CompSci 516: Database Systems

QUAL EXAM

Spring 2021

Total = 100

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.

Q1. SQL & RA (20 points)

Consider an online course repository (MOOC) where students from all over the world can sign up for courses offered by teachers from different institutions. Here is the schema:

- C(cid, title) -- “Course” information. **cid** is the unique course id.
- S(sid, name, country) -- “Student” information. **sid** is the unique student id.
- T(tid, name) -- “Teacher” information. **tid** is the unique teacher id.
- E(sid, cid) — “Enrollment” information. sid,cid are foreign keys referencing to Student and Course respectively
- O(tid, cid) – “Offered” information. tid,cid are foreign keys referencing to Instructor and Course respectively.

Q1a. SQL (10 points)

Write a SQL query to output the “**name**”s of the teachers who **only** taught courses such that **all** enrolled students in **each course** he/she taught are from the **same country** (two different courses taught by a teacher in the output might have students from two different countries).

- If there are multiple teachers satisfying the condition with the same name, all of them should appear in the output.
- Teachers who **did not** offer any course ever **should appear in the output**.

Q1b. RA (10 points)

Write the (almost) same query in Relational Algebra (RA). Schema and query repeated below for convenience).

- You can use either a relational algebra expression or a logical query tree in your answer.
- You can use standard relational algebra (set semantics), i.e., can use the following operators: SELECT (σ), PROJECT (π), JOIN (⋈), DIFFERENCE (-), RENAME (ρ).

Commented [1]: should be able to use $$\sigma$$ and then \pi, \join, -, \rho on gradescope.

Write a SQL query to output the “**name**”s of the teachers who **only** taught courses such that **all** enrolled students in **each course** he/she taught are from the **same country** (two different courses taught by a teacher in the output might have students from two different countries).

- **(Different from Q1a**) If there are multiple teachers satisfying the condition with the same name, **they should appear only once in the output.**
- Teachers who **did not** offer any course ever **should appear in the output**.

Q2. Normalization (12 points)

Consider a relation R(A,B,C,D,E) with the following functional dependencies:

A →B

C →B

DE →C

Commented [2]: should be able to use $$A \rightarrow B$$ etc. on gradescope.

Q2a. (7 points)

Decompose R into BCNF (Boyce Codd Normal Form). **Show your work**. **Your final answer should consist of a list of table names and attributes and an indication of the keys in each table (underlined attributes).**

Q2b. (5 points)

Give a proof why BCNF decomposition is "lossless", i.e., if the two tables generated after each decomposition step are joined back, no spurious tuples that were not present in the original relation can be generated. You can use the first step of your decomposition in Q2a as an example.

Q3. Join Algorithms (20 points)

Consider two relations R(A, B) and S(**B**, C) and a natural join on R, S on B.
R.B is a foreign key in R referring to the primary key S.B of S.

Assume

- R has n1 = 400 tuples, and a page of R can contain m1 = 5 tuples.
- S has n2 = 50 tuples, and a page of S can contain m2 = 5 tuples.
- There are M = 10 buffer pages available in the main memory (buffer pool).
- Each S.B value joins with the same number of R.B values.
- All of the R and S pages are initially on the disk.
- Each R and S page contains the same number of tuples.

Q3a. (5 points)

Consider the block-nested loop join algorithm where all M buffer pages are used with S as the outer relation and R as the inner relation.
Suppose **one buffer page is allocated to the inner relation, one buffer page is allocated to the output, and all M-2 pages are available to the outer relation.**

What is the I/O cost (in terms of the number of pages read into the memory, ignore the writing cost).

Q3b. (3 points)

Consider a variation of Q3a of the block-nested loop join algorithm where all M buffer pages are used with S as the outer relation and R as the inner relation.

Suppose **two buffer pages are allocated to the inner relation, two buffer pages are allocated to the output, and all M-4 pages are available to the outer relation.**

Does the I/O cost increase, decrease, or remain the same in this case (in terms of the number of pages read into the memory, ignore the writing cost).

Q3c. (2 points)

Is there any benefit of allocating two buffer pages to the inner relation (may or may not be related to I/O cost)?

Q3d. (2 points)

Is there any benefit of allocating two buffer pages to the output relation?

Q3e. (4 points)

What is the cost of index-nested loop join with S as the outer relation? Assume an **unclustered B+-tree index on R.B** for R. Assume the entire index structure is already in memory. Assume M = 10 pages are available for holding R,S, and an output page.

Again, the cost is in terms of the number of pages read into the memory, ignore the writing cost.

Q3f. (4 points)

How does the cost in Q3e change if the index on R.B is **a clustered B+-tree index**?

Q4. Transactions (20 points)

Q4a. (10 points) UNDO/REDO log/recovery

At the time of a system crash, let the log segment (in the UNDO/REDO logging scheme of fuzzy/non-quiescent checkpointing) involving four transactions S,T,U,V be as follows. Recall that an update record (T, A, 5, 6) means transaction T changed an object A from old value 5 to a new value 6.

1. (START S)

2. (S, X, 15, 22)

3. (COMMIT S)

4. (START T)

5. (T, X, 22, 37)

6. (START CKPT(T))

7. (T, Y, 12, 26)

8. (START U)

9. (COMMIT T)

10. (U, X, 37, 43)

11. (END CKPT)

12. (U, Y, 26, 31)

13. (START V)

14. (START CKPT(U,V))

15. (COMMIT U)

16. (V, Y, 31, 48)

(i) (2 points) Which updates are guaranteed to be on the disk after the crash? Write the step numbers and the update records.

(ii) (3 + 3 = 6 points) What are the final values of X and Y when the recovery is done?

(2 points) Explain your answer briefly.

Q4b. (10 points) Concurrency control

Consider a schedule with three transactions T1, T2, T3:

$R_3(C)$, $R_1(A)$, $W_1(B)$, $R_2(B)$, $W_2(D)$, $W_3(A)$

Commented [3]: it is okay to write R3(C) etc instead of subscripts.

Recall that $R_3(C)$ denotes C is being read by T3. Similarly $W_3(C)$ denotes C is being written by T3, etc.

(i) (2 + 3 = 5 points)

Is this schedule realizable by a **two-phase locking protocol** (2PL, NOT strict 2PL)?

Briefly explain your answer.

You can insert a "valid" sequence of "locks" and "unlocks" to justify your answer, e.g., L3(C) would denote C is being locked by T3 and U3(C) would denote C is being unlocked by T3.

(ii) (5 points) Find ALL the equivalent serial schedules of this schedule and explain your answer (otherwise argue that none exist).

Q5. Short Q/A (8 * 2 = 16 points)

No explanations are needed.

Q5a:

What is the output of the following SQL query on a relation R(A, B, C) with two tuples R(5, NULL, 6) and R(NULL, 7, NULL).

```
SELECT COUNT(*)
FROM R
WHERE ((A >= 5) AND (B <= 7)) OR ((C < 3) AND (B = 7))
```

Q5b.

Are the following queries q1, q2 equivalent on R(A, B) where all A and B values are not NULLs? Write Yes or No.

```
q1 =
SELECT DISTINCT Temp1.A,
        (SELECT SUM(Temp2.B)
          FROM R AS Temp2
          WHERE Temp1.A = Temp2.A) AS P
FROM R AS Temp1

q2 =
SELECT A, SUM(B) AS P
FROM R
GROUP BY A
```

Q5c.

Are the following queries q1, q2 equivalent on R(A, B) and S(B, C)? Write Yes or No.

```
q1 =
```

```
SELECT A
FROM R
WHERE EXISTS
        (SELECT *
         FROM S
         WHERE R.B = S.B)

q2 =
SELECT R.A
FROM R, S
WHERE R.B = S.B
```

Q5d.

Consider a relation *Emp*(*eid*, *age*) and the query

```
SELECT *
FROM Emp
WHERE age >= 65
```

We can use a hash index on age to answer this query: **True or False?**

Q5e.

Consider two relations R(A, B) and S(B, C) where the tuples in R are sorted by A on disk. The output of a sort-merge join between R and S will be naturally sorted by A: **True or False?**

Q5f.

Consider a table R(**A, B**,C) with the following properties:

R contains no duplicate rows, and {A,B} is a key of R;
$|R| = 1{,}000$
$|\pi_A R| = 800$.

Which of the following is the most accurate bound on the range of $|\pi_B R|$?

(a) $1 \leq |\pi_B R| \leq 800$

(b) $1 \leq |\pi_B R| \leq 999$

(c) $2 \leq |\pi_B R| \leq 999$

(d) $|\pi_B R| = 1{,}000$

Q5g.

Which of the following SQL queries from q1, q2, q3 are syntactically valid on R(A, B, C)?

Select all that apply.

q1:

```
SELECT A, B
FROM R
GROUP BY A, B
```

q2:

```
SELECT A, B
FROM R
GROUP BY A, B, C
```

q3:

```
SELECT A, B, C
FROM R
GROUP BY A, B
```

Q5h.

Which of the following ACID properties does the UNDO process aim to satisfy primarily?

(a) A = Atomicity
(b) C = Consistency
(c) I = Isolation
(d) D = Durability

Q6. Recursion (6 points)

Consider a relation E(U, V) storing edges in a directed graph G, e.g., a tuple E(1, 3) denotes that there is an edge from vertex 1 to vertex 3.

Write a recursive SQL query to output “1” if G has a cycle (if it does not have a cycle, the output will be empty).

Recall that WITH RECURSIVE keywords are used in recursive queries in SQL.

Q7. Join Order (6 points)

Consider three relations R1(A, B), R2 (B, C), and R3 (A, B, C, D), each with n tuples.

Consider the natural join of these three relations, where two relations will be joined first, and then the third relation will be joined with the result.

Q6a. (3 points) Show a join order (i.e., which two relations to be joined first) that may lead to about n^2 (square of n) intermediate or final output tuples. Give an example (with a small value of n like 2, 3, 4,., but that would generalize to arbitrary n) to illustrate such an example.

Q6b. (3 points) Show a join order (i.e., which two relations to be joined first) that would NEVER lead to more than n intermediate or final output tuples. Briefly explain your answer.