ename → genre

Decompose the schema into Boyce-Codd Normal Form (BCNF). Specify the tables in the final schema.

Show all the steps in your decomposition and explain your answer.

Q3. (24 pts) Sorting and Join Algorithms

Consider the following two relations and assume the following:

- A(aid, aname, country): no. of tuples $T_A = 20,000$; no. of tuples per page $n_A = 200$; no. of pages $N_A = 100$.
- P(aid, eid, rank): no. of tuples $T_P = 5000$; no. of tuples per page $n_P = 100$; no. of pages $N_P = 50$.
- Assume that the no. of buffer pages available is $B = 12$.
- Assume on average 20 artists participate in each event.
- Assume all index pages are in memory, and initially all relations are on disk.

Consider the following join query

```
SELECT *
FROM A, P
WHERE A.aid = P.aid
```

Consider three alternatives for the join:

- **option 1:** Block-oriented nested-loop join with A as outer relation (i.e., use all the available buffer pages for join).
- **option 2:** Sort-merge join.
- **option 3:** Index nested loop join with P as outer relation.

Write the estimated cost for all the combinations below (in terms of number of pages in I/O, and ignore the cost for final write to disk). Assume no other indexes exist.
Show your calculations briefly. (you can use the blank next page.)
If an option does not apply for a scenario, **write "N/A"**.

Scenario	cost: option 1	cost: option 2	cost: option 3
(A) Clustered hash index on A.aid for relation A			
(B) Both relations are sorted on aid			

(Extra page for calculations. Please use A-1, A-2, A-3, B-1, B-2, B-3 for different combinations.)

Q4. (16 pts) Transactions

Consider the following schedule with three transactions T_1, T_2, T_3:

- $R_2(B), W_2(B), R_3(C), W_3(C), R_3(A), W_3(A), C_3, R_2(C), W_2(C), R_1(A), R_1(B), W_1(A), W_1(B), C_2, C_1$

Note that $W_i(X)$ (resp. $R_i(X)$) denotes write to (resp. read) element X by transaction T_i, and C_i denotes that T_i has committed.

Answer the following questions. Give brief explanations.

Q4a. (4 points) Is this schedule **conflict-serializable**?
If yes, what is an **equivalent serial schedule**?

Q4b. (2 points) Is this schedule possible under **2PL (two-phase locking)**?

Q4c. (2 points) Is this schedule possible under **strict 2PL**?

Here is the schedule in Q4 again for your convenience:

$R_2(B), W_2(B), R_3(C), W_3(C), R_3(A), W_3(A), C_3, R_2(C), W_2(C), R_1(A), R_1(B), W_1(A), W_1(B), C_2, C_1$

Q4d. (2 points) Is this schedule **recoverable**?

Q4e. (2 points) Does this schedule **avoid cascading rollback**?

Q4f. (4 points) Suppose that right after a crash, the database log (using undo/redo logging) contains the following (and only these) entries. Note that the entry (T, A, u, v) implies that transaction T is updating A, the old value is u, and the new value is v.

- (T_1.start)
- (T_1, A, "good", "better")
- (T_2.start)
- (T_2, A, "better", "best")
- (T_1.commit)

What will be the value of A ("good", "better" or "best") when the recovery is complete?

Q5. (20 pts) write True/False.

No explanations are needed.

1. Given two relations $R(A,B)$ and $S(CD)$ without any nulls, the following equality holds (**True/False**):

$$R - \Pi_{AB}[R \bowtie_{B=C} S] = \Pi_{AB}[R \bowtie_{B \neq C} S]$$

2. One SQL query can have only one logical query plan but multiple physical query plans (**True/False**):

3. Given relations R and S with 100 and 10 pages on disk respectively, the cost of best possible join algorithm can be as low as 100 (**True/False**):

4. Consider the following relation R:

A	B
3	null
10	null

Consider the query

```
SELECT A
FROM R
WHERE B >= 5 OR B < 5
```

This query returns both tuples `(3, null)` and `(10, null)` in the output (**True/False**):

5. Every BCNF decomposition is lossless (**True/False**):

6. If all transactions are read-only (do not write to any element), then every schedule is serializable (**True/False**):

7. STEAL policy requires UNDO on recovery (**True/False**):

8. FORCE policy requires REDO on recovery (**True/False**):

9. A clustered index on a key attribute will have the same performance as an unclustered index on the same attribute (**True/False**):

10. Transaction recovery maintains only the A = atomicity property in ACID (**True/False**):