UCLA
Computer Science Department
Winter 2013

Instructor: J. Cho

Student Name and ID: _____

CS143 Midterm: Closed Book, 90 minutes

(IMPORTANT PLEASE READ **):**

- There are 3 problems on 9 pages for a total of 75 points to be completed in 90 minutes. *You should look through the entire exam before getting started, in order to plan your strategy.*
- *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	30	
2	25	
3	20	
Total	75	

Problem 1 (Queries): 30 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.

Assume the following for the two relations R and S referenced in the queries.

- Both R and S have two columns, $R(A, B)$ and $S(A, B)$.
- The attribute A is a key for each relation.
- The attribute B is not a key in either relation.
- NULL values are not allowed in any attribute.

Do not make any other assumptions about the data. To discourage randomly guessing the answers, we will give 3 points for each correct answer, deduct 1 point for each incorrect answer, and 0 point for each answer left blank.

Again, remember that A is a key, B is not, and NULL values are not allowed. These assumptions are crucial to derive the correct answers.

Note: you do not need provide an explanation to your answer. No partial point will be assigned to explanation.

1. Equivalent? _____
   (a) $\pi_A(R \cup S)$
   (b) $\pi_A(R) \cup \pi_A(S)$
2. Equivalent? _____
   (a) $\sigma_{R.A=1}(R)$
   (b) $\sigma_{R.A=1}(\pi_{R.A,R.B}(R \bowtie S))$
3. Equivalent? _____
   (a) $\pi_{R.A}(\sigma_{R.A=S.A}(R \times S))$
   (b) `SELECT R.A FROM R,S WHERE R.A=S.A`
4. Equivalent? _____
   (a) $\pi_{R.B}(\sigma_{R.A=S.A}(R \times S))$
   (b) `SELECT R.B FROM R,S WHERE R.A=S.A`

5. Equivalent? _____

   (a) $\pi_A(R) - \pi_{R1.A}(\sigma_{R1.A \leq R2.A}(\rho_{R1}(R) \times \rho_{R2}(R)))$

   (b) `SELECT MAX(A) FROM R`

6. Equivalent? _____

   (a) $\pi_{R1.B}(\sigma_{R1.B=R2.B \wedge R1.A \neq R2.A}(\rho_{R1}(R) \times \rho_{R2}(R)))$

   (b) `SELECT B FROM R GROUP BY B HAVING COUNT(*) > 1`

7. Equivalent? _____

   (a) `SELECT B FROM R GROUP BY B`

   (b) `SELECT DISTINCT B FROM R`

8. Equivalent? _____

   (a)
   ```
   SELECT B FROM R
   WHERE NOT EXISTS(SELECT * FROM S WHERE R.B = S.B)
   ```

   (b) `(SELECT B FROM R) EXCEPT (SELECT B FROM S)`

9. Equivalent? _____

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

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

10. Equivalent? _____

    (a) `SELECT SUM(B)/COUNT(*) FROM R`

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

Problem 2 (Database Integrity): 25 points

Consider a relation $T(A, B)$. For each of the three SQL assertions below, write in the space provided a *plain English* description of the constraint that is enforced by the assertion. Please be specific about the actual attributes involved in the constraint. In all three cases the correct answer corresponds to a single concept from the course and can be specified in a few words.

Note: keep your answer short and clear, and limit your answer to **at most 15 words.** You will lose partial (or even all) points if your answer is more complicated than necessary.

1. (a) (5 points)

```
CREATE ASSERTION First CHECK(
  NOT EXISTS(SELECT *
             FROM T T1, T T2
             WHERE T1.A <> T2.A))
```

Constraint: _____

(b) (5 points)

```
CREATE ASSERTION Second CHECK(
  NOT EXISTS(SELECT B
             FROM T
             WHERE B NOT IN (SELECT A FROM T)))
```

Constraint: _____

(c) (5 points)

```
CREATE ASSERTION Third CHECK(
  NOT EXISTS((SELECT * FROM T)
              EXCEPT
             (SELECT * FROM T WHERE A = A)))
```

Constraint: _____

2. Consider a table `Instructor(name,dept,salary)` with data about school instructors. Instructor name is a key of the table. Currently, `Instructor` contains the following five tuples:

Jason	ME	30
Jane	EE	10
Peter	EE	20
Susan	CS	10
Ted	CS	30

We have the following trigger:

```
CREATE TRIGGER IOnlyHope
AFTER UPDATE OF dept ON Instructor
  REFERENCING OLD ROW AS old_row, NEW ROW AS new_row
  FOR EACH ROW
WHEN (new_row.dept = 'CS' AND old_row.dept <> 'CS')
BEGIN
  UPDATE Instructor
  SET salary = 2 * (SELECT AVG(salary) FROM Instructor)
  WHERE name = new_row.name;
END
```

Suppose the following update command is executed:

```
UPDATE Instructor SET dept = 'CS' WHERE dept = 'EE'
```

List all of the possible final salaries for each of the five tuples in the table after the update command and all triggers have been invoked and completed. Note that when a SQL update command modifies multiple tuples, there is no guarantee about the order in which they are updated — they can be updated in any order.

(2 points each) Possible final salaries for

(a) Jason: _____

(b) Jane: _____

(c) Peter: _____

(d) Susan: _____

(e) Ted: _____

Problem 3 (Views and Authorization): 20 points

1. Consider three tables, `Student(sid, name, age)`, `Class(cid, title)`, and `Enroll(sid, cid)`, where `sid` is the student id and `cid` is the class id. The database user `U1` is the owner of `Student`, but she does not own `Class` or `Enroll` tables. She has the `UPDATE` privilege with `GRANT OPTION` and `SELECT` privilege without `GRANT OPTION` on `Enroll`. She does not have any privilege on `Class`.

   Now consider the following sequence of SQL statements issued by the user `U1`. Assume that `U2` is another valid user.

```
S1: CREATE VIEW YoungStudent AS
       SELECT * FROM Student WHERE age < 19

S2: CREATE VIEW EnrolledStudent AS
       SELECT * FROM Student
       WHERE sid IN (SELECT sid FROM Enroll)

S3: CREATE VIEW MultiEnrolledStudent AS
       SELECT S.sid, S.name, S.age
       FROM Student S, Enroll E
       WHERE S.sid = E.sid
       GROUP BY S.sid
       HAVING COUNT(*) > 1

S4: CREATE VIEW ClassJohn AS
       SELECT * FROM Class
       WHERE cid IN (SELECT cid FROM Student S, Enroll E
                     WHERE S.sid = E.sid AND S.name = 'John')

S5: GRANT SELECT ON YoungStudent TO 'U2'

S6: GRANT SELECT ON EnrolledStudent TO 'U2'

S7: GRANT UPDATE(name) ON EnrolledStudent TO 'U2'

S8: GRANT UPDATE(name, age) ON MultiEnrolledStudent TO 'U2'
```

   Which of the above statements will be executed without any error by DBMS? For each statement, either write "OK", or write "ERROR" along with a brief explanation of why the statement generates an error in the space provided on the next page.

(1.5 points each; if you write down "ERROR", keep your explanation short & clear and limit it to **at most 15 words**)

S1: _____

S2: _____

S3: _____

S4: _____

S5: _____

S6: _____

S7: _____

S8: _____

2. Consider the following table:

   `Sales(store, product, date, number)`

   Each tuple in the table records the sales number of a product at a store. The table contains sales information for three stores and twenty products for the year of 2012. Every product was sold at every store on each day of 2012.

   Now the sales manager wants to compute (1) the total sales number of each product and (2) the total sales number on each day of 2012. To help the sales manager obtain the answers quickly, you have created the following three materialized views. (Note: assume that `CREATE MATERIALIZED VIEW` is a command used to create a materialized view).

```
CREATE MATERIALIZED VIEW StoreProduct AS
   SELECT store, product, SUM(number) AS number FROM Sales
   GROUP BY store, product

CREATE MATERIALIZED VIEW StoreDate AS
   SELECT store, date, SUM(number) AS number FROM Sales
   GROUP BY store, date

CREATE MATERIALIZED VIEW ProductDate AS
   SELECT product, date, SUM(number) AS number FROM Sales
   GROUP BY product, date
```

   (a) In the space provided below, write the SELECT statements that compute the two statistics. In writing your statements, make sure that the answers are computed in a *most efficient way*, accessing the minimum number of tuples during the query execution:

   (1 points) Total sales number of each product

   (1 points) Total sales number on each day of 2012

   (2 points) Assuming that you have to drop one materialized view, which of the three views will you drop?

(More questions on the next page. Please continue.)

(b) (4 points) Assuming that you have drop *two* materialized views (not one), which two of the three views will you drop?

Note that when you drop two materialized views, the most efficient way to compute the two statistics may be different from when you drop one materialized view. In the space provided below, write down the SELECT statements that compute the two statistics in the most efficient way with the remaining one view. If the answer is the same as part (a), you can simply write "SAME AS BEFORE".

Total sales number of each product:

Total sales number on each day of 2012: