UCLA
Computer Science Department
Fall 2018

Instructor: J. Cho

Student Name and ID: _____

CS143 Midterm: Closed Book, 110 minutes

(IMPORTANT PLEASE READ **):**

- The exam is closed book and closed notes, but you may use one-page double-sided cheat-sheet during exam. You are also allowed to use a calculator.
- *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.
- Unless otherwise indicated, assume the SQL92 standard and the notation that we learned in the class.
- 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.
- 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	20	
4	20	
5	15	
6	15	
Total	100	

Problem 1 (Relational Algebra): 20 points

1. Consider the relation `Purchase(customer, item, time)`. A tuple in the relation represents the fact that the `customer` bought an `item` at `time`. Write down a relational algebra expression that returns all customers that did not buy the item `'Beer'`.

   ANSWER:
   $\pi_{customer}(Purchase) - \pi_{customer}(\sigma_{item='Beer'}(Purchase))$

2. Consider the following database on songs and albums:

   `Song(`<u>`title`</u>`, length, album)`
   `Album(`<u>`name`</u>`, artist, year)`

   Here, a tuple (t_1, l_1, a_1) in the Song table represents a song with title t_1 of play length l_1 in the album a_1. A tuple (n_1, a_1, y_1) in the Album table represents an album with title n_1 published in year y_1 by artist a_1. Assume that the play length of every song is different.

   Consider the following relational algebra expressions. The symbol := assigns the result from the right-hand side to the variable on the left-hand side.

   $T1 := Song \bowtie_{Song.album=Album.name} Album$
   $T2 := \Pi_{title,length}(\sigma_{artist='Beatles' \wedge year=1980}(T1))$
   $T3 := \Pi_{S.title,S.length}(\sigma_{T2.length>R.length \wedge R.length>S.length}(T2 \times \rho_R(T2) \times \rho_S(T2)))$
   $T4 := T2 - T3$
   $T5 := \Pi_{T2'.title,T2'.length}(\sigma_{T2.length>T2'.length}(T2 \times \rho_{T2'}(T2)))$
   $T6 := T4 - T5$

   In no more than two sentences, explain the meaning of $T6$.

   ANSWER:
   `The title and length of the longest song by Beatles published in 1980.`

Problem 2 (Number of Tuples): 10 points

Suppose the relation $R(A, B)$ has n tuples and the relation $S(B, C)$ has m tuples. Assume that the three attributes A, B, and C are of integer type. Consider the relational algebra expression $R \bowtie S$. Nulls are not allowed in the relations.

1. What is the minimum number of tuples that may result from the above expression? Explain your answer briefly.

   ANSWER:
   0. B values may not match between R and S.

2. What is the maximum number of tuples that may result from the above expression? Explain your answer briefly.

   ANSWER:
   $m \times n$. All tuples in R and S may have exactly the same B value.

Problem 3 (Query Equivalence): 20 points

Two queries are considered equivalent if they return exactly the same results for all database instances. For each of the pair of queries listed below, write "YES" if the two queries are equivalent and "NO" otherwise. Also, briefly explain your answer.

In all questions of Problem 3, assume two relations $R(\underline{A}, B)$ and $S(\underline{A}, B)$, where A **is the key of both relations**. Also, assume that **null values are not allowed.**

1. The expression $\sigma_C(R) - S$ is always equivalent to $\sigma_C(R) - \sigma_C(S)$, where C can be any condition.

   ANSWER:
   YES. $R-S$ returns tuples from R only. So removing the tuples that are not in R does not make any difference.

2. The following two queries are equivalent:

   (a)
   ```
   SELECT B
   FROM R R1
   WHERE B <= ALL (SELECT B FROM R R2 WHERE R1.A <> R2.A))
   ```

   (b)
   ```
   SELECT MIN(B) FROM R
   ```

   ANSWER:
   No. Consider $R = \{(1,0),(2,0)\}$.

Problem 4 (SQL Queries): 20 points

Given the following database tables, choose all queries in SQL that return the proper answer to the corresponding question.

```
Person(SSN, name, address)
Car(license, year, model)
Accident(license, accident-date, driver, damage-amount)
Owns(SSN, license)
```

Here, `Accident.driver` and `Owns.SSN` are foreign keys to `Person.SSN`. Similarly, `Accident.license` and `Owns.license` are foreign keys to `Car.license`. The driver involved in a car accident may not always be the owner of the car. We assume that a car cannot get involved in more than one accident on a certain date.

Please note that each question may have zero, one or more correct answers. Circle only the ones that are correct.

1. Who is the driver who participated in an accident with the most costly damage? Return the driver(s) and the amount of damage.

   (a)
   ```
   SELECT driver, damage-amount FROM Accident
   WHERE damage-amount IN (SELECT MAX(damage-amount) FROM Accident)
   ```

   (b)
   ```
   SELECT driver, damage-amount FROM Accident
   WHERE damage-amount = MAX(damage-amount)
   ```

   (c)
   ```
   (SELECT driver, damage-amount FROM Accident)
   EXCEPT
   (SELECT A1.driver, A1.damage-amount
   FROM Accident A1, Accident A2
   WHERE A1.damage-amount < A2.damage-amount)
   ```

   ANSWER:
   `a, c`

2. Find the license-plate number of all cars that have been involved in more than one accident (DO NOT RETURN DUPLICATES)

   (a)
   ```
   SELECT A1.license-plate FROM Accident A1, Accident A2
   WHERE A1.license-plate = A2.license-plate
       AND A1.accident-date <> A2.accident-date
   ```

   (b)
   ```
   SELECT DISTINCT A1.license-plate FROM Accident A1
   WHERE A1.license-plate IN (SELECT A2.license-plate FROM Accident A2
                   FROM Accident A2
                   WHERE A1.accident-date <> A2.accident-date)
   ```

(c)
```
SELECT license-plate FROM Accident
GROUP BY license-plate
HAVING COUNT(accident-date) > 1
```

ANSWER:
b, c

Problem 5 (Constraints): 15 points

Suppose we have the following table declarations:

```
CREATE TABLE A(w INT PRIMARY KEY);
CREATE TABLE B(x INT PRIMARY KEY REFERENCES A(w) ON DELETE SET NULL;
CREATE TABLE C(y INT REFERENCES A(w));
CREATE TABLE D(z1 INT REFERENCES B(x) ON DELETE SET NULL,
z2 INT REFERENCES A(w) ON UPDATE CASCADE);
```

Consider the following scripts:

I. `DELETE FROM C; DELETE FROM B; DELETE FROM A; DELETE FROM D;`
II. `DELETE FROM C; DELETE FROM D; DELETE FROM A; DELETE FROM B;`
III. `DELETE FROM B; DELETE FROM C; DELETE FROM D; DELETE FROM A;`

Which of the above scripts will empty all four tables, without error? Briefly explain your answer. An answer without any explanation may not get any points.

(a) III only (b) I only (c) II and III only (d) I and III only

ANSWER:

c. The foreign-key constraint from B to A doesn't cause problems if we delete from A first, since there is a clause that tells how to handle dangling tuples in B. However, the foreign-key constraint from C to A is a problem, since the default policy is to reject a deletion from A that causes a dangling tuple in C. Thus, in a valid sequence, C must precede A. The foreign-key constraint from D to B is not a problem, but the one from D to A is, again because the default policy will cause a rejection of certain deletions from A. Thus, D must also come before A. Of the three sequences, only I has C or D before A, so II and III are OK; I is not.

Problem 6 (Views): 15 points

Consider the following two tables. Enrollments table contains information about the number of students who took a course in a given year. Courses table shows the instructor name of a course in a given year.

`Enrollments(year,course,students)` // key is [year,course]
`Courses(course,year,instructor)` // key is [course,year]

Make no assumptions about the tables except for the specified keys. Consider the three view definitions below. Under the SQL92 standard, state whether a given modification for each view:

- ALWAYS generates a view-modification error;
- SOMETIMES (but not always) generates a view-modification error; or
- NEVER generates a view-modification error.

1. View:
```
CREATE VIEW V1 AS
        SELECT E.year, E.course
        FROM Enrollments E, Courses C
        WHERE E.year = C.year AND E.course = C.course
              AND C.instructor = 'Cho'
```
   Modification: `UPDATE V1.year ...`

   Will the above modification cause a view-modification error?

   Circle one: ALWAYS SOMETIMES NEVER

   ANSWER:
   `ALWAYS. A modifiable view can have only one table in FROM.`

2. View:
```
CREATE VIEW V3 AS
        SELECT course, year
        FROM Courses
        WHERE year > 2000
        WITH CHECK OPTION
```
   Modification: `DELETE FROM V3 ...`

   Will the above modification cause a view-modification error?

   Circle one: ALWAYS SOMETIMES NEVER

ANSWER:
NEVER. Deleting from V3 cannot violate the year > 2000 condition.

This page is intentionally left blank to provide scratch space.