CompSci 516: Database Systems

Midterm

Fall 2018

This booklet has 12 pages (including the cover page and 2 blank pages at the end).
You can use the reverse sides as additional space for writing your answers.

INSTRUCTIONS

1. No external help (books, notes, laptops, tablets, phones, etc.) or collaboration is allowed.
2. You have **75 minutes** to answer questions that add up to **100 points**. i.e. you have about 7.5 mins for 10 points, and about 15 mins for 20 points.
3. Some questions may need more time, some questions less. Please budget your time accordingly.
4. If you cannot solve a problem fully, write partial solution for partial credit. **Even if explanations are not sought in a question, if you cannot fully solve a problem, writing down your thought process may lead to some partial credit.**
5. Do not spend too much time on a problem that you find difficult to solve - move on to other problems.
6. **The problems are organized in no particular order, easier problems may appear later.**

****Write Your Name Here** (1 bonus point):**

First Name:

Last Name:

All the best!

DO NOT WRITE BELOW THIS LINE

Problem 1	/ 20	Problem 2	/ 20	Problem 3	/ 10
Problem 4	/ 18	Problem 5	/ 32	bonus point	/ 1
Total		/ 100			

Q1. (20 = 10 + 10 pts) RA, RC

Consider the following tables storing information about an international music competition with different events like guitar, violin, piano etc. (keys are underlined).

- `E(eid, event)` – stores event information.
- `A(aid, aname, country)` – stores artist information – name and country of origin.
- `P(aid, eid, rank)`
  Also `aid` and `eid` are foreign keys referring to `A` and `E` respectively. One artist can participate in multiple events.

Q1a: (10 pts) RC

Write an RC expression (TRC or first order logic form) to find names of all the artists (`aname`) who got rank = 1 in all events they participated .

(Hence if a rank-1 holder participated in only one event, s/he is going to appear in the solution.)
Ans: $\{q \mid \exists a \in A(a.aname = q.aname)\ \forall p \in P(a.aid = p.aid) \Rightarrow (p.rank = 1)\}$

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

Q1b: (10 pts) RA

(**same query in RA**) Write an RA expression (or logical query plan tree) to find names of all the artists (`aname`) who got rank = 1 in all events they participated.

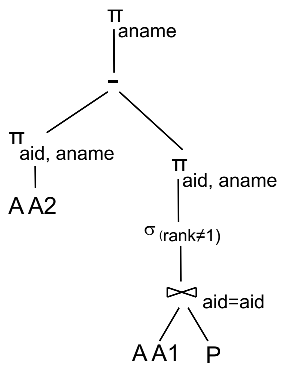
Ans:

$$
\begin{aligned}
& \{q \mid \exists a \in A[(A.aname = q.aname) \ \wedge \ \forall p \in P[(A.aid = P.aid) \Rightarrow (P.rank = 1)]\} \\
= \ & \{q \mid \exists a \in A(A.aname = q.aname) \ \wedge \ [\forall p \in P[\neg(A.aid = P.aid) \vee (P.rank = 1)]\} \\
= \ & \{q \mid \exists a \in A(A.aname = q.aname) \ \wedge \ [\neg\exists p \in P \neg[\neg(A.aid = P.aid) \vee (P.rank = 1)]]\} \\
= \ & \{q \mid \exists a \in A(A.aname = q.aname) \ \wedge \ [\neg\exists p \in P[(A.aid = P.aid) \wedge \neg(P.rank = 1)]]\} \\
= \ & \{q \mid \exists a \in A(A.aname = q.aname) \ \wedge \ [\neg\exists p \in P[(A.aid = P.aid) \wedge (P.rank \neq 1)]]\}
\end{aligned}
$$

The above transformation says the intuitive equivalent statement: we are looking for artists for whom there are no `P` tuples matching the `aid` where the rank is not equal to 1. The above transformation is not needed in the answer.

In other words, we are looking for all artists except the ones for whom there exists such a `P` tuple with rank $\neq 1$.

There are many possible solutions – one solution is given below:



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

Q2: (20 = 10 + 10 pts) SQL

Q2a: (10 pts) SQL for query in Q1

(same query from Q1 in SQL) Write a SQL query to find names of all the artists (`aname`) who got rank = 1 in all events they participated.

Ans:

Once again, there are many solutions possible. Here are two:
solution-1:

```
SELECT aname
FROM A
WHERE aid not in
(SELECT P.aid
FROM P
WHERE (P.rank <> 1)
)
```

solution-2:

```
SELECT aname
FROM A
WHERE NOT EXISTS
(SELECT *
FROM P
WHERE (P.rank <> 1)
AND P.aid = A.aid)
```

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

Q2b: (10 pts) More SQL

Write an SQL query **without using nested sub-queries or without using a WITH clause** that outputs the names of the events `event` where all participating artists (not necessarily rank = 1 holders) are from the same country.

Note: there can be ****exactly one**** `SELECT` clause in your solution.

Ans:

```
SELECT E.event
FROM A, P, E
WHERE A.aid = P.aid AND P.eid = E.eid
GROUP BY E.eid, E.event
HAVING COUNT(DISTINCT A.country) = 1
```

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

Q3. (10 = 1 * 10 pts) Indexing

Given the same schema as in Q1 (repeated above) and the following query, specify (write yes or no – NO explanations are needed) whether each of the following choices of indexes can speed up the query (for some data distribution), **assuming it is the only index that is available.**

```
SELECT A.aname
FROM A, P, E
WHERE A.aid = P.aid AND P.eid = E.eid
       AND event = 'guitar'
       AND rank > 3
```

1. B+-tree index on `P(rank)`. **Yes/No:**
2. Hash index on `P(rank)`. **Yes/No:**
3. B+-tree index on `A(aid, aname)` **Yes/No:**
4. B+-tree index on `A(aname, aid)` **Yes/No:**
5. Hash index on `E(eid)` **Yes/No:**
6. Hash index on `E(eid, event)` **Yes/No:**
7. Hash index on `E(event, eid)` **Yes/No:**
8. Hash index on `A(aid, country)` **Yes/No:**
9. Hash index on `A(country, aid)` **Yes/No:**
10. B+-tree index on `P(aid, rank)` **Yes/No:**

Ans:

1. B+-tree index on `P(rank)`. **Yes/No:** Yes
   (inequality selection on `rank`)
2. Hash index on `P(rank)`. **Yes/No:** No
   (inequality selection on `rank`, hash index cannot be used.)
3. B+-tree index on `A(aid, aname)` **Yes/No:** Yes
   (equality on `aid`, a prefix)
4. B+-tree index on `A(aname, aid)` **Yes/No:** No
   (no selection prefix on `aname`)
5. Hash index on `E(eid)` **Yes/No:** Yes
   (equality on `eid`)

6. Hash index on `E(eid, event)` **Yes/No:** Yes
   (equality on `eid` and `event`)

7. Hash index on `E(event, eid)` **Yes/No:** Yes
   (equality on `event` and `eid`)

8. Hash index on `A(aid, country)` **Yes/No:** No
   (no equality on `country`)

9. Hash index on `A(country, aid)` **Yes/No:** No
   (no equality on `country`)

10. B+-tree index on `A(aid, country)` **Yes/No:** Yes
    (selection on prefix `aid`)

Q4. (18 pts) Query Evaluation

Consider the following two relations from Q1 with the stated assumptions:

- `A(aid, aname, country)`: no. of tuples $T_A = 20,000$; no. of tuples/page $n_A = 200$; no. of pages $N_A = 100$.
- `P(aid, eid, rank)`: no. of tuples $T_P = 5000$; no. of tuples/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.
- Ignore page boundaries.

Consider the following 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.
- **option 2:** Sort-merge join.
- **option 3:** Index nested loop join with `P` as outer.

For the three scenarios below, for all three options, write the cost (in terms of I/O, assuming initially all relations are on disk, ignore final write). If an option does not apply for a scenario, **write "N/A"**.

No explanations are necessary. But you can show your calculations in the boxes or on reverse side of this page, which we may consider for partial credit.

Scenario	cost: option 1	cost: option 2	cost: option 3
(1) Clustered hash index on `A(aid)`			
(2) Both relations are sorted on `aid`			
(3) Clustered B+-index on `A(aid)` and `P` is sorted on aid			

Ans:

Scenario	**cost: option 1**	**cost: option 2**	**cost: option 3**
(1) Clustered hash index on `A(aid)`	600	750	5050
(2) Both relations are sorted on `aid`	600	150	N/A
(3) Clustered B+-index on `A(aid)` and `P` is sorted on `aid`	600	150	5050

- **Block-oriented nested-loop join with `A` as outer**: The cost is the same for all three scenarios. It is $N_A + \lceil \frac{N_A}{B-2} \rceil \times N_P = 100 + 10 \times 50 = 600$.

- **Sort-merge join:**

  - Cost of sorting `A` on `aid`: (i) Pass-0: create $\lceil \frac{N_A}{B} \rceil = \lceil 100/12 \rceil = 9$ sorted runs each with 12 pages; (ii) Pass-1: $B - 1 = 11$ buffers are available for a 9-way merge, so no further passes are needed. Total I/O for reading and writing back to disk = $2 \times 2 \times N_A = 400$.
  - Cost of sorting `P` on `aid`: (i) Pass-0: create $\lceil \frac{N_P}{B} \rceil = \lceil 50/12 \rceil = 5$ sorted runs each with 12 pages; (ii) Pass-1: $B - 1 = 11$ buffers are available for a 5-way merge, so no further passes are needed. Total I/O for reading and writing back to disk = $2 \times 2 \times N_A = 200$.

  When both relations are sorted, $B - 2 = 10$ buffers are available for the merge-phase, so the cost is $N_A + N_P = 100 + 50 = 150$ (ignoring final write).

  Depending on whether initially one or both relations are sorted, the costs are **(i) Scenario-1: (none sorted)** 400 + 200 + 150 = 750; **(ii) Scenario-2: (both sorted)** 150 = 150; **(i) Scenario-3: (`P` sorted)** But since there is a clustered index on `A(aid)`, `A` is sorted too. So cost is 150 = 150.

- **Index nested loop join with P as outer.** Given a `P` tuple there are 20 matching `A` tuples, and for a clustered index on `A(aid)`, only 1 I/O is needed since a page of `A` can hold $50 > 20$ `A` tuples.

  - **Scenario-1:** Cost is $N_P + T_P \times 1 = 50 + 5000 \times 1 = 5050$.
  - **Scenario-2:** Not applicable since no index available.
  - **Scenario-3:** Cost is again 5050.

Q5. (32 pts) Short Q/A

Q5a. (2 * 14 = 28 pts) Are the following statements True or False?

No explanations are needed.

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

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

   Ans: **False**: try $R(1,1), R(1,2), S(1,3), S(3,4)$.

2. A relation R (set semantic) has at least one superkey (**True/False**):
   Ans: **True**: The set of all attributes is always a superkey.

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**):
   Ans: **False**: The lowest cost of any join algorithm can be $100 + 10 = 110$ since every page has to be read from the disk.

4. Suppose relations R and S have the same schema and p is a predicate over this schema. The relational algebra expressions $\sigma_p(R - S)$ and $\sigma_p R - \sigma_p S$ are equivalent (i.e., their answers always agree with each other regardless of the predicate p and the contents of R and S). (**True/False**):
   Ans: **True**

5. Suppose relations R and S have the same schema and L is a subset of attributes. The relational algebra expressions $\pi_L(R - S)$ and $\pi_L R - \pi_L S$ are equivalent (**True/False**):
   Ans: **False**: Consider $R(A,B) = \{(0,0)\}$, $S(A,B) = \{0,1\}$, and $L = \{A\}$

6. Suppose relations R, S and T have the same schema. The relational algebra expressions $(R \bowtie T) - (S \bowtie T)$ and $(R - S) \bowtie T$ are equivalent (**True/False**):
   Ans: **True**.

7. Consider relation $R(A,B,C,D)$. Suppose we know $\{A,B\}$ is a key of R (and R may or may not have other keys). Then the FD $A \rightarrow B$ cannot hold in R. (**True/False**):
Ans: **True**: If $A \rightarrow B$, then $A^+ = (AB)^+ =$ all attributes. So A would be a key, and AB cannot be a key (it would be a superkey).

8. Consider relation $R(A,B,C,D)$. Suppose we know $\{A,B\}$ is a key of R (and R may or may not have other keys). Then the FD $C \rightarrow AB$ cannot hold in R. (**True/False**):
Ans: **False**: There can be two incomparable keys like SSN and $(name, address)$. In that case $AB \rightarrow C$ and $C \rightarrow AB$.

9. Consider the database schema below:

```
create table R(A integer not null PRIMARY KEY, B integer not null);
create table S(C integer not null PRIMARY KEY,
A integer not null REFERENCES R(A));
```

Regardless of the database instance, the number of distinct $S.A$ values must be no greater than the number of distinct $R.A$ values.

(**True/False**):
Ans: **True**: Every $S.A$ will point to at most one value of $R.A$.

10. In the above question, regardless of the database instance, the number of distinct $S.C$ values must be no greater than the number of distinct $R.A$ values.

(**True/False**):
Ans: **False**: R can have just one row, S can have 100 rows each with unique IDs $S.C$, but every $S.A$ value pointing to the same $R.A$ value.

11. The following two SQL queries are equivalent over a schema $Users(\underline{uid}, pop, date)$:
(i) `SELECT * FROM Users WHERE pop > 0.5 AND pop < 0.9;`
(ii) `(SELECT * FROM Users WHERE pop > 0.5) INTERSECT ALL (SELECT * FROM Users WHERE pop < 0.9);`

(Recall that INTERSECT ALL or UNION ALL preserves duplicates.)

(**True/False**):
Ans: **True**: The two queries are equivalent even if there are duplicates or NULL values.

12. The following two SQL queries are equivalent over a schema $Users(\underline{uid}, pop, age)$:
:
(i) `SELECT * FROM Users WHERE age < 6 OR pop > 0.9;`
(ii) `(SELECT * FROM Users WHERE age < 6) UNION ALL (SELECT * FROM Users WHERE pop > 0.9);`

(**True/False**):
Ans: **False**: If a user has age less than 6 and popularity higher than 0.9, this user will be returned once by the first query, but twice by the second.

13. Consider the natural join between two tables $R(A,B)$ and $S(B,C)$, where R is sorted by A and S is sorted by B . The output of a sort merge join between R and S on B will be naturally sorted by A.

(**True/False**):
Ans: **False**: The output will be ordered by B instead of A because we need to first sort R by the join attribute, which is B.

14. Consider the natural join between two tables $R(A,B)$ and $S(B,C)$, where R is sorted by A and S is sorted by B . Further, there is an index on $S.B$.The output of a index-nested loop join between R and S on B will be naturally sorted by A.

(**True/False**):
Ans: **True**: For every tuple of R, the matching tuples from S will be obtained using the index, hence the output will be sorted by A.

Q5b. (4 pts)

Consider the following relation R:

A	B
3	null
10	null

Consider the query

```
SELECT A
FROM R
WHERE B >= 7 OR B < 7
```

What is the output of the above query?

Briefly explain your answer in 1-2 sentences.
Ans: This query returns empty relation, i.e. no tuples are returned.

On both tuples, the predicate $(B \geq 7)$ evaluates to `unknown`, also $(B < 7)$ evaluates to `unknown`, since B is `null`. By 3-valued logic, $\texttt{unknown} \vee \texttt{unknown} = \texttt{unknown}$, and the WHERE clause returns a tuple if and only only if it evaluates to `true`. Hence no tuples are returned although the predicate itself covers the entire range of B.

Blank additional page to be used for rough calculations. DO NOT write solutions here.

Blank additional page to be used for rough calculations. DO NOT write solutions here.