UCLA
Computer Science Department
Fall 2019

Instructor: J. Cho

Student Name and ID: ______________________________

CS143 Midterm: Closed Book, 110 minutes

(IMPORTANT PLEASE READ **):**

- *Simplicity and clarity of your solutions will count.* You may get as few as 0 point for a problem if your solution is far more complicated than necessary, or if we cannot understand your solution.
- If you need to make any assumption to solve a question, please write down your assumptions.
- To get partial credits, you may want to write down how you arrived at your answer step by step.
- You may use one-page double-sided cheat-sheet during exam. You are also allowed to use a calculator.
- Please, write your answers neatly. Attach extra pages as needed. Write your name and ID on the extra pages.

Problem	**Score**	
1	20	
2	10	
3	10	
4	20	
Total	60	

Problem 1 Relational Algebra (20 points)

1. Consider a relation $R(A, B)$ that contains r tuples and a relation $S(B, C)$ that contains s tuples; assume $r \geq s > 0$. Make no assumptions about keys. For each of the following relational algebra expressions, write down the minimum and maximum number of tuples that could be in the result of the expression using r, s, and/or numbers. (15 points total, 1.5 point per answer)

   ANSWER:
   $R \cup \rho_{S(A,B)}(S)$
   `Min: r (when S` $\subseteq$ `R)`
   `Max: r + s (when R` $\cap$ `S = ∅)`

   $\pi_{A,C}(R \bowtie S)$
   `Min: 0 (when all R.B values are different from S.B values)`
   `Max: r x s (when all R.B = S.B = b)`

   $\pi_B(R) - (\pi_B(R) - \pi_B(S))$
   `Min: 0, Max: s. This expression is equivalent to` $\pi_B(R) \cap \pi_B(S)$

   $(R \bowtie R) \bowtie R$
   `Min: r, Max: r. R` $\bowtie$ `R is always R`

   $\sigma_{A>B}(R) \cup \sigma_{A<B}(R)$
   `Min: 0 (when A=B for every tuple in R), Max: r (when A≠B for every tuple in R)`

2. Consider a relation $R(A, B, C)$. You may assume there are no NULL values or duplicate tuples in R.

   Consider the following relational query:

   $\sigma_{Y \neq V \vee Z \neq W}(\rho_{R(X,Y,Z)}(R) \bowtie \rho_{R(X,V,W)}(R))$

   If the result of the above query is always empty, using 30 words or less, state a property/constraint/fact on R that is guaranteed to hold. Keep in mind that there is a very concise correct answer corresponding to a concept covered in class. (5 points)

   ANSWER:
   `X is a key of R. If the result is always empty, there exist no two tuples in R whose A values are the same, but B or C values are different.`

Problem 2 SQL (10 points)

1. Consider two SQL tables `T1(A,B)` and `T2(C)`. You may assume there are no NULL values in either table. Make no assumptions about keys. Consider the following three SQL queries:

```
Q1: SELECT A  FROM T1
    WHERE B IN (SELECT C FROM T2)

Q2: SELECT A  FROM T1, T2
    WHERE T1.B = T2.C

Q3: SELECT DISTINCT A  FROM T1, T2
    WHERE T1.B = T2.C
```

Circle one of the following statements. (Note: two queries are equivalent if they always return the same result.)

(a) Q1, Q2, and Q3 are all equivalent.

(b) Q1 and Q2 are equivalent; Q3 may produce a different answer sometimes.

(c) Q1 and Q3 are equivalent; Q2 may produce a different answer sometimes.

(d) Q2 and Q3 are equivalent; Q1 may produce a different answer sometimes.

(e) Q1, Q2, and Q3 may all produce different answers sometimes.

If you chose any answer other than (a), write down the *smallest* instances of `T1` and `T2` (i.e, `T1` and `T2` with fewest tuples) that demonstrate the nonequivalence(s). In addition, please show the result of all three queries on your tables. You may lose points if your tables have too many tuples. (6 points)

ANSWER:
`(e). T1(A, B) = {(1, 2), (1, 3)}. T2(C) = {2, 2, 3}. Q1 = {1, 1}, Q2 = {1, 1, 1}, Q3 = {1}. We gave 2 points if you selected (c) and provided a consistent example.`

2. Consider a SQL table `T(A)`, where `A` is of integer type. You may assume there are no NULL values in `T`. Make no assumptions about keys. Consider the following two SQL queries:

```
Q1: SELECT A FROM T
    WHERE NOT A >= SOME (SELECT A FROM T)

Q2: SELECT A FROM T
    WHERE A < (SELECT MAX(A) FROM T)
```

Circle one of the following statements. (Note: two queries are equivalent if they always return the same result.)

(a) Q1 and Q2 are equivalent.

(b) Q1 and Q2 may produce a different answer sometimes.

If you chose (b), write down the *smallest* instance of `T` (i.e., `T` with fewest tuples) that demonstrates the nonequivalence. In addition, show the result of both queries on your table. Again, you may lose points if your table has too many tuples. (4 points)

ANSWER:
`(b).  T(A) = {1, 2}.  Q1 = {}, Q2 = {1}.`

Problem 3 (Database Integrity): 10 points

Consider tables `T1(P,A)` and `T2(F,B)`. This problem explores using triggers to enforce two constraints:

(1) Key constraint on `T1.P`
(2) Referential integrity constraint from `T2.F` to `T1.P`

To keep things simple, you may assume there are never NULL values for T1.P.

1. List all possible data modification operations on `T1` and `T2` that could violate either the key constraint or the referential integrity constraint. For update operations, include the specific columns. You do not need to associate the operations with which constraint(s) they may affect. As an example, we have already included one modification operation, "Update to `T1.P`", in the answer box. Fill in the box with other modification operations that may cause a violation. (2 points)

ANSWER:

```
Key constraint violation:
INSERT ON T1.  UPDATE(P) ON T1.

RI violation:
DELETE ON T1, UPDATE(P) ON T1
INSERT ON T2, UPDATE(F) ON T2
```

2. Now you need to create a trigger to enforce the key constraint when column `T1.P` is updated. In this part of the problem, you will specify a *row-level before trigger* to enforce the key constraint.

   - Assume that when a tuple in `T1` is updated, the new value of `P` in that tuple is always different from the old one. Make no other assumptions about the updates.
   - The trigger should execute a special "`RAISE-ERROR`" command when the key constraint is violated. This command will abort the statement that caused the violation.

   Fill in the blanks in the following skeleton. You should make use of trigger features to enforce the constraint, but without getting overly complex. Note that it is fine to leave some boxes empty, as appropriate. You should use syntax suggested by the SQL99 standard, not by a particular vendor. (4 points)

   ANSWER:

```
CREATE TRIGGER UpdateKeyConstraint
BEFORE UPDATE OF P ON T1
REFERENCING NEW ROW AS NN
FOR EACH ROW
WHEN (EXISTS(SELECT * FROM T1 WHERE T1.P = NN.P))
RAISE-ERROR;
```

3. Now you need to create a trigger to enforce the referential-integrity constraint when column `T1.P` is updated. In this part of the problem, you will specify a *statement-level after trigger* to enforce the constraint.

   - Assume that when a tuple in `T1` is updated, the new value of `P` in that tuple is always different from the old one. Make no other assumptions about the updates.
   - For the referential integrity constraint, please implement the "`ON UPDATE SET NULL`" policy.

   Fill in the blanks in the following skeleton. You should make use of trigger features to enforce the constraint, but without getting overly complex. Note that it is fine to leave some boxes empty, as appropriate. You should use syntax suggested by the SQL99 standard, not by a particular vendor. (4 points)

   ANSWER:

```
CREATE TRIGGER UpdateForeignKeyConstraint
AFTER UPDATE OF P ON T1
REFERENCING OLD TABLE AS OO
UPDATE T2
SET F = NULL
WHERE F IN (SELECT P FROM OO)
```

Problem 4 (Authorization): 20 points

1. Consider the following two tables in a database:

   ```
   Student(sid, name, address, GPA)
   Major(sid, dept)
   ```

   `sid` is a key for `Student`, and `(sid, dept)` together is a key for `Major` (i.e., a student may have multiple majors). No attributes are permitted to be NULL. The owner (creator) of these tables is a user named Hennessy.

   (a) Hennessy wants to grant to a user named Plummer the following two permissions on all students with at least one major containing the string "Engineering" (and only those students).

      i. Read `(sid, name, address, GPA)` of those students.

      ii. Modify the address of those students.

   Is it possible to specify a command or a sequence of commands that achieves this goal? If so, write down such command(s). If not, explain why not. Make sure to adhere to the SQL standard. Remember that you may lose points if your example/explanation is overly complicated. (5 points)

ANSWER:

```
CREATE VIEW EngStud AS
SELECT * FROM Student
WHERE sid IN
(SELECT sid FROM Major WHERE dept LIKE '%Engineering%');

GRANT SELECT, UPDATE(address) ON EngStud To 'Plummer';
```

(b) Hennessy further wants to grant to a user named Elaine the following permissions on the student(s) whose GPA is the highest among all students.

i. Read `(sid, name, address, GPA)` of such student(s).

ii. Modify the address of such student(s).

Is it possible to specify a command or sequence of commands that achieves this goal? If so, write down such command(s). If not, explain why not. Make sure to adhere to the SQL standard. You may lose points if your example/explanation is overly complicated. (5 points)

ANSWER:

```
It is NOT possible.  To find the max, we need to use aggregates
or join the Student table with itself ( either directly in the
WHERE clause or indirectly using subqueries).  None of these are
allowed when we want to create a modifiable view.
```

(c) Finally, Hennessy wants to grant to a user named Sahami the following permissions.

   i. Read the sids of those students who are majoring in “CS”.

   ii. Add CS as a student’s major (presuming the student is already in the database but not majoring in CS).

   Is it possible to specify a command or sequence of commands that achieves this goal? If so, write down such command(s). If not, explain why not. Make sure to adhere to the SQL standard. You may lose points if your example/explanation is overly complicated. (5 points)

ANSWER:

```
CREATE VIEW CSMajor AS
SELECT * FROM Major WHERE dept='CS'
WITH CHECK OPTION;

GRANT SELECT, INSERT ON CSMajor TO 'Sahami';
```

2. Consider a table `T(A,B,C)` with owner Amy, and the following sequence of statements related to privileges on `T`. Each statement is prefaced with the user issuing it.

```
Amy: GRANT SELECT, UPDATE ON T TO 'Bob' WITH GRANT OPTION
Amy: GRANT SELECT, UPDATE ON T TO 'Carol' WITH GRANT OPTION
Amy: GRANT SELECT, UPDATE ON T TO 'Frank' WITH GRANT OPTION
Bob: GRANT SELECT, UPDATE(A, B) ON T TO 'David' WITH GRANT OPTION
Carol: GRANT SELECT ON T TO 'David' WITH GRANT OPTION
Frank: GRANT UPDATE(A, C) ON T TO 'David'
David: GRANT SELECT, UPDATE(A) ON T TO 'Eve'
Amy: REVOKE SELECT, UPDATE ON T FROM 'Bob' CASCADE
```

What privileges on table T does Eve have after this sequence of statements? (5 points)

ANSWER:
`Only SELECT on T. After REVOKE, there still exists a path for SELECT from Amy to Carol to David to Eva. In contrast, the only possible path for UPDATE(A) from Amy to Frank to David to Eva gets disconnected at David to Eva, because FRANK did not give GRANT OPTION to DAVID.`